CROSSREF™ Manual

Abstract This manual describes the CROSSREF cross-referencing utility, including how to use it with C, COBOL 74, COBOL85, EXTENDED BASIC, FORTRAN, Pascal, SCREEN COBOL, and TAL programs, and how to interpret the output.

Part Number 82597

Tandem Computers Incorporated

Document History

Edition	Part Number	Product Version	Operating System Version	Date
First Edition	82316 A00	CROSSREF A00	GUARDIAN A04/E05	October 1982
Second Edition	82516 A00	CROSSREF B00	GUARDIAN B00/E08	March 1985
Third Edition	82597	CROSSREF C00	GUARDIAN 90 C00	November 1987

New editions incorporate any updates issued since the previous edition.

NEW AND CHANGED INFORMATION

Sections describing how to use CROSSREF on C, COBOL85, and Pascal programs have been added to the *CROSSREF Manual* for the C00 release of the GUARDIAN Operating System.

CONTENTS

FIGURES

TABLES

PREFACE

This manual describes the CROSSREF program development tool. CROSSREF produces cross-reference listings of application programs.

This manual is written for application programmers who want to use CROSSREF on programs written in one or more of the following languages: C, COBOL 74, COBOL85, EXTENDED BASIC, FORTRAN, Pascal, SCREEN COBOL, and TAL, Tandem's Transaction Application Language. You should use this manual along with the reference manual of the language or languages that you are using.

This manual is divided into twelve sections and two appendixes. The first three sections introduce CROSSREF, describe how to run it, and explain how to interpret its output. The next section describes the syntax for CROSSREF commands. Each of the remaining eight sections describe how to use CROSSREF with a specific language. You need to read only the section on the language or languages that you are using.

Section 1, "Introduction," provides a brief introduction to CROSSREF.

Section 2, "Running CROSSREF," explains how to start CROSSREF. It gives the syntax for the CROSSREF command, describes interactive and noninteractive use, and provides examples.

Section 3, "Interpreting CROSSREF Output," describes the parts of a cross-reference listing and how to read the output. It also describes what happens if your source file contains compilation errors.

Section 4, "CROSSREF Commands," gives the syntax for each command, tells you what you should know when using the command, and provides examples.

Section 5, "C," describes using CROSSREF with C. It provides a sample C program and its cross-reference listing.

Section 6, "COBOL 74," describes using CROSSREF with COBOL 74. It provides a sample COBOL program and its cross-reference listing.

Section 7, "COBOL85," describes using CROSSREF with COBOL85. It provides a sample COBOL85 program and its cross-reference listing.

Section 8, "EXTENDED BASIC," describes using CROSSREF with EXTENDED BASIC. It provides a sample BASIC program and its cross-reference listing.

Section 9, "FORTRAN," describes using CROSSREF with FORTRAN. It provides a sample FORTRAN program and its cross-reference listing.

Section 10, "PASCAL," describes using CROSSREF with Pascal. It provides a sample Pascal program and its cross-reference listing.

Section 11, "SCREEN COBOL," describes using CROSSREF with SCREEN COBOL. It provides a sample SCREEN COBOL program and its cross-reference listing.

Section 12, "TAL," describes using CROSSREF with TAL. It provides a sample TAL program and its cross-reference listing.

Appendix A, "Syntax Summary," provides a syntax summary of all CROSSREF commands.

Appendix B, "Warning and Error Messages," lists and describes the error and warning messages that CROSSREF produces.

NOTATION CONVENTIONS

The following list summarizes the conventions for syntax notation in this manual.

Notation	**Meaning**
UPPERCASE LETTERS	Uppercase letters represent keywords and reserved words; enter these items exactly as shown.
italics	Lowercase italic letters represent variable items that you supply.
Brackets []	Brackets enclose optional syntax items. A group of vertically aligned items enclosed in brackets represents a list of selections from which you can choose one or none.
Braces {}	Braces enclose required syntax items. A group of vertically aligned items enclosed in braces represents a list of selections from which you must choose one.
Ellipsis ...	An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed syntax items any number of times.
Spaces	If a space separates two items, that space is required. If one of the items is a punctuation symbol, such as a parenthesis or a comma, spaces are optional.
Punctuation	Parentheses, commas, semicolons, and other symbols not described above must be entered precisely as shown. Quotation marks around any symbol indicate that it is not a syntax descriptor but a required character, and you must enter it as shown.

SECTION 1

INTRODUCTION

WHAT IS CROSSREF?

CROSSREF is a tool that produces a cross-reference listing of selected identifiers in your application program. CROSSREF quickly reduces the time and effort required for program development by answering such questions as:

- Where are the identifiers located and how are they used?
- Which statements, if any, will be affected if I change an identifier?
- Which identifiers, if any, are declared but not used?

Identifiers can be data variables, statement labels, subprograms, etc.

You can use this information to quickly and easily debug or maintain your programs. For example, sometimes a program mistakenly changes the value of a variable. The cross-reference listing shows where the variable appears and how it is used in the program, so you can easily find and fix the problem.

As another example, you might want to know where the transfer occurs when program control passes to a label. The cross-reference listing shows all locations in the program that use the label as a GOTO target.

WHICH LANGUAGES DOES CROSSREF SUPPORT?

CROSSREF supports the following languages: C, COBOL 74, COBOL85, EXTENDED BASIC, FORTRAN, Pascal, SCREEN COBOL, and TAL. (When you want to invoke COBOL 74, enter just COBOL at the system prompt. To invoke COBOL85, enter COBOL85.) You can use CROSSREF on application programs coded in any combination of these languages.

CROSSREF uses a set of generic identifier classes that are mapped to various data types in each language. Not all languages use all of the identifier classes. Instead, the mapping of data types to identifier classes is language specific; a CROSSREF identifier might or might not be used by a given language. The identifier classes used by each language are listed at the beginning of each language section in this book.

HOW DOES CROSSREF WORK?

CROSSREF has two modes of operation: stand alone and compiler dependent. Whichever mode you use, CROSSREF obtains information about your program from the compiler.

CROSSREF uses the compiler to scan the source file and pass information about the identifiers back to CROSSREF. CROSSREF then collects, combines, and sorts identifier information into a single, alphabetized cross-reference listing. Each entry includes information about the name of the identifier, the type of identifier (label, variable, and so forth), the type of reference it is (for example, a read or write reference), and where the reference was found (source file and line number). CROSSREF then writes the listing to a file that you specify.

You can select which identifiers appear in the cross-reference listing by using the SET CLASS command when running in stand-alone mode or by using control directives when running in compiler-dependent mode. Refer to Section 4 for information on the SET command and to the appropriate language manual for information on control directives.

STAND-ALONE CROSSREF

Stand-alone CROSSREF enables you to obtain cross-reference data from source files coded in one or more languages and combine this information into one cross-reference listing. The examples in this manual describe stand-alone CROSSREF.

When you run CROSSREF in stand-alone mode, it ignores the CROSSREF-related compiler directives in the source file. CROSSREF reads control information from the IN file that you specified at startup. (The IN file is usually the home terminal.) It obtains identifier reference information from the compiler through interprocess messages.

COMPILER-DEPENDENT CROSSREF

Compiler-dependent CROSSREF is integrated with a language compiler. It produces a cross-reference listing that is written to the compiler's OUT file. See the appropriate language manual for information on running compiler-dependent CROSSREF.

When CROSSREF runs in compiler-dependent mode, the compiler starts SYMSERV as part of the compilation. SYMSERV is a process that contains CROSSREF and the symbol table collector. SYMSERV uses the control directives that you include in the source file and receives identifier reference information from the compiler through interprocess messages.

SECTION 2

RUNNING CROSSREF

OVERVIEW

This section describes how to run stand-alone CROSSREF from the GUARDIAN command interpreter. It gives the syntax for the CROSSREF command, describes how to start CROSSREF in *interactive* or *noninteractive* mode, and provides examples, including how to use CROSSREF on programs coded in one or more languages.

CROSSREF COMMAND

CROSSREF resides in the file named $SYSTEM.SYSTEM.CROSSREF. Use the CROSSREF command to execute CROSSREF.

CROSSREF [/ *run-option-list* /] [;] [*command-list*]

run-option-list

is one or more of the standard GUARDIAN run options separated by commas (,). The options are described in the *GUARDIAN Operating System Utilities Reference Manual*.

⟶

If you specify the OUT option and omit the file name, then CROSSREF suppresses all output to the OUT file.

;

is an optional delimiter between the last GUARDIAN run option and the first CROSSREF command. You do not need to enter the semicolon; you can omit it without affecting the CROSSREF command.

command-list

is one or more CROSSREF commands. You must separate multiple commands with semicolons. If you specify a command list and an IN file, CROSSREF executes the listed commands and terminates without opening and reading from the IN file.

NOTE

CROSSREF creates its work files on the current default volume. If you are concerned about a shortage of disk space on the current volume, you can tell CROSSREF to create its work files on another volume using the following command:

12> PARAM SWAPVOL *volume-name*

You must specify this command before beginning the CROSSREF session.

Interactive Mode

CROSSREF operates in interactive mode when you omit both the IN *file-name* option and the *command-list* parameter from the CROSSREF command.

When run in interactive mode, CROSSREF sends the following product banner to the terminal:

```
CROSSREF - CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1985, 1986
```

T9622C00 is the product number and version of CROSSREF.

(15JUL87) is the release date for this version of CROSSREF.

CROSSREF also displays the name of the system you are using at the end of the banner.

CROSSREF then displays an ampersand (&) prompt to indicate that it is ready to receive commands. When you enter a command, CROSSREF executes the command or displays an error message. It then displays another ampersand and waits for additional input. This procedure continues until you enter an EXIT command or press CTRL/Y, the End-of-File (EOF) command.

You can enter more than one CROSSREF command in response to the prompt (&), but you must separate multiple commands with semicolons. For example, the following two sets of commands, which tell CROSSREF to scan the file named PROGRAM and generate a listing to the file named CROSS1, are equivalent:

```
&SCAN program; GENERATE /OUT cross1/
```

```
&SCAN program
&GENERATE /OUT cross1/
```

CROSSREF executes commands one at a time from left to right. If CROSSREF encounters an error, it stops processing the command line. CROSSREF, however, executes all commands up to the point of the error.

You can continue commands for more than one line by typing a & at the end of the line. CROSSREF prompts for the remainder of the line before executing the command. When CROSSREF receives a line without an ampersand at the end, it executes the command immediately.

Command lines, continued or not, cannot exceed 528 characters.

Noninteractive Mode

CROSSREF operates in noninteractive mode when you specify either the IN *file-name* or the *command-list* parameter in a CROSSREF run command.

The following command directs CROSSREF to execute the commands in the EDIT file named COMMS and to send the output to the file named LISTING:

```
13> CROSSREF /IN comms, OUT listing/
```

CROSSREF terminates when it reaches the end of the file or when it encounters an EXIT command in the IN file. CROSSREF then returns control to the command interpreter.

The same rules apply to command lines in a command file that apply to command lines that you enter from the terminal:

- Multiple commands can appear on a single line when they are separated by semicolons.
- Command lines can be continued by placing an ampersand (&) at the end of the command line.
- Command lines cannot exceed 528 characters.

EXAMPLES OF CROSSREF OPERATION

The remainder of this section provides examples of CROSSREF operation. The examples in "Single-Language Program" show how to produce a listing for a program coded in a single language. The example in "Multi-Language Program" shows how to produce a listing for a program coded in two languages.

See Section 4 for a complete explanation of the CROSSREF commands used in these examples.

Single-Language Program

The following examples show how CROSSREF produces a cross-reference listing for a program coded in a single language.

To invoke CROSSREF from the GUARDIAN command interpreter, type CROSSREF. The IN *file-name* and OUT *file-name* options default to the home terminal. CROSSREF displays the product banner and issues the prompt character to show that it is ready to receive commands.

```
14> CROSSREF
CROSSREF-CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
&
```

Example 1

Example 1 shows the easiest sequence of commands for producing a cross-reference listing. Simply follow these steps:

1. Specify which language you are using with the SET LANGUAGE command; in this example, it is COBOL 74:

   ```
   &SET LANGUAGE cobol
   ```

2. Tell CROSSREF to scan the program file; in this example, the program file is named DREAM9:

   ```
   &SCAN dream9
   ```

3. Then tell CROSSREF to generate a listing and print it to an output file; in this example, the output goes to a file named OUTFILE:

   ```
   &GENERATE /OUT outfile/
   ```

4. Enter EXIT to end the session and return control to the command interpreter:

   ```
   &EXIT
   15>
   ```

In summary, the following four commands produce a cross-reference listing of the program file DREAM9:

```
&SET LANGUAGE cobol
&SCAN dream9
&GENERATE /OUT outfile/
&EXIT
16>
```

Example 2

To produce a more selective cross reference, you can request that CROSSREF report only certain identifiers. Example 2 shows how to produce a cross-reference listing of just the conditions and labels in the program named DREAM9. (Labels are section and paragraph names in COBOL. DREAM9, as mentioned in Example 1, is a COBOL 74 program.)

```
&SET LANGUAGE cobol
&SET CLASS * OFF
&SET CLASS CONDITIONS ON
&SET CLASS PROGLABELS ON
&SCAN dream9
&GENERATE /OUT outfile/
&EXIT
12>
```

Multi-Language Program

The following example shows how CROSSREF produces a single cross-reference listing for a program coded in two languages. The CROSSREF scan takes place in two steps-- one step for each language.

1. Using the SET LANGUAGE command, specify the language you are using for the first file. In this example, the first file is a COBOL 74 program:

   ```
   &SET LANGUAGE cobol
   ```

2. Tell CROSSREF to scan the file, named TVC15, using all of the default settings for the attribute specifications:

   ```
   &SCAN tvc15
   ```

(Section 4 describes the attribute specifications and the available settings.)

3. After CROSSREF scans the COBOL file, change the language attribute and customize the listing for the next file. This file is a set of FORTRAN subprocedures named FPROCS:

```
&SET LANGUAGE fortran
```

4. Because this file contains only FORTRAN subprocedures with no main procedure, set the unreferenced identifier flag to ON. Otherwise, none of the subprocedures are listed.

```
&SET UNREF ON
```

5. Yet the subprocedure ERROR is called only by other FORTRAN subprocedures, so exclude it from the listing:

```
&SET EXCLUDE ERROR
```

6. Tell CROSSREF to scan the FPROCS file:

```
&SCAN fprocs
```

7. Direct CROSSREF to generate a listing and print it to the file named CSQUARE:

```
&GENERATE /OUT csquare/
```

If the file named CSQUARE does not exist, CROSSREF creates it.

8. Finally, terminate the session by entering EXIT:

```
&EXIT
15>
```

SECTION 3

INTERPRETING CROSSREF OUTPUT

This section describes the parts of a cross-reference listing and how to read its output. It also describes what happens if your source file contains compilation errors.

CROSSREF LISTING

The cross-reference listing consists of two parts: a file list and an identifier list.

File List

The file list appears at the beginning of a cross-reference listing. CROSSREF lists each file that it scans and assigns a number to each file. The file number helps you differentiate line numbers from different files. Figure 3-1 shows a sample file list.

```
FILE NO.     FILE NAME.
  [1]        $DATA.TEST.COBMANE
  [2]        $DATA.TEST.COBLIBA
  [3]        $DATA.TEST.COBLIBC
  [4]        $DATA.TEST.COBLIBB
```

Figure 3-1. Sample File List

When a file is referenced in a program, the number and name of the file being referenced appears followed by the number and name of the file that contains the reference. The line number where the reference occurs appears at the end of the line. If the file is referenced in more than one place in the program, the line numbers appear one on top of the other. For example,

```
FILE NO.    FILE NAME.
  [1]       $EQ1.TEST.WCC
  [2]       $THIO.SYSLIB.STDIOH       WCC[1]            12
```

Identifier List

The identifier list follows the file list. CROSSREF lists identifiers by name in alphabetic order. Each identifier has a header followed by one or more reference lines.

Identifier Header

The header defines the identifier. It contains six fields:

- Identifier Name
- Identifier Qualifier (not always used)
- Attribute List
- File Name
- File Number
- Line Number

Figure 3-2 shows the format of an identifier header.

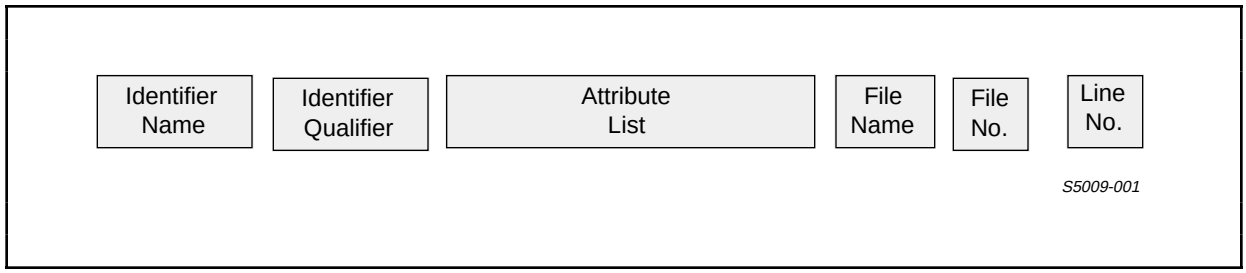


Figure 3-2. Identifier Header Format

Identifier Name. The identifier name is the name of the identifier as it is defined in the source file.

Identifier Qualifier. The identifier qualifier is a field that the compiler might use to further define the identifier. The exact meaning of the qualifier depends on the compiler. See the appropriate compiler section.

Attribute List. The attribute list is a set of identifier attributes sent by the compiler to CROSSREF. The exact meaning of the field depends on the compiler. See the appropriate compiler section.

This field always begins at column 35 of the header. If the combined length of the identifier name and qualifier exceeds 35 characters, the compiler attribute list is placed on the lines following the first header line.

File Name. The file name is the name of the file in which the identifier is defined. The fully qualified file name is shown in the file list.

File Number. The file number is the number assigned to the file name. Every file used in the cross-reference listing, and the file number associated with it, are shown in the file list at the beginning of the cross-reference listing.

Line Number. The line number is the number of the line in the specified file in which the identifier is defined.

Identifier Reference Lines

Following the header line are zero or more reference lines. These lines contain the number of the line (or lines) where the identifier is referenced followed by a code (or codes) that describes the type of reference (for example, read). The references from a single file might extend over several lines.

If an identifier is referenced in more than one file, CROSSREF begins the references for the next file on a new line. The file name and number are listed in the leftmost field followed by the reference line numbers and codes.

The reference line contains a minimum of four fields:

- File Name
- File Number
- Line Number
- Reference Code

Figure 3-3 shows the identifier header followed by an identifier reference line.

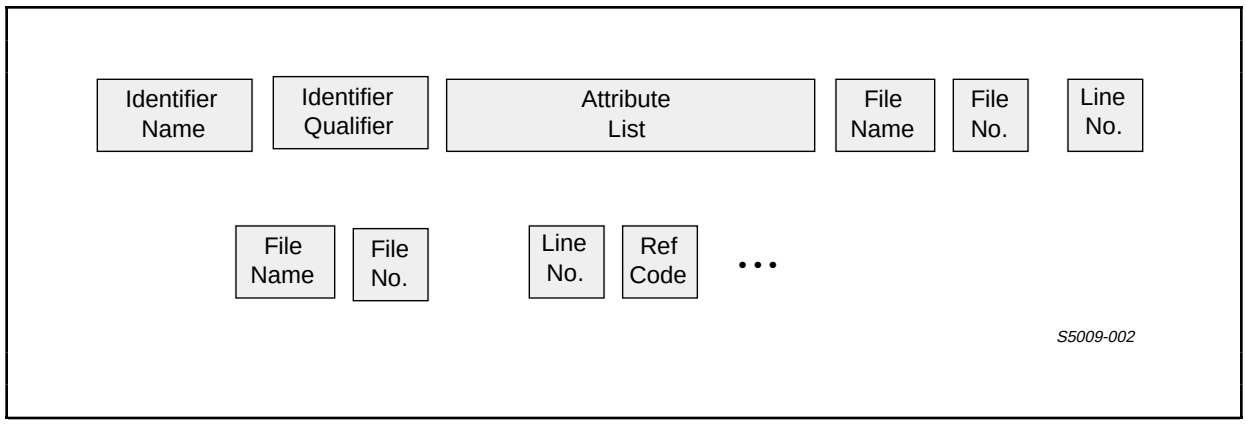


Figure 3-3. Sample Reference Entry

File Name. The file name is the name of the file in which the identifier references occur. It is printed on the leftmost field of the reference line.

File Number. The file number is the number of the file in which the identifier references occur. It is printed immediately after the file name.

Line Number. The line number is the number of the program line in which the identifier reference occurs. There can be several reference line numbers listed for a file. Each line number is followed by a reference code.

Reference Code. The reference code describes the type of reference. There are six codes: blank, D, I, M, P, and W. Table 3-1 describes the reference codes.

Table 3-1. Reference Codes

Code	Description
blank	designates a read reference. This refers to any statement that uses the value of the identifier; it also refers to any parameter that is passed by value.
D	designates a definition. This refers to any nonexecutable statement that defines the type of variable or its storage.
I	designates an invocation reference. This is only generated by procedures, subroutines, parametric procedures, and function calls.
M	designates other miscellaneous references. It refers to any reference that does not fit into another category. For example, a GOTO instruction generates a miscellaneous reference.
P	designates a parameter reference. This occurs only when the identifier is passed as a reference parameter.
W	designates a write reference. This occurs whenever a statement might change the value of the identifier. This does not include parameters to subroutine calls.

CROSSREF IDENTIFIER CLASSES

CROSSREF maintains a flag for each identifier class that it recognizes. By default, all CROSSREF identifier classes except for KEYWORDS and LITERALS are set to ON. The following identifier classes are set to ON:

BLOCKDATAS	INDEXES	PROGLABELS
BLOCKS	INLINES	REGISTERS
CONDITIONS	LINENOS	SCREENS
CONSTANTS	MACROS	SUBPROCS
FILES	MNEMONICS	SYSVARS
FMTLABELS	PROCEDURE PARAMS	TYPES
FUNCTIONS	PROCEDURES	VARIABLES

You can change the setting of a flag to ON or OFF using the SET command. If you set a flag to OFF, the identifier class does not appear in the CROSSREF listing.

NOTE

Additionally, unreferenced identifiers do not appear in the list unless you set the UNREF attribute specification to ON or ONLY. See the "SET Command" in Section 4 for details.

Not all identifier classes are used by all languages. See the appropriate language section for a list of identifier classes used by that language.

COMPILATION ERRORS

If your source file contains compilation errors, CROSSREF collects all of the compiler diagnostic messages and writes them to the file that you specify. See the ERRORS command in Section 4.

Consult the corresponding compiler reference manual for a description of the messages.

SECTION 4

CROSSREF COMMANDS

This chapter describes the commands that you can use when executing CROSSREF. It covers commands that are common to the Program Development Tools (CROSSREF, BINDER, and INSPECT) and commands that are used only by CROSSREF. Table 4-1 lists all commands executable by CROSSREF. Those commands that are common to the three tools are marked with an asterisk (*). Each command is explained in detail following the table.

Table 4-1. Summary of CROSSREF Commands (Page 1 of 2)

Command	Description
COMMENT	adds comments to log of CROSSREF session.
ENV*	displays current environment settings.
ERRORS	prints error messages generated by the compiler during last SCAN command.
EXIT*	stops CROSSREF and prints all remaining cross-reference information.
FC*	lets you edit or repeat a command line.
GENERATE	prints all cross-reference information gathered since the last GENERATE command or CROSSREF startup.

