

**UNCLASSIFIED**

---

---

**AD 296 332**

*Reproduced  
by the*

**ARMED SERVICES TECHNICAL INFORMATION AGENCY  
ARLINGTON HALL STATION  
ARLINGTON 12, VIRGINIA**



---

---

**UNCLASSIFIED**

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

296 332

CATALOGED BY ASTIA

AS AD NO.

296332

63-2-4

TM-555 063 00

JOVIAL For The Dilettante And Beyond

Complex Error Detection Lists

# TECHNICAL MEMORANDUM

(TM Series)

This document was produced in connection with a research project sponsored by SDC's independent research program.

---

JOVIAL For The Dilettante And Beyond:

Compiler Error Detection Lists

by

Millard H. Perstein

2 January 1963

SYSTEM

DEVELOPMENT

CORPORATION

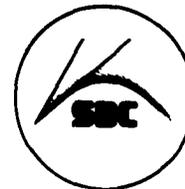
2500 COLORADO AVE.

SANTA MONICA

CALIFORNIA

---

Permission to quote from this document or to reproduce it, wholly or in part, should be obtained in advance from the System Development Corporation.



2 January 1963

1  
Page 2 Blank

TM-555/063/00

JOVIAL for the Dilettante and Beyond  
Compiler Error Detection Lists

| Contents  | Page |
|---|------|
| 1. Introduction . . . . .   | 3    |
| 2. Error Detection in Phase 1 of the Generator . . . . .          | 3    |
| 3. Error Detection in Phase 2 of the Generator . . . . .          | 8    |
| 4. Errors Found by the Generator in Lexington Only . . . . .      | 11   |
| 5. Error Detection in the Translator on the Q7 . . . . .          | 12   |
| 6. Error Detection in the Translator on the CDC 1604 . . . . .    | 15   |
| 7. Error Detection in the Translator on the Philco 2000 . . . . . | 17   |

## JOVIAL for the Dilettante and Beyond:

### Compiler Error Detection List

#### 1. Introduction

The error lists in later sections of this document are the latest versions.

The compilers for the version of JOVIAL with which this series is concerned, J3, are each divided into several phases. The first two phases of each compiler comprise what is known as the generator. The generators in use on all SDC computers are very similar to one another. Beyond the generator phases there is a great deal of difference between compilers. The organization of the following sections reflects this (incomplete) commonness among the generators and the diversity among the later phases.

Even those who have thoroughly studied parts 1 and 2 of "JOVIAL for the Dilettante" (TM-555/061/00 and TM-555/062/00) may make mistakes in writing JOVIAL programs. Many errors, but not all, will be detected by the compiler. In the main the generator will detect the use of ungrammatical or meaningless structures. Phase 1 of the generator will detect mostly errors among the declarations. Phase 2 of the generator will detect mostly errors among the statements. The later parts of the compilers, called the translators, detect some errors which should have been found by the generator and other errors which involve violation of the restrictions established because of machine or translator characteristics.

Notification of detected errors is by number or some other cryptic code. The following sections of this document explain the meanings of these error messages. Some errors, unfortunately, are not detected. Some structures which are technically erroneous will meet no objection and may even result in a program that does what might be expected and desired. It is not, of course, guaranteed that the result will be the same another time. For those who are familiar with only the JOVIAL which has been described for the dilettante, some of the following explanations may be about things which seem obscure. It is unlikely that simple programs will elicit references to such explanations. If such references do occur the mentor should be consulted (see part 1, section 1.2, rule 2).

#### 2. Error Detection in Phase 1 of the Generator

Errors detected in the first phase are numbered from 1 to 99. These numbers refer to the explanations in the list below. The error messages are usually interspersed, on separate lines, with a printed output of the source program.

The error detected will be somewhere in the one or two lines preceding the message. The message will also refer to the location of the error by mentioning "main program" or the procedure name, a statement name, and a numeric increment. If the error occurs before any statement label has been found, this field will be blank. The numeric increment tells how many statements beyond the named one to look for the error. In phase 1 only and only for purposes of locating errors, data declarations will be counted as statements. The statements are counted by counting the terminating dollar signs.

