STRUCTURED PROGRAMMING TRANSLATORS
STRUCTTRAN-2 User's Manual

General Research Corporation

COPY AVAILABLE TO DDC DOES NOT PERMIT FULLY LEGIBLE PRODUCTION

Approved for public release; distribution unlimited.

ROME AIR DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
GRIFFISS AIR FORCE BASE, NEW YORK 13441
This report consists of the following volumes:

I - Final Report
II - STRUCTRAN-1 User's Manual
III - STRUCTRAN-1 System Design and Implementation Manual
IV - STRUCTRAN-2 User's Manual
V - STRUCTRAN-2 System Design and Implementation Manual

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

This report has been reviewed and approved for publication.

APPROVED:  

DONALD L. MARK  
Project Engineer

APPROVED:  

ROBERT D. KRUTZ, Col, USAF  
Chief, Information Sciences Division

FOR THE COMMANDER:  

JOHN P. HUSS  
Acting Chief, Plans Office

Do not return this copy. Retain or destroy.
MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications ($C^3$) activities, and in the $C^3$ areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.
STRUCTURED PROGRAMMING TRANSLATORS. Volume IV.

R. A. Melton

General Research Corporation
P. O. Box 3587
Santa Barbara CA 93105

Rome Air Development Center (ISIS)
Griffiss AFB NY 13441

Same

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release: distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
Same

18. SUPPLEMENTARY NOTES
RADC Project Engineer:
Donald L. Mark (ISIS)

DMAAC Project Engineer
Ms. Opal Power

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Structured Programming
Precompilers
Translators
Software Tools

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
STRUCTTRAN-2 is a system which converts unstructured FORTRAN programs into structured form. The STRUCTTRAN-2 User's Manual describes the use and operation of this system. STRUCTTRAN-2 is compatible with the STRUCTTRAN-1 precompiler which is a special purpose language preprocessor intended to augment standard FORTRAN and permit its use as the basis of a structured programming effort.
ILLUSTRATIONS

<table>
<thead>
<tr>
<th>NO.</th>
<th>Description</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Use of STRUCTRAN-2</td>
<td>2</td>
</tr>
<tr>
<td>2.2</td>
<td>STRUCTRAN-2 Input Format</td>
<td>4</td>
</tr>
<tr>
<td>2.3</td>
<td>STRUCTRAN-2 Original FORTRAN Source Listing</td>
<td>4</td>
</tr>
<tr>
<td>2.4</td>
<td>STRUCTRAN-2 Output Format</td>
<td>4</td>
</tr>
<tr>
<td>4.1</td>
<td>STRUCTRAN-2 Control</td>
<td>8</td>
</tr>
<tr>
<td>C.1</td>
<td>Statement Descriptor Blocks for Module SUM</td>
<td>15</td>
</tr>
<tr>
<td>C.2</td>
<td>Symbol Table for Module SUM</td>
<td>15</td>
</tr>
<tr>
<td>C.3</td>
<td>DMATRAN Statements for Module SUM</td>
<td>16</td>
</tr>
<tr>
<td>C.4</td>
<td>DD-Paths for Module SUM</td>
<td>16</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

The techniques of Structured Programming are finding increasing application in the computing community. Structured programs are, however, difficult to write in programming languages that do not have the statement types necessary to produce GOTO-free code. DMATRAN is a GOTO-less programming language that augments standard FORTRAN so that it can be used as a basis for structured programming. The STRUCTRAN-1 preprocessor translates DMATRAN into standard FORTRAN. It is compatible with the output of STRUCTRAN-2, a system which converts unstructured FORTRAN programs into structured form in DMATRAN. This "structurization" capability is useful when existing FORTRAN programs are to be maintained, modified, documented, or understood. The enhanced visibility of program structure provided by STRUCTRAN-2 translation aids in these tasks.

General use of STRUCTRAN-2 is discussed in Sec. 2. Appendix A describes the use of STRUCTRAN-2 on the UNIVAC 1108. STRUCTRAN-2 constraints are summarized in Sec. 3. Error messages which may occur during a STRUCTRAN-2 run are summarized in Appendix B, and the high-level audit which may be printed during STRUCTRAN-2 processing is described in Appendix C.
STRUCTRAN-2 translates existing FORTRAN programs into structured DMATRAN programs, logically equivalent to the original FORTRAN programs. Structurization does not change the algorithm implemented by the original program, but reveals the structure of the algorithm so that it may be more readily understood. STRUCTRAN-2 is useful when existing FORTRAN programs are to be maintained, modified, documented, or studied. The DMATRAN program can be listed and executed by using the STRUCTRAN-1 preprocessor (refer to the STRUCTRAN-1 User's Manual).

