LEGAL NOTICE

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

"This technical report has been reviewed and is approved for publication."

HENRY J. EIDEN, Major, USAF
Program Manager

FOR THE COMMANDER

FRANK J. ZIMMA, Colonel, USAF
Director, Information Systems
Technology Applications Office
Deputy for Command & Management Systems
m: ESD-TR-75-353

18 June 74

Above TR not received in DTIC. Our copy may from

U Michigan Lit, Dept Industrial & Operations
Engineering 313-764-6473.

of all authors, Dr. Teichnor still a Michigan U.

2 June - Dr. Barron called - he to check rec.

2 July - Teichnor called. The above TR is obsolete.

PSL/PSA Language replaced above.

" " Users Manual (used at Turner AFB)

available commercially:

ISD, Inc., 325 Eisenhower Pkwy,
Suite 103, Ann Arbor Michigan 48104.

He'll look for a copy of TR & send.

19 July - Mr. Barron sent me a PSL/PSA - on ESD-TR.

30 July 74 - HAND-CARRIED TR to Demlin Corrazini - to
be sent to DTIC for photostating - ON LOAN - and
RETURNED TO AFGL Library with AD number.
The department record accordingly.

Although the total is assumed to be 40

*** INPUTS ***

- P.M.-employee-record USED

*** OUTPUTS ***

- P.M.-statement GENERATED
- P.M.-listing GENERATED
- P.M.-employee-report GENERATED
- P.M.-listing DERIVED
- P.M.-employee-report DERIVED
- P.M.-statement DERIVED

There are routines used by one or more processes in the system.

*** OUTPUTS ***

** INPUTS FOR THIS PROCESS **

*** OUTPUTS ***

** OUTPUTS FOR THIS PROCESS **

** Tag-esp-processing **

- contains personnel data for those employees who
  spent the evening after the filing. It also
  contains information for those employees who

** Printed on Form **

1. ** Print tag of employee by employment status item **
Figure 58

- Employee records
- Payroll records
- Tax records
- Benefits records

---

**OUTCOMES**

- Increase employee productivity
- Reduce errors
- Improve communication

---

**IMPACTS**

- Reduced time spent on payroll processing
- Increased accuracy of records
- Improved employee satisfaction

---

- The system provides accurate information about new employees and
  future employment status.
The User Requirements Analyzer (URA) is one of two major components of computer aided requirements analysis. URA is used in conjunction with the User Requirements Language (URL) to generate, maintain, and analyze URL data bases. URL is described in ESD-TR-75-88, Vol. II. This document describes URA version 2.1 in the H6180/Multics computing environment. URA version 2.0 is implemented in the IBM 370/158/TSO computing environment and described in ESD-TR-75-88, Vol. III.
H6180/MULTICS/URA USER'S MANUAL

DRI Call No. 83713
Copy No. 1 of 2

ISDOS Research Project

November 1975

Approved for Public Release; Distribution Unlimited.

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
HQ ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731
The User Requirements Analyzer (URA) is one of two major components of computer aided requirements analysis. URA is used in conjunction with the User Requirements Language (URL) to generate, maintain, and analyze URL data bases. URL is described in ESD-TR-75-88, Vol. II. This document describes URA version 2.1 in the H6180/Multics computing environment. URA version 2.0 is implemented in the IBM 370/158/ISO computing environment and described in ESD-TR-75-88, Vol. III.
SUMMARY

Part I User Requirements Analyzer

The goal of Part I is to assist the User Requirements Analyzer (URA) user in effectively using the URA Modifier Commands specified in Part IV, "User Requirements Analyzer Command Descriptions." This part illustrates the usage of Version A2.1 of the User Requirements Analyzer and specifies the steps in creating the URA data base, inputting User Requirements Language (URL) statements, modifying the contents of the data base, generating URA outputs, and correcting syntactical and logical errors.

Since URA is used in conjunction with an operating system, this part should be used in conjunction with Part II which presents installation dependent features to be considered when using URA. This part covers installation independent features of URA, general concepts, etc. Each section of this part has a corresponding section in Part II. This allows easy reference between general concepts and the actual practice of applying them in a particular installation.

Part II Usage of the User Requirements Analyzer under Multics

The purpose of Part II is to assist the URA user in effectively manipulating the URA command language under Multics. This part covers those installation dependent features of URA and is intended to be used in conjunction with the User Requirements Analyzer, Part I. Each section of this part has a corresponding section in the User Requirements Analyzer. This part illustrates the usage of Version A2.1 of the User Requirements Analyzer.

Part III URA Outputs

The goals of Part III are to assist the User Requirements Analyzer user in generating reports from the information in a URA data base, describe the standard reports available in URA, and finally, provide general guidelines on using these reports to aid in the logical system design process. In order to generate the reports described in this part it is necessary to understand the information presented in Part I and Part II for the installation in which URA is being used. It is also desirable to use the "User Requirements Analyzer Command Descriptions", Part IV, as a reference for a better understanding of the URA commands and parameters used to generate a particular report.

This part describes those URA reports available in Version A2.1 of the User Requirements Analyzer.
Part IV User Requirements Analyzer Command Descriptions

The objective of Part IV is to give the user of the User Requirements Analyzer (URA) the list of the commands available, the correct syntax of these commands, and the parameters allowable for each command. This part is not intended as a handbook on how to use URA, but rather as a reference for the commands available. Parts I and II describe how to use these commands and present a detailed description of each of the outputs generated by these commands. This part describes the facilities of Version A2.1 of the User Requirements Analyzer.

The manner in which URA directly interfaces with the Multics operating system is given in Appendix E. Also included in that Appendix is a formal description of the URA commands available only under that particular operating system.

Part V Automated Documentation System Users' Manual

The objective of Part V is to provide detailed operating instructions for the Automated Documentation System (ADS). ADS is a stand-alone program and file structure that operates on a URL data base to produce requirements specifications in standard formats (e.g. Mil Std 483).
# TABLE OF CONTENTS

## PART I  USER REQUIREMENTS ANALYZER

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>14</td>
</tr>
<tr>
<td>2.</td>
<td>16</td>
</tr>
<tr>
<td>2.1</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>17</td>
</tr>
<tr>
<td>3.</td>
<td>19</td>
</tr>
<tr>
<td>3.1</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>20</td>
</tr>
<tr>
<td>4.</td>
<td>22</td>
</tr>
<tr>
<td>4.1</td>
<td>22</td>
</tr>
<tr>
<td>4.2</td>
<td>22</td>
</tr>
<tr>
<td>4.3</td>
<td>23</td>
</tr>
<tr>
<td>4.4</td>
<td>23</td>
</tr>
<tr>
<td>4.5</td>
<td>23</td>
</tr>
<tr>
<td>5.</td>
<td>24</td>
</tr>
<tr>
<td>5.1</td>
<td>24</td>
</tr>
<tr>
<td>5.2</td>
<td>25</td>
</tr>
<tr>
<td>5.3</td>
<td>25</td>
</tr>
<tr>
<td>6.</td>
<td>26</td>
</tr>
<tr>
<td>7.</td>
<td>27</td>
</tr>
<tr>
<td>7.1</td>
<td>31</td>
</tr>
<tr>
<td>7.2</td>
<td>38</td>
</tr>
<tr>
<td>7.3</td>
<td></td>
</tr>
<tr>
<td>7.4</td>
<td></td>
</tr>
</tbody>
</table>
8. Report Commands

9. Error Conditions
   9.1 Initial Messages
   9.2 Abnormal Termination
   9.3 Data Base Already Open

PART III URA OUTPUTS

1. Introduction

2. Objects of Reports From a URA Data Base
   2.1 Purpose of URA Reports
   2.2 Relation of URA Reports to Logical System Design
   2.3 Advantages of Using URA Reports

3. Generation of Reports
   3.1 URA Command Language for Reports
   3.2 Retrieval of Information from the URA Data Base

| ATTRIBUTE REPORT | 119 |
| CONSISTS COMPARISON REPORT | 123 |
| CONSISTS MATRIX REPORT | 132 |
| CONTENTS REPORT | 144 |
| DATA BASE SUMMARY | 150 |
| DATA PROCESS REPORT | 153 |
| DICTIONARY REPORT | 174 |
| EXTENDED PICTURE REPORT | 180 |
| FORMATTED PROBLEM STATEMENT | 195 |
| FREQUENCY REPORT | 209 |
| IDENTIFIER INFORMATION REPORT | 212 |
| KWIC INDEX | 217 |
| NAME GEN | 219 |
A Note on Version A2.1 of URA
The URA Command Language
URA Command Language Installation Dependencies
Command Language Syntax Notation
Format of Command Descriptions
CHANGE-TYPE
CONSISTS-COMPARISON
CONSISTS-MATRIX
CONTENTS
DATA-PROCESS
DELETE
DELETE-COMMENT-ENTRY
DELETE-PSL
DICTIONARY
ENTITY-IDENTIFIER
EXTENDED-PICTURE
FORMATTED-PROBLEM-STATEMENT
FREQUENCY
PART V

AUTOMATED DOCUMENTATION SYSTEM USERS' MANUAL

INTRODUCTION

1.0 Document Initialization

1.1 Strategy for the document initialization and general process

1.2 Syntax for entries in the Documentation Schema

2.0 Documentation data base population

2.1 Strategy for the documentation data base population

2.2 Syntax for the Documentation Data Base

3.0 The Documentation Generation

4.0 Usage of the Documentation Generator
4.1 Development of the Documentation Schema 351
4.2 Development of the Documentation Data Base 356
4.3 Development of the Analyzer Data Base with the Document Generator in Mind 361

APPENDIX A: URA Command Abbreviations 364
APPENDIX B: Creating and Initializing URA Data Bases 374
APPENDIX C: Dump/Restore Facility 376
APPENDIX D: Specification Generator 377
APPENDIX E: Using the URA Command Language with Multics 378
APPENDIX F: Example of Document Schema and Source 388
APPENDIX G: Document Examples 398
APPENDIX H: Executing Automatic Documentation System 484
GLOSSARY 485
<table>
<thead>
<tr>
<th>FIGURE</th>
<th>Description</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Interaction between Operating System and URA processing modes</td>
<td>15</td>
</tr>
<tr>
<td>2.</td>
<td>CHANGE-TYPE Report</td>
<td>32</td>
</tr>
<tr>
<td>3.</td>
<td>CHANGE-TYPE Report</td>
<td>34</td>
</tr>
<tr>
<td>4.</td>
<td>CHANGE-TYPE Report</td>
<td>35</td>
</tr>
<tr>
<td>5.</td>
<td>CHANGE-TYPE Report</td>
<td>36</td>
</tr>
<tr>
<td>6.</td>
<td>DELETION Report</td>
<td>39</td>
</tr>
<tr>
<td>7.</td>
<td>DELETION Report</td>
<td>41</td>
</tr>
<tr>
<td>8.</td>
<td>DELETED COMMENT ENTRIES Report</td>
<td>43</td>
</tr>
<tr>
<td>9.</td>
<td>DELETED COMMENT ENTRIES Report</td>
<td>45</td>
</tr>
<tr>
<td>10.</td>
<td>DELETED URL Report</td>
<td>48</td>
</tr>
<tr>
<td>11.</td>
<td>AS-IS SOURCE LISTING</td>
<td>52</td>
</tr>
<tr>
<td>12.</td>
<td>AS-IS SOURCE LISTING</td>
<td>53</td>
</tr>
<tr>
<td>13.</td>
<td>PUNCHED COMMENT ENTRIES Report</td>
<td>56</td>
</tr>
<tr>
<td>14.</td>
<td>PUNCHED COMMENT ENTRIES Report</td>
<td>58</td>
</tr>
<tr>
<td>15.</td>
<td>PUNCHED COMMENT ENTRIES Report</td>
<td>59</td>
</tr>
<tr>
<td>16.</td>
<td>RENAME Report</td>
<td>61</td>
</tr>
<tr>
<td>17.</td>
<td>RENAME Report</td>
<td>63</td>
</tr>
<tr>
<td>18.</td>
<td>REPLACED COMMENT ENTRIES</td>
<td>66</td>
</tr>
<tr>
<td>19.</td>
<td>Interaction between MULTICS and URA Processing Modes</td>
<td>104</td>
</tr>
<tr>
<td>20.</td>
<td>ATTRIBUTE Report</td>
<td>121</td>
</tr>
<tr>
<td>21.</td>
<td>CONSISTS COMPARISON Report</td>
<td>126</td>
</tr>
<tr>
<td>22.</td>
<td>CONSISTS MATRIX Report</td>
<td>136</td>
</tr>
<tr>
<td>23.</td>
<td>CONSISTS MATRIX Report</td>
<td>140</td>
</tr>
<tr>
<td>24.</td>
<td>CONTENTS Report</td>
<td>147</td>
</tr>
<tr>
<td>25.</td>
<td>CONTENTS Report</td>
<td>148</td>
</tr>
</tbody>
</table>
FIGURE 26. DATA BASE SUMMARY

27. DATA PROCESS Report

28. DATA PROCESS Report

29. DICTIONARY Report

30. DICTIONARY Report

31. EXTENDED PICTURE Report

32. EXTENDED PICTURE Report

33. FORMATTED PROBLEM STATEMENT Margin Parameters

34. FORMATTED PROBLEM STATEMENT

35. FORMATTED PROBLEM STATEMENT

36. FREQUENCY Report

37. IDENTIFIER INFORMATION Report

38. KWIC INDEX Report

39. NAME GEN Report

40. NAME GEN Report

41. NAME GEN Report

42. NAME GEN Report

43. NAME GEN Report

44. NAME LIST

45. General PICTURE Format and Limits per Page

46. INTERFACE PICTURE Report

47. SET PICTURE Report

48. INPUT PICTURE Report

49. OUTPUT PICTURE Report

50. ENTITY PICTURE Report

51. GROUP/ELEMENT PICTURE Report
<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PROCESS/PATTERN Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>52.</td>
<td>PROCESS PICTURE Report</td>
</tr>
<tr>
<td>53.</td>
<td>PROCESS PICTURE Report</td>
</tr>
<tr>
<td>54.</td>
<td>PROCESS PICTURE Report</td>
</tr>
<tr>
<td>55.</td>
<td>PROCESS PICTURE Report</td>
</tr>
<tr>
<td>56.</td>
<td>PROCESS CHAIN Report</td>
</tr>
<tr>
<td>57.</td>
<td>PROCESS INPUT/OUTPUT Report</td>
</tr>
<tr>
<td>58.</td>
<td>PROCESS INPUT/OUTPUT Report</td>
</tr>
<tr>
<td>59.</td>
<td>PUNCHED COMMENT ENTRIES</td>
</tr>
<tr>
<td>60.</td>
<td>INPUT STRUCTURE Report</td>
</tr>
<tr>
<td>61.</td>
<td>OUTPUT STRUCTURE Report</td>
</tr>
<tr>
<td>62.</td>
<td>INTERFACE STRUCTURE Report</td>
</tr>
<tr>
<td>63.</td>
<td>PROCESS STRUCTURE Report</td>
</tr>
<tr>
<td>64.</td>
<td>PROCESS INPUT/OUTPUT Report</td>
</tr>
<tr>
<td>65.</td>
<td>Documentation Initializer</td>
</tr>
<tr>
<td>66.</td>
<td>Documentation Data Base Populator</td>
</tr>
<tr>
<td>67.</td>
<td>Documentation Generator</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>248</td>
</tr>
<tr>
<td>251</td>
</tr>
<tr>
<td>254</td>
</tr>
<tr>
<td>256</td>
</tr>
<tr>
<td>262</td>
</tr>
<tr>
<td>269</td>
</tr>
<tr>
<td>270</td>
</tr>
<tr>
<td>276</td>
</tr>
<tr>
<td>279</td>
</tr>
<tr>
<td>280</td>
</tr>
<tr>
<td>281</td>
</tr>
<tr>
<td>282</td>
</tr>
<tr>
<td>286</td>
</tr>
<tr>
<td>343</td>
</tr>
<tr>
<td>348</td>
</tr>
<tr>
<td>350</td>
</tr>
<tr>
<td>TABLE</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>E-1</td>
</tr>
</tbody>
</table>
PART I

USER REQUIREMENTS ANALYZER
1. INTRODUCTION

The first step in use of the URL/URA system consists of specifying a problem statement in URL statements. The second step consists of using URA to enter the problem statement into a computer data base. URA extracts information from the URL statements and stores it in a URA data base. Once this information (a problem statement) is in the data base, it can be modified, new information can be added to it, and reports presenting the status of the problem statement can be generated. These actions are implemented by the URA commands available in the URA (processing) mode. This mode of operation may be attained by accessing the URA software available on a particular operating system. Therefore, by understanding the operating commands that interact with URA and the URA command language, the problem definer (user) can effectively manipulate the contents of a URA data base. Part I serves as a guide to using the URA commands.

Operating system and URA commands can only be used in their respective (processing) modes. Operating System commands can be used from time of signing on to the system, to the time access to the URA software is acquired. At this point, only URA commands may be used until the problem definer returns control to the operating system or terminates processing to be done in URA mode (through use of the URA "STOP" command). Operating system commands then can again be used up to the time of signing off. This interaction between operating system and URA modes is illustrated by Figure 1.

The first five sections of Part I deal with URA at an introductory level. Section 2 presents introductory information about the use of URA. Section 3 explains the procedure of initializing URA once on the operating system. Sections 4 and 5 present practical concepts and conventions to be known before using URA. (Once access to URA has been achieved, various commands are available; these commands are described in Sections 6, 7 and 8.) Several examples are given in these sections in order to better illustrate the results of specific implementations. Sections 9 and 10 deal with handling errors encountered in the use of URA. Appendix A presents a list of all URA commands available (and the parameters for each command) as well as the abbreviations for all these to serve as a quick reference. Throughout Part I the long forms of URA commands and parameters are used interchangeably with their abbreviations.
Figure 1: Interaction between Operating System and URA processing modes
2. USING URA

2.1 The URA Command Language

The URA Command Language commands may be grouped into three major types.

**Control Commands**

Control Commands are used to pass certain control information to the User Requirements Analyzer. These commands are operating system dependent and are given in Part II. The use of these commands does not change the contents of the data base.

**Modifier Commands**

Modifier Commands are used to modify the contents of the URA data base in the manner defined by the problem definer. These commands take legal URL statements or URL names as input. URA then generates error diagnostics as well as an output report to present the outcome of the data base alteration.

**Report Commands**

Report Commands retrieve data from the URA data base and output it in some standard format. These reports do not change the contents of the data base whatsoever. Their purpose is solely that of displaying some or all of the contents of the data base.

2.2 Command Parameters

Most URA commands have parameters which may be set by the user to modify the actions of the command. There are basically five types of parameters:

- **Input data parameters** - these parameters specify the data to be used as input to the command. INPUT, FILE and NAME are examples of Input data parameters.

- **Input control parameters** - these parameters specify how the input data is to be used, changed, etc., by the command. The TYPE parameter for the CHANGE-, TYPE command and CONTAINED/CONSISTS parameter for the CONSISTS-MATRIX command are examples of this type.

- **Output data parameters** - these parameters specify if output is to be generated from the command and the form in which it is presented. The PUNCH and PRINT parameters are examples of this type of parameter.
Output option parameters - these parameters specify options which may be included or omitted from the output. The LEVELS parameter for the CONTENTS command and the DESCRIPTION parameter for the DICTIONARY command are examples of this type.

Output format parameters - these parameters specify alternate formats for presenting the information in the output from the command. The NEW-PAGE parameter and Hmargin parameter for the FPS command are examples of this type.

2.3 NAME-GEN Command

Although all of the commands can be issued independently of each other, it is often advantageous to use some commands in sequence since the output of one command may be used as input by another. The most common instance of this is when NAME-GEN is used to select certain names (say all PROCESSES for example) which can then be used as input to a Report Command (possibly PICTURE, for a PICTURE REPORT for all PROCESS names). Though NAME-GEN is technically classified as a Report Command one of its major functions is to selectively retrieve names stored in the data base. For more information about use of this command, see "URA Outputs"* and "URA Command Descriptions."**

2.4 The HELP Command

The HELP Command provides the user with information about the syntax and parameters of URA commands. The HELP command does not affect the manner in which URA operates nor accesses information in the data base. For this reason, it is not classified as a Control, Modifier or Report Command. When the HELP Command is given, URA displays a list of all available URA commands and their abbreviations. By specifying a particular URA command name as a parameter to the HELP command:

```
HELP CONTENTS
```

e.g., all the parameters available for this command will be printed. If the "LONG" parameter were given in conjunction with a command name:
HELP FPS LONG

all parameters for the FORMATTED-PROBLEM-STATEMENT Command would be printed out as well as a description of the function of each of these parameters for the command. This description is presented in the same format as that in "USER REQUIREMENTS ANALYZER COMMAND DESCRIPTIONS".* To illustrate an example, when "HELP CONTENTS" was given the following information was printed:

CONTENTS

Prototype: CONTENTS(CONT) [parameter] ...

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE [=fname], NAME(N)=user name</td>
<td>FILE</td>
</tr>
<tr>
<td>INDEX, NOINDEX</td>
<td>NOINDEX</td>
</tr>
<tr>
<td>LEVELS=integer, LEVELS=ALL</td>
<td>ALL</td>
</tr>
<tr>
<td>NCFLAG, NONCFLAG</td>
<td>NONCFLAG</td>
</tr>
</tbody>
</table>

*Part IV
3. THE URA ENVIRONMENT

The considerations necessary for using URA and preparing for using URA define the URA environment. The following points must be considered:

- How the data base is prepared.
- How URA is executed.
- How URA is used.
- How batch use of URA is different from terminal use.

The first three points are presented in section 3.1, "Initiating URA," and the last point is presented in section 3.2, "Batch Versus On-line Use of URA."*

3.1 Initiating URA

- The steps required to prepare a URA data base for access by URA are:

  i) The URA data base file must be created.

  Creation of the data base file occurs only once. Once created, any changes to be made (including emptying the whole file) can be made without destroying it.

  ii) The data base must then be initialized in order to be accessed by URA.

  Initialization, for the most part, occurs only once, at the time of creating the data base. The data base must be reinitialized if emptied.

- Executing URA involves running the URA program. This allows the user to specify URA commands to change the contents of the data base or generate reports about its contents.

- Once the URA program has been invoked the following steps are suggested in using URA:

  i) The data base to be accessed by the URA commands must be specified.

  This should be done any time URA is used to ensure that the user is accessing the correct data base. In some cases the data base to be used will set by default any time the user executes URA. Even if this is the case, the user should be aware that the default is in effect.

* The exact commands to carry out the above procedures are installation-dependent and can be found in Part II (Section 3.1).
ii) URA commands are given to modify, update and/or generate reports on the database information.

Any of the commands given in "User Requirements Analyzer Command Descriptions"* can be issued to accomplish these tasks. The order in which commands are given and which commands are given is determined by the user.

iii) The STOP command is given to terminate the URA session.

3.2 Batch Versus On-line Use of URA

The manner in which a user interacts with URA via batch processing differs from on-line (terminal) usage. There are various advantages and disadvantages to either approach. A few of these are given in Table 1.

The procedures given in Section 3.1 are the same for both on-line and batch use of URA. The specific manner in which URA commands are given by the user, however, is different.

In batch processing of URA commands, URA commands are given, one per card, following the card which executes the URA program. Any errors detected in specifying the command (or its parameters) cause the command to be ignored, and URA then moves to the next command.

When executing URA in on-line mode, the user must wait for the message:

ENTER COMMAND (AND ANY PARAMETERS)

before giving any commands. After typing in the command followed by a carriage return, the user must wait again until the command prompting message is issued before entering the next command. If an error occurs when specifying the command (or its parameters) the user will be prompted for a replacement.

* Part IV
### On-line Use of URA

- The user is able to handle errors as they occur. Errors can be corrected before any attempt is made to modify or retrieve information in the data base.
- Which URA commands to be issued and the order in which they are given does not have to be predetermined, i.e., the user may issue commands ad hoc.
- Utilizing the edit facility of the operating system allows the user to make changes to information in the data base quickly and efficiently.

### Batch Use of URA

- Requires the URA user to think out procedures (list of URA commands) to be executed before they are executed.
- Cheaper to run than on-line use.
- All output generated by URA is printed in a useable format, i.e., via the line printer. Output generated at the terminal may be part of the terminal listing and interspersed with other information.

### Advantages

- Loss of connection between terminal and computer (line hits) may cause the contents of the data base to become unusable. Recovery procedures would be required to restore the data base contents.
- On-line use is generally more expensive than batch because of connect time costs and other additional costs related to terminal access.

### Disadvantages

- Turn-around time for jobs may be long.
- Editing of information in the data base may require two batch runs; one to retrieve, one to replace.
- Errors are ignored and any subsequent URA commands would be run regardless.

Table 1. Comparison of Outline versus Batch Use of URA
4. SPECIFYING INPUT TO URA COMMANDS

For most URA commands one or more names (specified by the user) can be used as "input" to the command. This can be done by utilizing the "input data" parameters for the command. In the case of Modifier Commands, the modification is made for each name used as input. For Report Commands, information is retrieved for each of the names used as input. Except for the INPUT-PSL command, all names used as input to the Modifier and Report Commands must be names already stored in the problem definer's URA data base.

4.1 The NAME Parameter

There are two methods of specifying names to be input to a command. The simplest way is to use the NAME parameter. When this parameter is used, the modification will be made, or report will be generated, for only that name specified by the NAME parameter. For example, if NAME=T-CARD were used for the DELETE command, only T-CARD would be deleted from the data base. Likewise, if NAME=T-CARD were used as a parameter for the CONTENTS command, the CONTENTS REPORT would be generated for the name T-CARD, and no others.

4.2 The FILE and INPUT Parameters

The second way to specify names as input to a URA command is to put all the names for which the modification is to be made, or report generated, into a file and specify that the contents of that file are to be used as input. At most installations this specification can simply be done via the FILE and INPUT parameters, but varies slightly from one installation to the next. (See Part II, Section 4.) FILE and INPUT are different in the way names can be formatted within the file specified by these parameters. When using the FILE parameter, each name in the specified file must begin in the first column of each line of the file and only one name per line is allowed. The format for files specified by the INPUT parameter varies according to the URA command using this parameter. For example, if a file is used as input to the INPUT-PSL command, the file must consist of URL statements to be entered into the URA data base. For those Modifier Commands that allow the INPUT parameter, the particular format needed for the input file is specified in the description of each command.

If one particular file name is used throughout a given terminal session to contain various information and name lists to interact with URA commands, the user should remember to empty the file before each time new data is to be written into it. Otherwise, information used as input to a previous command may be leftover when using the file with subsequent commands.
4.3 Entering Data Into an Input File

The manner in which data is entered into a file is installation-dependent. Part II, Section 4, for the details of this procedure.

4.4 Using NAME-GEN

One alternative to specifying input to Modifier and Report Commands is to let NAME-GEN generate the input file. The various parameters for NAME-GEN allow selection of particular types of names that are desired (e.g., all GROUPS and ENTITIES) and retrieval of these names which are then put into a file by URA. The names are formatted, one name per line starting in the first column of the line. The contents of this file can be used as input to a Report Command or Modifier Command. For example, by specifying:

```
NAME-GEN ENTITY GROUP
CONTENTS
```

in sequence, the CONTENTS REPORT will be generated for all ENTITY and GROUP names in the URA data base. In addition, the contents of the file produced by NAME-GEN are maintained until the next NAME-GEN is issued.

4.5 Using PUNCH Files

PUNCH files are files which have formats acceptable by FILE or INPUT parameters. The file described in the previous section is a PUNCH file from the NAME-GEN command. Output from NAME-GEN is put into its assigned PUNCH file so that it may be used as input to any of the FILE parameters for Modifier and Report Commands. The PUNCH file format is different from the report format for the command that generates both of them. For example, upon execution of the NAME-GEN command for all PROCESSES, the report generated will consist of a report heading, line numbers for the contents of the report, the names of all PROCESSES in the data base and their corresponding name type (which is, of course, PROCESS). The contents of the PUNCH file produced by this command will only contain the names of the PROCESSES, without report headings, etc. In other words, the PUNCH file contains similar information to the report output from the command, but in a format acceptable to the FILE and INPUT parameters of other URA commands.

At most installations, the PUNCH file to be used (name of the file) can be specified by the PUNCH parameter for the command. The manner of assignment varies slightly from one installation to the next. (See Part II, Section 4.) Specific usage of the PUNCH parameter is given in the descriptions of the individual commands that utilize this parameter. The specific names of the PUNCH files used can be found in Appendix E.
5. RECEIVING OUTPUT FROM URA COMMANDS*

Several URA commands allow the user to specify whether output is generated from the command or not. This is done via the "output data" parameters for the particular command. When generating outputs from URA, the information is put into a file or printed on a device such as a line printer or terminal. If this file or device is not specified then all outputs are written to the main output file or device. This means that output will be written on the terminal when in conversational mode and on the line printer when running batch. There are several reasons why outputs might be routed elsewhere, especially for on-line processing:

- Large quantities of output would take too long to be printed at the terminal.

- Depending on the type of terminal, some portions of the output may not be printed because of physical restrictions imposed by the terminal.

- The handling of printout from the terminal can sometimes be awkward and the format not aesthetically pleasing.

- No copy of the output is desired. (Only the PUNCH file may be needed as a step in a modification procedure).

Most methods of receiving and controlling output from URA commands are installation dependent and therefore given in Part II, Section 5.

5.1 The NOPRINT Parameter

Several URA commands allow the option of not having the output printed via the NOPRINT parameter. The commands that allow this parameter are:

DELETE-COMMENT-ENTRY (DCOM)
FORMATTED-PROBLEM-STATEMENT (FPS)
NAME-GEN (NG)
PRINT-COMMENT-ENTRY (PCOM)
PROCESS-INPUT-OUTPUT (PRIO)
REPLACE-COMMENT-ENTRY (RCOM)

The two Modifier Commands RCOM and DCOM have this parameter available because the printout can be fairly large and may not be needed for future reference. The report heading for the RCOM or DCOM output and any error diagnostics are still printed to provide a hard-copy record of the command execution.

* This section only deals with receiving outputs in the form of reports (as specified in Part III). Receiving output as presented via the PUNCH parameter is discussed in the previous section.
The remaining four Report Commands can use this parameter in conjunction with the PUNCH parameter. The option of the NOPRINT parameter is provided because when PUNCH information is desired, there may be no need for the printout.

5.2 The INDEX Parameter

Several commands allow the user to specify that an index (alphabetical listing of names used) for the report be generated by the command. The index also specifies the pages on which these names occur in the report. This is done by specifying INDEX as a parameter for the command. The commands which allow this parameter are:

- CONTENTS
- DICTIONARY
- EXTENDED-PICTURE
- FORMATTED-PROBLEM-STATEMENT
- PICTURE
- PROCESS-CHAIN
- PROCESS-INPUT-OUTPUT
- STRUCTURE

5.3 The NOSOURCE and XREF Parameters

The NOSOURCE and XREF parameter have the same function for the INPUT-PSL and DELETE-PSL commands as the NOPRINT and INDEX parameter for other URA commands. The report produced by XREF, however, (the URA CROSS REFERENCE LISTING) specifies the lines rather than the pages where the names occur.
6. CONTROL COMMANDS

All control commands are installation-dependent. For this reason all control commands available to a particular installation and the descriptions and usage of these commands are given in Section 6 of Part II.
7. MODIFIER COMMANDS

All the commands in this section modify the URA data base in some manner and generate an output to present the result of the modification. These outputs provide the user with a permanent record of changes made to the problem statement in the data base.

Effect of Modifier Commands on the Data Base Structure

There are basically three types of information stored in a URA data base:

1) Names and types of objects defined by the user
2) Comment entries (narrative and free format descriptions of objects)
3) Connections among objects and between an object and comment entry

All this information is entered into the data base via INPUT-PSL from the URL statements used as input to this command. In most cases the section header statements define the type of objects and names of the objects, comment entry statements (DESCRIPTION, for example) define comment entries to be stored, and other URL statements define relationships or connections among the named objects in the data base. To present the structure of the data base in a graphical manner, the following symbols will be used:

- symbol for record used to describe a given named object (name record)

- symbol for comment entry stored in data base

- symbol for a NUB, a type of record used to make connections among objects and between an object and comment entry
Using the above notation, a simple relationship between two objects (name records) may look like:

Data in the NUB defines the type of connection (RECEIVES, for example) and the direction of the arrows defines the manner in which the relationship should be interpreted, i.e., which object does the RECEIVING.

A connection between an object and a comment entry may look like:

The data in the NUB defines the type of comment entry (PROCEDURE, for example).

It is important to note that the connections made among objects are different from the connection made between an object and comment entry.

INPUT-PSL creates records for named objects, the NUB records connecting the objects, and the comment entries. Commands must also be available to do the following:

1) Change a name record
   i) change the name of the object
   ii) change the object type

2) Delete a connection between objects

3) Delete a name record (and any connections it had with other objects)

4) Change a comment entry

5) Delete a comment entry

URA has facilities to perform all of these modifications on the data base information. The following URL commands perform the actions corresponding to the above:
1) Change a name record:
   1) Change the name of an object:
      RENAME - This command changes information within the name record only.
   11) Change the object type:
      CHANGE-TYPE - This command changes information within the name record only.

2) Delete connections among objects:
      DELETE-PSL - This command deletes NUBS (and thus connections) among name records in the data. It does not delete name records or comment entries.

3) Delete name records:
      DELETE - This command deletes name records. Names to be deleted having connections with other names and/or having comment entries associated with them, will have corresponding NUBS (and comment entries) deleted.

4) Change comment entries:
      REPLACE-COMMENT-ENTRY - This command changes information within the comment entry only.

5) Delete comment-entries:
      DELETE-COMMENT-ENTRY - This command deletes comment entries and also deletes corresponding NUBS.

Modifier Command Description Format

Each Modifier Command will be described in the following format:

Command name (command abbreviation)

When specifying the command to be executed, either the long form of the command name or legal abbreviations of the name may be given.

Modification made

All Modifier commands change information in the URA data base. What type of information is changed and how it is changed is
described. Also, the checks made by URA before the change is made are presented.

Output description

All Modifier commands generate an output to present the result of the modification(s) made. The name of the output, purpose of the output, contents, and diagnostics given in the output are presented to aid in using it.

Execution

The basic form of specifying the command to be executed is described. An example of how this is done and the results from the action are given.

Options and Alternatives

All Modifier commands can be executed in more than one way. For example, a different form of the command may be used to change one name in the data base versus a number of names. Also, the effect which the parameters for the command have on modifying the data base is described. Examples are given to illustrate the alternatives.

Common Errors

It is possible to make particular errors in the use of each Modifier command. Some of the particular logical and syntactical errors that occur when executing the command are given.

The following commands are described in this section in alphabetical order:

7.1 CHANGE-TYPE
7.2 DELETE
7.3 DELETE-COMMENT-ENTRY
7.4 DELETE-PSL
7.5 INPUT-PSL
7.6 PUNCH-COMMENT-ENTRY
7.7 RENAME
7.8 REPLACE-COMMENT-ENTRY
7.1 CHANGE-TYPE (CT)

Modification Made

Each name specified as input to this command has its corresponding name type changed if the new type does not conflict with the context in which the name has previously been used. This modification is most often used to change an undefined name (**UNDEFINED***) to a specific name type (such as GROUP or ELEMENT). Various "checking" facilities must be used to ascertain that legal changes are being made. For each name type change, URA must check to see that:

i) The name whose name type is to be changed exists in the data base.

ii) The assignment of the new name type is consistent with the context in which the name was used previously.

Output description

The output generated by this command is the CHANGE-TYPE REPORT. This report presents for each name used as input to the CHANGE-TYPE command, the name, the old name type associated with it and the new name type now assigned to it. Any error diagnostics which may occur during the name type change will also be printed. The names are printed out in the same order in which they were read as input to the CHANGE-TYPE command.

Execution

To change the name type of only one name, the following command format is issued:

```
CHANGE-TYPE NAME=gross-pay TYPE=ELEMENT
```

Previously in the problem statement, "gross-pay" had been defined as a GROUP. It was more appropriate to call it an ELEMENT, and the CHANGE-TYPE command made it easy to facilitate this change. The resulting CHANGE-TYPE REPORT is shown in Figure 2.

Options and Alternatives

1. The name types of several names can be changed at one time if the names are put into a file and the file is specified as input to the command.* (This is usually done via the FILE parameter.)

* The exact manner in which the file is specified is given in Part II, Section 7.1.
Figure 3 is an output resulting from using a file as input to CHANGE-TYPE. All the names in the example are shown to have been previously undefined (not assigned a name type). The report shows that "date" was changed to a GROUP, "employee-date" became a GROUP, and "valid-t-card" an ELEMENT. Each line of the input file consists of the name of an object followed by the new name type to be assigned to it. The format is acceptable if the name is followed by its new name type and the two are within the first eighty columns of the file line and there is at least one blank between them. The file used to generate Figure 3 was:

```
date  GROUP
employee-data  GROUP
valid-t-card  ELEMENT
```

2. The TYPE parameter can also be used effectively with an input file. All names in the input file will have their name types changed to the name type specified by the TYPE parameter. If the file from the previous example was specified as input and the TYPE parameter was also used, only the names in the file would be read as input; the name types would be ignored. All the names specified in the input file would have their name types changed to that given by the TYPE parameter. Figure 4 presents the output resulting from a CHANGE-TYPE command with TYPE=ELEMENT and the file used as input the same as that used for Figure 3.

3. It is sometimes advantageous to use NAME-GEN in conjunction with CHANGE-TYPE, i.e., use the output produced by NAME-GEN as input to CHANGE-TYPE. This is most often done when all names of a particular type (usually UNDEFINED) are changed to another type (GROUPS or ELEMENTS). To change all undefined names in the data base to GROUPS:

```
NAME-GEN UNDEFINED
CHANGE-TYPE FILE TYPE=GROUP
```

The NAME-GEN command selects all undefined names and places them in a file. The CHANGE-TYPE command then uses the file produced by NAME-GEN as its input file. The TYPE=GROUP parameter specifies that all the names in the input file should be changed to GROUPS. Figure 5 presents the result of this procedure.

Common Errors

If neither an input file nor NAME is specified for the command, the message: "NO NAME GIVEN" will be printed by URA. If a file or NAME is given, but no name types are given in the file or no TYPE is given, the message: "NO TYPE GIVEN WITH "NAME=" OR "FILE" PARAMETER will be printed. Should either of these messages be generated, URA will not execute the CHANGE-TYPE command. The command and its parameters should be reentered with the necessary corrections.
CHANGE-TYPE REPORT

PARAMETERS FOR: CHANGE-TYPE

FILE

1* date
  OLD TYPE = *** UNDEFINED ***
  NEW TYPE = GROUP

2* employee-data
  OLD TYPE = *** UNDEFINED ***
  NEW TYPE = GROUP

3* valid-t-card
  OLD TYPE = *** UNDEFINED ***
  NEW TYPE = ELEMENT

Figure 3

34
PARMETERS FOR: CHANGE-TYPE

FILE TYPE=ELEMENT

1* date
   OLD TYPE - GROUP
   NEW TYPE - ELEMENT

2* employee-data
   OLD TYPE - GROUP
   NEW TYPE - ELEMENT

3* valid-t-card
   OLD TYPE - ELEMENT
   NEW TYPE - ELEMENT
Another common error is attempting to assign a new name type to a name that has previously been used in a way that conflicts with its new name type. For example, if the name XYZ was previously defined to be CONTAINED in SET S1, this would imply that XYZ must be either an ENTITY, INPUT or OUTPUT. If an attempt was made to change its type to a GROUP (which cannot be CONTAINED in a SET) the error message:

URA030:CHKREL:  CONFLICT WITH EXISTING CONNECTIONS

would be given and the change would not be made. If it is still desired that XYZ be a GROUP, XYZ should be deleted from the data base and proper URL statements for XYZ should be entered via INPUT-PSL. All conflicting connections are listed. They all must be resolved before the change of type can occur.
7.2 **DELETE (DEL)**

**Modification Mode**

For each name specified as input to the DELETE command, all its relationships (i.e., USES, SUBPARTS, etc.), with other names in the data base are removed, its comment entries (such as DESCRIPTION or PROCEDURE) are deleted and finally the name is deleted from the data base. Before any of these modifications are made, URA checks that the name to be deleted exists in the data base. If the name cannot be found, no attempt will be made by URA to delete the name.

**Output Description**

The DELETION REPORT is produced each time this command is initiated. Each name used as input to the DELETE command is printed on the report along with the status of the change (i.e., if it did or did not work). The names on the output appear in the same order as read by the DELETE command.

This report serves as a permanent record of names that have been deleted from the URA data base. It is intended to aid the analyst in keeping track of modifications to the data base. Once there is a record of a particular name being deleted, the analyst has the option of re-using the name.

**Execution**

The following command deletes one name from the data base:

```
DELETE NAME=field-check-new
```

The DELETION REPORT for this action is shown in Figure 6.
Options and Alternatives

1. Several names can be deleted from the data base if the names are put into a file and the file is specified as input to the command.* (This is usually done via the FILE parameter.) The format of the input file consists of one name per line, beginning in column one. Figure 7 is an output resulting from using a file as input to the DELETE command. All the names in the report had been defined in the data base. The report shows that deletion of each name was successful.

Common Errors

DELETE should not be used to delete the entire contents of the data base. Operating system command should be utilized for this procedure. The data base should be emptied and then reinitialized.

When doing minor editing of a URL description for a name in the data base the information connected to that name (URL statements) should be saved before deleting the name. A FORMATTED-PROBLEM-STATEMENT for the name should be generated using the PUNCH parameter. Then the name can be deleted (via DELETE) from the data base. At this point, the old information in the PUNCH file can be edited to suit the problem definer and then re-entered via the INPUT-PSL command using the PUNCH file produced by the FPS as input.

If a file is used as input to the command, all names to be deleted must begin in the first column of the file line. Any preceding blanks will be interpreted as part of the name.

If neither an input file nor NAME is specified for the command, the message: "NO NAME OR FILE WAS SPECIFIED" will be printed by URA. Should this happen, URA will not execute the DELETE command. The command and its parameters should be reentered with the necessary corrections.

* The exact manner in which the file is specified is given in Part II, Section 7.2.
DELETION

PARAMETERS FOR: DELETE

FILE

DELETED - check
DELETED - new-employee-printing
DELETED - payrate

Figure 7
7.3 DELETE-COMMENT-ENTRY (DCOM)

Modification Made

The DELETE-COMMENT-ENTRY takes those names specified as input and deletes, for each input name, the comment entries associated with those comment entry types designated as command parameters.* If no comment entry types are specified by the parameters, no comment entries will be deleted. Checking is performed to see if the comment entry exists in the data base before it is deleted. If the comment entry cannot be found, no attempt is made to delete it.

Output description

The output report generated by this command is called DELETED COMMENT ENTRIES. For each comment entry to be deleted from the data base, the following information is printed on the output:

- name in the data base to which the comment entry belonged.
- the type of comment entry (i.e., DESCRIPTION, PROCEDURE, etc.) which is being deleted.
- the full text of the comment entry.

The order of the output names is the same as the order of the input names.

This serves as a hard copy record for those comment entries deleted from the system description. As stated before, it is desirable to have all modifications to the system description documented.

Execution

The following command deletes the PROCEDURE comment entry associated with one name.

```
DCOM NAME=new-info-validation PROCEDURE
```

This was done because the problem definers wanted to delete current PROCEDURE comment entry and did not want to replace it. If the comment entry could be correct if edited or a replacement was available, then it would be more appropriate to use the REPLACE-COMMENT-ENTRY command (see section 7.8). The output generated by this command is shown in Figure 8.

Options and Alternatives

1. Multiple comment entries can be deleted for a name:

```
DCOM NAME=new-info-validation PROCEDURE DESCRIPTION
```

In this case, the PROCEDURE and DESCRIPTION comment entries would be deleted for the PROCESS, "new-info-validation."

* An example of a comment entry type is a URL "DESCRIPTION" or "PROCEDURE" statement. The comment entry associated with this comment entry type would be the text specified by the user for the particular statement.
DELETED COMMENT ENTRIES

PARAMETERS FOR: DCOM

CASE=new-info-validation NODESCRIPTION PROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER
NOVOLATILITY-SET NODERIVATION NOTRUE-WHILE NOFALSE-WHILE PRINT NOPRINT

1* new-info-validation
   PROCEDURE;
   1) read a unit of new employee information
   2) check the ranges of the fields
   3) if: fields correct
      then: add to the data base
   4) else: reject entire unit of information

Figure 8

43
2. The following types of comment entries can be deleted when specified as parameters for DELETE-COMMENT-ENTRY:

   DERIVATION
   DESCRIPTION
   FALSE- WHILE
   PROCEDURE
   TRUE- WHILE
   VOLATILITY
   VOLATILITY-MEMBER
   VOLATILITY-SET

3. Several names can have their comment entries deleted if the names are put into a file and the file is specified as input. The comment entries deleted for these names are those specified as parameters for the command. If the user specifies that the DESCRIPTION and PROCEDURE comment entries be deleted for a GROUP name, the DESCRIPTION comment entry will be deleted and the message:

   URA104:DELS: PROCEDURE COMMENT-ENTRY NOT FOUND

will be given because GROUPS may not have PROCEDURE statements.

   Figure 9 shows the output resulting from executing the DELETE-COMMENT-ENTRY command with a file of names as input and DESCRIPTION and PROCEDURE comment entry types as parameters. The example shows for each name used as input, each comment entry deleted for the name.

4. Printing of the DELETED COMMENT ENTRIES output may be suppressed by specifying NOPRINT as a parameter:

   DCOM NAME=payroll-processing DESCRIPTION NOPRINT

Common Errors

If neither an input file nor NAME is specified for the command, the message "NONAME OR FILE SPECIFIED" will be printed by URA. Should this happen, URA will not execute the DELETE-COMMENT-ENTRY command. The command and its parameters should be reentered with the necessary corrections.
DELETED COMMENT ENTRIES

PARAMETERS FOR: DCOM

DESCRIPTION PROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION NOTRUE-WHILE NOFALSE-WHILE PRINT FILE

1* new-employee-printing
   DESCRIPTION;
   1 this process produces the new hire section of the h-t report

2* new-employee-printing
   PROCEDURE;
   1 1-) accept a valid unit of employee information
   2 2-) save the information
   3 3-) print the new hire section of the h-t report

3* new-info-validation
   DESCRIPTION;
   1 this process accepts correct input information and rejects
   2 the input otherwise.

A104:DELSET: PROCEDURE COMMENT ENTRY NOT FOUND FOR : new-info-validation

Figure 9
7.4 **DELETE-PSL (DPSL)**

**Modification Made**

This command takes as input, any URL statements in the format specified in the "User Requirements Language, Version 3.0 Language Reference Manual."\(^1\) For each section header statement (i.e., PROCESS, DEFINE, etc.\(^2\)), all the URL statements following this section header (up to the next section header statement) will be deleted from the URA data base for those names specified in the header statement. This command only deletes relationships between names and does not delete any comment entries (this is handled by the DCOM command) nor deletes names (this is handled by the DELETE command). If some of the information presented by the URL statements is contradictory, an error message will be given for that statement. Error diagnostics are also given when syntactical errors occur. URA attempts to continue the procedure until too many errors are encountered.\(^3\)

**Output Description**

The two outputs that may be generated by this command are DELETED URL and the URA CROSS REFERENCE. The DELETED URL output is a record of all information (except names and comment entries) deleted from the URA data base. It aids the analyst in finding errors in the deletion procedure and produces error diagnostics in sufficient detail to aid the analyst in correcting these errors. This output displays, line for line, the data used as input to the DELETE-PSL command. No reordering is done on the input data.

The URA CROSS REFERENCE is intended as an aid to the analyst in correcting errors that appear in the DELETED URL output. It consists of an alphabetical list of all user defined names, i.e., non-URL names that appear in the DELETED URL output. For each name that appears in the CROSS REFERENCE, its corresponding name type (as given in the DELETED URL output) is printed and a list of all lines in the DELETED URL output where the name appeared is also given.

**Execution**

The most common method of deleting URL statements is by first writing all statements to be deleted into a file (or punching them on cards) and then using this as input to the command.* (This

---

1 ISDOS Working Paper No. 68.
2 See Appendix F of ISDOS Working Paper No. 68 for a complete list of all possible section header types.
3 See Section 9 for the limit of errors allowed.

* The exact manner in which the file can be specified is given in Part II, Section 7.4.
is usually done via the INPUT parameter.) Only the first 72 columns of each line in the file may contain URL statements. Anything after column 72 is ignored. Figure 10 is the output resulting from this type of procedure. The EOF statement must be given to specify the end of URL statements to be deleted.

Options and Alternatives

1. In many cases when the amount of input is relatively large (a few hundred lines); hence, there may be a need for the URA CROSS REFERENCE. It will be generated by specifying the XREF parameter with the command.

2. If no input file is specified, URA will wait for URL statements to be typed in (from the terminal), or when in batch mode, interpret any following cards up through the first "EOF" as URL statements to be deleted. When URL statements are entered at the terminal, each line entered is echoed back by URA along with any errors encountered for that statement (i.e. an AS-IS SOURCE LISTING). This allows the user to correct errors as they occur.

3. Printing of the DELETED URL output may be suppressed by specifying NOSOURCE as a parameter.

Common Errors

The most common errors are typing errors encountered in interpreting the URL statements. A typing mistake can cause many different types of syntactical and semantic errors.

Only the first 72 columns of each line in the input file are read so all URL statements should fit in this region. Anything over column 72 will be truncated and an error message will be generated in most cases.

Omitting the semicolon at the end of a URL statement is a common cause for several errors.

DELETE-PSL will not delete comment entry statements from the data base so these statements are ignored if encountered in the input file. No names other than SYNONYMS can be deleted from the data base via DELETE-PSL.

The last line of the input file containing the URL statements should have the word EOF signifying the end of input. This should also be typed when inputting the data interactively. EOF allows the return to the URA command handler. No URL statements are read after EOF.
PARAMETERS FOR: DPSL

SOURCE NOXREF

LINE STATE

1 >GROUP:  check;
2 > CONTAINED: pay-statement;
3 > RELATION: comp-pay-info;
4 > SYNONYM: comp-pay-info;
5 > ELEMENT: payrate;
6 > ALPHA: type numeric-information;
7 > VALUES: 3 THRU 100;
8 > EOF
7.5 **INPUT-PSL (IP)**

Modification Made

This command takes as input, any URL statements in the format specified in "User Requirements Language, Version 3.0, Language Reference Manual." For each section header statement (i.e., PROCESS, DEFINE, etc.) the user defined names specified by that section header will be added to the list of names in the data base (if not already in the data base). All the URL statements following this section header up to the next section header statement, specify connections to be made with other names in the data base. URA first performs syntax and semantic checks on each input line before any more complex checking is performed. An "in context" check is made for each name used as input. If the name is not in the user's data base, it is added. If it is, a check is made to see that the context in which the name is used in the new input agrees with the manner in which the name is used in the data base. If there is a conflict, an error message will be produced and URA will skip to the next input statement. URA attempts to continue the input procedure until too many errors are encountered. If redundant information is given, i.e., specifying the same relationship more than once, the redundant information will not be added to the information already in the data base. No diagnostic message is given to denote redundant information.

**Output Description**

The two outputs that may be generated by this command are the URA AS-IS SOURCE LISTING and the URA CROSS REFERENCE. The URA AS-IS SOURCE LISTING is a record of all information input into the URA data base, and is intended as an aid to the analyst. It aids the analyst in finding errors in the input data and produces error diagnostics in sufficient detail to aid the analyst in correcting these errors. The output displays, line for line, the data used as input to the INPUT-PSL command. The order of the input data is not changed.

The URA CROSS REFERENCE is intended as an aid to the analyst in correcting errors that appear in the URA AS-IS SOURCE LISTING and also to resolve ambiguities in assigning name types to the undefined names in the listing. It is useful in correcting errors, as any name involved in an error can be quickly referenced to find all places in the AS-IS LISTING where the name is used, and the name type assigned to that name.

---

1 ISDOS Working Paper No. 68.

2 See Appendix F of Working Paper No. 68 for a complete list of all possible section header types.

3 See Section 9 for the limit of error allowed.
From this information, the analyst will be able to determine what information has to be reentered to correct the error. Since the CROSS REFERENCE also presents all those names which have an ambiguous name type (one that was not defined in previous input), the analyst can resolve those ambiguities by use of the CHANGE-TYPE or INPUT-PSL commands. The output consists of an alphabetical list of all user defined names, i.e., non-URL names, that appear in the AS-IS LISTING. For each name that appears in the CROSS REFERENCE, its corresponding name type (as given in the AS-IS LISTING) is printed and a list of all lines in the AS-IS LISTING where the name appeared is also given.

Execution

The most common method of inputting URL statements is by first writing all statements to be added into a file (or punching them on cards) and then using this as input to the command.* (This is usually done via the INPUT parameter.) Note that only the first 72 columns of each line in the file may contain URL statements. Anything after 72 is ignored. Figure 11 is the output resulting from this type of procedure. The EOF statement must be given to specify the end of URL statements to be added. The UPDATE parameter specifies that the URA data base is to be modified by the input. If this parameter is not given, none of the information will be added to the data base.

Options and Alternatives

1. In many cases, when the amount of input is relatively large (a few hundred lines) there may be a need for the URA CROSS REFERENCE. By simply specifying XREF as a parameter, it will be generated. Figure 12 presents an AS-IS LISTING and CROSS REFERENCE for a small problem statement.

2. In most cases, it is advantageous to first do a syntax and semantic check of the input data before attempting to put it in the data base. By not specifying the UPDATE parameter, these checks will be made without actually putting the information into the data base. This will generate an AS-IS LISTING with error diagnostics for the URL statements used as input. Since most problem statements have one or two typing errors anyway, this proves to be an inexpensive way to catch errors early. After the source of the errors has been determined and corrected, the command can be issued again using UPDATE as a parameter.

3. If no input file is specified, URA will wait for URL statements to be typed in (from the terminal), or when in batch mode, interpret any following cards up through the first "EOF" as URL statements to be added to the data base. When URL statements are entered at the terminal, each line entered is echoed back by URA along with any errors encountered for that statement (i.e., an AS-IS SOURCE LISTING). This allows the user to correct errors as they occur.

* The exact manner in which the file can be specified is given in Part II, Section 7.5.
4. Printing of the AS-IS SOURCE LISTING may be suppressed by specifying NOSOURCE as a parameter.

5. The DBREF parameter allows referencing of the data base when semantic as well as syntax checks are desired for URL statements used as input. This must be in effect when UPDATE is given as a parameter. NO DBREF may be specified if only a syntax check of the statements is desired and the data base is not to be updated.

**Common Errors**

The most common errors are typing errors encountered in interpreting the URL statements. A typing mistake can cause so many different types of syntactical and semantic errors that it will be handled in a later section (Section 10).

It is also possible to input the new information into the wrong data base. It is important that the data base to be used has been specified. Otherwise, the data base may be set to some default which may not be the data base file desired.

The INPUT-PSL command only reads the first 72 columns of each line in the input file so all URL statements must fit in this region. Anything over column 72 will be truncated and an error message will be given in most cases.

Omitting the semicolon at the end of a URL statement is a common cause for several errors. It is important that the syntax of each URL statement be correct.

The last line of the input file containing the URL statements should have the word EOF signifying the end of input. This should also be typed when inputting the data interactively. EOF allows the return to the URA command handler. (See Figures 11 and 12 to see how EOF is used correctly.) No URL statements are read after EOF.
PARAMETERS FOR: SYMU

SOURCE NOXREF UPDATE DBREF

LINE START

<table>
<thead>
<tr>
<th>ID FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 &gt;PROCESS: payroll-processing;</td>
</tr>
<tr>
<td>2 &gt; SYNONYM: payproc,pl;</td>
</tr>
<tr>
<td>3 &gt; DESCRIPTION:</td>
</tr>
<tr>
<td>4 &gt; This process represents the highest level process</td>
</tr>
<tr>
<td>5 &gt; in the target system. it accepts and processes</td>
</tr>
<tr>
<td>6 &gt; all inputs and produces all outputs.;</td>
</tr>
<tr>
<td>7 &gt;</td>
</tr>
<tr>
<td>8 &gt;EOF</td>
</tr>
</tbody>
</table>

Figure 11
AS-IS SOURCE LISTING

PARAMETERS FOR: SYMU

SOURCE XREF UPDATE DBREF

LINE      ID FIELD
          START
1 >/* System Flow 7 */
2 >
3 >INPUT: employee-information;
4 >
5 >OUTPUT: paysystem-outputs;
6 >
7 >SET: payroll-master-information;
8 >
9 >REAL-WORLD-ENTITY: departments-and-employees;
10 >    Generates: employee-information;
11 >    Receives: paysystem-outputs;
12 >
13 >PROCESS: payroll-processing;
14 >    Updates payroll-master-information;
15 >    Receives: employee-information;
16 >    Generates: paysystem-outputs;
17 >
18 >EOF

Figure 12
<table>
<thead>
<tr>
<th>SEQUENCE NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 departments-and-employees</td>
<td>INTERFACE</td>
</tr>
<tr>
<td>2 employee-information</td>
<td>INPUT</td>
</tr>
<tr>
<td>3 payroll-master-information</td>
<td>SET</td>
</tr>
<tr>
<td>4 payroll-processing</td>
<td>PROCESS</td>
</tr>
<tr>
<td>5 paysystem-outputs</td>
<td>OUTPUT</td>
</tr>
</tbody>
</table>

Figure 12 (Continued)
Modification Made

Technically, the output produced by this command is a report presenting narrative information in the manner of a glossary. It makes no modifications to the data base, but is presented here because its main objective is to aid the analyst in changing comment entries in conjunction with the REPLACE-COMMENT-ENTRY command. The idea of using the output as a glossary (for final specifications perhaps) however, should not be overlooked. A message is given when no comment entry is available for a particular comment entry type or the name specified is not in the data base.

Output Description

The PUNCHED COMMENT ENTRIES output is generated by this command. It presents selected comment entries for each name used as input to the command. Any type of name may be used as input to the command. Depending on the type of name the following comment entries may be retrieved:

- DERIVATION (DER)
- DESCRIPTION (DESC)
- FALSE-WHILE (FW)
- PROCEDURE (PRCD)
- TRUE-WHILE (TW)
- VOLATILITY (VOL)
- VOLATILITY-MEMBER (VOLM)
- VOLATILITY-SET (VOLS)

For each name used as input to the command, the name is printed on the output in the order in which it was read and associated with that name, the type of comment entry and the text for that comment entry (for each type of comment entry as specified in the parameter list).

Execution

To obtain the DESCRIPTION comment entry for one name the following command might be given:

```
PCOM NAME=payroll-processing DESCRIPTION
```

This will generate the report shown in Figure 13. A PUNCH file will also be generated with the same information as the report. The manner in which the file to contain the PUNCH data is specified is installation dependent and given in Part II, Section 7.6. If the procedure is done at the terminal, the PUNCH file can then be edited and used as input to the REPLACE-COMMENT-ENTRY command to modify the comment entry. If the procedure is done in batch, the contents of the PUNCH file produced should be punched on cards (by the system if possible). Then the deck of cards produced can be modified and used as input to REPLACE-COMMENT-ENTRY in the next batch run.
PUNCH：PRONETRATIONS

PRAMETERS FOR: PCOM

NAME=payroll-processing DESCRIPTION NOPROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET MODERIVATION NOTRUE-WHILE NOFALSE-WHILE PRINT PUNCH

* payroll-processing

DESCRIPTION:

1  This process represents the highest level process
2  in the target system. It accepts and processes
3  all inputs and produces all outputs.

Figure 13
Options and Alternatives

1. Several names can have comment entries printed and/or PUNCHED if the names are put into a file and the file is specified as input to the command.* (This is usually done via the FILE parameter.) Figure 14 is an output resulting from using a file as input to the PCOM command.

2. Multiple comment entries, such as DESCRIPTION and PROCEDURE, can be generated for several names when a file is specified as input and more than one comment entry type is specified as parameters. This is illustrated in Figure 15.

3. When the objective of executing this command is to generate a PUNCH file, printing of the report may be suppressed by issuing NOPRINT as a parameter.

4. When the objective of executing this command is to generate the report (and no PUNCH data is desired), production of data in the PUNCH file may be suppressed by issuing NOPUNCH as a parameter.

5. The NAME-GEN can also be used in conjunction with PCOM to retrieve all names of a particular name type (such as INTERFACE) to be used as input to the PCOM command. For example:

   NAME-GEN INTERFACE
   PCOM DESCRIPTION

   This procedure retrieves all INTERFACE names defined in the data base and produces the PUNCHED COMMENT ENTRIES report for all these names and their corresponding DESCRIPTIONS. This could also be done for more than one type of name:

   NAME-GEN SET PROCESS
   PCOM DESCRIPTION PROCEDURE

   Notice that the PROCEDURE parameter is given, but SETS cannot have PROCEDURE statements associated with them. Only the DESCRIPTION statements will be retrieved for SET names while both DESCRIPTION and PROCEDURE statements will be retrieved for PROCESS names.

Common Errors

The problem definer should note that most of the parameters indicating comment types (i.e., FALSE-WHILE, VOLATILITY, etc.) can apply to only one type of name (CONDITION, ENTITY, respectively).

* The exact manner in which the file is specified is given in Part II, Section 7.6.
PUNCHED COMMENT ENTRIES

PARAMETERS FOR: PCOM

FILE DESCRIPTION NOPROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION
NOTRUE-WHILE NOFALSE-WHILE PRINT PUNCH

employee

DESCRIPTION:
1 an employee is identified by an employee number.

Time-card

DESCRIPTION:
1 this input contains the information about the hours that an
2 employee worked the preceding week.

Pay-statement

DESCRIPTION:
1 this output is the payment to the employee for the previous
2 weeks work.
PUNCHED COMMENT ENTRIES

PARAMETERS FOR: PCOM

FILE DESCRIPTION PROCEDURE NOVOLATILITY NOVOLATILITY-MEMBER NOVOLATILITY-SET NODERIVATION
NOTRUE-WHILE NOFALSE-WHILE PRINT PUNCH

1× new-employee-printing
   DESCRIPTION;
   1 this process adds information to the h-t report. ;

2× new-employee-printing
   PROCEDURE;
   1 1-) accept a valid unit of new employee information
   2 2-) print the new hire section of the h-t report. ;

3× new-info-validation
   DESCRIPTION;
   1 this process accepts correct input information and rejects
   the input otherwise. ;

4× new-info-validation
   PROCEDURE;
   1 1-) read a unit of new employee information
   2 2-) check the ranges of the fields
   3 3-) if: fields correct
   4    then: add to the data base
   5 else: reject entire unit of information ;

Figure 15
7.7 **RENAME (REN)**

**Modification Made**

The RENAME command takes an old name (of some object in the problem statement data base) and a new name as input. If the new name is not a URL reserved word or a name already in the data base, the command will replace the old name by the new name. Before a name is changed, URA checks that:

- the old name exists in the data base
- the new name is not already used in the data base
- the new name is a legal URL name (see the "User Requirements Language, Version 3.0, Language Reference Manual"*)

If any of these requirements are violated, no change will be made.

**Output Description**

The output generated from this command is called the RENAME REPORT. For every name changed by the RENAME command, this report presents the "old name" which appeared in the data base and the "new name" which has taken its place. When the name change is not successful, error diagnostics are also printed specifying the cause of the error. Again, the names are printed on the output in the same order as they are read as input.

**Execution**

To change one name in the data base, the following command might be given:

```
RENAME OLD=employee-code NEW=employee-number
```

Upon first defining the target system, "employee-code" was used to represent a certain piece of data. Later it was found that this data was actually called "employee-number" and the change was made to be consistent with organization terminology. See Figure 16 for the report generated by this command. This command is also beneficial for changing misspelled names in the data base. Through typing errors, "employee-number" may have gone in as "employee-nuber". This mistake can be corrected by:

```
RENAME OLD=employee-nuber NEW=employee-number
```

* ISDOS Working Paper No. 68.
Options and Alternatives

1. As with most of the modifier commands, the problem definer has the option of changing several names at one time. The old-new name pairs must first be put in a file to be used as input to the command. (This is usually done via the INPUT parameter.) Figure 14 presents the output resulting from this procedure. Each line of the file must consist of an old name followed by the corresponding new name. The two names may be anywhere in the first 80 columns of the line and must be separated by one or more blanks. The format of the file used to produce Figure 17 is given below:

joseph-i-smith  henry-miller
varying-employee-data  changing-employee-data
level-1  11
error-listing  error-list

Common Errors

The most common error in using RENAME is specifying a name already in the data base or a URL reserved word as the new name. URA will not make the change if this is the case. The command would have to be reissued with another new name to take the place of the illegal one.

If neither an input file or an OLD/NEW pair of parameters is specified for the command, the message: "MUST GIVE OLD AND NEW, OR INPUT" will be printed by URA. Should this happen, URA will not execute the RENAME command. The command and its parameters should be reentered with the necessary corrections.
7.8 REPLACE-COMMENT-ENTRY (RCOM)

Modification Made

This command takes as input names which exist in the data base, each followed by a URL comment entry statement. If the comment is a DESCRIPTION comment entry, for example, then the command will replace the old DESCRIPTION comment entry by the one used as input. What RCOM actually does is delete the old comment entry and put the new comment entry in its place. This is done after a check has been made to ascertain that the "old comment entry" exists in the data base and the "new comment entry" is legal for the particular application being used (e.g., not attempting to enter a PROCEDURE comment entry for a SET name). A check is made to see if the input name is in the data base.

If an attempt is made to replace a comment entry for a name that did not have a comment entry specified for it, the message:

URA 126:REPSET: WARNING - THERE IS NO COMMENT ENTRY TO DELETE

will be given and the designated comment entry will be connected to the name.

Output Description

The output generated by this report is called REPLACED COMMENT ENTRIES. For each "old comment entry" to be replaced, the output depicts, in the following order:

- name to which the "old comment entry" belongs
- the type of comment entry which is being changed
- the entire text of the "old comment entry"
- the entire text of the "new comment entry" which replaces the old one

Error diagnostics referring to problems encountered in executing the command are also printed here.

Execution

Any information to be supplied as input to RCOM must first be placed in a file and the file must be designated as input.* (This is usually done via the INPUT parameter.)

---

* The exact manner in which the file is specified is given in Part II, Section 7.8.
The contents of the file must be in the following format:

```
name
  comment entry type;
  
  comment entry
  
  etc.
```

The contents of the file used to produce the output shown in Figure 18 was:

```
new-employee-printing
  DESC;
  This process produces the new hire section of the h-t report;
new-employee-printing
  PRCD;
  1-) accept a valid unit of new employee information
  2-) save the information
  3-) print the new hire section of the h-t report;
```

Options and Alternatives

1. It is often the case that only some minor editing of a comment entry need be done to make it correct. This can be done when the output from the PUNCH-COMMENT-ENTRY command is edited and used as input to the RCOM command. This is described in Section 7.6 of this paper.

2. To suppress printing of the REPLACED COMMENT ENTRIES report the NOPRINT parameter may be specified.

Common Errors

The major problem in using this command is specifying the file format correctly. (This is not a problem, however, if the contents of the file used was produced by PUNCH-COMMENT-ENTRY.) Although the command allows free formatting of the file, the order: name, comment-entry type, comment-entry must be maintained. Each must begin on a new line.
PARAMETERS FOR: HCOM

PRINT

DELETED COMMENT ENTRY **
* new-employee-printing
  DESCRIPTION :
  1 this process adds information to the h-t report.

DELETED COMMENT ENTRY **
* new-employee-printing
  DESCRIPTION :
  1 this process produces the new hire section of the h-t report.

DELETED COMMENT ENTRY **
* new-employee-printing
  PROCEDURE :
  1 1-) accept a valid unit of new employee information
  2 2-) print the new hire section of the h-t report.

INSERTED COMMENT ENTRY **
* new-employee-printing
  PROCEDURE :
  1 1-) accept a valid unit of new employee information
  2 2-) save the information
  3 3-) print the new hire section of the h-t report.

Figure 18
8. REPORT COMMANDS

Report Commands retrieve specific types of information from the database and present it in formats which aid the problem definer to analyze the problem statement for correctness and completeness. Many of the formats can serve as final specifications of the system being designed.

