NASIS DATA BASE MANAGEMENT SYSTEM - IBM 360 TSS IMPLEMENTATION

III - DATA SET SPECIFICATIONS

NEOTERICS, INC.

prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

NASA Lewis Research Center
Contract NAS 3-14979
NOTICE

THIS DOCUMENT HAS BEEN REPRODUCED FROM THE BEST COPY FURNISHED US BY THE SPONSORING AGENCY. ALTHOUGH IT IS RECOGNIZED THAT CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED IN THE INTEREST OF MAKING AVAILABLE AS MUCH INFORMATION AS POSSIBLE.
# Table of Contents

## Topic B - Data Base Executive

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1</td>
<td>DEPL/I Diagnostics</td>
<td>4</td>
</tr>
<tr>
<td>B.2</td>
<td>DBPL/I-DEPAC Interface</td>
<td>10</td>
</tr>
<tr>
<td>B.3</td>
<td>DBPAC Error Codes</td>
<td>13</td>
</tr>
<tr>
<td>B.4</td>
<td>Mainline File Control Block</td>
<td>19</td>
</tr>
<tr>
<td>B.5</td>
<td>List Structure</td>
<td>23</td>
</tr>
<tr>
<td>B.6</td>
<td>List Error Control Block</td>
<td>25</td>
</tr>
<tr>
<td>B.7</td>
<td>Descriptor Descriptor File</td>
<td>27</td>
</tr>
<tr>
<td>B.8</td>
<td>DEPL/I-DEBLIST Interface</td>
<td>55</td>
</tr>
</tbody>
</table>

## Topic C - Utilities

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1</td>
<td>NASSIS.USERIDS</td>
<td>57</td>
</tr>
<tr>
<td>C.2</td>
<td>NASSIS.JOINIDS</td>
<td>58</td>
</tr>
<tr>
<td>C.3</td>
<td>EDIT.LISRMLF</td>
<td>59</td>
</tr>
</tbody>
</table>

## Topic D - Maintenance

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.1</td>
<td>REDLOAD Error Codes Table</td>
<td>60</td>
</tr>
<tr>
<td>D.2</td>
<td>TRANSCT Descriptors</td>
<td>61</td>
</tr>
<tr>
<td>D.3</td>
<td>CORRECT Data Display Format</td>
<td>63</td>
</tr>
<tr>
<td>D.4</td>
<td>RELICAD Error Data Set</td>
<td>65</td>
</tr>
<tr>
<td>D.5</td>
<td>Inverted Index Format</td>
<td>66</td>
</tr>
<tr>
<td>D.6</td>
<td>Descriptor Editor Data Display Format</td>
<td>68</td>
</tr>
<tr>
<td>D.7</td>
<td>Descriptor Editor Field Name Display Format</td>
<td>70</td>
</tr>
<tr>
<td>D.8</td>
<td>REDLOAD Input Data Set</td>
<td>72</td>
</tr>
<tr>
<td>D.9</td>
<td>Descriptor Editor Listing Format</td>
<td>73</td>
</tr>
<tr>
<td>D.10</td>
<td>INVERT Restart File</td>
<td>75</td>
</tr>
<tr>
<td>D.11</td>
<td>INVERT SORTIN File</td>
<td>76</td>
</tr>
<tr>
<td>D.12</td>
<td>INVERT SORTCUT File</td>
<td>77</td>
</tr>
<tr>
<td>D.13</td>
<td>INVERT PLEX File</td>
<td>78</td>
</tr>
<tr>
<td>D.14</td>
<td>INVERT RANGE File</td>
<td>79</td>
</tr>
<tr>
<td>D.15</td>
<td>Descriptor Editor Checkpoint</td>
<td>80</td>
</tr>
<tr>
<td>D.16</td>
<td>MERGE INDEX File</td>
<td>82</td>
</tr>
<tr>
<td>D.17</td>
<td>Descriptor Editor REVIEW Display Format</td>
<td>83</td>
</tr>
<tr>
<td>D.18</td>
<td>Descriptor Editor DEFIELD Structure</td>
<td>86</td>
</tr>
<tr>
<td>D.19</td>
<td>Descriptor Editor DESECUR</td>
<td>88</td>
</tr>
<tr>
<td>D.20</td>
<td>Descriptor Editor DESECUR Structure</td>
<td>90</td>
</tr>
<tr>
<td>D.21</td>
<td>Descriptor Editor DESUPER Structure</td>
<td>92</td>
</tr>
<tr>
<td>D.22</td>
<td>Descriptor Editor DEVALID Structure</td>
<td>94</td>
</tr>
<tr>
<td>D.23</td>
<td>Descriptor Editor DEPLD Structure</td>
<td>96</td>
</tr>
<tr>
<td>D.24</td>
<td>Descriptor Editor DEXINIT</td>
<td>99</td>
</tr>
<tr>
<td>D.25</td>
<td>Descriptor Editor DEX Structure</td>
<td>100</td>
</tr>
<tr>
<td>D.26</td>
<td>Descriptor Editor DEHDR Structure</td>
<td>109</td>
</tr>
</tbody>
</table>

## Topic E - Terminal Support

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.1</td>
<td>TSPL/I Diagnostics</td>
<td>111</td>
</tr>
<tr>
<td>E.2</td>
<td>Terminal Control Block</td>
<td>114</td>
</tr>
<tr>
<td>Topic</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>E.3</td>
<td>TTEXT-TCB Declaration</td>
<td>118</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td><strong>DATA RETRIEVAL</strong></td>
<td></td>
</tr>
<tr>
<td>F.1</td>
<td>RETDATA - Retrieval Data Table</td>
<td>122</td>
</tr>
<tr>
<td>F.2</td>
<td>EXPAND Display Format</td>
<td>124</td>
</tr>
<tr>
<td>F.3</td>
<td>SELECT Display Format</td>
<td>126</td>
</tr>
<tr>
<td>F.4</td>
<td>DISPLAY Display Format</td>
<td>128</td>
</tr>
<tr>
<td>F.5</td>
<td>PARSED Table</td>
<td>132</td>
</tr>
<tr>
<td>F.6</td>
<td>SETS Display Format</td>
<td>136</td>
</tr>
<tr>
<td>F.7</td>
<td>EXECUTE Display Format</td>
<td>138</td>
</tr>
<tr>
<td>F.8</td>
<td>PRINT Data Set Format</td>
<td>140</td>
</tr>
<tr>
<td>F.9</td>
<td>EXPTAB - Expand Term Table</td>
<td>145</td>
</tr>
<tr>
<td>F.10</td>
<td>FLDTAB - Field Name Table</td>
<td>147</td>
</tr>
<tr>
<td>F.11</td>
<td>FORMATS Display Format</td>
<td>152</td>
</tr>
<tr>
<td>F.12</td>
<td>SETAE - Sets Table</td>
<td>153</td>
</tr>
<tr>
<td>F.13</td>
<td>USERTAB - User Data Table</td>
<td>155</td>
</tr>
<tr>
<td>F.14</td>
<td>EXPLAIN Display Format</td>
<td>158</td>
</tr>
<tr>
<td>F.15</td>
<td>GFIELDS Display Format</td>
<td>159</td>
</tr>
<tr>
<td>F.16</td>
<td>SEC_FORM - Sequential Format Table</td>
<td>160</td>
</tr>
<tr>
<td>F.17</td>
<td>NASISID.STRATEGY.DATASET</td>
<td>162</td>
</tr>
<tr>
<td>F.18</td>
<td>SRCFTAB - Linear Search Table</td>
<td>164</td>
</tr>
<tr>
<td>F.19</td>
<td>COL_FORM - Columnar Format Table</td>
<td>169</td>
</tr>
<tr>
<td>F.20</td>
<td>FIELDS Display Format</td>
<td>171</td>
</tr>
<tr>
<td>F.21</td>
<td>LIMIT Table</td>
<td>172</td>
</tr>
<tr>
<td>F.22</td>
<td>LIMIT Display Format</td>
<td>174</td>
</tr>
<tr>
<td><strong>G</strong></td>
<td><strong>USAGE STATISTICS</strong></td>
<td></td>
</tr>
<tr>
<td>G.1</td>
<td>STATIC Descriptors</td>
<td>176</td>
</tr>
<tr>
<td>G.2</td>
<td>Maintenance Report Format</td>
<td>186</td>
</tr>
<tr>
<td>G.3</td>
<td>Retrieval Report Format</td>
<td>188</td>
</tr>
<tr>
<td>G.4</td>
<td>Snapshot Report Format</td>
<td>190</td>
</tr>
<tr>
<td><strong>H</strong></td>
<td><strong>IMMEDIATE COMMANDS</strong></td>
<td></td>
</tr>
<tr>
<td>H.1</td>
<td>NASIS Message File</td>
<td>192</td>
</tr>
<tr>
<td>H.2</td>
<td>Strategy Data Set</td>
<td>193</td>
</tr>
<tr>
<td>H.3</td>
<td>Strategy Display Format</td>
<td>194</td>
</tr>
<tr>
<td>H.4</td>
<td>Strategy Names Display Format</td>
<td>195</td>
</tr>
<tr>
<td>H.5</td>
<td>User Profile Table</td>
<td>196</td>
</tr>
<tr>
<td>H.6</td>
<td>User Profile Data Set</td>
<td>197</td>
</tr>
<tr>
<td>H.7</td>
<td>VERETAE - Command Table</td>
<td>198</td>
</tr>
</tbody>
</table>
TOPIC B.1 - DATA BASE EXECUTIVE

A. DATA SET NAME:
   DBPL/I Diagnostics

E. CREATED BY:
   DB Preprocessor Function

C. TYPE OF FILE:
   (4) Table

D. ORGANIZATION:
   Keyed List

E. KEY IDENTIFIER (CONTROL FIELD):
   Each diagnostic comment has a five-character identification key having the form: DBnnn, where nnn is a unique identification number.

F. RECORD LENGTH:
   Variable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   DBPL/I Diagnostic Comments are generated into mainline source programs by the DB preprocessor function (see Section IV, Topic E.1 of the DWB).

I. DBPL/I DIAGNOSTIC COMMENTS:
   
   DB001  INITIALIZATION COMPLETE.
   Informative - the % INCLUDE DB statement has been successfully processed.
DB002 'DB' FOUND WITHOUT ARGUMENT. IT IS A RESERVED IDENTIFIER.

Severe error - a DB preprocessor function reference has no parenthesized argument.

DB003 MISSING LEFT PARENTHESIS.

Severe error - a DB preprocessor function reference does not begin with double left parentheses. Processing of this DB reference was abandoned because the closing right parenthesis would not be able to be found.

DB004 ARGUMENT ABANDONED. TOO MANY DBPL/I ERRORS.

Error - more than four errors have been noted from one DB preprocessor function reference so it is being abandoned. This diagnostic may arise when the right parenthesis are missing at the end of the argument (PL/I passes the remainder of the source program to the DB function).

DB005 EXTRANEOUS TEXT IGNORED.

Error - if this message immediately follows DB009, then the statement has been processed properly but additional clause(s) other than comments intervene before the semicolon. Verify that the statement has its own semicolon. If this message follows a diagnostic other than DB009, it means merely that part of the statement was ignored.

DB006 MISSING SEMICOLON OR ICION. 'text'

Severe error - the right parenthesis at the end of a DB preprocessor function reference has been encountered unexpectedly. The label or statement "text" shown was ignored.

DB007 MISSING SEMICOLON. 'text'

Severe error - the right parenthesis at the end of a DB preprocessor function reference has been encountered unexpectedly. The statement "text" shown was ignored.

DB009 DBPL/I STATEMENT: 'text; comments'

Informative - a non-null statement has been found. The "text" shown is the statement as internally rearranged into a standard format without embedded comments for further analysis. The "comments" shown
are those extracted from the statement.

**DB011 STATEMENT FOLLOWS FINISH.**

Severe error - the statement has been ignored because it follows the `DE(FINISH;)` reference.

**DB013 STATEMENT HAS 'n' MORE LEFT PARENTHESES THAN RIGHT PARENTHESES.**

Severe error - the statement semicolon has been found but the parentheses are unbalanced. The statement was ignored.

**DB015 UNKNOWN STATEMENT KEYWORD: 'word'.**

Severe error - the 'word' shown is not the first word of a DBPI/I statement. The statement was ignored.

**DB017 INVALID LISTERROR ACTION.**

Severe error - an ON LISTERROR statement was ignored because its action clause was neither SYSTEM nor GO TO.

**DB019 INVALID ERRORFILE.**

Severe error - an ON ERRORFILE statement was ignored because its filename was longer than eight characters or was not terminated by a right parenthesis.

**DB021 INVALID WHEN CONDITION.**

Severe error - an ON statement was ignored because it was neither ON LISTERROR nor ON ERRORFILE. These are the only ON statements recognized by the DB preprocessor function.

**DB023 MISSING LIST POINTER.**

A GET LIST or PUT LIST statement was ignored because it did not contain a required parenthesized list pointer.

**DB025 INVALID GET LIST CLAUSE.**

Severe error - a GET LIST statement was ignored because it did not contain either a KEY(O) or a KEY INTO clause.

**DB026 INVALID PUT LIST CLAUSE.**

Severe error - a PUT LIST statement was ignored
because it did not contain required INTERNAL KEY FROM clauses.

**DB027** INVALID FILE.
Severe error - a statement having a FILE clause was ignored because its filename was longer than eight characters or was not terminated by a right parenthesis.

**DB028** MISSING LIST.
Severe error - a SET statement was ignored because it did not contain a required LIST clause.

**DB029** MISSING FILE.
Severe error - a statement that should have a FILE clause was ignored because it did not have one.

**DB030** MISSING SIZE.
Severe error - a SET List statement was ignored because it did not contain a required SIZE clause.

**DB031** INVALID CN ACTION.
Severe error - an ON ERRORFILE statement was ignored because its action clause was neither SYSTEM nor GO TO.

**DB032** MISSING 'LIST' OR 'KEY' CLAUSE.
Severe error - a GET FILE statement was ignored because it did not contain either a LIST or a KEY clause.

**DB033** MISSING FIELD CLAUSE.
Severe error - a statement that should have a FIELD clause was ignored because it did not have one.

**DB034** MISSING 'LIKE LIST'.
Severe error - a SET LIST SIZE statement was ignored because it did not contain a required LIKE LIST clause.

**DB035** MISSING INTO.
Severe error - a GET FIELD statement was ignored because it did not contain a required INTO clause.
DB037  MISSING FROM.
Severe error - a PUT or REPUT statement was ignored because it did not contain a required FROM clause.

DB039  MORE ITEMS THAN FIELDS.
Severe error - excess INTO or FROM items in a GET, PUT or REPUT FIELD statement were ignored.

DB041  MORE FIELDS THAN ITEMS.
Severe error - excess fieldnames in a GET, PUT or REPUT FIELD statement were ignored because the INTO or FROM clause has too few items.

DB043  INVALID OPEN CLAUSE.
Severe error - an OPEN statement has been abandoned because one of its substatements has an invalid or out of order FILE, TITLE, access or function clause.

DB045  INVALID READ OPTION(S).
Severe error - a READ statement has been ignored because it has an invalid or out of order file-positioning or NOLOCK option.

DB047  INVALID CLOSE SYNTAX.
Severe error - a CLOSE statement has been abandoned because one of its substatements has an invalid or out of order FILE or ERASE clause.

DB049  MISSING FROM.
Severe error - a WRITE statement has been ignored because it did not contain a required FROM clause.

DB051  MISSING KEYFROM.
Severe error - a LOCATE statement has been ignored because it did not contain a required KEYFROM clause.

DB053  LIST OPTION MISSING.
Severe error - a FREE statement has been ignored because it did not contain a required LIST option.

DB055  INVALID LIST.
Severe error - a FREE LIST statement has been ignored because it did not contain a parenthesized set of list-pointers.

DB057 'filename' FILE HAS STATEMENT DIAGNOSTIC(S).

Severe error - the FINISH statement has not generated a Mainline File Control Block declaration (see Section III, Topic B.4 of the DWB) for the 'filename' shown because errors in its use have been previously detected. Note that a missing MFCB declaration will yield "undefined qualified name" diagnostics from the PL/I compiler for correct DBPL/I statements using the 'filename'.

DB059 'filename' FILE HAS NON-INPUT USE(S).

Informative - the 'filename' shown has a use that may conflict with the INPUT file function attribute.

DB061 'filename' FILE HAS NON-OUTPUT USE(S).

Informative - the 'filename' shown has a use that may conflict with the OUTPUT file function attribute.

DB063 'filename' FILE HAS NON-UPDATE USE(S).

Informative - the 'filename' shown has a use that may conflict with the UPDATE file function attribute.

DB065 'filename' FILE REQUIRES UPDATE ATTRIBUTE.

Informative - the 'filename' shown has a use that requires the UPDATE file function attribute, but this compilation does not contain a valid OPEN...UPDATE statement for the 'filename'.

DB067 'n' DBPL/I ERRORS.

Informative - the FINISH statement has been processed and 'N' errors were previously detected. The programmer should find and analyze the 'N' DBPL/I diagnostic comments.
TOPIC B.2 - DATA BASE EXECUTIVE

A. DATA SET NAME:
   DBPL/I - DEPAc Interface

B. CREATED BY:
   DB Preprocessor Function

C. TYPE OF FILE:
   (4) Table

D. ORGANIZATION:
   Documentary Table

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:

The DBPL/I - DEPAc Interface (see Table 1) specifies the MFCE. STATEMENT.OPERATION code, DBPAC entry point name, and argument types and order for the various DBPL/I statements and substatements. Thus, it serves to specify for the IF preprocessor function (see Section IV, Topic B.1 of the DWB) what MFCB assignments and CALL statements are to be generated for each DBPL/I statement. Conversely, it specifies for DBPAC (see Section II, Topic B.2 of the DWB) what entry points will be entered and what and how information will be available at execution time for the performance of the various statement actions.

The various entry points and their argument types are declared by source code in SOURCE.LISRMAC member DBTEXT. Any program that includes the DB preprocessor also is given DBTEXT by an INCLUDE statement in DB.
<table>
<thead>
<tr>
<th>ROUTINE</th>
<th>OPTION</th>
<th>OPERATION</th>
<th>ENTRYPONT</th>
<th>ARG-1</th>
<th>ARG-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS-close</td>
<td>00010000</td>
<td>DBPACFPV</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS-close</td>
<td>00011000</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SE-get-f</td>
<td>C1100000</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>SE-get-f</td>
<td>01100100</td>
<td>DBPACFP</td>
<td>f</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>GET INDEX-K</td>
<td>01100001</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>GET KEY-SET</td>
<td>01100000</td>
<td>DBPACP</td>
<td>p</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>GET SUBFILE-KEY SET</td>
<td>01010001</td>
<td>DBPACFP</td>
<td>f</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>GET LIST-SET</td>
<td>01010000</td>
<td>DBPACFP</td>
<td>f</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>GET INDEX-LIST SET</td>
<td>01010001</td>
<td>DBPACFP</td>
<td>f</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>GET RECORD</td>
<td>01000000</td>
<td>DBPACFR</td>
<td>f</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>LOCATE</td>
<td>11010000</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>LOCATE SUBFILE</td>
<td>11010010</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS open</td>
<td>00100000</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SE put-file</td>
<td>10010000</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>READ KEY</td>
<td>11100000</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>READ KEY NOLOCK</td>
<td>11100100</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>READ INDEX KEY</td>
<td>11100101</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>READ SUBFILE KEY</td>
<td>11100000</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>READ SUBFILE-KEY NOLOCK</td>
<td>11100110</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>READ PER SUBFILE NOLOCK</td>
<td>11101100</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>READ PER SUBFILE NOLOCK</td>
<td>11101110</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>READ LIST</td>
<td>11110100</td>
<td>DBPACFP*</td>
<td>f</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>READ LIST NOLOCK</td>
<td>11110110</td>
<td>DBPACFP*</td>
<td>f</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>READ seq.</td>
<td>11110000</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ seq. NOLOCK</td>
<td>11110100</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ INDEX seq.</td>
<td>11110101</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ SUBFILE seq.</td>
<td>11110010</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ SUBFILE seq. NOLOCK</td>
<td>11110110</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ BACK</td>
<td>11110000</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ BACK NOLOCK</td>
<td>11111100</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ INDEX BACK</td>
<td>11111101</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ SUBFILE BACK</td>
<td>11111101</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ SUBFILE BACK NOLOCK</td>
<td>11111110</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SE repsect-field</td>
<td>10100000</td>
<td>DBPACFP</td>
<td>f</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>UNLOCK</td>
<td>11000000</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNLOCK SUBFILE</td>
<td>11000010</td>
<td>DBPACFP</td>
<td>f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE</td>
<td>10000000</td>
<td>DBPACFP</td>
<td>f</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>
SS = substatement
SE = statement element
f = filename
p = list pointer
r = record work area
v = character string

*For READ LIST <NCLOCK> with the KEY (nn) clause, use entry point DBPACPF and a fullword subscript value as the third argument.
TOPIC B.3 - DATA BASE EXECUTIVE

A. DATA SET NAME:
   DBPAC Error Codes

B. CREATED BY:
   RDBPAC posts in FCB.ERRCR.ONCODE.

C. TYPE OF FILE:
   (4) Table

D. ORGANIZATION:
   Sequential

E. KEY IDENTIFIER (CONTROL FIELD):
   Error-code

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:

   The data base executive was written with the intent of handling all data base interfaces with the users of the NASIS system.

   This is handled by the user writing a PL/I program and in it using the DEPL/I language extension to handle all of the input/output operations to the data base.

   When the user's program is running, the data base executive will attempt to detect any and all errors which occur and communicate these errors back to the user's program.

   The method of performing this communication of errors is through the use of the data base executive error codes. These are fixed binary numbers which have unique meanings and are transmitted back to the user's program using the MFCB (mainline file control block) as a vehicle of communication.
<table>
<thead>
<tr>
<th>ERROR CODE</th>
<th>EXPLANATION OF ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Illegal attempt to imply open.</td>
</tr>
<tr>
<td>03</td>
<td>Tried to imply an open on a new file-name without the use of an open command.</td>
</tr>
<tr>
<td>20</td>
<td>Trying to open a file when the header descriptor's DESCOK switch is off.</td>
</tr>
<tr>
<td>21</td>
<td>Trying to open for OUTPUT or UPDATE and the file is not the anchor or an associate or a descriptor file.</td>
</tr>
<tr>
<td>22</td>
<td>End-of list (READ LIST Statement).</td>
</tr>
<tr>
<td>23</td>
<td>Number of files exceeds the number allowed in the MFCE file array.</td>
</tr>
<tr>
<td>25</td>
<td>The user is not the owner of the file, but he is attempting an open for UPDATE or OUTPUT.</td>
</tr>
<tr>
<td>26</td>
<td>Attempted open of the file for INPUT, the DATA switch indicates no data.</td>
</tr>
<tr>
<td>27</td>
<td>Open attempted but its function was not INPUT, OUTPUT, or UPDATE.</td>
</tr>
<tr>
<td>28</td>
<td>Open issued for UPDATE or OUTPUT was prohibited by the MNTING, MNTABLE, or the DATA switch.</td>
</tr>
<tr>
<td>30</td>
<td>Operation code error.</td>
</tr>
<tr>
<td>31</td>
<td>Key field failed general validation (READ KEY or LOCATE).</td>
</tr>
<tr>
<td>32</td>
<td>Key field failed specific validation (READ KEY or LCCATE).</td>
</tr>
<tr>
<td>34</td>
<td>Erase attempted on CLOSE but the file is not open for UPDATE.</td>
</tr>
<tr>
<td>35</td>
<td>Erase attempted on descriptor file other than the anchor.</td>
</tr>
<tr>
<td>36</td>
<td>The GET FIELD operation attempted but the last record operation was a LOCATE.</td>
</tr>
<tr>
<td>38</td>
<td>The GET FIELD operation attempted but there is no current record.</td>
</tr>
</tbody>
</table>
39 GET RECORD operation attempted but the user is not the owner.

40 GET LIST attempted, but file is not inverted index.

41 Key is null (READ KEY or LOCATE).

42 Key sequence error (LOCATE sequential).

43 Duplicate key error (LOCATE direct).

44 Not an OUTPUT file for WRITE.

46 No current record (PUT or REPUT).

47 Current record not locked (PUT or REPUT).

49 PUT or REPUT to INPUT file.

50 REPUT to non-UPDATE file.

51 REPUT following LOCATE.

52 GET operation (field is not in descriptor table).

53 Field failed general validation (PUT or REPUT).

54 Field failed special validation (PUT or REPUT).

55 Null value to be PUT.

56 Bit field too long (PUT or REPUT).

57 PUT to non-null bit switch.

58 PUT to non-null fixed length field.

59 PUT to non-null variable length (single element) field.

61 Field would make record too long (PUT).

62 Field would make record too long (REPUT).

63 Element would make record too long (PUT).

64 Element would make record too long (REPUT).

65 Element field too long (PUT or REPUT).

66 Too many (variable) elements (PUT).
67 No GET before REPUT (variable elements).
68 No good GET before REPUT (variable elements).
69 Undefined field (FUT or REPUT).
70 REPUT to never PUT (null) field (record not found).
71 Too many (fixed) elements (PUT).
72 (Fixed) element would make record too long (PUT).
73 No GET before REPUT (fixed elements).
74 No good GET before REPUT (fixed elements).
75 Field too long (FUT or REPUT).
76 Key would make cross reference record too long (PUT or REPUT).
77 Cross reference not found on record.
78 Target field 'actual' length checking indicates truncation.
79 Command system trying to open someone else's STATIC or TRNSCT dataplexes for UPDATE or OUTPUT.
80 Command system opening a dataplex (other than STATIC or TRNSCT) for either OUTPUT or UPDATE.
83 GET KEY incompatible with list.
84 GET KEY sequence error.
85 Field length less than 2 found. Data Base damage.
86 Field length beyond reclen found. Data Base damage.
87 Field length not equal to 2 plus a multiple of element length found. Data Base damage.
88 Element length less than 1 found. Data Base damage.
89 Element length beyond field length found. Data Base damage.
90 Invalid DB2 header descriptor. DB1 descriptor or damage.