STRUCTRAN-2 replaces FORTRAN control statements with the following four DMATRAN statement constructs:

- **IF...THEN...ELSE...END IF**—provides block structuring of conditionally executable sequences of statements
- **DO WHILE...END WHILE**—permits iteration of a code segment while a specified condition remains true
- **DO UNTIL...END UNTIL**—permits iteration until a specified condition becomes true
- **BLOCK<name>...END BLOCK** (and corresponding INVOKE<name> statement) —provide a facility for top-down programming and internal subroutines.

STRUCTRAN-2 file usage is shown in Fig. 2.1. The FORTRAN source program to be translated is read from the system card reader (unit number is installation-dependent). A listing of the original FORTRAN program (plus any FORTRAN

---

**Figure 2.1. Use of STRUCTRAN-2**
statements inserted by STRUCTRAN-2) is produced on the system printer. Infrequently, it is necessary to examine the high-level audit trail (Appendix C) produced by STRUCTRAN-2 during the structurization process. When this is the case, the scratch file associated with the audit must be copied to the system printer. The DMATRAN source code is written in card image form onto either the system punch or a permanent file. After the STRUCTRAN-2 run is complete, the STRUCTRAN-1 preprocessor can be used to list and execute the DMATRAN program produced.

2.1 STRUCTRAN-2 INPUT

The only input required by STRUCTRAN-2 is the FORTRAN source code to be structurized. It must be in card image form and should be compilable by the UNIVAC 1108 FORTRAN V compiler (or any compiler which implements a subset of FORTRAN V) with no errors. More than one routine may be structurized during a STRUCTRAN-2 run. No control cards between routines are required; STRUCTRAN-2 recognizes the end of each routine automatically. Additional constraints on the FORTRAN source code are summarized in Sec. 3. Figure 2.2 indicates the input format for STRUCTRAN-2.

2.2 STRUCTRAN-2 OUTPUT

STRUCTRAN-2 lists the FORTRAN source code, with any modifications which may have been made to it (Fig. 2.3). For example, a unique CONTINUE statement is added for any DO statement which was not associated with a unique CONTINUE in the original FORTRAN (line 11 of the example). The listing associates an internal STRUCTRAN-2 statement number with each statement. This statement number is useful in tracing the structurized code back to the original FORTRAN code. The DMATRAN source code is also output by STRUCTRAN-2 in card image format (Fig. 2.4), so that it can be readily processed by the STRUCTRAN-1 preprocessor. A sequence number in columns 73 through 80 of the DMATRAN source code relates each statement to the corresponding statement in the STRUCTRAN-2-produced FORTRAN listing. This sequence information is also displayed on the indented listing produced by STRUCTRAN-1.

It is often appropriate to examine the DMATRAN source code produced by STRUCTRAN-2 for the purpose of making simple but effective improvements. For example, if an original FORTRAN program contains duplicate segments of code, they will be highly visible in the structurized version because their structures will be identical. The BLOCK...END BLOCK and INVOKE statements can be conveniently used to eliminate these redundant segments of code by manually modifying the DMATRAN source code.
SUBROUTINE SUM(N,ARRAY,ISUM)
INTEGER ARRAY(10)
ISUM=0
IF(N.LE.0)GOTO 999
J=0
50 K=N-J
IF(K.GT.10)K=10
DO 100 I=1,K
ISUM=ISUM+ARRAY(I)
100 J=J+1
IF(J.LT.N)GOTO 50
999 RETURN
END

Figure 2.2. STRUCTRAN-2 Input Format

<table>
<thead>
<tr>
<th>STMT</th>
<th>LABEL</th>
<th>STATEMENT TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>SUBROUTINE SUM ( N * ARRAY * ISUM )</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>INTEGER ARRAY ( 10 )</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>ISUM = 0</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>IF ( N *GT. 0 ) THEN</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>J = 0</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>DO UNTIL ( J *GE. N )</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>K = N - J</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>IF ( K *GT. 10 ) K = 10</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>DO 10100 I = 1 + K</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>ISUM = ISUM * ARRAY ( I )</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>J = J + 1</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>CONTINUE</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>IF ( J *LT. N ) GOTO 50</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>RETURN</td>
</tr>
</tbody>
</table>