- 1 The START operator is missing.
- 2 Illegal triplet. A JOVIAL "entity" might be defined as a primitive JOVIAL word (such as BEGIN; ITEM or EQ), a constant, a label, a recognized pair of special characters (such as (\$) or (\*\*), or a single special character (such as \$, =, \*, or +). Most of the program is scanned entity by entity, considering the entities three at a time, each entity being considered in turn as the third, second, and first term of a triplet. Error type 2 means that the second entity is illegal in conjunction with the first and third.
- 3\* An illegal character was found.
- 4\* An illegal comment statement occurred.
- 5 A procedure declaration was found within a procedure declaration.
- 6 Illegal triplet. See error type 2. Error type 6 means that the combination of the first and third terms of the triplet is illegal regardless of the class of the second term.
- 7 Illegal dollar signs were found.
- 8 OPEN or SHUT is not followed by INPUT or OUTPUT in the statement.
- 9\* Illegal direct code statement occurred.
- 10\* Illegal double prime occurred.
- 11 Dollar sign is missing from DEFINE declaration.
- 12 Binary point designator in A( ) is not a constant. References the ASSIGN statement. A( ) is the accumulator designator, and any character contained within the parentheses must be a constant.
- 13 The label is missing in a switch declaration.

---

\* These messages are probably due to compiler error or machine malfunction and should be reported to the JOVIAL compiler staff.

- 14 = or ( is missing in switch declaration.
- 15 Item is missing in item switch declaration.
- 16 Procedure label is missing in procedure declaration.
- 17 CLOSE label is missing in CLOSE declaration.
- 18 Dollar sign is missing after CLOSE label.
- 19\* Indeterminable identifier or constant occurred.
- 20 TERM or dollar sign following TERM is missing.
- 21 Comment terminated by \$. This generally indicates the double prime terminating a comment is missing. If the second double prime were missing from a comment, the compiler would not recover until it found a dollar sign. In this case the statement following the comment will be lost.
- 22\* Control is transferred to the zero point in a switch.
- 23 Uneven number of brackets. Includes (, ), (\$, \$), (/ , and /).
- 24 Too many entries in the dictionary. Processing of source program terminates.
- 25 Too many OVERLAY declarations. Processing of source program terminates.
- 26 The total number of characters required for identifier names, constants and data is excessive. Processing of source program terminates.
- 27 Too many DEFINE declarations. Processing of source program terminates.
- 28 Too many characters of information in DEFINE declarations or too many characters in one definition and its associated nested defined labels. Processing of source program terminates.
- 29 Too large a literal constant (Hollerith or transmission code).
- 30 Item, mode, or string declaration type code is not recognized.
- 31 Item, mode, or string declaration number of bits or characters does not follow type code for type A, D, H or I.

---

\* These messages are probably due to compiler error or machine malfunction and should be reported to the JOVIAL Compiler Staff.

- 32 Item, mode, or string declaration number of bits or characters is not an unsigned integer for type A, D, H, T or S.
- 33 Signed or unsigned indication in item, mode, or string declaration is not given for type A or D.
- 34 Item declaration format is confused. Check the format preceding the \$ character.
- 35 A simple item declaration has no type code. Mode definition is used for its type.
- 36 Used only on the Philco 2000 in Lexington. See section 4.
- 37 One of two constants specifying range values in an item declaration is missing.
- 38 A subscripted item declaration in a table with specified packing has confused format preceding the \$ character.
- 39 A simple item or mode declaration has a data constant inconsistent with its variable type.
- 40 A status item, mode, or string declaration has no statuses.
- 41 A status constant assigned to a variable is not found among the variable's statuses. This error message indicates the absence of a status; misspelling of the status constant or the status item; or mistaken assumption about an item in the compool which is not declared therein as a status item.
- 42 The referenced transfer point (statement label, close, procedure, or switch) is missing. Error Type 42 is synonymous with undefined tag. Since a tag cannot be pronounced undefined until the entire program or procedure is examined, the error printout only appears after the main program or procedure. The particular statement label that provokes the error message is included in the error message at the extreme right.
- 43 A non-subscripted variable with a subscript or a subscripted variable without a subscript is referenced. This common error occurs when a table item is used without a subscript. Even though the item may appear in a one-entry table, thus making any subscript other than zero meaningless, the item still requires a subscript.
- 44 An overlay declaration has a format error or an identifier not already in the dictionary.