91 Field descriptor reclen less than 78. Descriptor damage.

92 Field length less than 2 in descriptor. Descriptor damage.

93 Field length beyond reclen in descriptor. Descriptor damage.

94 VALIDARG longer than 50 bytes would be truncated. Descriptor damage.

95 SECURITY field length invalid. Should be 2 plus a multiple of 8.

96 No descriptor found for key field. Descriptor damage.

97 Invalid field length in index record found. Database damage.

98 Record missing from index region. Database damage.

99 End of data. (VISAM)

104 Keys equal - sequence error. (VISAM+100)

108 Key not found. (VISAM+100)

112 Keys out of sequence. (VISAM+100)

115 Keys do not coincide. (VISAM+100)

120 Keys coincide. (VISAM+100)

124 Invalid retrieval address. (VISAM+100)

128 Invalid record length. (VISAM+100)

131 Position past end of data set. (VISAM+100)

136 Position before start of data set. (VISAM+100)

140 Exceed maximum number of overflow pages. (VISAM+100)

144 Exceed maximum size of shared data set. (VISAM+100)

145 No data set-name found which is like the given
one. (VISANM)

200 Attempt to (PUT or REPUT) null pattern.

201 Attempt to (PUT or REPUT) to readonly field.

202 Undefined subfile or indexed field (READ, LOCATE, or UNLOCK).

203 Not an indexed field.

204 Not a subfile <control> field.

205 LOCATE to INBUF file.

206 No current anchor record (LOCATE SUBFILE).

207 Anchor record not locked (LOCATE SUBFILE).

208 No current subrecord (READ PER SUBFILE).

209 Anchor record not current (delete subrecord).

210 The file is not open (to post FLDTAB).

211 Descriptor damage detected while posting FLDTAB.

212 Anchor record not locked (delete subrecord).

213 Anchor record not parent of subrecord (delete subrecord).

214 Subrecord id not found in control field (delete subrecord).

215 Duplicate fixed length element (PUT or REPUT).

216 Duplicate varying length element (PUT or REPUT).

217 LOCATE SUBFILE not done because 131071 regions used.

218 NAMEFLD field length invalid. Should be 2 plus a multiple of 9.

219 GET superfield requires current subfile record.

220 RSECTYCD field length invalid. Should be 2 plus a multiple of 9.

221 Non-owner attempted to open associate having record level security.
TOPIC B.4 - DATA BASE EXECUTIVE

A. DATA SET NAME:
Mainline File Control Block

B. CREATED BY:
Declared by DB Preprocessor Function.

C. TYPE OF FILE:
(4) Table

D. ORGANIZATION:
Linear structure followed by an array of linear structures.

E. KEY IDENTIFIER (CCNTBCI FIELD):
Within DEPAC the mainline file control block is a parameter named MFCB. Outside DBPAC each mainline file control block is an independent external controlled structure whose name is the DBPL/I file name (PLEX in the retrieval system). For this reason, file names must not conflict with other external name system. This file name is not padded with dollar signs the way a file title must be. The file name is passed as an argument in CALL statements to DBPAC and thus becomes the MFCB parameter.

F. RECORD LENGTH:
1324 bytes (hexadecimal 52C)
This is the length of the whole control block including the necessary dope vectors and a thirty-seven element array allowing up to thirty-seven data sets in a data base. The number of elements in the array may be adjusted, if necessary: - the control block size will be adjusted by 24 bytes per element and the MFCB.FILE.ARR_SIZE field must be suitably initialized but no changes are necessary in DBPAC or in other MFCB's.

G. BLOCKING FACTOR:
Not Applicable
H. PURPOSE:

The MFCB control block is used for communication between mainline programs and DBPAC. For DBPL/I statements in the mainline, the DB preprocessor function generates statements that post fields in the MFCB, such as the operation code. At execution time, the posted MFCB is passed as an argument to DBPAC. DBPAC performs the operation indicated in the MFCB, making reference to other fields in the MFCB as necessary and posting fields in the MFCB, such as MFCB.ERROR.ONCODE, which may subsequently be referenced in the mainline.

I. PL/I DECLARATION:

DECLARE
1 MFCB, /*MAINLINE FILE CONTROL BLOCK*/
   1 INITIALIZE BIT(2), /*00: NEVER INITIALIZED */
       /*10: INITIALIZED, CLOSED */
       /*11: INITIALIZED, OPENED */
   2 FILLER_1 BIT(6), /*NOT USED */
   2 STATEMENT, /*CR FUNCTION */
       3 OPERATION BIT(8), /*CODE */
       3 ONFIELD CHAR(8), /*FIELD NAME */
   2 FILLER_2 CHAR(3), /*NOT USED */
   2 ERROR,
       3 SYSTEM BIT(1), /*1: STANDARD DBPAC ACTION */
       /*0: USER ERROR ROUTINE */
       3 FILLER_3 BIT(7), /*NOT USED */
       3 ONCODE FIXED BINARY, /*USER'S */
       3 ROUTINE LABEL, /*IN MAINLINE */
       3 ONRETURN LABEL,
   2 FILE,
       3 ONFILE CHAR(8), /*FILE TITLE */
       3 OLFILE CHAR(8), /*TO SAVE FILE TITLE IF DYNAM*/
       3 OWNER_ID CHAR(8), /*OWNER OF THE FILE */
       3 DSNAME CHAR(35), /*DATA SET NAME */
       3 ATTRIBUTES
           4 ACCESS BIT(1), /*0: DIRECT */
               /*1: SEQUENTIAL */
           4 SAVE_FUNC BIT(2), /*TO SAVE FUNCTION IF DYNAMIC*/
           4 FILLER_4 BIT(3), /*NOT USED */
           4 FUNCTION BIT(2), /*TO SAVE FUNCTION IF DYNAMIC*/
               /*10: INPUT */
               /*01: OUTPUT */
               /*11: UPDATE */
   3 CURRENT_FILE FIXED BINARY, /*SUBSCRIPT IN FILE,ARRAY*/
   3 LAST_FILE FIXED BINARY, /*NUMBER OF FILES IN FILEPLEX OR DATAPLEX*/
   3 ARR_SIZE FIXED BINARY(15), /*DECLARED ARRAY SIZE */
   3 ARRAY (37),
       4 FILE_NAME CHAR(8), /*DESCRIPTION TABLE ADDRESS */
       4 DTP POINTER,
4 FCEP PCINTER, /*FILE CONTROL BLOCK ADDRESS */
4 KYC FIXED BINARY(15),/*SUBSCRIPT OF KEY FIELD */
/*DESCRIPTOR IS ALWAYS =1. */

4 SWITCHES,
5 CURRENT BIT(1),
5 LOCKED BIT(1),
5 WRITE BIT(1), /*FORCE WRITE */
5 REWRITE BIT(1), /*FORCE REWRITE */
5 ABSENT BIT(1), /*NULL OR SECURED RECORD */
5 OPENED BIT(1), /*THE FILE IS OPEN */
5 FILLER_5 BIT(2), /*NOT USED */
4 FILLER_6 CHAR(1), /*NOT USED */
4 RECORDCT FIXED BINARY(15); /* # OF USABLE DESC. */

J. DETAIL NOTES:

INITIALIZED - used entirely within DBPAC.

STATEMENT.CPERATION - see Section III, Topic B.2 of the
DWB for the codes that are posted here by the DB
preprocessor function.

STATEMENT.CNFIELD - posted by the DB preprocessor
function.

ERROR.SYSTEM - posted by the DB preprocessor
function.

ERROR.ONCODE - posted by DBPAC when an error is
detected but not reset for successful
operations. See Section III, Topic B.3 of the DWEB
for the DBPAC Error Codes.

ERROR.ROUTINE - posted by the DB preprocessor
function.

ERROR.ONRETURN - posted by DBPAC when an error is
detected.

FILE.ONFILE - posted by the DB preprocessor function.
When the first character is not a pound sign
indicating a descriptor file, DBPAC shifts the
value one character to the right and posts a
leading black character.

FILE.OLFILI - used within DBPAC to detect need for
reinitialization.

FILE.OWNER_ID - used within DBPAC.

FILE.DSNAME - used within DBPAC.

FILE.ATTRIBUTES.ACCESS and FUNCTION - posted by either
the DE preprocessor function or, when in default, by DBPAC.

FILE.SAVE_FUNC - used within DBPAC.

FILE.CURRENT_FILE - used within DBPAC.

FILE.LAST_FILE - used within DBPAC.

FILE.ARR_SIZE - set by the DB preprocessor function indicating the dimension of the FILE.ARRAY.

FILE.ARRAY - this array is used within DBPAC. Each element of the array is a linear structure of fields relating to a data set. When the mainline is accessing a descriptor region or an inverted index, only the first element is used. Otherwise, the first element relates to the anchor data set and subsequent elements relate to associated and subfile and inverted index data sets.

FILE.ARRAY.FILE_NAME - the "title" of the data set having a leading blank or pound sign and a trailing blank or suffix.

FILE.ARRAY.DTP - the address of the (dynamically allocated) descriptor table for this data set.

FILE.ARRAY.FCBP - the address of the (dynamically allocated) file control block for this data set.

FILE.ARRAY.KYC - the subscript of the key field descriptor in the descriptor table array is always 1.

FILE.ARRAY.KYC(1) - one (the anchor) plus the number of associate data sets.

FILE.ARRAY.KYC(2) - KYC(1) plus the number of subfile data sets.

FILE.ARRAY.SWITCHES - switches used by DBPAC for the status of the data set.

FILE.ARRAY.RECORDCT - the number of descriptors in the descriptor table array. This number does not include descriptors of fields a given user does not have field security clearance to access.
TOPIC B.5 - DATA BASE EXECUTIVE

A. DATA SET NAME:
List Structure

B. CREATED BY:
RDBPAC and RDBLIST

C. TYPE OF FILE:
(4) Table

D. ORGANIZATION:
Structure containing and adjustable array

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
40 byte prefix plus number-of-keys times internal key length

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
The list structure includes the primary list of keys and all supporting information. The list segments created by the data base executive are all linked together by the PREV pointer and the NEXT pointer.

I. PL/I DECLARATION:

DECLARE
  1 LIST BASED(LIST_PTR), /*MAIN STORAGE KEY LIST*/
     3 PREFIX,
     5 CHAIN,
     7 PTR,
     9 PREV PTR,
     9 NEXT PTR,
    7 CONTINUED BIT(1),
     5 SPARE BIT(31), /*NOT USED*/
     5 STRING_SIZE FIXED BIN, /*ONLY POSTED BY ALLOC.*/
    5 KEY,
7 MAX_COUNT FIXED BIN,
7 COUNT FIXED BIN,
7 CURSOR,
9 READ FIXED BIN,
9 GET FIXED BIN,
7 SIZE FIXED BIN,
7 FIELD_NAME CHAR(8),

7 CONVERSION CHAR(8),

3 KEYS CHAR(LIST STRING_SIZE
REFER(LIST STRING_SIZE));

/*COUNT LIMIT THIS SEGMENT*/
/*KEYS PER THIS SEGMENT*/
/* FOR THIS SEGMENT*/
/* FOR READ PER LIST*/
/* FOR GET KEY*/
/*BYTES PER KEY*/
/*KEY CONTROL FLD NAME*/
/*KEYS APPLY TO*/
/*RTN NAME FOR OUTPUT*/
/*NULL VALUE: NONE*/
/*CHAR (KEY SIZE)*/
/*MAX_COUNT*/
TOPIC B.6 - DATA BASE EXECUTIVE

A. DATA SET NAME:
LISTERB - List Error Control Block

B. CREATED BY:
Allocated by RDBMIT.

Error fields are posted by DB preprocessor and RDBLIST.
List chain anchors are initialized by RDBMTT and posted
by RDBPAC and RDBLIST.

C. TYPE OF FILE:
(4) Table

D. ORGANIZATION:
Simple structure

E. KEY IDENTIFIER (CONTROL FIELD):
Not applicable

F. RECORD LENGTH:
68 bytes (20 bytes data + 48 bytes PL/I dope vectors,
etc.)

G. BLOCKING FACTOR:
Not applicable

H. PURPOSE:
The list error control block holds the list segment
chain anchors. (The chain used by the FREE all LISTs
statement.) It also is the point of communication
between RDBLIST and mainline programs for list error
handling when no MFCB is involved in the operation.
(When an MFCB is involved in a list error situation,
the MFCB is used for error indication, etc.)

I. PL/I DECLARATION:
DECLARE

1 LISTERR CTL EXT, /*COMMON CONTROL BLOCK */
  3 ERROR,
  5 SYSTEM BIT(1) /*1: SYSTEM ACTION */
INIT('1'B),
5 ONCODE FIXED BIN(15)
  INIT(0),
/*0: GC TO USER ERROR RTNE */
/*1: INVALID LIST OPERATION */
/*2: INCOMPATIBLE LISTS */
/* GET LIST KEY SET ERRORS: */
/*4: NULL INPUT LIST */
/*5: NO GET KEY SINCE RESET */
/*6: INCOMPATIBLE LISTS */
/* GET LIST KEY INTO ERROR: */
/*7: KEY SEQUENCE ERROR */
/*8: TRUNC, TARGET TOO SHORT*/
/* SET LIST LIKE LIST ERROR: */
/*9: INVALID SIZE */
/* PUT LIST KEY FROM ERRORS: */
/*10: NULL TARGET LIST */
/*11: WRONG LENGTH KEY VALUE*/
/*12: KEY SEQUENCE ERROR */
/*USER ERROR RTNE ADDRESS */
/*NULL: NO CHAIN */
/*ALLOCATOR MUST INITIALIZE */
/*FORWARD CHAIN ANCHOR */
/*BACKWARD CHAIN ANCHOR */
5 ROUTINE LABEL
3 PTR,
5(FIRST, LAST) PTR;
TOPIC B.7 - DATA BASE EXECUTIVE

A. DATA SET NAME:
Data base Descriptor File,
for example:

SACOWNER.ASRDI$.ASREI$#

where:
"ownerid" is the 1-8 character TSS identification of
the owner of the data base.
"data base" is the 6 character data base name with the
dollar sign character used for padding.

B. CREATED BY:
RDBEDIT - the Descriptor Editor program

C. TYPE OF FILE:
(6) Data Ease Descriptor file

D. ORGANIZATION

VISAM - Virtual Indexed Sequential Access Method
(TSS) - organized in one or more regions of contiguous
records: one region for the anchor data set
descriptors and, as necessary, a region for each
associate, subfile and inverted index dataset's
descriptors.

Records have the varying length universal record format
(URF) built by DEPAC.

E. KEY IDENTIFIER (CONTROL FIELD):
The VISAM key length is 15 bytes consisting of:

7 character region name and 8 character FILENAME.

A DBPL/I descriptor FILE is a contiguous set of records
having the same region name. The DBPL/I FILE shall be
OPENed using an 8 character TITLE value consisting of:

1 pound sign character (#) (signifying descriptor
file).

6 character data base name with the dollar sign
character ($) used for padding.

1 suffix character.

The suffix character shall be from the following ranges:

<table>
<thead>
<tr>
<th>Range</th>
<th>Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>blank</td>
<td>anchor file descriptors</td>
</tr>
<tr>
<td>1-9</td>
<td>associate file descriptors</td>
</tr>
<tr>
<td>Z-O</td>
<td>subfile descriptors</td>
</tr>
<tr>
<td>A-P</td>
<td>inverted index descriptors</td>
</tr>
</tbody>
</table>

DBPAC uses the data base name and suffix to automatically generate the region name value for the keys.

The DBPL/I KEY value is only the 8 character FLDNAME (name of the field being described) that completes the VISAM key value.

F. RECORD LENGTH:

34 bytes minimum for a file descriptor record.
78 bytes minimum for a field descriptor record.

G. BLOCKING FACTORS:

One 4096 byte page (block) will hold about 40 average descriptors; enough for a few regions for a simple data base. For a complicated data base with many data sets, fields, secondary fields, and security codes, three or more pages (blocks) may be required.

H. PURPOSE:

A data base descriptor file describes a data base in terms of the datasets, records, and fields the data base is composed of, and indicates their interrelationships. A data base descriptor file is created and maintained or modified by RDBEDIT, the descriptor editor, which is a system service program.

Then the data base may be loaded, maintained, retrieved, etc. by programs using RDBPAC for data base access. RDBPAC, when OPENing a data base, reads the descriptor file and from it builds a table that governs its further actions.

I. SAMPLE DATA BASE:

A sample data base is used to illustrate in detail how the descriptors shall describe a complicated data base. See Figure 1.
The record layout in Figure 2 shows all eight record layouts in the sample data base for reference in the following sections of this specification.

J. DESCRIPTOR REGIONS:

The sample descriptor file consists of eight regions (having the suffixes 'I', 'J', 'Z', 'Y', 'A', 'B', 'C', 'D') corresponding to the eight data sets in the sample data base. (Of course VISAM alphabetizes them 'A', 'B', 'C', 'D', 'Y', 'Z', 'I'.)

Each descriptor region has at least one file descriptor record and two field descriptor records (key and RECLEN).

Dummy descriptor records are required in anchor regions when the data base has associate and/or subfile datasets. They are also required in associate regions when subfile(s) are controlled from the associate dataset.

Figure 3 tabulates all the file, field, and dummy descriptors required to describe the sample data base.

K. FILE DESCRIPTOR RECORD:

The file descriptor record describes the dataset as a whole. It is uniquely identified by having a key (FLDNAME) of 8 blanks. It has the field values shown below. See Figure 4 for the field values of the sample descriptor file. All values should be posted except that if RECSECFP is NULL, RSECTYCD does not apply.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLDNAME</td>
<td>8 blanks</td>
<td>DBPL/I key</td>
</tr>
<tr>
<td>DESCK</td>
<td>OFF</td>
<td>incomplete descriptors.</td>
</tr>
<tr>
<td></td>
<td>ON</td>
<td>descriptors are complete.</td>
</tr>
<tr>
<td>FILETYPE</td>
<td>ANCHOR</td>
<td>type of data set being described</td>
</tr>
<tr>
<td></td>
<td>ASSOCIATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SUBFILE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INDEX</td>
<td></td>
</tr>
<tr>
<td>DESCSRCT</td>
<td>numeric</td>
<td>number of field descriptors in this region.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(The file descriptor is not to be counted.)</td>
</tr>
<tr>
<td>ESELNGTH</td>
<td>numeric</td>
<td>length in bytes of fixed portion</td>
</tr>
</tbody>
</table>
of records including RECLEN, key and fixed primary fields. For a spanned index, this includes the key suffix byte.

SPANNEE
OFF ordinary records.
ON spanned records with internally suffixed keys.

DATA
OFF no data on file yet.
ON retrieval is possible.

MNTNABLE
ON maintenance is allowed.
OFF maintenance is prohibited.

MNTNING
OFF no maintenance is in progress.
ON maintenance is in progress.

LOADABLE
ON loading is allowed.
OFF loading this data set is prohibited.

RECSECFP
null records do not have a record security field.
numeric numeric offset in bytes of record security field in records.

RSECTYCD
9 byte elements zero or more record level security codes.
8 alphameric record security password.
1 byte mask for comparison with record security field.

L. FIELD DESCRIPTOR RECORDS:

A field descriptor record indicates that a particular named field occurs in the dataset being described. It is uniquely identified within the region by having a key (FLDNAME) that is the name of the field. There are two kinds of field descriptors:

primary direct
secondary (direct and indirect)

All field descriptor records may have the values shown below:

FIELD VALUE COMMENTS
-------- ----- -------------
FLDNAME 8 alphameric unique field name. DBPL/I
          blank padded key.
GENERCRT 8 alphamerics blank padded name of generic routine for testing input values.

VALIDRTN 8 alphamerics blank padded name of routine for testing and/or converting input values.

VALIDARG 0-50 bytes argument to be supplied to VALIDRTN

NUMALIGN OFF string alignment, left justification
ON numeric alignment, right justification

REFORMAT 8 alphamerics blank padded name of routine for converting output values.

SECURITY 8 alphamerics asterisk padded 0-18 field security pass-words

GENERCRT, VALIDRTN, VALIDARG, and NUMALIGN may even be posted for secondary (read only) fields because linear search, for example, may have to transform values to be used as comparands.

M. PRIMARY DIRECT FIELD DESCRIPTOR RECORDS:

A primary direct field descriptor record describes a maintainable field that occurs on each record of the dataset being described. There are five kinds of primary fields:

  single fixed bit
  single fixed byte
  single varying byte
  multiple fixed byte
  multiple varying byte

In addition to the field values shown in Section L, all primary descriptors have READONLY OFF and a selection of the following field values.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
<td>------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>READONLY</td>
<td>OFF</td>
<td>field value may be maintained (PUT and REPUT.)</td>
</tr>
<tr>
<td>VARFLD</td>
<td>FIXED</td>
<td>fixed length field in fixed portion of records.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>portion of records.</td>
</tr>
<tr>
<td>BITFLD</td>
<td>OFF</td>
<td>byte field</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td>CN</td>
<td>bit field</td>
</tr>
</tbody>
</table>

| FLDPOSIT | numeric | if VARFLD is FIXED, offset in bytes of field. |
|          |        | if VARFLD is VARYING, relative field in variable portion of records. |

| FLDLEN   | numeric | if BITFLD is ON, offset of bit (0, 2, 4 or 6) in byte specified by FLDPOSIT. |
|          |        | if VARFLD is FIXED, internal length of field in bytes. |
|          |        | if VARFLD is VARYING, maximum internal length of field in bytes with internal field length prefix. |

| ELTLEN   | numeric | field does not have elements. |
|          |        | maximum number of elements to be PUT into field or, for a control field, maximum number of subrecords per parent record. |

| VARELT   | FIXED | fixed length elements. |
|          | VARYING | varying length elements. |

| UNIQUELT | OFF | duplicate element values are allowed. |
|          | ON  | internal element values must be unique. |

| INVFILE  | alphabetic | descriptor region suffix for inverted index dataset. (null if none.) |

| INDEXEXT | OFF  | index internal values of field. |
|          | ON   | index external values of field. |
|          |      | (External length may require index key length greater than internal length.) |

A single fixed bit field descriptor has:

<table>
<thead>
<tr>
<th>VARFLD</th>
<th>FIXED</th>
</tr>
</thead>
<tbody>
<tr>
<td>BITFLD</td>
<td>CN</td>
</tr>
</tbody>
</table>
A single fixed byte field descriptor has:

- **VARFLD**: FIXED
- **BITFILE**: OFF
- **FLDPOSIT**: offset in bytes
- **FLDLEN**: internal length in bytes
- **INVFILE**: optional if FLDLEN less than 254

See EMPAYCI, EMFINSL and VEHMAKE in the sample data base.

A single varying byte field descriptor has:

- **VARFLD**: VARYING
- **FLDPOSIT**: relative varying field
- **FLDLEN**: maximum internal length including internal 2 byte field length prefix
- **ELTLIM**: 0 (zero)
- **INVFILE**: optional if (FLDLEN-Z) less than 254

See KIDNAME in the sample data base.

A multiple fixed byte field descriptor has:

- **VARFLD**: VARYING
- **FLDPOSIT**: relative varying field
- **FLDLEN**: maximum internal field length including internal 2 byte field length prefix
- **ELTLIM**: maximum number of elements
- **ELTLEN**: internal element length in bytes
- **VARELT**: FIXED
- **UNIQUELT**: optional
- **INVFILE**: optional if ELTLEN less than 254

See the EMPKID and EMPVEH control fields in the sample data base.

A multiple varying byte field descriptor has:

- **VARFLD**: VARYING
- **FLDPOSIT**: relative varying field
- **FLDLEN**: maximum internal field length including internal 2 byte field length prefix and internal 1 byte element length prefixes
- **ELTLIM**: maximum number of elements
- **ELTLEN**: maximum internal element length including internal 1 byte element length
prefix.

UNIQUELT optional
INVFILE optional if (ELTLEN-1) less than 254.

See the KIDPET field in the sample data base.

CONTROL FIELD DESCRIPTORS

Every descriptor region must have a key descriptor for the field that uniquely identifies records in a dataset. It is a primary direct field descriptor record for a single fixed byte field.

SECURITY must be null.
READONLY is OFF
VARFLE is FIXED
BITFLD is OFF
FLDPOSIT is 4
INVFILE must be null

Each associate key descriptor is identical (except the region suffix) to the anchor key descriptor. See the EMPNAME field in the sample data base.

Each subfile dataset in a data base requires:

1. a control field in the anchor or an associate dataset
2. a separate descriptor region for the subfile dataset containing:
   2a. file descriptor record
   2b. RECLEN secondary descriptor record
   2c. subrecord id key descriptor record
   2d. parent key secondary descriptor record
   2e. descriptor records for other subrecord fields.

1. A subfile control field descriptor describes a secondary multiple fixed byte field maintained by RDEPAC.

   FLDNAME is a six-character name suffixed by two blanks applying to the subfile.
   GENEPCRT is DBCVTID
   VALIDRTN is null
   NUMALIGN is ON
   REFORMAT is DBFMTID
   SECURITY is optional
   READONLY is ON
   VARFLE is VARYING
   FLDPOSIT is relative varying field
FLDLEN is maximum internal field length
ELTLIM is maximum number of elements.
Note that FLDLEN or ELTLIM limits the maximum number of subrecords per parent record.

ELTLEN is 3
VAREIT is FIXED
UNIQUEIT is ON
SUBCNTRI is CN
SUBFILE is alphabetic character descriptor region suffix for subfile dataset.
INVFILE must be null

See EMPKID and EMPVEH in the sample data base.

2a. A subfile file descriptor record has:

FILETYPE SUBFILE

2b. A subrecord RECLEN descriptor is standard.

2c. A subrecord id key descriptor describes a primary single fixed byte field:

FLDNAME is the six-character subfile name suffixed by "ID".
GENERCRT is EBCVTID
VALIDRTN is null
NUMALIGN is CN
FEFORMAT is DBFMTID
SECURITY must be null
READONLY is OFF
VARFID is FIXED
EITFLD is OFF
FLDPOSIT is 4
INVFILE must be null

See EMPKIDID and EMPVEHID in the sample data base.

2d. A subrecord parent key descriptor describes a secondary single fixed byte field.

FLDNAME is the six-character subfile name suffixed by "PK".
GENERCRT, VALIDRTN, NUMALIGN, REFORMAT and FLDLEN are the same as the anchor key descriptor.
SECURITY is optional
READONLY is ON
VARFID is FIXED
EITFLD is OFF
FLDPOSIT is 7
INVFILE must be null
See EMPKIDPK and EMPVEHUK in the sample data base.

2e. Record level security may be independently specified for the anchor, associate and/or subfile datasets. It may not be specified for an inverted index dataset. Each dataset to have record level security must have:

- RECSECFFP field position of record security field
- RSECTYCD optional

in its file descriptor record and a primary direct field descriptor record for a single fixed byte field as shown in Figure 5 and as follows:

- FLDNAME RECSEC suffixed by the descriptor region suffix (blank for the anchor and a blank)
- GENERCRT DBCVTX
- VALIDARG null
- NUMALIGN OFF
- REFORMAT DEPMTH
- SECURITY optional
- VARFID FIXED
- EITFLD OFF
- FIDPOSIT on anchor or associate datasets, after the key (ie. 4 + key FLDLEN + 1). on subfile datasets, after the parent key field (ie. 4 + 3 + parent key FLDLEN + 1).
- INVFILE optional

See the RECSEC, RECSEC1, EMPKIDRS and EMPVEHRS fields in the sample data base.