Table 4-1. Summary of CROSSREF Commands (Page 2 of 2)

Command	Description
HELP*	displays the syntax of CROSSREF commands.
LOG*	directs a copy of the input commands and output produced by CROSSREF to a file.
OBEY*	reads commands from a specified file.
OUT*	directs output listing to a specified file.
RESET	restores attribute specifications to their initial values.
SAVE	saves current attribute settings after a GENERATE command is executed.
SCAN	searches specified source files for cross-reference information.
SET	changes attribute specifications for subsequent SCAN commands.
SHOW	displays one or more current attribute settings.
SYSTEM*	changes the default system setting.
VOLUME*	changes the default volume and subvolume settings.

CROSSREF ATTRIBUTE SPECIFICATIONS

Within the CROSSREF commands, the RESET, SAVE, SCAN, and SET commands allow you to set *attribute specifications*, a group of features which control:

- The compiler that scans the source file and
- The information that CROSSREF collects and reports.

Table 4-2 lists and describes these attribute specifications.

Table 4-2. CROSSREF Attribute Specifications (Page 1 of 2)

Attribute	Definition
CLASS	controls which types of identifiers are reported by setting the identifier class mapped to the type ON or OFF.
CPU	specifies the CPU on which the compiler runs.
DEFINITIONS ONLY	prints only the definition of the reported identifiers.
DIRECTIVES	passes compiler directives to the compiler.
EXCLUDE	specifies which identifiers to exclude from the listing.
INCLUDE	specifies which identifiers to include in the listing.
LANGUAGE	sets the language compiler that CROSSREF uses to scan the file.
LIBRARY	specifies a library file to be passed to the COBOL compiler during start-up.
MEM	specifies the amount of memory to be reserved for the compiler during a SCAN.

Table 4-2. CROSSREF Attribute Specifications (Page 2 of 2)

Attribute	Definition
OMIT	excludes specified files from the listing.
PRIORITY	specifies the execution priority at which the compiler runs.
PROGRAM	specifies the file containing the compiler. (You need to use this attribute only when the subvolume containing the compiler is not the same as the one that contains CROSSREF.)
UNREF	specifies whether CROSSREF should include unreferenced identifiers in the listing.

COMMENT COMMAND

The COMMENT command allows you to add comments to the log of your CROSSREF session. The syntax of the COMMENT command is:

COMMENT *text*

text

is any text you want to print as a comment.

Considerations

- See the description of the LOG command in this section for information on creating a log of your CROSSREF session.
- To continue a comment on the following line, end the current line with an ampersand (&).
- You can use the COMMENT command to record comments that you intend to use as a CROSSREF input file.

Example

The following example prints comments to the log file named MYLOG:

```
&LOG TO mylog
&COMMENT  This is sample comment text...
&SET LANGUAGE tal
&SCAN ncgsect1
&LOG STOP
&GENERATE /OUT $s.#lp1/
&EXIT
15>
```

ENV COMMAND

The ENV command displays the current settings of the program environment parameters. The syntax of the ENV command is:

```
                              [ LOG    ]
ENV [ / OUT file-name / ]     [ SYSTEM ]
                              [ VOLUME ]
```

OUT *file-name*

directs the list of current program environment settings to the specified file.

LOG

displays the current log file-name if logging is in progress.

SYSTEM

displays the current system name.

VOLUME

displays the current volume and subvolume names.

Considerations

If you specify ENV without specifying an option, CROSSREF displays the values for all parameters.

Examples

The following example displays the system name:

```
&ENV SYSTEM
```

The next example displays the values for all the parameters (LOG, SYSTEM, and VOLUME):

```
&ENV
```

ERRORS COMMAND

The ERRORS command lists the diagnostic messages produced by the compiler during a SCAN command. The syntax of the ERRORS command is:

```
ERRORS [ / OUT file-name / ]

OUT file-name

   identifies a file to receive the list of compiler
   messages.

   file-name must be a valid GUARDIAN file name.
```

Considerations

- You must specify the ERRORS command after the SCAN command. If you specify an ERRORS command first, CROSSREF prints ERROR FILE EMPTY to the specified file.
- If you do not specify an OUT file, CROSSREF prints the diagnostic messages to the current CROSSREF OUT file.
- The CROSSREF OUT file is set when you start CROSSREF, but you can change it during the execution of CROSSREF using the system command OUT.

Example

The following example prints all diagnostic messages to the file named ERRFILE:

```
&SET LANGUAGE cobol
&SCAN build1
&ERRORS /OUT errfile/
&GENERATE /OUT $s.#lp1/
&EXIT
14>
```

EXIT COMMAND

The EXIT command stops CROSSREF and returns control to the command interpreter. The syntax of the EXIT command is:

```
EXIT
```

Considerations

- If you type EXIT before CROSSREF has sent the accumulated cross-reference information to the current OUT file, CROSSREF executes an implied GENERATE command. The implied GENERATE causes CROSSREF to send the output to the current CROSSREF OUT file before exiting and returning control to the command interpreter.
- You can stop CROSSREF and return to the command interpreter by pressing CTRL/Y at the CROSSREF prompt (&).

Example

The following example exits from CROSSREF and returns control to the command interpreter:

```
&EXIT
12>
```

FC COMMAND

The FC command allows you to edit or repeat a command. The syntax of the FC command is:

```
FC
```

Considerations

The FC command works the same way in CROSSREF as it does in the command interpreter or EDIT. When you enter FC, the FC prompt appears on the next line followed by the last command that you entered. The FC prompt is a period (.). You can enter the subcommands I, R, or D to edit the displayed line. I inserts characters, R replaces characters, and D deletes characters. See the *GUARDIAN 90 Operating System User's Guide* for details.

Example

The following example shows how to change LITTERALA ON to SET LITERALS ON.

```
&LITTERALA ON
          ^
 **** ERROR **** Invalid syntax
&FC
.LITTERALA ON
.   D                deletes the extra letter T
.LITERALA ON
.ISET                inserts the command SET (see * below)
.SET LITERALA ON
.           RS       replaces the letter A with S
.SET LITERALS ON
.                    executes the command
```

* You must type a blank space after ISET to insert a blank before LITERALA ON.

GENERATE COMMAND

The GENERATE command prints all cross-reference information that has accumulated since the last GENERATE command or the beginning of the session and empties the CROSSREF storage buffers. The syntax of the GENERATE command is:

```
GENERATE [ / OUT file-name / ]

OUT file-name

   identifies a file to receive the cross-reference
   listing.

   file-name must be a valid GUARDIAN file name.
```

Considerations

- If you have not issued a SCAN command during the current session, CROSSREF issues an error message and does not produce a listing.
- The CROSSREF OUT file is set when you start CROSSREF, but you can change it during CROSSREF execution using the system command OUT.
- If you do not specify an OUT file, CROSSREF prints the information to the current CROSSREF OUT file.
- If you type EXIT without issuing a GENERATE command, CROSSREF executes an implicit GENERATE command and sends the listing to the current CROSSREF OUT file.
- If you specify SAVE before entering a GENERATE command, the current attribute settings remain in effect after CROSSREF executes the GENERATE command. If you do not issue a SAVE command, CROSSREF restores all attribute settings to their initial values after executing the GENERATE command.

Example

The following example prints the cross-reference listing to the file named LISTING:

```
&SET LANGUAGE cobol
&SCAN empsal
&GENERATE /OUT listing/
&EXIT
12>
```

HELP COMMAND

The HELP command displays the syntax of CROSSREF commands. The syntax of the HELP command is:

```
                                [ command-name ]
HELP [ / OUT file-name / ] [ param-name   ]
                           [ <param-name> ]
```

OUT *file-name*

directs the help output to the specified file. See the description of the OUT command in this section for additional information.

command-name

is the name of a CROSSREF command.

param-name

is one of the following parameters: *class-name*, *class-list*, or *file-name*. The angle brackets are optional.

Considerations

- If you specify HELP without specifying an option, CROSSREF displays the names of all CROSSREF commands.
- If you do not know how to spell a particular *class-name*, enter HELP *command-name*, where *command-name* is the name of the command with which *class-name* is associated. CROSSREF then displays all parameters for that command. (*class-list* and *class-name* display the same parameters.)
- Make sure that the PDTHELP file resides in the same volume and subvolume as CROSSREF. If the file is in another place, CROSSREF cannot print the help information.

Examples

The following example displays the names of all CROSSREF and system commands:

```
&HELP
```

The next example displays the syntax of the SET command:

```
&HELP SET
```

LOG COMMAND

The LOG command writes a copy of the current session's input and output to a file. The syntax of the LOG command is:

```
    { TO file-name }
LOG {              }
    { STOP         }
```

TO *file-name*

identifies a file to receive the copy of the commands and output. If the file does not exist, CROSSREF creates one using the specified *file-name*.

STOP

closes the current log file and stops all logging.

Considerations

- If you specify the name of a disk file that does not exist, CROSSREF creates an EDIT file and sends the output to that file.
- If you specify the name of a disk file that already exists, CROSSREF appends the output to the existing EDIT file.
- If you issue another LOG *file-name* command when logging is already in progress, CROSSREF closes the previous log file and begins to log to the new file. If the specified file name is the same as the previous log file name, CROSSREF ignores the LOG command and continues to log to the same file.

Example

The following example writes a copy of all input commands and CROSSREF output to the file named LOGFILE. CROSSREF continues to write the output to the log file until you enter the LOG STOP command.

```
&LOG TO logfile
&SET LANGUAGE cobol
&SCAN filename
&LOG STOP
&GENERATE /OUT listing/
&EXIT
13>
```

OBEY COMMAND

The OBEY command reads commands from a specified file. The syntax of the OBEY command is:

```
OBEY [ / OUT file-name / ] file-name

OUT file-name

   directs any output listing to the specified file.
   However, error messages are also displayed at the
   terminal if you are using CROSSREF interactively.  For
   additional information, see the description of the OUT
   command in this section.

file-name

   is a standard GUARDIAN file name.
```

Considerations

- If you do not fully qualify the file name, CROSSREF expands the name using the currently set system, volume, and subvolume names. These names can be inherited from the command interpreter defaults or specified using the SYSTEM and VOLUME commands.
- CROSSREF reads commands from the specified file and processes them until it encounters an end-of-file. It then closes the OBEY file, and command input reverts to the previous input file.
- You can use an OBEY command within an OBEY file; you can nest OBEY files up to a depth of four.
- If you change the default setting of SYSTEM or VOLUME in an OBEY file, those settings remain in effect after you return from the OBEY file. To return to the previous settings, you must enter another SYSTEM or VOLUME command.

- If any part of the specification is invalid or if the file does not exist or cannot be opened, CROSSREF displays an error message and does not change the current source for command input.
- If CROSSREF detects an error while processing an OBEY file, it closes the file and, in the case of nested OBEY files, any other OBEY files currently open. If the original input file was a terminal, CROSSREF issues a prompt on the terminal. If the input file was not a terminal, CROSSREF terminates.

Example

In the following example, CROSSREF reads commands from the file named INSPCOM:

```
&OBEY inspcom
```

OUT COMMAND

The OUT command directs the output listing to a specified file. The syntax of the OUT command is:

```
{ OUT file-name                          }
{ command / OUT file-name / param-name }
```

file-name

is a standard GUARDIAN file name.

command

is a CROSSREF command.

param-name

specifies one or more parameters for *command*.

Considerations

- The first form of the OUT command redirects all CROSSREF output to *file-name* for the duration of the current CROSSREF session.
- The second form of the OUT command temporarily redirects the CROSSREF output to *file-name*. You use this form of the command within another command, for example SHOW /OUT listfile/ CLASS.
- If *file-name* is a disk file and the file does not exist, CROSSREF creates an EDIT file. If the named file is an existing disk file, CROSSREF appends the output to the existing EDIT file.
- If *file-name* is invalid or if CROSSREF cannot open the file, CROSSREF displays an error message and does not redirect the listing.

Examples

The following example sends all output to the file named EUSTACE:

```
&OUT EUSTACE
```

The next example sends the HELP command output to the file named FRED:

```
&HELP /OUT FRED/ SCAN
```

RESET COMMAND

The RESET command restores attribute specifications to their initial values. The syntax of the RESET command is:

```
      { *                          }
RESET {                            }
      { attribute-specification }
```

*

restores all attributes.

attribute-specification

is one of the following:

CLASS [*class-name*]

CLASS resets all identifier class flags to their default values.

CLASS *class-name* resets a particular *class-name* to its default value (ON or OFF); *class-name* can be one of the following:

BLOCKDATAS	INLINES	PROGLABELS
BLOCKS	KEYWORDS	REGISTERS
CONDITIONS	LINENOS	SCREENS
CONSTANTS	LITERALS	SUBPROCS
FILES	MACROS	SYSVARS
FMTLABELS	MNEMONICS	TYPES
FUNCTIONS	PROCEDURE PARAMS	VARIABLES
INDEXES	PROCEDURES	

See "CROSSREF Identifier Classes" in Section 3 for a list of default values.

→

CPU

restores the setting of the CPU to the same CPU that is running CROSSREF.

DEFINITIONS ONLY

turns off the DEFINITIONS ONLY setting.

DIRECTIVES

clears the string argument to be passed to the compiler during startup.

EXCLUDE

empties the identifier Exclude list.

INCLUDE

empties the identifier Include list.

LANGUAGE

clears the language setting.

LIBRARY

clears the library setting.

MEM

restores the MEM setting to its default value.

OMIT

empties the Omit list.

→

PRIORITY

restores the priority setting for compiler runs to the priority of CROSSREF.

PROGRAM

restores the program setting.

UNREF

restores the unreferenced identifier flag to OFF.

Considerations

See the SET command for more information on the attribute specifications.

Examples

The following example restores all attributes to their original state:

```
&RESET *
```

The next example restores all identifier classes to their default values:

```
&RESET CLASS
```

The next example restores the BLOCKS identifier to its default value:

```
&RESET CLASS BLOCKS
```

SAVE COMMAND

The SAVE command saves set attributes after CROSSREF executes a GENERATE command. The syntax of the SAVE command is:

```
     { *                          }
SAVE {                            }
     { attribute-specification    }
```

*

saves all set attributes.

attribute-specification

is one of the following:

CLASS

saves all set identifier class flags.

CPU

saves the CPU setting.

DEFINITIONS ONLY

saves the DEFINITIONS ONLY setting.

DIRECTIVES

saves the string argument to be passed to the compiler during startup.

LANGUAGE

saves the language setting.

→

LIBRARY

saves the library setting.

MEM

saves the MEM setting.

PRIORITY

saves the priority setting for compiler runs.

PROGRAM

saves the program setting.

UNREF

saves the unreferenced identifier flag setting to ON.

Considerations

- You must specify the SAVE command before the GENERATE command.
- If you do not specify SAVE, all attributes are reset to their default values after a GENERATE command.

Example

In the following example, the LITERALS class remains ON after the GENERATE command executes:

```
&SET LANGUAGE cobol
&SET CLASS LITERALS ON
&SAVE CLASS
&GENERATE
&EXIT
13>
```

SCAN COMMAND

The SCAN command specifies source files for CROSSREF to examine. The SCAN command can optionally set attribute specifications that are in effect only for the duration of the command. The syntax of the SCAN command is:

```
SCAN file-list [ , attribute-specification ] ...
```

file-list

is one or more standard GUARDIAN file names. The syntax is as follows:

```
{ file-name                          }
{ ( file-name [ , file-name ] ... ) }
```

The specified file must contain source code in the programming language specified in the SET LANGUAGE or SCAN command.

attribute-specification

is one of the attribute specifications listed below. The attribute specifications have effect only for the duration of the SCAN command.

```
      { { class-name } [ ON  ] }
CLASS { { *          } [ OFF ] }
      { class-list             }
```

The CLASS specification sets identifier class flags.

→

class-name

is one or more of the following:

BLOCKDATAS	INLINES	PROGLABELS
BLOCKS	KEYWORDS	REGISTERS
CONDITIONS	LINENOS	SCREENS
CONSTANTS	LITERALS	SUBPROCS
FILES	MACROS	SYSVARS
FMTLABELS	MNEMONICS	TYPES
FUNCTIONS	PROCEDURE PARAMS	VARIABLES
INDEXES	PROCEDURES	

Not all class names are valid for all languages. If you specify a class name that is not valid for the current language, the identifier class flag is set, but it has no effect on the operation of the SCAN command.

If you specify a class name and do not state whether it is ON or OFF, CROSSREF uses the default setting for that class. See "CROSSREF Identifier Classes" in Section 3 for a list of default values.

*

sets all classes to their default values or to the indicated value if one is specified.

class-list

is more than one class name. The syntax is as follows:

```
{ ( name [ ON  ] [ , name [ OFF ] ]... ) }
{        [ OFF ]           [ OFF ]        }
```

As mentioned above under *class-name*, if you do not specify ON or OFF, CROSSREF uses the default setting for the indicated *class-name*.

→

ON

tells CROSSREF to include the class in the listing.

OFF

tells CROSSREF to exclude the class from the listing.

CPU *cpu-number*

specifies the CPU that will run the compiler that works with CROSSREF. *cpu-number* must be a decimal number from 0 to 15, but the range of valid values depends on the number of processors in your system.

The initial setting is the CPU on which CROSSREF is running.

DEFINITIONS ONLY

prints only the definition of each identifier (excluding references) in the cross-reference listing.

DIRECTIVES " [;] *directive* ..."

specifies a string of one or more compiler directives to be passed to the compiler during startup.

Passing directives in this manner has the same effect as specifying those directives in the compiler's invocation line. A semicolon may optionally precede the string.

Initially, no directives are passed to the compiler.

→

```
        { class-name                                }
EXCLUDE {                                           }
        { ( class-name [ , class-name ] ... ) }
```

adds the specified identifier classes to the Exclude list. CROSSREF does not generate a cross-reference listing for identifiers found on this list, regardless of the identifier class settings. You cannot place literals on the Exclude list.

Initially, the Exclude list is empty.

```
        { class-name                                }
INCLUDE {                                           }
        { ( class-name [ , class-name ] ... ) }
```

adds the specified identifier classes to the Include list. CROSSREF generates a cross-reference listing for identifiers found on this list, regardless of the identifier class settings. You cannot place literals on the Include list.

Initially, the Include list is empty.

```
          { BASIC   }
          { C       }
          { COBOL   }
LANGUAGE  { COBOL85 }
          { FORTRAN }
          { PASCAL  }
          { SCOBOL  }
          { SCOBOLX }
          { TAL     }
```

selects the language for subsequent cross-reference scanning of source files. You can select only one language at a time.

Initially, the language setting is undefined.

→

LIBRARY *file-name*

specifies a library file name that CROSSREF passes to the COBOL 74, COBOL85, or SCREEN COBOL compiler during startup. The compiler reads text from the library file when it encounters unqualified COPY statements in the source file. Other compilers do not use the LIBRARY attribute.

file-name must be a valid GUARDIAN file name.

By default, no library file name is passed to the compiler.

MEM *pages*

specifies the number of memory pages to be reserved for the compiler during a SCAN command.

pages must be a decimal integer between 0 and 64. If you specify a number smaller than the minimum required for compiler operation, the minimum is used.

```
     { file-name                          }
OMIT {                                    }
     { ( file-name [ , file-name ] ... ) }
```

excludes references contained in the designated files from the cross-reference listing.

file-name must be a standard GUARDIAN file name.

Initially, the Omit list is empty.

PRIORITY *priority-number*

specifies the execution priority at which the compiler process will run during a SCAN. If you specify an execution priority greater than the CROSSREF priority, the CROSSREF priority is used. The CROSSREF priority is set to 140.

priority-number should be a decimal integer between 1 and 140.

→

The initial setting for compiler execution priority is the same as the CROSSREF priority.

PROGRAM *file-name*

specifies the file containing the compiler. This command is used when the compiler specified in LANGUAGE *attribute-specification* does not reside on the same subvolume as CROSSREF.

file-name must be a standard GUARDIAN file name.

The initial setting assumes that the compiler is on the same subvolume as CROSSREF.

```
      { ON   }
UNREF { OFF  }
      { ONLY }
```

indicates whether CROSSREF should or should not include unreferenced identifiers in the cross-reference listing. (The default setting is OFF.)

ON includes unreferenced identifiers (as well as referenced identifiers).

OFF excludes unreferenced identifiers.

ONLY includes unreferenced identifiers only.

Considerations

- You can specify the language before issuing a SCAN command or in the SCAN command. To specify the language beforehand, use the SET command.
- An identifier cannot be on both the Include list and the Exclude list.
- The use of the identifier class flags varies from language to language. See the appropriate language section for more information.

- The MEM attribute specification allows you to specify more memory than the compiler default.
- The default execution priority for the compiler is the same as the execution priority of CROSSREF. You can specify a lower execution priority, but the compiler can never execute at a priority greater than the CROSSREF priority.
- Every language has one or more constructs that are defined but never referenced, for example, the BASIC IMAGE statement or the FORTRAN COMMON block. You must set UNREF to ON or ONLY to have unreferenced identifiers appear in the CROSSREF listing.

Examples

The following example tells CROSSREF to scan the file named FILE1:

```
&SCAN file1
```

The next example tells CROSSREF to scan the file named FILE1 and include keywords in the listing:

```
&SCAN file1, CLASS KEYWORDS ON
```

SET COMMAND

The SET command changes the attribute specifications for the current execution of CROSSREF. The syntax of the SET command is:

SET *attribute-specification*

attribute-specification

is one of the following:

```
                      [ ON  ]
CLASS [ class-name    [     ] ]
                      [ OFF ]
```

CLASS sets all identifier class flags to the default values.

CLASS *class-name* sets the specified *class-name* to either its default value or to the indicated value if specified. *class-name* can be one of the following:

BLOCKDATAS	INLINES	PROGLABELS
BLOCKS	KEYWORDS	REGISTERS
CONDITIONS	LINENOS	SCREENS
CONSTANTS	LITERALS	SUBPROCS
FILES	MACROS	SYSVARS
FMTLABELS	MNEMONICS	TYPES
FUNCTIONS	PROCEDURE PARAMS	VARIABLES
INDEXES	PROCEDURES	

Not all class names are valid for all languages. If you specify a class name that is not valid for the current language, CROSSREF sets the identifier class flag, but the flag has no effect on the operation of the SCAN command.

If you specify a class name and do not state whether it is ON or OFF, CROSSREF uses the default setting for the flag. See "CROSSREF Identifier Classes" in Section 3 for a list of default values.

⟶

ON

tells CROSSREF to include the class in the listing.

OFF

tells CROSSREF to exclude the class from the listing.

CPU *cpu-number*

specifies the CPU that will run the compiler that works with CROSSREF.

cpu-number must be a decimal number from 0 to 15, but the range of valid values depends on the number of processors in your system.

The initial setting is the CPU on which CROSSREF is running.

DEFINITIONS ONLY

prints only the definition of each identifier (excluding references) in the cross-reference listing.

DIRECTIVES " [;] *directive* ..."

specifies a string of one or more compiler directives to be passed to the compiler during startup.

Passing directives in this manner has the same effect as specifying these directives in the compiler's invocation line. A semicolon may optionally precede the string.

Initially, no directives are passed to the compiler.

→

```
        { class-name                                }
EXCLUDE {                                           }
        { ( class-name [ , class-name ] ... ) }
```

adds the specified identifier classes to the Exclude list. CROSSREF does not generate a cross-reference listing for identifiers found on this list, regardless of the setting of the identifier's class flag. You cannot place literals on the Exclude list.