- 45 A file declaration has no statuses.
- 46 A file declaration has a confused format or uses a multiply-defined identifier.
- 47 A table declaration has a confused format or uses a multiply-defined identifier. The declaration is omitted.
- 48 The model is not found for a like-table declaration.
- 49 An illegal subscripted variable was generated as a result of a like-table declaration. This variable is omitted from the table.
- 50 A table declaration is confused between subscripted item declarations. Error type 50 occurs when either the BEGIN or the END that should be used to enclose the items of a table has been omitted. Error type 50 is generally followed by numerous error codes which result from the confusion created by the omission of the BEGIN or END.
- 51 A string or item declaration component of a specified table declaration has confused format.
- 52 The constant type of pre-set table data is inconsistent with other constants in the same set, or not compatible with the assigned variable. Constants of preset table data must be the same as other constants in the same set, and they must be compatible with the assigned variable. An octal constant is an exception; octal constants may be used as pre-set data for Hollerith, Standard Transmission Code, and A type items.
- 53 No identifier follows "ITEM" in a simple item declaration.
- 54 An item declaration identifier was already used for another item, table, string or file in the same scope.
- 55 A mode declaration with illegal format was not accepted.
- 56 Array format error. Caution - the array may not have been entered in the dictionary.
- 57 A set of array dimensions are not pure unsigned integers.
- 58 Pre-set table data exceeds the table capacity.
- 59 A constant has a format error. Caution, the reduced form may be junk.
- 60 Pre-set parameter constants assigned to an array with more than 3 dimensions is not currently implemented.

- 61 Pre-set parameter constants assigned to an array are excessive. The number of constants in a set of constants is greater than the number of columns.
- 62 A switch, close, or statement label is duplicated in the same scope.
- 63 Caution - mode definition was used. The compiler currently in use may or may not have this error message suppressed. Sometimes an Error Type 63 will bring to light errors in spelling declared items, since the normal mode will create new items with the new spelling (i.e., item PARITY declared and later used erroneously as PARTY). If Error Type 63 is suppressed, it is suggested that the programmer scan the dictionary for similar "surprises."
- 64 A subscripted variable was not previously declared. Mode definition was used to enable continued processing.
- 65 See 61. The number of second level sets of constants is greater than the number of rows.
- 66 See 61. The number of third level sets of constants is greater than the number of planes.
- 95 An end-of-file is the first thing encountered in the JOVIAL program.
- 96 There are more than 10 nested defined labels. (A label probably has been defined recursively.) Processing of the source program has been terminated.
- 97 There is an illegal single prime (').
- 98 There is an error in format of the library file.
- 99 An attempt was made to read past the end-of-file on the library tape.

### 3. Error Detection in Phase 2 of the Generator

Errors detected in the second phase are numbered from 100 to 199. These numbers refer to the explanations in the list below. These error messages are all printed out after the listing of the source program. The error message refers to the location of the error by means of a statement label and a count of unlabeled statements. Declarations are not counted here.

- 101 The statement begins with an illegal part of speech, either a constant, logical operator, relational operator, or arithmetic operator. This may be caused by another error in this statement or the immediately preceding statement.