Most report commands allow the report to be generated for a single name (via the NAME parameter) or for a number of names placed in a file and specified as input to the command.*

The descriptions of these report commands, their usage and interpretation, and the usage of reports produced by them are given in "JRA Outputs."**

* The exact manner in which this is done is installation dependent and is given in Part II, Section 8.

** Part III.
URA has extensive checking facilities to prevent errors in the problem statement. At the URA command mode level, checks are made to insure that all commands given are legal URA commands and that all parameters given are legal parameters for that command. If an illegal command, an illegal parameter, or illegal parameter for that command is given when in on-line mode, the following message will be generated:

INVALID PARAMETER - ENTER REPLACEMENT OR BLANK LINE

The user must enter the replacement following the question mark and then hit the carriage return key. If the command is accepted, processing of that command commences. Should an error be encountered while processing the command, one of the following three types of error diagnostics will be given:

i) Data Base Management System Errors

These errors are encountered when there may be some danger of destroying the contents of the URA data base or there is an error in the URA software. Even though the URA software might be the cause of the error, it is doubtful if it will do anything to harm the contents of the user's data base. A complete list of these error messages is given in "A Data Base Management System for URA Based on DBTG 71." *

*** ERROR 16 - DATA BASE FILE INCONSISTENT

This error message is given if the user attempts to modify or retrieve information from a URA data base which has had its contents altered so that it is unusable by the URA software.

**** ERROR # n FOUND IN ROUTINE # m

An error message of this format usually designates an error in the URA software, where n is the error number. To find out which routine the error occurred in, refer to "A Data Base Management System for URA Based on DBTG'71."* If the values of the variables n and m are 16 and 30 respectively, the error designates a data base inconsistency which is usually a user error. Any other errors of this form with different values should be brought to the attention of those persons maintaining the URA software.

ii) URA Command Errors

These errors are encountered in the processing of URA commands and are user errors. These diagnostics are generated when the user presents ambiguous or incorrect information to the URA commands. In most cases, URA will take no action to fulfill the user's request if an error is encountered. The command must be restated in corrected form, before action is taken. All these errors are presented in the following format:
**URA Error Messages**

Where "nnn" designates the URA error number, "subroutine" denotes the subroutine in the URA software where the error occurred, and "error-message" corresponds to some diagnostics which describe the error condition.

### iii) **URA Input Errors**

These errors are encountered when the INPUT-PSL or DELETE-PSL are used incorrectly. URA always attempts to recover from these errors unless an excessive number or errors have been encountered. Each of these errors is assigned a level number, 1 through 4. The user is allowed to make up to 24 level 1 and 24 level 2 errors, but a single level 3 or level 4 error will terminate processing of the command. The levels are described below.

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Warning</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>Serious user error</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>URA unable to recover</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Exceeded URA capabilities</td>
<td>0</td>
</tr>
</tbody>
</table>

These types of errors are presented in the following format:

```
**** LEVEL j, URAnnn:subroutine: error-message
```

where "j" designates the level number and "nnn" denotes the URA error number. The last part of the format is the same as the URA command errors.

After processing any URA command, a STOP status message is given. This message designates that processing of the command was successful (STOP 0, i.e., errors were handled effectively, etc.) or that processing was not totally successful (STOP 4 or STOP 8). STOP 4 is given when an error level limit is exceeded for INPUT-PSL errors, for example. The STOP 8 message designates a serious error in attempting to access the data base, usually resulting from an inconsistent data base.

The following is a list of all possible errors that can be encountered while using URA. A short description of each error accompanies it as well as suggested action to take should the error occur.
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
</table>
| 2      | NLEX:      | NAME TOO LONG  
A user defined name has exceeded the 30 character limit  
allowed by URL/URA. The name is truncated to 30, but  
is still put in the data base. (See Section 10.1,  
part v.) |
| 3      | NLEX:      | 'EOF' NOT FOUND BEFORE END-OF-FILE  
The user has terminated the input before specifying  
the URL 'EOF.' Processing of the input is terminated. |
| 4      | NLEX:      | ILLEGAL CHARACTER - TREATED AS BLANK  
Either an illegal character encountered in an input line  
or legal URL character used in the wrong context. See  
the "User Requirements Language, Version 3.0, Language  
Reference Manual"* for a list of legal characters. The  
illegal character is replaced by a blank and processing  
of the URL statement continues. |
| 5      | NLEX:      | END-OF-FILE IN MIDDLE OF COMMENT  
The end of input has been encountered following the '/*'  
comment characters. Processing of the input is terminated. |
| 6      | SCAN:      | INVALID LEXICAL TYPE RETURNED FROM NLEX  
URA software error. Please notify persons maintaining  
URA should this error occur. |
| 7      | SCAN:      | ILLEGAL CHARACTER - IGNORED  
An illegal character encountered when scanning an input  
line. See the "User Requirements Language, Version 3.0  
Language Reference Manual"* for a complete list of  
legal characters. The illegal character is ignored and  
processing of the URL statement continues. |
| 8      | COMLOP:    | PARSE STACK OVERFLOW  
URA software error. Please notify persons maintaining  
URA should this error occur. |
| 9      | PROK:      | BAD CASE  
URA software error. Please notify persons maintaining  
URA should this error occur. |
| 10     | REDUCE:    | NO APPLICABLE PRODUCTION - SYNTAX ERROR - START SKIPPING  
Illegal URL statement syntax is encountered. If this  
is a header statement, following statements will be  
assigned to the previous header statement. The error may  
be a result of incorrect usage of a URL reserved word.  
(See Section 10.1, part i.) |
| 11     | STACK      | ILLEGAL SYMBOL PAIR - SYNTAX ERROR - START SKIPPING  
Illegal URL statement syntax is encountered. If this  
is a header statement, following statements will be  
assigned to the previous header statement. This state-  
ment is not entered into the URA data base. (See Section  
10.1, part i.) |
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
</table>
| 12     | SYMBOL:    | SYMBOL TABLE OVERFLOW  
Exceeded limits of URA. Reissue INPUT-PSL command at point in the input file where this error occurred. |
| 13     | SYMBOL:    | TOO MANY SYMBOLS  
Exceeded limits of URA. Reissue INPUT-PSL command at point in the input file where this error occurred. |
| 14     | SETYPE:    | INVALID SYMBOL TABLE POINTER  
URA software error. Please notify persons maintaining URA should this error occur. |
| 15     | STACK:     | INVALID CASE  
URA software error. Please notify persons maintaining URA should this error occur. |
| 16     | COMENT:    | END-OF-FILE IN COMMENT ENTRY  
End of input encountered in URL comment entry. Processing of the input is terminated. |
| 17     | SKIP:      | END OF FILE WHILE SKIPPING  
Serious error. In attempt to recover from previous errors the end of input has been encountered. Processing of input is terminated. |
| 18     | IDENT:     | NO NAMES IN DATA BASE  
Attempt to retrieve names from an empty data base. |
| 19     | RECOV:     | UNABLE TO RECOVER AT THIS TIME  
Processing of input is terminated due to serious errors which make it unable to continue. |
| 20     | RECOV:     | LAST STATEMENT SKIPPED  
Statement where error occurred is skipped so that processing of input may continue. |
| 21     | SETINF:    | INVALID SYMBOL TABLE POINTER  
URA software error. Please notify persons maintaining URA should this error occur. |
| 22     | OTHERS:    | SAME ATTRIBUTE ALREADY GIVEN WITH DIFFERENT ATTRIBUTE VALUE  
An attempt was made to assign a second value to the same ATTRIBUTE for a given name. The new value is ignored. |
| 23     | SUBPRT:    | TOO MANY LEVELS - STACK OVERFLOW  
The limit allowed for retrieving names via the SUBPARTS-OF parameter has been exceeded. Reissue the NAME-GEN command with the last name retrieved as the value for the SUBPARTS-OF parameter. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
</table>
| 24     | MAINRCOM:  | MISSING SEMICOLON ON LINE AFTER NAME  
Semicolon is needed to terminate comment entry statement. |
| 25     | HEAD:      | INVALID HEADER STATEMENT - STATEMENTS WILL BE IGNORED  
Illegal syntax of header statement. All URL statements up to the next header statement will be ignored. (This error is also described in section 10.1, part iii.) |
| 26     | IGINFO:    | INVALID SYMBOL TABLE POINTER  
URA software error. Please notify persons maintaining URA should this error occur. |
| 27     | PTABIN:    | INVALID LEXICAL TYPE OR END-OF-FILE  
URA software error. Please notify persons maintaining URA should this error occur. |
| 28     | PTABIN:    | DUPLICATE RESERVED WORD - IGNORED  
URA software error. Please notify persons maintaining URA should this error occur. |
| 29     | CHKREL:    | CONFLICT WITH EXISTING CONNECTIONS (RELA) RELTYP #  
Attempt made to change name type to one which conflicts with the context in which the name is used. No change is made. |
| 30     | CHKREL:    | CONFLICT WITH EXISTING CONNECTIONS (RELB) RELTYP #  
Attempt made to change name type to one which conflicts with the context in which the name is used. No change is made. |
| 31     | MAINCT:    | BAD INPUT FORMAT  
The format of the file used as input to the command is incorrect. See the command description for correct format. No change is made. |
| 32     | MAINCT:    | NAME NOT IN DATA BASE  
Attempt to change name type of name not defined in the URA data base. |
| 33     | MAINCT:    | INVALID NAME TYPE  
Attempt to assign an illegal name type to a name. Probably a spelling error. |
| 34     | MAINCT:    | NAME TYPE TOO LONG  
Attempt to assign an illegal name type to a name. Probably a spelling error. |
| 35     | MAINCT:    | WARNING - STUFF AFTER NAME TYPE  
The input file contains more than just name and new name type. The extra data will be ignored by the command processor. |
| 36     | MAINCT:    | INVALID NAME - TOO LONG  
The name for which the change is to be made is over 30 characters. Check spelling. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>FNDPD:</td>
<td>THIS IS NOT A PD FOR ANY NAMES - This problem definer is not associated to any names defined in the data base.</td>
</tr>
<tr>
<td>38</td>
<td>MAINREN:</td>
<td>OLD NAME NOT IN D.B. Attempt to change name of some object which is not defined in the data base. Probably a spelling error.</td>
</tr>
<tr>
<td>39</td>
<td>MAINREN:</td>
<td>NEW NAME ALREADY IN D.B. Attempt to change old name to a name already defined in the data base. User must choose another name.</td>
</tr>
<tr>
<td>40</td>
<td>CLREN:</td>
<td>MUST GIVE OLD AND NEW, OR INPUT Parameters given for the command do not supply sufficient information for processing. Reissue command.</td>
</tr>
<tr>
<td>41</td>
<td>MAINDEL:</td>
<td>NAME TO BE DELETED NOT IN D.B. Attempt to delete a name which is not defined in the URA data base.</td>
</tr>
<tr>
<td>42</td>
<td>MAINDEL:</td>
<td>INVALID MEMBER TYPE URA software error. Please notify persons maintaining URA should this error occur.</td>
</tr>
<tr>
<td>43</td>
<td>OTHERS:</td>
<td>CARDINALITY ALREADY GIVEN AS SYSPAR Attempt to assign a numerical value to a CARDINALITY statement when previously assigned a SYSTEM-PARAMETER name. The value is ignored.</td>
</tr>
<tr>
<td>44</td>
<td>OTHERS:</td>
<td>CARDINALITY ALREADY GIVEN AS DIFFERENT VALUE Attempt to assign a second value to a CARDINALITY statement. The new value is ignored.</td>
</tr>
<tr>
<td>45</td>
<td>CLCT:</td>
<td>NO TYPE GIVEN WITH &quot;NAME=&quot; OR &quot;FILE&quot; PARAMETER No new name type has been specified. The command must be reissued.</td>
</tr>
<tr>
<td>46</td>
<td>CT:</td>
<td>NO NAME GIVEN No name has been specified to have its name type changed. The NAME or FILE parameter must be given.</td>
</tr>
<tr>
<td>47</td>
<td>MAINNG:</td>
<td>PD NOT FOUND IN DATA BASE The problem definer specified by the PD parameter is not defined in the data base. No names will be generated.</td>
</tr>
<tr>
<td>48</td>
<td>MAINNG:</td>
<td>KEYWORD NOT FOUND IN DATA BASE - The keyword specified by the KEYWORD parameter is not defined in the data base. No names will be generated.</td>
</tr>
<tr>
<td>49</td>
<td>MAINNG:</td>
<td>NO NAMES IN DATA BASE Attempt to retrieve names from an empty data base.</td>
</tr>
<tr>
<td>Number</td>
<td>Subroutine</td>
<td>Error Message</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| 50     | PLIST:     | TOO MANY NAMES - REST IGNORED  
Exceeded 50 name limit. Remaining names should be given in another statement. |
| 51     | RWLIST:    | MUST BE SUBSETTING CRITERION NAME  
Attempt to define a name which is not a GROUP or ELEMENT to be SUBSETTING CRITERION. |
| 52     | IDENTC:    | NAME NOT IN D.B. -  
Attempt to retrieve information about a name which is not defined in the data base. |
| 53     | OPTRW:     | NAME LIST TOO LONG - REST IGNORED  
Exceeded 50 name limit. Remaining names should be given in another statement. |
| 54     | OPTRW:     | NAME LIST TOO LONG - REST IGNORED  
Exceeded 50 name limit. Remaining names should be given in another statement. |
| 55     | OPTRW:     | NAME LIST TOO LONG - REST IGNORED  
Exceeded 50 name limit. Remaining names should be given in another statement. |
| 56     | USEDTO:    | TOO MANY NAMES - REST IGNORED  
Exceeded 50 name limit. Remaining names should be given in another statement. |
| 57     | USEDTO:    | TOO MANY NAMES - REST IGNORED  
Exceeded 50 name limit. Remaining names should be given in another statement. |
| 58     | OPTRW:     | NAME LIST TOO LONG - REST IGNORED  
Exceeded 50 name limit. Remaining names should be given in another statement. |
| 59     | MAINNG:    | SUBPARTS-OF NAME NOT IN DATA BASE -  
Attempt to retrieve names that are a part of a name not defined in the data base. |
| 60     | APPLES:    | SECOND MAILBOX FOR PD ILLEGAL  
Attempt to associate a second MAILBOX to a particular PROBLEM DEFINER. |
| 61     | RWLIST:    | ALREADY PART OF SOMETHING ELSE  
Attempt to define a structure where an object is PART OF more than one other object. This is contrary to the rules specified in the "User Requirements Language, Version 3.0, Language Reference Manual."* |
| 62     | RWLIST:    | SECOND PD FOR THIS ITEM ILLEGAL  
Attempt to assign a second RESPONSIBLE-PROBLEM-DEFINER to an object. This statement is ignored. |

* ISDOS Working Paper No. 68
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
</table>
| 63     | RWLIST:    | ALREADY PART OF SOMETHING ELSE  
Attempt to define a structure where an object is  
PART OF more than one other object. This is contrary  
to rules specified in the "User Requirements Language,  
| 64     | MAINDICT:  | NAME NOT FOUND IN D.B. -  
Attempt to retrieve information about a name that is  
not defined in the data base. |
| 65     | MAINCONT:  | NAME NOT FOUND IN D.B.  
Attempt to retrieve information about a name that is not  
defined in the data base. |
| 66     | MAINPIC:   | NAME NOT IN D.B. -  
Attempt to retrieve information about a name that is not  
defined in the data base. |
| 67     | MAINPIC:   | PICTURE NOT AVAILABLE FOR -  
Attempt to generate the report for a name which is not  
a SET, INPUT, OUTPUT, ENTITY, GROUP, ELEMENT, PROCESS  
or INTERFACE. Only these types of objects may have  
PICTURES generated for them. |
| 68     | REPSET:    | WARNING - MISSING SEMICOLON. NEW COMMENT ENTRY ADDED  
Semicolon not given to terminate comment entry. One  
is assumed and processing continues. |
| 69     | REPSET:    | NO NEW COMMENT ENTRY - OLD ENTRY HAS BEEN DELETED  
Since no new comment entry has been given to replace  
the old, the old comment entry statement is deleted. |
| 70     | CLDCOM:    | NO NAME OR FILE SPECIFIED  
Either the NAME or FILE parameter must be given for  
this command to be executed. Parameters given do not  
supply sufficient information for processing. Reissue command. |
| 76     | MAINPRIO:  | NAME NOT IN DATA BASE -  
Attempt to retrieve information about a name that is not  
defined in the data base. |
| 87     | CLDEL:     | NO NAME OR FILE WAS SPECIFIED  
Either the NAME or FILE parameter must be given for  
this command to be executed. Parameters given do not  
supply sufficient information for processing. Reissue command. |
| 88     | MAINPRIO:  | NAME NOT A PROCESS NAME -  
Attempt to retrieve information from a name that is not  
a PROCESS name. Only PROCESS names may be used as input  
to this command. |
| 89     | MAINNG:    | NAME MUST BE INPUT, OUTPUT, PROCESS, OR INTERFACE FOR "SO"  
PARAMETER -  
Attempt to retrieve SUBPARTS information for a name which  
is not an OUTPUT, INPUT, PROCESS or INTERFACE. Only these  
objects may have "SUBPARTS." |

* ISDOS Working Paper No. 68
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>RWLIST:</td>
<td>SSCN IS ONLY LEGAL TYPE IN DEFINE SECTION WHICH CAN BE MAINTAINED Attempt to use MAINTAINED statement for some object which is not SUBSETTING-CRITERION.</td>
</tr>
<tr>
<td>91</td>
<td>ADDUSE:</td>
<td>TOO MANY USAGES URA software error. Please notify persons maintaining URA should this error occur.</td>
</tr>
<tr>
<td>92</td>
<td>MAINPAV:</td>
<td>NAME NOT IN D.B. - Attempt to retrieve information about name which is not defined in the data base.</td>
</tr>
<tr>
<td>93</td>
<td>MAINPAV:</td>
<td>NAME HAS NO USAGES AS ATTRIBUTE FOR ANYTHING - Attempt to retrieve ATTRIBUTE information for a name which is not an ATTRIBUTE.</td>
</tr>
<tr>
<td>95</td>
<td>CLEI:</td>
<td>MUST GIVE EITHER ENTITY OR IDENTIFIER PARAMETER Either the ENTITY or IDENTIFIER parameter must be used in conjunction with this command for successful implementation.</td>
</tr>
<tr>
<td>98</td>
<td>CONCOL:</td>
<td>NAME NOT IN D.B. - Attempt to retrieve information about a name not defined in the data base.</td>
</tr>
<tr>
<td>99</td>
<td>IDENTR:</td>
<td>NAME NOT IN D.B. - Attempt to retrieve information about a name not defined in the data base.</td>
</tr>
<tr>
<td>100</td>
<td>ATLIST:</td>
<td>TOO MANY ATTRIBUTE VALUE PAIRS IN SINGLE STATEMENT Limit exceeded. Remaining pairs should be given in another statement.</td>
</tr>
<tr>
<td>101</td>
<td>ATLIST:</td>
<td>NAME MUST BE ATTRIBUTE NAME Attempt to use a name defined as something else as an ATTRIBUTE name.</td>
</tr>
<tr>
<td>102</td>
<td>ATLIST:</td>
<td>NAME MUST BE ATTRIBUTE VALUE NAME Attempt to use a name defined as something else as an ATTRIBUTE-VALUE name.</td>
</tr>
<tr>
<td>Number</td>
<td>Subroutine</td>
<td>Error Message</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>103</td>
<td>DELSET;</td>
<td>DESCRIPTION COMMENT ENTRY NOT FOUND FOR ; Attempt to delete a nonexistent DESCRIPTION statement.</td>
</tr>
<tr>
<td>104</td>
<td>DELSET;</td>
<td>PROCEDURE COMMENT ENTRY NOT FOUND FOR : Attempt to delete a nonexistent PROCEDURE statement.</td>
</tr>
<tr>
<td>105</td>
<td>DELSET;</td>
<td>VOLATILITY COMMENT NOT FOUND FOR : Attempt to delete a nonexistent VOLATILITY statement.</td>
</tr>
<tr>
<td>106</td>
<td>DELSET;</td>
<td>VOLATILITY-MEMBER COMMENT ENTRY NOT FOUND FOR : Attempt to delete a nonexistent VOLATILITY-MEMBER statement.</td>
</tr>
<tr>
<td>107</td>
<td>DELSET;</td>
<td>VOLATILITY-SET COMMENT ENTRY NOT FOUND FOR : Attempt to delete a nonexistent VOLATILITY-SET statement.</td>
</tr>
<tr>
<td>108</td>
<td>DELSET;</td>
<td>DERIVATION COMMENT ENTRY NOT FOUND FOR : Attempt to delete a nonexistent DERIVATION statement.</td>
</tr>
<tr>
<td>109</td>
<td>DELSET;</td>
<td>TRUE WHILE COMMENT ENTRY NOT FOUND FOR : Attempt to delete a nonexistent TRUE WHILE statement.</td>
</tr>
<tr>
<td>110</td>
<td>DELSET;</td>
<td>FALSE WHILE COMMENT NOT FOUND FOR : Attempt to delete a nonexistent FALSE WHILE statement.</td>
</tr>
<tr>
<td>111</td>
<td>MAINDCOM:</td>
<td>NAME NOT FOUND IN D.B. : Attempt to delete information for a name not defined in the data base.</td>
</tr>
<tr>
<td>113</td>
<td>CLCM:</td>
<td>MUST GIVE EITHER CONSISTS OR CONTAINED PARAMETER Either the CONSISTS or CONTAINED parameter must be used in conjunction with this command.</td>
</tr>
<tr>
<td>115</td>
<td>VLIST:</td>
<td>MIN NOT LESS THAN MAX - IGNORED If a number range is specified the first number must be less than the second number.</td>
</tr>
<tr>
<td>116</td>
<td>OTHERS</td>
<td>VALUES ONLY LEGAL FOR ELEMENT, SYSPAR, OR ATTRIBUTE-VALUE Attempt to use a VALUES statement for a name which is not an ELEMENT, SYSTEM-PARAMETER or ATTRIBUTE-VALUE.</td>
</tr>
<tr>
<td>117</td>
<td>OTHERS:</td>
<td>DIFFERENT VALUES ALREADY GIVEN Attempt to assign a second value for a given object. This statement is ignored.</td>
</tr>
</tbody>
</table>

* ISDOS Working Paper No. 68
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>118</td>
<td>SYSNL:</td>
<td>INVALID SYSPAR GIVEN Error encountered in using a SYSTEM-PARAMETER in a given statement. Interpretation of rest of statement becomes confused.</td>
</tr>
<tr>
<td>119</td>
<td>SYSNL:</td>
<td>SYSPAR MUST BE GREATER THAN ZERO Attempt to use zero as a SYSTEM-PARAMETER.</td>
</tr>
<tr>
<td>120</td>
<td>SNAMET:</td>
<td>NAME ALREADY USED IN DIFFERENT CONTEXT Attempt to use a name in a context which conflicts in the way it has previously been used.</td>
</tr>
<tr>
<td>121</td>
<td>MAINRCOM:</td>
<td>NAME NOT FOUND IN DATA BASE - Attempt to access information for a name not defined in the data base.</td>
</tr>
<tr>
<td>122</td>
<td>MAINRCOM:</td>
<td>INVALID TYPE OF COMMENT ENTRY - Attempt to replace unrecognizable comment entry statement. Probably a spelling error.</td>
</tr>
<tr>
<td>123</td>
<td>MAINRCOM:</td>
<td>CANNOT HAVE THIS TYPE OF COMMENT ENTRY - Attempt to assign a comment entry statement which is not legal for the particular name type.</td>
</tr>
<tr>
<td>124</td>
<td>REPSET:</td>
<td>...WITH THIS NAME - Used in conjunction with URA123. Specifies the name for which the comment entry was used.</td>
</tr>
<tr>
<td>125</td>
<td>MAINRCOM:</td>
<td>PROBLEMS SCANNING INPUT FILE - MUST ABORT Incorrect format of file used for input. See command description for correct format.</td>
</tr>
<tr>
<td>126</td>
<td>REPSET:</td>
<td>WARNING - THERE IS NO COMMENT ENTRY TO DELETE Attempt to delete nonexistent comment entry.</td>
</tr>
<tr>
<td>127</td>
<td>PUNSET:</td>
<td>DESCRIPTION COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent DESCRIPTION statement.</td>
</tr>
<tr>
<td>128</td>
<td>PUNSET:</td>
<td>PROCEDURE COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent PROCEDURE statement.</td>
</tr>
<tr>
<td>129</td>
<td>PUNSET:</td>
<td>VOLATILITY COMMENT NOT FOUND FOR: Attempt to retrieve nonexistent VOLATILITY statement.</td>
</tr>
<tr>
<td>130</td>
<td>PUNSET:</td>
<td>VOLATILITY-MEMBER COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent VOLATILITY-MEMBER statement.</td>
</tr>
<tr>
<td>131</td>
<td>PUNSET:</td>
<td>VOLATILITY-SET COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent VOLATILITY-SET statement.</td>
</tr>
<tr>
<td>Number</td>
<td>Subroutine</td>
<td>Error Message</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>132</td>
<td>PUNSET:</td>
<td>DERIVATION COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent DERIVATION statement.</td>
</tr>
<tr>
<td>133</td>
<td>PUNSET:</td>
<td>TRUE WHILE COMMENT ENTRY NOT FOUND FOR: Attempt to retrieve nonexistent TRUE WHILE statement.</td>
</tr>
<tr>
<td>134</td>
<td>PUNSET:</td>
<td>FALSE WHILE COMMENT NOT FOUND FOR: Attempt to retrieve nonexistent FALSE WHILE statement.</td>
</tr>
<tr>
<td>135</td>
<td>MAINPCOM:</td>
<td>NAME NOT FOUND IN D.B.: Attempt to retrieve information for a name not defined in the data base.</td>
</tr>
<tr>
<td>141^</td>
<td>CONROW:</td>
<td>NAME NOT IN D.B. - Attempt to retrieve information for a name not defined in the data base.</td>
</tr>
<tr>
<td>142</td>
<td>BETWEN:</td>
<td>THE TWO NAMES ARE NOT CONNECTED IN THAT FASHION Attempt to delete a relationship, between two names, which is not defined in the data base.</td>
</tr>
<tr>
<td>143</td>
<td>MAINDP:</td>
<td>NAME NOT IN D.B. - Attempt to retrieve information about a name which is not defined in the data base.</td>
</tr>
<tr>
<td>144</td>
<td>DPCOL:</td>
<td>TOO MANY COLUMNS Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.</td>
</tr>
<tr>
<td>145</td>
<td>DPCOL:</td>
<td>TOO MANY ROWS Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.</td>
</tr>
<tr>
<td>146</td>
<td>DPCOL:</td>
<td>SPARSE MATRIX SYSTEM OVERFLOW Exceeded limits of the software that produces the matrix.</td>
</tr>
<tr>
<td>147</td>
<td>SPROW:</td>
<td>TOO MANY ROWS Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.</td>
</tr>
<tr>
<td>148</td>
<td>DPROW:</td>
<td>TOO MANY COLUMNS Exceeded limits of the software that produces the matrix. Names omitted from the matrix should be used as input to another DP command.</td>
</tr>
<tr>
<td>149</td>
<td>DPROW:</td>
<td>SPARSE MATRIX SYSTEM OVERFLOW Exceeded limits of the software that produces the matrix.</td>
</tr>
<tr>
<td>Number</td>
<td>Subroutine</td>
<td>Error Message</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>150</td>
<td>DPSUM:</td>
<td>NO ROWS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No relationships can be specified about the names used as input, so no matrix will be generated.</td>
</tr>
<tr>
<td>151</td>
<td>DPSUM:</td>
<td>NO COLUMNS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No relationships can be specified about the names used as input, so no matrix will be generated.</td>
</tr>
<tr>
<td>152</td>
<td>DPSUM:</td>
<td>SPARSE MATRIX SYSTEM OVERFLOW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Exceeded limits of the software that produces the matrix.</td>
</tr>
<tr>
<td>153</td>
<td>MAINDP:</td>
<td>INVALID INPUT NAME TYPE -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attempt to use a name which is not a PROCESS name as input to the command.</td>
</tr>
<tr>
<td>154</td>
<td>MAINDP:</td>
<td>INVALID INPUT NAME TYPE -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attempt to use a name which is not a SET, INPUT, OUTPUT, ENTITY, GROUP or ELEMENT name as input to the command.</td>
</tr>
<tr>
<td>155</td>
<td>DPSUM:</td>
<td>INVALID ROW TYPE - SYSTEM ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>URA software error. Please notify persons maintaining URA should this error occur.</td>
</tr>
<tr>
<td>156</td>
<td>CNTAND:</td>
<td>NAME NOT IN DATA BASE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attempt to delete a relationship which has not been defined in the data base.</td>
</tr>
<tr>
<td>157</td>
<td>BETWEN:</td>
<td>NAME NOT IN DATA BASE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attempt to delete a relationship which has not been defined in the data base.</td>
</tr>
<tr>
<td>158</td>
<td>UDDERS:</td>
<td>NAME NOT IN DATA BASE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attempt to delete a relationship which has not been defined in the data base.</td>
</tr>
<tr>
<td>159</td>
<td>UDDERS:</td>
<td>NAME NOT IN DATA BASE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attempt to delete a relationship which has not been defined in the data base.</td>
</tr>
<tr>
<td>160</td>
<td>UDDERS:</td>
<td>NAME NOT IN DATA BASE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attempt to delete a relationship which has not been defined in the data base.</td>
</tr>
<tr>
<td>161</td>
<td>UDDERS:</td>
<td>NO CONNECTIVITY EXISTS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attempt to delete a relationship which does not exist for this name.</td>
</tr>
<tr>
<td>162</td>
<td>UDDERS:</td>
<td>DIFFERENT CONNECTIVITY IN DATA BASE - NOT DELETED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attempt to delete a relationship which is not stated exactly as it is in the data base.</td>
</tr>
<tr>
<td>Number</td>
<td>Subroutine</td>
<td>Error Message</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| 163    | UDDERS:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 164    | UDDERS:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 165    | UDDERS:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 166    | UDDERS:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 167    | UDDERS:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 168    | UDDERS:    | DIFFERENT VALUES IN DATA BASE - NOT DELETED  
Attempt to delete a number or range of numbers that was not defined for the statement. |
| 169    | UDDERS:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 170    | DELSYN:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 171    | DELSYN:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 172    | DELSYN:    | NAME IS NOT A SYNONYM FOR THIS NAME  
Attempt to delete a SYNONYM relationship which is not defined in the data base. |
| 173    | DRIVES:    | USES INFORMATION NOT IN DATA BASE  
Attempt to delete a relationship which is not specified exactly as it is in the data base. Relationship is not deleted. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
</table>
| 174    | DRIVES:    | WARNING - "USING" INFO IN DATA BASE  
Statement deleted although the relationship has not been specified exactly as in the data base (the "USING" clause has been omitted). |
| 175    | DRIVES:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 176    | DRIVES:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 177    | DRIVES:    | THESE TWO NAMES NOT CONNECTED IN THAT WAY  
Attempt to delete a relationship which is not defined in the data base. |
| 178    | Happns:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 179    | DRIVES:    | "USES" NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 180    | Happns:    | NAMES NOT CONNECTED  
Attempt to delete a relationship which is not defined in the data base. |
| 181    | DISCON:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 182    | DISCON:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
| 183    | DISCON:    | NAMES NOT CONNECTED  
Attempt to delete a relationship which is not defined in the data base. |
| 184    | ATVLST:    | NAME DOESN'T HAVE THIS ATTRIBUTE  
Attempt to delete a relationship which is not defined in the data base. |
| 185    | ATVLST:    | NAME NOT IN DATA BASE  
Attempt to delete a relationship which is not defined in the data base. |
<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATVLST:</td>
<td>NAME HAS NO ATTRIBUTES Attempt to delete ATTRIBUTE relationship for a name with no ATTRIBUTES.</td>
</tr>
<tr>
<td>NOCNST:</td>
<td>SYSPAR VALUE IN DATA BASE IS DIFFERENT - IGNORED Attempt to delete a statement using a different SYSTEM-PARAMETER. Statement not deleted.</td>
</tr>
<tr>
<td>NOCNST:</td>
<td>WARNING - SYSPAR IN DATA BASE Statement deleted though user did not include SYSTEM-PARAMETER with statement.</td>
</tr>
<tr>
<td>NOCNST:</td>
<td>NAME NOT IN DATA BASE Attempt to delete a CONSISTS relationship using a name not defined in the data base.</td>
</tr>
<tr>
<td>NOCNST:</td>
<td>CONSISTS/CONTAINED INFORMATION NOT IN DATA BASE Attempt to delete a CONSISTS or CONTAINED relationship not defined in the data base.</td>
</tr>
<tr>
<td>NOCNST:</td>
<td>NO SYSPAR IN DATA BASE - IGNORED Attempt to delete a relationship which is not defined exactly in the same way as defined in the data base. Statement not deleted.</td>
</tr>
<tr>
<td>CONN:</td>
<td>NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.</td>
</tr>
<tr>
<td>CONN:</td>
<td>RELATION HAS NO CONNECTIVITY Attempt to delete a CONNECTIVITY relationship for a name with no CONNECTIVITY statements associated with it.</td>
</tr>
<tr>
<td>CONN:</td>
<td>DIFFERENT CONNECTIVITY IN DATA BASE Attempt to delete a CONNECTIVITY relationship not defined exactly as is in the data base. Statement not deleted.</td>
</tr>
<tr>
<td>SYSVAL:</td>
<td>NAME NOT IN DATA BASE Attempt to delete a relationship which is not defined in the data base.</td>
</tr>
<tr>
<td>PLONG:</td>
<td>COMMENT NOT FOUND *** SYSTEM ERROR *** URL software error. Please notify persons maintaining URL should this error occur.</td>
</tr>
<tr>
<td>COMNT:</td>
<td>COMMENT-ENTRIES NOT ALLOWED IN DPSL Attempt to delete comment-entry statements. This can only be done using the DCOM and RCOM commands. Statement not deleted.</td>
</tr>
<tr>
<td>Number</td>
<td>Subroutine</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
</tr>
</tbody>
</table>
| 198    | COMNT:     | EOF WHILE LOOKING FOR SEMICOLON   
Improper statement syntax has been encountered. 
Semicolons are needed to end all URL statements. |
| 199    | HAPPNS:    | DIFFERENT SYSPAR - NOT DELETED   
Attempt to delete a HAPPENS relationship using a 
different SYSTEM-PARAMETER than defined in the data base. Statement not deleted. |
| 200    | HAPPNS:    | INTERVAL NOT IN DATA BASE   
Attempt to delete a HAPPENS relationship using an 
INTERVAL not defined in the data base. |
| 201    | PLIST:     | NAME NOT PART OF HEADER   
An illegal statement header has been given. Probably 
a spelling error. The statement is ignored. |
| 202    | NLIST:     | NAME PREVIOUSLY USED DIFFERENTLY - IGNORED   
Attempt to use a name in a context different than the 
way it is defined. (This error is also described in section 10.1.) |
| 203    | DRIVES:    | "USING" NAME NOT IN DATA BASE   
Attempt to delete a relationship not defined in the 
data base. |
| 204    | DEFN:      | INVALID NAME TYPE   
Attempt to assign a name type to a name which is used 
in a different context. |
| 205    | SETSYN:    | ALREADY SYNONYM FOR SOMETHING ELSE   
Attempt to assign a name to be a SYNONYM for more than one 
object. |
| 206    | SETSYN:    | UNABLE TO MAKE SYNONYM - TOO COMPLICATED   
See Section 10.1 part vii for explanation and solution to this 
error. |
| 207    | SETSYN:    | CANNOT BE MADE SYNONYM - DIFFERENT TYPES   
Attempt to assign a name as a SYNONYM to a different 
name, both with different name types. |
| 209    | SYSNL:     | NAME MUST BE INTERVAL   
Attempt to use a name which is not an INTERVAL in a 
CONSISTS statement for an INTERVAL section. |
| 210    | SYSNL:     | INVALID NAME TYPE   
Attempt to use a name in a context different than the 
way the name is defined. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
</table>
| 211    | OTHERS:    | NAME MUST BE ENTITY NAME  
Attempt to use a name in a context where only an ENTITY name is acceptable. |
| 212    | OTHERS:    | RELATION ALREADY EXISTS BETWEEN TWO OTHER ENTITIES  
Attempt to specify the same RELATION for a different pair of ENTITIES. Different ENTITY pairs imply different RELATIONS. |
| 213    | OTHERS:    | CAN ONLY HAVE ONE CARDINALITY  
Attempt to specify a second CARDINALITY statement for a name. Objects may have only one CARDINALITY. |
| 214    | OTHERS:    | CONNECTIVITY ALREADY GIVEN FOR THIS RELATION  
Attempt to specify a second CONNECTIVITY statement for a name. RELATIONS may have only one CONNECTIVITY. |
| 215    | OTHERS:    | ALREADY CONTAINS WITH DIFFERENT SYSTEM PARAMETER  
Attempt to specify the same CONSISTS statement, but with two different SYSTEM-PARAMETERS. |
| 216    | OTHERS:    | NAME MUST BE ENTITY NAME BEFORE VIA  
Attempt to use a name in a statement where only an ENTITY name is allowed. |
| 217    | OTHERS:    | NAME MUST BE RELATION AFTER VIA  
Attempt to use a name in a statement where only a RELATION name is allowed. |
| 218    | OTHERS:    | RELATION ALREADY EXISTS BETWEEN DIFFERENT ENTITY PAIR  
Attempt to specify the same RELATION for a different pair of ENTITIES. Different ENTITY pairs imply different RELATIONS. |
| 219    | OTHERS:    | NAME MUST BE CONDITION  
Attempt to use a name in a statement where only a CONDITION name is allowed. |
| 220    | SETSYN:    | CANNOT MAKE A NAME A SYNONYM OF ITSELF  
Attempt to specify a basic name as a synonym for itself. Basic names cannot also be synonyms. |
| 221    | NLIST:     | TOO MANY NAMES - REST IGNORED  
Attempt to specify a list of names for a statement where only a single name is acceptable. |
| 222    | RWLIST:    | NAME MUST BE GROUP OR ELEMENT  
Attempt to use a name in a statement where only a GROUP or ELEMENT name is acceptable. |
| 223    | RWLIST:    | NAME MUST BE SET, ENTITY, GROUP, ELEMENT, OR INPUT  
Attempt to use a name in a statement where only a SET, ENTITY, GROUP, ELEMENT, or INPUT is acceptable. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
</table>
| 224    | RWLIST:    | IDENTIFIER MUST BE GROUP OR ELEMENT  
|        |            | Attempt to use a name in a statement where only a GROUP or ELEMENT name is acceptable. Only GROUPS and ELEMENTS may be used as IDENTIFIERS. |
| 225    | RWLIST:    | CANNOT HAVE KEYWORD FOR KEYWORD  
|        |            | Attempt to assign a KEYWORD to a KEYWORD name. |
| 226    | RWLIST:    | ONLY RELATIONS AND SEC'S CAN BE MAINTAINED  
|        |            | Attempt to use a URL statement in the wrong context. |
| 227    | RWLIST:    | PD CANNOT BE RESPONSIBLE FOR PD  
|        |            | Attempt to assign a RESPONSIBLE-PROBLEM-DEFINER to a PROBLEM-DEFINER name. |
| 228    | RWLIST:    | CANNOT HAVE SECURITY FOR SECURITY  
|        |            | Attempt to assign a SECURITY statement to a SECURITY name. |
| 229    | RWLIST:    | CANNOT HAVE SOURCE FOR SOURCE  
|        |            | Attempt to assign a SOURCE to a SOURCE name. |
| 230    | RWLIST:    | SSC MUST BE SSC, GROUP, OR ELEMENT  
|        |            | Attempt to define a name, which is not a GROUP or ELEMENT, as SUBSETTING-CRITERIA for a SET name. |
| 231    | RWLIST:    | SYNONYMS ONLY APPLIED TO FIRST NAME  
|        |            | Attempt to assign a SYNONYM to more than one name. The SYNONYM is given only to the first name. |
| 232    | APPLES:    | APPLIES STATEMENT ILLEGAL WITH THIS NAME TYPE  
|        |            | Attempt to use APPLIES statement for a name which is not a KEYWORD, MAILBOX, SECURITY or SOURCE. |
| 233    | DEFN:      | TOO MANY NAMES IN DEFINE HEADER - REST IGNORED  
|        |            | Exceeded 50 name limit, remaining names should be given in another statement. |
| 234    | OPRTRW:    | NAME MUST BE PROCESS  
|        |            | Attempt to use a name in a wrong context. Only a PROCESS name can be used in this context. |
| 235    | OPRTRW:    | NAME MUST BE ELEMENT, GROUP, ENTITY, OR SET  
|        |            | Attempt to use a name in wrong context for an UPDATES relationship. |
| 236    | OPRTRW:    | MUST BE ELEMENT, GROUP, INPUT, ENTITY, OR SET  
<p>|        |            | Attempt to use a name in wrong context for a USES or USING relationship. |</p>
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
</table>
| 237    | USEDTO:    | MUST BE PROCESS NAME  
Attempt to use a name in a wrong context. Only a PROCESS name can be used in this context. |
| 238    | USEDTO:    | MUST BE ELEMENT, GROUP, ENTITY, OUTPUT, OR SET  
Attempt to use a name in the wrong context for a DERIVES relationship. |
| 239    | USEDTO:    | MUST BE ELEMENT, GROUP, ENTITY, OR SET  
Attempt to use a name in the wrong context for an UPDATE relationship. |
| 240    | APPLES:    | KEYWORD CANNOT APPLY TO KEYWORD  
Attempt to use the APPLIES statement in the wrong context. |
| 241    | APPLES:    | MAILBOX CAN ONLY APPLY TO PD  
Attempt to use the APPLIES statement in the wrong context. |
| 246    | APPLES:    | SECURITY CANNOT APPLY TO SECURITY  
Attempt to use the APPLIES statement in the wrong context. |
| 247    | APPLES:    | SOURCE CANNOT APPLY TO SOURCE  
Attempt to use the APPLIES statement in the wrong context. |
| 248    | APPLES:    | MEMO CANNOT APPLY TO MEMO  
Attempt to use the APPLIES statement in the wrong context. |
| 249    | APPLES:    | INVALID SECTION - WOOPS  
URA software error. Please notify persons maintaining URA should this error occur. |
| 251    | SNAMET:    | ATTEMPT TO CHANGE TYPE WHEN ALREADY TYPED  
URA software error. Please notify persons maintaining URA should this error occur. |
| 252    | SETSYN:    | SYNONYM TABLE OVERFLOW  
Exceeded URA limits. The user should reissue INPUT-PSL command at point in the input file where this error occurred. |
| 253    | RWLIST:    | INVALID STATEMENT NUMBER  
URA software error. Please notify persons maintaining URA should this error occur. |
| 254    | NAMDBK:    | CANNOT CREATE SYNONYM  
URA software error. Please notify persons maintaining URA should this error occur. |
| 256    | CONTND:    | INVALID SECTION  
URA software error. Please notify persons maintaining URA should this error occur. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>257</td>
<td>CONTND:</td>
<td>MUST BE GROUP, INPUT, OUTPUT, OR ENTITY Attempt to use the CONTAINED relationship in a wrong context.</td>
</tr>
<tr>
<td>258</td>
<td>CONTND:</td>
<td>MUST BE SET Attempt to use the CONTAINED relationship in a wrong context. INPUTS, OUTPUTS and ENTITIES can only be CONTAINED in a SET.</td>
</tr>
<tr>
<td>259</td>
<td>CONTND:</td>
<td>MUST BE GROUP, INPUT, OUTPUT, OR ENTITY Attempt to use a name in the wrong context for a CONTAINED relationship.</td>
</tr>
<tr>
<td>263</td>
<td>OPTRW:</td>
<td>MUST BE ELEMENT, GROUP, OUTPUT, ENTITY OR SET Attempt to use a name in wrong context for a DERIVES relationship.</td>
</tr>
<tr>
<td>264</td>
<td>SYNTH:</td>
<td>NAME MUST BE SYSPAR Attempt to use a name, defined to be something else, as a SYSTEM-PARAMETER.</td>
</tr>
<tr>
<td>265</td>
<td>HAPENS:</td>
<td>SAME THING, SAME INTERVAL, DIFFERENT SYSPAR Attempt to specify same relationship between two INTERVAL names though with different SYSTEM-PARAMETER. Not allowed.</td>
</tr>
<tr>
<td>268</td>
<td>USEDTO:</td>
<td>NAME MUST BE SET, ENTITY, GROUP, ELEMENT, OR INPUT Attempt to use a name in the wrong context for a USES relationship.</td>
</tr>
<tr>
<td>269</td>
<td>NLIST:</td>
<td>NAME LIST TOO LONG, REST IGNORED Limit of 50 names has been exceeded. Remaining names should be given in another statement.</td>
</tr>
<tr>
<td>270</td>
<td>INPAR:</td>
<td>ERROR OPENING DATA BASE - MUST ABORT Attempt to use an inconsistent data base. No processing can be done on it. (See Section 10.1.)</td>
</tr>
<tr>
<td>271</td>
<td>MAINCNC:</td>
<td>NAME NOT IN D.B. - Attempt to retrieve information for a name not defined in the data base.</td>
</tr>
</tbody>
</table>

* ISDOS Working Paper No. 68.
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
</table>
| 272    | CNCBLD:    | TOO MANY ROWS - STOPPING HERE  
Exceeded limits of software that produces the matrix.  
Names omitted from the matrix should be used as input to another CNC command. |
| 273    | CNCBLD:    | NAME DOESN'T CONSIST OF ANYTHING -  
No information can be presented for this name in the matrix. |
| 274    | CNCBLD:    | TOO MANY LEVELS - LOWER LEVEL STUFF IGNORED  
Too many levels of CONSISTS information to be presented. |
| 275    | CNCBLD:    | ***THE FOLLOWING NAMES ARE INVOLVED IN A LOOP:  
This problem should be corrected by modifying the CONSISTS statements for these names. |
| 276    | CNCBLD:    | TOO MANY LEVELS - LOWER LEVEL STUFF IGNORED  
Too many levels of CONSISTS information to be presented. |
| 277    | CNCBLD:    | TOO MANY COLUMNS - STOPPING HERE  
Exceeded limits of software that produces the matrix.  
Names omitted from the matrix should be used as input to another CONSISTS-COMPARISON command. |
| 278    | CNCBLD:    | SPARSE MATRIX OVERFLOW - STOPPING HERE  
Exceeded limits of software that produces the matrix. |
| 279    | CNCSUM:    | ***NO COLUMNS, OR NO ROWS - STOPPING  
No relationships can be specified about the names used as input, so no matrix will be generated. |
| 280    | CNCSUM:    | LESS THAN 2 ROWS, NO SIMILARITY MATRIX  
Not enough information is available to generate a matrix. |
| 281    | CNCSUM:    | SPARSE MATRIX OVERFLOW - STOPPING  
Exceeded limits of software that produces the matrix. |
| 282    | MUST:      | STACK OVERFLOW - CONTINUING  
Exceeded limits of software that produces the report.  
An attempt is made to recover and process as much data as possible. |
| 283    | HAVE:      | STACK OVERFLOW - CONTINUING  
Exceeded limits of software that produces the report.  
An attempt is made to recover and process as much data as possible. |
| 284    | MAINSTR:   | TOO MANY LEVELS - CONTINUING  
Exceeded limits of software that produces the report.  
An attempt is made to process as much data as possible. |
<table>
<thead>
<tr>
<th>Number</th>
<th>Subroutine</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>285</td>
<td>ERRPS:</td>
<td>THE FOLLOWING NAMES ARE INVOLVED IN LOOPS Through incorrect specification of PARTS/SUBPARTS statements, loops have been implied in structures of objects in the problem statement. The user should determine which PARTS/SUBPARTS relationships should be changed and delete them via the DELETE-PSL command.</td>
</tr>
<tr>
<td>286</td>
<td>FRINTV:</td>
<td>NO FREQUENCY INFORMATION IN DATA BASE No HAPPENS statements have been specified in the problem statement stored in the data base. If any output is desired from this report, at least one HAPPENS statement must be in the data base.</td>
</tr>
<tr>
<td>287</td>
<td>MAINIDX:</td>
<td>NO NAMES IN INDEX Attempt to generate an index into a report presenting no information about any names. This is merely a warning.</td>
</tr>
<tr>
<td>288</td>
<td>STATPS:</td>
<td>NO NAMES AT LEVEL ONE Attempt to generate STRUCTURE report for names of particular name type (i.e., PROCESS, INPUT, OUTPUT, or INTERFACE), but no names of this type currently exist in the data base. To generate this report for PROCESS names, for example, at least one PROCESS must be defined in the data base.</td>
</tr>
<tr>
<td>289</td>
<td>PCLRBT:</td>
<td>NO PICTURE AVAILABLE FOR Attempt to generate a PICTURE for names which legally have a PICTURE, but no information that was specified for this name can be presented in PICTURE format. Information that may be presented in a PICTURE is any dealing with interaction of data and PROCESSES and structure (CONSISTS and SUBPARTS statements).</td>
</tr>
<tr>
<td>290</td>
<td>SETSYN:</td>
<td>NAME ALREADY USED IN DIFFERENT CONTEXT Attempt to assign a name of UNDEFINED name type as a SYNONYM to another name which has been used in some context in conflict with the manner in which the UNDEFINED name has been used.</td>
</tr>
<tr>
<td>301</td>
<td>IDENTR:</td>
<td>***TOO MANY COLUMNS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another ENTITY-IDENTIFIER command.</td>
</tr>
<tr>
<td>302</td>
<td>IDENTR:</td>
<td>***TOO MANY ROWS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another ENTITY-IDENTIFIER command.</td>
</tr>
<tr>
<td>303</td>
<td>IDENTR:</td>
<td>***MATRIX OVERFLOW -- MUST STOP HERE *** Exceeded limits of software that produces the matrix.</td>
</tr>
<tr>
<td>Number</td>
<td>Subroutine</td>
<td>Error Message</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>---------------</td>
</tr>
<tr>
<td>304</td>
<td>IDENTR:</td>
<td>THE FOLLOWING NAMES DO NOT IDENTIFY ANYTHING: No information can be presented in the matrix for these names because they do not &quot;IDENTIFY&quot; any ENTITIES.</td>
</tr>
<tr>
<td>305</td>
<td>IDENTC:</td>
<td>*** TOO MANY COLUMNS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another ENTITY-IDENTIFIER command.</td>
</tr>
<tr>
<td>306</td>
<td>IDENTC:</td>
<td>*** TOO MANY ROWS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another ENTITY-IDENTIFIER command.</td>
</tr>
<tr>
<td>307</td>
<td>IDENTC:</td>
<td>*** MATRIX OVERFLOW - MUST STOP HERE *** Exceeded limits of software that produces the matrix.</td>
</tr>
<tr>
<td>308</td>
<td>IDENTC:</td>
<td>THE FOLLOWING NAMES ARE NOT IDENTIFIED BY ANYTHING: No information can be presented in the matrix for these names because no &quot;IDENTIFIED&quot; relationships have been specified for them.</td>
</tr>
<tr>
<td>309</td>
<td>CONROW:</td>
<td>*** TOO MANY COLUMNS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CONSISTS-MATRIX command.</td>
</tr>
<tr>
<td>310</td>
<td>CONROW:</td>
<td>*** TOO MANY ROWS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CONSISTS-MATRIX command.</td>
</tr>
<tr>
<td>311</td>
<td>CONROW:</td>
<td>THE FOLLOWING ARE NOT CONTAINED IN ANYTHING: No information can be presented in the matrix for these names because no &quot;CONTAINED IN&quot; relationships have been specified for them.</td>
</tr>
<tr>
<td>312</td>
<td>CONROW:</td>
<td>*** MATRIX OVERFLOW -- MUST STOP HERE *** Exceeded limits of software that produces this matrix.</td>
</tr>
<tr>
<td>313</td>
<td>CONCOL:</td>
<td>*** TOO MANY COLUMNS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CONSISTS-MATRIX command.</td>
</tr>
<tr>
<td>314</td>
<td>CONCOL:</td>
<td>*** TOO MANY ROWS -- MUST STOP HERE *** Exceeded limits of software that produces the matrix. Names omitted from the matrix should be used as input to another CONSISTS-MATRIX command.</td>
</tr>
<tr>
<td>315</td>
<td>CONCOL:</td>
<td>THE FOLLOWING DO NOT CONSIST OF ANYTHING: No CONSISTS statements have been used in conjunction with the names listed.</td>
</tr>
<tr>
<td>Number</td>
<td>Subroutine</td>
<td>Error Message</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>---------------</td>
</tr>
</tbody>
</table>
| 316    | CONCOL:    | *** MATRIX OVERFLOW -- MUST STOP HERE ***  
Exceeded limits of software that produces this matrix. |
| 317    | CHKREL:    | NAME ALREADY USED IN DIFFERENT CONTEXT  
Attempt to use a name in a context different from  
its initial context. |
10. HOW TO CORRECT ERRORS

Once error situations are detected, there must be some method to deal with them. When the errors are caused by problems in generating a report, no action need be taken as no harm will come to the data base. The Report Command can simply be restated in correct format to solve the problem. If, however, an error is encountered in making modifications to the data base (via Modifier Commands) then some immediate action should be taken if the problem definer desires to maintain a correct and complete problem statement.

The errors discovered in making modifications to the data base can be "Input Errors" which are errors discovered by URA in its attempt to process the information needed to update the data base. All these errors are specified by one or more URA error messages. The majority of these errors occur in the process of using the INPUT-PSL command.

The errors discovered in the problem statement by the problem definer are called "Logical Errors." No error diagnostics are generated by URA to denote that an error has occurred. If a name was misspelled in the input information used for INPUT-PSL, the name could be legal by URL/URA conventions yet not correct from the problem definer's standpoint. "BATCH" and "BATHC" are both names that would be perfectly acceptable to URA but not to the problem definer.

The following two sections deal with aiding the problem definer in correcting both Input Errors and Logical Errors should they occur. Treatment of the error correction methodology is still at a cursory level and no attempt is made to present procedures to correct all possible errors.

10.1 Input Errors

As stated before, all input errors cause URA error diagnostics to be printed. There are a few classes of errors which happen again and again and so will be described below.

Inconsistent Data Base

This error is usually identified by getting the URA error: "URA270: INPAR: ERROR OPENING DATA BASE." This error might occur after issuing a URA Modifier or Report Command and it specifies that the contents of the file being used as a URA data base cannot be accessed by the URA software. Methods for correcting this situation are installation-dependent since it involves the manner in which files are created and initialized. See Section 10.1 of Part II for solutions to this problem at a particular installation.

URA Statement Errors

These errors account for the majority of the errors encountered when inputting information into the data base via INPUT-PSL. These errors are caused by improper use of URL statements according to the rules specified in the "User Requirements Language, Version
3.0, Language Reference Manual."* An occurrence of any of these errors results in the statement, where the error occurred, being ignored by the system.

The "$" character printed by URA is usually fairly close in pointing out where the error occurred. Some of the more common errors (and solutions) are presented here in hopes that the users will be able to apply the methods of solving these errors to their own, specific needs.

i) Syntax Errors

These errors are often encountered through misspellings, improper format of the statement or improper usage of URL reserved words. URA usually generates either of the two error messages:

URA010: REDUCE: NO APPLICABLE PRODUCTION-SYNTAX ERROR-START SKIPPING

or,

URA011: STACK: ILLEGAL SYMBOL PAIR-SYNTAX ERROR-START SKIPPING

For example, if upon misspelling the RECEIVES statement:

RECIEVES FOLDER-A,FOLDER-B;

URA will react by printing the URA010 error message and skip that statement to go on to the next. There are some further problems that can then occur. If the error occurred in a header statement, such as PROCESS, then the header statement is skipped and all statements intended to be related to the header statement will be related to the previous header statement. When a reserved word is misspelled URA has no way of knowing if the statement was to be a header statement or not. Take the example:

GROUP: G1;
CSTS: E1, E2, G2;
PROCESS: P1;
RCVS: I1, I2;
SUBPARTS P2, P3;
ELEMENT E1, E2;

PROCESS has been misspelled which results in having that header statement skipped. All the statements following this header are related to the previous header which leads to more problems since statements which can only be associated to a PROCESS are being attributed to a GROUP name. More errors will occur from this resulting; hence, the PROCESS, RCVS and SUBPARTS statements will not be entered into the data base.

To correct

* ISDOS Working Paper No. 68.
this error, the statements that were omitted could be entered by another INPUT-PSL command. A far more serious problem occurs if the "previous" header was also a PROCESS. For example:

```
PROCESS PX;
RCVS: I1,I3;
GENS: 01,02;
PROCESSEX P1;
RCVS I1,I2;
SUBPARTS P2,P3;
ELEMENT E1,E2;
```

If this were the case, then only one error would be caught by URA (the misspelled "PROCESS") and the following RCVS and SUBPARTS statements would be attributed to PROCESS PX. If this mistake were discovered, the user would have to delete the two statements from PX and then reinput the information for P1:

```
DELETE-PSL
PROCESS PX;
RCVS I1,I2;
SUBPARTS P2,P3;
EOF
INPUT-PSL UPDATE
PROCESS P1;
RCVS I1,I2;
SUBPARTS P2,P3;
EOF
```

ii) **Illegal Statement**

This error is designated by the URA error:

```
URA266:ILLST:  ILLEGAL STATEMENT IN THIS SECTION
```

This error can be caused simply by using a statement that is not allowed for that particular section. Using a CONSISTS statement in a PROCESS section would obviously generate this error. The other case occurs when an error is made in a header section statement and all the following statements might be incompatible with the previous header section name. Whenever this error is encountered, the statement is not put into the data base.

iii) **Illegal Header Statement**

If an error occurs in a header statement and URA Is able to identify it as a header statement, the following error will be given:

```
URA025:HEAD:  INVALID HEADER STATEMENT-STATEMENTS WILL BE IGNORED
```

This means that all the statements up to the next header statement will be ignored and not input into the data base.
All the statements ignored must be reinputted using another INPUT-PSL command to be put into the data base.

iv) Input Line Too Long

If a number of URL statements are used on one line of the input file or if the URL statement is very long, it may run over the 72 column restriction. Should this occur, usually URA010 or URA011 will be generated specifying that improper syntax has been encountered. Note that no error message is generated for the fact that the statement runs over the 72 column restriction. Errors are encountered because anything over column 72 is ignored. Therefore, names may be truncated or a semicolon lost.

v) Name Too Long

It is an easy thing to mistake a 31 or 32 character word for 30 characters. Names longer than 30 characters are caught by URA and flagged by the error:

URA 022:NLEX: NAME TOO LONG

The statement that used the name is still entered into the data base but the name is stored in a truncated form in the data base. If the truncated form of the name is not satisfactory, it is a simple matter to change the name via the RENAME command.

vi) Using URA Reserved Words Incorrectly

Most syntax errors are fairly easy to detect; a misspelled word, improper format, etc., but one of the hardest to detect is the improper use of a URL reserved word. For example, the following statement would be flagged by a URADIO or URIAU error message as having a syntax error.

ATTRIBUTE TYPE A;

The letter "A" happens to be a URL optional word and cannot be used as a user-defined name. Detecting these reserved words can get trickier than this, however, as the statement:

PROCESS D,F,G,K;

seems correct, but "F" is the abbreviation of the URL reserved word "FALSE." The key to finding these errors is to watch where the "$" character is printed by URA. It is usually printed directly after the location of the source of the error. The statement is ignored should this type of error occur and the only solution is to reinput the data using a different name. A list of all URL reserved words is given in Appendix B of the "User Requirements Language, Language Reference Manual."*

* ISDOS Working Paper No. 68.
vii) **Synonym Too Complicated**

This error is specified by the **URA** error:

**URA206:SETSYN: UNABLE TO MAKE SYNONYM-TOO COMPLICATED**

This is caused by specifying various relationships about two names and then attempting to make one a SYNONYM of the other. The problem lies in combining these relationships. The statements:

```
GROUP G1;
USED BY P2;
PROCESS LONG-PROCESS-NAME;
SUBPARTS P3,P4;
SYNONYM P2;
```

will generate the error. P2 is implicitly defined to be a PROCESS just in the context in which it is used in the second statement; it also has a relationship with G1. Now LONG-PROCESS-NAME is defined and has relationships formed with P3 and P4. In the last statement, an attempt was made to make P2 and LONG-PROCESS-NAME the same PROCESS and the error is generated. The whole problem could have been avoided if the user had maintained the convention of issuing SYNONYM statement directly after the header statement as shown below:

```
GROUP G1;
USED BY P2;
PROCESS LONG-PROCESS-NAME;
SYNONYM P2;
SUBPARTS P3,P4;
```

If the statements had been inputted in this manner, LONG-PROCESS-NAME would not have had any relationships formed with other names (P3 and P4 in the previous example) and the assignment of P2 as a SYNONYM would have been successful.

Since the error does occur, there exists a method of correcting this problem:

1. Retrieve all information for one of the names via the PUNCH parameter for the FPS command.
2. Delete the name for which the information was retrieved from the data base.
3. Alter the PUNCH information so that all the information now pertains to the name still in the data base.
4. Enter the modified PUNCH information as input to the INPUT-PSL command.
In this way, all information about P2 is given to LONG-PROCESS-NAME and P2 is assigned as a SYNONYM if future references to the name are necessary. It is much easier to maintain the convention of assigning SYNONYMS directly after the header statement.

viii) Names Used in Wrong Context

This type of error accounts for the majority of the error messages presented in Section 9.

URA202:NLIST: NAME PREVIOUSLY USED DIFFERENTLY-IGNORED

and,

URA222:RWLIST: NAME MUST BE GROUP OR ELEMENT

are examples of diagnostics presented for this type of error. The statement will be ignored and the only way to resolve the problem is to reinput the information in an acceptable format.

ix) Breaking Section/Statement Rules

Several error messages can be generated by attempting to break the rules set forth in the "User Requirements Language, Version 3.0, Language Reference Manual," for statements within a particular section. In using the PART statement, for example, an object may be PART of only one object and failure to comply with this rule will result in:

URA061:RWLIST: ALREADY PART OF SOMETHING ELSE

or some analogous URA error message. These error checks are made to enforce the rules set forth in the "Language Reference Manual" and ensure that the problem statement is still meaningful. Other messages presented for this type of error are:

URA214:OTHERS: CONNECTIVITY ALREADY GIVEN FOR THIS RELATION

or,

URA060:APPLES: SECOND MAILBOX FOR PD ILLEGAL

If the user wishes to replace the information stated in the data base, e.g., replace the MAILBOX for a problem definer, the relationship should be deleted via DELETE-PSL and then the correct information should be inputted using the INPUT-PSL command.

* ISDOS Working Paper No. 68.
10.2 Logical Errors

These errors occur when inputting information into the data base (as input errors do), but no diagnostics are given in the AS-IS SOURCE LISTING. These errors might be detected by scanning the complete list of names in the data base (NAME-GEN) and the complete problem statement (FORMATTED-PROBLEM-STATEMENT). These errors can also be detected when reviewing the contents of any of the other reports available on URA.

Misspelled Names

A simple spelling error can result in two names which look very similar, but which are treated as two different objects in the data base.

For example, if the name, "CALENDAR-DAY" was used to specify a particular INTERVAL in the data base and then "CALENDAR-DAYS" is used in the statements:

```
INTERVAL:  CALENDAR-week;
CONSISTS:  7 CALENDAR-DAYS;
```

the two names become completely different objects (to URA). URA does not know that the two are the same object and it is up to the user to detect and correct this mistake which can be done in the following manner.

1. Retrieve all information for one of the names via the PUNCH parameter for the FPS command.

2. Delete the name for which the information was retrieved from the data base.

3. Alter the PUNCH information so that all the information now pertains to the name still in the data base.

4. Enter the modified PUNCH information as input to the INPUT-PSL command.

The effect of doing this for the previous example would be that

i) All information given about CALENDAR-DAYS is transferred to CALENDAR-DAY and then CALENDAR-DAYS is deleted from the data base. If it is desirable to use the plural form of the name in the data base then it should be a SYNONYM, this can be done in a DESIGNATE statement:

```
INPUT-PSL
DESIGNATE CALENDAR-DAYS SYNONYM CALENDAR-DAY;
E0F
```
ii) Since names can consist of letters or numbers, another common misspelling error is to substitute the letter "0" for the number "0". It is often very difficult to detect this and so there appear to be two names, spelled exactly the same in the data base. This can be corrected in the same way as the previous problem.

iii) When the spelling error only involves one name (if TIME-CARD was spelled TIMECARD in all instances) then this problem could easily be solved by using the RENAME command:

```
RENAME 0=TIMECARD  N=TIME-CARD
```

If both TIME-CARD and TIMECARD are defined in the data base, then the same procedure used to change CALENDAR-DAYS must be performed.

Redundant Objects

Another error which occurs quite frequently is to define one object by two different names, not realizing that they are representing the same thing. EMPLOYEE-RECORD and EMPLOYEE-DATA may be defined separately in the data base, but represent the same thing. To resolve this redundancy, the information for the two names must be combined. This can be done in the same manner as given for correcting the misspelled names (involving two names).

Missing Semicolons

Most often, a missing semicolon will be detected as a syntax error (as described in Section 10.1). There is one particular case where a missing semicolon would not generate any error message:

```
PROCESS P1;
DESCRIPTION;
    THIS IS A DESCRIPTION COMMENT ENTRY THAT IS MISSING THE SEMICOLON.
RCVS I1,I2;
GENS 01,02;
```

What happens here is that the RCVS statement becomes part of the DESCRIPTION comment entry. A semicolon was omitted in terminating the lines intended to be the comment entry, but URA simply searches for the first semicolon to signify the end of the comment entry. To solve this problem the DESCRIPTION statement must be replaced and the RCVS statement must be added to the data base. This can be accomplished by the following procedure.

1. Generate the incorrect comment entry in the form of PUNCH information (via the PUNCH-COMMENT-ENTRY command).
2. Alter the PUNCH information so that the comment entry is correct.
3. Use the modified PUNCH information as input to the REPLACE-
COMMENT-ENTRY command.

4. Add the RCVS statement via an INPUT-PSL command.

Correctness and Completeness

For the most part, it is up to the problem definer to maintain
correctness of the problem statement and URA maintains correctness
of the data base. The problem definer has the ability to do this
through usage of the DELETE-PSL and INPUT-PSL commands. Completeness
can also be determined by the problem definer or improved
through use of the INPUT-PSL command.
Part II

USAGE of the USER

REQUIREMENTS ANALYZER

under MULTICS
1. INTRODUCTION

URA extracts information from URL statements and stores it in a URA data base. Once this information (a requirements statement) is in the data base, it can be modified, new information can be added to it, and reports can be generated presenting the status of the requirements statement. These actions are implemented by the URA commands available in the URA processing mode. This mode of operation may be attained by accessing the URA software available under Multics. Therefore, by understanding the Multics commands that aid in interacting with URA, and the URA command language, the URA user can effectively manipulate the contents of a URA data base.

This paper specifies those Multics commands commonly used in interacting with URA and how to use them.* Multics and URA commands can only be used in their respective processing modes.** Multics commands can be used from the time of logging on to Multics, to the time access is made to the URA software. Once the user enters URA mode, only URA commands may be used until the URA user terminates processing to be done in URA mode (through use of the URA "stop" command). Multics commands can then be issued until the user either logs out of Multics, or re-enters URA mode. This interaction between Multics and URA modes is better illustrated by Figure 19.

The format of this paper serves an important purpose. The first five sections deal with Multics and URA at an introductory level. Section 2 presents necessary information on the basic use of Multics. Section 3 explains the procedure of accessing URA once on Multics. Sections 4 and 5 present practical concepts and conventions which aid in using URA under Multics. Once access to URA has been achieved, sections 6, 7 and 8 present the manner in which Multics interacts with the various commands. Several examples are given in those sections in order to better illustrate the results of specific implementations. Section 9 deals with using Multics to handle errors encountered in the use of URA.

2. Using Multics

For an introduction to Multics, see the Multics User's Guide, Honeywell Order Number AL40. For a more detailed explanation of Multics facilities, see the Multics Programmers Manual (MPM).

3. The URA Environment

Assuming that the user is already logged in to Multics, a URA data base with the name "psabb.dbf" may be initialized by the following command:

* For a better understanding of the whole Multics system and commands available, see the Multics Programmer's Manual, the Multics User's Guide (Honeywell Order Number AL40) and any documentation provided by your local installation.