Figure 2.3. STRUCTRAN-2 Original FORTRAN Source Listing

SUBROUTINE SUM ( N * ARRAY * ISUM )
INTEGER ARRAY ( 10 )
ISUM = 0
IF ( N *GT. 0 ) THEN
J = 0
DO UNTIL ( J *GE. N )
K = N - J
IF ( K *GT. 10 ) K = 10
I = 1
DO UNTIL ( I *GT. K )
ISUM = ISUM * ARRAY ( I )
J = J + 1
I = I + 1
END UNTIL
END UNTIL
END IF
RETURN
END

Figure 2.4. STRUCTRAN-2 Output Format
2.3 DEFAULT PARAMETERS

No provision has been made for user modification of the default parameters. The values for each are determined during STRUCTRAN-2 installation. Default values currently used are listed below:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRUCTRAN-2 input unit</td>
<td>5LINPUt</td>
</tr>
<tr>
<td>STRUCTRAN-2 listing unit</td>
<td>6LOUTPUT</td>
</tr>
<tr>
<td>STRUCTRAN-2 error diagnostics unit</td>
<td>6LOUTPUT</td>
</tr>
<tr>
<td>STRUCTRAN-2 output unit</td>
<td>6LLPUNCH</td>
</tr>
<tr>
<td>Include comments in structurized source code</td>
<td>YES</td>
</tr>
<tr>
<td>BLOCK generation threshold</td>
<td></td>
</tr>
<tr>
<td>Number of duplicated statements</td>
<td>15</td>
</tr>
<tr>
<td>Number of statements</td>
<td>1000</td>
</tr>
<tr>
<td>Number of paths</td>
<td>1,000,000</td>
</tr>
<tr>
<td>Working storage size</td>
<td>10,000</td>
</tr>
</tbody>
</table>
3 STRUCTTRAN-2 CONSTRAINTS

The following constraints should be kept in mind when using STRUCTTRAN-2:

1. There must be at least one blank after every source-text FORTRAN keyword (e.g., REAL I or DO 100 I=1,10, not REALI or DO100).
2. Only UNIVAC 1108 FORTRAN V statement forms are acceptable.
3. Only the ANSI X3.9 form of the ASSIGNED GOTO statement is processed (e.g., all labels to which control can transfer must be indicated in the ASSIGNED GOTO statement).
4. At most 250 keywords, operators, delimiters, and symbols are permitted in any statement (except FORMAT statements and Hollerith strings).
5. DO-statement targets are automatically retargeted when necessary, each to a distinct CONTINUE statement.
6. No decision-to-decision path (DD-path) in the program may contain more than 100 successive assignment statements or unconditional I/O statements.
7. Iterations which have more than one entry may not be correctly structurized. An example of a multiple-entry iteration follows:

   IF(P) GO TO 10
   5 IF(Q) GO TO 20
      I=I+1
   10 J=J+1
   GO TO 5
   20 CONTINUE

The iteration in this example can be entered at label "5" or label "10". The complexity of individual multiple entry iterations determines the results of the structurization. Several outcomes are possible:

(1) Structurized source code is okay (the simple example above will be correctly structurized).

(2) Structurized source code contains a DO WHILE statement that is always true (modify the structurized source code).

(3) Structurization is not completed and an error message describing the program graph is printed (use the program graph to identify the multiple-entry iteration and modify the original FORTRAN source code).
4.1 STRUCTTRAN-2 PROGRAM ORGANIZATION

STRUCTTRAN-2 is organized into eight components, each of which consists of a hierarchy of related modules with specific processing capabilities. These components are used in a definite order to translate FORTRAN to DMATRAN. Table 4.1 shows the STRUCTTRAN-2 components; detailed descriptions of their functions appear in Sec. 4 of the STRUCTTRAN-2 System Design and Implementation Manual.

