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
<table>
<thead>
<tr>
<th>Document History</th>
<th>Edition</th>
<th>Part Number</th>
<th>Product Version</th>
<th>Operating System Version</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Edition</td>
<td>82316 A00</td>
<td>CROSSREF A00</td>
<td>GUARDIAN A04/E05</td>
<td>October 1982</td>
<td></td>
</tr>
<tr>
<td>Second Edition</td>
<td>82516 A00</td>
<td>CROSSREF B00</td>
<td>GUARDIAN B00/E08</td>
<td>March 1985</td>
<td></td>
</tr>
<tr>
<td>Third Edition</td>
<td>82597</td>
<td>CROSSREF C00</td>
<td>GUARDIAN 90 C00</td>
<td>November 1987</td>
<td></td>
</tr>
</tbody>
</table>

New editions incorporate any updates issued since the previous edition.

**Trademarks or Service Marks**
The following are trademarks or service marks of Tandem Computers Incorporated:

- GAX
- BATCH-PLUS
- CCS
- CLX
- DB-BATCHFILE
- DONAH
- DNS
- ENABLE
- ENCOMPASS
- ENFORM
- ENVoy
- EXCHANGE
- EXPAND
- EXT
- FAKLINK
- FOX
- GUARDIAN
- HITS NONSTOP
- INSPECT
- INF
- LaserPLUS
- Lighthouse
- Lighthouse Keeper
- LIN
- MEASURE
- MULTILAN
- NetBath
- NonStop
- PATHMAKER
- PCFORMAT
- PSMAIL
- PS TEXT
- PSX
- RDF
- SAFE-GUARD
- SAFE-T-Net
- SEEoney
- SNAX
- T-TEXT
- TACL
- TAIL
- Tandem
- Tandem Logo
- TGAL
- THL
- TIL
- T.I.M.E.
- T.M.F.
- TRANsfER
- TUNEx
- TWINAC
- TWINCOG
- TWINPRO
- TAP
- T6
- T20
- VIEWPOINT
- VIEWSYS
- VLM
- VUX
- WFLINK
- XL8
- X1600
- XRAY

**Copyright**
All rights reserved. No part of this document may be reproduced in any form, including photocopying or translation to another language, without the prior written consent of Tandem Computers Incorporated. Copyright © 1987 Tandem Computers Incorporated.
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

PREFACE ...................................................... ix
NOTATION CONVENTIONS ............................................... xi

SECTION 1. INTRODUCTION ............................................. 1-1
  What Is CROSSREF? ............................................. 1-1
  Which Languages Does CROSSREF Support? ...................... 1-2
  How Does CROSSREF Work? ..................................... 1-2
  Stand-Alone CROSSREF ......................................... 1-3
  Compiler-Dependent CROSSREF .................................. 1-3

SECTION 2. RUNNING CROSSREF ....................................... 2-1
  Overview .................................................. 2-1
  CROSSREF Command ........................................... 2-1
    Interactive Mode ......................................... 2-3
    Noninteractive Mode ..................................... 2-4
  Examples of CROSSREF Operation .............................. 2-4
    Single-Language Program .................................. 2-5
    Multi-Language Program .................................. 2-6

SECTION 3. INTERPRETING CROSSREF OUTPUT ......................... 3-1
  CROSSREF Listing .......................................... 3-1
  File List ............................................... 3-1
  Identifier List ......................................... 3-2
  CROSSREF Identifier Classes ............................... 3-7
  Compilation Errors ...................................... 3-7

SECTION 4. CROSSREF COMMANDS ..................................... 4-1
  CROSSREF Attribute Specifications ........................... 4-3
  COMMENT Command .......................................... 4-5
  ENV Command ............................................. 4-6
  ERRORS Command ........................................... 4-8
  EXIT Command .............................................. 4-10
  FC Command ............................................... 4-11
  GENERATE Command ......................................... 4-12
  HELP Command ............................................. 4-14
CONTENTS

LOG Command .............................................. 4-16
OBEY Command ............................................. 4-18
OUT Command ............................................. 4-20
RESET Command ............................................ 4-22
SAVE Command ............................................. 4-25
SCAN Command ............................................. 4-28
SET Command .............................................. 4-35
SHOW Command ............................................. 4-41
SYSTEM Command ........................................... 4-45
VOLUME Command ........................................... 4-46

SECTION 5. C ............................................... 5-1
  C Identifiers ............................................. 5-1
  Sample Listing .......................................... 5-2

SECTION 6. COBOL 74 ........................................ 6-1
  COBOL 74 Identifiers ...................................... 6-1
  Using Compiler Directives in CROSSREF ................. 6-3
  Sample Listing ........................................... 6-4
  Compiler Attributes .................................... 6-13
    Alphabet-Name ......................................... 6-13
    Condition-Name ....................................... 6-13
    Data-Name ............................................ 6-14
    File-Name ............................................ 6-15
    Index-Name .......................................... 6-16
    Literals ............................................. 6-16
    Mnemonic-Name ....................................... 6-16
    Paragraph-Name ...................................... 6-17
    Section-Name ......................................... 6-17

SECTION 7. COBOL85 ......................................... 7-1
  COBOL85 Identifiers ..................................... 7-1
  Using Compiler Directives in CROSSREF ................. 7-3
  Sample Listing .......................................... 7-4
  Compiler Attributes .................................... 7-13
    Alphabet-Name ......................................... 7-13
    Class-Name ............................................ 7-13
    Condition-Name ....................................... 7-13
    Data-Name ............................................ 7-14
    File-Name ............................................ 7-15
    External Switch ...................................... 7-16
    Index-Name .......................................... 7-16
    Literals ............................................. 7-17
    Mnemonic-Name ....................................... 7-17
    Paragraph-Name ...................................... 7-17
    Program-Name ......................................... 7-18
    Section-Name ......................................... 7-18
    Symbolic-Character .................................... 7-18

SECTION 8. EXTENDED BASIC .................................. 8-1
  EXTENDED BASIC Identifiers ............................. 8-1
  Sample Listing .......................................... 8-2
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.
The following list summarizes the conventions for syntax notation in this manual.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPERCASE LETTERS</td>
<td>Uppercase letters represent keywords and reserved words; enter these items exactly as shown.</td>
</tr>
<tr>
<td>italics</td>
<td>Lowercase italic letters represent variable items that you supply.</td>
</tr>
<tr>
<td>Brackets []</td>
<td>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.</td>
</tr>
<tr>
<td>Braces {}</td>
<td>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.</td>
</tr>
<tr>
<td>Ellipsis ...</td>
<td>An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed syntax items any number of times.</td>
</tr>
<tr>
<td>Spaces</td>
<td>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.</td>
</tr>
<tr>
<td>Punctuation</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.
INTRODUCTION
Which Languages Does CROSSREF Support?

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)

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.
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:

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

**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.)

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

**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:

```plaintext
&SET LANGUAGE cobol
```

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

```plaintext
&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:

   \texttt{\&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.

   \texttt{\&SET UNREF ON}

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

   \texttt{\&SET EXCLUDE ERROR}

6. Tell CROSSREF to scan the FPROCS file:

   \texttt{\&SCAN fprocs}

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

   \texttt{\&GENERATE /OUT csquare/}

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

8. Finally, terminate the session by entering EXIT:

   \texttt{\&EXIT}

15>
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.
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,

<table>
<thead>
<tr>
<th>FILE NO.</th>
<th>FILE NAME.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>$EQ1.TEST.WCC</td>
</tr>
</tbody>
</table>

**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.

![Identifier Header Format](image)

**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.
INTERPRETING CROSSREF OUTPUT
Identifier List

**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.

![Sample Reference Entry](image)

**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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>blank</td>
<td>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.</td>
</tr>
<tr>
<td>D</td>
<td>designates a definition. This refers to any nonexecutable statement that defines the type of variable or its storage.</td>
</tr>
<tr>
<td>I</td>
<td>designates an invocation reference. This is only generated by procedures, subroutines, parametric procedures, and function calls.</td>
</tr>
<tr>
<td>M</td>
<td>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.</td>
</tr>
<tr>
<td>P</td>
<td>designates a parameter reference. This occurs only when the identifier is passed as a reference parameter.</td>
</tr>
<tr>
<td>W</td>
<td>designates a write reference. This occurs whenever a statement might change the value of the identifier. This does not include parameters to subroutine calls.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>BLOCKDATAS</th>
<th>INDEXES</th>
<th>PROGLABELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCKS</td>
<td>INLINES</td>
<td>REGISTERS</td>
</tr>
<tr>
<td>CONDITIONS</td>
<td>LINENOS</td>
<td>SCREENS</td>
</tr>
<tr>
<td>CONSTANTS</td>
<td>MACROS</td>
<td>SUBPROCS</td>
</tr>
<tr>
<td>FILES</td>
<td>MNEMONICS</td>
<td>SYSVARS</td>
</tr>
<tr>
<td>FMTLABELS</td>
<td>PROCEDURE</td>
<td>TYPES</td>
</tr>
<tr>
<td>FUNCTIONS</td>
<td>PARAMS</td>
<td>VARIABLES</td>
</tr>
</tbody>
</table>

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)

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

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

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASS</td>
<td>controls which types of identifiers are reported by setting the identifier class mapped to the type ON or OFF.</td>
</tr>
<tr>
<td>CPU</td>
<td>specifies the CPU on which the compiler runs.</td>
</tr>
<tr>
<td>DEFINITIONS ONLY</td>
<td>prints only the definition of the reported identifiers.</td>
</tr>
<tr>
<td>DIRECTIVES</td>
<td>passes compiler directives to the compiler.</td>
</tr>
<tr>
<td>EXCLUDE</td>
<td>specifies which identifiers to exclude from the listing.</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>specifies which identifiers to include in the listing.</td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>sets the language compiler that CROSSREF uses to scan the file.</td>
</tr>
<tr>
<td>LIBRARY</td>
<td>specifies a library file to be passed to the COBOL compiler during start-up.</td>
</tr>
<tr>
<td>MEM</td>
<td>specifies the amount of memory to be reserved for the compiler during a SCAN.</td>
</tr>
</tbody>
</table>
Table 4-2. CROSSREF Attribute Specifications (Page 2 of 2)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMIT</td>
<td>excludes specified files from the listing.</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>specifies the execution priority at which the compiler runs.</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>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.)</td>
</tr>
<tr>
<td>UNREF</td>
<td>specifies whether CROSSREF should include unreferenced identifiers in the listing.</td>
</tr>
</tbody>
</table>
**COMMENT COMMAND**

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

```plaintext
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:

```plaintext
&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:

```
ENV [ / OUT file-name / [ LOG ]
  [ 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
CROSSREF COMMANDS

ERRORS Command

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:

```device
&SET LANGUAGE cobol
&SCAN build1
&ERRORS /OUT errfile/
&GENERATE /OUT $s.#lpl/
&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
.LITERAL ON
  .ISET
  SET LITERAL ON
  .RS
  SET LITERALS ON
  .
```

- deletes the extra letter T
- inserts the command SET (see * below)
- replaces the letter A with S
- executes the command

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

```cobol
&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:

```
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>
CROSSREF COMMANDS
OBEY Command

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
CROSSREF COMMANDS
OUT Command

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.
CROSSREF COMMANDS
Considerations

<table>
<thead>
<tr>
<th>PRIORITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>restores the priority setting for compiler runs to the priority of CROSSREF.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>restores the program setting.</td>
</tr>
</tbody>
</table>

| 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:

```plaintext
{ *                       }
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    | Procs       | 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:

```plaintext
{ ( 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.
SCAN Command

EXCLUDE

{ class-name }

{ ( 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.

INCLUDE

{ class-name }

{ ( 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.

LANGUAGE

{ BASIC }

{ C }

{ COBOL }

{ 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.
CROSSREF COMMANDS

Examples

- 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.
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.

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.

LANGUAGE { BASIC }  
{ C }  
{ COBOL }  
{ 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.
CROSSREF COMMANDS

Examples

- 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:

\[
\text{SHOW} \ [ \ / \ \text{OUT} \ \text{file-name} \ / \ ] \{ \ast \} \{ \ \text{attribute-specification} \}
\]

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.

\ast

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:
CROSSREF COMMANDS
SHOW Command

| 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:

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

<table>
<thead>
<tr>
<th>CROSSREF Class</th>
<th>Default Setting</th>
<th>C Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONSTANTS</td>
<td>ON</td>
<td>Enumeration constants</td>
</tr>
<tr>
<td>FUNCTIONS</td>
<td>ON</td>
<td>Functions</td>
</tr>
<tr>
<td>MACROS</td>
<td>ON</td>
<td>#define(s)</td>
</tr>
<tr>
<td>TYPES</td>
<td>ON</td>
<td>User-defined and compiler-defined (anonymous) types</td>
</tr>
<tr>
<td>VARIABLES</td>
<td>ON</td>
<td>All variables</td>
</tr>
</tbody>
</table>

**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)
&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.
/* usage:  wc <file1> [<file2>...] */

This program will count the number of lines, words, and characters in a text file and report them to standard out. When multiple files are specified a running total of all of the files is printed in addition to the individual files' totals.