** A processing mode is defined by the software system in control when any type of command is issued. The debug facility, the editor, URA, etc., all define processing modes and thus, a particular set of
Figure 19. Interaction Between Multics and URA Processing Modes
exec_com >udd>project>personid>dbin  psadb  >udd>project>personid>psa

This will create an empty initialized data base of the default size in the current working directory. To create a data base with a non-default size see Appendix B for creating data bases.

Once a data base has been initialized, the user may enter URA mode with the Multics commands:

exec_com >udd>project>personid>ura

This command is used whenever the user wishes to enter URA mode, whereas a data base need only be initialized at the beginning of a project. Should a user desire to expand or reorganize his data base, the dump/restore programs described in Appendix C may be used.

4. Specifying Input to URA Commands

This section specifies information relevant to using and manipulating Multics segments to be used as input files to URA commands. The "input" and "file" parameters of URA commands may be used with standard Multics segments or input may be specified as coming from the terminal in an interactive mode.

4.1 Entering Data into a Data Set

Using one of the Multics editors, the user can create segments to be used with the "input" and "file" parameters. Such segments may contain names, URL statements, etc.

If the user has URL statements in the segment named mos.input.psl in the current working directory, he may use it as input for the ip command as follows:

ip  input=mos.input.psl  update

4.2 Specifying input data interactively

It is usually advantageous to enter data into segments before it is used as input to URA as the user may correct errors and omissions without difficulty using the Multics editor. There are times when the user may wish to enter data directly to the Analyzer. In such cases, the user specifies "term" instead of the segment name. For example, to provide input via the terminal for the ip command, use:

ip  input=term  update

5. Receiving Output from URA Commands

This section specifies information relevant to using and manipulating Multics segments to be used as output for URA commands. In Multics, output files are specified in the output parameter of the URA set command, and via the punch parameter for other URA commands.
5.1 Using the Output Parameter

The output parameter in the URA set command allows the problem definer to specify where all reports generated by URA commands are to be printed. If nothing is specified, all output is sent to the terminal. To specify that output is to be sent to a segment:

```
set output=segment-name
```

From this point, all reports generated by URA will be written into this segment. The report output may be reassigned to the terminal by:

```
set output=term
```

5.2 Using Punch Segments

For some commands, a punch file is produced. If the user does not explicitly assign the punch file, a default segment name will be used. These default segment names are given in Appendix E.

6. Control Commands

These commands control the operation of URA in some way without actually causing any output themselves.

6.1 Set Command

The most common use of this command is to change the default data base, or to reroute the report output. To change the data base to be used on subsequent commands:

```
set db=data-base-segment-name
```

If the user has a non-default size data base, he may wish to use the dbt parameter:

```
set dbt=table-segment-name
```

6.2 Stop Command

This command return to Multics command mode. All parameters changed via the URA set command return to their default values.
7. Modifier Commands

change-type
delete
delete-comment-entry
delete-psl
input-psl
punch-comment-entry
rename
replace-comment-entry

8. Report Commands

consist-comparison
consist-matrix
data-process
dictionary
entity-identifier
extended-picture
formatted-problem-statement
kwic
picture
print-attribute-values
process-chain
process-input-output
punch-comment-entry

9. Error Conditions

In addition to error comments generated by the URA system, there are occasional errors associated with the interaction between the URA system and Multics.

9.1 Initial messages

When the user first enters URA mode, he will receive messages from Multics that "io-switch does not exist." These may be ignored.

9.2 Abnormal Termination

If any of the commands is unable to terminate in the normal manner, the Fortran monitor may be entered before URA has had a chance to close all files. The Fortran monitor will ask the user if these files are to be closed. In all cases, the user should answer "yes."
9.3 Data Base Already Open

It is possible for the user to get into a state where URA cannot open his data base because it thinks that it is already open. If this happens, the user should return to Multics mode, then issue the Multics command new_process.
Part III

URA OUTPUTS
1. INTRODUCTION.

Once a URL description of an information processing system has been entered into a URA data base the user has the option of retrieving the stored information in several different standard formats called URA reports. Each URA report has particular characteristics with respect to its purpose, the amount of retrieval and analysis required to generate the report, the information presented in the report, the format, etc. In this sense, each report can be classified by its characteristics to provide an overall description of the report and to aid in determining how the report may be used to aid problem definers in checking the validity of the URL description and to improve on its completeness.

Only the reports generated by report commands will be presented in this paper. The reports generated by modifier commands are described in the "User Requirements Analyzer, Part I. The manner of specifying input to report commands is given in Section 4 of Part I and Section 4 of Part II. The manner of receiving output from report commands is given in Section 5 of Part I and Section 5 of Part II.

Section 2 of Part III presents the objectives of PSA reports with respect to those relationships to the logical system design process and their advantages in the system documentation procedure. Section 3 describes the manner in which information is extracted from a URA data base to produce reports.

Section 4 presents several categories for classifying and describing URA reports based on contents, format and usage. Section 5 consists of descriptions of each of the standard URA reports available in Version A2.1 of URA.
2. OBJECTS OF REPORTS FROM A URA DATA BASE

2.1 Purpose of URA Reports

The purpose of URA Reports is to present information retrieved from a URA data base (which contains the description of a particular system) in a format which is useful to persons who are documenting the target system, who desire to understand the target system, or who are involved in the design of the target system.

This requires that the reports must be of various formats, contain various types and levels of information, and be oriented at the various types of user's.

With respect to those persons documenting the target system, the reports must present information resulting from modifications to the URA data base. (These reports are presented in "User Requirements Analyzer, Part I"") In addition, reports must present the status of the URA data base after modifications have been made (i.e., successful modifications). Those reports are basically the same as those used by people who desire to understand the target system. They present selected portions of the information in the data base in various formats. A few particular reports may also be used to aid those persons designing the target system. They usually present the results of extensive analysis on information in the data base.

2.2 Relation of URA Reports to Logical System Design

There are two major objectives in logical system design:

- To produce a proposed system that is the best possible in terms of what it will cost to build, cost to operate and what it will contribute to the organization.

- To minimize the cost and time to produce this "optimum" target system.

The goal of developing computer-aided methods for use in logical system design is to contribute to the above objectives. At the present time, it is not possible to achieve an optimum solution for both of these objectives. One contribution that can be made by a computer-aided methods is to improve the "quality" of the description of the target system. Quality is defined in terms of consistency, unambiguity and completeness.

Consistency means that no statements made in the description contradict, or are incompatible with, other statements and
any particular object is referred to by the same name throughout the description.

Unambiguity means that statements and relationships are made so precisely that interpretation is uniform by all readers.

Completeness means that all necessary relationships are given and no objects have been omitted from the description.

The quality objectives can be aided at three levels:

1. URA will enforce consistency and unambiguity through the syntax analysis and reference checks made when data is entered into the URA data base.

2. The URA reports will make it easier for the problem definer to detect logic errors in the problem statement, unresolved conditions, etc.

3. Some reports are available to the problem definer to aid in detecting incompleteness and inconsistencies of the problem statement.

Therefore, the first objective in logical system design can be attained by improving the quality of the documentation.

The second objective, that of minimizing design cost and time, is aided by transferring much of the clerical workload to the computer. The Analyzer URA maintains an up-to-date record of all information collected. The preparation of this information for use by analysts,* management, etc., can be readily retrieved at request.

To meet these objectives, URA offers three classes of outputs:

- Reports to aid the analyst
- Reports to aid the project management
- Reports to aid the designer
- Final specifications

Reports to aid the analyst are basically those which aid in resolving inconsistencies, ambiguities, and incompleteness in the logical system design problem statement.

* NOTE: Throughout this paper, the terms analyst and problem definer are used synonymously. The problem definer is a person responsible for writing a system description (or part of one) in URB. A URA user is a person who uses the URA software to update or retrieve information from a URA data base.
The reports of concern to project management pertain to status of the project in the form of amount of information entered into the URA data base, etc. The reports which are of benefit to the designer are those which reflect inconsistencies and incompleteness in the problem statement which must be resolved in order to develop a design for the proposed system. They also present information in a manner that optimal or feasible designs may be formulated.

The final specifications are the end result of the logical system design phase using URL/URA. They express all the information in the URA data base in an easy-to-read format. These specifications consist of a series of URA reports generated in a particular order and format.

2.3 Advantages of Using URA Reports

In contrast to manually produced documentation, (i.e., handwritten or typed, the URA reports have several advantages with respect to maintainability, format, usefulness, etc.).

Maintainability

All changes made to the URL description of an Information Processing System are made via the URA data base. All documentation (URA reports) produced from the information in the data base after a change is up-to-date. There is no need to change previous documentation.

Changes made to manually maintained descriptions usually require modification of all existing sets of documentation, and modification of each portion of the documentation affected by the change. This process can cause serious errors in the documentation or require extensive rewriting (and retyping) of the previous documentation.

Format

Each URA report is formatted according to the purpose of the report, taking into consideration the orientation of persons intended to use the report. Therefore, information consisting of many complex relationships may be presented in a graphical format, to make the information easier to interpret. Likewise, a matrix format may be used to present a large amount of information at one time. Formats are standardized so that interpretation is uniform by all users.

Formats of manually produced documentation are often not standardized leading to problems and conflicts in interpretation. Presentations of large amounts of information on a few pages is often a problem because of difficulties in completion and keeping the information up-to-date.
Usefulness

Each URA report has been designed for a particular purpose, i.e., to meet the system documentation needs of one or more users. In an effort to maintain this, all information in the reports is presented in a well-structured manner and the reports are designed to be consistent in their presentations.

Many manually produced documents are done ad hoc with possibly no serious considerations on how the documents may be used to benefit the system building process. In addition, the documentation is often difficult to use because of inconsistencies in the manner in which information is presented and seemingly lack of organization.

Availability

Documentation can be produced anytime (on request) once information has been stored in the URA database. This reduces the lag time usually encountered in the manual production of documentation. This permits important decisions to be made when the problem is encountered rather than "next week, when all the information is available." Physical production of the reports is very fast since this is usually done using a line printer or terminal.

Too often the documentation is produced after the fact. This is especially common if the people are very technically inclined. They would rather sit at the terminal or write program code rather than write documentation. Therefore, when documentation is needed, little is available or the organization of the information makes it very difficult to use. Physical production of the documentation then requires many hours of manual labor behind pencils and typewriters excluding the time and costs of reproducing the original. (It is important to note that in most instances that documentation produced in this way is usually out-of-date by the time it leaves the typewriters.)

Selectivity

URA reports may be generated containing all the information known about the target system as containing specific information about one particular name relevant to the description. This capability allows the user to see only what is desired rather than getting too much or too little.

It would be virtually impossible to incorporate all possible combinations of information in a manually produced system in an effort to anticipate any type of request. Therefore, the description of the target system is presented in a select number of ways and any information not directly available must be desired from the available information (assuming it is in some way desirable).
Analysis

URA offers reports which aid in checking completeness and consistency of the problem statement as it exists in the URA data base. Many of the reports present information in a manner which allows visual-analysis of the information to check for these qualities. Other reports incorporate these checks as part of the information presented in the report and is referred to as computer-aided analysis. (Visual analysis implies that the checks are made by the user where computer-aided analysis is performed by the computer.)

In most forms of manually produced documentation the formats used make it difficult to check for completeness and inconsistencies in the documentation. Inconsistencies, in particular, go undetected very easily as a result of various spellings of the same name in the description. Where one person may know that all the versions of the name refer to the same thing, others may not.

Extensions

In addition to the standard reports available in a particular version of URA, the users may write their own programs which generate reports to present information particular to their applications of URA.* The new reports can be incorporated into any future documentation packages.

Manually produced documentation may incorporate complex reports involving a great amount of computation and analysis in its production, but this requires the same amount of computation and analysis anytime the report is to be presented. Rather than consuming the analyst's time in producing the report, it is more feasible to let the computer do it (and in less time).

* The procedures necessary to write these programs and which effectively interface with the URA data base will be given in a forthcoming paper.
3. GENERATION OF REPORTS

All URA reports are produced by issuing commands to URA. All commands available for URA and the reports generated by each are given in "User Requirement Analyzer Command Descriptions".* Descriptions of each report and the name(s) of the command(s) used to generate it are given in section 5 of this paper.

3.1 URA Command Language for Reports

The programs which are initiated by a particular report command, retrieve information from the URA data base and output it in some meaningful format. No modifications are made to the information in the data base; their sole function is to retrieve the information and display it in some manner. In the process of collecting and displaying the information analysis may be done and the results printed as part of the report.

Most report commands have one or more parameters that may be used in conjunction with the command for one of the following purposes:

- To specify data to be used as input to the command (Input data parameters).
- To specify how data used as input is to be interpreted (Input control parameters).
- To specify what options the user has in generating the report with respect to content (Output option parameters).
- To specify what options the user has in generating the report with respect to format (Output format parameters).

The manner of specifying commands and their parameters is given in Part IV and the manner of using the commands and parameters is given in Parts I and II.
3.2 Retrieval of Information from the URA Data Base

There are basically three types of information stored in a URA data base:

1) Names and types of objects defined by the user.
2) Comment entries (narrative and free format descriptions of objects).
3) Connections among objects and between an object and comment entry.

The first type of information is stored as name records, the second as comment entry records, and the last as NUB records.*

Each URA report presents information taken from one or more of these different types of records. Table 2 gives the information presented by each report. In addition each URA report may also present information derived from the information in one or more of these different types of records.

The manner in which the information is presented differs from one report to another. The relationship between two name records may be either implied by the report format or explicitly defined. For example, the FORMATTED PROBLEM STATEMENT would present that "employee-address" CONSISTS of "street", "city", "state" and "zip-code". The CONTENTS REPORT, on the other hand would present this in the following format:

```
 1 employee-address
  2 street
  2 city
  2 state
  2 zip-code
```

* See the description of these records as presented in Section 7 of Part I for a better understanding of the data base structure.
Table 2

Types of Information taken and presented by URA Reports

Therefore, there are three major aspects relative to what is contained in a URA report:

1) What type of information is presented:
   a) Names and types of objects
   b) Comment Entries
   c) Connections among objects and between an object and comment entry

2) How the information is presented:
   a) As taken from the URA data base
   b) Derived from information in the data base

3) How Relationships are defined:
   a) explicitly
   b) implied

All information in this report is "derived" from the information taken from each of the different types of records.

118
ATTRIBUTE REPORT

Purpose

This report is intended to present the system properties aspect of the target system description with respect to the ATTRIBUTES defined and used in the description.

Information Presented

The report presents, for each ATTRIBUTE name used as input, all names in the data base to which the ATTRIBUTE applies and the associated ATTRIBUTE-VALUES for the names. In effect, this presents information given by the ATTRIBUTE statement.

Format

Each ATTRIBUTE name is numbered, 1*, 2*, etc., as it is encountered as input and printed on the report. Each name to which a particular ATTRIBUTE applies is also numbered.

The URA statements:

```
INPUT: time-card;
ATTRIBUTE arrival-type  scheduled;
```

would be presented as:

```
1* ATTRIBUTE:  arrival-type

APPLIES TO:         VALUE:
1 time-card         scheduled
```

in the ATTRIBUTE REPORT for the ATTRIBUTE arrival-type. Given a particular ATTRIBUTE name, the software generating the report searches the data base for all names the ATTRIBUTE APPLIES to (is connected to) and lists them under the "APPLIES TO:" heading in the report with corresponding ATTRIBUTE-VALUE under the "VALUE:" heading.

The format for the ATTRIBUTE REPORT is considered to be a list format.

Option and Alternatives

The report may be generated for a single ATTRIBUTE name (via the NAME parameter) or for a collection of ATTRIBUTE names specified by the user or retrieved via NAME-GEN.

Analysis

Each name given as input in generating the report must be checked that it is an ATTRIBUTE name before further processing continues. The name is then printed on the report. If the input name is not an ATTRIBUTE the message:

```
URA093: NAME HAS NO USAGES AS ATTRIBUTE FOR ANYTHING
```

is printed. Names and corresponding ATTRIBUTE-VALUES are then retrieved pair by pair and listed under the given ATTRIBUTE until no more pairs are found for the ATTRIBUTE.
The next ATTRIBUTE name from the input stream is taken (if there is one) and the process repeats.

Usages

In many applications of the use of the Language and Analyzer, certain ATTRIBUTE names may be designated as mandatory for a description. For example, some applications may require that every PROCESS defined as part of the description must be specified as being either manual or automated via the ATTRIBUTE process-type. Generation of the ATTRIBUTE REPORT for the name process-type allows the analyst to determine if the current description is accurate, complete and consistent in this respect.

The report also presents those names which are logically related because of common ATTRIBUTE-VALUES among them. In the previous example, there were two logical groups, manual and automated. In practice there are usually several possible ATTRIBUTE-VALUES.

Examples

Figure 20 presents the output resulting from generating the ATTRIBUTE REPORT using the names produced by NAME-GEN as input. The following Analyzer commands were given:

```
NAME-GEN ATTRIBUTE
PRINT-ATTRIBUTE-VALUES
```
A IBUTE: arrival-type

APPLIES TO:                      VALUE:
  time-card                     scheduled
  salaried-employment-form      random
  hourly-employment-form        random
  tax-withholding-certificate   random
  employment-termination-form  random

A IBUTE: complexity-level

APPLIES TO:                      VALUE:
  salaried-employee-processing  high
  hourly-employee-processing    high
  new-employee-processing       medium
  terminating-emp-processing    low

A IBUTE: copies

APPLIES TO:                      VALUE:
  hourly-employee-report        three
  salaried-employee-report      three
  terminated-employee-report    two
  hired-employee-report         two

A IBUTE: data-standard

APPLIES TO:                      VALUE:

Figure 20
121
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>birthdate</td>
<td>date</td>
</tr>
<tr>
<td>current-date</td>
<td>date</td>
</tr>
<tr>
<td>employment-date</td>
<td>date</td>
</tr>
<tr>
<td>pay-date</td>
<td>date</td>
</tr>
<tr>
<td>termination-date</td>
<td>date</td>
</tr>
<tr>
<td>5th Attribute: type</td>
<td></td>
</tr>
<tr>
<td>applied to:</td>
<td></td>
</tr>
<tr>
<td>- employee-identification-number</td>
<td>numeric</td>
</tr>
<tr>
<td>- department</td>
<td>numeric</td>
</tr>
<tr>
<td>- job-number</td>
<td>numeric</td>
</tr>
<tr>
<td>- pay-grade-code</td>
<td>numeric</td>
</tr>
<tr>
<td>- salary</td>
<td>numeric</td>
</tr>
<tr>
<td>- total-hours</td>
<td>numeric</td>
</tr>
<tr>
<td>- number-of-deductions</td>
<td>numeric</td>
</tr>
<tr>
<td>- job-title</td>
<td>character</td>
</tr>
<tr>
<td>- supervisor</td>
<td>character</td>
</tr>
<tr>
<td>- city</td>
<td>character</td>
</tr>
<tr>
<td>- state</td>
<td>character</td>
</tr>
<tr>
<td>- street</td>
<td>character</td>
</tr>
<tr>
<td>- employee-name</td>
<td>character</td>
</tr>
</tbody>
</table>
CONSISTS COMPARISON REPORT

Purpose

To present data structure information for SET, INPUT, OUTPUT, ENTITY and GROUP names given as input. The report by-passes all intermediate levels of data structure and only presents the lowest level constituents of those names given as input.

Information Presented

Structure information based on CONSISTS statements in the data base can be presented for SET, INPUT, OUTPUT, ENTITY and GROUP names given as input. However, rather than presenting all levels of the data structure, only the highest and lowest levels are present ignoring all other levels. For example, the structure:

1 hourly-employee-record
  2 employee-name
    3 surname
    3 initial
    3 first-name
  2 employee-identification-number
  2 social-security-number

might appear in the CONTENTS REPORT which presents all levels of the data structure for hourly-employee-record. The CONSISTS COMPARISON REPORT, however, would present the names:

surname
initial
first-name
employee-identification-number
social-security-number

as the low level components of the data structure for hourly-employee-record.

In addition to the data structure relationships presented, similarities among data structures are identified and summarized in the report.

Format

The first part of the report specifies those names given as input, but do not have data structure relationships (CONSISTS statements). Two lists are the given, one giving all names given as input (and having CONSISTS information), and the other giving all the low level constituents of those names in the first list. Next comes the BASIC CONTENTS MATRIX where each row of the matrix represents one of those names given in the list of low level constituents and each column of the matrix represents one of the names given in the list of input names. All names in the list and rows and columns are numbered so that the correspondence between a particular row or column and the name that it represents is one to one. A relationship between a name represented by a particular row and a name represented by a particular column is designated by an asterisk entry (*) at the intersection of the row and column.
in the matrix. A blank entry designates that no such relationship exists.

A second matrix called the CONTENTS SIMILARITY MATRIX is also generated to present similarities in the data structures (for those names given as input) as represented in the BASIC CONTENTS MATRIX. All information in the CONTENTS SIMILARITY MATRIX is derived from information presented in the BASIC CONTENTS MATRIX. All the names used as input are represented by a column number and row number in the matrix. The numbers are the same so that any given object is represented by row J and column J. The matrix should be read from row to column as saying: the data object represented by row I has an integer number of low level constituents in common with the data object represented in column J. When I=J, the number of low level constituents of any object in common with itself is presented. This is, of course, the total number of constituents for that given data object. The final section of this report is the CONTENTS SIMILARITY ANALYSIS which presents those input names that have identical lowest level constituents or which are strict subsets (at the lowest level) of other input names.

Options and Alternatives

No options are available to change format or content of the report.

Analysis

Each name given as input is searched for in the Analyzer data base. If the name is not found, the message:

    URA271: MAINCNC: NAME NOT IN D.B.-

is printed and is not represented in the matrices.

For each name defined in the data base with CONSISTS information, its components are then found via the CONSISTS statement. If its components have CONSISTS information then the procedure is continued until only ELEMENTS are encountered (which may not have any sub-components) or no more CONSISTS information can be found. The lists of input names and low level constituents are then printed on the report.

The BASIC CONTENTS MATRIX is then printed out to illustrate the relationships between the names in the two lists and each relationship is designated by an asterisk.

The CONTENTS SIMILARITY MATRIX is produced by counting the number of column entries in common between any two rows of the BASIC CONTENTS MATRIX. The diagonal is produced by counting the total number of asterisks in the BASIC CONTENTS MATRIX for a given row.

The CONTENTS SIMILARITY SUMMARY is produced by inspecting the numerical values in the CONTENTS SIMILARITY MATRIX. If a particular number presented in the diagonal occurs elsewhere in the same row of the matrix, it means that a particular name (represented by the row) has all of its constituents in common with another name. If the other name has the same number of constituents, then the data structures are identical. If the other name has more constituents, then the first name has a data structure which is a subset of the others.
Usage

One use of the CONSISTS COMPARISON REPORT is to detect redundant or similar data structures. Its ability to do this lies in its presentation (ignoring all intermediate levels of data structure).

This aids the analyst in detecting possible errors in the target system description and simplifying it should similarities in structures occur. This is usually beneficial when the report is generated for all INPUT, OUTPUT and ENTITY names in the description as input.

The report is also beneficial to the system designer who may utilize it to optimize structures that the software will eventually have to access. Identical or similar structures for different ENTITIES, for example, may be mapped into the same type of storage structure to reduce complexity of the software.

Examples

Figure 21 presents the results of generating the CONSISTS COMPARISON REPORT for all INPUT and OUTPUT names in a particular data base. The following commands were used to generate the example:

```
NAME-GEN  INPUT  OUTPUT
CONSISTS-COMPARISON
```

Note that the two names employee-information and paysystem-outputs are not included in the matrices because they do not have CONSISTS information.
URA 73:0 OLD : NAME DOESN'T CONSIST OF ANYTHING - employee-information
URA 73:0 OLD : NAME DOESN'T CONSIST OF ANYTHING - paysystem-outputs
## CONSISTS COMPARISON REPORT

### HOMOGENEOUS MATRIX

The rows are the given input names.

The columns are the lowest level objects which are contained in the rows, with intermediate groups ignored.

If any columns are group names, then the definition is incomplete.

If any columns are ambiguous names, they are possible elements.

<table>
<thead>
<tr>
<th>HOMOGENEOUS NAMES</th>
<th>COLUMN NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. employee-termination-form</td>
<td>1. surname</td>
</tr>
<tr>
<td>2. initial</td>
<td>2. initial</td>
</tr>
<tr>
<td>3. first-name</td>
<td>3. first-name</td>
</tr>
<tr>
<td>4. social-security-number</td>
<td>4. social-security-number</td>
</tr>
<tr>
<td>5. termination-date</td>
<td>5. termination-date</td>
</tr>
<tr>
<td>6. employee-identification-number</td>
<td>6. employee-identification-number</td>
</tr>
<tr>
<td>7. employment-status</td>
<td>7. employment-status</td>
</tr>
<tr>
<td>8. error-code</td>
<td>8. error-code</td>
</tr>
<tr>
<td>9. employment-date</td>
<td>9. employment-date</td>
</tr>
<tr>
<td>10. department</td>
<td>10. department</td>
</tr>
<tr>
<td>11. gross-pay</td>
<td>11. gross-pay</td>
</tr>
<tr>
<td>12. status-code</td>
<td>12. status-code</td>
</tr>
<tr>
<td>13. total-hours</td>
<td>13. total-hours</td>
</tr>
<tr>
<td>14. sex</td>
<td>14. sex</td>
</tr>
<tr>
<td>15. birthdate</td>
<td>15. birthdate</td>
</tr>
<tr>
<td>16. house-number</td>
<td>16. house-number</td>
</tr>
<tr>
<td>17. street</td>
<td>17. street</td>
</tr>
<tr>
<td>18. apartment-number</td>
<td>18. apartment-number</td>
</tr>
<tr>
<td>19. city</td>
<td>19. city</td>
</tr>
<tr>
<td>20. state</td>
<td>20. state</td>
</tr>
<tr>
<td>21. zip-code</td>
<td>21. zip-code</td>
</tr>
<tr>
<td>22. phone</td>
<td>22. phone</td>
</tr>
<tr>
<td>23. job-title</td>
<td>23. job-title</td>
</tr>
<tr>
<td>24. pay-rate</td>
<td>24. pay-rate</td>
</tr>
</tbody>
</table>
**COLUMN NAMES**

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>current-date</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>26</td>
<td>job-number</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>27</td>
<td>pay-grade-code</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>28</td>
<td>supervisor</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>29</td>
<td>pay-date</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>30</td>
<td>check-number</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>31</td>
<td>total-deductions</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>32</td>
<td>net-pay</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>33</td>
<td>federal-tax</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>34</td>
<td>state-tax</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>35</td>
<td>fica-tax</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>36</td>
<td>number-of-deductions</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>37</td>
<td>cumulative-gross-pay</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>38</td>
<td>cumulative-tax-deductions</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>39</td>
<td>cumulative-fica-deductions</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>40</td>
<td>cumulative-hours</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>41</td>
<td>cumulative-state-deductions</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>42</td>
<td>cumulative-federal-deductions</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>43</td>
<td>regular-hours-worked</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>44</td>
<td>overtime-hours-worked</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>45</td>
<td>hours-per-day</td>
<td>ELEMENT</td>
</tr>
</tbody>
</table>
CONISTS COMPARISON REPORT

**COMPUTER MATRIX**

"in a column j" means that column j is contained directly or indirectly in row i. The columns do not consist of anything further. Intermediate groups are ignored.

```
  1 11111111 2222222223 3333333334 44444
  1234567890 1234567890 1234567890 12345

+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| 1 | I******* | I | I | I |
| 2 | I****** | ** | I | I | I |
| 3 | I***** | ** | * | I | I | I |
| 4 | I**** | * | **** | I | I | I |
| 5 | I*** | ** | I | I | I | I | I |
| 6 | I** | | I | I | I | I | I | I |
| 7 | I* | | * | I | I | I | I | I |
| 8 | I | | ** | I | I | I | I | I |
| 9 | | | ** | I | I | I | I | I |
| 10 | | | ** | I | I | I | I | I |
| 11 | | | | I | I | I | I | I |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
```

Figure 21  (Continued)
CONSISTS COMPARISON REPORT

SOMATIC SIMILARITY MATRIX

The sum in (i,i) is the number of objects at the same level contained in row i from above.

The sum in (i,j) (i not equal j) is the sum of objects at the lowest level in common seen rows i and j from above.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>51</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>71</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>51</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>4</td>
<td>51</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>71</td>
<td>51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>51</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>41</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>221</td>
<td>4</td>
<td>5</td>
<td>20</td>
<td>11</td>
<td>131</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>41</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>41</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>11</td>
<td>131</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>211</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>I</td>
<td>I</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 21 (Continued)

130
CONTRACT RELATIONSHIP SUMMARY

<table>
<thead>
<tr>
<th>ROWNUM</th>
<th>NAME</th>
<th>IS A SUBSET OF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>item-termination-form</td>
<td>10 terminated-employee-report</td>
</tr>
<tr>
<td>2</td>
<td>item-employee-report</td>
<td>10 terminated-employee-report</td>
</tr>
<tr>
<td>3</td>
<td>item-employee-report</td>
<td>4 hourly-employee-report</td>
</tr>
<tr>
<td>4</td>
<td>item-employment-form</td>
<td>5 hourly-employment-form</td>
</tr>
</tbody>
</table>

Figure 21 (Continued)
CONSISTS MATRIX REPORT

Purpose

To present the data structure (as specified by the CONSISTS statements), either above or below, each SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT name given as input is involved in.

Information Presented

For each name used as input (which must be a SET, INPUT, OUTPUT, ENTITY, GROUP or ELEMENT) the report presents names the input name CONSISTS of (if the report is generated with the CONSISTS parameter in effect) or names the input name is CONTAINED in (if the report is generated with the CONTAINED parameter in effect). Take the following structure, for example, and assume that its description exists in an Analyzer data base in the form of CONSISTS relationships between the names:

(SET) -- SET-A level 1

(ENTITIES) -- ENT-A ENT-B ENT-C level 2

(GROUPS) -- GR-A GR-B GR-C GR-D GR-E level 3

(ELEMENTS) -- EL-A EL-B EL-C level 4

If the CONSISTS MATRIX REPORT were generated for all the above GROUP names, GR-A, GR-B, GR-C, GR-D and GR-E, and the CONSISTS parameter was in effect when generating the report, the ELEMENT names EL-A, EL-B and EL-C would be presented in the report. The report would also designate GR-A and GR-E as not having any CONSISTS information available.

If the report was generating with the CONTAINED parameter in effect, and the same GROUP names as input, the ENTITY names ENT-A, ENT-B and ENT-C would be presented in the report.
In essence, the CONSISTS MATRIX REPORT presents one level above or one level below designated starting points in the data structures described in an Analyzer data base.

The report could not be generated for SET names with the CONTAINED parameter in effect because the Language does not allow SETS to be CONTAINED in higher level structures. Likewise, the report could not be generated for ELEMENT names with the CONSISTS parameter in effect because the Language does not allow ELEMENTS to CONSIST of lower level structures.

The report also presents statistics on the data structure information such as how many low level components a particular name consists of or how many structures it is contained in.

Format

If the CONSISTS parameter is used when generating the report, any names given as input which do not have CONSISTS statements as part of their descriptions are flagged at the beginning of the report. If the CONTAINED parameter is used, any names given as input which do not have CONTAINED statements as part of their descriptions are flagged.

Two lists of names are then presented, one labeled ROW NAMES and the other COLUMN NAMES. If the CONSISTS parameter was used when generating the report, the names designated as COLUMN NAMES are those which were given as input. If the CONTAINED parameter was used when generating the report, the names designated as ROW NAMES are those which were given as input.

In any case, each name under COLUMN NAMES CONSISTS of zero or more names under ROW NAMES and each name under ROW NAMES IS CONTAINED in zero or more names under COLUMN NAMES. A matrix is then printed to show the relationships between the names designated as ROW NAMES (which are represented by the rows of the matrix) and the names designated as COLUMN NAMES (which are represented by the columns of the matrix). The rows and columns of the matrix are numbered to correspond to the number assigned to each name in the list of ROW NAMES and COLUMN NAMES, respectively.

An asterisk (*) entry at the intersection of a particular row and column of the matrix designates that the name represented by the row is CONTAINED in the name represented by the column.

Inspection of an entire row reveals all names that a particular name (represented by the row) is CONTAINED in. Inspection of an entire column reveals all names that a particular name (represented by the column) CONSISTS of.
A summary section is also included in the report presenting for
each ROW NAME:
- The row it was represented by in the matrix (ROW)
- Its name type (TYPE)
- The number of * entries in its row (or the number of names
  CONTAINING it) (COUNT)

The summary presents for each COLUMN NAME:
- The column it was represented by (COLUMN)
- Its name type (TYPE)
- The number of * entries in its column (or the number of
  names it CONSISTS of) (COUNT)

The summary section for ROW and COLUMN names is ordered in decreasing
order of COUNT.

Options and Alternatives

The report must be generated using either the CONSISTS or CONTAINED
parameter. If the CONSISTS parameter is used, all names given as
input must be SET, INPUT, OUTPUT, ENTITY and/or GROUP names. If the
CONTAINED parameter is used all names given as input must be INPUT,
OUTPUT, ENTITY, GROUP and/or ELEMENT names.

The report may be generated for a single input name (via the NAME
parameter) or for a collection of input names either specified by
the user or retrieved via NAME-GEN.

Analysis

Each name given as input is searched for in the data base. If it is
not found, the message:

URA098: CONCOL: NAME NOT IN D.B.

is printed.

If the CONSISTS parameter is used, each name given as input must be
checked that CONSISTS information is available for it. If no CONSISTS
information is available, the name is listed under the message:

URA315: CONCOL : THE FOLLOWING DO NOT CONSIST OF ANYTHING:

The components of those input names which do have CONSISTS information
are found. The list of all input names and the list of all components
are then printed on the report.

If the CONTAINED parameter is used, each name given as input must be
checked that CONTAINED information is available for it. If no CONTAINED
information is available, the name is listed under the message:

URA411: CONCOL : THE FOLLOWING ARE NOT CONTAINED IN ANYTHING:

Those names which the input names are CONTAINED in are found. The
list of all names they are and the list of names they are CONTAINED in
are then printed on the report.
A matrix is printed out to illustrate the relationships between the names in the two lists and each relationship is designated by an asterisk.

A summary is then produced by counting the number of asterisks appearing in each row and each column of the matrix.

Usages

When a list of ELEMENT and/or GROUP names are given as input to the command producing the report and the CONTAINED parameter is specified, the report aids the analyst by identifying which GROUP and ELEMENT names are not incorporated into higher level information structures. It aids the physical system designer by determining utilization of ELEMENT and GROUP names by the logical information structures within the target system description.

Generating the report (using the CONSISTS parameter) for SET, INPUT, OUTPUT, ENTITY and GROUP names determines which of these have identical structures, with respect to one another, and which are empty, i.e., have no CONSISTS relationships. This also identifies similarities in the structures for these types of names. The analyst may then use this information to determine if redundant structures have been defined, if the description is incomplete, etc.

Example

Figure 22 presents the CONSISTS MATRIX REPORT generated with the CONSISTS parameter in effect and using all INPUT and OUTPUT names in a particular data base as input.

Figure 23 presents the report generated with the CONTAINED parameter in effect and using all GROUP names in a particular data base as input. The Analyzer commands used to generate this example were:

NAME-GEN GROUP
CONSISTS-MATRIX CONTAINED
CONSISTS MATRIX REPORT

The following do not consist of anything:

GROUP  ELEMENT
1. job  -name  GROUP
2. cia.  -security-number  ELEMENT
3. cia.  -title  ELEMENT
4. job.  -identification-number  ELEMENT
5. job.  -status  ELEMENT
6. port.  -entry  GROUP
7. det.  -port-entry  GROUP
8. def.  -port-entry  GROUP
9. con.  -date  GROUP
10. cci.  -job-date  GROUP
11. cia.  -GROSS  GROUP
12. def.  -port-entry  GROUP
13. def.  -job-date  GROUP
14. def.  -salaries  GROUP
15. def.  -late  ELEMENT
16. def.  -port-entry  ELEMENT
17. def.  -mode  ELEMENT
18. def.  -product  ELEMENT
19. def.  -hours-worked  ELEMENT
20. def.  -hours-worked  ELEMENT
21. def.  -day  ELEMENT

Column names

1. Employee-information  INPUT
2. Employment-termination-form  INPUT
3. Error-listing  OUTPUT
4. Hired-employee-report  OUTPUT
5. Hourly-employee-report  OUTPUT
6. Hourly-employment-form  OUTPUT
7. Pay-statement  OUTPUT
8. Pay-system-outputs  OUTPUT
9. Salared-employee-report  OUTPUT
10. Salared-employment-form  INPUT
11. Tax-withholding-certificate  INPUT
12. Terminated-employee-report  OUTPUT
13. Time-card  INPUT

The rows are contained in the columns with *'s.
**CONSISTS MATRIX REPORT**

### NUMBER OF COLUMNS THAT CONTAIN THE ROWS**

<table>
<thead>
<tr>
<th>No.</th>
<th>Column Name</th>
<th>Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ee-name</td>
<td>GROUP</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>ssn</td>
<td>ELEMENT</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>res-identification-number</td>
<td>ELEMENT</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>job-data</td>
<td>GROUP</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>station-date</td>
<td>ELEMENT</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>adopt-status</td>
<td>ELEMENT</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>listing-entry</td>
<td>GROUP</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>-report-entry</td>
<td>GROUP</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>job-data</td>
<td>GROUP</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>sub</td>
<td>GROUP</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>-report-entry</td>
<td>GROUP</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>old-job-data</td>
<td>GROUP</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>all</td>
<td>GROUP</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>net-of-deductions</td>
<td>ELEMENT</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>stat-data</td>
<td>ELEMENT</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>-report-entry</td>
<td>GROUP</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>-cole</td>
<td>ELEMENT</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>pi</td>
<td>ELEMENT</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>hrs-hours-worked</td>
<td>ELEMENT</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>ov-hours-hours-worked</td>
<td>ELEMENT</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>job-gpr-ing</td>
<td>ELEMENT</td>
<td>1</td>
</tr>
</tbody>
</table>

### NUMBER OF ROWS CONTAINED IN THE COLUMNS**

<table>
<thead>
<tr>
<th>Count</th>
<th>TYPE</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ti-third</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>-report-termination-form</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>withholding-certificate</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>non-employment-form</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>status</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>non-employment-form</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>-listing</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>-report</td>
<td>OUTPUT</td>
</tr>
</tbody>
</table>

Figure 22 (Continued)
URAA - EXAM

CONSISTS MATRIX REPORT

EMP. JOB: CM

FILE CODE: INDEX

URAA 11: HOW : THE FOLLOWING ARE NOT CONTAINED IN ANYTHING:
- Hires-update-data
- Terminations-data
- Hires-pay-data
- S-emp-pay-data
- S-terminated-pay-data
- Empl-pay-data
- Card-data

COLUMN NAMES

<table>
<thead>
<tr>
<th>GROUP</th>
<th>ENTITY</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 hourly-employee-record</td>
<td>GROUP</td>
<td></td>
</tr>
<tr>
<td>2 term-report-entry</td>
<td>GROUP</td>
<td></td>
</tr>
<tr>
<td>3 tax-withholding-certificate</td>
<td>INPUT</td>
<td></td>
</tr>
<tr>
<td>4 salaried-employee-record</td>
<td>ENTITY</td>
<td></td>
</tr>
<tr>
<td>5 personal-data</td>
<td>GROUP</td>
<td></td>
</tr>
<tr>
<td>6 pay-statement</td>
<td>OUTPUT</td>
<td></td>
</tr>
<tr>
<td>7 check</td>
<td>GROUP</td>
<td></td>
</tr>
<tr>
<td>8 time-card</td>
<td>INPUT</td>
<td></td>
</tr>
<tr>
<td>9 s-emp-report-entry</td>
<td>GROUP</td>
<td></td>
</tr>
<tr>
<td>10 pay-stub</td>
<td>GROUP</td>
<td></td>
</tr>
<tr>
<td>11 hired-report-entry</td>
<td>GROUP</td>
<td></td>
</tr>
<tr>
<td>12 s-emp-report-entry</td>
<td>GROUP</td>
<td></td>
</tr>
<tr>
<td>13 error-listing-entry</td>
<td>GROUP</td>
<td></td>
</tr>
<tr>
<td>14 employment-termination-form</td>
<td>INPUT</td>
<td></td>
</tr>
<tr>
<td>15 error-listing</td>
<td>OUTPUT</td>
<td></td>
</tr>
<tr>
<td>16 hourly-employee-report</td>
<td>OUTPUT</td>
<td></td>
</tr>
<tr>
<td>17 hired-employee-report</td>
<td>OUTPUT</td>
<td></td>
</tr>
<tr>
<td>18 hourly-employment-form</td>
<td>INPUT</td>
<td></td>
</tr>
<tr>
<td>19 salaried-employment-form</td>
<td>INPUT</td>
<td></td>
</tr>
<tr>
<td>20 salaried-employee-report</td>
<td>OUTPUT</td>
<td></td>
</tr>
<tr>
<td>21 terminated-employee-report</td>
<td>OUTPUT</td>
<td></td>
</tr>
</tbody>
</table>

THE ROWS ARE CONTAINED IN THE COLUMNS WITH *5

Figure 23
Figure 23 (Continued)

141
The number of columns that contain the rows:

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>Group</td>
<td>13</td>
</tr>
<tr>
<td>4-6</td>
<td>Group</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>Group</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>Group</td>
<td>1</td>
</tr>
</tbody>
</table>

The number of rows contained in the columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>Entity</td>
<td>2</td>
</tr>
<tr>
<td>4-6</td>
<td>Group</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Input</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Entity</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>Group</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>Group</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>Input</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>Input</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>Group</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>Input</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>Group</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 23 (Continued)
**CONSISTS MATRIX REPORT**

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>report-entry</td>
<td>GROUP</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>report-entry</td>
<td>GROUP</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>listing-entry</td>
<td>GROUP</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>report-termination-form</td>
<td>INPUT</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>listing</td>
<td>OUTPUT</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>employee-report</td>
<td>OUTPUT</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>employee-report</td>
<td>OUTPUT</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>employee-report</td>
<td>OUTPUT</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>employee-report</td>
<td>OUTPUT</td>
<td>1</td>
</tr>
</tbody>
</table>

*Figure 23 (Continued)*

143
CONTENTS REPORT

Purpose

To allow the user to view entire data structures (all levels) described in an Analyzer data base as implied by the use of the CONSISTS statement.

Information Presented

The CONTENTS REPORT presents all lower levels of data structures for SET, INPUT, OUTPUT, ENTITY and GROUP names used as input. (Since ELEMENTS may not have CONSISTS information they cannot be used as input to produce the report.) All names which the input names CONSIST of are designated as level 2 names; the names that the level 2 names CONSIST of are designated as level 3 names, etc.

The CONSISTS statement allows network structures to be constructed and so any given name may be CONTAINED in more than one structure, and at different levels in the different structures. The types of names presented in the structures will be INPUTS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS.

Format

Each name given as input to the report is identified by a number 1, 2, etc. designating its position in the list of input names and also by the number 1 designating it as a level 1 name. All names that are part of its structure are numbered 1 through n according to its position in the structure when printed out and also numbered according to the name's relative level in the structure. Each level 2, 3 and so on is indented to further accent the idea of structure. Each group of names of a given level number are CONTAINED in the preceding name of the next highest level number. (Level 1 is the highest level number.) For example, the following URL description:

ENTITY hourly-employee-record;
  CONSISTS employee-name,
          employee-identification-number,
          social-security-number;

GROUP employee-name;
  CONSISTS surname,
          initial,
          first-name;

would appear as:

1* 1 hourly-employee-record
1   2 employee-name
2   3 surname
3   3 initial
4   4 first-name
5   2 employee-identification-number
6   2 social-security-number
in the CONTENTS REPORT if the report was generated for hourly-
employee-record. If the report was generated for employee-name
the following structure would appear:

1* 1 employee-name
1  2 surname
2  2 initial
3  2 first-name

Options and Alternatives

The user may restrict the number of levels of the data structures
presented when a numerical value is assigned to the LEVELS parameter.
For example, when LEVELS=2 is given only the names at levels one and
two of the data structure are presented in the report. The report
normally prints out ALL levels of the data structures.

GROUPS in the structures which do not CONSIST of lower level infor-
mation or those UNDEFINED names in the structure are flagged by the
message:

**NO CONSISTS FOR GROUP OR UNDEF**

when the NCFLAG parameter is used. An INDEX for the report is produced
when the INDEX parameter is used.

The report may be generated for a single input name (via the NAME para-
meter) or for a collection of input names either specified by the user
or retrieved via NAME-GEN.

Analysis

Each name given as input is searched for in the data base. If the name
is not found, the message:

URA065: MAINCONT: NAME NOT FOUND IN D.B.-

is printed and no structure information is printed for the name.

For each name given as an input, each name which it CONSISTS of is
designated as a level 2 name as it is printed out. If this level 2
name CONSISTS of any information, then each name which it CONSISTS
of is designated as a level 3 name as it is printed out. This process
continues until no more CONSISTS relationships are found or the level
specified by the LEVELS parameter is reached.

Names are printed out as they are encountered in the structure.

Usage

For the analyst, the report presents information structures as defined
by the use of the CONSISTS statement in a format in which the entire
structure can be seen. It is usually most beneficial to generate the
report on records for INPUT, OUTPUT and ENTITY names in the data base
since all major information constructs are based around these types of
names.

Completeness checks on the structure information in the data base can
be performed by specifying the NCFLAG parameter when generating the
report. Since all GROUPS should consist of other information (by
definition of GROUP) and UNDEFINED names should be resolved, this
is not an indicator of the completeness of the structure.
For a complete set of final specifications for a particular target system, the report may be generated to present the logical information structures to be handled by the target. For this purpose it is recommended that the report be generated for all SET names with LEVELS=2 so that the relationships among INPUTS, OUTPUTS and ENTITIES with SETS can be presented, and generated for all INPUT, OUTPUT and ENTITY names so that structures below these names can be viewed. The commands to generate this information are:

\begin{verbatim}
NAME-GEN SET CONTENTS LEVELS=2
NAME-GEN INPUT OUTPUT ENTITY ORDER=BYTYPE CONTENTS
\end{verbatim}

the BYTYPE option is used so that all names of a particular name type are defined together.

When the volume of information which will be presented by the CONTENTS REPORT is unknown, it is a good practice to be conservative in specifying a value for LEVELS rather than allow LEVELS to have the value ALL and risk the possibility of generating dozens of pages of output.

**Examples**

**Figure 24** presents the full data structure for the name hourly-employee-record. This example was produced by the following command:

\begin{verbatim}
CONTENTS NAME=hourly-employee-record
\end{verbatim}

**Figure 25** presents the data structures down to level 2 for all ENTITIES defined in a particular Analyzer data base. The Analyzer commands used to generate this example were:

\begin{verbatim}
NAME-GEN ENTITY CONTENTS LEVELS=2
\end{verbatim}
1. hourly-employee-record (ENTITY)
   2. employee-name (GROUP)
      3. surname (ELEMENT)
      4. initial (ELEMENT)
      5. first-name (ELEMENT)
   6. employee-identification-number (ELEMENT)
   7. social-security-number (ELEMENT)
   8. pay-grade-code (ELEMENT)
   9. address (GROUP)
      10. house-number (ELEMENT)
      11. street (ELEMENT)
      12. apartment-number (ELEMENT)
      13. city (ELEMENT)
      14. state (ELEMENT)
      15. zip-code: (ELEMENT)
   16. phone (ELEMENT)
   17. employment-date (ELEMENT)
   18. number-of-deductions (ELEMENT)
   19. department (ELEMENT)
   20. cumulative-gross-pay (ELEMENT)
   21. cumulative-federal-deductions (ELEMENT)
   22. cumulative-state-deductions (ELEMENT)
   23. cumulative-fica-deductions (ELEMENT)
   24. age (ELEMENT)
   25. sex (ELEMENT)
   26. status-code (ELEMENT)
   27. cumulative-hours (ELEMENT)
1.  department-record (ENTITY)
   |   department (ELEMENT)
   |   supervisor (ELEMENT)
   |   number-of-employees (ELEMENT)
   |   total-budget (ELEMENT)
   |   remaining-funds (ELEMENT)

2.  hourly-employee-record (ENTITY)
   |   employee-name (GROUP)
   |   employee-identification-number (ELEMENT)
   |   social-security-number (ELEMENT)
   |   pay-grade-code (ELEMENT)
   |   address (GROUP)
   |   phone (ELEMENT)
   |   employment-date (ELEMENT)
   |   number-of-employments (ELEMENT)
   |   department (ELEMENT)
   |   cumulative-gross-pay (ELEMENT)
   |   cumulative-federal-deductions (ELEMENT)
   |   cumulative-state-deductions (ELEMENT)
   |   cumulative-fica-deductions (ELEMENT)
   |   age (ELEMENT)
   |   sex (ELEMENT)
   |   status-code (ELEMENT)
   |   cumulative-hours (ELEMENT)

3.  fulltime-employee-record (ENTITY)
   |   employee-name (GROUP)
   |   employee-identification-number (ELEMENT)
   |   social-security-number (ELEMENT)
   |   pay-grade-code (ELEMENT)
   |   address (GROUP)
   |   phone (ELEMENT)
   |   employment-date (ELEMENT)
   |   number-of-employments (ELEMENT)
   |   department (ELEMENT)
DATA BASE SUMMARY

Purpose

This report provides statistical information with respect to the usage of different name types (e.g. PROCESSES, ELEMENTS, etc.) and can be used as an aid in estimating size of the Language description in an Analyzer data base.

Information Presented

The report provides an entry for each different name type (PROCESS, ELEMENT, etc.) defined in a particular data base. For example, if only PROCESSES have been defined, only a PROCESS entry will appear in the report. For each name type the report entry presents:

- The number of names in the data base of that name type (COUNT)
- The number of these names that have SYNONYMS (#W/SYN)
- The percentage of these names with SYNONYMS (PERCENT)
- The number of these names that have a DESCRIPTION (#W/DESC)
- The percentage of these names with a DESCRIPTION (PERCENT)

An entry is also provided consisting of all of the above considering all name types (**TOTAL**) in the data base.

Format

The report presents six columns of information as follows:

<table>
<thead>
<tr>
<th>name-type</th>
<th>COUNT</th>
<th>#W/SYN</th>
<th>PERCENT</th>
<th>#W/DESC</th>
<th>PERCENT</th>
</tr>
</thead>
</table>

in a table format. A row of the table consists of values for each of the above columns. A row is presented for each name type defined in the data base and the name types are presented in alphabetical order. A row is also printed presenting the total for each column. Since there are a limited number of name types, the report always fits on one page.

Options and Alternatives

There are no options available for this report.

Analysis

The software generating the report looks at every name in the data base and updates three fields according to its name type (COUNT), whether or not it has a DESCRIPTION (#W/DESC), and whether or not it has any SYNONYMS (#W/SYN). After all names have been looked at, percentages (PERCENT) for those names with DESCRIPTIONS and SYNONYMS are computed within each name type.
Finally, totals are produced for values retrieved for each table heading and are printed.

If the report is generated for an empty data base, the message:

    NO NAMES IN D.B.-

is printed.

Usage

The major benefit of this report is to project management personnel in recording progress being made in the target system description procedure. Comparing a particular DATA BASE SUMMARY with previous summaries allows management to see where some of the work effort has been concentrated. For example, if 52 PROCESSES are counted in the report compared to 12 previously, it shows that the major project effort has been centered around definition of PROCESSES.

General progress may be evaluated by looking at the totals presented for successive reports.

Examples

Figure 26 presents the DATA BASE SUMMARY generated from a small description of a target system.
**DATA BASE SUMMARY**

<table>
<thead>
<tr>
<th>COUNT</th>
<th>#/SYN</th>
<th>PERCENT</th>
<th>#/DESC</th>
<th>PERCENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td></td>
<td>4</td>
<td>89.00</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td></td>
<td>1</td>
<td>10.00</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>0</td>
<td></td>
<td>11</td>
<td>21.57</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>100.00</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
<td>4</td>
<td>57.14</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td></td>
<td>5</td>
<td>23.32</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>100.00</td>
<td>6</td>
<td>100.00</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>100.00</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>100.00</td>
<td>7</td>
<td>100.00</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>2.78</td>
<td>8</td>
<td>22.22</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>100.00</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>100.00</td>
<td>4</td>
<td>100.00</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>24</td>
<td>14.12</td>
<td>61</td>
<td>35.88</td>
</tr>
</tbody>
</table>

**Figure 26**

152
DATA PROCESS REPORT

Purpose

This report shows the interaction between information (SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS) defined and the PROCESSES defined for the target system. It also shows the data dependencies among PROCESSES as implied by the Language descriptions of the PROCESSES and possible deficiencies in the descriptions of these PROCESSES.

Information Presented

The information presented in this report is slightly different depending on whether the DATA or PROCESS parameter was used in generating the report.

If the DATA parameter is used in generating the report, the names used as input must be SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS and/or ELEMENTS and the report presents all those PROCESSES which manipulate the input names via the RECEIVED, USED, UPDATED, DERIVED and GENERATED statements in a particular Analyzer data base.

If the PROCESS parameter is used in generating the report, the names used as input must be PROCESS names and the report presents all those SET, INPUT, OUTPUT, ENTITY, GROUP and ELEMENT names that each input PROCESS name manipulates via the RECEIVES, USES, UPDATES, DERIVES and GENERATES statements in a particular Analyzer data base.

The interactions among data and PROCESSES are shown as a matrix. An analysis on this matrix presents a summary describing the incomplete aspect of the Language description with respect to the information presented in the matrix. For example, a PROCESS producing information without using any information would be identified in this summary.

A second matrix presents the manner in which the PROCESSES (presented in the first matrix) interact, i.e., how they depend on data produced by other PROCESSES. Again, the information in this matrix is derived from the information in the first matrix.

Finally, an analysis is performed on this matrix and presented in the form of a summary. The summary identifies those PROCESSES with no predecessors (i.e., do not USE or RECEIVE data produced by other PROCESSES) and those with no successors (i.e., do not DERIVE, UPDATE or GENERATE any data used by other PROCESSES).

The two summaries and the second matrix are all produced based on the information presented in the first matrix. Therefore, items which are designated incomplete in the report may actually be resolved elsewhere in the description in the data base. For example, if the following description was in the data base:
PROCESS: payroll-processing;
USES: employee-information;
DERIVES: paysystem-outputs;

and the DATA PROCESS REPORT was produced for the name employee-information, the PROCESS payroll-processing would be presented since it USES it. The report would then identify payroll-processing as USING data but not UPDATING or DERIVING anything though elsewhere in the description it does. For this reason, it is important to recognize that the comments in the report are made with respect to the information in the first matrix rather than the entire description as it exists in the data base.

Format

Two lists of names are first presented, one labeled ROW NAMES and the other COLUMN NAMES. If the DATA parameter was used in generating the report, the names designated as ROW NAMES are those which were given as input. If the PROCESS parameter was used in generating the report the names designated as COLUMN NAMES are those which were given as input.

In any case, each name under COLUMN NAMES in some way (RECEIVES, USES, etc.) interacts with zero or more names given under ROW NAMES and each name under ROW NAMES is manipulated (USED, DERIVED, etc.) by zero or more, PROCESS names given under COLUMN NAMES.

The DATA PROCESS INTERACTION MATRIX is then printed out to show the relationships between the names designated as ROW NAMES (which are represented by the rows of the matrix) and the names designated as COLUMN NAMES (which are represented by the columns of the matrix). The rows and columns of the matrix are numbered to correspond to the number assigned to each name in the list of ROW NAMES and COLUMN NAMES, respectively.

An entry (R, U, D, A, F, 1 or 2) at the intersection of a particular row and column of the matrix designates that the name represented by the row is manipulated in some way (as defined by the meaning of the entry) by the name represented by the column. A legend is provided as part of the report that defines the meaning of each possible entry. This legend is shown below for purposes of clarification.

<table>
<thead>
<tr>
<th>(i, j) value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>Row i is received or used by column j (input)</td>
</tr>
<tr>
<td>U</td>
<td>Row i is updated by column j</td>
</tr>
<tr>
<td>D</td>
<td>Row i is derived or generated by column j (output)</td>
</tr>
<tr>
<td>A</td>
<td>Row i is input to, updated by, and output of column j (all)</td>
</tr>
<tr>
<td>F</td>
<td>Row i is input to and output of column j (flow)</td>
</tr>
<tr>
<td>1</td>
<td>Row i is input to and updated by column j</td>
</tr>
<tr>
<td>2</td>
<td>Row i is updated by and output of column j</td>
</tr>
</tbody>
</table>
A summary section called the DATA PROCESS INTERACTION MATRIX ANALYSIS is then presented specifying those inconsistencies found in analysis of the DATA PROCESS INTERACTION MATRIX. Inconsistencies for ROW NAMES and COLUMN NAMES are handled separately. Inconsistencies found for ROW NAMES are presented under the DATA heading and are of the following format:

row name    (name-type) (row number) inconsistency message

Inconsistencies found for COLUMN NAMES are presented under the PROCESS heading and are of the following format:

column name    (row number) inconsistency message

No name-type is necessary because all names under COLUMN NAMES are PROCESSES.

A second matrix, the PROCESS INTERACTION MATRIX, is printed to show relationships implied between PROCESSES in the DATA PROCESS INTERACTION MATRIX. In this matrix both the rows and columns represent those PROCESS names listed under COLUMN NAMES and the rows and columns are numbered to correspond to the appropriate name in COLUMN NAMES.

An asterisk (*) entry at the intersection of a particular row and column of the matrix designates that the PROCESS represented by the row DERIVES or UPDATES some information which is USED by the PROCESS represented by the column.

A summary section called the PROCESS INTERACTION MATRIX ANALYSIS then presents observations on the information in the PROCESS INTERACTION MATRIX. These observations are presented in the following format:

process name    (row or column number) observation

An observation is given for each PROCESS which does not use information produced by any of the other PROCESSES, or does not produce information used by any of the other PROCESSES.

Options and Alternatives

The report must be generated using either the DATA or PROCESS parameter. If the DATA parameter is used, all names given as input must be SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names. If the PROCESS parameter is used, all names given as input must be PROCESS names.

Various sections of the report can be printed or left out depending on the parameters used when generating it. These parameters and their effects are presented below:

1. DPMAT - the DATA PROCESS INTERACTION MATRIX is included in the report

   NODPMAT - the matrix is not printed

2. DJANAL - the DATA PROCESS INTERACTION MATRIX ANALYSIS is included in the report

   NODJANAL - the analysis is not printed
3. PMAT - the PROCESS INTERACTION MATRIX is included in the report.

NOPMAT - the matrix is not printed.

4. PANL - the PROCESS INTERACTION ANALYSIS is included in the report.

NOPANL - the analysis is not printed.

The report may be generated for a single input name (via the name parameter) or for a collection of input names either specified by the user or retrieved via NAME-GEN.

Analysis

Each name given as input is searched for in the data base. If it is not found, the message:

URAI43:MAINDP: NAME NOT IN D.B.-

is printed. If the name is found and the DATA parameter is used, the name is checked that it is a SET, INPUT, OUTPUT, ENTITY, GROUP or ELEMENT name. If it is not, the message:

URAI54:MATINDP: INVALID INPUT NAME TYPE

is printed and the name is not included in the matrix.

If the DATA parameter is used, all PROCESS names which RECEIVE, USE, UPDATE, DERIVE and/or GENERATE each input name are found. The list of all input names and the list of all PROCESS names found are then printed on the report.

If the PROCESS parameter is used, all SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names which interact with each input name are found. The list of all input names and the list of all SET, INPUT, OUTPUT, ENTITY, GROUP and ELEMENT names found are then printed on the report.

The DATA PROCESS INTERACTION MATRIX is then printed with the appropriate value (R, U, D, A, F, 1 or 2) designating a relationship between a column name and row name.

The matrix is then analyzed and inconsistency messages are printed as data diagnostics (for row names representing SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names) and process diagnostics (for column names representing PROCESS names).

These diagnostics are presented below categorized by the name types to which they may apply.
I. DATA DIAGNOSTICS (ROWS)

1. INPUT names

- not RECEIVED by any PROCESS

If no PROCESS names RECEIVE the INPUT name of interest, this diagnostic is printed. If at least one PROCESS RECEIVES the INPUT name, the message is not printed.

- not USED by any PROCESS

If no PROCESS USES the INPUT name of interest, this diagnostic is printed. If at least one PROCESSES USES the INPUT name, the message is not printed.

2. OUTPUT names

- not GENERATED by any PROCESS

If no PROCESS names GENERATE the OUTPUT name of interest, this diagnostic is printed. If at least one PROCESS GENERATES the OUTPUT name, the message is not printed.

- not DERIVED by any PROCESS

If no PROCESS names DERIVE the OUTPUT name of interest, this diagnostic is printed. If at least one PROCESS DERIVES the OUTPUT name, the message is not printed.

3. ENTITY or SET names

- not DERIVED by any PROCESS

If no PROCESS names DERIVE the ENTITY or SET name of interest, this diagnostic is printed. If at least one PROCESS DERIVES the name, the message is not printed.

- DERIVED but not USED by any PROCESS

If at least one PROCESS name DERIVES and no PROCESS names USE the ENTITY or SET name of interest, then this diagnostic is printed. There are 3 conditions that will cause this message not to be printed:

i) the name is not DERIVED by any PROCESS

ii) the name is USED by at least one PROCESS

iii) both 1 and 2
- UPDATED but not USED by any PROCESS

If at least one PROCESS UPDATES and no PROCESS USES the ENTITY or SET name of interest, this diagnostic is printed. There are 3 conditions that will cause this message not to be printed:

1) the name is not UPDATED by any PROCESS
2) the name is USED by at least one PROCESS
3) both 1 and 2

4. GROUP or ELEMENT

- not DERIVED, UPDATED, or USED by any PROCESS

If no PROCESS names DERIVE, UPDATE or USE the GROUP or ELEMENT name of interest, this diagnostic is printed. There are several conditions that will cause this message not to be printed:

1) at least one PROCESS DERIVES the name
2) at least one PROCESS UPDATES the name
3) at least one PROCESS USES the name
4) all 3 or any combination of the above

NOTE: it is only necessary that one of the first 3 conditions is satisfied for the message not to be printed.

II. PROCESS DIAGNOSTICS (COLUMNS)

- does not interact with any data

If the PROCESS of interest does not interact with any SET, INPUT, OUTPUT, ENTITY, GROUP or ELEMENT names, this diagnostic is printed. If at least one name interacts with this PROCESS, this message is not printed.

- USES data, but does not DERIVE or UPDATE anything

If the PROCESS of interest USES at least one SET, INPUT, ENTITY, GROUP or ELEMENT name and does not DERIVE or UPDATE any, this diagnostic is printed. There are several conditions where this message will not be printed:
i) the PROCESS does not USE any SET, INPUT, ENTITY, GROUP or ELEMENT

ii) the PROCESS DERIVES at least one SET, OUTPUT, ENTITY, GROUP or ELEMENT

iii) the PROCESS UPDATES at least one SET, ENTITY, GROUP or ELEMENT

iv) all 3 or any combination of the above

- DERIVES something but does not USE anything

If the PROCESS of interest DERIVES at least one SET, ENTITY, GROUP or ELEMENT and does not USE any, this diagnostic is printed. There are 3 conditions where this message will not be printed:

i) the PROCESS does not DERIVE any SET, OUTPUT, ENTITY, GROUP or ELEMENT

ii) the PROCESS USES at least one SET, INPUT, ENTITY, GROUP or ELEMENT

iii) both 1 and 2

- UPDATES something but does not USE anything

If the PROCESS of interest UPDATES at least one SET, ENTITY, GROUP or ELEMENT and does not USE any, this diagnostic is printed. There are 3 conditions where this message will not be printed:

i) the PROCESS does not UPDATE any SET, ENTITY, GROUP or ELEMENT

ii) the PROCESS USES at least one SET, INPUT, ENTITY, GROUP or ELEMENT

iii) both 1 and 2

The PROCESS INTERACTION MATRIX is then produced using the data in the first matrix. The entries in a given column (representing a PROCESS name) of the DATA PROCESS INTERACTION MATRIX are compared with the entries for each other column. If an entry in the given column designates that the PROCESS USES information which is DERIVED or UPDATED by the column being compared with, an * entry is made in the PROCESS INTERACTION matrix at the column or row, respectively, representing these two PROCESS names. The matrix is then analyzed and observations are produced from this analysis. These observations are presented below.
- no interaction with other PROCESSES

If a PROCESS does not USE an object and also does not DERIVE or UPDATE an object, this diagnostic is printed. There are 3 conditions that will cause this message not to be printed:

1) the PROCESS USES at least one SET, INPUT, ENTITY, GROUP or ELEMENT

ii) the PROCESS UPDATES or DERIVES a SET, OUTPUT, ENTITY, GROUP or ELEMENT

iii) both 1 and 2

- no predecessors for this PROCESS

If the PROCESS of interest does not USE any objects but DERIVES or UPDATES at least one object, this diagnostic is printed. The 3 conditions that will cause this message not to be printed are:

1) the PROCESS USES at least one SET, INPUT, ENTITY, GROUP or ELEMENT

ii) the PROCESS does not DERIVE or UPDATE any SETS, OUTPUTS, ENTITIES, GROUPS or ELEMENTS

iii) both 1 and 2

- no successors for this PROCESS

If the PROCESS of interest USES at least one object but does not DERIVE or UPDATE any object, this diagnostic is printed. The 3 conditions that will cause this message not to be printed are:

1) the PROCESS does not USE any SETS, INPUTS, ENTITIES, GROUPS or ELEMENTS

ii) the PROCESS DERIVES or UPDATES at least one SET, OUTPUT, ENTITY, GROUP or ELEMENT

iii) both 1 and 2

Usage

When the report is generated using the DATA parameter, it aids in presenting the utilization of SET, INPUT, OUTPUT, ENTITY, GROUP and ELEMENT names defined. This aids in identifying which names are not being utilized. For those names which are being utilized, it presents all the PROCESS names which utilize them.

When the report is generated using the PROCESS parameter, it presents all the data required for each particular PROCESS. It also aids in identifying PROCESS names which do not interact with data or are not consistently defined with respect to the manner in which they use data.
The PROCESS INTERACTION MATRIX may be used by designers to plan out
the logic of the target system because it presents the data dependencies
among the PROCESSES defined.

The following completeness checks can be made for a target system
description based on information presented in the report:

- All INPUTS RECEIVED by some PROCESS
- All INPUTS USED by some PROCESS
- All OUTPUTS GENERATED by some PROCESS
- All OUTPUTS DERIVED by some PROCESS
- All ENTITIES and SETS DERIVED by some PROCESS
- All ENTITIES and SETS DERIVED and USED by some PROCESS
- All ENTITIES and SETS are UPDATED and USED by some PROCESS
- All GROUPS and ELEMENTS are DERIVED or UPDATED or USED by some
  PROCESS
- All PROCESSES USE data and DERIVE or UPDATE data
- All PROCESSES which DERIVE data also USE data
- All PROCESSES which UPDATE data also USE data
- All PROCESSES interact with data in some way

The report may also be used to aid in determining if the description
of the target system was specified consistently with respect to the
use of language statements. In particular, it determines:

- whether or not the use of RECEIVES and GENERATES statements
  in describing the system flow aspect of the system is cons-
  sistent.

- whether or not the use of USES, UPDATES and DERIVES statements
  in describing the data derivation aspect of the system is
  consistent.