<table>
<thead>
<tr>
<th>Component</th>
<th>Memory (Octal)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRUC0</td>
<td>77K</td>
<td>Storage Manager, Data Base Services, Support Subroutines, and Control</td>
</tr>
<tr>
<td>STRUC1</td>
<td>3K</td>
<td>Initialization of Data Base</td>
</tr>
<tr>
<td>STRUC2</td>
<td>14K</td>
<td>Lexical Analysis</td>
</tr>
<tr>
<td>STRUC3</td>
<td>14K</td>
<td>FORTRAN Parsing</td>
</tr>
<tr>
<td>STRUC4</td>
<td>13K</td>
<td>Graph Identification</td>
</tr>
<tr>
<td>STRUC5</td>
<td>16K</td>
<td>Graph Analysis</td>
</tr>
<tr>
<td>STRUC6</td>
<td>20K</td>
<td>Formation of DMATRAN Program</td>
</tr>
<tr>
<td>STRUC7</td>
<td>5K</td>
<td>Print/Punch Services</td>
</tr>
</tbody>
</table>

The UNIVAC-1108 installation of STRUCTTRAN-2 operates in an overlay mode. STRUC0 is the root of the overlay structure and causes each of the other components to be loaded and executed in the proper order. All routines that are required by two or more components are included in STRUC0. STRUC0 is always in core, with at most one other component. The approximate size of each component as overlayed on the UNIVAC 1108 is shown in Table 4.1.

The code for all STRUCTTRAN-2 components was developed by GRC personnel, utilizing structured programming techniques throughout.
4.2 STRUCTRAN-2 OPERATION

Each STRUCTRAN-2 component is devoted to a unique task; communication between components is through a common data base called a data base library. The data base library is initially created by STRUC1, modified by STRUC2, STRUC3, STRUC4, and STRUC5, and utilized by STRUC6 and STRUC7. The sequence in which the components are executed is shown in Fig. 4.1, which uses DMATRAN to express a high-level, structured representation of STRUCTRAN-2's operation.

```plaintext
PROCEDURE STRUCTRAN2 CONTROL
INVOKF ( STRUCI = INITIALIZE DATA BASE )
DO WHILE ( END OF THE CURRENT MODULE HAS NOT BEEN REACHED )
  INVOKF ( STRUC2 = PERFORM LEXICAL SCAN OF MODULE TO IDENTIFY
  KEYWORDS AND VARIABLES )
  IF ( END OF THE CURRENT MODULE HAS NOT BEEN REACHED ) THEN
    INVOKF ( STRUC3 = PERFORM PARSE OF MODULE TO IDENTIFY STATEMENT
    TYPES )
    INVOKF ( STRUCT = PRINT FORTRAN MODULE )
    INVOKF ( STRUC4 = IDENTIFY PROGRAM GRAPH )
    INVOKF ( STRUC5 = RECOGNIZE STRUCTURED FORMS IN PROGRAM GRAPH AND
    ABSTRACTLY REPRESENT AS A DMATRAN PROGRAM )
    INVOKF ( STRUC6 = USE STRUCTURED FORM REPRESENTATION TO REWRITE
    MODULE IN DMATRAN )
    INVOKF ( STRUC7 = PUNCH DMATRAN MODULE )
  END IF
END WHILE
END
```

Figure 4.1. Description of STRUCTRAN-2 Control

STRUCO consists of the main program for STRUCTRAN-2, all the routines it causes to be loaded, and all routines which are referred to directly or indirectly in two or more components. Since STRUC1 through STRUC7 communicate through the data base library, all data base service routines and storage manager routines are in STRUCO. Also all support routines performing commonly utilized functions are in STRUCO. The data base library is initialized by STRUC1. STRUC2 inputs the FORTRAN text to be structurized, adding it to the data base library after performing a lexical analysis to recognize FORTRAN keywords and variables. STRUC3 reads the lexically analyzed text from the data base library and parses each statement, adding information on statement types and symbol usage to the data base library. STRUC4 utilizes all of the information accumulated so far in the data base library to identify the program graph of the module. This is added to the data base library in tabular form. STRUC5 performs a reduction-type analysis to recognize structured forms in the graph of the module. It updates the information on the data base library to indicate the structured form of the module. STRUC6 uses this representation to build the DMATRAN version of the original FORTRAN program. STRUC7 is used to print and punch statement text from the data base library. Since a single
program unit in UNIVAC 1108 FORTRAN may contain many routines (due to the internal subroutine capability), the iteration indicated in Fig. 4.1 is necessary to process modules consisting of more than one routine.

4.3 STRUCTRAN-2 DATA STRUCTURES

STRUCTRAN-2 utilizes the data structures shown in Table 4.2. The FORTRAN source text to be structurized is input on L1N. A random-access data base library is created on LIBNEW. The FORTRAN source code is listed on LOUT, translated DMATRAN source code is written on LPUNCH, and a high-level audit trail is output to LDEBUG. The file activity performed by STRUCTRAN-2 was summarized in Fig. 2.1.

