MODIFICATION OF SAYANT TO RUN ON THE PDP RSX-11M OPERATING SYSTEM(U) ROYAL AIRCRAFT ESTABLISHMENT FARNBOROUGH (ENGLAND) R J BLUFF ET AL. JAN 82
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A
MODIFICATION OF SAVANT TO RUN ON THE PDP RSX-11M OPERATING SYSTEM

by

R. J. Bluff
P. T. May

SUMMARY

SAVANT (System Architecture Verification and Analysis Technique) is a computer program developed specifically to investigate the feasibility of an avionic system design, in terms of problems such as completeness and consistency.

The program was originally developed on a Prime 300 computer system. This Memorandum describes alterations and modifications of SAVANT, to enable it to be used on a PDP 11 RSX-11M operating system.
# LIST OF CONTENTS

<table>
<thead>
<tr>
<th></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
</tr>
<tr>
<td>2</td>
<td>MAIN MODIFICATION TO SAVANT PROGRAM</td>
</tr>
<tr>
<td>3</td>
<td>INSTALLATION OF SAVANT PROGRAM</td>
</tr>
<tr>
<td>4</td>
<td>CONCLUSIONS</td>
</tr>
<tr>
<td></td>
<td>Acknowledgments</td>
</tr>
<tr>
<td></td>
<td>References</td>
</tr>
<tr>
<td></td>
<td>Appendices A, B, C, D, E1, E2, E3, F1, F2, F3, F4, F5, F6</td>
</tr>
<tr>
<td></td>
<td>Report documentation page</td>
</tr>
</tbody>
</table>
INTRODUCTION

SAVANT¹ (System Architecture Verification and Analysis Technique) is a computer program which was developed specifically to investigate problems of completeness, consistency and feasibility of avionic system designs.

SAVANT was originally developed on a Prime 300 computer system and was programmed in Coral 66 using the System Designers standard SDL portable compiler. It was decided to modify the original program to run on a PDP 11/34 computer, with an RSX-11M operating system. This Memorandum describes the major program modifications to convert the original SAVANT program to Dec Coral 66 (version 3.2) and the installation of it to run under an RSX-11M operating system, see Ref 2 and 3. This Memorandum does not attempt to describe the development and application of SAVANT as Ref 1 gives an excellent detailed description.

The exercise of converting a program written in a high level language, eg Coral 66, from one machine to another machine has shown that it can be a tedious and time consuming process and some of the problems encountered are described below. A full listing of the program and installation files are provided to give a complete software document which will assist other users to install SAVANT, on their own machines with relative ease.

1 MAIN MODIFICATION TO SAVANT PROGRAM

The original program was developed on a Prime 300 computer using the SDL portable compiler. One of the major design aims was to make the program as transportable as possible. This aim was achieved by minimising the implementation dependent features in the main body of the code. Also, all the library calls and channel numbers for input/output were embedded in macro definitions and only the macros were called within the program. The changes to the original program fall broadly into two categories which are discussed in more detail below.

The first category includes implementation dependent differences between the two compilers. The main difference occurs in the libraries, and Appendix E.1 contains a listing of the Dec Library specification - CORIOL.CRL - required by SAVANT. The file MACDEF.COR (Appendix E.1) contains all the new macro definitions concerned with the input/output etc. All the other global macro definitions are in the file SAVGBL.COR (Appendix E.1) together with the global declarations. Wherever possible, byte arrays have replaced integer arrays in an attempt to minimise program space. The use of the character "!" as an escape character, before and after a string of layout characters, in a text string, such as:

(a) '!L!' - newline,
(b) '!S!' - space, etc

is not a feature of the Dec Coral. It was decided to edit the modification manually and this task turned out to be very tedious and with hindsight the changes should have been completed automatically with the Teco editor.
The second category of modification to the SAVANT program are as a result of the RSX-11M operating system on the PDP 11/34. The original program was too large to run as a single task, because a task is limited to a virtual address space of 32K words on the PDP 11/34. Hence, it was necessary to reduce the virtual address requirements by creating an overlaid task. Briefly, overlaying a task means it must be divided into segments; a single root segment which is always resident in memory, and a number of overlay segments which can be loaded into memory as they are required. It is not intended to discuss in detail the technique for overlaying tasks as Ref 4 provides all the relevant details.

It should be stressed that the task of creating an overlay structure was considerably reduced by the fact, that the original program was designed and written in a top-down and well structured manner. In fact, the structure of the program very nearly maps directly on to the main overlay structure of the program.

The necessity to produce an overlaid task also provided an opportunity to incorporate all program modules into a library called SAVLIB. The library feature results in a considerable saving of time when program modification and maintenance are necessary. It allows single modules to be compiled and inserted into the library without the need to recompile the complete program. This is useful because, if, for example a global parameter is to be changed then it will usually be safer to recompile the whole program, which could take several hours. To assist library building and overlay construction every SAVANT procedure was copied into an individual file. These files were given an identical name to the corresponding procedure name. The procedure names were also truncated into a six letter combination, to ensure uniqueness of module and module entry names at task build time. Appendix A contains cross-reference tables between the old SAVANT procedure and the new file/procedure names. The first table lists the root six letter file/procedure name in column 1; column 3 is the corresponding SAVANT command name where it is appropriate; column 4 is the old SAVANT procedure name; and column 5 is the program level, where level 0 is the top level, eg SAV2MN. The other tables are similar; the first column contains new file/procedure names and old procedure names, in alphabetical order, respectively.

All the procedures are in separate files which have the same names as the procedure and these files are then placed in either an OVL or LIB file together with the four global files (viz CORIOL.CRL, PROCS.COR, SAVGBL.COR, and MACDEF.COR) using the Coral 'INCLUDE' facility. These files are the basic program modules, which are used to construct SAVLIB and they are compiled as individual Coral segments.

Appendix C contains examples of LIB and OVL files respectively. The first file is LIBOO.COR and includes the procedure file ACTPRC.COR and is headed 'Coral' MODOO. Appendix D contains a cross reference of LIB/MOD and OVL/NUM with a segment name which is the same as the procedure filename. The OVL files contain the procedure which correspond to the SAVANT commands and the LIB files contain the remainder of the procedures.

Appendix E.1 contains the listings of the four global files. Appendix B illustrates the excellent structure of the original SAVANT program, and is a cross reference
table of procedure calls versus program levels. The level 0 and 1 are excluded for the
sake of clarity and only consist of SAV2MN at level 0 and SETCMD and OBEYCM at level 1.
Consider the command CDR which is called from the main program SAV2MN at level 0 via
OBEYCM at level 1. This in turn calls the procedure CHDART at level 2. CHDART calls
CHDWRN and CHDAV at level 3. CHDAV calls GETNV at level 4 and LOCNAM at level 5, while
LOCNAM calls FINNAM at level 6.

The file handling differences between the two operating systems necessitated
extensive modification to the old 'FIND FILE' procedure and the substitution of two new
procedures to replace the old 'READ VALID FILENAME'. Also two other procedures CMPSTR
and GETNUM were required and the listing of all these procedures is given in Appendix E.2
together with the remainder of the procedures in Appendix E.3.

The original procedure 'OBEY COMMAND' caused the operating system to abort the
Dec Coral compiler task and was modified. The Dec compiler is not capable of handling
the long "'IF' condition 'THEN' consequence 'ELSE' ...." conditional statement.

This long statement, was reconstructed into three shorter conditional statements
by the introduction of compound statements, see listing of procedure OBEYCM.COR in
Appendix E.2.

The use of byte arrays introduced several minor errors at run time and these were
overcome by ensuring that byte arrays started on word boundaries.

3 INSTALLATION OF SAVANT PROGRAM

The installation of SAVANT is straight forward and is accomplished by running the
indirect command file SAVMN.CMD listed in Appendix F.6. This command file expects to
find all files in SY: and in the current default UIC. The main command file calls compi-
lation indirect command files OVLCMP.CMD (Appendix F.3) and LIBCMP.CMD (Appendix F.2);
it then builds the SAVANT library file SAVLIB.OLB with LIBGEN.CMD (Appendix F.4) and
finally runs the task builder using indirect command file SAVBLD.CMD (Appendix F.5) which
requires the overlay descriptor file SAVANT.ODL (Appendix F.1).

4 CONCLUSIONS

It was assumed that it would be a fairly straight forward exercise to transfer a
well structured program written in Coral from the Prime 300 to the PDP 11/34 as both
machines had Coral compilers. In practice it turned out to be a time consuming exercise.
This Memorandum describes the major modification necessary as a result of the differences
between the compilers and operating systems, on the two machines. The technique of
partitioning a large program into several individual segments which are then compiled
separately and installed into a library has proved a well worthwhile feature. It has
saved a considerable amount of time during program development and has introduced flexi-
bility into the software which assists task building of overlaid tasks. The program has
now been running for several weeks on the PDP, but has only been subjected to fairly
limited use. It is impossible to guarantee that the program is free from error and it is
hoped that any errors found by other users will be reported to the author. It is also
intended to run this program on a Dec VAX/11-750.
Acknowledgments

The author wishes to thank Dr. N. Tatham and Dr. J.M. Barrett of Flight Systems Department, for the initial part of the conversion exercise.
<table>
<thead>
<tr>
<th>No.</th>
<th>Author</th>
<th>Title, etc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A.A. Callaway</td>
<td>SAVANT - a database manipulation program for system architecture verification and design analysis. RAE Technical Report 80101 (1981)</td>
</tr>
<tr>
<td>2</td>
<td>Dec</td>
<td>IAS/RSX/UMS Coral 66 users guide; AA-D112A-TK.</td>
</tr>
<tr>
<td>3</td>
<td>Dec</td>
<td>Coral 66 language reference manual; AD-A11A-TC.</td>
</tr>
<tr>
<td>4</td>
<td>Dec</td>
<td>RSX-11M-plus task builder manual; AA-H266A-TC.</td>
</tr>
</tbody>
</table>
## APPENDIX A

### CROSS REFERENCE LISTS

**LIST OF FILENAMES AND PROCEDURE NAMES BY LEVEL NUMBER**

<table>
<thead>
<tr>
<th>File No</th>
<th>CMD</th>
<th>Procedure name</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAV2MN</td>
<td></td>
<td>MAIN PROGRAM</td>
<td>0</td>
</tr>
<tr>
<td>SETCMD</td>
<td>2</td>
<td>SET COMMAND</td>
<td>1</td>
</tr>
<tr>
<td>OBEYCM</td>
<td>3</td>
<td>OBEY COMMAND</td>
<td>1</td>
</tr>
<tr>
<td>RESET</td>
<td>4</td>
<td>RSP</td>
<td>2</td>
</tr>
<tr>
<td>INTTYE</td>
<td>5</td>
<td>ITE</td>
<td>2</td>
</tr>
<tr>
<td>INFIL</td>
<td>6</td>
<td>IFE</td>
<td>2</td>
</tr>
<tr>
<td>INFILM</td>
<td>7</td>
<td>IFM</td>
<td>2</td>
</tr>
<tr>
<td>LISENT</td>
<td>8</td>
<td>LEN</td>
<td>2</td>
</tr>
<tr>
<td>LISSSD</td>
<td>9</td>
<td>LSD</td>
<td>2</td>
</tr>
<tr>
<td>LISSSN</td>
<td>10</td>
<td>LSN</td>
<td>2</td>
</tr>
<tr>
<td>LISDNM</td>
<td>11</td>
<td>LDN</td>
<td>2</td>
</tr>
<tr>
<td>TRDATA</td>
<td>12</td>
<td>TDP</td>
<td>2</td>
</tr>
<tr>
<td>CHSSN</td>
<td>13</td>
<td>CSN</td>
<td>2</td>
</tr>
<tr>
<td>CHDNM</td>
<td>14</td>
<td>CDN</td>
<td>2</td>
</tr>
<tr>
<td>CHRATE</td>
<td>15</td>
<td>CRE</td>
<td>2</td>
</tr>
<tr>
<td>CHRATG</td>
<td>16</td>
<td>CRG</td>
<td>2</td>
</tr>
<tr>
<td>CHDART</td>
<td>17</td>
<td>CDR</td>
<td>2</td>
</tr>
<tr>
<td>CHSSRT</td>
<td>18</td>
<td>CSR</td>
<td>2</td>
</tr>
<tr>
<td>CHPRCE</td>
<td>19</td>
<td>CPE</td>
<td>2</td>
</tr>
<tr>
<td>CHPRCG</td>
<td>20</td>
<td>CPG</td>
<td>2</td>
</tr>
<tr>
<td>CHDAPR</td>
<td>22</td>
<td>CDF</td>
<td>2</td>
</tr>
<tr>
<td>CHSSPR</td>
<td>23</td>
<td>CSP</td>
<td>2</td>
</tr>
<tr>
<td>CHUNIE</td>
<td>24</td>
<td>CUE</td>
<td>2</td>
</tr>
<tr>
<td>CHUNIG</td>
<td>25</td>
<td>CUG</td>
<td>2</td>
</tr>
<tr>
<td>CHDAUN</td>
<td>26</td>
<td>CDU</td>
<td>2</td>
</tr>
<tr>
<td>DELDAR</td>
<td>27</td>
<td>DDR</td>
<td>2</td>
</tr>
<tr>
<td>DELIEN</td>
<td>28</td>
<td>DOE</td>
<td>2</td>
</tr>
<tr>
<td>FILENT</td>
<td>29</td>
<td>FEN</td>
<td>2</td>
</tr>
<tr>
<td>FILESD</td>
<td>30</td>
<td>FSD</td>
<td>2</td>
</tr>
<tr>
<td>CLRD</td>
<td>31</td>
<td>CLR</td>
<td>2</td>
</tr>
<tr>
<td>CONFIG</td>
<td>32</td>
<td>CFS</td>
<td>2</td>
</tr>
<tr>
<td>LISPAP</td>
<td>33</td>
<td>LDP</td>
<td>2</td>
</tr>
<tr>
<td>LISCOS</td>
<td>34</td>
<td>LCS</td>
<td>2</td>
</tr>
<tr>
<td>LISDAT</td>
<td>35</td>
<td>LDT</td>
<td>2</td>
</tr>
<tr>
<td>MAPDAT</td>
<td>36</td>
<td>MDT</td>
<td>2</td>
</tr>
<tr>
<td>SSPRDA</td>
<td>37</td>
<td>SSP</td>
<td>2</td>
</tr>
<tr>
<td>CKONS</td>
<td>38</td>
<td>CCS</td>
<td>2</td>
</tr>
<tr>
<td>CCR</td>
<td>39</td>
<td>CCR</td>
<td>2</td>
</tr>
<tr>
<td>LISMES</td>
<td>40</td>
<td>LBM</td>
<td>2</td>
</tr>
<tr>
<td>LISSDA</td>
<td>41</td>
<td>LDD</td>
<td>2</td>
</tr>
<tr>
<td>CALLOA</td>
<td>42</td>
<td>CLD</td>
<td>2</td>
</tr>
<tr>
<td>SUMMES</td>
<td>43</td>
<td>SUM</td>
<td>2</td>
</tr>
<tr>
<td>LISIME</td>
<td>44</td>
<td>LOM</td>
<td>2</td>
</tr>
<tr>
<td>SETRET</td>
<td>45</td>
<td>SRC</td>
<td>2</td>
</tr>
<tr>
<td>SETTAD</td>
<td>46</td>
<td>STA</td>
<td>2</td>
</tr>
<tr>
<td>LISSUB</td>
<td>47</td>
<td>LSS</td>
<td>2</td>
</tr>
<tr>
<td>REORDM</td>
<td>48</td>
<td>ROM</td>
<td>2</td>
</tr>
<tr>
<td>REFTHM</td>
<td>49</td>
<td>RFM</td>
<td>2</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLM</td>
<td>FILE MESSAGES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DMS</td>
<td>DISMANTLE SYSTEM</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>FMS</td>
<td>FORM SYSTEM</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LSA</td>
<td>LIST SUBADDRESSES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>FLS</td>
<td>FILE SCHEDULES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>UFS</td>
<td>UNIFORM SYSTEM</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>FLSCH</td>
<td>FLS FILE SCHEDULES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNFSY</td>
<td>UNFORM SYSTEM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INIT</td>
<td>INITIALISE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CMDWRN</td>
<td>COMMAND WARN</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>INTPCH</td>
<td>INT PARAM CHANGE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FLOPCH</td>
<td>FLO PARAM CHANGE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>INDAT</td>
<td>INPUT DATA</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>INNAM</td>
<td>INPUT NAMELIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LISREF</td>
<td>LIST REFLIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LISNAM</td>
<td>LIST NAMES</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CHNAM</td>
<td>CHANGE NAME</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CHNMEM</td>
<td>CHANGE NUMERIC ENTRY</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>GENNCH</td>
<td>GENERAL NUMBER CHANGE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CHDAV</td>
<td>CHANGE DATA VALUE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CHSSV</td>
<td>CHANGE SS VALUE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DELENT</td>
<td>DELETE ENTRY</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FILREF</td>
<td>FILE REFLIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FRMXTS</td>
<td>FORM TX SS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FRMXD</td>
<td>FORM RX SS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FATAL</td>
<td>FATAL CHECK</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FRMMESS</td>
<td>FORM MESSAGES</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FINBSS</td>
<td>FIND BUS SS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>PARNUM</td>
<td>PARTITION NUMBER</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SETMAP</td>
<td>SET MAP</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LISSSP</td>
<td>LIST SS PAIR</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FINMES</td>
<td>FIND MESSAGE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CKTA</td>
<td>CHECK TA</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LISHMA</td>
<td>LIST MESSAGE DATA</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FILNAM</td>
<td>FILE NAMELIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FNCOD</td>
<td>FIND CODE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FRMSUB</td>
<td>FORM SUBSETS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>OPHEL</td>
<td>OPEN LIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CLOLIS</td>
<td>CLOSE LIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CKTAST</td>
<td>CHECK TA SET</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FRMSAD</td>
<td>FORM SUBADDRESSES</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FILTAB</td>
<td>FILE TABLES</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FILSAD</td>
<td>FILE SUBADDRESSES</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LISCMD</td>
<td>LIST COMMANDS</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FNFIL</td>
<td>FIND FILE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CLRCEF</td>
<td>CLEAR CF LISTS</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CLRMEL</td>
<td>CLEAR MESSAGE LISTS</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>OPTION</td>
<td>OPTION</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>READEN</td>
<td>READ ENTRY</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>COMNM</td>
<td>CONVERT NAME</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>GETNV</td>
<td>GET NEW VALUE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FINENT</td>
<td>FIND ENTRY</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CKREF</td>
<td>CHECK REFERENCE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>GET&quot;LV&quot;</td>
<td>GET OLD VALUE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CHK &quot;N&quot;</td>
<td>CHANGE NUMBER</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>WORDS</td>
<td>WORDS</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FATLPK</td>
<td>FATAL PRINT</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>SENMES</td>
<td>SEND MESSAGE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CKSUB</td>
<td>CHECK SUBSET</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FINBC</td>
<td>FIND BUSCON</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FLSCH</td>
<td>FILE CW</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>ROW</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>FILISA</td>
<td>FILE ONE SA</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FILDIR</td>
<td>FILE DIRECTS</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>RECNAM</td>
<td>RECORD NAME</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>GETTR</td>
<td>GET TR</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>LOCNNAM</td>
<td>LOCATE NAME</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>DELNAM</td>
<td>DELETE NAME</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>REPOIN</td>
<td>REPOINT</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>SENCON</td>
<td>SEND CONTENTS</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>FILHDR</td>
<td>FILE HEADER</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>FINNAM</td>
<td>FIND NAME</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>MOVSTR</td>
<td>MOVE STRING</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>ACTPRC</td>
<td>ACTUAL PREC</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>CLISTR</td>
<td>CLISTREF</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>FILNUM</td>
<td>FILE NUMBER</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>OPFIRE</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>OPFIWR</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>File No</td>
<td>CMD</td>
<td>Procedure name</td>
<td>Level</td>
</tr>
<tr>
<td>---------</td>
<td>-----</td>
<td>---------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>ACTPRC 121</td>
<td>ACTUAL PREC</td>
<td>ACTUAL PREC</td>
<td>6</td>
</tr>
<tr>
<td>CALLDA 42</td>
<td>CLD</td>
<td>CALCULATE LOADINGS</td>
<td>2</td>
</tr>
<tr>
<td>CHDAFP 22</td>
<td>CDP</td>
<td>CHANGE DATA PREC</td>
<td>2</td>
</tr>
<tr>
<td>CHDAART 17</td>
<td>CDR</td>
<td>CHANGE DATA RATE</td>
<td>2</td>
</tr>
<tr>
<td>CHDAUN 26</td>
<td>CDU</td>
<td>CHANGE DATA UNITS</td>
<td>2</td>
</tr>
<tr>
<td>CHDAV 47</td>
<td>CHANGE DATA VALUE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CHDNM 14</td>
<td>CDN</td>
<td>CHANGE DATA NAME</td>
<td>2</td>
</tr>
<tr>
<td>CHNAM 64</td>
<td>CHANGE NAME</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CHNUM 103</td>
<td>CHANGE NUMBER</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CHNUME 65</td>
<td>CHANGE NUMERIC ENTRY</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CHPRCE 19</td>
<td>CPE</td>
<td>CHANGE PREC ENTRY</td>
<td>2</td>
</tr>
<tr>
<td>CHPRCG 20</td>
<td>CPG</td>
<td>CHANGE PREC GENERAL</td>
<td>2</td>
</tr>
<tr>
<td>CHRATE 15</td>
<td>CRE</td>
<td>CHANGE RATE ENTRY</td>
<td>2</td>
</tr>
<tr>
<td>CHRATG 16</td>
<td>CRG</td>
<td>CHANGE RATE GENERAL</td>
<td>2</td>
</tr>
<tr>
<td>CHSSN 13</td>
<td>CSN</td>
<td>CHANGE SS NAME</td>
<td>2</td>
</tr>
<tr>
<td>CHSSPR 23</td>
<td>CSP</td>
<td>CHANGE SS PREC</td>
<td>2</td>
</tr>
<tr>
<td>CHSSRT 18</td>
<td>CSR</td>
<td>CHANGE SS RATE</td>
<td>2</td>
</tr>
<tr>
<td>CHSSV 68</td>
<td>CHANGE SS VALUE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CHUNIE 24</td>
<td>CUE</td>
<td>CHANGE UNITS ENTRY</td>
<td>2</td>
</tr>
<tr>
<td>CHUNIG 25</td>
<td>CUG</td>
<td>CHANGE UNITS GENERAL</td>
<td>2</td>
</tr>
<tr>
<td>CKCONS 38</td>
<td>CCS</td>
<td>CHECK CONSISTENCY</td>
<td>2</td>
</tr>
<tr>
<td>CKCORR 39</td>
<td>CCR</td>
<td>CHECK CORRELATION</td>
<td>2</td>
</tr>
<tr>
<td>CKREF 101</td>
<td>CHECK REFERENCE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CKSUB 107</td>
<td>CHECK SUBSET</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CKTA 81</td>
<td>CHECK TA</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CKTAST 88</td>
<td>CHECK TA SET</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CLISTR 122</td>
<td>CLISTREF</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>CLOLIS 87</td>
<td>CLOSE LIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CLRCFL 94</td>
<td>CLEAR CF LISTS</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CLRDB 31</td>
<td>CLR</td>
<td>CLEAR DATABASE</td>
<td>2</td>
</tr>
<tr>
<td>CLRML 95</td>
<td>CLEAR MESSAGE LISTS</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CMDWRN 57</td>
<td>COMMAND WARN</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CONFIG 32</td>
<td>CFS</td>
<td>CONFIGURE SYSTEM</td>
<td>2</td>
</tr>
<tr>
<td>CONVNM 98</td>
<td>CONVERT NAME</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DEL1EN 28</td>
<td>DOE</td>
<td>DELETE ONE ENTRY</td>
<td>2</td>
</tr>
<tr>
<td>DELDAR 27</td>
<td>DDR</td>
<td>DELETE DATA REFERENCE</td>
<td>2</td>
</tr>
<tr>
<td>DELENT 69</td>
<td>DELETE ENTRY</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DELNAM 115</td>
<td>DELETE NAME</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>DISMAN 51</td>
<td>DMS</td>
<td>DISMANTLE SYSTEM</td>
<td>2</td>
</tr>
<tr>
<td>FATLCK 74</td>
<td>FATAL CHECK</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FATLP 105</td>
<td>FATAL PRINT</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FILESA 110</td>
<td>FILE ONE SA</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FILEW 109</td>
<td>FILE CW</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FILDIR 111</td>
<td>FILE DIRECTS</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FILENT 29</td>
<td>FEN</td>
<td>FILE ENTRIES</td>
<td>2</td>
</tr>
<tr>
<td>FILESD 30</td>
<td>FSD</td>
<td>FILE SUBSYSTEM DATA</td>
<td>2</td>
</tr>
<tr>
<td>FILHDR 118</td>
<td>FILE HEADER</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>FILMES 50</td>
<td>FLM</td>
<td>FILE MESSAGES</td>
<td>2</td>
</tr>
<tr>
<td>FILNAM 93</td>
<td>FILE NAMELIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FILNUM 124</td>
<td>FILE NUMBER</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>FILREF 70</td>
<td>FILE REFLIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FILSAD 91</td>
<td>FILE SUBADDRESSES</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FILLSCH 54</td>
<td>FLS</td>
<td>FILE SCHEDULES</td>
<td>2</td>
</tr>
<tr>
<td>FILTAB 90</td>
<td>FILE TABLES</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FINBC 108</td>
<td>FIND BUSCON</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FINBSS 76</td>
<td>FIND BUS SS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FINCOD 84</td>
<td>FIND CODE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Arguments</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>FINENT</td>
<td>FIND ENTRY</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FINFIL</td>
<td>FIND FILE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FINMES</td>
<td>FIND MESSAGE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FINNAM</td>
<td>FIND NAME</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>FLOPCH</td>
<td>FLO PARAM CHANGE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FRMMES</td>
<td>FORM MESSAGES</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FRMRXS</td>
<td>FORM RX SS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FRMSAD</td>
<td>FORM SUBADDRESSES</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FRMSUB</td>
<td>FORM SUBSETS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FRMSYS</td>
<td>FORM SYSTEM</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>FRMTXD</td>
<td>FORM TX DATA</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FRMTXS</td>
<td>FORM TX SS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>GENNCH</td>
<td>GENERAL NUMBER CHANGE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>GETNV</td>
<td>GET NEW VALUE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>GETOLV</td>
<td>GET OLD VALUE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>GETTR</td>
<td>GET TR</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>INDAT</td>
<td>INPUT DATA</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>INFILE</td>
<td>INPUT FILED ENTRIES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>INFILM</td>
<td>INPUT FILED MESSAGES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>INIT</td>
<td>INITIALISE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>INNAM</td>
<td>INPUT NAMELIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>INTPCH</td>
<td>INT PARAM CHANGE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>INTTYE</td>
<td>INPUT TERMINAL ENTRIES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISIME</td>
<td>LIST ONE MESSAGE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISCMD</td>
<td>LIST COMMANDS</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>LISCOS</td>
<td>LIST CONFIGURED SUBSYSTEMS</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISDAP</td>
<td>LIST DATA PATHS</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISDAT</td>
<td>LIST DATA TRAFFIC</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISDDA</td>
<td>LIST DIRECT DATA</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISDNN</td>
<td>LIST DATA NAMES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISENT</td>
<td>LIST ENTRIES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISMDA</td>
<td>LIST MESSAGE DATA</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LISMES</td>
<td>LIST BUS MESSAGES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISNAM</td>
<td>LIST NAMES</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LISREF</td>
<td>LIST REFLIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LISSAD</td>
<td>LIST SUBADDRESSES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISSSD</td>
<td>LIST SS DATA</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISSSN</td>
<td>LIST SS NAMES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LISSSP</td>
<td>LIST SS PAIR</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LISSUB</td>
<td>LIST SUBSETS</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LCMNAM</td>
<td>LOCATE NAME</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>MAPDAT</td>
<td>MAP DATA TRAFFIC</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MOVSTR</td>
<td>MOVE STRING</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>OBEYCM</td>
<td>OBEY COMMAND</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>OPELIS</td>
<td>OPEN LIST</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>OPTION</td>
<td>OPTION</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>READEN</td>
<td>READ ENTRY</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>RECNAM</td>
<td>RECORD NAME</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>REFMTH</td>
<td>REFORMAT MESSAGES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>REORDM</td>
<td>REORDER MESSAGE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>REPOIN</td>
<td>REPOINT</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>RESET PARAMETERS</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>SAV2MN</td>
<td>MAIN PROGRAM</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SENCON</td>
<td>SEND CONTENTS</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>SENMES</td>
<td>SEND MESSAGE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>SETCMD</td>
<td>SET COMMAND STRINGS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SETMAP</td>
<td>SET MAP</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SETRET</td>
<td>SET RETRY CODE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>SETTAD</td>
<td>SET TERMINAL ADDRESS</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>PARNUM</td>
<td>PARTITION NUMBER</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SSPRDA</td>
<td>SOURCE SINK PAIR DATA</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>SUMMES</td>
<td>SUM SUMMARISE MESSAGES</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>TRDATA</td>
<td>12</td>
<td>Trace Data Path</td>
<td>2</td>
</tr>
<tr>
<td>UNFSYS</td>
<td>55</td>
<td>Uniform System</td>
<td>2</td>
</tr>
<tr>
<td>WORDS</td>
<td>104</td>
<td>Words</td>
<td>4</td>
</tr>
</tbody>
</table>
## LIST OF FILENAMES AND PROCEDURE NAMES BY PROCEDURE NAME