Examples

Figure 27 presents the DATA PROCESS REPORT generated for all INPUT,
OUTPUT and ENTITY names defined for a particular target system descrip-
tion. This example was produced using the Analyzer commands:

NAME-GEN INPUT OUTPUT ENTITY
Figure 28 presents the report generated for low level PROCESSES defined in the description. These PROCESSES were identified by the KEYWORD "terminal" and the report was produced by the following Analyzer commands:

NAME-GEN PROCESS KEY=terminal
DATA-PROCESS PROCESS
The table below lists the columns and their corresponding types:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>t-record</td>
<td>ENTITY</td>
</tr>
<tr>
<td>Information</td>
<td>INPUT</td>
</tr>
<tr>
<td>t-termination-form</td>
<td>INPUT</td>
</tr>
<tr>
<td>t-termination-report</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-holiday-employee-record</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-holiday-employee-report</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-holiday-employee-form</td>
<td>INPUT</td>
</tr>
<tr>
<td>t-event</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-employee-output</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-employee-record</td>
<td>ENTITY</td>
</tr>
<tr>
<td>t-employee-report</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-employee-form</td>
<td>INPUT</td>
</tr>
<tr>
<td>t-termination-employee-report</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-termination-employee-form</td>
<td>INPUT</td>
</tr>
<tr>
<td>t-holiday-employee-record</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-holiday-employee-report</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-holiday-employee-form</td>
<td>INPUT</td>
</tr>
<tr>
<td>t-termination-employee-report</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-termination-employee-form</td>
<td>INPUT</td>
</tr>
<tr>
<td>t-holiday-employee-record</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-holiday-employee-report</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>t-holiday-employee-form</td>
<td>INPUT</td>
</tr>
<tr>
<td>t-termination-employee-report</td>
<td>OUTPUT</td>
</tr>
</tbody>
</table>

**Figure 27**

163
**DETRACTION MATRIX**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>R</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>R</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>D</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>I</td>
<td>D</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>I</td>
<td>D</td>
<td>R</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>K</td>
<td>K</td>
<td>E</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>D</td>
<td>D</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>R</td>
<td>I</td>
<td>R</td>
<td>D</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>K</td>
<td>K</td>
<td>R</td>
<td>K</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>R</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 27 (Continued)
### Data Process Report

#### Entity-Interaction Matrix Analysis

<table>
<thead>
<tr>
<th>Entity</th>
<th>(Entity)</th>
<th>(Row 1)</th>
<th>Not Derived by Any Process</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity</th>
<th>(Column 7)</th>
<th>Uses Data, But Does Not Derive or Update Any</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity</th>
<th>(Column 8)</th>
<th>Uses Data, But Does Not Derive or Update Any</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVENT</td>
<td>ACTIVITY MATRIX ANALYSIS</td>
<td>(ROW/COL)</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>1.1.</td>
<td>testing</td>
<td>1) NO INTERACTION, BUT HAS SUBPARTS</td>
</tr>
<tr>
<td>1.2.</td>
<td>map-processing</td>
<td>2) NO INTERACTION, BUT HAS SUBPARTS AND IS PART OF</td>
</tr>
<tr>
<td>1.3.</td>
<td>log-processing</td>
<td>3) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td>1.4.</td>
<td>con-processing</td>
<td>4) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td>1.5.</td>
<td>proc-processing</td>
<td>5) NO INTERACTION, BUT HAS SUBPARTS AND IS PART OF</td>
</tr>
<tr>
<td>1.6.</td>
<td>con-creation</td>
<td>6) NO PREDECESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td>1.7.</td>
<td>entity-generation</td>
<td>7) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td>1.8.</td>
<td>entity-creation</td>
<td>8) NO INTERACTION, BUT IS PART OF ANOTHER PROCESS</td>
</tr>
<tr>
<td>1.9.</td>
<td>old-creation</td>
<td>9) NO PREDECESSORS FOR THIS PROCESS</td>
</tr>
</tbody>
</table>

Figure 27 (Continued)
167
**Column Names**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. federal-deductions-update</td>
<td>PROCESS</td>
</tr>
<tr>
<td>2. p fica-deductions-update</td>
<td>PROCESS</td>
</tr>
<tr>
<td>3. funds-update</td>
<td>PROCESS</td>
</tr>
<tr>
<td>4. gross-pay-update</td>
<td>PROCESS</td>
</tr>
<tr>
<td>5. h-gross-pay-computation</td>
<td>PROCESS</td>
</tr>
<tr>
<td>6. hours-update</td>
<td>PROCESS</td>
</tr>
<tr>
<td>7. n-t-pay-computation</td>
<td>PROCESS</td>
</tr>
<tr>
<td>8. pay-computation-validation</td>
<td>PROCESS</td>
</tr>
<tr>
<td>9. s-gross-pay-computation</td>
<td>PROCESS</td>
</tr>
<tr>
<td>10. state-deductions-update</td>
<td>PROCESS</td>
</tr>
<tr>
<td>11. tax-computation</td>
<td>PROCESS</td>
</tr>
<tr>
<td>12. time-card-validation</td>
<td>PROCESS</td>
</tr>
<tr>
<td>13. total-deductions-computation</td>
<td>PROCESS</td>
</tr>
<tr>
<td>14. total-hours-computation</td>
<td>PROCESS</td>
</tr>
</tbody>
</table>
### 3 Interaction Matrix

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Row $i$ is received or used by column $j$ (input)</td>
</tr>
<tr>
<td>1</td>
<td>Row $i$ is updated by column $j$</td>
</tr>
<tr>
<td>2</td>
<td>Row $i$ is derived or generated by column $j$ (output)</td>
</tr>
<tr>
<td>3</td>
<td>Row $i$ is input to, updated by, and output of column $j$ (all)</td>
</tr>
<tr>
<td>4</td>
<td>Row $i$ is input to and output of column $j$ (flow)</td>
</tr>
<tr>
<td>5</td>
<td>Row $i$ is input to and updated by column $j$</td>
</tr>
<tr>
<td>6</td>
<td>Row $i$ is updated by and output of column $j$</td>
</tr>
</tbody>
</table>

Figure 28 (Continued)
Figure 28 (Continued)
<table>
<thead>
<tr>
<th>Action Matrix Analysis</th>
<th>Action</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>actions-update</td>
<td>(ROW/COL 1) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td></td>
<td>zones-update</td>
<td>(ROW/COL 2) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 3) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 4) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 5) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 6) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 7) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 8) NO INTERACTION, BUT IS PART OF ANOTHER PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 9) NO PREDECESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 10) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 11) NO SUCCESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 12) NO INTERACTION, BUT IS PART OF ANOTHER PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 13) NO PREDECESSORS FOR THIS PROCESS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ROW/COL 14) NO PREDECESSORS FOR THIS PROCESS</td>
</tr>
</tbody>
</table>

Figure 28 (Continued)
DICTIONARY REPORT

Purpose

This report presents definitions attached to names used in a Language description and is intended as an aid in communication among persons interested in the description.

Information Presented

This report prints out the following information about each name used as input when generating the report and the appropriate parameters are used:

- Name type of the name
- DESCRIPTION comment entry for the name
- SYNONYMS associated with the name
- RESPONSIBLE PROBLEM DEFINER for the name's description
- KEYWORDS associated with the name

All of the above information is readily available from the contents of the data base.

Format

A dictionary entry is presented for each name given as input to the software producing the report. The first line of each entry consists of:

- A number designating the order the name was read from the input (and consequently presented in the report)
- The name the entry is for
- The name type of the name

The DESCRIPTION, SYNONYMS, KEYWORDS and RESPONSIBLE-PROBLEM-DEFINER statements for each name are presented in the following format after the first line of the entry:

DESCRIPTION:

[DESCRIPTION comment entry]

SYNONYMS: [all SYNONYMS for the name listed two per line]

KEYWORDS: [all KEYWORDS for the name listed two per line]

RESP-PROB: [RESPONIBLE DEFINER NAME]

Spacing between dictionary entries may be modified by the NUM-SPACE parameter.
Options and Alternatives

The number of lines skipped between dictionary entries is specified by the NUM-SPACE parameter. By default, 3 lines are skipped but NUM-SPACE may take any value 0 through 10.

The different types of information presented in a report entry can be included in or left out of the report depending on the parameters used when generating it. Each parameter and its effect is presented below:

1. DESCRIPTION - the DESCRIPTION comment entry for each name is printed
   NODESCRIPTION - DESCRIPTION comment entries are not printed

2. KEYWORDS - all KEYWORDS associated to each name is printed out
   NOKEYWORDS - KEYWORDS are not printed

3. RESPONSIBLE-PD - the RESPONSIBLE-PROBLEM-DEFINER for each name is printed
   NORESPONSIBLE-PD - RESPONSIBLE-PROBLEM-DEFINERS are not printed

4. SYNONYMS - all SYNONYMS associated to each name are printed out
   NOSYNONYMS - SYNONYMS are not printed

An INDEX for the report is provided when the INDEX parameter is used.

Analysis

For each name given as input, the software finds the name in the data base. If the name cannot be found, the message:

URA064: MAINDICT: NAME NOT FOUND IN D.B. -

is printed. If the name is found, the information specified by the parameters for the command is retrieved if available for the name.

Usage

The report is a valuable aid to analysts in maintaining definitions for names in the data base and as a tool for communicating with users of the target system. The DESCRIPTIONS for each name may be okayed or disapproved by the users with respect to what the users require of the target system. As DESCRIPTIONS are modified, the analyst can add, delete or modify other statements to correspond to the DESCRIPTIONS.

If conventions are imposed on the language description required for particular types of names, an effective data dictionary can be formed. For example, by requiring certain keywords to be assigned to OPER and
ELEMENT names and that each GROUP and ELEMENT name have a DESCRIPTION, the DICTIONARY REPORT would be a good reference for anyone interested in the data described in the target system.

Examples

Figure 29 presents the DICTIONARY REPORT for a single PROCESS name. This example was produced by the command:

```
DICTIONARY NAME=payroll-processing
```

Figure 30 presents the report for several PROCESS names which have the KEYWORD 'independent'. This example was produced by the following Analyzer commands:

```
NAME-GEN PROCESS KEY=independent
DICTIONARY
```
PROCESS

DESCRIPTION:
This process represents the highest level process in the target system. It accepts and processes all inputs and produces all outputs.

SYNONYMS: payproc

RESP PD: michel-j-bastarache
1. Weekly-employee-processing PROCESS

DESCRIPTION:
This process performs those actions needed to interpret time cards to produce a pay statement for each hourly employee.

KEYWORDS: independent

2. New-employee-processing PROCESS

DESCRIPTION:
This process stores information about new employees and then prints out a corresponding report.

KEYWORDS: independent

3. Salaried-employee-processing PROCESS

DESCRIPTION:
This process produces the pay statement for salaried employees once a month.

KEYWORDS: independent

4. Sub-routines PROCESS

DESCRIPTION:
These are routines used by one or more processes in the system.

Figure 30
178
DESCRIPTION:
This process deletes data for those employees who are no longer on the payroll, from the files. It also prints a list of all employees no longer on the payroll.

KEYWORDS: independent
Purpose

To present in graphical format, for each name input, a network of names related to it by structure or by data flow.

Information Presented

Names input to the report may have any of the following types:

- ELEMENT
- ENTITY
- GROUP
- INPUT
- INTERFACE
- OUTPUT
- PROCESS
- SET

Starting with each input name, one of four pictures will be obtained. These are referred to as structure downward, structure upward, data-flow forward, and data-flow backward.

For each name used as input, the report presents all successors to that name, where successors are found using the relationships listed in the appropriate column of either Table 3 or Table 4. For each of these new names, the report presents all of its successors, finding them in a similar manner. This network continues until the desired number of relationships has been traced, a loop is encountered, or no more relationships are found.

<table>
<thead>
<tr>
<th>Name Type</th>
<th>Relationships Displayed Structure Downward</th>
<th>Relationships Displayed Structure Upward</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELEMENT</td>
<td></td>
<td>CONTAINED</td>
</tr>
<tr>
<td>ENTITY</td>
<td>CONSISTS</td>
<td>CONTAINED</td>
</tr>
<tr>
<td>GROUP</td>
<td>CONSISTS</td>
<td>CONTAINED</td>
</tr>
<tr>
<td>INPUT</td>
<td>SUBPARTS, CONSISTS</td>
<td>PART, CONTAINED</td>
</tr>
<tr>
<td>INTERFACE</td>
<td>SUBPARTS</td>
<td>PART</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>SUBPARTS, CONSISTS</td>
<td>PART, CONTAINED</td>
</tr>
<tr>
<td>PROCESS</td>
<td>SUBPARTS, UTILIZES</td>
<td>PART, UTILIZED</td>
</tr>
<tr>
<td>SET</td>
<td>SUBSETS, CONSISTS</td>
<td>SUBSET</td>
</tr>
</tbody>
</table>

Table 3
Structure Relationships Displayed in Extended Picture Report

180
<table>
<thead>
<tr>
<th>Name Type</th>
<th>Relationships Displayed</th>
<th>Relationships Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data-Flow Forward</td>
<td>Data-Flow Backward</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>USED</td>
<td>DERIVED</td>
</tr>
<tr>
<td></td>
<td>USED TO DERIVE</td>
<td>UPDATED</td>
</tr>
<tr>
<td></td>
<td>USED TO UPDATE</td>
<td></td>
</tr>
<tr>
<td>ENTITY</td>
<td>USED</td>
<td>DERIVED</td>
</tr>
<tr>
<td></td>
<td>USED TO DERIVE</td>
<td>UPDATED</td>
</tr>
<tr>
<td></td>
<td>USED TO UPDATE</td>
<td></td>
</tr>
<tr>
<td>GROUP</td>
<td>USED</td>
<td>DERIVED</td>
</tr>
<tr>
<td></td>
<td>USED TO DERIVE</td>
<td>UPDATED</td>
</tr>
<tr>
<td></td>
<td>USED TO UPDATE</td>
<td></td>
</tr>
<tr>
<td>INPUT</td>
<td>RECEIVED</td>
<td>GENERATED</td>
</tr>
<tr>
<td></td>
<td>USED</td>
<td></td>
</tr>
<tr>
<td></td>
<td>USED TO DERIVE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>USED TO UPDATE</td>
<td></td>
</tr>
<tr>
<td>INTERFACE</td>
<td>GENERATES</td>
<td>RECEIVES</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>RECEIVED</td>
<td>GENERATED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DERIVED</td>
</tr>
<tr>
<td>PROCESS</td>
<td>GENERATES</td>
<td>RECEIVES</td>
</tr>
<tr>
<td></td>
<td>DERIVES</td>
<td>USES</td>
</tr>
<tr>
<td></td>
<td>UPDATES</td>
<td>USES TO DERIVE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USES TO UPDATE</td>
</tr>
<tr>
<td>SET</td>
<td>USED</td>
<td>DERIVED</td>
</tr>
<tr>
<td></td>
<td>USED TO DERIVE</td>
<td>UPDATED</td>
</tr>
<tr>
<td></td>
<td>USED TO UPDATE</td>
<td></td>
</tr>
</tbody>
</table>

Table 4
Data Flow Relationships Displayed in Extended Picture Report

181
Format

Each name which appears on the output is shown within a box. The top line of the box indicates the name type, while the bottom line shows the relationship with the name's predecessor. Boxes containing related names are linked by dotted lines.

If a name joins two or more chains (strings of related names) into a loop or loops, every appearance of that name after the first will be linked to a box containing the message, "LOOPS TO PREVIOUS ENTRY."

Output is continued across page boundaries. If the right edge of one page continues to the left edge of a second, the rightmost column of boxes on the first page will be repeated as the leftmost column of boxes on the second page, in order to facilitate matching of edges. Similarly, if the bottom edge of one page continues to the top edge of a second, the bottom row of boxes on the first page will be repeated as the top row of boxes on the second page.

Options and Alternatives

The report may be generated for a single name (via the NAME parameter) or for a collection of names, either input by the user or obtained by use of NAME-GEN.

The type of picture to be produced is selected by specifying one of the following parameter pairs:

- STRUCTURE DOWNWARD
- STRUCTURE UPWARD
- DATA-FLOW FORWARD
- DATA-FLOW BACKWARD

The number of columns and rows used on the page may be decreased from their default values of 119 and 39, respectively, via the COLUMNS and ROWS parameters. The minimum acceptable values for COLUMNS and ROWS are 38 and 14, respectively.

The number of boxes arranged horizontally or vertically on a page may be decreased from the defaults, which are the maximum numbers that will fit in each direction (depending on COLUMNS and ROWS), in order to make the output less cluttered. The parameters which can be used to do this are HORIZONTAL-BOXES and VERTICAL-BOXES. Their maximum values (for COLUMNS=119 and ROWS=39) are 6. Due to the scheme for continuing pages, their minimum values are 2.

The number of connections to be traced, starting at the given name, may be set at any positive value via the LINKS parameter. The same LINKS value is used for all names input when the FILE parameter is used.

An index, containing each name used on the report and the page(s) on which it appears, may be obtained by specifying the INDEX parameter.
Each name given as input is first checked to see that it is in the data base and that it has one of the legal types (ELEMENT, ENTITY, GROUP, INPUT, INTERFACE, OUTPUT, PROCESS, or SET). If the name is not in the data base, the message

**URA370: MAINEP: NAME NOT FOUND IN DATA BASE**

will be given. If the name has a type which is not in the list above, the user will receive the message

**URA371: EPSUCC: NAME NOT ACCEPTABLE TYPE FOR EP REPORT.**

If the name passes these two tests, it is placed in a data structure which will later be used for output. The name is then used to generate a tree structure as follows. Using the relationships in the appropriate column of Table 3 or Table 4, all successors for the name are retrieved from the data base and placed on a stack. Then, the first name is removed from the stack, placed in its proper location in the data structure, and all successors for that name are retrieved and placed on the stack. This procedure continues, with names being removed from the top of the stack, placed in the data structure, and used to obtain further names, which are then placed on the stack. At any stage of this procedure, no names will be put on the stack if one of the following is true:

1. The current name has no successors

2. The current name has been encountered earlier, and is therefore at the end of a chain or forms a loop with some portion of a chain traced earlier.

3. The number of links that has been traced on the current chain is equal to the limit set by the LINKS parameter. (For every input name for which this occurs, the message

**URA372: EPSUCC: LINK LIMIT SPECIFIED WAS REACHED**

will be printed.)

Thus, in any of these cases the size of the stack will decrease. The entire procedure is complete when, after any search, the stack is empty.

The data structure constructed from all names found as above is broken into page-sized units and is printed a page at a time.

The process described above is repeated until no more names remain in the input stream.
Usage

The EXTENDED PICTURE is very similar in content to the PICTURE report and therefore, most usages of the PICTURE report apply to the EXTENDED PICTURE report.

In addition, the EXTENDED PICTURE report provides a comprehensive view of the information flow and structure aspects of the target system for inclusion in the final specifications of the system, or as an aid in communicating this information to others.

Problem definers may use the EXTENDED PICTURE report to visually analyze the description of particular objects and the system as a whole, for completeness. Table 5 presents all completeness checks that can be made by visually scanning EXTENDED PICTURE reports.

Examples

Figure 31 presents an EXTENDED PICTURE of structure information for the PROCESS "hourly-employee-processing."

Figure 32 presents an EXTENDED PICTURE of data flow information for the INTERFACE "payroll-department." The amount of information presented was limited by setting LINKS=3.
<table>
<thead>
<tr>
<th>STRUCTURE:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SET</strong></td>
<td>check that SET is broken down into ENTITIES, or INPUTS or SETS.</td>
</tr>
<tr>
<td><strong>INPUT/OUTPUT/GROUP/ENTITY</strong></td>
<td>check that these are eventually broken down into ELEMENTS.</td>
</tr>
<tr>
<td><strong>GROUP/ELEMENT</strong></td>
<td>check that these are contained within some larger data structure.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FLOW:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PROCESS</strong></td>
<td>check that information produced is used in some manner.</td>
</tr>
<tr>
<td></td>
<td>check that information used has been made available (produced) in some manner.</td>
</tr>
<tr>
<td></td>
<td>check that all PROCESSES interact with data in some manner.</td>
</tr>
<tr>
<td><strong>INTERFACE</strong></td>
<td>check that these all generate INPUTS to the system and/or receive OUTPUTS.</td>
</tr>
<tr>
<td></td>
<td>check that the OUTPUTS received have been generated in some manner.</td>
</tr>
<tr>
<td></td>
<td>check that the INPUTS generated are used in some manner.</td>
</tr>
<tr>
<td><strong>SET</strong></td>
<td>check that all these are USED, UPDATED and/or DERIVED in some manner.</td>
</tr>
<tr>
<td><strong>INPUT/OUTPUT/ENTITY</strong></td>
<td>check that all these are produced in some manner and/or used in some manner.</td>
</tr>
<tr>
<td><strong>GROUP/ELEMENT</strong></td>
<td>check that all these are produced in some manner and/or used in some manner.</td>
</tr>
</tbody>
</table>

Table 5
Completeness Checks that may be made by Visual Analysis of the EXTENDED PICTURE report
EXTENDED PICTURE

+---PROCESS---
IN-GROSS- I
IPAY- I
ICONPUTATION
+---PART----+

+---PROCESS---
IN-REPORT- I
ENTRY- I
IGENERATION I
+---PART----+

+---PROCESS---
IFUNDS- I
IUPDATE I
I UTILIZED---+

+---PROCESS---
IFICA- I
IDEDUCTIONS- I
+---PROCESS---
IUPDATE I
I UTILIZED---+

+---PROCESS---
IHOJRAL- I
IEMP- I
IUPDATE I
+---PART----+

+---PROCESS---
ISTATE- I
IDEDUCTIONS- I
IUPDATE I
I UTILIZED---+

+---PROCESS---
IFEDERAL- I
IDEDUCTIONS- I
IUPDATE I
I UTILIZED---+

Figure 31 (Continued)

188
EXTENDED PICTURE

+---PROCESS---+
IFEDERAL- I
INEDUCTIONS- I
UPDATE I
+---UTILIZED---+

+---PROCESS---+
IGROSS- I
IPAY- I
UPDATE I
+---UTILIZED---+

+---PROCESS---+
IHOURS- I
UPDATE I
I
+---PART-----+

+---PROCESS---+
IHOURLY-PAY-I
ICONP- I
VALIDATION I
+---PROCESS---+
IHOURLY- I
+---UTILIZED---+
IPAYCHECK- I
VALIDATION I
PART-----+
TIME- I
ICARD- I
VALIDATION I
PART-----+
PARAMETER FOR: EP

NAME= ROLL-DEPARTMENT DATA-FLOW NOSTRUCTURE NOFORWARD BACKWARD LINKS=3 NOINDEX COLUMNS=100
ROWS=5 HORIZONTAL-BOXES=5 VERTICAL-BOXES=6

Figure 32

190
EXTENDED PICTURE

---PROCESS---
ISALARIED- I  ISALARIED- I
IEMPLOYEE- I  IEMPLOYEE- I
IPROCESSING I  IRECORD I
---GENERATES---
---INPUT---
I I
I TIME-CARD I
I I
---RECEIVED---

---PROCESS---
IHOURLY- I  IHOURLY- I
IEMPLOYEE- I  IEMPLOYEE- I
IPROCESSING I  IRECORD I
---GENERATES---
---INPUT---
I I
I TIME-CARD I
I I
---RECEIVED---

---PROCESS---
IHOURLY- I  I LOOPS TO I
IEMPLOYEE- I  I PREVIOUS I
IPROCESSING I  I ENTRY I
---DERIVES---
---PROCESS---
ISALARIED- I  I LOOPS TO I
IEMPLOYEE- I  I PREVIOUS I
IPROCESSING I  I ENTRY I
---DERIVES---

Figure 32 (Continued)

191
EXTENDED PICTURE

---PROCESS---+ "---INPUT---+
ISALARIED-  I  I  LOKS TO  I
EMPLOYEE-  I  I  PREVIOUS  I
PROCESSING  I  I  ENTRY  I
---DERIVES---+ "---RECEIVED--+

---INPUT---+
ITAX-WITHHO- I
ILDING-CERT-I
IFIICATE  I
---RECEIVED--+

---INPUT---+
ISALARIED-  I
EMPLOYMENT-I
IFORM  I
---RECEIVED--+

---PROCESS---+ ---INPUT---+
HIREP-  I  I  I  I  I  I  I  I
EMPLOYEE-  I  I  I  I  I  I  I  I
REPORT  I  I  I  I  I  I  I  I
---RECEIVED---+ "---GENERATES---+

---INPUT---+
ITAX-WITHHO- I
ILDING-CERT-I
IFIICATE  I
---USED---+

---INPUT---+
ISALARIED-  I
EMPLOYMENT-I
IFORM  I
---USED---+

Figure 32 (Continued)
192
Figure 32 (Continued)
EXTENDED PICTURE

+---PROCESS--+
ISALARIED- I I LOOPS TO I
EMPLOYEE- I I PREVIOUS I
+---OUTPUT--+
ISALARIED- I I PROCESSING I I ENTRY I
EMPLOYEE- I
REPORT I
+---RECEIVED--+
ISALARIED- I I LOOPS TO I
EMPLOYEE- I I PREVIOUS I
PROCESSING I I ENTRY I
+---DERIVES--+
+---INPUT--+
EMPLOYMENT- I
TERMINATI- I
+---PROCESS--+
TERMINATI- I I FORM I
+---OUTPUT--+
TERMINATED- I I GENERATES-+
EMPLOYEE- I
REPORT I
+---RECEIVED--+
TERMINATI- I I LOOPS TO I
EMPLOYEE- I I PREVIOUS I
PROCESSING I I ENTRY I
+---DERIVES--+

Figure 32 (Continued)
FORMATTED PROBLEM STATEMENT

Purpose

To present in the Language format, all description given about one or more names in an Analyzer data base.

Information Presented

The report presents for each name used as input when generating the report, all information directly available in the data base for that name and its relationships with other names in the data base. Since this report can be generated for any type of name and the report presents all Language relationships specified for each name type, it must present all relationships as specified in "User Requirement Language, Language Reference Manual".*

Format

All information presented in this report is presented as legal Language statements and is formatted according to the values of the margin parameters. The margin parameters have the following effects on the format:

AMARG - indicates the column at which the first name of a name pair is to be outputted.

BMARG - indicates the column at which the second name of a name pair is to be outputted.

CMARG - specifies the number of columns between SMARG and where the text starts for a comment entry.

NMARG - indicates the column where the user defined name in a section header is to be outputted.

RNMARG - indicates the column where the first name of a name list or name used in a Language statement is outputted.

SMARG - specifies the right hand margin for names in a name list.

*ISBOS Working Paper No. 68..
Figure 33 illustrates the margin parameters with respect to a part of an actual FORMATTED PROBLEM STATEMENT.

| HMARC | PROCESS: | salaried-employee-processing; |
| CMARC | DESCRIPTION: | This process produces the pay statement for salaried employees once a month; |
| SMARC | SYNONYMS ARE: | s-emp-proc, s3; |
| KMARC | ATTRIBUTES ARE: | complexity-level | high |
| AMARC | |
| BMARC | |

All Language statements presented in the FPS are numbered sequentially.

For each type of Language section, the statements within the section are ordered as given in Table 6. Sections in the FPS which describe undefined names or relationships not allowed by the syntax of the Language are presented as comment statements, i.e., preceded by the characters /* and succeeded by the characters */.
CONDITION section

SYNONYMS
DESCRIPTION
SEE-MEMO
KEYWORDS
ATTRIBUTES
BECOMING TRUE IS CALLED
BECOMING FALSE IS CALLED
TRUE WHILE
FALSE WHILE
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE

DEFINE section

SYNONYMS
DESCRIPTION
SEE-MEMO
KEYWORDS
ATTRIBUTES
APPLIES
SUBSETTING-CRITERION
MAINTAINED
/* CONTAINED */
/* CONNECTIVITY */
/* CARDINALITY */
/* HAPPENS */
/* VALUE */
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE

ELEMENT section

SYNONYMS
DESCRIPTION
SEE-MEMO
KEYWORDS
ATTRIBUTES
CONTAINED
ASSOCIATED WITH
IDENTIFIES
SUBSETTING-CRITERION
DERIVED
USED
UPDATED
VALUES
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE

TABLE 6

197
ENTITY section

SYNONYMS
DESCRIPTION
SEE-MEMO
KEYWORDS
ATTRIBUTES
CONTAINED
ASSOCIATED WITH
IDENTIFIES
SUBSETTING-CRITERION
DERIVED
USED
UPDATED
VALUES
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE

EVENT section

SYNONYMS
DESCRIPTION
SEE-MEMO
KEYWORDS
ATTRIBUTES
TRIGGERS
WHEN TRUE
WHEN FALSE
ON INCEPTION
ON TERMINATION
HAPPENS
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE

GROUP section

SYNONYMS
DESCRIPTION
SEE-MEMO
KEYWORDS
ATTRIBUTES
CONTAINED
CONSISTS
IDENTIFIES
SUBSETTING-CRITERION
ASSOCIATED-WITH
DERIVED
USED
UPDATED
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE

TABLE 6 (Continued)
INPUT section

SYNONYMS
DESCRIPTION
SEE-MEMO
KEYWORDS
ATTRIBUTES
PART
SUBPARTS
CONTAINED
CONSISTS
GENERATED
RECEIVED
USED
HAPPENS
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE

INTERFACE section

SYNONYMS
DESCRIPTION
SEE-MEMO
KEYWORDS
ATTRIBUTES
PART
SUBPARTS
GENERATES
RECEIVES
RESPONSIBLE
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE

INTERVAL section

SYNONYMS
DESCRIPTION
SEE-MEMO
KEYWORDS
ATTRIBUTES
CONSISTS
/* CONTAINED */
/* HAPPENS */
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE

MEMO section

SYNONYMS
DESCRIPTION
KEYWORDS
ATTRIBUTES
APPLIES
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE

TABLE 6 (Continued)
OUTPUT section

SYNONYMS
DESCRIPTION
SEE-MEMO
KEYWORDS
ATTRIBUTES
PART
SUBPARTS
CONTAINED
CONSISTS
DERIVED
GENERATED
RECEIVED
HAPPENS
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE

PROBLEM-DEFINER section

SYNONYMS
DESCRIPTION
SEE-MEMO
KEYWORDS
ATTRIBUTES
MAILBOX
RESPONSIBLE
SECURITY
SOURCE

PROCESS section

SYNONYMS
DESCRIPTIONS
SEE-MEMO
KEYWORDS
ATTRIBUTES
PART
SUBPARTS
UTILIZED
UTILIZES
RECEIVES
GENERATES
DERIVES
MAINTAINS
UPDATES
USES
PROCEDURE
INCEPTION-CAUSES
TERMINATION-CAUSES
TRIGGERED
HAPPENS
RESPONSIBLE-PROBLEM-DEFINER
SECURITY
SOURCE
TABLE 6 (Continued)
Options and Alternatives

Aside from the variations in the report format as the result of assignment of the margin parameters, there are a few other parameters which modify the format in some way. These parameters are given below with the effect they have on the report format.

1. **NEW-LINE** - specifies that the first name in a list of names for Language statements is started on the line following the statement header.

2. **NONEW-LINE** - specifies that the names in the name list begin on the same line as the Language statement.

For example, with **NEW-LINE** in effect, a Language statement would be printed as follows:

```
SYNONYMS ARE:
s-emp-proc,
s3;
```

With **NONEW-LINE** in effect, the statements are printed as follows:

```
SYNONYMS ARE:  s-emp-proc,
                s3;
```

2. **NEW-PAGE** - specifies that each section (description of single name) presented in the report would be printed beginning at the top of a new page.

3. **NONEW-PAGE** - specifies that sections are printed one after another.

When maintaining an up-to-date copy of the FPS, it is often desirable to generate the FPS for all names in the data base with the **NEW-PAGE** option. Any modifications to the description in the data base can be recorded by generating an FPS (with the **NEW-PAGE** option again) for those names affected by the modification.

3. **ONE-PER-LINE** - specifies that the names in a name list within a given statement be presented one per line.

4. **SEVERAL-PER-LINE** - specifies that the names in a name list be presented as many as possible on a line.

Some information in the FPS can be included or left out, depending on the parameters used when generating it. Each parameter and its effect is given below.

1. **COMMENT** - specifies that comment statements for descriptions of undefined names and relationships not allowed by the Language syntax are to be included where applicable in the report.
2. DEFINE - specifies that descriptions for names which are described by a DEFINE section (ATTRIBUTE, ATTRIBUTE-VALUE, KEYWORD, MAILBOX, SECURITY, SOURCE, SUBSETTING-CRITERION, and SYSTEM-PARAMETER names) are included in the report when these names are given as input.

NODEFINE - specifies that the description of any name described by a DEFINE section is not presented in the report.

3. DESG - specifies that the descriptions for names which are SYNONYMS for other names in the data base are presented in the report by the DESIGNATE section.

NODESG - specifies that the descriptions for names that are SYNONYMS are not presented in the report.

For each name given as input to the command producing the report, the report presents the appropriate section to describe the name. For example, when a PROCESS name is given, it is described in a PROCESS section. Therefore, when a SYNONYM name is given as input, the report describes the SYNONYM by a DESIGNATE section rather than presenting the description of the name the SYNONYM name applies to.

An INDEX for the report is generated when the INDEX parameter is used.

The report may be generated for a single input name (via the NAME parameter) or for a collection of input names either specified by the user or retrieved via NAME-GEN.

Analysis

For each name given as input, the software finds the name in the data base. If the name cannot be found, the message:

/* NAME NOT FOUND IN D.B. */

is printed on the report. Each relationship the name has with other names is printed in the format of a legal Language statement. If no statement exists, the relationship is presented as a comment entry. Since no Language section is available to describe an UNDEFINED name, the description of the name and relationships it has with other names are presented as a comment statement.

Usage

Since the FPS presents all the description given about each name in the data base, the report is beneficial in checking the accuracy of each description. It is usually recommended that an FPS for all names be maintained as a reference and updated when changes are made to the data base.

203
Examples

Figure 34 presents an FPS for a single name. This example was generated by the following command:

```
FORMATTED-PROBLEM-STATEMENT NAME=payroll-processing
```

Figure 35 presents the report for all ENTITY names defined in a particular data base. This example was generated by the following commands:

```
NAME-GEN ENTITY
FPS
```
FORMATTED PROBLEM STATEMENT

1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
payroll-processing;

SUBS AS: payproc,
pl;

OPTION;
This process represents the highest level process in the target system. It accepts and processes all inputs and produces all outputs;

SUBS AS: goals-memo,
process-memo;
SUBS AS: new-employee-processing,
terminating-employee-processing,
hourly-employee-processing,
salaried-employee-processing,
shared-routines;
SUBS AS: employee-information;
SUBS AS: paysia-system-outputs;
SUBS AS: payroll-master-information;
SUBS AS: employee-information,
payroll-master-information;
SUBS AS: PROBLEM-DEFINED AS:
michael-bastardo;

END OF EOF EOF

Figure 34.
department-record;

This record holds all current data relevant about each department;

and IS: department-file;

and OF:

department,

payroll-master-information;

and OF:

department,

supervisor,

number-of-employees,

total-budge,

remaining-funds;

/ IT W/ RELATED TO:

hourly-employee-record

VIA dept-hourly-emp-relation;

/ IT W/ RELATED TO:

all-time-employee-record

VIA dept-relation-emp-relation;

/ MADE BY: department;

/ EXITS: no-of-departments;

hourly-employee-record;

/ EXITS:

/ IT IS: h-emp-rec;

/ EXIT;

This record holds all current data about each hourly employee;

/ MADE BY: hourly-employee-file;

/ EXITS: payroll-master-information;

/ EXITS OF:

employee-name,

employee-id-termination-number,

social-security-number,

pay-state-social,

address,

phone,

annual-ear-date.

Figure 35

206
number-of-deductions,
cumulative-gross-pay,
cumulative-federal-deductions,
cumulative-state-deductions,
cumulative-fica-deductions,
age,
sex,
status-code,
cumulative-hours;

/ ISAT % RELATED TO:
department-record
VIA dept-hourly-emp-relation;
filed BY: employee-identification-number;
ensured BY: hourly-record-creation;
ensured BY: hourly-employee-processing;

if:
term-employee-generation
TO DERIVE term-employee;
valid;
valid;
This record is changed about once a week;
validity IS: no-restrictions;
salaried-employee-record;

YES ARE: s-emp-rec;
filed;
This record holds all current data about salaried employees;
REFER TO: payroll-master-information,
salaried-employee-file;

LIST OF:
employee-name,
employee-identification-number,
social-security-number,
pay-grade-code,
address,
phone,
employment-info,
number-of-deductions,
department,
cumulative-federal-deductions,
FORMATTED PROBLEM STATEMENT

cumulative-gross-pay,
cumulative-state-deductions,
cumulative-fica-deductions,
age,
sex,
status-code;

/* RELATED TO:
department-record
via dept-salaried-emp-relation;
FIED BY: employee-identification-number;
SO BY: salaried-record-creation;
IF: salaried-employee-processing;
IF:
term-report-entry-generation
TO DESIRE term-report-entry;
RECORD: no-of-salaried-employees;
VALID;
This record is changed about once a month;
ITY IS: no-restrictions;

IF EOF EOF

Figure 35 (Continued)
208
FREQUENCY REPORT

Purpose

To present all information based on the use of the HAPPENS statement in a particular Analyzer data base.

Information Presented

The report presents size and volume information about all INPUT, OUTPUT, PROCESS and EVENT names defined in the data base with respect to the HAPPENS statements connected to those types of names. An entry is made in the report for each INPUT, OUTPUT, PROCESS and EVENT with a HAPPENS statement and the entries are grouped by the INTERVAL over which the HAPPENS statement is effective. SYSTEM-PARAMETERS are contained in each entry to define the number of times the name associated to the entry occurs within the INTERVAL.

Format

The report presents all HAPPENS statements relative to a specific INTERVAL whose name is printed on a heading. For each INTERVAL, three headings are printed and the frequency information pertaining to the particular INTERVAL is listed below these headings. The headings are:

- **NAME**: the names of the INPUTS, OUTPUTS, PROCESSES and EVENTS which HAPPEN within the designated INTERVAL are listed.
- **TYPE**: the name type of each of the names given under NAME is listed.
- **TIMES HAPPENS**: the SYSTEM-PARAMETER or number used in the HAPPENS statement for each of the names given under NAME is listed.

The INTERVALS are presented alphabetically in the report. No ordering is imposed on the names listed.

Options and Alternatives

No options are available for this report.

Analysis

The data base is searched for an INTERVAL name that is connected to a HAPPENS relationship. If none are found, the message:

**URA286:FRINTV: NO FREQUENCY INFORMATION IN DATA BASE**

is printed. If an INTERVAL name is found, it is printed out and for each HAPPENS statement connected to it the name the HAPPENS statement applies to, its name type and associated SYSTEM-PARAMETER are retrieved and printed out.
The search for another INTERVAL is continued until all names have been tested.

Usage

The report is helpful to analysts in checking that all items in the description which are to be logically related via their frequency are grouped together.

The report is also beneficial to system designers when considering the relationships of various parts of system with respect to frequency, and the amount of input and output to be handled by the target system.

Examples

Figure 36 presents the FREQUENCY REPORT for a small Language description of a target system.
### Frequency Report

<table>
<thead>
<tr>
<th>Event</th>
<th>Type</th>
<th>Times Happens</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp-processing-init</td>
<td>EVENT</td>
<td>one</td>
</tr>
<tr>
<td>termination-form</td>
<td>INPUT</td>
<td>several</td>
</tr>
<tr>
<td>reporting</td>
<td>OUTPUT</td>
<td>no-of-payroll-processing</td>
</tr>
<tr>
<td>monthly-report</td>
<td>INPUT</td>
<td>several</td>
</tr>
<tr>
<td>employee-form</td>
<td>OUTPUT</td>
<td>one</td>
</tr>
<tr>
<td>military-certificate</td>
<td>INPUT</td>
<td>several</td>
</tr>
<tr>
<td>processing</td>
<td>PROCESS</td>
<td>no-of-payroll-processing</td>
</tr>
<tr>
<td>salary-processing</td>
<td>PROCESS</td>
<td>one</td>
</tr>
</tbody>
</table>

### Frequency Report

<table>
<thead>
<tr>
<th>Event</th>
<th>Type</th>
<th>Times Happens</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp-processing-init</td>
<td>EVENT</td>
<td>one</td>
</tr>
<tr>
<td>mp-processing-init</td>
<td>EVENT</td>
<td>one</td>
</tr>
<tr>
<td>mp-processing-init</td>
<td>EVENT</td>
<td>one</td>
</tr>
<tr>
<td>- employee-report</td>
<td>OUTPUT</td>
<td>one</td>
</tr>
<tr>
<td>- employee-report</td>
<td>OUTPUT</td>
<td>one</td>
</tr>
<tr>
<td>- employee-report</td>
<td>OUTPUT</td>
<td>one</td>
</tr>
<tr>
<td>- employee-report</td>
<td>OUTPUT</td>
<td>one</td>
</tr>
<tr>
<td>- employee-report</td>
<td>OUTPUT</td>
<td>one</td>
</tr>
<tr>
<td>- employee-report</td>
<td>OUTPUT</td>
<td>one</td>
</tr>
<tr>
<td>- employee-report</td>
<td>OUTPUT</td>
<td>one</td>
</tr>
<tr>
<td>- employee-report</td>
<td>OUTPUT</td>
<td>one</td>
</tr>
</tbody>
</table>

Figure 36

211
IDENTIFIER INFORMATION REPORT

Purpose

To present all information based on the use of IDENTIFIERS for ENTITIES in a particular Analyzer data base.

Information Presented

If IDENTIFIER names are used as input when generating this report (and the IDENTIFIER parameter is specified), those ENTITIES which the input names IDENTIFY are presented in the report.

If ENTITY names are used as input when generating the report (and the ENTITY parameter is specified), those IDENTIFIERS which the input names are IDENTIFIED by are presented in the report.

In either case, the information is presented as a matrix. An analysis of the information in the matrix produces some statistics showing the number of IDENTIFIERS each ENTITY in the matrix had and the number of ENTITIES each IDENTIFIER identifies in the matrix.

Format

If the IDENTIFIER parameter is used when generating the report, any names given as input which do not IDENTIFY any ENTITY in the data base are flagged at the beginning of the report. If the ENTITY parameter is used when generating the report, any names given as input which are not IDENTIFIED by any IDENTIFIER in the data base are flagged at the beginning of the report.

Two lists of names are then presented, one labeled ROW NAMES and the other COLUMN NAMES. If the IDENTIFIER parameter was used when generating the report, the names designated as ROW NAMES are those which were given as input. If the ENTITY parameter was used when generating the report, the names designated as COLUMN NAMES are those which were given as input.

In any case, each name under ROW NAMES IDENTIFIES one or more names under COLUMN NAMES and each name under COLUMN NAMES is IDENTIFIED by one or more names under ROW NAMES.

A matrix is then printed to show the relationships between the names designated as ROW NAMES (which are represented by the rows of the matrix) and the names designated as COLUMN NAMES (which are represented by the columns of the matrix). The rows and columns of the matrix are numbered to correspond to the number assigned to each name in the list of ROW NAMES and COLUMN NAMES, respectively.

An asterisk (*) entry at the intersection of a particular row and column of the matrix designates that the name represented by the row IDENTIFIES the ENTITY represented by the column.

Inspection of an entire row reveals all ENTITIES that a particular name (represented by the row) IDENTIFIES. Inspection of an entire column reveals all IDENTIFIERS for the particular name represented by the column.
A summary section is also included in the report presenting for each ROW NAME:

- The row it was represented by in the matrix (ROW)
- Its name type (TYPE)
- The number of * entries in its row (or the number of ENTITIES it IDENTIFIES) (COUNT)

The summary presents for each COLUMN NAME:

- The column it was represented by (COLUMN)
- Its name type (TYPE)
- The number of * entries in its column (or the number of IDENTIFIERS for it) (COUNT)

The summary section for ROW and COLUMN names is ordered in decreasing order of COUNT.

Options and Alternatives

The report must be generated using either the IDENTIFIER or ENTITY parameter. If the IDENTIFIER parameter is used, all names given as input must be IDENTIFIER names. If the ENTITY parameter is used, all names given as input must be ENTITY names.

The report may be generated for a single input name (via the NAME parameter) or for a collection of input names either specified by the user or retrieved via NAME-GEN.

Analysis

For each name given as input, the software finds the name in the database. If the name is not found, the message:

URA099:IDENTR: NAME NOT IN D. B. - or
URA052:IDENTC: NAME NOT IN D. B. -

is printed depending on whether the IDENTIFIER or ENTITY parameter was specified for the command, respectively.

If the IDENTIFIER parameter is used, each name given as input is checked that it IDENTIFIES one or more ENTITIES. If it does not, the name is listed under the message:

URA304:IDENTR: THE FOLLOWING NAMES DO NOT IDENTIFY ANYTHING

The ENTITIES that are IDENTIFIED by the input names are found. The list of all input names and the list of all ENTITIES retrieved is then printed.

If the ENTITY parameter is used, each name given as input is checked that it is IDENTIFIED by one or more IDENTIFIERS. If it does not, the name is listed under the message:

URA305:IDENTC: THE FOLLOWING NAMES ARE NOT IDENTIFIED BY ANYTHING
The IDENTIFIERS for the ENTITIES given as input are found. The list of all input names and the list of all IDENTIFIERS retrieved is then printed.

A matrix is printed out to illustrate the relationships between the names in the two lists and each relationship is designated by an asterisk.

A summary is then produced by counting the number of asterisks appearing in each row and column of the matrix.

Usage

The report presents information that aids the analyst in checking the completeness and consistency of the problem statement by:

1 - identifying those ENTITIES which do not have IDENTIFIERS. This can be accomplished by the following Analyzer commands:

```
NAME-GEN ENTITY
ENTITY-IDENTIFIER ENTITY
```

2 - being in an easy-to-analyze format to check that the IDENTIFIERS defined for the problem statement are being used properly. For example, a typing error may result in an IDENTIFIER being used in the wrong context.

The report aids the system designer by presenting those ENTITIES with the same IDENTIFIERS and aids in determining a consistent and well-defined identifier coding structure.

Examples

Figure 37. presents the report using the ENTITY parameter. The Analyzer commands used to generate this example were:

```
NAME-GEN ENTITY
ENTITY-IDENTIFIER ENTITY
```
IDENTIFIER INFORMATION REPORT

COLUMN NAMES

<table>
<thead>
<tr>
<th>ELEMENT</th>
<th>ENTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 department-record</td>
<td>ENTITY</td>
</tr>
<tr>
<td>2 hourly-employee-record</td>
<td>ENTITY</td>
</tr>
<tr>
<td>3 salaried-employee-record</td>
<td>ENTITY</td>
</tr>
</tbody>
</table>

THE ROWS ARE IDENTIFIERS OF THE COLUMNS WITH *S

<table>
<thead>
<tr>
<th>123</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1* 1</td>
</tr>
<tr>
<td>2 1 **1</td>
</tr>
<tr>
<td>-----</td>
</tr>
</tbody>
</table>

Figure 37

215
IDENTIFIER INFORMATION REPORT

THE NUMBER OF COLUMNS IDENTIFIED BY THE ROWS**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>employee-identification-number</td>
<td>ELEMENT</td>
<td>2</td>
</tr>
<tr>
<td>column</td>
<td>ELEMENT</td>
<td>1</td>
</tr>
</tbody>
</table>

THE NUMBER OF ROWS THAT IDENTIFY THE COLUMNS**

<table>
<thead>
<tr>
<th>C</th>
<th>TYPE</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>sent-record</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>hourly-employee-record</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>full-employee-record</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 37 (Continued)

216
KWIC INDEX

Purpose

To present, in an easy to inspect format, logical groupings of names defined in a particular Analyzer data base with respect to the spelling of the names.

Information Presented

The report presents, for all those names used as input, an alphabetical listing consisting of an entry for each name as it appears as input and entries for each permutation of the name (about the dashes). For example, if the name hourly-employee-form was supplied as input, there would be entries for

- employment-form
- form
- hourly-employment-form
- hourly
- employment

in the report. When there are several names used as input then all names with the word "employment" in them would have entries grouped together, all those with "form" would be grouped together, etc.

Format

The entries in the report are ordered alphabetically and numbered sequentially. There are two parts of each entry, the right hand side of the entry presents that part of the user defined name that has been stripped off for a permutation of the name and the left hand side of the entry presents the remaining part of the name. The distance (the number of columns) between the right and left sides of the entry can be varied by the value assigned to the DIF parameter.

Options and Alternatives

The DIF parameter may take on any value from 2 to 52.

Analysis

Each name given as input to the software generating the report is inspected, separated at the dashes in the name and a list is formed consisting of the original name and all permutations of the name.

After all names in the input list have been processed, the newly formed list is sorted and presented as the report.

Usage

The KWIC INDEX aids analysts in maintaining naming conventions used in the target system description process and for finding names in the Description based on the keyword within the names.

It is often desirable to use some conventions in assigning names to objects defined in a target system description and the KWIC INDEX aids in maintaining these. For example, by issuing the following Analyzer
commands:

    NAME-GEN   ELEMENT
    KWIC

a KWIC INDEX is presented for all ELEMENT names so that consistency of naming can be checked.

Examples

Figure 38 presents a KWIC INDEX for INPUT names defined in a small problem statement.
NAME GEN

Purpose

To present all names in the data base with respect to some selection criteria (as specified by parameters).

Information Presented

The report presents a list of names and their corresponding name types. The types of the names in the list are specified by the parameters used in the command producing the report. For example, when the INTERFACE and PROCESS parameters are given, the report presents all INTERFACE and PROCESS names in the data base. When the PROCESS and KEYWORD=terminal parameters are given, the report presents all PROCESSES in the data base that have the KEYWORD "terminal" associated as part of their description.

Format

An entry in the report is printed for each name retrieved and consists of:

- the name retrieved, and
- the name type of the name

The entries within the report are ordered in one of two ways: alphabetically on the names when the ORDER=ALPHA parameter is in effect, and alphabetically on the names within name type (which are also ordered alphabetically) when the ORDER=HYTYPE parameter is in effect.

Options and Alternatives

Several parameters for the command generating the report specify the types of names that will be retrieved from the data base and presented in the report. The following types of names can be retrieved when the parameter (of the same name) is given for the command:

ATTRIBUTE
ATTRIBUTE-VALUE
CONDITION
ELEMENT
ENTITY
EVENT
GROUP
INPUT
INTERFACE
INTERVAL
KEYWORD
MAILBOX
MEMO
OUTPUT
PROBLEM-DEFINER
PROCESS
SECURITY
SET
SOURCE
SUBSETTING-CRITERION
SYSTEM-PARAMETER
UNDEFINED

If none of these parameters are specified, no names will be presented. If the ALL parameter is specified, the names of all name types except SYNONYM and UNDEFINED will be presented. If the TOTAL parameter is specified, every name in the data base is presented.
If all names in the data base except for SYNONYMS, UNDEFINED names and
PROBLEM-DEFINERS were presented, the following parameters would be
used:

\textbf{ALL NOPROBLEM-DEFINER}

A 'NO' prefix for any of the name type parameters designates that names
of that type are not to be presented. Specifying ALL has the effect of
setting switches to indicate that all name types are to be presented.
Specifying NOPROBLEM-DEFINER after the ALL parameter has the effect of
setting a switch to indicate that PROBLEM-DEFINER names are not to be
retrieved.

Four other parameters specify the types of names retrieved from the
data base. Names used as IDENTIFIERS of ENTITIES can be presented by
specifying IDENTIFIER as a parameter. Names which are used as IDENTIFIERS
and are also defined as ELEMENTS are presented when the IDENTIFIER-
ELEMENT parameter is specified. Names which are used as IDENTIFIERS and
are also defined as GROUPS are presented when the IDENTIFIER-GROUP para-
meter is specified.

All names which belong to the SUBPARTS structure for a given name (as
would be retrieved for the STRUCTURE report) can be retrieved by specifying:

\textbf{SUBPARTS-OF=name}

where the name is an INPUT, OUTPUT, PROCESS or INTERFACE name which has
SUBPARTS information defined for it. The number of levels to go down
and retrieve names to present in the report is specified by the SUBLEVEL
parameter. If SUBLEVEL=0, then all levels of names are presented.
If SUBLEVEL=1, then only those names which are PART OF the SUBPARTS OF
name are presented. The following picture may clarify the association
between the value of SUBLEVEL and the names presented.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{subparts-diagram.png}
\caption{Subparts Structure Diagram}
\end{figure}

Generation of the report with \texttt{SUBPARTS-OF=S1} and \texttt{SUBLEVEL=3} would
present S2, S3, S4, S5, S6, S7, S8, S9, S10 and S11 in the report.
Generation of the report with \texttt{SUBPARTS-OF=S1} and \texttt{SUBLEVEL=1} would
present the names S2, S3, and S4.
Of the various types of names retrieved by any of the above parameters, additional selectivity is provided by the KEY and PD parameters. The KEY parameter specifies that for all names retrieved, only those which have the designated KEYWORD are presented in the report. For example, if the following parameters were given:

\[ \text{PROCESS \hspace{1em} KEY=terminal} \]

only those PROCESSES with the KEYWORD "terminal" would be presented in the report.

Likewise, the PD parameters specifies that for all names retrieved, only those which have the designated PROBLEM-DEFINER are presented by the report. When both of these parameters are used together, only those names which satisfy both requirements are presented. For example, if the parameters:

\[ \text{INPUT \hspace{1em} OUTPUT \hspace{1em} KEY=weekly \hspace{1em} PD=michel-j-bastarache} \]

were given only those INPUTS and OUTPUTS which had the KEYWORD "weekly" and the PROBLEM-DEFINER "michel-j-bastarache" would be presented in the report.

Specification of the SYNONYM parameter presents all SYNONYMS for each name retrieved in the report in addition to the basic form of the name. If only the SYNONYMS are desired, the basic names may be suppressed by specifying the NOBASIC and SYNONYM parameters. With standard defaults in effect, the BASIC and NOSYNONYM parameters are used.

Analysis

Each name defined in the data base is checked against the parameters for the command. If it satisfies the requirements as specified by the parameters, it is placed in a list. After all names in the data base have been checked, the list is sorted as the report.

If NAME-GEN is generated for an empty data base, the message:

\[ \text{URA049: MAINNG: NO NAMES IN DATA BASE} \]

will be printed.

If the PD parameter is used in generating the report and the name given as the PD value is not a name defined in the data base, the message:

\[ \text{URA047: MAINNG: PD NOT FOUND IN DATA BASE} \]

will be printed. If the name is found, but is not a PROBLEM-DEFINER and related to any names in the data base, the message:

\[ \text{URA037: FNDPD: THIS IS NOT A PD FOR ANY NAMES} \]
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>tax-withholding</td>
</tr>
<tr>
<td>tax-withholding</td>
<td>hourly</td>
</tr>
<tr>
<td>hourly</td>
<td>salaried</td>
</tr>
<tr>
<td>salaried</td>
<td>employment-termination</td>
</tr>
<tr>
<td>employment-termination</td>
<td>hourly-employment</td>
</tr>
<tr>
<td>employee</td>
<td>employment-termination</td>
</tr>
<tr>
<td>employment-termination</td>
<td>salaried-employment</td>
</tr>
<tr>
<td>salaried-employment</td>
<td>employment</td>
</tr>
<tr>
<td>employment</td>
<td>tax</td>
</tr>
</tbody>
</table>

Figure 38

222
If the KEY parameter is used in generating the report and the name given as the KEY value is not a name defined in the data base, the message:

    URA048: MAINNG:  KEYWORD NOT FOUND IN DATA BASE -

will be printed. If the name is found, but is not a KEYWORD and not related to any names in the data base, the message:

    URA030: FNDKEY:  THIS IS NOT A KEYWORD FOR ANY NAMES -

is printed.

If the SUBPARTS-OF parameter is used in generating the report and the name given as the SUBPARTS-OF value is not a name defined in the data base, the message:

    URA059: MAINNG: SUBPARTS-OF NAME NOT IN DATA BASE -

is printed. If the name is found but it is not a PROCESS, INPUT, OUTPUT or INTERFACE name, the message:

    URA089: MAINNG: NAME MUST BE INPUT, OUTPUT, PROCESS OR INTERFACE FOR "SO" PARAMETER

is printed.

Usage

It is an important aid to the analyst in obtaining other reports and outputs. For example, the analyst can ask for a list of all SET, ENTITY and GROUP names and with this list then ask for a CONTENTS REPORT for these names.

It is also used by the analyst as a reference to what names have been used and how they have been used (i.e., what their name types are).

The output can also be used effectively by project management to measure productivity of the project members. This can be done by retrieving a list of all names in the data base defined by a particular problem definier (analyst) and comparing it to previous lists.

Finally, the NAME GEN output can become an integral part of the final specifications as it acts as a directory in specifying name lists corresponding to certain selection criteria (a directory of all data elements may be desired before a section which deals with the definition of each element in detail).

Examples

Figure 39 presents a NAME GEN report produced for all INPUT names defined in a data base. The command used to generate this example was:

    NAME-GEN INPUT
Figure 40 presents a NAME-GEN report produced for all INPUT, OUTPUT and ENTITY names in a data base given in alphabetical order. The command used to generate this example was:

    NAME-GEN  INPUT  OUTPUT  ENTITY

Figure 41 presents the report for all names in the data base which have "michel-j-bastarache" defined as their PROBLEM DEFINER. The command used to generate this example was:

    NAME-GEN  ALL  PD=michel-j-bastarache

Figure 42 presents the report for all PROCESS names in the data base which have "terminal" defined as one of their KEYWORDS. The command used to generate this example was:

    NAME-GEN  PROCESS  KEY=terminal

Figure 43 presents the report for SUBPARTS for the PROCESS name "payroll-processing". Only the first sublevel of the SUBPARTS structure is presented. Also, the synonym of "payroll-processing" was specified for the SUBPARTS-OF parameter rather than the full name. The command used to generate this example was:

    NAME-GEN  SUBPARTS-OF=payproc  LEVELS=1
<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>point-record</td>
<td>ENTITY</td>
</tr>
<tr>
<td>2</td>
<td>point-information</td>
<td>INPUT</td>
</tr>
<tr>
<td>3</td>
<td>point-termination-form</td>
<td>INPUT</td>
</tr>
<tr>
<td>4</td>
<td>listing</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>5</td>
<td>employee-report</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>6</td>
<td>employee-record</td>
<td>ENTITY</td>
</tr>
<tr>
<td>7</td>
<td>employee-report</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>8</td>
<td>employee-report</td>
<td>INPUT</td>
</tr>
<tr>
<td>9</td>
<td>employee-form</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>10</td>
<td>output</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>11</td>
<td>patient-record</td>
<td>ENTITY</td>
</tr>
<tr>
<td>12</td>
<td>patient-report</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>13</td>
<td>patient-record</td>
<td>INPUT</td>
</tr>
<tr>
<td>14</td>
<td>hospital-certificate</td>
<td>INPUT</td>
</tr>
<tr>
<td>15</td>
<td>hospital-report</td>
<td>OUTPUT</td>
</tr>
<tr>
<td>16</td>
<td>record</td>
<td>INPUT</td>
</tr>
</tbody>
</table>

**Figure 40**

226
Figure 41
NAME LIST

Purpose

To present a list of all names defined in a particular Analyzer
data base, the name type associated with the name and SYNONYMS
defined for each name.

Information Presented

The report presents every name currently defined in the user's
data base, the name type associated with each name and the
SYNONYMS associated with each name.

Format

An entry in the report is printed for each name in the Analyzer
data base and consists of:

- the name,
- the name type of the name, and
- any SYNONYMS for the name,

which are listed under the headings: NAME, TYPE, and SYNonyms,
respectively.

The entries within the report are ordered in one of two ways:
apphabetically on the names when the ORDER=ALPHA parameter is
in effect and, alphabetically on the names within name type
(which are also ordered alphabetically) when the ORDER=BYTYPE
parameter is in effect.

If no SYNONYMS are available for a particular name a dash (-) is
printed under the SYNONYM heading. If more than one SYNONYM exists
for a name, they are listed beneath each other.

Options and Alternatives

No options other than the ORDER=ALPHA and ORDER=BYTYPE features
are available for this report.

Analysis

Each name in the data is inspected and its name type and any
SYNONYMS for the name are retrieved. After this information has
been collected for all names in the data, it is sorted and
presented as the report.

Usage

The report is intended to be used as a director facility by anyone
needing a reference including all names defined in the data base.
Examples

Figure 44 presents the NAME LIST report generated with the ORDER=BYTYPE option.
<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Synonym</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>self-type</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>security-level</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>standard</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>normal</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>nvisc</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>metric</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>near</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>time</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>scheduled</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>metadata</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>data-for-employee-found</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>employee-id-valid</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>the-url-for-employee</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>cards-ready</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>client-number</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>date</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>client-number</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>client-number</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>client-number</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>client-number</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>client-number</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>client-number</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>client-number</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>client-number</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>client-number</td>
<td></td>
</tr>
</tbody>
</table>

Figure 44
<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Type</th>
<th>Synonym</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>current-status</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>name</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>pay</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>per-day</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>number</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>date</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>department-change</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>sex</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>result-of-inductions</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>title-employee</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>total-hours-worked</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>result-of-inductions</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>result-of-employees</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>total-hours-worked</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>result</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>title</td>
<td>ELEMENT</td>
<td></td>
</tr>
</tbody>
</table>

Figure 44 (Continued)
<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>SYNONYM</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>EVENT</td>
<td>e-emp-processing-init</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>EVENT</td>
<td>e-employ-processing-init</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>GROUP</td>
<td>e-error-checks</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>EVENT</td>
<td>e-ok-emp-processing-init</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>EVENT</td>
<td>e-action-processing-init</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>EVENT</td>
<td>e-actioning</td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>EVENT</td>
<td>e-key-check</td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>GROUP</td>
<td>e-action</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>EVENT</td>
<td>e-action-ntale-data</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>EVENT</td>
<td>e-action-data</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>EVENT</td>
<td>e-action-entry</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>EVENT</td>
<td>e-action-pay-data</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>EVENT</td>
<td>e-action-entry</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>EVENT</td>
<td>e-action-pay-data</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>EVENT</td>
<td>e-action-ntale-data</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>EVENT</td>
<td>e-action-print</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>EVENT</td>
<td>e-action-print</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>EVENT</td>
<td>e-action-print</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>EVENT</td>
<td>e-action-print</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>EVENT</td>
<td>e-action-print</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>INPUT</td>
<td>e-action-termination-form</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>INPUT</td>
<td>e-action-termination-form</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>INPUT</td>
<td>e-action-termination-form</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>INTERFACE</td>
<td>e-action-termination-form</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>INTERFACE</td>
<td>e-action-termination-form</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>INTERFACE</td>
<td>e-action-termination-form</td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>INTERFACE</td>
<td>e-action-termination-form</td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>INTERFACE</td>
<td>e-action-termination-form</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>INTERFACE</td>
<td>e-action-termination-form</td>
<td></td>
</tr>
</tbody>
</table>

Figure 44 (Continued)
<table>
<thead>
<tr>
<th>TYPE</th>
<th>SYNONYM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL</td>
<td></td>
</tr>
<tr>
<td>FEATURE</td>
<td></td>
</tr>
<tr>
<td>PROCESS</td>
<td></td>
</tr>
<tr>
<td>OUTPUT</td>
<td></td>
</tr>
<tr>
<td>TERMINAL</td>
<td></td>
</tr>
<tr>
<td>TERMINAL-OUTPUTS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 44 (Continued)
<table>
<thead>
<tr>
<th>TYPE</th>
<th>SYNONYM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCESS</td>
<td>-</td>
</tr>
<tr>
<td>PROCESS</td>
<td>-</td>
</tr>
<tr>
<td>PROCESS</td>
<td>-</td>
</tr>
<tr>
<td>PROCESS</td>
<td>-</td>
</tr>
<tr>
<td>PROCESS</td>
<td>-</td>
</tr>
<tr>
<td>PROCESS</td>
<td>-</td>
</tr>
<tr>
<td>PROCESS</td>
<td>-</td>
</tr>
<tr>
<td>RELATION</td>
<td>-</td>
</tr>
<tr>
<td>RELATION</td>
<td>-</td>
</tr>
<tr>
<td>SECURITY</td>
<td>-</td>
</tr>
<tr>
<td>SECURITY</td>
<td>-</td>
</tr>
<tr>
<td>SET</td>
<td>dept-file</td>
</tr>
<tr>
<td>SET</td>
<td>h-emp-file</td>
</tr>
<tr>
<td>SET</td>
<td>master-file</td>
</tr>
<tr>
<td>SET</td>
<td>pay-mast</td>
</tr>
<tr>
<td>SET</td>
<td>s1</td>
</tr>
<tr>
<td>SET</td>
<td>s-emp-file</td>
</tr>
<tr>
<td>SOURCE</td>
<td>-</td>
</tr>
<tr>
<td>SYSTEM-PARAMETER</td>
<td>-</td>
</tr>
<tr>
<td>NAME TYPE</td>
<td>RELATIONSHIPS DISPLAYED</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>INTERFACE</td>
<td>FLOW - RECEIVES</td>
</tr>
<tr>
<td></td>
<td>Generates</td>
</tr>
<tr>
<td></td>
<td>STRUCTURE - PART OF</td>
</tr>
<tr>
<td></td>
<td>SUBPARTS ARE</td>
</tr>
<tr>
<td></td>
<td>DATA - RESPONSIBLE FOR</td>
</tr>
<tr>
<td>SET</td>
<td>FLOW - DERIVED</td>
</tr>
<tr>
<td></td>
<td>UPDATED</td>
</tr>
<tr>
<td></td>
<td>USED</td>
</tr>
<tr>
<td></td>
<td>STRUCTURE - SUBSET OF</td>
</tr>
<tr>
<td></td>
<td>SUBSETS ARE</td>
</tr>
<tr>
<td></td>
<td>CONSISTS</td>
</tr>
<tr>
<td></td>
<td>DATA - RESPONSIBLE-INTERFACE</td>
</tr>
<tr>
<td></td>
<td>SUBSETTING-CRITERIA</td>
</tr>
<tr>
<td>INPUT</td>
<td>FLOW - GENERATED</td>
</tr>
<tr>
<td></td>
<td>RECEIVED</td>
</tr>
<tr>
<td></td>
<td>USED</td>
</tr>
<tr>
<td></td>
<td>STRUCTURE - PART OF</td>
</tr>
<tr>
<td></td>
<td>SUBPARTS ARE</td>
</tr>
<tr>
<td></td>
<td>CONTAINED</td>
</tr>
<tr>
<td></td>
<td>CONSISTS</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>FLOW - GENERATED</td>
</tr>
<tr>
<td></td>
<td>DERIVED</td>
</tr>
<tr>
<td></td>
<td>RECEIVED</td>
</tr>
<tr>
<td></td>
<td>STRUCTURE - PART OF</td>
</tr>
<tr>
<td></td>
<td>SUBPARTS ARE</td>
</tr>
<tr>
<td></td>
<td>CONTAINED</td>
</tr>
<tr>
<td></td>
<td>CONSISTS</td>
</tr>
<tr>
<td>ENTITY</td>
<td>FLOW - DERIVED</td>
</tr>
<tr>
<td></td>
<td>UPDATED</td>
</tr>
<tr>
<td></td>
<td>USED</td>
</tr>
<tr>
<td></td>
<td>STRUCTURE - CONTAINED</td>
</tr>
<tr>
<td></td>
<td>CONSISTS</td>
</tr>
<tr>
<td></td>
<td>DATA - IDENTIFIED</td>
</tr>
<tr>
<td>NAME TYPE</td>
<td>RELATIONSHIPS DISPLAYED</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>GROUP/ELEMENT</td>
<td>FLOW - DERIVED UPDATED USED USED TO DERIVE USED TO UPDATE</td>
</tr>
<tr>
<td></td>
<td>STRUCTURE - CONTAINED CONSISTS*</td>
</tr>
<tr>
<td></td>
<td>DATA - ASSOCIATED IDENTIFIED SUBSETTING-CRITERION</td>
</tr>
<tr>
<td>PROCESS</td>
<td>FLOW - RECEIVES USES USES TO DERIVE USES TO UPDATE DERIVES GENERATES UPDATES MAINTAINS</td>
</tr>
<tr>
<td></td>
<td>STRUCTURE - PART OF SUBPARTS ARE UTILIZED BY UTILIZES</td>
</tr>
</tbody>
</table>

Table 7 (cont'd)

*This relationship only applies to GROUPS, i.e., an ELEMENT cannot CONSIST of anything.
General PICTURE Format and Limits per Page

Figure 45

* if more, report is continued on next page (except for PART OF relationship which only one per PICTURE).
PICTURE

Purpose

The PICTURE report presents flow and structure information about the problem statement in a graphical format. The PICTURE report provides the user with a detailed view of one part of the target system description (i.e., it presents all flow and structure relationships a particular name has with other names).