Initially, the Exclude list is empty.

```
        { class-name                                }
INCLUDE {                                           }
        { ( class-name [ , class-name ] ... ) }
```

adds the specified identifiers to the Include list. CROSSREF generates a cross-reference listing for identifiers found on this list, regardless of the setting of the identifier's class flag. You cannot place literals on the Include list.

Initially, the Include list is empty.

```
          { BASIC   }
          { C       }
          { COBOL   }
LANGUAGE  { COBOL85 }
          { FORTRAN }
          { PASCAL  }
          { SCOBOL  }
          { SCOBOLX }
          { TAL     }
```

selects the language for subsequent cross-reference scanning of source files. You can select only one language at a time.

Initially, the language setting is undefined.

→

LIBRARY *file-name*

specifies a library file name that CROSSREF passes to the COBOL 74, COBOL85, or SCREEN COBOL compiler during startup. The compiler reads text from the library file when it encounters unqualified COPY statements in the source file. Other compilers do not use the LIBRARY attribute.

By default, no library file name is passed to the compiler.

MEM *pages*

specifies the number of memory pages to be reserved for the compiler during a SCAN command.

pages must be a decimal integer between 0 and 64. If you specify a number smaller than the minimum required for compiler operation, CROSSREF uses the minimum.

```
     { file-name                          }
OMIT {                                    }
     { ( file-name [, file-name ] ... ) }
```

excludes references contained in the designated files from the cross-reference listing.

file-name must be a valid GUARDIAN file name.

Initially, the Omit list is empty.

PRIORITY *priority-number*

specifies the execution priority at which the compiler process will run during a SCAN command. If you specify an execution priority greater than the CROSSREF priority, the CROSSREF priority is used. The CROSSREF priority is set at 140.

priority-number should be a decimal integer between 1 and 140.

→

The initial setting for compiler execution priority is the same as the CROSSREF priority.

PROGRAM *file-name*

specifies the file containing the compiler. You must use this command if the compiler specified in LANGUAGE *attribute-specification* does not reside on the same subvolume as CROSSREF.

file-name must be a valid GUARDIAN file name.

The initial setting assumes that the compiler is on the same subvolume as CROSSREF.

```
      { ON   }
UNREF { OFF  }
      { ONLY }
```

indicates whether CROSSREF should or should not include unreferenced identifiers in the cross-reference listing. The default setting is OFF.

ON includes unreferenced identifiers as well as referenced identifiers.

OFF excludes unreferenced identifiers.

ONLY includes unreferenced identifiers only.

Considerations

- An identifier cannot be on both the Include list and the Exclude list.
- The MEM attribute specification allows you to specify more memory than the compiler default.
- The default execution priority for the compiler is the same as the execution priority of CROSSREF. You can specify a lower execution priority, but the compiler can never execute at a priority greater than the CROSSREF priority.

- The use of the identifier class flags varies from language to language. See the appropriate language section for more information.
- All compilers require that a semicolon precede the directive string specified in the startup message. If you do not include the preceding semicolon in a SET DIRECTIVES command, CROSSREF puts one in for you.
- Every language has one or more constructs that are defined but never referenced, for example, the BASIC IMAGE statement and the FORTRAN COMMON block. You must set UNREF ON to have unreferenced identifiers appear in the CROSSREF listing.

Examples

The following example tells CROSSREF that the file it should scan is a COBOL 74 file:

```
&SET LANGUAGE COBOL
```

The next example sets all identifier classes ON:

```
&SET CLASS * ON
```

The next example sets the LITERALS class ON:

```
&SET CLASS LITERALS ON
```

SHOW COMMAND

The SHOW command displays the current settings of attribute specifications. By default, CROSSREF sends the output to the current CROSSREF OUT file. The syntax of the SHOW command is:

```
                                  { *                          }
SHOW [ / OUT file-name / ] ]      {                            }
                                  { attribute-specification }
```

OUT *file-name*

identifies a file to receive the output from the SHOW command.

file-name must be a valid GUARDIAN file name.

If you do not specify an OUT file, CROSSREF prints the output to the current CROSSREF OUT file.

*

displays the current settings of all attributes.

attribute-specification

is one of the following:

CLASS [*class-name*]

CLASS displays the status of all identifier class flags.

CLASS *class-name* displays the status of a particular *class-name*. *class-name* is one of the following:

⟶

BLOCKDATAS	INLINES	PROGLABELS
BLOCKS	KEYWORDS	REGISTERS
CONDITIONS	LINENOS	SCREENS
CONSTANTS	LITERALS	SUBPROCS
FILES	MACROS	SYSVARS
FMTLABELS	MNEMONICS	TYPES
FUNCTIONS	PROCEDURE PARAMS	VARIABLES
INDEXES	PROCEDURES	

CPU

displays the current CPU setting.

DEFINITIONS ONLY

displays the current setting of the DEFINITIONS ONLY attribute.

DIRECTIVES

displays the current setting of the directive string.

EXCLUDE

displays the contents of the Exclude list.

INCLUDE

displays the contents of the Include list.

LANGUAGE

displays the current language setting.

LIBRARY

displays the current library file.

→

MEM

displays the current memory page setting.

OMIT

displays the files on the Omit list.

PRIORITY

displays the execution priority setting for the compiler during a SCAN command.

PROGRAM

displays the file to be executed as the compiler during a SCAN command.

UNREF

displays the setting of the unreferenced identifier flag.

Considerations

- The CROSSREF OUT file is set when you start CROSSREF, but you can change it during the CROSSREF execution using the system command OUT.

Example

The following example prints the current settings for all attributes to the file named SHOWME:

```
&SHOW /OUT showme/ *
```

This command produces the following output:

```
   CLASS            BLOCKS  ON            BLOCKDATAS  ON
   CLASS            CONDITIONS  ON        CONSTANTS  ON
   CLASS            FILES  ON             FMTLABELS  ON
   CLASS            FUNCTIONS  ON         INDEXES  ON
   CLASS            INLINES  ON           KEYWORDS   OFF
   CLASS            LINENOS  ON           LITERALS  OFF
   CLASS            MACROS  ON            MNEMONICS  ON
   CLASS            PROCEDURE PARAMS  ON  PROCEDURES  ON
   CLASS            PROGLABELS  ON        REGISTERS  ON
   CLASS            SCREENS  ON           SUBPROCS  ON
   CLASS            SYSVARS  ON           TYPES  ON
   CLASS            VARIABLES  ON
   CPU              6
   DEFINITIONS ONLY FALSE
   DIRECTIVES       UNDEFINED
   EXCLUDE          LIST EMPTY
   INCLUDE          LIST EMPTY
   LANGUAGE         UNDEFINED
   LIBRARY          UNDEFINED
   MEM              UNDEFINED
   OMIT             LIST EMPTY
   PRIORITY         140
   PROGRAM          UNDEFINED
   UNREF            OFF
```

SYSTEM COMMAND

The SYSTEM command changes the default system setting. CROSSREF uses the default setting to find a filename if you do not explicitly specify a system name. The syntax of the SYSTEM command is:

```
SYSTEM [ system ]

system

   is a GUARDIAN system name.  If you do not specify a
   system, the system in which CROSSREF is running becomes
   the default system.
```

Example

The following example changes the default system to \NYPD:

```
&SYSTEM \nypd
```

VOLUME COMMAND

The VOLUME command changes the volume and subvolume settings. CROSSREF uses these settings to find a filename if you do not explicitly specify a volume and subvolume name. The syntax of the VOLUME command is:

```
VOLUME volume [ .subvol ]

volume

   is a GUARDIAN volume name.

subvol

   is a GUARDIAN subvolume name.
```

Examples

The following example changes the default volume to $MKT:

```
&VOLUME $mkt
```

The next example changes the default volume and subvolume to $MKT.ABC:

```
&VOLUME $mkt.abc
```

SECTION 5

C

This section describes the C identifier classes and provides a sample C program and its cross-reference listing.

C IDENTIFIERS

The CROSSREF utility indexes C programs according to the identifier classes listed in Table 5-1.

Table 5-1 also shows the default settings for each class and what C data types correspond to each of these classes.

Table 5-1. C Identifier Classes

CROSSREF Class	Default Setting	C Type
CONSTANTS	ON	Enumeration constants
FUNCTIONS	ON	Functions
MACROS	ON	#define(s)
TYPES	ON	User-defined and compiler-defined (anonymous) types
VARIABLES	ON	All variables

NOTE

CROSSREF includes a data type description for all identifier classes except MACROS and FUNCTIONS in the output listing.

SAMPLE LISTING

The following example starts CROSSREF, scans the file named WCC, and generates a listing to $s.#lp:

```
17> CROSSREF
CROSSREF-CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1985, 1986
&SET LANGUAGE c
&SCAN wcc
&GENERATE /OUT $s.#lp/
&EXIT
18>
```

On the following pages, you can see the program and its cross-reference listing.

Figure 5-1 shows the C program that CROSSREF scanned to produce the cross-reference listing.

```
 1     /*
 2      * usage:  wc <file1> [<file2>...]
 3      *
 4      * This program will count the number of lines, words, and characters in
 5      * a text file and report them to standard out.  When multiple files are
 6      * specified a running total of all of the files is printed in addition
 7      * to the individual files' totals.
 8      */
 9
10     #pragma runnable
11
12     #include <stdioh>
13     #include <ctypeh>
14     #include <stdlibh>
15
16     #define TRUE        1
17     #define FALSE       0
18
19     #define MAXLINE     256   /* maximum length of an input line */
20
21     long total_lines, total_words, total_chars;
22
23     void count(FILE *fp, char *file);
24
25     int main(int argc, char *argv[])
26     {
27     int i;
28     FILE *fp;
29
30     if (argc < 2)
31          {
32          fprintf(stderr, "usage:  wc <file1> [<file2>...]\n");
33          return EXIT_SUCCESS;
34          }
35     for (i = 1; i < argc; i++)
36          {
37          if ((fp = fopen(argv[i], "r")) == NULL)
38               {
39               fprintf(stderr, "can't open %s\n", argv[i]);
40               continue;
41               }
42          count(fp, argv[i]);
43          if (fclose(fp) < 0)
44               fprintf(stderr, "can't close %s\n", argv[i]);
45          }
46     printf("\n*** Total:  %ld line(s), %ld word(s), %ld character(s)\n",
47            total_lines,total_words,total_chars);
48     }
49
50     /*
51      * Count the number of lines, words, and characters in a text file.
52      */
53     void count(FILE *fp, char *file)
54     {
55     unsigned int nl;
56     long nw, nc;
```

Figure 5-1. C Sample Program (Page 1 of 2)

```
57     char line[MAXLINE];
58     int inword;
59     char *p;
60
61     for (nl = 0, nw = 0, nc = 0; fgets(line, MAXLINE, fp) != NULL; nl++)
62          {
63          for (p = line, inword = FALSE; *p != '\0'; p++, nc++)
64               if (!inword && isalnum(*p))
65                    {
66                    nw++;
67                    inword = TRUE;
68                    }
69               else if (isspace(*p)
70                  || (ispunct(*p) && !((*p == '\'' || *p == '-')
71                  && (isalpha(*(p - 1)) && isalpha(*(p + 1))))))
72                    inword = FALSE;
73          }
74     total_lines += nl;
75     total_words += nw;
76     total_chars += nc;
77     printf("%s:  %d line(s), %ld word(s), %ld character(s)\n",
78            file, nl, nw, nc);
79     }
```

Figure 5-1. C Sample Program (Page 2 of 2)

Figure 5-2 shows the first page of the cross-reference listing. This is the cross-reference file list. It lists the name of each source file scanned. In this example four files were scanned. WCC is the C source code and STDIOH, CTYPEH, and STDLIBH are C library declaration files. See your *C Reference Manual* for details on the library declaration files.

```
PAGE 1


CROSSREF - CROSS-REFERENCE PROGRAM - T9622C00 - (15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1984, 1985, 1986


FILE NO.     FILE NAME
  [1]        $EM2.UCREF.WCC
  [2]        $RTOOL.TOOLSC10.STDIOH          WCC[1]              12
  [3]        $RTOOL.TOOLSC10.CTYPEH          WCC[1]              13
  [4]        $RTOOL.TOOLSC10.STDLIBH         WCC[1]              14
```

Figure 5-2. CROSSREF Listing--File List

The identifier list makes up the rest of the cross-reference listing. See Figure 5-3 on the following page. The identifier list describes each identifier in alphabetic order, showing:

- How it is defined (its attributes)
- Where it is defined (file name and number and line number)
- Where and how it is used in the program

Look at the entry for the identifier named *count*. The identifier reference line indicates that the function is defined in the file WCC at line 54 (indicated by a code D). It is declared at line 23 and called at line 42 (indicated by a code M).

Look at the entry for the identifier named *fp*. The identifier header indicates that it is an automatic variable that is a pointer to a structure. It is defined in the file WCC at line 28 (indicated by code D). It is referenced at lines 37, 42, and 43 (indicated by code M).

```
PAGE 2
NAME AND NAME QUALIFIER          LANGUAGE-DEPENDENT ATTRIBUTES                    DEFINITION POINT


106 TOTAL SYMBOLS COLLECTED WITH 205 TOTAL REFERENCES COLLECTED


 EXIT_SUCCESS                                                                     STDLIBH[4]   23
      WCC[1]          33 M

 FALSE                                                                            WCC[1]       17
      WCC[1]          63 M      72 M

 FILE                                                                             STDIOH[2]    35
      WCC[1]          23 M      28 M       53 M
      STDIOH[2]       37 M      65 M       68 M      73 M      76 M     81 M      84 M     89 M      95 M
                      98 M      101 M      104 M     113 M     116 M    121 M     128 M    132 M     142 M
                      147 M     151 M      154 M     159 M     169 M    172 M     183 M    189 M     194 M

 MAXLINE                                                                          WCC[1]       19
      WCC[1]          57 M      61 M

 NULL                                                                             STDIOH[2]    11
      WCC[1]          37 M      61 M

 TRUE                                                                             WCC[1]       16
      WCC[1]          67 M

 _L                                                                               CTYPEH[3]    16
      WCC[1]          64 M      71 M

 _N                                                                               CTYPEH[3]    17
      WCC[1]          64 M

 _P                                                                               CTYPEH[3]    19
      WCC[1]          70 M

 _S                                                                               CTYPEH[3]    18
      WCC[1]          69 M

 _U                                                                               CTYPEH[3]    15
      WCC[1]          64 M      71 M

 __IOB                          pointer to struct, in block __IOB                 WCC[1]       32
      WCC[1]          32 M      39 M       44 M
      STDIOH[2]       37 M

 _ctype                         array[] of uns char, in block ._ctype             WCC[1]       64
      WCC[1]          64 M      69 M       70 M      71
      CTYPEH[3]       24 M

 argc  of main                  short formal                                      WCC[1]       26
      WCC[1]          26 D      30 M      35 M
```

Figure 5-3. CROSSREF Listing--Identifier List (Page 1 of 3)

```
PAGE 3
NAME AND NAME QUALIFIER     LANGUAGE-DEPENDENT ATTRIBUTES                    DEFINITION POINT

argv  of main                pointer to pointer to uns char formal            WCC[1]        26
     WCC[1]         26 D    37 M        39 M        42 M        44 M

count                                                                         WCC[1]        54
     WCC[1]         23 M    42 M        54 D

file  of count               pointer to uns char formal                       WCC[1]        54
     WCC[1]         54 D    78 M

fp  of main                  pointer to struct automatic                      WCC[1]        28
     WCC[1]         28 D    37 M        42 M        43 M

fp  of count                 pointer to struct formal                         WCC[1]        54
     WCC[1]         54 D    61 M

fprintf
     WCC[1]         32 M    39 M        44 M
     STDIOH[2]      65 M

i of main                    short automatic                                  WCC[1]        27
     WCC[1]         27 D    35 M        37 M        39 M        42 M     44 M

inword   of count            short automatic                                  WCC[1]        58
     WCC[1]         58 D    63 M        64 M        67 M        72 M

isalnum                                                                       CTYPEH[3]     33
     WCC[1]         64 M

isalpha                                                                       CTYPEH[3]     26
     WCC[1]         71 M

ispunct                                                                       CTYPEH[3]     32
     WCC[1]         70 M

isspace                                                                       CTYPEH[3]     31
     WCC[1]         69 M

line   of count              array[256] of uns char automatic                 WCC[1]        57
     WCC[1]         57 D    61 M        63 M

nc   of count                long automatic                                   WCC[1]        56
     WCC[1]         56 D    61 M        64 M        76 M        78 M

nl   of count                uns short automatic                              WCC[1]        55
     WCC[1]         55 D    61 M        74 M        78 M

nw   of count                long automatic                                   WCC[1]        56
     WCC[1]         56 D    61 M        66 M        75 M        78 M

p  of count                  pointer to uns char automatic                    WCC[1]        59
     WCC[1]         59 D    63 M        64 M        69 M        70 M     71 M
```

Figure 5-3. CROSSREF Listing--Identifier List (Page 2 of 3)

```
PAGE 4
NAME AND NAME QUALIFIER        LANGUAGE-DEPENDENT ATTRIBUTES              DEFINITION POINT


 printf
      WCC[1]          46 M     77 M
      STDIOH[2]       64 M

 size_t                                                                   STDLIOH[2]    19
      STDIOH[2]       137 M    140 M    141 M   164 M   167 M   168 M
      STDLIBH[4]      54 M     55 M     62 M    70 M

 stderr                                                                   STDIOH[2]     41
      WCC[1]          32 M     39 M     44 M

 toascii                                                                  CTYPEH[3]     43
      WCC[1]          64 M     69 M     70 M    71 M

 total_chars                long, in block total_chars                    WCC[1]        21
      WCC[1]          21 D     47 M     76 M

 total_lines                long, in block total_lines                    WCC[1]        21
      WCC[1]          21 D     47 M     74 M

 total_words                long, in block total_words                    WCC[1]        21
      WCC[1]          21 D     47 M     75 M
```

Figure 5-3. CROSSREF Listing--Identifier List (Page 3 of 3)

SECTION 6

COBOL 74

This section describes the COBOL 74 identifier classes and provides a sample COBOL 74 program and its cross-reference listing. It also describes the compiler attributes that might appear in a cross-reference listing.

COBOL 74 IDENTIFIERS

The CROSSREF utility indexes COBOL 74 programs according to the identifier classes listed in Table 6-1.

Table 6-1 also shows the the default settings for each identifier class and what COBOL 74 data types correspond to each of these classes.

Table 6-1. COBOL 74 Identifier Classes

CROSSREF Class	Default Setting	COBOL 74 Type
CONDITIONS	ON	Condition names
FILES	ON	COBOL 74 file names
FUNCTIONS	ON	Routines that return a value
INDEXES	ON	Index names
LITERALS	OFF	Numeric and nonnumeric constants
MNEMONICS	ON	Mnemonic names, alphabet names
PROCEDURES	ON	PROGRAMS
PROGLABELS	ON	Labels, procedure names (paragraph names, section names)
VARIABLES	ON	Data names

Notice that the default setting for LITERALS is OFF. If you want numeric and nonnumeric constants to appear in the cross-reference listing, you must set LITERALS to ON using the SET command.

By default, CROSSREF does not report unreferenced identifiers for COBOL 74, COBOL85, or SCREEN COBOL. If you want unreferenced identifiers to appear, you must set the UNREF attribute specification to ON or ONLY. If you set UNREF to ON, CROSSREF collects all identifiers, referenced and unreferenced, that belong to all classes set to ON. If you set UNREF to ONLY, CROSSREF collects only the unreferenced identifiers that belong to all classes set to ON.

USING COMPILER DIRECTIVES IN CROSSREF

Because CROSSREF actually invokes the COBOL 74 compiler to collect the identifier information, you might need to pass a default library file name or one or more directives to the compiler. (You supply the library file name for COPY statements that do not specify one.)

The SET DIRECTIVES command enables you to pass one or more compiler directives to the COBOL 74 compiler while the SET LIBRARY command lets you pass a default library file name. In the following example, the SET DIRECTIVES command sets the ANSI formatting directive and a conditional compilation toggle; the SET LIBRARY command alters the default copy library name from COPYLIB to MYLIB.

```
12> CROSSREF
CROSSREF-CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1985, 1986
&SET LANGUAGE cobol
&SET LIBRARY mylib
&SET DIRECTIVES "ANSI;SETTOG 1"
&SCAN bprog
&GENERATE /OUT $s.#cros/
&EXIT
13>
```

SAMPLE LISTING

The following example invokes CROSSREF, scans the file named COBEX, and generates a listing to $s.#lp:

```
14> CROSSREF
CROSSREF-CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1985, 1986
&SET LANGUAGE cobol
&SCAN cobex
&GENERATE /OUT $s.#lp/
&EXIT
15>
```

On the following pages, you can see the program and its cross-reference listing. The listing includes all identifier classes except literals.