<table>
<thead>
<tr>
<th>Procedure name</th>
<th>Level</th>
<th>File</th>
<th>No</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTUAL PREC</td>
<td>6</td>
<td>ACTPRC</td>
<td>121</td>
<td></td>
</tr>
<tr>
<td>CALCULATE LOADINGS</td>
<td>2</td>
<td>CALLOA</td>
<td>42</td>
<td>CLD</td>
</tr>
<tr>
<td>CHANGE DATA NAME</td>
<td>2</td>
<td>CHDMN</td>
<td>14</td>
<td>CDN</td>
</tr>
<tr>
<td>CHANGE DATA PREC</td>
<td>2</td>
<td>CHDAFP</td>
<td>22</td>
<td>CDF</td>
</tr>
<tr>
<td>CHANGE DATA RATE</td>
<td>2</td>
<td>CHDART</td>
<td>17</td>
<td>CDR</td>
</tr>
<tr>
<td>CHANGE DATA UNITS</td>
<td>2</td>
<td>CHDAUN</td>
<td>26</td>
<td>CDU</td>
</tr>
<tr>
<td>CHANGE DATA VALUE</td>
<td>3</td>
<td>CHDAV</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>CHANGE NAME</td>
<td>3</td>
<td>CHNAM</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>CHANGE NUMBER</td>
<td>4</td>
<td>CHNUM</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td>CHANGE NUMERIC ENTRY</td>
<td>3</td>
<td>CHNUME</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>CHANGE PREC ENTRY</td>
<td>2</td>
<td>CHPRCE</td>
<td>19</td>
<td>CPE</td>
</tr>
<tr>
<td>CHANGE PREC GENERAL</td>
<td>2</td>
<td>CHPRCG</td>
<td>20</td>
<td>CPG</td>
</tr>
<tr>
<td>CHANGE RATE ENTRY</td>
<td>2</td>
<td>CHRATE</td>
<td>15</td>
<td>CRE</td>
</tr>
<tr>
<td>CHANGE RATE GENERAL</td>
<td>2</td>
<td>CHRATG</td>
<td>16</td>
<td>CRG</td>
</tr>
<tr>
<td>CHANGE SS NAME</td>
<td>2</td>
<td>CHSSN</td>
<td>13</td>
<td>CSN</td>
</tr>
<tr>
<td>CHANGE SS PREC</td>
<td>2</td>
<td>CHSSPR</td>
<td>23</td>
<td>CSP</td>
</tr>
<tr>
<td>CHANGE SS RATE</td>
<td>2</td>
<td>CHSSRT</td>
<td>18</td>
<td>CSR</td>
</tr>
<tr>
<td>CHANGE SS VALUE</td>
<td>3</td>
<td>CHSSV</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>CHANGE UNITS ENTRY</td>
<td>2</td>
<td>CHUNIE</td>
<td>24</td>
<td>CUE</td>
</tr>
<tr>
<td>CHANGE UNITS GENERAL</td>
<td>2</td>
<td>CHUNIG</td>
<td>25</td>
<td>CUG</td>
</tr>
<tr>
<td>CHECK CONSISTENCY</td>
<td>2</td>
<td>CKCONS</td>
<td>38</td>
<td>CCS</td>
</tr>
<tr>
<td>CHECK CORRELATION</td>
<td>2</td>
<td>CKCORR</td>
<td>39</td>
<td>CCR</td>
</tr>
<tr>
<td>CHECK REFERENCE</td>
<td>4</td>
<td>CKREF</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>CHECK SUBSET</td>
<td>4</td>
<td>CKSUB</td>
<td>107</td>
<td></td>
</tr>
<tr>
<td>CHECK TA</td>
<td>3</td>
<td>CKTA</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>CHECK TA SET</td>
<td>3</td>
<td>CKTAST</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>CLEAR CF LISTS</td>
<td>4</td>
<td>CLRCLF</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>CLEAR DATABASE</td>
<td>2</td>
<td>CLRDB</td>
<td>31</td>
<td>CLR</td>
</tr>
<tr>
<td>CLEAR MESSAGE LISTS</td>
<td>4</td>
<td>CLRMES</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>CLISTREF</td>
<td>6</td>
<td>CLISTR</td>
<td>122</td>
<td></td>
</tr>
<tr>
<td>CLOSE LIST</td>
<td>3</td>
<td>CLOLIS</td>
<td>87</td>
<td></td>
</tr>
<tr>
<td>COMMAND WARN</td>
<td>3</td>
<td>CMDWRN</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>CONFIGURE SYSTEM</td>
<td>2</td>
<td>CONFIG</td>
<td>32</td>
<td>CFS</td>
</tr>
<tr>
<td>CONVERT NAME</td>
<td>4</td>
<td>CONVNM</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>DELETE DATA REFERENCE</td>
<td>2</td>
<td>DELDAR</td>
<td>27</td>
<td>DDR</td>
</tr>
<tr>
<td>DELETE ENTRY</td>
<td>3</td>
<td>DELENT</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>DELETE NAME</td>
<td>5</td>
<td>DELNAM</td>
<td>115</td>
<td></td>
</tr>
<tr>
<td>DELETE ONE ENTRY</td>
<td>2</td>
<td>DELENI</td>
<td>28</td>
<td>DOE</td>
</tr>
<tr>
<td>DISMANTLE SYSTEM</td>
<td>2</td>
<td>DISMAN</td>
<td>51</td>
<td>DMS</td>
</tr>
<tr>
<td>FATAL CHECK</td>
<td>3</td>
<td>FATLCK</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>FATAL PRINT</td>
<td>4</td>
<td>FATLPR</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>FILE CW</td>
<td>4</td>
<td>FILCW</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>FILE DIRECTS</td>
<td>4</td>
<td>FILDIR</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>FILE ENTRIES</td>
<td>2</td>
<td>FILENT</td>
<td>29</td>
<td>FEN</td>
</tr>
<tr>
<td>FILE HEADER</td>
<td>5</td>
<td>FILHDR</td>
<td>118</td>
<td></td>
</tr>
<tr>
<td>FILE MESSAGES</td>
<td>2</td>
<td>FILMES</td>
<td>50</td>
<td>FLM</td>
</tr>
<tr>
<td>FILE NAMELIST</td>
<td>3</td>
<td>FILNAM</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>FILE NUMBER</td>
<td>6</td>
<td>FILNUM</td>
<td>123</td>
<td></td>
</tr>
<tr>
<td>FILE ONE SA</td>
<td>4</td>
<td>FILISA</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>FILE REFLIST</td>
<td>3</td>
<td>FILREF</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>FILE SCHEDULES</td>
<td>2</td>
<td>FILSCH</td>
<td>54</td>
<td>FLS</td>
</tr>
<tr>
<td>FILE SUBADDRESSES</td>
<td>3</td>
<td>FILSAD</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>FILE SUBSYSTEM DATA</td>
<td>2</td>
<td>FILESD</td>
<td>30</td>
<td>FSD</td>
</tr>
<tr>
<td>FILE TABLES</td>
<td>3</td>
<td>FILTAB</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>FIND BUS SS</td>
<td>3</td>
<td>FINBSS</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>FIND BUSCON</td>
<td>4</td>
<td>FINRCS</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>FIND CODE</td>
<td>3</td>
<td>FINCTD</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>Trace Data Path</td>
<td>Uniform System</td>
<td>Words</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRDATA 12</td>
<td>UNFSYS 55</td>
<td>WORDS 104</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX B

PROGRAM STRUCTURE

LEVEL NO 1 2 3 4 5 6

SAVANT COMMAND

CLD CALLOG-CMDWRN
|---------------------FINBC---------------------FINNAM

CDP CHDAPR-CMDWRN
|-----CHDAV--GETNV
|---------------------LOCNAM--FINNAM

CDR CHDART-CMDWRN
|-----CHDAV--GETNV
|---------------------LOCNAM--FINNAM

CDU CHDAUN-CMDWRN
|---------------------LOCNAM--FINNAM

|---------------------RECNAM--FINNAM
|---------------------CKREF
|                     |
|---------------------DELNM
|---------------------REPOIN

CDN CHDMN-CMDWRN
|-----CHNA----FINENT--LOCNAM--FINNAM

|---------------------MOVSTR
|---------------------CONVNM--DELNM
|                     |
|---------------------REPOIN
|                     |
|---------------------OPIN

CPE CHPRCE-CMDWRN
|-----CHNME--FINENT--LOCNAM--FINNAM

|---------------------GETTR
|---------------------GEINV

DOE DEL1EN-CMDWRN
|---------------------FINENT--LOCNAM--FINNAM

|---------------------GETTR
|---------------------DELENT
|---------------------CKREF--DELNM

|---------------------REPOIN

DDR DELDAR-CMDWRN
|---------------------LOCNAM--FINNAM

|---------------------DELENT
|---------------------CKREF--DELNM

|---------------------REPOIN
|---------------------DELNM
|---------------------REPOIN
DMS  DISMAN--CMDWRN
      --------------CLRML
      --------------CLRCFL

FEN  FILENT--CMDWRN
      -------FILREF--FINFIL--!-------OPFIKE
          !-------OPFIWR
          OPTIONS
          ------------ACTPRC
          ------------FILNUM

FSD  FILESD--CMDWRN
      ---------------------LOCNAM--FINNAM
      -------FILREF--FINFIL--!-------OPFIKE
          !-------OPFIWR
          OPTIONS
          ------------ACTPRC
          ------------FILNUM

FLM  FILMES--CMDWRN
      ---------------------FINFIL--!-------OPFIKE
          !-------OPFIWR
          OPTIONS
          ------------FINNAM
          ------------FILNUM

CPG  CHPRCG--CMDWRN
      !-------GENNCH--GETOLV
          !-------GETNV
          !-------CHNUM

CRE  CHRATE--CMDWRN
      !-------CHNUME--FINENT--LOCNAM--FINNAM
          !-------GETTR
          !-------GETNV

CRG  CHRATG--CMDWRN
      !-------GENNCH--GETOLV
          !-------GETNV
          !-------CHNUM

CSN  CHSSN--CMDWRN
      !-------CHNAM--FINNAM
          !-------MOVSTR
          !-------CONVNM--DELMAN
          !-------REPOIN
          !-------OPTION

CSP  CHSSPR--CMDWRN
      !-------CHSSV--LOCNAM--FINNAM
          !-------GETOLV
          !-------CHNUM
          !-------GETNV
<table>
<thead>
<tr>
<th>CUG</th>
<th>CHUNIG---CMDWRN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--------------CHNAM---------FINNAM</td>
</tr>
<tr>
<td></td>
<td>------------MOVSTR</td>
</tr>
<tr>
<td></td>
<td>---------------------OPTION</td>
</tr>
<tr>
<td></td>
<td>------------CONVNM---DELNAM</td>
</tr>
<tr>
<td></td>
<td>-----REPOIN</td>
</tr>
<tr>
<td>CCS</td>
<td>CKCONS---CMDWRN</td>
</tr>
<tr>
<td></td>
<td>-----FATLCK--FATLPR---------ACTPRC</td>
</tr>
<tr>
<td></td>
<td>---------------------ACTPRC</td>
</tr>
<tr>
<td>CCR</td>
<td>CKCORR---CMDWRN</td>
</tr>
<tr>
<td>CLR</td>
<td>CLRDB---CMDWRN</td>
</tr>
<tr>
<td></td>
<td>-----INIT-----CLRMEL</td>
</tr>
<tr>
<td></td>
<td>!-----CLRCFL</td>
</tr>
<tr>
<td></td>
<td>!-----LISCMD</td>
</tr>
<tr>
<td></td>
<td>---------------------OPTION</td>
</tr>
<tr>
<td>CFS</td>
<td>CONFIG---CMDWRN</td>
</tr>
<tr>
<td></td>
<td>-----FRMTXD</td>
</tr>
<tr>
<td></td>
<td>-----FRMRXS--WORDS</td>
</tr>
<tr>
<td></td>
<td>!---------------------ACTPRC</td>
</tr>
<tr>
<td></td>
<td>-----FATLCK--FATLPR</td>
</tr>
<tr>
<td></td>
<td>!---------------------ACTPRC</td>
</tr>
<tr>
<td></td>
<td>!------FRMMES</td>
</tr>
<tr>
<td></td>
<td>!-----FINBSS</td>
</tr>
<tr>
<td></td>
<td>!-----FRMTXS</td>
</tr>
<tr>
<td></td>
<td>!-----PARNUM</td>
</tr>
<tr>
<td>LDP</td>
<td>LISDAP---CMDWRN</td>
</tr>
<tr>
<td></td>
<td>--------------OPTION</td>
</tr>
<tr>
<td></td>
<td>!---------------ACTPRC</td>
</tr>
<tr>
<td>LDT</td>
<td>LISDAT---CMDWRN</td>
</tr>
<tr>
<td></td>
<td>!--------------OPTION</td>
</tr>
<tr>
<td></td>
<td>!------LISSSP</td>
</tr>
<tr>
<td>LDD</td>
<td>LISDDA---CMDWRN</td>
</tr>
<tr>
<td></td>
<td>!------LISMDA</td>
</tr>
<tr>
<td></td>
<td>!------OPTION</td>
</tr>
<tr>
<td></td>
<td>!------SENMES---SENCON</td>
</tr>
<tr>
<td>LDN</td>
<td>LISDNM---CMDWRN</td>
</tr>
<tr>
<td></td>
<td>!------LISNAM--OPTION</td>
</tr>
<tr>
<td>LEN</td>
<td>LISENT---CMDWRN</td>
</tr>
<tr>
<td></td>
<td>!------LISREF--OPTION</td>
</tr>
<tr>
<td></td>
<td>!---------------------ACTPRC</td>
</tr>
<tr>
<td>LBM</td>
<td>LISMES---CMDWRN</td>
</tr>
<tr>
<td></td>
<td>!------LISMDA--OPTION</td>
</tr>
<tr>
<td></td>
<td>!------SENMES---SENCON</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LSA</td>
<td>LISSAD--CMDWRN</td>
</tr>
<tr>
<td>LSD</td>
<td>LISSSD--CMDWRN                                                             LOCNAM--FINNAM</td>
</tr>
<tr>
<td></td>
<td>LISREF--OPTION</td>
</tr>
<tr>
<td></td>
<td>ACTPRC</td>
</tr>
<tr>
<td>LSN</td>
<td>LISSSN--CMDWRN                                                             LOCNAM--OPTION</td>
</tr>
<tr>
<td>LSS</td>
<td>LISSUB--CMDWRN                                                             FRMSUB--CKSUB</td>
</tr>
<tr>
<td></td>
<td>OPTION</td>
</tr>
<tr>
<td>LOM</td>
<td>LISEME--CMDWRN                                                             FINMES</td>
</tr>
<tr>
<td></td>
<td>SENMES--SENCN</td>
</tr>
<tr>
<td>MDT</td>
<td>MAPDA1--CMDWRN                                                             CLISTR</td>
</tr>
<tr>
<td></td>
<td>OPTION</td>
</tr>
<tr>
<td></td>
<td>FINFIL--OPFIRE</td>
</tr>
<tr>
<td></td>
<td>SETMAP--OPFWR</td>
</tr>
<tr>
<td></td>
<td>OPTION</td>
</tr>
<tr>
<td>RFM</td>
<td>REFMTM--CMDWRN                                                             FINMES</td>
</tr>
<tr>
<td></td>
<td>OPELIS</td>
</tr>
<tr>
<td></td>
<td>CLOLIS</td>
</tr>
<tr>
<td></td>
<td>OPTION</td>
</tr>
<tr>
<td></td>
<td>SENMES--SENCN</td>
</tr>
<tr>
<td>ROM</td>
<td>REORDM--CMDWRN                                                             FINMES</td>
</tr>
<tr>
<td></td>
<td>SENMES--SENCN</td>
</tr>
<tr>
<td></td>
<td>MODABL</td>
</tr>
<tr>
<td>RSP</td>
<td>RESET--CMDWRN                                                              INTPCH--OPTION</td>
</tr>
<tr>
<td></td>
<td>FLOPCH--OPTION</td>
</tr>
<tr>
<td>SRC</td>
<td>SETRET--CMDWRN                                                             FINMES</td>
</tr>
<tr>
<td></td>
<td>FINCOD</td>
</tr>
<tr>
<td></td>
<td>OPTION</td>
</tr>
<tr>
<td>STA</td>
<td>SETTAD--CMDWRN                                                             CKTA</td>
</tr>
<tr>
<td></td>
<td>LOCNAM--FINNAM</td>
</tr>
<tr>
<td>SSP</td>
<td>SSPRDA--CMDWRN                                                             LOCNAM--FINNAM</td>
</tr>
<tr>
<td></td>
<td>LOCNAM--CLISTR</td>
</tr>
<tr>
<td></td>
<td>LISSSP</td>
</tr>
<tr>
<td>SUM</td>
<td>SUMMES--CMDWRN                                                             FINMES</td>
</tr>
<tr>
<td></td>
<td>OPTION</td>
</tr>
<tr>
<td>TDP</td>
<td>TRDATA---CMDWRN</td>
</tr>
<tr>
<td>------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>LOCNAM---FINNAM</td>
</tr>
<tr>
<td></td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>ACTPRC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UFS</th>
<th>UNFSYS---CMDWRN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX C

TYPICAL LIB/OVL FILES

`CORAL' NUM00
`INCLUDE' "SY:CORIOL.CRL"
`COMMON' (LABEL 'OVLOO
  `INCLUDE' "SY:PROC5.COR"
);
`INCLUDE' "SY:SAVGBL.COR"
`SEGMENT' OVLOO
`BEGIN'
  `INCLUDE' "SY:MACDEF.COR"
  `INCLUDE' "SY:CALLOA.COR"
`END'
`FINISH'

`CORAL' MOD00
`INCLUDE' "SY:CORIOL.CRL"
`COMMON' (LABEL 'LIB00
  `INCLUDE' "SY:PROC5.COR"
);
`INCLUDE' "SY:SAVGBL.COR"
`SEGMENT' LIB00
`BEGIN'
  `INCLUDE' "SY:MACDEF.COR"
  `INCLUDE' "SY:ACTPRC.COR"
`END'
`FINISH'
## APPENDIX D

### OVL/LIB SEGMENT NAME CORRESPONDENCE

<table>
<thead>
<tr>
<th>LIB/MOD</th>
<th>SEGMENT NAME</th>
<th>OVL/NUM</th>
<th>SEGMENT NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIB00</td>
<td>ACTPRC</td>
<td>OVL00</td>
<td>CALLOA</td>
</tr>
<tr>
<td>LIB01</td>
<td>CHDAPR</td>
<td>OVL01</td>
<td>CHDART</td>
</tr>
<tr>
<td>LIB02</td>
<td>CHDART</td>
<td>OVL02</td>
<td>CHDART</td>
</tr>
<tr>
<td>LIB03</td>
<td>CHDAUN</td>
<td>OVL03</td>
<td>CHDAUN</td>
</tr>
<tr>
<td>LIB04</td>
<td>CHDMM</td>
<td>OVL04</td>
<td>CHDMM</td>
</tr>
<tr>
<td>LIB05</td>
<td>CHFRCE</td>
<td>OVL05</td>
<td>CHFRCE</td>
</tr>
<tr>
<td>LIB06</td>
<td>CHFRCG</td>
<td>OVL06</td>
<td>CHFRCG</td>
</tr>
<tr>
<td>LIB07</td>
<td>CHPATE</td>
<td>OVL07</td>
<td>CHPATE</td>
</tr>
<tr>
<td>LIB08</td>
<td>CHRATG</td>
<td>OVL08</td>
<td>CHRATG</td>
</tr>
<tr>
<td>LIB09</td>
<td>CHSSN</td>
<td>OVL09</td>
<td>CHSSN</td>
</tr>
<tr>
<td>LIB10</td>
<td>CHSSPR</td>
<td>OVL10</td>
<td>CHSSPR</td>
</tr>
<tr>
<td>LIB11</td>
<td>CHSSRT</td>
<td>OVL11</td>
<td>CHSSRT</td>
</tr>
<tr>
<td>LIB12</td>
<td>CHUNIE</td>
<td>OVL12</td>
<td>CHUNIE</td>
</tr>
<tr>
<td>LIB13</td>
<td>CHUNIG</td>
<td>OVL13</td>
<td>CHUNIG</td>
</tr>
<tr>
<td>LIB14</td>
<td>CKCONS</td>
<td>OVL14</td>
<td>CKCONS</td>
</tr>
<tr>
<td>LIB15</td>
<td>CKCORR</td>
<td>OVL15</td>
<td>CKCORR</td>
</tr>
<tr>
<td>LIB16</td>
<td>CLRDDB</td>
<td>OVL16</td>
<td>CLRDDB</td>
</tr>
<tr>
<td>LIB17</td>
<td>CONFIG</td>
<td>OVL17</td>
<td>CONFIG</td>
</tr>
<tr>
<td>LIB18</td>
<td>DELIEN</td>
<td>OVL18</td>
<td>DELIEN</td>
</tr>
<tr>
<td>LIB19</td>
<td>DELLAR</td>
<td>OVL19</td>
<td>DELLAR</td>
</tr>
<tr>
<td>LIB20</td>
<td>DISMAN</td>
<td>OVL20</td>
<td>DISMAN</td>
</tr>
<tr>
<td>LIB21</td>
<td>FILENT</td>
<td>OVL21</td>
<td>FILENT</td>
</tr>
<tr>
<td>LIB22</td>
<td>FILESMD</td>
<td>OVL22</td>
<td>FILESMD</td>
</tr>
<tr>
<td>LIB23</td>
<td>FIMES</td>
<td>OVL23</td>
<td>FIMES</td>
</tr>
<tr>
<td>LIB24</td>
<td>FILSCH</td>
<td>OVL24</td>
<td>FILSCH</td>
</tr>
<tr>
<td>LIB25</td>
<td>FRMSYS</td>
<td>OVL25</td>
<td>FRMSYS</td>
</tr>
<tr>
<td>LIB26</td>
<td>INFFILE</td>
<td>OVL26</td>
<td>INFFILE</td>
</tr>
<tr>
<td>LIB27</td>
<td>INFILFM</td>
<td>OVL27</td>
<td>INFILFM</td>
</tr>
<tr>
<td>LIB28</td>
<td>INIT</td>
<td>OVL28</td>
<td>INIT</td>
</tr>
<tr>
<td>LIB29</td>
<td>INTITYE</td>
<td>OVL29</td>
<td>INTITYE</td>
</tr>
<tr>
<td>LIB30</td>
<td>LISCOS</td>
<td>OVL30</td>
<td>LISCOS</td>
</tr>
<tr>
<td>LIB31</td>
<td>LISDAP</td>
<td>OVL31</td>
<td>LISDAP</td>
</tr>
<tr>
<td>LIB32</td>
<td>LISDAT</td>
<td>OVL32</td>
<td>LISDAT</td>
</tr>
<tr>
<td>LIB33</td>
<td>LISDDA</td>
<td>OVL33</td>
<td>LISDDA</td>
</tr>
<tr>
<td>LIB34</td>
<td>LISTSMM</td>
<td>OVL34</td>
<td>LISTSNM</td>
</tr>
<tr>
<td>LIB35</td>
<td>LISNENT</td>
<td>OVL35</td>
<td>LISNENT</td>
</tr>
<tr>
<td>LIB36</td>
<td>LISMES</td>
<td>OVL36</td>
<td>LISMES</td>
</tr>
<tr>
<td>LIB37</td>
<td>LISSAD</td>
<td>OVL37</td>
<td>LISSAD</td>
</tr>
<tr>
<td>LIB38</td>
<td>LISSSAD</td>
<td>OVL38</td>
<td>LISSSAD</td>
</tr>
<tr>
<td>LIB39</td>
<td>LISSSD</td>
<td>OVL39</td>
<td>LISSSD</td>
</tr>
<tr>
<td>LIB40</td>
<td>LISSSN</td>
<td>OVL40</td>
<td>LISSSN</td>
</tr>
<tr>
<td>LIB41</td>
<td>LISSUB</td>
<td>OVL41</td>
<td>LISSUB</td>
</tr>
<tr>
<td>LIB42</td>
<td>MAPDAT</td>
<td>OVL42</td>
<td>MAPDAT</td>
</tr>
<tr>
<td>LIB43</td>
<td>REFMTM</td>
<td>OVL43</td>
<td>REFMTM</td>
</tr>
<tr>
<td>LIB44</td>
<td>REORDM</td>
<td>OVL44</td>
<td>REORDM</td>
</tr>
<tr>
<td>LIB45</td>
<td>RESET</td>
<td>OVL45</td>
<td>RESET</td>
</tr>
<tr>
<td>LIB46</td>
<td>SETRET</td>
<td>OVL46</td>
<td>SETRET</td>
</tr>
<tr>
<td>LIB47</td>
<td>SETTAD</td>
<td>OVL47</td>
<td>SETTAD</td>
</tr>
<tr>
<td>LIB48</td>
<td>SSPRDA</td>
<td>OVL48</td>
<td>SSPRDA</td>
</tr>
<tr>
<td>LIB49</td>
<td>SUMMES</td>
<td>OVL49</td>
<td>SUMMES</td>
</tr>
<tr>
<td>LIB50</td>
<td>TRDATA</td>
<td>OVL50</td>
<td>TRDATA</td>
</tr>
<tr>
<td>LIB51</td>
<td>UNFSYS</td>
<td>OVL51</td>
<td>UNFSYS</td>
</tr>
<tr>
<td>LIB52</td>
<td>LISREF</td>
<td>OVL52</td>
<td>LISREF</td>
</tr>
</tbody>
</table>
LIB53  LISSSP
LIB54  LOCNAM
LIB55  MOVSTR
LIB56  OPELIS
LIB57  READEN
LIB58  OPFIRE
LIB59  RECNAM
LIB60  REPOIN
LIB61  SENCON
LIB62  SENMES
LIB63  SETMAP
LIB64  PARNUM
LIB65  WORDS
LIB66  OPFIWR
LIB67  MODABL
APPENDIX E1
---------

GLOBAL FILES
---------

'NLIST'

'COMMENT' COROL.CRL - DEC CORAL LIBRARY SPECS;

'COMMENT' MACRO DEFINITIONS FOR LIBRARY SPECS:

'DEFINE' VB '"VALUE''BYTE"';
'DEFINE' VI '"VALUE''INTEGER"';
'DEFINE' VF '"VALUE''FLOATING"';
'DEFINE' LB '"LOCATION''BYTE''';
'DEFINE' LI '"LOCATION''INTEGER''';
'DEFINE' LF '"LOCATION''FLOATING''';
'DEFINE' BA '"BYTE''ARRAY''';

'COMMENT' LIBRARY ROUTINES FROM COROTS.QLB)

'LIBRARY' ( 'PROCEDURE' NEWLINE (VB,VB);
'PROCEDURE' DEFLINE (VB,VI,"LABEL");
'PROCEDURE' DELETE (VB);
'PROCEDURE' CREATE (VB,VI,VB,"LABEL");
'PROCEDURE' OPEN (VB,VI,VB,"LABEL");
'PROCEDURE' WRITETEXT (VB,VI);
'PROCEDURE' REACH (VB,LB);
'PROCEDURE' SPACES (VB,VB);
'PROCEDURE' WRITE (VB,VI,VI);
'PROCEDURE' WRITEVF (VB,VI,VI);
'PROCEDURE' WRITEX (VB,VI);
'PROCEDURE' IREAD (VB,VI,LI);
'PROCEDURE' FREAD (VB,LF);
'PROCEDURE' TIDEF (VB,VI,"LABEL");
'PROCEDURE' CLOSE (VB);
'PROCEDURE' RELEASE (VB);
'PROCEDURE' READTEXT (VB,VI,BA,LI);
'PROCEDURE' WRITEREG (VB,VI,BA,LI);
'PROCEDURE' ERASE (VI,VI,VI,VI,VI,VI,BA) );

'LIST'

'NLIST'

'PROC'S.COR
---------

'INTEGER'