Information Presented

The PICTURE report can be produced for any names in the data base of the following name types:

<table>
<thead>
<tr>
<th>INTERFACE</th>
<th>SET</th>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTITY</td>
<td>GROUP</td>
<td>ELEMENT</td>
<td>PROCESS</td>
</tr>
</tbody>
</table>

The information presented in the report varies depending on which name type the report is produced for and the parameters used when generating the report. For each name type the FLOW, STRUCTURE and DATA parameters present different types of PSL relationships as retrieved from the data base. Table 7 shows the relationships presented for each name type.

Format

The PICTURE report is generated in a graphical format. The basic template for the format is shown in Figure 45. A given PICTURE report describes a single named object.

The object being described is represented by a rectangular box printed in the center of the report page. All named objects related to the center object are also represented by rectangular boxes but are arranged around the perimeter of the page. The name type (PROCESS, ELEMENT, etc.) of the represented object is printed along the top line of each box. The relationship of an object (represented by one of the perimeter boxes) with the center object is printed along the bottom line of the box. For illustrative purposes, lines extend from each box to the center box.
The preceding example is to be interpreted as follows:

```
PROCESS pr-x USES GROUP gr-z and,
PROCESS pr-x DERIVES ELEMENT el-y.
```

Should a PICTURE of a particular name exceed the page limitation as given in Figure 45, the PICTURE will be continued on succeeding pages.

The format of the relationships presented varies depending on the name type of the object being described (just as the types of relationships vary). See Figures 46-52H on how relationships are formatted.

**Options and Alternatives**

The user has the option of displaying any combination of the three types of relationships (DATA, FLOW and STRUCTURE) on the report. See Table 7 for which relationships are displayed in each of these categories for a particular name type. For example, if only the CONSISTS and CONTAINED information (STRUCTURE) were to be displayed for a particular ENTITY name, DATA and FLOW relationships could be suppressed via the NODATA and NOFLOW parameters. The parameters allowed and their effect on the report are described below:

1. **DATA**
   - specifies that data type relationships be included in each PICTURE.
   - NODATA specifies that these relationships are not included.

2. **FLOW**
   - specifies that flow type relationships be included in each PICTURE.
   - NOFLOW specifies that these relationships are not included.

3. **STRUCTURE**
   - specifies that structure type relationships be included in each PICTURE.
   - NOSTRUCTURE specifies that these relationships are not included.

An INDEX for the report is produced when the INDEX parameter is used.

The report may be generated for a single input name (via the NAME parameter) or for a collection of names either specified by the USER or retrieved via NAME-GEN.

**Analysis**

For each name given as input the software finds the name in the data base. If the name is not found the message:

```
URA066: MAINPIC: NAME NOT IN D.B. -
```

is printed. If the name is found it is checked if it is of a legal name type for which a PICTURE may be generated, i.e. a SET, INPUT, OUTPUT, ENTITY, GROUP, ELEMENT, PROCESS, or INTERFACE name. If it is not one of these name types the message:
SET PICTURE

Figure 47

243
INPUT PICTURE

Figure 48

244
GROUP/ELEMENT PICTURE

Figure 51
247

* Pertains to GROUP PICTURE only
PROCESS PICTURE

Figure 52

248
is printed. If the name is of a legal name type it is then checked if any of the relationships that can be presented in the PICTURE for that name type exist for the particular name. If none of these relationships exist, the message:

URA289: PCLRBT: NO PICTURE AVAILABLE FOR 

is printed. Otherwise, those relationships available for the name are presented in the report.

Usage

Project management can use this report to gain a basic understanding of the functions of the target system by viewing PICTURES of high level target system objects.

PICTURES provide a good means of communication among people in the project and those external to it. A graphical format is often easier to interpret than a matrix, narrative text, etc.

Problem Definors may use the PICTURE report to visually analyze the description of particular objects to check for completeness. For each type of name (e.g., PROCESS or ELEMENT) several checks can be made depending on the relationships presented in the report. Table 8 presents completeness checks that can be made by visually scanning PICTURE reports.

Examples

Figure 53 presents a PICTURE for an INPUT name "time-card."

Figure 54 presents a PICTURE for an INTERFACE name "employee."

Figure 55 presents a PICTURE for a PROCESS name "hourly-employee-processing." This example was produced by giving the following Analyzer command:

PICTURE NAME=hourly-employee-processing
<table>
<thead>
<tr>
<th>INTERFACE</th>
<th>An INTERFACE should RECEIVE an OUTPUT, GENERATE an INPUT and/or be RESPONSIBLE for a SET.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET</td>
<td>A SET should be USED by a PROCESS, DERIVED BY A PROCESS, and/or be UPDATED by a PROCESS.</td>
</tr>
<tr>
<td></td>
<td>A check can also be made that the SET has a RESPONSIBLE-INTERFACE.</td>
</tr>
<tr>
<td></td>
<td>If the SET has SUBSETS, it should also have SUBSETTING-CRITERIA.</td>
</tr>
<tr>
<td>INPUT</td>
<td>An INPUT should be RECEIVED by a PROCESS and GENERATED by an INTERFACE. An INPUT should also be USED by a PROCESS</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>An OUTPUT should be GENERATED by a PROCESS and RECEIVED by an INTERFACE. An OUTPUT should also be DERIVED by a PROCESS</td>
</tr>
<tr>
<td>ENTITY</td>
<td>An ENTITY should be USED by a PROCESS, DERIVED by a PROCESS, and/or be UPDATED by a PROCESS.</td>
</tr>
<tr>
<td></td>
<td>A check can also be made that the ENTITY is IDENTIFIED by a GROUP or ELEMENT.</td>
</tr>
<tr>
<td>GROUP/ELEMENT</td>
<td>A GROUP/ELEMENT should be USED by a PROCESS, DERIVED by a PROCESS, and/or be UPDATED by a PROCESS.</td>
</tr>
<tr>
<td></td>
<td>A check can be made that the GROUP/ELEMENT may IDENTIFY an ENTITY, be SUBSETTING-CRITERION for a SET and/or be ASSOCIATED with a RELATION.</td>
</tr>
<tr>
<td>PROCESS</td>
<td>A PROCESS should RECEIVE an INPUT, GENERATE an OUTPUT, USE a SET, ENTITY, INPUT, GROUP or ELEMENT, DERIVE a SET, OUTPUT, ENTITY, GROUP or ELEMENT, UPDATE a SET, ENTITY, GROUP or ELEMENT, and/or MAINTAIN a RELATION.</td>
</tr>
</tbody>
</table>

Table 8

Completeness Checks that may be made by using the PICTURE REPORT.
Figure 53 (continued)
*---INPUT---*
I
I time-card I
I
*---ELEMENT---*  *---ELEMENT---*  *---ELEMENT---*
I overtime- I hours- I employee-I-I
I hours- I day- I identification-I
I worked- I I-I I lion-number I
*---CONSISTS---*  *---CONSISTS---*  *---CONSISTS---*

Figure 53 (Continued)
Figure 54 (Continued)
PROCESS_CHAIN_REPORT

Purpose

To present in a graphical format the sequence of EVENTS and PROCESSES which occurs as a result of each EVENT or PROCESS specified as input.

Information Presented

For each EVENT name given as input to the software producing the report, the report presents all PROCESS names which the EVENT TRIGGERS. For each of these PROCESS names presented the report presents all EVENTS occurring ON-INCEPTION or ON-TERMINATION of the PROCESS. The report then presents all the PROCESSES which these EVENTS TRIGGER, etc., and the network continues until a loop is encountered or no more relationships are found.

For each PROCESS name given as input to the software producing the report, the report presents all EVENTS occurring ON-INCEPTION or ON-TERMINATION of the PROCESS. For each of these EVENT names presented the report presents all PROCESS names which the EVENT TRIGGERS. The report then presents all the EVENTS occurring ON-INCEPTION or ON-TERMINATION of the PROCESS, etc., and the network continues until a loop is encountered or no more relationships are found.

Format

Each name which appears on the output is shown within a box. The top line of the box indicates the name type (EVENT or PROCESS), while the bottom line shows the relationship with the preceding EVENT or PROCESS (TRIGGERED, ON-INCEPTION, ON-TERMINATION). Boxes containing related names are linked by dotted lines.

If a name joins two or more chains (strings of related names) into a loop or loops, every appearance of that name after the first will be linked to a box containing the message, "LOOPS TO PREVIOUS ENTRY."

Output is continued across page boundaries. If the right edge of one page continues to the left edge of a second, the rightmost column of boxes on the first page will be repeated as the leftmost column of boxes on the second page, in order to facilitate matching of edges. Similarly, if the bottom edge of one page continues to the top edge of a second, the bottom row of boxes on the first page will be repeated as the top row of boxes on the second page.

Options and Alternatives

The report may be generated for a single EVENT or PROCESS name (via the NAME parameter) or for a collection of such names, either input by the user or obtained by use of NAME-GEN.
The number of columns and rows used on the page may be decreased from their default values of 119 and 39, respectively, via the COLUMNS and ROWS parameters. The minimum acceptable values for COLUMNS and ROWS are 38 and 14, respectively.

The number of boxes arranged horizontally or vertically on a page may be decreased from the defaults, which are the maximum numbers that will fit in each direction (depending on COLUMNS and ROWS), in order to make the output less cluttered. The parameters which can be used to do this are HORIZONTAL-BOXES and VERTICAL-BOXES. Their maximum values (for COLUMNS=119 and ROWS=39) are 6. Due to the scheme for continuing pages, their minimum values are 2.

The number of connections to be traced, starting at the given name, may be set at any positive value via the LINKS parameter. The same LINKS value is used for all names input when the FILE parameter is used.

An index, containing each name used on the report and the page(s) on which it appears, may be obtained by specifying the INDEX parameter.

Analysis

Each name given as input is first checked to see that it is in the data base and that it is either a PROCESS or EVENT name. If the name is not in the data base, the message

URA360: FILMAT: NAME NOT IN DATA BASE

will be given. If the name is of a type other than PROCESS or EVENT, the user will receive the message

URA361: FILMAT: NAME NOT EVENT OR PROCESS.

If the name passes these two tests, it is placed in a data structure which will later be used for output. The name is then used to generate a tree structure of PROCESSES and EVENTS as follows. If the name is an EVENT, all PROCESSES which it TRIGGERS are retrieved from the data base and placed on a stack. Then, the first name is removed from the stack, placed in its proper location in the data structure, and all EVENTS occurring ON-INCEPTION or ON-TERMINATION of that PROCESS are retrieved and placed on the stack. This procedure continues, with names being removed from the top of the stack, placed in the data structure, and used to obtain further names, which are then placed on the stack. At any stage of this procedure, no names will be put on the stack if one of the following is true:

1. The current name is an EVENT which TRIGGERS no PROCESSES, or a PROCESS which has no EVENTS occurring ON-INCEPTION or ON-TERMINATION.

2. The current name has been encountered earlier, and is

259
therefore at the end of a chain or forms a loop with some portion of a chain traced earlier.

3. The number of links that has been traced on the current chain is equal to the limit set by the LINKS parameter. (For every input name for which this occurs, the message

URA364: FILMAT: LINK LIMIT SPECIFIED WAS REACHED

will be printed.)

Thus, in any of these cases, the size of the stack will decrease. The entire procedure is complete when, after any search, the stack is empty. If the name input is a PROCESS name, the first search is for EVENTS occurring ON-INCEPTION or ON-TERMINATION of that PROCESS. From there, the procedure is identical to that described above.

The data structure constructed from all names found as above is broken into page-sized units and is printed a page at a time.

The process described above is repeated until no more names remain in the input stream.

Usage

The PROCESS CHAIN report presents a comprehensive view of the dynamic behavior of the PROCESSES within the target system for inclusion in the final specifications of the system or as an aid in communicating this information to others.

Programmers and System Designers in particular will find this report helpful in identifying and optimizing the system logic. If the PROCESSES are defined to the level of computable statements, the PROCESS CHAIN report will essentially chart out the program logic.

Problem Definiers may use the PROCESS CHAIN report to visually analyze the description of particular objects and the system as a whole, for completeness. Table 9 presents completeness checks that can be made by visually scanning PROCESS CHAIN reports.

Example

Figure 56 presents a PROCESS CHAIN for the EVENT "salaried-emp-processing-unit."
| PROCESS | The absence of any EVENT as a result of INCEPTION or TERMINATION of the PROCESS should be rationalized. |
| EVENT | The absence of any PROCESS being TRIGGERED by an EVENT should be rationalized. |
| System Description | All chains should terminate in one of three ways: |
| | 1) In a loop back into the chain |
| | 2) By a PROCESS designating the last activity in the procedure represented by the chain |
| | 3) By an EVENT designating termination of a procedure represented by a chain |

Given a particular PROCESS or EVENT, the report allows a trace to be made through the system of actions taken. Checks can be made that based on a particular starting point all EVENT-PROCESS chains evolving from the starting point terminate correctly.

Table 9
Completeness Checks that may be made by Visual Analysis of the PROCESS CHAIN Report
<table>
<thead>
<tr>
<th>Process Chain</th>
</tr>
</thead>
<tbody>
<tr>
<td>+----EVENT----+ +----PROCESS----+ +----EVENT----+</td>
</tr>
<tr>
<td>ISALARIED- I  ISALARIED- I  ISALARIED- I</td>
</tr>
<tr>
<td>DATA- I       IPAYCHECK- I     IVALIDITY- I</td>
</tr>
<tr>
<td>ICOLLECTION I  IVALIDATION I   ICHECK I</td>
</tr>
<tr>
<td>+ON INCEPTN-+ +TRIGGERED-+ +ON INCEPTN-+</td>
</tr>
<tr>
<td>+----EVENT----+ +----PROCESS----+ +----EVENT----+</td>
</tr>
<tr>
<td>ISALARIED- IX- I  ISALARIED- I</td>
</tr>
<tr>
<td>LIP-PROCESS- I  LEMPLOYEE- I</td>
</tr>
<tr>
<td>TING-INIT I  IPROCESSING I</td>
</tr>
<tr>
<td>+TRIGGERED-+</td>
</tr>
<tr>
<td>+----EVENT----+ +----PROCESS----+ +----EVENT----+</td>
</tr>
<tr>
<td>IHOURLY-EMP-I  IHOURLY- I  IING-INIT I</td>
</tr>
<tr>
<td>IPROCESSING-I  IEMPLOYEE- I</td>
</tr>
<tr>
<td>INIT I  IPROCESSING I  IING-INIT</td>
</tr>
<tr>
<td>+TRIGGERED-+</td>
</tr>
<tr>
<td>+----EVENT----+ +----PROCESS----+ +----EVENT----+</td>
</tr>
<tr>
<td>IHOURLY- I  IDATA- I  IPROCESSING I</td>
</tr>
<tr>
<td>ICURRENT- I  ICURRENT- I  IPROCESSING I</td>
</tr>
<tr>
<td>+ON INCEPTN-+ +TRIGGERED-+ +ON INCEPTN-+</td>
</tr>
</tbody>
</table>

Figure 56  (Continued)
Figure 56 (Continued)
264
**Figure 56 (Continued)**

265
PROCESS INPUT/OUTPUT

Purpose

To present in an easy to examine outline form, the basic functions of one or more PROCESSES in the Language description and how these PROCESSES interact with information.

Information Presented

The report presents, for the PROCESS names given as input, four types of information which may be printed or suppressed by the specification of appropriate parameters for the command generating the report.

The DESCRIPTION parameter permits the printing of the DESCRIPTION comment entry for each PROCESS if available. The PROCEDURE parameter permits the printing of the PROCEDURE comment entry for each PROCESS if available.

The INPUT parameter permits the printing of the names of all SETS, INPUTS, ENTITIES, GROUPS and/or ELEMENTS that are RECEIVED and/or USED by each PROCESS. The OUTPUT parameter permits the printing of the names of all SETS, OUTPUTS, ENTITIES, GROUPS and/or ELEMENTS that GENERATED, UPDATED and/or DERIVED by each PROCESS.

Format

An entry in the report is printed for each PROCESS name given as input. Each name is identified by a number, 1*, 2*, etc. designating its position in the input stream. The following format is used to print out information about each process:

#* process name

[DESCRIPTION comment entry]

[PROCEDURE comment entry]

*** INPUTS ***

[All INPUTS RECEIVED by the PROCESS]

[All SETS, INPUTS, ENTITIES, GROUPS and ELEMENTS USED by the PROCESS]

*** OUTPUTS ***

[All OUTPUTS GENERATED by the PROCESS]

[All SETS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS DERIVED by the PROCESS]

[All SETS ENTITIES, GROUPS and ELEMENTS UPDATED by the PROCESS]
All the names listed under the INPUTS and OUTPUTS headings are numbered sequentially.

If a DESCRIPTION or PROCEDURE comment entry is not available for a particular name, that part of the format is not included in the report. If no names are listed under the INPUTS heading, the message:

NO INPUTS FOR THIS PROCESS

will be printed. If no names are listed under the OUTPUTS heading, the message:

NO OUTPUTS FOR THIS PROCESS

will be printed.

Options and Alternatives

Any part of the information presented for each PROCESS name can be included in or omitted from the report depending on the parameters used when generating it: The parameters and their effect on the report are given below.

1. DESCRIPTION - specifies that the DESCRIPTION comment entry for each name be included in the report.
   NODESCRIPTION - specifies that the comment entry is not printed.

2. PROCEDURE - specifies that the PROCEDURE comment entry for each name be included in the report.
   NOPROCEDURE - specifies that the comment entry is not printed.

3. INPUT - specifies that names USED or RECEIVED by the PROCESS are presented in the report.
   NOINPUT - specifies that those names are not printed.

4. OUTPUT - specifies that names DERIVED, UPDATED or GENERATED by the PROCESS are presented in the report.
   NOOUTPUT - specifies that these names are not printed.

Each entry of the report is started at the beginning of a new page when the NEW-PAGE parameter is specified. When NONEW-PAGE is in effect, the entries are printed one after another in the report.

An INDEX for the report is produced when the INDEX parameter is specified.

The report may be generated for a single name (via the name parameter) or for a collection of names each specified by the user or retrieved via NAME-GET.
Analysis

Each name given as input to the software producing the report is searched for in the database. If it is not found the message:

\text{URA076: MAINPRIO: NAME NOT IN DATA BASE -}

is printed. If it is found a check is made that it is a PROCESS name. If it is not, the message:

\text{URA088: MAINPRIO: NAME NOT A PROCESS NAME -}

is printed. If the name is a PROCESS name then the information available for it, as requested by the parameters, is presented on the report.

Usage

This report is beneficial in presenting a general description of the functions of target system (as described by the PROCESS) for purposes of communication between analysts and users.

It may also be used by analysts to check that the DESCRIPTION and PROCEDURE defined for each PROCESS is in agreement with the information which is input to or output from the PROCESS.

Examples

Figure 57 presents a PROCESS INPUT/OUTPUT report for a single name "payroll-processing."

Figure 58 presents the report generated for the SUBPARTS of "payroll-processing." This was done by the following commands:

\text{NAME-GEN SUBPARTS-OF=payroll-processing LEVELS=1}
\text{PROCESS-INPUT-OUTPUT PROCEDURE}
process input/output

1. This input/output description procedure nonex-page noindex print nopunch

2. processing

3. process represents the highest level process

4. in the input system, it accepts and processes

5. all inputs and produces all outputs.

*** INPUTS ***

- weather-information RECEIVED
- biology-information USED
- system-master-information USED

*** OUTPUTS ***

- system-outputs GENERATED
- system-outputs DERIVED
- system-master-information UPDATED
This process performs those actions needed to interpret the cards to produce a pay statement for each hourly employee.

---

**INPUTS**

- Time card
- Hourly-employee-record

**OUTPUTS**

- Pay-statement
- Gross-listing
- Hourly-employee-report
- Pay-statement

---

Figure 58

270
PROCESS INPUT/OUTPUT

- Employee-processing

This process stores information about new employees and prints out a corresponding report.

1. New employee record
2. Count of number of employees in appropriate category
3. Update relationship between employee record and count
4. Update all appropriate fields in employee record.

*** INPUTS ***

- Daily-employment-form  RECEIVED
- Initial-employment-form  RECEIVED
- K- withholding-certificate  RECEIVED
- Letter-employment-form  USED
- Initial-employment-form  USED
- K- withholding-certificate  USED

*** OUTPUTS ***

- New-employee-report  GENERATED
- New-employee-report  DERIVED

- Employee-processing

This process produces the pay statement for salaried employees once a month.

1. Calculate gross pay.
2. Subtract TSS from gross pay.
3. Subtract tax from gross pay to obtain net pay.
4. Update the previous employee record accordingly.

Figure 58 (Continued)
The department record accordingly.

Rate bonus if

The index is assumed to be 40

*** INPUTS ***

4.1 intrud-employee-record 

4.1.1 intrud-employee-record 

*** OUTPUTS ***

4.1.2 intrud-employee-record

4.1.3 intrud-employee-report

4.1.4 intrud-employee-report

4.1.5 intrud-employee-report

4.1.6 intrud-employee-report

4.1.7 intrud-employee-report

*** INPUTS ***

No inputs for this process

*** OUTPUTS ***

No outputs for this process

**-ing-processing

Process deletes data, for those employees who
are no longer on the payroll, from the files. It also
creates a list of all employees no longer on the
payroll.

Figure 58 (Continued)

272
**PROCESS INPUT/OUTPUT**

1. main type of employee by employment status item.
2. then, retrieve the contents of the appropriate employee record and print in report format.
3. fill asc of employees field in appropriate record.
4. to employee record.

*** INPUTS ***

employment-termination-form RECEIVED
employment-termination-form USED

*** OUTPUTS ***

terminated-employee-report GENERATED
terminated-employee-report DERIVED
PUNCHED COMMENT ENTRIES

Purpose

To present selected comment entries given for one or more names in a particular data base.

Information Presented

The report presents, for those names given as input, any comment entries which are specified as parameters and are available for the names. The types of comment entries available for each name are dependent on the name type of the name the report is being generated for. The table below shows the types of comment entries that may be presented and the types of names they may be presented for.

<table>
<thead>
<tr>
<th>Comment Entry Type</th>
<th>Name Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>All name types</td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>PROCESS</td>
</tr>
<tr>
<td>VOLATILITY</td>
<td>ENTITY</td>
</tr>
<tr>
<td>VOLATILITY-MEMBER</td>
<td>SET</td>
</tr>
<tr>
<td>VOLATILITY-SET</td>
<td>SET</td>
</tr>
<tr>
<td>DERIVATION</td>
<td>RELATION and SET</td>
</tr>
<tr>
<td>TRUE-WHILE</td>
<td>CONDITION</td>
</tr>
<tr>
<td>FALSE-WHILE</td>
<td>CONDITION</td>
</tr>
</tbody>
</table>

Format

An entry in the report is printed for each comment entry presented for each name given as input. Each entry is numbered 1*, 2*, etc. The format of each entry is:

```plaintext
/*
 name
 comment-entry-statement;
 [comment entry text];
```

Each line of the comment entry text is numbered within a given report entry.

Options and Alternatives

An entry in the report is produced for each comment entry (as specified by the parameters for the command generating the report) available for each name given as input. The following parameters designate the comment entries to be presented:

<table>
<thead>
<tr>
<th>DERIVATION</th>
<th>DESCRIPTION</th>
<th>FALSE-WHILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCEDURE</td>
<td>TRUE-WHILE</td>
<td>VOLATILITY</td>
</tr>
<tr>
<td>VOLATILITY-MEMBER</td>
<td>VOLATILITY-SET</td>
<td></td>
</tr>
</tbody>
</table>

Any of these parameters prefixed with "NO" specifies that the corresponding comment entries are not to be presented in the report.
Analysis

Each name given as input to the software generating the report is searched for in the data base. If it is not found the message:

    URA135: MAINPCOM: NAME NOT FOUND IN D.B. -

is printed. If the name is found then those comment entries (as specified by the parameters) which are available for it are printed on the report.

Usage

The report may be used by analysts to check the availability of specific types of comment entries for a class of names. For example, to check that all SETS have VOLATILITY-MEMBER, VOLATILITY-SET, and DERIVATION comment entries the commands:

    NAME-GEN SET
    PUNCH-COMMENT-ENTRY VOLATILITY-MEMBER VOLATILITY-SET DERIVATION

can be given.

Examples

Figure 59 presents the report for the name "salaried-employee-processing."
This process produces the pay statement for salaried employees once a month.

1. salary defines gross pay.
2. compute taxes from gross pay.
3. subtract taxes from gross pay to obtain net pay.
4. update salaried employee record accordingly.
5. update department record accordingly.
6. generate paycheck

Note: hours worked is assumed to be 40;
STRUCTURE:

Purpose

To present the implied hierarchy of PROCESSES, or INPUTS, or OUTPUTS or INTERFACES defined in the Analyzer data base, from the use of the SUBPARTS statements relating them.

Information Presented

The report presents all SUBPARTS structures for a given class of names (INTERFACES, INPUTS, OUTPUTS, or PROCESSES) as specified by the parameters when generating the report.

The structures start with all names which are not PART of any higher structure. These names are designated level 1 names. The SUBPARTS of level 1 names are presented as level 2 names. The SUBPARTS of the level 2 names are then presented and so on.

Format

The report presents the structures under three headings: COUNT, LEVEL, and NAME. NAME presents the name of the object in the structure, LEVEL presents the level number associated to the name corresponding to its position in the structure and COUNT presents the position (line) in the report where the name is printed out. Each level is indented (as specified by the INDENT parameter) to further accent the idea of structure.

A summary section for the report provides a count (under the COUNT heading) of the number of names presented at a given level (as designated by the LEVEL heading).

Options and Alternatives

The INPUT, OUTPUT, PROCESS and INTERFACE parameters for the command specify which type of names the report will be produced for. One and only one of these parameters may be specified for the command producing the report.

The number of spaces which each level of the structure is indented may be assigned by the INDENT parameter. If no value is given INDENT defaults to 3, but may take on any value from 1 to 10.

An INDEX for the report is produced when the INDEX parameter is used.

Analysis

A check is made that at least one name (of the name type designated by the parameters) exists in the data base which is not PART of a larger structure. If no such name exists the message:

URA 288: STATPS: No name at level one
is printed. For each name found, its SUBPARTS structure is traced and presented in the report.

If a loop is encountered in the structure the message:

URA 285: ERRPS: THE FOLLOWING NAMES ARE INVOLVED IN LOOPS -

is printed with the names involved.

Should the structure consist of more than 50 levels (the limit that the software has been designed to handle) the message:

URA 284: MAINSTR: TOO MANY LEVELS - CONTINUING -

is printed.

Usage

The report is an aid to analysts in maintaining consistency in structures defined for the target system description. Especially where a top-down approach is being used, the analyst is concerned that names have been inserted into the proper level of a structure.

Examples

Figure 60 presents a STRUCTURE report for INPUT names. Figure 61, 62 and 63 present STRUCTURE reports for OUTPUT, INTERFACE and PROCESS names, respectively.
INPUT STRUCTURE

employee-information
  time-card
  hourly-employment-form
  salaried-employment-form
  tax-withholding-certificate
  employment-termination-form
### Output Structure

<table>
<thead>
<tr>
<th>LEVEL</th>
<th>COUNT</th>
<th>LEVEL</th>
<th>COUNT</th>
<th>LEVEL</th>
<th>COUNT</th>
<th>LEVEL</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

- system-outputs
- pay-statement
- error-listing
- hourly-employee-report
- salaried-employee-report
- hired-employee-report
- terminated-employee-report

---

Figure 61

280
INTERFACE STRUCTURE

1. EMPLOYEES
   1.1. PAYROLL EMPLOYEES
   1.2. PAYROLL DEPARTMENT

Figure 62
281
PROCESS STRUCTURE

1. Payroll-processing
   2. New-employee-processing
   3. Salaried-record-creation
   4. Hourly-record-creation
   5. Hire-report-entry-generation
   6. Department-file-addition
   7. Terminating-emp-processing
   8. Salaried-record-deletion
   9. Hourly-record-deletion
   10. Te. a-report-entry-generation
   11. Department-file-removal
   12. Hourly-employee-processing
   13. Hourly-paycheck-validation
   14. Time-card-validation
   15. Hourly-emp-update
   16. Hour-update
   17. H-report-entry-generation
   18. Hourly-paycheck-production
   19. H-gross-pay-computation
   20. Total-hours-computation
   21. Salaried-employee-processing
   22. Salaried-paycheck-validation
   23. Salaried-emp-update
   24. S-report-entry-generation
   25. Salaried-paycheck-production
   26. S-gross-pay-computation
   27. Shared-routines
   28. Pay-computation-validation
   29. Tax-computation
   30. F.E.T.-pay-computation
   31. Total-deductions-computation
   32. F.E.T.-pay-update
   33. Local-deductions-update
   34. State-deductions-update

Figure 63
**Process Structure**

<table>
<thead>
<tr>
<th>Level Count</th>
<th>Level Count</th>
<th>Level Count</th>
<th>Level Count</th>
<th>Level Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 63** (Continued)
INDEX

Purpose

To provide a reference into a particular URA report to locate all occurrences of the use of a particular user defined name in the report.

Information Presented

The report presents all the user defined names used in a given URA report, which must be one of the:

- CONTENTS REPORT
- DICTIONARY REPORT
- EXTENDED PICTURE
- FORMATTED PROBLEM STATEMENT
- PICTURE
- PROCESS CHAIN
- PROCESS INPUT/OUTPUT or STRUCTURE,

the page numbers in the report where each name occurs, and the number of times the name appears within the pages it occurs (if more than once).

Format

The report consists of an entry for each name used in one of the above reports. The entry consists of:

- the name presented in the report
- the page numbers (separated by commas) that the name occurs on in the report
- the number of occurrences (enclosed in brackets) of the name on a particular page.

Each entry is numbered and the entries are arranged in alphabetical order by name.

Options and Alternatives

There are no options for this report.

Analysis

The input for the software producing the INDEX is obtained from one of the report-producing modules which allow the INDEX parameter. If no input is available, because the report presents no information, the message:

URA-007: REPORT: NO NAMES IN INDEX

is printed. If input is available (in the form of names and line numbers) the names are sorted and presented as the index report.
Usage

The INDEX is intended as a reference into a report for purpose of locating all occurrences of the use of a particular name in the report. The INDEX is usually desirable whenever a report over a few pages in size is to be generated.

Examples

Figure 64 presents a PROCESS INPUT/OUTPUT REPORT with an INDEX.
INPUT DESCRIPTION PROCEDURE MONY-PAGE INDEX PRINT NOPUNCH

1. Employ-processing

   This process performs those actions needed to interpret
   time-cards to produce a pay-statement for each hourly
   employee.

   1. Compute gross pay from time card.
   2. Subtract tax from gross pay.
   3. Subtract tax from gross pay to obtain net pay.
   4. Create hourly employee record accordingly.
   5. Enter department record accordingly.
   6. Prepare paycheck.

   Note: If status code specifies that the employee did not work
   this week, no processing will be done for this employee.

*** INPUTS ***

<table>
<thead>
<tr>
<th>Item</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Card</td>
<td>Received</td>
</tr>
<tr>
<td>Hr Card</td>
<td>Used</td>
</tr>
<tr>
<td>Hourly Employee</td>
<td>Used</td>
</tr>
</tbody>
</table>

*** OUTPUTS ***

<table>
<thead>
<tr>
<th>Item</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pay Stmt</td>
<td>Generated</td>
</tr>
<tr>
<td>Cross-listing</td>
<td>Generated</td>
</tr>
<tr>
<td>Hourly Employee Report</td>
<td>Generated</td>
</tr>
<tr>
<td>Cross-listing</td>
<td>Derived</td>
</tr>
<tr>
<td>Net Pay</td>
<td>Derived</td>
</tr>
<tr>
<td>Editorial Report</td>
<td>Derived</td>
</tr>
</tbody>
</table>

Figure 64
The process stores information about new employees and prints out a corresponding report.

1. new employee record
2. count of number of employees in appropriate staff
3. relationship between employee record and staff
4. update appropriate fields in employee record.

*** INPUTS ***

- employment-form
- employee-employment-form
- mailing-certificate
- employment-form
- employee-employment-form
- mailing-certificate

*** OUTPUTS ***

- employee-report
- employee-report

Employee-processing

This process produces the pay statement for salaried employees once a month.

1. pay declines gross pay.
2. cut taxes from gross pay.
3. correct taxes from gross pay to obtain net pay.
4. update related employee record accordingly.

Figure 64 (continued)
6. Enter department record accordingly.
7. Enter paycheq
   > hours worked is assumed to be 40

*** INPUTS ***

salaries-employee-record

*** OUTPUTS ***

pay-statement
error-listing
salaries-employee-report
error-listing
salaries-employee-report

** Figure 64 (continued) **

288
1. call type of employee by employment status item
2. add retrieve the contents of the appropriate employee record and print in report form
3. store number of employee still in appropriate record
4. store employee record.

*** INPUTS ***

employment-termination-form RECEIVED
employee-termination-form USED

*** OUTPUTS ***

unintended-employee-report GENERATED
unintended-employee-report DERIVED

Figure 64 (continued)
<table>
<thead>
<tr>
<th>Category</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 (2)</td>
<td></td>
</tr>
<tr>
<td>3 (2), 5 (2)</td>
<td></td>
</tr>
<tr>
<td>4 (2)</td>
<td></td>
</tr>
<tr>
<td>4 (2)</td>
<td></td>
</tr>
<tr>
<td>3 (2), 5 (2)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4 (2)</td>
<td></td>
</tr>
<tr>
<td>6 (2)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Figure 64 (continued)

290
PART IV

USER REQUIREMENTS ANALYZER

COMMAND DESCRIPTIONS
A Note on Version A2.1 of URA

There are several important changes that have been made in URA which makes Version A2.1 different than Version A2.0.

First, several internal modifications (which are transparent to the user) have been made to make the URA more efficient. Second a few errors in Version A2.0 have been corrected so that the URA commands perform properly. Third, the availability of a select number of URA commands and parameters have been removed. Last, two new commands are available in this version, and some commands have additional parameters not available in Version A2.0.

The following is a list of all modifications that fall into the latter two classes of changes:

1) The DATA-BASE-STATISTICS command has been removed from the URA command language, but is still available as a stand-alone program.

2) The PUNCH/NOPUNCH and EMPTY/NOEMPTY parameters are no longer operative for the PROCESS-INPUT-OUTPUT command.

3) The following parameters are now available for the DATA-PROCESS command: DPMAT/NODPMAT, DPANL/NODPNL, PMAT/NOMAT, and PANL/NOPANL.

4) The PROCESS-CHAIN and EXTENDED-PICTURE commands have been added.
The URA Command Language

The URA Command Language consists of three basic types of commands:

- **Report Commands**
- **Modifier Commands**
- **Control Commands**

Report Commands retrieve data from the URA data base and output it in some meaningful format. These reports do not change the contents of the data base whatsoever. Their purpose is solely that of displaying orderings and/or relationships within the current problem statement. The following Report Commands are available in this version of the Analyzer:

- CONSISTS-COMPARISON
- CONSISTS-MATRIX
- CONTENTS
- DATA-PROCESS
- DICTIONARY
- ENTITY-IDENTIFIER
- EXTENDED-PICTURE
- FORMATTED-PROBLEM-STATEMENT
- FREQUENCY
- KWIC
- NAME-GEN
- NAME-LIST
- PICTURE
- PRINT-ATTRIBUTE-VALUES
- PROCESS-INPUT-OUTPUT
- PROCESS-CHAIN
- PUNCH-COMMENT-ENTRY
- STRUCTURE
- SUMMARY

Modifier Commands are intended to modify the contents of the URA data base in the manner defined by the problem definer. These commands take legal URL statements or URL names as input. URA then generates error diagnostics as well as an output report to present the outcome of the data base alteration. The following Modifier Commands are available in this version of the Analyzer:

- CHANGE-TYPE
- DELETE
- DELETE-COMMENT-ENTRY
- DELETE-PSL
- INPUT-PSL
- RENAME
- REPLACE-COMMENT-ENTRY

Control Commands are the means to pass certain control information to the User Requirements Analyzer. The SET command, for example, allows the user to define which URA data base is to be accessed by the Report and Modifier Commands as well as setting various switches and assigning input and output files. Control Commands are installation dependent and therefore are given in the Appendix E.

293
Although any of the commands can be issued independently of each other, it is often advantageous to use some commands in sequence. This means that output of one command can be used as input by another. The most common instance of this is when NAME-GEN is used to select certain names (say all PROCESSES for example) which can then be used as input to a Report Command (possibly PICTURE, for a PICTURE REPORT for all PROCESS names.)

**URA Command Language/Installation Dependencies**

URA is a software system that is designed to be used interactively in a time sharing system environment. It contains its own data base management system but it is dependent on the time sharing operating system for the usual facilities of sign on, identification and security, file creation and editing, etc.

Differences in operating systems and operations at particular installations affect the way in which URA is executed at a particular installation. There are basically three aspects of URA that are installation "dependent":

1) Control commands
2) Method of initiating and executing URA
3) File names used by URA

These dependencies are presented in Appendix E for the particular installation.

URA can also be used in batch mode at most installations. The commands necessary to accomplish this are also installation dependent and are not covered in this paper.
ABBREVIATIONS

To enable the user to fit a lengthy command on the allotted line and eliminate some of the tedium of command specification, abbreviated forms for both commands and parameters may be used. Each abbreviation can be found in parentheses immediately following the word it represents. For example, the command:

```
CHANGE-TYPE NAME=GROSS-PAY TYPE=ELEMENT
```

can be written as

```
CT N=GROSS-PAY T=ELEMENT
```

BLANKS

A blank must appear between the command and any accompanying parameter, and between successive parameters. Several blanks are treated as a single blank and may be inserted whenever a single blank is necessary. For example,

```
CT N=GROSS-PAY T=ELEMENT
```

can be written as

```
CT  N=GROSS-PAY  T=ELEMENT
```

BRACES

In the following examples, when parameters or parameter values are enclosed in braces ({}), a choice among the two or more entries must be made. It is important to note that one and only one of the options must be chosen. For example, the braces used in describing the syntax of the CHANGE-TYPE command specifies that the command must either be of the form:

```
CT N=user-name  T=name-type
```

or

```
CT F=fdname [T=name-type]
```

BRACKETS

Whenever parameter notation in an example appears within brackets ([ ]), it indicates a feature the user may optionally use. For example, the TYPE parameter in the CHANGE-TYPE command is optional when the FILE parameter is also used. Therefore, the command may be of the form:

```
CT FILE=fdname TYPE=name-type
```

or

```
CT FILE=fdname
```
The syntax of the FILE parameter shows that the parameter may be given either as:

FILE=fdname

or just

FILE

No other variations are acceptable (except those already specified, i.e., abbreviations, etc.).

COMMAND LINE

Each command must appear on a separate line and totally on that line. A command cannot be split on succeeding lines. Only columns 1 through 80 of each line can be used.

GENERAL COMMAND SYNTAX

The command identifiers (name of the command) must precede any accompanying parameter or list of parameters.

COMMAND PARAMETERS

Parameters for a command separated by one or more blanks. Parameters may be given in any order, but are processed from left to right. If conflicting parameters are used, the right-most parameter, i.e., the last one given, is considered to be correct and is the one used in the processing of the command.

ELLIPSIS

The ellipsis (...) signifies that the command construct immediately preceding the ellipsis can be repeated as many times as desired by the user.

INTEGERS

The integers required for parameters must be positive integers. If a value range is given for a particular parameter description, that restriction must also be met.

NAMES

All user defined names (user-name) must meet the following restrictions to be a legal URL name.

A name can be formed only from the following characters:

A, B, C, ..., Z (letters)
0, 1, 2, ..., 9 (integers)
- (dash)
-- A name can be any combination of thirty characters or less where the first character is a letter.

-- Blanks cannot be used in the name.

-- A user defined name cannot be a URL RESERVED WORD. For a list of Reserved words see APPENDIX B of ISDOS Working Paper No. 68, USER REQUIREMENTS LANGUAGE, LANGUAGE REFERENCE MANUAL.

For example,

GROSS-PAY, EMPLOYEE-NUMBER and PAYROLL-PROCESSING are all legal names.

PROCESS, EMPLOYEE-# and 123-HILL-STREET are illegal names. "PROCESS" cannot be used as a user-name because it is a URL Reserved Word. "EMPLOYEE-#" uses a character (the "#" sign) which is not allowed and "123-HILL-STREET" is illegal because it starts with an integer rather than a letter.
Format of Command Descriptions

All the URA commands in this paper are described in the following format:

**Command:** COMMAND-NAME

**type:** command type

**Purpose:** This presents the function of the command in the URA system whether it generates a report, modifies the data base or gives control information to URA. (The "URA Users Manual" and "URA Reports" present detailed descriptions of the reports generated by each command.)

**Prototype:** This presents the legal syntax for the command. The Command Language Syntax Notation specifies what the special symbols (such as braces and brackets) represent in interpreting the syntax.

**Parameters:** For each parameter available for the command, this section provides a brief description explaining how the parameter changes the action of the command. (The "URA Users Manual" and "URA Reports" explain how to use these parameters effectively.)

There are basically five types of parameters:

- **Input data parameters** - these parameters specify the data to be used as input to the command. FILE and NAME are examples of Input data parameters.

- **Input control parameters** - these parameters specify how the input data is to be used, changed, etc., by the command. The TYPE parameter for the CHANGE-TYPE command and CONTAINED/CONSISTS parameter for the CONSISTS-MATRIX command are examples of this type.

- **Output data parameters** - these parameters specify if output is to be generated from the command and the form in which it is presented. The PUNCH and PRINT parameters are examples of this type of parameter.

- **Output option parameters** - these parameters specify options which may be included or omitted from the output. The LEVELS parameter for the CONTENTS command and the DESCRIPTION parameter for the DICTIONARY command are examples of this type.

- **Output format parameters** - these parameters specify alternate formats for presenting the information in the output from the command. The NEW-PAGE parameter and HMARG parameter for the FPS command are examples of this.
The parameters for each command will be presented in the above order according to type.

Defaults: These present which parameters will be used for the command, or what value a parameter will have, if the parameter, or value, is not explicitly defined. For example, if

```
CONTENTS
```

is specified, the defaults for this command are such that this has the same effect as specifying:

```
CONTENTS FILE NOINDEX LEVELS=ALL NONCFLAG
```

If a "no default" is given, this means that if not explicitly defined, the corresponding parameter will not be used for the command.

Messages: These are the possible error messages that may occur if the parameters for the command are not specified correctly. The "URA Users Manual" explains what to do should these messages be encountered.

Examples: Actual example of the command syntax are presented. (The results from these examples are presented in the "URA Users Manual" and "URA Reports."
Command: CHANGE-TYPE

Type: modifier command

Purpose: To change the name type of a user name defined in the user's data base. A record of this change is generated in the form of the CHANGE-TYPE REPORT.

Prototype: CHANGE-TYPE(CT)  
\{NAME(N)=user-name TYPE(T)=name-type \}  
\{FILE(F)[=fdname] [TYPE(T)=name-type]\}

Parameters:

Input-data

FILE(F)[=fdname]  
Default: no default  
When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. The file format for each line of the input file must be of the form:

user-name [name-type]

Free format is allowed so the user-name does not have to start in the first position in the line. The two names must be separated by one or more blanks. The name-type is optional. If a name-type is not specified for each user-name, the name type for each of these names will be changed to the type specified in the TYPE parameter. One of these alternative methods of assigning a type must be used, but not both. If both are used, all the names in the file will be assigned the name type specified by the TYPE parameter.

NAME(N)=user-name  
Default: no default

The given user-name is the name for which the change is to be made. When the NAME parameter is used, the TYPE parameter must be used in conjunction with it.

Input-Control

TYPE(T)=name-type  
Default: no default  
This parameter specifies the new name type to be used in the change. See Appendix E of The User Requirements Language, Language Reference Manual** for a list of all possible name types.

Messages: If neither FILE nor NAME parameter are given, the error message:

NO NAME GIVEN

* The name of the default file is installation dependent and consequently is given in Appendix E.

** ISDOS Working Paper No. 68
will be generated by URA. If one of these two parameters are given, but no TYPE is specified, the error message:

**NO TYPE GIVEN WITH "NAME=" OR "FILE" PARAMETER.**

will be given.

**Examples:**

CHANGE-TYPE NAME=GROSS-PAY TYPE=ELEMENT

CT  F  T=ELEMENT

CT  FILE  T=GROUP
Command: CONSISTS-COMPARISON  Type: report command

Purpose: To produce the CONSISTS-COMPARISON REPORT.

Prototype: CONSISTS-COMPARISON(CNC) [parameter]...

Parameters:

- **Input-Data**: FILE(F)[=fdname]  Default: FILE

  When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command and the report is produced using all the names in the file. In any case, the names in the input file must be SET, INPUT, OUTPUT, ENTITY and/or GROUP names. The format of the input file must be one name per line.

Examples: CNC

CNC FILE

---

* The name of the default file is installation dependent and consequently is given in Appendix E
Command: CONISTS-MATRIX

Type: report command

Purpose: To produce the CONISTS MATRIX REPORT.

Prototype: CONISTS-MATRIX(CM) \{CONTAINE(CNTD)\} [parameter]...

Parameters:

Input-File
Data
When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. When a name is specified via the NAME parameter, the report is produced only for that name. The format of the input file must be one name per line.

Input-
Control
CONTAINED(CNTD), CONISTS(CSTS) Default: no default

Since no default exists, one of the above must be specified. If CONTAINED is given, the names used as input must be ELEMENT, GROUP, ENTITY, INPUT and/or OUTPUT names. If the CONSISTS parameter is given the names used as input must be SET, ENTITY, INPUT, OUTPUT and/or GROUP names.

Messages: If neither CONTAINED nor CONSISTS is specified, the message:

MUST GIVE EITHER CONSISTS OR CONTAINED PARAMETER

will be printed.

Examples:

CM N=EMPLOYEE-NUMBER CNTD
CM FILE CSTS
CM CSTS

* The name of the default file is installation dependent and consequently is given in Appendix E
Command: CONTENTS

Type: report command

Purpose: To produce the CONTENTS REPORT.

Prototype: CONTENTS(CONT) [parameter]...

Parameters:

Input-Data
FILE(F)[=fdname], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command and the report is produced for all the names in the file. When a name is specified by the NAME parameter, the report is produced for that name alone. In any case, the names used as input to the command must be SET, INPUT, OUTPUT, ENTITY and/or GROUP names. The format of the input file must be one name per line.

Output-Data
INDEX, NOINDEX Default: NOINDEX

The INDEX parameter specifies the production of an index for the report consisting of an alphabetical listing of all names used in the report and the pages on which they occur.

Output-Option
LEVELS= \{integer\} Default: LEVELS=ALL

The LEVELS parameter specifies the lowest level of subordinate names to be outputted. The ALL value indicates that all subordinate names should be outputted. LEVELS can take on any integer value from 1 to 50.

NCFLAG, NONCFLAG Default: NONCFLAG

The NCFLAG parameter flags all GROUPS in the output reports that do not CONSIST of anything else, and those undefined names which are contained in a GROUP, INPUT, OUTPUT, ENTITY or SET.

Examples: CONTENTS N=VARYING-EMPLOYEE-DATA

CONT F

* The name of the default file is installation dependent and consequently is given in Appendix E.
Command: DATA-PROCESS  
Type: report command

Purpose: To produce the DATA PROCESS REPORT.

Prototype: DATA-PROCESS(dp) \{\text{DATA(D)}\} \{\text{PROCESS(P)}\} [\text{parameter}]...

Parameters:

- **Input-Data**
  - FILE(F)[=fdname], NAME(N)=user-name  
  Default: FILE
  
  When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. The format of the input file must be one name per line.

  When a name is given via the NAME parameter, the report is produced only for that name.

- **Input-Control**
  - DATA(D), PROCESS(P)  
  Default: no default
  
  Since no default exists, one of the above must be specified. If DATA is specified, the names used as input to the command must be SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names. If PROCESS is specified, the names used as input to the command must be PROCESS names.

- **Output-Options**
  - DPMAT, NODPMAT  
  Default: DPMAT

  With the DPMAT parameter in effect, the DATA PROCESS INTERACTION MATRIX is presented as part of the report. When NODPMAT is specified, this matrix is not printed.

  DPNAL, NODPNAL  
  Default: DPNAL

  With the DPNAL parameter in effect, analysis is done on the DATA PROCESS INTERACTION MATRIX (whether printed or not) and presented as the DATA PROCESS INTERACTION ANALYSIS. When NODPNAL is specified, this analysis is not done.

  PMAT, NOPMAT  
  Default: PMAT

  With the PMAT parameter in effect, the PROCESS INTERACTION MATRIX is presented as part of the report. When NOPMAT is specified, this matrix is not printed.

  PANL, NOPANL  
  Default: PANL

  With the PANL parameter in effect, analysis is done on the PROCESS INTERACTION MATRIX (whether printed or not) and presented as the PROCESS INTERACTION MATRIX ANALYSIS. When NOPANL is specified, this analysis is not done.

* The name of the default file is installation dependent and consequently is given in Appendix E
Messages: If neither DATA nor PROCESS is specified, an error message:

MUST GIVE EITHER "DATA" OR "PROCESS" PARAMETER
will be printed.

Examples: DP N=PAYROLL-PROCESSING PROCESS
           DP DATA
Command: DELETE

Type: modifier command

Purpose: To delete a name or list of names from the data base. When a name is deleted all of its connections to other names in the data base are also deleted. A permanent record of the change is also generated in the form of the DELETION REPORT.

Prototype: DELETE(DEL) \{FILE(F)\[=fdname\] \}
\{NAME(N)=user-name\}

Parameters:

Input Data
FILE(F)\[=fdname\], NAME(N)=user-name Default: no default

When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command and all names in the file are deleted from the data base. The format of the input must be one name per line. When a name is specified by the NAME parameter, that name is deleted from the data base.

Messages: If neither the FILE nor NAME parameter is specified, the message:

NO NAME OR FILE WAS SPECIFIED

will be given.

Examples: DELETE N=FIELD-CHECK-NEW

DEL FILE

* The name of the default file is installation dependent and consequently is given in Appendix E
Command: DELETE-COMMENT-ENTRY  Type: modifier command

Purpose: To delete from the data base, for the given name or list of names, those comment entries associated with each comment entry statement specified in the list of parameters. A permanent record of the change is generated in the form of the DELETED COMMENT ENTRIES report.

Prototype: DELETE-COMMENT-ENTRY(DCOM) \{FILE(F)[=fdname] \} \{NAME(N)=user-name \} [parameter]...

Parameters:

Input-FILE(F)[=fdname], NAME(N)=user-name  Default: no default

Data

When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. When a name is given via the NAME parameter, only the specified comment entries for that name are deleted. Either FILE or NAME must be given but not both. The format of the input file must be one name per line.

Input-When given as parameters, the comment entries for the following comment entry statements are deleted.

Control

DERIVATION(DER), NODERIVATION(NDER)  Default: NODERIVATION
DESCRIPTION(DESC), NODESCRIPTION(NDESC)  Default: NODESCRIPTION
FALSE-WHILE(FW), NOFALSE-WHILE(NFW)  Default: NOFALSE-WHILE
PROCEDURE(PRCD), NOPROCEDURE(NPRCD)  Default: NOPROCEDURE
TRUE-WHILE(TW), NOTRUE-WHILE(NTW)  Default: NOTRUE-WHILE
VOLATILITY(VOL), NOVOLATILITY(NVOL)  Default: NOVOLATILITY
VOLATILITY-MEMBER(VOLM), NOVOLATILITY-MEMBER(NVOLM)  Default: NOVOLATILITY-MEMBER
VOLATILITY-SET(VOLS), NOVOLATILITY-SET(NVOLS)  Default: NOVOLATILITY-SET

Output-PRINT, NOPRINT(NP)  Default: PRINT

Data

The PRINT parameter initiates the production of a printed DELETED COMMENT ENTRIES report. NOPRINT suppresses the printing.

Messages: If neither the FILE nor NAME parameter are given, the message:

NO NAME OR FILE SPECIFIED

is printed.

* The name of the default file is installation dependent and consequently is given in Appendix E
Examples: DELETE-COMMENT-ENTRY N=NEW-INFO-VALIDATION PRCD DESC
DCOM FILE DESC
Command: DELETE-PSL

Type: modifier command

Purpose: To delete specific URL statements in the problem definer's data base. Those statements used as input to the command are deleted. A permanent record for the change is generated in the form of the DELETED URL report.

Prototype: DELETE-PSL(DPSL) [parameter]...

Parameters:

**Input-Data**

- **INPUT(I)[=fname]**
  - Default: INPUT=terminal
  - When INPUT is used and a fname is specified, the contents of the designated fname are used as input to the command. This input must be in the same format allowable by the INPUT-PSL command (i.e., legal URL statements). The only exception is that no comment entry statements are allowed in the input (DESCRIPTION, for example). The EOF statement terminates input. If no fname is specified, its value defaults to the terminal so that the URL statements can be entered interactively.

**Output-Data**

- **SOURCE(S), NOSOURCE(NS)**
  - Default: SOURCE
  - When the SOURCE parameter is in effect, an AS-IS SOURCE LISTING (the DELETED URL output) of the deleted URL statements is produced. When the NOSOURCE parameter is given, no AS-IS SOURCE LISTING is produced.

- **XREF(X), NOXREF(NX)**
  - Default: NOXREF
  - The user may desire a cross reference for the AS-IS SOURCE LISTING. This consists of a list of all user-defined names from the input file and the line numbers on which they occur in the DELETED URL report. To accomplish this, the problem definer should specify XREF. When NOXREF is in effect, no cross reference is produced.

Example: DELETE-PSL NS

DPSL X
Command: DICTIONARY

Type: report command

Purpose: To produce the DICTIONARY REPORT for a name or list of
names in the user's data base.

Prototype: DICTIONARY(DICT) [parameter]...

Parameters:

Input-Data
FILE(F) [=fdname], NAME(N) = user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the
command. This file is the default PUNCH file for NAME-GEN.
If an fdname is indicated, that file is used as the input file for the production of the DICTIONARY REPORT. When a
name is specified via the NAME parameter, the report is generated for that name alone. The format of the input file must be one name per line.

Output-Data
INDEX, NOINDEX Default: NOINDEX

When given, the INDEX parameter specifies the production of an index to the report. This index consists of an alphabetical listing of all names used in the report and the page(s) on which they occur in the report.

Output-Option

The following four parameters specify the information to be included in the DICTIONARY. The "NO" prefix on a parameter specifies that such information not be included for the name(s).

DESCRIPTION(DESC), NODESCRIPTION(NDESC) Default: DESCRIPTION
KEYWORDS(KEY), NOKEYWORDS(NKEY) Default: KEYWORDS
RESPONSIBLE-PD(RPD), NORESPONSIBLE-PD(NRPD) Default: RESPONSIBLE-PD
SYNONYMS(SYN), NOSYNONYMS(NSYN) Default: SYNONYMS

Output-Format
NUM-SPACE(NS) = integer Default: NUM-SPACE=2

For ease in reading, the number of lines skipped between dictionary entries can be modified by varying this parameter. NUM-SPACE may take on any value between 0 and 10.

Examples: DICTIONARY  N=PAYROLL_PROCESSING
DICTIONARY  DICT  FILE

* The name of the default file is installation dependent and consequently is given in Appendix E
Command: ENTITY-IDENTIFIER

Type: report command

Purpose: To produce the IDENTIFIER INFORMATION REPORT.

Prototype: ENTITY-IDENTIFIER(EI) \{IDENTIFIER(I)\} \{ENTITY(E)\} [parameter]...

Parameters:

Input-Data

FILE(F)[=fdname], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as in the input file for the command. If a name is specified via the NAME parameter, the report is generated for that name alone. The format of the input file must be one name per line.

Input-Control

IDENTIFIER(I), ENTITY(E) Default: no default

Since no default is allowed, one of the above must be specified. If IDENTIFIER is specified, the names used as input to the command must be names used as IDENTIFIERS in the data base. If the ENTITY parameter is given, the names used as input must be defined ENTITY names.

Messages: If neither the IDENTIFIER nor ENTITY parameter is specified, the message:

MUST GIVE EITHER ENTITY OR IDENTIFIER PARAMETER

will be given.

Examples: EI N=EMPLOYEE-NUMBER I

EI FILE ENTITY

* The name of the default file is installation dependent and consequently is given in Appendix E
Command: EXTENDED-PICTURE  Type: report command

Purpose: To produce the EXTENDED PICTURE report.

Prototype: EXTENDED-PICTURE(EP) [parameter]...

Parameters:

Input-

Data FILE(F)[=fdname], NAME(N)=user-name Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is given, that file is used as the input file for the command. The format for the input file must be one name per line.

When a name is given via the NAME parameter, the report is produced only for that name. Regardless of whether FILE or NAME is specified, all names used as input to this command must be PROCESS, INTERFACE, INPUT, ELEMENT, GROUP, OUTPUT, ENTITY, or SET names.

Output-

Data INDEX, NOINDEX Default: NOINDEX

The INDEX parameter specifies the production of an index for the EXTENDED PICTURE report. This index consists of all user-defined names used in the report, in alphabetical order, along with the pages on which they appear in the report.

Output-

Option STRUCTURE(STR), DATA-FLOW(DF) Default: no default

When the STRUCTURE parameter is used, information is obtained for each input name from UTILIZES, SUBPARTS, CONSISTS, and SUBSETS statements, or their complementary statements. This information is then presented in graphical format. When the DATA-FLOW parameter is used, the graphical output shows information obtained for each input name from USED, USED TO DERIVE, USED TO UPDATE, RECEIVED, UPDATES, DERIVES, and GENERATES statements or their complementary statements. Since there is no default, either STRUCTURE or DATA-FLOW must be specified.

FORWARD(FWD), BACKWARD(BWD) Default: no default

DOWNWARD(DOWN), UPWARD(UP)

It is frequently convenient to think of FORWARD and BACKWARD as referring to DATA-FLOW and UPWARD and DOWNWARD as referring to STRUCTURE. However, FORWARD and DOWNWARD may be used interchangeably, as may

* The name of the default file is installation dependent and consequently is given in Appendix E
BACKWARD and UPWARD. The FORWARD (or DOWNWARD) parameter causes retrieval of information about each input name based on UTILIZES, SUBPARTS, CONSISTS and SUBSETS statements (for STRUCTURE) or USED, USED TO DERIVE, USED TO UPDATE, RECEIVED, UPDATES, DERIVES, and GENERATES statements (for DATA-FLOW). The BACKWARD (or UPWARD) parameter causes retrieval of information about each input name based on UTILIZED, PART, CONTAINED, and SUBSET statements (for STRUCTURE) or USES, USES TO DERIVE, USES TO UPDATE, RECEIVES, UPDATED, DERIVED, and GENERATED statements (for DATA-FLOW). Since there is no default, one of the above parameters must be specified.

**LINKS=integer**  
Default: **LINKS=1000**

The LINKS parameter specifies the maximum number of links (connections between names) to be followed in producing the report. LINKS can take on any integer value between 1 and 1000, inclusive.

**OUTPUT-Format**

**COLUMNS(COLS)=integer**  
Default: **COLUMNS=119**

The COLUMNS parameter specifies the number of columns to be used for output. The maximum value allowed is 119.

**ROWS=integer**  
Default: **ROWS=39**

The ROWS parameter specifies the number of rows to be used for output. The maximum value allowed is 39.

**HORIZONTAL-BOXES(HB)=integer**  
Default: (see text)

HORIZONTAL-BOXES specifies the maximum number of boxes containing names to be arranged across the page. The default value is the largest possible value for the given value of COLUMNS, and is computed as the greatest integer in \((COLUMNS-4)/17\).

**VERTICAL-BOXES(VB)=integer**  
Default: (see text)

VERTICAL-BOXES specifies the maximum number of boxes containing names to be arranged down the page. The default value is the largest possible value for the given value of ROWS, and is computed as the greatest integer in \((ROWS-2)/6\).

314
Messages: If the HORIZONTAL-BOXES value used will not fit in the number of COLUMNS specified, the message:

HORIZONTAL-BOXES TOO LARGE FOR NUMBER OF COLUMNS ON PAGE

will be given.

If the VERTICAL-BOXES value used will not fit in the number of ROWS specified, the message:

VERTICAL-BOXES TOO LARGE FOR NUMBER OF ROWS ON PAGE

will be given.

If neither the DATA-FLOW nor STRUCTURE parameter is specified, the message:

NEITHER DATA-FLOW NOR STRUCTURE WAS SPECIFIED

will be printed.

If none of FORWARD, BACKWARD, UPWARD and DOWNWARD is specified, one of the following messages will be printed:

NEITHER FORWARD NOR BACKWARD WAS SPECIFIED
NEITHER UPWARD NOR DOWNWARD WAS SPECIFIED.

Examples: EXTENDED-PICTURE STR DOWN N=PROCESS1
EP FILE DF BACKWARD LINKS=5
Command: FORMATTED-PROBLEM-STATEMENT Type: report command

Purpose: To produce the FORMATTED PROBLEM STATEMENT for a given name, or list of names and/or to produce this information in the form of PUNCH data.

Prototype: FORMATTED-PROBLEM-STATEMENT(FPS) [parameter]...

Parameters:

Input-Data

FILE(F)[=fdname], NAME(N)=user-name Default: FILE
When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. When a name is given via the NAME parameter, the report is produced only for the name specified. The format of the input file must be one name per line.

Output-Data

INDEX, NOINDEX Default: NOINDEX
The INDEX parameter specifies the production of an index for the FPS. This index consists of an alphabetical listing of all user defined names used in the FORMATTED PROBLEM STATEMENT and the page(s) on which they occur.

PRINT, NOPRINT(NP) Default: PRINT
The NOPRINT parameter specifies that no printed output report will be produced. The PRINT parameter specifies the production of the FORMATTED PROBLEM STATEMENT.

PUNCH(P)[=fdname], NOPUNCH Default: NOPUNCH
The PUNCH parameter specifies that PUNCH data should be generated and written into the designated PUNCH file. When the PUNCH parameter is used and no fdname is designated, the data is written into the default PUNCH file.* This file is the default PUNCH file for the command. If an fdname is indicated, that file is used as the PUNCH file. With the NOPUNCH parameter in effect, no action is taken to generate PUNCH data.

Output-Option

COMMENT(COM), NOCOMMENT(NCOM). Default: COMMENT
The COMMENT option, when in effect, specifies the inclusion of comments for undefined names. The NOCOMMENT option suppresses these comments.

* The name of the default file is installation dependent and consequently is given in Appendix E
With the DEFINE option in effect, DEFINE sections are included in the report. The NODEFINE option specifies that no DEFINE sections are included in the FORMATTED PROBLEM STATEMENT.

DES(DG), NODESG(NDG)  
Default: DESG

The DESG option, when in effect, indicates that DESIGNATE sections are provided for SYNONYM names in the FORMATTED PROBLEM STATEMENT. The NODESG option suppresses the production of such output.

EMPTY, NOEMPTY  
Default: (see text)

When EMPTY is in effect (the default when the PUNCH parameter is also used) the PUNCH file is emptied before PUNCH data is written into it. When NOEMPTY is in effect (the default when PUNCH is not used) no action is taken to empty the PUNCH file.

Output-format

AMARG(AM)=integer  
Default: AMARG=10

The AMARG parameter indicates the column at which the first name of a name pair is to be outputted. An example of a name pair can be found in the ATTRIBUTE statement where the syntax requires an ATTRIBUTE name and ATTRIBUTE-VALUE. AMARG must take on some value greater than SMARG and less than BMARG.

BMARG(BM)=integer  
Default: BMARG=25

The BMARG parameter indicates the column at which the second name of a pair is to be outputted. BMARG must take on some value greater than AMARG and less than RMARG-30.

CMARG(CM)=integer  
Default: CMARG=1

The CMARG parameter specifies the number of columns from SMARG the text (comment entry) for a comment entry statement begins. CMARG must take on some value greater than 0 and less than RMARG-72.*

HMARG(HM)=integer  
Default: HMARG=40

This parameter specifies the column where the user defined name in a section header statement are to be printed on the output. HMARG must take on some value greater than SMARG and less than RMARG-30.*

NEW-LINES(NL), NONEW-LINE(NNL)  
Default: NONEW-LINE

When the NEW-LINE parameter is given, the first name of a name list associated with a statement will appear on the line succeeding the statement identifier (name of the statement). The NONEW-LINE parameter initiates the list on the same line as the statement identifier.

* RMARG has the value 119 at most installations.
NEW-PAGE(NPG), NONEW-PAGE(NNPG)  Default: NONEW-PAGE

When given, the NEW-PAGE parameter specifies that each section of the FPS be printed on a separate page. NONEW-PAGE signifies that the sections will follow one another on a page within the page size restrictions. In any case, interrupted sections will be continued on succeeding pages.

NMARG(NM)=integer  Default: NMARG=20

The NMARG parameter indicates the column in which the name or first name of a name list for any statement will be outputted. NMARG must take on some value greater than SMARG and less than RNMARG-30.

ONE-PER-LINE(OPL), SEVERAL-PER-LINE(SPL)  Default: ONE-PER-LINE

The ONE-PER-LINE option indicates that the names in a name list for any statement will appear on succeeding lines. SEVERAL-PER-LINE option signifies that names in a name list will appear on the same line.

RNMARG(RM)=integer  Default: RNMARG=70

Specifies the right-hand margin for names in a name list when the SEVERAL-PER-LINE parameter is in effect. RNMARG must take on some value greater than MAX(BMARG, NMARG)+29 and less than RNMARG-30.*

SMARG(SM)=integer  Default: SMARG=5

The SMARG parameter indicates the column in which the statement identifier (name of the statement) will be started. SMARG must take on some value greater than 0 and less than MIN(AMARG, NMARG).

Examples:

FPS  N=FIELD-CHECK-NEW
FPS  FILE

* RMARG has the value 119 at most installations.
<table>
<thead>
<tr>
<th>Command</th>
<th>FREQUENCY</th>
<th>Type: report command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>To produce the FREQUENCY REPORT.</td>
<td></td>
</tr>
<tr>
<td>Prototype</td>
<td>FREQUENCY(FREQ)</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Examples</td>
<td>FREQ</td>
<td></td>
</tr>
</tbody>
</table>
Command: HELP

Purpose: To provide the on-line user with a list of possible commands for URA or information about the parameters for a particular URA command.

Prototype: HELP [parameter]...

Parameters: Command-name

Default: (see text)

If no command-name is given, a list of currently available URA commands is given. If a command-name is given, then the parameters for that command are presented. Abbreviations for the command-name are also acceptable.

SHORT, LONG

Default: SHORT

If SHORT is given, only the parameters for the given command are printed. If LONG is given, explanations of the various parameters are also printed.

Examples: HELP FPS LONG
HELP CONTENTS
Command: INPUT-PSL
Type: modifier command

Purpose: To add information to the URA data base to expand or modify the problem statement. A permanent record of the change can be generated in the form of the AS-IS SOURCE LISTING and URA CROSS REFERENCE.

Prototype: INPUT-PSL(IP) [parameter]...

Parameters:

Input-Data

INPUT(I)=fdname

Default: INPUT=terminal

When INPUT is specified, the contents of the designated fdname are used as input to the command. This input must be in the format of legal URL statements as specified by ISDOS Working Paper No. 68. The EOF statement terminates input. If the INPUT parameter is not specified, input is read from the terminal so that the URL statements can be entered interactively.

Input-Control

DBREF(D), NODBREF(ND)

Default: DBREF

The DBREF parameter allows the referencing of the data base by URA in its syntax and semantic analysis. When given, NODBREF allows the analyzer to only perform a syntax check of the input.

UPDATE(U), NOUPDATE(NU)

Default: NOUPDATE

With the UPDATE parameter given, the input will update the URA data base. NOUPDATE indicates that the data base is not to be changed.

Output-Data

SOURCE(S), NOSOURCE(NS)

Default: SOURCE

When the SOURCE parameter is in effect, AS-IS SOURCE LISTING of the input URL is produced. When the NOSOURCE parameter is given, the listing is not produced.

XREF(X), NOXREF(NX)

Default: NOXREF

XREF specifies that a URA CROSS REFERENCE is to be generated for the AS-IS SOURCE LISTING. This consists of a list of all user defined names from the input file and the line numbers on which they occur in the AS-IS SOURCE LISTING.