<table>
<thead>
<tr>
<th>Logical Unit Number</th>
<th>File Name</th>
<th>Data Structure</th>
<th>Mode</th>
<th>Storage Form</th>
<th>Record Format</th>
<th>Recommended Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6L1LIBNEW</td>
<td>LIBNEW</td>
<td>FORTRAN Program Description</td>
<td>Binary</td>
<td>Random</td>
<td>System Standard (installation-dependent)</td>
<td>Scratch file (mass storage)</td>
</tr>
<tr>
<td>6LOUTPUT</td>
<td>LOUT</td>
<td>Reports</td>
<td>BCD</td>
<td>Sequential</td>
<td>Maximum 132 characters per line</td>
<td>System printer</td>
</tr>
<tr>
<td>6LDEBUG</td>
<td>LDEBUG</td>
<td>Reports</td>
<td>BCD</td>
<td>Sequential</td>
<td>Maximum 132 characters per line</td>
<td>Scratch file (mass storage)</td>
</tr>
<tr>
<td>6LPUNCH</td>
<td>LPUNCH</td>
<td>DMATRAN Translated Text</td>
<td>BCD</td>
<td>Sequential</td>
<td>Card image</td>
<td>System punch or permanent file*</td>
</tr>
<tr>
<td>5LINPUT</td>
<td>LIN</td>
<td>FORTRAN Text</td>
<td>BCD</td>
<td>Sequential</td>
<td>Card image</td>
<td>Card reader or permanent file*</td>
</tr>
</tbody>
</table>

*Permanent files must have been previously created.

NOTE: The above files, along with other intermediate scratch files, are defined in subroutine FLINIT.
APPENDIX A
USING STRUCTRAN-2 ON THE UNIVAC 1108

Various modifications have been made to the UNIVAC 1108 installation of STRUCTRAN-2 to enhance its utility in the UNIVAC 1108 environment. The following points briefly summarize the enhancements:

- STRUCTRAN-2 accepts input either as source cards (run stream data) or as FORTRAN source elements referred to with the @ADD statement.
- STRUCTRAN-2 produces a program file in TPF$ containing DMATRAN source code, translated FORTRAN source code, and FORTRAN relocatable elements for each program unit structurized.

As described in Sec. 2, no special control cards are required. Section 2 describes the use of STRUCTRAN-2, and illustrates STRUCTRAN-2 input and output.

The following run stream translates two FORTRAN source elements from program file PF2, assuming that the absolute element for STRUCTRAN-2 is on file PF1.

```
: Assign permanent files
@XQT PF1.STRUCTRAN2
@ADD PF2.MAIN
@XQT PF1.STRUCTRAN2
@ADD PF2.XAMPLE
@COPY TPF$.,PF3.
@FIN
```

STRUCTRAN-2 causes the STRUCTRAN-1 translator to process each DMATRAN element it outputs. A main program and subroutine XAMPLE are the FORTRAN source elements on program file PF2. The result of this run stream includes STRUCTRAN-2 FORTRAN listings of MAIN and XAMPLE, STRUCTRAN-1 indented listings of the structurized MAIN and XAMPLE, FORTRAN V listings of the translated FORTRAN versions for MAIN and XAMPLE, and a program file on PF3 containing the DMATRAN source code, the FORTRAN source code, and the relocatable elements for MAIN and XAMPLE. The DMATRAN source-code elements have MAIN and XAMPLE as element names; both have the version identifier STR. The first card image of DMATRAN source element MAIN is a DMATRAN control card (see STRUCTRAN-1 User's Manual, Appendix A) containing the name MAIN. This is followed by the DMATRAN source code for MAIN. The FORTRAN source code and relocatable elements have MAIN as the element name. Elements for XAMPLE are similar.
APPENDIX B
STRUCTRAN-2 SYSTEM DIAGNOSTICS (FOR USE WITH HIGH-LEVEL AUDIT)