PROCEDURE' ACTREC ('VALUE''INTEGER''');
'PROCEDURE' CALLDA;
'PROCEDURE' CHDFAP,
'PROCEDURE' CHDFAT
'PROCEDURE' CHDAUN;
'PROCEDURE' CDAV ('VALUE''INTEGER''');
'PROCEDURE' CHDSAV ('VALUE''INTEGER''');
'PROCEDURE' CHDNN;
'PROCEDURE' CHMM ('BYTE''ARRAY'');
'PROCEDURE' CPHM ('VALUE''INTEGER''; 'VALUE''INTEGER''; 'VAL.: ''INTEGER''; 'VALUE''INTEGER''; 'VALUE''INTEGER'';
PROCEDURE' CPHREC;
'PROCEDURE' CFRCG;
'PROCEDURE' CHRATE I
'PROCEDURE' CRCHR 1
'PROCEDURE' CHSSNM I
'PROCEDURE' CHSSPR I
'PROCEDURE' CHSSRT I
'PROCEDURE' CHSSVM (VALUE 'INTEGER') I
'PROCEDURE' CHVNM I
'PROCEDURE' CHVNM I
'PROCEDURE' CSKON I
'PROCEDURE' CSKOR I
'PROCEDURE' CSKFR (BYTE 'ARRAY', VALUE 'INTEGER') I
'PROCEDURE' CKSUB (VALUE 'INTEGER', VALUE 'INTEGER', LOCATION 'INTEGER') I
'PROCEDURE' CKTA (VALUE 'INTEGER', VALUE 'INTEGER', VALUE 'INTEGER', LOCATION 'INTEGER') I
'PROCEDURE' CKTAST (LOCATION 'INTEGER') I

'INTEGER'
'PROCEDURE' CLISTR (VALUE 'INTEGER') I
'PROCEDURE' CLOLIS (VALUE 'INTEGER') I
'PROCEDURE' CLRFL I
PROCEDURE' CLRDB I
'PROCEDURE' CLRML I

'INTEGER'
'PROCEDURE' CMPSTR (VALUE 'INTEGER', VALUE 'INTEGER') I
'PROCEDURE' CONFIG I
'PROCEDURE' CMPWM (BYTE 'ARRAY', VALUE 'INTEGER', VALUE 'INTEGER') I
'PROCEDURE' CNEL I
'PROCEDURE' CDEL I
'PROCEDURE' DLENT (VALUE 'INTEGER') I
'PROCEDURE' DLEN (BYTE 'ARRAY', VALUE 'INTEGER') I
'PROCEDURE' DLETH I
'PROCEDURE' DSTM I
'PROCEDURE' FALCK (LOCATION 'INTEGER') I
'PROCEDURE' FALLP (VALUE 'INTEGER', VALUE 'INTEGER', VALUE 'INTEGER', LOCATION 'INTEGER') I
'PROCEDURE' FILEM (VALUE 'INTEGER', VALUE 'INTEGER', VALUE 'INTEGER', VALUE 'INTEGER') I
'PROCEDURE' FILET I
'PROCEDURE' FILES I

'PROCEDURE' FLDIR (VALUE 'INTEGER', VALUE 'INTEGER', LOCATION 'INTEGER') I
'PROCEDURE' FILET I
'PROCEDURE' FILEREF (VALUE 'INTEGER') I
'PROCEDURE' FILES I
'PROCEDURE' FILSAD I
'PROCEDURE' FILSCH I
'PROCEDURE' FILTAB I
'PROCEDURE' FILISA (VALUE 'INTEGER', VALUE 'INTEGER', VALUE 'INTEGER', VALUE 'INTEGER', VALUE 'INTEGER', LOCATION 'INTEGER') I
'PROCEDURE' FINDC (LOCATION 'INTEGER') I
'PROCEDURE' FINDS (LOCATION 'INTEGER') I
'PROCEDURE' FINDCD (LOCATION 'INTEGER', LOCATION 'INTEGER') I
'PROCEDURE' FINEST (LOCATION 'INTEGER', LOCATION 'INTEGER', LOCATION 'INTEGER') I
'PROCEDURE' FINIL (VALUE 'INTEGER', VALUE 'INTEGER', LOCATION 'INTEGER') I
'PROCEDURE' FINM (VALUE 'INTEGER', LOCATION 'INTEGER') I
'PROCEDURE' FINMG (BYTE 'ARRAY', LOCATION 'INTEGER') I
'PROCEDURE' FLOCH (VALUE 'INTEGER', VALUE 'FLOATING', LOCATION 'FLOATING') I
'PROCEDURE' FRAMES I
'PROCEDURE' FRMXXS I
'PROCEDURE' FRMSAD (LOCATION 'INTEGER') I
'PROCEDURE' FRMSUB I
'PROCEDURE' FRMSYS I
'PROCEDURE' FRMTXD I
'PROCEDURE' FRMTXS I
'PROCEDURE' GENMCH (VALUE 'INTEGER') I
'PROCEDURE' GETHUM (VALUE 'BYTE', BYTE 'ARRAY') I
'PROCEDURE' GETIV (VALUE 'INTEGER', LOCATION 'INTEGER') I
'PROCEDURE' GETOLV (VALUE 'INTEGER', LOCATION 'INTEGER') I

'FLOATING'
'DEFINE' MAXENTRIES "500" (### Max number of database entries allowed)
'DEFINE' MAXMESS "200" (### Max number of data bus messages allowed)
'DEFINE' MAXSS "50" (### Max number of subsystem names)
'DEFINE' MAXDATA "120" (### Max number of data names)
'DEFINE' MAXSTUDS "13" (### Max number of words to hold a STRING - gives 13 char. STRING)
'DEFINE' TTO '1' (### Channel number for terminal input)
'DEFINE' TTO '2' (### Channel number for terminal output)
'DEFINE' INFIL '3' (### Channel number for disc file input access)
'DEFINE' OUTFIL '4' (### Channel number for disc file output access)

'COMMENT' PROGRAM STATES.
*** These macros define the various values taken
*** by the variable STATE!

'DEFINE' STOPPED '0'
'DEFINE' EMPTY '1'
'DEFINE' OPENX '2'
'DEFINE' CONFIGURED '3'
'DEFINE' LIMITED '4'
'DEFINE' FORMED '5'
'DEFINE' LIMITED FORMED '6'
'DEFINE' INCONSISTENT '7'

'COMMENT' PROGRAMMING AIDS.
*** These macros are designed to make coding more
*** explicit and to aid structured programming.

'DEFINE' WHILE (COND) 'FOR' WHILF=0 'WHILE' COND
### Gives contrived while-loop clause
'DEFINE' TRUE '1' (### For pseudo boolean operations)
'DEFINE' FALSE '0'
'DEFINE' NOT(X) 'IF' X=TRUE 'THEN' FALSE 'ELSE' TRUE' (### Ditto)
'DEFINE' INC(X) 'X=X+1'
'DEFINE' DEC(X) 'X=X-1'
'DEFINE' DO NOTHING ''
'DEFINE' STRING(LOC) 'LOCATION'(LOC(O)) (### Points to start of string in 'LOC' BUFFER)

'COMMON' GLOBAL:

'COMMON' INTEGER CONSTANTS.
### These are the preset integers defining the particular
### system architecture, whose values can be changed by
### commanding RSP:

'INTEGER' LOWEST RATE, WORD LENGTH, MESSAGE LENGTH, WORD OVERHEAD, BIT RATE, TERMINAL LIMIT
10 16 32 4 1000 30

'COMMON' REAL CONSTANTS.
### Similarly, these are the real preset constants
### whose values can be changed by commanding RSP:

'FLOATING' CONT OVERHEAD, TERM OVERHEAD := 2.6, 4.91

'COMMON' GLOBAL INTEGERS:

'INTEGER' ENTRIES, TOTAL TX DATA, TOTAL MESSAGES, CONF SS COUNT, STATE, WHILF, OBEYED)

FS 452
'COMMENT'
### ENTRIES: Count of REFLIST entries.
### TOTAL TX DATA: Count of TXLIST entries.
### TOTAL MESSAGES: Count of MESSAGE HEADER & MESSAGE DATA entries.
### CONF SS COUNT: Count of configured subsystems.
### STATE: The current database state.
### WHDPOD: Dummy variable for contrived while-loop.

'COMMENT' INTENTIONS FOR FREQUENTLY USED STRINGS.
### These hold pointers to text strings which appear frequently
### within the program.

'COMMENT' EACH PAGE, CONTINUE:
### EACH PAGE: "TO PAUSE ON EACH PAGE".
### CONTINUE: "TO CONTINUE".

'COMMENT' TEMPORARY VARIABLES:
'BYTE' 'ARRAY' AITEMP(0:19), ERTSTATE(0:23);
'INTEGER' ITEMPP, ITEMP;

'COMMENT' DATABASE ARRAYS
'BYTE' 'ARRAY' SS NAME, UNITS NAME(0:MAXSS), MAXSTWDS, DATA NAME(0:MAXDATA), MAXSTWDS)
'INTEGER' 'ARRAY' CONF SS LIST, TERM ADDRESS, SUBADD COUNT(0:MAXSS), MESSAGE DATA(0:MAXMESS), MAP(-1:MAXSS, -1:MAXSS)

'COMMENT' SS NAME: Namelist array for subsystem name strings.
### DATA NAME: Namelist array for data item name strings.
### UNITS NAME: Namelist array for units name strings.
### CONF SS LIST: Array for list of configured subsystem codes.
### TERM ADDRESS: Array for terminal address values of subsystems in
### CONF SS LIST. Value of 0 means subsystem on bus
### but not set. Value of -1 means ss not on bus.
### SUBADD COUNT: Array for counts of subaddress assignments made
### for subsystem in CONF SS LIST.
### MESSAGE DATA: Array to hold codes of data items in bus messages,
### each "nm" representing one message.
### MAP: Array used for formulation of source/sink pair map.

'COMMENT' TEXT BUFFERS:
'BYTE' 'ARRAY' STRBUFF, FNAME(0:MAXSTWDS)
'BYTE' 'ARRAY' REMARKS(0:413), INBUFF, OUTBUFF(0:101)

'COMMENT' STRBUFF: Buffer for all text strings typed by user except file
### names and remarks for files.
### FNAME: Buffer for all name text.
### REMARKS: Buffer for file remarks text.
### INBUFF: Character buffer required by library input procedures.
### OUTBUFF: ditto for output procedures.

'COMMENT' ARRAY FOR COMMAND LISTING STRINGS:
'INTEGER' 'ARRAY' CMD(0:52)

'COMMENT' PRESET ARRAY DEFINING COMMANDS IN EACH STATE
'INTEGER' 'ARRAY' CNMLIST(17+0:403)=
7 1 2 0 3 4 0 5 2
40 5 6 0 7 8 0 9 0 0 1 1 1 2 0 1 3 1 4 0 1 5 0 1 6 0 1 7 0 1 8 0 1 9 0 2 0 0 2 1 0 2 2 0 2 3 0 2 4 0 2 5 0 2 6 0 2 7 0 2 8 0 2 9 0 3 0 3 1 0 3 2 0 3 3 0 3 4 0 3 5 0 3 6 0 3 7 0 3 8 0 3 9 0 4 0 0 4 1 0 4 2 0 4 3 0 4 4 0 4 5 0 4 6 0 4 7 0 4 8 0 4 9 0 5 0 0 5 1 0 5 2 0 5 3 0 5 4 0 5 5 0 5 6 0 5 7 0 5 8 0 5 9 0 6 0 0 6 1 0 6 2 0 6 3 0 6 4 0 6 5 0 6 6 0 6 7 0 6 8 0 6 9 0 7 0 0 7 1 0 7 2 0 7 3 0 7 4 0 7 5 0 7 6 0 7 7 0 7 8 0 7 9 0 8 0 0 8 1 0 8 2 0 8 3 0 8 4 0 8 5 0 8 6 0 8 7 0 8 8 0 8 9 0 9 0 0 9 1 0 9 2 0 9 3 0 9 4 0 9 5 0 9 6 0 9 7 0 9 8 0 9 9
'COMMENT'

'CMD: Set up by SET COMMAND STRINGS with pointers to all
the text strings used when displaying available commands.
CMD010 is set to a string which just gives a new line, for
convenience when interpreting the CMDLIST array.
CMDLIST: The procedure used to list the commands available
in the current state is LIST COMMANDS. It does
this with reference to the preset CMDLIST array, where
each row represents the state whose value is the row
number (eg: EMPTY = 1, OPEN = 2, etc) and references
the commands valid in that state. This array is
preset with pointers to the CMD array entries which
point to the actual strings. The first number in
each row is the number of following entries in
the row and then the commands are arranged in two;
separated by 0's, which are pointers to the new line
strings. Thus the commands are listed two to a line
on the display. For example: Row 1, which represents
the EMPTY state has 7 entries: 1 (RESP), 2 (ITE),
0 (newline), 3 (IFM), 4 (IFM), 0 (newline) and
52 (STOP)

'COMMENT' DATABASE TABLES.
RELIST holds the list of basic database entries; each defining
one data input or output requirement for a subsystem.
TXLIST is used when the system is configured to hold the list
of all transmitted data items, with their associated transmission
subsystems, rate, precision and units identifiers. RXLIST is
used to hold information on receivers of the transmitted data
held in TXLIST. For each transmitted data item RXLIST holds the
receiving subsystem codes together with their rate, precision
and units requirements. MESSAGE HEADER holds the specification
of each bus message (or direct data transfer) in the configured
system; the contents of which are held in the corresponding rows
of the MESSAGE DATA array.
This is a convenient point to say that since table entries
are referenced from index 0 the counts of entries on the various
tables (ENTRIES, TOTAL TX DATA and TOTAL MESSAGES) always point
to the next available entry. Search loops, then are always
indexed from 0 to 1 less than the count value - eg:
'FOR REFFTR := 0 'STEP' 1 'UNTIL' ENTRIES - 1 'DO'...
SAVANT always increments message numbers by 1 when interchanging with
the user, so that the message which the user knows as MESSAGE 1,
also, is actually held in entry 0 of the table. In comments
within the program we talk about the 'user message number'
and the 'actual message number'; and this is the difference.

'TABLE' RELIST03 MAXENTRIES)
(55 'UNSIGNED' (6) O 'BIT'-0)
DATA 'UNSIGNED' (9) O 'BIT'-0)
TR 'UNSIGNED' (1) O 'BIT'-15)
RATE 'UNSIGNED' (4) I 'BIT'-0)
PREC 'UNSIGNED' (10) I 'BIT'-1)
CONF 'UNSIGNED' (1) I 'BIT'-14)
UNITS 'UNSIGNED' (6) 2 'BIT'-0)
'COMMENT'
### SS: The subsystem name code in the entry.
### DATA: The data name code.
### TR: The transmit/receive bit (1: transmit; 0: receive).
### RATE: The rate group number.
### PREC: Precision in bits.
### CONF: Flag to indicate entry dealt with during configuration.
### UNITS: The units name code.

'TABLE' TXLIST[4+MAXDATA]
[TXDATA 'UNSIGNED' (9) '0`BIT'0]
[TXSS 'UNSIGNED' (6) '0`BIT'9]
[TXRATE 'UNSIGNED' (4) '1`BIT'0]
[TXPREC 'UNSIGNED' (10) '1`BIT'4]
[TXUNIT 'UNSIGNED' (6) '2`BIT'0]
[TXCHECK 'UNSIGNED' (1) '2`BIT'6]
[MRX 'UNSIGNED' (6) '3`BIT'0]
[RXSTART 'UNSIGNED' (10) '3`BIT'63]

'COMMENT'
### TXDATA: The transmitted data item.
### TXSS: Its transmitted subsystem.
### TXRATE: Its rate.
### TXPREC: Its precision.
### TXUNIT: Its units.
### TXCHECK: Flag used when checking consistency.
### MRX: The number of receivers for the data.
### RXSTART: Pointer to the first RXLIST entry for this data.

'TABLE' RXLIST[2+MAXENTRIES]
[RXSS 'UNSIGNED' (6) '0`BIT'0]
[RXRATE 'UNSIGNED' (4) '0`BIT'6]
[RXPREC 'UNSIGNED' (10) '1`BIT'10]
[RXWORDS 'UNSIGNED' (6) '1`BIT'103]

'COMMENT'
### RXSS: The receiving subsystem.
### RXRATE: Its rate requirement.
### RXPREC: Its precision requirement.
### RXUNIT: Its units requirement.
### RXWORDS: Number of words represented by RXPREC.

'TABLE' MESSAGE HEADER[3+MAXMESS]
[TX 'UNSIGNED' (6) '0`BIT'0]
[RX 'UNSIGNED' (6) '0`BIT'6]
[RB 'UNSIGNED' (4) '0`BIT'12]
[WS 'UNSIGNED' (6) '1`BIT'0]
[MC 'UNSIGNED' (5) '1`BIT'0]
[TXSA 'UNSIGNED' (5) '1`BIT'6]
[RXSA 'UNSIGNED' (5) '1`BIT'11]
[RETRY 'UNSIGNED' (4) '2`BIT'0]
[SUBSET 'UNSIGNED' (8) '2`BIT'4]
[EXCEPT 'UNSIGNED' (1) '2`BIT'12]

'COMMENT'
### TX: The source subsystem.
### RX: The sink subsystem.
'LIST'

'MACDEF.COR'

'COMMENT' -- FILE HANDLING MACROS

'DEFINE' OPEN READ FILE (L1:L2)'DEFUN(INFL,'LOCATION'(FILENAME0),L1)OPEN(INFL,'SS','RP**01,L2)'

'DEFINE' OPEN WRITE FILE (L1:L2)'DEFUN(OUTFL,'LOCATION'(FILENAME0),L1)CREATE(OUTFL,'SS','01,L2)'

'DEFINE' CLOSE READ FILE 'CLOSE(INFL)RELEASE(INFL)'

'DEFINE' CLOSE WRITE FILE 'CLOSE(OUTFL)RELEASE(OUTFL)'

'DEFINE' FILENAME VALID 'TRUE'

'DEFINE' FILE EXISTS 'TRUE'

'COMMENT' -- INPUT MACROS

'DEFINE' READ NUM(DEV) 'GETNUM(DEV,INBUF)' (** Reads a number from device - assumed type procedure)

'DEFINE' INPUT TEXT(DEV) 'READTEXT(DEV:o,STRBUFF,ITEMP)' (** Reads a text string of up to 13 characters from 'DEV' into STRBUFF)

'DEFINE' READ FILE NAME 'READTEXT(TTI:o,FILENAME,ITEMP)' (** Reads a text string of up to 7 characters from 'TTI' into FNAME)

'DEFINE' READ ONE(CHAR)

'BEGIN'

'READTEXT(TTI:o,ATEMP,ITEMP)'

CHAR=ATEMP[2]

'END' (** Reads one character from 'TTI' into CHAR)

'DEFINE' READ REMARKS(DEV) 'READTEXT(DEV:o,REMARKS,ITEMP)' (** Reads a text string of up to 80 characters from 'DEV' into REMARKS)

'COMMENT' -- OUTPUT MACROS

'DEFINE' WRITE NUM(DEV:NUM:D1&D2) 'WRITE(DEV:(NUM)2)'

'DEFINE' SEND CHAR(DEV:CH) 'WRITECH(DEV:(CH))'

'DEFINE' OUTPUT SPC(DEV:N) 'SPACES(DEV:N)'

'DEFINE' ENDLINE(DEV:N) 'NEWLINE(DEV:N)'

'END' (** Writes 'N' new lines to device)

FS 452
'DEFINE' OUTPUT TEXT(DEV,TEXT) "Writetext(DEV,TEXT)"

'DEFINE' NAME(DEV,NL,POS) "Writetext(DEV,'LOCATION'(NL[POS+0]),"

'DEFINE' NAME TAB(DEV,NL,POS,NS) "Spaces(DEV,NS,NL,[POS+0])"

'DEFINE' NAME TAB(DEV,NL,POS,NS) "Spaces(DEV,NS,NL,[POS+0])"

('## Follow up output of name with spaces to give a total

# field width of NS spaces, for tabulation purposes)

'COMMENT' STRING COMPARISON MACROS.

'COMMENT' STRING COMPARISON MACROS.

'COMMENT' STRING COMPARISON MACROS.

'# Note: STRBUFF is where all input strings, apart from file

# names and file remarks, are buffered prior to comparison)

'DEFINE' TEXT IS(STR) 'CMPSTR("LOCATION"(STRBUFF[0]),STR)=0"'

('## Delivers TRUE value if string in 'STRBUFF' = string 'STR"

'DEFINE' NAME MATCHES(NL,POS)

'CMPSTR("LOCATION"(STRBUFF[0]),"LOCATION"(NL[POS+0]))=0"

('## TRUE if string in 'STRBUFF' = name in namelist 'NL' at index 'POS"

'COMMENT' PART WORD MACROS.

'COMMENT' PART WORD MACROS.

'COMMENT' PART WORD MACROS.

'COMMENT' PART WORD MACROS.

'DEFINE' PARTITION COUNT(M+W) "BITS[0:3]MESSAGE DATA[M,W]"

'DEFINE' DATA WORD(M+W) "BITS[0:3]MESSAGE DATA[M,W]"

'LIST'
APPENDIX E2

ROOT SEGMENT

'COREL' SWAIN
'INCLUDE' 'LB1131CORIDL.CRL'
'COMMON' ('LABEL' 'ROOT')
'INCLUDE' 'LB1131PROCS.COR'
'INCLUDE' 'LB1131SAVGBL.COR'
'ENTER' 'ROOT'
'SEGMENT' 'ROOT'
'BEGIN'
'INCLUDE' 'LB1131MACDEF.COR'
'INTEGER' 'PROCEDURE' 'OPTION' ('VALUE' 'INTEGER' 'PROMPT')
'BEGIN'

'COMMENT'
--- Used whenever the user needs to respond to a question with
--- a simple yes/no answer. The question is posed - the nature
--- of the question being delivered as the PROMPT string - and
--- OPTION takes the value TRUE only if 'Y' is typed in response.
--- Called by: INTYPE
--- CLRDB
--- LISDAT
--- MAPDAT
--- SUMMES
--- SETAD
--- SETTB
--- SETMN
--- INTPN
--- INTPH
--- LISREF
--- CHMN
--- FNTXS
--- LISHDA

'INTEGER' 'REPLY'

'COMMENT'
--- REPLY: The user's reply.

ENDLINE(TT0;1)
OUTPUT TEXT(TT0; 'TYPE 'Y'")
OUTPUT TEXT(TT0; 'PROMPT') OUTPUT TEXT(TT0; "...")
READ ONE(REPLY)
ENDLINE(TT0;1)
OUTPUT TEXT(TT0; ".... OK")
ENDLINE(TT0;1)
'ANSWER' ('IF' 'REPLY'= 'LITERAL' 'Y') 'THEN' TRUE 'ELSE' FALSE
'END' 'OPTION'

'PROCEDURE' CMDWRN ('VALUE' 'INTEGER' 'TYPE')

'BEGIN'

'COMMENT'
--- Warns of invalid command and reminds of listing option.
--- If TYPE is delivered as 0, the command is completely
--- unknown whereas if TYPE is 1 then the command has been
--- recognised but the user has attempted to invoke it in
--- a state in which it is not valid. The erroneous command is
--- repeated back to the user in the warning message.

CMDWRN (LEVEL 3)
### Called by: OBEY COMMAND and all Level 2 procedures

```plaintext
ENDLINE(TTD):1)
OUTPUT TEXT(TTD:*COMMAND ')
OUTPUT TEXT(TTD:*STRING(STRBUFF))
'IF' TYPE=0 'THEN'
'BEGIN' OUTPUT TEXT(TTD:* NOT UNDERSTOOD')
'END'
'ELSE'
'BEGIN'
OUTPUT TEXT(TTD:* NOT VALID IN THIS STATE')
ENDLINE(TTD):1)
'END'
'END' CMDWRN!

'FLOATING'PROCEDURE'GETNUM('VALUE'BYTE'BYTE'BYTE'ARRAY'INBUFF))
'BEGIN'
'FLOATING' NUMBER
FREAD(DEV+NUMBER)
'ANSWER' NUMBER
'END' GETNUM!

'INTEGER'PROCEDURE'CMPSRT('VALUE'INTEGER'STRING1;STRING2))
'BEGIN'
'COMMENT' Compares STRING1 and STRING2 character by character.
Returns a value of: +1 STRING1 longer than STRING2
     -1 STRING2 longer than STRING1
      0 STRING1 and STRING2 are identical!
     'BYTE' 'ARRAY' CHAR[1:123]
     'INTEGER' LENGTH1;LENGTH2;NEXT CHAR
     LENGTH1=[STRING1]
     LENGTH2=[STRING2]
     'IF' LENGTH1>LENGTH2 'THEN' 'ANSWER' +1!
     'IF' LENGTH1<LENGTH2 'THEN' 'ANSWER' -1!
     'FOR' NEXTCHAR=1 'STEP' 1 'UNTIL' LENGTH1 'DO'
     'BEGIN'
     CHAR[1]=STRING1+NEXT CHAR[1]
     'END'
     'ANSWER' 0
     'END' CMPSRT!

PROCEDURE'SETCMD:
'BEGIN'

'COMMENT'
### Sets up pointers to the command strings in the CMD array for
### use when listing commands. Also sets up two other frequently
### used strings.
### Called by: SAVD21

CMD[0]=*

1
CMD[1]="RSP - RESET PARAMETERS"
CMD[2]="ITE - INPUT TERMINAL ENTRIES"
CMD[3]="IFE - INPUT FILED ENTRIES"
```
PROCEDURE OBEYCM

BEGIN

COMMENT

OBEYCM - OBEY COMMAND (LEVEL 1)

The command interpreter.

Calls the appropriate Level 2 procedure depending on the
input command text string. If the command is not recognised
then a notification is sent.