Examples: INPUT-PSL XREF UPDATE

IP  U
**Command:**  
KWIC  

**Default:**  
report command

**Purpose:**  
To produce a KWIC INDEX for a list of names.

**Prototype:**  
KWIC [parameter]...

**Parameters:**

<table>
<thead>
<tr>
<th>Input-Data</th>
<th>Default: FILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE(F)[=fdname]</td>
<td></td>
</tr>
<tr>
<td>When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. The format of the input file must be one name per line.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output-Format</th>
<th>Default: DIF=20</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIF=integer</td>
<td></td>
</tr>
<tr>
<td>DIF is the number of spaces allowed between the keyword and the rest of the name as it appears in the output. DIF must take on some value greater than 1 and less than 53.</td>
<td></td>
</tr>
</tbody>
</table>

**Examples:**  

<table>
<thead>
<tr>
<th>KWIC</th>
<th>DIF=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>KWIC</td>
<td></td>
</tr>
</tbody>
</table>

* The name of the default file is installation dependent and consequently is given in Appendix E
Command: NAME-GEN

Default: report command

Purpose: To produce the NAME-GEN report and/or retrieve certain names to be put in a PUNCH file and used as input to other commands.

Prototype: NAME-GEN(NG) [parameter]...

Parameters:

Output-Data

PRINT, NOPRINT(NP)  Default: PRINT

The PRINT parameter initiates the production of a printed output (NAME GEN); NOPRINT suppresses printing of such a report.

PUNCH(P)[=fdname], NOPUNCH  Default: PUNCH

The PUNCH parameter specifies that PUNCH data should be generated and written into the designated PUNCH file. When the PUNCH parameter is used and no fdname is designated, the data is written into the default PUNCH file.* This file is used as the PUNCH file. With the NOPUNCH parameter in effect, no action is taken to generate PUNCH data.

Output-Option

EMPTY, NOEMPTY  Default: (see text)

When EMPTY is in effect (the default when the PUNCH parameter is also used) the PUNCH file is emptied before the list of names is written into it. When NOEMPTY is in effect (the default when PUNCH is not used) no action is taken to empty the PUNCH file.

The following retrieval parameters indicate the types of names to be retrieved and placed in the output file. A "NO" prefix attached to a parameter means that names of that type are not to be retrieved.

<table>
<thead>
<tr>
<th>Name Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIBUTE(ATTR), NOATTRIBUTE(NATTR)</td>
<td>NOATTRIBUTE</td>
</tr>
<tr>
<td>ATTRIBUTE-VALUE(ATTRV), NOATTRIBUTE-VALUE(NATTRV)</td>
<td>NOATTRIBUTE-VALUE</td>
</tr>
<tr>
<td>CONDITION(COND, NOCONDITION(NCOND)</td>
<td>NOCONDITION</td>
</tr>
<tr>
<td>ELEMENT(ELE), NOELEMENT(NELE)</td>
<td>NOELEMENT</td>
</tr>
<tr>
<td>ENTITY(ENT), NOENTITY(NENT)</td>
<td>NOENTITY</td>
</tr>
<tr>
<td>EVENT(EV), NOEVENT(NEV)</td>
<td>NOEVENT</td>
</tr>
</tbody>
</table>

* The name of the default file is installation dependent and consequently is given in Appendix E
**Name Type**

- **GROUP (GR)**, NOGROUP (NGR)
- **INPUT (INP)**, NOINPUT (NINP)
- **INTERFACE (INTF)**, NOINTERFACE (NINTF)
- **INTERVAL (INT)**, NOINTERVAL (NINT)
- **KEYWORD (KEY)**, NOKEYWORD (NKEY)
- **MAILBOX (BOX)**, NOMAILBOX (NBOX)
- **MEMO**, NOMEMO (NMEMO)
- **OUTPUT (OUT)**, NOOUTPUT (NOUT)
- **PROBLEM-DEFINER (PD)**, NOPROBLEM-DEFINER (NPD)
- **PROCESS (PROC)**, NOPROCESS (NPROC)
- **REAL-WORLD-ENTITY (RWE)**, NOREAL-WORLD-ENTITY (RWE)
- **RELATION (RLN)**, NORELATION (NRLN)
- **SECURITY (SEC)**, NOSECURITY (NSEC)
- **SET**, NOSET
- **SOURCE (SRC)**, NOSOURCE (NSRC)
- **SUBSETTING-CRITERION (SSCN)**, NOSUBSETTING-CRITERION (NSSCN)
- **SYSTEM-PARAMETER (SYSP)**, NOSYSTEM-PARAMETER (NSYSP)
- **UNDEFINED (UNDF)**, NOUNDEFINED (NUNDF)

**Default**

- **GROUP**
- **INPUT**
- **INTERFACE**
- **INTERVAL**
- **KEYWORD**
- **MAILBOX**
- **MEMO**
- **OUTPUT**
- **PROBLEM-DEFINER**
- **PROCESS**
- **REAL-WORLD-ENTITY**
- **RELATION**
- **SECURITY**
- **SET**
- **SOURCE**
- **SUBSETTING-CRITERION**
- **SYSTEM-PARAMETER**
- **UNDEFINED**

**None, All**

Default: NONE

With the None parameter, all Name Type switches are set to an "off" position and names of no types (hence, no names) are retrieved. When the ALL parameter is given, all types of names are included in the output except SYNONYMS and UNDEFINED names.

**TOTAL**

Default: no default

The TOTAL parameter specifies the inclusion of all name types including SYNONYMS and UNDEFINED names in the output.

**SUBLEVEL (SL)=integer**

Default: SUBLEVEL=0

This parameter (to be used in conjunction with the SUBPARTS-OF parameter) specifies the level to which the SUBPARTS tree should be traversed and names retrieved. The zero (0) value indicates all levels should be retrieved.

**SUBPARTS-OF (SO)=user-name**

Default: no default

This parameter retrieves all SUBPARTS (down to the level specified by the SUBLEVEL parameter) of the designated user-name. The user-name may be a PROCESS, INPUT, OUTPUT or INTERFACE.

* REAL-WORLD-ENTITY may be used in place of INTERFACE.
IDENTIFIER(ID),
IDENTIFIER-ELEMENT(IDE), IDENTIFIER-GROUP(IDG)

Default: no default

IDENTIFIER retrieves only those names which are used as
IDENTIFIERS in the data base. IDENTIFIER-ELEMENT retrieves
only those that are defined as ELEMENT names and are used
as IDENTIFIERS, and IDENTIFIER-GROUP retrieves only those
defined as GROUP names and are used as IDENTIFIERS.

Specific names can be selected from those names retrieved
by the retrieval parameters when using the following
selection parameters:

BASIC, NOBASIC

Default: BASIC

The BASIC parameter specifies that basic names be included
in the output list of the requested name types. "Basic"
names are those names which are not SYNONYMS.

SYNONYM(SYN), NOSYNONYM(NSYN)

Default: NOSYNONYM

The SYNONYM parameter specifies that in addition to the
basic names retrieved, the synonyms for these basic names
should be included in the output list of requested name
types. If only synonyms are desired then NOBASIC should
also be specified when using the SYNONYM parameter.

KEY=user-name

Default: no default

Only those names with the given user-name as a KEYWORD
are selected to be part of the output. The user-name
must be a name defined as a KEYWORD in the data base.

PD=user-name

Default: no default

Only those names associated with the specified PROBLEM-
DEFINER are selected to be part of the output. The user-
name must be a PROBLEM-DEFINER name defined in the data
base.

Output- ORDER= 
Format

ORDER= \{ALPHA \} \{BYTYPE \}

Default: ORDER=ALPHA

With ORDER equal to ALPHA, the report presents the
retrieved names in alphabetical order by name. BYTYPE
signifies that the names are grouped by name type with
the types alphabetically ordered and names within the same
type ordered alphabetically by name.

Examples:

NG PROCESS RWE
NG ALL
NG ALL KEY=L1
NG SO=TIME-CARD PD=WALTER-J-RATAJ KEY=L1
Command: NAME-LIST

Default: report command

Purpose: To produce the NAME LIST report.

Prototype: NAME-LIST(NL) [parameter]...

Parameters:

Output-Format

ORDER= \{ ALPHA \}

\{ BYTYPE \}

Default: ORDER=ALPHA

If ORDER=ALPHA is specified, the list is ordered by the .URA name of the object, if ORDER=BYTYPE is specified, the order is alphabetical by name type, with objects of the same type being ordered by name.

Examples:

NAME-LIST

NL ORDER=BYTYPE
Command: **PICTURE**

Type: report command

Purpose: To produce the PICTURE report.

Prototype: **PICTURE**(PIC) [parameter]...

Parameters:

- **Input-Data**
  - FILE(F) [=fdname], NAME(N)=user-name
  - Default: FILE

  When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. When a name is given via the NAME parameter, the report is produced only for that name. In any case, the names used as input to this command must be INTERFACE, PROCESS, SET, INPUT, OUTPUT, ENTITY, GROUP and/or ELEMENT names. The format of the input file must be one name per line.

- **Output-Data**
  - INDEX, NOINDEX
  - Default: NOINDEX

  The INDEX parameter specifies the production of an index for the PICTURE report. This index consists of all user defined names used in the report, in alphabetical order and the pages on which they appear in the report.

- **Output-Option**
  - DATA(D), NODATA(ND)
  - Default: DATA

  With the DATA parameter in effect, information applicable and available for the given name, or list of names other than structure or flow data is printed on the output. NODATA inhibits such action.

  - FLOW, NOFLOW
  - Default: FLOW

  This parameter presents flow information in the PICTURE report. It presents RECEIVES and GENERATES information between INPUTS and OUTPUTS with PROCESSES and INTERFACES. It also presents USES and DERIVES information between PROCESSES and data (such as SETS, ENTITIES, GROUPS and ELEMENTS).

- **STRUCTURE**(STR), **NOSTRUCTURE**(NSTR)
  - Default: STRUCTURE

  When the STRUCTURE parameter is in effect, the information available in the SUBPARTS, CONSISTS and/or SUBSETS statements and their complementary statements for the input name(s) appears in the report.

* The name of the default file is installation dependent and consequently is given in Appendix E
Examples:   PICTURE  N=PAYCALC-UPDATING
            PIC  N=PAYROLL-PROCESSING  NODATA  NOFLOW
Command: PRINT-ATTRIBUTE-VALUES  Type: report command

Purpose: To produce the ATTRIBUTE REPORT.

Prototype: PRINT-ATTRIBUTE-VALUES(PAV) [parameter]...

Parameters:

Input-Data  FILE(F) [=fdname], NAME(N)=user-name  Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. The input file format must be one ATTRIBUTE name per line. When a name is specified by the NAME parameter, the report is generated for that name only. In any case, only ATTRIBUTE names may be used as input to this command.

Examples:

PRINT-ATTRIBUTE-VALUES  FILE
PAV  N=TYPE

* The name of the default file is installation dependent and consequently is given in Appendix E
Command: PROCESS-CHAIN
Type: report command

Purpose: To produce the PROCESS CHAIN report.

Prototype: PROCESS-CHAIN(PC) [parameter]...

Parameters:

Input-Data
FILE(F)=[fdname], NAME(N)=user-name  Default: FILE
When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is given, that file is used as the input file for the command. The format for the input file must be one name per line. When a name is given via the NAME parameter, the report is produced only for that name. Regardless of whether FILE or NAME is specified, all names used as input to this command must be EVENT or PROCESS names.

Output-Data
INDEX, NOINDEX  Default: NOINDEX
The INDEX parameter specifies the production of an index for the PROCESS CHAIN report. This index consists of all user-defined names used in the report, in alphabetical order, along with the pages on which they appear in the report.

Output-Option
LINKS=integer  Default: LINKS=1000
The LINKS parameter specifies the maximum number of links (connections between names) to be followed in producing the report. LINKS can take on any integer value between 1 and 1000, inclusive.

Output-Format
COLUMNS(COLS)=integer  Default: COLUMNS=119
The COLUMNS parameter specifies the number of columns to be used for output. The maximum value allowed is 119.
ROWS=integer  Default: ROWS=39
The ROWS parameter specifies the number of rows to be used for output. The maximum value allowed is 39.

* The name of the default file is installation-dependent and consequently is given in Appendix E.
HORIZONTAL-BOXES(HB)=integer  
Default: (see text)

HORIZONTAL-BOXES specifies the maximum number of boxes containing names to be arranged across the page. The default value is the largest possible value for the given value of COLUMNS, and is computed as the greatest integer in (COLUMNS-4)/17.

VERTICAL-BOXES(VB)=integer  
Default: (see text)

VERTICAL-BOXES specifies the maximum number of boxes containing names to be arranged down the page. The default values is the largest possible value for the given value of ROWS, and is computed as the greatest integer in (ROWS-2)/6.

Messages:  
If the HORIZONTAL-BOXES value used will not fit in the number of COLUMNS specified, the message

HORIZONTAL-BOXES TOO LARGE FOR NUMBER OF COLUMNS ON PAGE

will be given.

If the VERTICAL-BOXES value used will not fit in the number of ROWS specified, the message

VERTICAL-BOXES TOO LARGE FOR NUMBER OF ROWS ON PAGE

will be given.

Examples:  
PROCESS-CHAIN N=EVENT1

PC FILE LINKS=4 INDEX
Command: PROCESS-INPUT-OUTPUT

Type: report command

Purpose: To produce the PROCESS INPUT/OUTPUT report.

Prototype: PROCESS-INPUT-OUTPUT(PRIO) [parameter]...

Parameters:

**Input**

- **FILE(F)=[fdname]**, **NAME(N)=user-name**  
  Default: FILE

  When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command and the report is produced using all the names in the file. When a single name is specified by the NAME parameter, the report is produced for that name alone. Either FILE or NAME can be used but not both. In any case, all the names used as input to this command must be PROCESS names. The input file format is one PROCESS name per line.

**Output**

- **INDEX, NOINDEX**  
  Default: NOINDEX

  When given, the INDEX parameter specifies the production of an index into the report. The index consists of all input and output names in the report, in alphabetical order and the page(s) on which they occur in the report.

- **PRINT, NOPRINT(NP)**  
  Default: PRINT

  The NOPRINT parameter specifies that no printed output report will be produced. The PRINT parameter specifies the production of the PROCESS INPUT/OUTPUT report.

- **DESCRIPTION(Desc), NODESCRIPTION(NDesc)**  
  Default: DESCRIPTION

  When the DESCRIPTION parameter is in effect, the comment entry associated with the DESCRIPTION statement, for all PROCESS used as input, is retrieved and printed on the report. NODESCRIPTION specifies that this information is not to be retrieved.

- **PROCEDURE(PRCD), NOPROCEDURE(NPRCD)**  
  Default: NOPROCEDURE

  When the PROCEDURE parameter is specified, the comment entry associated with the PROCEDURE statement, for each PROCESS name used as input, is retrieved and printed on the report. With the NOPROCEDURE parameter in effect, this information is not retrieved.

* The name of the default file is installation dependent and consequently is given in Appendix E
INPUT(INP), NOINPUT(NINP)  Default:  INPUT

When the INPUT parameter is in effect, all the names of objects used as input to each PROCESS (i.e., names associated with the RECEIVES and USES statements) are retrieved and printed on the report. The NOINPUT parameter specifies that this information is not to be retrieved.

OUTPUT(OUT), NOOUTPUT(NOUT)  Default:  OUTPUT

When the OUTPUT parameter is in effect, all the names of objects designated as output from each PROCESS (i.e., names associated with the GENERATES, DERIVES, and UPDATES statements) are retrieved and printed on the report. The NOOUTPUT parameter specifies that this information is not to be retrieved.

NEW-PAGE(NPG), NONEW-PAGE(NNPG)  Default:  NONEW-PAGE

When given, the NEW-PAGE parameter specifies that each section of the PROCESS INPUT/OUTPUT report be printed on a separate page. NONEW-PAGE signifies that the sections will follow one another on a page within the page size restrictions. In any case, interrupted sections will be continued on succeeding pages.

Examples:

PRIO  N=PAYROLL-PROCESSING

PRIO  F  NDESC  NPG  PRCD
Command: PUNCH-COMMENT-ENTRY  Type: report command

Purpose: To produce the PUNCHED COMMENT ENTRIES report and/or punch the specified comment entries into a PUNCH file.

Prototype: PUNCH-COMMENT-ENTRY(PCOM)[parameter]...

Parameter:

Input-Data  FILE(F)[=fdname], NAME(N)=user-name  Default: FILE

When the FILE parameter is used and no fdname is designated, the contents of the default file* are used as input to the command. This file is the default PUNCH file for NAME-GEN. If an fdname is indicated, that file is used as the input file for the command. When a name is specified by the NAME parameter, the output is produced for that name alone. The format of the input file must be one name per line.

Output-Data  PRINT, NOPRINT(NP)  Default: PRINT

The PRINT parameter initiates the production of printed output for the report. When the NOPRINT parameter is given, the PUNCH COMMENT ENTRIES report is not produced.

PUNCH(F)[=fdname], NOPUNCH  Default: PUNCH

The PUNCH parameter specifies that PUNCH data should be generated and written into the designated PUNCH file. When the PUNCH parameter is used and no fdname is designated, the data is written into the default PUNCH file.* This file is the default PUNCH file for the command. If an fdname is indicated, that file is used as the PUNCH file. With the NOPUNCH parameter in effect, no action is taken to generate PUNCH data.

Output-Option  EMPTY, NOEMPTY  Default: (see text)

When EMPTY is in effect (the default when the PUNCH parameter is also used) the PUNCH file is emptied before PUNCH data is written into it. When NOEMPTY is in effect (the default when PUNCH is not used) no action is taken to empty the PUNCH file.

* The name of the default file is installation dependent and consequently is given in Appendix E
The comment entries associated with the following types of comment entry statements are retrieved when given as parameters.

<table>
<thead>
<tr>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>DERIVATION(der)</td>
<td>NODERIVATION</td>
</tr>
<tr>
<td>DESCRIPTION(desc)</td>
<td>NODESCRIPTION</td>
</tr>
<tr>
<td>FALSE-WHILE(fw)</td>
<td>NOFALSE-WHILE</td>
</tr>
<tr>
<td>PROCEDURE(prcd)</td>
<td>NOPROCEDURE</td>
</tr>
<tr>
<td>TRUE-WHILE(tw)</td>
<td>NOTRUE-WHILE</td>
</tr>
<tr>
<td>VOLATILITY(vol)</td>
<td>NOVOLATILITY</td>
</tr>
<tr>
<td>VOLATILITY-MEMBER(volm)</td>
<td>NOVOLATILITY-MEMBER</td>
</tr>
<tr>
<td>VOLATILITY-SET(vols)</td>
<td>NOVOLATILITY-SET</td>
</tr>
</tbody>
</table>

Examples:

PCOM N=PAYROLL-PROCESSING DESC
PCOM F DESC PRCD
Command: RENAME

Type: modifier command

Purpose: To change the name of some object in the data base and to produce the RENAME REPORT as a permanent record of the change.

Prototype: RENAME(REN) \{ OLD(O)=user-name \ NEW(N)=user-name \ INPUT(I)=fname \}

Parameters:

Input-Data

INPUT(I)=fname Default: no default

For multiple name changes, an input file can be used. Each line of the file must consist of the old name followed by the new name. The two names must be separated by one or more blanks.

OLD(O)=user-name Default: no default

The user-name specified here is the name that is to be changed. This name must be defined in the data base.

NEW(N)=user-name Default: no default

The user-name specified here is the name to replace the old name. If the new name is already in the data base, the name will not be changed.

For a single change, both OLD and NEW must be given with legal values.

Messages: If neither INPUT nor the OLD and NEW parameters are specified the message:

MUST GIVE OLD AND NEW, OR INPUT will be given.

Examples: RENAME OLD=EMPLOYEE-CODE NEW=EMPLOYEE-NUMBER
Command: REPLACE-COMMENT-ENTRY  Type: modifier command

Purpose: To replace, for a given name, specific comment entries associated to it. A REPLACED COMMENT ENTRIES report is also printed as a permanent record of the change.

Prototype: REPLACE-COMMENT-ENTRY(RCOM) [parameter]...

Parameters:

Input-
Data  INPUT(I)=fname  Default: (see text)

The designated fname contains the new comment entries that will replace specified old comment entries in the data base. The required format of the file is the same as that punched by PUNCH-COMMENT-ENTRY. If INPUT is not given, the input will be taken from the default file.* This file is the default PUNCH file for PCOM. For each comment entry to be replaced, the following format must be given in the input file:

name

comment-entry type;


comment entry text


Where name is a name defined in the data base, the comment-entry-type (e.g., DESCRIPTION, VOLATILITY, etc.) must be followed by a semicolon. The text following this must also be followed by a semicolon. This sequence of lines can be repeated as many times as necessary in the input file.

Output-
Data  PRINT, NOPRINT(NP)  Default: PRINT

The PRINT parameter initiates the production of the REPLACED COMMENT ENTRIES report; NOPRINT suppresses printing. The report, if produced, contains both the old and new comment entries.

Examples:  RCOM  NOPRINT

* The name of the default file is installation dependent and consequently is given in Appendix E
Command: STRUCTURE  
Type: report command

Purpose: To produce a STRUCTURE report for INPUTS, OUTPUTS, PROCESSES, or INTERFACES.

Prototype: STRUCTURE(STR) [parameter]...

Parameters:

Output-Data

INDEX, NOINDEX  Default: NOINDEX

The INDEX parameter, when given, specifies the production of an index to the report, giving the pages on which each undefined name used in the report occurs. NOINDEX specifies that no index should be generated.

Output-Option

URA will produce structure reports for the following name types when given as parameters. (Only one may be given for each report.)

INPUT(INP)  Default: PROCESS
OUTPUT(OUT)
PROCESS (PROC)
INTERFACE (INTF)*

Output-Format

INDENT(IND)=integer  Default: INDENT=3

The number is the number of spaces to indent each succeeding level in the report. INDENT must take on some value greater than 0 and less than 11.

Examples: STRUCTURE
STR  INPUT

* REAL-WORLD-ENTITY (RWE) may be used in place of INTERFACE.
Command: SUMMARY

Default: report command

Purpose: To produce the DATA BASE SUMMARY output.

Prototype: SUMMARY(SUM)

Parameters: no parameters

Examples: SUMMARY
SUM
PART V

AUTOMATED DOCUMENTATION SYSTEM

USERS' MANUAL
INTRODUCTION

Whenever an organization develops an information system, one of the major problems is maintaining it over its life. The developers of the system are rarely around, or available to help the maintenance group in their task. A frequent result of this situation is a complete redesign and redevelopment of the system to make relatively minor modifications to the system. Such practice is an obvious waste of resources that could have been avoided with proper documentation. While the above is a strong motivator for the need for documentation standards, there are other advantages involved, namely, better communication between developers of the system, easier training of new employees, etc. Hence, most organizations have developed their own standards which must be adhered to for documenting the system. Examples of such standards are MILITARY standards 483 and 490 and Department of Defense Manual 4120.17-M.

Much of the information that is needed in the final document describing the system may be obtained during the system's analysis, design, and construction phases of the development life cycle. Therefore, what is needed is a mechanism that will allow the capture of the information as and when it becomes available and storing it in a readily usable data base. This is where the URL/URA data base comes into the picture.

The Automated Documentation System basically is composed of three parts. These are the documentation schema, the documentation source, and the Analyzer data base. These areas are discussed in this part.
1.0 Document Initialization

The initialization for the Automated Documentation System must be performed for each project to be documented. This task may be performed by the project leader. It involves taking the Military Standard and deriving from them relevant headings for the particular project in mind, as shown in the example (Appendix F). The initialization allows the project director to identify the major and detailed areas of documentation to be produced by the project team members. It can be most succinctly expressed as the table of contents to be shown in the final documentation. In addition, the project manager can decide to allow for up to three generic types of pages such as figure, appendices, etc., by specifying these in the initialization. The details of the syntax will be discussed in section 1.2. Development of a documentation schema is discussed in section 4.1. Section 1.1 will discuss some of the strategy to be followed, and the general initialization process. The end result of the initialization process is a documentation object schema.

1.1 Strategy for the document initialization, and general process

There are basically three types of entries that may be entered into the documentation schema. An entry line is a maximum of 80 characters. The entries may be typed in on cards, or other media; in this form they are known as documentation source schema. When these same entries have been entered on a file on the target computer which also has the Analyzer database on file, the entries are known as the documentation object schema. The process of converting the source schema into the object schema would generally involve nothing more than reading in the cards, etc. into a file. One of the most common ways of doing this is by a local text editor, or some other file populating mechanism. This processor (editor, etc.) is known as the documentation initializer (Figure 65).

The three types of entries that may be entered into the documentation schema are:

- the table of contents: (SECTION ENTRIES)
- formatting commands (TEXT FORMATTING ENTRIES)
- comments (COMMENT ENTRIES)

The table of contents entries are all preceded by a distinguishing character, "#" which must be entered on every heading to appear in the table of contents. Formatting commands are proceeded by the distinguishing character "*".

Finally, the manager may want to insert, into the documentation schema, personal comments which are not to appear in the final table of contents. These may be specified by a ' &' followed by a comment of up to 79 characters. Should the comment be longer, it may be started on the next line but preceded by another '&'. Comments may be used to guide the documentors, to assign specific portions of the documentation to particular individuals, etc. Any formatting command that appears in the schema will be executed directly before the section is printed in the document. This characteristic will be discussed in Section 4.1.
1.2 Syntax for entries in the Documentation Schema

SECTION-ENTRIES have the following syntax:

#SECTION IDENTIFIER [HEADING-ENTRY]

The section-identifier must be preceded by a "#" in column 1. The section-identifier is the unique string corresponding to that section of the documentation schema (e.g., 2.2.3). This string may be made up of numbers, characters and/or special characters. No particular restrictions are placed on the format of the identifier; for example, an identifier of 1.2.a.1.b is a perfectly legitimate identifier. The section entry may not exceed 80 characters. It is important to note that there must be at least one space between the identifier and the heading entry. The final documentation will be produced in the order that the identifiers were entered in the documentation schema. Hence, it is the responsibility of the individual preparing the documentation schema to enter them in the proper order. The documentation generator will not check to see that these are sorted. The section-identifier is the title to be given to the section. This is the title that will appear in the final document.

TEXT-FORMATTING-ENTRIES have the following syntax:

*CONTROL-INFORMATION-ENTRY

The text formatting commands may also be interspersed anywhere in the schema, but must be preceded by a "**" in column 1. There are two types of formatting commands. One type controls what is going to be done at a certain point in regards to spacing; the other type controls the global options of formatting.

The following options are invoked as needed:

*NEW-PAGE - Skip to the top of the next page.

*SKIP [N] - Skip N lines. If N is greater than the number of lines left on a page, then this command causes carriages to skip to the top of the next page. N defaults to 1.

*HOLD-BLANK [N] - Check to see if N lines are left on this page. If there are, skip N lines. If there are not, skip to the next page and then skip N lines. This option insures that enough space will be left on a single page for a diagram or able to be written in manually. N defaults to 1.

*HOLD [N] - The rest of the current page is checked to see if N lines are left on the page (not including HEADING or BOTTOM-LINES). If N lines are left, the next line of the source data base is produced. If N lines are not left on the current page, skip to the top of the next page and produce the next line of source. Default is 1.

These options are present at the commencement (under their default conditions and can be changed throughout the source):
*TITLE (OFF) - If a title is given, this title is put on the top of each page. If OFF is given, no title is printed on each page. Default is OFF.

*HEADING (ON) - If ON is specified, the HEADING-ENTRY for the last section printed is put at the bottom of each page. If OFF is specified, the HEADING-ENTRY is not put at the bottom of each page. Default is ON.

*TOP-LINES [N] - Skip N lines at the top of each new page. N defaults to 3. Range is from 0 to 10.

*BOTTOM-LINES [N] - Skip N lines at the bottom of each page. N defaults to 2. Range is from 0 to 10.

*LINES [N] - N is the number of lines to be printed on a page, including Title and Heading. N defaults to 60. Range is from 5 to 60.

*SOURCE-MARGIN [N] - N is the number of spaces to indent the actual text material. This does not effect Analyzer report indentation. N defaults to 5. Range is from 1 to 40.

*HEADING-MARGIN [N] - N is the number of spaces to indent a section heading - N defaults to 10. Range is from 1 to 40.

*HEADING-SKIP [N] - N is the number of lines to skip before printing a section-header. The amount skipped will at most be to the top of the next page. N defaults to 2. Range is from 0 to 60.

*REPORT-CONTROL (OFF) - When set to ON, the Analyzer report carriage control is engaged when outputting Analyzer reports. This affects page feeding of the document produced. When OFF, the line skipping and page feeding is controlled by only formatting commands. Default is ON. 

COMMENT-ENTRIES have the following syntax:

& COMMENT

Comments may be interspersed anywhere in the documentation schema. They will not appear in the final output. The "&" must appear in column 1. The comment on a given line can be a maximum of 79 characters.

There is no limit to the number of comment entries and header entries in a documentation source schema. An example of a documentation source schema is:

#2. PROJECT DEVELOPMENT ENVIRONMENT/DOCUMENTATION SYSTEM

#2.1 DEVELOPMENTAL PHASES

#2.1.1 INITIATION

& THERE SHOULD BE NO NEED TO HAVE ANY OTHER LISTS FOR US.

(A larger example is in appendix F)
2.0 Documentation data base population

The actual documentation to be written would probably be done by more than one individual. These individuals may enter the documentation of the areas they are responsible for, into different files, or into one common file assigned to be shared by them. In the former case, the documentation manager must merge the different pieces of the documentation into a single file. The order of this file is not important, as the final documentation will be produced in the order specified by the documentation schema (see section 1).

2.1 Strategy for the documentation data base population

It is important that the individuals who are going to document the system know what is in the Analyzer data base. They should be given a complete NAME-LIST* (or NAME-GEN*) of names in the Analyzer data base, and have either a FORMATTED-PROBLEM-STATEMENT* of all these names, or have access, via a terminal, to the Analyzer data base in order to easily determine what is already in the Analyzer data base. For example, if the documentation calls for the description of a particular input (say TEST-RESULTS), this should be in the Analyzer data base, and the documentor should be required to invoke its description from the Analyzer data base rather than to have to re-type it. This has two obvious advantages: the one mentioned above (reduction of redundant work), and the other one of always having an up-to-date description of the item of interest. It is the duty of the project manager to see that all Analyzer entries have been made, and do exist. The above also implies that the documentor be quite familiar with the use of the Analyzer.

The individuals populating the documentation data base must also have access to the documentation schema, and must use the same section-identifiers as used in the schema. It is also feasible to have a limited amount of text formatting capability such as getting to the top of the next page, or skipping to next half page, etc. This capability would be used where a diagram, or table (which could not be drawn by the documentation generator) had to be inserted, and room had to be left for it. An entry line for the documentation data base is a maximum of 120 characters.

To distinguish between the various kinds of entries in the documentation data base, Analyzer entries are preceded by a "%" in column, text-formatting entries are preceded by a "*" in column 1, and the actual text itself has nothing preceding it, and begins in column 1. Preceding blanks are considered part of the text. An entry line for the documentation data base is a maximum of 120 characters.

Part IV, "Input Requirements Analyzer Command Descriptions" and Part III, "URA Outputs" for an explanation of these commands and corresponding outputs.
The documentation data base may be populated by the documentors in much the same way as the documentation schema (Figure 66), i.e., any processor such as the test editor, file populator, etc., may be used to store the source in the documentation data base which must be a file. If several documentors are working simultaneously, different files may be assigned to individual documentors. These portions of the data base can then be merged at documentation generation time. The order of the entries is unimportant.

2.2 Syntax for the Documentation Data Base

SECTION-ENTRIES have the following syntax:

```
#SECTION-IDENTIFIER [HEADING-COMMENT]
```

The section entries are the same as in the documentation schema. However, the text portion (heading comment) in the documentation data base is ignored. Only the section-identifier is used to match against the documentation schema. The "#" must be in Column 1.

ANALYZER-COMMAND-ENTRIES have the following syntax:

```
% ANALYZER-COMMAND
```

Analyzer commands may be interspersed anywhere in the text with the provision that they be preceded by a "%" in column 1. Any Analyzer commands from Part IV are allowed except the STOP command and the SET command with the option PROMPT=ON in effect (by default it is not in effect). The commands are not executed immediately. They are just stored as is until the documentation generation process.

TEXT-FORMATTING-ENTRIES have the following syntax:

```
* CONTROL-INFORMATION-ENTRY
```

The text formatting commands available to use in the schema will also be available to use in the source. They may be interspersed anywhere in the documentation source. The "*" must again be in column 1.

---

* Part IV,
  Description "User Requirements Analyzer Command"
COMMENT-ENTRIES have the following syntax:

&COMMENT

Comments may be interspersed anywhere in the documentation source. They will not appear in the final output (same as in the schema). The "&" must be in column 1.

In the documentation source, the line size maximum is 120 characters. Text entries have no specific format.

3.0 The Documentation Generation

The documentation generation may be performed by the project manager when one wishes to have the completed documentation. The manager may first wish to ascertain that the documentation is indeed complete. It is possible for the manager to find this out. However, the extent of checking is limited to the absence, or presence of an entry in the data base corresponding to every entry in the schema. There is no check to find out if the corresponding Analyzer data base entries also exist.

The manager also has the flexibility of specifying which data base, and schema are to be considered. If one should so desire, the generator will produce only the table of contents.

If the manager finds that all entries in the schema have corresponding entries in the database, one can now go ahead and ask the generator to produce the final documentation as per the specification in the schema. Optionally, a title may be specified on the final document. The manager must realize that this will be a lengthy process, as it involves the invocation of several analyzer commands, the generation of intermediate files, and the merging of several files to produce the final documentation (Figure 67). Hence, it is recommended that this be done in batch mode. However, because the Analyzer processor is being run when the documentation is desired, it will be using the latest version of the descriptions, etc., that are available.

The process that the Documentation Generator goes through is shown in Figure 67. It involves two steps. Phase I (CHECK) checks to see what Section Entries in the schema are present in the documentation source. It also extracts all Analyzer commands from the source and writes them into a separate file to be used as input to the Analyzer. Phase II (MERGE) merges the output from the Analyzer with the documentation source and produces the document. It is at this point that the text formatting commands are processed.

The procedure for documentation generation is dependent on the operating system under which the Automated Documentation System is installed. Hence there is no one way to invoke the Automated Documentation System. Appendix H will include the system dependent information required to execute the Automated Documentation System and generate a document.
DOCUMENTATION GENERATOR

Figure 67

350
4.0 Usage of the Documentation Generator

In using the documentation generator, there are several areas that need to be considered in order to achieve a desired quality level. Coordination between the schema, source and Analyzer data base are mandatory in order to develop a document which will contain complete information. Development of the methods to produce these three inputs into the system so that they can be used in an on-going way (camera-ready documentation) is the topic of this section. Considerations as to what should be included in each component, what pitfalls should be watched out for, and what variations might arise in usage of the documentation generator will be discussed. It is important for the reader to realize at this point that while this system has been developed to provide a generalized format for producing documents under a set of standards, the system still allows a great deal of flexibility to the user.

Department of Defense Manual 4120.17-M along with MIL Standard 483 will be used as example documentation standards in this discussion. Development of a Functional Description and Data Requirements Document of the Pay system Analyzer data base (W.P. 74*) will be used as a specific example (Appendix F).

4.1 Development of the Documentation Schema

The documentation schema consists of a structural outline or table of contents of the particular document being produced. From this schema a table of contents for the document being generated will be produced. Hence, it is important that the user include enough section levels in order to make the table of contents meaningful. It is also important to note that the final documentation will be produced in the order stated in the documentation schema, so the schema should be set up appropriately.

Format for the schema is discussed in section 1. The documentation initializer (see Figure 65) takes the documentation schema and produces a documentation object schema, which is stored permanently on file after any require editing is done.

Within the schema, formatting commands are allowed. The schema is checked to see if it contains any formatting commands. If it does, each formatting command goes into effect prior to the printing of the section header and immediately following the formatting command. This is also true for commands at the beginning of the schema.

Although it is suggested that the table of contents of a particular document or standard be used as the schema, many times a more detailed schema is warranted. Situations where more or less detail might be deemed necessary are in the following subsections.

* "An Example of the Use of URL Using Top-Down Analysis."
4.1.1 Where less detail might be necessary

The table of contents of a certain standard is a skeleton type set up, giving suggestions on how the documents should be developed, but leaving room for variances in structure from one document to another written according to the standard. In this case it might be wise to use less detail in the schema and keep it as general as possible.

An example of where this situation arises is in MIL standard 483. In Section 3.2, the standard sets up the criteria for describing each separate function of a system. Since there is going to be a varying number of functions in different systems, the standard sets up the description then as follows:

3.2.X Function X
3.2.X.1 INPUTS
3.2.X.2 PROCESSING
3.2.X.3 OUTPUTS

Thus, this basic structure is repeated for every function in the system. This causes a definite problem when using the document generator. Since it is desirable to have one schema for all documents produced, it can be seen that since the number of functions will vary from one system to the next, the standard's table of contents cannot be followed strictly.

Three options are open to the documentor in cases like this.

a. Include in the schema only the section header (#3.2 DETAILED FUNCTIONAL REQUIREMENTS) and leave the rest to be described in the source. Comment entries could be included in the schema to describe what should be included with these. The advantage of this option is that the schema will be constant and independent of each instance of this document. The disadvantage is that the schema will now lack information and the table of contents will be somewhat incomplete.

#3.2 DETAILED FUNCTION REQUIREMENTS
& PARAGRAPH 3.2X FUNCTION X.
& THE BASIC PARAGRAPH FOR EACH FUNCTION SHALL BEGIN WITH
& DESCRIPTIVE AND INTRODUCTORY MATERIAL WHICH DEFINES THE
& FUNCTION AND ITS RELATIONSHIP TO OTHER FUNCTIONS. THEN, THE
& FOLLOWING THREE SUBPARAGRAPHS SHALL SPECIFY THE QUALITATIVE
& REQUIREMENTS CONCERNING THE FUNCTION.
& PARAGRAPH 3.2.X.1 INPUTS.
& PARAGRAPH 3.2.X.2 PROCESSING.
& PARAGRAPH 3.2.X.3 OUTPUTS.

b. Set up the schema in the following way:

#3.2.1 FUNCTION ONE
#3.2.1.1 INPUTS
#3.2.1.2 PROCESSING
#3.2.1.3 OUTPUTS
#3.2.2 FUNCTION TWO
#3.2.2.1 INPUTS
#3.2.2.2 PROCESSING
#3.2.2.3 OUTPUTS
 FUNCTION TWELVE
  03.2.12.1 INPUTS
  03.2.12.2 PROCESSING
  03.2.12.3 OUTPUTS

The advantage of this option is that the table of contents is complete. The disadvantage is that the entries in the table are not meaningful.

c. Include in the schema the actual function names for each system. An example of this would be:

  #3.2 DETAIL FUNCTION REQUIREMENTS
  #3.2.1 SENSOR CALIBRATION (INITIAL) FUNCTION
    #3.2.1.1 INPUTS
    #3.2.1.2 PROCESSING
    #3.2.1.3 OUTPUTS
  #3.2.2 ELEMENT CORRECTION (INITIAL) FUNCTION
    #3.2.2.1 INPUTS
    #3.2.2.2 PROCESSING
    #3.2.2.3 OUTPUTS
  #3.2.3 MANEUVER DETERMINATION CONTROL FUNCTION
    #3.2.3.1 INPUTS
    #3.2.3.2 PROCESSING
    #3.2.3.3 OUTPUTS
  #3.2.4 SIMULTANEOUS SOLUTION OF MANEUVER AND ELEMENTS FUNCTION
    #3.2.4.1 INPUTS
    #3.2.4.2 PROCESSING
    #3.2.4.3 OUTPUTS
  #3.2.5 ELEMENT MAINTENANCE CONTROL FUNCTION
    #3.2.5.1 INPUTS
    #3.2.5.2 PROCESSING
    #3.2.5.3 OUTPUTS
  #3.2.6 ELEMENT CORRECTION (ROUTINE) FUNCTION
    #3.2.6.1 INPUTS
    #3.2.6.2 PROCESSING
    #3.2.6.3 OUTPUTS
  #3.2.7 OBSERVATION EDITING FUNCTION
    #3.2.7.1 INPUTS
    #3.2.7.2 PROCESSING
    #3.2.7.3 OUTPUTS
  #3.2.8 SENSOR CALIBRATION (ROUTINE) FUNCTION
    #3.2.8.1 INPUTS
    #3.2.8.2 PROCESSING
    #3.2.8.3 OUTPUTS
  #3.2.9 ELEMENT RECOVERY CONTROL FUNCTION
    #3.2.9.1 INPUTS
    #3.2.9.2 PROCESSING
    #3.2.9.3 OUTPUTS
  #3.2.10 ELEMENT CORRECTION (RECOVERY) FUNCTION
    #3.2.10.1 INPUTS
    #3.2.10.2 PROCESSING
    #3.2.10.3 OUTPUTS

...
The advantage of this method is that the schema is complete and descriptive. The disadvantage is that the schema will have to be changed for each application document.

The decision as to which one of these options should be used is up to the discretion of the user.

4.1.2 Where more detail might be necessary

If within the standard, references are made to information that is deemed necessary to include within a certain section, it might be advisable to include this information in the schema. An example of this would be in the Data Requirements Document of D.O.D. Manual 4120-17-M. In this document, the table of contents looks as follows:

1. GENERAL DATA REQUIREMENTS
   1.1 PURPOSE OF DATA REQUIREMENTS
   1.2 PROJECT REFERENCES
   1.3 MODIFICATION OF DATA REQUIREMENTS
2. DATA DESCRIPTION
   2.1 LOGICAL ORGANIZATION OF STATIC SYSTEM DATA
   2.2 LOGICAL ORGANIZATION OF DYNAMIC INPUT DATA
   2.3 LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA
   2.4 INTERNALLY GENERATED DATA
   2.5 SYSTEM DATA CONSTRAINTS
3. USER SUPPORT FOR DATA COLLECTION
   3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
   3.2 RECOMMEND SOURCE OF INPUT DATA
   3.3 DATA COLLECTION AND TRANSFER PROCEDURES
   3.4 DATA BASE IMPACTS

Within Section 3.1, the standard refers to nine areas of supplementary information. If this information is considered to be of such a nature that it will be included in most all the documents being produced, the user might want to include the information in the schema (therefore in the table of contents of the document). The schema would then look like:
II. GENERAL DATA REQUIREMENTS

1. PURPOSE OF DATA REQUIREMENTS
2. PROJECT REFERENCES
3. MODIFICATION OF DATA REQUIREMENTS

2. DATA DESCRIPTION

1. LOGICAL ORGANIZATION OF STATIC SYSTEM DATA
2. LOGICAL ORGANIZATION OF DYNAMIC input DATA
3. LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA
4. INTERNALLY GENERATED DATA
5. SYSTEM DATA CONSTRAINTS

3. USER SUPPORT FOR DATA COLLECTION

1. DATA COLLECTION REQUIREMENTS AND SCOPE

1. INPUT SOURCE(S) OF THE DATA ELEMENT
2. INPUT MEDIUM
3. RECIPIENTS
4. CRITICAL VALUE
5. SCALES OF MEASUREMENT
6. CONVERSION FACTORS
7. OUTPUT FORM/DEVICE
8. EXPANSION FACTORS
9. FREQUENCY OF UPDATE

2. RECOMMEND SOURCE OF INPUT DATA

3. DATA COLLECTION AND TRANSFER PROCEDURES

4. DATA BASE IMPACTS

It might be that the documentor just needs to be reminded of the supplementary information. In this case it might be appropriate just to include an appropriate comment.

1. GENERAL DATA REQUIREMENTS

1. PURPOSE OF DATA REQUIREMENTS
2. PROJECT REFERENCES
3. MODIFICATION OF DATA REQUIREMENTS

2. DATA DESCRIPTION

1. LOGICAL ORGANIZATION OF STATIC SYSTEM DATA
2. LOGICAL ORGANIZATION OF DYNAMIC INPUT DATA
3. LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA
4. INTERNALLY GENERATED DATA
5. SYSTEM DATA CONSTRAINTS

3. USER SUPPORT FOR DATA COLLECTION

1. DATA COLLECTION REQUIREMENTS AND SCOPE

1. INPUT SOURCE(S) OF THE DATA ELEMENT
2. INPUT MEDIUM
3. RECIPIENTS
4. CRITICAL VALUE
5. SCALES OF MEASUREMENT
6. CONVERSION FACTORS
7. OUTPUT FORM/DEVICE
8. EXPANSION FACTORS
9. FREQUENCY OF UPDATE

2. RECOMMEND SOURCE OF INPUT DATA

3. DATA COLLECTION AND TRANSFER PROCEDURES

4. DATA BASE IMPACTS
The decision as to what should be in the schema is based primarily on how much detail is wanted and/or expected in the table of contents.

4.2 Development of the Documentation Source Data Base

The format of the documentation source and a description of how it is transformed into the documentation database is in Section 2. There are several areas that the user need be aware of when developing the documentation database. Proficiency in these areas is necessary if the documentor is to attain full range capabilities of the document generator.

4.2.1 Knowledge of the Analyzer Data Base

It is of utmost importance that the documentor have a thorough knowledge of the Analyzer data base as well as the Analyzer command language. The documentor must be able to pick out the information needed for the document. When it is known that a system will have to be documented using a certain standard, it is also important that the person developing the Analyzer data base have a full understanding of what information is needed to fulfill the standard's documentation requirements. Since the standard usually requires complete information about the system, writing the Analyzer data base with the standard in mind will usually serve to insure a complete system description.

4.2.2 Text Material

The documentation source data base for various projects written under a certain standard should be developed in such a way as to minimize the differences in the various Source data bases. This can cut down substantially on the documentor's time spent on each system. Another advantage of doing this is that once a certain combination of commands has been determined as the best way to convey certain information required by a particular standard, then that combination can be repetitively used for all projects documented under this particular standard.

With this in mind, the documentor should try to include in the source only such text that will be present in every instance of the document being produced. Examples of such text are in paragraphs found in both the Functional Description and Data Requirements Documents. These paragraphs describe the purpose of each of the documents:

Section 1.1

This Functional Description for (Project Name) (Project Number) is written to provide:

a. The system requirements to be satisfied which will serve as a basis for mutual understanding between the user and the developer.
b. Information on performance requirements, preliminary
design, and user impacts, including fixed and continuing
costs.

c. A basis for the development of system tests.

Section 1.1

This objectives of this Data Requirements Document for
(Project Name) (Project Number) are to list and define
data elements which the system must handle and to
communicate data collection requirements to the user.

Another example of where text should be used is when a
common segment is being described. An example of this is
Section 3.3 of the Functional Description. Here is an example
of how the source might look:

```
3.3 INPUTS/OUTPUTS

INPUTS:
%NG INPUT NOPRINT
%FPS
%SKIP 2

OUTPUTS:
%NG OUTPUT NOPRINT
%FPS
ZSTR OUTPUT
```

Since the documentation source data base will be different for
each project documented, it is not necessary that the text be
constant. However, this practice reduces redundancy in work
done by the documentor.

4.2.3 Analyzer Commands

As mentioned before, the documentor must have a full understanding
of the Analyzer command capabilities (described in Part IV*)
including all possible options. This knowledge is
essential for the document generator to produce complete information
in a readable form. Some of the commands and options that might be
useful are listed below:

<table>
<thead>
<tr>
<th>Command</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCOM</td>
<td>Options: DESCRIPTION</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOPUNCH</td>
</tr>
<tr>
<td>NG</td>
<td>Options: SUBLEVEL=#</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SUBPART-OF=NAME</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOPRINT, PRINT</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>KEY=keyword-name</td>
</tr>
<tr>
<td>CONTENTS</td>
<td></td>
</tr>
<tr>
<td>PICTURE</td>
<td>Options: NODETA</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOFLOW</td>
</tr>
<tr>
<td></td>
<td>NOSTRUCTURE</td>
</tr>
</tbody>
</table>

* Part IV, "User Requirements Analyzer Command Descrip-
tions."
The Data-Process Report could be very beneficial, especially since it produces a very complete description of data flow and process functions. It can be used when an overall interaction description is needed. This report also shows if consistency is present as well as showing completeness (or incompleteness).

The STOP command should not be used. There are three options of the SET command that should not be used: PROMPT=ON will virtually destroy the document being produced. It is also recommended that OUTPUT=XXX, HEADING=ON, PARM=ON not be used. (All four of these options default to OFF.)

4.2.4 Formatting Entries

As shown in Section 2.2, the text formatting entries allow the user a great deal of maneuverability in regards to the final document format. At this point, a diagram might help to show just what is going on a page of a document.
It is important for the documentor to understand the formatting commands. There are several points that should be remembered when using the commands. These are covered in the following subsection.

4.2.4.1 Defaults of Formatting Commands

The formatting options can be changed at any time in the Source. The formatting commands default affect how the document will appear. If one does not specify a value for these commands, then the defaults will be used. When the documentor does not specify these command values, then the final document will appear as if the defaults were included in the Source database. The formatting command defaults are:

* REPORT-CC ON
* TITLE OFF
* HEADING ON
* TOP-LINES 3
* BOTTOM-LINES 2
* LINES 60
* SOURCE-MARGIN 5
* HEADING-MARGIN 5
* HEADING-SKIP 2

The last occurrence of each option is the one in effect.

4.2.4.2 Analyzer Report Indentation

Analyzer report indentation is not affected by either HEADING-MARGIN or SOURCE-MARGIN. Care must be taken in setting these two options so the document being produced is in a logical readable format.

4.2.4.3 LINE SKIPPING

HEADING-SKIP reacts the same way as SKIP does when an end of page is encountered. Also, the documentor should be careful when trying to set up a section-header so it starts on a new page. (The combination of NEW-PAGE with a HEADING-SKIP will cause the Section-Identifier not to be printed at the top of the new page. If the value of SKIP or HEADER-SKIP is more than the current LINES values then the generator automatically skips to the top of the next page.

4.2.4.4 Usage of REPORT-CC

Usage of the REPORT-CC command will have a direct bearing on the paging of the document being processed. Since when REPORT-CC is ON, analyzer reports will be produced according to their normal paging sequence. This is helpful when a PICTURE Report is being produced and more than one name is input.
or the PICTURE must be continued on the next page. Other reports where this option is helpful are the DATA PROCESS Report and the CONSISTS-COMPARISON MATRIX Report. In cases where a report will be taking up more than one page it is important to start the report at the top of a fresh page so that the Analyzer paging will match the documentation generator's paging. There are cases, though, where the documentor will wish to set REPORT-CC OFF, such as where several names are being entered into the FPS Report or CONTENTS Report.

4.2.4.5 The HOLD and HOLD-BLANK Commands

The documentor should understand the usefulness of the HOLD and HOLD-BLANK commands. If a diagram, table or figure must be drawn manually, the HOLD-BLANK command should always be used. This will insure enough space is saved on one complete page to insert the figure. If an analyzer report or some text material is going to be produced and the documentor wishes that this material be contained on one page, the HOLD command should be used. An example of where it would be used is with the PICTURE Report. Since the PICTURE Report uses 40 lines, the documentor should place a HOLD 40 preceding the PICTURE command to insure that it is produced on a single page.

4.2.5 Usage of Analyzer SYNONYMS

A SYNONYM is a short abbreviation form of a name that can be stored in the Analyzer data base. A SYNONYM is a reserve word of the Analyzer.

In order to generalize the documentation Source data base, SYNONYMS should be used whenever possible to describe certain items, such as main functions, master files, etc. The implication of this practice on the Analyzer data base formulation is discussed in section 4.3.2.

By using SYNONYMS, the documentor is relieved of changing even more of the Source data base from one project to the next. An example of how this can be done is in Section 3.2 of the Functional Description. The Source data base will look as follows:

```
%3.2 SYSTEM FUNCTIONS
ZNG SO=MAIN-PROCESS SL=1 NOPRINT
    THE VARIOUS SUB-FUNCTIONS OF THE SYSTEM ARE:
    ZPROM DESC PRCR NOPUNCH
```

In this example, MAIN-PROCESS is a SYNONYM.
Section 2.4.2.4 is another example.

#2.4.2.4 OPERATIONAL IMPACTS

A. OPERATIONAL STRUCTURE

ZSTR
*SKIP 2

B. TIMELINESS (OPERATIONS AND DATA)

ZFREQ
*SKIP 2

C. INPUTS

ZSTR INPUT
*SKIP 2

D. DATA RETENTION

1. MASTER FILE

ZP COM N=MASTER-FILE DESC DER NOPUNCH
*NEW-PAGE

ZPIC N=MASTER-FILE NODATA NOFLOW

Here, MASTER-FILE is a SYNONYM.

4.3 Development of the Analyzer Data Base with the Document Generator in Mind

There are two situations that can occur in relation to the Analyzer data base when using the Document Generator.

The Analyzer data base has either been written with the document to be produced in mind, or it has not been. In any case, it is going to be beneficial for the documentor to write or update the Analyzer data base in such a way as to facilitate ease of production of the document using the generator. Various practices can be used by the documentor to accomplish this. These practices are explained in the following sub-sections.

4.3.1 Usage of MEMO

When it is necessary to describe certain aspects of a proposed system that are general in nature but do not apply directly to the structural, functional, or data flow of the system, then the usage of MEMOs in the Analyzer data base can be very helpful. When objectives, background material, requirements, impacts, etc., are needed, a MEMO is the obvious answer. Hence, simple PUNCH-COMMENT-ENTRY is all that is needed in the documentation source. An example of a typical MEMO is shown below. This MEMO relates the objectives required for Section 2.2 of the Functional Description:

* Note: All capitalized names in this section refer to URL Reserve Words. For a more detailed explanation, refer to ISDOS Working Papers No. 68 and No. 98.
DESCRIPTION;
THIS PROJECT HAS BEEN AUTHORIZED TO DEVELOP A PAY SYSTEM
FOR THIS ORGANIZATION. THIS PAY SYSTEM WILL PERFORM PAYROLL
PROCESSING USING EMPLOYEE INFORMATION COMING FROM DEPART-
MENTS AND EMPLOYEES AND WILL PRODUCE OUTPUTS WHICH WILL GO
TO THE DEPARTMENTS AND EMPLOYEES. THE SYSTEM WILL ALSO
MAINTAIN THE PAYROLL MASTER INFORMATION.

4.3.2 Usage of NAMES and SYNONYMS

When the documentor is constructing Analyzer data bases, one should
keep in mind what Analyzer NAMES and SYNONYMS have been used in
previous documentation Source data bases. The documentor's work
can be reduced if common names are used to describe similar sections
of Analyzer data bases which are describing systems that will
have to be documented under the same standard. For instance, a
documentor might want to include sections called BACKGROUND-MEMO
and OBJECTIVE-MEMO when developing Analyzer data bases to be
documented using the Functional Description Standards. The
documentor might also want to have an ATTRIBUTE named DEVELOPMENT-
tIMES in these Data Bases. An examples of ATTRIBUTE usage follows:

DEFINe
AS A ATTRIBUTE;
/* VALUES ARE:
MAN-HOURS-120 FOR
PROGRAMMING,
PLENTY FOR
COMPUTER-TIME,
MAN-HOURS-100 FOR
DATA-BASE-DEVELOPMENT,
MAN-HOURS-20 FOR
PAYROLL-CLERICAL,
MAN-HOURS-30 FOR
OPERATIONS,
*/

SYNONYMS can be used in similar fashion. If the documentor
uses similar synonyms in each data base, his editing can be
reduced. Examples of this are using MASTER-FILE as a SYNONYMS
for the main file of the system and using MAIN-PROCESS as a
synonym for the highest level function in the system.

4.3.3 Completeness

There are three areas in which the documentor should strive to
achieve completeness in the Analyzer data base. These areas
are data flow, system structure, and functional flow.

Data flow should be complete in that all files or data sets are
accounted for. This means that data definition should be
included. Elements of data sets need be defined only to the
level of description required. The CONTENTS Report is a good
means of checking data description completeness. Usage of
UPDATES, DERIVES, GENERATES, RECEIVED, and USES statements is
the vehicle by which data flow information is presented.

362
System structure must be complete for obvious reasons. The STRUCTURE Report is helpful in checking and representing the information.

Functional flow is perhaps the trickiest facet in the development of the Analyzer data base. Care must be taken to insure that a functional chain of EVENTS, CONDITIONS and PROCESSES exists in a meaningful fashion. Usage of TRIGGERS, ON TERMINATION OF, ON INCEPTION OF, TRIGGERS, HAPPENS, and WHEN Statements is suggested.

4.3.4 DESCRIPTIONS

The documents should keep in mind that in many cases standards will need descriptions in the form of text on different aspects of systems. The DESCRIPTION statement can be used by the documentor to meet these text requirements. The documentor should include a DESCRIPTION with all major sections of an Analyzer data base, as well as any other descriptive statements that are deemed necessary (i.e., PROCEDURE, VOLATILITY-SET, etc.)

4.3.5 Usage of KEYWORDS

When one or more objects in a system description need to be identified for selection and analysis purposes, KEYWORDS should be attached to each of the objects. By doing this, the documentor has a link between related objects. For example, if all functions (PROCESSES) described as being manual procedures in a system description were to be listed and analyzed together, the KEYWORD "manual" could be attached to each PROCESS for this purpose. By doing a NAME-GEN with the option KEY=MANUAL, a list of these PROCESSES could then be produced.
APPENDIX A - URA Command Abbreviations

This appendix presents all URA commands, with their parameters and defaults. It also presents the acceptable abbreviations for these commands and parameters. There are several conventions in choosing these abbreviations that may aid the user:

1. For command names that consist of only one word, for example, CONTENTS or PICTURE, the first three or four letters of the name are used as their abbreviation. CONT is the abbreviation for CONTENTS and PIC is the abbreviation for PICTURE.

2. For most command names that consist of more than one word, e.g., CONSISTS-MATRIX or FORMATTED-PROBLEM-STATEMENT, the abbreviation is derived by using the first letter from each word in the command name. This gives CM for CONSISTS-MATRIX and FPS for FORMATTED-PROBLEM-STATEMENT. This convention is not strictly followed however, so that the abbreviations may be more meaningful. For example, DCOM is the abbreviation for DELETE-COMMENT-ENTRY which is more mnemonic than DCE.

3. Abbreviations for parameters used in the same way by several commands are the same. The FILE parameter then, always has an abbreviation, F, allowed no matter which command is using it. Some of the more common parameters are listed below:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>F</td>
</tr>
<tr>
<td>NAME</td>
<td>N</td>
</tr>
<tr>
<td>INPUT</td>
<td>I</td>
</tr>
<tr>
<td>NOPRINT</td>
<td>NP</td>
</tr>
<tr>
<td>PUNCH</td>
<td>P</td>
</tr>
</tbody>
</table>

4. Whenever an abbreviation exists for a parameter that has a "NO" prefix, such as NOSYNONYMS or NOPRINT, the abbreviation is always prefixed with "N." For example, NSYN is the abbreviation for NOSYNONYMS and NP is the abbreviation for NOPRINT.