Figure 6-1 shows the COBOL 74 program that CROSSREF scanned to produce the cross-reference listing.

```
1      ?SAVE ALL
2      ?SEARCH $SYSTEM.SYSTEM.COBOLLIB
3       IDENTIFICATION DIVISION.
4         PROGRAM-ID.    FUPPERWARE.
5         AUTHOR.        ANN COBOL.
6         INSTALLATION.  TRANSACTIONS ANONYMOUS.
7         DATE-WRITTEN.  29 FEBRUARY 1984.
8         DATE-COMPILED.
9      ****************************************************************
10     *                                                              *
11     *  This program creates a FUP process and watches for its      *
12     *  termination.                                                *
13     *                                                              *
14     ****************************************************************
15      ENVIRONMENT DIVISION.
16        CONFIGURATION SECTION.
17          SOURCE-COMPUTER.  TANDEM TXP.
18          OBJECT-COMPUTER.  TANDEM TXP.
19          SPECIAL-NAMES.
20        INPUT-OUTPUT SECTION.
21          FILE-CONTROL.
22             SELECT MESSAGE-IN-FILE
23                    ASSIGN TO "$RECEIVE"
24                    FILE STATUS IS RECEIVE-FILE-STATUS.
25          I-O-CONTROL.
26          RECEIVE-CONTROL.
27            TABLE OCCURS 1 TIMES
28            SYNCDEPTH LIMIT IS 1
29            REPLY CONTAINS 32 CHARACTERS
30            MESSAGE SOURCE IS MESSAGE-SOURCE-REC
32            REPORT SYSTEM MESSAGES.
34      DATA DIVISION.
35        FILE SECTION.
36          FD MESSAGE-IN-FILE
37             LABEL RECORDS ARE OMITTED.
39          01 MESSAGE-IN.
40             05 SYS-MSG-CODE                PIC S9(4) COMP.
41                88 SYS-MSG-STOP             VALUE -5.
42                88 SYS-MSG-ABEND            VALUE -6.
43             05 SYS-MSG-PROCNAME            PIC X(6).
44             05 FILLER                      PIC X(1024).
47        WORKING-STORAGE SECTION.
48          01 MESSAGE-SOURCE-REC.
49             05 SYSTEM-FLAG       PIC S9  COMP.
50             05 ENTRY-NUMBER      PIC 999 COMP.
51             05 FILLER            PIC X(4).
52             05 PROCESS-ID.
53                10 PROCESS-NAME   PIC X(6).
54                10 CPU-PIN.
55                   15 CPU-PART    PIC X.
56                   15 PIN-PART    PIC X.
57             05 FILLER            PIC X(16).
60          01 CPU-PIN-REDEF.
61             05 ALPHA-CPU.
62                10 CPU-HIGH-BYTE     PIC X.
63                10 CPU-LOW-BYTE      PIC X.
```

Figure 6-1. COBOL 74 Sample Program (Page 1 of 4)

```
64             05 NUMERIC-CPU          REDEFINES ALPHA-CPU
65                                     PIC S9999 COMP.
66             05 ALPHA-PIN.
67                10 PIN-HIGH-BYTE     PIC X.
68                10 PIN-LOW-BYTE      PIC X.
69             05 NUMERIC-PIN          REDEFINES ALPHA-PIN
70                                     PIC S9999 COMP.
72          01 FILE-DATA.
73             05 RECEIVE-FILE-STATUS.
74                10 STAT-1                    PIC 9.
75                   88  CLOSE-FROM-REQUESTER VALUE 1 THRU 3.
76                10 STAT-2                    PIC 9.
77          01  SAVE-MESSAGE-STUFF.
78             05 FUP                  PIC X(21)
79                                         VALUE "$SYSTEM.SYSTEM.FUP".
80             05 FUP-NAME             PIC X(5)
81                                         VALUE SPACES.
82             05 SEND-ALL-MSGS        PIC S9(4)
83                                         VALUE ZERO COMP.
84             05 PRIORITY-EQ-MINE     PIC S9(4)
85                                         VALUE 0 COMP.
86             05 PROCESSOR-EQ-MINE    PIC S9(4)
87                                         VALUE -1 COMP.
88             05 MEMORY-AS-USUAL      PIC S9(4)
89                                         VALUE ZERO COMP.
90             05 SU-ERROR             PIC S9(4)
91                                         VALUE ZERO COMP.
92             05 NEWPROCESS-ERR-LEFT  PIC 9(4).
93             05 NEWPROCESS-ERR-RIGHT PIC 9(4).
94             05 FUP-FAILED           PIC X(19)
95                                         VALUE "Failed to start FUP".
96             05 STRING-PORTION       PIC X(7)
97                                         VALUE "STRING".
98             05 INFO-COMMAND         PIC X(7)
99                                         VALUE "INFO *".
100            05 STARTUP-RESULT       PIC S9(4)
101                                        VALUE ZERO COMP.
102            05 NULL-CPLIST          PIC S9(9)
103                                        VALUE ZERO COMP.
104
106     PROCEDURE DIVISION.
107     DECLARATIVES.
108       HANDLE-INFILE-ERRORS SECTION.
109         USE AFTER STANDARD ERROR PROCEDURE ON MESSAGE-IN-FILE.
110       INFILE-ERROR.
111         IF STAT-1 = 1
112            DISPLAY "EOF on $RECEIVE"
113         ELSE
114            DISPLAY "RECEIVE FILE ERROR STATUS = "
115                    RECEIVE-FILE-STATUS
116         .
117     END DECLARATIVES.
119
120     AA SECTION.
121     AA-1.
122        OPEN INPUT MESSAGE-IN-FILE.
```

Figure 6-1. COBOL 74 Sample Program (Page 2 of 4)

```
123        MOVE ZERO TO SU-ERROR
124                     SYS-MSG-CODE.
125
127     * Inject INFO command into STARTUP message to pass to FUP
128        ENTER "PUTSTARTUPTEXT"
129              USING STRING-PORTION,
130                    INFO-COMMAND,
131                    NULL-CPLIST
132              GIVING STARTUP-RESULT.
133
134     * Start FUP
135        ENTER "CREATEPROCESS"
136              USING FUP,
137                    FUP-NAME,
138                    SEND-ALL-MSGS,
139                    PRIORITY-EQ-MINE,
140                    PROCESSOR-EQ-MINE,
141                    MEMORY-AS-USUAL,
142                    OMITTED
143              GIVING SU-ERROR.
144
145     * Await termination of FUP, or report it never started
146        IF SU-ERROR = 0
147           PERFORM WATCH
148             UNTIL SYS-MSG-STOP
149                OR SYS-MSG-ABEND
150        ELSE
151           PERFORM DISPLAY-STARTUP-FAILURE
152        .
153
154        STOP RUN.
156
158     DISPLAY-STARTUP-FAILURE.
159        IF SU-ERROR = 1
160           DISPLAY FUP-FAILED
161                   " -- REQUIRED PARAMETER MISSING OR ILLEGAL"
162        ELSE IF SU-ERROR = 2
163                DISPLAY FUP-FAILED
164                        " -- ILLEGAL PROGRAM FILE NAME ("
165                        FUP ")"
166        ELSE IF SU-ERROR = 3
167                DISPLAY FUP-FAILED
168                        " -- INFILE, OUTFILE, OR DEFAULT VOLUME"
169                DISPLAY " NAME CANNOT BE CONVERTED TO NETWORK FORM"
170        ELSE IF SU-ERROR < 256
171                DISPLAY FUP-FAILED
172                        " -- File management error #"
173                        SU-ERROR
174        ELSE
175     *        -- Received raw error from NEWPROCESS system procedure.
176     *        -- Decompose it into left byte and right byte values.
177              DIVIDE        SU-ERROR
178                  BY        256
179                  GIVING    NEWPROCESS-ERR-LEFT
180                  REMAINDER NEWPROCESS-ERR-RIGHT.
180.1
```

Figure 6-1. COBOL 74 Sample Program (Page 3 of 4)

```
181             DISPLAY FUP-FAILED
182                     " -- NEWPROCESS error #"
183                     SU-ERROR
184                     " = ("
185                     NEWPROCESS-ERR-LEFT
186                     ","
187                     NEWPROCESS-ERR-RIGHT
188                     ")"
189         .
191
192     WATCH.
193        READ MESSAGE-IN-FILE.
194        PERFORM CAPTURE-CPU-PIN.
195        DISPLAY "-----"
196                "(" NUMERIC-CPU "," NUMERIC-PIN ")"
197                "-----"
198                SYS-MSG-CODE.
199
200     CAPTURE-CPU-PIN.
201        MOVE CPU-PART TO CPU-LOW-BYTE.
202        MOVE LOW-VALUES TO CPU-HIGH-BYTE.
203        MOVE PIN-PART TO PIN-LOW-BYTE.
204        MOVE LOW-VALUES TO PIN-HIGH-BYTE.
```

Figure 6-1. COBOL 74 Sample Program (Page 4 of 4)

Figure 6-2 shows the first page of the cross-reference listing. This is the cross-reference file list. It lists the name of each source file that CROSSREF scanned. In this example, only one file, COBEX, was scanned.

```
PAGE 1


CROSSREF - CROSS-REFERENCE PROGRAM - T9622C00 - (15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1984, 1985, 1986


FILE NO.     FILE NAME
  [1]        $EM2.UCREF.COBEX
```

Figure 6-2. CROSSREF Listing--File List

The identifier list makes up the rest of the cross-reference listing. See Figure 6-3. The identifier list describes each identifier in alphabetic order, showing:

- How it is defined (its attributes)
- Where it is defined (file name and number and line number)
- Where and how it is used in the program

Look at the entry for CAPTURE-CPU-PIN. The identifier header indicates that it is the name of a paragraph that belongs to the section named AA. It is defined in the file COBEX at line 200.

The reference line indicates that the identifier is referenced in the file COBEX at line 194.

Now look at the entry for the identifier named MEMORY-AS-USUAL. The identifier header indicates that it is part of the group data item SAVE-MESSAGE-STUFF. It is a level-5 numeric computation item in the Working-Storage section. Its size is 2 bytes, indicating that it is a 1-word integer. Its offset is 32, which means that it begins at the thirty-second position of the record SAVE-MESSAGE-STUFF.

The reference line indicates that the identifier is referenced in the file COBEX at line 141.

```
PAGE 2
NAME AND NAME QUALIFIER           LANGUAGE-DEPENDENT ATTRIBUTES                DEFINITION POINT


50 TOTAL SYMBOLS COLLECTED WITH 55 TOTAL REFERENCES COLLECTED

 ALPHA-CPU OF CPU-PIN-REDEF        05    AN   GROUP WSS  SIZE=2     OFFSET=0     COBEX[1]       61
      COBEX[1]        64 M

 ALPHA-PIN OF CPU-PIN-REDEF        05    AN   GROUP WSS  SIZE=2     OFFSET=2     COBEX[1]       66
      COBEX[1]        69 M

 CAPTURE-CPU-PIN OF AA             PARA                                          COBEX[1]       200
      COBEX[1]        194 M

 CPU-HIGH-BYTE OF ALPHA-CPU OF CPU-PIN-REDEF                                     COBEX[1]       62
                                   10    AN   DISP  WSS  SIZE=1     OFFSET=0
      COBEX[1]        202 W

 CPU-LOW-BYTE OF ALPHA-CPU OF CPU-PIN-REDEF                                      COBEX[1]       63
                                   10    AN   DISP  WSS  SIZE=1     OFFSET=1
      COBEX[1]        201 W

 CPU-PART  OF CPU-PIN OF PROCESS-ID OF MESSAGE-SOURCE-REC                        COBEX[1]       55
                                   15    AN   DISP  WSS  SIZE=1     OFFSET=14
      COBEX[1]        201

 DISPLAY-STARTUP-FAILURE OF AA     PARA                                          COBEX[1]       158
      COBEX[1]        151 M

 FUP OF SAVE-MESSAGE-STUFF         05    AN   DISP  WSS  SIZE=21    OFFSET=0     COBEX[1]       78
      COBEX[1]        136 M       165

 FUP-FAILED  OF SAVE-MESSAGE-STUFF                                               COBEX[1]       94
                                   05    AN   DISP  WSS  SIZE=19    OFFSET=44
      COBEX[1]        160         163         167         171     181

 FUP-NAME  OF SAVE-MESSAGE-STUFF   05    AN   DISP  WSS  SIZE=5     OFFSET=21    COBEX[1]       80
      COBEX[1]        137 M

 INFO-COMMAND  OF SAVE-MESSAGE-STUFF                                 OFFSET=70    COBEX[1]       98
                                   05    AN   DISP  WSS  SIZE=7
      COBEX[1]        130 M

 MEMORY-AS-USUAL OF SAVE-MESSAGE-STUFF                                           COBEX[1]       88
                                   05    NM   COMP  WSS  SIZE=2     OFFSET=32
      COBEX[1]        141 M

 MESSAGE-IN-FILE                   FD    IS $RECEIVE                             COBEX[1]       36
                                                    RECORD=1032 F ORG=SEQ ACC=SEQ
      COBEX[1]         22 M       109 M       122 M       193 M

 MESSAGE-SOURCE-REC                01    AN   GROUP WSS  SIZE=32    OFFSET=0     COBEX[1]       48
```

Figure 6-3. CROSSREF Listing--Identifier List (Page 1 of 3)

```
PAGE 3
NAME AND NAME QUALIFIER        LANGUAGE-DEPENDENT ATTRIBUTES              DEFINITION POINT


   COBEX[1]       32 M

 NEWPROCESS-ERR-LEFT OF SAVE-MESSAGE-STUFF                                 COBEX[1]   92
                                05   NM   DISP  WSS  SIZE=4    OFFSET=36
   COBEX[1]       179 W         185

 NEWPROCESS-ERR-RIGHT  OF SAVE-MESSAGE-STUFF                               COBEX[1]   93
                                05   NM   DISP  WSS  SIZE=4    OFFSET=40
   COBEX[1]       180 W         187

 NULL-CPLIST OF SAVE-MESSAGE-STUFF                                         COBEX[1]   102
                                05   NM   COMP  WSS  SIZE=4    OFFSET=80
   COBEX[1]       131 M

 NUMERIC-CPU OF CPU-PIN-REDEF     05   NM   COMP  WSS  SIZE=2    OFFSET=0     COBEX[1]   64
   COBEX[1]       196

 NUMERIC-PIN OF CPU-PIN-REDEF     05   NM   COMP  WSS  SIZE=2    OFFSET=2     COBEX[1]   69
   COBEX[1]       196

 PIN-HIGH-BYTE OF ALPHA-PIN OF CPU-PIN-REDEF                               COBEX[1]   67
                                10   AN   DISP  WSS  SIZE=1    OFFSET=2
   COBEX[1]       204 W

 PIN-LOW-BYTE  OF ALPHA-PIN OF CPU-PIN-REDEF                               COBEX[1]   68
                                10   AN   DISP  WSS  SIZE=1    OFFSET=3
   COBEX[1]       203 W

 PIN-PART  OF CPU-PIN OF PROCESS-ID OF MESSAGE-SOURCE-REC                  COBEX[1]   56
                                15   AN   DISP  WSS  SIZE=1    OFFSET=15
   COBEX[1]       203

 PRIORITY-EQ-MINE  OF SAVE-MESSAGE-STUFF                                   COBEX[1]   84
                                05   NM   COMP  WSS  SIZE=2    OFFSET=28
   COBEX[1]       139 M

 PROCESSOR-EQ-MINE OF SAVE-MESSAGE-STUFF                                   COBEX[1]   86
                                05   NM   COMP  WSS  SIZE=2    OFFSET=30
   COBEX[1]       140 M

 RECEIVE-FILE-STATUS OF FILE-DATA 05   AN   GROUP WSS  SIZE=2    OFFSET=0     COBEX[1]   73
   COBEX[1]       24 M          115

 SEND-ALL-MSGS OF SAVE-MESSAGE-STUFF                                       COBEX[1]   82
                                05   NM   COMP  WSS  SIZE=2    OFFSET=26
   COBEX[1]       138 M

 STARTUP-RESULT  OF SAVE-MESSAGE-STUFF                                     COBEX[1]   100
                                05   NM   COMP  WSS  SIZE=2    OFFSET=78
   COBEX[1]       132 W
```

Figure 6-3. CROSSREF Listing--Identifier List (Page 2 of 3)

```
PAGE 4
NAME AND NAME QUALIFIER        LANGUAGE-DEPENDENT ATTRIBUTES                DEFINITION POINT


 STAT-1  OF RECEIVE-FILE-STATUS OF FILE-DATA                                  COBEX[1]   74
                                10   NM   DISP  WSS  SIZE=1    OFFSET=0
      COBEX[1]       111

 STRING-PORTION  OF SAVE-MESSAGE-STUFF                                        COBEX[1]   96
                                05   AN   DISP  WSS  SIZE=7    OFFSET=63
      COBEX[1]       129 M

 SU-ERROR  OF SAVE-MESSAGE-STUFF  05   NM   COMP  WSS  SIZE=2    OFFSET=34    COBEX[1]   90
      COBEX[1]       123 W       143 W     146     159     162     166     170     173     177
                     183

 SYS-MSG-ABEND OF SYS-MSG-CODE OF MESSAGE-IN OF MESSAGE-IN-FILE               COBEX[1]   42
                                88   NM   COMP  FS   SIZE=2    OFFSET=0
     COBEX[1]        149

 SYS-MSG-CODE  OF MESSAGE-IN-OF MESSAGE-IN-FILE                               COBEX[1]   40
                                05   NM   COMP  FS   SIZE=2    OFFSET=0
     COBEX[1]        124 W       198

 SYS-MSG-STOP  OF SYS-MSG-CODE OF MESSAGE-IN OF MESSAGE-IN-FILE               COBEX[1]   41
                                88   NM   COMP  FS   SIZE=2    OFFSET=0
     COBEX[1]        148

 WATCH OF AA                    PARA                                          COBEX[1]   192
     COBEX[1]        147 M
```

Figure 6-3. CROSSREF Listing--Identifier List (Page 3 of 3)

COMPILER ATTRIBUTES

CROSSREF collects identifier attribute information from the COBOL 74 compiler and prints it in the identifier header. The attributes are explained below.

Alphabet-Name

If the identifier is an alphabet-name, it is described in the header as MNEM. (MNEM might also indicate a mnemonic-name. See "Mnemonic-Name" in this section.)

Alphabet-name has no significance on the Tandem implementation of COBOL 74 because the ASCII character set is the only one used. When used to document the ASCII set, it is defined as NATIVE or STANDARD-1.

Condition-Name

If the identifier is a condition-name, it is defined in one of two ways: either as a Level 88 Item or as COND IS SWITCH-*ss*.

If the identifier is a Level 88 Item, it is described in the header as

88 *category usage loc* SIZE=*size* [V] OFFSET=*offset* [*i* INX[S]]

category, *usage*, *loc*, *size*, V, *offset*, and *i* are all explained under "Data-Name" in this section.

The identifier qualifier indicates the parent conditional variable of the Level 88 item.

If the identifier represents a condition-name that tests an external switch, it is described in the header as

COND IS SWITCH-*ss* {ON }
{OFF}

ss is a decimal integer from 1 to 15 that identifies which switch the program is testing. ON or OFF indicates which state is being tested.

Data-Name

If the identifier is a data item, it is described in the header as

ln category usage loc SIZE=*size* [V] OFFSET=*offset* [*i* INX[S]] [SPCL-REG]

ln is the level number of the data-item.

category is one of the following:

AL	Alphabetic
ALE	Alphabetic with Bs in its PICTURE
AN	Alphanumeric
ANE	Alphanumeric edited
NM	Numeric
NME	Numeric edited

usage is one of the following:

GROUP	Group item
DISP	DISPLAY item
COMP	COMPUTATIONAL item
INX	INDEX item

loc shows the site of the identifier declaration and is one of the following:

ESS	Extended-Storage Section
FS	File Section
WSS	Working-Storage Section
LS	Linkage Section

size is the size in bytes of the identifier's value in decimal notation (up to nine places without leading zeros).

V indicates that the size is variable, due to an OCCURS DEPENDING ON clause.

offset is the byte offset of the value of the data-item from the relevant base address; for example, from the start of the containing record. This corresponds to the identifier's position within a record. The offset is shown in decimal notation up to nine places without leading zeros.

i INX[S] appears if the references to the data-item require subscripting or indexing. In that case, *i* is a number from 1 to 7 showing the number of subscripts. If the value of *i* is 1, INX appears instead of INXS.

SPCL-REG appears if the data-name is a Special Register.

File-Name

If the identifier is a file name, it is described in the header as

```
   FD IS external-file-name                          kk KEY[S]
                                                     SET s:pp
       BLOCK=b [R]  RECORD=r {F} ORG=org ACC=acc
                             {V}
```

external-file-name is the name that you specified in the ASSIGN clause of the File-Control entry.

kk KEY[S] indicates the number of record keys the file has in decimal notation. If the value of *kk* is 1, KEY appears instead of KEYS.

SET *s:pp* appears if the file is a member of a multiple file tape set. *s* is a digit that identifies the set and *pp* is a decimal integer from 1 to 31 that identifies the file's position in that set.

If the block size is not equal to the record size, the BLOCK field shows the block size in decimal notation. R indicates a record multiple size; otherwise, it is the block size in bytes.

The RECORD entry shows record size in bytes, expressed in decimal notation. F indicates the records are of fixed length; V indicates the records are of variable length, each having up to the displayed maximum size.

org marks the file's organization. It can be:

SEQ	Sequential
REL	Relative
INX	Indexed

acc marks the file's access mode. It can be:

SEQ	Sequential
RAN	Random
DYN	Dynamic

Index-Name

If the identifier is the name of an index item, it is described in the header as INX. The identifier qualifier shows the name of the table item that the index-name belongs to. An index item is always two bytes long.

Literals

CROSSREF prints literals before any other identifiers. They are shown exactly as they appear in the source file; if they have quotation marks in the source file, they have them in the cross-reference listing.

Mnemonic-Name

If the identifier is a mnemonic-name, it is described in the header as MNEM.

When the mnemonic refers to a channel name, CHANNEL-*cc* identifies the channel.

When the mnemonic refers to an external switch, SWITCH-*ss* identifies the switch.

Paragraph-Name

If the identifier is the name of a paragraph, it is described in the header as PARA. The identifier qualifier also indicates what section (if any) the paragraph belongs to.

If code is generated for the program, %*offset* shows the code offset for this paragraph relative to the program's base. The offset is an octal number of six digits.

Section-Name

If the identifier is the name of a section, it is described in the header as SECT.

If code is generated for the program, %*offset* shows the code offset for this section relative to the program's base. The offset is an octal number of six digits.

SECTION 7

COBOL85

This section describes the COBOL85 identifier classes and provides a sample COBOL85 program and its cross-reference listing. It also describes the compiler attributes that might appear in a cross-reference listing.

COBOL85 IDENTIFIERS

The CROSSREF utility indexes COBOL85 programs according to the identifier classes listed in Table 7-1.

Table 7-1 also shows the the default settings for each identifier class and what COBOL85 data types correspond to each of these classes.

Table 7-1. COBOL85 Identifier Classes

CROSSREF Class	Default Setting	COBOL85 Type
CONDITIONS	ON	Condition names
CONSTANTS	ON	Symbolic characters
FILES	ON	COBOL file names
FUNCTIONS	ON	Routines that return a value
INDEXES	ON	Index names
LITERALS	OFF	Numeric and nonnumeric constants
MNEMONICS	ON	Mnemonic names, alphabet names, class names
PROCEDURES	ON	PROGRAMS
PROGLABELS	ON	Labels, procedure names (paragraph names, section names)
VARIABLES	ON	Data names

Notice that the default setting for LITERALS is OFF. If you want numeric and nonnumeric constants to appear in the cross-reference listing, you must set LITERALS to ON using the SET command.

By default, CROSSREF does not report unreferenced identifiers for COBOL 74, COBOL85, or SCREEN COBOL. If you want unreferenced identifiers to appear, you must set the UNREF attribute specification to ON or ONLY. If you set UNREF to ON, CROSSREF collects all identifiers, referenced and unreferenced, that belong to all classes set to ON. If you set UNREF to ONLY, CROSSREF collects only the unreferenced identifiers that belong to all classes set to ON.

(STAT-2 and CLOSE-FROM-REQUESTOR do not appear in the cross-reference listing shown in Figure 7-3 because the listing was created with the UNREF attribute specification set to OFF.)

USING COMPILER DIRECTIVES IN CROSSREF

Because CROSSREF actually invokes the COBOL85 compiler to collect the identifier information, you might need to pass a default library file name or one or more directives to the compiler. (You supply the library file name for COPY statements that do not specify one.)

The SET DIRECTIVES command enables you to pass one or more compiler directives to the COBOL85 compiler while the SET LIBRARY command lets you pass a default library file name. In the following example, the SET DIRECTIVES command sets the ANSI formatting directive and a conditional compilation toggle; the SET LIBRARY command alters the default copy library name from COPYLIB to MYLIB.

```
12> CROSSREF
CROSSREF-CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1985, 1986
&SET LANGUAGE cobol85
&SET LIBRARY mylib
&SET DIRECTIVES "ANSI;SETTOG 1"
&SCAN bprog
&GENERATE /OUT $s.#cros/
&EXIT
13>
```

SAMPLE LISTING

The following example invokes CROSSREF, scans the file named COBEX85, and generates a listing to $s.#lp:

```
16> CROSSREF
CROSSREF-CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1985, 1986
&SET LANGUAGE cobol85
&SCAN cobex85
&GENERATE /OUT $s.#lp/
&EXIT
17>
```

On the following pages, you can see the program and its cross-reference listing. The listing includes all identifier classes except literals.