END SETCMD
*** Called by: MAIN PROGRAM
*** Calls                   CMDWN   LISCMD
*** All Level 2 Procedures

OBEYED:=TRUE;
'IF' TEXT IS('RSP') 'THEN' RESET
'ELSE' 'IF' TEXT IS('ITE') 'THEN' INITYE
'ELSE' 'IF' TEXT IS('IFE') 'THEN' INFILE
'ELSE' 'IF' TEXT IS('IFM') 'THEN' INFILM
'ELSE' 'IF' TEXT IS('LEM') 'THEN' LIENT
'ELSE' 'IF' TEXT IS('LSD') 'THEN' LISSSD
'ELSE' 'IF' TEXT IS('LSM') 'THEN' LISSMN
'ELSE' 'IF' TEXT IS('LDN') 'THEN' LISSDM
'ELSE' 'IF' TEXT IS('TDP') 'THEN' TRDATA
'ELSE' 'IF' TEXT IS('CSN') 'THEN' CHSSN
'ELSE' 'IF' TEXT IS('CDM') 'THEN' CHDMM
'ELSE' 'IF' TEXT IS('CRE') 'THEN' CHRATE
'ELSE' 'IF' TEXT IS('CRG') 'THEN' CHRATG
'ELSE' 'IF' TEXT IS('CDR') 'THEN' CHDART
'ELSE' 'IF' TEXT IS('CSR') 'THEN' CHSSRT
'ELSE' 'IF' TEXT IS('CPE') 'THEN' CHFCE
'ELSE' 'IF' TEXT IS('CPG') 'THEN' CHFCO
'ELSE' OBEYED:=FALSE;
'IF' OBEYED = FALSE 'THEN'

'BEGIN';
OBEYED:= TRUE;
'IF' TEXT IS('CDP') 'THEN' CHDAPR
'ELSE' 'IF' TEXT IS('CSF') 'THEN' CHSSPR
'ELSE' 'IF' TEXT IS('CUE') 'THEN' CHMICE
'ELSE' 'IF' TEXT IS('CUG') 'THEN' CHMIG
'ELSE' 'IF' TEXT IS('EDU') 'THEN' CHDAUN
'ELSE' 'IF' TEXT IS('DDR') 'THEN' DELDAR
'ELSE' 'IF' TEXT IS('DOE') 'THEN' DELIEN
'ELSE' 'IF' TEXT IS('FEN') 'THEN' FILENT
'ELSE' 'IF' TEXT IS('FSO') 'THEN' FILESD
'ELSE' 'IF' TEXT IS('CLR') 'THEN' CLRDB
'ELSE' 'IF' TEXT IS('CFS') 'THEN' CONFIG
'ELSE' 'IF' TEXT IS('LDP') 'THEN' LISDAP
'ELSE' 'IF' TEXT IS('CS') 'THEN' LISCOS
'ELSE' 'IF' TEXT IS('LDT') 'THEN' LISDAT
'ELSE' 'IF' TEXT IS('MDT') 'THEN' MAPDAT
'ELSE' 'IF' TEXT IS('SSF') 'THEN' SSSFRDA
'ELSE' 'IF' TEXT IS('CSS') 'THEN' CKCONS
'ELSE' 'IF' TEXT IS('CCR') 'THEN' CKCORR
'ELSE' OBEYED:=FALSE;
'END';
'IF' OBEYED = FALSE 'THEN'

'BEGIN';
'IF' TEXT IS('LHM') 'THEN' LISHES
'ELSE' 'IF' TEXT IS('LDD') 'THEN' LISDDA
'ELSE' 'IF' TEXT IS('LDM') 'THEN' CALLDA
'ELSE' 'IF' TEXT IS('SUM') 'THEN' SUMMES
'ELSE' 'IF' TEXT IS('LOM') 'THEN' LISIME
'ELSE' 'IF' TEXT IS('LSS') 'THEN' LISSUB
'ELSE' 'IF' TEXT IS('SRE') 'THEN' SETRET
'ELSE' 'IF' TEXT IS('STA') 'THEN' SETTAD
'ELSE' 'IF' TEXT IS('ROM') 'THEN' REOKDM
'ELSE' 'IF' TEXT IS('RIP') 'THEN' REFTMT
'ELSE' 'IF' TEXT IS('FHL') 'THEN' FILEM
'ELSE' 'IF' TEXT IS('DMS') 'THEN' DISMAY
'ELSE' 'IF' TEXT IS('FNS') 'THEN' FAMSYS


'ELSE' IF TEXT IS('LSA') THEN' LISSA
'ELSE' IF TEXT IS('FLS') THEN' FILSCH
'ELSE' IF TEXT IS('UFS') THEN' UNFSYS
'ELSE' IF TEXT IS('H') THEN' LISCHD
ELSE
'BEGIN'
'COMMENT' *** command not recognised
CMDWRN(0)
'END'
'END' OBEYCM

'PROCEDE' INIT;
'BEGIN'

'COMMENT'
*** Returns the database to the EMPTY state, initialising counts
*** and all database tables and arrays.
*** Called by: SAVD21 CLRDB
*** Calls: LISCMD CLRCFL
***
'INTEGER' PTR1, PTR2;

'COMMENT'
*** PTR1: Pointer used as index for clearing REFList and namelists.
*** PTR2: Pointer used as index to individual namelists words

CLRMEL;
CLRCFL;
'FOR' PTR1 := 0 'STEP' 1 'UNTIL' 3#MAXENTRIES - 1 'DO' REFList(PTR1) := 0;
ENTRIES := 0;
'FOR' PTR2 := 0 'STEP' 1 'UNTIL' 6 'DO'
'BEGIN'
'FOR' PTR1 := 0 'STEP' 1 'UNTIL' MAXSS 'DO'
'BEGIN'
SS NAME(PTR1, PTR2) := 0;
UNITS NAME(PTR1, PTR2) := 0
'END'
'FOR' PTR1 := 0 'STEP' 1 'UNTIL' MAXDATA 'DO' DATA NAME(PTR1, PTR2) := 0
'END'
SS NAME0; 13 := MAXSS;
SS NAME0; 23 := 0;
DATA NAME0; 13 := MAXDATA;
DATA NAME0; 23 := 11
UNITS NAME0; 13 := MAXSS;
UNITS NAME0; 23 := 2;
STATE := EMPTY;
LISCMD
'END' INIT:
'COMMENT' System Architecture Verification & Analysis Technique

SAVANT 2 - Version 3/1 of 26 JUL 81

Originally written by A A Callaway FSS Div RAE Farnborough Hants
for a PRIME S300.
This version of SAVANT has been modified by R Bluff and
P T May to run on a PDP 11/34

FS 452
TIDEF(TTO,TTI,TIERR);
ERRSET(0,0,-1+0+0,ERRSTAT);
ENDLINE(TTO+1);
OUTPUT TEXT(TTO,"WELCOME TO SAVANT 2 - VERSION 1 5/1 OF 26 JUL 81");
ENDLINE(TTO+1);
SETCMD;
INIT;
WHILE (STATE<>STOPPED) 'DO'
  'BEGIN'
    ENDLINE(TTO+2);
    OUTPUT TEXT(TTO,"COMMAND .... ");
    INPUT TEXT(TTI);
    ENDLINE(TTO+1);
    'IF' TEXT IS('STOP') 'THEN'
      'BEGIN'
        ENDLINE(TTO+1);
        OUTPUT TEXT(TTO,"HAVE YOU FILED ALL REQUIRED DATA ?");
        'IF' OPTION('IF YOU REALLY WANT TO STOP')=TRUE
          'THEN' STATE=STOPPED
      'END'
    ELSE OBEYCM
  'END';
TIERR;
'END'
'FINISH'
LIST OF USER PROCEDURES

'COMMENT'
## Please note!!! This file will not compile and is only a list
## of all the SAVANT procedures for reference purposes.

'PROCEDURE' SETCMD:
'BEGIN'

'COMMENT'
## Sets up pointers to the command strings in the CMD array, for
## use when listing commands. Also sets up two other frequently
## used strings.
## Called by: SAVD21

CMD0311='RSP - RESET PARAMETERS
CMD0221='ITE - INPUT TERMINAL ENTRIES
CMD0231='IFE - INPUT FILED ENTRIES
CMD0411='IFM - INPUT FILED MESSAGES
CMD0511='LEM - LIST ENTRIES
CMD0631='LSD - LIST SUBSYSTEM DATA
CMD0731='LSH - LIST SUBSYSTEM NAMES
CMD0831='LDM - LIST DATA NAMES
CMD0931='TDP - TRACe DATA PATH
CMD1031='CSW - CHANGE SUBSYSTEM NAME
CMD1131='CDN - CHANGE DATA NAME
CMD1231='C,R - CHANGE DAta Entry
CMD1331='CRC - CHANGE RATE GENERAL
CMD1431='CDR - CHANGE DATA RATE
CMD1531='CSR - CHANGE SUBSYSTEM RATE
CMD1631='CPF - CHANGE PREC ENTRY
CMD1731='CPG - CHANGE PREC GENERAL
CMD1831='CDP - CHANGE DATA PREC
CMD1931='CSP - CHANGE SUBSYSTEM PREC
CMD2031='CUF - CHANGE UNITS ENTRY
CMD2131='CUG - CHANGE UNITS GENERAL
CMD2231='CUD - CHANGE DATA UNITS
CMD2331='DBR - DELETE DATA REFERENCE
CMD2431='DEE - DELETE ONE ENTRY
CMD2531='DEF - FILE ENTRIES
CMD2631='FSD - FILE SUBSYSTEM DATA
CMD2731='CLR - CLEAR DATABASE
CMD2831='CFW - CONFIGURE SYSTEM
CMD2931='LDP - LIST DATA PATHS
CMD3031='LCS - LIST CONFIGURED SUBSYSTEMS
CMD3131='LDT - LIST DATA TRAFFIC
CMD3231='MBT - MAP DATA TRAFFIC
CMD3331='SSP - SOURCE SINK PAIR DATA
CMD3431='CCS - CHECK CONSISTENCY
CMD3531='CCR - CHECK CORRELATION
CMD3631='SUM - SUMMARISE MESSAGES
CMD3731='LSM - LIST BUS MESSAGES
CMD3831='LDD - LIST DIRECT DATA
CMD3931='CLD - CALCULATE LOADINGS
CMD4031='LDM - LIST ONE MESSAGE
CMD4131='LSS - LIST SUBSETS

SETCMD (LEVEL 1)
PROCEDURE OBEYCM

'BEGIN'

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.

'PROCEDURE OBEYCM

'OBEYCM - OBEY COMMAND (LEVEL 1)

'COMMENT'

The command interpreter.
'ELSE' 'IF' TEXT IS('LDP') 'THEN' LISPAP
'ELSE' 'IF' TEXT IS('LSC') 'THEN' LISCOS
'ELSE' 'IF' TEXT IS('LDT') 'THEN' LISDAT
'ELSE' 'IF' TEXT IS('MDT') 'THEN' MAPDAT
'ELSE' 'IF' TEXT IS('SSP') 'THEN' SSPRDA
'ELSE' 'IF' TEXT IS('CSS') 'THEN' CKCONS
'ELSE' 'IF' TEXT IS('CCR') 'THEN' CKCORR
'ELSE' OBEYED:=FALSE;
'END';
'IF' OBEYED = FALSE 'THEN'
'BEGIN';
'IF' TEXT IS('LM') 'THEN' LIMES
'ELSE' 'IF' TEXT IS('LDD') 'THEN' LISDBA
'ELSE' 'IF' TEXT IS('LDL') 'THEN' CALLDA
'ELSE' 'IF' TEXT IS('LDU') 'THEN' SUMMES
'ELSE' 'IF' TEXT IS('LIM') 'THEN' LISIME
'ELSE' 'IF' TEXT IS('LSS') 'THEN' LISSUB
'ELSE' 'IF' TEXT IS('SRC') 'THEN' SETRET
'ELSE' 'IF' TEXT IS('STA') 'THEN' SETTAD
'ELSE' 'IF' TEXT IS('ROM') 'THEN' REORDM
'ELSE' 'IF' TEXT IS('RFS') 'THEN' REFMT
'ELSE' 'IF' TEXT IS('MLM') 'THEN' FILMES
'ELSE' 'IF' TEXT IS('DNS') 'THEN' DISMAN
'ELSE' 'IF' TEXT IS('FMS') 'THEN' FRMSYS
'ELSE' 'IF' TEXT IS('LSA') 'THEN' LISSAD
'ELSE' 'IF' TEXT IS('FLS') 'THEN' FLSCH
'ELSE' 'IF' TEXT IS('UFS') 'THEN' UHFSSY
'ELSE' 'IF' TEXT IS('H') 'THEN' LISHD
'ELSE'
'BEGIN';
'COMMENT' '### command not recognised:
CMWARN(0)
'END';
'END';
'END' OBEYCH;

'PROCEDURE' RESET;
'BEGIN';

'COMMENT';
'### SAVANT Command.
'### This procedure allows the reset system parameters to be reset.
'### Each is offered in turn, and the new value enterd if IVAR
'### or RVAL are returned non-negative.
'### Called by: OBEY COMMAND
'### Calls: CMWARN INTCH;
'### FLOCH;

'IF' STATE <> EMPTY
'THEN' CMWARN (1)
'ELSE'
'BEGIN';
'INTEGER' IVAL;
'FLOATING' FVAL;

'COMMENT';
'### IVAR: integer value returned from change procedure.
'### FVAL: floating value returned from change procedure
INTCH ('WORD LENGTH', 'BITS', 'WORD LENGTH', 0, IVAR);
'IF' IVAL = 0 'THEN' 'WORD LENGTH := IVAL

FS 452
INTPC('MESSAGE LENGTH', 'WORDS', MESSAGE LENGTH, 63,IVAL);  
'IF' IVAL >= 0 'THEN' MESSAGE LENGTH := IVAL;  
INTPC('NUMBER OF RATE GROUPS', '', LOWEST RATE, 15,IVAL);  
'IF' IVAL >= 0 'THEN' LOWEST RATE := IVAL;  
INTPC('WORD OVERHEAD (SYNC + PARITY)', 'BITS', WORD OVERHEAD, 0,IVAL);  
'IF' IVAL >= 0 'THEN' WORD OVERHEAD := IVAL;  
INTPC('TRANSMISSION BIT RATE', 'KBITS/SEC', BIT RATE, 0,IVAL);  
'IF' IVAL >= 0 'THEN' BIT RATE := IVAL;  
INTPC('MAX NUMBER OF DATA BUS TERMINALS', '', TERMINAL LIMIT, 0,IVAL);  
'IF' IVAL >= 0 'THEN' TERMINAL LIMIT := IVAL;  
FLOPCH('CONTROLLER/Terminal OVERHEAD', CONT OVERHEAD, FVAL);  
'IF' FVAL >= 0 'THEN' CONT OVERHEAD := FVAL;  
FLOPCH('Terminal/Terminal OVERHEAD', TERM OVERHEAD, FVAL);  
'IF' FVAL >= 0 'THEN' TERMINAL OVERHEAD := FVAL;  
'END'  
'END' RESET;  

PROCEDURE INTTYE;  
'BEGIN'  

'COMMENT'  
*** SAVANT Command.  
*** Causes a number of entries related to a specific subsystem  
*** or to a number of miscellaneous subsystems, to be added to the  
*** REPLIST from the terminal, with prompts to assist the user.  
*** Can also be used to create a REPLIST starting from the EMPTY  
*** state. For dedicated entries, the subsystem name is read  
*** first. This is, therefore, buffered in STRBUFF. The input  
*** procedure is then called twice - once for transmitted entries  
*** and once for received entries. For miscellaneous entries  
*** the input procedure is only called once.  
*** Called by: OBEY COMMAND  
*** Calls: CMDWRN !INDAT  
*** OPTION  

'IF' STATE <> EMPTY  
'AND' STATE <> OPENX  
'THEN' CMDWRN (1)  
'ELSE'  
'BEGIN'  
'INTEGER' ERROR, SSCODE;  

'COMMENT'  
*** ERROR: Flag to indicate error in reading.  
*** SSCODE: Subsystem name code for dedicated entries - set  
*** in READEN  
SSCODE := 0I (### to indicate name not recorded)  
'IF' OPTION ('IF ALL DATA APPERTAINS TO ONE SUBSYSTEM') = TRUE 'THEN'  
'BEGIN'  
'COMMENT'  
### dedicated entries  
ENDLINE(TTD, ** SUBSYSTEM NAME **')  
WRITE TEXT (TTD, * SUBSYSTEM NAME *...*')  
INPUT TEXT (TTI);  
INDAT (TTI, 1, SSCODE, ERROR);  
### read tx entries  
'IF' ERROR = FALSE 'THEN' INDAT (TTI, 0, SSCODE, ERROR);  
### read rx entries  
'END'  
'ELSE' INDAT (TTI, -1, SSCODE, ERROR);  
### miscellaneous data  
'IF' ERROR = FALSE 'THEN'  
'BEGIN'
'BEGIN'

'COMMENT'
### SAVANT Command.
### Inputs previously filed messages data and changes database state
### to LIMITED.
### Called by: OBEY COMMAND
### Calls: CMDURN INNAM

* FINIL

'IF' STATE <> EMPTY
'THEN' CMDURN (1)
'ELSE'

'BEGIN'

'INTEGER' CSSPTR, MPTR, MDPTR, LENGTH, ERROR

'COMMENT'
### CSSPTR: Pointer to configured subsystem entries.
### MPTR: Pointer to message entries.
### MDPTR: Pointer to data words in message data array.
### LENGTH: Buffer for message length.
### ERROR: Error flag for FINIL

FINIL (*MESSAGE*, 0, ERROR);
'IF' ERROR = FALSE 'THEN'

'BEGIN'

INPUT TEXT (INFIL);
'IF' TEXT IS ('MESSAGE FILE') 'THEN'

'BEGIN'

'COMMENT' *** first input namelists

INNAM (SS NAME)

INNAM (DATA NAME)

'COMMENT' *** now configured subsystem arrays

CONF SS COUNT := READ NUM (INFIL)

'FOR' CSSPTR := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'

'BEGIN'

CONF SS LIST(CSSPTR) := READ NUM (INFIL)

TERM ADDRESS(CSSPTR) := READ NUM (INFIL)

SUBADD COUNT(CSSPTR) := 0

'END';

'COMMENT' *** now message structure

TOTAL MESSAGES := READ NUM (INFIL)

'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'

'BEGIN'

TXINMPTR := READ NUM (INFIL)
RXINMPTR := READ NUM (INFIL)
LENGTH := READ NUM (INFIL)
RGMPTR := LENGTH
RETRY(RXINMPTR) := READ NUM (INFIL)
LENGTH := READ NUM (INFIL)
WDISPMPTR := LENGTH

'RETRACT' MDPTR := 1 'STEP' 1 'UNTIL' LENGTH 'DO'

MESSAGE DATA(MPTR, MDPTR) := READ NUM (INFIL)

'END'

CLOSE READ FILE;
STATE := LIMITED;
ENDLINE(ITIO); WRITE TEXT (ITIO, * MESSAGES READ *);
ENDLINE(ITIO);

'END'

'ELSE'

'BEGIN'
'PROCEDURE' LISSEN;
'BEGIN'

'COMMENT'
*** SAVANT Command.
*** Causes the complete RELIST to be listed.
*** Called by: OBEY Command
*** Calls: CMDWRN

'IF' STATE <> OPENX
'AND' STATE <> CONFIGURED
'AND' STATE <> FORMED
'AND' STATE <> INCONSISTENT
'THEN' CMDWRN (1)
'ELSE' LISREF (0)
'END' LISSEN;

'PROCEDURE' LISSSD;
'BEGIN'

'COMMENT'
*** SAVANT Command.
*** Causes all RELIST entries relating to a particular subsystem
*** to be listed.
*** Called by: OBEY Command
*** Calls: CMDWRN
*** LOCNAME

'IF' STATE <> OPENX
'AND' STATE <> CONFIGURED
'AND' STATE <> FORMED
'AND' STATE <> INCONSISTENT
'THEN' CMDWRN (1)
'ELSE'
'BEGIN'
'INTEGER' SSCORE;

'COMMENT'
*** SSCORE: The subsystem name code

LOCNAME ('*SUBSYSTEM', SS NAME, SSCORE);
'IF' SSCORE > 0 'THEN' LISREF (SSCODE)

'END'
'END' LISSSD;

'PROCEDURE' LISSSN;
'BEGIN'

'COMMENT'
*** SAVANT Command.

LISSEN (LEVEL 2)
LISSSD (LEVEL 2)
LISSSN (LEVEL 2)
### Causes all subsystem names in the SS NAME list to be listed.
### Since all possible names can be contained on one vdu page,
### the option to pause is not offered and is delivered as FALSE
### to the listing procedure.
### Called by: OBREY COMMAND
### Calls: CMDWRN   LISNAM

'IF' STATE = EMPTY
'THEN' CMDWRN (1)
'ELSE' LISNAM (SS NAME, FALSE)
'END' LISSNH

'PROCEDURE' LISDNN;
'BEGIN'

'COMMENT'
### SAVANT Command.
### Causes all data names in the DATA NAME list to be listed. Since
### the possible number of names could cause the list to exceed one
### page, the user is offered the pause option which is passed on
### to the listing procedure.
### Called by: OBREY COMMAND
### Calls: CMDWRN   LISNAM
### OPTION;

'IF' STATE = EMPTY
'THEN' CMDWRN (1)
'ELSE'
'BEGIN'
'INTEGER' PAUSE;

'COMMENT'
### PAUSE: Flag to indicate response to pause option

PAUSE = OPTION (EACH PAGE)
LISNAM (DATA NAME, PAUSE)
'END'
'END' LISDNN;

'PROCEDURE' TRDATA;
'BEGIN'

'COMMENT'
### SAVANT Command.
### Causes all REFLIST references to a specified data item to be listed,
### giving the associated subsystem name, whether the item is
### transmitted or received by the subsystem, and the rate, precision
### and units for each reference.
### Called by: OBREY COMMAND
### Calls: CMDWRN   LOCNAM
### ACTFRC;

'IF' STATE<>OPENX
'AND' STATE<>CONFIGURED
'AND' STATE<>FORMED
'AND' STATE< INCONSISTENT
'THEN' CMDWRN (1)
'ELSE'
'BEGIN'
'INTEGER' DACODE, REFPTR;
'COMMENT
### DACODE: The specific data name code.
### REFPTR: Pointer to the REFLIST entries!

LOCNAME('DATA'+DATA NAME+DACODE);
'IF' DACODE>0 'THEN'
'BEGIN'
ENDLINE(TTD+1); OUTPUT TEXT(TTD)^ RATE PREC UNITS'^'
ENDLINE(TTD+1); OUTPUT TEXT(TTD)^'
'COMMENT' ### data name located - search through REFLIST:
'FOR' REFPTR=0 'STEP' 1 'UNTIL' ENTRIES-1 'DO'
'IF' DATA[REFPTR]=DACODE 'THEN'
'BEGIN'
'COMMENT' ### relevant entry located - list details
ENDLINE(TTD+1); NAME(TTD+SS NAME+SSREFPTR));
'IF' TREFPTR=0 'THEN' OUTPUT TEXT(TTD+*RECEIVES *) 'ELSE' OUTPUT TEXT(TTD+*TRANSmits*)
OUTPUT SPC(TTD+2); WRITE NUM(TTD+RATE[REFPTR+2]+0)); OUTPUT SPC(TTD+3);
WRITE NUM(TTD+ACTPRC(PRECEFFPTR)+4+0)); OUTPUT SPC(TTD+7);
NAME(TTD+UNITs NAME+UNITsREFPTR))
'END'
'END'
'END' TRDATA;

'PROCEDURE' CHSSN;
'BEGIN'

### SAVANT Command.
### Causes a subsystem to have its name changed.
### Called by: OBREAL COMMAND
### Calls: CMDNAM CHNNAM

'IF' STATE <> OPENX 'THEN' CMDNAM (1)
'ELSE' CHNNAM (SS NAME)
'END' CHSSN;

'PROCEDURE' CHNNAM;
'BEGIN'

### SAVANT Command.
### Causes a data item to have its name changed.
### Called by: OBREAL COMMAND
### Calls: CMDNAM CHNNAM

'IF' STATE <> OPENX 'THEN' CMDNAM (1)
'ELSE' CHNNAM (DATA NAME)
'END' CHNNAM;

FS 452
'PROCEDURE' CHRATE
'BEGIN'

'COMMENT'
### SAVANT Command.
### Causes rate value of particular REFLIST entry to be changed.
### Called by: OBEY COMMAND
### Calls: CMDWRN CHNUME)

'IF' STATE <> OPENX
'THEN' CMDWRN (1)
'ELSE' CHNUME (0)
'END' CHRATE;

'PROCEDURE' CHRTAG
'BEGIN'

'COMMENT'
### SAVANT Command.
### Causes all rates of a particular value to be changed
to another value throughout the REFLIST.
### Called by: OBEY COMMAND
### Calls: CMDWRN GENNCH

'IF' STATE <> OPENX
'THEN' CMDWRN (1)
'ELSE' GENNCH (0)
'END' CHRTAG;

'PROCEDURE' CHDART
'BEGIN'

'COMMENT'
### SAVANT Command.
### Causes the rate value associated with a particular data item
to be set up at each occurrence of the item on the REFLIST.
### Called by: OBEY COMMAND
### Calls: CMDWRN CHDAVI

'IF' STATE <> OPENX
'THEN' CMDWRN (1)
'ELSE' CHDAVI (0)
'END' CHDART;

'PROCEDURE' CHSSRT
'BEGIN'

'COMMENT'
### SAVANT Command.
### Causes all rates of a particular value to be changed to
### another value, but only for the REFLIST entries relating to
### a particular subsystem.
### Called by: OBEY COMMAND
### Calls: CMDWRN CHSSV

'IF' STATE <> OPENX
'THEN' CMDWRN (1)
'ELSE' CHSSV (0)
PROCEDURE CHPRCE
BEGIN
COMMENT
*** SAVANT Command.
*** Causes precision value of particular REFLIST entry to be changed.
*** Called by: OBEY COMMAND
*** Calls: CHDWRN CHNUNE

IF STATE <> OPENX
THEN CHDWRN (1)
ELSE CHNUNE (1)
END CHPRCE

PROCEDURE CHPRCG
BEGIN
COMMENT
*** SAVANT Command.
*** Causes all precisions of a particular value to be changed
*** to another value throughout the REFLIST.
*** Called by: OBEY COMMAND
*** Calls: CHDWRN CHNUNCG

IF STATE <> OPENX
THEN CHDWRN (1)
ELSE CHNUNCG (1)
END CHPRCG

PROCEDURE CHDAPR
BEGIN
COMMENT
*** SAVANT Command.
*** Causes the precision value associated with a particular data item
*** to be set up at each occurrence of the item on the REFLIST.
*** Called by: OBEY COMMAND
*** Calls: CHDWRN CHDAV

IF STATE <> OPENX
THEN CHDWRN (1)
ELSE CHDAV (1)
END CHDAPR

PROCEDURE CHSSPR
BEGIN
COMMENT
*** SAVANT Command.
*** Causes all precisions of a particular value to be changed to
*** another value; but only for the REFLIST entries relating to
*** a particular subsystem.
*** Called by: OBEY COMMAND
*** Calls: CHDWRN CHSSV

IF STATE <> OPENX
THEN CHDWRN (1)
'ELSE' CHSSV (1)
'END' CHSSPRI

PROCEDURE' CHUNIE
'BEGIN'

'COMMENT'
**** SAVANT Command.
**** Changes the units identifier of a specific REFLIST entry, checking
**** whether the change has caused the old name to disappear.
**** Called by: OBEY COMMAND
**** Calls: CMDWRN FINENT
**** CKREF RECNAM

'IF' STATE<>OPENX
'THEN' CMDWRN (1)
'ELSE'
'BEGIN'

'INTEGER' LPOS, UNCODE, ERR, DUM1, DUM2;

'COMMENT'
LPOS: Pointer to the REFList entry.
UNCODE: The old units name code in the entry.
ERR: Flag to indicate UNITS NAME list full.
DUM1: Unused name code.
DUM2: ditto;

FINENT(LPOS,DUM1,DUM2);
'IF' LPOS=0 'THEN'
'BEGIN'

'COMMENT' *** entry located - save old units name and replace;
UNCODE=UNITSLPOS);
ENLNE(TTD+1);
OUTPUT TEXT(TTD,*NEW UNITS NAME .... ');
INPUT TEXT(TTTI);
RECNAM(UNITSLNAME,LPOS,DUM1,ERR);
'IF' ERR=FALSE 'THEN'
'BEGIN'

'COMMENT' *** check REFList for old name:
CKREF(UNITSLNAME,UNCODE);
ENLNE(TTD+1);
OUTPUT TEXT(TTD,*NAME CHANGED ');
ENLNE(TTD+1);
'END'

'END'

'END' CHUNIE

PROCEDURE' CHUNIG
'BEGIN'

'COMMENT'
**** SAVANT Command.
**** Invokes the general change of name of a units identifier.
**** Called by: OBEY COMMAND
**** Calls: CMDWRN CHNAM

'IF' STATE <> OPENX
'THEN' CMDWRN (1)
'ELSE' CHNAM (UNITSLNAME)
'END' CHUNIG
'PROCEDURE' CHDAUN;
'BEGIN' CHDAUN (LEVEL 2)

'COMMENT'
'*** SAVANT Command.
'*** Sets up the units identifier associated with a specific data
'*** item to the specified name at every occurrence of the data
'*** item in the REFLIST, whatever units identifiers may have
'*** been associated before. At each setting, the old units
'*** name must be checked to see if it has now disappeared from
'*** the REFLIST.
'*** Called by: OBRY COMMAND
'*** C: ls!
'*** CMDWN CKREF
'*** RECNAM LOCNAM

'IF' STATE<>OPENX
' THEN' CMDWN(1)
' ELSE'
'BEGIN'
'  ' ' INTEGER' DACODE, REFCTR, UNCODE, ERR, DUMMY)

'COMMENT'
'*** DACODE: The specific data name code.
'*** REFCTR: Pointer to the REFLIST entries.
'*** UNCODE: The old units name code in each entry.
'*** ERR: Flag to indicate UNITS NAME list is full.
'*** DUMMY: Value of new units name code (not needed)

LOCNAME("DATA",DATA NAME,DACODE))
' IF' DACODE=0 ' THEN'
'BEGIN'
'  ' ' COMMENT' *** data name code located - set required units name
ENDLINE(TTO-1))
OUTPUT TEXT(TTO,"NEW UNITS NAME . . . .",)
INPUT TEXT(TT))
ERR:=false;
REFCTR:=0;
WHILE(REFCTR<ENTRIES 'AND' ERR=False) ' DO'
'BEGIN'
'  ' ' COMMENT' *** work through REFLIST recording the new units name
'  ' *** code and checking that replaced at each occurrence
'  ' *** of the data item
'  ' ' IF' DATAREFCTR=DACODE ' THEN'
'BEGIN'
'  ' ' COMMENT' *** entry refers to the data item
UNCODE:=UNITS(REFCTR));
RECNAM(UNITS NAME=REFCTR,DUMMY,ERR))
CKREF(UNITS NAME=UNCODE)
'END'
INCREFCTR)
'END';
' IF' ERR=False ' THEN'
'BEGIN'ENDLINE(TTO-1))OUTPUT TEXT(TTO,"MOD COMPLETE");
ENDLINE(TTO-1)
'END'
'END'
'END' CHDAUN;
'PROCEDURE' DELDAR;
'BEGIN'

'COMMENT'
### SAVANT Command

Deletes all REFLIST entries relating to a particular data item.

closing up the REFLIST to overwrite. For each entry thus removed,

the procedure also checks whether this has removed the last

reference to a subsystem or units name and takes appropriate action.

Also removes the data name from the DATA NAME name list and REPOINS

the REFLIST DATA entries to take account of this.

Called by: OBEY COMMAND

Calls: CMDWRN, DELENT

### COMMENT

REFFTR: Pointer to the REFLIST entries.