A blank entry in the "Parameters" column for a command designates that there are no parameters for the command. A blank entry in the "Abbreviations" column designates that there is no abbreviation for the command or parameter name. A blank entry in the "Defaults" column designates that there is no default for this parameter for the given command.
<table>
<thead>
<tr>
<th>Command Name</th>
<th>Parameters</th>
<th>Abbreviations</th>
<th>Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE-TYPE</td>
<td>FILE</td>
<td>CT</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>NAME</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TYPE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>CONSISTS-COMPARISON</td>
<td>FILE</td>
<td>CNC</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>CONSISTS-MATRIX</td>
<td>FILE</td>
<td>CM</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>NAME</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONTAINED</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONSISTS</td>
<td>CONTD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CSTS</td>
<td></td>
</tr>
<tr>
<td>CONTENTS</td>
<td>FILE</td>
<td>CONT</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>NAME</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INDEX</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOINDEX</td>
<td></td>
<td>NOINDEX</td>
</tr>
<tr>
<td></td>
<td>LEVELS</td>
<td></td>
<td>LEVELS=ALL</td>
</tr>
<tr>
<td></td>
<td>NCFLAG</td>
<td></td>
<td>NONCFLAG</td>
</tr>
<tr>
<td></td>
<td>NONCFLAG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA-BASE-STATISTICS</td>
<td>NAMES</td>
<td>DBS</td>
<td>NAMES</td>
</tr>
<tr>
<td></td>
<td>NONAMES</td>
<td></td>
<td>NUBS</td>
</tr>
<tr>
<td></td>
<td>NUBS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NONUBS</td>
<td></td>
<td>NONAMNUBS</td>
</tr>
<tr>
<td></td>
<td>NAMNUBS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NONAMNUBS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNONYMS</td>
<td>SYN</td>
<td>NOSYNONYMS</td>
</tr>
<tr>
<td></td>
<td>NOSYNONYMS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA-PROCESS</td>
<td>FILE</td>
<td>DP</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>NAME</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATA</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PROCESS</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>Command Name</td>
<td>Parameters</td>
<td>Abbreviations</td>
<td>Defaults</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FILE</td>
<td>DEL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAME</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>DELETE-COMMENT-ENTRY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DERIVATION</td>
<td>DCOM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FALSE-WHILE</td>
<td>DER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PROCEDURE</td>
<td>FW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRUE-WHILE</td>
<td>TW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VOLATILITY</td>
<td>VOL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VOLATILITY-MEMBER</td>
<td>VOLM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VOLATILITY-SET</td>
<td>VOLS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FILE</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAME</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRINT</td>
<td></td>
<td>PRINT</td>
</tr>
<tr>
<td></td>
<td>NOPRINT</td>
<td>NP</td>
<td></td>
</tr>
<tr>
<td>DELETE-PSL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>INPUT</td>
<td>DPSL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SOURCE</td>
<td>I</td>
<td>INPUT</td>
</tr>
<tr>
<td></td>
<td>NOSOURCE</td>
<td>S</td>
<td>SOURCE</td>
</tr>
<tr>
<td></td>
<td>XREF</td>
<td>NS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOXREF</td>
<td>X</td>
<td>NOXREF</td>
</tr>
<tr>
<td>DICTIONARY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FILE</td>
<td>DICT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NAME</td>
<td>F</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>INDEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOINDEX</td>
<td></td>
<td>NOINDEX</td>
</tr>
<tr>
<td></td>
<td>NUM-SPACE</td>
<td>NS</td>
<td>NUM-SPACE=2</td>
</tr>
<tr>
<td></td>
<td>DESCRIPTION</td>
<td>DESC</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td></td>
<td>NODescription</td>
<td>NDESC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KEYWORDS</td>
<td>KEY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOKEYWORDS</td>
<td>NKEY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RESPONSIBLE-PD</td>
<td>RPD</td>
<td>RESPONSIBLE-PD</td>
</tr>
<tr>
<td></td>
<td>NONRESPONSIBLE-PD</td>
<td>NRPD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNONYMS</td>
<td>SYN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOSYNONYMS</td>
<td></td>
<td>SYNONYMS</td>
</tr>
<tr>
<td>Command Name</td>
<td>Parameters</td>
<td>Abbreviations</td>
<td>Defaults</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>ENTITY-IDENTIFIER</td>
<td></td>
<td>EI</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>FILE</td>
<td>F</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>NAME</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IDENTIFIER</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENTITY</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>EXTENDED-PICTURE</td>
<td></td>
<td>EP</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>FILE</td>
<td>F</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>NAME</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INDEX</td>
<td></td>
<td>NOINDEX</td>
</tr>
<tr>
<td></td>
<td>NOINDEX</td>
<td></td>
<td>NOINDEX</td>
</tr>
<tr>
<td></td>
<td>STRUCTURE</td>
<td>STR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATA-FLOW</td>
<td>DF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FORWARD</td>
<td>FWD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BACKWARD</td>
<td>BWD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOWNWARD</td>
<td>DOWN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UPWARD</td>
<td>UP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LINKS</td>
<td></td>
<td>LINKS=1000</td>
</tr>
<tr>
<td></td>
<td>COLUMNS</td>
<td>COLS</td>
<td>COLUMNS=119</td>
</tr>
<tr>
<td></td>
<td>ROWS</td>
<td></td>
<td>ROWS=39</td>
</tr>
<tr>
<td></td>
<td>HORIZONTAL-BOXES</td>
<td>HB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VERTICAL-BOXES</td>
<td>VB</td>
<td></td>
</tr>
<tr>
<td>FORMATTED-PROBLEM-STATEMENT</td>
<td></td>
<td>FPS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AMARG</td>
<td>AM</td>
<td>AMARG=10</td>
</tr>
<tr>
<td></td>
<td>BMARG</td>
<td>BM</td>
<td>BMARG=25</td>
</tr>
<tr>
<td></td>
<td>COMMENT</td>
<td>COM</td>
<td>COMMENT</td>
</tr>
<tr>
<td></td>
<td>NOCOMMENT</td>
<td>NCOM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CMARG</td>
<td>CM</td>
<td>CMARG=1</td>
</tr>
<tr>
<td></td>
<td>DEFINE</td>
<td>DEF</td>
<td>DEFINE</td>
</tr>
<tr>
<td></td>
<td>NODEFINE</td>
<td>NDEF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DESG</td>
<td>DG</td>
<td>DESG</td>
</tr>
<tr>
<td></td>
<td>NODESG</td>
<td>NDG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EMPTY</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOEMPTY</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Cont'd)
<table>
<thead>
<tr>
<th>Command Name</th>
<th>Parameters</th>
<th>Abbreviations</th>
<th>Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>F</td>
<td>FILE</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMARG</td>
<td>HM</td>
<td>HMARG=40</td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td></td>
<td>NOINDEX</td>
<td></td>
</tr>
<tr>
<td>NOINDEX</td>
<td>NL</td>
<td>NONEW-LINES</td>
<td></td>
</tr>
<tr>
<td>NEW-LINES</td>
<td>NL</td>
<td>ONE-W-LINE</td>
<td></td>
</tr>
<tr>
<td>NONEW-LINES</td>
<td>NNL</td>
<td>ONE-W-LINE</td>
<td></td>
</tr>
<tr>
<td>NEW-PAGE</td>
<td>NPG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NONEW-PAGE</td>
<td>NNPG</td>
<td>NONEW-PAGE</td>
<td></td>
</tr>
<tr>
<td>NMARG</td>
<td>NM</td>
<td>NMARG=20</td>
<td></td>
</tr>
<tr>
<td>ONE-PER-LINE</td>
<td>OPL</td>
<td>ONE-W-LINE</td>
<td></td>
</tr>
<tr>
<td>SEVERAL-PER-LINE</td>
<td>SPL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRINT</td>
<td></td>
<td>PRINT</td>
<td></td>
</tr>
<tr>
<td>NOPRINT</td>
<td>NP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PUNCH</td>
<td>P</td>
<td>NOPUNCH</td>
<td></td>
</tr>
<tr>
<td>NOPUNCH</td>
<td></td>
<td>RNMARG=70</td>
<td></td>
</tr>
<tr>
<td>RNMARG</td>
<td>RM</td>
<td>SMARG=5</td>
<td></td>
</tr>
<tr>
<td>SMARG</td>
<td>SM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FREQUENCY</td>
<td>FREQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HELP</td>
<td>command-name</td>
<td>SHORT</td>
<td>SHORT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LONG</td>
<td></td>
</tr>
<tr>
<td>INPUT-PSL</td>
<td>IP</td>
<td>DBREF</td>
<td></td>
</tr>
<tr>
<td>DBREF</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NODBREF</td>
<td>ND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INPUT</td>
<td>I</td>
<td>INPUT</td>
<td></td>
</tr>
<tr>
<td>SOURCE</td>
<td>S</td>
<td>SOURCE</td>
<td></td>
</tr>
<tr>
<td>NOSOURCE</td>
<td>NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UPDATE</td>
<td>U</td>
<td>NOUPDATE</td>
<td></td>
</tr>
<tr>
<td>NOUPDATE</td>
<td>NU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XREF</td>
<td>X</td>
<td>NOXREF</td>
<td></td>
</tr>
<tr>
<td>NOXREF</td>
<td>NX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command Name</td>
<td>Parameters</td>
<td>Abbreviations</td>
<td>Defaults</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>KWIC</td>
<td>FILE</td>
<td>F</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>DIF</td>
<td></td>
<td>DIF=20</td>
</tr>
<tr>
<td>NAME-GEN</td>
<td></td>
<td>NG</td>
<td>NOATTRIBUTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATTR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NATTR</td>
<td>NOATTRIBUTE-VALUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATTV</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NATTV</td>
<td>NOATTRIBUTE-VALUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COND</td>
<td>NOCONDITION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NCOND</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ELE</td>
<td>NOELEMENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NELE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENT</td>
<td>NOENTITY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NENT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>EV</td>
<td>NOEVENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEV</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GR</td>
<td>NOGROUP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NGR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INP</td>
<td>NOINPUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NINP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INTF</td>
<td>NOINTERFACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NINTF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INT</td>
<td>NOINTERVAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NINT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>KEY</td>
<td>NOKEYWORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NKEY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BOX</td>
<td>NOMAILBOX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NBOX</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMEMO</td>
<td>NOMEMO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMEMO</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT</td>
<td>NOOUTPUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PD</td>
<td>NOPROBLEM-DEFINER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NPD</td>
<td></td>
</tr>
</tbody>
</table>

(Cont'd)
<table>
<thead>
<tr>
<th>Command Name</th>
<th>Parameters</th>
<th>Abbreviations</th>
<th>Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME-GEN (cont'd)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PROCESS</td>
<td>PROC</td>
<td>NOPROCESS</td>
</tr>
<tr>
<td></td>
<td>NOPROCESS</td>
<td>NPROC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REAL-WORLD-ENTITY*</td>
<td>RWE</td>
<td>NOREAL-WORLD-ENTITY</td>
</tr>
<tr>
<td></td>
<td>NOREAL-WORLD-ENTITY</td>
<td>NRWE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RELATION</td>
<td>RLN</td>
<td>NORELATION</td>
</tr>
<tr>
<td></td>
<td>NORELATION</td>
<td>NRLN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SECURITY</td>
<td>SEC</td>
<td>NOSECURITY</td>
</tr>
<tr>
<td></td>
<td>NOSECURITY</td>
<td>NSEC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SET</td>
<td></td>
<td>NOSET</td>
</tr>
<tr>
<td></td>
<td>NOSET</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SOURCE</td>
<td>SRC</td>
<td>NOSOURCE</td>
</tr>
<tr>
<td></td>
<td>NOSOURCE</td>
<td>NSRC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SUBSETTING-CRITERION</td>
<td>SSCN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOSUBSETTING-CRITERION</td>
<td>NSSCN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNONYMS</td>
<td>SYN</td>
<td>NOSYNONYMS</td>
</tr>
<tr>
<td></td>
<td>NOSYNONYMS</td>
<td>NSYN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYSTEM-PARAMETER</td>
<td>SYSP</td>
<td>NOSYSTEM-PARAMETER</td>
</tr>
<tr>
<td></td>
<td>NOSYSTEM-PARAMETER</td>
<td>NSYSP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UNDEFINED</td>
<td>UNDF</td>
<td>NOUNDEFINED</td>
</tr>
<tr>
<td></td>
<td>NOUNDEFINED</td>
<td>NUNDF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BASIC</td>
<td></td>
<td>BASIC</td>
</tr>
<tr>
<td></td>
<td>NOBASIC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EMPTY</td>
<td></td>
<td>EMPTY</td>
</tr>
<tr>
<td></td>
<td>NOEMPTY</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IDENTIFIER</td>
<td>ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IDENTIFIER-GROUP</td>
<td>IDG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IDENTIFIER-ELEMENT</td>
<td>IDE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NONE</td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td></td>
<td>ORDER=ALPHA</td>
</tr>
<tr>
<td></td>
<td>ORDER</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PD</td>
<td></td>
<td>PRINT</td>
</tr>
<tr>
<td></td>
<td>PRINT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*REAL-WORLD-ENTITY is synonymous with INTERFACE.
<table>
<thead>
<tr>
<th>Command Name</th>
<th>Parameters</th>
<th>Abbreviations</th>
<th>Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOPRINT</td>
<td></td>
<td>NP</td>
<td>PUNCH</td>
</tr>
<tr>
<td>PUNCH</td>
<td></td>
<td>P</td>
<td>PUNCH</td>
</tr>
<tr>
<td>NOPUNCH</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>SL</td>
<td>SUBLEVEL=0</td>
</tr>
<tr>
<td>.SUBLEVEL</td>
<td></td>
<td>SO</td>
<td></td>
</tr>
<tr>
<td>SUBPARTS-OF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAME-LIST</td>
<td>ORDER</td>
<td>NL</td>
<td>ORDER=ALPHA</td>
</tr>
<tr>
<td>PICTURE</td>
<td></td>
<td>PIC</td>
<td>DATA</td>
</tr>
<tr>
<td>DATA</td>
<td></td>
<td>D</td>
<td>DATA</td>
</tr>
<tr>
<td>NODATA</td>
<td></td>
<td>ND</td>
<td></td>
</tr>
<tr>
<td>FILE</td>
<td></td>
<td>F</td>
<td>FILE</td>
</tr>
<tr>
<td>NAME</td>
<td></td>
<td>N</td>
<td>FLOW</td>
</tr>
<tr>
<td>FLOW</td>
<td></td>
<td></td>
<td>NOFLOW</td>
</tr>
<tr>
<td>NOFLOW</td>
<td></td>
<td></td>
<td>NOINDEX</td>
</tr>
<tr>
<td>INDEX</td>
<td></td>
<td></td>
<td>STRUCTURE</td>
</tr>
<tr>
<td>NOINDEX</td>
<td></td>
<td></td>
<td>NOSTRUCTURE</td>
</tr>
<tr>
<td>STRUCTURE</td>
<td></td>
<td>STR</td>
<td>STRUCTURE</td>
</tr>
<tr>
<td>NOSTRUCTURE</td>
<td></td>
<td>NSTR</td>
<td></td>
</tr>
<tr>
<td>PRINT-ATTRIBUTE-VALUES</td>
<td>FILE</td>
<td>PAV</td>
<td>FILE</td>
</tr>
<tr>
<td>.NAME</td>
<td></td>
<td>F</td>
<td>FILE</td>
</tr>
<tr>
<td>PROCESS-CHAIN</td>
<td>FILE</td>
<td>PC</td>
<td>FILE</td>
</tr>
<tr>
<td>.NAME</td>
<td></td>
<td>F</td>
<td>FILE</td>
</tr>
<tr>
<td>INDEX</td>
<td></td>
<td></td>
<td>NOINDEX</td>
</tr>
<tr>
<td>NOINDEX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LINKS</td>
<td></td>
<td></td>
<td>LINKS=1000</td>
</tr>
<tr>
<td>COLUMNS</td>
<td></td>
<td>COLS</td>
<td>COLUMNS=119</td>
</tr>
<tr>
<td>ROWS</td>
<td></td>
<td></td>
<td>ROWS=39</td>
</tr>
<tr>
<td>HORIZONTAL-BOXES</td>
<td></td>
<td>HB</td>
<td></td>
</tr>
<tr>
<td>VERTICAL-BOXES</td>
<td></td>
<td>VB</td>
<td></td>
</tr>
<tr>
<td>Command Name</td>
<td>Parameters</td>
<td>Abbreviations</td>
<td>Defaults</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>PROCESS-INPUT-OUTPUT</td>
<td>FILE</td>
<td>PRIO</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>NAME</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DESCRIPTION</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NODESCRIPTION</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PROCEDURE</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOPROCEDURE</td>
<td>N</td>
<td>NOPROCEDURE</td>
</tr>
<tr>
<td></td>
<td>INPUT</td>
<td>IN</td>
<td>INPUT</td>
</tr>
<tr>
<td></td>
<td>NOINPUT</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OUTPUT</td>
<td>O</td>
<td>OUTPUT</td>
</tr>
<tr>
<td></td>
<td>NOOUTPUT</td>
<td>N</td>
<td>NOOUTPUT</td>
</tr>
<tr>
<td></td>
<td>NEW-PAGE</td>
<td>N</td>
<td>NNEW-PAGE</td>
</tr>
<tr>
<td></td>
<td>PRINT</td>
<td>N</td>
<td>PRINT</td>
</tr>
<tr>
<td>PUNCH-COMMENT-ENTRY</td>
<td>DERIVATION</td>
<td>PCOM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DESCRIPTION</td>
<td>DER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FALSE-WHILE</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PROCEDURE</td>
<td>PCCD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRUE-WHILE</td>
<td>TW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VOLATILITY</td>
<td>VOL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VOLATILITY-MEMBER</td>
<td>VOLS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VOLATILITY-SET</td>
<td>VC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EMPTY</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOEMPTY</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FILE</td>
<td>F</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>NAME</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRINT</td>
<td>NP</td>
<td>PRINT</td>
</tr>
<tr>
<td></td>
<td>NOPRINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PUNCH</td>
<td>P</td>
<td>NOPUNCH</td>
</tr>
<tr>
<td></td>
<td>NOPUNCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command Name</td>
<td>Parameters</td>
<td>Abbreviations</td>
<td>Defaults</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>RENAME</td>
<td>INPUT</td>
<td>REN I</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OLD</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NEW</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>REPLACE-COMMENT-ENTRY</td>
<td>INPUT</td>
<td>RCOM I</td>
<td>INPUT</td>
</tr>
<tr>
<td></td>
<td>PRINT</td>
<td>NPRINT NP</td>
<td>PRINT</td>
</tr>
<tr>
<td>STRUCTURE</td>
<td>INDENT</td>
<td>STR IND</td>
<td>INDENT=3</td>
</tr>
<tr>
<td></td>
<td>INDEX</td>
<td></td>
<td>NOINDEX</td>
</tr>
<tr>
<td></td>
<td>NOINDEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>INPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTERFACE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OUTPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PROCESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REAL-WORLD-ENTITY*</td>
<td>RWE</td>
<td></td>
</tr>
<tr>
<td>SUMMARY</td>
<td>SUM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* REAL-WORLD-ENTITY is synonymous with INTERFACE.
Appendix B. Creating and Initializing URA data bases

Before the user can add information to a URA data base, the data base must exist and be initialized properly. Creating an initialized data base is, in general, a two-step process. Step one is to create the table-file containing a description of the data base. Step two is to create the data base itself. Several data bases may be created using the same description, hence, step two may be done several times for a single table file created in step one.

Step one - Create Table File

The table file contains a description of the URA data base which depends on the DDL (Data Definition Language) description of the data base. The important aspect of this description from the user point of view is the number of pages in the data base. The first statement in the DDL is the number of pages in the data base. To create a table file:

```plaintext
eexec_com >udd>project>personid>ddla name
```

where the segment name.ddl contains the ddl and the segment name.dbt will contain the tables. For instance, if the user wished to set up a table file for a fifty page data base, he would make a copy of the DDL file

```plaintext
copy >udd>project>personid>psa.ddl user.ddl
```

He would then edit the new DDL file, user.ddl, to change the first line to read:

```plaintext
npages 50
```

Note that the number of pages must be right justified in columns 20-24.

Once the user has changed the DDL, he may use ddda to create a table file:

```plaintext
eexec_com >udd>project>personid>ddla user
```

This would create a segment named user.dbt.

Step two - Data Base Initialization

The user may initialize an empty data base segment using dbin.

```plaintext
eexec_com >udd>project>personid>dbin data-base table-file
```

The data base has a qualifier of ".dbf" added, and the table file has a qualifier of ".dbt" added.
For the above example,
exec_com >udd>project>personid>dbin mydb user
would initialize a fifty page data base segment named mydb.dbf.
Appendix C. Dump/Restore facility

1. Dump.

To produce a card image dump of a data base:

exec_com >udd>project>personid>pdum data-base output punch

where:

data-base is the data base to be dumped (the ".dbf" qualifier is added)
output is a printed listing of the dump
punch is the card image dump segment

2. Restore

To restore a previously dumped data base:

exec_com >udd>project>personid>pres data-base output input

where:

data-base is an empty initialized data base (the ".dbf" qualifier is added)
output is a printed listing of the input
input is the punch segment from pdum
Appendix D. Specification Generator

To execute the specification generator:

    exec_com >udd>project>personid>spg schema source

where:

    schema is the segment containing the schema
    source is the segment containing the source
APPENDIX E

Using the URA Command Language with Multics*
(Version A2.1 of URA)

Contents

There are three parts to this addendum:

1) URA Control Commands for Multics
   set Command ..............................................
   stop Command ............................................

   These commands can be used when using URA under Multics (in addition to the other commands presented in Part IV).

2) Notes on Executing URA under Multics ..................

   This part explains how to create and initialize a URA data base in Multics and how to enter URA mode once in Multics. Example Multics runs are also given to illustrate the manner in which URA interacts with Multics.

3) Default Segment Names .................................

   This part presents the default segment names used by the URA commands and referenced in Part IV.

   A brief "Multics Glossary" is included to define some of the terminology used in this addendum.

* For a better understanding of Multics, see the Multics Programmers' Manual (MPM) or Multics Users' Guide, Honeywell Order No. AL40.

* For a more detailed explanation of how URA interacts with Multics, see Part II.
Special Note Concerning the Use of URA on Multics:

All commands (and their abbreviations) described in Part IV must be issued in lower case. Multics does not recognize the upper case versions of these commands.
Command: set Type: control command

Purpose: To set various global switches and parameters.

Prototype: set [parameter]...

Parameters:
- `data-base(db)=segment name` Default: `data-base=psa.dbf`
  
  The data base is the segment in which the data base information is assembled and stored. This facility allows the user to change data bases conveniently and quickly. If the "set data-base" command is not given, the data set "psadb.dbf" on the user's working storage is used.

- `dbt=segment name` Default: `dbt*=udd>project>personid>psa.dbt`
  
  The data tables used in accessing the user's URA data base may be specified if other than the normal default values. When this parameter is not used, the segment >udd>project>personid>psa.dbt is used.

- `echo(e)= (on | off)` Defaults: `echo=off`
  
  With "echo" set equal to on, the commands are printed on the current output device (term) as they are encountered. This is more desirable in batch (command is printed on line printer) than in on-line mode (you always see the command you type in).

- `heading(h)= (on | off)` Default: `heading=on`
  
  With "heading" set off, printing of headings (date, title, page number, etc.) of each URA report will be suppressed. Headings will be printed when the switch is set on.

- `lines(l)=integer` Default: `lines=45`
  
  The number of lines printed by URA per page is set to the indicated number. The default number fits the output to an 8-1/2 x 11 inch page for convenient binding. "Lines" may take on any value between 10 and 500.

- `mode= {batch(b) | terminal(t)}` Default: `mode=terminal`
  
  This switch sets other switches particular to the mode of operation such as "prompt" and "echo."

- `output(o)=segment name` Default: `output=term`
  
  This parameter specifies the data set where all output resulting from any subsequent URA commands should be written into. If no output file is specified, all output is written at the terminal (term) (Note: The output is appended to the end of the segment specified.)
parameters(p)= \{\text{on} \} \quad \text{Default: parameters=on}\]

With "parameters" set off, printing of parameters (in effect in the production of each URA report generated) will be suppressed. Parameters will be printed when the switch is set on.

problem-name(pnam)=user-name \quad \text{Default: no default}\]

The "problem-name" is the title that goes on each page of the output. It must be a URL name, that is, it must begin with a letter, be no more than 30 characters in length, and be composed of alphanumeric characters with no intervening blanks.

prompt(p)= \{\text{on} \} \quad \text{Defaults: prompt=on}\]

If "prompt" is set to on, URA will prompt the user for the correct command or parameters when an error is encountered. With "prompt" set to OFF, URA will ignore invalid commands and parameters and proceed to the next command or parameter.

Examples: set db=psa.dbf pnam=ura-example
set output=term
Command: stop  

Type: control command

Purpose: To terminate execution of the URA software and return control to Multics.

Prototype: stop

Parameters: None

Example: stop

* Issuing this command returns all storage used by URA for tables, scratch files, etc. To restart execution of URA after issuing this command the user must give:

   exec-com >udd>project>personid>ura

If a user leaves URA mode and enters Multics without use of the "stop" command, the user should issue the Multics "start" command to return to URA.
2. Notes on Executing URA under Multics

To create and initialize a URA data base:

```
exec-com >udd>project>personid>dbin databasename databasetables
```

where "databasename" is a segment name defined by the user and "databasetables" is an existing segment containing information to initialize the data base. In most instances the segment >udd>project>personid>psa can be used as the data base tables. If any other name is used, it must also be specified by the "dbt" parameter for the URA "set" command.

To run the Analyzer (URA) and have the ability to use the URA command language, the Multics command:

```
exec-com >udd>project>personid>ura
```

must be given. URA will prompt the user for a URA command by the message:

Enter command (and any parameters)

To specify the data base and data base tables to be accessed by subsequent URA commands use the "set" command:

```
set data-base=databasename dbt=databasetables
```

where the two names given were those used in the creation and initialization procedure above. When the segment name is psadb.dbf and the data base tables are contained in >udd>project>personid>psa.dbt then the assignments are not required.
Example URA sessions

login Userid
ec >udd>project>personid>dbin mydb
>udd>project>personid>ura
ec >udd>project>personid>ura
set db=mydb.dbf o=temp.print
input-psl input=afile.psl update
name-gen all
fps
stop
dprint temp.print
logout

[login to Multics]
[initialize an empty data base]
[enter URA mode]
[set control information]
[update URA data base using contents of afile.psl in current working directory]
[get list of all names]
[get FPS for all names]
[leave URA mode]
[print output = line printer]
[logout of Multics]
A subsequent session may enter more input and generate some reports.

login Userid
ed >udd>project>personid>ura
set db=mydb.dbf pnam=Multics-example
input-psl input=somefile.psl update
name-gen process
picture
name-gen all
fps
stop
logout

[login to Multics]
[enter URA mode]
[set control information]
[update the data base using the contents of somefile.psl]
[get list of process names]
[get Picture report for each process name]
[get list of all names]
[get FPS for all names]
[leave PSA mode]
[logout of Multics]
3. Default Segment Names

Table E-1 presents a summary of Multics default segment names used in conjunction with each URA command. The "Default Input File" is the default segment used as input to the command if no other segment set has been specified (i.e., via, the "INPUT=" or "FILE=" parameter.) A dash in the entry designates that the command has no Input data parameters.

The "Default PUNCH File" is the default segment used to store PUNCH data from the command if no PUNCH segment is designated via the "PUNCH=" parameter. A dash in the entry designates that no PUNCH data can be obtained directly from the command. (PUNCH data is data that is directly acceptable as input to a URA command.)
## Multics Default Segment Names

<table>
<thead>
<tr>
<th>Command</th>
<th>Default Input File</th>
<th>Default PUNCH File</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE-TYPE</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>CONSISTS-COMPARISON</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>CONSISTS-MATRIX</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>CONTENTS</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>DATA-PROCESS</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>DELETE-COMMENT-ENTRY</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>DELETE-PSL</td>
<td>term</td>
<td></td>
</tr>
<tr>
<td>DICTIONARY</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>ENTITY-IDENTIFIER</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>FORMATTED-PROBLEM-STATEMENT</td>
<td>psaname.uratemp</td>
<td>psafps.psl</td>
</tr>
<tr>
<td>FREQUENCY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INPUT-PSL</td>
<td>term</td>
<td></td>
</tr>
<tr>
<td>KWIC</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>NAME-GEN</td>
<td></td>
<td>psaname.uratemp</td>
</tr>
<tr>
<td>NAME-LIST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PICTURE</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>PRINT-ATTRIBUTE-VALUES</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>PROCESS-INPUT-OUTPUT</td>
<td>psaname.uratemp</td>
<td></td>
</tr>
<tr>
<td>PUNCH-COMMENT-ENTRY</td>
<td>psaname.uratemp</td>
<td>psapcom.comment</td>
</tr>
<tr>
<td>RENAME</td>
<td>no default</td>
<td></td>
</tr>
<tr>
<td>REPLACE-COMMENT-ENTRY</td>
<td>psapcom.comment</td>
<td></td>
</tr>
<tr>
<td>STRUCTURE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE E-1**
APPENDIX F

Schema for the Functional Description ........
Source for the Functional Description ........
Schema for the Data Requirements ............
Source for the Data Requirements ............
APPENDIX F

Schema for the Functional Description

-SCHEMA-

1 *TITLE FUNCTIONAL DESCRIPTION
2 *HEADING-MARGIN 3
3 *LISTS 50
4 #1. FUNCTIONAL DESCRIPTION - GENERAL
5 #4. THIS IS AN EXAMPLE DOCUMENTATION SCHEMA OF THE FUNCTIONAL
6 #3 DESCRIPTION FOUND IN JOB MANUAL 4120.17-1.
7 #1.1 PURPOSE OF FUNCTIONAL DESCRIPTION
8 #1.2 PROJECT REFERENCES
9 #2. SYSTEM SUMMARY
10 #2.1 BACKGROUNDS/PURPOSES
11 #2.2 OBJECTIVES
12 #2.3 EXISTING METHODS AND PROCEDURES
13 #2.4 PROPOSED METHODS AND PROCEDURES
14 #2.4.1 SUMMARY OF IMPROVEMENTS
15 #2.4.2 SUMMARY OF IMPACTS
16 #2.4.2.1 EQUIPMENT IMPACTS
17 #2.4.2.2 SOFTWARE IMPACTS
18 #2.4.2.3 ORGANIZATIONAL IMPACTS
19 #2.4.2.4 OPERATIONAL IMPACTS
20 #2.4.2.5 DEVELOPMENT IMPACTS
21 #2.4.5 EXPECTED LIMITATIONS
22 #3. DETAILED CHARACTERISTICS
23 #3.1 SPECIFIC PERFORMANCE REQUIREMENTS
24 #3.1.1 ACCURACY AND VALIDITY
25 #3.1.2 TIMING
26 #3.2 SYSTEM FUNCTIONS
27 #3.3 INPUTS/OUTPUTS
28 #3.4 DATA CHARACTERISTICS
29 #3.5 FAILURE CONTINGENCIES
30 #4. ENVIRONMENT
31 #4.1 EQUIPMENT ENVIRONMENT
32 #4.2 SUPPORT SOFTWARE ENVIRONMENT
33 #4.3 INTERFACES
34 #4.4 SECURITY
35 #4.5 COST FACTORS
36 #6. DEVELOPMENT PLAN

389
#1. FUNCTIONAL DESCRIPTION

*SET DB=SE11:NAVY.DBF

1.1 PURPOSE OF FUNCTIONAL DESCRIPTION

**THIS FUNCTIONAL DESCRIPTION FOR THE PAYSTATEMENT EXAMPLE, PROJECT #1 IS WRITTEN TO PROVIDE:
   A. THE SYSTEM REQUIREMENTS TO BE SATISFIED WHICH WILL SERVE AS A BASIS FOR MUTUAL UNDERSTANDING BETWEEN THE USER AND THE DEVELOPER,
   B. INFORMATION ON PERFORMANCE REQUIREMENTS, PRELIMINARY DESIGN, AND USER IMPACTS, INCLUDING FIXED AND CONTINUING COSTS,
   C. A BASIS FOR THE DEVELOPMENT OF SYSTEM TESTS.

1.2 PROJECT REFERENCES

A. A COPY OF THE PROJECT REQUEST IS IN THE APPENDIX.

B. STANDARDS AND REFERENCE DOCUMENTATION

1. ISDOS WORKING PAPERS 74, 86, 90
2. DOD ADS DOCUMENTATION STANDARDS MANUAL 4120.17-M

#2. SYSTEM SUMMARY

THIS SECTION SHALL PROVIDE A GENERAL DESCRIPTION, WRITTEN IN NON-ADP TERMINOLOGY, OF THE PROPOSED ADS.

2.1 BACKGROUND/PURPOSES

2.2 OBJECTIVES

2.3 EXISTING METHODS AND PROCEDURES

2.4 PROPOSED METHODS AND PROCEDURES

SINCE THIS IS A NEW COMPANY, THERE IS NO APPLICABLE EXISTING PROCEDURE FOR DOING THE PAYROLL.
61 \%FPS N=ONE
62 \#2.4.2 SUMMARY OF IMPACTS
63 *SKIP 1
64 \#2.4.2.1 EQUIPMENT IMPACTS
65 *SKIP 1
66 A LINE PRINTER CAPABLE OF PRINTING CHECKS IS NECESSARY.
67 EQUIPMENT CAPABILITIES ARE DISCUSSED IN PARAGRAPH 4.1.
68 \#2.4.2.2 SOFTWARE IMPACTS
69 *SKIP 1
70 THERE ARE FIVE BASIC SOFTWARE AREAS WITHIN THE PAYROLL SYSTEM.
71 *SKIP 1
72 \%NG SO=MAIN-PROCESS SL=1
73 *HOLD 43
74 \%PIC N=MAIN-PROCESS NODATA NOPLOW
75 \#2.4.2.3 ORGANIZATIONAL IMPACTS
76 *SKIP 1
77 \%FPS N=RESPONSIBILITIES
78 *HEADING-SKIP 50
79 \#2.4.2.4 OPERATIONAL IMPACTS
80 *HEADING-SKIP 2
81 *REPORT-CC OFF
82 *SKIP 1
83 A. OPERATIONAL STRUCTURE
84 \%STR
85 *NEW-PAGE
86 B. TIMELINESS (OPERATIONS AND DATA)
87 *SKIP 1
88 \%FREQ
89 *NEW-PAGE
90 C. INPUTS
91 \%STR INPUT
92 *SKIP 2
93 D. DATA RETENTION
94 *SKIP 1
95 \%COM N=MASTER-FILE DESC DER NOPUNCH
96 *HOLD 43
97 \%PIC N=MASTER-FILE NODATA NOPLOW
98 *HEADING-SKIP 59
99 \#2.4.2.5 DEVELOPMENT IMPACTS
100 *REPORT-CC ON
101 *SKIP 1
102 \%FPS N=COMPLEXITY-LEVEL
103 *SKIP 2
104 \%FPS N=DEVELOPMENT-NEEDS
105 *HEADING-SKIP 3
106 \#2.5 EXPECTED LIMITATIONS
107 *SKIP 1
108 ERRORS WILL BE NECESSARY FOR BAD DATA INPUT
109 *SKIP 1
110 \%COM N=ERROR-LISTING DESC NOPUNCH
111 *HOLD 43
112 \%PIC N=ERROR-LISTING
113 *HOLD 43
114 \%PIC N=ERROR-LISTING-ENTRY
115 *HEADING-SKIP 59
116 \#3. DETAILED CHARACTERISTICS
117 *HEADING-SKIP 3
118 *SKIP 1
119 \#3.1 SPECIFIC PERFORMANCE REQUIREMENTS
120 *SKIP 1
3.1.1 ACCURACY AND VALIDITY

3.1.2 TIMING

The following is a description of the functional flow.

For salaried employees -

For hourly employees -

The various sub-functions of the system are:

### System Functions

*PCOM DESC PROC NOPUNCH

### Inputs/Outputs

*SKIP 1

**INPUTS:**

*SKIP 1

**OUTPUTS:**

*SKIP 1
*SKIP 2
*STR OUTPUT
*3.4 DATA CHARACTERISTICS
*REPORT-CC ON
*SKIP 1
*FPS N=PAYROLL-MASTER-INFORMATION
*SKIP 2
*THE FOLLOWING LINES ARE EXCLUSIVE TO THIS DOCUMENT
%PCOM N=DEPARTMENT-FILE DESC DER VOLM NOPUNCH
*SKIP 1
%PCOM N=HOURLY-EMPLOYEE-FILE DESC DER VOLM NOPUNCH
*HOLD 43
*PIC N=HOURLY-EMPLOYEE-FILE
*NEW-PAGE
%PCOM N=SALARIED-EMPLOYEE-FILE DESC DER VOLM NOPUNCH
*HOLD 43
*PIC N=SALARIED-EMPLOYEE-FILE
#3.5 FAILURE CHARACTERISTICS
*SKIP 1
%PCOM N=BACKUP-MEMO DESC NOPUNCH
*HEADING-SKIP 61
#4. ENVIRONMENT
*HEADING-SKIP 3
*SKIP 1
#4.1 EQUIPMENT ENVIRONMENT
*SKIP 1
%PCOM N=EQUIPMENT-MEMO DESC NOPUNCH
#4.2 SUPPORT SOFTWARE ENVIRONMENT
*SKIP 2
%PCOM N=SOFTWARE-MEMO DESC NOPUNCH
#4.3 INTERFACES
*SKIP 1
%STR RWE
*HOLD 43
%PIC N=ACCOUNTING-SYSTEMS
*NEW-PAGE
%PIC N=PAYROLL-DEPARTMENT
#4.4 SECURITY
*REPORT-CC OFF
*SKIP 1
%PCOM N=SECURITY-MEMO DESC NOPUNCH
%NG SEC NOPRINT
%FPS
*HEADING-SKIP 61
#5. COST FACTORS
%PCOM N=COSTS-MEMO DESC NOPUNCH
#6. DEVELOPMENT PLAN
*SKIP 1
%PCOM N=DEVELOPMENT-MEMO DESC NOPUNCH
*SKIP 2
%FPS N=DEVELOPMENT-TIMES
1 *TITLE DATA REQUIREMENTS DOCUMENT
2 *HEADING-MARGIN 3
3 *LINES 50
4 #1. GENERAL DATA REQUIREMENTS
5 & THIS IS AN EXAMPLE DOCUMENTATION SCHEM OF THE DATA
6 & REQUIREMENTS DOCUMENT FOUND IN DOD MANUAL 4120.17-M
7 #1.1 PURPOSE OF DATA REQUIREMENTS
8 #1.2 PROJECT REFERENCES
9 #1.3 MODIFICATION OF DATA REQUIREMENTS
10 #2. DATA DESCRIPTION
11 #2.1 LOGICAL ORGANIZATION OF STATIC SYSTEM DATA
12 #2.2 LOGICAL ORGANIZATION OF DYNAMIC INPUT DATA
13 #2.3 LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA
14 #2.4 INTERNALLY GENERATED DATA
15 #2.5 SYSTEM DATA CONSTRAINTS
16 #3. USER SUPPORT FOR DATA COLLECTION
17 #3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
18 #3.2 RECOMMEND SOURCE OF INPUT DATA
19 #3.3 DATA COLLECTION AND TRANSFER PROCEDURES
20 #3.4 DATA BASE IMPACTS
# 1. GENERAL DATA REQUIREMENTS

This is an example documentation source of the data requirements document found in DOD Manual 4120.17-M.

# 1.1 PURPOSE OF DATA REQUIREMENTS

*SKIP 1

*SOURCE-MARGIN 1

The objectives of this data requirements document for the paystatement example, project #1 are to list and define data elements which the system must handle and communicate data collection requirements to the user.

# 1.2 PROJECT REFERENCES

*SKIP 1

*SOURCE-MARGIN 1

#2. DATA DESCRIPTION

*SKIP 1

*SOURCE-MARGIN 5

*HEADING-Skip 2

The data described in this section shall be separated into two categories, static data and dynamic data. Static data is defined as that data which is used mainly for reference during system operation and is usually generated or updated in widely separated frames independent of normal system runs. Dynamic data includes all data which is intended to be updated and which is input to a system during a normal run (including "real time" data such as targeting data) or is output by the system. Static data as described above is frequently referred to as parametric data and dynamic data as non-parametric data. Both, however, are composed of data elements. The data element names listed in paragraphs 2.1, 2.2, and 2.3 shall be those contained in standard data element libraries, whenever applicable.

# 2.1 LOGICAL ORGANIZATION OF STATIC SYSTEM DATA

*SKIP 1

%CONTENTS N=MASTER-FILE

*SKIP 1

%CONTENTS N=DEPT-FILE

*SKIP 1

%CONTENTS N=H-EMP-FILE

*SKIP 1

%CONTENTS N=S-EMP-FILE

# 2.2 LOGICAL ORGANIZATION OF DYNAMIC INPUT DATA

*SKIP 1

*REPORT-CC OFF

%N: SD=EMP-INFO SL=1 NOPRINT

*CONTENTS

# 2.3 LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA

*REPORT-CC ON

*CONTENTS N=PAY-STATEMENT

*SKIP 1

*CONTENTS N=HOURLY-EMPLOYEER-REPORT

*SKIP 1

*CONTENTS N=SALARIED-EMPLOYEE-REPORT

*SKIP 1
%CONTENTS N=HIRED-EMPLOYEE-REPORT
*SKIP 1
%CONTENTS N=TERMINATED-EMPLOYEE-REPORT
*SKIP 1
%CONTENTS N=ERROR-LISTING
%2.4 INTERNALLY GENERATED DATA
*SKIP 1
%CONTENTS N=ERROR-LISTING
%2.5 SYSTEM DATA CONSTRAINTS
*SKIP 1
%CONTENTS N=ERROR-LISTING
%2.6 USER SUPPORT FOR DATA COLLECTION
*HEADING-SKIP 61
%3. USER SUPPORT FOR DATA COLLECTION
*HEADING-SKIP 2
%31 INPUT OUTPUT ORDER=3YTYPE NOPRINT
%CNC
*NEW-PAGE
%3. USER SUPPORT FOR DATA COLLECTION
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
*SKIP 1
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
INFORMATION NEEDED IN ORDER TO ESTABLISH THE VALUES
OF EACH DATA ELEMENT:
*SKIP 2
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=SALARIED-EMPLOYMENT-FORM
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=HOURLY-EMPLOYMENT-FORM
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=PAY-ROLL-DEPARTMENT
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=EMPLOYMENT-TERMINATION-FORM
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
*SKIP 2
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
INFORMATION TO BE COLLECTED BY THE USER:
*SKIP 1
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=PARTICIPANT DESC NOPUNCH
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=EMPLOYMENT-TERMINATION-FORM
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=PAY-ROLL-DEPARTMENT
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=EMPLOYMENT-TERMINATION-FORM
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
*SKIP 2
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
B. RECIPIENTS
*SKIP 1
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=EMPLOYMENT-TERMINATION-FORM
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=EMPLOYMENT-TERMINATION-FORM
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=EMPLOYMENT-TERMINATION-FORM
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
*SKIP 2
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
D. CRITICAL VALUES
*SKIP 1
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=ACCOUNTING-SYSTEMS
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=ACCOUNTING-SYSTEMS
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%FPS N=ACCOUNTING-SYSTEMS
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
*SKIP 2
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
E. OUTPUT FORM/DEVICE
*SKIP 1
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%PCOM N=1-EMP-REPORT DESC NOPUNCH
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%PCOM N=1-EMP-REPORT DESC NOPUNCH
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%PCOM N=1-EMP-REPORT DESC NOPUNCH
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
*SKIP 2
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
F. FREQUENCY OF UPDATE
*SKIP 1
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%PCOM N=2-EMP-REPORT DESC NOPUNCH
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%PCOM N=2-EMP-REPORT DESC NOPUNCH
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%PCOM N=2-EMP-REPORT DESC NOPUNCH
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
*SKIP 2
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
G. RECOMMENDED SOURCE OF INPUT DATA
*SKIP 1
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%PCOM N=3-EMP-REPORT DESC NOPUNCH
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%PCOM N=3-EMP-REPORT DESC NOPUNCH
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%PCOM N=3-EMP-REPORT DESC NOPUNCH
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
*SKIP 2
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
%3.1 DATA COLLECTION REQUIREMENTS AND SCOPE
SOURCE
396
07-24-75 16:13:04 SOURCE
#3.3 DATA COLLECTION AND TRANSFER PROCEDURES

This topic has been discussed above (Section 3.1.A)

#3.4 DATA BASE IMPACTS

This topic has been discussed above (Section 3.1.F)

*PCOM N=DEPARTMENT-FILE DESC DER VOLS NOPUNCH

*PCOM N=HOURLY-EMPLOYEE-FILE DESC DER VOLS NOPUNCH

*PCOM N=SALARIED-EMPLOYEE-FILE DESC DER VOLS NOPUNCH
APPENDIX G

Examples of the Automated Documentation System

This appendix contains examples of the Functional Description and Data Requirements Document written using the Automated Documentation System. Included are the schemas and source data bases used to produce these documents. The Analyzer data base used is the Pay System example described in Part IV.* Due to space limitations, listings of the Analyzer data base are not included.

The following list of Analyzer commands have been used to produce these examples:

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Parameters Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTENTS</td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>NAME</td>
</tr>
<tr>
<td>DATA PROCESS</td>
<td>DATA</td>
</tr>
<tr>
<td></td>
<td>FILE</td>
</tr>
<tr>
<td>FPS</td>
<td>NAME</td>
</tr>
<tr>
<td></td>
<td>FILE</td>
</tr>
<tr>
<td></td>
<td>PRINT</td>
</tr>
<tr>
<td>FREQUENCY</td>
<td>ENTITY</td>
</tr>
<tr>
<td>NAME-GEN</td>
<td>INPUT</td>
</tr>
<tr>
<td></td>
<td>OUTPUT</td>
</tr>
<tr>
<td></td>
<td>PRINT</td>
</tr>
<tr>
<td></td>
<td>NOPRINT</td>
</tr>
<tr>
<td></td>
<td>SUBLEVEL</td>
</tr>
<tr>
<td></td>
<td>SUBPART OF</td>
</tr>
<tr>
<td></td>
<td>SECURITY</td>
</tr>
</tbody>
</table>

* User Requirements Analyzer Command Description.
<table>
<thead>
<tr>
<th>Command Name</th>
<th>Parameters Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUNCH-COMMENT-ENTRY</td>
<td>DER, DESC, PROC, VOLM, VOLS, FILE, NAME, NOPUNCH, PRINT</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>STRUCTURE</td>
<td>INPUT, PROCESS, RWE, OUTPUT</td>
</tr>
</tbody>
</table>

Although these are the only commands used, all Analyzer commands may be used with the Automated Documentation Generator (with the exception of the STOP command).
1. FUNCTIONAL DESCRIPTION - GENERAL

1.1 PURPOSE OF FUNCTIONAL DESCRIPTION

This functional description for the paystatement example, Project #1 is written to provide:
A. The system requirements to be satisfied which will serve as a basis for mutual understanding between the user and the developer.
B. Information on performance requirements, preliminary design, and user impacts, including fixed and continuing costs.
C. A basis for the development of system tests.

1.2 PROJECT REFERENCES

The payroll department is the sponsor and user of this management information system. The actual operation of this system will be handled by both the payroll department and the data processing department.

A. A copy of the project request is in the appendix.
B. Standards and reference documentation
   1. ISDOS working papers 74 and 86, and Part I.
   2. DOD ADS documentation standards manual 4120.17-M
2. SYSTEMS

This section shall provide a general description, written in non-ADP terminology, of the proposed ADS.

2.1 BACKGROUND/PURPOSES

1. Background-Memo
   Description;
   This system will work in conjunction with various other accounting systems to form a functional information network within the company;

2.2 OBJECTIVES

2. Objectives-Memo
   Description;
   This project has been authorized to develop a pay system for this organization. This pay system will perform payroll processing using employee information coming from departments and employees and will produce outputs which will go to the departments and employees. The system will also maintain the payroll master information.

2.3 EXISTING METHODS AND PROCEDURES

Since this is a new company, there is no applicable existing procedure for doing the payroll.

2.4 PROPOSED METHODS AND PROCEDURES

1. Process-Memo
   Description;
THE PAYROLL PROCESS WILL BE DIVIDED IN TWO SECTIONS —
HOURLY AND SALARIED. CERTAIN PROCESSES THAT CAN BE
USED BY BOTH SECTIONS WILL BE IN A SHARED PROCESS SECTION.

****
ONLY LOWEST LEVEL PROCESSES, I.E. TERMINAL PROCESSES,
DO ACTUAL MANIPULATION OF DATA ELEMENTS. ALL SUCH
PROCESSES WILL HAVE THE KEYWORD TERMINAL;
2.4.1 SUMMARY OF IMPROVEMENTS
A. STAFFING

1* PAYROLL-DEPT-EMPLOYEES
DESCRIPTION:
1       HERE IS A LIST OF THE NUMBER OF EMPLOYEES NEEDED IN THE
2       PAYROLL DEPARTMENT:
3       TWO ACCOUNTANTS
4       TWO SECRETARIES
5       THREE PAYROLL-CLERKS
6       TWO MANAGERS;

B. TIMELINESS

1 DEFINE
2     AS A SYSTEM-PARAMETER;
3 /* LEFT CONNECTIVITY OF:
4     DEPT-HOURLY-EMP-RELATION,
5     DEPT-SALARIED-EMP-RELATION */
6 /* HIRED-EMPLOYEE-REPORT HAPPENS
7     ONE TIMES-PER WEEK */
8 /* TERMINATION-PROCESSING-INIT HAPPENS
9     ONE TIMES-PER WEEK */
10 /* TERMINATING-EMP-PROCESSING HAPPENS
11     ONE TIMES-PER WEEK */
12 /* TERMINATED-EMPLOYEE-REPORT HAPPENS
13     ONE TIMES-PER WEEK */
14 /* SALARIED-PAYCHECK-PROD-INIT HAPPENS
15     ONE TIMES-PER MONTH */
16 /* SALARIED-EMPLOYEE-REPORT HAPPENS
17     ONE TIMES-PER MONTH */
18 /* SALARIED-EMPLOYEE-PROCESSING HAPPENS
19     ONE TIMES-PER MONTH */
20 /* SALARIED-EMP-PROCESSING-INIT HAPPENS
21     ONE TIMES-PER MONTH */
22 /* NEW-EMPLOYEE-PROCESSING-INIT HAPPENS
23     ONE TIMES-PER WEEK */
24 /* NEW-EMPLOYEE-PROCESSING HAPPENS
25     ONE TIMES-PER WEEK */
26 /* HOURLY-PAYCHECK-PROD-INIT HAPPENS
27     ONE TIMES-PER WEEK */
28 /* HOURLY-EMPLOYEE-REPORT HAPPENS
29     ONE TIMES-PER WEEK */
30 /* HOURLY-EMPLOYEE-PROCESSING HAPPENS

SUMMARY OF IMPROVEMENTS
2.4.2 SUMMARY OF IMPACTS

2.4.2.1 EQUIPMENT IMPACTS

A LINE PRINTER CAPABLE OF PRINTING CHECKS IS NECESSARY. EQUIPMENT CAPABILITIES ARE DISCUSSED IN PARAGRAPH 4.1.

2.4.2.2 SOFTWARE IMPACTS

THERE ARE FIVE BASIC SOFTWARE AREAS WITHIN THE PAYROLL SYSTEM.

1  HOURLY-EMPLOYEE-PROCESSING  PROCESS
2  NEW-EMPLOYEE-PROCESSING  PROCESS
3  SALARIED-EMPLOYEE-PROCESSING  PROCESS
4  SHARED-ROUTINES  PROCESS
5  TERMINATING-EMP-PROCESSING  PROCESS
2.4.2.3 ORGANIZATIONAL IMPACTS
DESCRIPTION:

The Payroll Dept. Manager will be responsible for the development of the computer software and maintenance of the system while the other manager will be in charge of all non-computer personnel. All software development is the responsibility of the EDI Department.

EOF EOF EOF EOF EOF

ORGANIZATIONAL IMPACTS

OPERATIONAL IMPACTS
## A. Operational Structure

<table>
<thead>
<tr>
<th>Count Level Name</th>
<th>Action/Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1</td>
<td>Payroll-Processing</td>
</tr>
<tr>
<td>2 2</td>
<td>New-Employee-Processing</td>
</tr>
<tr>
<td>3 3</td>
<td>Salaried-Record-Creation</td>
</tr>
<tr>
<td>4 3</td>
<td>Hourly-Record-Creation</td>
</tr>
<tr>
<td>5 3</td>
<td>Hire-Report-Entry-Generation</td>
</tr>
<tr>
<td>6 3</td>
<td>Department-File-Addition</td>
</tr>
<tr>
<td>7 2</td>
<td>Terminating-Emp-Processing</td>
</tr>
<tr>
<td>8 3</td>
<td>Salaried-Record-Deletion</td>
</tr>
<tr>
<td>9 3</td>
<td>Hourly-Record-Deletion</td>
</tr>
<tr>
<td>10 3</td>
<td>Terms-Report-Entry-Generation</td>
</tr>
<tr>
<td>11 3</td>
<td>Department-File-Removal</td>
</tr>
<tr>
<td>12 2</td>
<td>Hourly-Employee-Processing</td>
</tr>
<tr>
<td>13 3</td>
<td>Hourly-Paycheck-Validation</td>
</tr>
<tr>
<td>14 4</td>
<td>Time-Card-Validation</td>
</tr>
<tr>
<td>15 3</td>
<td>Hourly-Emp-Update</td>
</tr>
<tr>
<td>16 4</td>
<td>Hours-Update</td>
</tr>
<tr>
<td>17 3</td>
<td>H-Report-Entry-Generation</td>
</tr>
<tr>
<td>18 3</td>
<td>Hourly-Paycheck-Production</td>
</tr>
<tr>
<td>19 4</td>
<td>H-Gross-Pay-Computation</td>
</tr>
<tr>
<td>20 4</td>
<td>Total-Hours-Computation</td>
</tr>
<tr>
<td>21 3</td>
<td>Hourly-Paycheck-Computation</td>
</tr>
<tr>
<td>22 2</td>
<td>Salaried-Employee-Processing</td>
</tr>
<tr>
<td>23 3</td>
<td>Salaried-Paycheck-Validation</td>
</tr>
<tr>
<td>24 3</td>
<td>Salaried-Emp-Update</td>
</tr>
<tr>
<td>25 3</td>
<td>S-Report-Entry-Generation</td>
</tr>
<tr>
<td>26 3</td>
<td>Salaried-Paycheck-Production</td>
</tr>
<tr>
<td>27 4</td>
<td>S-Gross-Pay-Computation</td>
</tr>
<tr>
<td>28 3</td>
<td>Salaried-Paycheck-Computation</td>
</tr>
<tr>
<td>29 2</td>
<td>Shared-Routines</td>
</tr>
<tr>
<td>30 3</td>
<td>Pay-Computation-Validation</td>
</tr>
<tr>
<td>31 3</td>
<td>Tax-Computation</td>
</tr>
<tr>
<td>32 3</td>
<td>Net-Pay-Computation</td>
</tr>
<tr>
<td>33 3</td>
<td>Total-Deductions-Computation</td>
</tr>
<tr>
<td>34 3</td>
<td>Gross-Pay-Update</td>
</tr>
<tr>
<td>35 3</td>
<td>Federal-Deductions-Update</td>
</tr>
<tr>
<td>36 3</td>
<td>State-Deductions-Update</td>
</tr>
<tr>
<td>37 3</td>
<td>Fica-Deductions-Update</td>
</tr>
</tbody>
</table>
**B. Timeliness (Operations and Data)**

**Interval: Month**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Times Happens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employment-Termination-Form</td>
<td>Input</td>
<td>Several</td>
</tr>
<tr>
<td>Tax- withholding-Certificate</td>
<td>Input</td>
<td>Several</td>
</tr>
<tr>
<td>Salaries-Paycheck-Prod-Init</td>
<td>Event</td>
<td>One</td>
</tr>
<tr>
<td>Salaries-Employee-Form</td>
<td>Input</td>
<td>Several</td>
</tr>
<tr>
<td>Salaries-Employee-Report</td>
<td>Output</td>
<td>One</td>
</tr>
<tr>
<td>Salaries-Employee-Processing</td>
<td>Process</td>
<td>One</td>
</tr>
<tr>
<td>Salaries-Emp-Processing-Init</td>
<td>Event</td>
<td>One</td>
</tr>
<tr>
<td>Payroll-Processing</td>
<td>Process</td>
<td>No-of-payroll-processing</td>
</tr>
<tr>
<td>Hourly-Employment-Form</td>
<td>Input</td>
<td>Several</td>
</tr>
<tr>
<td>Error-Listing</td>
<td>Output</td>
<td>No-of-payroll-processing</td>
</tr>
</tbody>
</table>

**Interval: Week**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Times Happens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hired-Employee-Report</td>
<td>Output</td>
<td>One</td>
</tr>
<tr>
<td>Time-Card</td>
<td>Input</td>
<td></td>
</tr>
<tr>
<td>Termination-processing-Init</td>
<td>Event</td>
<td>One</td>
</tr>
<tr>
<td>Terminating-Emp-Processing</td>
<td>Process</td>
<td>One</td>
</tr>
<tr>
<td>Terminated-Employee-Report</td>
<td>Output</td>
<td>One</td>
</tr>
<tr>
<td>New-Employee-processing-Init</td>
<td>Event</td>
<td>One</td>
</tr>
<tr>
<td>New-Employee-Processing</td>
<td>Process</td>
<td>One</td>
</tr>
<tr>
<td>Hourly-Paycheck-Prod-Init</td>
<td>Event</td>
<td>One</td>
</tr>
<tr>
<td>Hourly-Employee-Report</td>
<td>Output</td>
<td>One</td>
</tr>
<tr>
<td>Hourly-Employee-Processing</td>
<td>Process</td>
<td>One</td>
</tr>
<tr>
<td>Hourly-Emp-Processing-Init</td>
<td>Event</td>
<td>One</td>
</tr>
</tbody>
</table>

**Operational Impacts**

411
C. INPUT

<table>
<thead>
<tr>
<th>LEVEL COUNT</th>
<th>LEVEL COUNT</th>
<th>LEVEL COUNT</th>
<th>LEVEL COUNT</th>
<th>LEVEL COUNT</th>
<th>LEVEL COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>EMPLOYEE-INFORMATION</td>
<td>TIME-CARD</td>
<td>HOURLY-EMPLOYMENT-FORM</td>
<td>SALARIED-EMPLOYMENT-FORM</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>TAX-WITHHOLDING-CERTIFICATE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>EMPLOYMENT-TERMINATION-FORM</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D. DATA RETENTION

1* PAYROLL-MASTER-INFORMATION

DESCRIPTION;

THIS SET CONTAINS ONE UNIT OF INFORMATION
FOR EACH EMPLOYEE ON THE PAYROLL, THAT IS,
THOSE EMPLOYEES WHO ARE TO RECEIVE PAYCHECKS.;

2* PAYROLL-MASTER-INFORMATION

DERIVATION;

NEW-EMPLOYEE-PROCESSING ADDS MEMBERS TO THIS SET
AND TERMINATING-EMPLOYEE-PROCESSING DELETES MEMBERS.
(MEMBERS IN THIS CASE ARE EMPLOYEE RECORDS.);
PAYROLL-MASTER-INFORMATION (CONTINUED)
2.4.2.5 DEVELOPMENT IMPACTS

DEFINE
AS A ATTRIBUTE;
/* VALUES ARE:
HIGH FOR HOURLY-EMPLOYEE-PROCESSING,
LOW FOR TERMINATING-EMP-PROCESSING,
HIGH FOR SALARIED-EMPLOYEE-PROCESSING,
MEDIUM FOR NEW-EMPLOYEE-PROCESSING,
*/
DESCRIPTION;
THIS ATTRIBUTE DESIGNATES RELATIVE COMPLEXITY OF
SOFTWARE NEEDED TO PERFORM A PARTICULAR FUNCTION.

EOF EOF EOF EOF

GROUP
CONSISTS OF:
PROGRAMMING,
OPERATIONS,
PAYROLL-CLERICAL,
DATA-BASE-DEVELOPMENT,
COMPUTER-TIME;

EOF EOF EOF EOF

2.5 EXPECTED LIMITATIONS

ERRORS WILL BE NECESSARY FOR BAD DATA INPUT

* ERROR-LISTING
DESCRIPTION;
1 THIS OUTPUT IS A LISTING OF THAT INPUT DATA THAT FAILED
2 THE INPUT VALIDATION RULES.

EXPECTED LIMITATIONS 415
EXPECTED LIMITATIONS 416
3.1 SPECIFIC PERFORMANCE REQUIREMENTS

(SEE SECTION 2)

1 PROCESS
2 SYNONYMS ARE: MAIN-PROCESS,
3 PAYPROC,
4 P1;
5 DESCRIPTION:
6 THIS PROCESS REPRESENTS THE HIGHEST LEVEL PROCESS
7 IN THE TARGET SYSTEM. IT ACCEPTS AND PROCESSES
8 ALL INPUTS AND PRODUCES ALL OUTPUTS;
9 SEE: MEMO: OBJECTIVES-MEMO,
10 PROCESS-MEMO;
11 KEYWORDS: MAIN;
12 SUBPARTS ARE: NEW-EMPLOYEE-PROCESSING,
13 TERMINATING-EMP-PROCESSING,
14 HOURLY-EMPLOYEE-PROCESSING,
15 SALARIED-EMPLOYEE-PROCESSING,
16 SHARED-ROUTINES;
17 RECEIVES: EMPLOYEE-INFORMATION;
18 GENERATES: PAYSYSTEM-OUTPUTS;
19 DERIVES: PAYSYSTEM-OUTPUTS;
20 UPDATES: PAYROLL-MASTER-INFORMATION;
21 USES: EMPLOYEE-INFORMATION,
22 PAYROLL-MASTER-INFORMATION;
23 HAPPENS:
24 NO-OP-PAYROLL-PROCESSING
25 TIMES-PER MONTH;
26 RESPONSIBLE-PROBLEM-DEFINER IS:
27 MICHEL-J-BASTARACHE;
28
29 EOF EOF EOF EOF EOF

DATA RECEIVED:
1 EMPLOYEE-INFORMATION INPUT
2 EMPLOYMENT-TERMINATION-FORM INPUT
3 HOURLY-EMPLOYMENT-FORM INPUT

SPECIFIC PERFORMANCE REQUIREMENTS
4 ARID-EMPLOYMENT-FORM INPUT
5 TAX-WITHHOLDING-CERTIFICATE INPUT
6 TIME-CARD INPUT
7 DEPARTMENT-FILE SET
8 HOURLY-EMPLOYEE-FILE SET
9 PAYROLL-MASTER-INFORMATION SET
10 SALARIED-EMPLOYEE-FILE SET

OUTPUTS GENERATED:
1 ERROR-LISTING OUTPUT
2 HIRED-EMPLOYEE-REPORT OUTPUT
3 HOURLY-EMPLOYEE-REPORT OUTPUT
4 PAY-STATIONMENT OUTPUT
5 PAYSYSTEM-OUTPUTS OUTPUT
6 SALARIED-EMPLOYEE-REPORT OUTPUT
7 TERMINATED-EMPLOYEE-REPORT OUTPUT

3.1.1 ACCURACY AND VALIDITY

1* ACCURACY-MEMO
DESCRIPTION:
1 ACCURACY IS IMPORTANT IN THIS SYSTEM TO THE EFFECT THAT NOTHING
2 SHOULD BE OPEN TO ESTIMATION. ALL SALARIES AND WAGES WILL BE
3 SPECIFICALLY DEFINED. ALSO, A NATURAL ERROR-CHECKING SYSTEM EXISTS
4 IN THAT EACH EMPLOYEE OF THE COMPANY WILL USUALLY CHECK OVER
5 HIS/HER PAY CAREFULLY.;

1 EVENT VALIDITY-CHECK;
2 TRIGGERS: PAY-COMPUTATION-VALIDATION,
3 TIME-CARD-VALIDATION;
4 WHEN: EMPLOYEE-ID-VALID BECOMES FALSE;
5 OR INCEPTION OF:
6 HOURLY-PAYCHECK-VALIDATION,
7 SALARIED-PAYCHECK-VALIDATION;
8
9 EOF EOF EOF EOF EOF EOF

ACCUARITY AND VALIDITY

419
FOR SALARIED EMPLOYEES -

1 EVENT
   SALARIED-EMP-PROCESSING-INIT;
   DESCRIPTION;
   THIS EVENT INITIATES THE PROCESSING OF SALARIED EMPLOYEE
   RECORDS TO PRODUCE PAYCHECKS;
   TRIGGERS: SALARIED-EMPLOYEE-PROCESSING;
   HAPPENS: ONE TIMES-PER MONTH;

2 EOF EOF EOF EOF EOF
   EVENT
   VALIDITY-CHECK;
   TRIGGERS: PAY-COMPUTATION-VALIDATION,
   TIME-CARD-VALIDATION;
   WHEN: EMPLOYEE-ID-VALID BECOMES FALSE;
   ON INCEPTION OF:
   HOURLY-PAYCHECK-VALIDATION,
   SALARIED-PAYCHECK-VALIDATION;

3 EOF EOF EOF EOF EOF
   EVENT
   PASSED-ERROR-CHECKS;
   TRIGGERS: HOURLY-PAYCHECK-COMPUTATION,
   SALARIED-PAYCHECK-COMPUTATION;
   WHEN: ALL-DATA-FOR-EMPLOYEE-FOUND BECOMES TRUE;

4 EOF EOF EOF EOF EOF
   EVENT
   SALARIED-PAYCHECK-PROD-INIT;
   DESCRIPTION;
   THIS EVENT OCCURS AFTER SALARIED PAYCHECK COMPUTATION
   IS COMPLETE.;
   TRIGGERS: SALARIED-PAYCHECK-PRODUCTION;
   ON TERMINATION OF:
   SALARIED-PAYCHECK-COMPUTATION;
   HAPPENS: ONE TIMES-PER MONTH;

5 EOF EOF EOF EOF EOF

FOR HOURLY EMPLOYEES -

1 EVENT
   HOURLY-EMP-PROCESSING-INIT;

TIMING
DESCRIPTION:
* THIS EVENT INITIATES THE PROCESSING of HOURLY EMPLOYEE
  RECORDS TO PRODUCE PAYCHECKS. INITIATION OF SALARIED AND
  AND HOURLY EMPLOYEE PROCESSING MAY BE DONE CONCURRENTLY.

TRIGGERS:  HOURLY-EMPLOYEE-PROCESSING;
WHEN:  TIME-CARDS-READY BECOMES TRUE;
ON INCEPTION OF:  SALARIED-EMPLOYEE-PROCESSING;
HAPPENS:
  ONE TIMES-PER WEEK;

EVENT  VALIDITY-CHECK;
TRIGGERS:  PAY-COMPUTATION-VALIDATION,
TIME-CARD-VALIDATION;
WHEN:  EMPLOYEE-ID-VALID BECOMES FALSE;
ON INCEPTION OF:  HOURLY-PAYCHECK-VALIDATION,
SALARIED-PAYCHECK-VALIDATION;

EVENT  TIME-CARD-MISSING;
TRIGGERS:  TIME-CARD-VALIDATION,
PAY-COMPUTATION-VALIDATION;
WHEN:  NO-TIME-CARD-FOR-EMPLOYEE BECOMES FALSE;

EVENT  PASSED-ERROR-CHECKS;
TRIGGERS:  HOURLY-PAYCHECK-COMPUTATION,
SALARIED-PAYCHECK-COMPUTATION;
WHEN:  ALL-DATA-FOR-EMPLOYEE-FOUND BECOMES TRUE;

EVENT  HOURLY-PAYCHECK-PROD-INIT;
DESCRIPTION:
* THIS EVENT OCCURS AFTER HOURLY PAYCHECK COMPUTATION IS COMPLETE;
TRIGGERS:  HOURLY-PAYCHECK-PRODUCTION;
ON TERMINATION OF:  HOURLY-PAYCHECK-COMPUTATION;
HAPPENS:
  ONE TIMES-PER WEEK;

EVENT  NEW-EMPLOYEE-PROCESSING-INIT;

TIMING  423
DESCRIPTION:
This event occurs after Payroll has run for Hourly employees and initiates processing of new employees.
TRIGGERS: NEW-EMPLOYEE-PROCESSING;
ON TERMINATION OF:
HOURLY-EMPLOYEE-PROCESSING;
HAPPENS:
ONE TIMES-PER WEEK;

EVENT
DESCRIPTION:
This event occurs after new employee processing is completed and initiates terminating employee processing.
TRIGGERS: TERMINATING-EMP-PROCESSING;
ON TERMINATION OF:
NEW-EMPLOYEE-PROCESSING;
HAPPENS:
ONE TIMES-PER WEEK;
3.2 SYSTEM FUNCTIONS

THE VARIOUS SUB-FUNCTIONS OF THE SYSTEM ARE:

1* HOURLY-EMPLOYEE-PROCESSING
DESCRIPTION:
1. THIS PROCESS PERFORMS THOSE ACTIONS NEEDED TO INTERPRET
2. TIME CARDS TO PRODUCE A PAY STATEMENT FOR EACH HOURLY
3. EMPLOYEE.

2* HOURLY-EMPLOYEE-PROCESSING
PROCEDURE:
1. COMPUTE GROSS PAY FROM TIME CARD.
2. COMPUTE TAX FROM GROSS PAY.
3. SUBTRACT TAX FROM GROSS PAY TO OBTAIN NET PAY.
4. UPDATE HOURLY EMPLOYEE RECORD ACCORDINGLY.
5. UPDATE DEPARTMENT RECORD ACCORDINGLY.
6. GENERATE PAYCHECK.
NOTE: IF STATUS CODE SPECIFIES THAT THE EMPLOYEE DID NOT WORK
7. THIS WEEK, NO PROCESSING WILL BE DONE FOR THIS EMPLOYEE
8. RECORD.

3* NEW-EMPLOYEE-PROCESSING
DESCRIPTION:
1. THIS PROCESS STORES INFORMATION ABOUT NEW EMPLOYEES AND
2. THEN PRINTS OUT A CORRESPONDING REPORT.

4* NEW-EMPLOYEE-PROCESSING
PROCEDURE:
1. ADD NEW EMPLOYEE RECORD
2. INCREMENT COUNT OF NUMBER OF EMPLOYEES IN APPROPRIATE
3. DEPARTMENT
4. SPECIFY RELATIONSHIP BETWEEN EMPLOYEE RECORD AND
5. DEPARTMENT
6. INITIALIZE ALL APPROPRIATE FIELDS IN EMPLOYEE RECORD.

5* SALARIED-EMPLOYEE-PROCESSING
DESCRIPTION:
1. THIS PROCESS PRODUCES THE PAY STATEMENT FOR SALARIED
2. EMPLOYEES ONCE A MONTH.
6* SALARIED-EMPLOYEE-PROCESSING
PROCEDURE:
1. SALARY DEFINES GROSS PAY.
2. COMPUTE TAXES FROM GROSS PAY.
3. SUBTRACT TAXES FROM GROSS PAY TO OBTAIN NET PAY.
4. UPDATE SALARIED EMPLOYEE RECORD ACCORDINGLY.
5. UPDATE DEPARTMENT RECORD ACCORDINGLY.
6. GENERATE PAYCHECK
NOTE: HOURS WORKED IS ASSUMED TO BE 40;

7* SHAPED-ROUTINES
DESCRIPTION:
1. THESE ARE ROUTINES USED BY ONE OR MORE PROCESSES
   IN THE SYSTEM;

8* TERMINATING-EMP-PROCESSING
DESCRIPTION:
1. THIS PROCESS DELETES DATA, FOR THOSE EMPLOYEES WHO
   ARE NO LONGER ON THE PAYROLL, FROM THE FILES. IT ALSO
2. PRINTS A LIST OF ALL EMPLOYEES NO LONGER ON THE
3. PAYROLL;

9* TERMINATING-EMP-PROCESSING
PROCEDURE:
1. DETERMINE TYPE OF EMPLOYEE BY EMPLOYMENT STATUS ITEM
2. FROM THIS, RETRIEVE THE CONTENTS OF THE APPROPRIATE
   EMPLOYEE RECORD AND PRINT IN REPORT FORMAT
3. UPDATE NUMBER OF EMPLOYEES FIELD IN APPROPRIATE
   DEPARTMENT RECORD
4. DELETE EMPLOYEE RECORD;

3.3 INPUTS/OUTPUTS

INPUTS:
1 INPUT SYNONYMS ARE: EMP-INFO,
2 I1;
3 DESCRIPTION:
4 THIS INPUT REPRESENTS ALL THE NECESSARY INFORMATION TO
5 PRODUCE THE OUTPUTS FROM THE PAYSYSTEM;

INPUTS/OUTPUTS
INPUT
SYNONYMS ARE: H-EMP-FORM;
DESCRIPTION:
THIS INPUT CONTAINS THE INFORMATION NECESSARY TO
ADD A NEW HOURLY EMPLOYEE TO THE PAYROLL.
KEYWORDS: NEW;
ATTRIBUTES ARE:
ARRIVAL-TYPE RANDOM;

INPUTS/OUTPUTS

EMPLOYMENT-TERMINATION-FORM:

SYNONYMS ARE: TERM-FORM;
DESCRIPTION:
THIS INPUT CONTAINS THE INFORMATION NECESSARY TO
DELETE AN EMPLOYEE FROM THE PAYROLL.
KEYWORDS: TERM-KEY;
ATTRIBUTES ARE:
ARRIVAL-TYPE RANDOM;
PART OF: EMPLOYEE-INFORMATION;
CONSISTS OF:
EMPLOYEE-NAME,
SOCIAL-SECURITY-NUMBER,
TERMINATION-DATE,
EMPLOYEE-IDENTIFICATION-NUMBER,
EMPLOYMENT-STATUS;
GENERATED BY: PAYROLL-DEPARTMENT;
RECEIVED BY: TERMINATING-EMP-PROCESSING;
USED BY: TERMINATING-EMP-PROCESSING;
HAPPENS:
SEVERAL TIMES-PER MONTH;

INPUT
SYNONYMS ARE: I-O-CONSTRAINTS-MEMO;
KEYWORDS: MAIN;
SUBPARTS ARE: TIME-CARD,
HOURLY-EMPLOYMENT-FORM,
SALARIED-EMPLOYMENT-FORM,
TAX-WITHHOLDING-CERTIFICATE,
EMPLOYMENT-TERMINATION-FORM;
GENERATED BY: DEPARTMENTS-AND-EMPLOYEES;
RECEIVED BY: PAYROLL-PROCESSING;
USED BY: PAYROLL-PROCESSING;
RESPONSIBLE-PROBLEM-DEFINER IS:
MICHEL-J-BASTARACHE;
PART OF: EMPLOYEE-INFORMATION;
CONSISTS OF:
PERSONAL-DATA,
HOURLY-JOB-DATA;
GENERATED BY: PAYROLL-DEPARTMENT;
RECEIVED BY: NEW-EMPLOYEE-PROCESSING;
USED BY:
HOURLY-RECORD-CREATION,
NEW-EMPLOYEE-PROCESSING;
USED BY:
HIRE-REPORT-ENTRY-GENERATION
TO DERIVE HIRED-REPORT-ENTRY;
HAPPENS:
SEVERAL
TIMES-PER MONTH;

INPUT
SYNONYMS ARE: S-EMP-FORM;
DESCRIPTION:
THIS INPUT CONTAINS THE INFORMATION NECESSARY TO
ADD A NEW SALARIED EMPLOYEE TO THE PAYROLL.;
KEYWORDS:
NEW;
ATTRIBUTES ARE:
ARRIVAL-TYPE RANDOM;
PART OF:
EMPLOYEE-INFORMATION;
CONSISTS OF:
PERSONAL-DATA,
SALARIED-JOB-DATA;
GENERATED BY: PAYROLL-DEPARTMENT;
RECEIVED BY: NEW-EMPLOYEE-PROCESSING;
USED BY:
NEW-EMPLOYEE-PROCESSING,
SALARIED-RECORD-CREATION;
USED BY:
HIRE-REPORT-ENTRY-GENERATION
TO DERIVE HIRED-REPORT-ENTRY;
HAPPENS:
SEVERAL
TIMES-PER MONTH;

INPUT
SYNONYMS ARE: TAX-CERT;
DESCRIPTION:
THIS INPUT CONTAINS TAX INFORMATION NECESSARY TO
COMPUTE THE EMPLOYEE'S PAYCHECK.;

INPUTS/OUTPUTS 428
KEYWORDS: NEW;

ATTRIBUTES ARE:
ARRIVAL-TYPE RANDOM;

PART OF: EMPLOYEE-INFORMATION;

CONSISTS OF:
  EMPLOYEE-NAME,
  ADDRESS,
  SOCIAL-SECURITY-NUMBER,
  NUMBER-OF-DEDUCTIONS,
  CURRENT-DATE;

GENERATED BY: EMPLOYEE;
RECEIVED BY: NEW-EMPLOYEE-PROCESSING;
USED BY: NEW-EMPLOYEE-PROCESSING;

HAPPENS:
  SEVERAL TIMES-PER MONTH;

INPUT

SYNONYMS ARE: T-CARD;

DESCRIPTION;
  THIS INPUT CONTAINS THE INFORMATION ABOUT THE HOURS THAT AN
  HOURLY EMPLOYEE WORKED THE PRECEDING WEEK;

SEE-MEMO: I-O-CONSTRAINTS-MEMO;

KEYWORDS: HOURLY;

ATTRIBUTES ARE:
ARRIVAL-TYPE SCHEDULED;

PART OF: EMPLOYEE-INFORMATION;

CONSISTS OF:
  EMPLOYEE-NAME,
  SOCIAL-SECURITY-NUMBER,
  STATUS-CODE,
  PAY-DATE,
  REGULAR-HOURS-WORKED,
  OVERTIME-HOURS-WORKED,
  7 HOURS-PER-DAY,
  EMPLOYEE-IDENTIFICATION-NUMBER;

GENERATED BY: EMPLOYEE;
RECEIVED BY: HOURLY-EMPLOYEE-PROCESSING;
USED BY: HOURLY-EMPLOYEE-PROCESSING;

HAPPENS:
  NO-OF-HOURLY-EMPLOYEES TIMES-PER WEEK;

INPUTS/OUTPUTS
OUTPUTS:

1 OUTPUT
SYNONYMS ARE:  E-LIST;
DESCRIPTION:
   THIS OUTPUT IS A LISTING OF THAT INPUT DATA THAT FAILED
   THE INPUT VALIDATION RULES.
KEYWORDS:  HOURLY,
SALARIED;
PART OF:  PAYSYSTEM-OUTPUTS;
CONSISTS OF:
   ERROR-LISTING-ENTRY,
   ERROR-CODE;
   SEVERAL
   ERROR-CODE;
   Derived by:  SALARIED-EMPLOYEE-PROCESSING;
   Derived by:  HOURLY-EMPLOYEE-PROCESSING;
   Generated by:  HOURLY-EMPLOYEE-PROCESSING,
   SALARIED-EMPLOYEE-PROCESSING;
   Received by:  PAYROLL-DEPARTMENT;
   Happens:
   NO-OF-PAYROLL-PROCESSING
   TIMES-PER MONTH;

21 OUTPUT
SYNONYMS ARE:  HIRED-REPORT;
DESCRIPTION:
   THIS REPORT PRESENTS INFORMATION ABOUT ALL EMPLOYEES
   HIRED IN THE PREVIOUS WEEK.
KEYWORDS:  NEW;
ATTRIBUTES ARE:
COPY
TWO;
PART OF:  PAYSYSTEM-OUTPUTS;
CONSISTS OF:
   HIRED-REPORT-ENTRY;
   Derived by:  NEW-EMPLOYEE-PROCESSING;
   Generated by:  NEW-EMPLOYEE-PROCESSING;
   Received by:  PAYROLL-DEPARTMENT;
   Happens:
   ONE TIMES-PER WEEK;

38 OUTPUT

HOURLY-EMPLOYEE-REPORT;

INPUTS/OUTPUTS
SYNONYMS ARE: H-EMP-REPORT;

DESCRIPTION;
THIS IS AN ADMINISTRATIVE RECORD OF ALL HOURLY
EMPLOYEES PAID IN ONE WEEK;

KEYWORDS: HOURLY;

ATTRIBUTES ARE:

COPIES THREE;

PART OF: PAYSYSTEM-OUTPUTS;

CONSISTS OF:

H-EMP-REPORT-ENTRY;

DERIVED BY: HOURLY-EMPLOYEE-PROCESSING;

GENERATED BY: HOURLY-EMPLOYEE-PROCESSING;

RECEIVED BY: PAYROLL-DEPARTMENT;

HAPPENS:

ONE TIMES-PER WEEK;

OUTPUT

SYNONYMS ARE: PAYCHECK;

DESCRIPTION;

THIS OUTPUT IS THE PAYMENT TO THE EMPLOYEE FOR THE PREVIOUS
WORK PERIOD;

SEE-MEMO: I-O-CONSTRAINTS-MEMO;

KEYWORDS: HOURLY,

SALARIED;

PART OF: PAYSYSTEM-OUTPUTS;

CONSISTS OF:

PAY-STUB,

CHECK;

DERIVED BY: SALARIED-EMPLOYEE-PROCESSING;

DERIVED BY: HOURLY-EMPLOYEE-PROCESSING;

GENERATED BY: HOURLY-EMPLOYEE-PROCESSING,

SALARIED-EMPLOYEE-PROCESSING;

RECEIVED BY: EMPLOYEE;

OUTPUT

SYNONYMS ARE: O1,

PAYOUTS;

DESCRIPTION;

THIS OUTPUT REPRESENTS ALL THE REQUIRED OUTPUTS OF THE
TARGET PAYSYSTEM AS DEFINED BY POLICY;

SEE-MEMO: I-O-CONSTRAINTS-MEMO;

KEYWORDS: MAIN;

SUBPARTS ARE: PAY-STATION,
OUTPUT

SYNONYMS ARE: S-EMP-REPORT;

DESCRIPTION:
THIS IS AN ADMINISTRATIVE RECORD OF ALL SALARIED
EMPLOYEES PAID IN ONE MONTH.