#pragma runnable

#include <stdioh>
#include <ctypeh>
#include <stdlibh>

#define TRUE        1
#define FALSE       0
#define MAXLINE     256   /* maximum length of an input line */

long total_lines, total_words, total_chars;

void count(FILE *fp, char *file);

int main(int argc, char *argv[])
{
    int i;
    FILE *fp;

    if (argc < 2)
    {
        fprintf(stderr, "usage:  wc <file1> [<file2>...]\n\n"); return EXIT_SUCCESS;
    }
    for (i = 1; i < argc; i++)
    {
        if ((fp = fopen(argv[i], "r")) == NULL)
            {
                fprintf(stderr, "can't open %s\n", argv[i]);
                continue;
            }
        count(fp, argv[i]);
        if (fclose(fp) < 0)
            fprintf(stderr, "can't close %s\n", argv[i]);
    }
    printf("\n*** Total:  %ld line(s), %ld word(s), %ld character(s)\n",
            total_lines,total_words,total_chars);
}

/* Count the number of lines, words, and characters in a text file. */

void count(FILE *fp, char *file)
{
    unsigned int nl; long nw, nc;

# C Sample Listing

```c
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 == '\n' || *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.

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).
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>106 TOTAL SYMBOLS COLLECTED WITH 205 TOTAL REFERENCES COLLECTED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXIT_SUCCESS</td>
<td></td>
<td>STDLIBH[4] 23</td>
</tr>
<tr>
<td>WCC[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FALSE</td>
<td></td>
<td>WCC[1] 17</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>63 M</td>
<td></td>
</tr>
<tr>
<td>FILE</td>
<td></td>
<td>STDIOTH[2] 35</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>23 M 28 M 53 M</td>
<td></td>
</tr>
<tr>
<td>STDIOTH[2]</td>
<td>37 M 65 M 68 M 73 M 76 M 81 M 84 M 89 M 95 M</td>
<td></td>
</tr>
<tr>
<td>98 M 101 M 104 M 113 M 116 M 121 M 128 M 132 M 142 M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>147 M 151 M 154 M 159 M 169 M 172 M 183 M 189 M 194 M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXLINE</td>
<td></td>
<td>WCC[1] 19</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>57 M 61 M</td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td></td>
<td>STDIOTH[2] 11</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>37 M</td>
<td></td>
</tr>
<tr>
<td>TRUE</td>
<td></td>
<td>WCC[1] 16</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>67 M</td>
<td></td>
</tr>
<tr>
<td>_L</td>
<td>WCC[1] 64 M 71 M</td>
<td></td>
</tr>
<tr>
<td>_N</td>
<td>WCC[1] 64 M</td>
<td></td>
</tr>
<tr>
<td>_P</td>
<td>WCC[1] 70 M</td>
<td></td>
</tr>
<tr>
<td>_S</td>
<td>WCC[1] 69 M</td>
<td></td>
</tr>
<tr>
<td>_U</td>
<td>WCC[1] 64 M 71 M</td>
<td></td>
</tr>
<tr>
<td>__IOB</td>
<td>pointer to struct, in block __IOB</td>
<td>WCC[1] 32</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>32 M 39 M 44 M</td>
<td></td>
</tr>
<tr>
<td>STDIOTH[2]</td>
<td>37 M</td>
<td></td>
</tr>
<tr>
<td>_ctype</td>
<td>array[] of uns char, in block _ctype</td>
<td>WCC[1] 64</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>64 M 69 M 70 M 71 M</td>
<td></td>
</tr>
<tr>
<td>CTYPEH[3]</td>
<td>24 M</td>
<td></td>
</tr>
<tr>
<td>argc</td>
<td>short formal</td>
<td>WCC[1] 26</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>26 D 30 M 35 M</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5-3. CROSSREF Listing--Identifier List (Page 1 of 3)
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>argv of main</td>
<td>pointer to pointer to uns char formal</td>
<td>WCC[1] 26</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>26 D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>37 M</td>
<td></td>
</tr>
<tr>
<td>count</td>
<td>WCC[1]</td>
<td>54 D</td>
</tr>
<tr>
<td></td>
<td>23 M</td>
<td></td>
</tr>
<tr>
<td>file of count</td>
<td>pointer to uns char formal</td>
<td>WCC[1] 54</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>54 D</td>
<td></td>
</tr>
<tr>
<td>fp of main</td>
<td>pointer to struct automatic</td>
<td>WCC[1] 28</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>28 D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>37 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>42 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>43 M</td>
<td></td>
</tr>
<tr>
<td>fprintf</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>39 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>44 M</td>
<td></td>
</tr>
<tr>
<td>i of main</td>
<td>short automatic</td>
<td>WCC[1] 27</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>27 D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>37 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>39 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>42 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>44 M</td>
<td></td>
</tr>
<tr>
<td>inword of count</td>
<td>short automatic</td>
<td>WCC[1] 58</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>58 D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>63 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>67 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>72 M</td>
<td></td>
</tr>
<tr>
<td>isalnum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WCC[1]</td>
<td>64 M</td>
<td></td>
</tr>
<tr>
<td>isalpha</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WCC[1]</td>
<td>71 M</td>
<td></td>
</tr>
<tr>
<td>ispunct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WCC[1]</td>
<td>70 M</td>
<td></td>
</tr>
<tr>
<td>isspace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WCC[1]</td>
<td>69 M</td>
<td></td>
</tr>
<tr>
<td>line of count</td>
<td>array[256] of uns char automatic</td>
<td>WCC[1] 57</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>57 D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>61 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>63 M</td>
<td></td>
</tr>
<tr>
<td>nc of count</td>
<td>long automatic</td>
<td>WCC[1] 56</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>56 D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>61 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>76 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>78 M</td>
<td></td>
</tr>
<tr>
<td>nl of count</td>
<td>uns short automatic</td>
<td>WCC[1] 55</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>55 D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>61 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>74 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>78 M</td>
<td></td>
</tr>
<tr>
<td>nw of count</td>
<td>long automatic</td>
<td>WCC[1] 56</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>56 D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>61 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>66 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>75 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>78 M</td>
<td></td>
</tr>
<tr>
<td>p of count</td>
<td>pointer to uns char automatic</td>
<td>WCC[1] 59</td>
</tr>
<tr>
<td>WCC[1]</td>
<td>59 D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>63 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>69 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>71 M</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5-3. CROSSREF Listing--Identifier List (Page 2 of 3)
### PAGE 4

<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>printf</td>
<td>WCC[1] 46 M 77 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STDIOH[2] 64 M</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STDLIBH[4] 54 M 55 M 62 M 70 M</td>
<td></td>
</tr>
<tr>
<td>total_chars</td>
<td>long, in block total_chars</td>
<td>WCC[1] 21</td>
</tr>
<tr>
<td></td>
<td>WCC[1] 21 D 47 M 76 M</td>
<td></td>
</tr>
<tr>
<td>total_lines</td>
<td>long, in block total_lines</td>
<td>WCC[1] 21</td>
</tr>
<tr>
<td></td>
<td>WCC[1] 21 D 47 M 74 M</td>
<td></td>
</tr>
<tr>
<td>total_words</td>
<td>long, in block total_words</td>
<td>WCC[1] 21</td>
</tr>
<tr>
<td></td>
<td>WCC[1] 21 D 47 M 75 M</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5-3. CROSSREF Listing--Identifier List (Page 3 of 3)
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 default settings for each identifier class and what COBOL 74 data types correspond to each of these classes.
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.
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.

```
&SET LANGUAGE cobol
&SET LIBRARY mylib
&SET DIRECTIVES "ANSI;SETTOG 1"
&SCAN bprog
&GENERATE /OUT $s.#cros/
&EXIT
```

13>
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)
   &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.
IDENTIFICATION DIVISION.
  PROGRAM-ID.  FUPPERWARE.
  AUTHOR.  ANN COBOL.
  INSTALLATION.  TRANSACTIONS ANONYMOUS.

ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
    SOURCE-COMPUTER.  TANDEM TXP.
    OBJECT-COMPUTER.  TANDEM TXP.

INPUT-OUTPUT SECTION.
  FILE-CONTROL.
    SELECT MESSAGE-IN-FILE
      ASSIGN TO "$RECEIVE"
      FILE STATUS IS RECEIVE-FILE-STATUS.

WORKING-STORAGE SECTION.
  01 MESSAGE-SOURCE-REC.
    05 SYSTEM-FLAG   PIC S9  COMP.
    05 ENTRY-NUMBER PIC 999 COMP.
    05 FILLER       PIC X(1024).
  01 PROCESS-ID.
    10 PROCESS-NAME PIC X(6).
    10 CPU-PIN.
    15 CPU-PART    PIC X.
    15 PIN-PART    PIC X.
    05 FILLER     PIC X(16).
  01 CPU-PIN-REDEF.
    05 ALPHA-CPU.
    10 CPU-HIGH-BYTE PIC X.
    10 CPU-LOW-BYTE PIC X.

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

DECLARATIVES.
HANDLE-INFILE-ERRORS SECTION.
USE AFTER STANDARD ERROR PROCEDURE ON MESSAGE-IN-FILE.

INFILE-ERROR.
IF STAT-1 = 1
DISPLAY "EOF on $RECEIVE"
ELSE
DISPLAY "RECEIVE FILE ERROR STATUS = "
RECEIVE-FILE-STATUS
.
END DECLARATIVES.

AA SECTION.
AA-1.
OPEN INPUT MESSAGE-IN-FILE.
MOVE ZERO TO SU-ERROR
                  SYS-MSG-CODE.

* Inject INFO command into STARTUP message to pass to FUP
ENTER "PUTSTARTUPTEXT"
      USING STRING-PORTION,
      INFO-COMMAND,
      NULL-CPLIST
      GIVING STARTUP-RESULT.

* Start FUP
ENTER "CREATEPROCESS"
      USING FUP,
      FUP-NAME,
      SEND-ALL-MSGS,
      PRIORITY-EQ-MINE,
      PROCESSOR-EQ-MINE,
      MEMORY-AS-USUAL,
      OMITTED
      GIVING SU-ERROR.

* Await termination of FUP, or report it never started
IF SU-ERROR = 0
      PERFORM WATCH
      UNTIL SYS-MSG-STOP
      OR SYS-MSG-ABEND
ELSE
      PERFORM DISPLAY-STARTUP-FAILURE
.

STOP RUN.

DISPLAY-STARTUP-FAILURE.
IF SU-ERROR = 1
      DISPLAY FUP-FAILED
      " -- REQUIRED PARAMETER MISSING OR ILLEGAL"
ELSE IF SU-ERROR = 2
      DISPLAY FUP-FAILED
      " -- ILLEGAL PROGRAM FILE NAME ("
      FUP "))"
ELSE IF SU-ERROR = 3
      DISPLAY FUP-FAILED
      " -- INFILE, OUTFILE, OR DEFAULT VOLUME"
      DISPLAY " NAME CANNOT BE CONVERTED TO NETWORK FORM"
ELSE IF SU-ERROR < 256
      DISPLAY FUP-FAILED
      " -- File management error #"
      SU-ERROR
ELSE
      -- Received raw error from NEWPROCESS system procedure.
      -- Decompose it into left byte and right byte values.
      DIVIDE SU-ERROR
      BY 256
      GIVING NEWPROCESS-ERR-LEFT
      REMAINDER NEWPROCESS-ERR-RIGHT.

Figure 6-1. COBOL 74 Sample Program (Page 3 of 4)
DISPLAY FUP-FAILED
" -- NEWPROCESS error #"
SU-ERROR
" = ("
NEWPROCESS-ERR-LEFT
",""
NEWPROCESS-ERR-RIGHT
")"
.

WATCH.
READ MESSAGE-IN-FILE.
PERFORM CAPTURE-CPU-PIN.
DISPLAY "------"
"(" NUMERIC-CPU "," NUMERIC-PIN ")"
"------"
SYS-MSG-CODE.

CAPTURE-CPU-PIN.
MOVE CPU-PART TO CPU-LOW-BYTE.
MOVE LOW-VALUES TO CPU-HIGH-BYTE.
MOVE PIN-PART TO PIN-LOW-BYTE.
MOVE LOW-VALUES TO PIN-HIGH-BYTE.
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.

<table>
<thead>
<tr>
<th>FILE NO.</th>
<th>FILE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>$EM2.UCREF.COBEX</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Name and Name Qualifier</th>
<th>Language-Dependent Attributes</th>
<th>Definition Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHA-CPU OF CPU-PIN-REDEF</td>
<td>05 AN GROUP WSS SIZE=2 OFFSET=0</td>
<td>COBEX[1] 61</td>
</tr>
<tr>
<td>ALPHA-PIN OF CPU-PIN-REDEF</td>
<td>05 AN GROUP WSS SIZE=2 OFFSET=2</td>
<td>COBEX[1] 66</td>
</tr>
<tr>
<td>CAPTURE-CPU-PIN OF AA</td>
<td>PAR</td>
<td>COBEX[1] 200</td>
</tr>
<tr>
<td>CPU-HIGH-BYTE OF ALPHA-CPU OF CPU-PIN-REDEF</td>
<td>10 AN DISP WSS SIZE=1 OFFSET=0</td>
<td>COBEX[1] 62</td>
</tr>
<tr>
<td>CPU-LOW-BYTE OF ALPHA-CPU OF CPU-PIN-REDEF</td>
<td>10 AN DISP WSS SIZE=1 OFFSET=1</td>
<td>COBEX[1] 63</td>
</tr>
<tr>
<td>CPU-PART OF CPU-PIN OF PROCESS-ID OF MESSAGE-SOURCE-REC</td>
<td>15 AN DISP WSS SIZE=1 OFFSET=14</td>
<td>COBEX[1] 55</td>
</tr>
<tr>
<td>DISPLAY-STARTUP-FAILURE OF AA</td>
<td>PAR</td>
<td>COBEX[1] 158</td>
</tr>
<tr>
<td>FUP OF SAVE-MESSAGE-STUFF</td>
<td>05 AN DISP WSS SIZE=21 OFFSET=0</td>
<td>COBEX[1] 78</td>
</tr>
<tr>
<td>FUP-FAILED OF SAVE-MESSAGE-STUFF</td>
<td>05 AN DISP WSS SIZE=19 OFFSET=44</td>
<td>COBEX[1] 94</td>
</tr>
<tr>
<td>FUP-NAME OF SAVE-MESSAGE-STUFF</td>
<td>05 AN DISP WSS SIZE=5 OFFSET=21</td>
<td>COBEX[1] 80</td>
</tr>
<tr>
<td>INFO-COMMAND OF SAVE-MESSAGE-STUFF</td>
<td>05 AN DISP WSS SIZE=7 OFFSET=70</td>
<td>COBEX[1] 98</td>
</tr>
<tr>
<td>MEMORY-AS-USUAL OF SAVE-MESSAGE-STUFF</td>
<td>05 NM COMP WSS SIZE=2 OFFSET=32</td>
<td>COBEX[1] 88</td>
</tr>
<tr>
<td>MESSAGE-IN-FILE</td>
<td>FD IS $RECEIVE RECORD=1032 F ORG=SEQ ACC=SEQ</td>
<td>COBEX[1] 36</td>
</tr>
<tr>
<td>MESSAGE-SOURCE-REC</td>
<td>01 AN GROUP WSS SIZE=32 OFFSET=0</td>
<td>COBEX[1] 48</td>
</tr>
</tbody>
</table>

Figure 6-3. CROSSREF Listing--Identifier List (Page 1 of 3)
Figure 6-3. CROSSREF Listing--Identifier List (Page 2 of 3)
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT-1 OF RECEIVE-FILE-STATUS OF FILE-DATA</td>
<td>10 NM DISP WSS SIZE=1 OFFSET=0</td>
<td>COBEX[1] 74</td>
</tr>
<tr>
<td>STRING-PORTION OF SAVE-MESSAGE-STUFF</td>
<td>05 AN DISP WSS SIZE=7 OFFSET=63</td>
<td>COBEX[1] 96</td>
</tr>
<tr>
<td>SU-ERROR OF SAVE-MESSAGE-STUFF</td>
<td>05 NM COMP WSS SIZE=2 OFFSET=34</td>
<td>COBEX[1] 90</td>
</tr>
<tr>
<td>SYS-MSG-ABEND OF SYS-MSG-CODE OF MESSAGE-IN OF MESSAGE-IN-FILE</td>
<td>88 NM COMP FS SIZE=2 OFFSET=0</td>
<td>COBEX[1] 42</td>
</tr>
<tr>
<td>WATCH OF AA PARA</td>
<td>COBEX[1] 147 M</td>
<td></td>
</tr>
</tbody>
</table>

**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

\[ \text{ln category usage loc SIZE=\text{size} [V] OFFSET=\text{offset} [i \text{ INX[S]}]} \]  
\[ \text{[SPCL-REG]} \]

\text{ln} is the level number of the data-item.

\text{category} is one of the following:
- \text{AL} Alphabetic
- \text{ALE} Alphabetic with Bs in its PICTURE
- \text{AN} Alphanumeric
- \text{ANE} Alphanumeric edited
- \text{NM} Numeric
- \text{NME} Numeric edited

\text{usage} is one of the following:
- \text{GROUP} Group item
- \text{DISP} DISPLAY item
- \text{COMP} COMPUTATIONAL item
- \text{INX} INDEX item

\text{loc} shows the site of the identifier declaration and is one of the following:
- \text{ESS} Extended-Storage Section
- \text{FS} File Section
- \text{WSS} Working-Storage Section
- \text{LS} Linkage Section

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

\text{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 INXS 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.
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

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

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)
&SET LANGUAGE cobol85
&SET LIBRARY mylib
&SET DIRECTIVES "ANSI;SETTOG 1"
&SCAN bprog
&GENERATE /OUT $s.#cros/
&EXIT
13>
```
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)
  &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.
IDENTIFICATION DIVISION.
PROGRAM-ID.  FUPPERWARE.
*  AUTHOR.  ANN COBOL.
*  INSTALLATION.  TRANSACTIONS ANONYMOUS.
*  DATE-Compiled.