SSCODE: The subsystem name code in an entry.

UCODE: The units name code to be deleted.

LOCNAM('DATA'='DATA NAME=DATA CODE')

IF 'DATA CODE'='DATA CODE'

BEGIN

'COMMENTS' data name code located - find all REFLIST references

REFFTR=0

WHILE (REFFTR<ENTRIES) 'DO'

'IF' 'DATA CODE=DATA CODE' 'THEN'

'BEGIN'

'COMENTS' entry refers to data - delete it and check other names

SSCODE=SSREFFTR)

DELENT(REFFTR)

CRREF('SS NAME',SSCDE)

'END'

'ELSE' INC(REFFTR)

DELMAN ('DATA NAME='DATA CODE')

REPOINS('DATA CODE')

ENDLINE(TTD,1)

OUTPUT TEXT(TTD,'DATA DELETED')

ENDLINE(TTD,1)

'END'

'END' DELMAN

'PROCEDURE' DELIEN

'BEGIN'

'COMENTS'

SAVANT Command.

Removes one specific entry from the REFLIST, and checks whether

this action has caused the appearance from the database

of any of the names in the entry.

Called by: OBEY COMMAND

Calls: CMDWRN, DELENT

'IF' STATE<OPENX

'THEN' CMDWRN(1)
'ELSE'
'BEGIN'
'INTEGER' SS_CODE, DA_CODE, UN_CODE, LPOS;

'COMMENT'
### SS_CODE: The subsystem name code in the entry.
### DA_CODE: The data name code in the entry.
### UN_CODE: The units name code in the entry.
### LPOS: Pointer to the entry.
FINENT(LPOS, SS_CODE, DA_CODE);
'IF' LPOS<>0 'THEN'
'BEGIN'
'COMMENT' ### entry exists - delete it
UNICODE=UNIT$[LPOS];
DELENT(LPOS);
CRREF$[SS NAME, SS_CODE];
CRREF$[DATA NAME, DA_CODE];
CRREF$[UNITS NAME, UN_CODE];
ENDLINE(TTO+1);
OUTPUT TEXT(TTO, "ENTRY DELETED");
ENDLINE(TTO+1);
'END'
'END' DELIEN;

'PROCEDURE' FILET;
'BEGIN'

'COMMENT'
### SAVANT Command.
### Causes the complete REFLIST to be sent to a disc file.
### Called by: OBEY COMMAND
### Calls: CMDRN

FILET (LEVEL 2)

'IF' STATE<OPENX
'AND' STATE<CONFIGURED
'AND' STATE<FORMED
'AND' STATE<INCONSISTENT
' THEN' CMDRN(1)
'ELSE'
'BEGIN'
FILE REF(0);
ENDLINE(TTO+1);
OUTPUT TEXT(TTO, "REFLIST FILED");
ENDLINE(TTO+1);
'END'
'END' FILET;

'PROCEDURE' FILES;
'BEGIN'

'COMMENT'
### SAVANT Command.
### Causes all REFLIST entries relating to an individual
### subsystem to be sent to a disc file.
### Called by: OBEY COMMAND
### Calls: CMDRN
### Calls: LOCAMRN

FILES (LEVEL 2)

FS 452
'IF' STATE=OPENX
'AND' STATE=CONFIGURED
'AND' STATE=FORMED
'AND' STATE=INCONSISTENT
'THEN' CMDWN(1)
'ELSE'
'BEGIN'
'INTEGER' SSCODE;
'COMMENT'
### SSCODE: The subsystem name code;
LOCNAM('SUBSYSTEM',SS_NAME,SSCODE);
'IF' SSCODE=0 'THEN'
'BEGIN'
FREF(SSCODE);
ENDLINE(TTO+1);
OUTPUT TEXT(TTO,"DATA FILED");
ENDLINE(TTO+1);
'END'
'END' FILES;

'PROCEDURE' CLRDB;
'BEGIN'

'COMMENT'
### SAVANT Command.
### Returns database state from open or LIMITED to EMPTY,
### Initialises the database structures. The user is requested
### to confirm the intention to clear.
### Called by: OBEX COMMAND
### Calls: CMDWN INIT OPTION

'IF' STATE <> OPENX
'AND' STATE <> LIMITED
'THEN' CMDWN (1)
'ELSE' IF OPTION ("IF YOU REALLY WISH TO CLEAR") = TRUE 'THEN' INIT
'ELSE' DO NOTHING
'END'
'PROCEDURE' CONFIG;
'BEGIN'

'COMMENT'
### SAVANT Command.
### Configures a system from the basic REFLIST by identifying
### the required subsystems, then the appropriate data items and
### their valid receivers, then forms the message structure.
### If fatal inconsistencies exist the database state is set
### to the inconsistent state instead of the configured state.
### Called by: OBEX COMMAND
### Calls: CMDWN FRMTXS
### FRMTXID FRMNXS
### FATLCK FRMMS
### FINBSS PARTITION NUMBER

'IF' STATE=OPENX
'THEN' CMDWN(1)
'ELSE'
'BEGIN'
'INTEGER' FATAL;
'COMMENT'
### FATAL: Flag to indicate fatal inconsistencies!

FATAL=FALSE
FRMTXS:
'IF' CONF SS COUNT>0 'THEN'
'BEGIN'
FRMTXD1
FRMXXS:
ENDLINE(TTO;1)
OUTPUT TEXT(TTO,'SYSTEM CONFIGURED')
ENDLINE(TTO;1)
FATLCK(FATAL)
'IF' FATAL=TRUE 'THEN'
'BEGIN'
FRMXXS:
FINBS(FATAL);
'IF' FATAL=FALSE 'THEN' PARNUM
'END';
'IF' FATAL=TRUE 'THEN' BEGIN ENDLINE(TTO;2) OUTPUT TEXT(TTO," NO "); 'END';
ELSE ENDLINE(TTO;1)
OUTPUT TEXT(TTO,'MESSAGES FORMED')
ENDLINE(TTO;1)
STATE="IF' FATAL=TRUE 'THEN' INCONSISTENT 'ELSE' CONFIGURED
'END';
'END' CONFIG)

'PROCEDURE' LISDAP;
'BEGIN'

'COMMENT'
### SAVANT Command.
### For the configured subsystem, this procedure lists all transmitted
### data items. Each item on the list comprises the data name,
### together with its transmitting subsystem and the associated
### precision and rate capabilities, followed by a list of receiving
### subsystems, together with their precision and rate requirements.
### The option is given to pause on each page, but the pause actually
### occurs before listing the next item which would cause the pause
### length to be exceeded.
### Called by: OBEY COMMAND
### Calls: CMDWRN OPTION
### ACTFRC1

'IF' STATE<>CONFIGURED
'AND' STATE<>FORMED
'AND' STATE<>INCONSISTENT
'THEN' CMDWRN(1)
'ELSE'
'BEGIN'
'INTEGER' TXPTR, DATARX, RXPTR, PAUSE, LINES, STOP
'COMMENT'
### TXPTR: Pointer to TXLIST entries.
### DATARX: Index to current receiver of data item being examined.
### RXPTR: Pointer to RXLIST entries.
### PAUSE: Flag to indicate response to pause option.
### LINES: Line count.
### STOP: Flag to indicate response to continue option

LISDAP (LEVEL 2)
LINES:=1;
STOP:=FALSE;
PAUSE:=OPTION(EACH PAGE);
'FOR' TXPTR:=0 'STEP' 1 'UNTIL' TOTAL TX DATA-1 'DO'
'BEGIN'
'COMMENT' ### work through tx data items;
'IF' PAUSE=TRUE 'AND' LINES+NRXITXPTR]+1>24 'THEN'
'BEGIN'
'COMMENT' ### check continue option if pausing and page full;
LINES:=1;
ENDLINE(TTO+1);
'IF' OPTION(CONTINUE)=FALSE 'THEN'
'BEGIN'
'COMMENT' ### discontinuing;
TXPTR:=TOTAL TX DATA;
STOP:=TRUE
'END'
'END'!
'IF' STOP=FALSE 'THEN'
'BEGIN'
'COMMENT' ### output transmitting info;
ENDLINE(TTO+1));
NAME(TTO,DATA NAME,TXDATA[TXPTR]);
NAME TAB(TTO,DATA NAME,TXDATA[TXPTR]+15)
OUTPUT TEXT(TTO,'FROM
'));
NAME(TTO,SS NAME,TXSS[TXPTR]);
NAME TAB(TTO,SS NAME,TXSS[TXPTR]+24)
WRITE NUM(TTO,ACTPRC(TAPREXTXPTR)+4*0)
OUTPUT TEXT(TTO,'* BITS AT RATE
');
WRITE NUM(TTO,XTRATE[TXPTR]+2*0)
'IF' NIXITXPTR=0 'THEN' 'BEGIN' ENDLINE(TTO+1)) OUTPUT TEXT(TTO,'NO RECEIVER')
'END'
'ELSE'
'FOR' DATARX:=0 'STEP' 1 'UNTIL' NRXITXPTR]=1 'DO'
'BEGIN'
'COMMENT' ### output info on receivers!
RXPTR:=RXSTART[TXPTR]+DATARX
ENDLINE(TTO+1));
OUTPUT SPC(TTO+15)
OUTPUT TEXT(TTO,'TO
'));
NAME(TTO,SS NAME,RXSS[RXPTR]);
NAME TAB(TTO,SS NAME,RXSS[RXPTR]+15)
OUTPUT TEXT(TTO,'* REQUIRES
');
WRITE NUM(TTO,ACTPRC(RXPRERE[RXPTR]+4*0)
OUTPUT TEXT(TTO,'* BITS AT RATE
');
WRITE NUM(TTO,RRXRATE[RXPTR]+2*0)
'END'
'END'
LINES:=LINES+NRXITXPTR]+1
'END'
'END'
'END' LIDAP!

'PROCEDURE' LISCOS;
'BEGIN'
'COMMENT'
### SAVANT Command.
### Produces a list of all configured subsystems tabulated in
### three columns. Each name is preceded by a prefix as follows.
### If the subsystem is not on the bus then the prefix is 'nob'.

LISCOS (LEVEL 2)
### If the subsystem is on the bus but its terminal address has not
### been set then the prefix is 'uns'; otherwise the prefix is the
### terminal address.
### Called by: OBEY COMMAND
### Calls: CMDWRN

'IF' STATE CONFIGURED
'AND' STATE LIMITED
'AND' STATE FORMED
'AND' STATE LIMITED FORMED
'AND' STATE INCONSISTENT
'THEN' CMDWRN(1)
'ELSE'
'BEGIN'
'INTEGER' CSSPTR, COLUMN

'COMMENT'
### CSSPTR: Pointer to configured subsystem entries.
### COLUMN: Column count
COLUMN=3;
'FOR' CSSPTR=1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
'BEGIN'
'COMMENT' ### work through configured subsystems
INC(COLUMN);
'IF' COLUMN=3 'THEN'
'BEGIN'
'COMMENT' ### start in first column
COLUMN=1;
ENDLINE(TTO+1)
'END';
'COMMENT' ### send prefix followed by name, and tab next column
OUTPUT TEXT(TTO+1);
'IF' TERM ADDRESSICCSSPTR>0 'THEN' OUTPUT TEXT(TTO+1)
'ELSE' IF TERM ADDRESSICCSSPTR=0 'THEN' OUTPUT TEXT(TTO+1)
'ELSE' WRITE NUM(TTO+1)
OUTPUT TEXT(TTO+1)
NAME(TTO+1)
'IF' COLUMN=3 'AND' CSSPTR<CONF SS COUNT 'THEN'
NAME TAB(TTO+1)
'COMMENT' ### send count of configured subsystems
HDLINE(TTO+1)
WRITE NUM(TTO+1)
OUTPUT TEXT(TTO
'CONFIGURED SUBSYSTEMS')
ENDLINE(TTO+1)
'END';
'END' LISCOS;

'PROCEDURE' LISDAT;
'BEGIN'

'COMMENT'
### SAVANT Command.
### This procedure produces a list of all source/sink pairs in
### the configured system, together with details of the data
### passing between each. The user is offered the option to
### pause after each s/s pair listed.
### Called by: OBEY COMMAND
### Calls: CMDWRN LISSSP
### OPTION'

LISDAT (LEVEL 2)
"IF" STATE <> CONFIGURED
"AND" STATE <> FORMED
"THEN" CMDWRN (1)
"ELSE"
"BEGIN"
"INTEGER" TXCODE, RXCODE, PAUSE, DONE:

"COMMENT"
### TXCODE: Pointer to source on CONF SS LIST.
### RXCODE: Pointer to sink on CONF SS LIST.
### PAUSE: Flag to indicate response to pause option.
### DONE: Flag to indicate traffic found for s/s pair.
PAUSE = 1: OPTION ('TO PAUSE AFTER EACH SOURCE/SINK PAIR')
'FOR' TXCODE = 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
'FOR' RXCODE = 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
'IF' TXCODE <> RXCODE 'THEN'
"BEGIN"
"COMMENT" *** work through source/sink pairs.
LISSP (TXCODE, RXCODE, DONE)
"IF" PAUSE = TRUE 'AND' DONE = TRUE 'THEN'
"BEGIN"
"COMMENT" *** check continue OPTION if pausing and output has occurred.
"IF" OPTION (CONTINUE) = FALSE 'THEN'
"BEGIN"
"COMMENT" *** discontinuing - set pointers to escape:
TXCODE := CONF SS COUNT
RXCODE := CONF SS COUNT
"END"
"END"
"END" LISDAT

"PROCEDURE" MAPDAT:
"BEGIN"

"COMMENT"
### SAVANT Command.
### This command displays a visual map of data traffic between source/sink pairs.
### The map is formed as a matrix, using the MAP array, which is
### first primed by SETMAP. Taking each row as representing a
### configured subsystem acting as source and each column as
### representing a configured subsystem acting as sink, the message
### list is scanned and an appropriate letter character placed
### at the appropriate element of the matrix to indicate the
### type of traffic: B - bus traffic only; D - direct traffic
### only and C - both direct and bus traffic. When it comes
### to deciding on a C the direct transfers will have been determined
### first since they are earlier in the message list. Thus, if
### a bus transfer is discovered for a particular source/sink pair
### then the appropriate element can be examined. If it already
### contains a B, then a C is inserted; otherwise a B is inserted.
### To output the map, the user is given an option. It can either
### be displayed on the terminal or it can be sent to a disk file
### for future listings. The latter alternative is attractive if
### there are many subsystems (e.g., 20) since the map would then be
### too large to display on a terminal screen. During output, the
### corresponding subsystem name is listed for each row so that
### the numbers labelling the columns can be identified.
**Called by:** OBEY COMMAND
**Calls:** CNDWRN SETMAP
**FILFIL** OPTION
**CLISTR**

'IF' STATE <> CONFIGURED
'AND' STATE <> FORMED
'AND' STATE <> LIMITED
'AND' STATE <> LIMITED FORMED
'THEN' CNDWRN (1)
'ELSE'
'BEGIN'
'INTEGER' TXCODE, RXCODE, MPTR, DEST, ERROR

'COMMENT'
**TXCODE** Pointer to source on CONF SS LIST.
**RXCODE** Pointer to sink on CONF SS LIST.
**MPTR** Pointer to message entries.
**DEST** Destination device.
**ERROR** Dummy error flag for FILFIL.

SETMAP

'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES = 1 'DO'
'BEGIN'
'COMMENT' **work through message list taking each source/sink**
**pair as encountered**
TXCODE := CLISTR (TXMPTR));
RXCODE := CLISTR (RXMPTR));
'IF' RGEMPTR <> 0 'THEN'
'BEGIN'
'IF' MAP[TXCODE, RXCODE] <> 195 'THEN'
MAP[TXCODE, RXCODE] := ('IF' MAP[TXCODE, RXCODE] = 196 'THEN' 195 'ELSE' 194)
'END'
'ELSE' MAP[TXCODE, RXCODE] = 196
'END'
'IF' OPTION (*TO SEND MAP TO FILE*) = TRUE 'THEN'
'BEGIN'
'COMMENT' **destination is disc file**
FILFIL (*MAP*, 1, ERROR);
WRITE TEXT (OUTFILE, "MAP FILE");
ENDLINE(OUTFIL, 1);
DEST := OUTFIL
'END'
'ELSE' DEST := TTO (### destination is terminal)
'FOR' TXCODE := -1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
'BEGIN'
'COMMENT' **send each row of map array to destination**
ENDLINE (DEST, 1));
'IF' TXCODE > 0 'THEN'
'BEGIN'
'COMMENT' **send subsystem name**
NAME (DEST, SS NAME, CONF SS LIST[TXCODE]);
NAME TAB (DEST, SS NAME, CONF SS LIST[TXCODE], 17)
'END'
'ELSE' SPACES (DEST, 17);
'FOR' RXCODE := -1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
'BEGIN'
'COMMENT' **send row elements**
'IF' RXCODE > 0 'THEN' SEND CHAR (DEST, 'OCTAL'(240));
SEND CHAR (DEST, MAP[TXCODE, RXCODE]);
'END'
'END'
'END'
`IF` Best <> TTO `THEN`
`BEGIN`
  ENDLINE(TTO2);
  WRITE TEXT (TTO, "MAPPING COMPLETE");
  ENDLINE(TTO1);
  NEWLINE (OUTFILE.1)
  CLOSE WRITE FILE
`END`
`END` MAPDAT

`PROCEDURE` SSPRDA
`BEGIN`

  `COMMENT` SSPRDA (LEVEL 2)
  `SAVANT Command.
  `Produces a list of all data passing between one source/sink pair.
  `Called by: OBEY COMMAND
  `Calls: CMDWRN, LISSSP, LOCHIA
  `IF` STATE <> CONFIGURED
  `AND` STATE <> FORMED
  `THEN` CMDWRN (1)
  `ELSE`
  `BEGIN`
  `INTEGER` TXCODE, RXCODE, DONE, ERRMESS
  `COMMENT`
  `TXCODE! Pointer to source on CONF SS LIST.
  `RXCODE! Pointer to sink on CONF SS LIST.
  `DONE! Flag to indicate traffic exists.
  `ERRMESS! Pointer to error message string.
  ERRMESS := ""
  SUBSYSTEM NOT CONFIGURED
  ""
  LOCHIA ("SOURCE SUBSYSTEM", SS NAME, TXCODE);
  `IF` TXCODE > 0 `THEN`
  `BEGIN`
  `COMMENT` source exists - find it on CONF SS LIST
  TXCODE := CLISTR (TXCODE);
  `IF` TXCODE > CONF SS COUNT `THEN` WRITE TEXT (TTO, ERRMESS)
  `ELSE`
  `BEGIN`
  `COMMENT` found it - locate sink
  LOCHIA ("SINK SUBSYSTEM", SS NAME, RXCODE);
  `IF` RXCODE > 0 `THEN`
  `BEGIN`
  `COMMENT` sink exists - find it on CONF SS LIST
  RXCODE := CLISTR (RXCODE);
  `IF` RXCODE > CONF SS COUNT `THEN` WRITE TEXT (TTO, ERRMESS)
  `ELSE` `IF` TXCODE = RXCODE `THEN`
  `BEGIN`
  ENDLINE(TTO1);
  WRITE TEXT (TTO, "SAME SINK AS SOURCE");
  ENDLINE(TTO1);
  `END`
  `ELSE`
  `BEGIN`  `COMMENT` found it - list traffic;

`END`
LISSP (TXCODE, RXCODE, DONE);
'IF' DONE = FALSE 'THEN'
'BEGIN'
ENDLINE(TTD:1))
WRITE TEXT (TTD, "NO DATA")
ENDLINE(TTD:1)
'END'
'END'
'END'
'END'
'END' SSPRDA)

'PROCEDURE' CKCONS
'BEGIN'

'COMMENT'

*** SAVANT Command.
*** Determines the fatal inconsistencies in the inconsistent system.
*** If there are no fatal data inconsistencies, it concludes that there
*** must be too many potential bus terminals.
*** Called by: OBEY COMMAND
*** Called: CHWPNR FATLCK!

'IF' STATE <> INCONSISTENT
'THEN' CHWPNR (1)
'ELSE'
'BEGIN'
'INTEGER' FATAL;

'COMMENT'

*** FATAL: Flags to confirm fatal data inconsistencies!

FATAL := FALSE;
FATLCK (FATAL);
'IF' FATAL = FALSE 'THEN'
'BEGIN'
ENDLINE(TTD:1))
WRITE TEXT (TTD, "NO FATAL DATA INCONSISTENCIES - ");
WRITE TEXT (TTD, "TOO MANY TERMINALS");
ENDLINE(TTD:1)
'END'
'END'
'END' CKCONS;

'PROCEDURE' CKCORR
'BEGIN'

'COMMENT'

*** SAVANT Command.
*** Performs a check that all input and output requirements in the
*** configured system are satisfied; reporting the details of any
*** deficiencies. Firstly examines the NRX entries of TXLIST in order
*** to determine whether any transmitted data items have no configured
*** receivers. Then scans the CONF field of the REFLIST to check if
*** any received entries involving configured subsystems have not
*** been dealt with; which implies a received data item requirement
*** which is not supplied by any configured transmitter.
*** Called by: OBEY COMMAND

CKCONS (LEVEL 2)

CKCORR (LEVEL 2)
```assembly
*** Calls: CMDWRN

'IF' STATE <> CONFIGURED
'AND' STATE <> FORMED
'AND' STATE <> INCONSISTENT
'THEN' CMDWRN (1)
'ELSE'
'BEGIN'
'INTEGER' TXPTR, REFPTR, CSSPTR;

'COMMENT'
*** TXPTR Pointer to TXLIST entries.
*** REFPTR Pointer to the RXLIST entries.
*** CSSPTR Pointer to configured subsystem entries.

NEWLINE (TTO; 1);
'FOR' TXPTR := 0 'STEP' 1 'UNTIL' TOTAL TX DATA = 1 'DO'
'IF' NRX(TXPTR) = 0 'THEN'
'BEGIN'
'COMMENT' *** no receiver for tx data
NAME (TTO, DATA NAME, TXDATA[TXPTR]);
WRITE TEXT (TTO, ' FROM '
NAME (TTO, SS NAME, TXSS[TXPTR]);
WRITE TEXT (TTO, ' NOT USED');
ENDLINE(TTO;1);
'END';
'FOR' REFPTR := 0 'STEP' 1 'UNTIL' ENTRIES = 1 'DO'
'IF' CONF(REFPTR) = FALSE 'THEN'
'BEGIN'
'COMMENT' *** entry not dealt with - see if subsystem configured
CSSPTR := 1;
WHILE (CSSPTR <= CONF SS COUNT) 'DO'
'BEGIN'
'IF' SSREFPFR = CONF SS LIST(CSSPTR) 'THEN'
'BEGIN'
'COMMENT' *** subsystem configured - data not supplied
NAME (TTO, DATA NAME, DATAREFPFR);
WRITE TEXT (TTO, ' TO ');
NAME (TTO, SS NAME, SSSREFPFR);
WRITE TEXT (TTO, ' NOT SUPPLIED');
ENDLINE(TTO;1);
CSSPTR := CONF SS COUNT
'END';
INC (CSSPTR);
'END'
'END';
ENDLINE(TTO;1)
WRITE TEXT (TTO, ' CHECK COMPLETE');
ENDLINE(TTO;1)
'END';
'END' CKCORR;

'PROCEDURE' LISMES;
'BEGIN'

'COMMENT'
*** SAVANT Command.
*** Causes details of all bus messages to be listed.
*** Called by: OBEY COMMAND
*** Calls: CMDWRN LISMES;
```

LISMES (LEVEL 2)
'IF' STATE <> CONFIGURED
'AND' STATE <> LIMITED
'AND' STATE <> FORMED
'AND' STATE <> LIMITED FORMED
'THEN' CMDRUN (1)
'ELSE' LISDA (1)
'END' LISTEN

'PROCEDURE' LISDA
'BEGIN'

'COMMENT'
### SAVANT Command.
### Causes details of all direct data transfers to be listed.
### Called by: OBRE COMMAND
### Calls: CMDRUN LISDA

'IF' STATE <> CONFIGURED
'AND' STATE <> LIMITED
'AND' STATE <> FORMED
'AND' STATE <> LIMITED FORMED
'THEN' CMDRUN (1)
'ELSE' LISDA (0)
'END' LISDA

'PROCEDURE' CALLOA
'BEGIN'

'COMMENT'
### SAVANT Command.
### This procedure calculates the percentage bus loadings which
### would be achieved were the configured system implemented in
### practice, using the system parameters for word and message
### overheads, word length and transmission rate. The user specifies
### which subsystem is bus controller - if a name which is not one
### of the configured subsystems is given then a dedicated controller
### is assumed - and also specifies the maximum iteration rate;
### which is that represented by Rate 1. The procedure first
### assembles the number of words, including message overheads,
### associated with each rate group, and also updates several running
### counts in the process. AVLOAD assembles the sum of messages
### times rate for use in average bus loadings calculation; MAXLOAD
### assembles a sum of all messages; for use in peak lumped loadings
### calculations; and MAXDIS is the number of messages at whichever
### rate other than Rate 1 involves the most messages; for use in the
### peak distributed loading calculations. After listing the
### individual rate load figures, the procedure then calculates and
### displays these bus loading percentages.
### Called by: OBRE COMMAND
### Calls: CMDRUN FINSC

'IF' STATE <> CONFIGURED
'AND' STATE <> LIMITED
'AND' STATE <> FORMED
'AND' STATE <> LIMITED FORMED
'THEN' CMDRUN (1)
'ELSE
'BEGIN'
'INTEGER' NPCR, NDIV, RGRP, BCU;
'FLOATING' MAXRATE, MAXLOAD, AVLOAD, MAXDIS, LOAD, FACTOR,
ACTUAL RATE, OHEAD, LENGTH
'FLOATING' ARRAY RATELOAD(11:15)

'COMMENT'
*** MPRT: Pointer to message entries.
*** NDIV: Division counter for determining actual rate.
*** RGRP: The current rate group.
*** BCU: The name code of the bus controller (0 if dedicated).
*** MAXRATE: The maximum iteration rate in Hz.
*** MAXLOAD: See main comment.
*** AVLOAD: ditto
*** MAXDIS: ditto
*** LOAD: The current individual rate load, for listing & comparison.
*** FACTOR: Multiplication factor for forming loading percentages.
*** ACTUAL RATE: The current actual rate in Hz.
*** OHEAD: The overhead for the current message.
*** LENGTH: The number of words in the message (inc. overheads).
*** RATELOAD: Array for individual rate load counts;

FINBB (BCU)
ENDLINE(TTD+1)
WRITE TEXT (TTD, "MAX ITERATION RATE .... ")
MAXRATE := READ NUM (TTI)
MAXLOAD := 0
AVLOAD := 0

'COMMENT' *** initialise RATELOAD array!
'FOR' RGRP := 1 'STEP' 1 'UNTIL' LOWEST RATE 'DO' RATELOAD(RGRP) := 0;
'FOR' MPRT := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES = 1 'DO'

'BEGIN'

'COMMENT' *** work through message list, updating appropriate sums;
RGRP := RGENPRTR31
'IF' RGRP > 0 'THF'

'BEGIN'

'COMMENT' *** message is on the bus:
ACTUAL RATE := 2*MAXRATE;

'FOR' NDIV := 1 'STEP' 1 'UNTIL' RGRP 'DO' ACTUAL RATE := ACTUAL RATE/2;
OHEAD := 'IF' TXMPTR3 = BCU 'OR' RXMPTR3 = BCU
'THEN' CON_DICT OVERHEAD ELSE TERM OVERHEAD
LENGTH := OHEAD + WSDCNPTR31
AVLOAD := AVLOAD + LENGTH*ACTUAL RATE;
MAXLOAD := MAXLOAD + LENGTH;
RATELOAD[RGRP] := RATELOAD[RGRP] + LENGTH

'END'

'COMMENT' *** list loads at each rate and form MAXDIS
ENDLINE(TTD+2)
WRITE TEXT (TTD, "LOAD AT EACH RATE!")
ENDLINE(TTD+2)
ACTUAL RATE := 2*MAXRATE
MAXDIS := 0

'FOR' RGRP := 1 'STEP' 1 'UNTIL' LOWEST RATE 'DO'

'BEGIN'

ACTUAL RATE := ACTUAL RATE/2
LOAD := RATELOAD[RGRP];
WRITE NUM (TTD, ACTUAL RATE, 4, 3)
WRITE TEXT (TTD, ' Hz:
WRITE NUM (TTD, LOAD, 4, 3)
WRITE TEXT (TTD, ' Words:
ENDLINE(TTD+1)
'IF' RGRP > 0 'AND' LOAD > MAXDIS 'THEN' MAXDIS := LOAD

'END'

'COMMENT' *** form multiplication factor to convert word sums into
**PROCEDURE** SUMMES;

**BEGIN**

```
'COMMENT'

SAVANT Command.

Causes a summary list of all messages in the configured system
above to be displayed. The listing cease after each page; with the
option to discontinue. For each message the procedure tabulates
the user message number, source and sink subsystems, rate group,
retry code and number of words. For direct transfers, the rate
is displayed as 'D' and no retry code is output.

Called by: OBEY COMMAND

Calls: CMDWRN OPTIONS

'IF' STATE <> CONFIGURED

'AND' STATE <> LIMITED

'AND' STATE <> FORMED

'THEN' CMDWRN (1)

'ELSE'

'BEGIN'

'INTEGER' NPTR, LINES

'COMMENT'

MPTR: Pointer to message entries.

LINES: Line count.

LINES := 1;

'FOR' NPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'

'BEGIN'

'COMMENT' *** work through message list

'IF' LINES = 1 'THEN'

'BEGIN'

'COMMENT' *** title columns at start of page:

ENDLINE(TTO+1)

WRITE TEXT (TTO, " No. FROM TO RATE RETRY WORDS")

ENDLINE(TTO+1)

WRITE TEXT (TTO, "--- ---- --- ---- ---- ---- ----")

ENDLINE(TTO+1)

'END'

'COMMENT' *** list info

WRITE NUM (TTO, NPTR+1, 3, 0)
```
WRITE TEXT (TTO, "*"");
NAME (TTO, SS NAME, TX(MPTR));
NAME TAB (TTO, SS NAME, TX(MPTR), 17);
NAME (TTO, SS NAME, RX(MPTR));
NAME TAB (TTO, SS NAME, RX(MPTR), 17);
'IF' RG(MPTR) = 0
'THEN' WRITE TEXT (TTO, "  D   ")
'ELSE' 'BEGIN'
WRITE NUM (TTO, RG(MPTR), 5, 0); WRITE TEXT(TTO:"*");
WRITE NUM (TTO, RETRY(MPTR), 6, 0)
'END')
WRITE NUM (TTO, WS(MPTR), 6, 0);
NEWLINE (TTO, 1);
INC (LINES);
'IF' LINES = 21 'THEN'
'BEGIN'
'COMMENT' *** check continue option if page full
LINES := 1;
'IF' OPTION (CONTINUE) = FALSE 'THEN' MPTR := TOTAL MESSAGES
'END'
'END'
'END' SUMMES;

'PROCEDURE' LISIME;
'BEGIN'

'COMMENT'
### SAVANT Command.
### Causes the contents of a single bus message to be listed.
### Called by: OBLOY COMMAND
### Calls: CMDWR massmes
### SENMES

'IF' STATE <> CONFIGURED
'AND' STATE <> LIMITED
'AND' STATE <> FORMED
'AND' STATE <> LIMITED FORMED
'THEN' CMDWR (1)
'ELSE'.
'BEGIN'
'INTEGER' MPR

'COMMENT'
### MPR: Pointer to message to be listed.