- 102 The FOR statement to initialize a subscript is in the range of a FOR statement which initialized the same subscript.
- 103 The 2 or 3 factor FOR statement is not terminated by a  $\$$ .
- 104 An expression of the statement contains an odd number of parentheses or brackets, or does not terminate with a comma or  $\$$ .
- 105 A TEST statement is used outside the range of any FOR statement.
- 106 A TEST statement specifies a subscript activated by a one-factor FOR statement.
- 107 A TEST statement is used in the range only of a one-factor FOR statement.
- 108 An inactive subscript is specified by a TEST statement.
- 109 At the end of the program, there are not enough ENDS to match the BEGINS.
- 110 A non-label is used as a switch point in a switch declaration or as the object of a GOTO statement.
- 111 A procedure is used as a switch point in a switch declaration or as the object of a GOTO statement.
- 112 An error in an adjacent statement causes a declaration other than SWITCH, CLOSE or procedure to be missed in phase one.
- 113 A separator other than  $\$$  was found between statements.
- 114 The relational operator EQ is used instead of an = in an assignment statement, or else an IF, IPRITH, or ORIF operator is missing from a relational statement.
- 115 The separator = is used instead of EQ in a relational statement, or else a boolean variable is used in an arithmetic expression.
- 116 An inactive subscript is specified as a factor in a FOR statement.
- 117 There are too many ENDS to match the BEGINS in the program.
- 118 An error is found in the analysis of a FOR factor. See a previous error message for the specific error discovered.
- 119 An expression, rather than a whole statement was found.
- 120 At the start of this procedure, there are too many BEGINS to match the ENDS. Check BEGINS and ENDS in the preceding procedure or main program.

- 121 The statement begins with a parenthesis or subscript bracket, or else a BEGIN, END, START, or TERM is within the statement.
- 122 The program has too many FOR loops.
- 123 A table name is used as a variable; or an ENTRY, ALL, NEXT, or NWDGEN modifier is missing.
- 125 The statement contains an illegal part of speech. Most likely this is caused by a missing \$ or previous error which caused part of a statement to be eliminated. Specifically, a BEGIN, END, START, or TERM or a declaration, I/O operator, or direct code operator occurs within the statement.
- 126 A \*, /, or \*\* operator immediately follows another operator. An operand is missing in the statement.
- 127 A boolean variable is used in an arithmetic rather than a logical expression.
- 128 The statement has an odd number of parentheses or brackets.
- 129 An absolute value modifier or a unary minus sign does not have an object.
- 130 A bead expression has an absolute value modifier.
- 131 An exchange operator, ==, is found in a statement which is probably meant to be a boolean assignment statement.
- 132 An undetermined error, probably due to a missing \$ which causes a part of speech to be used illegally, occurs in the statement.
- 133 A procedure is used as a function in the statement.
- 134 A subscript is used outside of the range of its activating FOR statement.
- 135 One of the left operands for an operator in the statement could not be located. The operand is probably in the beaded variable of a bead expression.
- 136 An excessive number of commas occurs in a bead expression.
- 137 An illegal part of speech occurs within a subscript expression.
- 138 A non-boolean item is used as a boolean item in a relational statement.

- 139 Over 200 error messages have been given by GEN2. Correct these and try again. Compilation is terminated.
- 140 The number of procedure calls exceeds the capacity permitted by GEN2.
- 141 The number of procedure declarations exceeds the capacity of GEN2.
- 142 Procedures in the program are used recursively, or procedures are called from within procedures to a depth of more than twenty levels.

Note Statements are examined for an operand-operator-operand-operator... pattern. Brackets and separators are determined from context to be either operands, operators, or null. Errors 150, 151, 153, 154 are mentioned when a break is encountered in this pattern.

- 150 A sequential operator, declaration, I/O command, or a BEGIN, END, START, or TERM was found instead of an expected operand.
- 151 A separator, logical operator, relational operator, arithmetic operator \* or /, or a subscript bracket, or odd parenthesis was found instead of an expected operand.
- 153 A constant, odd parenthesis, unary minus sign, or absolute value modifier was found instead of an expected operator.
- 154 A statement label, file, item, procedure, subscript, sequential operator, declaration, I/O command, BIT/BYTE modifier, direct code, direct assign, or a BEGIN, END, START, or TERM was found instead of an expected operator.