UNKNOWN MESSAGES

- KJONES: UNKNOWN LANGUAGE.
- KJONES: GRAPHICAL ANALYSIS TERMINATED ABNORMALLY.
- HRCNT: WRONG STATEMENT TYPE.
- HRCNT: NO ENDING PARENTHESIS IN GOTO STATEMENT.
- PDAAS: ILLEGAL PREDICATE INDEX.
- DOLAH: LABEL OVERFLOW.
- TRACE: TOO MANY WARNINGS.
- TROH: TOO MANY ERRORS.
- IGNSRA: TOO MANY NON-EXECUTABLE KEYWORDS.
- LABSTM: LABSTM ERROR.
- ADMODL: TO MANY STATEMENT LABELS.
- ADMODL: TOO MANY DO-STATEMENTS.
- CHKLS: COMMON BLOCK OVERFLOW.
- CLASY: TOO MANY KEYWORDS.
- CLASY: TOO MANY KEYPLUSES.
- COMPC: DIMENSIONED SYMBOL DIMENSIONED AGAIN IN COMMON STATEMENT.
- IFPROC: ILLEGAL IF-ASSIGNED-GOTO IN MODULE.
- NTRSYM: SYMBOL EQUIVALENCED BEFORE DIMENSIONED.
- OFFSET: SUBSCRIPT ERROR.
- PKASA: ILLEGAL ASSIGNED-GOTO IN MODULE.
- STHLKL: MODULE HAS UNRECOGNIZABLE LANGUAGE.
- DECLAR: DECLARATION LIST WITH MORE THAN THREE DIMENSIONS.
- GETHLK: SELECTED BLOCK OUT OF KNOWN RANGE.
- GETHLK: ILLEGAL MODULE NUMBER.
- GETLST: GETLST CALLED FOR NON-VARIABLE LENGTH TABLE.
- GETLST: ILLEGAL MODULE NUMBER.
- IGTWU: INDEX BEYOND BLOCK SIZE.
- IGTWU: SELECTED BLOCK OUT OF KNOWN RANGE.
- IGTWU: ILLEGAL MODULE NUMBER.
- LOTHKL: ILLEGAL BLOCK NUMBER.
- LOTHWU: ILLEGAL BLOCK NUMBER.
- LPTHLK: ILLEGAL BLOCK NUMBER.
- LPTWU: ILLEGAL BLOCK NUMBER.
- PUTHLK: SELECTED BLOCK OUT OF KNOWN RANGE.
- PUTLST: ATTEMPT TO REPLACE LIST WITH LONGER ONE.
- PUTLST: NO SUCH BLOCK.
- PUTLST: PUTLST CALLED FOR NON-VARIABLE LENGTH TABLE.
- PUTLST: TOO MANY ITEMS IN VARIABLE LIST.
- PUTLST: ILLEGAL MODULE NUMBER.
- PUTWU: INDEX BEYOND BLOCK SIZE.
- PUTWU: TABLE NOT KNOWN IN SYSTEM.
- PUTWU: ILLEGAL MODULE NUMBER.
C 5100 DMFERO CHANGED MODULE NOT YET ON NEW LIBRARY.
C 5101 DMFERO ACTIVE CORE FRAGMENT NOT FOUND IN EDT.
C 5102 DMFMOD MODULE NOT IN CORE TO DUMP.
C 5103 GETLIB ILLEGAL TABLE NUMBER.
C 5104 GETLIB ILLEGAL LIBRARY NUMBER.
C 5105 GETLIB FRAGMENT BEYOND RANGE.
C 5106 GETMO MODULE NUMBER NOT YET ASSIGNED.
C 5107 GETMO MOD NOT ON LIBRARY.
C 5108 GETMO MODULE OUTSIDE LEGAL RANGE.
C 5109 GETTAK MODULE NUMBER NOT YET ASSIGNED.
C 5110 LHOPEN NO INFORMATION ON READ ONLY LIBRARY.
C 5111 LHRAD ATTEMPT TO READ ILLEGAL LIBRARY IN LHRAD.
C 5112 LHRAD ATTEMPT TO WRITE ON ILLEGAL LIBRARY IN LHRAD.
C 5113 LHRAD NO MORE TABLE SPACE AVAILABLE.
C 5114 MUGET MODULE CAN ONLY BE USED WITH ACTIVE MODULES.
C 5115 MUGPUT MODULE CAN ONLY BE USED WITH ACTIVE MODULES.
C 5116 MUGPUT ILLEGAL DOPATH NUMBER.
C 5117 MUGPUT INDICATED STATEMENT HAS NO BLOCK STRUCTURE.
C 5118 WSGFRO ILLEGAL CORE FRAGMENT NUMBER.
C 5119 WSGFRO ILLEGAL LIBRARY FRAGMENT NUMBER.
C 5120 WSGFRO ILLEGAL LIBRARY ADDRESS.
C 5121 WSGFRO ILLEGAL CORE FRAGMENT NUMBER.
C 5122 WSGFRO ILLEGAL LIBRARY FRAGMENT NUMBER.
C 7100 ACTDAT DID NOT KNOW MODULE WAS ACTIVE.
C 7101 ACTFRO NO SUCH FRAGMENT.
C 7102 ACTFRO FRAGMENT NUMBER GIVEN AS ZERO.
C 7103 ACTFRO ACTMOD DOES NOT KNOW AN ACTIVE MODULE IS
    ACTIVE.
C 7104 ISIF UNBALANCED PARENTHESES.
C 7105 GETFPR TOO MANY CARDS.
C 7106 ISASYN UNBALANCED PARENTHESES.
C 4200 NTSMTM UNKNOWN LANGUAGE.
C 4201 RSOPDK UNKNOWN LANGUAGE.
C 4202 ENEDM NO END STATEMENT.
C 4203 FOPDM POPUP ERROR.
C 4204 PRSTMTM ILLEGAL CALL.
C 4205 STMSUM TOO MANY DOPATHS BEGINNING AT ONE STATEMENT.
C 4206 SUMSTK SUMSTK OVERFLOW.
C 4207 SUNPOH SUMSTK UNDERFLOW.
C 4208 ADDJNC OVERFLOW IN ADDJNC.
C 4209 INTRVL OVERFLOW IN INTRVL.
C 4210 NAPT LOOP WITH NO EXIT.
C 4211 NHEF LOOP WITH NO EXIT.
C 4212 NOSTK NOSTK OVERFLOW.
C 4213 NOSTK NOSTK OVERFLOW.
C 4214 STRUCTM UNKNOWN TYPE.
C 4215 TT UNKNOWN TYPE.
C 4216 SESSTK SSTACK OVERFLOW.
C 4217 PT UNKNOWN TYPE.
C 4218 PROSTK SSTACK OVERFLOW.
C 4219 NEXIT OVERFLOW IN NEXIT.
C 4220 NEXIT UNDERFLOW IN NEXIT.
C 4221 JHVALU RANGE ERROR IN JHVALU.
C 4222 ISRSTAB ILLEGAL TABLE NUMBER.
C 4223 ISRSTAB ISRSTAB CANNOT BE USED WITH VARIARLF
    LENGTH TABLES.
C 4224 ISRSTAB ILLEGAL MODULE NUMBER.
C 4225 STK OVERFLOW.
C 4226 STK OVERFLOW.
C 4227 DOPGEN OVERFLOW IN DOGEN.
C 4228 DOPGEN INFINITE LOOP FOUND.