Figure 7-1 shows the COBOL85 program that CROSSREF scanned to produce the cross-reference listing.

```
 1     ?SAVE ALL
 2     ?SEARCH $SYSTEM.SYSTEM.COBOLLIB
 3      IDENTIFICATION DIVISION.
 4        PROGRAM-ID.    FUPPERWARE.
 5     *  AUTHOR.        ANN COBOL.
 6     *  INSTALLATION.  TRANSACTIONS ANONYMOUS.
 7     *  DATE-WRITTEN.  24 June 1987.
 8     *  DATE-COMPILED.
 9     ****************************************************************
10     *                                                              *
11     *  This program creates a FUP process and watches for its      *
12     *  termination.                                                *
13     *                                                              *
14     ****************************************************************
15      ENVIRONMENT DIVISION.
16        CONFIGURATION SECTION.
17          SOURCE-COMPUTER.  TANDEM TXP.
18          OBJECT-COMPUTER.  TANDEM TXP.
19        SPECIAL-NAMES.
20         INPUT-OUTPUT SECTION.
21          FILE-CONTROL.
22             SELECT MESSAGE-IN-FILE
23                    ASSIGN TO "$RECEIVE"
24                    FILE STATUS IS RECEIVE-FILE-STATUS.
25          I-O-CONTROL.
26          RECEIVE-CONTROL.
27            TABLE OCCURS 1 TIMES
28            SYNCDEPTH LIMIT IS 1
29            REPLY CONTAINS 32 CHARACTERS
30            MESSAGE SOURCE IS MESSAGE-SOURCE-REC
32            REPORT SYSTEM MESSAGES.
34      DATA DIVISION.
35        FILE SECTION.
36          FD MESSAGE-IN-FILE
37             LABEL RECORDS ARE OMITTED.
39          01 MESSAGE-IN.
40             05 SYS-MSG-CODE                PIC S9(4) COMP.
41                88 SYS-MSG-STOP             VALUE -5.
42                88 SYS-MSG-ABEND            VALUE -6.
43             05 SYS-MSG-PROCNAME            PIC X(6).
44             05 FILLER                      PIC X(1024).
47        WORKING-STORAGE SECTION.
48          01 MESSAGE-SOURCE-REC.
49             05 SYSTEM-FLAG       PIC S9  COMP.
50             05 ENTRY-NUMBER      PIC 999 COMP.
51             05 FILLER            PIC X(4).
52             05 PROCESS-ID.
53                10 PROCESS-NAME   PIC X(6).
54                10 CPU-PIN.
55                   15 CPU-PART    PIC X.
56                   15 PIN-PART    PIC X.
57             05 FILLER            PIC X(16).
60          01 CPU-PIN-REDEF.
61             05 ALPHA-CPU.
62                10 CPU-HIGH-BYTE     PIC X.
63                10 CPU-LOW-BYTE      PIC X.
```

Figure 7-1. COBOL85 Sample Program (Page 1 of 4)

```
 64             05 NUMERIC-CPU          REDEFINES ALPHA-CPU
 65                                     PIC S9999 COMP.
 66             05 ALPHA-PIN.
 67                10 PIN-HIGH-BYTE     PIC X.
 68                10 PIN-LOW-BYTE      PIC X.
 69             05 NUMERIC-PIN          REDEFINES ALPHA-PIN
 70                                     PIC S9999 COMP.
 72          01 FILE-DATA.
 73             05 RECEIVE-FILE-STATUS.
 74                10 STAT-1                    PIC 9.
 75                   88  CLOSE-FROM-REQUESTER VALUE 1 THRU 3.
 76                10 STAT-2                    PIC 9.
 77          01  SAVE-MESSAGE-STUFF.
 78             05 FUP                  PIC X(21)
 79                                         VALUE "$SYSTEM.SYSTEM.FUP".
 80             05 FUP-NAME             PIC X(5)
 81                                         VALUE SPACES.
 82             05 SEND-ALL-MSGS        PIC S9(4)
 83                                         VALUE ZERO COMP.
 84             05 PRIORITY-EQ-MINE     PIC S9(4)
 85                                         VALUE 0 COMP.
 86             05 PROCESSOR-EQ-MINE    PIC S9(4)
 87                                         VALUE -1 COMP.
 88             05 MEMORY-AS-USUAL      PIC S9(4)
 89                                         VALUE ZERO COMP.
 90             05 SU-ERROR             PIC S9(4)
 91                                         VALUE ZERO COMP.
 92             05 NEWPROCESS-ERR-LEFT  PIC 9(4).
 93             05 NEWPROCESS-ERR-RIGHT PIC 9(4).
 94             05 FUP-FAILED           PIC X(19)
 95                                         VALUE "Failed to start FUP".
 96             05 STRING-PORTION       PIC X(7)
 97                                         VALUE "STRING".
 98             05 INFO-COMMAND         PIC X(7)
 99                                         VALUE "INFO *".
100             05 STARTUP-RESULT       PIC S9(4)
101                                         VALUE ZERO COMP.
102             05 NULL-CPLIST          PIC S9(9)
103                                         VALUE ZERO COMP.
104
106       PROCEDURE DIVISION.
107       DECLARATIVES.
108         HANDLE-INFILE-ERRORS SECTION.
109           USE AFTER STANDARD ERROR PROCEDURE ON MESSAGE-IN-FILE.
110         INFILE-ERROR.
111           IF STAT-1 = 1
112              DISPLAY "EOF on $RECEIVE"
113           ELSE
114              DISPLAY "RECEIVE FILE ERROR STATUS = "
115                      RECEIVE-FILE-STATUS
115.1         END-IF
116           .
117       END DECLARATIVES.
119
120       AA SECTION.
121       AA-1.
```

Figure 7-1. COBOL85 Sample Program (Page 2 of 4)

```
122         OPEN INPUT MESSAGE-IN-FILE
123         MOVE ZERO TO SU-ERROR
124                      SYS-MSG-CODE
125
127     * Inject INFO command into STARTUP message to pass to FUP
128         ENTER "PUTSTARTUPTEXT"
129               USING STRING-PORTION,
130                     INFO-COMMAND,
131                     NULL-CPLIST
132               GIVING STARTUP-RESULT
133
134     * Start FUP
135         ENTER "CREATEPROCESS"
136               USING FUP,
137                     FUP-NAME,
138                     SEND-ALL-MSGS,
139                     PRIORITY-EQ-MINE,
140                     PROCESSOR-EQ-MINE,
141                     MEMORY-AS-USUAL,
142                     OMITTED
143               GIVING SU-ERROR
144
145     * Await termination of FUP, or report it never started
146         IF SU-ERROR = 0
147            PERFORM UNTIL SYS-MSG-STOP OR SYS-MSG-ABEND
150                READ MESSAGE-IN-FILE
150.1              PERFORM CAPTURE-CPU-PIN
150.2              DISPLAY "-----"
150.3                      "(" NUMERIC-CPU "," NUMERIC-PIN ")"
150.4                      "-----"
150.5                      SYS-MSG-CODE
150.51          END-PERFORM
150.6       ELSE
151            PERFORM DISPLAY-STARTUP-FAILURE
152         END-IF
153
154         STOP RUN
155         .
156
158      DISPLAY-STARTUP-FAILURE.
159         EVALUATE SU-ERROR
159.1          WHEN 1
160               DISPLAY FUP-FAILED
161                       " -- REQUIRED PARAMETER MISSING OR ILLEGAL"
162            WHEN 2
163               DISPLAY FUP-FAILED
164                       " -- ILLEGAL PROGRAM FILE NAME ("
165                         FUP ")"
166            WHEN 3
167               DISPLAY FUP-FAILED
168                       " -- INFILE, OUTFILE, OR DEFAULT VOLUME"
169               DISPLAY " NAME CANNOT BE CONVERTED TO NETWORK FORM"
170            WHEN 4 THRU 255
171               DISPLAY FUP-FAILED
172                       " -- File management error #"
173                       SU-ERROR
```

Figure 7-1. COBOL85 Sample Program (Page 3 of 4)

```
174             WHEN OTHER
175      *        -- Received raw error from NEWPROCESS system procedure.
176      *        -- Decompose it into left byte and right byte values.
177                 DIVIDE        SU-ERROR
178                     BY        256
179                     GIVING    NEWPROCESS-ERR-LEFT
180                     REMAINDER NEWPROCESS-ERR-RIGHT
180.1
181                 DISPLAY FUP-FAILED
182                         " -- NEWPROCESS error #"
183                         SU-ERROR
184                         " = ("
185                         NEWPROCESS-ERR-LEFT
186                         ","
187                         NEWPROCESS-ERR-RIGHT
188                         ")"
189          END-EVALUATE
190          .
191
200       CAPTURE-CPU-PIN.
201          MOVE CPU-PART TO CPU-LOW-BYTE
202          MOVE LOW-VALUES TO CPU-HIGH-BYTE
203          MOVE PIN-PART TO PIN-LOW-BYTE
204          MOVE LOW-VALUES TO PIN-HIGH-BYTE
205          .
```

Figure 7-1. COBOL85 Sample Program (Page 4 of 4)

Figure 7-2 shows the first page of the cross-reference listing. This is the cross-reference file list. It lists the name of each source file that CROSSREF scanned. In this example, only one file, COBEX85, was scanned.

```
PAGE 1


CROSSREF - CROSS-REFERENCE PROGRAM - T9622C00 - (15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1984, 1985, 1986


FILE NO.     FILE NAME
  [1]        $EM2.UCREF.COBEX85
```

Figure 7-2. CROSSREF Listing--File List

The identifier list makes up the rest of the cross-reference listing. See Figure 7-3. The identifier list describes each identifier in alphabetic order, showing:

- How it is defined (its attributes)
- Where it is defined (file name and number and line number)
- Where and how it is used in the program

Look at the entry for CAPTURE-CPU-PIN on page two of the listing. The identifier header indicates that it is the name of a paragraph that belongs to the section named AA. It is defined in the file COBEX85 at line 200.

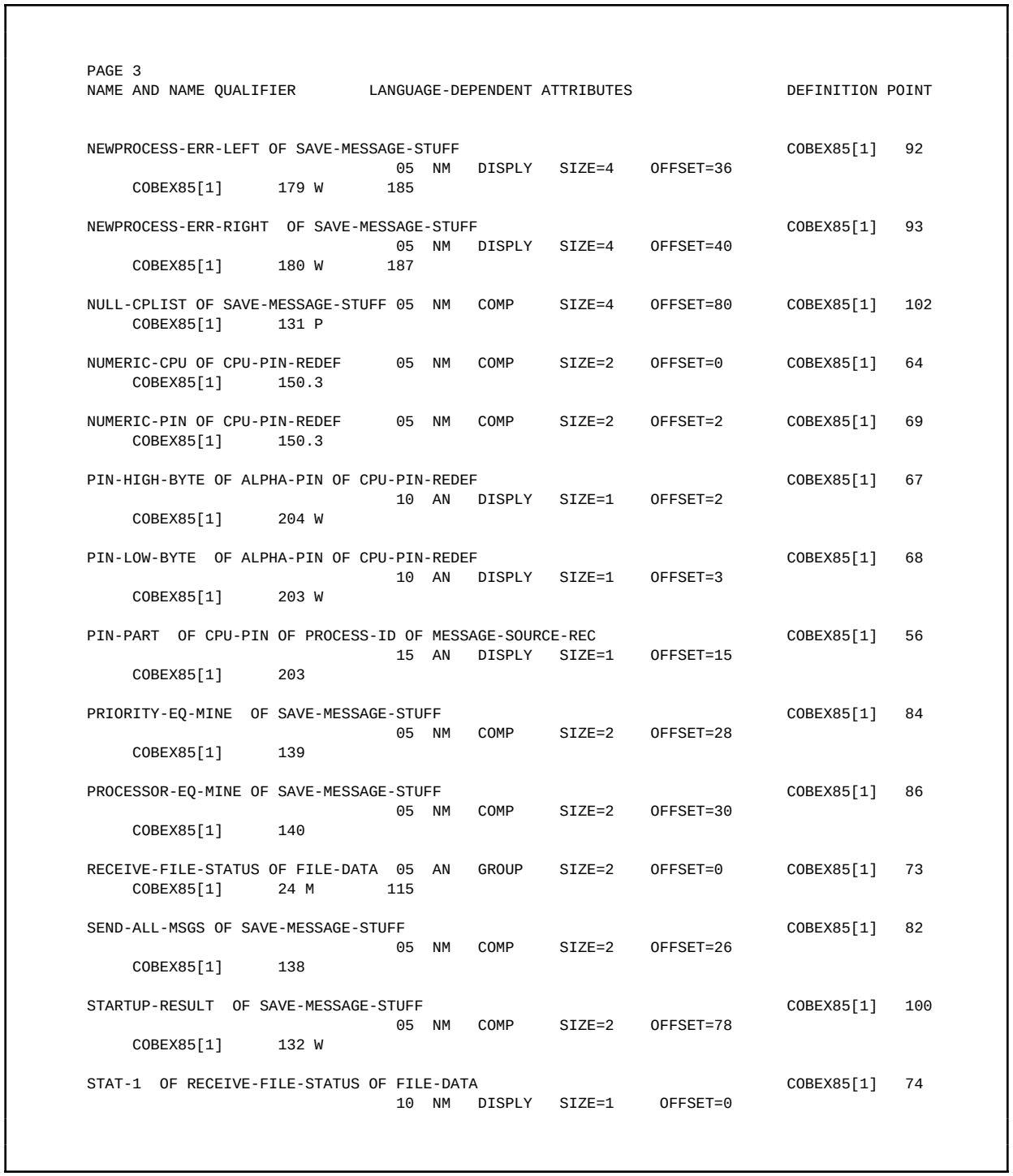
The reference line indicates that the identifier is referenced in the file COBEX85 at line 150.1 (indicated by code M).

Now look at the entry for the identifier named MEMORY-AS-USUAL on page two of the listing. The identifier header indicates that it is part of the group data item SAVE-MESSAGE-STUFF. It is a level-5 numeric computation item. Its size is 2 bytes, indicating that it is a 1-word integer. Its offset is 32, which means that it begins at the thirty-third byte of the record SAVE-MESSAGE-STUFF.

The reference line indicates that the identifier is referenced in the file COBEX85 at line 141.

```
PAGE 2
NAME AND NAME QUALIFIER           LANGUAGE-DEPENDENT ATTRIBUTES                DEFINITION POINT

 54 TOTAL SYMBOLS COLLECTED WITH 51 TOTAL REFERENCES COLLECTED

 ALPHA-CPU OF CPU-PIN-REDEF         05   AN   GROUP     SIZE=2     OFFSET=0      COBEX85[1]     61
      COBEX85[1]      64 M

 ALPHA-PIN OF CPU-PIN-REDEF         05   AN   GROUP     SIZE=2     OFFSET=2      COBEX85[1]     66
      COBEX85[1]      69 M

 CAPTURE-CPU-PIN OF AA              PARAGRAPH                                     COBEX85[1]     200
      COBEX85[1]      150.1 M

 CPU-HIGH-BYTE OF ALPHA-CPU OF CPU-PIN-REDEF                                      COBEX85[1]     62
                                    10   AN   DISPLY    SIZE=1     OFFSET=0
      COBEX85[1]      202 W

 CPU-LOW-BYTE  OF  ALPHA-CPU OF CPU-PIN-REDEF                                     COBEX85[1]     63
                                    10   AN   DISPLY    SIZE=1     OFFSET=1
      COBEX85[1]      201 W

 CPU-PART  OF CPU-PIN OF PROCESS-ID OF MESSAGE-SOURCE-REC                         COBEX85[1]     55
                                    15   AN   DISPLY    SIZE=1    OFFSET=14
      COBEX85[1]      201

 DISPLAY-STARTUP-FAILURE OF AA      PARAGRAPH                                     COBEX85[1]     158
      COBEX85[1]      151 M

 FUP OF SAVE-MESSAGE-STUFF          05    AN   DISPLY    SIZE=21    OFFSET=0      COBEX85[1]     78
      COBEX85[1]      136 P       165

 FUP-FAILED  OF SAVE-MESSAGE-STUFF 05    AN   DISPLY    SIZE=19    OFFSET=44     COBEX85[1]     94
      COBEX85[1]      160         163          167         171   181

 FUP-NAME  OF SAVE-MESSAGE-STUFF    05    AN   DISPLY    SIZE=5     OFFSET=21     COBEX85[1]     80
      COBEX85[1]      137 P

 INFO-COMMAND  OF SAVE-MESSAGE-STUFF                                              COBEX85[1]
                                    05    AN   DISPLY    SIZE=7     OFFSET=70
      COBEX85[1]      130 P

 MEMORY-AS-USUAL OF SAVE-MESSAGE-STUFF                                            COBEX85[1]
                                    05    NM   COMP      SIZE=2     OFFSET=32
      COBEX85[1]      141

 MESSAGE-IN-FILE                    FILE (FD) IS $RECEIVE                         COBEX85[1]
                                    ORG=SEQ   ACC=SEQ    RECORD=1032 F
      COBEX85[1]      36 M        109 M       122 M        150 M

 MESSAGE-SOURCE-REC                 01    AN   GROUP     SIZE=32    OFFSET=0      COBEX85[1]
      COBEX85[1]      30 M
```

Figure 7-3. CROSSREF Listing--Identifier List (Page 1 of 3)



```
PAGE 3
NAME AND NAME QUALIFIER        LANGUAGE-DEPENDENT ATTRIBUTES                 DEFINITION POINT


NEWPROCESS-ERR-LEFT OF SAVE-MESSAGE-STUFF                                    COBEX85[1]   92
                                05  NM   DISPLY   SIZE=4    OFFSET=36
     COBEX85[1]      179 W       185

NEWPROCESS-ERR-RIGHT  OF SAVE-MESSAGE-STUFF                                  COBEX85[1]   93
                                05  NM   DISPLY   SIZE=4    OFFSET=40
     COBEX85[1]      180 W       187

NULL-CPLIST OF SAVE-MESSAGE-STUFF 05  NM   COMP     SIZE=4    OFFSET=80     COBEX85[1]   102
     COBEX85[1]      131 P

NUMERIC-CPU OF CPU-PIN-REDEF      05  NM   COMP     SIZE=2    OFFSET=0      COBEX85[1]   64
     COBEX85[1]      150.3

NUMERIC-PIN OF CPU-PIN-REDEF      05  NM   COMP     SIZE=2    OFFSET=2      COBEX85[1]   69
     COBEX85[1]      150.3

PIN-HIGH-BYTE OF ALPHA-PIN OF CPU-PIN-REDEF                                  COBEX85[1]   67
                                10  AN   DISPLY   SIZE=1    OFFSET=2
     COBEX85[1]      204 W

PIN-LOW-BYTE  OF ALPHA-PIN OF CPU-PIN-REDEF                                  COBEX85[1]   68
                                10  AN   DISPLY   SIZE=1    OFFSET=3
     COBEX85[1]      203 W

PIN-PART  OF CPU-PIN OF PROCESS-ID OF MESSAGE-SOURCE-REC                     COBEX85[1]   56
                                15  AN   DISPLY   SIZE=1    OFFSET=15
     COBEX85[1]      203

PRIORITY-EQ-MINE  OF SAVE-MESSAGE-STUFF                                      COBEX85[1]   84
                                05  NM   COMP     SIZE=2    OFFSET=28
     COBEX85[1]      139

PROCESSOR-EQ-MINE OF SAVE-MESSAGE-STUFF                                      COBEX85[1]   86
                                05  NM   COMP     SIZE=2    OFFSET=30
     COBEX85[1]      140

RECEIVE-FILE-STATUS OF FILE-DATA  05  AN   GROUP    SIZE=2    OFFSET=0      COBEX85[1]   73
     COBEX85[1]      24 M        115

SEND-ALL-MSGS OF SAVE-MESSAGE-STUFF                                          COBEX85[1]   82
                                05  NM   COMP     SIZE=2    OFFSET=26
     COBEX85[1]      138

STARTUP-RESULT  OF SAVE-MESSAGE-STUFF                                        COBEX85[1]   100
                                05  NM   COMP     SIZE=2    OFFSET=78
     COBEX85[1]      132 W

STAT-1  OF RECEIVE-FILE-STATUS OF FILE-DATA                                  COBEX85[1]   74
                                10  NM   DISPLY   SIZE=1     OFFSET=0
```

Figure 7-3. CROSSREF Listing--Identifier List (Page 2 of 3)

```
PAGE 4
NAME AND NAME QUALIFIER        LANGUAGE-DEPENDENT ATTRIBUTES                DEFINITION POIN


      COBEX85[1]      111

 STRING-PORTION  OF SAVE-MESSAGE-STUFF                                       COBEX85[1]   9
                                  05  AN   DISPLY   SIZE=7    OFFSET=63
      COBEX85[1]      129 P

 SU-ERROR  OF SAVE-MESSAGE-STUFF   05  NM   COMP     SIZE=2    OFFSET=34      COBEX85[1]   9
      COBEX85[1]      123 W       143 W       146      159      173       177       183

 SYS-MSG-ABEND OF SYS-MSG-CODE OF MESSAGE-IN OF MESSAGE-IN-FILE              COBEX85[1]   4
                                  88  NM   COMP     SIZE=2    OFFSET=0
      COBEX85[1]      147

 SYS-MSG-CODE  OF MESSAGE-IN OF MESSAGE-IN-FILE                              COBEX85[1]   4
                                  05  NM   COMP     SIZE=2    OFFSET=0
      COBEX85[1]      124 W       150.5

 SYS-MSG-STOP  OF SYS-MSG-CODE OF MESSAGE-IN OF MESSAGE-IN-FILE              COBEX85[1]   4
                                  88  NM   COMP     SIZE=2    OFFSET=0
      COBEX85[1]      147
```

Figure 7-3. CROSSREF Listing--Identifier List (Page 3 of 3)

COMPILER ATTRIBUTES

CROSSREF collects identifier attribute information from the COBOL85 compiler and prints it in the identifier header. The attributes are explained below.

Alphabet-Name

If the identifier is an alphabet-name, it is described in the header as ALPHABET IS *definition*.

definition can be STANDARD-1, SPECIAL, or *system-name* depending on what you entered in your program. If you specified one of the reserved words STANDARD-1, STANDARD-2, or NATIVE, STANDARD-1 appears. If you specified a literal phrase, SPECIAL appears. If you specified EBCDIC for the *system-name*, EBCDIC appears.

Class-Name

If the identifier is a class-name, it is described in the header as CLASS.

Condition-Name

If the identifier is a condition-name, it is described in one of two ways: either as a Level 88 Item or as CONDITIONAL IS SWITCH-*ss*.

If the identifier is a Level 88 Item, it is described in the header as

88 *category usage* SIZE=*size* [V] OFFSET=*offset* [*ss* SUB[S]]

category, *usage*, *size*, V, *offset*, and *ss* are all explained under "Data-Name" in this section.

Level 88 condition-names are always associated with a data-name called the conditional-variable. The first or only identifier qualifier indicates which conditional-variable the Level 88 Item is associated with.

If the identifier represents a condition-name that tests an external switch, it described in the header as

```
CONDITION IS SWITCH-ss {ON }
                       {OFF}
```

ss is a decimal integer from 1 to 15 that identifies which switch the program is testing. ON or OFF indicates which state is being tested.

If the condition name is qualified, the qualifier identifies the mnemonic-name with which the condition-name is associated.

Data-Name

If the identifier is a data item, it is described in the header as

```
ln category usage SIZE=size [V] OFFSET=offset [ss SUB[S]]
                                              [SPCL-REG]
```

ln is the level number of the data-item.

category is one of the following:

AL	Alphabetic
AN	Alphanumeric
ANE	Alphanumeric edited
NM	Numeric
NME	Numeric edited

If the *usage* is INDEX, the *category* is blank.

usage is one of the following:

GROUP	Group item
DISPLY	DISPLAY item
COMP	COMPUTATIONAL item
INDEX	INDEX item
NATIVE	NATIVE-2, NATIVE-4, or NATIVE-8 item

size shows the size in bytes of the identifier's value in decimal notation (up to nine places without leading zeros).

V appears in the attribute list if the data-item contains a subordinate item that is a table with a variable number of occurrences.

offset shows the byte offset of the value of the data-item from the relevant base address; for example, from the start of the containing record. The offset is shown in decimal notation up to nine places without leading zeros.

ss SUB[S] appears if the references to the data-item require subscripting. In that case, *ss* is a number from 1 to 7 showing the number of subscripts. If the value of *ss* is 1, SUB appears instead of SUBS.