*******************************************************************************

*  
*  This program creates a FUP process and watches for its 
*  termination. 
*  
*******************************************************************************

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  TANDEM TXP.
OBJECT-COMPUTER.  TANDEM TXP.
SPECIAL-NAMES.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT MESSAGE-IN-FILE
ASSIGN TO "$RECEIVE"
FILE STATUS IS RECEIVE-FILE-STATUS.
I-O-CONTROL.
RECEIVE-CONTROL.
TABLE OCCURS 1 TIMES
SYNCDEPTH LIMIT IS 1
REPLY CONTAINS 32 CHARACTERS
MESSAGE SOURCE IS MESSAGE-SOURCE-REC
REPORT SYSTEM MESSAGES.

DATA DIVISION.
FILE SECTION.
FD MESSAGE-IN-FILE
LABEL RECORDS ARE OMITTED.
01 MESSAGE-IN.
05 SYS-MSG-CODE PIC S9(4) COMP.
88 SYS-MSG-STOP VALUE -5.
88 SYS-MSG-ABEND VALUE -6.
05 SYS-MSG-PROCNAME PIC X(6).
05 FILLER PIC X(1024).

WORKING-STORAGE SECTION.
01 MESSAGE-SOURCE-REC.
05 SYSTEM-FLAG PIC S9 COMP.
05 ENTRY-NUMBER PIC 999 COMP.
05 FILLER PIC X(4).
05 PROCESS-ID.
10 PROCESS-NAME PIC X(6).
10 CPU-PIN.
15 CPU-PART PIC X.
15 PIN-PART PIC X.
05 FILLER PIC X(16).
01 CPU-PIN-REDEF.
05 ALPHA-CPU.
10 CPU-HIGH-BYTE PIC X.
10 CPU-LOW-BYTE PIC X.
/**
 * Sample COBOL85 program listing.
 *
 * 05 NUMERIC-CPU
 *    05 ALPHA-PIN.
 *        10 PIN-HIGH-BYTE
 *        10 PIN-LOW-BYTE
 *    05 NUMERIC-Pin
 *        05 ALPHA-PIN
 *
 * 01 FILE-DATA.
 *    05 RECEIVE-FILE-STATUS.
 *        10 STAT-1
 *        88 CLOSE-FROM-REQUESTER VALUE 1 THRU 3.
 *        10 STAT-2
 *
 * 01 SAVE-MESSAGE-STUFF.
 *    05 FUP
 *        VALUE "$SYSTEM.SYSTEM.FUP".
 *    05 FUP-NAME
 *        VALUE SPACES.
 *    05 SEND-ALL-MSGS
 *        VALUE ZERO COMP.
 *    05 PRIORITY-EQ-MINE
 *        VALUE 0 COMP.
 *    05 PROCESSOR-EQ-MINE
 *        VALUE -1 COMP.
 *    05 MEMORY-AS-USUAL
 *        VALUE ZERO COMP.
 *    05 SU-ERROR
 *        VALUE ZERO COMP.
 *    05 NEWPROCESS-ERR-LEFT
 *    05 NEWPROCESS-ERR-RIGHT
 *    05 FUP-FAILED
 *        VALUE "Failed to start FUP".
 *    05 STRING-PORTION
 *        VALUE "STRING".
 *    05 INFO-COMMAND
 *        VALUE "INFO *".
 * 05 STARTUP-RESULT
 *    05 NULL-CPLIST
 *
 *
 * PROCEDURE DIVISION.
 * DECLARATIVES.
 * HANDLE-INFILE-ERRORS SECTION.
 * USE AFTER STANDARD ERROR PROCEDURE ON MESSAGE-IN-FILE.
 * INFILE-ERROR.
 * IF STAT-1 = 1
 *    DISPLAY "EOF on $RECEIVE"
 * ELSE
 *    DISPLAY "RECEIVE FILE ERROR STATUS = "
 *    RECEIVE-FILE-STATUS
 * END-IF
 */

Figure 7-1. COBOL85 Sample Program (Page 2 of 4)
OPEN INPUT MESSAGE-IN-FILE
MOVE ZERO TO SU-ERROR
SYS-MSG-CODE

* Inject INFO command into STARTUP message to pass to FUP
ENTER "PUTSTARTUPTEXT"
USING STRING-PORTION,
INFO-COMMAND,
NULL-CPLIST
GIVING STARTUP-RESULT

* Start FUP
ENTER "CREATEPROCESS"
USING FUP,
FUP-NAME,
SEND-ALL-MSGS,
PRIORITY-EQ-MINE,
PROCESSOR-EQ-MINE,
MEMORY-AS-USUAL,
OMITTED
GIVING SU-ERROR

* Await termination of FUP, or report it never started
IF SU-ERROR = 0
PERFORM UNTIL SYS-MSG-STOP OR SYS-MSG-ABEND
READ MESSAGE-IN-FILE
PERFORM CAPTURE-CPU-PIN
DISPLAY "-----"
"(" NUMERIC-CPU "," NUMERIC-PIN ")"
"-----"
SYS-MSG-CODE
END-PERFORM
ELSE
PERFORM DISPLAY-STARTUP-FAILURE
END-IF

STOP RUN
.

DISPLAY-STARTUP-FAILURE.
EVALUATE SU-ERROR
WHEN 1
DISPLAY FUP-FAILED
" -- REQUIRED PARAMETER MISSING OR ILLEGAL"
WHEN 2
DISPLAY FUP-FAILED
" -- ILLEGAL PROGRAM FILE NAME (" FUP ")"
WHEN 3
DISPLAY FUP-FAILED
" -- INFILE, OUTFILE, OR DEFAULT VOLUME"
DISPLAY " NAME CANNOT BE CONVERTED TO NETWORK FORM"
WHEN 4 THRU 255
DISPLAY FUP-FAILED
" -- File management error #"
SU-ERROR
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.

<table>
<thead>
<tr>
<th>FILE NO.</th>
<th>FILE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>$EM2.UCREF.COBE85</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHA-CPU OF CPU-PIN-REDEF</td>
<td>05 AN GROUP SIZE=2 OFFSET=0</td>
<td>COBEX85[1] 61</td>
</tr>
<tr>
<td>ALPHA-PIN OF CPU-PIN-REDEF</td>
<td>05 AN GROUP SIZE=2 OFFSET=2</td>
<td>COBEX85[1] 66</td>
</tr>
<tr>
<td>CAPTURE-CPU-PIN OF AA</td>
<td>PARAGRAPH</td>
<td>COBEX85[1] 200</td>
</tr>
<tr>
<td>CPU-HIGH-BYTE OF ALPHA-CPU OF CPU-PIN-REDEF</td>
<td>10 AN DISPLY SIZE=1 OFFSET=0</td>
<td>COBEX85[1] 62</td>
</tr>
<tr>
<td>CPU-LOW-BYTE OF ALPHA-CPU OF CPU-PIN-REDEF</td>
<td>10 AN DISPLY SIZE=1 OFFSET=1</td>
<td>COBEX85[1] 63</td>
</tr>
<tr>
<td>CPU-PART OF CPU-PIN OF PROCESS-ID OF MESSAGE-SOURCE-REC</td>
<td>15 AN DISPLY SIZE=1 OFFSET=14</td>
<td>COBEX85[1] 55</td>
</tr>
<tr>
<td>DISPLAY-STARTUP-FAILURE OF AA</td>
<td>PARAGRAPH</td>
<td>COBEX85[1] 158</td>
</tr>
<tr>
<td>FUP OF SAVE-MESSAGE-STUFF</td>
<td>05 AN DISPLY SIZE=21 OFFSET=0</td>
<td>COBEX85[1] 78</td>
</tr>
<tr>
<td>FUP-FAILED OF SAVE-MESSAGE-STUFF</td>
<td>05 AN DISPLY SIZE=19 OFFSET=44</td>
<td>COBEX85[1] 94</td>
</tr>
<tr>
<td>FUP-NAME OF SAVE-MESSAGE-STUFF</td>
<td>05 AN DISPLY SIZE=5 OFFSET=21</td>
<td>COBEX85[1] 80</td>
</tr>
<tr>
<td>INFO-COMMAND OF SAVE-MESSAGE-STUFF</td>
<td></td>
<td>COBEX85[1]</td>
</tr>
<tr>
<td>MEMORY-AS-USUAL OF SAVE-MESSAGE-STUFF</td>
<td>05 NM COMP SIZE=2 OFFSET=32</td>
<td>COBEX85[1] 141</td>
</tr>
<tr>
<td>MESSAGE-IN-FILE</td>
<td>FILE (FD) IS $RECEIVE</td>
<td>COBEX85[1]</td>
</tr>
<tr>
<td>MESSAGE-SOURCE-REC</td>
<td>01 AN GROUP SIZE=32 OFFSET=0</td>
<td>COBEX85[1]</td>
</tr>
</tbody>
</table>