N. SECONDARY READONLY FIELD DESCRIPTOR RECORDS:

A secondary field descriptor record describes a derived field made up of one or more component fields. There are two types: direct and indirect.

A direct secondary field descriptor redescribes part of all of one primary field or a field automatically maintained by RDEPAC such as RECLEN and subfile control and parent key fields. A direct secondary descriptor has the same field values as a primary descriptor with the following qualifications:

- VARFID is always ON. Field may only be retrieved (GET).
INVFILE must be null.

FLDPOSIT and FLDLEN will specify an internal location within or equal to a primary field.

Otherwise, the descriptor fields may specify a direct secondary field like any of the five types of primary fields. A secondary of the same type of length provides renaming and perhaps an alternate REFORMAT. A secondary "single fixed byte" with a shorter FDLLENGTH provides for subfields. A secondary "single varying byte" redefining a primary "multiple fixed byte" obtains the concatenation of the internal element values.

Every descriptor region shall have a secondary direct field descriptor for a single fixed byte RECLEN as follows:

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLDNAME</td>
<td>is RECLEN</td>
<td></td>
</tr>
<tr>
<td>GENERCRT</td>
<td>is DBCVTRL</td>
<td></td>
</tr>
<tr>
<td>VALIDRTN</td>
<td>is null</td>
<td></td>
</tr>
<tr>
<td>NUMALIGN</td>
<td>is ON</td>
<td></td>
</tr>
<tr>
<td>REFORMAT</td>
<td>is DBMFTRL</td>
<td></td>
</tr>
<tr>
<td>SECURITY</td>
<td>is optional</td>
<td></td>
</tr>
<tr>
<td>READONLY</td>
<td>is ON</td>
<td></td>
</tr>
<tr>
<td>VARFLE</td>
<td>is FIXED</td>
<td></td>
</tr>
<tr>
<td>BITFLE</td>
<td>is OFF</td>
<td></td>
</tr>
<tr>
<td>FLDPOSIT</td>
<td>is 0 (zero)</td>
<td></td>
</tr>
<tr>
<td>FLDLEN</td>
<td>is 4</td>
<td></td>
</tr>
<tr>
<td>INVFILE</td>
<td>must be null</td>
<td></td>
</tr>
</tbody>
</table>

(No dummy descriptors are used for RECLEN. Direct secondary descriptors may be specified by the Data Base Analyst to provide unique field names for the various RECLENs in a data base.)

An indirect secondary field descriptor describes a "superfield" made of one or more primary or direct secondary component fields. No more than one of the component fields may be a multi-element field. The component field values will be concatenated in NAMEFLD order for retrieval. (If there is a multi-element component field, then the superfield will yield multiple values.) An indirect secondary descriptor has the field values shown below.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>READONLY</td>
<td>ON</td>
<td>field may only be retrieved</td>
</tr>
<tr>
<td>NAMEFLD</td>
<td>9 byte</td>
<td>one to 16 component field</td>
</tr>
</tbody>
</table>
elements specifications
hex '00' use external value of component
hex '80' use internal value of component
followed by primary or direct secondary
8 alphameric component fieldname
blank padded

REFORMAT 8 alphameric name of routine for converting
blank padded concatenated output value

If the component fields specified in NAMEFLD are all from the same dataset (anchor, associate or subfile), then the indirect secondary descriptor goes in the descriptor region for that dataset. (Dummy descriptor(s) are required for the indirect secondary descriptor if it is on an associate or subfile.) See the KIDID field in the sample data base.

If the components are from an associate file and from a subfile controlled from that associate file, then the indirect secondary descriptor goes in the descriptor region for that associate.

Otherwise the indirect secondary descriptor goes in the anchor descriptor region and no dummy descriptors are required. See the EMPTYPE field in the sample data base.

If any component is from a subfile dataset, the (1.) no components may be from any other subfile dataset and (2.) to GET such a field, a mainline program must first obtain a current record in the subfile -- FDBPAC will automatically ensure that the parent and associate record(s) are available when required.

Every anchor and associate descriptor region shall have a secondary indirect field descriptor for the key field, see Figure 6.

FLDNAME is 'FILEKEY'
READONLY is ON
NAMEFLD has one element consisting of hex '00'
followed by the name of the primary key field.
REFORMAT is NULL.

(No dummy descriptor is used for FILEKEY on associate datasets.)

O. DUMMY DESCRIPTOR RECORDS
A dummy field descriptor indicates that the field occurs in an associate or subfile data set and if it has an inverted index data set. It has the field values shown below.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLNAME</td>
<td>8 alphameric field name. DBPL/I key. blank padded</td>
<td>descriptor region suffix for associate data set. (Null if none.)</td>
</tr>
<tr>
<td>ASSOCFIL</td>
<td>numeric character</td>
<td>descriptor region suffix for associate data set. (Null if none.)</td>
</tr>
<tr>
<td>SUBFILE</td>
<td>alphabetic character</td>
<td>descriptor region suffix for subfile data set. (Null if none.)</td>
</tr>
<tr>
<td>SUBCNTRL</td>
<td>OFF</td>
<td>dummy descriptor for subfile field. primary (dummy if ASSOCFIL is posted) descriptor for sub-file control field.</td>
</tr>
<tr>
<td>INVFILE</td>
<td>alphabetic character</td>
<td>descriptor region suffix for inverted index data set. (Null if none.)</td>
</tr>
</tbody>
</table>

The descriptor file, shown graphically in Figure 7 and 8, is so designed that the anchor region describes the anchor records and also has dummy descriptors for all other fields on associate (1) or subfile (Z,Y) records thus indicating the presence of all associate and subfile data sets. It also indicates the presence of all inverted indexes for the data base (A,B,C,D).

An associate region (1) describes a data set of associate records and has dummy descriptors for all other fields on subfiles (Y) depending on the associate record. It also indicates the presence of all inverted indexes for the associate and dependent subfiles (C,D).

A subfile region (Z) describes a data set of subfile records and indicates the presence of all inverted indexes for the subrecords (B).

An index region (A) describes a data set of inverted index records.

This all enables DBPAC to access a whole data base, an associate portion of a data base, a subfile, or an
inverted index as if it were a degenerate case of a
data base.

P. INVERTED INDEX DESCRIPTORS:

If INVFILE is posted for a primary direct field, then a
separate descriptor region must exist for the inverted
index dataset. It contains only the following:

FILE DESCRIPTOR RECORD

FILETYPE is INDEX
SPANned is optional
BSELNGTH is 4 + index key FLDLEN (+1 if
SPANned).

REClEN SEcoNDARY DIRECT FIELD DESCRIPTOR RECORD

Single Fixed Byte

INDEX KEY SECONDARY DIRECT FIELD DESCRIPTOR RECORD

Single Fixed Byte

FLDNAME is same as indexed FLDNAME
READONLY is ON
FLDPOSIT is 4

If indexed field descriptor has INDEXEXT OFF, the
FLDLEN is maximum internal indexed field value
length (without 2 byte internal field length or 1
byte internal element length) and REFORMAT is same
as indexed field REFORMAT. If indexed field
descriptor has INDEXEXT ON, then FLDLEN is
maximum external indexed field value length and
REFORMAT is null or a blank stripper. In either
case, FLDLEN does not include the internal
"sequence number" suffix if SPANNED.

CROSS REFERENCES SECONDARY DIRECT FIELD DESCRIPTOR

Record Multiple Fixed Byte

FLDNAME is same as indexed field's record key
FLDNAME.
READONLY is ON.
FLDPOSIT is 1.
FLDLEN is 4C00.
ELTLIM is 4C00.
ELTLEN is same as indexed field's record key
FLDLEN.
REFORMAT is same as indexed field's record key
REFORMAT.
**FILE AND FIELD DESCRIPTOR RECORD FORMATS:**

**File Descriptor Record Format**

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 RECLEN</td>
<td>0-3</td>
<td>Fixed</td>
<td>Binary Length of header record, in bytes, including itself.</td>
</tr>
<tr>
<td>2 KEY</td>
<td>4-18</td>
<td>Fixed</td>
<td>EBCDIC Identifier for this descriptor. Contains file name.</td>
</tr>
<tr>
<td>3 FLENAM</td>
<td>4-10</td>
<td>Fixed</td>
<td>EBCDIC Seven-character file name for this descriptor. Contains data base and suffix.</td>
</tr>
<tr>
<td>4 DATAPLEX</td>
<td>4-9</td>
<td>Fixed</td>
<td>EBCDIC Dataplex name padded with $s to 6 characters.</td>
</tr>
<tr>
<td>5 SUF</td>
<td>10</td>
<td>Fixed</td>
<td>EBCDIC Identifier dataset.</td>
</tr>
<tr>
<td>6 FLDNAM</td>
<td>11-18</td>
<td>Fixed</td>
<td>EBCDIC Contains blanks.</td>
</tr>
<tr>
<td>7 FILETY</td>
<td>19</td>
<td>Fixed</td>
<td>EBCDIC 1: Anchor 2: Associate 3: Subfile 4: Inverted index</td>
</tr>
<tr>
<td>8 DESCRCT</td>
<td>20-21</td>
<td>Fixed</td>
<td>Binary Number of field descriptors for this dataset.</td>
</tr>
<tr>
<td>9 BSELNGTH</td>
<td>22-23</td>
<td>Fixed</td>
<td>Binary Length of fixed portion of record.</td>
</tr>
<tr>
<td>10 DESCK</td>
<td>24.0</td>
<td>Fixed</td>
<td>Bit 0: incomplete descriptors. 1: complete descriptors.</td>
</tr>
<tr>
<td>11 SPANNED</td>
<td>24.2</td>
<td>Fixed</td>
<td>Bit Applicable if FILETYPE=4: 0: ordinary records 1: spanned records with internally suffixed keys.</td>
</tr>
<tr>
<td>12 DATA</td>
<td>24.4</td>
<td>Fixed</td>
<td>Bit 0: No data on file. 1: Retrieval is possible.</td>
</tr>
</tbody>
</table>
13 MNTNAEL 24.6 Fixed Bit 0: Maintenance prohibited.
1: Maintenance allowed.

14 MNTNING 25.0 Fixed Bit 0: Maintenance not in progress.
1: Maintenance in progress.

15 -------- 25.2 Fixed Bit Currently not applicable-Null.
1: Check record security.

16 LOADABLE 25.4 Fixed Bit 0: Loading prohibited.
1: Loading allowed.

17 -------- 25.6 Fixed Bit Currently not applicable-Null.

18 REMAINS 26-29 Fixed ----- Currently not applicable-Null.

19 RECSECFP 30-31 Fixed Binary Record security field offset in records; null if none.

20 RSECTYCD VrFld1 Var ----- 0-18 record security specifications consisting of a NASIS-id padded with $s to 8 characters and a one byte mask.
### Field Descriptor Record Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Length/Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 RECLEN</td>
<td>0-3 Fixed Binary</td>
<td>Length of entire descriptor, in bytes, including itself.</td>
</tr>
<tr>
<td>2 KEY</td>
<td>4-18 Fixed EBCDIC</td>
<td>Identifier for this descriptor. Contains file and field names.</td>
</tr>
<tr>
<td>3 FLENAME</td>
<td>4-10 Fixed EBCDIC</td>
<td>Seven character file name for these descriptors.</td>
</tr>
<tr>
<td>4 FLDNAME</td>
<td>11-18 Fixed EBCDIC</td>
<td>Name of field within file.</td>
</tr>
<tr>
<td>5 ASSOCFIL</td>
<td>19 Fixed EBCDIC</td>
<td>Suffix of associated linear file which contains this field; null if none.</td>
</tr>
<tr>
<td>6 SUBFILE</td>
<td>20 Fixed EBCDIC</td>
<td>Suffix of subfile which contains this field or which is controlled by this field; null if none.</td>
</tr>
<tr>
<td>7 INVFILE</td>
<td>21 Fixed EBCDIC</td>
<td>Suffix of inverted file which indexes this field; null if none.</td>
</tr>
<tr>
<td>8 READONLY</td>
<td>22.0 Fixed Bit</td>
<td>0: (Re)Put allowed. 1: (Re)Put prohibited.</td>
</tr>
<tr>
<td>9 SUBCTRL</td>
<td>22.2 Fixed Bit</td>
<td>Applicable if SUBFILE is non-null: 0: Field is on subfile. 1: This is control field.</td>
</tr>
<tr>
<td>10 VARFLD</td>
<td>22.4 Fixed Bit</td>
<td>0: Fixed length field. 1: Varying length field.</td>
</tr>
<tr>
<td>11 BITFLD</td>
<td>22.6 Fixed Bit</td>
<td>Applicable if VARFLD=0 0: Byte field. 1: Bit field.</td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
<td>Type</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>12 NUMALIGN</td>
<td>23.0</td>
<td>Fixed Bit</td>
</tr>
<tr>
<td>13 VARELT</td>
<td>23.2</td>
<td>Fixed Bit</td>
</tr>
<tr>
<td>14 UNIQUELT</td>
<td>23.4</td>
<td>Fixed Bit</td>
</tr>
<tr>
<td>15 INDEXEXT</td>
<td>23.6</td>
<td>Fixed Bit</td>
</tr>
<tr>
<td>16 GENERCRT</td>
<td>24-31</td>
<td>Fixed EBCDIC</td>
</tr>
<tr>
<td>17 VALIDRIN</td>
<td>32-39</td>
<td>Fixed EBCDIC</td>
</tr>
<tr>
<td>18 REFORMAT</td>
<td>40-47</td>
<td>Fixed EBCDIC</td>
</tr>
<tr>
<td>19 SPARE</td>
<td>48-55</td>
<td>Fixed</td>
</tr>
<tr>
<td>20 NAMECNT</td>
<td>56-57</td>
<td>Fixed</td>
</tr>
<tr>
<td>21 FLDOPOSIT</td>
<td>58-59</td>
<td>Fixed Binary</td>
</tr>
</tbody>
</table>
offset in record.
If VARFLD=1: relative varying field.

22 FLDLEN 60-61 Fixed Binary
If BITFLD=1: bit offset in byte.
If VARFLD=0: field length in bytes.
If VARFLD=1: maximum field length in bytes including 2 byte length indicator.

23 DFLDLEN 62-63 Fixed -----
Currently not applicable-Null.

24 ETLIM 64-65 Fixed Binary
Applicable if VARFLD=1:
0: not multi-element.
>0: maximum number of elements allowed.

25 DELTLEM 66-67 Fixed -----
Currently not applicable-Null.

26 ETLLEN 68-69 Fixed Binary
If VARELT=0: element length in bytes.
If VARELT=1: maximum element length in bytes including 1 byte length indicator.

27 DELETLEM 70-71 Fixed -----
Currently not applicable-Null.

28 VALIDARG VrFld1 VarHex
Argument to be used with VALIDRTN (test mask, limit, etc.). Fifty bytes maximum. Null if none.

29 NAMEFLD VrFld2 Var
0-18 Superfield components consisting of a one byte function code (80x: external field element, 00x: internal field element) and an 8 character component field name.
30 SECURITY VrFld3 Var EBCDIC 0-18 field security codes consisting of a NASIS-id padded with *s to 8 characters.
(Note: A simple data plex consists of only an anchor data set. A more complicated dataplex than the sample may have multiple index, associate, and/or subfile datasets, but the principles for describing it are shown in the sample.)
<table>
<thead>
<tr>
<th>INDEX</th>
<th>A'</th>
<th>RECLEN</th>
<th>EMPNAME (CROSS REFERENCES)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>B'</th>
<th>RECLEN</th>
<th>KIDNAME (KEY)</th>
<th>EMPKIDID (CROSS REFERENCES)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>C'</th>
<th>RECLEN</th>
<th>EMPINSCL</th>
<th>EMPNAME (CROSS REFERENCES)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDEX</th>
<th>D'</th>
<th>RECLEN</th>
<th>VEHMAKE (KEY)</th>
<th>EMPVENID (CROSS REFERENCES)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 2 - Sample Dataplex Record Layout**
**Anchor** | **Associate** | **Subfile** | **Subfile**  
---|---|---|---
Region: 'PLEX$$' | 'PLEX$$1' | 'PLEX$$Z' | 'PLEX$$Y'

file descriptor | file descriptor | file descriptor | file descriptor

**Field descriptors:**
- RECLEN
- EMPNAME (Key)
- FILEKEY
- RECSEC
- EMPTYTYPE
- EMPPAYCL
- EMPKID (control)
  - EMPKIDID
  - EMPKIDPK
  - EMPKIDS
  - KIDNAME
  - KIDPRT
  - KIDID
- RECSECl
- EMPINSCL
- EMPVEH
  - EMPVEHID
  - EMPVEHPK
  - EMPVEHRS
  - VEHAIRCD
  - VEHMAKE
- dummy descriptors

**Index** | **Index** | **Index** | **Index**  
---|---|---|---
Region: 'PLEX$$A' | 'PLEX$$B' | 'PLEX$$C' | 'PLEX$$D'

file descriptor | file descriptor | file descriptor | file descriptor

**Field descriptors:**
- RECLEN
- EMPPAYCL
- EMPNAME

**Key:**
- EMPNAME
- EMPKIDID

**Cross references:**
- EMPNAME
- EMPKIDID

---

FIGURE 3. SAMPLE DATAPLEX DESCRIPTOR LIST
<table>
<thead>
<tr>
<th>REGION</th>
<th>DESCOK</th>
<th>FILETYPE</th>
<th>DESCRT</th>
<th>BSELCNT</th>
<th>SPANNED</th>
<th>DATA</th>
<th>MNTNABLE</th>
<th>MNTNING</th>
<th>LOADABLE</th>
<th>RECSECFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLEX$$</td>
<td>ON</td>
<td>ANCHOR</td>
<td>21</td>
<td>17</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>14</td>
</tr>
<tr>
<td>PLEX$$1</td>
<td>ON</td>
<td>ASSOCIATE</td>
<td>11</td>
<td>20</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>14</td>
</tr>
<tr>
<td>PLEX$$2</td>
<td>ON</td>
<td>SUBFILE</td>
<td>7</td>
<td>18</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>17</td>
</tr>
<tr>
<td>PLEX$$Y</td>
<td>ON</td>
<td>SUBFILE</td>
<td>6</td>
<td>21</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>17</td>
</tr>
<tr>
<td>PLEX$$A</td>
<td>ON</td>
<td>INDEX</td>
<td>3</td>
<td>7</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>null</td>
</tr>
<tr>
<td>PLEX$$B</td>
<td>ON</td>
<td>INDEX</td>
<td>3</td>
<td>14</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>null</td>
</tr>
<tr>
<td>PLEX$$C</td>
<td>ON</td>
<td>INDEX</td>
<td>3</td>
<td>9</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>null</td>
</tr>
<tr>
<td>PLEX$$D</td>
<td>ON</td>
<td>INDEX</td>
<td>3</td>
<td>14</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>null</td>
</tr>
</tbody>
</table>

**FIGURE 4. SAMPLE FILE DESCRIPTORS**
## Sample Primary Direct Field Descriptors

<table>
<thead>
<tr>
<th>FLDNAME</th>
<th>SECURITY</th>
<th>READONLY</th>
<th>VARTLD=FIXED/VARYING</th>
<th>FLTDOSIT</th>
<th>FLDLEN</th>
<th>NUMALIGN</th>
<th>EITLIM</th>
<th>VARTLD=FIXED/VARYING</th>
<th>UNIQUE</th>
<th>GENERIC</th>
<th>VALIDRATION</th>
<th>REFORMAT</th>
<th>INVFILE</th>
<th>INDEXEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLEX$$ EMPNAME</td>
<td>X</td>
<td>OFF</td>
<td>F</td>
<td>OFF</td>
<td>4</td>
<td>10</td>
<td>OFF</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PLEX$$ RECSEC</td>
<td>( )</td>
<td>OFF</td>
<td>F</td>
<td>OFF</td>
<td>14</td>
<td>1</td>
<td>OFF</td>
<td>DBCYTHX</td>
<td>X</td>
<td>X</td>
<td>DBFMTHX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLEX$$ EMPPAYCL</td>
<td>( )</td>
<td>OFF</td>
<td>F</td>
<td>OFF</td>
<td>15</td>
<td>2</td>
<td>OFF</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>A</td>
<td>OFF</td>
</tr>
<tr>
<td>PLEX$$1 EMPNAME</td>
<td>X</td>
<td>OFF</td>
<td>F</td>
<td>OFF</td>
<td>10</td>
<td>1</td>
<td>*</td>
<td>DBCVTHX</td>
<td>X</td>
<td>X</td>
<td>DBFMTHX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLEX$$1 RECSEC1</td>
<td>( )</td>
<td>OFF</td>
<td>F</td>
<td>OFF</td>
<td>14</td>
<td>1</td>
<td>OFF</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>C</td>
<td>OFF</td>
</tr>
<tr>
<td>PLEX$$1 EMPINSCL</td>
<td>( )</td>
<td>OFF</td>
<td>F</td>
<td>OFF</td>
<td>15</td>
<td>5</td>
<td>OFF</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLEX$$ EMPKIDID</td>
<td>X</td>
<td>OFF</td>
<td>F</td>
<td>OFF</td>
<td>4</td>
<td>3</td>
<td>ON</td>
<td>DBCVTID</td>
<td>X</td>
<td>X</td>
<td>DBFMID</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLEX$$ EMPKIDRS</td>
<td>( )</td>
<td>OFF</td>
<td>F</td>
<td>OFF</td>
<td>17</td>
<td>1</td>
<td>OFF</td>
<td>DBCVTHX</td>
<td>X</td>
<td>X</td>
<td>DBFMTHX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLEX$$ ZKIDNAME</td>
<td>( )</td>
<td>OFF</td>
<td>V</td>
<td>OFF</td>
<td>1</td>
<td>10</td>
<td>OFF</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>B</td>
<td>OFF</td>
</tr>
<tr>
<td>PLEX$$ ZKIDPET</td>
<td>( )</td>
<td>OFF</td>
<td>V</td>
<td>OFF</td>
<td>2</td>
<td>40</td>
<td>OFF</td>
<td>5 10 V OFF</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td></td>
</tr>
<tr>
<td>PLEX$$ YEMPVEHID</td>
<td>X</td>
<td>OFF</td>
<td>F</td>
<td>OFF</td>
<td>4</td>
<td>3</td>
<td>ON</td>
<td>DBCVTID</td>
<td>X</td>
<td>X</td>
<td>DBFMID</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLEX$$ YEMPVEHRS</td>
<td>( )</td>
<td>OFF</td>
<td>F</td>
<td>OFF</td>
<td>17</td>
<td>1</td>
<td>OFF</td>
<td>DBCVTHX</td>
<td>X</td>
<td>X</td>
<td>DBFMTHX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLEX$$ YEHAIRCD</td>
<td>( )</td>
<td>OFF</td>
<td>F</td>
<td>ON</td>
<td>18</td>
<td>0</td>
<td>OFF</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PLEX$$ YEHMMAK</td>
<td>( )</td>
<td>OFF</td>
<td>F</td>
<td>OFF</td>
<td>19</td>
<td>2</td>
<td>OFF</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>D</td>
<td>ON</td>
</tr>
</tbody>
</table>

## Sample Secondary Direct Field Descriptors

<table>
<thead>
<tr>
<th>FLDNAME</th>
<th>SECURITY</th>
<th>READONLY</th>
<th>VARTLD=FIXED/VARYING</th>
<th>FLTDOSIT</th>
<th>FLDLEN</th>
<th>NUMALIGN</th>
<th>EITLIM</th>
<th>VARTLD=FIXED/VARYING</th>
<th>UNIQUE</th>
<th>GENERIC</th>
<th>VALIDRATION</th>
<th>REFORMAT</th>
<th>INVFILE</th>
<th>INDEXEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLEX$$ RECLEN</td>
<td>( )</td>
<td>ON</td>
<td>F</td>
<td>OFF</td>
<td>0</td>
<td>4</td>
<td>ON</td>
<td>DBCVTRL</td>
<td>X</td>
<td>X</td>
<td>DBFMTRL</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PLEX$$ EMPKID</td>
<td>( )</td>
<td>ON</td>
<td>V</td>
<td>1 4000</td>
<td>ON</td>
<td>20</td>
<td>3</td>
<td>F ON</td>
<td>DBCVTID</td>
<td>X</td>
<td>DBFMID</td>
<td>X</td>
<td></td>
<td>ON Z</td>
</tr>
<tr>
<td>PLEX$$1 RECLEN</td>
<td>( )</td>
<td>ON</td>
<td>F</td>
<td>OFF</td>
<td>0</td>
<td>4</td>
<td>ON</td>
<td>DBCVTRL</td>
<td>X</td>
<td>X</td>
<td>DBFMTRL</td>
<td>X</td>
<td>X</td>
<td>ON Y</td>
</tr>
<tr>
<td>PLEX$$1 EMPVEH</td>
<td>( )</td>
<td>ON</td>
<td>V</td>
<td>1 4000</td>
<td>ON</td>
<td>5</td>
<td>3</td>
<td>F ON</td>
<td>DBCVTID</td>
<td>X</td>
<td>DBFMID</td>
<td>X</td>
<td>X</td>
<td>ON Y</td>
</tr>
<tr>
<td>PLEX$$2 RECLEN</td>
<td>( )</td>
<td>ON</td>
<td>F</td>
<td>OFF</td>
<td>7</td>
<td>10</td>
<td>*</td>
<td>DBCVTRL</td>
<td>X</td>
<td>X</td>
<td>DBFMTRL</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PLEX$$2 EMPKIDPK</td>
<td>( )</td>
<td>ON</td>
<td>F</td>
<td>OFF</td>
<td>7</td>
<td>10</td>
<td>*</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PLEX$$3 RECLEN</td>
<td>( )</td>
<td>ON</td>
<td>F</td>
<td>OFF</td>
<td>0</td>
<td>4</td>
<td>ON</td>
<td>DBCVTRL</td>
<td>X</td>
<td>X</td>
<td>DBFMTRL</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PLEX$$3 EMPVEHPK</td>
<td>( )</td>
<td>ON</td>
<td>F</td>
<td>OFF</td>
<td>7</td>
<td>10</td>
<td>*</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>( )</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5. Sample Direct Field Descriptors**
<table>
<thead>
<tr>
<th>FLDNAME</th>
<th>SECURITY</th>
<th>READONLY</th>
<th>NAMEFILD</th>
<th>REFORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLEX$$ FILEKEY</td>
<td>X</td>
<td>ON</td>
<td>(EMPNAME)</td>
<td>X</td>
</tr>
<tr>
<td>PLEX$$ EMPTYPE</td>
<td>( )</td>
<td>ON</td>
<td>(EMPPAYCL,EMPINSCL)</td>
<td>( )</td>
</tr>
<tr>
<td>PLEX$$1FILEKEY</td>
<td>X</td>
<td>ON</td>
<td>(EMPNAME)</td>
<td>X</td>
</tr>
<tr>
<td>PLEX$$2KIDID</td>
<td>( )</td>
<td>ON</td>
<td>(KIDNAME,EMPKIDPK)</td>
<td>( )</td>
</tr>
</tbody>
</table>

**FIGURE 6. SAMPLE INDIRECT SECONDARY FIELD DESCRIPTORS**
FIGURE 7. SAMPLE DESCRIPTOR FILE
<table>
<thead>
<tr>
<th>FLDNAME</th>
<th>ASSOCFIL</th>
<th>SUBFILE</th>
<th>SUBCNTRL</th>
<th>INVFILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLEX$$</td>
<td>EMPKIDID</td>
<td>Z</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EMPKIDPK</td>
<td>Z</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EMPKIDRS</td>
<td>Z</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KIDNAME</td>
<td>Z</td>
<td>OFF</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>KIDPET</td>
<td>Z</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KIDID</td>
<td>Z</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RECSEC1</td>
<td>1</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>EMPINSCL</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EMPVEH</td>
<td>1</td>
<td>Y</td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td>EMPVEHID</td>
<td>1</td>
<td>Y</td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td>EMPVEHPK</td>
<td>1</td>
<td>Y</td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td>EMPVEHRS</td>
<td>1</td>
<td>Y</td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td>VEHARCD</td>
<td>1</td>
<td>Y</td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td>VEHMAKE</td>
<td>1</td>
<td>Y</td>
<td>OFF</td>
</tr>
<tr>
<td>PLEX$$1</td>
<td>EMPVEHID</td>
<td>Y</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EMPVEHPK</td>
<td>Y</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EMPVEHRS</td>
<td>Y</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VEHARCD</td>
<td>Y</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VEHMAKE</td>
<td>Y</td>
<td>OFF</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 8. SAMPLE DUMMY FIELD DESCRIPTORS
TOPIC B.8 - DATA EASE EXECUTIVE

A. DATA SET NAME:
   DBPL/I - DBLIST Interface

B. CREATED BY:
   DB Preprocessor Function

C. TYPE OF FILE:
   (4) Table

D. ORGANIZATION:
   Documentary Table

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:

The DBPL/I - DBLIST Interface (see Table 1) specifies the DBLIST entry point name and the argument types and order for the various DBPL/I statements. Thus, it serves to specify for the DB Preprocessor function (see Section IV, Topic B.5) what CALL statements are to be generated for each DBPI/I statement. Conversely, it specifies for DBLIST (see Section IV, Topic B.5) what entry points will be entered and what and how information will be available at execution time for the performance of the various statement actions.