SPCL-REG appears if the data-name is a Special Register.

File-Name

If the identifier is a file name, it is described in the header as

```
   FILE { (FD) } IS Tandem-name                      [kk KEY[S] ]
        { (SD) }                                     [SET s:pp  ]

         ORG=org  ACC=acc  RECORD=r {F}  BLOCK=b [R]
                                    {V}
```

FILE (FD) appears when the file-name identifies a data file. FILE (SD) appears when the file-name identifies a sort-merge file.

Tandem-name is the Tandem file-name that you specified in the ASSIGN clause of the File-Control entry.

kk KEY[S] appears if the file has record keys. *kk* shows the number of keys in decimal notation. If the value of *kk* is 1, KEY appears instead of KEYS.

SET *s:pp* appears if the file is part of a multiple file tape set. *s* is a digit that identifies the set and *pp* is a decimal integer from 1 to 31 that identifies the file's position in that set.

ORG marks the file's organization. It can be:

SEQ	Sequential
REL	Relative
INX	Indexed

ACC marks the file's access mode. It can be:

SEQ	Sequential
RAN	Random
DYN	Dynamic

The RECORD entry shows record size in bytes, expressed in decimal notation. F indicates the records are of fixed length; V indicates the records are of variable length, each having up to the displayed maximum size.

If the block size is not equal to the record size, the BLOCK field shows the block size in decimal notation. If R appears, *b* indicates the size of the block in terms of the number of records that the block contains; if R does not appear, b indicates the size of the block in terms of the number of bytes that the block contains.

External Switch

If the identifier is an external switch referenced by a mnemonic-name, it is described in the header as EXTERNAL SWITCH.

Index-Name

If the identifier is the name of an index item, it is described in the header as INDEX SIZE=4.

The first or only identifier qualifier shows the name of the table item that the index-name belongs to. An index item is always four bytes long.

Literals

CROSSREF prints literals before any other identifiers. They are shown exactly as they appear in the source file; if they have quotation marks in the source file, they have them in the cross-reference listing.

Mnemonic-Name

If the identifier is a mnemonic-name, it is described in the header as

```
                { CHANNEL-cc  }
   MNEMONIC IS  { Tandem-name }
                { SWITCH-ss   }
```

When the mnemonic-name refers to a channel, CHANNEL-*cc* appears in the header. *cc* is a decimal integer from 1 to 12 that identifies the channel.

When the mnemonic-name refers to a system-name, the appropriate *Tandem-name* appears in the header. If you specify CONSOLE as the system-name, $0 appears; if you specify MYTERM, #TERM appears.

When the mnemonic-name refers to an external switch, SWITCH-*ss* appears in the header. *ss* is a decimal integer from 1 to 15 that identifies the switch. The external switch also appears as a separate entry in the cross-reference listing. See "External Switch" in this section for details.

Paragraph-Name

If the identifier is the name of a paragraph, it is described in the header as PARAGRAPH. If the paragraph-name is qualified, the qualifier also indicates what section the paragraph belongs to.

If code is generated for the program, %*offset* shows the code offset for this paragraph relative to the base of the containing program. %*offset* is an octal number of six digits.

Program-Name

If the identifier is the name of a program, it is described in the header as PROGRAM.

Section-Name

If the identifier is the name of a section, it is described in the header as SECTION.

If code is generated for the program, %*offset* shows the code offset for this section relative to the base of the containing program. %*offset* is an octal number of six digits.

Symbolic-Character

If the identifier is a symbolic-character, it is described in the header as SYMBOLIC IS *vvv*.

vvv is a decimal integer from 1 to 256. 1 corresponds to the first character in the Tandem character set and 256 to the last character in the set.

SECTION 8

EXTENDED BASIC

This section describes the EXTENDED BASIC identifier classes and provides a sample BASIC program and its cross-reference listing.

EXTENDED BASIC IDENTIFIERS

The CROSSREF utility indexes EXTENDED BASIC programs according to the identifier classes listed in Table 8-1.

Table 8-1 also shows the default settings for each identifier class and what EXTENDED BASIC data types correspond to each of these classes.

Table 8-1. EXTENDED BASIC Identifier Classes

CROSSREF Class	Default Setting	BASIC Type
FUNCTIONS	ON	User-defined functions
KEYWORDS	OFF	Keywords (reserved words)
LINENOS	ON	Line numbers
LITERALS	OFF	Literals
SYSVARS	ON	System variables
VARIABLES	ON	Variables

Notice that the default setting for KEYWORDS and LITERALS is OFF. If you want keywords and literals to appear in the cross-reference listing, you must set them to ON using the SET command. See Section 4 for details.

SAMPLE LISTING

The following example starts CROSSREF, scans the file named BASEX, and generates a listing to $s.#lp:

```
13> CROSSREF
CROSSREF-CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1985, 1986
&SET LANGUAGE basic
&SCAN basex
&GENERATE /OUT $s.#lp/
&EXIT
14>
```

On the following pages, you can see the program and its cross-reference listing. The listing includes line numbers, variables, functions, and system variables. It does not include keywords and literals, however, because only the default settings were used.

The line numbers shown in the CROSSREF listing represent the EXTENDED BASIC line number plus the continuation number of the BASIC statement. For example, line 230 in the sample program (see Figure 8-1) contains statements 230, 230.01, and 230.02. Lines 230.01 and 230.02 represent the THEN and ELSE statements of line 230:

```
230  IF MID$(tdate$,4,3) = currentmo$                &
        THEN fnthismonth = 1  ELSE fnthismonth = 0
```

There are several ways to declare the type and size of an EXTENDED BASIC variable; it may be defined by a DECLARE, DIM, or MAP statement. The CROSSREF priority is MAP, DIM, DECLARE. Consequently, CROSSREF lists the MAP statement as the defining statement followed by subsequent references to DIM and DECLARE statements.

Figure 8-1 shows the EXTENDED BASIC program that CROSSREF scanned to produce the cross-reference listing.

```
100  !  Print monthly account activity summary.

200  !  Set up function to find current month's stuff.
210  currentmo$ = MID$(DAT$,4,3)
220  DEF fnthismonth(tdate$)
230      IF MID$(tdate$,4,3)=currentmo$                    &
          THEN fnthismonth=1  ELSE fnthismonth=0
240  FNEND

300  !  Set up map and open account file.
310  MAP (accounts) id$=6,    lname$=20, firname$=20,      &
                    FILL$=60, balance,   lasttrandate$=8
320  OPEN "accounts" as #1, ORGANIZATION INDEXED,          &
                            MAP accounts,                 &
                            ACCESS READ, ALLOW READ

400  !  Print header for report.
410  PRINT USING 710 \ PRINT USING 720 \ PRINT USING 730

500  !  Read records and print lines of report.
510  ON ERROR GOTO 900
520  WHILE ERR = 0
530     GET #1
540     IF (fnthismonth(lasttrandate$))                    &
           THEN PRINT USING 740, id$,                      &
                         TRM$(lname$) + ", " + firname$,   &
                         balance, lasttrandate$           \&
           total = total + balance
550  NEXT

600  !  Print total.
610  PRINT  \  PRINT USING 750
620  PRINT USING 760, total  \ PRINT

700  !  All the print formats are kept here.
710  : ACCOUNT                       CURRENT     TRANSACTION
720  : NUMBER     NAME               BALANCE        DATE
730  : ------------------------------------------------
740  : 'RRRRR  |  'LLLLLLLLL  |  $$##,###.##-  |  'LLLLLL
750  :                            ____________
760  :              TOTAL IS:  $$#,###,###.##-


900  !  Error processing code.
910  IF ERR=1 AND ERL=530 THEN RESUME 600                &
                          ELSE ON ERROR GOTO 0
```

Figure 8-1. EXTENDED BASIC Sample Program

Figure 8-2 shows the first page of the cross-reference listing. This is the cross-reference file list. It lists the name of each source file scanned. In this example, only one file, BASEX, was scanned.

```
PAGE 1


CROSSREF - CROSS-REFERENCE PROGRAM - T9622C00 - (15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1984, 1985, 1986


FILE NO.     FILE NAME
  [1]        $EM2.UCREF.BASEX
```

Figure 8-2. CROSSREF Listing--File List

The identifier list makes up the rest of the cross-reference listing. See Figure 8-3 on the following page. The identifier list describes each identifier in alphabetic order, showing:

- How it is defined (its attributes)
- Where it is defined (file name and number and line number)
- Where and how it is used in the program

Look at the entry for FNTHISMONTH in the listing. The identifier header indicates that FNTHISMONTH is a user-defined function returning a floating-point, REAL(64) value. It is defined in the file BASEX at line 220.

The reference line indicates that FNTHISMONTH is referenced in the file BASEX at lines 220, 230.01, 230.02, and 540. It is defined (indicated by code D) at line 220, and it is write referenced (indicated by code W) at lines 220, 230.01, and 230.02. The function is invoked (indicated by code I) at line 540.

The value of FNTHISMONTH is initialized at line 220; that is, value 0 is written to it. Thus, the function is both declared and write referenced at line 220.

```
PAGE 2
NAME AND NAME QUALIFIER          LANGUAGE-DEPENDENT ATTRIBUTES                    DEFINITION POINT


94 TOTAL SYMBOLS COLLECTED WITH 106 TOTAL REFERENCES COLLECTED


 600                             NUMBERED BASIC LINE                                BASEX[1]      600
      BASEX[1]     910.01

 710                             NUMBERED BASIC LINE WITH PRINT IMAGE               BASEX[1]      710
      BASEX[1]     410

 720                             NUMBERED BASIC LINE WITH PRINT IMAGE               BASEX[1]      720
      BASEX[1]     410.01

 730                             NUMBERED BASIC LINE WITH PRINT IMAGE               BASEX[1]      730
      BASEX[1]     410.02

 740                             NUMBERED BASIC LINE WITH PRINT IMAGE               BASEX[1]      740
      BASEX[1]     540.01

 750                             NUMBERED BASIC LINE WITH PRINT IMAGE               BASEX[1]      750
      BASEX[1]     610.01

 760                             NUMBERED BASIC LINE WITH PRINT IMAGE               BASEX[1]      760
      BASEX[1]     620

 900                             NUMBERED BASIC LINE                                BASEX[1]      900
      BASEX[1]     510

 BALANCE                         FLOATING-POINT, REAL(64), VARIABLE WITHIN MAP      BASEX[1]      310
      BASEX[1]     310 D       540.01       540.02

 CURRENTMO$                      STRING VARIABLE
      BASEX[1]     210 W       230

 ERL                             SYSTEM VARIABLE WITH INTEGER VALUE
      BASEX[1]     910

 ERR                             SYSTEM VARIABLE WITH INTEGER VALUE
      BASEX[1]     520         910

 FIRMNAME$                       STRING VARIABLE WITHIN MAP, SIZE=20 BYTES          BASEX[1]      310
      BASEX[1]     310 D       540.01

 FNTHISMONTH                     USER-DEFINED FUNCTION RETURNING FLOATING-POINT, REAL(64),   BASEX[1]      220
                                 VALUE
      BASEX[1]     220 D       220 W        230.01 W    230.02 W    540 I

 ID$                             STRING VARIABLE WITHIN MAP, SIZE=6 BYTES           BASEX[1]      310
      BASEX[1]     310 D       540.01

 LASTTRANDATES                   STRING VARIABLE WITHIN MAP, SIZE=8 BYTES           BASEX[1]      310
```

Figure 8-3. CROSSREF Listing--Identifier List (Page 1 of 2)

```
PAGE 3
NAME AND NAME QUALIFIER        LANGUAGE-DEPENDENT ATTRIBUTES                DEFINITION POINT


      BASEX[1]        310 D    540 P       540.01

 LNAME$                         STRING VARIABLE WITHIN MAP, SIZE=20 BYTES    BASEX[1]      310
      BASEX[1]        310 D    540.01

 TDATE$                         STRING VARIABLE
      BASEX[1]        220 D    230

 TOTAL                          FLOATING-POINT, REAL(64), VARIABLE
      BASEX[1]        540.02   540.02 W    620
```

Figure 8-3. CROSSREF Listing--Identifier List (Page 2 of 2)

SECTION 9

FORTRAN

This section describes the FORTRAN identifier classes and provides a sample FORTRAN program and its cross-reference listing.

FORTRAN IDENTIFIERS

The CROSSREF utility indexes FORTRAN programs according to the identifier classes listed in Table 9-1.

Table 9-1 also shows the default settings for each identifier class and what FORTRAN data types correspond to each of these classes.

Table 9-1. FORTRAN Identifier Classes

CROSSREF Class	Default Setting	FORTRAN Type
BLOCKDATAS	ON	BLOCK DATA subprograms
BLOCKS	ON	COMMON blocks
CONSTANTS	ON	PARAMETERs (named constants)
FMTLABELS	ON	Labels of FORMAT statements
FUNCTIONS	ON	FUNCTION subprograms
INLINES	ON	Inline functions
LITERALS	OFF	Unnamed constants
PROCEDURE PARAMS	ON	Dummy procedures
PROCEDURES	ON	SUBROUTINE subprograms
PROGLABELS	ON	Labels of executable statements
SUBPROCS	ON	Statement functions
VARIABLES	ON	Variables, arrays, records

Notice that the default setting for LITERALS is OFF. If you want unnamed constants to appear in the cross-reference listing, you must set LITERALS to ON using the SET command. See Section 4 for details.

SAMPLE LISTING

The following example starts CROSSREF, scans the file named FORTEX, and generates a listing to $s.#lp:

```
15> CROSSREF
CROSSREF-CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1985, 1986
&SET LANGUAGE fortran
&SCAN fortex
&GENERATE /OUT $s.#lp/
&EXIT
16>
```

On the following pages, you can see the program and its cross-reference listing. The listing includes all identifier classes except literals.

Figure 9-1 shows the FORTRAN program that CROSSREF scanned to produce the cross-reference listing.

```
 1           PROGRAM SORT
 2
 3     C     Program reads and sorts a file of up to 50 numbers.
 4
 5           INTEGER   count, maxcount
 6           REAL      numbers
 7           PARAMETER (maxcount=50)
 8           COMMON    count, numbers(maxcount)
 9
10           OPEN (UNIT=2, FILE = 'datafile')
11           DO 100  count = 1, maxcount
12              READ (UNIT=2, FMT=901, END=200)  numbers(count)
13     100      CONTINUE
14     C     Check for too many values.
15           READ (UNIT=2, FMT=901, END=200) dummy
16           WRITE(4,FMT=902)  maxcount
17           STOP
18     200   count = count - 1
19
20           CALL PRTNUMS ('Before sorting:')
21           CALL SORTNUMS (count, numbers)
22           CALL PRTNUMS ('After sorting:')
23
24           STOP
25     901   FORMAT (F6.2)
26     902   FORMAT (1X, 'Data file has too many values.', /,
27          +        1X, 'Maximum number of values for sort: ', I6)
28           END
29
30           SUBROUTINE SORTNUMS (icount, values)
31
32     C     Sorts array (values) with icount elements in ascending order.
33
34           DIMENSION  values(icount)
35
36           DO 600  i = icount-1, 1, -1
37              DO 500 j = 1, i
38                   IF ( values(j) .LT. values(j+1) )  GOTO 500
39                   temp        = values(j)
40                   values(j)   = values(j+1)
41                   values(j+1) = temp
42      500     CONTINUE
43      600  CONTINUE
44           RETURN
45           END
46
47           SUBROUTINE PRTNUMS (message)
48
49     C     Prints message passed as parameter, numbers from common block.
50
51           CHARACTER  message*(*)
52           INTEGER   count, maxcount
53           REAL      numbers
54           PARAMETER (maxcount=50)
55           COMMON    count, numbers(maxcount)
56
57           WRITE (UNIT=4,FMT=100)  message
58           WRITE (UNIT=4,FMT=200)  (numbers(i), i = 1, count)
59           RETURN
60     100   FORMAT(/,1X,A,/)
61     200   FORMAT(1X,F6.2)
62           END
```

Figure 9-1. FORTRAN Sample Program

Figure 9-2 shows the first page of the cross-reference listing. This is the cross-reference file list. It lists the name of each source file scanned. In this example, only one file, FORTEX, was scanned.

```
PAGE 1


CROSSREF - CROSS-REFERENCE PROGRAM - T9622C00 - (15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1984, 1985, 1986


FILE NO.     FILE NAME
  [1]        $EM2.UCREF.FORTEX
```

Figure 9-2. CROSSREF Listing--File List

The identifier list makes up the rest of the cross-reference listing. See Figure 9-3 on the following page. The identifier list describes each identifier in alphabetic order, showing:

- How it is defined (its attributes)
- Where it is defined (file name and number and line number)
- Where and how it is used in the program

Look at the entry for the identifier named COUNT. The identifier header indicates that it is an INTEGER * 2 VARIABLE (a 1-word, 2-byte integer) stored in blank common.

The identifier is first defined in the file FORTEX at line 5 and again at line 8 (indicated by code D). Then it is write referenced at lines 11 and 18 (indicated by code W); it is read referenced at lines 12 and 18 (indicated by code blank); and it is parameter referenced at line 21 (indicated by code P).

```
PAGE 2
NAME AND NAME QUALIFIER          LANGUAGE-DEPENDENT ATTRIBUTES                  DEFINITION POINT

 34 TOTAL SYMBOLS COLLECTED WITH 78 TOTAL REFERENCES COLLECTED

 100                              PROGRAM LABEL                                  FORTEX[1]       13
       FORTEX[1]        11 M       13 D

 100                              FORMAT LABEL                                   FORTEX[1]       60
       FORTEX[1]        57 M       60 D

 200                              PROGRAM LABEL                                  FORTEX[1]       18
       FORTEX[1]        12 M       15 M         18 D

 200                              FORMAT LABEL                                   FORTEX[1]       61
       FORTEX[1]        58 M       61 D

 500                              PROGRAM LABEL                                  FORTEX[1]       42
       FORTEX[1]        37 M       38 M         42 D

 600                              PROGRAM LABEL                                  FORTEX[1]       43
       FORTEX[1]        36 M       43 D

 901                              FORMAT LABEL                                   FORTEX[1]       25
       FORTEX[1]        12 M       15 M        25 D

 902                              FORMAT LABEL                                   FORTEX[1]       26
       FORTEX[1]        16 M       26 D

 COUNT                            INTEGER*2 VARIABLE, IN /BLANK^/                FORTEX[1]       5
       FORTEX[1]        5 D        8 D         11 W        12       18 W   18   21 P

 COUNT                            INTEGER*2 VARIABLE, IN /BLANK^/                FORTEX[1]       52
       FORTEX[1]        52 D       55 D        58

 DUMMY                            REAL VARIABLE                                  FORTEX[1]       15
       FORTEX[1]        15 W

 I                                INTEGER*2 VARIABLE                             FORTEX[1]       36
       FORTEX[1]        36 W       37

 I                                INTEGER*2 VARIABLE                             FORTEX[1]       58
       FORTEX[1]        58         58 M

 ICOUNT                           INTEGER*2 DUMMY VARIABLE                       FORTEX[1]       30
       FORTEX[1]        30 D       34          36

 J                                INTEGER*2 VARIABLE                             FORTEX[1]       37
       FORTEX[1]        37 W       38          39          40         41

 MAXCOUNT                         INTEGER*2 PARAMETER:     50                    FORTEX[1]        5
       FORTEX[1]         5 D       7 D         8           11         16
```

Figure 9-3. CROSSREF Listing--Identifier List (Page 1 of 2)

```
PAGE 3
NAME AND NAME QUALIFIER        LANGUAGE-DEPENDENT ATTRIBUTES                 DEFINITION POINT


 MAXCOUNT                    INTEGER*2 PARAMETER:     50                     FORTEX[1]      52
      FORTEX[1]     52 D       54 D        55

 MESSAGE                      CHARACTER*(*) DUMMY VARIABLE                  FORTEX[1]      47
      FORTEX[1]     47 D       51 D        57

 NUMBERS                      ARRAY(1:50) OF REAL, IN /BLANK^/              FORTEX[1]      6
      FORTEX[1]     6 D        8 D         12 W        21 P

 NUMBERS                      ARRAY(1:50) OF REAL, IN /BLANK^/              FORTEX[1]      53
      FORTEX[1]     53 D       55 D        58

 PRTNUMS                      SUBROUTINE                                    FORTEX[1]      47
      FORTEX[1]     20 I       22 I        47 D

 SORTNUMS                     SUBROUTINE                                    FORTEX[1]      30
      FORTEX[1]     21 I       30 D

 TEMP                         REAL VARIABLE                                 FORTEX[1]      39
      FORTEX[1]     39 W       41

 VALUES                       DUMMY ARRAY(1:?) OF REAL                      FORTEX[1]      30
      FORTEX[1]     30 D       34 D        38      39      40 W     40     41 W
```

Figure 9-3. CROSSREF Listing--Identifier List (Page 2 of 2)

SECTION 10

PASCAL

This section describes the Pascal identifier classes and provides a sample Pascal program and its cross-reference listing.

PASCAL IDENTIFIERS

The CROSSREF utility indexes Pascal programs according to the identifier classes listed in Table 10-1.

Table 10-1 also shows the default settings for each class and what Pascal data types correspond to each of these classes.

Table 10-1. Pascal Identifier Classes

CROSSREF Class	Default Setting	Pascal Type
BLOCKS	ON	Public variables
CONSTANTS	ON	Named constants
PROCEDURES	ON	Procedures and functions
PROGLABELS	ON	GOTO labels
TYPES	ON	All data types
VARIABLES	ON	Non-public variables

SAMPLE LISTING

The following example starts CROSSREF, scans the file named PASCALEX, and generates a listing to $s.#lp:

```
14> CROSSREF
CROSSREF-CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1985, 1986
&SET LANGUAGE pascal
&SCAN pascalex
&GENERATE /OUT $s.#lp/
&EXIT
15>
```

On the following pages, you can see the program and its cross-reference listing.