Figure 7-3. CROSSREF Listing--Identifier List (Page 1 of 3)
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEWPROCESS-ERR-LEFT OF SAVE-MESSAGE-STUFF</td>
<td>05 NM DISPLY SIZE=4 OFFSET=36</td>
<td>COBEX85[1] 92</td>
</tr>
<tr>
<td>NEWPROCESS-ERR-RIGHT OF SAVE-MESSAGE-STUFF</td>
<td>05 NM DISPLY SIZE=4 OFFSET=40</td>
<td>COBEX85[1] 93</td>
</tr>
<tr>
<td>NULL-CPLIST OF SAVE-MESSAGE-STUFF</td>
<td>05 NM COMP SIZE=4 OFFSET=80</td>
<td>COBEX85[1] 102</td>
</tr>
<tr>
<td>NUMERIC-CPU OF CPU-PIN-REDEF</td>
<td>05 NM COMP SIZE=2 OFFSET=0</td>
<td>COBEX85[1] 64</td>
</tr>
<tr>
<td>NUMERIC-PIN OF CPU-PIN-REDEF</td>
<td>05 NM COMP SIZE=2 OFFSET=2</td>
<td>COBEX85[1] 69</td>
</tr>
<tr>
<td>PIN-HIGH-BYTE OF ALPHA-PIN OF CPU-PIN-REDEF</td>
<td>10 AN DISPLY SIZE=1 OFFSET=2</td>
<td>COBEX85[1] 67</td>
</tr>
<tr>
<td>PIN-LOW-BYTE OF ALPHA-PIN OF CPU-PIN-REDEF</td>
<td>10 AN DISPLY SIZE=1 OFFSET=3</td>
<td>COBEX85[1] 68</td>
</tr>
<tr>
<td>PIN-PART OF CPU-PIN OF PROCESS-ID OF MESSAGE-SOURCE-REC</td>
<td>15 AN DISPLY SIZE=1 OFFSET=15</td>
<td>COBEX85[1] 56</td>
</tr>
<tr>
<td>PRIORITY-EQ-MINE OF SAVE-MESSAGE-STUFF</td>
<td>05 NM COMP SIZE=2 OFFSET=28</td>
<td>COBEX85[1] 84</td>
</tr>
<tr>
<td>PROCESSOR-EQ-MINE OF SAVE-MESSAGE-STUFF</td>
<td>05 NM COMP SIZE=2 OFFSET=30</td>
<td>COBEX85[1] 86</td>
</tr>
<tr>
<td>RECEIVE-FILE-STATUS OF FILE-DATA</td>
<td>05 AN GROUP SIZE=2 OFFSET=0</td>
<td>COBEX85[1] 73</td>
</tr>
<tr>
<td>SEND-ALL-MSGS OF SAVE-MESSAGE-STUFF</td>
<td>05 NM COMP SIZE=2 OFFSET=26</td>
<td>COBEX85[1] 82</td>
</tr>
<tr>
<td>STARTUP-RESULT OF SAVE-MESSAGE-STUFF</td>
<td>05 NM COMP SIZE=2 OFFSET=78</td>
<td>COBEX85[1] 100</td>
</tr>
<tr>
<td>STAT-1 OF RECEIVE-FILE-STATUS OF FILE-DATA</td>
<td>10 NM DISPLY SIZE=1 OFFSET=0</td>
<td>COBEX85[1] 74</td>
</tr>
<tr>
<td>NAME AND NAME QUALIFIER</td>
<td>LANGUAGE-DEPENDENT ATTRIBUTES</td>
<td>DEFINITION POINT</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>COBEX85[1]</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>STRING-PORTION OF SAVE-MESSAGE-STUFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>05 AN DISPLY SIZE=7 OFFSET=63</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COBEX85[1]</td>
<td>129</td>
<td></td>
</tr>
<tr>
<td>SU-ERROR OF SAVE-MESSAGE-STUFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>05 NM COMP SIZE=2 OFFSET=34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COBEX85[1]</td>
<td>123 W</td>
<td>143 W</td>
</tr>
<tr>
<td>SYS-MSG-ABEND OF SYS-MSG-CODE OF MESSAGE-IN OF MESSAGE-IN-FILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>88 NM COMP SIZE=2 OFFSET=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COBEX85[1]</td>
<td>147</td>
<td></td>
</tr>
<tr>
<td>SYS-MSG-CODE OF MESSAGE-IN OF MESSAGE-IN-FILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>05 NM COMP SIZE=2 OFFSET=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COBEX85[1]</td>
<td>124 W</td>
<td>150.5</td>
</tr>
<tr>
<td>SYS-MSG-STOP OF SYS-MSG-CODE OF MESSAGE-IN OF MESSAGE-IN-FILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>88 NM COMP SIZE=2 OFFSET=0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COBEX85[1]</td>
<td>147</td>
<td></td>
</tr>
</tbody>
</table>

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 KEYS] { (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 KEYS 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.
COBOL85
External Switch

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

<table>
<thead>
<tr>
<th>CROSSREF Class</th>
<th>Default Setting</th>
<th>BASIC Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTIONS</td>
<td>ON</td>
<td>User-defined functions</td>
</tr>
<tr>
<td>KEYWORDS</td>
<td>OFF</td>
<td>Keywords (reserved words)</td>
</tr>
<tr>
<td>LINENOS</td>
<td>ON</td>
<td>Line numbers</td>
</tr>
<tr>
<td>LITERALS</td>
<td>OFF</td>
<td>Literals</td>
</tr>
<tr>
<td>SYSVARS</td>
<td>ON</td>
<td>System variables</td>
</tr>
<tr>
<td>VARIABLES</td>
<td>ON</td>
<td>Variables</td>
</tr>
</tbody>
</table>

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)
&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$, 4, 3)=currentmo$               &
240     THEN fnthismonth=1 ELSE fnthismonth=0
250 FNEND
300 ! Set up map and open account file.
310 MAP (accounts) id$=6, lname$=20, firname$=20, &
320     FILL$=60, balance, lasttrandate$=8
330 OPEN "accounts" as #1, ORGANIZATION INDEXED, &
340     MAP accounts, &
350     ACCESS READ, ALLOW READ
400 ! Print header for report.
410 PRINT USING 710 \ PRINT USING 720 \ PRINT USING 730
500 ! Read records and print lines of report.
510 ON ERROR GOTO 900
520 WHILE ERR = 0
530 GET #1
540 IF (fnthismonth(lasttrandate$))                  &
550     THEN PRINT USING 740, id$,                      &
560     TRM$(lname$) + "", " + firname$, &
570     balance, lasttrandate$ \&
580     total = total + balance
590 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 | 'LLLLLLLLLL | $###,###.##- | 'LLLLLLL
750 : \\ 
760 : TOTAL IS: $##,###,###.##- \\
900 ! Error processing code.
910 IF ERR=1 AND ERL=530 THEN RESUME 600 \&
920 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.

<table>
<thead>
<tr>
<th>FILE NO.</th>
<th>FILE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>$EM2.UCREF.BASEX</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>94 TOTAL SYMBOLS COLLECTED WITH 106 TOTAL REFERENCES COLLECTED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>NUMBERED BASIC LINE</td>
<td>BASEX[1] 600</td>
</tr>
<tr>
<td>710</td>
<td>NUMBERED BASIC LINE WITH PRINT IMAGE</td>
<td>BASEX[1] 710</td>
</tr>
<tr>
<td>720</td>
<td>NUMBERED BASIC LINE WITH PRINT IMAGE</td>
<td>BASEX[1] 720</td>
</tr>
<tr>
<td>730</td>
<td>NUMBERED BASIC LINE WITH PRINT IMAGE</td>
<td>BASEX[1] 730</td>
</tr>
<tr>
<td>740</td>
<td>NUMBERED BASIC LINE WITH PRINT IMAGE</td>
<td>BASEX[1] 740</td>
</tr>
<tr>
<td>750</td>
<td>NUMBERED BASIC LINE WITH PRINT IMAGE</td>
<td>BASEX[1] 750</td>
</tr>
<tr>
<td>760</td>
<td>NUMBERED BASIC LINE WITH PRINT IMAGE</td>
<td>BASEX[1] 760</td>
</tr>
<tr>
<td>900</td>
<td>NUMBERED BASIC LINE</td>
<td>BASEX[1] 900</td>
</tr>
<tr>
<td>BALANCE</td>
<td>FLOATING-POINT, REAL(64), VARIABLE WITHIN MAP</td>
<td>BASEX[1] 310</td>
</tr>
<tr>
<td>CURRENTMO$</td>
<td>STRING VARIABLE</td>
<td>BASEX[1] 210 W 230</td>
</tr>
<tr>
<td>ERL</td>
<td>SYSTEM VARIABLE WITH INTEGER VALUE</td>
<td>BASEX[1] 910</td>
</tr>
<tr>
<td>ERR</td>
<td>SYSTEM VARIABLE WITH INTEGER VALUE</td>
<td>BASEX[1] 520 910</td>
</tr>
<tr>
<td>FIRMNAME$</td>
<td>STRING VARIABLE WITHIN MAP, SIZE=20 BYTES</td>
<td>BASEX[1] 310</td>
</tr>
<tr>
<td>FNTHISMONTH</td>
<td>USER-DEFINED FUNCTION RETURNING FLOATING-POINT, REAL(64), VALUE</td>
<td>BASEX[1] 220</td>
</tr>
<tr>
<td>IDS</td>
<td>STRING VARIABLE WITHIN MAP, SIZE=6 BYTES</td>
<td>BASEX[1] 310 540.01</td>
</tr>
<tr>
<td>LASTTRANDATES</td>
<td>STRING VARIABLE WITHIN MAP, SIZE=8 BYTES</td>
<td>BASEX[1] 310</td>
</tr>
</tbody>
</table>

Figure 8-3. CROSSREF Listing--Identifier List (Page 1 of 2)
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASEX[1]</td>
<td></td>
<td>310 D 540.01</td>
</tr>
<tr>
<td>LNAME$</td>
<td>STRING VARIABLE WITHIN MAP, SIZE=20 BYTES</td>
<td>BASEX[1] 310</td>
</tr>
<tr>
<td>TDATE$</td>
<td>STRING VARIABLE</td>
<td>BASEX[1] 220 D 230</td>
</tr>
<tr>
<td>TOTAL</td>
<td>FLOATING-POINT, REAL(64), VARIABLE</td>
<td>BASEX[1] 540.02 540.02 W 620</td>
</tr>
</tbody>
</table>

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

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

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.
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)
   &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.
FORTRAN
Sample 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     901      FORMAT (F6.2)
24     902      FORMAT (1X, 'Data file has too many values.', /
25                   +        1X, 'Maximum number of values for sort: ', I6)
26           END
27
28           SUBROUTINE SORTNUMS (icount, values)
29
30     C     Sorts array (values) with icount elements in ascending order.
31
32           DIMENSION  values(icount)
33
34           DO 600  i = icount-1, 1, -1
35              DO 500 j = 1, i
36                   IF ( values(j) .LT. values(j+1) )  GOTO 500
37                   temp        = values(j)
38                   values(j)   = values(j+1)
39                   values(j+1) = temp
40      500     CONTINUE
41      600  CONTINUE
42           RETURN
43           END
44
45           SUBROUTINE PRTNUMS (message)
46
47     C     Prints message passed as parameter, numbers from common block.
48
49           CHARACTER  message*(*)
50           INTEGER   count, maxcount
51           REAL      numbers
52           PARAMETER (maxcount=50)
53           COMMON    count, numbers(maxcount)
54
55           WRITE (UNIT=4,FMT=100)  message
56           WRITE (UNIT=4,FMT=200)  (numbers(i), i = 1, count)
57           RETURN
58     100      FORMAT(/,1X,A,/)  
59     200      FORMAT(1X,F6.2)
60           END
61

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.

<table>
<thead>
<tr>
<th>FILE NO.</th>
<th>FILE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>$EM2.UCREF.FORTEX</td>
</tr>
</tbody>
</table>

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).
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FORTEX[1]</td>
</tr>
<tr>
<td>34 TOTAL SYMBOLS COLLECTED WITH 78 TOTAL REFERENCES COLLECTED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>PROGRAM LABEL</td>
<td>FORTEX[1] 13</td>
</tr>
<tr>
<td>100</td>
<td>FORMAT LABEL</td>
<td>FORTEX[1] 60</td>
</tr>
<tr>
<td>200</td>
<td>PROGRAM LABEL</td>
<td>FORTEX[1] 18</td>
</tr>
<tr>
<td>200</td>
<td>FORMAT LABEL</td>
<td>FORTEX[1] 61</td>
</tr>
<tr>
<td>500</td>
<td>PROGRAM LABEL</td>
<td>FORTEX[1] 42</td>
</tr>
<tr>
<td>600</td>
<td>PROGRAM LABEL</td>
<td>FORTEX[1] 43</td>
</tr>
<tr>
<td>901</td>
<td>FORMAT LABEL</td>
<td>FORTEX[1] 25</td>
</tr>
<tr>
<td>902</td>
<td>FORMAT LABEL</td>
<td>FORTEX[1] 26</td>
</tr>
<tr>
<td>COUNT</td>
<td>INTEGER<em>2 VARIABLE, IN /BLANK</em>/</td>
<td>FORTEX[1] 5</td>
</tr>
<tr>
<td>COUNT</td>
<td></td>
<td>FORTEX[1] 52</td>
</tr>
<tr>
<td>DUMMY</td>
<td>REAL VARIABLE</td>
<td>FORTEX[1] 15</td>
</tr>
<tr>
<td>I</td>
<td>INTEGER*2 VARIABLE</td>
<td>FORTEX[1] 36</td>
</tr>
<tr>
<td>I</td>
<td>INTEGER*2 VARIABLE</td>
<td>FORTEX[1] 58</td>
</tr>
<tr>
<td>ICOUNT</td>
<td>INTEGER*2 DUMMY VARIABLE</td>
<td>FORTEX[1] 30</td>
</tr>
<tr>
<td>J</td>
<td>INTEGER*2 VARIABLE</td>
<td>FORTEX[1] 37</td>
</tr>
<tr>
<td>MAXCOUNT</td>
<td>INTEGER*2 PARAMETER:</td>
<td>FORTEX[1] 5</td>
</tr>
</tbody>
</table>