The various entry points and their argument types are declared by source code in SOURCER.LISRMAC member DBTEXT. Any program that includes the DB preprocessor also is given DBIEXT by an INCLUDE statement in DB.
### TABLE 1.

<table>
<thead>
<tr>
<th>DBPL/I</th>
<th>GENERATED PL/I</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREE LIST;</td>
<td>CALL DEPAC;</td>
</tr>
<tr>
<td>FREE LIST (p1,p2);</td>
<td>CALL DBPACP(p1);</td>
</tr>
<tr>
<td>GET LIST (p1) KEY(0);</td>
<td>CALL DBGLKO(p1);</td>
</tr>
<tr>
<td>GET LIST (p1) KEY INTL (st);</td>
<td>CALL DBGLKI(p1,st);</td>
</tr>
<tr>
<td>GET LIST (p1) INTERNAL KEY INTO (st);</td>
<td>CALL DBGLIK(p1,st);</td>
</tr>
<tr>
<td>GET LIST (p1) KEY (n) INTO (st);</td>
<td>CALL DBGLKN(p1,n,st);</td>
</tr>
<tr>
<td>GET LIST (p1) KEY SET (p2);</td>
<td>CALL DBGLKS(p1,p2);</td>
</tr>
<tr>
<td>SET LIST (p2) SIZE (n) LIKE LIST (p1);</td>
<td>CALL DBSLLL(p2,n,p1);</td>
</tr>
</tbody>
</table>

where:
- p1, p2 are POINTER
- n is FIXED BINARY(31)
- st is CHARACTER(-)VARYING
TOPIC C.1 - UTILITIES

A. DATA SET NAME:
NASIS USERIES

B. CREATED BY:
CREATIDS (procdef)

C. TYPE OF FILE:
VISAM

D. ORGANIZATION:
Variable format

E. KEY IDENTIFIER (CCNTRL FIELD):
8 Character NASISIC, key length 8, key position 4.

F. RECORD LENGTH:
4000 bytes

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
Maintain for each NASIS ID, the corresponding password, timeslice, user authority and list of valid files.
TOPIC C.2 - UTILITIES

A. DATA SET NAME:
JOINIDS

B. CREATED BY:
USERJOIN

C. TYPE OF FILE:
VISAM

D. ORGANIZATION:
Variable format

E. KEY IDENTIFIER (CCNTRL FIELD):
A JOINed TSS-ID, key length 8, key position 4.

F. RECORD LENGTH:
30 bytes.

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
This is the list of JOINed TSS-IDs for the program MERGE.
A. DATA SET NAME:
   EDIT.LISRMLF

B. CREATED BY:
   Not Applicable

C. TYPE OF FILE:
   VISAM

D. ORGANIZATION:
   Index Sequential

E. KEY IDENTIFIER (CONTROL FIELD):
   The fifteen byte key is composed of the eight byte message key concatenated to the seven byte line number.

F. RECORD LENGTH:
   V(132)

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   This data set contains a copy of the NASIS system messages used by the various modules for prompting and diagnostic messages. After editing it, the DBA will use this file to replace the current system message file, LISRLIP(0) (LISRMLF).
TOPIC D.1 - MAINTENANCE

A. DATA SET NAME:
   RDBLOAD ERROR_CODES Table

B. CREATED BY:
   RDBLOAD

C. TYPE OF FILE:
   Core table

D. ORGANIZATION:
   Sequential array

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Defined as (0:216) character (1)

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   This table contains codes to be used to control the action taken for each RDBAC error that may occur.
TOPIC D.2 - MAINTENANCE

A. DATA SET NAME:
   TRNSCT Data Set Descriptors

E. CREATED BY:
   CORRECT (RECORR)

C. TYPE OF FILE:
   VI (Indexed) - Anchor

D. ORGANIZATION:
   Indexed Sequential (VISAM)

F. KEY IDENTIFIER (CONTROL FIELD):
   Offset of 4, fixed field (255)

F. RECORD LENGTH:
   4000

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   The purpose of the transaction data base is to serve as a temporary repository for changes to one of the database datafiles, until the datafile owner can validate the change and execute the maintenance program to apply it.

The TRNSCT data base will consist of fields defined as follows:

KEY - This will be an alphanumeric Fixed field 255 bytes in length consisting of the data base name (padded to six characters with $) concatenated with the OWNER-ID, (padded to eight characters with *) of the affected database concatenated with the anchor's key. The last fourteen (14) bytes of the key will be a time stamp field.

NASISID - This will be an alphanumeric Fixed field, 8 bytes in length, and will contain the
NASIS-ID of where the transaction was created.

**OPCODE** - This is an alphanumeric Fixed field, 3 bytes in length, which indicates the operation to be performed.

**FIELD** - This is an alphanumeric Fixed field, 8 bytes in length, which indicates the field of a data base which is to be updated.

**START** - For field context operations, this field will contain the starting location of the context. It will be an alphanumeric Fixed field 4 bytes in length.

**END** - For field context operations, this field will contain the ending location of the context. It will be an alphanumeric Fixed field, 4 bytes in length.

**OLDDATA** - A field which will contain data. This will be the cld data field in the instance of the field context.

**NEWDATA** - A field which will contain data. This will be the new data field in the instance of the field context.

**SUBKEY** - This is an alphanumeric fixed field, 10 bytes in length, which indicates the subfile key to be corrected.

**SUBCTL** - This is an alphanumeric fixed field, 8 bytes in length, which is the subfile control field for subfile being corrected.

One important consideration is that conversion, validation and reformatting routines can be written and used to check the data as it enters the data base, thus causing many errors to be detected before maintenance is ever run.
TOPIC D.3 - MAINTENANCE

A. DATA SET NAME:
   CORRECT Data Display Format

F. CREATED BY:
   CORRECT (RDBCORR)

C. TYPE OF FILE:
   Terminal Display

D. ORGANIZATION:
   Not Applicable

E. KEY IDENTIFIER (CCNTRCl FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   The purpose of this display is to present as much of the data contained in the field as possible to the user.
CORRECT XXXXXXXX, XXXXXXXXXXXXXX

(LENGTH = 80, ELEMENTS = 4)

E001 : XXXXXXX
E002 : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
002  : XXXXXX
E003 : XXXXXXXX
E004 : XXXXXXXX
TOPIC D.4 - MAINTENANCE

A. DATA SET NAME:
   RDBLOAD Error Data Set

B. CREATED BY:
   Not Applicable

C. TYPE OF FILE:
   VISAM or VSAM

D. ORGANIZATION:
   Sequential

E. KEY IDENTIFIER (CONTROL FIELD):
   (Must be the same as that of the RDBLOAD input data set).

F. RECORD LENGTH:
   (Must be the same as that of the RDBLOAD input data set).

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   This data set serves as the repository for all input records that cannot be successfully loaded.
TOPIC D.5 - MAINTENANCE

A. DATA SET NAME:
   Database Inverted Index Format

B. CREATED BY:
   A descriptor region is created by RDBEDIT (the descriptor editor) for each inverted index. It generally consists of a header record and three field descriptor records.

Inverted index records are originally written by either RINVRSS (the sort method of inversion) or RDBINVRT (an inversion utility program) or RDBPAC (inverting concurrently with database loading).

C. TYPE OF FILE:
   (6) Database File

D. ORGANIZATION:
   TSS VISAM - Virtual Indexed Sequential Access Method

E. KEY IDENTIFIER (CCNTRCI FIELD):
   The index key field name is the same as the indexed field name, e.g. AUTHOR, SUBJTERM, KEYWORD etc. The VISAM key length is the maximum (internal) length of the indexed values (plus 1 for a spanned index) and may not exceed 255 bytes.

F. RECORD LENGTH:
   Variable - 4000 bytes maximum.

G. BLOCKING FACTOR:
   VISAM blocks records into pages of 4096 bytes.

H. PURPOSE:
   The purpose of the inverted index files is to give the system a fast, efficient method of storing and accessing the list of records in a database file that contain a particular data element value.

   The records of inverted files consist of a universal record form with a specialized structure. The structure consists of the concatenation of the
following fields:

the VISAM record length field (RECLEN)
4 bytes, fixed length, binary

the VISAM key field
1-254 bytes, fixed length, indexed field element value
optionally suffixed by
1 byte, fixed length, binary record number within region if the index is SPANNED.

the cross references field
2 bytes, fixed length, binary field length
followed by one or more fixed length anchor or subfile internal key values in ascending collating sequence.

In a SPANNED index, the first record of a region has key suffix zero, and possible continuation records (up to 255) in a continuous 'region'. All records in a region, except possibly the last, are maximum length (have the maximum number of whole cross references).
TOPIC D.6 - MAINTENANCE

A. DATA SET NAME:
   Descriptor Editor Data Display Format

E. CREATED BY:
   Descriptor Editor DISPLAY Command (RDBEDDP)

C. TYPE OF FILE:
   Terminal Display

D. ORGANIZATION:
   Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   The purpose of this display is to allow the user of the
   Descriptor Editor to review the specifications for a
   particular field descriptor.

I. SAMPLE DISPLAY:

   FIELD NAME.................xxxxxxxxxx
   FIELD TYPE..................xx
   ALIGNMENT...................x
   FIELD FORMAT................xx
   FIELD LENGTH.................xxxx
   ELEMENT LENGTH...............xxx
   ELEMENT NUMBER...............xxxx
   UNIQUE ELEMENTS..............x
   CONVERSION ROUTINE.........xxxxxxx
   FORMATTING ROUTINE.........xxxxxxx
   VALIDATION ROUTINE.........xxxxxxx
   VALIDATION ARGUMENT........xxxxxxx
                           xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                           xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
INDEX FILE ID
INDEX KEY FORM
EXTERNAL KEY LENGTH
INDEX SPANDED
ASSOCIATE FILE ID
SUBFILE CONTROL FIELD
SUBFILE ID
BASE FIELD NAME
BASE FIELD Offset
SUPERFIELD COMPONENTS

SECURITY
TOPIC D.7 - MAINTENANCE

A. DATA SET NAME:
Descriptor Editor Field Name Display Format

B. CREATED BY:
Descriptor Editor (FIELDS) Command (RDBEDFD)

C. TYPE OF FILE:
Terminal Display

D. ORGANIZATION:
Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
Not Applicable

H. PURPOSE:
The purpose of this display is to allow the user of the Descriptor Editor to review the names of all of the fields described thus far in CREATE mode. In UPDATE mode the user is presented a list of the descriptor descriptor field names.
I. SAMPLE DISPLAY:

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXX XXXXXXXXX XXXXXXXX
XXXXXXXXXXX XXXXXXXXX XXXXXXXX
XXXXXXXXXXX XXXXXXXXX
```

NOTE: A Total of 57 names can be displayed on one screen.
TOPIC D.8 - MAINTENANCE

A. DATA SET NAME:
   RDBLOAD Input Data Set

B. CREATED BY:
   Not Applicable

C. TYPE OF FILE:
   VISAM or VSAM

D. ORGANIZATION:
   Sequential

E. KEY IDENTIFIER (CONTROL FIELD):
   Usually same as that of the data base to be loaded.

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   This data set serves as the source of input records for RDBLOAD.
TOPIC D.9 - MAINTENANCE

A. DATA SET NAME:
   Descriptor Editor Listing Format

B. CREATED BY:
   Descriptor Editor Print Command (RDBEDPR)

C. TYPE OF FILE:
   1403 Printer Display

D. ORGANIZATION:
   Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   133

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   The purpose of this display is to list the contents of each descriptor field and each file descriptor in character form.

I. SAMPLE DISPLAY:
   See Figure 1
TOPIC D.10 - MAINTENANCE

A. DATA SET NAME:
   INVERT Restart File
   'INVERT.PARM.'||FILENAME -
   where FILENAME is the six character dataplex name.

B. CREATED BY:
   RDBSIVRT module.

C. TYPE OF FILE:
   Sequential

D. ORGANIZATION:
   VSAM

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   255 bytes (Variable)

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   This file provides a restart key for the first phase of
   DBSIVRT.
TOPIC D.11 - MAINTENANCE

A. DATA SET NAME:

INVERT SORTIN File

'SORTIN.'||FILENAME'||'.'||FIELD

1. FILENAME is the six character data base name.
2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

First step of BESIVRT.

C. TYPE OF FILE:

Sequential

D. ORGANIZATION:

VSAM

E. KEY IDENTIFIER (CCNTECL FIELD):

First field is the maximum length value of the field being inverted.

F. RECORD LENGTH:

4000 bytes (Variable). Record consists of maximum length value of field being inverted concatenated with file key.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to the second step, which is the TSS sort step of LESIVRT.
TOPIC D.12 - MAINTENANCE

A. DATA SET NAME:

INVERT SORTOUT File
'SORTOUT.'||FILENAME||'.'||FIELD

1. FILENAME is the six character data base name.
2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

Sort step of DBSIVRT.

C. TYPE OF FILE:

Sequential

D. ORGANIZATION:

VSAM

E. KEY IDENTIFIER (CCNTRCI FIELD):

First field is the maximum length value of the field being inverted.

F. RECORD LENGTH:

4000 bytes (Variable). Record consists of maximum length value of field being inverted concatenated with the file key.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to step three of DBSIVRT.
A. DATA SET NAME:

INVERT PLEX File

'PLEX.'\|\|FILENAME\|\|'.\|\|FIELD

1. FILENAME is the six character dataplex name.

2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

Step three of DBSIVRT.

C. TYPE OF FILE:

Indexed Sequential

D. ORGANIZATION:

VISAM

E. KEY IDENTIFIER (CONTROL FIELD):

Key of file is internal field value being inverted concatenated with blanks up to the maximum external field length. If index file is spanned, span character is concatenated as last position of Key.

F. RECORD LENGTH:

4000 bytes (Variable). Record is identical in format as an index file record.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to the last step, translation step, of DBSIVRT.
TOPIC D.14 - MAINTENANCE

A. DATA SET NAME:

INVERT RANGE file
' RANGE,' ||FILENAME||'.','||FIELD

1. FILENAME is the six character data base name.
2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

Step three of DESIVRT if not indexed external, if indexed external, step four creates this data set.

C. TYPE OF FILE:

Indexed Sequential

D. ORGANIZATION:

VISAM

E. KEY IDENTIFIER (CONTROL FIELD):

Key of the file is the maximum length value of the field being inverted. If index file is spanned, span character is concatenated as last position of Key.

F. RECORD LENGTH:

4000 bytes (Variable). Record is identical to index file record.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to the combine module for index updates.
TOPIC D.15 - MAINTENANCE

A. DATA SET NAME:
   DESCRIPTOR.CHKPOINT

B. CREATED BY:
   Descriptor Editor Checkpoint (RDBEDCP)

C. TYPE OF FILE:
   TSS VAM

D. ORGANIZATION:
   Sequential

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   The records are of varying length of which the maximum
   is dynamically determined at execution time. The
   maximum possible value is 4000.

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   The purpose of this dataset is for storing sufficient
   information from the descriptor tables so that the user
   can continue creating the descriptor file at a future
   time through use of the RESTORE command.

   The first record consists of those items from the X
   structure whose value must be preserved. The second
   record consists of the entire content of the FIELD
   structure. The next group of records will contain the
   field descriptor information. There will be one record
   for each existing field, consisting of the information
   in the appropriate FIELD structure concatenated with the
   information contained in the appropriate SECURITY,
   SUPER, and VALID structures where applicable.
   Following the field descriptor records are records
   containing the header descriptor information, one for
   each existing file. These records consist of the
   information from the appropriate HDR structure
concatenated to the information from the proper RECSEC structure when applicable.
TOPIC D.16 - MAINTENANCE

A. DATA SET NAME:

MERGE INDEX File

'INDMRG.PARM.'//FILENAME - FILENAME is the six character data base name.

B. CREATED BY:

RDBINDM MODULE

C. TYPE OF FILE:

SEQUENTIAL

D. ORGANIZATION:

VSAM

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

255 Bytes (variable)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file provides a restart key for restart of DBINDM.
TOPIC D.17 - MAINTENANCE

A. DATA SET NAME:
   Descriptor Editor REVIEW Display Format

B. CREATED BY:
   Descriptor Editor REVIEW Command (RDBEDRV)

C. TYPE OF FILE:
   Terminal Display

D. ORGANIZATION:
   Not Applicable

E. KEY IDENTIFIER (CCNTRL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   The purpose of this display is to allow the user of the Descriptor Editor to review the exact contents of any descriptor record in any descriptor region.
### I. Sample Displays:

**File Descriptor**

<table>
<thead>
<tr>
<th>SUFFIX</th>
<th>FILETYPE=</th>
<th>DESCRIPT =</th>
<th>BSENGTH =</th>
</tr>
</thead>
<tbody>
<tr>
<td>=x</td>
<td></td>
<td>=xxxxx</td>
<td>xxxxx</td>
</tr>
<tr>
<td>DESCOK</td>
<td></td>
<td>=x</td>
<td></td>
</tr>
<tr>
<td>=x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNTNING</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECSECFP=</td>
<td></td>
<td>REMAINS =</td>
<td></td>
</tr>
<tr>
<td>xxxxx</td>
<td></td>
<td>=xxxxxxx</td>
<td></td>
</tr>
</tbody>
</table>

**RSECTYCD=**

```
xxxxxxxxxx:xx
  xxxxxxxxx:xx
  xxxxxxxxx:xx
  xxxxxxxxx:xx
  xxxxxxxxx:xx
  xxxxxxxxx:xx
  xxxxxxxxx:xx
  xxxxxxxxx:xx
  xxxxxxxxx:xx
  xxxxxxxxx:xx
```

```
<table>
<thead>
<tr>
<th>FIELD DESCRICTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>________________</td>
</tr>
<tr>
<td>FLDNAME =xxxxxxx</td>
</tr>
<tr>
<td>SUBFILE =x</td>
</tr>
<tr>
<td>READONLY=x</td>
</tr>
<tr>
<td>VARFLD =x</td>
</tr>
<tr>
<td>NUMALIGN=x</td>
</tr>
<tr>
<td>UNIQUELT=x</td>
</tr>
<tr>
<td>GENERCRT=xxxxxxx</td>
</tr>
<tr>
<td>REFORMAT=xxxxxxx</td>
</tr>
<tr>
<td>FLDLEN =xxxxx</td>
</tr>
<tr>
<td>FLDLEN =xxxxx</td>
</tr>
<tr>
<td>ELTLEN =xxxxx</td>
</tr>
</tbody>
</table>
| SPARE =xxxxxxxxx | VALIDARG=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
TOPIC D.18 - MAINTENANCE

A. DATA SET NAME:
DEFIELD which consists of the external structure FIELD

B. CREATED BY:
Not Applicable

C. TYPE OF FILE:
Table

D. ORGANIZATION:
PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
Not Applicable

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
This table is used to contain the names and core locations of all the field descriptor during the running of a retrieval session.
I. PL/I DECLARATION:

THIS STRUCTURE IS USED TO CONTAIN THE FIELD NAMES AND THEIR RESPECTIVE FLD POINTERS.

1 FIELD BASED (X.FIELD_PTR), /* FIELD NAMES AND */
    /* POINTERS STRUCTURE. */
3 RECLLEN BIN (31) FIXED, /* RECORD LENGTH FOR */
    /* ASMPUT. THIS IS USED IN */
    /* CHKPOINT COMMAND. IT IS */
    /* SET EQUAL TO ELT LENGTH OF */
    /* FIELD STRUCTURE. */
3 LAST BIN FIXED, /* INDEX OF LAST TABLE ENTRY. */
3# BIN FIXED, /* NUMBER OF ENTRIES IN TABLE. */
3 A (X.#FN REFER (FIELD.#)),
    5 NAME CHAR (8), /* FIELDNAME ARRAY. */
7 PTR PTR; /* FLD STRUCTURE POINTERS. */
TOPIC D.19 - MAINTENANCE

A. DATA SET NAME:

DERECSEC which consists of the structures RECSEC and RECSEC STR

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The RECSEC structure is used to contain the record security codes that pertain to a given file. RECSEC_STR is a character string overlay of the RECSEC structure.
I. **PL/I DECLARATION:**

The RECSEC structure is used to store the record security codes and security masks used to determine record security. The RECSEC structure is pointed to be HDR.RSECTYCD field when the file has record security defined on it.

```pli
1 RECSEC BASED (X.RSEC_PTR), /* RECORD SECURITY */
   /* CODES STRUCTURE. */
3 # BIN FIXED, /* NUMBER OF SECURITY CODES. */
3 SECURITY (18), /* USER PASSWORD. */
   5 MASK CHAR (8), /* RECORD ACCESS CODE. */
3 CHANGED (18) BIT (1), /* ONE FLAG FOR EACH SECURITY */
   /* CODE. IF ON THEN REPUT NEW */
   /* VALUE. */
3 FILLER CHAR (8); /* NEEDED FOR PLI BUG. */

This structure is a character string overlay of the RECSEC structure. It is used for making copies of the RECSEC structure.

DCL RECSEC_STR CHAR (193) BASED (X.RSEC_PTR);
   /* RECSEC STRUCTURE OVERLAY. */
```
TOPIC D.20 - MAINTENANCE

A. DATA SET NAME:
DESECCUR which consists of the structure SECURITY and SECURITY_STR.

B. CREATED BY:
Not Applicable

C. TYPE OF FILE:
Table

D. ORGANIZATION:
PL/I Data Structure

E. KEY IDENTIFIER (CCNTRCI FIELD):
Not Applicable

F. RECORD LENGTH:
Not Applicable

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
The SECURITY structure is used to contain the security codes defining field security for a given field. SECURITY_STR is a character string overlay of the SECURITY structure.
I. PL/I DECLARATION

THE SECURITY STRUCTURE IS USED TO STORE THE FIELD SECURITY CODES DEFINED FOR A GIVEN FIELD. IT IS POINTED TO BY THE FLD.SECURITY FIELD ON WHICH THIS FIELD SECURITY IS DEFINED.

```pli
1 SECURITY BASED (X.FSEC_PTR), /* FIELD SECURITY */  
   /* CODES STRUCTURE. */  
3 # BIN FIXED, /* NUMBER OF SECURITY CODES */  
   /* FOR THIS FIELD. */  
3 CODE (18) CHAR (8), /* SECURITY CODE VALUES. */  
3 CHANGED (18) BIT (1), /* ONE FLAG FOR EACH SECURITY */  
   /* CODE. IF ON THEN REPUT THE */  
   /* NEW VALUE. */  
3 FILLER CHAR (8); /* NEEDED FOR PL/I BUG. */
```

THIS STRUCTURE IS A CHARACTER STRING OVERLAY OF THE SECURITY STRUCTURE. IT IS USED FOR MAKING COPIES OF THE SECURITY STRUCTURE.

```pli
DCL SECURITY_STR CHAR (157) BASED (X.FSEC_PTR);  
   /* SECURITY STRUCTURE OVERLAY. */
```
TOPIC D.21 - MAINTENANCE

A. DATA SET NAME:
   DESUPER which consists of the structure SUPER and SUPER_STR.

B. CREATED BY:
   Not Applicable

C. TYPE OF FILE:
   Table

D. ORGANIZATION:
   PL/I Data Structure.

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   The SUPER structure is used to contain the superfield component information of a field descriptor. SUPER_STR is a character string overlay of the SUPER structure.
I. PL/I DECLARATION:

THE SUPER STRUCTURE IS USED TO STORE THE SUPERFIELD COMPONENTS OF A SUPER DESCRIPTOR. IT IS POINTED TO BY THE FLD.NAMEFLD OF THE DEFINING SUPERFIELD.

1 SUPER BASED (X.SUPER_PTR), /* SUPER FIELD */
    /* COMPONENT FIELDNAMES */
    /* STRUCTURE. */
3 # EIN FIXED, /* NUMBER OF COMPONENT NAMES. */
3 NAME (16),
    5 CODE CHAR (1), /* INTERNAL-EXTERNAL INDICATOR*/
    5 FIELD CHAR(8), /* COMPONENT FIELD NAMES. */
3 CHANGED (16) BIT (1), /* ONE FLAG FOR EACH COMPONENT*/
    /* NAME. IF ON THEN REPUT THIS*/
    /* COMPONENT NAME. */
3 FILLER CHAR (8); /* NEEDED FOR PL/I BUG. */

THIS STRUCTURE IS A CHARACTER STRING OVERLAY OF THE SUPER STRUCTURE. IT IS USED FOR MAKING COPIES OF THE SUPER STRUCTURE.