#### 4. Errors Found by the Generator in Lexington Only

The error numbers listed below are produced by the first phase of the generator on the Philco 2000 computer in Lexington.

- 36 Too many COMPOOL identifiers are used.
- 92 The COMPOOL format is illegal.
- 93 Use of an identifier in the program is not compatible with its definition in the COMPOOL.
- 94 An item, table, or file name was not found in the COMPOOL.

### 5. Error Detection in the Translator on the Q7

Error messages produced by the translator on the Q7 contain a reference number between 200 and 300. The list below contains the explanations corresponding to these numbers.

Messages 200 to 224 arise during processing of the dictionary. They are printed before the listing of the dictionary and refer to the appropriate dictionary entry number. Following that, any error message from 225 and up are printed. They will contain references to the intermediate language or other internal tables. The dictionary, which contains the descriptions of all variables, labels, and constants, is printed next.

- 201\* The dictionary capacity was exceeded.
- 202\* The overlay capacity was exceeded.
- 203\* The identifier string capacity was exceeded.
- 204 A floating type item was detected.
- 205 The name of the I/O device is unrecognizable.
- 206 A table with no items was detected.
- 207 A specified table with item overlays was detected.
- 208 There is an error in overlaid subscripted items.
- 209 An item size equals zero.
- 210 The number of characters in a constant for initial data exceeds the number of characters in the item.
- 211 A specified table was found with "number of words per entry" omitted.
- 212 A specified table was found with "number of entries" omitted.
- 213 A literal item greater than five bytes doesn't start in bit zero.
- 214 A non-literal item exceeds machine word size.
- 215 A dual item exceeds half word size.

---

\* These messages are probably due to compiler error or machine malfunction and should be reported to the JOVIAL compiler staff.

- 216 A literal array variable exceeds five bytes.
- 217\* The class of a dictionary entry is unknown.
- 218 The number of parameters exceeds the number of entries of a table.
- 219 (not used)
- 220 The product of the first two array dimensions exceeds 1024.
- 221 The number of parameters exceeds the number of array elements.
- 222 Type of a constant is unknown.
- 223 Illegal octal constant was found.
- 224 Array storage exceeds 32,768.
- 225\* Internal error - length of intermediate language table is unknown.
- 226\* The TERM operator is missing in the intermediate language.
- 227\* Internal error - intermediate language table is too big.
- 228\* Internal error - operators are missing in the intermediate language.
- 229 An illegal subscript was found with an item.
- 230\* Too many unique array subscript combinations were found.
- 231 There are too few or too many subscripts on an array variable.
- 232\* Too many subscripts are used (declared with FOR statements).
- 233 Too many arrays are declared (total number of dimensions exceeds 100).
- 234 "BIT or BYTE" modifier was found on a subscript (declared with a FOR statement).
- 235 A subscript declared by a FOR statement was not accepted by the generator.
- 236 A subscript declared by a one- or two-factor FOR statement is not used.
- 237 A subscript is used incorrectly.

---

\* These messages are probably due to compiler error or machine malfunction and should be reported to the JOVIAL compiler staff.

- 238 Illegal use of modifier was found.
- 239 There are too many uses of a FOR-statement subscript.
- 240\* Internal error - transfer table was exceeded.
- 241 Too much machine language code (DIRECT code) is used.
- 242\* Internal error - unrecognizable operator was found in intermediate language.
- 243 A switch is in error.
- 244\* Translator error - statement analysis phase.
- 245\* Translator error - statement expansion phase.
- 246\* Translator error - premature end of statement analysis.
- 247 (not used)
- 248\* Internal error - operand is missing in the intermediate language.
- 249 Variable is illegally subscripted.
- 250\* Translator error - item subscript analysis phase.
- 251\* Translator error - index register assignment for the statement.
- 252 An array has too many dimensions (more than 10).
- 253  
to (not used)  
257
- 258\* Internal error - Statement Analysis Table was exceeded.
- 259\* Internal error - Data Analysis Table was exceeded.
- 260 A dummy parameter and an actual parameter are incompatible for this procedure call.
- 261 The number of actual input parameters does not agree with the number of dummy input parameters.