Figure 10-1 shows the Pascal program that CROSSREF scanned to produce the cross-reference listing.

```
 1     { This program reads data base input from the terminal and enters    }
 2     { it into a memory resident data base.                               }
 3     PROGRAM CrossrefExample(INPUT,OUTPUT);
 4
 5     IMPORT BEGIN
 6      TYPE
 7       Date       =INTEGER;
 8       PlaceName  =STRING[20];
 9       PersonName =STRING[20];
10       PersonKind =(National,Alien);
11       Level      = 1..4;
12     END;
13
24     TYPE
25       Person = RECORD
26                  Name:        PersonName;
27                  DateOfBirth: Date;
28                  JobLevel:    Level;
29                  NextRecord: ^Person;
30                  CASE Origin: PersonKind OF
31                    National: (BirthPlace:      PlaceName);
32                    Alien:    (CountryOfOrigin: PlaceName;
33                               DateOfEntry:     Date;
34                               PortOfEntry:     PlaceName);
35                END;
36
37      PtrToPerson = ^Person;
38
41
42     VAR
44      Name            :PersonName;
45      JobLevel        :Level;
46      BirthDate       :Date;
47      BirthPlace      :PlaceName;
48      CountryOfOrigin :PlaceName;
49      DateOfEntry     :Date;
50      PortOfEntry     :PlaceName;
51
52      PersonRecPointer :PtrToPerson;
53      TempRecPointer   :PtrToPerson;
55
56     PROCEDURE MakeThePerson(VAR ThePerson       :PtrToPerson;
57                                 Name            :PersonName;
58                                 JobLevel        :Level;
59                                 BirthDate       :Date;
60                                 BirthPlace      :PlaceName;
61                                 CountryOfOrigin :PlaceName;
62                                 DateOfEntry     :Date;
63                                 PortOfEntry     :PlaceName );
64      BEGIN
65
67          ThePerson^.Name        := Name;
68          ThePerson^.DateOfBirth := BirthDate;
69          ThePerson^.JobLevel    := JobLevel;
70          CASE ThePerson^.Origin OF
71            National: ThePerson^.BirthPlace       := BirthPlace;
```

Figure 10-1. Pascal Sample Program (Page 1 of 2)

```
 72            Alien:    BEGIN
 73                       ThePerson^.CountryOfOrigin := CountryOfOrigin;
 74                       ThePerson^.DateOfEntry     := DateOfEntry;
 75                       ThePerson^.PortOfEntry     := PortOfEntry;
 76                      END;
 77          END; {CASE}
 79
 80      END; {MakePerson}
 81
 82     BEGIN
 83
 84      NEW(PersonRecPointer);
 85
 86      WHILE NOT EOF(INPUT) DO BEGIN
 87
 88        WRITELN( 'Name:' );           READLN( Name);
 89        WRITELN( 'JobLevel: ');       READLN( JobLevel);
 90        WRITELN( 'BirthDate:');       READLN( BirthDate);
 91        WRITELN( 'BirthPlace:');      READLN( BirthPlace);
 92        WRITELN( 'CountryOfOrigin:'); READLN( CountryOfOrigin);
 93        WRITELN( 'DateOfEntry:');     READLN( DateOfEntry);
 94        WRITELN( 'PortOfEntry:');     READLN( PortOfEntry);
 95
 97
 98        MakeThePerson(PersonRecPointer, Name, JobLevel, BirthDate, BirthPlace,
 99                      CountryOfOrigin, DateOfEntry, PortOfEntry );
100
101        TempRecPointer := PersonRecPointer;
102        NEW(PersonRecPointer);
103        TempRecPointer^.NextRecord := PersonRecPointer;
104
105      END; {WHILE}
106
111     END.
```

Figure 10-1. Pascal Sample Program (Page 2 of 2)

Figure 10-2 shows the first page of the cross-reference listing. This is the cross-reference file list. It lists the name of each source file scanned. In this example only one file, PASCALEX, was scanned.

```
PAGE 1


CROSSREF - CROSS-REFERENCE PROGRAM - T9622C00 - (15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1984, 1985, 1986


FILE NO.     FILE NAME
  [1]        $EM2.UCREF.PASCALEX
```

Figure 10-2. CROSSREF Listing--File List

The identifier list makes up the rest of the cross-reference listing. See Figure 10-3 on the following page. The identifier list describes each identifier in alphabetic order, showing:

- How it is defined (its attributes)
- Where it is defined (file name and number and line number)
- Where and how it is used in the program

Look at the entry for the identifier named PERSONKIND. The identifier header indicates that is a public variable of type enumeration. It is defined in the file PASCALEX at line 10 (indicated by code D) and referenced at line 30 (indicated by code M).

Now look at the second entry for the identifier named PORTOFENTRY. The identifier header indicates that it is contained in the routine MAKETHEPERSON and is defined in the file PASCALEX at line 63. The reference line indicates that it is a parameter of the type PLACENAME and is read referenced in the file at line 75 (indicated by a blank code).

```
PAGE 2
NAME AND NAME QUALIFIER       LANGUAGE-DEPENDENT ATTRIBUTES                DEFINITION POINT


106 TOTAL SYMBOLS COLLECTED WITH 217 TOTAL REFERENCES COLLECTED


 ALIEN PUBLIC                      ENUMERATION CONST                       PASCALEX[1]    10
      PASCALEX[1]    10 D         32          72

 BIRTHDATE OF ROUTINE MAKETHEPERSON                                        PASCALEX[1]    59
                                   INT16 PARAM
      PASCALEX[1]    59 D         68

 BIRTHDATE OF ROUTINE CROSSREFEXAMPLE                                      PASCALEX[1]    46
                                   INT16 VAR
      PASCALEX[1]    46 D         90 W        98

 BIRTHPLACE  OF RECORD PERSON      PLACENAME FIELD                         PASCALEX[1]    31
      PASCALEX[1]    31 D         71 W

 BIRTHPLACE  OF ROUTINE MAKETHEPERSON                                      PASCALEX[1]    60
                                   PLACENAME PARAM
      PASCALEX[1]    60 D         71

 BIRTHPLACE  OF ROUTINE CROSSREFEXAMPLE                                    PASCALEX[1]    47
                                   PLACENAME VAR
      PASCALEX[1]    47 D         91 W        98

 COUNTRYOFORIGIN OF RECORD PERSON  PLACENAME FIELD                         PASCALEX[1]    32
      PASCALEX [1]   32 D         73 W

 COUNTRYOFORIGIN OF ROUTINE MAKETHEPERSON                                  PASCALEX[1]    61
                                   PLACENAME PARAM
      PASCALEX[1]    61 D         73

 COUNTRYOFORIGIN OF ROUTINE CROSSREFEXAMPLE                                PASCALEX[1]    48
                                   PLACENAME VAR
      PASCALEX[1]    48 D         92 W        99

 DATE  PUBLIC                      INT16 TYPE                              PASCALEX[1]    7
      PASCALEX[1]    7 D          27 M        33 M     46 M     49 M     59 M     62 M

 DATEOFBIRTH OF RECORD PERSON      INT16 FIELD                             PASCALEX[1]    27
      PASCALEX[1]    27 D         68 W

 DATEOFENTRY OF RECORD PERSON      INT16 FIELD                             PASCALEX[1]    33
      PASCALEX[1]    33 D         74 W

 DATEOFENTRY OF ROUTINE MAKETHEPERSON                                      PASCALEX[1]    62
                                   INT16 PARAM
      PASCALEX[1]    62 D         74

 DATEOFENTRY  OF ROUTINE CROSSREFEXAMPLE                                   PASCALEX[1]    49
```

Figure 10-3. CROSSREF Listing--Identifier List (Page 1 of 3)

```
PAGE 3
NAME AND NAME QUALIFIER          LANGUAGE-DEPENDENT ATTRIBUTES              DEFINITION POINT


                                  INT16 VAR
     PASCALEX[1]      49 D       93 W        99

 EOF PREDEFINED                    ROUTINE
     PASCALEX[1]      86 I

 INPUT PUBLIC                      TEXT VAR                                  PASCALEX[1]    3
     PASCALEX[1]      3 D        86 W        88 W     89 W    90 W    91 W     92 W    93 W        94 W

 INTEGER PREDEFINED                INT16 TYPE
     PASCALEX[1]      7 M

 JOBLEVEL  OF RECORD PERSON        LEVEL FIELD                               PASCALEX[1]    28
     PASCALEX[1]     28 D        69 W

 JOBLEVEL  OF ROUTINE MAKETHEPERSON                                          PASCALEX[1]    58
                                   LEVEL PARAM
     PASCALEX[1]     58 D        69

 JOBLEVEL  OF ROUTINE CROSSREFEXAMPLE                                        PASCALEX[1]    45
                                   LEVEL VAR
     PASCALEX[1]     45 D        89 W        98

 LEVEL PUBLIC                      SUBRANGE TYPE                             PASCALEX[1]    11
     PASCALEX[1]     11 D        28 M        45 M     58 M

 MAKETHEPERSON OF ROUTINE CROSSREFEXAMPLE                                    PASCALEX[1]    56
                                   ROUTINE
     PASCALEX[1]     56 D        98 I

 NAME  OF RECORD PERSON            PLACENAME FIELD                           PASCALEX[1]    26
     PASCALEX[1]     26 D        67 W

 NAME  OF ROUTINE MAKETHEPERSON    PLACENAME PARAM                           PASCALEX[1]    57
     PASCALEX[1]     57 D        67

 NAME  OF ROUTINE CROSSREFEXAMPLE  PLACENAME VAR                             PASCALEX[1]    44
     PASCALEX[1]     44 D        88 W        98

 NATIONAL  PUBLIC                  ENUMERATION CONST                         PASCALEX[1]    10
     PASCALEX[1]     10 D        31          71

 NEW PREDEFINED                    ROUTINE                                                  .
     PASCALEX[1]     84 I        102 I

 NEXTRECORD  OF RECORD PERSON      POINTER FIELD                             PASCALEX[1]    29
     PASCALEX[1]     29 D        103 W

 ORIGIN OF RECORD CROSSREFEXAMPLE PERSONKIND FIELD                           PASCALEX[1]    30
     PASCALEX[1]     30 D        70
```

Figure 10-3. CROSSREF Listing--Identifier List (Page 2 of 3)

```
PAGE 4
NAME AND NAME QUALIFIER          LANGUAGE-DEPENDENT ATTRIBUTES                DEFINITION POINT


 OUTPUT PUBLIC                    TEXT VAR                                    PASCALEX[1]    3
      PASCALEX[1]      3 D        88 W        89 W    90 W    91 W    92 W    93 W       94 W

 PERSON  OF ROUTINE CROSSREFEXAMPLE                                           PASCALEX[1]    25
                                  RECORD TYPE
      PASCALEX[1]     25 D        29 M       37 M

 PERSONKIND  PUBLIC               ENUMERATION TYPE                            PASCALEX[1]    10
      PASCALEX[1]     10 D        30 M

 PERSONNAME  PUBLIC               PLACENAME TYPE                              PASCALEX[1]    9
      PASCALEX[1]     9 D         26 M        44 M    57 M

 PERSONRECPOINTER  OF ROUTINE CROSSREFEXAMPLE                                 PASCALEX[1]    52
                                  PTRTOPERSON VAR
      PASCALEX[1]     52 D        84 W        98 W    101     102 W   103

 PLACENAME PUBLIC                 STRING TYPE                                 PASCALEX[1]    8
      PASCALEX[1]     8 D         31 M        32 M    34 M    47 M    48 M    50 M       60 M        61 M
                      63 M

 PORTOFENTRY OF RECORD PERSON     PLACENAME FIELD                             PASCALEX[1]    34
      PASCALEX[1]     34 D        75 W

 PORTOFENTRY OF ROUTINE MAKETHEPERSON                                         PASCALEX[1]    63
                                  PLACENAME PARAM
      PASCALEX[1]     63 D        75

 PORTOFENTRY OF ROUTINE CROSSREFEXAMPLE                                       PASCALEX[1]    50
                                  PLACENAME VAR
      PASCALEX[1]     50 D        94 W        99

 PTRTOPERSON OF ROUTINE CROSSREFEXAMPLE                                       PASCALEX[1]    37
                                  POINTER TYPE
      PASCALEX[1]     37 D        52 M        53 M    56 M

 READLN  PREDEFINED               ROUTINE
      PASCALEX[1]     88 I        89 I        90 I    91 I    92 I    93 I    94 I

 TEMPRECPOINTER OF ROUTINE CROSSREFEXAMPLE                                    PASCALEX[1]    53
                                  PTRTOPERSON VAR
      PASCALEX[1]     53 D        101 W       103

 THEPERSON OF ROUTINE MAKETHEPERSON                                           PASCALEX[1]    56
                                  PTRTOPERSON PARAM
      PASCALEX[1]     56 D        67          68      69      70      71      73         74          75

 WRITELN PREDEFINED               ROUTINE
      PASCALEX[1]     88 I        89 I        90 I    91 I    92 I    93 I    94 I
```

Figure 10-3. CROSSREF Listing--Identifier List (Page 3 of 3)

SECTION 11

SCREEN COBOL

This section describes the SCREEN COBOL identifier classes and provides a sample SCREEN COBOL program and its cross-reference listing.

SCREEN COBOL IDENTIFIERS

The CROSSREF utility indexes SCREEN COBOL programs according to the identifier classes listed in Table 11-1.

Table 11-1 also shows the default settings for each identifier class and what SCREEN COBOL data types correspond to each of these classes.

Table 11-1. SCREEN COBOL Identifier Classes

CROSSREF Class	Default Setting	SCREEN COBOL Type
CONDITIONS	ON	Condition names
LITERALS	OFF	Numeric and nonnumeric constants
MNEMONICS	ON	Mnemonic names, alphabet names
PROCEDURES	ON	PROGRAMS
PROGLABELS	ON	Labels, procedure names (paragraph names, section names)
SCREENS	ON	Screen names
VARIABLES	ON	Data names

Notice that the default setting for literals is OFF. If you want numeric and nonnumeric constants to appear in the cross-reference listing, you must set LITERALS to ON using the SET command. See Section 4 for details.

By default, CROSSREF does not report unreferenced identifiers for COBOL 74, COBOL85, or SCREEN COBOL. If you want unreferenced identifiers to appear, you must set the UNREF attribute specification to ON or ONLY. If you set UNREF to ON, CROSSREF collects all identifiers, referenced and unreferenced, that belong to all classes set to ON. If you set UNREF to ONLY, CROSSREF collects only the unreferenced identifiers that belong to all classes set to ON.

SAMPLE LISTING

The following example starts CROSSREF, scans the file named SCOBEX, and generates a listing to $s.#lp:

```
14> CROSSREF
CROSSREF-CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1985, 1986
&SET LANGUAGE scobolx
&SCAN scobex
&GENERATE /OUT $s.#lp/
&EXIT
15>
```

On the following pages, you can see the program and its cross-reference listing. The listing includes all identifier classes except literals.

Figure 11-1 shows the SCREEN COBOL program that CROSSREF scanned to produce the cross-reference listing.

```
 1      IDENTIFICATION DIVISION.
 2      Program-id.
 3                      ScdragnA.
 4      ENVIRONMENT DIVISION.
 5      CONFIGURATION SECTION.
 6      SOURCE-COMPUTER.
 7                      T16.
 8      OBJECT-COMPUTER.
 9                      T16,
10                      Terminal is T16-6530.
11     *                Character-set is US ASCII
12     *                Character-set is USASCII.
13      SPECIAL-NAMES.
14                      F1         is F1,
15                      F2         is F2,
16                      F3         is F3,
17                      F4         is F4,
18                      F5         is F5,
19                      F6         is F6,
20                      F7         is F7,
21                      F14        is F14
22                      SF16       is SF16,
23                      RETURN-KEY is RETURN-KEY.
24
25     *-----------------------------------------------------------------------
26     *
27     *   ---->>>>>> This version for 6530 terminal.
28     *   ---->>>>>> The terminal must have the appropriate language
29     *   ---->>>>>>     installed and the RETURN key configured as a function
30     *   ---->>>>>>     key.
31     *   The purpose of this test is to check out the national use character
32     *   features of SCOBOL.
33     *
34     *       1. A menu screen will appear and ask you to select the
35     *          character set you wish to test.
36     *
37     *-----------------------------------------------------------------------
38
39      DATA DIVISION.
40      WORKING-STORAGE SECTION.
41
42      01 Info1                            Pic x(65)          Value spaces.
43      01 Info2                            Pic x(65)          Value spaces.
44      01 Info3                            Pic x(65)          Value spaces.
45      01 Info4                            Pic x(65)          Value spaces.
46      01 Info5                            Pic x(65)          Value spaces.
47      01 Err-msg                          Pic x(67)          Value spaces.
48      01 END-OF-TEST                      Pic x(03)          Value "NO ".
49
50      LINKAGE SECTION.
51
52      SCREEN SECTION.
53
54      01 DRAGON-SCREEN SIZE 24, 80 .
55        05 FILLER AT 1, 18
56                  VALUE "Test of European National Use Characters" .
```

Figure 11-1. SCREEN COBOL Sample Program (Page 1 of 4)

```
 57        05 FILLER AT 3, 14
 58                  VALUE "Press the appropriate function key to test the ".
 59        05 FILLER AT 4, 14
 60                  VALUE "character set desired. ".
 61        05 FILLER AT 5, 14
 62                  VALUE "The screens will contain all instructions necessary".
 63        05 FILLER AT 6, 14
 64                  VALUE "to run the tests".
 65        05 L1 AT  8, 14
 66              PIC X(65)
 67              FROM INFO1 .
 68        05 L2 AT  9, 14
 69              PIC X(65)
 70              FROM INFO2 .
 71        05 L3 AT 10, 14
 72              PIC X(65)
 73              FROM INFO3 .
 74        05 L4 AT 11, 14
 75              PIC X(65)
 76              FROM INFO4 .
 77        05 L5 AT 12, 14
 78              PIC X(65)
 79              FROM INFO5 .
 80        05 FILLER AT 14, 25
 81                  VALUE "Character set selection keys".
 82        05 FILLER AT 16, 22
 83                  VALUE "F1 French (AZ)     F2 French (QW)".
 84        05 FILLER AT 17, 22
 85                  VALUE "F3 German          F4 Spanish".
 86        05 FILLER AT 18, 22
 87                  VALUE "F5 United Kingdom  F6 Swedish".
 88        05 FILLER AT 19, 22
 89                  VALUE "F7 Danish          F14 Recover ".
 90        05 FILLER AT 20, 22
 91                  VALUE "RETURN message    SF16 End Test ".
 92        05 ERR-WINDOW AT 24, 4
 93                      PIC X(67)
 94                      FROM ERR-MSG
 95                      ADVISORY .
 96
 97      PROCEDURE DIVISION.
 98      DECLARATIVES.
 99        RECOVER-SCRN SECTION.
100          USE FOR SCREEN RECOVERY.
101          DISPLAY DRAGON-SCREEN.
102          MOVE "SCREEN RECOVERY ACTIVATED" TO ERR-MSG.
103          DISPLAY TEMP ERR-WINDOW.
104        END DECLARATIVES.
105
106      MAIN-DRIVER.
107        Move space to info1.
108        MOVE "NO" TO END-OF-TEST.
109        PERFORM STEP1
110                UNTIL END-OF-TEST = "YES".
111      ALL-DONE.
112        EXIT PROGRAM.
```

Figure 11-1. SCREEN COBOL Sample Program (Page 2 of 4)

```
113
114
115      Step1.
116
117            Display base DRAGON-SCREEN.
118            Display L1.
119            Display " " in L2.
120            Display " " in L3.
121            Display " " in L4.
122            Accept
123                UNTIL F1 F2 F3 F4 F5 F6 F7 F14 RETURN-KEY
124                Escape on SF16.
125            If  termination-status = 10
126                move "YES" to end-of-test
127            else
128                move space to info1.
129                perform one of
130                        TEST1
131                        TEST2
132                        TEST3
133                        TEST4
134                        TEST5
135                        TEST6
136                        TEST7
137                        RECOVER-SCRN1
138                        DISPLAY-RETURN
139                   depending on termination-status.
140     /
141      TEST1.
142     *      FRENCH (AZ)
143            Call scdragn6
144                 on error
145                     perform TEST1-ERROR.
146
147      TEST2.
148     *      FRENCH (QW)
149            Call scdragn5
150                 on error
151                     perform TEST2-ERROR.
152
153      TEST3.
154     *      GERMAN
155            Call scdragn3
156                 on error
157                     perform TEST3-ERROR.
158
159      TEST4.
160     *      SPANISH
161            Call scdragn7
162                 on error
163                     perform TEST4-ERROR.
164
165      TEST5.
166     *      UNITED KINGDOM
167            Call scdragn8
168                 on error
```

Figure 11-1. SCREEN COBOL Sample Program (Page 3 of 4)

```
169                  perform TEST5-ERROR.
170
171      TEST6.
172     *     SWEDISH/FINNISH
173           Call scdragn2
174                on error
175                    perform TEST6-ERROR.
176
177      TEST7.
178     *     DANISH/NORWEGIAN
179           Call scdragn4
180                on error
181                    perform TEST7-ERROR.
182
183      RECOVER-SCRN1.
184        DISPLAY RECOVERY.
185
186      DISPLAY-RETURN.
187        MOVE "RETURN KEY PRESSED" TO ERR-MSG.
188        DISPLAY TEMP ERR-WINDOW.
189
190     /
191      TEST1-ERROR.
192           Move "on error returned from French (AZ) test" to info1.
193
194      TEST2-ERROR.
195           Move "on error returned from French (QW) test" to info1.
196
197      TEST3-ERROR.
198           Move "on error returned from German test" to info1.
199
200      TEST4-ERROR.
201           Move "on error returned from Spanish" to info1.
202
203      TEST5-ERROR.
204           Move "on error returned from United Kingdom test" to info1.
205
206      TEST6-ERROR.
207           Move "on error returned from Swedish/Finnish test" to info1.
208
209      TEST7-ERROR.
210           Move "on error returned from Danish/Norwegian test" to info1.
```

Figure 11-1. SCREEN COBOL Sample Program (Page 4 of 4)

Figure 11-2 shows the first page of the cross-reference listing. This is the cross-reference file list. It lists the name of each source file scanned. In this example, only one file, SCOBEX, was scanned.

```
PAGE 1


CROSSREF - CROSS-REFERENCE PROGRAM - T9622C00 - (15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1984, 1985, 1986


FILE NO.     FILE NAME
  [1]        $EM2.UCREF.SCOBEX
```

Figure 11-2. CROSSREF Listing--File List

The rest of the listing consists of the identifier list. See Figure 11-3. The identifier list describes each identifier in alphabetic order, showing:

- How it is defined (its attributes)
- Where it is defined (file name and number and line number)
- Where and how it is used in the program

Look at the entry for the identifier named ERR-WINDOW. The identifier header indicates that it is a level-5 screen field, which is part of the DRAGON-SCREEN screen. It is defined in the file SCOBEX at line 92.

The reference line indicates that ERR-WINDOW is referenced in the file SCOBEX at lines 103 and 188. Both of these references are read referenced (indicated by code blank).

Now look at the entry for the identifier named INFO1. The identifier header indicates that it is a level-1 alphanumeric display item. It is defined at line 42.