DCL SUPER_STR CHAR (156) BASED (X.SUPER_PTR); /* SUPER STRUCTURE OVERLAY. */
TOPIC D.22 - MAINTENANCE

A. DATA SET NAME:
DEVALID which consists of the structure VALID

B. CREATED BY:
Not Applicable

C. TYPE OF FILE:
Table

D. ORGANIZATION:
PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
Not Applicable

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
This structure is used to contain the validation argument for a field descriptor.
I. PL/I DECLARATION:

THE VALID STRUCTURE IS USED TO STORE A VALIDATION ARGUMENT IF ONE IS DEFINED FOR THE FIELD. IT IS POINTED TO BY FLD.VALIDARG IN THE FIELD TO WHICH THIS ARGUMENT BELONGS.

1 VALID BASED (X.ARG_PTR), /* VALIDATION ARGUMENT */
    /* STRUCTURE. */
3 LENGTH BIN FIXED, /* LENGTH OF VALIDATION */
    /* ARGUMENT. */
3 ARGUMENT CHAR (X.LVA REFER (VALID,LENGTH));
    /* VALIDATION ARGUMENT. */
A. DATA SET NAME:
   DEFLD which consists of the based structures FLD and FLD STRING

B. CREATED BY:
   Not Applicable

C. TYPE OF FILE:
   Table

D. ORGANIZATION:
   PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   The FLD structure is used to contain the information describing a field descriptor. FLD_STRING is a character string used to overlay the FLD structure.
I. **PL/I DECLARATION:**

This structure is used to store the information defining a field descriptor.

```plaintext
1 FLD BASED (X.FID PTR), /* FIELD DESCRIPTOR */
   /* STRUCTURE. */
3 BACKWARD PTR, /* BACKWARD FLD POINTER. */
3 FORWARD PTR, /* FORWARD FLD POINTER. */
3 FLDNAME CHAR (8),  /* FIELD NAME. */
3 ASSOCFIL CHAR (1), /* ASSOCIATE FILE IDENTIFIER. */
3 SUBFIL CHAR (1),  /* SUBFILE IDENTIFIER. */
3 INVFILE CHAR (1), /* INVERTED FILE IDENTIFIER. */
3 READONLY CHAR (1), /* FIELD READ ONLY FLD. */
3 SUBCTRL CHAR (1), /* FIELD A SUBFILE CONTROL FLD. */
3 VARFLD CHAR (1), /* VARYING LENGTH FIELD FLD. */
3 BITFLD CHAR (1), /* BIT STRING OF LENGTH ONE. */
3 NUMALIGN CHAR (1), /* FIELD ALIGNMENT FLD. */
3 VARELT CHAR (1), /* ELEMENTS OF VARYING LENGTH FLD. */
3 UNIQUELT CHAR (1), /* ELEMENTS UNIQUE FLD. */
3 INDEXEXT CHAR (1), /* INDEX KEYS TO BE IN INTERNAL OR EXTERNAL FORM. */
3 FILLER CHAR (1), /* BOUNDARY ALIGMENT. */
3 GENERCRT CHAR (8), /* CONVERSION ROUTINE NAME. */
3 VALIDRTN CHAR (8), /* VALIDATION ROUTINE NAME. */
3 REFORMAT CHAR (8), /* FORMATTING ROUTINE NAME. */
3 SPARE CHAR (16),  /* UNUSED DESCRIPTOR FIELD. */
3 FLDPOSIT BIN FIXED, /* FIELD POSITION VALUE. */
3 FLDLEN BIN FIXED, /* FIELD LENGTH VALUE. */
3 DFLDLEN BIN FIXED, /* MAXIMUM FIELD LENGTH OF ALL VALUES STORED ON THE DATA BASE. */
3 ELTLIM BIN FIXED, /* MAX NUMBER OF ELEMENTS/FLD. */
3 DELTLM BIN FIXED, /* MAXIMUM ELEMENTS STORED IN THIS FIELD IN THE DATA BASE. */
3 ELTLLEN BIN FIXED, /* ELEMENT LENGTH VALUE. */
3 DELTLENN BIN FIXED, /* MAXIMUM ELEMENT LENGTH OF ALL THE ELEMENTS STORED FOR THIS FIELD IN THE DATA BASE. */
3 VALIDARG PTR, /* POINTER TO VALIDATION ARGUMENT IF ANY. */
3 NAMEFLD PTR, /* POINTER TO LIST OF FIELD NAMES MAKING UP SUPER FIELD. */
3 SECURITY PTR, /* POINTER TO FIELD SECURITY CODES IF ANY. */
3 BASEFLD CHAR (8), /* THE FIELDNAME ON WHICH A SUBFIELD IS TO BE DEFINED. */
3 OFFSET BIN FIXED, /* THE OFFSET WITHIN THE BASE FIELD THAT THE SUBFIELD */
```
3 FILE_LIST BIN FIXED, /* ON WHICH ENTRY IN FLD_TAB */
     /* HAS THIS FIELD BEEN HUNG. */
3 FLDTYPE BIN FIXED, /* ENTRY INTO FIELD TYPE TABLE*/
     /* DEFINING WHICH TYPE OF */
     /* FIELD THIS IS. */
3 CHANGED (28) BIT (1), /* ONE FLAG FOR EACH ITEM IN */
     /* FLD STRUCTURE. IF ON THEN */
     /* PUT NEW VALUE IN DESCRIPTOR*/
     /* FILE. */
3 FILLER2 CHAR (8);/* NEEDED FOR PL/I BUG. */

THIS STRUCTURE IS A CHARACTER STRING OVERLAY ON THE FLD
STRUCTURE. IT IS USED FOR MAKING COPIES OF THE FLD
STRUCTURE.

DCL FLD_STRING CHAR (122) BASED (X.FLD_PTR);
     /* FLD STRUCTURE OVERLAY. */
TOPIC D.24 - MAINTENANCE

A. DATA SET NAME:
   DEXINIT which consists of the X external data structure including all initialization values.

B. CREATED BY:
   Not Applicable

C. TYPE OF FILE:
   Table

D. ORGANIZATION:
   PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   The X structure is used to contain common variables and information used to control the flow through the descriptor editor
TOPIC D.25 - MAINTENANCE

A. DATA SET NAME:

DEX which consists of the X external data structure minus all initialization values.

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The X structure is used to contain common variables and information used to control the flow through the descriptor editor.
I. PL/I DECLARATION:

THE X STRUCTURE IS A COLLECTION OF MINOR STRUCTURES AND SINGLE VARIABLES USED IN THE RUNNING OF THE DESCRIPTOR EDITOR. THESE MINOR STRUCTURES CONSIST OF PREDEFINED FLD, HDR, RECSEC SECURITY, AND SUPER STRUCTURES, AS WELL AS THE INPUT OUTPUT WORK AREAS FOR THE VARIOUS STRUCTURES. THE OTHER MINOR STRUCTURES ARE A LIST OF RESERVED FIELD NAMES AND A LIST OF FIELDS WHICH ARE TO BE DELETED FROM THE DESCRIPTOR FILE WHEN THE CURRENT DESCRIPTORS ARE FILED TO MAKE THE DESCRIPTOR FILE ACCURATE.

1 X

EXT CTL, /* EXTERNAL CONTROLLED */
/* BLOCKED VARIABLES. */

THIS MINOR STRUCTURE IS THE PREDEFINED COMMENTS FIELD DESCRIPTOR.

3 FLD_COMMENTS LIKE FLD,/*COMMENTS FIELD DESCRIPTOR*/

THIS MINOR STRUCTURE IS THE PREDEFINED FREEFORM FIELD DESCRIPTOR.

3 FLD_FREEFORM LIKE FLD, /*USER ENTERED KEYWORDS */

THIS MINOR STRUCTURE IS THE PREDEFINED RECORD SECURITY FIELD DESCRIPTOR.

3 FLD_RS LIKE FLD, /* RECORD SECURITY DESCRIPTOR */

THIS MINOR STRUCTURE IS THE PREDEFINED BASE FOR A SUBFILE CONTROL FIELD DESCRIPTOR.

3 FLD_SUBCNTRL LIKE FLD,

THIS MINOR STRUCTURE IS THE PREDEFINED BASE FOR A SUBFILE KEY FIELD DESCRIPTOR.

3 FLD_SUBKEY LIKE FLD,

THIS MINOR STRUCTURE IS THE PREDEFINED BASE FOR A SUBFILE PARENT KEY FIELD DESCRIPTOR.

3 FLD_SUBPK LIKE FLD,

THIS MINOR STRUCTURE IS THE PREDEFINED HEADER DESCRIPTOR RECORD FOR THE ASSOCIATE FILE CONTAINING COMMENTS AND FREEFORM FIELD DESCRIPTORS.

3 HDR ASSOC LIKE HDR, /* HEADER FOR COMMENTS AND */
THIS MINOR STRUCTURE IS THE PREDEFINED HEADER DESCRIPTOR RECORD THE INDEX FILE ON WHICH THE FIELD FREEFORM IS INDEXED.

3 HDR_INDEX LIKE HDR,
    /* INDEX FILE HEADER FOR USER */
    /* KEYWORDS STORED IN FREEFORM*/

THIS MINOR STRUCTURE IS A PREDEFINED INITIALIZED FLD STRUCTURE. IT IS USED TO INITIALIZE A NEWLY ALLOCATED FLD STRUCTURE.

3 INIT_FLD LIKE FLD,
    /* FIELD DESCRIPTOR INITIAL */
    /* VALUES. */

THIS MINOR STRUCTURE IS A PREDEFINED INITIALIZED HDR STRUCTURE. IT IS USED TO INITIALIZE A NEWLY ALLOCATED HDR STRUCTURE.

3 INIT_HDR LIKE HDR,
    /* HEADER DESCRIPTOR INITIAL */
    /* VALUES. */

THIS MINOR STRUCTURE IS A PREDEFINED INITIALIZED SECURITY STRUCTURE. IT IS USED TO INITIALIZE A NEWLY ALLOCATED SECURITY STRUCTURE.

3 INIT_SECURITY, /* FIELD SECURITY STRUCTURE */
    /* INITIAL VALUES. */
    5 #
    5 EIN FIXED,
    5 CODE (18) CHAR (8),
    5 CHANGED (18) BIT (1),
    5 FILLER CHAR (8), /* NEEDED FOR PFL BUG. */

THIS MINOR STRUCTURE IS USED FOR ALL IO OPERATIONS TO AND FROM THE DESCRIPTOR FILE INVOLVING FIELD DESCRIPTOR RECORDS. ALL FIELD DESCRIPTOR DATA FROM THE DESCRIPTOR FILE IS PLACED INTO THIS WORK AREA BEFORE BEING MOVED TO AN ALLOCATED FLD STRUCTURE. BEFORE OUTPUTTING TO A FIELD DESCRIPTOR ON THE DESCRIPTOR FILE, THE FIELD INFORMATION IS MOVED INTO THE IO_FLD STRUCTURE. THIS IS NECESSARY BECAUSE DBPAC REQUIRES ALL IO INTO AND FROM TO BE DONE FROM VARYING LENGTH CHARACTER STRINGS.

3 IO_FLD, /* FIELD DESCRIPTOR WORK AREA */
    /* STRUCTURE. */
    5 BACKWARD PTR, /* BACKWARD FIELD POINTER. */
    5 FORWARD PTR, /* FORWARD FIELD POINTER. */
    5 FLDNAME CHAR (8) VAR, /* FIELD NAME. */
    5 ASSCFIL CHAR (1) VAR, /* ASSOCIATE FILE ID. */
    5 SUBFILE CHAR (1) VAR, /* SUBFILE IDENTIFIER. */
5 INVFILE CHAR (1) VAR, */ INVERTED FILE ID. */
5 READONLY CHAR (1) VAR, */ FIELD READ ONLY FLAG. */
5 SUBCNRCL CHAR (1) VAR, */ FIELD IS A SUBFILE */
           /* CONTROL FIELD. */
5 VARFLD CHAR (1) VAR, */ VARGING LENGTH FIELD. */
5 BITFLD CHAR (1) VAR, */ FIXED LENGTH BIT */
           /* STRING OF LENGTH ONE. */
5 NUMALIGN CHAR (1) VAR, */ FIELD ALIGNMENT FLAG. */
5 VARELT CHAR (1) VAR, */ FIELD ELEMENTS OF */
           /* LENGTH FLAG. */
5 UNIQELT CHAR (1) VAR, */ ELEMENTS UNIQUE FLAG. */
5 INDEXEXT CHAR (1) VAR, */ INDEX KEYS TO BE IN */
           /* INTERNAL OR EXTERNAL FORM */
           /* FLAG. */
5 FILLER CHAR (1) VAR, */ BOUNDARY ALIGNMENT. */
5 GENERCRT CHAR (8) VAR, */ CONVERSION RTN NAME. */
5 VALIDRTN CHAR (8) VAR, */ VALIDATION RTN NAME. */
5 REFORMAT CHAR (8) VAR, */ FORMATTING RTN NAME. */
5 SPARE CHAR (8) VAR, */ UNUSED DESCRIPTOR FIELD. */
5 FLDPOSIT CHAR (2) VAR, */ FIELD POSITION VALUE. */
5 FLDLEN CHAR (2) VAR, */ FIELD LENGTH VALUE. */
5 DELFIELD CHAR (2) VAR, */ MAXIMUM FIELD LENGTH. */
5 ELTLM CHAR (2) VAR, */ MAX NUMBER OF */
           /* ELEMENTS / FIELD. */
5 DELTLM CHAR (2) VAR, */ MAXIMUM # OF ELEMENTS*/
5 ELEN LEN CHAR (2) VAR, */ ELEMENT LENGTH VALUE. */
5 DELTLEN CHAR (2) VAR, */ MAXIMUM ELEMENT LENGTH*/
5 VALIARG PTR, /* POINTER TO VALIDATION */
           /* ARGUMENT IF ANY. */
5 NAMEFLD PTR, /* POINTER TO LIST OF FIELD */
           /* NAMES MAKING UP SUPER FIELD*/
5 SECURITY PTR, /* POINTER TO FIELD SECURITY */
           /* CODES IF ANY. */
5 BASEFLD CHAR (8), /* SUBFIELD DEFINING BASE. */
5 OFFSET BIN FIXED, */ OFFSET IN BASEFIELD */
           /* SUBFIELD IS TO START. */
5 FLDLIST BIN FIXED, */ WHICH ENTRY IN FLD_TAB. */
5 FLDTYPE BIN FIXED, */ TYPE OF FIELD. */
5 CHANGED (28) BIT (1), */ ONE FLAG FOR EACH ITEM */
           /* IN FLD STRUCTURE. IF ON */
           /* THEN PUT NEW VALUE IN */
           /* DESCRIPTOR FILE. */
5 FILLER2 CHAR (8), /* NEEDED FOR PLI BUG. */

THIS MINOR STRUCTURE IS USED FOR ALL IO OPERATIONS TO
AND FROM THE DESCRIPTOR FILE INVOLVING HEADER RECORDS.
ALL INPUT FROM THE DESCRIPTOR FILE INVOLVING HEADER
RECORDS IS PlACED IN THE IO_HDR STRUCTURE BEFORE BEING
MOVED TO AN ALLOCATED HDR STRUCTURE. TO OUTPUT A
HEADER RECORD, THE INFORMATION IS MOVED INTO THE IO-HDR
STRUCTURE BEFORE BEING PLACED ON THE FILE. THIS IS
NECESSARY BECAUSE DBPAC REQUIRES ALL IO TO BE DONE IN
INTO AND FROM VARYING LENGTH CHARACTER STRINGS.
3 IO_HDR, /* HEADER DESCRIPTOR WORK AREA*/
5 BACKWARD PTR, /* BACKWARD HEADER POINTER. */
5 FORWARD PTR, /* FCPWARE HEADER POINTER. */
5 SUFFIX CHAR (1) VAR,/* WHICH FILE THIS */
/* HEADER BELONGS TO. */
5 FILETYPE CHAR (1) VAR,/* TYPE OF FILE INDICATOR*/
5 DESCRT CHAR (2) VAR,/* NUMBER OF FIELD */
/* DESCRIPTORS ON THIS FILE. */
5 BSLENTH CHAR (2) VAR,/* TOTAL LENGTH OF FIXED*/
/* FIELDS ON THIS FILE. */
5 DESCCK CHAR (1) VAR,/* DESCRIPTORS ON FLAG. */
5 SPANNED CHAR (1) VAR,/* THIS INDEX TO CONSIST*/
/* OF SPANNED RECORDS FLAG. */
5 DATA CHAR (1) VAR,/* DATA IS ON FILE FLAG. */
5 MNTNABLE CHAR (1) VAR,/* FILE CAN BE */
/* MAINTAINED FLAG. */
5 MNTNING CHAR (1) VAR,/* FILE BEING MAINTAINED*/
/* FLAG. */
5 LOADABLE CHAR (1) VAR,/* FILE CAN BE LOADED. */
5 REMAINS CHAR (4) VAR,/* UNUSED DESCRIPTOR FLD*/
5 RECSECFF CHAR (2) VAR,/* FILE HAS RECORD */
/* SECURITY FLAG. */
5 RSECTYCD PTR, /* POINTER TO RECORD */
/* SECURITY CODES IF ANY. */
5 CHANGED (13) BIT (1),/* ONE FLAG FOR EACH ITEM */
/* IN HEADER STRUCTURE. IF */
/* ON THEN PUT NEW VALUE */
/* IN THE DESCRIPTOR FILE. */
5 FILLER CHAR (8),/* NEEDED FOR PLI BUG. */

THIS MINOF STRUCTURE IS USED FOR ALL IO OPERATIONS TO
AND FROM THE DESCRIPTOR FILE INVOLVING FIELD SECURITY
CODES.

3 IO_SECURITY, /* FIELD SECURITY STRUCTURE. */
5 # BIN FIXED,/* NUMBER OF SECURITY CODES*/
/* FOR THIS FIELD. */
5 CODE (18) CHAR (8) VAR,/* USER PASSWORD. */
5 CHANGED (18) BIT (1),/* ONE FLAG FOR EACH */
/* SECURITY CODE. IF ON THEN */
/* REPUT THE NEW VALUE. */
5 FILLER CHAR (8),/* NEEDED FOR PLI BUG. */
3 GF, /* PARAMETERS TO GET FIELD */
/* SUBROUTINE. */
5 ALLOC_NEW BIT (1),/* ALLOCATE AND INITIALIZE */
/* NEW FLD STRUCTURE. */
5 FLD_LEN BIN FIXED,/* MAXIMUM ALLOWABLE LENGTH*/
/* FOR THE FIELDNAME. */
5 FLD_MSG CHAR (8),/* MSGID TO PROMPT FOR THE */
/* FIELDNAME. */
5 FLD# BIN FIXED,/* FOR AN EXISTING FIELD, */
/* THE ENTRY IN FIELD STRUCTURES*/
/* ELSE 0. */
**NEW_FLD** BIT (1), /* ON - GET A BRAND NEW FIELD*/
/* CFF - AN EXISTING FIELD. */

**PRMPT_ERR** BIT (1), /* VALUE TO SET TC.PROMPT.ERR*/
**RESERVED** BIT (1), /* A RETURNED VALUE INDICATING*/
/* IF THE FIELDNAME IS */
/* RESERVED. */

**RES_FLD** BIT (1), /* ON - RESERVED NAMES ARE */
/* ACCEPTABLE. */
/* OFF - RESERVED NAMES ARE */
/* NOT ACCEPTABLE. */

**ADD_FLAG** BIT (1), /* IN ADD OR CHANGE COMMAND. */
**ALPHA** CHAR (26), /* ALL ACCEPTABLE ALPHABETIC */
/* CHARACTERS. */

**ALPHANUMERIC** CHAR (36),
/* ALL ACCEPTABLE ALPHA- */
/* NUMERIC CHARACTERS. */

**ARG_PTR** PTR, /* PTR TO VALIDATION ARGUMENT. */
**ASSOC_NAMES** CHAR (9), /* ALL POSSIBLE ASSOCIATE */
/* FILE ID'S. */

**COMND_CALL** BIT (1), /* A COMMAND CALL OR AN */
/* INTERNAL CALL. */

**COMND_NAME** CHAR (8), /* NAME OF COMMAND CALLED. */
**ERR_FLAG** BIT (1), /* ERROR FLAG USED FOR */
/* INTER MODULE COMMUNICATION. */

**FIELDNAME** CHAR (8), /* FIELD TO BE PROCESSED. */
**FIELDTYPE** (0:10) CHAR (2), /* ALL VALID FIELD TYPES. */
**FIELD_PTR** PTR, /* PTR TO FIELD NAME STRUCTURE*/

**FLD_LAST** (60) PTR, /* PTRS TO LAST FIELD ENTRY */
/* IN EACH FIELD STRING. */

**FLD_PTR** PTR, /* PTR TO FIELD DESCRIPTOR. */
**FLD_TAB** (60) PTR, /* PTRS TO FIRST FIELD ENTRY */
/* IN EACH FIELD STRING. */

**FLDTYPE** BIN FIXED, /* FIELD TYPE USED BY ADD. */
**FSEC_PTR** PTR, /* PTR TO FIELD SECURITY */
/* STRUCTURE. */

**HDR_PTR** PTR, /* PTR TO FIELD DESCRIPTOR. */
**HEAD_TAB** (36) PTR, /* CNE PTR FOR EACH HEADER. */

**HEX_CHARS** CHAR (16),
/* ALL ACCEPTABLE HEXADECIMAL */
/* CHARACTERS. */

**INDEX_NAMES** CHAR (16),
/* LIST OF ALL POSSIBLE INDEX */
/* FILE ID'S. */

**IOAREA** CHAR (256) VAR, /* COMMON TERMINAL INPUT */
/* OUTPUT AREA. */

**LOAD_FILE** CHAR (1), /* ID OF FILE TO LOAD FROM. */
**LOAD_ONE** BIT (1), /* LOAD JUST CNE RECORD. */
**LVA** BIN FIXED, /* LENGTH OF VALIDATION */
/* ARGUMENT. */

**PAT_FILE** CHAR (1), /* ID OF FILE BEING WORKED */
/* ON BY REVIEW - PATCH. */

**PAT_FIELD** CHAR (8), /* NAME OF FIELD BEING */
/* WORKED ON BY REVIEW-PATCH. */
3 REV_MODE BIT (1), /* IN REVIEW OR UPDATE MODE. */
3 RSEC_PTR PTR, /* PTR TO RECORD SECURITY */
    /* CODES. */
3 SAVE_STRING CHAR (150) VAP, /* AREA TO BUILD */
    /* COMMAND STRINGS. */
3 SUBFILE_NAMES CHAR (10),
    /* LIST OF ALL POSSIBLE */
    /* SUB-FILE ID'S. */
3 SUFFIX CHAR (36),
    /* ALL POSSIBLE FILE */
    /* IDENTIFIER SUFFICIES. */
3 SUPER_PTR PTR, /* PTR TO SUPERFIELD COMPONENT*/
3 TRANS, /* TRANSITORY CALL LABELS. */
    5 CALL CHAR (8), /* ROUTINE TO BE CALLED */
    5 RET CHAR (8), /* ROUTINE TO RETURN TO. */

THIS SUBSEQUENT PART OF THE $X$ STRUCTURE IS SEPARATED FROM THE REST OF THE $X$ ITEMS AS THIS PART OF $X$ IS THE PART THAT MUST BE SAVED WHEN USING THE CHKPOINT COMMAND. THIS PREVIOUS INFORMATION OF $X$ NEED NOT BE SAVED, AS IT IS SETUP PROPERLY WHENEVER $X$ IS ALLOCATED, OR THOSE ITEMS WHOSE VALUES MATTERS NOT BETWEEN COMMAND EXECUTION.

3 CHKPOINT_RECLN EIN (31) FIXED,
    /* OUTPUT RECORD LENGTH FOR */
    /* ASMPUT ROUTINE. IT IS SET */
    /* SET TO THE LENGTH OF THE */
    /* $X$ STRUCTURE THAT MUST BE */
    /* SAVED WHEN CHECKPOINT IS */
    /* EXECUTED. */

THIS MINOR STRUCTURE IS THE PREDEFINED RECLN FIELD DESCRIPTOR. IT IS PLACED IN IN THIS PART OF $X$ BECAUSE IT MAY HAVE FIELD SECURITY APPLIED TO IT.

3 FLD_RECLN LIKE FLD,

THIS MINOR STRUCTURE IS A LIST OF RESERVED FIELDNAMES. THE USER MAY NOT DEFINE BY USE OF THE ADD, SUPERFLD, CREATSUP, ADELIKE, AND RENAME COMMANDS A FIELD DESCRIPTOR WITH A FIELDNAME THAT APPEARS IN THIS TABLE.

3 RESERVED, /* LIST OF RESERVED FIELDNAMES*/
    5 LAST_# BIN FIXED INIT (14), /* INDEX OF LAST ENTRY. */
    5 FIELDNAME (40) CHAR (8), /* RESERVED FIELDNAMES */
3 ASSOC_LIST CHAR (9), /* LIST OF ASSOCIATE FILE */
    /* ID'S AVAILABLE FOR */
    /* ASSIGNMENT. */
3 CREATEF BIT (1), /* CREATE-UPDATE MODE FLAG. */
3 DATAPLEX CHAR (6), /* FILE BEING DEFINED. */
3 DELETE_FILES CHAR (36), /* LIST OF DESCRIPTOR REGIONS TO BE DELETED FROM DISC. */
3 EXIST_FILES CHAR (36), /* FILE IDS OF ALL FILES EXISTING ON DISC. */
3 FILE_EXISTS BIT (1), /* DESCRIPTOR FILE EXISTS. */
3 INDEX_LIST CHAR (16), /* LIST OF UNASSIGNED INDEX*. */
3 #FN BIN FIXED, /* NUMBER OF ENTRIES IN FIELD ID'S. */
3 LCAD_ERROR BIT (1), /* ERROR IN LOADING DESCRIPTORS*/
3 NEED_FILE BIT (1), /* USER SHOULD FILE TO SAVE. */
3 SUBFILE_LIST CHAR (10), /* LIST OF ALL UNASSIGNED SUB_FILE ID'S. */

THIS MINOR STRUCTURE IS USED TO STORE THAT INFORMATION NECESSARY TO THE EXECUTION OF MODULES THAT CAN HAVE PAGEABLE INFORMATION DISPLAYS.