---

\* These messages are probably due to compiler error or machine malfunction and should be reported to the JOVIAL compiler staff.

- 262 The number of actual output parameters does not agree with the number of dummy output parameters.
- 263 A constant "BYTE" modifier specifies a segment of a literal item which is outside the item.
- 264 "BIT" or "BYTE" modifier contains a negative constant bead.
- 265 The statement causes loss of all significant bits of an operand.
- 266 Possible error - "BIT" or "BYTE" modifier specifies zero bits.
- 267 There is an illegal assignment or exchange operation. It may be a constant due to error 266.
- 268 Possible error in assignment - there is one dual operand.
- 269\* Internal error - address modification table was exceeded.
- 270 A constant used in a relational statement exceeds the maximum magnitude of the variable involved.
- 271 Multiplication by zero or one was detected.
- 272 Division by zero or one was detected.
- 273\* Translator error - statement analysis and arithmetic scaling phase.
- 274 A variable used in an arithmetic expression is larger than fifteen magnitude bits.
- 275 Exponentiation involves a negative base raised to a fractional power.
- 276 A dual constant is used as an exponent.

#### 6. Error Detection in the Translator on the CDC 1604

The following list contains explanations of the error message numbers printed by the 1604 translator. The error messages also refer to the JOVIAL statement which is erroneous. Message numbers between 200 and 300 refer to direct code.

\* These messages are probably due to compiler error or machine malfunction and should be reported to the JOVIAL compiler staff.

- 201 MTERM contains more than 5 digits.
- 204 "DEC" is illegal; (a) too many digits; (b) non-numerical characters.
- 205 "BSS" is illegal. Non-numerical characters exist in MTERM.
- 206 "OCT" contains more than 16 digits and/or 8 or 9 in MTERM.
- 207 The LOCN or MTERM has more than 6 characters (symbolic).
- 208 Duplicate tags. The tag referenced by the error message has been compared against all of the identifiers in the main program that are defined in the dictionary, and a match was found. For example, a main program item (e.g., AA) defined in the dictionary would prohibit the use of a tag with the same label (i.e., AA) in the direct code. This is also true if the item and the direct code tag are local to the same procedure.
- 301 BIT or BYTE size greater than the item size was found.
- 302 A subscript is used outside of the range of its activating FOR statement.
- 303 The FOR statement to initialize a subscript is in the range of a FOR statement which initialized the same subscript.
- 304\* SIGN of an unsigned item was found.
- 305 An impossible subscript computation is attempted.
- 306\* ILD is not equal to 2 for a subscript.
- 307 A subscript range is partially within a procedure.
- 308 Too many machine language subroutines are used.
- 309 An illegal I/O unit is referenced.
- 310 An illegal I/O status request is made.
- 311 An attempt is made to read or write part of a multiple block parallel table.
- 312\* Unpacking of the IL is out of phase.

\* These messages are probably due to compiler error or machine malfunction and should be reported to the JOVIAL compiler staff.

313 An ungrammatical structure was detected.

316 A procedure call is used without a corresponding procedure declaration.

#### 7. Error Detection in the Translator on the Philco 2000

Following the generator on the 2000, there are two phases of the compiler known, together, as the translator. Following the translator the TAC assembly program operates. Part of the compiler output is the Code Bbit listing from TAC. Errors detected by the first phase of the translator cause messages to be listed as TAC comments interspersed with the TAC data. The comments with their meanings are listed here.

\*\*\*\*\*OVERLAY ERROR DICT. CH. xxxxxxxx \$

A subscripted item occurs in an OVERLAY declaration or an identifier which has appeared in a previous OVERLAY declaration is not the first identifier in the current overlay declaration. Referring to the cited dictionary channel (entry) will give the name of the identifier in question.