FINMES ('MESSAGE NUMBER', MPR);
'IF' MPR >= 0 'THEN' SENMES (MPR)
'END'
'END' LISIME;

'PROCEDURE' SETRET;
'BEGIN'

'COMMENT'
### SAVANT Command.
### Sets retry codes for bus messages. The procedure has two
### modes of operation. In the first mode the user can set the
'PROCEDURE' SETTAD;
'BEGIN'

'COMMENT'
### SAVANT Command.
### Sets terminal addresses, either for all configured subsystems
### or for that specified. If all are being set, any valid ta value
### can be assigned except one already used in the current setting.
### If an individual ta is being set, no value already assigned
### can be used.
### Called by: DVERIFY COMMAND 
### Calls: CMDWRN CKTA
### OPTION LOCNAM
### CONF SS REF;

'IF' STATE <> CONFIGURED
'AND' STATE <> LIMITED
'THEN' CMDWRN (1)
'ELSE'
'BEGIN'
'INTEGER' REPLY, SSCODE, CSSPTR, ERROR;

'COMMENT'
### REPLY: TA value from user.
### SSCODE: Subsystem name code for individual settings.
### CSSPTR: Pointer to configured subsystem entries.
### ERROR: Flag to indicate erroneous reply.

'IF' OPTION ('IF ALL TA'S ARE TO BE SET') = TRUE 'THEN'
'BEGIN'
'COMMENT' ### general settings;
ENDLINE(TTO:1)
WRITE TEXT(TTO: 'TYPE TA VALUE AFTER SUBSYSTEM NAME');
ENDLINE(TTO:1)
CSSPTR := 1
WHILE (CSSPTR <= CONF SS COUNT) 'DO'
'BEGIN'
'COMMENT' ### work through configured subsystems;
'IF' TERM ADDRESS(CSSPTR) < 0 'THEN' INC (CSSPTR)
'ELSE'
'BEGIN'
'COMMENT' ### subsystem is on bus - offer name and set
### valid reply;
ENDLINE(TTO:1)
NAME (TTO, SS NAME, CONF SS LIST(CSSPTR));
WRITE TEXT(TTO: " .... ");
REPLY := READ NUM(TTO);
CKTA (CSSPTR, REPLY, CSSPTR, ERROR);
'IF' ERROR = FALSE 'THEN'
'BEGIN'
'COMMENT' ### record reply
TERM ADDRESS(CSSPTR) := REPLY;
INC (CSSPTR)
'END'
'END'
'END'

'END'

'END'

'END'

'END'

'END'
'END'
'ELSE'
'BEGIN'
'COMMENT' *** Individual setting:
LOCNUM (*SUBSYSTEM*, SS NAME, SS CODE);
'IF' SS CODE > 0 'THEN'
'BEGIN'
'COMMENT' *** Subsystem exists - find it on CONF SS LIST:
CSSPTR := CLISTR (SS CODE);
'IF' CSSPTR > CONF SS COUNT
'THEN'
'BEGIN'
ENDLINE(TTO+1);
WRITE TEXT (TTO, "This subsystem not configured");
ENDLINE(TTO+1);
'END'
'ELSE'
'BEGIN'
'COMMENT' *** See if it's on the bus:
'IF' TERM ADDRESS(CSSPTR) < 0
'THEN'
'BEGIN'
ENDLINE(TTO+1);
WRITE TEXT (TTO, "This subsystem is not on the bus");
ENDLINE(TTO+1);
'END'
'ELSE'
'BEGIN'
'COMMENT' *** Set valid to value and record:
ERROR := TRUE;
WHILE (ERROR = TRUE) 'DO'
'BEGIN'
WRITE TEXT (TTO, "TA value ... ");
REPLY := READ NUM (TTO);
CKTA (CSSPTR, REPLY, CONF SS COUNT, ERROR)
'END'
TERM ADDRESS(CSSPTR) := REPLY
'END'
'END'
'END'
'END' 'SETTAB;

'PROCEDURE' LISSUB;
'BEGIN'

'COMMENT'
*** Savant Command:
*** Causes message subsets to be formed and marked, listing
*** the results. The option is given to pause on each page.
*** Called by: OBEY COMMAND
*** Calls: CHWRN FRMSUB
*** OPTION
'IF' STATE <> CONFIGURED
'AND' STATE <> LIMITED
'AND' STATE <> FORMED
'AND' STATE <> LIMITED FORMED
'THEN' CHWRN (1)
'ELSE'

LISSUB (LEVEL 2)
'BEGIN'
   'INTEGER' MPTR, PAUSE, LINES;

   'COMMENT'
   *** MPTR: Pointer to message entries.
   *** PAUSE: Flag to indicate response to pause option.
   *** LINES: Line count.
   PAUSE := OPTION (EACH PAGE);
   FRMSUB:
   LINES := 1;
   'FOR' MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
   'BEGIN'
      'COMMENT' *** work through message list:
      'IF' SUBSET (MPTR) > 0 'THEN'
      'BEGIN'
         'COMMENT' *** message is a subset - list info:
         WRITE TEXT (TTO, 'MESSAGE*');
         WRITE NUM (TTO, MPRH, 3, 0);
         WRITE TEXT (TTO, ' IS A SUBSET OF MESSAGE*');
         WRITE NUM (TTO, SUBSET (MPRH) + 3, 0);
         NEWLINE (TTO, 1);
         INC (LINES);
      'END'
      'IF' PAUSE = TRUE 'AND' LINES = 23 'THEN'
      'BEGIN'
         'COMMENT' *** check continue option if pausing and msg full:
         LINES := 1;
         'IF' OPTION (CONTINUE) = FALSE 'THEN' MPRH := TOTAL MESSAGES
      'END'
   'END'
   'END' LIISSUB;

   'PROCEDURE' REORDM;
   'BEGIN'

   'COMMENT'
   ***ANTITY Command.
   *** This procedure effects the reordering of data words in a message.
   *** It moves a specified block of data within a message to a new
   *** specified position.
   *** Called by: OBEY COMMAND
   *** Calls: CMDWRN FINMES
   *** SEGMES MODABL;
   'IF' STATE <> CONFIGURED
   'AND' STATE <> LIMITED
   'THEN' CMDWRN (1)
   'ELSE'
   'BEGIN'
      'INTEGER' MPRH, LOW, HIGH, NEW, BLOCK, SIZE;
   'COMMENT'
      *** MPRH: Pointer to message being reordered.
      *** LOW: Pointer to first word of block being moved.
      *** HIGH: Pointer to last word of block being moved.
      *** NEW: Pointer to new position of first word.
      *** BLOCK: Number of words in data block being moved.
      *** SIZE: Length of message being reordered.

   REORDM (LEVEL 2)
FINMES (* MESSAGE TO BE REORDERED*, MPTR)
'IF' MPTR >= 0 'THEN'
'BEGIN'
'COMMENT' *** located message to be reordered - display its contents
*** and locate data block
SENMES (MPTR);
SIZE := WSIZE(MPTR);
ENDLINE(TTO:1);
WRITE TEXT (TTO, 'LOW POINTER .... ');
LOW := READ NUM (TTI);
ENDLINE(TTO:1);
WRITE TEXT (TTO, 'HIGH POINTER .... ');
HIGH := READ NUM (TTI);
ENDLINE(TTO:1);
WRITE TEXT (TTO, 'NEW POSITION .... ');
NEW := READ NUM (TTI);
'IF' LOW > SIZE 'OR' HIGH > SIZE 'OR' LOW > HIGH
'THEN'
'BEGIN'
ENDLINE(TTO:1);
WRITE TEXT (TTO, 'POINTERS INCONSISTENT');
ENDLINE(TTO:1);
'END'
'ELSE'
'BEGIN'
'COMMENT' *** pointers ok - check new position
BLOCK := HIGH - LOW + 1;
'IF' NEW + BLOCK - 1 > SIZE
'THEN'
'BEGIN'
ENDLINE(TTO:1);
WRITE TEXT (TTO, 'SORRY - WONT FIT');
ENDLINE(TTO:1);
'END'
'ELSE'
'BEGIN'
'COMMENT' *** move data block
NDABL (MPTR, MPTR, LOW, NEW, BLOCK);
ENDLINE(TTO:1);
WRITE TEXT (TTO, 'REORDERING COMPLETE');
ENDLINE(TTO:1);
SENMES (MPTR)
'END'
'END'
'END' REORD

'PROCEDURE' REFHTM
'BEGIN'

'REFTM (LEVEL 2)

'REFTM

'PROCEDURE' REFHTM
'BEGIN'

'SAVANT Command...
This procedure performs the reformatting task. It removes a
specified block of data from a yielding message and deposits
it either as a new message or in another existing message.
Known as the accepting message, provided that the source-sink
and rate of the accepting message are identical to those of the
yielding message.
Called bw: OBEY COMMAND
Calls: CMDRNM FINMES
% OPELIS  CLOLIS
% OPTION  SENMES

'IF' STATE <> CONFIGURED
'AND' STATE <> LIMITED
'THEN' ENDWAN (1)
'ELSE'
'BEGIN'
'INTEGER' YMPTR, AMPTR, BLOCK, LOW, HIGH, SIZE, LIST OPENED;

'COMMENT'
*** YMPTR: Pointer to message yielding data.
*** AMPTR: Pointer to message accepting data.
*** BLOCK: Number of words in data block being moved.
*** LOW: Pointer to first word of block being moved.
*** HIGH: Pointer to last word of block being moved.
*** SIZE: Number of words in yielding or accepting message.
*** LIST OPENED: Flag to indicate void message created!

FINMES ('MESSAGE YIELDING DATA*, YMPTR)
'IF' YMPTR > 0 'THEN'
'BEGIN'
'COMMENT' *** yielding message identified!
'IF' OPTION ('IF YIELDED PORTION IS TO CREATE A NEW MESSAGE*) = TRUE 'THEN'
'BEGIN'
'COMMENT' *** new message to be created - open a new at the next
*** message entry to accept the data

AMPTR := YMPTR + 1;
OPELIS (YMPTR);
LIST OPENED := TRUE
'END'
'ELSE'
'BEGIN'
'COMMENT' *** accepting message already exists - locate it!
FINMES ('MESSAGE ACCEPTING DATA*, AMPTR)
LIST OPENED := FALSE
'END'
'IF' AMPTR >= 0 'THEN'
'BEGIN'
'COMMENT' *** check parameters of two messages match!
'IF' TX(AMPTR) <> TX[YMPTR]
'OR' RX(AMPTR) <> RX[YMPTR]
'OR' RX(AMPTR) <> RX(YMPTR)
'THEN'
'BEGIN'
ENDLINE(TTO.1)
WRITE TEXT (TTO, 'MESSAGE PARAMETERS DONT MATCH');
ENDLINE(TTO.1)
'END'
'ELSE'
'BEGIN'
'COMMENT' *** display yielding message and identify block
*** of data to be moved!

SIZE := WDS[YMPTR];
SENMES (YMPTR);
ENDLINE(TTO.1)
WRITE TEXT (TTO, 'LOW POINTER ..... ');
LOW := READ NUM (TTO.1);
ENDLINE(TTO.1)
WRITE TEXT (TTO, 'HIGH POINTER ..... ');
HIGH := READ NUM (TTO.1);
'IF' LOW SIZE 'OR' HIGH SIZE 'OR' LOW > HIGH 'THEN'
'BEGIN'
'COMMENT' *** pointers supplied dont tally!
ENDLINE(TTO:1))
WRITE TEXT (TTO, 'POINTER INCONSISTENT')
ENDLINE(TTO:1))
'IF' LIST OPENED = TRUE 'THEN' CLOLIST (AMPTR)
'END'
'ELSE'
'BEGIN'
'COMMENT' *** check accepting message can accommodate data
BLOCK := HIGH - LOW + 1
'IF' WDS[AMPTR] + BLOCK > MESSAGE LENGTH
'THEN'
'BEGIN'
ENDLINE(TTO:1))
WRITE TEXT (TTO, 'ACCEPTING MESSAGE CANT ACCOMMODATE')
ENDLINE(TTO:1))
'END'
'ELSE'
'BEGIN'
'COMMENT' *** move the data
MODABL (YMPIR, AMPTR, LOW, 1, BLOCK)
ENDLINE(TTO:1))
WRITE TEXT (TTO, 'REFORMATTING COMPLETE')
ENDLINE(TTO:1))
'COMMENT' *** display reformatted messages
'IF'重要举措] > 0
'THEN' SENDMESS (YMPIR)
'ELSE'
'BEGIN'
'COMMENT' *** yielding message now empty!
CLOLIST (YMPIR))
'IF' AMPTR > YMPIR 'THEN' DEC (AMPTR)
'SENDMESS (AMPTR)
'END'
'END'
'END'
'END'
'END' REFMTX;

'PROCEDURE' FILMES;
'BEGIN'

'COMMENT'
*** SAVANT Command.
*** This procedure sends the configured system message structure and
*** supporting arrows to a specified file.
*** Called by: OBEY COMMAND
*** Calls: CHDRARH FILMAN
*** FINFIL FILNUM;

'IF' STATE <> CONFIGURED
'AND' STATE <> LIMITED
'AND' STATE <> FORMED
'AND' STATE <> LIMITED FORMED
'THEN' CHDRARH (1)
'ELSE'

FILMES (LEVEL 2)

FS 452
BEGIN
"INTEGER" CSSPTR; MPTR; MDPTR; SIZE; ERROR;

"COMMENT"
### CSSPTR! Pointer to configured subsystem entries.
### MPTR! Pointer to message entries.
### MDPTR! Pointer to data words in message data array.
### SIZE! Buffer for message length.
### ERROR! Dummy error flag for FINIL;

FINIL (*MESSAGE*, 1; ERROR)
WRITE TEXT (OUTFIL, "MESSAGE FILE")
ENDLINE(OUTFIL,1);

"COMMENT" ### first file required names list
FILMAN (SS NAME);
FILMAN (DATA NAME);
FILNUM (CONF SS COUNT);
"FOR" CSSPTR := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
"BEGIN"
"COMMENT" ### file configured subsystem arrays
FILNUM (CONF SS LIST(CSSPTR));
FILNUM (TERM ADDRESS(CSSPTR));
"END"
FILNUM (TOTAL MESSAGES);
"FOR" MPTR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
"BEGIN"
"COMMENT" ### file message header and contents for each message
FILNUM (TX(MPTR));
FILNUM (RX(MPTR));
FILNUM (POL(MPTR));
FILNUM (RETRY(MPTR));
SIZE := EOS(MPTR);
FILNUM (SIZE);
"FOR" MDPTR := 1 'STEP' 1 'UNTIL' SIZE 'DO' FILNUM (MESSAGE DATA(MPTR; MDPTR))
"END"
CLOSE WRITE FILE;
ENDLINE(TTO+1);
WRITE TEXT (TTO, "MESSAGES FILED")
ENDLINE(TTO+1);
"END"
FINES;

"PROCEDURE" DISMAN
"BEGIN"

"COMMENT"
### SAVANT Command.
### This procedure restores the configured database to the open state.
### initialising affected database structures.
### Called by: OBEY COMMAND
### Calls: CHRN RNL RLRCFL
### CLRMNL;

"IF" STATE <> CONFIGURED
"AND" STATE <> INCONSISTENT
"THEN" CHRN (!)
"ELSE"
"BEGIN"
"INTEGER" REFPRTR

"COMMENT"
*** REF PTR: Pointer to the REFLIST entries:

CLMNL:
CLCFL:
'FOR' REF PTR := 1 'STEP' 1 'UNTIL' ENTRIES = 1 'DO' CONE (REF PTR) := FALSE
ENDLINE (TTO+1);
WRITE TEXT (TTO, "SYSTEM DISMANTLED");
ENDLINE (TTO+1);
STATE := OPEN X
'END'
'END' DISMAN

'PROCEDURE' FRMSYS
'BEGIN'

'COMMENT'
*** SAVANT Command.
*** This procedure is used to form the configured system. It checks
*** that all terminal addresses have been set, and if this is so,
*** forms subsets, then subaddresses. Provided subaddress forms
*** succeeds: the database state is changed.
*** Called by: OBEY COMMAND
*** Calls:
CMDWRN FRMSUB
*** CKTS ST FRMSAD

'IF' STATE <> CONFIGURED
'AND' STATE <> LIMITED
'THEN' CMDWRN (1)
'ELSE'
'BEGIN'

'INTEGER' ERROR:

'COMMENT'
*** ERROR: Flag to indicate error in forming:

CKTS ST (ERROR);
'IF' ERROR = TRUE 'THEN'
'BEGIN'
ENDLINE (TTO+1);
WRITE TEXT (TTO, "ALL TA'S NOT SET");
ENDLINE (TTO+1);
'END'
'ELSE'
'BEGIN'
FRMSUB;
FRMSAD (ERROR);
'IF' ERROR = FALSE 'THEN'
'BEGIN'
ENDLINE (TTO+1);
WRITE TEXT (TTO, "SYSTEM FORMED");
ENDLINE (TTO+1);
STATE := 'IF' STATE = CONFIGURED 'THEN' FORMED 'ELSE' LIMITED FORMED
'END'
'END'
'END' FRMSYS

'PROCEDURE' LISSAD
'BEGIN'

FS 452
'COMMENT'
### SAVANT Command.
### This procedure produces a list of transmitted and received
### subaddress assignments for each bus message in the formed system.
### The user is offered the option to pause on each page and to
### discontinue listing after any page.
### Called by: OBUR COMAND
### Calls: CHDVR (OPTION)
'IF' STATE <> FORMED
'AND' STATE <> LIMITED FORMED
'THEN' CHDVR (1)
'ELSE'.
'BEGIN'
'INTEGER' MPR; PAUSE, LINES;

'COMMENT'
### MPR: Pointer to message entries.
### PAUSE: Flag to indicate response to pause option.
### LINES: Line count.
PAUSE := OPTION (EACH PAGE);
LINES := 1;
'COMMENT' *** Work through message list;
'FOR' MPR := 0 'STEP' 1 'UNTIL' TOTAL MESSAGES - 1 'DO'
'IF' RCMPRJ <> 0 'THEN'
'BEGIN'
'COMMENT' *** Message is on bus - LISSAD
WRITE NUM (TTO, MPR+1: 3: 0);
OUTPUT SPCT(TTO+4);
WRITE TEXT (TTO: "SA")
WRITE NUM (TTO, TXC(MPR), 2: 0);
WRITE TEXT (TTO, " IN ");
NAME (TTO, SS NAME, TXC(MPR));
NAME TAB (TTO, SS NAME, TX(MPR), 17);
WRITE TEXT (TTO, " TO SA ");
WRITE NUM (TTO, RXC(MPR), 2: 0);
WRITE TEXT (TTO, " IN ");
NAME (TTO, SS NAME, RXC(MPR));
NAME TAB (TTO, SS NAME, RXC(MPR), 17);
WRITE NUM (TTO, WD(MPR), 2: 0);
WRITE TEXT (TTO, " WORDS")
ENDLINE(TTO+1);
INC (LINES);
'IF' PAUSE = TRUE 'AND' LINES = 23 'THEN'
'BEGIN'
'COMMENT' *** check continue option if pausing and page full
LINES := 1;
'IF' OPTION (CONTINUE) = FALSE 'THEN' MPR := TOTAL MESSAGES
'END'
'END'
'END' LISSAD;

PROCEDURE FLSCH;
'BEGIN'

'COMMENT'
### SAVANT Command.
### For the formed system, this procedure causes a list of bus control
### schedules to be sent to a specified file, and a list of subaddress
### contents and direct data requirements for each subsystem to another.

```
IF STATE <> FORMED
  AND STATE <> LIMITED FORMED
  THEN CMDWRN (I)
  ELSE
    BEGIN
      INTEGER MPTR, CSSPTR;
      FOR MPTR=0 TO 1 UNTIL TOTAL MESSAGES=0 DO
        BEGIN
          TXSM1(MPTR);=0
          RXSACMPTR;=0
        END
        FOR CSSPTR=1 TO 1 UNTIL CONF SS COUNT 'DO' SUBADD COUNT(CSSPTR);=0
        ENDLINE(TTO;)
        OUTPUT TEXT(TTO;'SYSTEM UNFORMED');
      ENDLINE(TTO;)
      STATE='IF' STATE=FORMED 'THEN' CONFIGURED 'ELSE' LIMITED
    END
  END UNFSYS;
```

'PROCEDURE' INIT
'BEGIN'

'COMMENT'
### Returns the database to the EMPTY state, initialising counts
### and all database tables and arrays.
### Called by SAVD2I, CLRB
### Calls: LISCMDB, CLRCFL
###
'INTEGER' PTR1, PTR2:

'COMMENT'
### PTR1: Pointer used as index for clearing REFLIST and namelists.
### PTR2: Pointer used as index to individual namelist words.

CLRCFL:
CLRCFL:
'FOR' PTR1 := 0 'STEP' 1 'UNTIL' MAXENTRIES - 1 'DO' REFLIST[PTR1] := 0;
ENTRIES := 0;
'FOR' PTR2 := 0 'STEP' 1 'UNTIL' &DO'
'BEGIN'
'FOR' PTR1 := 0 'STEP' 1 'UNTIL' MAXSS 'DO'
'BEGIN'
SS NAME[PTR1], PTR2 := 0;
UNITS NAME[PTR1], PTR2 := 0;
'END';
'FOR' PTR1 := 0 'STEP' 1 'UNTIL' MAXDATA 'DO'
DATA NAME[PTR1], PTR2 := 0;
'END';
SS NAME[0], 13 := MAXSS;
SS NAME[2], 21 := 0;
DATA NAME[0], 13 := MAXDATA;
DATA NAME[2], 21 := 18;
UNITS NAME[0], 13 := MAXSS;
UNITS NAME[2], 21 := 29;
STATE := EMPTY;
LISCMDB
'END' INIT;
'PROCEDURE' CMDWRN ('VALUE''INTEGER' TYPE);
'BEGIN'

'COMMENT'
### Warns of invalid command and reminds of listing option.
### If TYPE is delivered as 0, the command is completely
### unknown; whereas, if TYPE is 1 then the command has been
### recognised but the user has attempted to invoke it in
### a state in which it is not valid. The erroneous command is
### repeated back to the user in the warnings message.
### Called by OB1 COMMAND and all Level 2 procedures.