12
WARNING MESSAGES

5101 SITHELK DO TARGET IS NOT A CONTINUE STATEMENT.
5400 INDUP ARRAY IN INDUP FULL.
5401 INOWN ARRAY IN INOWN FULL.

FATAL ERROR MESSAGES

5200 FROM ERROR LIMIT EXCEEDED.
5200 PUTWRU SELECTED BLOCK OUT OF TABLE BOUNDS.
5201 PUTHLK SELECTED BLOCK OUT OF TABLE BOUNDS.
5112 LHOWN OLD LIBRARY OPENED WITH IMPROPER FRAGMENT SIZE
5119 LHOWN NEW LIBRARY OPENED WITH IMPROPER FRAGMENT SIZE
5120 LHOWN R/W LIBRARY OPENED WITH IMPROPER PASSWORD
7400 SETFOR $ IN COUNT STATS. MUST BE MANUALLY REMOVED.
APPENDIX C

HIGH-LEVEL AUDIT

STRUCTRAN-2 produces a high-level audit which may be printed by copying the normal scratch (unit 8) audit file to the printer. The audit describes the contents of the data base created and used by STRUCTRAN-2 in structuring a FORTRAN program, as well as an indented listing of the structured program. The high-level audit for the program in Fig. 2.2 is shown in Figs. C.1, C.2, C.3, and C.4. For an explanation of the reports generated, refer to the SDB, STB, and DDP Table definitions in STRUCTRAN-2 System Design and Implementation Manual (Sec. 3.3).
### Statement Descriptor Blocks for Module SUM