Figure 9-3. CROSSREF Listing--Identifier List (Page 1 of 2)
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXCOUNT</td>
<td>INTEGER*2 PARAMETER: 50</td>
<td>FORTEX[1] 52</td>
</tr>
<tr>
<td></td>
<td>FORTEX[1] 52 D 54 D 55</td>
<td></td>
</tr>
<tr>
<td>MESSAGE</td>
<td>CHARACTER*(*) DUMMY VARIABLE</td>
<td>FORTEX[1] 47</td>
</tr>
<tr>
<td></td>
<td>FORTEX[1] 47 D 51 D 57</td>
<td></td>
</tr>
<tr>
<td>NUMBERS</td>
<td>ARRAY(1:50) OF REAL, IN /BLANK^/</td>
<td>FORTEX[1] 6</td>
</tr>
<tr>
<td></td>
<td>FORTEX[1] 6 D 8 D 12 W 21 P</td>
<td></td>
</tr>
<tr>
<td>NUMBERS</td>
<td>ARRAY(1:50) OF REAL, IN /BLANK^/</td>
<td>FORTEX[1] 53</td>
</tr>
<tr>
<td></td>
<td>FORTEX[1] 53 D 55 D 58</td>
<td></td>
</tr>
<tr>
<td>PRTNUMS</td>
<td>SUBROUTINE</td>
<td>FORTEX[1] 47</td>
</tr>
<tr>
<td></td>
<td>FORTEX[1] 20 I 22 I 47 D</td>
<td></td>
</tr>
<tr>
<td>SORTNUMS</td>
<td>SUBROUTINE</td>
<td>FORTEX[1] 30</td>
</tr>
<tr>
<td></td>
<td>FORTEX[1] 21 I 30 D</td>
<td></td>
</tr>
<tr>
<td>TEMP</td>
<td>REAL VARIABLE</td>
<td>FORTEX[1] 39</td>
</tr>
<tr>
<td></td>
<td>FORTEX[1] 39 W 41</td>
<td></td>
</tr>
<tr>
<td>VALUES</td>
<td>DUMMY ARRAY(1:?) OF REAL</td>
<td>FORTEX[1] 30</td>
</tr>
<tr>
<td></td>
<td>FORTEX[1] 30 D 34 D 38 39 40 W 40 41 W</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>CROSSREF Class</th>
<th>Default Setting</th>
<th>Pascal Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCKS</td>
<td>ON</td>
<td>Public variables</td>
</tr>
<tr>
<td>CONSTANTS</td>
<td>ON</td>
<td>Named constants</td>
</tr>
<tr>
<td>PROCEDURES</td>
<td>ON</td>
<td>Procedures and functions</td>
</tr>
<tr>
<td>PROGLABELS</td>
<td>ON</td>
<td>GOTO labels</td>
</tr>
<tr>
<td>TYPES</td>
<td>ON</td>
<td>All data types</td>
</tr>
<tr>
<td>VARIABLES</td>
<td>ON</td>
<td>Non-public variables</td>
</tr>
</tbody>
</table>

**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)
&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.
(This program reads data base input from the terminal and enters it into a memory resident data base.)

PROGRAM CrossrefExample(INPUT,OUTPUT);

IMPORT BEGIN

TYPE
  Date = INTEGER;
  PlaceName = STRING[20];
  PersonName = STRING[20];
  PersonKind = (National, Alien);
  Level = 1..4;
END;

TYPE
  Person = RECORD
      Name: PersonName;
      DateOfBirth: Date;
      JobLevel: Level;
      NextRecord: ^Person;
      CASE Origin: PersonKind OF
          National: (BirthPlace: PlaceName);
          Alien: (CountryOfOrigin: PlaceName;
                 DateOfEntry: Date;
                 PortOfEntry: PlaceName);
      END;
  END;

PtrToPerson = ^Person;

VAR
  Name :PersonName;
  JobLevel: Level;
  BirthDate :Date;
  BirthPlace :PlaceName;
  CountryOfOrigin :PlaceName;
  DateOfEntry :Date;
  PortOfEntry :PlaceName;
  PersonRecPointer :PtrToPerson;
  TempRecPointer :PtrToPerson;

PROCEDURE MakeThePerson(VAR ThePerson :PtrToPerson;
                         Name :PersonName;
                         JobLevel :Level;
                         BirthDate :Date;
                         BirthPlace :PlaceName;
                         CountryOfOrigin :PlaceName;
                         DateOfEntry :Date;
                         PortOfEntry :PlaceName );
BEGIN
  ThePerson^.Name := Name;
  ThePerson^.DateOfBirth := BirthDate;
  ThePerson^.JobLevel := JobLevel;
  CASE ThePerson^.Origin OF
    National: ThePerson^.BirthPlace := BirthPlace;
    Alien: ThePerson^.DateOfEntry := DateOfEntry;
    PortOfEntry := PortOfEntry;
  END;
END;

Figure 10-1. Pascal Sample Program (Page 1 of 2)
Alien: BEGIN
  ThePerson^.DateOfEntry := DateOfEntry;
  ThePerson^.PortOfEntry := PortOfEntry;
  END; {CASE}
END; {MakePerson}
BEGIN
NEW(PersonRecPointer);
WHILE NOT EOF(INPUT) DO BEGIN
  WRITELN( 'Name:' );           READLN( Name);
  WRITELN( 'JobLevel: ' );       READLN( JobLevel);
  WRITELN( 'BirthDate:' );       READLN( BirthDate);
  WRITELN( 'BirthPlace:');      READLN( BirthPlace);
  WRITELN( 'CountryOfOrigin:'); READLN( CountryOfOrigin);
  WRITELN( 'DateOfEntry:');     READLN( DateOfEntry);
  WRITELN( 'PortOfEntry:');     READLN( PortOfEntry);
  MakeThePerson(PersonRecPointer, Name, JobLevel, BirthDate, BirthPlace, CountryOfOrigin, DateOfEntry, PortOfEntry );
  TempRecPointer := PersonRecPointer;
  NEW(PersonRecPointer);
  TempRecPointer^.NextRecord := PersonRecPointer;
END; {WHILE}
END.
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.

<table>
<thead>
<tr>
<th>FILE NO.</th>
<th>FILE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>$EM2.UCREF.PASCALEX</td>
</tr>
</tbody>
</table>

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).
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIEN PUBLIC</td>
<td>ENUMERATION CONST</td>
<td>PASCALEX[1] 10</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1] 10 D 32</td>
<td>72</td>
</tr>
<tr>
<td>BIRTHDATE OF ROUTINE</td>
<td>INT16 PARAM</td>
<td>PASCALEX[1] 59</td>
</tr>
<tr>
<td>MAKETHEPERSON</td>
<td>PASCALEX[1] 59 D 68</td>
<td></td>
</tr>
<tr>
<td>BIRTHDATE OF ROUTINE</td>
<td>INT16 VAR</td>
<td>PASCALEX[1] 46</td>
</tr>
<tr>
<td>CROSSREFEXAMPLE</td>
<td>INT16 PARAM</td>
<td>PASCALEX[1] 59</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1] 46 D 90 W 98</td>
<td></td>
</tr>
<tr>
<td>BIRTHPLACE OF RECORD</td>
<td>PLACENAME FIELD</td>
<td>PASCALEX[1] 31</td>
</tr>
<tr>
<td>PERSON</td>
<td>PASCALEX[1] 31 D 71 W</td>
<td></td>
</tr>
<tr>
<td>BIRTHPLACE OF ROUTINE</td>
<td>PLACENAME PARAM</td>
<td>PASCALEX[1] 60</td>
</tr>
<tr>
<td>MAKETHEPERSON</td>
<td>PASCALEX[1] 60 D 71</td>
<td></td>
</tr>
<tr>
<td>BIRTHPLACE OF ROUTINE</td>
<td>PLACENAME VAR</td>
<td>PASCALEX[1] 47</td>
</tr>
<tr>
<td>CROSSREFEXAMPLE</td>
<td>PLACENAME VAR</td>
<td>PASCALEX[1] 47</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1] 47 D 91 W 98</td>
<td></td>
</tr>
<tr>
<td>COUNTRYOFORIGIN OF</td>
<td>PLACENAME FIELD</td>
<td>PASCALEX[1] 32</td>
</tr>
<tr>
<td>RECORD PERSON</td>
<td>PASCALEX[1] 32 D 73 W</td>
<td></td>
</tr>
<tr>
<td>COUNTRYOFORIGIN OF</td>
<td>PLACENAME PARAM</td>
<td>PASCALEX[1] 61</td>
</tr>
<tr>
<td>ROUTINE MAKETHEPERSON</td>
<td>PASCALEX[1] 61 D 73</td>
<td></td>
</tr>
<tr>
<td>COUNTRYOFORIGIN OF</td>
<td>PLACENAME VAR</td>
<td>PASCALEX[1] 48</td>
</tr>
<tr>
<td>ROUTINE CROSSREFEXAMPLE</td>
<td>PLACENAME VAR</td>
<td>PASCALEX[1] 48</td>
</tr>
<tr>
<td>DATE PUBLIC</td>
<td>INT16 TYPE</td>
<td>PASCALEX[1] 7</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1] 7 D 27 M 33 M 46 M 49 M 59 M 62 M</td>
<td></td>
</tr>
<tr>
<td>DATEOFBIRTH OF RECORD</td>
<td>INT16 FIELD</td>
<td>PASCALEX[1] 27</td>
</tr>
<tr>
<td>PERSON</td>
<td>PASCALEX[1] 27 D 68 W</td>
<td></td>
</tr>
<tr>
<td>DATEOFENTRY OF RECORD</td>
<td>INT16 FIELD</td>
<td>PASCALEX[1] 33</td>
</tr>
<tr>
<td>PERSON</td>
<td>PASCALEX[1] 33 D 74 W</td>
<td></td>
</tr>
<tr>
<td>DATEOFENTRY OF ROUTINE</td>
<td>INT16 PARAM</td>
<td>PASCALEX[1] 62</td>
</tr>
<tr>
<td>MAKETHEPERSON</td>
<td>PASCALEX[1] 62 D 74</td>
<td></td>
</tr>
<tr>
<td>DATEOFENTRY OF ROUTINE</td>
<td>PLACENAME FIELD</td>
<td>PASCALEX[1] 49</td>
</tr>
<tr>
<td>CROSSREFEXAMPLE</td>
<td>PLACENAME FIELD</td>
<td>PASCALEX[1] 49</td>
</tr>
</tbody>
</table>

Figure 10-3. CROSSREF Listing--Identifier List (Page 1 of 3)
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT16 VAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 49 D 93 W 99</td>
<td>EOF PREDEFINED ROUTINE</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 86 I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INPUT PUBLIC PASCALEX[1] 3 D 86 W 88 W 89 W 90 W 91 W 92 W 93 W 94 W</td>
<td>INTEGER PREDEFINED INT16 TYPE</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 7 M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JOBLEVEL OF RECORD PERSON PASCALEX[1] 28 D 69</td>
<td>INTEGER PREDEFINED INT16 TYPE</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 28 D 69 W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JOBLEVEL OF ROUTINE MAKE THE PERSON PASCALEX[1] 58 D 69</td>
<td>LEVEL PARAM</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 58 D 69 W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JOBLEVEL OF ROUTINE CROSSREF EXAMPLE PASCALEX[1] 45 D 89 W 98</td>
<td>LEVEL VAR</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 45 D 89 W 98</td>
<td>LEVEL PUBLIC SUBRANGE TYPE</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 11 D 28 M 45 M 58 M</td>
<td>LEVEL PUBLIC SUBRANGE TYPE</td>
<td></td>
</tr>
<tr>
<td>MAKE THE PERSON OF ROUTINE CROSSREF EXAMPLE PASCALEX[1] 56 D 98</td>
<td>ROUTINE</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 56 D 98 I</td>
<td>NAME OF RECORD PERSON PLACENAME FIELD</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 26 D 67 W</td>
<td>NAME OF ROUTINE MAKE THE PERSON PLACENAME PARAM</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 57 D 67</td>
<td>NAME OF ROUTINE CROSSREF EXAMPLE PLACENAME VAR</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 44 D 88 W 98</td>
<td>NATIONAL PUBLIC ENUMERATION CONST</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 10 D 31 71</td>
<td>NEW PREDEFINED ROUTINE</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 84 I 102 I</td>
<td>NEXT RECORD OF RECORD PERSON POINTER FIELD</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 29 D 103 W</td>
<td>ORIGIN OF RECORD CROSSREF EXAMPLE PERSON KIND FIELD</td>
<td></td>
</tr>
<tr>
<td>PASCALEX[1] 30 D 70</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 10-3. CROSSREF Listing--Identifier List (Page 2 of 3)
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT PUBLIC</td>
<td>TEXT VAR</td>
<td>PASCALEX[1] 3</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>3 D 88 W 89 W 90 W 91 W 92 W 93 W 94 W</td>
</tr>
<tr>
<td>PERSON</td>
<td>RECORD TYPE</td>
<td>PASCALEX[1] 25</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>25 D 29 M 37 M</td>
</tr>
<tr>
<td>PERSONKIND PUBLIC</td>
<td>ENUMERATION TYPE</td>
<td>PASCALEX[1] 10</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>10 D 30 M</td>
</tr>
<tr>
<td>PERSONNAME PUBLIC</td>
<td>PLACENAME TYPE</td>
<td>PASCALEX[1] 9</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>9 D 26 M 44 M 57 M</td>
</tr>
<tr>
<td>PERSONRECPOINTER</td>
<td>PTRTOPERSON VAR</td>
<td>PASCALEX[1] 52</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>52 D 84 W 98 W 101 102 W 103</td>
</tr>
<tr>
<td>PLACENAME PUBLIC</td>
<td>STRING TYPE</td>
<td>PASCALEX[1] 8</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>8 D 31 M 32 M 34 M 47 M 48 M 50 M 60 M 61 M 63 M</td>
</tr>
<tr>
<td>PORTOFENTRY OF RECORD PERSON</td>
<td>PLACENAME FIELD</td>
<td>PASCALEX[1] 34</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>34 D 75 W</td>
</tr>
<tr>
<td>PORTOFENTRY OF ROUTINE MAKETHEPERSON</td>
<td>PLACENAME PARAM</td>
<td>PASCALEX[1] 63</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>63 D 75</td>
</tr>
<tr>
<td>PORTOFENTRY OF ROUTINE CROSSREFEXAMPLE</td>
<td>PLACENAME VAR</td>
<td>PASCALEX[1] 50</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>50 D 94 W 99</td>
</tr>
<tr>
<td>PTRTOPERSON OF ROUTINE CROSSREFEXAMPLE</td>
<td>POINTER TYPE</td>
<td>PASCALEX[1] 37</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>37 D 52 M 53 M 56 M</td>
</tr>
<tr>
<td>READLN PREDEFINED ROUTINE</td>
<td>PASCALEX[1]</td>
<td>88 I 89 I 90 I 91 I 92 I 93 I 94 I</td>
</tr>
<tr>
<td>TIMEPRECPINTER OF ROUTINE CROSSREFEXAMPLE</td>
<td>PTRTOPERSON VAR</td>
<td>PASCALEX[1] 53</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>53 D 101 W 103</td>
</tr>
<tr>
<td>THEPERSON OF ROUTINE MAKETHEPERSON</td>
<td>PTRTOPERSON PARAM</td>
<td>PASCALEX[1] 56</td>
</tr>
<tr>
<td></td>
<td>PASCALEX[1]</td>
<td>56 D 67 68 69 70 71 73 74 75</td>
</tr>
<tr>
<td>WRITELN PREDEFINED ROUTINE</td>
<td>PASCALEX[1]</td>
<td>88 I 89 I 90 I 91 I 92 I 93 I 94 I</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>CROSSREF Class</th>
<th>Default Setting</th>
<th>SCREEN COBOL Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDITIONS</td>
<td>ON</td>
<td>Condition names</td>
</tr>
<tr>
<td>LITERALS</td>
<td>OFF</td>
<td>Numeric and nonnumeric constants</td>
</tr>
<tr>
<td>MNEMONICS</td>
<td>ON</td>
<td>Mnemonic names, alphabet names</td>
</tr>
<tr>
<td>PROCEDURES</td>
<td>ON</td>
<td>PROGRAMS</td>
</tr>
<tr>
<td>PROGLABELS</td>
<td>ON</td>
<td>Labels, procedure names</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(paragraph names, section names)</td>
</tr>
<tr>
<td>SCREENS</td>
<td>ON</td>
<td>Screen names</td>
</tr>
<tr>
<td>VARIABLES</td>
<td>ON</td>
<td>Data names</td>
</tr>
</tbody>
</table>