3 PAGE_INFO,
  5 RTN LABEL, /* WHAT ROUTINE TO PAGE. */
  5 RTN_NAME CHAR (8), /* THE ROUTINE NAME THE PAGING MODULE IS TO CALL. */
  5 PTR PTR, /* ADDRESS OF STRUCTURE BEING DISPLAYED. */
  5 DIR CHAR (1), /* PAGING DIRECTION. */
  5 FILE_ID CHAR (1), /* SUFFIX OF FILE BEING REVIEWED. */
  5 FLD_NAME CHAR (8), /* NAME OF FIELD BEING REVIEWED */
  5 ITEM BOOLEAN FIXED, /* LAST ITEM PUT ON SCREEN */
  5 ITEMS BIN FIXED, /* # OF ITEMS TO DISPLAY. */
  5 LIMIT BIN FIXED, /* ALL ITEMS AFTER LIMIT TO BE DISPLAYED ONE PER LINE. */
  5 # BIN FIXED, /* # OF PAGE BEING DISPLAYED */
  5 LAST BIN FIXED, /* LAST ENTRY USED. */
  5 START (100) BIN FIXED, /* ITEM # USED TO START */
                /* EACH PAGE. */

THIS MINOR STRUCTURE IS USED TO STORE THOSE FIELD NAMES AND IDS OF THE DESCRIPTOR REGIONS IN WHICH THEY APPEAR THAT MUST BE DELETED FROM THE DESCRIPTOR FILE THE NEXT TIME A FILE IS DELETED.
5  A (100),
7  FIELD CHAR (8),/* NAMES OF FIELDS TO BE DELETED. */
7  IDS  CHAR (4);/* IDS OF FILES ON WHICH THE FIELD APPEARS. */
TOPIC D.26 - MAINTENANCE

A. DATA SET NAME:

DEHDR which consists of the structures HDR and HDR_STRING

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The HDR structure is used to contain the information describing a file descriptor. HDR_STRING is a character string used to overlay the HDR structure.
I. PL/I DECLARATION:

THE HDR STRUCTURE IS USED STORE THE INFORMATION DEFINING A FILE DESCRIPTOR.

1 HDR BASED (X.HDR_PTR), /* FILE DESCRIPTOR */
   /* STRUCTURE. */
3 BACKWARD PTR, /* BACKWARD HDR POINTER. */
3 FORWARD PTR, /* FORWARD HDR POINTER. */
3 SUFFIX CHAR (1), /* WHICH FILE THIS HEADER BELONGS TO. */
3 FILETYPE CHAR (1) /* TYPE OF FILE INDICATOR. */
3 DESCRIPT BIN FIXED, /* NUMBER OF FIELD DESCRIPTORS ON THIS FILE. */
3 DESCRIPT CHAR (1), /* DESCRIPTORS OK FLAG. */
3 SPANNED CHAR (1), /* SPANNED RECORDS. */
3 DATA CHAR (1), /* DATA ON FILE SWITCH. */
3 MNTNABLE CHAR (1), /* FILE CAN BE MAINTAINED FLAG. */
3 MNTNING CHAR (1), /* FILE BEING MAINTAINED FLAG. */
3 LOADABLE CHAR (1), /* WHETHER OR NOT TO PLACE DATA ON THIS FILE. */
3 REMAINS CHAR (8), /* UNUSED HDR DESCRIPTOR FIELD. */
3 RECSFEPF BIN FIXED, /* POINTER TO RECORD SECURITY. */
3 RSECTYCD PTR, /* CODES IF ANY. */
3 CHANGED (13) BIT (1), /* ONE FLAG FOR EACH ITEM IN HEADER STRUCTURE. IF ON THEN PUT NEW VALUE IN DESCRIPTOR FILE. */
3 FILLER CHAR (8); /* NEEDED FOR PLI BUG. */

THIS STRUCTURE IS A CHARACTER STRING OVERLAY ON THE HDR STRUCTURE. IT IS USED FOR MAKING COPIES OF THE HDR STRUCTURE.

DCL HDR_STRING CHAR (46), BASED (X.HDR_PTR); /* HDR STRUCTURE OVERLAY. */
TOPOIIC E.1 - TERMINAL SUPPORT

A. DATA SET NAME:
TSPL/I Diagnostics

B. CREATED BY:
TS Preprocessor Function

C. TYPE OF FILE:
(4) Table

D. ORGANIZATION:
Keyed List

E. KEY IDENTIFIER (CONTROL FIELD):
Each diagnostic comment has an identification key having the form: '---ERROR---nn' where nn is a unique identification number.

F. RECORD LENGTH:
Variable

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
TSPL/I Diagnostic comments are generated into mainline source programs by the TS preprocessor function (see Section IV, Topic E.1 of the DWB). They are tabulated here with additional notes for reference. In Paragraph I, single quotes denote that characters from the TS preprocessor function or its argument are filled into the message to make its meaning clearer.

I. TSPL/I DIAGNOSTIC COMMENTS:

TS001  MISSING ARGUMENT ON TSPL/I REFERENCE.

Severe error - a TS preprocessor function reference has no parenthesized argument.

TS002  TSPL/I ARGUMENT DOES NOT BEGIN WITH A '('.

Severe error - a TS preprocessor function reference
does not begin with double left parentheses. Processing of this TS reference was abandoned because the closing right parenthesis would not be able to be found.

**TS003** **MISSING DELIMITING IN TSPL/I STATEMENT.**

Severe error - the right parenthesis at the end of a TS preprocessor function reference has been encountered unexpectedly.

**TS004** **STATEMENT HAS A MISSING ';'.**

Severe error - the right parenthesis at the end of a TS preprocessor function reference has been encountered unexpectedly.

**TS005** **STATEMENT FOUND FOLLOWING FINISH.**

Severe error - the statement has been ignored because it follows the TS {{FINISH;}} reference.

**TS006** **STATEMENT CONTAINS EXCESS '{' (s).**

Severe error - the statement semicolon has been found, but the parentheses are unbalanced. The statement was ignored.

**TS007** **STATEMENT KEYWORD UNKNOWN.**

Severe error - an unknown word was found as the first word of a TSPL/I statement. The statement was ignored.

**TS008** **'text' STATEMENT CONTAINS INVALID SYNTAX.**

The statement type identified by 'text' was found to contain invalid syntax. The statement was ignored.

**TS009** **EXTRANEOUS TEXT IGNORED.**

This message merely means that part of the statement was ignored.

**TS010** **IMPROPER OR MULTIPLE ENABLE STATEMENTS.**

An improper placement of or multiple use of an enable statement has been encountered. The statement was ignored.
The finish statement has been processed and NNNNN errors were previously detected.
TOPIC E.2 - TERMINAL SUPPORT

A. DATA SET NAME:
Terminal Control Block

B. CREATED BY:
TS Preprocessor Function

C. TYPE OF FILE:
(4) Table

D. ORGANIZATION:
Linear Structure of Fields

E. KEY IDENTIFIER (CONTROL FIELD):
TC

The terminal control block is an automatic internal data table.

F. RECORD LENGTH:
236 Bytes (Hexidecimal EC)

This is the length of the whole control block including the dope vectors.

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
The TC control block is used for communication between mainline programs and TSSUP. The declaration is generated by the TS preprocessor function. For TSPL/I statements in the mainline, the TS preprocessor function generates statements that post fields in TC, such as a prompt message key. At execution time, TSSUP refers to fields in TC and posts error code fields in TC which may subsequently be referenced in the mainline.
I. PL/I DECLARATION:

/* TERMINAL CONTROL BLOCK (TC) FOR (TS2) TSPL/I */

DECLARE
  1 TC, /*DEFINE THE TC STRUCTURE */
  2 FUNCTION CHAR(8), /*TS FUNCTION IDENTIFIER */
  2 ENTRY 'ENTRY */
  2 READ 'READ */
  2 WRITE 'WRITE */
  2 FLUSH 'FLUSH */
  2 PUT 'PUT */
  2 PROMPT-C='COMMAND PROMPT*/
  2 PROMPT-D='DATA PROMPT */
  2 PROMPT-M='MESSAGE */
  2 PAGING_ENTRY CHAR(8), /*TS PAGING ENTRY POINT */
  2 TO_NAME OF THE CURRENT */
  2 MODULE'S PAGING ENTRY */
  2 LINE_SIZE FIXED BIN(15), /*TS LINE WIDTH (BYTES) */
  2 INPUT, /*TS SCREEN INPUT FIELDS */
  3 ERROR BIT(1), /*READ ERROR BIT SWITCH */
  3 SET BY TSSUP AFTER READ */
  3 'O'=NO ERROR '1'=ERROR */
  3 EXTRA_BITS BIT(7), /*RESERVED BIT SWITCHES */
  2 OUTPUT, /*TS SCREEN OUTPUT FIELDS */
  3 SIZE FIXED BIN(15), /*OUTPUT AREA SIZE (LINES) */
  3 INDENT FIXED BIN(15), /*INDENTATION COLUMN NUMBER*/
  3 WRITTEN FIXED BIN(15), /*PUT OUTPUT COUNT (BYTES) */
  3 IF AUTO_WRITE IS SET ON */
  3 DIRECTION BIT(1), /*PUT DIRECTION BIT SWITCH */
  3 SET BY TS PREPROCESSOR */
  3 'O'=FORWARD '1'=BACKWARD */
  3 PUT_PARTIAL BIT(1), /*PUT OUTPUT MODE SWITCH */
  3 SET BY USER AT ANYTIME */
  3 'O'=PUT FULL RECORD ONLY */
  3 '1'=PUT PARTIAL RECORD */
  3 AUTO_WRITE BIT(1), /*PUT END OF BUFFER ONLY */
  3 SET BY USER AT ANYTIME */
  3 'O'=RETURN TO USER */
  3 '1'=AUTOMATIC WRITE */
  3 WORD_BREAK BIT(1), /*PUT LINE SPLIT SWITCH */
  3 SET BY USER AT ANYTIME */
  3 'O'=TRUNCATE AT LINE END */
  3 '1'=BREAK AT LAST WORD */
  3 OVERFLOW BIT(1), /*PUT OVERFLOW BIT SWITCH */
/* SET BY TSSUP WHEN PUT */ /* CAUSES BUFFER OVERFLOW */ /* 0' = NO OVERFLOW */ /* 1' = OVERFLOW */

3 CONTINUATION BIT(1), /* PUT CONTINUATION SWITCH */ /* SET BY USER WHEN HE IS */ /* PUTTING CONTINUED DATA */ /* 0' = NO CONTINUATION */ /* 1' = PUT CONTINUED DATA */

3 POSITION BIT(1), /* PUT POSITIONING SWITCH */ /* SET BY TS PREPROCESSOR */ /* 0' = PUT TO NEXT LINE */ /* 1' = PUT TO TOP OF SCREEN */

3 MORE_DATA BIT(1), /* SCREEN OVERFLOW SWITCH */ /* SET BY THE USER WHEN HE */ /* HAS MORE DATA TO OUTPUT */ /* VIA THE PAGING MECHANISM */ /* 0' = NO MORE DATA REMAINS */ /* 1' = MORE DATA AVAILABLE */

2 PROMET, /* TS SCREEN PROMPT FIELDS */

3 SIZE FIXED BIN(15), /* PROMPT AREA SIZE (LINES) */ /* SET BY TSSUP ON ENTRY */

3 MESSAGE_KEY CHAR(8), /* KEY OF CURRENT MESSAGE */ /* SET BY TS PREPROCESSOR */

3 KEYWORD CHAR(8), /* KEYWORD FOR DATA PROMPT */ /* SET BY TS PREPROCESSOR */

3 BYPASS BIT(1), /* PROMPTING BYPASS SWITCH */ /* SET BY USER AT ANYTIME */ /* 0' = PROMPT IF NO DATA */ /* 1' = RETURN NULL VALUE */

3 ERROR BIT(1), /* PROMPTING ERROR SWITCHZZZ */ /* SET BY USER WHEN A DATA */ /* ERROR FORCES REPROMPTING */ /* 0' = PROCESS NORMALLY */ /* 1' = REPrompt FOR DATA */

3 TRUNCATION BIT(1), /* DATA TRUNCATION SWITCH */ /* SET BY TSSUP FOR PROMPT */ /* 0' = NO TRUNCATION */ /* 1' = DATA TRUNCATED */

3 DEFAULT BIT(1), /* DEFAULT VALUE BIT SWITCH */ /* SET BY TSSUP FOR PROMPT */ /* 0' = DATA ENTERED BY USER */ /* 1' = DATA WAS A DEFAULT */

3 UOTED BIT(1), /* UOTED UOTED BIT SWITCH */ /* SET BY TSSUP FOR PROMPT */ /* 0' = NORMAL DATA VALUE */ /* 1' = UOTED STRING */

3 MORE_DATA BIT(1), /* PARENTHORIZED LIST SWITCH */ /* SET BY TSSUP FOR PROMPT */ /* 0' = LAST DATA VALUE */ /* 1' = MORE VALUES FOLLOW */

3 SKIP BIT(1), /* SKIP INPUT PARSING BIT */ /* SET BY THE USER WHEN HE */
3 PAGE BIT(1);

/* WISHES TO BYPASS PARSING */
/* '0' = DO NORMAL PARSING */
/* '1' = SKIP NORMAL PARSING */
/* PAGING CONTROL SWITCH */
/* '0' = IGNORE PAGING ENTRY */
/* '1' = ALTER PAGING ENTRY */
TOPIC E.3 - TERMINAL SUPPORT

A. DATA SET NAME:
TSTEXT - Terminal Control Block Declaration

B. CREATED BY:
Included by TS processor function.

C. TYPE OF FILE:
Table

D. ORGANIZATION:
PL/I Source Statements

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
92 bytes

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
The purpose of the TSTEXT is to define the terminal control block (TC) within every program using it. This enables the programmer and the preprocessor to refer to the fields of the TC block in order to specify the various functions and options needed by the program.

I. PL/I DECLARATION:

DECLARE
TSFLUSH ENTRY(), /* FLUSH ENTRY POINT */
TSREAD ENTRY(CHAR(*) VAR), /* READ ENTRY POINT */
TSWRITE ENTRY(CHAR(*) VAR), /* WRITE ENTRY POINT */
TSPUT ENTRY(CHAR(*) VAR), /* PUT ENTRY POINT */
TSPRMTC, /* COMMAND PROMPT ENTRY */
TSPRMTD, /* DATA PROMPT ENTRY */
TSPRMTM ) ENTRY(CHAR(*) VAR, /* MESSAGE ENTRY POINT */
CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR,
CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR,
CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR,
CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR,
CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR):
DECLARE
1 TC,
2 FUNCTION CHAF(8),

1* DEFINE THE TC STRUCTURE */
/* TS FUNCTION IDENTIFIER */
/* SET BY TS PREPROCESSOR */
/* 'ENTRY *=ENTRY */
/* 'READ *=READ */
/* 'WRITE *=WRITE */
/* 'FLUSH *=FLUSH */
/* 'PUT *=PUT */
/* 'PROMPT-C'=COMMAND PROMPT */
/* 'PROMPT-D'=DATA PROMPT */
/* 'PROMPT-M'=MESSAGE */

2 PAGING_ENTRY CHAR(8),
/* TS PAGING ENTRY POINT */
/* SET BY TS PREPROCESSOR */
/* TO NAME OF THE CURRENT */
/* MODULE'S PAGING ENTRY */

2 LINE_SIZE FIXED BIN(15),
/* TS LINE WIDTH (BYTES) */
/* SET BY TSSUP ON ENTRY */

2 INPUT,
3 ERROR BIT(1),
/* TS SCREEN INPUT FIELDS */
/* READ ERROR BIT SWITCH */
/* SET BY TSSUP AFTER READ */
/* '0'=NO ERROR '1'=ERROR */

3 EXTRA_BITS BIT(7),
/* RESERVED BIT SWITCHES */

2 OUTPUT,
3 SIZE FIXED BIN(15),
/* OUTPUT AREA SIZE (LINES) */
/* SET BY TSSUP ON ENTRY */

3 INDENT FIXED BIN(15),
/* INDENTATION COLUMN NUMBER */
/* SET BY USER AT ANYTIME */

3 WRITTEN FIXED BIN(15),
/* PUT OUTPUT COUNT (BYTES) */
/* SET BY TSSUP ON OVERFLOW */
/* IF AUTO_WRITE IS SET ON */

3 DIRECTION BIT(1),
/* PUT DIRECTION BIT SWITCH */
/* SET BY TS PREPROCESSOR */
/* '0'=FORWARD '1'=BACKWARD */

3 PUT_PARTIAL BIT(1),
/* PUT OUTPUT MODE SWITCH */
/* SET BY USER AT ANYTIME */
/* '0'=PUT FULL RECORD ONLY */
/* '1'=PUT PARTIAL RECORD */

3 AUTO_WRITE BIT(1),
/* PUT END OF BUFFER SWITCH */
/* SET BY USER AT ANYTIME */
/* '0'=RETURN TO USER */
/* '1=AUTOMATIC WRITE */

3 WORD_BREAK BIT(1),
/* PUT LINE SPLIT SWITCH */
/* SET BY USER AT ANYTIME */
/* '0'=TRUNCATE AT LINE END */
/* '1'=BREAK AT LAST WORD */

3 OVERFLOW BIT(1),
/* PUT OVERFLOW BIT SWITCH */
/* SET BY TSSUP WHEN PUT */
/* CAUSES BUFFER OVERFLOW */
/* '0'=NO OVERFLOW */
/* '1'=OVERFLOW */
3 CONTINUATION BIT(1), /* PUT CONTINUATION SWITCH */
/* SET BY USER WHEN HE IS */
/* PUTTING CONTINUED DATA */
/* '0' = NO CONTINUATION */
/* '1' = PUT CONTINUED DATA */
3 POSITION BIT(1), /* PUT POSITIONING SWITCH */
/* SET BY TS PREPROCESSOR */
/* '0' = PUT TO NEXT LINE */
/* '1' = PUT TO TOP OF SCREEN */
3 MORE_DATA BIT(1), /* SCREEN OVERFLOW SWITCH */
/* SET BY THE USER WHEN HE */
/* HAS MORE DATA TO OUTPUT */
/* VIA THE PAGING MECHANISM */
/* '0' = NO MORE DATA REMAINS */
/* '1' = MORE DATA AVAILABLE */
2 PROMPT,
3 SIZE FIXED BIN(15), /* PROMPT AREA SIZE (LINES) */
/* SET BY TSSUP ON ENTRY */
3 MESSAGE_KEY CHAR(8), /* KEY OF CURRENT MESSAGE */
/* SET BY TS PREPROCESSOR */
3 KEYWORD CHAR(8), /* KEYWORD FOR DATA PRCMT */
/* SET BY TS PREPROCESSOR */
3 BYPASS BIT(1), /* PROMPTING BYPASS SWITCH */
/* SET BY USER AT ANYTIME */
/* '0' = PROMPT IF NO DATA */
/* '1' = RETURN NULL VALUE */
3 ERROR BIT(1), /* PROMPTING ERROR SWITCH */
/* SET BY USER WHEN A DATA */
/* ERROR FORCES REPROMPTING */
/* '0' = PROCESS NORMALLY */
/* '1' = REPROMPT FOR DATA */
3 TRUNCATION BIT(1), /* DATA TRUNCATION SWITCH */
/* SET BY TSSUP FOR PROMPT */
/* '0' = NO TRUNCATION */
/* '1' = DATA TRUNCATED */
3 DEFAULT BIT(1), /* DEFAULT VALUE BIT SWITCH */
/* SET BY TSSUP FOR PROMPT */
/* '0' = DATA ENTERED BY USER */
/* '1' = DATA WAS A DEFAULT */
3 QUOTED BIT(1), /* QUOTED QUOTED BIT SWITCH */
/* SET BY TSSUP FOR PROMPT */
/* '0' = NORMAL DATA VALUE */
/* '1' = QUOTED STRING */
3 MORE_DATA BIT(1), /* PARENTHIZED LIST SWITCH */
/* SET BY TSSUP FOR PROMPT */
/* '0' = LAST DATA VALUE */
/* '1' = MORE VALUES FOLLOW */
3 SKIP BIT(1), /* SKIP INPUT PARSING BIT */
/* SET BY THE USER WHEN HE */
/* WISHES TO BYPASS PARSING */
/* '0' = DO NORMAL PARSING */
/* '1' = SKIP NORMAL PARSING */
3 PAGE BIT(1); /* PAGING CONTROL SWITCH */
/* '0'=IGNORE PAGING ENTRY */
/* '1'=ALTER PAGING ENTRY */
TOPIC F.1 - DATA RETRIEVAL

A. DATA SET NAME

RETDATA - Retrieval Data Table

B. CREATED BY:

RDBINIT

C. TYPE OF FILE:

Table

D. ORGANIZATION

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE

1. RETDATA EXTERNAL CONTROLLED.
   This table contains data fields unique to the retrieval sub-system and referenced by various modules of that sub-system.

   2. PRINTDE CHARACTER (8) VARYING.
      This field contains the ddname of the print file for retrieval.

   2. PRINTDS CHARACTER (35) VARYING.
      This field contains the dsname of the print file for retrieval.

   2. SRT98DE CHARACTER (8) VARYING.
      This field contains the ddname of the save file for retrieval.

   2. SRT98DS CHARACTER (35) VARYING.
      This field contains the dsname of the save file for retrieval.
2. **BITS.**

The following bit switches are used to communicate status between the various retrieval modules.

3. **PRTUSED BIT (1).**

Describes whether any data has been written to the retrieval print file.

I. **PL/I DECLARATION**

```pli
/*

NASIS SYSTEM RETRIEVAL DATA TABLE */

DCL
1 RETDATA EXTERNAL CONTROLLED, /*DEFINE RETRIEVAL DATA */
  2 PRINTDD CHAR(8) VAR, /*PRINT FILE DDNAME */
  2 PRINTDS CHAR(35) VAR, /*PRINT FILE DSNAME */
  2 SET98DD CHAR(8) VAR, /*SAVE FILE DDNAME */
  2 SET98DS CHAR(35) VAR, /*SAVE FILE DSNAME */
  2 BITS, /*RETRIEVAL BIT SWITCHES */
  3 PRTUSED BIT(1), /*PRINT FILE USED BIT */
  3 UNUSED BIT(7); /*UNASSIGNED BIT SWITCHES */
```
TOPIC F.2 - DATA RETRIEVAL

A. DATA SET NAME:
   EXPAND Display Format

B. CREATED BY:
   EXPAND (RDEXPND)

C. TYPE OF FILE:
   (3) Terminal Communication

D. ORGANIZATION:
   Character Display Screen

E. KEY IDENTIFIER (CCNTRL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Variable (Enter output area of the screen or pseudo screen)

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:

   This is a series of on-line output displays produced by the EXPAND command giving the user full access to the inverted indexes of a data base assisting him in an on-line search for information.

   The display is adapted to the size of the display screen being used. If the end of the inverted index or the end of the range of E-numbers (000-999) is encountered in either direction, a line such as

   (--START OF INDEX--)
   (--END OF INDEX----)

   is displayed in the appropriate row. The primary term is always regenerated on the appropriate row when multiple paging operations are done in either direction even if the primary term is not found in the inverted index.
**SAMPLE EXPAND DISPLAY**

**SYSTEM: -ENTER:**

**USER:** expand pli, language

**SYSTEM:**

<table>
<thead>
<tr>
<th>LINE</th>
<th>XREFS</th>
<th>LANGUAGE(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>E097</td>
<td>28</td>
<td>ASM</td>
</tr>
<tr>
<td>EC98</td>
<td>6</td>
<td>ENG</td>
</tr>
<tr>
<td>EC99</td>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>-E100</td>
<td>43</td>
<td>PLI</td>
</tr>
<tr>
<td>E101</td>
<td>4</td>
<td>TSS</td>
</tr>
</tbody>
</table>

****

**(--START OF INDEX--)**

****

**(--END OF INDEX--)**
TOPIC F.3 - DATA RETRIEVAL

A. DATA SET NAME:
   SELECT Display Format

B. CREATED BY:
   SELECT (RDBSLCT and RDBSETS)

C. TYPE OF FILE:
   (3) Terminal communication

D. ORGANIZATION:
   Character Display Screen

E. KEY IDENTIFIER (CCMPCL FIELD):
   Not Applicable

F. RECORD LENGTH:
   480 Byte typical - 40 column, 22 line output area apart from the prompting area.

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   This is the output generated by the SELECT command. RDBSLCT calls the DBPSET entry point of RDBSETS to post the users new set and RDBSETS sends this display to the prompt area of the screen.
SELECT COMMAND SCREEN DISPLAY

aa bbbbb cccccc

or

aa bbbbb (FROM: ddddd) cccccc

where:

aa = set number
bbbb = number of references
cc etc. = SELECT expression
ddddd = control field name, if applicable
TOPIC F.4 - DATA RETRIEVAL

A. DATA SET NAME:
   DISPLAY Display Format

B. CREATED BY:
   DISPLAY (RDBDSFI)

C. TYPE OF FILE:
   (3) Terminal Communication

D. ORGANIZATION:
   Character Display Screen

E. KEY IDENTIFIER (CCNTRCL FIELD):
   Not Applicable

F. RECORD LENGTH:
   480 Bytes typical - 40 column, 12 line output area
   apart from the prompting area.

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   This is a series of on-line output displays produced by
   the DISPLAY command giving the user full access to the
   anchor and associated files of a data base assisting
   him in an on-line search for information. Each screen
   image is built in a PACTAB buffer and then transmitted
   in a single output operation to the display screen. A
   special use of the DISPLAY command is to retrieve saved
   screen images and redisplay them. Usually a stored
   screen image is one of the formats produced by the
   various commands, but it may even be a screen image the
   user has keyed in.

   The display is adapted to the size of the display
   screen being used including the degenerate case of a
   typewriter terminal (120 columns by one line).
The first row under the heading rows always has a field name tag, even when it is a continuation of an element value begun on the previous screen.
DISPLAY Command Screen Display

DISPLAY aa,b,cccc (original command parameters)

wwwwwwww: mmmmm CF SET aa, FORMAT b, ITEM nn
xxxxxxx: n
yyyyyyyy: p,
: q
zzzzzzzz: rrrrrrrrrrrrrrrrrrrrrrrrrrrrr
rrrrrrrrrrrrrrrrrrrrrrrrrrr
rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
rrrrrrrrrrrrrrrrr etc.

where:

nn = relative record in set/key aa
mmm etc. = up to 30 characters of key value.
p, q = up to 30 characters of element value.
rrr etc. = 77 character element value.
www etc. = key field name.
xxx etc. = field name having a single short element.
yyy etc. = field name having two short elements.
zzz etc. = field name having a single long element.
COLUMNS

DISPLAY aa, PPP, ccccc (original command parameters)

<table>
<thead>
<tr>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1</td>
<td>h2</td>
<td>h3</td>
<td>h4</td>
<td>h5</td>
</tr>
</tbody>
</table>

F1 F2 F5
F2
F3 F4
F4
F5
F3
F3

where:

xx = page number.
t1 = one or more title lines.
h1 = one or more header lines.
F1,F5 = one element field on the anchor or associate file.
F2 = a multi-element field.
F3,F4 = an elemental field on a subfile.
TOPIC F.5 - DATA RETRIEVAL

A. DATA SET NAME:
   PARSED Table

B. CREATED BY:
   Not Applicable

C. TYPE OF FILE:
   Table

D. ORGANIZATION:
   Linear Structure of elements

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   PARSED is a group of structures used by SELECT to save the information from a parsed expression, when that expression requires a search or contains "S" numbers which cannot be resolved with set numbers until after execution at the actual linear search.