KEYWORDS: SALARIED;

ATTRIBUTES ARE:

COPIES THREE;

PART OF: PAYSYSTEM-OUTPUTS;

CONSISTS OF:
S-EMP-REPORT-ENTRY;

DERIVED BY: SALARIED-EMPLOYEE-PROCESSING;

GENERATED BY: SALARIED-EMPLOYEE-PROCESSING;

RECEIVED BY: PAYROLL-DEPARTMENT;

HAPPENS:
ONE TIMES-PER MONTH;

OUTPUT

SYNONYMS ARE: TERM-REPORT;

DESCRIPTION:
THIS REPORT PRESENTS A LISTING OF ALL EMPLOYEES THAT
ARE NO LONGER ON THE PAYROLL.

KEYWORDS: TERM-KEY;

ATTRIBUTES ARE:

COPIES TWO;

PART OF: PAYSYSTEM-OUTPUTS;

CONSISTS OF:
TERM-REPORT-ENTRY;

DERIVED BY: TERMINATING-EMP-PROCESSING;

GENERATED BY: TERMINATING-EMP-PROCESSING;

RECEIVED BY: PAYROLL-DEPARTMENT;
3.4 DATA CHARACTERISTICS

1 SET
SYNONYMS ARE: MASTER-FILE,
PAY-MAST,
S1;

DESCRIPTION;
THIS SET CONTAINS ONE UNIT OF INFORMATION
FOR EACH EMPLOYEE ON THE PAYROLL, THAT IS,
THOSE EMPLOYEES WHO ARE TO RECEIVE PAYCHECKS.;
SEE-MEMO: OBJECTIVES-MEMO;
KEYWORDS: MAIN;
CONSISTS OF:
12 HOUMLY-EMPLOYEE-RECORD,
13 SALARIED-EMPLOYEE-RECORD,
14 DEPARTMENT-RECORD;
SUBSET ARE: DEPARTMENT-FILE,

PAYROLL-MASTER-INFORMATION;
HOURLY-EMPLOYEE-FILE
SALARIED-EMPLOYEE-FILE;
SU SETTING-Criteria ARE:
EMPLOYMENT-STATUS,
DEPARTMENT;
DERIVATION;
NEW-EMPLOYEE-PROCESSING ADDS MEMBERS TO THIS SET
AND TERMINATING-EMPLOYEE-PROCESSING DELETES MEMBERS.
(MEMBERS IN THIS CASE ARE EMPLOYEE RECORDS.);
USED BY: PAYROLL-PROCESSING;
UPDATED BY: PAYROLL-PROCESSING;
OCURRENCES: NO-OF-EMPLOYEES;
RESPONSIBLE-REAL-WORLD-ENTITY IS:
PAYROLL-DEPARTMENT;
RESPONSIBLE-PROBLEM-DEFINER IS:
MICHEL-J-DASTARACHE;
EOF EOF EOF EOF EOF

1* DEPARTMENT-FILE
DESCRIPTION;
THIS FILE CONSISTS OF ALL EMPLOYEE RECORDS FOR A
GIVEN DEPARTMENT. ONE FILE IS AVAILABLE FOR EACH
DEPARTMENT.;

2* DEPARTMENT-FILE
VOLATILITY-SET;
THIS FILE IS CHANGED WHENEVER A NEW EMPLOYEE IS HIRED
INTO THE DEPARTMENT OR EMPLOYEE LEAVES THE DEPARTMENT.
THIS HAPPENS ABOUT FOUR TIMES A MONTH.;

3* DEPARTMENT-FILE
DERIVATION;
NEW-EMPLOYEE-PROCESSING ADDS MEMBERS TO THIS SET
AND TERMINATING-EMPLOYEE-PROCESSING DELETES MEMBERS.
(MEMBERS IN THIS CASE ARE EMPLOYEE RECORDS.);

1* HOURLY-EMPLOYEE-FILE
DESCRIPTION;
THIS FILE CONSISTS OF ALL HOURLY EMPLOYEE RECORDS.;
2. **HOURLY-EMPLOYEE-FILE**
   VOLATILITY-MEMBER;
   1. MEMBERS OF THIS SET ARE MODIFIED ABOUT ONCE A WEEK.

3. **HOURLY-EMPLOYEE-FILE**
   DERIVATION;
   1. NEW-EMPLOYEE-PROCESSING ADDS MEMBERS TO THIS SET
   2. AND TERMINATING-EMPLOYEE-PROCESSING DELETES MEMBERS.
   3. (MEMBERS IN THIS CASE ARE EMPLOYEE RECORDS.)
1* SALARIED-EMPLOYEE-FILE
DESCRIPTION:
THIS FILE CONSISTS OF ALL SALARIED EMPLOYEE RECORDS;

2* SALARIED-EMPLOYEE-FILE
VOLATILITY-MEMBER;
MEMBERS OF THIS SET ARE MODIFIED ABOUT ONCE A MONTH;

3* SALARIED-EMPLOYEE-FILE
DERIVATION:
NEW-EMPLOYEE-PROCESSING ADDS MEMBERS TO THIS SET
AND TERMINATING-EMPLOYEE-PROCESSING DELETES MEMBERS.
(MEMBERS IN THIS CASE ARE EMPLOYEE RECORDS);
3.5 FAILURE CONTINGENCIES
SYSTEM BACKUP IS PROVIDED BY THE DATA BASE MANAGEMENT SYSTEM AS WELL AS THE OPERATING SYSTEM ITSELF. IN CASE OF COMPUTER FAILURE, THE SYSTEM KEEPS BACKUP FILES ON TAPES SO THE DATA BASE CANNOT BE LOST. IN CASE OF A TOTAL SYSTEM FAILURE, PROCEDURES FOR DOING THE PAYROLL BY HAND HAVE ALSO BEEN SET UP.
4.1 EQUIPMENT ENVIRONMENT

1* EQUIPMENT-MEMO
DESCRIPTION:
1 THIS ORGANIZATION IS CURRENTLY USING AN IBM 370-168,
2 WHICH IS MORE THAN ADEQUATE FOR DOING THE PAYROLL.
3 THE ONLY EQUIPMENT LACKING AT THIS POINT IS A SPECIAL
4 TYPE OF LINE PRINTER FOR DOING THE CHECKS.
5 ;

4.2 SUPPORT SOFTWARE ENVIRONMENT

1* SOFTWARE-MEMO
DESCRIPTION:
1 THIS SYSTEM WILL USE INSTALLATION SORT ROUTINES AND VARIOUS
2 OTHER SOFTWARE PACKAGES.
3 ;

4.3 INTERFACES

COUNT LEVEL NAME

<table>
<thead>
<tr>
<th></th>
<th>1 ACCOUNTING-SYSTEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2 PAYROLL-SYSTEM</td>
</tr>
<tr>
<td>3</td>
<td>1 DEPARTMENTS-AND-EMPLOYEES</td>
</tr>
<tr>
<td>4</td>
<td>2 EMPLOYEE</td>
</tr>
<tr>
<td>5</td>
<td>2 PAYROLL-DEPARTMENT</td>
</tr>
</tbody>
</table>
Interfaces
4.4 SECURITY
The security of the system is based upon the fact that information in the data base is something that should stay between the employer and each individual employee. It would not be good management to let employees know what their fellow employees are making. While at the same time it would not be advisable to let information on managers' salaries be available. It must be remembered that payroll information is the type of information that is treated in confidence.

Define as a security;
Applies to: Hourly-Employee-Processing,
Salaried-Employee-Processing,
New-Employee-Processing;
Define as a security;
Applies to: Hourly-Employee-Record,
Salaried-Employee-Record;
COSTS INVOLVED WITH THIS PROJECT INCLUDE:
1 DEVELOPMENT COSTS
2 PROGRAMMING COSTS
3 COMPUTER COSTS
4 DATA BASE COSTS
5 OPERATIONS COSTS
6 DATA COLLECTION COSTS
7 SECURITY COSTS
DEVELOPMENT PLAN

1# DEVELOPMENT-MEMO
DESCRIPTION:
DEVELOP OF THIS SYSTEM ENTAILS THE CONCENTRATED EFFORT
OF BOTH THE PAYROLL DEPARTMENT AND THE EDP DEPARTMENT

1 DEFINE
2 AS A ATTRIBUTE;
3 /* VALUES ARE:
4 PLENTY FOR MAN-HOURS-120 COMPUTER-TIME,
5 FOR MAN-HOURS-20 PROGRAMMING,
6 FOR MAN-HOURS-30 PAYROLL-CLERICAL,
7 FOR MAN-HOURS-100 OPERATIONS,
8 FOR DATA-BASE-DEVELOPMENT,
9 */
10
15 EOF EOF EOF EOF EOF

DEVELOPMENT-TIMES
<table>
<thead>
<tr>
<th></th>
<th>DATA REQUIREMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>SURVEY OF DATA REQUIREMENTS</td>
</tr>
<tr>
<td>1.2</td>
<td>PROJECT REFERENCES</td>
</tr>
<tr>
<td>1.3</td>
<td>MODIFICATION OF DATA REQUIREMENTS</td>
</tr>
<tr>
<td>2.</td>
<td>DATA DESCRIPTION</td>
</tr>
<tr>
<td>2.1</td>
<td>LOGICAL ORGANIZATION OF STATIC SYSTEM DATA</td>
</tr>
<tr>
<td>2.2</td>
<td>LOGICAL ORGANIZATION OF DYNAMIC INPUT DATA</td>
</tr>
<tr>
<td>2.3</td>
<td>LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA</td>
</tr>
<tr>
<td>2.4</td>
<td>INTERNALLY GENERATED DATA</td>
</tr>
<tr>
<td>2.5</td>
<td>SYSTEM DATA CONSTRAINTS</td>
</tr>
<tr>
<td>3.</td>
<td>USER SUPPORT FOR DATA COLLECTION</td>
</tr>
<tr>
<td>3.1</td>
<td>DATA COLLECTION REQUIREMENTS AND SCOPE</td>
</tr>
<tr>
<td>3.2</td>
<td>RECOMMEND SOURCE OF INPUT DATA</td>
</tr>
<tr>
<td>3.3</td>
<td>DATA COLLECTION AND TRANSFER PROCEDURES</td>
</tr>
<tr>
<td>3.4</td>
<td>DATA BASE IMPACTS</td>
</tr>
</tbody>
</table>

TABLE OF CONTENTS
1 PURPOSE OF DATA REQUIREMENTS

THE OBJECTIVES OF THIS DATA REQUIREMENTS DOCUMENT FOR THE PAYROLL DEPARTMENT EXAMPLE, PROJECT #1 ARE TO LIST AND DEFINE DATA ELEMENTS WHICH THE SYSTEM MUST HANDLE AND COMMUNICATE DATA COLLECTION REQUIREMENTS TO THE USER.

1.2 PROJECT REFERENCES

1* OBJECTIVES-MEMO
DESCRIPTION:
1  THIS PROJECT HAS BEEN AUTHORIZED TO DEVELOP A PAY SYSTEM
2  FOR THIS ORGANIZATION. THIS PAY SYSTEM WILL PERFORM PAYROLL
3  PROCESSING USING EMPLOYEE INFORMATION COMING FROM
4  DEPARTMENTS AND EMPLOYEES AND WILL PRODUCE OUTPUTS WHICH
5  WILL GO TO THE DEPARTMENTS AND EMPLOYEES. THE SYSTEM WILL
6  ALSO MAINTAIN THE PAYROLL MASTER INFORMATION.

1* PAYROLL-DEPARTMENT
DESCRIPTION:
1  THIS DEPARTMENT IS RESPONSIBLE FOR ALL PAYROLL DATA.
2  THIS DEPARTMENT IS ALSO THE SPONSOR OF THIS PROJECT
3  ;

1.3 MODIFICATION OF DATA REQUIREMENTS

NOT APPLICABLE TO THIS PROJECT.
THE DATA DESCRIBED IN THIS SECTION SHALL BE SEPARATED INTO TWO CATEGORIES, STATIC DATA AND DYNAMIC DATA. STATIC DATA IS DEFINED AS THAT DATA WHICH IS USED MAINLY FOR REFERENCE DURING SYSTEM OPERATION AND IS USUALLY GENERATED OR UPDATED IN WIDELY SEPARATED TIME PERIODS INDEPENDENT OF NORMAL SYSTEM RUNS. DYNAMIC DATA INCLUDES ALL DATA WHICH IS INTENDED TO BE UPDATED AND WHICH IS INPUT TO A SYSTEM DURING A NORMAL RUN (INCLUDING "REAL TIME" DATA SUCH AS TARGETING DATA) OR IS OUTPUT BY THE SYSTEM. STATIC DATA AS DESCRIBED ABOVE IS FREQUENTLY REFERRED TO AS PARAMETRIC DATA AND DYNAMIC DATA AS NON-PARAMETRIC DATA. BOTH, HOWEVER, ARE COMPOSED OF DATA ELEMENTS. THE DATA ELEMENT NAMES LISTED IN PARAGRAPHS 2.1, 2.2, AND 2.3 SHALL BE THOSE CONTAINED IN STANDARD DATA ELEMENT LIBRARIES, WHENEVER APPLICABLE.

2.1 LOGICAL ORGANIZATION OF STATIC SYSTEM DATA

<table>
<thead>
<tr>
<th>#</th>
<th>PAYROLL-MASTER-INFORMATION (SET)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 HOURLY-EMPLOYEE-RECORD (ENTITY)</td>
</tr>
<tr>
<td>2</td>
<td>3 EMPLOYEE-NAME (GROUP)</td>
</tr>
<tr>
<td>3</td>
<td>4 SURNAME (ELEMENT)</td>
</tr>
<tr>
<td>4</td>
<td>4 INITIAL (ELEMENT)</td>
</tr>
<tr>
<td>5</td>
<td>4 FIRST-NAME (ELEMENT)</td>
</tr>
<tr>
<td>6</td>
<td>3 EMPLOYEE-IDENTIFICATION-NUMBER (ELEMENT)</td>
</tr>
<tr>
<td>7</td>
<td>3 SOCIAL-SECURITY-NUMBER (ELEMENT)</td>
</tr>
<tr>
<td>8</td>
<td>3 PAY-GRADE-CODE (ELEMENT)</td>
</tr>
<tr>
<td>9</td>
<td>3 ADDRESS (GROUP)</td>
</tr>
<tr>
<td>10</td>
<td>4 HOUSE-NUMBER (ELEMENT)</td>
</tr>
<tr>
<td>11</td>
<td>4 STREET (ELEMENT)</td>
</tr>
<tr>
<td>12</td>
<td>4 APARTMENT-NUMBER (ELEMENT)</td>
</tr>
<tr>
<td>13</td>
<td>4 CITY (ELEMENT)</td>
</tr>
<tr>
<td>14</td>
<td>4 STATE (ELEMENT)</td>
</tr>
<tr>
<td>15</td>
<td>4 ZIP-CODE (ELEMENT)</td>
</tr>
<tr>
<td>16</td>
<td>3 PHONE (ELEMENT)</td>
</tr>
<tr>
<td>17</td>
<td>3 EMPLOYMENT-DATE (ELEMENT)</td>
</tr>
<tr>
<td>18</td>
<td>3 NUMBER-OF-DEDUCTIONS (ELEMENT)</td>
</tr>
<tr>
<td>19</td>
<td>3 DEPARTMENT (ELEMENT)</td>
</tr>
<tr>
<td>20</td>
<td>3 CUMULATIVE-GROSS-PAY (ELEMENT)</td>
</tr>
<tr>
<td>21</td>
<td>3 CUMULATIVE-FEDERAL-DEDUCTIONS (ELEMENT)</td>
</tr>
<tr>
<td>22</td>
<td>3 CUMULATIVE-STATE-DEDUCTIONS (ELEMENT)</td>
</tr>
</tbody>
</table>
23  3  CUMULATIVE-FICA-DEDUCTIONS (ELEMENT)
24  3  AGE (ELEMENT)
25  3  SEX (ELEMENT)
26  2  STATUS-CODE (ELEMENT)
27  3  CUMULATIVE-HOURS (ELEMENT)
28  2  SALARIED-EMPLOYEE-RECORD (ENTITY)
29  3  EMPLOYEE-NAME (GROUP)
30  4  SURNAME (ELEMENT)
31  4  INITIAL (ELEMENT)
32  4  FIRST-NAME (ELEMENT)
33  3  EMPLOYEE-IDENTIFICATION-NUMBER (ELEMENT)
34  3  SOCIAL-SECURITY-NUMBER (ELEMENT)
35  3  PAY-GRADE-CODE (ELEMENT)
36  3  ADDRESS (GROUP)
37  4  HOUSE-NUMBER (ELEMENT)
38  4  STREET (ELEMENT)
39  4  APARTMENT-NUMBER (ELEMENT)
40  4  CITY (ELEMENT)
41  4  STATE (ELEMENT)
42  4  ZIP-CODE (ELEMENT)
43  3  PHONE (ELEMENT)
1. DEPARTMENT-DATE (ELEMENT)
2. NUMBER-OF-DEDUCTIONS (ELEMENT)
3. DEPARTMENT (ELEMENT)
4. CUMULATIVE-FEDERAL-DEDUCTIONS (ELEMENT)
5. CUMULATIVE-GROSS-PAY (ELEMENT)
6. CUMULATIVE-STATE-DEDUCTIONS (ELEMENT)
7. CUMULATIVE-FICA-DEDUCTIONS (ELEMENT)
8. AGE (ELEMENT)
9. SEX (ELEMENT)
10. STATUS-CODE (ELEMENT)

2. DEPARTMENT-RECORD (ENTITY)
3. DEPARTMENT (ELEMENT)
4. SUPERVISOR (ELEMENT) (2)
5. NUMBER-OF-EMPLOYEES (ELEMENT)
6. TOTAL-BUDGET (ELEMENT)
7. REMAINING-FUNDS (ELEMENT)

1. DEPARTMENT-FILE (SET)
2. DEPARTMENT-RECORD (ENTITY) (NO-OF-DEPARTMENTS)
3. DEPARTMENT (ELEMENT)
4. SUPERVISOR (ELEMENT) (2)
5. NUMBER-OF-EMPLOYEES (ELEMENT)
6. TOTAL-BUDGET (ELEMENT)
7. REMAINING-FUNDS (ELEMENT)

1. HOURLY-EMPLOYEE-FILE (SET)
2. HOURLY-EMPLOYEE-RECORD (ENTITY)
3. EMPLOYEE-NAME (GROUP)
4. SURNAME (ELEMENT)
5. INITIAL (ELEMENT)
6. FIRST-NAME (ELEMENT)
7. EMPLOYEE-IDENTIFICATION-NUMBER (ELEMENT)
8. SOCIAL-SECURITY-NUMBER (ELEMENT)
9. PAY-GRADE-CODE (ELEMENT)
10. ADDRESS (GROUP)
11. HOUSE-NUMBER (ELEMENT)
12. STREET (ELEMENT)
13. APARTMENT-NUMBER (ELEMENT)
14. CITY (ELEMENT)
15. STATE (ELEMENT)
16. ZIP-CODE (ELEMENT)

LOGICAL ORGANIZATION OF STATIC SYSTEM DATA
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>PHONE (ELEMENT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>3</td>
<td>EMPLOYMENT-DATE (ELEMENT)</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>NUMBER-OF-DEDUCTIONS (ELEMENT)</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
<td>DEPARTMENT (ELEMENT)</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>CUMULATIVE-GROSS-PAY (ELEMENT)</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>CUMULATIVE-FEDERAL-DEDUCTIONS (ELEMENT)</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
<td>CUMULATIVE-STATE-DEDUCTIONS (ELEMENT)</td>
</tr>
<tr>
<td>22</td>
<td>3</td>
<td>CUMULATIVE-FICA-DEDUCTIONS (ELEMENT)</td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>AGE (ELEMENT)</td>
</tr>
<tr>
<td>24</td>
<td>3</td>
<td>SEX (ELEMENT)</td>
</tr>
<tr>
<td>25</td>
<td>3</td>
<td>STATUS-CODE (ELEMENT)</td>
</tr>
<tr>
<td>26</td>
<td>3</td>
<td>CUMULATIVE-HOURS (ELEMENT)</td>
</tr>
</tbody>
</table>

1* 1 SALARIED-EMPLOYEE-FILE (SET)
   2 SALARIED-EMPLOYEE-RECORD (ENTITY)
   3 EMPLOYEE-NAME (GROUP)
   4   SURNAME (ELEMENT)
   5   INITIAL (ELEMENT)
   6   FIRST-NAME (ELEMENT)
   7   EMPLOYEE-IDENTIFICATION-NUMBER (ELEMENT)
   8   SOCIAL-SECURITY-NUMBER (ELEMENT)
   9   PAY-RATE-CODE (ELEMENT)
   10  ADDRESS (GROUP)
   11  HOUSE-NUMBER (ELEMENT)
   12  STREET (ELEMENT)
   13  APARTMENT-NUMBER (ELEMENT)
   14  CITY (ELEMENT)
   15  STATE (ELEMENT)
   16  ZIP-CODE (ELEMENT)
   17  PHONE (ELEMENT)
   18  EMPLOYMENT-DATE (ELEMENT)
   19  NUMBER-OF-DEDUCTIONS (ELEMENT)
   20  DEPARTMENT (ELEMENT)
   21  CUMULATIVE-FEDERAL-DEDUCTIONS (ELEMENT)
   22  CUMULATIVE-GROSS-PAY (ELEMENT)
   23  CUMULATIVE-STATE-DEDUCTIONS (ELEMENT)
   24  CUMULATIVE-FICA-DEDUCTIONS (ELEMENT)
   25  AGE (ELEMENT)
   26  SEX (ELEMENT)
   27  STATUS-CODE (ELEMENT)
### Logical Organization of Dynamic Input Data

<table>
<thead>
<tr>
<th>1st Level</th>
<th>2nd Level</th>
<th>3rd Level</th>
<th>4th Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYMENT-TERMINATION-FORM (INPUT)</td>
<td>EMPLOYEE-NAME (GROUP)</td>
<td>SURNAME (ELEMENT)</td>
<td>INITIAL (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FIRST-NAME (ELEMENT)</td>
<td>SOCIAL-SECURITY-NUMBER (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATION-DATE (ELEMENT)</td>
<td>EMPLOYEE-IDENTIFICATION-NUMBER (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EMPLOYMENT-STATUS (ELEMENT)</td>
<td></td>
</tr>
<tr>
<td>HOURSLY-EMPLOYMENT-FORM (INPUT)</td>
<td>PERSONAL-DATA (GROUP)</td>
<td>EMPLOYEE-NAME (GROUP)</td>
<td>SURNAME (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INITIAL (ELEMENT)</td>
<td>FIRST-NAME (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOCIAL-SECURITY-NUMBER (ELEMENT)</td>
<td>SEX (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIRTHDATE (ELEMENT)</td>
<td>ADDRESS (GROUP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HOUSE-NUMBER (ELEMENT)</td>
<td>STREET (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>APARTMENT-NUMBER (ELEMENT)</td>
<td>CITY (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STATE (ELEMENT)</td>
<td>ZIP-CODE (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHONE (ELEMENT)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>HOURSLY-JOB-DATA (GROUP)</td>
<td>JOB-TITLE (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAY-RATE (ELEMENT)</td>
<td>CURRENT-DATE (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EMPLOYMENT-DATE (ELEMENT)</td>
<td>JOB-NUMBER (ELEMENT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAY-RATE-CODE (ELEMENT)</td>
<td>SUPERVISOR (ELEMENT) (2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEPARTMENT (ELEMENT)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>EMPLOYEE-IDENTIFICATION-NUMBER (ELEMENT)</td>
<td></td>
</tr>
<tr>
<td>EMPLOYEE-NAME (GROUP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SURNAME (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INITIAL (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIRST-NAME (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOCIAL-SECURITY-NUMBER (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEX (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIRTHDATE (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDRESS (GROUP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOUSE-NUMBER (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STREET (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APARTMENT-NUMBER (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CITY (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATE (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP-CODE (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHONE (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALARIZED-JOB-DATA (GROUP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JOB-TITLE (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAY-GRADE-CODE (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CURRENT-DATE (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMPLOYMENT-DATE (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUPERVISOR (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEPARTMENT (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMPLOYEE-IDENTIFICATION-NUMBER (ELEMENT)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4* 1 TAX-WITHHOLDING-CERTIFICATE (INPUT)
   1 2 EMPLOYEE-NAME (GROUP)
   2 3 SURNAME (ELEMENT)
   3 3 INITIAL (ELEMENT)
   4 3 FIRST-NAME (ELEMENT)
   5 2 ADDRESS (GROUP)
   6 3 HOUSE-NUMBER (ELEMENT)
   7 3 STREET (ELEMENT)
   8 3 APARTMENT-NUMBER (ELEMENT)
   9 3 CITY (ELEMENT)
  10 3 STATE (ELEMENT)
  11 3 ZIP-CODE (ELEMENT)
  12 2 SOCIAL-SECURITY-NUMBER (ELEMENT)
  13 2 NUMBER-OF-DEDUCTIONS (ELEMENT)
  14 2 CURRENT-DATE (ELEMENT)

5* 1 TIME-CARD (INPUT)
   1 2 EMPLOYEE-NAME (GROUP)
   2 3 SURNAME (ELEMENT)

LOGICAL ORGANIZATION OF DYNAMIC INPUT DATA
2.3 LOGICAL ORGANIZATION OF DYNAMIC OUTPUT DATA

1* 1. PAY-STATEMENT (OUTPUT)
  1   2. PAY-STUB (GROUP)
      2.1 EMPLOYEE-NAME (GROUP)
          2.1.1 INITIAL (ELEMENT)
          2.1.2 FIRST-NAME (ELEMENT)
          2.1.3 SURNAME (ELEMENT)
          2.1.4 INITIAL (ELEMENT)
          2.1.5 FIRST-NAME (ELEMENT)
      2.2 SOCIAL-SECURITY-NUMBER (ELEMENT)
      2.3 PAY-DATE (ELEMENT)
      2.4 CHECK-NUMBER (ELEMENT)
      2.5 TOTAL-HOURS (ELEMENT)
      2.6 GROSS-PAY (ELEMENT)
      2.7 TOTAL-DEDUCTIONS (ELEMENT)
      2.8 NET-PAY (ELEMENT)
      2.9 FEDERAL-TAX (ELEMENT)
      2.10 STATE-TAX (ELEMENT)
      2.11 FICA-TAX (ELEMENT)
      2.12 CHECK (GROUP)
          2.12.1 EMPLOYEE-NAME (GROUP)
              2.12.1.1 INITIAL (ELEMENT)
              2.12.1.2 FIRST-NAME (ELEMENT)
              2.12.1.3 SURNAME (ELEMENT)
              2.12.1.4 INITIAL (ELEMENT)
              2.12.1.5 FIRST-NAME (ELEMENT)
          2.12.2 NET-PAY (ELEMENT)
          2.12.3 CHECK-NUMBER (ELEMENT)
          2.12.4 PAY-DATE (ELEMENT)

1* 1. HOURLY-EMPLOYEE-REPORT (OUTPUT)
  1   2. EMP-REPORT-ENTRY (GROUP)
      2.1 EMPLOYEE-NAME (GROUP)
1 SALARIED-EMPLOYEE-REPORT (OUTPUT)
1 2 S-EMP-REPORT-ENTRY (GROUP)
 3 EMPLOYEE-NAME (GROUP)
   4 SURNAME (ELEMENT)
   4 INITIAL (ELEMENT)
   4 FIRST-NAME (ELEMENT)
 3 EMPLOYEE-IDENTIFICATION-NUMBER (ELEMENT)
 3 DEPARTMENT (ELEMENT)
 3 GROSS-PAY (ELEMENT)
 3 STATUS-CODE (ELEMENT)

1 HIRED-EMPLOYEE-REPORT (OUTPUT)
1 2 HIRED-REPORT-ENTRY (GROUP)
 3 EMPLOYEE-IDENTIFICATION-NUMBER (ELEMENT)
 3 EMPLOYEE-NAME (GROUP)
   4 SURNAME (ELEMENT)
   4 INITIAL (ELEMENT)
   4 FIRST-NAME (ELEMENT)
 3 SOCIAL-SECURITY-NUMBER (ELEMENT)
 3 EMPLOYMENT-STATUS (ELEMENT)
 3 EMPLOYMENT-DATE (ELEMENT)

1 TERMINATED-EMPLOYEE-REPORT (OUTPUT)
1 2 TERM-REPORT-ENTRY (GROUP)
 3 EMPLOYEE-IDENTIFICATION-NUMBER (ELEMENT)
 3 EMPLOYEE-NAME (GROUP)
   4 SURNAME (ELEMENT)
   4 INITIAL (ELEMENT)
   4 FIRST-NAME (ELEMENT)
 3 SOCIAL-SECURITY-NUMBER (ELEMENT)
 3 TERMINATION-DATE (ELEMENT)
ADDRESS (GROUP)
   4  HOUSE-NUMBER (ELEMENT)
   11  STREET (ELEMENT)
   12  APARTMENT-NUMBER (ELEMENT)
   13  CITY (ELEMENT)
   14  STATE (ELEMENT)
   15  ZIP-CODE (ELEMENT)
   16  3  EMPLOYMENT-DATE (ELEMENT)
   17  3  PAY-RANK-CODE (ELEMENT)
   18  3  CUMULATIVE-GROSS-PAY (ELEMENT)
   19  3  CUMULATIVE-TAX-DEDUCTIONS (ELEMENT)
   25  3  CUMULATIVE-FICA-DEDUCTIONS (ELEMENT)
   21  3  CUMULATIVE-HOURS (ELEMENT)
   22  3  EMPLOYMENT-STATUS (ELEMENT)
   23  3  CUMULATIVE-STATE-DEDUCTIONS (ELEMENT)
   24  3  CUMULATIVE-FEDERAL-DEDUCTIONS (ELEMENT)

2.4 INTERNALLY GENERATED DATA

1*  1 ERROR-LISTING (OUTPUT)
1  2  ERROR-LISTING-ENTRY (GROUP)
  3  EMPLOYEE-NAME (GROUP)
   3  SURNAME (ELEMENT)
   4  INITIAL (ELEMENT)
   5  FIRST-NAME (ELEMENT)
   6  3  EMPLOYEE-IDENTIFICATION-NUMBER (ELEMENT)
   7  3  SOCIAL-SECURITY-NUMBER (ELEMENT)
   8  3  ERROR-CODE (ELEMENT)
   9  2  ERROR-CODE (ELEMENT) (SEVERAL)

2.5 SYSTEM DATA CONSTRAINTS

1  MEMO
   DESCRIPTION;
   FOR THE MOST PART, INPUT AND OUTPUT FROM THE SYSTEM
   IS DIRECTLY PROPORTIONAL TO THE NUMBER OF EMPLOYEES.
   LIMITATIONS ON THE SYSTEM NEED NOT BE WORRIED ABOUT IN
   REGARDS TO SIZE AND NUMBER OF FILES.
   APPLIES TO:  EMPLOYEE-INFORMATION.

SYSTEM DATA CONSTRAINTS
USER SUPPORT FOR DATA COLLECTION

FILE:
FILE: NAME DOESN'T CONSIST OF ANYTHING - EMPLOYEE-INFORMATION
FILE: NAME DOESN'T CONSIST OF ANYTHING - PAYSYSTEM-OUTPUTS
**BASIC CONTENTS MATRIX**

The rows are the given input names.

The columns are the lowest level objects which are continued in the rows, with intermediate groups ignored.

If any columns are group names, then the definition is incomplete.

If any columns are ambiguous names, they are possible elements.

<table>
<thead>
<tr>
<th>ROW NAMES</th>
<th>COLUMN NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 EMPLOYMENT-TERMINATION-FORM INPUT</td>
<td>1 SURNAME</td>
</tr>
<tr>
<td>2 HOURLY-EMPLOYMENT-FORM INPUT</td>
<td>2 INITIAL</td>
</tr>
<tr>
<td>3 SALARIED-EMPLOYMENT-FORM INPUT</td>
<td>3 FIRST-NAME</td>
</tr>
<tr>
<td>4 TAX-SITTING-CERTIFICATE INPUT</td>
<td>4 SOCIAL-SECURITY-NUMBER</td>
</tr>
<tr>
<td>5 TITLE-CARD</td>
<td>5 TERMINATION-DATE</td>
</tr>
<tr>
<td>6 ERROR-LISTING OUTPUT</td>
<td>6 EMPLOYEE-IDENTIFICATION-NUMBER</td>
</tr>
<tr>
<td>7 HIRSD-EMPLOYEE-REPORT OUTPUT</td>
<td>7 EMPLOYMENT-STATUS</td>
</tr>
<tr>
<td>8 HOURLY-EMPLOYEE-REPORT OUTPUT</td>
<td>8 SEX</td>
</tr>
<tr>
<td>9 PAY-STATEMENT OUTPUT</td>
<td>9 BIRTHDATE</td>
</tr>
<tr>
<td>10 SALARIED-EMPLOYEE-REPORT OUTPUT</td>
<td>10 HOUSE-NUMBER</td>
</tr>
<tr>
<td>11 TERMINATED-EMPLOYEE-REPORT OUTPUT</td>
<td>11 STREET</td>
</tr>
<tr>
<td></td>
<td>12 APARTMENT-NUMBER</td>
</tr>
<tr>
<td></td>
<td>13 CITY</td>
</tr>
<tr>
<td></td>
<td>14 STATE</td>
</tr>
<tr>
<td></td>
<td>15 ZIP-CODE</td>
</tr>
<tr>
<td></td>
<td>16 PHONE</td>
</tr>
<tr>
<td></td>
<td>17 JOB-TITLE</td>
</tr>
<tr>
<td></td>
<td>18 PAY-RATE</td>
</tr>
<tr>
<td></td>
<td>19 CURRENT-DATE</td>
</tr>
<tr>
<td></td>
<td>20 EMPLOYMENT-DATE</td>
</tr>
<tr>
<td></td>
<td>21 JOB-NUMBER</td>
</tr>
<tr>
<td></td>
<td>22 PAY-RATENO-CODE</td>
</tr>
<tr>
<td></td>
<td>23 SUPERVISOR</td>
</tr>
<tr>
<td></td>
<td>24 DEPARTMENT</td>
</tr>
<tr>
<td></td>
<td>25 NUMBER-OF-DEDUCTIONS</td>
</tr>
<tr>
<td></td>
<td>26 STATUS-CODE</td>
</tr>
<tr>
<td></td>
<td>27 PAY-DATE</td>
</tr>
<tr>
<td>COLUMN NAMES</td>
<td>ELEMENT</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>OVERTIME-HOURS-WORKED</td>
<td>29</td>
</tr>
<tr>
<td>HOURS-PER-DAY</td>
<td>30</td>
</tr>
<tr>
<td>GROSS-PAY</td>
<td>31</td>
</tr>
<tr>
<td>TOTAL-HOURS</td>
<td>32</td>
</tr>
<tr>
<td>CHECK-MONTH</td>
<td>33</td>
</tr>
<tr>
<td>NET-PAY</td>
<td>34</td>
</tr>
<tr>
<td>STATE-TAX</td>
<td>35</td>
</tr>
<tr>
<td>FICA-TAX</td>
<td>36</td>
</tr>
<tr>
<td>CUMULATIVE-GROSS-PAY</td>
<td>37</td>
</tr>
<tr>
<td>CUMULATIVE-TAX-DEDUCTIONS</td>
<td>38</td>
</tr>
<tr>
<td>CUMULATIVE-FICA-DEDUCTIONS</td>
<td>39</td>
</tr>
<tr>
<td>CUMULATIVE-STATE-DEDUCTIONS</td>
<td>40</td>
</tr>
<tr>
<td>CUMULATIVE-FEDERAL-DEDUCTIONS</td>
<td>41</td>
</tr>
<tr>
<td>CUMULATIVE-STATE-HOURS</td>
<td>42</td>
</tr>
<tr>
<td>CUMULATIVE-FICA-HOURS</td>
<td>43</td>
</tr>
<tr>
<td>CUMULATIVE-STATE-DEDUCTIONS</td>
<td>44</td>
</tr>
<tr>
<td>CUMULATIVE-FEDERAL-DEDUCTIONS</td>
<td>45</td>
</tr>
</tbody>
</table>
## CONTENTS SIMILARITY MATRIX

The number in \((i,i)\) is the number of objects at the lowest level contained in row \(i\) from above.

The number in \((i,j)\) (\(i\) not equal \(j\)) is the number of objects at the lowest level in common between rows \(i\) and \(j\) from above.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

This table represents the similarity matrix for the given contents, where each entry \((i,j)\) indicates the number of objects in common between row \(i\) and row \(j\) at the lowest level.
<table>
<thead>
<tr>
<th>ROW#</th>
<th>NAME</th>
<th>IS A SUBSET OF</th>
<th>ROW#</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EMPLOYMENT-TERMINATION-FORM</td>
<td>11 TERMINATED-EMPLOYEE-REPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SUFFICIENT-EMPLOYMENT-FORM</td>
<td>2 HOURLY-EMPLOYMENT-FORM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MILD-EMPLOYEE-REPORT</td>
<td>11 TERMINATED-EMPLOYEE-REPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SALARIED-EMPLOYEE-REPORT</td>
<td>8 HOURLY-EMPLOYEE-REPORT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In regards to data flow, the rows are data names, the columns are process names.

<table>
<thead>
<tr>
<th>Row Names</th>
<th>Column Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 EMPLOYEE-INFORMATION</td>
<td>1 PAYROLL-PROCESSING</td>
</tr>
<tr>
<td>2 EMPLOYMENT-TERMINATION-FORM</td>
<td>2 TERMINATING-EMP-PROCESSING</td>
</tr>
<tr>
<td>3 HOURLY-EMPLOYMENT-FORM</td>
<td>3 HIRE-REPORT-ENTRY-GENERATION</td>
</tr>
<tr>
<td>4 SALARIED-EMPLOYMENT-FORM</td>
<td>4 NEW-EMPLOYEE-PROCESSING</td>
</tr>
<tr>
<td>5 TAX-WITHHOLDING-CERTIFICATE</td>
<td>5 HOURLY-RECORD-CREATION</td>
</tr>
<tr>
<td>6 TIME-CARD</td>
<td>6 SALARIED-RECORD-CREATION</td>
</tr>
<tr>
<td>7 ERROR-LISTING</td>
<td>7 HOURLY-EMPLOYEE-PROCESSING</td>
</tr>
<tr>
<td>8 HIRE-EMPLOYEE-REPORT</td>
<td>8 SALARIED-EMPLOYEE-PROCESSING</td>
</tr>
<tr>
<td>9 HOURLY-EMPLOYEE-REPORT</td>
<td></td>
</tr>
<tr>
<td>10 PAY-STATEMENT</td>
<td></td>
</tr>
<tr>
<td>11 PAYSYSTEM-OUTPUTS</td>
<td></td>
</tr>
<tr>
<td>12 SALARIED-EMPLOYEE-REPORT</td>
<td></td>
</tr>
<tr>
<td>13 TERMINATED-EMPLOYEE-REPORT</td>
<td></td>
</tr>
</tbody>
</table>
### Interaction Matrix

<table>
<thead>
<tr>
<th>(i, j) value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>Row i is received or used by column j (input)</td>
</tr>
<tr>
<td>U</td>
<td>Row i is updated by column j</td>
</tr>
<tr>
<td>O</td>
<td>Row i is derived or generated by column j (output)</td>
</tr>
<tr>
<td>A</td>
<td>Row i is input to, updated by, and output of column j (all)</td>
</tr>
<tr>
<td>F</td>
<td>Row i is input to and output of column j (flow)</td>
</tr>
<tr>
<td>1</td>
<td>Row i is input to and updated by column j</td>
</tr>
<tr>
<td>2</td>
<td>Row i is updated by and output of column j</td>
</tr>
</tbody>
</table>

#### Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>R</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>R</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>I</td>
<td>R</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>8</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>11</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>I</td>
<td>I</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### User Support for Data Collection
<table>
<thead>
<tr>
<th>PROCESSES</th>
<th>(COLUMN 3) Uses data, but does not derive or update anything</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIRE-RECORD-ENTRY-GENERATION</td>
<td>(COLUMN 5) Uses data, but does not derive or update anything</td>
</tr>
<tr>
<td>ROUBLE-RECORD-CREATION</td>
<td>(COLUMN 6) Uses data, but does not derive or update anything</td>
</tr>
<tr>
<td>SALARY-RECORD-CREATION</td>
<td>(COLUMN 8) Derives something, but does not use anything</td>
</tr>
<tr>
<td>SALARY-EMPLOYEE-PROCESSING</td>
<td></td>
</tr>
</tbody>
</table>

**USER SUPPORT FOR DATA COLLECTION**
The rows and columns are process names from above. An asterisk in (i,j) means that something derived or updated by process i is used by process j.

*** MATRIX EMPTY FOR ROWS 1 THRU 8 AND COLUMNS 1 THRU 8
<table>
<thead>
<tr>
<th>Process</th>
<th>Interaction Matrix Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-</td>
<td>(ROW/COL 1) NO INTERACTION,</td>
</tr>
<tr>
<td>Collection Processing</td>
<td>BUT HAS SUBPARTS</td>
</tr>
<tr>
<td>Terminus-Entry-Generation</td>
<td>(ROW/COL 2) NO INTERACTION,</td>
</tr>
<tr>
<td>(ROW/COL 3) NO INTERACTION,</td>
<td>BUT HAS SUBPARTS AND IS PART OF ANOTHER PROCESS</td>
</tr>
<tr>
<td>(ROW/COL 4) NO INTERACTION,</td>
<td>BUT HAS SUBPARTS AND IS PART OF ANOTHER PROCESS</td>
</tr>
<tr>
<td>(ROW/COL 5) NO INTERACTION,</td>
<td>BUT HAS SUBPARTS AND IS PART OF ANOTHER PROCESS</td>
</tr>
<tr>
<td>(ROW/COL 6) NO INTERACTION,</td>
<td>BUT HAS SUBPARTS AND IS PART OF ANOTHER PROCESS</td>
</tr>
<tr>
<td>(ROW/COL 7) NO INTERACTION,</td>
<td>BUT HAS SUBPARTS AND IS PART OF ANOTHER PROCESS</td>
</tr>
<tr>
<td>(ROW/COL 8) NO INTERACTION,</td>
<td>BUT HAS SUBPARTS AND IS PART OF ANOTHER PROCESS</td>
</tr>
</tbody>
</table>

**USER SUPPORT FOR DATA COLLECTION**

470
We refer to the given input names.
The columns are the lowest level objects which are contained in the rows, with intermediate groups ignored.

If any columns are group names, then the definition is incomplete.

If any columns are ambiguous names, they are possible elements.

<table>
<thead>
<tr>
<th>ROW NAMES</th>
<th>COLUMN NAMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 DEPARTMENT-RECORD</td>
<td>1 DEPARTMENT</td>
</tr>
<tr>
<td>2 HOURLY-EMPLOYEE-RECORD</td>
<td>2 SUPERVISOR</td>
</tr>
<tr>
<td>3 SALARIED-EMPLOYEE-RECORD</td>
<td>3 NUMBER-OF-EMPLOYEES</td>
</tr>
<tr>
<td></td>
<td>4 TOTAL-BUDGET</td>
</tr>
<tr>
<td></td>
<td>5 REMAINING-FUNDS</td>
</tr>
<tr>
<td></td>
<td>6 SURNAME</td>
</tr>
<tr>
<td></td>
<td>7 INITIAL</td>
</tr>
<tr>
<td></td>
<td>8 FIRST-NAME</td>
</tr>
<tr>
<td></td>
<td>9 EMPLOYEE-IDENTIFICATION-NUMBER</td>
</tr>
<tr>
<td></td>
<td>10 SOCIAL-SECURITY-NUMBER</td>
</tr>
<tr>
<td></td>
<td>11 PAY-GRADE-CODE</td>
</tr>
<tr>
<td></td>
<td>12 HOUSE-NUMBER</td>
</tr>
<tr>
<td></td>
<td>13 STREET</td>
</tr>
<tr>
<td></td>
<td>14 APARTMENT-NUMBER</td>
</tr>
<tr>
<td></td>
<td>15 CITY</td>
</tr>
<tr>
<td></td>
<td>16 STATE</td>
</tr>
<tr>
<td></td>
<td>17 ZIP-CODE</td>
</tr>
<tr>
<td></td>
<td>18 PHONE</td>
</tr>
<tr>
<td></td>
<td>19 EMPLOYMENT-DATE</td>
</tr>
<tr>
<td></td>
<td>20 NUMBER-OF-DEDUCTIONS</td>
</tr>
<tr>
<td></td>
<td>21 CUMULATIVE-GROSS-PAY</td>
</tr>
<tr>
<td></td>
<td>22 CUMULATIVE-FEDERAL-DEDUCTIONS</td>
</tr>
<tr>
<td></td>
<td>23 CUMULATIVE-STATE-DEDUCTIONS</td>
</tr>
<tr>
<td></td>
<td>24 CUMULATIVE-FICA-DEDUCTIONS</td>
</tr>
<tr>
<td></td>
<td>25 AGE</td>
</tr>
<tr>
<td></td>
<td>26 SEX</td>
</tr>
</tbody>
</table>

USER SUPPORT FOR DATA COLLECTION
Ann: a (i, j) means that column j is contained
directly or indirectly in row i. The columns
do not consist of anything further. Intermediate
groups are ignored.

```
1 1111111112 22222222
1234567890 1234567890 12345678
```

```
+----------+----------+----------+
1 |I*****| I | I | I |
2 | I* | *****I**********I**********I |
3 | I* | *****I**********I**********I |
+----------+----------+----------+
```
the number of objects
in level contained in row 1 from above.

The cell in (i, j) (i not equal j) is
the number of objects at the lowest level in
cells between rows i and j from above.

```
    1  2  3
+--------+
1 5 1 1
2 24 23
3 23
+--------+
```
1 INPUT
SYNONYMS ARE: S-EMP-FORM;
DESCRIPTION:
THIS INPUT CONTAINS THE INFORMATION NECESSARY TO
ADD A NEW SALARIED EMPLOYEE TO THE PAYROLL;
KEYWORDS: NEW;
ATTRIBUTES ARE:
ARRIVAL-TYPE RANDOM;
PART OF: EMPLOYEE-INFORMATION;
CONSISTS OF:
PERSONAL-DATA,
SALARIED-JOB-DATA;
GENERATED BY: PAYROLL-DEPARTMENT;
RECEIVED BY: NEW-EMPLOYEE-PROCESSING;
USED BY: NEW-EMPLOYEE-PROCESSING,
SALARIED-RECORD-CREATATION;
USED BY:
HIRE-REPORT-ENTRY-GENERATION
TO DERIVE HIRED-REPORT-ENTRY;
HAPPENS:
SEVERAL TIMES-PER MONTH;

24 EOF
EOF
EOF
EOF
EOF
1 INPUT
SYNONYMS ARE: H-EMP-FORM;
DESCRIPTION:
THIS INPUT CONTAINS THE INFORMATION NECESSARY TO
ADD A NEW HOURLY EMPLOYEE TO THE PAYROLL;
KEYWORDS: NEW;
ATTRIBUTES ARE:
ARRIVAL-TYPE RANDOM;
PART OF: EMPLOYEE-INFORMATION;
CONSISTS OF:
PERSONAL-DATA,
DATA COLLECTION REQUIREMENTS AND SCOPE

1. Input

   SYNONYMS ARE: TAX-CERT;
   DESCRIPTION:
   THIS INPUT CONTAINS TAX INFORMATON NECESSARY TO
   COMPUTE THE EMPLOYEE'S PAYCHECK;
   KEYWORDS:
   NEW;
   ATTRIBUTES ARE:
   ARRIVAL-TYPE RANDOM;
   PART OF: EMPLOYEE-INFORMATION;
   CONSISTS OF:
   EMPLOYEE-NAME,
   ADDRESS,
   SOCIAL-SECURITY-NUMBER,
   NUMBER-OF-Deductions,
   CURRENT-DATE;
   GENERATED BY: EMPLOYEE;
   RECEIVED BY: NEW-EMPLOYEE-PROCESSING;
   USED BY: NEW-EMPLOYEE-PROCESSING;
   HAPPENS:
   SEVERAL TIMES-PER MONTH;

2. Input

   SYNONYMS ARE: TERM-FORM;
   DESCRIPTION:
   THIS INPUT CONTAINS THE INFORMATION NECESSARY TO
   DELETE AN EMPLOYEE FROM THE PAYROLL;
   KEYWORDS: TERM-KEY;
   ATTRIBUTES ARE:

EMPLOYMENT-TERMINATION-FORM;
INFORMATION TO BE COLLECTED BY THE USER:

INPUT TIME-CARD:
SYNONYMS ARE: T-CARD;
DESCRIPTION: THIS INPUT CONTAINS THE INFORMATION ABOUT THE HOURS THAT AN HOURLY EMPLOYEE WORKED THE PRECEDING WEEK;
SEE-MEMO: I-O-CONSTRAINTS-MEMO;
KEYWORDS: HOURLY;
ATTRIBUTES ARE:
ARRIVAL-TYPE SCHEDULED;
PART OF: EMPLOYEE-INFORMATION;
CONSISTS OF:

- EMPLOYEE-NAME,
- SOCIAL-SECURITY-NUMBER,
- STATUS-CODE,
- PAY-DATE,
- REGULAR-HOURS-WORKED,
- OVERTIME-HOURS-WORKED,
- HOURS-PER-DAY,
- EMPLOYEE-IDENTIFICATION-NUMBER;

GENERATED BY: EMPLOYEE;
RECEIVED BY: HOURLY-EMPLOYEE-PROCESSING;
USED BY: HOURLY-EMPLOYEE-PROCESSING;
HAPPENS: SEVERAL TIMES-PER MONTH;
ELEMENTARY INFORMATION:

A. INPUT SOURCE(S) OF THE DATA ELEMENT

1 INTERFACE
SYNONYMS ARE: DEPT-EMP, R1;
DESCRIPTION;
THIS IS THE ENTITY WHICH WILL RECEIVE ALL THE OUTPUTS AND
SUPPLY ALL THE INPUTS. ;
SEE-MEMO: OBJECTIVES-MEMO;
SUBPARTS ARE: EMPLOYEE,
PAYROLL-DEPARTMENT;
GENERATES: EMPLOYEE-INFORMATION;
RECEIVES: PAYSYSTEM-OUTPUTS;
RESPONSIBLE-PROBLEM-DEFINER IS:
MICHIEL-J-BASTARACHE;

EOF EOF EOF EOF EOF
1 INTERFACE
SYNONYMS ARE: EMP;
DESCRIPTION;
AN EMPLOYEE IS IDENTIFIED BY AN EMPLOYEE NUMBER;
PART OF: DEPARTMENTS-AND-EMPLOYEES;
GENERATES: TIME-CARD,
TAX-WITHHOLDING-CERTIFICATE;
RECEIVES: PAY-STATEMENT;

EOF EOF EOF EOF EOF
1 INTERFACE
SYNONYMS ARE: PAY-DEPT;
DESCRIPTION;
THIS DEPARTMENT IS RESPONSIBLE FOR ALL PAYROLL DATA;
THIS DEPARTMENT IS ALSO THE SPONSOR OF THIS PROJECT
PART OF: DEPARTMENTS-AND-EMPLOYEES;
GENERATES: EMPLOYMENT-TERMINATION-FORM,

DATA COLLECTION REQUIREMENTS AND SCOPE
B. Recipients

1 INTERFACE ACCOUNTING-SYSTEMS;
2 DESCRIPTION;
3 THE ORGANIZATION HAS VARIOUS ACCOUNTING SYSTEMS THAT
4 USE THE INFORMATION GENERATED BY THE PAYROLL SYSTEM.
5 INTERNAL AUDITING ALSO RECEIVES THE PAYROLL INFORMATION.
6 ;
7 SUBPARTS ARE: PAYROLL-SYSTEM;
8 RECEIVES: PAYSYSTEM-OUTPUTS;
9
10 EOF EOF EOF EOF EOF

D. Critical Values

1 ELEMENT REMAINING-FUNDS;
2 DESCRIPTION;
3 CURRENT AMOUNT OF FUNDS IN DEPARTMENT.
4 EACH INDIVIDUAL DEPARTMENT MUST HAVE A SUFFICIENT
5 BALANCE TO MEET ITS PAYROLL REQUIREMENTS.
6 ;
7 KEYWORDS: SHARED-KEY;
8 CONTAINED IN: DEPARTMENT-RECORD,
9 DEPARTMENT-UPDATE-DATA;
10 UPDATED BY: FUNDS-UPDATE
11 USING: DEPARTMENT,
12 GROSS-PAY;
13
14 EOF EOF EOF EOF EOF
1.

**DAILY EMPLOYEE-REPORT**

**DESCRIPTION:**
1. This is an administrative record of all hourly employees paid in one week.
2. 

**SALARIED-EMPLOYEE-REPORT**

**DESCRIPTION:**
1. This is an administrative record of all salaried employees paid in one month.
2. 

**F. FREQUENCY OF UPDATE**

**INTERVAL: MONTH**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>TIMES HAPPENS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYMENT-TERMINATION-FORM</td>
<td>INPUT</td>
<td>SEVERAL</td>
</tr>
<tr>
<td>TAX-WITHOLDING-CERTIFICATE</td>
<td>INPUT</td>
<td>SEVERAL</td>
</tr>
<tr>
<td>SALARIED-PAYCHECK-PROD-INIT</td>
<td>EVENT</td>
<td>ONE</td>
</tr>
<tr>
<td>SALARIED-EMPLOYEE-FORM</td>
<td>INPUT</td>
<td>SEVERAL</td>
</tr>
<tr>
<td>SALARIED-EMPLOYEE-REPORT</td>
<td>OUTPUT</td>
<td>ONE</td>
</tr>
<tr>
<td>SALARIED-EMPLOYEE-PROCESSING</td>
<td>PROCESS</td>
<td>ONE</td>
</tr>
<tr>
<td>SALARIED-EMP-PROCESSING-INIT</td>
<td>EVENT</td>
<td>ONE</td>
</tr>
<tr>
<td>PAYROLL-PROCESSING</td>
<td>PROCESS</td>
<td>NO-OF-PAYROLL-PROCESSING</td>
</tr>
<tr>
<td>HOURLY-EMPLOYMENT-FORM</td>
<td>INPUT</td>
<td>SEVERAL</td>
</tr>
<tr>
<td>ERROR-LISTING</td>
<td>OUTPUT</td>
<td>NO-OF-PAYROLL-PROCESSING</td>
</tr>
</tbody>
</table>

**INTERVAL: WEEK**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>TIMES HAPPENS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIRED-EMPLOYEE-REPORT</td>
<td>OUTPUT</td>
<td>ONE</td>
</tr>
<tr>
<td>TIME-CARD</td>
<td>INPUT</td>
<td>NO-OF-HOURLY-EMPLOYEES</td>
</tr>
<tr>
<td>TERMINATION-PROCESSING-INIT</td>
<td>EVENT</td>
<td>ONE</td>
</tr>
<tr>
<td>TERMINATING-EMP-PROCESSING</td>
<td>PROCESS</td>
<td>ONE</td>
</tr>
<tr>
<td>TERMINATED-EMPLOYEE-REPORT</td>
<td>OUTPUT</td>
<td>ONE</td>
</tr>
<tr>
<td>NEW-EMPLOYEE-PROCESSING-INIT</td>
<td>EVENT</td>
<td>ONE</td>
</tr>
<tr>
<td>NEW-EMPLOYEE-PROCESSING</td>
<td>PROCESS</td>
<td>ONE</td>
</tr>
<tr>
<td>MOM-PAYCHECK-PROD-INIT</td>
<td>EVENT</td>
<td>ONE</td>
</tr>
<tr>
<td>HOURLY-PAYCHECK-PROD-INIT</td>
<td>OUTPUT</td>
<td>ONE</td>
</tr>
</tbody>
</table>

**DATA COLLECTION REQUIREMENTS AND SCOPE**

481
3.3 DATA COLLECTION AND TRANSFER PROCEDURES

This topic has been discussed above (Section 3.1.6).

3.4 DATA BASE IMPACTS

1* DEPARTMENT-FILE
DESCRIPTION;
1
2
3

DEPARTMENT-FILE
DESCRIPTION;
THIS FILE CONSISTS OF ALL EMPLOYEE RECORDS FOR A
GIVEN DEPARTMENT. ONE FILE IS AVAILABLE FOR EACH
DEPARTMENT.

2* DEPARTMENT-FILE
VOLATILITY-SET;
1
2
3

DEPARTMENT-FILE
VOLATILITY-SET;
THIS FILE IS CHANGED WHENEVER A NEW EMPLOYEE IS HIRED
INTO THE DEPARTMENT OR EMPLOYEE LEAVES THE DEPARTMENT.
THIS HAPPENS ABOUT FOUR TIMES A MONTH.

3* DEPARTMENT-FILE
DERIVATION;
1
2
3

DEPARTMENT-FILE
DERIVATION;
NEW-EMPLOYEE-PROCESSING ADDS MEMBERS TO THIS SET
AND TERMINATING-EMPLOYEE-PROCESSING DELETES MEMBERS.
(MEMBERS IN THIS CASE ARE EMPLOYEE RECORDS.)

1* HOURLY-EMPLOYEE-FILE
DESCRIPTION;
1

HOURLY-EMPLOYEE-FILE
DESCRIPTION;
THIS FILE CONSISTS OF ALL HOURLY EMPLOYEE RECORDS.

2* HOURLY-EMPLOYEE-FILE
VOLATILITY-MEMBER;
1

HOURLY-EMPLOYEE-FILE
VOLATILITY-MEMBER;
MEMBERS OF THIS SET ARE MODIFIED ABOUT ONCE A WEEK.
NON-EMPLOYEE-PROCESSING ADDS MEMBERS TO THIS SET
AND TERMINATING-EMPLOYEE-PROCESSING DELETES MEMBERS.
(MEMBERS IN THIS CASE ARE EMPLOYEE RECORDS.):

1. Salaried-Employee-File
   DESCRIPTION;
   THIS FILE CONSISTS OF ALL SALARIED EMPLOYEE RECORDS.

2. Salaried-Employee-File
   Volatility-Member;
   MEMBERS OF THIS SET ARE MODIFIED ABOUT ONCE A MONTH.

3. Salaried-Employee-File
   Derivation;
   NEW-EMPLOYEE-PROCESSING ADDS MEMBERS TO THIS SET
   AND TERMINATING-EMPLOYEE-PROCESSING DELETES MEMBERS.
   (MEMBERS IN THIS CASE ARE EMPLOYEE RECORDS.)
APPENDIX H

Execution of the Documentation Generator in a Specific Machine Environment

Using Multics, the Automated Documentation System can run using the following command:

```
exec-com >udd>project>personid*spg schema-segment source-segment
```

This command will do all syntax checking needed and generate the document. Output for the document will go to the file spg.output. The system assumes the Analyzer data base is in the file psadb.dbf unless the SET command is used to change the default (i.e., SET DB=

Output from this section looks as follows:

<table>
<thead>
<tr>
<th>LINE</th>
<th>TEXT</th>
<th>CMNDS</th>
<th>SECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>&amp;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>&amp; A SMALL TEST EXAMPLE &amp;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1. SCOPE</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2</td>
<td>1.1 IDENTIFICATION</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1.2 FUNCTIONAL SUMMARY</td>
</tr>
<tr>
<td>8</td>
<td><em>MISSING</em></td>
<td></td>
<td>1.3</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>1.4</td>
</tr>
</tbody>
</table>

*** INVALID SECTION IDENTIFIER

<table>
<thead>
<tr>
<th>LINE</th>
<th>TEXT</th>
<th>CMNDS</th>
<th>SECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>&amp;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>2. APPLICABLE DOCUMENTS</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>2</td>
<td>2.1 GOVERNMENT DOCUMENTS</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>1</td>
<td>2.2 NON-GOVERNMENT DOCUMENTS</td>
</tr>
</tbody>
</table>

SUMMARY

8 SECTIONS
1 OF WHICH ARE MISSING FROM THE SOURCE FILE.
18 TOTAL TEXT LINES, AND
6 TOTAL COMMAND LINES.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>analyst</td>
<td>Name used synonymously for &quot;problem definer.&quot; One who aids to develop the problem statement or logical system design.</td>
</tr>
<tr>
<td>Analyzer</td>
<td>Synonym for &quot;URA&quot;. Is the software package that processes problem stated in the Language.</td>
</tr>
<tr>
<td>comment entry</td>
<td>The text associated with a comment entry statement. DESCRIPTION, PROCEDURE and VOLATILITY are examples of statements which are specified by comment entries.</td>
</tr>
<tr>
<td>control commands</td>
<td>Those URA commands which allow the URA user to pass certain control information to the Analyzer. They are particular to an individual operating system.</td>
</tr>
<tr>
<td>conversational mode</td>
<td>Interactive use of the computer system through a terminal device. Used synonymously with on-line terminal, or interactive mode.</td>
</tr>
<tr>
<td>data base</td>
<td>The data base referred to throughout this paper is the user's data base which is populated by URA from the user inputted URL statements.</td>
</tr>
<tr>
<td>data object</td>
<td>Any URA name type that represents some form of data. SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS are all data objects described by URL.</td>
</tr>
<tr>
<td>fdname</td>
<td>Any legal file or device name. Allowable names are dependent on the operating system being used.</td>
</tr>
<tr>
<td>filename</td>
<td>Any legal temporary or permanent file name that is to be specified by the user.</td>
</tr>
<tr>
<td>input file</td>
<td>Any temporary or permanent file which contains data to be used by URA commands via INPUT or FILE parameters.</td>
</tr>
<tr>
<td>logical description</td>
<td>Synonym to &quot;problem statement.&quot; Set of requirements for a new system.</td>
</tr>
<tr>
<td>logical system design</td>
<td>The process of specifying a problem statement for any particular system.</td>
</tr>
</tbody>
</table>
modifier commands: Commands which modify the contents of a URA data base in the manner specified by the user.

Multics Command: A command to the Multics system; and element of the Multics command language.

Multics Command Language: The set of commands, subcommands and operands recognized by the Multics Operating System.

name type: Any of the many types of names allowed by URL (i.e., PROCESS, SET, GROUP, etc.). See "The User Requirements Language, Language Reference Manual"**, Appendix E, for a list of all possible name types.

physical system design: The process of specifying a physical system (consisting of software, machinery, etc.) given a particular problem statement.

physical system designer: Person responsible for deriving a physical system design form the problem statement generated from the logical system design process.

problem definer: Used synonymously with "analyst." That person who develops the requirements stated by the users into a format understandable by others and in sufficient detail to be usable by the physical system designer. The product of his work is the problem statement.

problem statement: A set of requirements specified by users of a proposed system and expressed by the problem definer into a format acceptable by the organization.

prompt: A system function that requests the terminal user to supply operands necessary to continue processing.

PUNCH file: A file which contains data (usually URL user-names) in a format that can be used as input to one or more URA commands.

report commands: Those commands which retrieve data from a URA data base and output it in some meaningful format.

segment: A named collection of data which is accessible by the system. The data set usually resides on an auxiliary storage device.
term

Identifier to designate that the terminal is to be used as a source of input or area for output.

undefined name

A name of a URL object that has been entered into the URA data base, but has no name type associated with it.

URA

The User Requirements Analyzer. A software package which stores information in the URA data base by interpreting URL statements given as input, and retrieves information from the data base in the form of reports.

URA command

Any of the commands that can be used to operate URA. See "User Requirements Analyzer Command Descriptions"* for complete descriptions about each command available in URA.

URA data base

File where URL information is stored (in a coded format) which can then be accessed by URA commands.

URL


URL statement

A statement specified by "The User Requirements Language, Language Reference Manual"**. Each statement may define a URL object, define a comment entry, or define a relationship among two or more URL objects.

user-name

Any legal URL name specified by problem definer. Also called a "user defined name."

* Part IV.
** ISDOS Working Paper No. 68.