The reference line indicates that INFO1 is referenced in the file SCOBEX at lines 67, 107, 128, 192, 195, 198, 201, 204, 207, and 210. The reference at line 67 is a read reference indicated by code blank. All other references are write references (indicated by code W).

```
PAGE 2
NAME AND NAME QUALIFIER          LANGUAGE-DEPENDENT ATTRIBUTES                    DEFINITION POINT


 84 TOTAL SYMBOLS COLLECTED WITH 61 TOTAL REFERENCES COLLECTED

  DISPLAY-RETURN                    PARA                                          SCOBEX[1]       186
       SCOBEX[1]       138 M

  DRAGON-SCREEN                     01     SCREEN  BASE                           SCOBEX[1]       54
       SCOBEX[1]       117

  END-OF-TEST                       01     AN   DISP                             SCOBEX[1]       48
       SCOBEX[1]       108 W        110          126 W

  ERR-MSG                           01     AN   DISP                             SCOBEX[1]       47
       SCOBEX[1]       94           187 W

  ERR-WINDOW   OF DRAGON-SCREEN     05     SCREEN  FIELD                          SCOBEX[1]       92
       SCOBEX[1]       188

  F1                                MNEM    FUNC-KEY                              SCOBEX[1]       14
       SCOBEX[1]       123 M

  F14                               MNEM    FUNC-KEY                              SCOBEX[1]       21
       SCOBEX[1]       123 M

  F2                                MNEM    FUNC-KEY                              SCOBEX[1]       15
       SCOBEX[1]       123 M

  F3                                MNEM    FUNC-KEY                              SCOBEX[1]       16
       SCOBEX[1]       123 M

  F4                                MNEM    FUNC-KEY                              SCOBEX[1]       17
       SCOBEX[1]       123 M

  F5                                MNEM    FUNC-KEY                              SCOBEX[1]       17
       SCOBEX[1]       123 M

  F6                                MNEM    FUNC-KEY                              SCOBEX[1]       19
       SCOBEX[1]       123 M

  F7                                MNEM    FUNC-KEY                              SCOBEX[1]       20
       SCOBEX[1]       123 M

  INFO1                              01     AN   DISP                             SCOBEX[1]       42
       SCOBEX[1]       67           107 W        128 W    192 W    195 W    198 W    201 W    204 W     207 W
                       210 W

  INFO2                              01     AN   DISP                             SCOBEX[1]       43
       SCOBEX[1]       70

  INFO3                              01     AN   DISP                             SCOBEX[1]       44
```

Figure 11-3. CROSSREF Listing--Identifier List (Page 1 of 3)

```
PAGE 3
NAME AND NAME QUALIFIER          LANGUAGE-DEPENDENT ATTRIBUTES                DEFINITION POINT


     SCOBEX[1]        73

 INFO4                          01     AN   DISP                            SCOBEX[1]       45
     SCOBEX[1]        76

 INFO5                          01     AN   DISP                            SCOBEX[1]       46
     SCOBEX[1]        79

 L1   OF DRAGON-SCREEN          05     SCREEN  FIELD                         SCOBEX[1]       65
     SCOBEX[1]        118

 L2   OF DRAGON-SCREEN          05     SCREEN  FIELD                         SCOBEX[1]       68
     SCOBEX[1]        119

 L3   OF DRAGON-SCREEN          05     SCREEN  FIELD                         SCOBEX[1]       71
     SCOBEX[1]        120

 L4   OF DRAGON-SCREEN          05     SCREEN  FIELD                         SCOBEX[1]       74
     SCOBEX[1]        121

 RECOVER-SCRN1                  PARA                                         SCOBEX[1]       183
     SCOBEX[1]        137 M

 RETURN-KEY                     MNEM    FUNC-KEY                             SCOBEX[1]       23
     SCOBEX[1]        123 M

 SCDRAGN2
     SCOBEX[1]        173 I

 SCDRAGN3
     SCOBEX[1]        155 I

 SCDRAGN4
     SCOBEX[1]        179 I

 SCDRAGN5
     SCOBEX[1]        149 I

 SCDRAGN6
     SCOBEX[1]        143 I

 SCDRAGN7
     SCOBEX[1]        161 I

 SCDRAGN8
     SCOBEX[1]        167 I

 SF16                           MNEM    FUNC-KEY                             SCOBEX[1]       22
     SCOBEX[1]        124 M

 STEP1                          PARA                                         SCOBEX[1]       115
```

Figure 11-3. CROSSREF Listing--Identifier List (Page 2 of 3)

```
PAGE 4
NAME AND NAME QUALIFIER          LANGUAGE-DEPENDENT ATTRIBUTES              DEFINITION POINT


     SCOBEX[1]        109 M

 TERMINATION-STATUS              01     NM    COMP    SPCL-REG
     SCOBEX[1]        125       139

 TEST1                           PARA                                       SCOBEX[1]       142
     SCOBEX[1]        130 M

 TEST1-ERROR                     PARA                                       SCOBEX[1]       191
     SCOBEX[1]        145 M

 TEST2                           PARA                                       SCOBEX[1]       148
     SCOBEX[1]        131 M

 TEST2-ERROR                     PARA                                       SCOBEX[1]       194
     SCOBEX[1]        151 M

 TEST3                           PARA                                       SCOBEX[1]       154
     SCOBEX[1]        132 M

 TEST3-ERROR                     PARA                                       SCOBEX[1]       197
     SCOBEX[1]        157 M

 TEST4                           PARA                                       SCOBEX[1]       160
     SCOBEX[1]        133 M

 TEST4-ERROR                     PARA                                       SCOBEX[1]       200
     SCOBEX[1]        163 M

 TEST5                           PARA                                       SCOBEX[1]       166
     SCOBEX[1]        134 M

 TEST5-ERROR                     PARA                                       SCOBEX[1]       203
     SCOBEX[1]        169 M

 TEST6                           PARA                                       SCOBEX[1]       172
     SCOBEX[1]        135 M

 TEST6-ERROR                     PARA                                       SCOBEX[1]       206
     SCOBEX[1]        175 M

 TEST7                           PARA                                       SCOBEX[1]       178
     SCOBEX[1]        136 M

 TEST7-ERROR                     PARA                                       SCOBEX[1]       209
     SCOBEX[1]        181 M
```

Figure 11-3. CROSSREF Listing--Identifier List (Page 3 of 3)

SECTION 12

TAL

This section describes the TAL identifier classes and provides a sample TAL program and its cross-reference listing.

TAL IDENTIFIERS

The CROSSREF utility indexes TAL programs according to the identifiers listed in Table 12-1.

Table 12-1 also shows the default settings for each identifier class and what data types correspond to each of these classes.

Table 12-1. TAL Identifier Classes

CROSSREF Class	Default Setting	TAL Type
BLOCKDATAS	ON	Data BLOCKs
CONSTANTS	ON	LITERALs
MACROS	ON	DEFINEs
PROCEDURE PARAMS	ON	PROCs that are formal parameters
PROCEDURES	ON	PROCs
PROGLABELS	ON	LABELs
REGISTERS	ON	Index registers named in USE statements
SUBPROCS	ON	SUBPROCs
TYPES	ON	STRUCT templates
VARIABLES	ON	Variables including nontemplate STRUCTs

SAMPLE LISTING

The following example starts CROSSREF, scans the file named TALEX, and generates a listing to $s.#lp:

```
16> CROSSREF
CROSSREF-CROSS-REFERENCE PROGRAM-T9622C00-(15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1985, 1986
&SET LANGUAGE tal
&SCAN talex
&GENERATE /OUT $s.#lp/
&EXIT
17>
```

On the following pages, you can see the program and its cross-reference listing. The listing includes all TAL types because all classes are ON by default.

Figure 12-1 shows the TAL program that CROSSREF scanned to produce the cross-reference listing.

```
 99          INT PROC Ascii(v,rjust,stg);
100            INT        v;              ! the integer value to convert
101            INT        rjust;          ! right justify result flag
102            STRING    .stg;            ! target STRING
103          BEGIN
104          STRING       b[0:5] := [5*[" "],"0"];
105
106          INT          n;              ! number of digits converted
107          INT          sgn := 0;       ! nonzero if 'v' is negative
108          INT          k := 5;         ! index for converted digit
109
110          IF v < 0                     ! value is negative
111          THEN
112            BEGIN
113            sgn := 1;                  ! set negative value flag
114            v := -v;                   ! take absolute value
115            END;
116
117          WHILE v                      ! while there is a value left....
118          DO
119            BEGIN
120            b[k] := $udbl(v) '\' 10 + "0";  ! convert a character
121            v := v / 10;               ! compute the remainder
122            k := k - 1;                ! count the converted character
123            END;
124
125          IF sgn                       ! number is negative
126          THEN
127            BEGIN
128            b[k] := "-";               ! insert the sign
129            k := k - 1;                ! count it as a character
130            END;
131
132          IF NOT (n:=5-k)              ! check for an overflow
133          THEN
134            n := 1;                    ! return one character in that case
135
136          IF rjust                     ! move the resultant string to the
137          THEN                         ! user's target
138            stg[n-1] '=:' b[5] FOR n   ! reverse move if right justified
139          ELSE
140            stg ':=' b[6-n] FOR n;     ! otherwise forward move
141
142          RETURN n;                    ! return the string's length
143          END !ascii! ;
144
145
```

Figure 12-1. TAL Sample Program

Figure 12-2 shows the first page of the cross-reference listing. This is the cross-reference file list. It lists the name of each source file scanned. In this example, only one file, TALEX, was scanned.

```
PAGE 1


CROSSREF - CROSS-REFERENCE PROGRAM - T9622C00 - (15JUL87)
Copyright Tandem Computers Incorporated 1982, 1983, 1984, 1985, 1986


FILE NO.     FILE NAME
  [1]        $EM2.UCREF.TALEX
```

Figure 12-2. CROSSREF Listing--File List

The identifier list makes up the rest of the cross-reference listing. See Figure 12-3 on the following page. The identifier list describes each identifier in alphabetic order, showing:

- How it is defined (its attributes)
- Where it is defined (file name and number and line number)
- Where and how it is used in the program

Look at the entry for the identifier named STG. The identifier header indicates that it is part of the procedure named ASCII and that it is a STRING indirect variable. It is defined in the file TALEX at line 102.

It is defined again at line 99 as shown on the identifier reference line. The remainder of the reference line indicates that it is both write referenced (indicated by code W) and read referenced (indicated by code blank) at line 138 and line 140.

```
PAGE 2
NAME AND NAME QUALIFIER          LANGUAGE-DEPENDENT ATTRIBUTES                  DEFINITION POINT


8 TOTAL SYMBOLS COLLECTED WITH 34 TOTAL REFERENCES COLLECTED

 B  OF ASCII                      STRING DIRECT VARIABLE                       TALEX[1]       6
      TALEX[1]      22 W         30 W         40          42

 K  OF ASCII                      INT DIRECT VARIABLE                          TALEX[1]       10
      TALEX[1]      22           24 W         30          31 W    34

 N  OF ASCII                      INT DIRECT VARIABLE                          TALEX[1]       8
      TALEX[1]      34 W         36 W         40          42      44

 RJUST  OF ASCII                  INT DIRECT VARIABLE                          TALEX[1]       3
      TALEX[1]      1 D          38

 SGN  OF ASCII                    INT DIRECT VARIABLE                          TALEX[1]       9
      TALEX[1]      15 W         27

 STG  OF ASCII                    STRING INDIRECT VARIABLE                     TALEX[1]       4
      TALEX[1]      1 D          40 W         40          42 W        42

 V  OF ASCII                      INT DIRECT VARIABLE                          TALEX[1]       2
      TALEX[1]      1 D          12           16 W        16      19      22     23 W       23
```

Figure 12-3. CROSSREF Listing--Identifier List

APPENDIX A

SYNTAX SUMMARY

The following is a syntax summary of the commands that you can use when executing CROSSREF.

COMMENT

COMMENT *text*

ENV

```
                          [ LOG    ]
ENV [ / OUT file-name / [ SYSTEM ]
                          [ VOLUME ]
```

ERRORS

ERRORS [/ OUT *file-name* /]

EXIT

```
EXIT
```

FC

```
FC
```

GENERATE

```
GENERATE [ / OUT file-name / ]
```

HELP

```
                              [ command-name ]
HELP [ / OUT file-name / ] [ param-name   ]
                              [ <param-name> ]
```

LOG

```
    { TO file-name }
LOG {              }
    { STOP         }
```

OBEY

```
OBEY [ / OUT file-name / ] file-name
```

OUT

```
{ OUT file-name                          }
{ command / OUT file-name / param-name }
```

RESET

```
      { *                        }
RESET {                          }
      { attribute-specification }
```

attribute-specification

is one of the following:

CLASS [*class-name*]

class-name is one of the following:

BLOCKDATAS	INLINES	PROGLABELS
BLOCKS	KEYWORDS	REGISTERS
CONDITIONS	LINENOS	SCREENS
CONSTANTS	LITERALS	SUBPROCS
FILES	MACROS	SYSVARS
FMTLABELS	MNEMONICS	TYPES
FUNCTIONS	PROCEDURE PARAMS	VARIABLES
INDEXES	PROCEDURES	

CPU

DEFINITIONS ONLY

DIRECTIVES

EXCLUDE

INCLUDE

LANGUAGE

LIBRARY

MEM

OMIT

PRIORITY

PROGRAM

UNREF

SAVE

```
     { *                       }
SAVE {                         }
     { attribute-specification }
```

attribute-specification

is one of the following:

CLASS

CPU

DEFINITIONS ONLY

DIRECTIVES

LANGUAGE

LIBRARY

MEM

PRIORITY

PROGRAM

UNREF

SCAN

SCAN *file-list* [, *attribute-specification*] ...

file-list

has the following syntax:

```
{ file-name                          }
{ ( file-name [ , file-name ] ... ) }
```

attribute-specification

is one of the following:

```
      { { class-name }  [ ON  ] }
CLASS { { *           }  [ OFF ] }
      { class-list               }
```

includes the following detailed parameters:

class-name

is one of the following:

BLOCKDATAS	INLINES	PROGLABELS
BLOCKS	KEYWORDS	REGISTERS
CONDITIONS	LINENOS	SCREENS
CONSTANTS	LITERALS	SUBPROCS
FILES	MACROS	SYSVARS
FMTLABELS	MNEMONICS	TYPES
FUNCTIONS	PROCEDURE PARAMS	VARIABLES
INDEXES	PROCEDURES	

class-list

has the following syntax:

```
{        [ ON  ]          [ ON  ]        }
{ ( name [     ] [ , name [     ] ]... ) }
{        [ OFF ]          [ OFF ]        }
```

```
CPU cpu-number
```

```
DEFINITIONS ONLY
```

```
DIRECTIVES " [ ; ] directive... "
```

```
        { class-name                          }
EXCLUDE {                                     }
        { ( class-name [ , class-name ] ... ) }
```

```
        { class-name                          }
INCLUDE {                                     }
        { ( class-name [ , class-name ] ... ) }
```

```
          { BASIC   }
          { C       }
          { COBOL   }
LANGUAGE  { COBOL85 }
          { FORTRAN }
          { PASCAL  }
          { SCOBOL  }
          { TAL     }

LIBRARY file-name
MEM pages

     { file-name                              }
OMIT {                                        }
     { ( file-name [ , file-name ] ... ) }

PRIORITY priority-number

PROGRAM file-name

      { ON   }
UNREF { OFF  }
      { ONLY }
```

SET

SET *attribute-specification*

attribute-specification

is one of the following:

```
                    [ ON  ]
CLASS [ class-name  [     ] ]
                    [ OFF ]
```

class-name is one of the following:

```
       BLOCKDATAS     INLINES            PROGLABELS
       BLOCKS         KEYWORDS           REGISTERS
       CONDITIONS     LINENOS            SCREENS
       CONSTANTS      LITERALS           SUBPROCS
       FILES          MACROS             SYSVARS
       FMTLABELS      MNEMONICS          TYPES
       FUNCTIONS      PROCEDURE PARAMS   VARIABLES
       INDEXES        PROCEDURES
```

CPU *cpu-number*

DEFINITIONS ONLY
DIRECTIVES " [;] *directive*... "

```
        { class-name                              }
EXCLUDE {                                         }
        { ( class-name [ , class-name ] ... ) }

        { class-name                              }
INCLUDE {                                         }
        { ( class-name [ , class-name ] ... ) }

         { BASIC   }
         { C       }
         { COBOL   }
LANGUAGE { COBOL85 }
         { FORTRAN }
         { PASCAL  }
         { SCOBOL  }
         { TAL     }
```

LIBRARY *file-name*

MEM *pages*

```
     { file-name                          }
OMIT {                                    }
     { ( file-name [ , file-name ] ... ) }
```

PRIORITY *priority-number*

```
PROGRAM file-name

      { ON   }
UNREF { OFF  }
      { ONLY }
```

SHOW

```
                              { *                          }
SHOW [ / OUT file-name / ] {                               }
                              { attribute-specification }
```

attribute-specification

is one of the following:

CLASS [*class-name*]

class-name is one of the following:

BLOCKDATAS	INLINES	PROGLABELS
BLOCKS	KEYWORDS	REGISTERS
CONDITIONS	LINENOS	SCREENS
CONSTANTS	LITERALS	SUBPROCS
FILES	MACROS	SYSVARS
FMTLABELS	MNEMONICS	TYPES
FUNCTIONS	PROCEDURE PARAMS	VARIABLES
INDEXES	PROCEDURES	

CPU

DEFINITIONS ONLY

DIRECTIVES

EXCLUDE

INCLUDE

LANGUAGE

LIBRARY

MEM

OMIT

PRIORITY

PROGRAM

UNREF

SYSTEM

SYSTEM [*system*]

VOLUME

VOLUME *volume* [.*subvol*]

APPENDIX B

WARNING AND ERROR MESSAGES

This appendix contains a summary of all the warning and error messages that CROSSREF generates.

WARNING MESSAGES

Warning messages indicate a minor discrepancy in the operation of CROSSREF. CROSSREF continues to execute the command. Warning messages are printed for your information.

Warning messages are either prefixed with "**** WARNING ****" or displayed exactly as shown in this appendix.

Exclude list identifier not found in crossref - *identifier*

Cause: CROSSREF placed the specified identifier on the exclude list but was unable to find any references to exclude.

Recovery: This message merely informs you of the situation; no action is required.

Include list identifier not found in crossref - *identifier*

Cause: CROSSREF placed the specified identifier on the include list but was unable to find any references to include.

Recovery: You might have specified the wrong files for CROSSREF to scan. If so, you can correct the problem with a subsequent SCAN command.

No Cross-Reference information available

Cause: You issued a GENERATE command without issuing a SCAN command since the last GENERATE or CROSSREF startup.

Recovery: Issue a SCAN command before issuing the GENERATE command.

ERROR MESSAGES

Error messages indicate that CROSSREF is unable to execute a command for some reason. You must correct the situation before CROSSREF can execute the command.

Error messages are either prefixed with "**** ERROR ****" or displayed exactly as shown in this appendix.

```
Compiler Communication Lost
```

Cause: The compiler process halted for some reason. CROSSREF also generates this message when the processor running the compiler fails.

Recovery: Examine the compiler error file with the ERRORS command.

```
Crossref:  Compiler version not compatible
```

Cause: CROSSREF attempted to run an invalid version of the compiler. If you did not set the PROGRAM attribute specification, then the version of the compiler on the system and volume that you are using was not updated to run with CROSSREF. If you did set the PROGRAM attribute specification, then the compiler you specified is not compatible with CROSSREF.

Recovery: Either update the compiler on $SYSTEM.SYSTEM, or specify a valid version of the compiler in the PROGRAM attribute specification.

Effective input record is too long

Cause: You entered a line greater than 528 characters.

Recovery: Break the line into several commands and reissue it.

file-error-message (*error-number*) *file-name*

Cause: CROSSREF produces this error when it encounters a GUARDIAN file error on one of the files that it is scanning.

Recovery: See the *Operator Messages: Console Format* for a description of the error.

Identifier found in exclude list

Cause: The identifier specified in the last SET INCLUDE command was already on the exclude list. CROSSREF ignores the command, and the identifier remains on the exclude list.

Recovery: If you want to place the identifier on the include list, you must first reset the exclude list.

```
Identifier found in include list
```

Cause: The identifier specified in the last SET EXCLUDE command was already on the include list. CROSSREF ignores the command, and the identifier remains on the include list.

Recovery: If you want to place the identifier on the exclude list, you must first reset the include list.

```
Identifier too long
```

Cause: You specified an identifier name greater than 31 characters.

Recovery: You must limit identifier names to 31 characters. This rule does not interfere with any language's rules for symbol names, since all languages supported by Tandem require that the first 31 characters be unique for all identifiers.

```
Illegal LOG file - ignored
```

Cause: The LOG file specified was the same as the current IN file or OUT file. CROSSREF continues using only the current OUT file for printing output and error messages.

Recovery: Specify a log file whose name is not the same as the name of the current IN file or OUT file.

```
Illegal OBEY file - ignored
```

Cause: The OBEY file specified was the same as the current IN file. CROSSREF continues reading commands from the current IN file.

Recovery: Reissue the command with the correct file name.

```
Illegal OUT file - ignored
```

Cause: The OUT file specified in the last command was the same as the current IN file. CROSSREF continues using the current OUT file.

Recovery: Reissue the command with a valid file name.

```
Integer conversion error
```

Cause: You entered an invalid number for an attribute specification.

Recovery: All numbers must be entered as decimal integers within the range specified for the given attribute specification.

```
Invalid file name
```

Cause: The file name specified in the last command did not conform to GUARDIAN standards.

Recovery: Reissue the command with a valid file name.

```
Invalid Subvolume name
```

Cause: The volume or subvolume name specified in the last command did not conform to GUARDIAN standards.

Recovery: Reissue the command with a valid GUARDIAN volume and/or subvolume name.

```
Invalid syntax
```

Cause: You made an error typing the previous command.

Recovery: Examine the correct syntax for the command, and type the command correctly. You can use FC to fix the error.

```
Invalid System name
```

Cause: The system specified in the last command did not exist, or the name did not conform to GUARDIAN standards.

Recovery: Reissue the command with a valid GUARDIAN system name.

```
Newprocess error error-number [ , file-error-number ]
```

Cause: An error occurred while CROSSREF was attempting to start the compiler. The cause of the error depends on the error number specified in the error message. If the newprocess error represents a file management error, then that error number is displayed also.

Recovery: See the *Operator Messages: Console Format* for a description of the error.

```
No help available for command
```

Cause: The command specified has no help message associated with it. You might have misspelled the command.

Recovery: Type HELP to get a list of all CROSSREF commands.

```
No language specified, scan is aborted
```

Cause: You did not set the LANGUAGE attribute before issuing a SCAN command.

Recovery: Specify the LANGUAGE attribute in a SET command or in a SCAN command.

```
OBEY nesting exceeds maximum
```

Cause: You attempted to nest your OBEY files greater than four deep.

Recovery: Only four OBEY files can be active at any one time. CROSSREF ignores the fifth OBEY command and continues executing commands from the fourth OBEY file.

```
SORT error error-number
An error has occurred with the SORT/MERGE product
```

Cause: CROSSREF encountered a SORT error while sorting its output.

Recovery: See the *SORT/MERGE Users Guide* for a description of the error.

Unterminated continuation line

Cause: You placed a continuation character at the end of a line, but the line was followed by an end of file.

Recovery: Either type the line with no continuation character, or complete the command on the following line before terminating input.

Unterminated string

Cause: You failed to put a closing quote on a DIRECTIVES attribute specification.

Recovery: Reissue the command with a closing quote included.

FATAL ERROR MESSAGES

The following is a list of fatal error messages generated by CROSSREF. Fatal errors indicate situations in which CROSSREF cannot continue operation. When CROSSREF encounters a fatal error, it displays an error message to indicate the cause and halts all processing.

Fatal error messages are either prefixed with "**** FATAL ERROR ****" or displayed exactly as shown in this appendix.

ALLOCATESEGMENT on *vol* failed due to lack of space

Cause: CROSSREF could not allocate the swap file for its extended memory segment on the indicated volume because there was insufficient disk space.

Recovery: Set the SWAPVOL param to a disk volume that has more space.

ALLOCATESEGMENT on *vol* failed with error *error*

Cause: CROSSREF could not allocate the swap file for its extended memory segment on the indicated volume because the listed GUARDIAN error occurred.

Recovery: The recovery method depends on the error. See the *Operator Messages: Console Format* for a description of the error.

```
Internal error at P=%address
```

Cause: This is a CROSSREF internal error.

Recovery: If you get this error, contact your Tandem representative. Save all input and output files, and, if no LOG file was produced, record all commands issued prior to the error.

```
Paging file error:  error number
```

Cause: CROSSREF encountered an error on one of its temporary files. The cause of the error depends on the exact error number listed.

Recovery: See the *Operator Messages: Console Format* for a description of the error.

```
Trap %n at p=%p-addr
```

Cause: CROSSREF has failed due to a run-time trap. *n* is a two digit trap number in octal notation and *p-addr* is the five digit CROSSREF code location where the trap occurred. See the *GUARDIAN Programmer's Guide* for a description of the trap number.

Recovery: This message appears as a result of a CROSSREF internal error. If you get this message, contact your Tandem representative.

```
Unable to allocate memory; swap file error = error
```

Cause: CROSSREF was unable to allocate the additional memory needed to complete the cross-reference.

Recovery: The recovery action depends on the three-digit GUARDIAN error number listed in the error message. See the *Operator Messages: Console Format* for a description of the error message. Changing volumes with the SWAPVOL param might eliminate the problem.

INDEX