<table>
<thead>
<tr>
<th>NO.</th>
<th>LABEL</th>
<th>KEYWORD</th>
<th>TYPE</th>
<th>WORD</th>
<th>PMENTS</th>
<th>(GB)</th>
<th>IF</th>
<th>PTR</th>
<th>LABEL</th>
<th>PTR</th>
<th>PRIMARY</th>
<th>NODE</th>
<th>AUX.</th>
<th>NODE</th>
<th>STATIC</th>
<th>COMPLEXITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SUMROUTINE</td>
<td>900</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>INTEGER</td>
<td>1</td>
<td>5</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>ASSIGNMENT</td>
<td>100</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>IF-GOTO</td>
<td>200</td>
<td>8</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>ASSIGNMENT</td>
<td>100</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>DO</td>
<td>100</td>
<td>5</td>
<td>57</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>IF</td>
<td>200</td>
<td>9</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>ON</td>
<td>100</td>
<td>7</td>
<td>85</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>ASSIGNMENT</td>
<td>100</td>
<td>4</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>10100</td>
<td>CONTINUE</td>
<td>200</td>
<td>1</td>
<td>163</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>IF-GOTO</td>
<td>200</td>
<td>4</td>
<td>167</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>RETURN</td>
<td>100</td>
<td>1</td>
<td>143</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>END</td>
<td>100</td>
<td>1</td>
<td>145</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure C.1. Statement Descriptor Blocks for Module SUM**

### Symbol Table

<table>
<thead>
<tr>
<th>NO.</th>
<th>NAME</th>
<th>CLASS</th>
<th>TYPE</th>
<th>MODE</th>
<th>BLOCK</th>
<th>DIM</th>
<th>1ST</th>
<th>3RD</th>
<th>3RD</th>
<th>TIL</th>
<th>LAST</th>
<th>LAST</th>
<th>EQ.</th>
<th>EQ.</th>
<th>COM</th>
<th>STR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SUM</td>
<td>ENTRY</td>
<td>ENTRY</td>
<td>TYPED</td>
<td>SUM</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>N</td>
<td>PARAMETER</td>
<td>VARIABLE</td>
<td>MISMATCH</td>
<td>N</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>ADBY</td>
<td>PARAMETER</td>
<td>ARRAY</td>
<td>TYPED</td>
<td>ADBY</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>COM</td>
<td>PARAMETER</td>
<td>VARIABLE</td>
<td>MISMATCH</td>
<td>COM</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>LOCAL</td>
<td>VARIABLE</td>
<td>INTEGER</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>V</td>
<td>LOCAL</td>
<td>CONSTANT</td>
<td>MISMATCH</td>
<td>V</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>U</td>
<td>LOCAL</td>
<td>LABEL</td>
<td>LABEL</td>
<td>U</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>J</td>
<td>LOCAL</td>
<td>VARIABLE</td>
<td>MISMATCH</td>
<td>J</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>X</td>
<td>LOCAL</td>
<td>LABEL</td>
<td>LABEL</td>
<td>X</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>LOCAL</td>
<td>VARIABLE</td>
<td>MISMATCH</td>
<td>X</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
<td>LOCAL</td>
<td>LABEL</td>
<td>LABEL</td>
<td>X</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>X</td>
<td>LOCAL</td>
<td>VARIABLE</td>
<td>MISMATCH</td>
<td>X</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>N</td>
<td>LOCAL</td>
<td>CONSTANT</td>
<td>MISMATCH</td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>N</td>
<td>LOCAL</td>
<td>LABEL</td>
<td>LABEL</td>
<td>N</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure C.2. Symbol Table for Module SUM**
### Figure C.3. DMATRAN Statements for Module SUM

```plaintext
1  SUMMULINE SUM (N, ARRAY, ISUM)
2  BEGIN ARRAY (10)
3  ISUM = 0
4  IF (N, 0, 1, THEN
5    J = 0
6    DO Until (J, 0, N)
7      K = N - J
8      IF (N, 0, 10) X = 10
9      I = 1
10     DO Until (1, 0, K)
11       ISUM = ISUM * ARRAY (I)
12       J = J + 1
13       END Until
14     END Until
15     END IF
16     RETURN;
17     END
```

### Figure C.4. DD-Paths for Module SUM

<table>
<thead>
<tr>
<th>STATEMENT</th>
<th>END</th>
<th>NO.</th>
<th>NOE</th>
<th>EOG</th>
<th>SUM</th>
<th>PRO</th>
<th>PPN</th>
<th>KNO.</th>
<th>ODS</th>
<th>INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>12</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>13</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>14</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>15</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
</tr>
</tbody>
</table>
```