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.
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)
   &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.
IDENTIFICATION DIVISION.
Program-id.
  ScdragnA.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.
T16.
OBJECT-COMPUTER.
T16,
  Terminal is T16-6530.
  * Character-set is US ASCII
  * Character-set is USASCII.
SPECIAL-NAMES.
  F1         is F1,
  F2         is F2,
  F3         is F3,
  F4         is F4,
  F5         is F5,
  F6         is F6,
  F7         is F7,
  F14        is F14
  SF16       is SF16,
  RETURN-KEY is RETURN-KEY.

 *---------------------------------------------------------------------
 *   ---->>>>>> This version for 6530 terminal.
 *   ---->>>>>> The terminal must have the appropriate language
 *   ---->>>>>>      installed and the RETURN key configured as a function
 *   ---->>>>>>      key.
 *---------------------------------------------------------------------

 DATA DIVISION.
 WORKING-STORAGE SECTION.
  01 Info1                           Pic x(65)           Value spaces.
  01 Info2                           Pic x(65)           Value spaces.
  01 Info3                           Pic x(65)           Value spaces.
  01 Info4                           Pic x(65)           Value spaces.
  01 Info5                           Pic x(65)           Value spaces.
  01 Err-msg                         Pic x(67)           Value spaces.
  01 END-OF-TEST                     Pic x(03)           Value "NO ".

 LINKAGE SECTION.
 SCREEN SECTION.
  01 DRAGON-SCREEN SIZE 24, 80 .
    05 FILLER AT 1, 18
    VALUE "Test of European National Use Characters" .
05 FILLER AT 3, 14
   VALUE "Press the appropriate function key to test the ".
05 FILLER AT 4, 14
   VALUE "character set desired. ".
05 FILLER AT 5, 14
   VALUE "The screens will contain all instructions necessary".
05 FILLER AT 6, 14
   VALUE "to run the tests".
05 L1 AT 8, 14
   PIC X(65)
   FROM INFO1 .
05 L2 AT 9, 14
   PIC X(65)
   FROM INFO2 .
05 L3 AT 10, 14
   PIC X(65)
   FROM INFO3 .
05 L4 AT 11, 14
   PIC X(65)
   FROM INFO4 .
05 L5 AT 12, 14
   PIC X(65)
   FROM INFO5 .
05 FILLER AT 14, 25
   VALUE "Character set selection keys".
05 FILLER AT 16, 22
   VALUE "F1 French (A2)      F2 French (QW)".
05 FILLER AT 17, 22
   VALUE "F3 German         F4 Spanish".
05 FILLER AT 18, 22
   VALUE "F5 United Kingdom F6 Swedish".
05 FILLER AT 19, 22
   VALUE "F7 Danish         F14 Recover ".
05 FILLER AT 20, 22
   VALUE "RETURN message   SF16 End Test ".
05 ERR-WINDOW AT 24, 4
   PIC X(67)
   FROM ERR-MSG
   ADVISORY .

PROCEDURE DIVISION.
DECLARATIVES.
RECOVER-SCRN SECTION.
USE FOR SCREEN RECOVERY.
DISPLAY DRAGON-SCREEN.
MOVE "SCREEN RECOVERY ACTIVATED" TO ERR-MSG.
DISPLAY TEMP ERR-WINDOW.
END DECLARATIVES.
MAIN-DRIVER.
MOVE space to info1.
MOVE "NO" TO END-OF-TEST.
PERSIST STEP1
UNTIL END-OF-TEST = "YES".
ALL-DONE.
EXIT PROGRAM.
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       / TEST1-ERROR.
191       Move "on error returned from French (A2) test" to infol.
192
193       TEST2-ERROR.
194       Move "on error returned from French (QW) test" to infol.
195
196       TEST3-ERROR.
197       Move "on error returned from German test" to infol.
198
199       TEST4-ERROR.
200       Move "on error returned from Spanish" to infol.
201
202       TEST5-ERROR.
203       Move "on error returned from United Kingdom test" to infol.
204
205       TEST6-ERROR.
206       Move "on error returned from Swedish/Finnish test" to infol.
207
208       TEST7-ERROR.
209       Move "on error returned from Danish/Norwegian test" to infol.

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.

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).
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAY-RETURN</td>
<td>PARA</td>
<td>SCOBEX[1] 186</td>
</tr>
<tr>
<td>DRAGON-SCREEN</td>
<td>01 SCREEN BASE</td>
<td>SCOBEX[1] 54</td>
</tr>
<tr>
<td>END-OF-TEST</td>
<td>01 AN DISP</td>
<td>SCOBEX[1] 48</td>
</tr>
<tr>
<td>ERR-MSG</td>
<td>01 AN DISP</td>
<td>SCOBEX[1] 47</td>
</tr>
<tr>
<td>ERR-WINDOW OF DRAGON-SCREEN</td>
<td>05 SCREEN FIELD</td>
<td>SCOBEX[1] 92</td>
</tr>
<tr>
<td>F1</td>
<td>MNEM FUNC-KEY</td>
<td>SCOBEX[1] 14</td>
</tr>
<tr>
<td>F14</td>
<td>MNEM FUNC-KEY</td>
<td>SCOBEX[1] 21</td>
</tr>
<tr>
<td>F2</td>
<td>MNEM FUNC-KEY</td>
<td>SCOBEX[1] 15</td>
</tr>
<tr>
<td>F3</td>
<td>MNEM FUNC-KEY</td>
<td>SCOBEX[1] 16</td>
</tr>
<tr>
<td>F4</td>
<td>MNEM FUNC-KEY</td>
<td>SCOBEX[1] 17</td>
</tr>
<tr>
<td>F5</td>
<td>MNEM FUNC-KEY</td>
<td>SCOBEX[1] 17</td>
</tr>
<tr>
<td>F6</td>
<td>MNEM FUNC-KEY</td>
<td>SCOBEX[1] 19</td>
</tr>
<tr>
<td>F7</td>
<td>MNEM FUNC-KEY</td>
<td>SCOBEX[1] 20</td>
</tr>
<tr>
<td>INFO1</td>
<td>01 AN DISP 107 W 128 W 192 W 195 W 198 W 201 W 204 W 207 W</td>
<td>SCOBEX[1] 42</td>
</tr>
<tr>
<td>INFO2</td>
<td>01 AN DISP</td>
<td>SCOBEX[1] 43</td>
</tr>
<tr>
<td>INFO3</td>
<td>01 AN DISP</td>
<td>SCOBEX[1] 44</td>
</tr>
</tbody>
</table>

Figure 11-3. CROSSREF Listing--Identifier List (Page 1 of 3)
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOBEX[1] 73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INFO4 SCOBEX[1] 76</td>
<td>01 AN DISP</td>
<td>SCOBEX[1] 45</td>
</tr>
<tr>
<td>INFO5 SCOBEX[1] 79</td>
<td>01 AN DISP</td>
<td>SCOBEX[1] 46</td>
</tr>
<tr>
<td>L1 OF DRAGON-SCREEN</td>
<td>05 SCREEN FIELD</td>
<td>SCOBEX[1] 65</td>
</tr>
<tr>
<td></td>
<td>SCOBEX[1] 118</td>
<td></td>
</tr>
<tr>
<td>L2 OF DRAGON-SCREEN</td>
<td>05 SCREEN FIELD</td>
<td>SCOBEX[1] 68</td>
</tr>
<tr>
<td></td>
<td>SCOBEX[1] 119</td>
<td></td>
</tr>
<tr>
<td>L3 OF DRAGON-SCREEN</td>
<td>05 SCREEN FIELD</td>
<td>SCOBEX[1] 71</td>
</tr>
<tr>
<td></td>
<td>SCOBEX[1] 120</td>
<td></td>
</tr>
<tr>
<td>L4 OF DRAGON-SCREEN</td>
<td>05 SCREEN FIELD</td>
<td>SCOBEX[1] 74</td>
</tr>
<tr>
<td></td>
<td>SCOBEX[1] 121</td>
<td></td>
</tr>
<tr>
<td>SCOBEX[1] 173 I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCOBEX[1] 155 I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCOBEX[1] 179 I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCOBEX[1] 149 I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCOBEX[1] 143 I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCOBEX[1] 161 I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCOBEX[1] 167 I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STEP1 PARA</td>
<td>SCOBEX[1] 115</td>
<td></td>
</tr>
</tbody>
</table>

Figure 11-3. CROSSREF Listing--Identifier List (Page 2 of 3)
<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOBEX[1]</td>
<td>109 M</td>
<td></td>
</tr>
<tr>
<td>TERMINATION-STATUS</td>
<td>01 NM COMP SPCL-REG</td>
<td>139</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>TEST1</td>
<td>PARA</td>
<td>SCOBEX[1] 142</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>130 M</td>
<td></td>
</tr>
<tr>
<td>TEST1-ERROR</td>
<td>PARA</td>
<td>SCOBEX[1] 191</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>145 M</td>
<td></td>
</tr>
<tr>
<td>TEST2</td>
<td>PARA</td>
<td>SCOBEX[1] 148</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>131 M</td>
<td></td>
</tr>
<tr>
<td>TEST2-ERROR</td>
<td>PARA</td>
<td>SCOBEX[1] 194</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>151 M</td>
<td></td>
</tr>
<tr>
<td>TEST3</td>
<td>PARA</td>
<td>SCOBEX[1] 154</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>132 M</td>
<td></td>
</tr>
<tr>
<td>TEST3-ERROR</td>
<td>PARA</td>
<td>SCOBEX[1] 197</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>157 M</td>
<td></td>
</tr>
<tr>
<td>TEST4</td>
<td>PARA</td>
<td>SCOBEX[1] 160</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>133 M</td>
<td></td>
</tr>
<tr>
<td>TEST4-ERROR</td>
<td>PARA</td>
<td>SCOBEX[1] 200</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>163 M</td>
<td></td>
</tr>
<tr>
<td>TEST5</td>
<td>PARA</td>
<td>SCOBEX[1] 166</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>134 M</td>
<td></td>
</tr>
<tr>
<td>TEST5-ERROR</td>
<td>PARA</td>
<td>SCOBEX[1] 203</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>169 M</td>
<td></td>
</tr>
<tr>
<td>TEST6</td>
<td>PARA</td>
<td>SCOBEX[1] 172</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>135 M</td>
<td></td>
</tr>
<tr>
<td>TEST6-ERROR</td>
<td>PARA</td>
<td>SCOBEX[1] 206</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>175 M</td>
<td></td>
</tr>
<tr>
<td>TEST7</td>
<td>PARA</td>
<td>SCOBEX[1] 178</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>136 M</td>
<td></td>
</tr>
<tr>
<td>TEST7-ERROR</td>
<td>PARA</td>
<td>SCOBEX[1] 209</td>
</tr>
<tr>
<td>SCOBEX[1]</td>
<td>181 M</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>CROSSREF Class</th>
<th>Default Setting</th>
<th>TAL Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCKDATAS</td>
<td>ON</td>
<td>Data BLOCKs</td>
</tr>
<tr>
<td>CONSTANTS</td>
<td>ON</td>
<td>LITERALs</td>
</tr>
<tr>
<td>MACROS</td>
<td>ON</td>
<td>DEFINEs</td>
</tr>
<tr>
<td>PROCEDURE PARAMS</td>
<td>ON</td>
<td>PROCs that are formal parameters</td>
</tr>
<tr>
<td>PROCEDURES</td>
<td>ON</td>
<td>PROCs</td>
</tr>
<tr>
<td>PROGLABELS</td>
<td>ON</td>
<td>LABELs</td>
</tr>
<tr>
<td>REGISTERS</td>
<td>ON</td>
<td>Index registers named in USE statements</td>
</tr>
<tr>
<td>SUBPROCS</td>
<td>ON</td>
<td>SUBPROCs</td>
</tr>
<tr>
<td>TYPES</td>
<td>ON</td>
<td>STRUCT templates</td>
</tr>
<tr>
<td>VARIABLES</td>
<td>ON</td>
<td>Variables including nontemplate STRUCTs</td>
</tr>
</tbody>
</table>
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)
   &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.
INT PROC Ascii(v,rjust,stg);

INT v; ! the integer value to convert
INT rjust; ! right justify result flag
STRING .stg; ! target STRING

BEGIN

STRING b[0:5] := [5*" ","0*"];

INT n; ! number of digits converted
INT sgn := 0; ! nonzero if 'v' is negative
INT k := 5; ! index for converted digit

IF v < 0 ! value is negative
  THEN
    BEGIN
      sgn := 1; ! set negative value flag
      v := -v; ! take absolute value
    END;

WHILE v ! while there is a value left....
  DO
    BEGIN
      b[k] := $udbl(v) ' \ 10 + "0"; ! convert a character
      v := v / 10; ! compute the remainder
      k := k - 1; ! count the converted character
    END;

IF sgn ! number is negative
  THEN
    BEGIN
      b[k] := "-"; ! insert the sign
      k := k - 1; ! count it as a character
    END;

IF NOT (n:=5-k) ! check for an overflow
  THEN
    n := 1; ! return one character in that case

IF rjust ! move the resultant string to the
  THEN ! user's target
    stg[n-1] '=': b[5] FOR n ! reverse move if right justified
  ELSE
    stg ':=' b[6-n] FOR n; ! otherwise forward move

RETURN n; ! return the string's length
END !ascii! ;
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.