ENDLINE(TTO+1);
OUTPUT TEXT(TTO,'COMMAND *');
OUTPUT TEXT(TTO,string(STRBUFF));
'IF' TYPE=0 'THEN'
'BEGIN' OUTPUT TEXT(TTO,' NOT UNDERSTOOD');
ENDLINE(TTO+1)
'END';
'ELSE'
'BEGIN' OUTPUT TEXT(TTO,' NOT VALID IN THIS STATE');
ENDLINE(TTO+1)

"PROCEDURE" CMDWRN ("VALUE" 'INTEGER' TYPE)"
'PROCEDURE' INTPCH
('VALUE' 'INTEGER' PARAM, UNITS, IDENT, LIMIT; 'LOCATION' 'INTEGER' VALUE);
'BEGIN'

'COMMENT'
*** Offers the user the option of changing one of the integer
*** preset parameters. If the option is accepted the new value
*** is returned as the parameter VALUE. If the option to change
*** is not accepted, VALUE is returned negative. This means that
*** a new value which is input negative will not be recognised
*** by the calling procedure. PARAM is a text string identifying
*** the parameter in question; IDENT is the current value; UNITS is a
*** text string defining the units in which the quantity is represented
*** and LIMIT is the maximum allowable value. If there is no limit,
*** LIMIT is delivered as 0.
*** Called by: RESET
*** Calls: OPTION

'INTEGER' REPLY;

'COMMENT'
*** REPLY: Buffer for reply;

ENDLINE(TTO);  
OUTPUT TEXT(TTO,PARAM);  
OUTPUT TEXT(TTO,VALUE IS");  
WRITE NUM(TTO,IDENT,4.0);  
OUTPUT SPC(TTO,1) OUTPUT TEXT(TTO,UNITS);  
'IF' OPTION="IF YOU WISH TO CHANGE"=TRUE 'THEN'
'BEGIN'

'COMMENT' *** change required - GETNU;
ENDLINE(TTO);  
OUTPUT TEXT(TTO,TYPE NEW VALUE .... ");  
REPLY:=READ NUM(TTI);  
'IF' LIMIT=0 'THEN'
'BEGIN'

'COMMENT' *** if there is a limit, check the value;
WHILE(REPLY>LIMIT) 'DO'

'BEGIN'

'COMMENT' *** reply over limit - keep trying;
ENDLINE(TTO);  
OUTPUT TEXT(TTO,LIMIT IS");  
WRITE NUM(TTO,LIMIT,2.0);  
OUTPUT TEXT(TTO, .. THEN .... ");  
REPLY:=READ NUM(TTI)

'END'

'END'

'ELSE' REPLY:=-1
VALUE:=REPLY

'END' INTPCH;

'PROCEDURE' FLOPCH
('VALUE' 'INTEGER' PARAM, 'VALUE' 'FLOATING' IDENT, 'LOCATION' 'FLOATING' VALUE);
'BEGIN'

'COMMENT'

'VALUE' offers the user the option of changing one of the floating
'preset parameters. If the option is accepted the new value
is returned as the parameter VALUE. If the option to change
is not accepted, VALUE is returned negative. This means that
a new value which is input negative will not be recognised
by the calling procedure. PARAM is a text string identifying
the parameter in question and IDENT is the current value.
Called by: RESET

Calls: OPTION

ENDLINE(TTD:1);
OUTPUT TEXT(TTD:PARAM);
OUTPUT TEXT(TTD:"VALUE IS");
WRITE NUM(TTD:IDENT:1:1);
OUTPUT TEXT(TTD:"WORDS");
'IF' OPTION("IF YOU WISH TO CHANGE")=TRUE 'THEN'
'BEGIN'
'COMMENT' *** change required - GETNU
ENDLINE(TTD:1);
OUTPUT TEXT(TTD:"TYPE NEW VALUE .... ");
VALUE:=READ NUM(TTI)
'END'
'ELSE' VALUE=1
'END' FLOCH

'PROCEDURE' INDAT
('VALUE' 'BYTE' 'CHNL' 'VALUE' 'INTEGER' 'TRBIT' 'LOCATION' 'INTEGER' 'SCODE' 'ERR');
'BEGIN'

'COMMENT'

'VALUE' inputs a number of entries to the REFLIST, either from a disc
file or from the terminal, depending on the value of CHNL.
Ascertains the number required either by prompting the user or
by reading the first file item. If the number to be read would
cause the REFLIST table bounds to be exceeded, a warning is sent
and no input takes place. The value of TRBIT is set by the
calling procedure. If it is "1 then this indicates that
miscellaneous entries must be read. If it is not negative
then the procedure is expected to read a number of entries
related to a single subsystem, whose name has been read and
is buffered in STRBUFF. A value of 1 indicates that transmitted
entries are to be read and 0, received entries. Each entry
is actually input by the procedure READEN. SCODE is the
name code of the subsystem for the single subsystem type of input.
Its value is used in the calling procedure, and if it is passed on
to the procedure READEN for setting ERR is used to
indicate that the input process has been aborted. This
procedure uses it to indicate too many entries; and it is
passed on to READEN for that procedure to indicate
a full namelist.
Called by: INTTYE INFILE

Calls: READEN

'INTEGER' NUMBER, RDCOUNT

'COMMENT'

'VALUE', NUMBER: Number of entries to be read.
'RDCOUNT': Running count of entries read

FLOCH (LEVEL 3)

INDAT (LEVEL 3)
'COMMENT' *** Find how many entries to be read
'IF' CHNL=TI 'THEN'
'BEGIN'
ENDLINE(TTO,1)
OUTPUT TEXT(TTO, "HOW MANY ");
'IF' TRBIT<0 'THEN' OUTPUT TEXT(TTO, "ENTRIES ... ");
'ELSE' 'IF' TRBIT=0 'THEN' OUTPUT TEXT(TTO, "TRANSMITTED ENTRIES ... ");
'ELSE' OUTPUT TEXT(TTO, "RECEIVED ENTRIES ... ");
'END'
NUMBER:=READ NUM(CHNL);
'IF' ENTRIES+NUMBER>MAXENTRIES 'THEN'
'BEGIN'
'COMMENT' *** this is too many - warn and don't read!
ERR:=TRUE
ENDLINE(TTO, 1);!
OUTPUT TEXT(TTO, "***ERROR*** ONLY ROOM FOR ");
WRITE NUM(TTO, MAXENTRIES-ENTRIES, 4, 0);!
OUTPUT TEXT(TTO, "ENTRIES");
ENDLINE(TTO, 2);
'END'
'ELSE'
'BEGIN'
'COMMENT' *** otherwise, read the entries!
RDCOUNT:=1;
ERR:=FALSE;
WHILE (RDCOUNT<>NUMBER) 'DO'
'BEGIN'
READEN(CHNL, TRBIT, SSCORE, ERR);
'IF' ERR=FALSE 'THEN' INC(ENTRIES) 'ELSE' RDCOUNT:=NUMBER;
INC(RDCOUNT);
'END'
'END'
'END' INDATI;

'PROCEDURE' INNAM ('BYTE', 'ARRAY', 'NAMELIST');
'BEGIN'

'COMMENT' *** During the input of field messages, inputs a namelist from
*** a file to which it has been sent in whole word form by the
*** procedure FILNAM.
*** Called by INFIL;

'INTEGER' MLPTR, STWDPTR, SIZE;

'COMMENT' *** MLPTR: Pointer to namelist entries.
*** STWDPTR: Pointer to individual string words.
*** SIZE: Buffer for namelist size;

SIZE := READ NUM (INFIL);
NAMELIST[0, 0] := SIZE;
FOR MLPTR := 1 'STEP' 1 'UNTIL' SIZE 'DO'
FOR STWDPTR := 0 'STEP' 1 'UNTIL' & 'DO'
NAMELIST[MLPTR, STWDPTR] := READ NUM (INFIL)
'END' INNAM;

'PROCEDURE' LISREF ('VALUE', 'INTEGER', 'CODE');
'BEGIN'

'COMMENT'
### Lists either partial or complete REFLIST. If CODE is 0 the 
### complete REFLIST is listed, rowwise subsystem by subsystem, 
### otherwise only those entries related to the subsystem whose 
### code is the value of CODE are listed. The option is given to 
### pause after each page of the listings; and to discontinue 
### listings after any page. If the listing runs to completion 
### then a count of entries is displayed. 
### Called by: LISENT LISSD 
### Calls: OPTION ACTPRC 

'REFPR: Pointer to the REFLIST. 
ENTRYCOUNT: Count of entries listed. 
SSCODE: The current subsystem name code. 
LOWER: Lower bound subsystem code for output loop. 
UPPER: Upper bound subsystem code for output loop. 
LINES: Line count. 
PAUSE: Flag to indicate response to pause option. 
STOP: Flag to indicate response to continue option. 

PAUSE<OPTION(EACH PAGE)) 
LINES=I; 
STOP=FALSE; 
ENTRYCOUNT+=0; 

'COMMENT' set lower and upper subsystem bounds for listings 
LOWER: 'IF' CODE=0 'THEN' 'ELSE' CODE; 
UPPER: 'IF' CODE=0 'THEN' SS NAME[0+0] 'ELSE' CODE; 
'FOR' SS CODE<LOWER 'STEP' 1 'UNTIL' UPPER 'DO' 
'FOR' REFPTR=0 'STEP' 1 'UNTIL' ENTRIES=1 'DO' 
'IF' SS[REFPR]=SS CODE 'THEN' 

'BEGIN' 
'COMMENT' for each subsystem in turn; list relevant entries 
'IF' LINES=1 'THEN' 
'BEGIN' 
'COMMENT' title columns at beginning of listings and at 
beginning of each page if pausing 
ENDLINE(TTO+1)) 
OUTPUT TEXT(TTO,"SUBSYSTEM DATA T/R RATE PREC UNITS") 
ENDLINE(TTO+1)); 
OUTPUT TEXT(TTO,"" ); 
ENDLINE(TTO+1)); 

'END'; 
NAME(TTO,SS NAME,SSCODE); 
NAME TAB(TTO,SS NAME,SSCODE,17); 
NAME(TTO,DATA NAME,DATA[REFPR]); 
NAME TAB(TTO,DATA NAME,DATA[REFPR],17); 
'IF' TR[REFPR]=1 'THEN' OUTPUT TEXT(TTO,"T") 'ELSE' OUTPUT TEXT(TTO,"R") 
OUTPUT SPC(TTO,9); 
WRITE MNU(TTO,RATE[REFPR]+2,0) 
OUTPUT SPC(TTO,6); 
WRITE MNU(TTO,ACTPRC[PRECE[REFPR]],4+0); 
OUTPUT SPC(TTO,5); 
NAME(TTO,UNITS NAME,UNITS[REFPR]); 
ENDLINE(TTO+1)); 
INC(LINES); 
'IF' PAUSE=TRUE 'AND' LINES=21 'THEN' 
'BEGIN' 
'COMMENT' check continue option if pausing and page full 
LINES=I; 
'IF' OPTION(CONTINUE)=FALSE 'THEN' 

---
'BEGIN'
'COMMENT' *** discontinuing listing
REFPTR=ENTRIES;
SSCODE=UPPER;
STOP=TRUE
'END'
'REND'
INC(ENTRYCOUNT)
'END'
ENDLINE(TTO-1)
'IF' STOP=FALSE 'THEN'
'BEGIN'
'COMMENT' *** if appropriate send count
WRITE NUM(TTO-ENTRYCOUNT+4,0);
OUTPUT TEXT(TTO-"ENTRIES")
ENDLINE(TTO-1)
'END'
'END' LISRFR:

'PROCEDURE' LISNM ('BYTE''ARRAY' NAMELIST) 'VALUE''INTEGER' PAUSE):
'BEGIN'

'COMMENT'
*** Lists the names on the appropriate namelist in five columns.
*** PAUSE specifies whether the option to pause on each pass has
*** been accepted in the calling procedure. If the list is allowed
*** to run to completion then the count of names is also sent.
*** Called by LISSNM
*** Calls: OPTION

'INTEGER' NLPTR, LIST TYPE, SIZE, COLUMN, LINES, STOP:

'COMMENT'
*** NLPTR: Pointer to namelist entries.
*** LIST TYPE: Buffer for namelist type.
*** SIZE: Buffer for namelist size.
*** COLUMN: Column count.
*** LINES: Line count.
*** STOP: Flag to indicate response to continue option.

LINES=11
STOP=FALSE
SIZE=NAMELIST[0,0]
LIST TYPE=NAMELIST[0,2]
COLUMN=5
'FOR' NLPTR=1 'STEP' 1 'UNTIL' SIZE 'DO'
'BEGIN'
'COMMENT' *** work through namelist
INC(COLUMN)
'IF' COLUMN=5 'THEN'
'BEGIN'

'COMMENT' *** start in column 1
COLUMN=11
ENDLINE(TTO-1)
INC(LINES)
'IF' PAUSE=TRUE 'AND' LINES=23 'THEN'
'BEGIN'
'COMMENT' *** check continue option if pausing and pass full
LINES=11
'IF' OPTION(CONTINUE)=FALSE
'THEN' STOP=TRUE

FS 452
'END'

'COMMENT' *** escape if discontinuing
' IF' STOP=TRUE 'THEN' NLPTR=SIZE
' ELSE'
' BEGIN'

'COMMENT' *** otherwise output name
NAME(TTO,NAMELIST, NLPTR);)
' IF' COLUMN<5 'AND' NLPTR< SIZE 'THEN' NAME TAB(TTO,NAMELIST, NLPTR,17)

'END'

'END';

'IF' STOP=FALSE 'THEN'
'BEGIN'

'COMMENT' *** if appropriate send count
ENDLINE(TTO:2);
WRITE NUM(TTO:SIZE,3:0);
' IF' LIST TYPE=0 'THEN'
'BEGIN'

OUTPUT TEXT(TTO,* SUBSYSTEMS*);
ENDLINE(TTO:1);
'END'

'ELSE'
'BEGIN'

OUTPUT TEXT(TTO,* DATA ITEMS*);
ENDLINE(TTO:1);
'END'

'END' LISTNH

'PROCEDURE' CHNAM ('BYTE','ARRAY',NAMELIST))
'BEGIN'

'COMMENT'

'CHNAM (LEVEL 3)

'CHNAM' Changes the name of a subsystem, data item or units identifier.
*** The old name is requested and its code found; if it exists.
*** Then the new name is requested and a check made to find whether
*** the new name is one already existing in the appropriate namelist.
*** If the old name doesn't already exist then the text string for
*** the new name is simply moved to the namelist to overwrite the
*** old name string. If, however, the new name is already
*** existing then the CHNAM procedure is used to change the
*** appropriate REFLIST occurrences of the old name code to the new
*** code and remove the old name. A subsystem name cannot be
*** changed to another existing one without confirmation. If the
*** new name is specified as the same as the old then nothing is
*** changed.
*** Called by CMSSN CHDNM
*** CMUNIG
*** Calls: OPTION CONVNM
*** FINNM MOVSTR

'INTEGER' LIST TYPE, OLD CODE, NEW CODE

'COMMENT'

'LIST TYPE: Buffer for namelist type.
*** OLD CODE: The code of the old name.
*** NEW CODE: The code of the new name if it exists.

LIST TYPE=NAMELIST[0,23]
ENDLINE(TTO:1)
OUTPUT TEXT(TTO,"OLD NAME ..... ");
INPUT TEXT(TTI);
FINNAM(NAMELIST;OLD CODE);
'IF' OLD CODE=0 'THEN' 'BEGIN'
ENDLINE(TTO:11)
OUTPUT TEXT(TTO;"NAME DOESNT EXIST");
ENDLINE(TTO:11)
'END'
'ELSE';
'BEGIN'
'COMMENT' *** old name code found;
ENDLINE(TTO:11)
OUTPUT TEXT(TTO;"NEW NAME ..... ");
INPUT TEXT(TTI)
FINNAM(NAMELIST;NEW CODE);
'IF' NEW CODE=0 'THEN'
'BEGIN'
'COMMENT' *** new name doesn't already exist;
MOVSTR(NAMELIST;OLD CODE);
ENDLINE(TTO:11)
OUTPUT TEXT(TTO;" NAME CHANGED");
ENDLINE(TTO:11)
'END'
'ELSE'; IF NEW CODE< OLD CODE 'THEN'
'BEGIN'
'COMMENT' *** new name already exists;
'IF' LIST TYPE=0 'THEN'
'BEGIN'
ENDLINE(TTO:11)
OUTPUT TEXT(TTO;"SUBSYSTEM NAME ALREADY EXISTS");
'IF' OPTION('TO GO AHEAD WITH CHANGE')=TRUE
'THEN' CONVNM(IS NAME;OLD CODE;NEW CODE)
'END';
'ELSE' CONVNM(NAMELIST;OLD CODE;NEW CODE)
'END';
'ELSE' DO NOTHING; (*** when new and old names are the same)
'END';
'END' CHNAM;

'PROCEDURE' CHNUME ('VALUE'; 'INTEGER' FIELD);
'BEGIN'
'COMMENT'
*** Changes rate or precision value of a specific REFLIST entry.
*** FIELD specifies rate or prec change.
*** Called by CHRATE
*** Calls: GETNV FINENT
'RETURNS' LPOS, NEW, MAX, DUM1, DUM2;

'COMMENT'
*** LPOS! Pointer to specific entry.
*** NEW! New value of rate or prec.
*** DUM1: Subsystem code delivered by FINENT - not used.
*** DUM2! Data code delivered by FINENT - also not used;
FINENT(LPOS,DUM1,DUM2);
'IF' LPOS>=0 'THEN'
'BEGIN'
'COMMENT' *** entry located - proceed with change
GETNV(FIELD;NEW);

FS 452
ENDLNE(TTD=1)
OUTPUT TEXT(TTD,’VALUE CHANGED’);
ENDLNE(TTD=1))
‘END’
‘END’ CHNUM;

‘PROCEDURE’ GENMCH (‘VALUE’ ‘INTEGER’ FIELD);
‘BEGIN’

‘COMMENT’
### Causes all rates or precisions of a particular value to be changed
### to another specified value throughout the REFLIST. The change is
### made subsystem by subsystem. FIELD specifies rate or prec change.
### Called by: CHMAYM CHMPRD
### Calls: GETNV GETOLV
###
### INTEGER’ SSOCDE, SIZE, OLD, NEW

‘COMMENT’
### SSOCDE: The current subsystem name code.
### SIZE: Buffer for namelist size.
### OLD: Rate or prec value to be changed.
### NEW: New value of rate or prec
GETOLV(FIELD=OLD);
GETNV(FIELD=NEW);
SIZE=SS NAMEEO-D1)
‘FOR’ SSOCDE=1 ‘STEP’ 1 ‘UNTIL’ SIZE ‘DO’ CHNUM(SSOCDE,FIELD,OLD,NEW));
ENDLNE(TTD=1))
OUTPUT TEXT(TTD,’MOD COMPLETE’));
ENDLINE(TTD=1))
‘END’ GENMCH

‘PROCEDURE’ CHDAV (‘VALUE’ ‘INTEGER’ FIELD);
‘BEGIN’

‘COMMENT’
### Sets all rates or precisions associated with a particular data
### item to the specified value, whatever they were before. FIELD
### specifies rate or precision change.
### Called by: CHMAYM CHMREF
### Calls: GETNV LOCNUM

### INTEGER’ DACODE, NEW, REFPR1

‘COMMENT’
### DACODE: Specified data name code.
### NEW: Desired rate or prec value.
### REFPR1: Pointer to the REFLIST
LOCNUM(*DATA*,DATA NAME,DA CODE);
‘IF’ DACODE>0 ‘THEN’
‘BEGIN’
‘COMMENT’ ## data name exists – Proceed with channel
GETNV(FIELD=NEW);
‘COMMENT’ ## search REFLIST for data name occurrence
‘FOR’ REFPR1=O ‘STEP’ 1 ‘UNTIL’ ENTRIES=1 ‘DO’
‘IF’ DATA(REFPR1)=DACODE ‘THEN’
'BEGIN
'COMMENT' *** REFLIST entry involves specified data item
'IF' FIELD=0 'THEN' RATE[REFPTR]=NEW 'ELSE' PREC[REFPTR]=NEW
'END'
EOLINE(TTD;1)
OUTPUT TEXT(TTD,"MOD COMPLETE");
EOLINE(TTD;1)
'END'
'END' CHDAV)

'PROCEDURE' CHSSV ("VALUE" 'INTEGER' FIELD)
'BEGIN'
'COMMENT'
*** Invokes the change of all rates or precisions of a particular
*** value to another specified value for relevant REFLIST entries
*** relating to the subsystem specified by the user. FIELD specifies
*** rate or precision change.
*** Called by: CHSSRT  CHSSPR
*** Calls: GETNV GETOLV
*** CHNUM LOCNUM

'INTEGER' OLD, NEW, SSCODE

'COMMENT'
*** OLD: Rate or prec value to be changed.
*** NEW: New value of rate or prec.
*** SSCODE: Code of specified subsystem
LOCNUM('SUBSYSTEM', 'SS NAME', SSCODE))
'IF' SSCODE=0 'THEN'
'BEGIN'
'COMMENT' *** subsystem exists - invoke change
GETOLV(FIELD;OLD);
GETNV(FIELD;NEW);
CHNUM(SSCODE;FIELD;OLD,NEW);  
EOLINE(TTD;1)
OUTPUT TEXT(TTD,"MOD COMPLETE");
EOLINE(TTD;1)
'END'
'END' CHSSV)

'PROCEDURE' DELENT ("VALUE" 'INTEGER' POS)
'BEGIN'
'COMMENT'
*** Removes the entry at position POS from the REFLIST database
*** by overwriting. All higher entries are moved down one place.
*** Called by: DELDR, DELENT

'INTEGER' REFPTR

'COMMENT'
*** REFPTR: Pointer to the REFLIST

'FOR' REFPTR := POS 'STEP' 1 'UNTIL' ENTRIES - 1 'DO'
'BEGIN'
SS[REFPTR] := SS[REFPTR+1];
DATA[REFPTR] := DATA[REFPTR+1];
TR[REFPTR] := TR[REFPTR+1];
'PROCEDURE' FRMTXS;
'BEGIN'

'COMMENT'
'*** During configuration of a system, this procedure determines which
*** subsystems are to be included; moving the appropriate codes onto
*** the CONF SS LIST and initialising the corresponding TERM ADDRESS
*** entries. The user is first offered the option to include all
*** subsystems. If this is declined, each subsystem in turn is offered
*** and the user indicates whether or not it is to be included. If the
*** user declines all subsystems the fact is reported.
*** Called by CONFIG
*** Calls: OPTION

'INTEGER' SSPTMP SIZE)

'COMMENT'
'*** SSPTMP Pointer to SS NAME entries,
*** SIZE: Buffer for namelist size
'IF' OPTION('TO INCLUDE ALL SUBSYSTEMS')=TRUE 'THEN'
'BEGIN'
'COMMENT' *** all subsystems included
'*** CONF SS COUNT:=SIZE;
'FOR' SSPTMP= 'STEP' 1 'UNTIL' SIZE 'DO'
'BEGIN'
'COMMENT' *** transfer all name codes
'*** CONF SS LIST[SSPTMP]=SSPTMP
'*** TERM ADDRESS[SSPTMP]=1;
'END'
'END'
'ELSE'
'BEGIN'
'COMMENT' *** not all subsystems included
'*** CONF SS COUNT:=0;
'FOR' SSPTMP=1 'STEP' 1 'UNTIL' SIZE 'DO'
'BEGIN'
'COMMENT' *** offer each name in turn
'ENDLINE(TTD1);
'NAME(TTD1, SS NAME, SSPTMP);
'IF' OPTION('TO INCLUDE')=TRUE 'THEN'
'BEGIN'
'COMMENT' *** this subsystem included - conv name code
'INC(CONF SS COUNT);
'CONF SS LIST[CONF SS COUNT]:=SSPTMP;
'TERM ADDRESS[CONF SS COUNT]:=1;
'END'
'END'
'END';
'IF' CONF SS COUNT=0 'THEN'
'BEGIN'
'ENDLINE(TTD1);
'OUTPUT TEXT(TTD1, "NO SYSTEM CONFIGURED");
'ENDLINE(TTD1);'
'END'
'END' FRMTXS;

'PROCEDURE' FRMTXD1
'BEGIN'
'COMMENT'
### During the configuration of a system, this procedure searches the
### REFLIST to locate all data items transmitted by each configured
### subsystem in turn. Each item thus found is recorded in the next
### available TXDATA element of TXLIST, with associated subsystem
### rate, precision and units values being set up in the appropriate
### fields of the entry.
### Called by: CONFIG1

'INTEGER' CSSPTR, REFPTR, SCODE)

'COMMENT'
### CSSPTR: Pointer to configured subsystem entries.
### REFPTR: Pointer to the REFList entries.
### SCODE: The current subsystem name code.
TOTAL TX DATA := 0
FOR CSSPTR := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
BEGIN
### take each configured subsystem in turn.
SCODE := CONF SS LIST(CSSPTR);
FOR REFPTR := 0 'STEP' 1 'UNTIL' ENTRIES - 1 'DO'
BEGIN
### search REFList entries;
'IF' (SCREFPTR) = SCODE 'AND' TR(REFPTR) = 1 'THEN'
'BEGIN'
### located data item transmitted by the subsystem;
TXSS[TOTAL TX DATA] := SCODE;
TXDATA[TOTAL TX DATA] := DATA(REFPTR);
TXRATE[TOTAL TX DATA] := RATE(REFPTR);
TXPREC[TOTAL TX DATA] := PREC(REFPTR);
TXUNITS[TOTAL TX DATA] := UNIT(REFPTR);
NXRTOTAL TX DATA] := 0;
CONF(REFPTR) := TRUE;
INC (TOTAL TX DATA)
'END'
'END'
'END' FRMTXD
'PROCEDURE' FRMRXS;
'BEGIN'

'COMMENT'
### During the configuration of a system, this procedure searches the
### REFList in order to locate all configured receivers of each
### transmitted data item in TXLIST, placing their codes in a sublist
### of the RXUNIT elements of the RXLIST and noting their rates,
### precisions and units in the appropriate RXRATE, RXPREC and
### RXUNIT elements. Places a pointer to the start of the sublist
### in the RXSTART element of TXLIST for this data item and records
### the size of the sublist, the number of receivers for the item,
### and the fact is noted in the CONF element of REFList.
### Called by: CONFIG1
### Calls: WORDS
### ACTPRC1

'INTEGER' TXPTR, REFPTR, RXPTR, CSSPTR, DACODE, CFLAG, RXCOUNT;

'COMMENT'
### TXPTR: Pointer to TXLIST entries.
### REFPTR: Pointer to the REFList entries.
### RXPTR: Pointer to RXLIST entries.
TXPTR := 0; RXCOUNT := 0;

WHILE (TXPTR < TOTAL TX DATA) 'DO'
  'BEGIN'
    'COMMENT' *** take each transmitted data item in turn
    DACODE := TXDATA[TXPTR]
    REFPT := 0;
    WHILE (REFPT < ENTRIES) 'DO'
      'BEGIN'
        'COMMENT' *** search REFLIST for receive entries of data item
        'IF' DATACODE 'AND' TR[REFPT] = 0 'THEN'
        'BEGIN'
          'COMMENT' *** located a receive entry - check if receiver configured
          CFLAG := FALSE
          'FOR' CSSPTR := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
            'IF' SSREFPTR = CONF SS LIST [CSSPTR] 'THEN' CFLAG := TRUE
            'IF' CFLAG = TRUE 'THEN'
          /* COMMT */ RXSTART[TXPTR] := RXCOUNT
          RXPTR := RXCOUNT + RX[TXPTR]
          RXSS[TXPTR] := SSREFPTR
          RXRATE[TXPTR] := RATE[REFPT]
          RXPREC[TXPTR] := PRECREFPTR
          RXUNIT[TXPTR] := UNITS[REFPT]
          RXWORDS[TXPTR] := WORDS (ACTPRC (PRECREFPTR))
          INC (NRXIXTPTR)
          CONF[REFPTR] := TRUE
        /* COMMT */
        'END'
      /* COMMT */
      INC (REFPT)
      'END'
      RXCOUNT := RXCOUNT + RX[TXPTR]
      INC (TXPTR)
    /* COMMT */
    'END'
  /* COMMT */
  'END' FRNRXS

PROCEDURE FATLCK ('LOCATION' 'INTEGER' 'FATAL')
  'BEGIN'
    'COMMENT' *** Checks for fatal inconsistencies in a configured system.
    'COMMENT' *** Checks whether the same data is transmitted by more than one
    'COMMENT' *** subsystem; if a data item is received by the subsystem which
    'COMMENT' *** transmits it, and whether any rate, precision or units
    'COMMENT' *** requirement is incompatible with that transmitted.
    'COMMENT' *** If any such inconsistencies exist, the facts are reported and
    'COMMENT' *** FATAL returned true.
    'COMMENT' *** Called by CONFIG, CKCONS
    'COMMENT' *** Called: FATLPR, ACTPRC
    'INTEGER' TXPTR1, TXPTR2, NRXPTR, HEADER, RXPTR
    'COMMENT'
    'COMMENT' TXPTR1: Pointer to TLXLIST entries.
    'COMMENT' TXPTR2: Pointer to TLXLIST entries for multiple tx check.
    'COMMENT' DATAX: Index to current receiver of data item being examined.
    'COMMENT' HEADER: Flag to indicate header needed on multiple tx output list.
    'COMMENT' RXPTR: Pointer to TXLIST entries.

FS 452
'COMMENT' *** first initialise TXCHECK entries
'FOR' TXPTR1=0 'STEP' 1 'UNTIL' TOTAL TX DATA-1 'DO' TXCHECK(TXPTR1)=0
'FOR' TXPTR1=0 'STEP' 1 'UNTIL' TOTAL TX DATA-1 'DO'
'BEGIN'
'COMMENT' *** take each transmitted data item in turn
HEADER=TRUE
TXPTR2=TXPTR1+1
WHILE (TXPTR2<TOTAL TX DATA) 'DO'
'BEGIN'
'COMMENT' *** look for another transmitter of same data
'IF' TXDATA(TXPTR1)=TXDATA(TXPTR2) 'AND' TXCHECK(TXPTR1)=0 'THEN'
'BEGIN'
'COMMENT' *** we've found one not already noted
'IF' HEADER=TRUE 'THEN'
'BEGIN'
'COMMENT' *** identify data and first transmitter
ENDLINE(TTO:1)
OUTPUT TEXT(TTO,*** *)
NAME(TTO,DATA NAME+TXDATA(TXPTR1))
OUTPUT TEXT(TTO,* TRANSMITTED BY '*)
NAME(TTO,SS NAME+TXSS(TXPTR1))
ENDLINE(TTO:1)
HEADER=FALSE
'END'
'COMMENT' *** identify other transmitter
OUTPUT TEXT(TTO, AND ')
NAME(TTO,SS NAME+TXSS(TXPTR2))
ENDLINE(TTO:1)
TXCHECK(TXPTR2)=1
FATAL=TRUE
'END'
INC(TXPTR2)
'END'
'COMMENT' *** now check receiver requirements
NRXPTR=0
WHILE (NRXPTR<NRXETXPTR1) 'DO'
'BEGIN'
'COMMENT' *** look at each receiver in turn
RXPTR=RXSTART[TXPTR1]+NRXPTR
'IF' RXSS[RXPTR]=TXSS[TXPTR1] 'THEN'
'BEGIN'
'COMMENT' *** this receiver is the same subsystem as transmitter
ENDLINE(TTO:1)
OUTPUT TEXT(TTO,*** *)
NAME(TTO,DATA NAME+TXDATA[TXPTR1])
OUTPUT TEXT(TTO,* TRANSMITTED AND RECEIVED BY ')
NAME(TTO,SS NAME+TXSS[TXPTR1])
ENDLINE(TTO:1)
FATAL=TRUE
'END'
'COMMENT' *** check incompatible rates, precisions and units
'IF' RXRATE(RXPTR)<RXRATE[TXPTR1] 'OR' RXRATE(RXPTR)>RXRATE[TXPTR1]
'THEN' FATALPR(0,RXPTR,RXPR+FAIL)
'END'
'IF' ACTPRC(RXPRC[RXPTR])<ACTPRC(TXPREC[TXPTR1])
'THEN' FATALPR(1,TXPTR1,RXPR+FATAL)
'IF' RXUNITS[RXPTR]<TXUNITS[TXPTR1] 'THEN' FATALPR(2,TXPTR1,RXPR+FATAL)
INC(NRXPTR)
'END'
'PROCEDURE' FRAMES:
'BEGIN'

'COMMENT'
*** During the configuration of a system, this procedure forms the***
*** bus messages, creating the MESSAGE HEADER table entries and***
*** assembling the associated data word information in the MESSAGE***
*** DATA array. The formation of messages takes place in order***
*** of rates, with Rate 0 transfers dealt with first, then Rate 1***
*** and so on. For each rate, the messages are formed in order***
*** of source subsystem and within this in order of sink subsystem.***
*** The rate, source and sink having been set up, the TXLIST table is***
*** searched for a data item with the correct source. For each entry***
*** thus found, the appropriate RXLIST entries are examined to***
*** find a receiving subsystem which matches the sink and with the***
*** desired rate requirement. The first complete match discovered***
*** for this rate; source and sink causes the next entry of the***
*** MESSAGE HEADER table to be set up and initialises the pointer***
*** to the MESSAGE DATA array entries. The data item is then***
*** placed in the first element of the corresponding row of the***
*** MESSAGE DATA array. Subsequent matches cause the appropriate***
*** data codes to be recorded in the following elements of the***
*** MESSAGE DATA array row. If more than one word is required to***
*** represent a data item, as discovered from the RXWORDS field***
*** of RXLIST, then the RXWORDS field is decremented and the***
*** pointer to TXLIST is decremented, so that it will point to***
*** the same TXLIST entry again, instead of moving on to examine the***
*** next. This means that the same data item is taken account of***
*** as many times as necessary to build up the correct number of***
*** entries in the MESSAGE DATA array. If at any time the number of***
*** words created for the message equals the value of MESSAGE***
*** LENGTH, then the message information is completed by inserting***
*** the correct value in the WDS field of the MESSAGE HEADER entry,***
*** and the next data item found which matches the criteria if***
*** any, will cause a new message entry to be created. When all***
*** data words have been discovered for the target rate, source and***
*** sink, the message entry being filled at this time is completed***
*** by having its WDS field set. The search then continues for the***
*** next sink, source or rate.