At search time EXSEARCH calls SELECT to begin evaluation of the boolean expressions and to post sets to be searched. SELECT uses the information contained in PARSED to do this and to replace all "S" numbers with their corresponding set numbers. After the search SELECT proceeds with the final evaluation of the expression.

1. PARSED is a based structure consisting of pointers to the other structures containing the various pieces of information that needs to be saved when a boolean expression contains "S" numbers.

2. PARS_TAB_PTR is a pointer to the structure
which is used to describe each element of the expression.

2. **FTAB_INFC_PTR** is a pointer to the structure which holds additional information about each element of the expression.

2. **FTAB_PTRS_PTR** is a pointer to the array of pointers, each pointer corresponding to an element of the expression.

2. **INST_LIST_PTR** is a pointer to the list of instructions generated by SELECT to provide for evaluation of the expression.

2. **WAS_PTR** is a pointer to the work string in which the expression and other necessary character strings are stored.

2. **LNTH** is the length for allocation of all tables listed here except WAS. It is determined from the length of the input expression.

2. **S#** contains the S# in which the PARSED pointer is stored, i.e. the PARSED pointer is stored in SRCHTAB.ENTRYDEF.(PARSED.S#)

1. **PARS_TAB** is the primary table for storage of information as the expression is parsed.

2. **LNTH** is the number of array elements in the table.

2. **EL** is an element of the table. One element in the table is used to describe each syntactic item in the expression.

3. **IDX** is the relative position of the item in the string WAS.

3. **LTH** is the length of the item.

3. **ID** is the identifier of the item which distinguishes between items of the same general type.

3. **TYPE** is the general type of the item, such as a relation operator, character string, etc.

3. **TERM** is used to mark an item as being a term during expression evaluation, or
to mark an item so that it will be ignored by later passes.

3. SKIP causes later program passes to skip over a particular number of items (or elements in PARS_TAB).

1. PTAB_PTRS is an array of pointers. Each pointer corresponds directly to an element in PARS_TAB—the Nth pointer in PTAB_PTRS corresponds to the Nth element in PARS_TAB. When an expression item results in the formation of a set, the pointer to the set is stored in the corresponding element of PTAB_PTRS.

2. INTH is number of array elements.

2. EL is a pointer array element.

1. PTAB_INFO is a table for storage of additional information about an expression item, and again each element corresponds directly to an element in PARS_TAB.

2. INTH is number of array elements.

2. EL is an array element.

3. IDX relative position of item, in string WAS, which is associated with item to which this element corresponds.

3. SFX indicates subfile which applies to item.

3. INDEXD on if item (Fieldname or value) is indexed.

3. NNDXE on if item (Fieldname or value) is not indexed.

3. CTL on when item (Fieldname) is control field name.

1. INST_LIST is a list of "instructions" created and executed by SELECT. The instructions guide the creation of sets, both from index files and through linear search, as well as the boolean combination of all sets, once formed, to yield the final set.

2. LNTH number of instruction elements in this list.
2. EL an instruction.
3. OP is the operation code.
3. IDX1 first parameter.
3. IDX2 second parameter.
3. IDX3 third parameter.

1. WAS is a work string containing the input expression and other necessary character strings.
2. INTH length of work string.
2. S actual string.

1. WAA is a one-character-per-element array which is defined on top of WAS to allow easy access to a single character.
2. LNTH is number of elements.
2. A is a one character element.

PARSER_LIST is base pointer for PARSED structure.
PARS_TAB_PTR is base pointer for PARS_TAB.
PTAP_INFO_PTR is base pointer for PTAP_INFO.
PTAP_PTRS_PTR is base pointer for PTAP_PTRS.
WAS_PTR is base pointer for WAS and WAA.
INST_LIST_PTR is base pointer for INST_LIST.
WAS_SIZE is set to adjust size of WAS at allocation.
TOPIC F.6 - DATA RETRIEVAL

A. DATA SET NAME:
SETS Display Format

B. CREATED BY:
SETS (RDBSETS)

C. TYPE OF FILE:
Terminal Communication

D. ORGANIZATION:
Character Screen Display

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
320 Bytes typical - 40 column, 8 line output area apart from the prompting area.

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
This is the output created by the SETS command. It is a display on the user's screen or typewriter terminal of the sets created during the current strategy session.

This display consists of the set number or S-number, the number of index references in the set, and the expression (including the control field name, if applicable) that formed the set. The expression will wrap around if it exceeds one line.

Paging forwards and backwards is available. The word 'MORE:' will appear at the bottom of the list if there is more data forward.
## I. SAMPLE OUTPUT:

**ENTER: SETS**

<table>
<thead>
<tr>
<th>SET#</th>
<th>XREFS</th>
<th>EXPRESSION</th>
<th>PAGE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>tbbb</td>
<td>ccccccccccccccccccccccccccc</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ccccccccccccccccccccccccccc</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ccccccccccccccccccccccccccc</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ccccccccccccccccccccccccccc</td>
<td></td>
</tr>
<tr>
<td>aa</td>
<td>bbbb</td>
<td>(FROM: ddddd) ccccccc</td>
<td></td>
</tr>
</tbody>
</table>

**-MORE:**

Where:

- **aa** = set number,
- **bbbb** = number of references,
- **cccc etc.** = expression,
- **dddddd** = control field name,
- **-MORE:** = forward continuation indicator.
TOPIC F.7 - DATA RETRIEVAL

A. DATA SET NAME:
EXECUTE Display Format

B. CREATED BY:
EXECUTE (RDBEXSR)

C. TYPE OF FILE:
Terminal Communication

D. ORGANIZATION:
Character Display Screen

E. KEY IDENTIFIER (CCNTROL FIELD):
Not Applicable

F. RECORD LENGTH:
880 Bytes - typical 40 column, 22 line output area apart from the prompting area.

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
This is the output generated by the EXECUTE command which may appear in the output area of the screen. If set numbers are not being displayed in the output area, then the output from EXECUTE will appear in the prompt area of the screen.

When the EXECUTE output is being placed in the output area of the screen, the screen image is transmitted in a single output operation to the display screen.

The display is adapted to the size of the display screen being used, including the degenerate case of a typewriter terminal (one line).

The output screen may contain from one line to the whole output area as output.
EXECUTE Command Screen Display:

aa bbbbb ccccc ... 

where:

aa = set number
bbbb = number of references
cccccc ... = SELECT (search option) expression.
TOPIC F.8 - DATA RETRIEVAL

A. DATA SET NAME:
   PRINT Data Set Format

B. CREATED BY:
   PRINT (RDBPRINT)

C. TYPE OF FILE:
   (5) Non-data base file and
   (2) Formatted print-out

D. ORGANIZATION:
   VSAM

E. KEY IDENTIFIER (CCNTROL FIELD):
   Not Applicable

F. PRINT LENGTH:
   132 bytes maximum printed plus record length and
   carriage control fields (5 bytes).

G. BLOCKING FACTOR:
   Block size = 4096 bytes.

H. PURPOSE:

   This is an output data set produced by the PRINT
   command. It consists of line images written using a
   PL/I file named PRINTER. At the end of a terminal
   session a TSS PRINT task is initiated to print the data
   set off-line on a line printer.

   A leader page shows the user's name and mail stop for
   distribution. Following the output produced for each
   PRINT command is a separator page having 36 dollar
   signs on the first line.
PRINT Command - LEADER PAGE

DISTRIBUTE TO: xxxxxxxxxxx etc.
MAIL STOP: yyyyyyy etc.

where:

xxx etc. = user's name
yyy etc. = mail stop
PRINT Command - TYPICAL FORMAT 1 PAGE

PRINT OF SET xx, Format 1.

aaaaaaaa: ddddddd
aaaaaaaaa: eeeeee
aaaaaaaa: ffffffff

where:

aaaaaaaa = key field name
d thru f = key value (wraps around to column 1 if more than 122 characters).
PRINT Command — TYPICAL FORMAT 2, 3 or 4 PAGE

PRINT OF SET xx, FORMAT y, zzzzzzzz:vvvvvvvvvvv PAGE wwww

aaaaaaa: d
bbbbbbb: e
   : f
cccccc: qqqq ..... qqqq
gggggggggggggg
qqqqq

where:

aaaaaaa = field name having a single short element.
bbbbbbb = field name having two short elements.
cccccc = field name having a long element.
d, f = element value up to 122 characters (no maximum number of elements).
qqq etc. = 379 character element value (no maximum).
zzz etc. = key field name.
vvv etc. = first 74 characters of key value.
PRINT Command - TYPICAL SET 98 (saved screen image) PAGE

DISPLAY OF SCREEN nnnnnnn

where:

nnnnnn = relative saved screen number.
TOPIC F.9 - DATA RETRIEVAL

A. DATA SET NAME:
   EXPTAB - Expand Term Table

B. CREATED BY:
   RDBXPNM

C. TYPE OF FILE:
   (4) Table

D. ORGANIZATION:
   PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   1. EXPTAF EXTERNAL CONTROLLED.
      This table contains a list of alphabetically sequential terms taken from an inverted index file and information relating the terms to reference numbers (E-numbers) used in the SELECT command.

   2. TERMS AREA (2500).
      This area contains a linked list of terms read from the inverted index.

   2. FIRST-FTR POINTER.
      Points to the first term in the linked list.

   2. LAST-FTR POINTER.
      Points to the last term in the linked list.

   2. TOP-FTR POINTER
      Points to the first term displayed on the current page of data.
2. **BOTTOM-PTR POINTER**  
Points to the last term displayed on the current page of data.

2. **FIRST - E# BINARY FIXED.**  
Contains the reference number for the first terms in the list.

2. **LAST E# BINARY FIXED.**  
Contains the reference number for the last term in the list.

2. **TOP-E# BINARY FIXED.**  
Contains the reference number for the first term on the current page.

2. **BOTTOM-E# BINARY FIXED.**  
Contains the reference number for the last term on the current page.

2. **LO-E# BINARY FIXED.**  
Contains the lowest valid reference number.  
(Either 1 or the reference number of the index origin.)

2. **HI-E# BINARY FIXED.**  
Contains the highest valid reference number.  
(Either 999 or the reference number of the index end.)

I. **PL/I DECLARATION**

```
DCL 1 EXPTAB EXT CONTROLED, /*DEFINE THE TERM TABLE */
2 TERMS AREA(2500), /*TERM STORAGE AREA */
2 FIRST_PTR POINTER, /*FIRST LIST ENTRY POINTER */
2 LAST_PTR POINTER, /*LAST LIST ENTRY POINTER */
2 TOP_PTR POINTER, /*FIRST LINE ON PAGE PTR */
2 BOTTOM_PTR POINTER, /*LAST LINE ON PAGE PTR */
2 FIRST_E# BIN FIXED, /*FIRST ENTRY'S E# */
2 LAST_E# BIN FIXED, /*LAST ENTRY'S E# */
2 TOPE# BIN FIXED, /*FIRST E# ON PAGE */
2 BOTTOM_E# BIN FIXED, /*LAST E# ON PAGE */
2 LO_E# BIN FIXED, /*LOWEST VALID E# */
2 HI_E# BIN FIXED; /*HIGHEST VALID E# */
```
A. DATA SET NAME:
FLDTAB - Field Name Table

B. CREATED BY:
DBPFLDT entry of module PDBPAC

C. TYPE OF FILE:
(4) Table

D. ORGANIZATION:
Linear structure containing adjustable arrays.

E. KEY IDENTIFIER (CONTROL FIELD):
FLDTAB is the major structure name. It is the name of an external variable containing the data.

F. RECORD LENGTH:
918 bytes (396 hexadecimal) minimum plus 10 (A hex) bytes per field name.

The minimum length includes the necessary PL/I dope vector (188 bytes = BC hexadecimal) and space for the RECLEN and keyname field names.

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
The field name table, FLDTAB, contains a list of the names of the fields in the current data base organized particularly for use by the DISPLAY and PRINT commands and index indicators, particularly for the EXPAND and SELECT commands, of field names for which there are inverted indexes. The sizes, base addresses and names of sequential format definition tables are tabulated. The base addresses and names of columnar format definition tables are tabulated.
**FIGURE 1. SAMPLE LISTING FORMAT**
I. SCHEMATIC DIAGRAM:
See Figure 1

J. PL/I DECLARATION:

/* FLDTAB: NASIS SYSTEM FIELD NAME TABLE FOR DATABASE-2. */

THIS TABLE IS ALLOCATED (CH FREE AND REALLOCATED) AND
INITIALIZED BY A CALL TO THE ENTRY POINT DBPFLDT OF
MODULE PEBFAC. EACH CALL TO THIS ENTRY POINT CAUSES
THE ENTIRE TABLE TO BE RE-INITIALIZED TO THE VALUES FOR
THE CURRENTLY SPECIFIED DATABASE FILE. THE VALUES WILL
BE ADJUSTED TO REFLECT THE DATA AVAILABILITY BASED UPON
THE SECURITY CODE ENTERED BY THE USER.

DECLARE
1 FLDTAB EXT CONTROLLED, /*NASIS FIELD NAME TABLE */
3 DATAPLEX CHARACTER(8), /*THE DATABASE FILE NAME */
3 FIELD, /*THE NUMBER OF FIELD NAMES */
  5 # FIXED BINARY, /*NAMES IN THE TABLE */
  5 RECLLEN CHARACTER(8), /*THE RECORD LENGTH FIELD*/
  5 KEYNAME CHARACTER(8), /*THE KEY FIELD ENTRY */
  5 NAME(*) CHARACTER(8), /*THE REMAINING FIELD */
      /*NAMES FOR THIS FILE */
  5 INDEXXD(*) CHARACTER(1), /*THE INDEX FILE SUFFIX */
      /*FOR EACH INDEXED FIELD */
      /*FIELD.INDEX(0:FIELD.##)*/
  5 SUBFILE(*) CHARACTER(1), /*THE SUBFILE SUFFIX FOR */
      /*EACH CONTROL FIELD OR */
      /*SUBFILE FIELD */
      /*FIELD.SUBFILE(0:FLD.##)*/
3 SEQ_FORMAT(25), /*SEQUENTIAL FORMAT INDEX*/
  5 # FIXED BINARY, /*THE NUMBER OF FIELD */
  5 BASE POINTER, /*NAMES IN THE FORMAT */
  5 NAME CHARACTER(8), /*THE FORMAT DESCRIPTION */
      /*TABLE ADDRESS */
      /*THE NAME ASSIGNED TO */
      /*THIS FORMAT (OPTIONAL) */
3 COL_FORMAT(25), /*COLUMNAR FORMAT INDEX */
  5 BASE POINTER, /*THE FORMAT DESCRIPTION */
      /*TABLE ADDRESS */
  5 NAME CHARACTER(8); /*THE NAME ASSIGNED TO */
      /*THIS FORMAT (OPTIONAL) */
K. FIELD DETAILS:

DATA_BASE - has the name of the current dataplex.

FIELD - the subscript in FIELD.NAME of the last field name. Thus it is the number of field names excepting RECLEN.

FIELD.RECLEN - the name of the anchor record length field. This may be referenced as FIELD.NAME(0) when convenient.

FIELD.KEYNAME - the name of the current data base's anchor key field. This may be referenced as FIELD.NAME(1) when convenient.

FIELD.NAME - an array containing the names of the fields in the current data base arranged as shown in Paragraph I.

FIELD.INDEXID - An array parallel with FIELD.NAME. A non-blank indicates an indexed field.

FIELD.SUFFIX - An array parallel with FIELD.NAME. In the anchor and associated portion of the array (subscripts C through FLDTABSEQ_FORMAT(3)) a non-blank indicates a subfile control field. In the subfile portion of the array (subscripts above FLDTABSEQ_FORMAT(3)) the character indicates which subfile a field is on.

SEQ_FORMAT - an array serving as a directory of the sequential format definition tables. The first four entries are posted by RDBPAC to overlay FIELD.NAME beginning with FIELD.KEYNAME as shown in Paragraph I. The remaining entries are posted by REBFORM to refer to dynamically allocated sequential format definition tables.

SEQ_FORMAT - the number of field names in a sequential format definition table.

SEQ_FORMAT.BASE - the address of a sequential format definition table or a NULL pointer value if it is undefined.

SEQ_FORMAT.NAME - the name assigned to a sequential format or blanks.

COL_FORMAT - an array serving as a directory of the columnar format definition tables. The entries are posted by RDBFCRM to refer to dynamically
allocated columnar format definition tables.

COL_FORMAT.BASE - the address of a columnar format definition table or a NULL pointer value if it is undefined.

COL_FORMAT.NAME - the name assigned to a columnar format or blanks.
Alphabetic Sequences.

* Subfile Control Fields

** Subfile id Key Fields

Figure 1. Schematic Diagram of FLDTAB
TOPIC F.11 - DATA RETRIEVAL

A. DATA SET NAME:
   FORMATS Display Format

B. CREATED BY:
   Formats - FIBSTRT

C. TYPE OF FILE:
   Terminal Display (Pageable)

D. ORGANIZATION:
   Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:

This terminal display is created to display the names of the formats currently available to him. A title, identifying the display, is generated, followed by the format names. The eight character names are sorted into alphabetic sequence, tagged with an asterisk if the format is in core, separated by a blank and grouped into a SCRNWTH size line before they are written to the display.
TOPIC F.12 - DATA RETRIEVAL

A. DATA SET NAME:
   SETAB Sets Table

B. CREATED BY:
   Not Applicable

C. TYPE OF FILE:
   Table

D. ORGANIZATION:
   PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:

1. SETAB EXTERNAL CONTROLLED.
   This structure contains the sets, i.e., current
   strategy, that the user is creating and associated
   information.

2. CURRENT # BINARY FIXED (15,0).
   This is the value of the last set number that was
   created.

2. SET (0:99).
   There is one set created for each select, search
   and LIMIT COMMAND.

3. POINTER (97) POINTER.
   There is a pointer to a list of keys for
   every set that is created. POINTER (J)
   points to the list for SET (J).

3. SIZE BINARY FIXED (31,0).
   This is the number of keys associated with
the corresponding set number.

3. **TYPE CHARACTER (1).**
   This is the SUBFILE SUFFIX that describes the origin of the keys in the set.

I. **PL/I DECLARATION:**

```pli
/*
 * NASIS SYSTEM SET TABLE
 */

DCL 1 SETAB EXTERN CONTROlLED, /*DEFINE THE SET TABLE */
  2 CURRENT # BIN FIXED(15,0), /*LAST ASSIGNED SET NUMBER */
  2 SET(0:99), /*DEFINE THE SET ENTRIES */
  3 POINTER PTR, /*THE SET LIST POINTER */
  3 SIZE BIN FIXED(15,0), /*THE SET SIZE (# OF KEYS) */
  3 TYPE CHAR(1); /*THE SET TYPE (SUBFILE ID) */
```
TOPIC F.13 - DATA RETRIEVAL

A. DATA SET NAME:
USERTAB User Data Table

B. CREATED BY:
RDBMTT

C. TYPE OF FILE:
Table

D. ORGANIZATION:
PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
Not Applicable

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:

1. USERTAB EXTERNAL CONTROLLED.
This structure contains user oriented and status information useful to all NASIS sub-systems.

   2. NIASIS_ID CHARACTER (8) VARYING.
   This field contains the id specified by the user when initiating his NASIS session.

   2. SECURITY CHARACTER (8) VARYING.
   This field contains the user's most recently specified security code, i.e. his PASSWORD at logon or in respond to a SECURE Command.

   2. OWNER_ID CHARACTER (8) VARYING.
   This field contains the TSS userid of the owner of the file specified by the user.

   2. STRATEGY CHARACTER (16) VARYING.
   This field contains the name of the strategy in the event of a BERUN.
2. **TASK_ID** BINARY FIXED (31,0).
   This field contains the task identification number assigned to the user at logon time.

2. **SEQUENCE** BINARY FIXED (15,).
   This field contains a sequence number used by the system in defining unique ddnames to dynamically specified files.

2. **EITS.**
   The status of the user's session is reflected by the settings of the following bit switches.

3. **MTTFLAG** BIT(1).
   Describes whether the task is running under MTT or not.

3. **DISABLED** BIT(1).
   Defines the status of attention interrupts.

3. **RETEIVER** BIT(1).
   Describes whether the task is running under the retrieval system or not.

3. **RESTART** BIT(1).
   Describes whether the session is a restart.

3. **RERUN** BIT(1).
   Describes whether the session is a rerun.

3. **TESTMODE** BIT(1).
   Describes whether the session is productive or a debugging run.

3. **CONVFLAG** BIT(1).
   Describes whether the task is conversational or not.

I. **PL/I DECLARATION:**

```pli
/*
  NASIS SYSTEM USER DATA TABLE */

DCL
  1 USERTAB EXT CONTROLLED, /*NASIS USER DATA TABLE */
  2 NASIS_ID CHAR(8) VAR, /*USER'S IDENTIFICATION */
  2 SECURITY CHAR(8) VAR, /*USER'S SECURITY CODE */
  2 OWNER_ID CHAR(8) VAR, /*FILE OWNER'S IDENTIFIER */
  2 STRATEGY CHAR(16) VAR, /*STRATEGY NAME FOR RERUN */
  2 TASK_ID BIN FIXED(31,0), /*TASK IDENTIFICATION */
```
2 SEQUENCE BIN FIXED(15,0), /*DDNAME SEQUENCE NUMBER */
2 BITS, /*SYSTEM STATUS FLAGS */
3 MTTFLAG EIT(1), /*'1'=IN MTT MODE */
3 DISABLED BIT(1), /*'1'=ATTN'S DISABLED */
3 RETRIEVE EIT(1), /*'1'=RUNNING RETRIEVAL */
3 RESTART EIT(1), /*'1'=IN RESTART MODE */
3 RERUN EIT(1), /*'1'=IN RERUN MODE */
3 TESTMODE EIT(1), /*'1'=NO STRATEGY SAVING */
3 CONVFLAG BIT=1), /*'1'=CONVERSATIONAL */
3 EXTRABIT EIT(1); /*'1'= (NO ASSIGNED VALUE) */
3. $X$ is actual value.
TOPIC F.14 - DATA RETRIEVAL

A. DATA SET NAME:
   EXPLAIN Display Format

E. CREATED BY:
   EXPLAIN (message, RESPONSE and term options) - RDBEXPL

C. TYPE OF FILE:
   Terminal Display (Pageable)

D. ORGANIZATION:
   Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   This terminal display is created to display to the user the results of his message response or term explanation. The date will be written to the screen as read from the message file with no indentation or data tagging, but with word-break specified.
TOPIC F.15 - DATA RETRIEVAL

A. DATA SET NAME:
   GPFIELDS Display Format

B. CREATED BY:
   GPFIELDS - FBGFDLDS

C. TYPE OF FILE:
   Terminal Display

D. ORGANIZATION:
   Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:

   This terminal display is created to display the names assigned to the various generic levels of the key of the generic file. A title, identifying the display, is generated, followed by the names. The eight character names, separated by two blanks, are grouped into SCRNWTH size lines before they are written to the display.
TOPIC F.16 - DATA RETRIEVAL

A. DATA SET NAME:
SEQ_FORM - Sequential Format Definition Table

E. CREATED BY:
DBPFLDT entry of RDEPAC (formats 1-4) overlay FLDTAB.FIELD.NAME
RDBFORM (formats 5-25) - by the FORMAT command.

C. TYPE OF FILE:
(4) Table

D. ORGANIZATION:
Adjustable linear array of 8-character field names.

E. KEY IDENTIFIER (CONTROL FIELD):
SEQ_FORM is the major structure name.

F. RECORD LENGTH:
8 bytes times the number of field names in FLDTAB.

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
A sequential format definition table, SEQ_FORM, contains a list of the names of the fields in a sequential format for use by the REVIEW and RECORD commands. The key field name is always the first in the list. The number of names in the list and the base address of the list is posted in FLDTAB.SEQ_FORMAT.
I. PL/I DECLARATION:

DECLARE
  1 SEQ_FORM BASED(SEQ_BASE), /* SEQUENTIAL FORMAT SPECS */
  3 FIELD_1_, /* OVERLAID BY FIELD(1) */
  5 #FIELDS FIXED FIN, /* NOT USED */
  5 PAD CHAR(6), /* FILLER TO CHAR(8) */
  3 FIELD(2:I)
    REFER(SEC_FORM.#FIELDS)), /* NOT USED */
  5 NAME CHAR(8); /* NAME LIST */
TOPIC F.17 - DATA RETRIEVAL

A. DATA SET NAME:
   NASISID.STRATEGY.DATASET

B. CREATED BY:
   RTSSTRRT

C. TYPE OF FILE:
   (5) Non-Data Base

D. ORGANIZATION:
   VISAM

E. KEY IDENTIFIER (CONTROL FIELD):
   Strategy name supplied by the user (region name) plus a seven digit integer generated key.

F. RECORD LENGTH:
   256 Bytes
   1. Key length - 23
      a. Region name - 16
      b. Integer key - 7
   2. Data length - 231
   3. Record length field - 4
   4. Keyboard/Cardboard Indicator - 1

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   This data set is used to contain the command strings that the user has entered through the NASIS system. The current strategy is kept in a CURRENT_STRATEGY as each command is entered. After a strategy session, the user may save the current strategy under his specified name; otherwise, it will be deleted. These saved strategies can then, at some later time, be rerun by
use of the RERUN command. The saving of the current strategy command strings also provides for a restart capability if TSS crashes while the user is running the NASIS system.
TOPIC F.18 - DATA RETRIEVAL

A. DATA SET NAME:
SRCHTAB - Linear Search Table of Pseudo-sets

B. CREATED BY:
SELECT (search option) - RDBSLCT

C. TYPE OF FILE:
Table

D. ORGANIZATION:
Linear structure containing arrays.

E. KEY IDENTIFIER (CCNTPOL FIELD):
SRCHTAB is the major structure name; it is the name of
the control section containing the data.

F. RECORD LENGTH:
(hexadecimal) data bytes.

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
The linear search table of pseudo-sets, SRCHTAB, contains the displavable information for every
pseudo-set, whether a PRINT is to be performed on each, the set upon which the pseudo-set is based (as a result
of the search), pointers to search lists and parsing
lists (for those pseudo-sets defined as boolean combinations of other pseudo-sets and/or sets), and the
testing criteria for each.

I. SCHEMATIC DIAGRAM:
See Figure 1

J. VARIABLE DETAILS:
3 CURRENT_S# is the last pseudo-set number
assigned.
MAX_S is the maximum allowable pseudo-set number.

ENTRY is an array of displayable pseudo-set information containing the S# and its related EXPRESSION.