<table>
<thead>
<tr>
<th>FILE NO.</th>
<th>FILE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>$EM2.UCREF.TALEX</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>NAME AND NAME QUALIFIER</th>
<th>LANGUAGE-DEPENDENT ATTRIBUTES</th>
<th>DEFINITION POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 TOTAL SYMBOLS COLLECTED WITH 34 TOTAL REFERENCES COLLECTED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B OF ASCII</td>
<td>STRING DIRECT VARIABLE</td>
<td>TALEX[1] 6</td>
</tr>
<tr>
<td></td>
<td>TALEX[1] 22 W 30 W 40 42</td>
<td></td>
</tr>
<tr>
<td>K OF ASCII</td>
<td>INT DIRECT VARIABLE</td>
<td>TALEX[1] 10</td>
</tr>
<tr>
<td></td>
<td>TALEX[1] 22 24 W 30 31 W 34</td>
<td></td>
</tr>
<tr>
<td>N OF ASCII</td>
<td>INT DIRECT VARIABLE</td>
<td>TALEX[1] 8</td>
</tr>
<tr>
<td></td>
<td>TALEX[1] 34 W 36 W 40 42 44</td>
<td></td>
</tr>
<tr>
<td>RJUST OF ASCII</td>
<td>INT DIRECT VARIABLE</td>
<td>TALEX[1] 3</td>
</tr>
<tr>
<td></td>
<td>TALEX[1] 1 D 38</td>
<td></td>
</tr>
<tr>
<td>SGN OF ASCII</td>
<td>INT DIRECT VARIABLE</td>
<td>TALEX[1] 9</td>
</tr>
<tr>
<td></td>
<td>TALEX[1] 15 W 27</td>
<td></td>
</tr>
<tr>
<td>STG OF ASCII</td>
<td>STRING INDIRECT VARIABLE</td>
<td>TALEX[1] 4</td>
</tr>
<tr>
<td></td>
<td>TALEX[1] 1 D 40 W 40 42 W 42</td>
<td></td>
</tr>
<tr>
<td>V OF ASCII</td>
<td>INT DIRECT VARIABLE</td>
<td>TALEX[1] 2</td>
</tr>
<tr>
<td></td>
<td>TALEX[1] 1 D 12 16 W 16 19 22 23 W 23</td>
<td></td>
</tr>
</tbody>
</table>

**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

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

_________________________________________________________________

ERRORS

ERRORS [ / OUT file-name / ]

_________________________________________________________________
EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT
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
SAVE

DEFINITIONS ONLY

DIRECTIVES

EXCLUDE

INCLUDE

LANGUAGE

LIBRARY

MEM

OMIT

PRIORITY

PROGRAM

UNREF

__________________________________________

SAVE

{ *                     }
SAVE {                  }
    { attribute-specification }

attribute-specification

    is one of the following:

    CLASS
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:
SYNTAX SUMMARY

SCAN

\[
\{ \{ \text{class-name} \} \ [ \text{ON} \ ] \}
\]

CLASS \{ \{ * \} \ [ \text{OFF} \ ] \}
\{ \text{class-list} \}

includes the following detailed parameters:

class-name

is one of the following:

\[
\begin{array}{lll}
\text{BLOCKDATAS} & \text{INLINES} & \text{PROGLABELS} \\
\text{BLOCKS} & \text{KEYWORDS} & \text{REGISTERS} \\
\text{CONDITIONS} & \text{LINENOS} & \text{SCREENS} \\
\text{CONSTANTS} & \text{LITERALS} & \text{SUBPROCS} \\
\text{FILES} & \text{MACROS} & \text{SYSVARS} \\
\text{FMTLABELS} & \text{MNEMONICS} & \text{TYPES} \\
\text{FUNCTIONS} & \text{PROCEDURE PARAMS} & \text{VARIABLES} \\
\text{INDEXES} & \text{PROCEDURES} & \\
\end{array}
\]

class-list

has the following syntax:

\[
\begin{array}{ll}
\{ \ [ \text{ON} \ ] \ [ \text{ON} \ ] \} \\
\{ \ ( \text{name} \ [ \ ] \ [ , \text{name} \ [ \ ] ] \} \} \\
\{ \ [ \text{OFF} \ ] \ [ \text{OFF} \ ] \} \\
\end{array}
\]

CPU \text{cpu-number}

DEFINITIONS ONLY

DIRECTIVES " [ ; ] directive... "

\[
\{ \text{class-name} \}
\]

EXCLUDE \{ \}
\{ \ ( \text{class-name} \ [ , \text{class-name} ] \) \}

\[
\{ \text{class-name} \}
\]

INCLUDE \{ \}
\{ \ ( \text{class-name} \ [ , \text{class-name} ] \) \}
SYNTAX SUMMARY

SET

{ 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:

CLASS [ class-name [ ] ]

[ ON ]
[ OFF ]

class-name is one of the following:
SYNTAX SUMMARY

SET

<table>
<thead>
<tr>
<th>BLOCKDATAS</th>
<th>INLINES</th>
<th>PROGLABELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCKS</td>
<td>KEYWORDS</td>
<td>REGISTERS</td>
</tr>
<tr>
<td>CONDITIONS</td>
<td>LINENOS</td>
<td>SCREENS</td>
</tr>
<tr>
<td>CONSTANTS</td>
<td>LITERALS</td>
<td>SUBPROCS</td>
</tr>
<tr>
<td>FILES</td>
<td>MACROS</td>
<td>SYSVARS</td>
</tr>
<tr>
<td>FMTLABELS</td>
<td>MNEMONICS</td>
<td>TYPES</td>
</tr>
<tr>
<td>FUNCTIONS</td>
<td>PROCEDURE PARAMS</td>
<td>VARIABLES</td>
</tr>
<tr>
<td>INDEXES</td>
<td>PROCEDURES</td>
<td></td>
</tr>
</tbody>
</table>

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
- BLOCKS
- CONDITIONS
- CONSTANTS
- FILES
- FMTLABELS
- FUNCTIONS
- INDEXES
- 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.
WARNING AND ERROR MESSAGES
Warning Messages

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 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.
WARNING AND ERROR MESSAGES
Error Messages

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.
WARNING AND ERROR MESSAGES
Error Messages

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.
WARNING AND ERROR MESSAGES
Fatal Error Messages

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

Alphabet names
COBOL85 7-2
COBOL85 compiler attribute 7-13
COBOL 74 6-2
COBOL 74 compiler attribute 6-13
SCREEN COBOL 11-2

Arrays
FORTRAN 9-2

Attribute list 3-4
Attribute specifications
displaying 4-41/44
list of
CLASS 4-22, 4-25, 4-28/30, 4-35/36, 4-41/42
CPU 4-23, 4-25, 4-30, 4-36, 4-42
DEFINITIONS ONLY 4-23, 4-25, 4-30, 4-36, 4-42
DIRECTIVES 4-23, 4-25, 4-30, 4-36, 4-42
EXCLUDE 4-23, 4-31, 4-37, 4-42
INCLUDE 4-23, 4-31, 4-37, 4-42
LANGUAGE 4-23, 4-25, 4-31, 4-37, 4-42
LIBRARY 4-23, 4-26, 4-32, 4-38, 4-42
MEM 4-23, 4-26, 4-32, 4-38, 4-43
OMIT 4-23, 4-32, 4-38, 4-43
PRIORITY 4-24, 4-26, 4-32, 4-38, 4-43
PROGRAM 4-24, 4-26, 4-33, 4-39, 4-43
UNREF 4-24, 4-26, 4-33, 4-39, 4-43
restoring 4-22/24
saving 4-25/27
specifying
SCAN command 4-28/34
SET command 4-35/40
summary of 4-3/4

BLOCK DATA subprograms
FORTRAN 9-2

BLOCKDATA$ identifier class
FORTRAN 9-2
TAL 12-2
INDEX

BLOCKS identifier class
  FORTRAN  9-2
  Pascal  10-2

C
  data types  5-2
  default settings  5-2
  identifier classes
    CONSTANTS  5-2
    FUNCTIONS  5-2
    MACROS  5-2
    TYPES  5-2
    VARIABLES  5-2
  sample identifier listing  5-6/9
  sample program  5-3/4
CLASS attribute specification
  displaying  4-41/42
  restoring  4-22
  saving  4-25
  specifying
    SCAN command  4-28/30
    SET command  4-35/36
Class names
  COBOL85  7-2
  COBOL85 compiler attribute  7-13
COBOL85
  compiler attributes
    alphabet-name  7-13
    class-name  7-13
    condition-name  7-13
    data-name  7-14/15
    external switch  7-16
    file-name  7-15/16
    index-name  7-16
    literals  7-17
    mnemonic-name  7-17
    paragraph-name  7-17
    program-name  7-18
    section-name  7-18
    symbolic-character  7-18
  compiler directives  7-3
  data types  7-2
  default settings  7-2
  identifier classes
    CONDITIONS  7-2
    CONSTANTS  7-2
    FILES  7-2
    FUNCTIONS  7-2
    INDEXES  7-2
    LITERALS  7-2
    MNEMONICS  7-2
    PROCEDURES  7-2
    PROGLABELS  7-2
INDEX

VARIABLES  7-2
sample identifier listing  7-10/13
sample program  7-5/9
COBOL  74
compiler attributes
  alphabet-name  6-13
  condition-name  6-13
  data-name  6-14/15
  file-name  6-15/16
  index-name  6-16
  literals  6-16
  mnemonic-name  6-16
  paragraph-name  6-17
  section-name  6-17
compiler directives  6-3
data types  6-2
default settings  6-2
identifier classes
  CONDITIONS  6-2
  FILES  6-2
  FUNCTIONS  6-2
  INDEXES  6-2
  LITERALS  6-2
  MNEMONICS  6-2
  PROCEDURES  6-2
  PROGLABELS  6-2
  VARIABLES  6-2
sample identifier listing  6-10/13
sample program  6-5/9
Command files
  specifying  4-18/19
Command lines
  continuation  2-3, 2-4
  maximum length  2-3, 2-4
  multiple commands  2-3, 2-4
COMMENT command  4-5, A-1
COMMON blocks
  FORTRAN  9-2
Compiler
CPU
  displaying  4-42
  restoring  4-23
  saving  4-25
  specifying  4-30, 4-36
diagnostic messages  4-8
directive string
  clearing  4-23
  displaying  4-42
  saving  4-25
  specifying  4-30, 4-36
INDEX

execution priority
   displaying  4-43
   restoring  4-24
   saving  4-26
   specifying  4-32, 4-38
memory page setting
   displaying  4-43
   restoring  4-23
   saving  4-26
   specifying  4-32, 4-38
program file setting
   displaying  4-43
   restoring  4-24
   saving  4-26
   specifying  4-33, 4-39
Compiler-dependent mode  1-3
Condition names
   COBOL85  7-2
   COBOL85 compiler attribute  7-13
   COBOL 74  6-2
   COBOL 74 compiler attribute  6-13
   SCREEN COBOL  11-2
CONDITIONS identifier class
   COBOL85  7-2
   COBOL 74  6-2
   SCREEN COBOL  11-2
CONSTANTS identifier class
   C  5-2
   COBOL85  7-2
   FORTRAN  9-2
   Pascal  10-2
   TAL  12-2
CPU attribute specification
   displaying  4-42
   restoring  4-23
   saving  4-25
   specifying
      SCAN command  4-30
      SET command  4-36
CROSSREF
   examples of operation  2-4/7
   execution command  2-1/2
   exiting  4-10
   location  2-1
   modes of operation  1-2, 1-3
   product banner  2-3
   work files  2-2
CROSSREF commands
   list of
      COMMENT  4-5, A-1
      ENV  4-6/7, A-1
      ERRORS  4-8/9, A-1
      EXIT  4-10, A-2
INDEX