'COMMENT'

'INTEGER' RGRP, TXCODE, RXCODE, HEADER, DITEM, DATAX, RXPTR, MCPTR, WCNT;

'COMMENT'
*** RGRP: The current value of rate group.
*** TXCODE: The position in the CONF SS LIST of the current source.
*** RXCODE: The position in the CONF SS LIST of the current sink.
*** HEADER: Flag to indicate MESSAGE HEADER needs to be set up.
*** TXPRT: Pointer to TXLIST entries.
*** DATAX: Index to current receiver of data item being examined.
*** RXPTR: Pointer to RXLIST entries.
*** MCPTR: Pointer to data words in message data array.
*** WCNT: Buffer for RXWORDS value.

TOTAL MESSAGES: 01
MODIFICATION OF SAYANT TO RUN ON THE PDP RSX-11M OPERATING SYSTEM(U) ROYAL AIRCRAFT ESTABLISHMENT FARNBOROUGH (ENGLAND) R J BLUFF ET AL. JAN 82 RAE-TM-FS(F)452 DRIC-BR-85532
ENDLINE(TTO+1)
OUTPUT TEXT(TTO, "THIS MIGHT TAKE SOME TIME........")
'FOR' RGRP=0 'STEP' 1 'UNTIL' LOWEST RATE 'DO'
'FOR' TXCODE=1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
'FOR' RXCODE=1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
'IF' TXCODE>RXCODE 'THEN'
'BEGIN'
"COMMENT" ### rate and source/sink pair set up
HEADER=TRUE
DITEM=0
WHILE (DITEM<TOTAL TX DATA) 'DO'
'BEGIN'
"COMMENT" ### work through TXLIST
'IF' TXSID[DITEM]=CONF SS LIST[TXCODE] 'THEN'
'BEGIN'
"COMMENT" ### found message where source matches
DATAX=0
WHILE (DATAX<HRX[DITEM]) 'DO'
'BEGIN'
"COMMENT" ### investigate each receiver in turn
RXPRTIX=RXSTART[DITEM]+DATAX
'IF' RXRATE[RXPRTIX]=RGRP 'AND' RXSID[RXPRTIX]=CONF SS LIST[RXCODE] 'THEN'
'BEGIN'
"COMMENT" ### sink and rate match
'IF' HEADER=TRUE 'THEN'
'BEGIN'
"COMMENT" ### set up message header entries
TXITOTAL MESSAGE[S]=CONF SS LIST[TXCODE]
RXITOTAL MESSAGE[S]=CONF SS LIST[RXCODE]
RXITOTAL MESSAGE[S]=RGRP
RETRYTOTAL MESSAGE[S]=0
WBSTOTAL MESSAGE[S]=0
MCPTIX=1
HEADER=FALSE
"END"
MESSAGE DATA[TOTAL MESSAGE[S]]=MCPTIX=TXDATA[DITEM]
("### place data code in MESSAGE DATA array")
INC(MCPTIX)
'IF' MCPTIX-MESSAGE LENGTH 'THEN'
'BEGIN'
"COMMENT" ### message is full
WBSTotal MESSAGE[S]=MESSAGE LENGTH
INC(TOTAL MESSAGE[S])
HEADER=TRUE
"END"
'COMMENT' ### see if more words needed for the data item
WCNT=RXWORD[RXPRTIX]
'IF' WCNT=0 'THEN' WCNT=64
DEC(WCNT)
RXWORD[RXPRTIX]=WCNT
DATAX=HRX[DITEM]
'IF' WCNT=0 'THEN' DEC(DITEM)
"END"
INC(DATAX)
"END"
INC(DITEM)
"END"
'IF' WBSTOTAL MESSAGE[S]=0 'AND' HEADER=FALSE 'THEN'
'BEGIN'
"COMMENT" ### round off message with word count
WBSTOTAL MESSAGE[S]=MCPTIX-1
INC(TOTAL_MESSAGES)
'END'
'END' FRAMES

'PROCEDURE' FINBSS ('LOCATION' 'INTEGER' 'FATAL')
'BEGIN'

'COMMENT'

+++ After forswind messages, locates all subsystems which are
+++ on the data bus, marking them as such in the TERM ADDRESS
+++ array. The search takes place in order of configured subsystems,
+++ with the message list being examined either until a message
+++ is found which involves the subsystem in a transfer with
+++ a rate group value >0 or until the list is exhausted.
+++ If the first case holds, then the appropriate entry of TERM
+++ ADDRESS is marked with a 0; otherwise it is left with the
+++ value of -1 set up when the array was initialized. A running
+++ count of bus terminals is kept, and if this exceeds TERMINAL
+++ LIMIT then a message is displayed and the parameter FATAL is
+++ returned TRUE to the calling procedure.
+++ Called by: CONFIG;

'INTEGER' SSCODE, CSSPTR, MPTR, BUSCOUNT;

'COMMENT'

+++ SSCODE: The current subsystem name code.
+++ CSSPTR: Pointer to configured subsystem entries.
+++ MPTR: Pointer to message entries.
+++ BUSCOUNT: The count of bus-connected subsystems.

BUSCOUNT := 0;
'FOR' CSSPTR = 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
'BEGIN'

'COMMENT' +++ work through each subsystem in turn!
SSCODE = CONF SS LIST(CSSPTR);
MPTR := 0;
WHILE (MPTR < TOTAL_MESSAGES) 'DO'
'BEGIN'

'COMMENT' +++ look for messages involving this subsystem:
'IF' TX(MPTR) = SSCODE 'OR' RX(MPTR) = SSCODE 'THEN'
'BEGIN'

'COMMENT' +++ found relevant message:
'IF' RX(MPTR) > 0 'THEN'
'BEGIN'

'COMMENT' +++ and rate value is >0;
TERM ADDRESS [CSSPTR] := 0;
INC (BUSCOUNT);
MPTR := TOTAL_MESSAGES; (+++ search no more)

'END'
'END';
INC (MPTR)
'END'
'END';

'IF' BUSCOUNT > TERMINAL LIMIT 'THEN'
'BEGIN'

'COMMENT' +++ too many bus subsystems:
ENDLINE (TTO + 1);
OUTPUT TEXT (TTO, "+++ FATAL");
ENDLINE (TTO + 1);
OUTPUT TEXT (TTO, "THIS SYSTEM CONTAINS MORE THAN");

FINBSS (LEVEL 3)
WRITE NUM(T0; TERMINAL LIMIT:2:01)
ENDLINE(T0:11)
OUTPUT TEXT(T0: TERMINALS AND IS THEREFORE INVALID FOR FURTHER ANALYSIS): FATAL:=TRUE
"END"
"END" FINISH:

'PROCEDURE' PARMUN:
'BEGIN':

'COMMENT':

'PARNUM (LEVEL 3)

After the formation of messages, places the partition count with
data items which have been partitioned. It does this by
examining the contents of each message in turn. If the current
message is the same as the previous one in the message; then
the count of partitions is incremented and placed with the
current word. If the previous word was not itself marked as
being partitioned, determined with reference to the COUNTING
flag; then this causes the flag to be set; the previous word
to be marked, and the partition count to be initialised. If
the current word is not the same as the previous; and
partitioning has been taking place up to now; then this causes
the COUNTING flag to be reset. The case where partitioning
may extend over a word boundary has to be taken into account,
and for this purpose the O'th element of the MESSAGE DATA
array row is utilised. If the current message being examined
has the same parameters as the previous; meaning that this
message is a continuation; then the last word in the previous
message is transferred into the O'th word of the current; in
order that the comparison algorithm can operate throughout
the message. The only other check that has to be done is that
if a data partition has been discovered in the first data word
of the current message, by comparison with the last word of
the previous; then the count value I has to be implanted in
the partition count of the previous message's last word.
Called by CONFIG:

'INTEGER' MPTR, HDPTR, COUNTING, COUNT:

'COMMENT':

MPTR: Pointer to message entries.
MPTR: Pointer to data words in message data array.
COUNTING: Flag to indicate partitioning is taking place.
COUNT: Value of the partition count:

COUNTING := FALSE: FOR MPTR := 0 'STEP' 1 'UNTIL' TOTAL MSSES - 1 'DO' BEGIN:

'COMMENT': Work through messages!
DATAWORD (MPTR, 0) := 'IF' MPTR > 0,
'AND' RD[MPTR] = RD[MPTR-1]
'AND' TX[MPTR] = TX[MPTR-1]
'AND' RX[MPTR] = RX[MPTR-1]
'THEN' DATAWORD (MPTR-1, MESSAGE LENGTH)
'ELSE' 0:
('SET UP O'th element if relevant:
'FOR' HDPTR := 1 'STEP' 1 'UNTIL' WDS[MPTR] 'DO'
'IF' DATAWORD (MPTR, HDPTR) = DATAWORD (MPTR, HDPTR-1) 'THEN'
BEGIN:
'COMMENT': Found partitioned data word:
'IF' COUNTING = FALSE 'THEN'

P5-58
BEGIN
  'COMMENT' *** first encounter with this partitioned word!
  COUNTING := TRUE
  COUNT := 2 (*** initialise count)
  PARTITION COUNT (NPTR, HDPTR-1) := 11 (*** plant start of count in previous word)
  'IF' HDPTR = 1 THEN' PARTITION COUNT (NPTR, MESSAGE LENGTH) := 1
  'END'
  PARTITION COUNT (NPTR, HDPTR) := COUNT1
  INC (COUNT1)
  'END'
  ELSE' COUNTING := FALSE
  'END'
  'END' PARMUM)

'PROCEDURE' SETMAP
'BEGIN'

'COMMENT'

'INTEGER' TENS, UNITS, ROW, COLUMN, INDEX, CHAR

'COMMENT'

TENS: Tens character of row or column label.
UNITS: Units character of row or column label.
ROW: Row pointer to element.
COLUMN: Column pointer to element.
INDEX: Count of rows or columns labelled.
CHAR: Internal value of telex code character.

'COMMENT' *** first put tens characters in row and column labels;
'FOR' TENS := 0 'STEP' 1 'UNTIL' 6 'DO'
  'FOR' UNITS := 0 'STEP' 1 'UNTIL' 9 'DO'
  'BEGIN'
  'COMMENT' *** count up to number of configured subsystems;
  INDEX := 10 + TENS + UNITS;
  'IF' INDEX <= CONF SS COUNT 'THEN'
  'BEGIN'
    CHAR := 'IF' TENS = 0 'THEN' 160 'ELSE' 176 + TENS;
    MAP[INDEX] := CHAR (*** set up tens character in column label)
    MAP[index] := -11 := CHAR (*** set up tens character in row label)
  'END'
  'END'

'COMMENT' *** now put units character in row and column labels;
'FOR' UNITS := 0 'STEP' 1 'UNTIL' 6 'DO'
  'FOR' TENS := 0 'STEP' 10 'UNTIL' 60 'DO'
  'BEGIN'
  INDEX := TENS + UNITS1
  'IF' INDEX <= CONF SS COUNT 'THEN'
  'BEGIN'
    CHAR := 176 + UNITS;
    MAP[INDEX] := CHAR (*** set up units character in column label)
    MAP[index] := 0 := CHAR (*** set up units character in row label)
  'END'
  'END'

FOR 452
'COMMENT' row put in space dot and asterisk characters
MAPC-1; -1JI = 16j1
MAPC0; 0] := 1601
'FOR' ROW := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
'FOR' COLUMN := 1 'STEP' 1 'UNTIL' CONF SS COUNT 'DO'
MAPCROW, COLUMNJ := 'IF' ROW = COLUMN 'THEN' 170 'ELSE' 174
'END' SETMAP1

'PROCEDURE' L1SSSP ('VALUE' 'INTEGER' TXCODE, RXCODE) 'LOCATION' 'INTEGER' OK)
'BEGIN'

'COMMENT'
### Lists all data items passing from the subsystem whose position
### on CONF SS LIST is delivered as TXCODE to the subsystem whose
### position on CONF SS LIST is delivered as RXCODE. OK is a flag
### returned to the calling procedure to indicate whether or not
### any traffic has been found to exist between this source and
### sink - this is determined by whether or not a header has been

L1SSSP (LEVEL 3)
APPENDIX F1

SAVANT.DLL


OVERLY: .FCTR (SAV00,SAV01,SAV02,SAV03,SAV04,SAV05,SAV06,SAV07,CONT1)
CONT1: .FCTR (SAV08,SAV09,SAV10,SAV11,SAV12,SAV13,SAV14,SAV15,CONT2)
CONT2: .FCTR (SAV16,SAV17,SAV18,SAV19,SAV20,SAV21,SAV22,SAV23,CONT3)
CONT3: .FCTR (SAV24,SAV25,SAV26,SAV27,SAV28,SAV29,SAV30,SAV31,CONT4)
CONT4: .FCTR (SAV32,SAV33,SAV34,SAV35,SAV36,SAV37,SAV38,SAV39,CONT5)
CONT5: .FCTR (SAV40,SAV41,SAV42,SAV43,SAV44,SAV45,SAV46,SAV47,CONT6)
CONT6: .FCTR (SAV48,SAV49,SAV50,SAV51)

SAV00: .FCTR SAVLIB/LB:NUM00-LB:[32,1]SAVLIB/LB
SAV01: .FCTR SAVLIB/LB:NUM01-LB:[32,1]SAVLIB/LB
SAV02: .FCTR SAVLIB/LB:NUM02-LB:[32,1]SAVLIB/LB
SAV03: .FCTR SAVLIB/LB:NUM03-LB:[32,1]SAVLIB/LB
SAV04: .FCTR SAVLIB/LB:NUM04-LB:[32,1]SAVLIB/LB
SAV05: .FCTR SAVLIB/LB:NUM05-LB:[32,1]SAVLIB/LB
SAV06: .FCTR SAVLIB/LB:NUM06-LB:[32,1]SAVLIB/LB
SAV07: .FCTR SAVLIB/LB:NUM07-LB:[32,1]SAVLIB/LB
SAV08: .FCTR SAVLIB/LB:NUM08-LB:[32,1]SAVLIB/LB
SAV09: .FCTR SAVLIB/LB:NUM09-LB:[32,1]SAVLIB/LB
SAV34: .FCTR SAVLIB/LB:NUM34-LB:[32,1]SAVLIB/LB
SAV41: .FCTR SAVLIB/LB:NUM41-LB:[32,1]SAVLIB/LB
SAV45: .FCTR SAVLIB/LB:NUM45-LB:[32,1]SAVLIB/LB
LIB1: .FCRT LB:[1,1]COROT5/LB

END
APPENDIX F2

LIBCMP.COM

COR LIB00;LIB00/-SP=LIB00
COR LIB01;LIB01/-SP=LIB01
COR LIB02;LIB02/-SP=LIB02
COR LIB03;LIB03/-SP=LIB03
COR LIB04;LIB04/-SP=LIB04
COR LIB05;LIB05/-SP=LIB05
COR LIB06;LIB06/-SP=LIB06
COR LIB07;LIB07/-SP=LIB07
COR LIB08;LIB08/-SP=LIB08
COR LIB09;LIB09/-SP=LIB09
COR LIB10;LIB10/-SP=LIB10
COR LIB11;LIB11/-SP=LIB11
COR LIB12;LIB12/-SP=LIB12
COR LIB13;LIB13/-SP=LIB13
COR LIB14;LIB14/-SP=LIB14
COR LIB15;LIB15/-SP=LIB15
COR LIB16;LIB16/-SP=LIB16
COR LIB17;LIB17/-SP=LIB17
COR LIB18;LIB18/-SP=LIB18
COR LIB19;LIB19/-SP=LIB19
COR LIB20;LIB20/-SP=LIB20
COR LIB21;LIB21/-SP=LIB21
COR LIB22;LIB22/-SP=LIB22
COR LIB23;LIB23/-SP=LIB23
COR LIB24;LIB24/-SP=LIB24
COR LIB25;LIB25/-SP=LIB25
COR LIB26;LIB26/-SP=LIB26
COR LIB27;LIB27/-SP=LIB27
COR LIB28;LIB28/-SP=LIB28
COR LIB29;LIB29/-SP=LIB29
COR LIB30;LIB30/-SP=LIB30
COR LIB31;LIB31/-SP=LIB31
COR LIB32;LIB32/-SP=LIB32
COR LIB33;LIB33/-SP=LIB33
COR LIB34;LIB34/-SP=LIB34
COR LIB35;LIB35/-SP=LIB35
COR LIB36;LIB36/-SP=LIB36
COR LIB37;LIB37/-SP=LIB37
COR LIB38;LIB38/-SP=LIB38
COR LIB39;LIB39/-SP=LIB39
COR LIB40;LIB40/-SP=LIB40
COR LIB41;LIB41/-SP=LIB41
COR LIB42;LIB42/-SP=LIB42
COR LIB43;LIB43/-SP=LIB43
COR LIB44;LIB44/-SP=LIB44
COR LIB45;LIB45/-SP=LIB45
COR LIB46;LIB46/-SP=LIB46
COR LIB47;LIB47/-SP=LIB47
COR LIB48;LIB48/-SP=LIB48
COR LIB49;LIB49/-SP=LIB49
COR LIB50;LIB50/-SP=LIB50
COR LIB51;LIB51/-SP=LIB51
COR LIB52;LIB52/-SP=LIB52
COR LIB53;LIB53/-SP=LIB53
COR LIB54;LIB54/-SP=LIB54
COR LIB55/SP=LIB55
COR LIB56/SP=LIB56
COR LIB57/SP=LIB57
COR LIB58/SP=LIB58
COR LIB59/SP=LIB59
COR LIB60/SP=LIB60
COR LIB61/SP=LIB61
COR LIB62/SP=LIB62
COR LIB63/SP=LIB63
COR LIB64/SP=LIB64
COR LIB65/SP=LIB65
COR LIB66/SP=LIB66
COR LIB67/SP=LIB67
APPENDIX F3
--------

OVL CMP CMD
--------

COR OVL00, OVL00/.SP=OVL00
COR OVL01, OVL01/.SP=OVL01
COR OVL02, OVL02/.SP=OVL02
COR OVL03, OVL03/.SP=OVL03
COR OVL04, OVL04/.SP=OVL04
COR OVL05, OVL05/.SP=OVL05
COR OVL06, OVL06/.SP=OVL06
COR OVL07, OVL07/.SP=OVL07
COR OVL08, OVL08/.SP=OVL08
COR OVL09, OVL09/.SP=OVL09
COR OVL10, OVL10/.SP=OVL10
COR OVL11, OVL11/.SP=OVL11
COR OVL12, OVL12/.SP=OVL12
COR OVL13, OVL13/.SP=OVL13
COR OVL14, OVL14/.SP=OVL14
COR OVL15, OVL15/.SP=OVL15
COR OVL16, OVL16/.SP=OVL16
COR OVL17, OVL17/.SP=OVL17
COR OVL18, OVL18/.SP=OVL18
COR OVL19, OVL19/.SP=OVL19
COR OVL20, OVL20/.SP=OVL20
COR OVL21, OVL21/.SP=OVL21
COR OVL22, OVL22/.SP=OVL22
COR OVL23, OVL23/.SP=OVL23
COR OVL24, OVL24/.SP=OVL24
COR OVL25, OVL25/.SP=OVL25
COR OVL26, OVL26/.SP=OVL26
COR OVL27, OVL27/.SP=OVL27
COR OVL28, OVL28/.SP=OVL28
COR OVL29, OVL29/.SP=OVL29
COR OVL30, OVL30/.SP=OVL30
COR OVL31, OVL31/.SP=OVL31
COR OVL32, OVL32/.SP=OVL32
COR OVL33, OVL33/.SP=OVL33
COR OVL34, OVL34/.SP=OVL34
COR OVL35, OVL35/.SP=OVL35
COR OVL36, OVL36/.SP=OVL36
COR OVL37, OVL37/.SP=OVL37
COR OVL38, OVL38/.SP=OVL38
COR OVL39, OVL39/.SP=OVL39
COR OVL40, OVL40/.SP=OVL40
COR OVL41, OVL41/.SP=OVL41
COR OVL42, OVL42/.SP=OVL42
COR OVL43, OVL43/.SP=OVL43
COR OVL44, OVL44/.SP=OVL44
COR OVL45, OVL45/.SP=OVL45

---

COR OVL46, OVL46/.SP=OVL46
COR OVL47, OVL47/.SP=OVL47
COR OVL48, OVL48/.SP=OVL48
COR OVL49, OVL49/.SP=OVL49
COR OVL50, OVL50/.SP=OVL50
COR OVL51, OVL51/.SP=OVL51
APPENDIX F4

LIBGEN.CMD

;THIS BUILDS THE LIBRARY FOR SAVANT
PIP SAVLIB.OLB;*/DE
LBR SAVLIB/CR
LBR SAVLIB/IN=LIB00
LBR SAVLIB/IN=LIB01
LBR SAVLIB/IN=LIB02
LBR SAVLIB/IN=LIB03
LBR SAVLIB/IN=LIB04
LBR SAVLIB/IN=LIB05
LBR SAVLIB/IN=LIB06
LBR SAVLIB/IN=LIB07
LBR SAVLIB/IN=LIB08
LBR SAVLIB/IN=LIB09
LBR SAVLIB/IN=LIB10
LBR SAVLIB/IN=LIB11
LBR SAVLIB/IN=LIB12
LBR SAVLIB/IN=LIB13
LBR SAVLIB/IN=LIB14
LBR SAVLIB/IN=LIB15
LBR SAVLIB/IN=LIB16
LBR SAVLIB/IN=LIB17
LBR SAVLIB/IN=LIB18
LBR SAVLIB/IN=LIB19
LBR SAVLIB/IN=LIB20
LBR SAVLIB/IN=LIB21
LBR SAVLIB/IN=LIB22
LBR SAVLIB/IN=LIB23
LBR SAVLIB/IN=LIB24
LBR SAVLIB/IN=LIB25
LBR SAVLIB/IN=LIB26
LBR SAVLIB/IN=LIB27
LBR SAVLIB/IN=LIB28
LBR SAVLIB/IN=LIB29
LBR SAVLIB/IN=LIB30
LBR SAVLIB/IN=LIB31
LBR SAVLIB/IN=LIB32
LBR SAVLIB/IN=LIB33
LBR SAVLIB/IN=LIB34
LBR SAVLIB/IN=LIB35
LBR SAVLIB/IN=LIB36
LBR SAVLIB/IN=LIB37
LBR SAVLIB/IN=LIB38
LBR SAVLIB/IN=LIB39
LBR SAVLIB/IN=LIB40
LBR SAVLIB/IN=LIB41
LBR SAVLIB/IN=LIB42
LBR SAVLIB/IN=LIB43
LBR SAVLIB/IN=LIB44
LBR SAVLIB/IN=LIB45
LBR SAVLIB/IN=LIB46
LBR SAVLIB/IN=LIB47
LBR SAVLIB/IN=LIB48
LBR SAVLIB/IN=LIB49
LBR SAVLIB/IN=LIB50
LBR SAVLIB/IN=LIB51
LBR SAVLIB/IN=LIB52
LBR SAVLIB/IN=LIB53
LBR SAVLIB/IN=LIB54
LBR SAVLIB/IN=LIB55
LBR SAVLIB/IN=LIB56
LBR SAVLIB/IN=LIB57
LBR SAVLIB/IN=LIB58
LBR SAVLIB/IN=LIB59
LBR SAVLIB/IN=LIB60
LBR SAVLIB/IN=LIB61
LBR SAVLIB/IN=LIB62
LBR SAVLIB/IN=LIB63
LBR SAVLIB/IN=LIB64
LBR SAVLIB/IN=LIB65
LBR SAVLIB/IN=LIB66
LBR SAVLIB/IN=LIB67
LBR SAVLIB/IN=OVL00
LBR SAVLIB/IN=OVL01
LBR SAVLIB/IN=OVL02
LBR SAVLIB/IN=OVL03
LBR SAVLIB/IN=OVL04
LBR SAVLIB/IN=OVL05
LBR SAVLIB/IN=OVL06
LBR SAVLIB/IN=OVL07
LBR SAVLIB/IN=OVL08
LBR SAVLIB/IN=OVL09
LBR SAVLIB/IN=OVL10
LBR SAVLIB/IN=OVL11
LBR SAVLIB/IN=OVL12
LBR SAVLIB/IN=OVL13
LBR SAVLIB/IN=OVL14
LBR SAVLIB/IN=OVL15
LBR SAVLIB/IN=OVL16
LBR SAVLIB/IN=OVL17
LBR SAVLIB/IN=OVL18
LBR SAVLIB/IN=OVL19
LBR SAVLIB/IN=OVL20
LBR SAVLIB/IN=OVL21
LBR SAVLIB/IN=OVL22
LBR SAVLIB/IN=OVL23
LBR SAVLIB/IN=OVL24
LBR SAVLIB/IN=OVL25
LBR SAVLIB/IN=OVL26
LBR SAVLIB/IN=OVL27
LBR SAVLIB/IN=OVL28
LBR SAVLIB/IN=OVL29
LBR SAVLIB/IN=OVL30
LBR SAVLIB/IN=OVL31
LBR SAVLIB/IN=OVL32
LBR SAVLIB/IN=OVL33
LBR SAVLIB/IN=OVL34
LBR SAVLIB/IN=OVL35
LBR SAVLIB/IN=OVL36
LBR SAVLIB/IN=OVL37
LBR SAVLIB/IN=OVL38
LBR SAVLIB/IN=OVL39
LBR SAVLIB/IN=OVL40
LBR SAVLIB/IN=OVL41
LBR SAVLIB/IN=OVL42
LBR SAVLIB/IN=OVL43
LBR SAVLIB/IN=OVL44
LBR SAVLIB/IN=OVL45
LBR SAVLIB/IN=OVL46
LBR SAVLIB/IN=OVL47
LBR SAVLIB/IN=OVL48
LBR SAVLIB/IN=OVL49
LBR SAVLIB/IN=OVL50
LBR SAVLIB/IN=OVL51
PIP /NV/CO=SAVLIB.OLB
PIP SAVLIB.OLB/PU
PIP *.OBJ**/DE
COR SAVOVL,SAVOVL/-SP=SAVOVL
APPENDIX F5

--------

SAVBLD, CMD

--------

SAVANT, SAVANT/CR/-SP=SAVANT/MP
MAXBUF=560
//
APPENDIX F6

-----------

SAVMN.CMD

-------------

$STARTING COMPILATION OF OVL??,COR FILES
@OVLCMP
$STARTING COMPILATION OF LIB??,COR FILES
@LIBCMP
$DELETE LISTING FILES
PIF *.LST*#/DE
$STARTING TO BUILD SAVANT LIBRARY
@LIBGEN
$STARTING TO TASK BUILD SAVANT
TKB @SAVBLD
$COMPLETION OF SAVANT BUILD
SAVANT (System Architecture Verifications and Analysis Technique) is a computer program developed specifically to investigate the feasibility of an avionic system design, in terms of problems such as completeness and consistency.

The program was originally developed on a Prime 300 computer system. This document describes alterations and modifications of SAVANT, to enable it to be used on a PDP 11/10H-11N operating system.