ENTRYDEF is an array with detailed pseudo-set information:

RECORD is a pointer used for the parameters of those pseudo-sets to be RECORDed after a search execution. This points to SPRNTAB structure.

CREATED_BY is two bits identifying the type of SELECT command used to create this pseudo-set where,

- SELECT_IF bit is on if the search option was used, or
- SELECT_BOOL bit is on if the boolean option was used.

REF_SET is a structure used for identifying the searching universe (or set) wherein,

- PTR is pointer to set to search within
- S# is a bit on if the set to be searched is a pseudo-set.

CORRESP_SET# is the value of the set resulting from this pseudo-set.

LIST_PTR is a pointer to the search list structure for this pseudo-set.

PARSED is a pointer to a parsing structure for boolean-created pseudo-sets.

DELETED is a bit on if this pseudo-set has been deleted.

FIELDNAM is the field name to be tested.

OP_CODE is a value of the operator to be used for the test, as follows:

1. greater than
2. less than
3. equal
4. greater than or equal
5. less than or equal
6. not equal
7. between
8. containing

5 VALUES is a pointer to test values; this points to VALUTAB.

3 SRCH_IN_PROGRESS is a bit on if a search is being executed.

3 IFSO is set on if any pseudo-set is to be printed.

3 SLCT_ERROR is a bit on if error occurs in SELECT during execution of search.

3 SLCT_FINISH is a bit on if all SELECT functions are complete during execution of search.

1 S#_XREFS is a bit array used to record the S#'s which reference the S# in question.

3 SPRNTTAB is a table of record parameters of an S#.

3 FORMAT is a table of record format parameters.

5 TYPE

5 FIRST 5. LAST

3 NEXT_SPRNTTAB is a pointer to next SPRNTTAB structure.

1 VALUE_# is set for adjustment of valutab at allocation.

1 VALUTAB is a table of pointers to values.

3 #OF is number of pointers in this table.

3 VALUPTR is an array of pointers to values.

1 VALUE_SIZE is set for size of value at allocation.

1 VALUE is a table containing a value to be used during search. Pointer is in VALUTAB.

3 SIZE is length of value.
### Table 1: Pseudo-Set Information

<table>
<thead>
<tr>
<th>Pseudo-Set Number</th>
<th>Expression</th>
<th>RecordParms. PTR.</th>
<th>Creator Bits</th>
<th>Reference Set</th>
<th>Ref. Set Is</th>
<th>Resultant Set #</th>
<th>Temp. Search List</th>
<th>Parse Info. PTR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'S01 5 IF AGE=21'</td>
<td>10</td>
<td>5</td>
<td>0</td>
<td>8</td>
<td>LOCN.</td>
<td>0</td>
<td>AGE</td>
</tr>
<tr>
<td>2</td>
<td>'S02 5 IF MAN=Y'</td>
<td>10</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td>LOCN.</td>
<td>0</td>
<td>MAN</td>
</tr>
<tr>
<td>3</td>
<td>'S03 S01*S02'</td>
<td>LOCN.</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>LOCN.</td>
</tr>
</tbody>
</table>

### Diagram

Figure 1. SCHEMATIC DIAGRAM
TOPIC F.19 - DATA RETRIEVAL

A. DATA SET NAME:
COL_FORM - Columnar Format Definition Table

B. CREATED BY:
RDBFORM - the FORMAT command

C. TYPE OF FILE:
(4) Table

D. ORGANIZATION:
Structure containing miscellaneous items, a linear array, and an adjustable array of structures.

E. KEY IDENTIFIER (CONTROL FIELD):
COL_FORM is the major structure name.

F. RECORD LENGTH:
1340 bytes plus 40 bytes per field specification, i.e.,
1540 bytes minimum (5 field specifications) to 3980 bytes maximum (66 field specifications).

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
A columnar format definition table, COL_FORM, contains coded specifications for a columnar display. It is
used by the DISPLAY and PRINT commands and may be revised by the FORMAT command. The optional line for
the page number, lines for titles, and lines for headers hold literal text for output. For each field
specified, the field name, column, width, summary requirements, tally, and summation values are carried.
(Average is not carried in COLFORM; it is computed in
DISPLAY or PRINT.)

I. PL/I DECLARATION:
DECLARE
1 COL_FORM BASED(COL_BASE), /*COLUMNAR FORMAT SPECS */
3 LINESIZE FIXED BIN(31), /*SCRNCOL OR 132 */
3 RECORD_COUNT FIXED BIN(31), /*INIT(0) */
3 TOP,
5 (PAGE#, /*1 OR 0 LINES */
@TITLES, /*0 OR MORE LINES */
#HEADERS, /*0 OR MORE LINES */
DEFAULT_HDR) FIXED BIN, /*0 OR RELATIVE HEADER LINE */
5 LINE(10) CHAR(132),
3 COL_GIVEN BIT(1), /*1: FIELD COLUMNS GIVEN */
/*0: FIELD COLUMNS DEFAULT */
3 #FIELDS FIXED BIN,
3 FIELD( I REFER(CCL_FORM.#FIELDS)),
5 NAME CHAR(8),
5 COLUMN FIXED BIN, /*FOR TRUNCATION INDICATOR */
5 WIDTH FIXED BIN, /*USE COLUMN+1...FOR VALUE */
5 ELEMENT_LIMIT FIXED BIN, /*FOR TRUNCATION INDICATOR */
5 ELEMENT_TALLY,
  7 REQUIRED BIT(1), /*INIT('0'B) */
  7 # FIXED BIN(31), /*INIT(0) */
5 ELEMENT_SUM,
  7 REQUIRED BIT(1), /*INIT('0'B) */
  7 ZONED BIT(1), /*1: ZONED VALUE (INIT) */
  /*0: BINARY VALUE */
  7 VALUE FLOAT BIN(53), /*INIT(0) */
5 ELEMENT_AVERAGE_REQUIRED
  BIT(1); /*INIT('0'B) */
A. DATA SET NAME:
FIELDS Display Format

B. CREATED BY:
FIELDS - RDBPLDS

C. TYPE OF FILE:
Terminal Display (Pageable)

D. ORGANIZATION:
Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
Not Applicable

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
This terminal display is created to display the names of the fields available to him from the current file. A title, identifying the display, is generated, followed by the field names. The eight character names, flagged by an asterisk, if indexed, and separated by a blank, are grouped into SCRNWTTH size lines before they are written to the display. As each subfile is encountered, a heading, identifying it and its control field, is generated.
TOPIC F.21 - RETRIEVAL

A. DATA SET NAME:
LIMIT

B. CREATED BY:
The module whose name is formed by concatenating "RL" to the data base name i.e. "RLERTS". The writeup on creating this module is in the DBA User's Guide.

C. TYPE OF FILE:
table.

D. ORGANIZATION:
PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
Not Applicable

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
The LIMIT structure contains the anchor key subfield names, their starting position within the key, and their length. This table is used by the LIMIT command when limiting a set.

I. PL/I DECLARATION:

DECLARE
1 LIMIT
EXTERNAL CONTROLLED, /* DEFINE THE TAB*/
3 KEY SIZE BIN FIXED, /* LENGTH OF FORMATTED KEY */
3 #_ENTRIES BIN FIXED, /* NUMBER OF SUBFIELDS */
/* DEFINED ON THE ANCHOR KEY */
/* ONE ENTRY FOR EACH SUBFIELD */
3 FIELD (16),
5 NAME CHAR (8) VAR, /* NAME OF SUBFIELD */
5 START BIN FIXED, /* START OF SUBFIELD */
/* WHERE 1 = FIRST CHARACTER */
/* POSITION OF THE KEY */
5 LENGTH BIN FIXED, /* LENGTH OF SUBFIELD */
5 TEST BIT (1), /* WHETHER OR NOT TO APPLY */
/* THIS TEST WHEN LIMIT IS CALLED. */

5 VALUE (2) CHAR (50) VAR;
/* THE VALUES TO COMPARE A */
/* KEY SUBFIELD AGAINST. */
TOPIC F.22 - RETRIEVAL

A. DATA SET NAME:
   LIMIT Command Display Format

B. CREATED BY:
   RDBLMT

C. TYPE OF FILE:
   Terminal Display

D. ORGANIZATION:
   Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   This display allows the user to view the new set created by LIMIT.

I. SAMPLE DISPLAY:

<table>
<thead>
<tr>
<th>SET#</th>
<th>XRETS</th>
<th>EXPRESSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX</td>
<td>XXXX</td>
<td>XXXXXXXXXX</td>
</tr>
</tbody>
</table>
TOPIC G.1 - USAGE STATISTICS

A. DATA SET NAME:
   STATIC Data Set Descriptors

B. CREATED BY:
   Command System and Maintenance System

C. TYPE OF FILE:
   Dataplex

D. ORGANIZATION:
   VISAM

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   4000/V

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   Maintain system statistics:
   1. Retention of Statistics

   In order to maintain the usage statistics, a file is required; and, in order to enhance the usage of these statistics (to interface smoothly with the NASIS system), a data base is required. With the statistics on a data base, the full power of the NASIS system can be used to manipulate them. If any special retrieval or report programs are required, then, currently, DBPL/I and TSPL/I are available; and the Report Generator and Linear Search features are also available.

   Each TSS-ID joined to the NASIS system will have its own statistics data base, and it can be shared with other TSS-ID's just as any other data base.
The name of the statistic data base will follow a uniform format for each TSS-ID joined to NASIS. That is, the name of the statistic data base will be STATIC and the fully qualified data base name will follow the standard naming conventions, eg.

tss-id.STATIC.STATIC#
tss-id.STATIC.STATIC

for the descriptors and anchor file, respectively.

This design also facilitates the integrity and security of the statistic file in that only the owner or those permitted by the owner can access the file.

2. Accumulation of Statistics

The STATIC data base will be composed of two different record types. The data required, and how it will be kept, is as follows:

a. KEY

SEPARATE - A single character which will distinguish the record type. A value of zero will indicate a maintenance record. A value of one will indicate a retrieval record.

IDENTIFIER - For maintenance records, it will be the data base name padded with dollar signs. For retrieval records, it will be the NASIS-ID padded with asterisks (to eight characters). Appended to the NASIS-ID will be the data base owner's TSS-ID and the name of the inverted index file.

b. The maintenance fields are as follows:

TOTALTRN - the number of transactions processed.

ANCOUNT - the number of records on the anchor file.

TOTALRUN - the number of maintenance runs.

MAINDATE - the date of each maintenance run:
element 1 will be the dates of the data base creation, elements 2 through 13 will be the dates of individual maintenance runs. After the dates have been filled, the second one will be dropped and the newest date added on the end. This field is variable-length with fixed-length elements. There is a maximum of thirteen elements.

TRANCNEW
TRANCDEL
TRANCUPD
TRSUBNEW
TRSUBDEL
TRSUBUPD
TRINVNEW
TRINVDDEL
TRINVUPD

- where TR indicates transaction; ANC, the anchor file; SUB, the subrecord files; INV, inverted files; NEW, new records; DEL, deletions; and UPD, updates.

These are the transaction count fields required for maintenance statistics. These fields will be used in conjunction with the data field. They will also have a maximum of thirteen elements. The elements will correspond directly with the date field and will represent the number of that given type of transaction encountered during the maintenance run. When all of the elements are present, the next count inserted will cause the second count to be added to the first element and the second element dropped. The newest element will go on the end.

c. The retrieval fields are:

CONNTIME - the connect time
CPU TIME - the CPU time
TOTALES - number of sessions
STRATLEN - the strategy length
STRATSTR - the number of strategies stored
STARDTE - the date of the first terminal session
LASTDATE - the date of the last terminal session
STRATNME - the names of the stored strategies - maximum of four.

NOTE: The eight fields above are to be accumulated for each NASIS-ID. There may be many records for each NASIS-ID; therefore, these statistics will be kept in a special record. The OWNER-ID and the inverted file name in this special record will be equal to blanks.

#EXPANDS - number of EXPANDS per session
#SELECTS - number of SELECTs per session
#SEARCHES - number of SEARCHes per session
#CORRECTS - number of CORRECTs per session
SESSIDATE - the date of each session

These field all contain a maximum of thirteen elements. The first element represents an accumulator and contains the total for all occurrences up to the SESSIDATE, which is the date of the last ejected session of the list (the earliest session). Regardless of the actual number of sessions within one calendar day, the statistics will be accumulated as if there were only one session.

All of the maintenance statistics will be automatically updated with the Load/Create program and the Maintenance Mainline program. If the data base owner wishes to modify certain data pertaining to the maintenance statistics, he has the ability to use the CORRECT command to update the STATIC data base interactively.
All of the retrieval statistics will be automatically updated with the FINISH module of the command system. If required, at maintenance time, a 'snapshot' of the statistics will be printed. If the data base owner (system manager) wishes to modify certain data pertaining to the retrieval statistics, he has the ability to use the CORRECT command to interactively update the STATIC data base.
APPENDIX A.

The STATIC data base is composed of the following fields:

A. KEY

1. Alphanumeric.
2. Fixed field.
3. Length of 24 bytes.
   a. First byte is maintenance or retrieval record indicator.
      1. 0 = maintenance record.
      a. data base name left justified.
      b. remainder padded with '*'s.
      2. 1 = retrieval record.
         a. NASIS-ID//OWNER-ID//DATA BASE file-name.
            1. The NASIS-ID is eight characters long and padded with '*'s.
            2. The OWNER-ID is really a TSS-ID, eight characters long and padded with '*'s.
            3. The data base file-name is seven characters long and conforms to the data base dataset naming conventions.

B. TOTALTRN (Maintenance)

1. Alphanumeric
2. Fixed field
3. Length of 6 bytes
4. Contains the total number of transactions.

C. ANCOUNT (Maintenance)

1. Alphanumeric
2. Fixed field
3. Length of 6 bytes
4. Contains number of records on the anchor file.

D. TOTALRUN (Maintenance)

1. Alphanumeric
2. Fixed field
3. Length of 3 bytes
4. Contains the number of maintenance runs.

E. MAINDATE (Maintenance)

1. Alphanumeric
2. Fixed element.
   a. Total of 13 elements, each 6 bytes long.
   b. In the form MM/DD/YY to indicate the month, day, and year of each maintenance run.
Element 1 will contain the data base creation date while elements 2-13 will be the dates of the individual maintenance runs. After the dates have been filled, the second one will be dropped and the newest date added to the end.

3. Total length of 78 bytes.

F. TRANCNEW (Maintenance)
   1. Alphanumeric.
   2. Fixed elements.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

G. TRANCDEL (Maintenance)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

H. TRANCUPD (Maintenance)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

I. TRSUBNEW (Maintenance)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

J. TRSUBDEL (Maintenance)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

K. TRSUBUPD (Maintenance)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

L. TRINVNEW (Maintenance)
   1. Alphanumeric.
   2. Fixed element.
a. Each 6 bytes long.

b. Maximum of 13 elements.

3. Total length of 78 bytes.

M. TRINVDEL (Maintenance)
1. Alphanumeric.
2. Fixed element.
   a. Each 6 bytes long.
   b. Maximum of 13 elements.
3. Total length of 78 bytes.

N. TRINVUPD (Maintenance)
1. Alphanumeric.
2. Fixed element.
   a. Each 6 bytes long.
   b. Maximum of 13 elements.
3. Total length of 78 bytes.

NOTE: Items F through N are transaction count fields for the maintenance statistics and correspond directly to MAINDATE.

TR indicates TRANSACTION
ANC indicates ANCHOR FILE
INV indicates INVERTED FILE
NEW indicates NEW RECORDS
DEL indicates DELETIONS
UPD indicates UPDATES

O. CONNTIME (Retrieval)
1. Alphanumeric.
2. Fixed field.
3. Length of 10 bytes.
4. Contains connect time.

P. CPUTIME (Retrieval)
1. Alphanumeric.
2. Fixed field.
3. Length of 10 bytes.
4. Contains total CPU time.

Q. TOTALSES (Retrieval)
1. Alphanumeric.
2. Fixed field.
3. Length of 10 bytes.
4. Contains total number of sessions.

R. STRATLEN (Retrieval)
1. Alphanumeric.
2. Fixed field.
3. Length of 6 bytes.
4. Contains strategy length.
S. STRATSTR (Retrieval)
   1. Alphanumeric.
   2. Fixed field.
   3. Length of 1 byte.
   4. Contains number of strategies stored.

T. STRATNAME (Retrieval)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 35 bytes long.
      b. Maximum of 4 elements.
   3. Total length of 140 bytes.

U. STARTDATE (Retrieval)
   1. Alphanumeric.
   2. Fixed field.
   3. Length of 6 bytes.
   4. Contains date of first terminal session.

V. LASTDATE (Retrieval)
   1. Alphanumeric.
   2. Fixed field.
   3. Length of 6 bytes.

NOTE: The right fields above are accumulated for each NASIS-ID. The owner-ID and the file-name have no meaning.

Therefore, the key of the record where these statistics are meaningful will be composed of an owner-ID and a file-name which are blank.

W. #EXPANDS (Retrieval)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

X. #SELECTS (Retrieval)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

Y. #SEARCHS (Retrieval)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.
Z. #CORECTS (Retrieval)
   1. Alphanumerics.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 SESSDATE (Retrieval)

AA. SESSDATE (Retrieval)
   1. Alphanumerics.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

NOTE: In the last 5 fields there is a one for one correspondence in the elements.

   first SESSDATE - the date of the newest session in the accumulated counts.

   first (others) - the accumulated counts on all indicated.

Regardless of the actual number of sessions within one given calendar day, the statistics will be accumulated as if there were only one session.

When (during UPDATE) a record is encountered with the variable fields having all 13 elements filled, the 'snapshot' of the given record will be taken. The last 12 elements will then be cleared, by summing them and adding them to the first element, the first element SESSDATE will be made equal to the last element SESSDATE.
TOPIC G.2 - USAGE STATISTICS

A. DATA SET NAME:
   Maintenance Statistics Report Format

B. CREATED BY:
   STATIC Report (RDEPRNTM)

C. TYPE OF FILE:
   VS (print)

D. ORGANIZATION:
   Sequential

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   133 Bytes

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   To display (via listing) the status of the maintenance statistics.
<table>
<thead>
<tr>
<th>DATAPLEX NAME</th>
<th>TOTAL TRNS</th>
<th>ANCHOR NUMBER RECORDS</th>
<th>TRANS RUNS</th>
<th>MAINTENANCE RUN DATES</th>
<th>FILEPLEX ADDS</th>
<th>DELETES</th>
<th>UPDATES</th>
<th>SUBPLEX ADDS</th>
<th>DELETES</th>
<th>UPDATES</th>
<th>XPLEX ADDS</th>
<th>DELETES</th>
<th>UPDATES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASRD1$</td>
<td>3,132</td>
<td>1</td>
<td>12/19/72</td>
<td>3,132</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FILEPLEX ADDS DELETES UPDATES**
- **TOTAL**: 3,132
- **AVERAGE**: 3,132

FOR ALL RUNS PER RUN
TOPIC G.3 - USAGE STATISTICS

A. DATA SET NAME:
   Retrieval Statistics Report Format

B. CREATED BY:
   Report Print (RDBFPNTR)

C. TYPE OF FILE:
   VS (print)

D. ORGANIZATION:
   Sequential

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   133 Bytes

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   To display (via listing) the status of the retrieval statistics.
### RETRIEVAL STATISTICS

<table>
<thead>
<tr>
<th>NASISID</th>
<th>CONN-TIME</th>
<th>CPU-TIME</th>
<th># STRAT</th>
<th>STORED</th>
<th>OWNER</th>
<th>FILE</th>
<th>FIELD</th>
<th>ACTUAL</th>
<th>TOTAL</th>
<th>NUMBER OF</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE01</td>
<td>0:53:30</td>
<td>0:00:48:790</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **SAOWNER ASRD1$A**: AUTHOR
  - EXP: 3
  - SEL: 0
  - SRCH: 0
  - CORR: 0
- **SAOWNER ASRD1$B**: KEYWORDS
  - EXP: 13
  - SEL: 0
  - SRCH: 0
  - CORR: 0
- **SAOWNER DB2TDBA**: EMPAGE
  - EXP: 1
  - SEL: 0
  - SRCH: 0
  - CORR: 0
- **SAOWNER DB2TDBB**: TOTALCAR
  - EXP: 1
  - SEL: 0
  - SRCH: 0
  - CORR: 0
- **SAOWNER DB2TDBC**: KIDAGE
  - EXP: 1
  - SEL: 0
  - SRCH: 0
  - CORR: 0
- **SAOWNER DB2TDBD**: PET
  - EXP: 1
  - SEL: 0
  - SRCH: 0
  - CORR: 0
- **SAOWNER DB2TDBE**: SVCDATE
  - EXP: 1
  - SEL: 0
  - SRCH: 0
  - CORR: 0
TOPIC G.4 - USAGE STATISTICS

A. DATA SET NAME:
   Snapshot Statistics Report Format

B. CREATED BY:
   Snapshot Print (RDECCHKPT)

C. TYPE OF FILE:
   VS (print)

D. ORGANIZATION:
   Sequential

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   133 Bytes

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   To display (via listing) those records which have undergone the reinitialization process.
SNAPSHOT (CHECKPOINT) OF RETRIEVAL STATISTICS RECORDS BEFORE REINITIALIZATION

12/18/72

<table>
<thead>
<tr>
<th>LISR ID</th>
<th>CONN-TIME</th>
<th>CPU-TIME</th>
<th># STRAT</th>
<th>OWNER-ID</th>
<th>FIELD</th>
<th>FILE</th>
<th>SESSION</th>
<th>#</th>
<th>#</th>
<th>#</th>
<th>#</th>
<th>#</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEOL</td>
<td>19:40</td>
<td>0:00:12:399</td>
<td>2</td>
<td>SAOWNER</td>
<td>KEYWORDS</td>
<td>ASRD1$B</td>
<td>721215</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>721215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>721215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>721215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>721215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>721215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>721215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>721215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>721215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>721215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>721215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>721215</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TOPIC H.1 - IMMEDIATE COMMANDS

A. DATA SET NAME:
   NASIS Message File

B. CREATED BY:
   Not Applicable

C. TYPE OF FILE:
   VISAM

D. ORGANIZATION:
   Sequential

E. KEY IDENTIFIER (CONTROL FIELD):
   The fifteen byte key is composed of the eight byte
   message key concatenated to the seven byte line
   number.

F. RECORD LENGTH:
   V(132)

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   This data set contains the NASIS system messages used
   by the various modules for prompting and diagnostic
   messages.
TOPIC H.2 - IMMEDIATE COMMANDS

A. DATA SET NAME:
   Strategy Data Set

B. CREATED BY:
   RTSSTRT

C. TYPE OF FILE:
   VISAM

D. ORGANIZATION:
   Regional Sequential

E. KEY IDENTIFIER (CONTROL FIELD):
   Strategy Name (16 characters)

F. RECORD LENGTH:
   328

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   To contain the stored strategies and formats created by and used by the various NASIS commands.
TOPIC H.3 - IMMEDIATE COMMANDS

A. DATA SET NAME:
   Strategy Display Format

B. CREATED BY:
   RDBSTRT

C. TYPE OF FILE:
   Screen Display

D. ORGANIZATION:
   Header = STRATEGY name (centered)
   Data Lines = full width, word split
   Overflow Lines = indented three characters
   Page Overflow = full record

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   Not Applicable

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   To display the contents of the data lines comprising a stored strategy.
TOPIC H.4 - IMMEDIATE COMMANDS

A. DATA SET NAME:
Strategy Names Display Format

B. CREATED BY:
RDBSTRT

C. TYPE OF FILE:
Screen Display

D. ORGANIZATION:
Data Lines = complete 16 character strategy names separated by two blanks (as many as will fit on a line).

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
Not Applicable

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
To display the names of the strategies present in the strategy data set,
TOPIC H.5 - IMMEDIATE COMMANDS

A. DATA SET NAME:
   User Profile Table

B. CREATED BY:
   TS2-Supervisor

C. TYPE OF FILE:
   Segmented Array

D. ORGANIZATION:
   Segment - 1 (Synonyms) - sequential
   Segment - 2 (Default Keywords) - sequential
   Segment - 3 (Default-Data) - random

E. KEY IDENTIFIER (CONTROL FIELD):
   Not Applicable

F. RECORD LENGTH:
   V(32,768)

G. BLOCKING FACTOR:
   Not Applicable

H. PURPOSE:
   To contain the user defined synonyms and defaults.
TOPIC H.6 - IMMEDIATE COMMANDS

A. DATA SET NAME:
NASIS User Profile Dataset

B. CREATED BY:
TS2-Supervisor

C. TYPE OF FILE:
VSAM

D. ORGANIZATION:
Sequential

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
V(32,768)

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:
To contain the lists of user defined synonyms and defaults for a particular NASISID.
TOPIC H.7 - DATA RETRIEVAL

A. DATA SET NAME:
VERETAE Table.

B. CREATED BY:
The NASIS modules which prompt for commands.

C. TYPE OF FILE:
Table

D. ORGANIZATION:
PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):
Not Applicable

F. RECORD LENGTH:
Not Applicable

G. BLOCKING FACTOR:
Not Applicable

H. PURPOSE:

1. **VERETAE EXTERNAL CONTROLLED.**
   This table contains the information necessary to associate a set of valid verbs (commands) and their respective entry points.

2. **# ENTRIES BINARY FIXED.**
   This field contains the count of the number of valid entries in the list below.

2. **SIZE BINARY FIXED.**
   This field contains the count of the number of entries that can be contained in the list below.

2. **SYMBOLIC_ID CHARACTER (8).**
   This field contains the default symbol that can be used to define user written extensions to this list of verbs.
2. **COMMAND (VERB_COUNT).**
   This list is used to describe the commands recognized by the defining module.

3. **NAME CHARACTER (8).**
   This field contains the command name.

3. **ROUTINE CHARACTER (8).**
   This field contains the name of the entry point to be called when this command is entered.

1. **VERB_COUNT BINARY FIXED.**
   This field must be set to the maximum number of entries allowable in the verb list, before the table is allocated.

I. **PL/I DECLARATION:**

```pli
/*
   GENERALIZED NASIS SYSTEM VERB TABLE */

DCL 1 VERBTAB EXT CONTROLED, /*DEFINE THE VERB TABLE */
   2 #_ENTRIES BIN FIXED, /*DEFINE THE CURRENT SIZE */
   2 SIZE BIN FIXED, /*DEFINE THE MAXIMUM SIZE */
   2 SYMBOLIC_ID CHAR(8), /*DEFINE THE DEFAULT TERM */
   2 COMMAND (VERB_COUNT), /*DEFINE THE VERB ENTRIES */
      3 NAME CHAR(8), /*DEFINE THE VERB NAME */
      3 ROUTINE CHAR(8); /*DEFINE THE ROUTINE NAME */

DCL VERB_COUNT BIN FIXED: /*DEFINE THE TABLE SIZE */
```