FC  4-11, A-2  
GENERATE  4-12/13, A-2  
HELP  4-14/15, A-2  
LOG  4-16/17, A-2  
OBEY  4-18/19, A-3  
OUT  4-20/21, A-3  
RESET  4-22/24, A-3/4  
SAVE  4-25/27, A-4/5  
SCAN  4-28/34, A-5-7  
SET  4-35/40, A-7/9  
SHOW  4-41/44, A-10/11  
SYSTEM  4-45, A-11  
VOLUME  4-46, A-11  
summary of  4-1/2

CROSSREF output  
directing the output  4-12/13  
format  3-1/7

Data BLOCKS  
TAL  12-2

Data names  
COBOL85  7-2  
COBOL85 compiler attribute  7-14/15  
COBOL 74  6-2  
COBOL 74 compiler attribute  6-14/15  
SCREEN COBOL  11-2

DEFINES  
TAL  12-2

Definition reference  3-6/7

DEFINITIONS ONLY attribute specification  
displaying  4-42  
restoring  4-23  
saving  4-25  
specifying  
   SCAN command  4-30  
   SET command  4-36

DIRECTIVES attribute specification  
displaying  4-42  
restoring  4-23  
saving  4-25  
specifying  
   SCAN command  4-30  
   SET command  4-36

Displaying  
attribute specifications  4-41/44  
command syntax  4-14  
environment parameters  4-6

Dummy procedures  
FORTRAN  9-2
INDEX

Editing commands 4-11
End-of-File
  character 2-3
  command 2-3
Enumeration constants
  C 5-2
ENV command 4-6/7, A-1
Environment parameters
  displaying 4-6
  setting
    LOG 4-16
    SYSTEM 4-45
    VOLUME 4-46
Error messages B-3/13
ERRORS command 4-8/9, A-1
EXCLUDE attribute specification 4-23, 4-31, 4-37, 4-42
EXCLUDE list
  adding identifiers to
    SCAN command 4-31
    SET command 4-37
  displaying 4-42
  emptying 4-23
EXIT command 4-10, A-2
EXTENDED BASIC
  data types 8-2
  default settings 8-2
  identifier classes
    FUNCTIONS 8-2
    KEYWORDS 8-2
    LINENOS 8-2
    LITERALS 8-2
    SYSVARS 8-2
    VARIABLES 8-2
  sample identifier listing 8-6/8
  sample program 8-5
External switch
  COBOL85 compiler attribute 7-16
FC command 4-11, A-2
file
  list 3-1/2
  name
    file list 3-1/2
    identifier header 3-4
    identifier reference lines 3-6
  number
    file list 3-1/2
    identifier header 3-4
    identifier reference lines 3-6
File names
    COBOL85  7-2
    COBOL85 compiler attribute  7-15/16
    COBOL 74  6-2
    COBOL 74 compiler attribute  6-15/16
FILES identifier class
    COBOL85  7-2
    COBOL 74  6-2
FMTLABELS identifier class
    FORTRAN  9-2
FORTRAN
    data types  9-2
    default settings  9-2
    identifier classes
        BLOCKDATAS  9-2
        BLOCKS  9-2
        CONSTANTS  9-2
        FMTLABELS  9-2
        FUNCTIONS  9-2
        INLINES  9-2
        LITERALS  9-2
        PROCEDURE PARAMS  9-2
        PROCEDURES  9-2
        PROGLABELS  9-2
        SUBPROCS  9-2
        VARIABLES  9-2
        sample identifier listing  9-6/8
        sample program  9-5
FUNCTION subprograms
    FORTRAN  9-2
Functions
    C  5-2
        EXTENDED BASIC  8-2
FUNCTIONS identifier class
    C  5-2
    COBOL85  7-2
    COBOL 74  6-2
    EXTENDED BASIC  8-2
    FORTRAN  9-2

GENERATE command  4-12/13, A-2
GOTO labels
    Pascal  10-2

HELP command  4-14/15, A-2

Identifier classes
    defaults  3-8
    displaying  4-41/42
    list of
        BLOCKDATAS  4-22, 4-29, 4-35, 4-42
        BLOCKS  4-22, 4-29, 4-35, 4-42
        CONDITIONS  4-22, 4-29, 4-35, 4-42
INDEX

CONSTANTS 4-22, 4-29, 4-35, 4-42
FILES 4-22, 4-29, 4-35, 4-42
FMTLABELS 4-22, 4-29, 4-35, 4-42
FUNCTIONS 4-22, 4-29, 4-35, 4-42
INDEXES 4-22, 4-29, 4-35, 4-42
INLINES 4-22, 4-29, 4-35, 4-42
KEYWORDS 4-22, 4-29, 4-35, 4-42
LINENOS 4-22, 4-29, 4-35, 4-42
MACROS 4-22, 4-29, 4-35, 4-42
MNEMONICS 4-22, 4-29, 4-35, 4-42
PROCEDURE PARAMS 4-22, 4-29, 4-35, 4-42
PROCEDURES 4-22, 4-29, 4-35, 4-42
PROGLABELS 4-22, 4-29, 4-35, 4-42
REGISTERS 4-22, 4-29, 4-35, 4-42
SCREENS 4-22, 4-29, 4-35, 4-42
SUBPROCS 4-22, 4-29, 4-35, 4-42
SYSVARS 4-22, 4-29, 4-35, 4-42
TYPES 4-22, 4-29, 4-35, 4-42
VARIABLES 4-22, 4-29, 4-35, 4-42

restoring 4-22
saving 4-25
specifying
SCAN command 4-28/30
SET command 4-35/36
used by C 5-2
used by COBOL85 7-2
used by COBOL 74 6-2
used by EXTENDED BASIC 8-2
used by FORTRAN 9-2
used by Pascal 10-2
used by SCREEN COBOL 11-2
used by TAL 12-2

Identifier header
attribute list 3-4
file name 3-4
file number 3-4
identifier name 3-3
identifier qualifier 3-3
line number 3-4

Identifier reference lines
file name 3-6
file number 3-6
line number 3-6
reference codes 3-6/7

INCLUDE attribute specification 4-23, 4-31, 4-37, 4-42

INCLUDE list
adding identifiers to
SCAN command 4-31
SET command 4-37
displaying 4-42
emptying 4-23
INDEX

Index names
  COBOL85  7-2
  COBOL85 compiler attribute  7-16
  COBOL 74  6-2
  COBOL 74 compiler attribute  6-16

Index registers
  TAL USE statements  12-2

INDEXES identifier class
  COBOL85  7-2
  COBOL 74  6-2

Inline functions
  FORTRAN  9-2

INLINES identifier class
  FORTRAN  9-2

Input file
  recording comments for input  4-5

Interactive mode  2-3

Invocation reference  3-6/7

KEYWORDS identifier class
  EXTENDED BASIC  8-2

LABELs
  TAL  12-2

Labels
  executable statements in FORTRAN  9-2
  FORMAT statements in FORTRAN  9-2

LANGUAGE attribute specification
  clearing  4-23
  displaying  4-42
  saving  4-25
  specifying
    SCAN command  4-31
    SET command  4-37

Languages compatible with CROSSREF  1-2

LIBRARY attribute specification
  clearing  4-23
  displaying  4-42
  saving  4-26
  specifying
    SCAN command  4-32
    SET command  4-38

Line number
  identifier header  3-4
  identifier reference lines  3-6

LINENOS identifier class
  EXTENDED BASIC  8-2

LITERALs
  TAL  12-2

Literals
  COBOL85 compiler attribute  7-17
  COBOL 74 compiler attribute  6-16
INDEX

LITERALS identifier class
- COBOL85 7-2
- COBOL 74 6-2
- EXTENDED BASIC 8-2
- FORTRAN 9-2
- SCREEN COBOL 11-2
- LOG command 4-16/17, A-2

MACROS identifier class
- C 5-2
- TAL 12-2

MEM attribute specification
- displaying 4-43
- restoring 4-23
- saving 4-26
- specifying
  - SCAN command 4-32
  - SET command 4-38

Messages
- error B-3/10
- fatal B-10/13
- warning B-1/2

Miscellaneous references 3-6/7

Mnemonic names
- COBOL85 7-2
- COBOL85 compiler attribute 7-17
- COBOL 74 6-2
- COBOL 74 compiler attribute 6-16
- SCREEN COBOL 11-2

MNEMONONICS identifier class
- COBOL85 7-2
- COBOL 74 6-2
- SCREEN COBOL 11-2

Modes of operation
- compiler-dependent 1-3
- stand-alone 1-3

Named constants
- FORTRAN 9-2
- Pascal 10-2

Non-public variables
- Pascal 10-2

Noninteractive mode 2-4

Nonnumeric constants
- COBOL85 7-2
- COBOL 74 6-2
- SCREEN COBOL 11-2

Nontemplate STRUCTs
- TAL 12-2

Numeric constants
- COBOL85 7-2
- COBOL 74 6-2
- SCREEN COBOL 11-2

Index-10
OBEY command  4-18/19, A-3
OMIT attribute specification  4-23, 4-32, 4-38, 4-43
OMIT list
   adding files to
      SCAN command  4-32
      SET command  4-38
displaying  4-43
emptying  4-23
OUT command  4-20/21, A-3
Output listing
   direct ing the output  4-12/13
   format  3-1/7
Output listing file  4-20

Paragraph names
COBOL85  7-2
   COBOL85 compiler attribute  7-17
COBOL 74  6-2
   COBOL 74 compiler attribute  6-17
SCREEN COBOL  11-2
Parameter reference  3-6/7
PARAMETERS
   FORTRAN  9-2
Pascal
   data types  10-2
   default settings  10-2
   identifier classes
      BLOCKS  10-2
      CONSTANTS  10-2
      PROCEDURES  10-2
      PROGLABELS  10-2
      TYPES  10-2
      VARIABLES  10-2
   sample identifier listing  10-6/9
   sample program  10-3/4
PRIORITY attribute specification
   displaying  4-43
   restoring  4-24
   saving  4-26
   specifying
      SCAN command  4-32
      SET command  4-38
PROCEDURE PARAMS identifier class
   FORTRAN  9-2
   TAL  12-2
PROCEDURES identifier class
COBOL85  7-2
   COBOL 74  6-2
   FORTRAN  9-2
   Pascal  10-2
   SCREEN COBOL  11-2
   TAL  12-2
INDEX

PROCs
  TAL 12-2
PROGLABELS identifier class
  COBOL85 7-2
  COBOL 74 6-2
  FORTRAN 9-2
  Pascal 10-2
  SCREEN COBOL 11-2
  TAL 12-2
PROGRAM attribute specification
  displaying 4-43
  restoring 4-24
  saving 4-26
  specifying
    SCAN command 4-33
    SET command 4-39
Program names
  COBOL85 7-2
  COBOL85 compiler attribute 7-18
  COBOL 74 6-2
  SCREEN COBOL 11-2
Public variables
  Pascal 10-2

Read reference 3-6/7
Records
  FORTRAN 9-2
Reference codes
  blank 3-6/7
  D 3-6/7
  I 3-6/7
  M 3-6/7
  P 3-6/7
  W 3-6/7
REGISTERS identifier class
  TAL 12-2
RESET command 4-22/24, A-3/4
Routines
  COBOL85 7-2
  COBOL 74 6-2
Running CROSSREF
  compiler-dependent mode 1-3
  interactive mode 2-3
  invoking 2-1/2
  noninteractive mode 2-4
  stand-alone mode 1-3

SAVE command 4-25/27, A-4/5
SCAN command 4-28/34, A-5/7

Index-12
SCREEN COBOL

data types  11-2
default settings  11-2
identifier classes
  CONDITIONS  11-2
  LITERALS  11-2
  MNEMONICS  11-2
  PROCEDURES  11-2
  PROGLABELS  11-2
  SCREENS  11-2
  VARIABLES  11-2
  sample identifier listing  11-9/12
  sample program  11-4/8
Screen names
  SCREEN COBOL  11-2
SCREENS identifier class
  SCREEN COBOL  11-2
Section names
  COBOL85  7-2
  COBOL85 compiler attribute  7-18
  COBOL 74  6-2
  COBOL 74 compiler attribute  6-17
  SCREEN COBOL  11-2
SET command  4-35/40, A-7/9
SHOW command  4-41/44, A-10/11
Statement functions
  FORTRAN  9-2
  STRUCT templates
    TAL  12-2
SUBPROCs
  TAL  12-2
SUBPROCS identifier class
  FORTRAN  9-2
    TAL  12-2
SUBROUTINE subprograms
  FORTRAN  9-2
Symbolic characters
  COBOL85  7-2
Symbolic-character
  COBOL85 compiler attribute  7-18
SYSTEM command  4-45, A-11
System variables
  EXTENDED BASIC  8-2
SYSVARS identifier class
  EXTENDED BASIC  8-2

TAL
  data types  12-2
  default settings  12-2
identifier classes
    BLOCKDATAS  12-2
    CONSTANTS  12-2
    MACROS  12-2
    PROCEDURE PARAMS  12-2
    PROCEDURES  12-2
    PROGLABELS  12-2
    REGISTERS  12-2
    SUBPROCS  12-2
    TYPES  12-2
    VARIABLES  12-2
    sample identifier listing  12-7
    sample program  12-5
TYPES identifier class
    C  5-2
    Pascal  10-2
    TAL  12-2

Unnamed constants
    FORTRAN  9-2
UNREF attribute specification
    displaying  4-43
    restoring  4-24
    saving  4-26
    specifying
        SCAN command  4-33
        SET command  4-39

Variables
    FORTRAN  9-2
VARIABLES identifier class
    C  5-2
    COBOL85  7-2
    COBOL 74  6-2
    EXTENDED BASIC  8-2
    FORTRAN  9-2
    Pascal  10-2
    SCREEN COBOL  11-2
    TAL  12-2
VOLUME command  4-46, A-11

Warning messages  B-1/2
work files, CROSSREF  2-2
Write reference  3-6/7

#define
    C  5-2