\*\*\*\*\* Name of array identifier ARRAY \$

An array has been declared. This is not permitted in this version.

\*\*\*\*\* Name of dual identifier DUAL ITEM \$

A dual item, not permitted in this version, has been declared.

\* Dual constant DUAL \$

A dual constant, not permitted in this version, has been used.

\*\*\*\*\* NEXT OVER 8191 \$

This comment will precede the data for a table declared to have more than the stated number of entries. Storage will be allocated for tables with more entries, but FOR loops involving these tables may not have correct tests generated. In Lexington the limiting number is 16383.

\*\*\*\*\* SIZE OVER 8191 \$

This comment will precede the data for a literal item declared to have more than the stated number of characters. Storage will be allocated for literals with up to 32767 characters, but their size may be considered modulo 8192 for use in the STRING routines, when referencing the item without the BIT/BYTE modifier(s). The use of these modifiers on the long literal item with variable beads will cause the STRING routines to utilize the bead values, and not the dictionary value of the operand's size (13-bit maximum). In Lexington the limiting size is 16383.

Errors detected by the second phase of the translator on the 2000 cause a jump to be generated to a unique undefined label for each kind of error detected. These labels will then appear among the temporaries listed at the beginning of the TAC Code Edit. Their header will be that of the program or procedure in which they occur. This should distinguish them from harmless temporaries such as ZVAROUT.AREA which are sometimes caused by the translator. If a program is run without correction, these errors will cause a return to SYSERR.

Calls to undefined procedures and jumps to undefined labels are not detected by the translator, but these labels are listed by TAC among the temporaries and jumps to them are flagged as possible errors in the Code Edit.

Following is a list of the undefined labels which may occur:

- 2I01 An illegal hardware name has been used in a file declaration.
- 2I02 An attempt has been made to open the CARD file as an output file.
- 2I03 An illegal operand has been used in an input-output statement. For the full list see FN-5530/001/00, page 5.
- 2I04 Maximum record size for a DRUM file and an operand used in a DRUM file input-output statement are both greater than 4095 words.
- 2I05 Maximum record size for a TYPE file and an operand used in a TYPE file input-output statement are both greater than 9 words.
- 2MANYTEM Too many temporary storages are required by the program. The translator "T-REG" table with 15 entries has been exceeded.
- 2MANYIR Too many index registers are required for the single letter subscripts which are active at the same time. Compiler tables allow for 14 index registers, real and pseudo, for this purpose.
- 2TESTERR In a descending FOR loop with a variable C factor, the FOR subscript indexes an item in a table which starts above 8191 in memory. (The test generated to detect when the index register becomes negative is not valid.) The limit is 16383 in Lexington.

**2ERRBITE** An operand having a BIT or BYTE modifier has an undefined subset (one or both beads are integral constants whose value is outside the domain of the operand).

**2BIGITEM** A Hollerith or STC item of more than 8 characters is used in an item switch.

**2BIGCONS** A Hollerith or STC constant of more than 8 characters is used in an item switch.

**2CONERR** A negative, floating constant is used in an item switch. Negation will not be recognized.

**UNCLASSIFIED**

System Development Corporation,  
Santa Monica, California  
JOVIAL FOR THE DILETTANTE AND BEYOND:  
COMPILER ERROR DETECTION LISTS.  
Scientific rept., TM-555/063/00, by  
M. H. Perstein. 2 January 1963,  
19p.

Unclassified report

DESCRIPTORS: Digital Computers.  
Machine Translation.

Presents the latest generator and  
translator error lists for J3

**UNCLASSIFIED**

---

version on AN/P8Q-7, CDC 1604 and  
Philco 2000 computers. Reports  
that the first two phases of each  
JOVIAL compiler comprise the generator,  
and that the generators in use on all  
SDC computers are very similar. Further  
reports that beyond the generator phases  
there is a great deal of difference  
between compilers. States that many  
errors will be detected by the compilers.  
Explains the meanings of error messages.

**UNCLASSIFIED**

**UNCLASSIFIED**