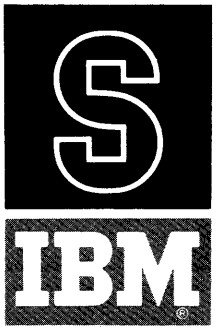


709/7090 DATA PROCESSING SYSTEM BULLETIN

PRELIMINARY REFERENCE MANUAL

IBM 709/7090 COMMERCIAL TRANSLATOR PROCESSOR

This is the second distribution of material which will constitute a reference manual for the 709/709 Commercial Translator Processor. This distribution replaces all previous publications concerning the 709/7090 Commercial Translator Processor. The material augments the information contained within four other IBM publications:

General Information Manual: IBM Commercial Translator
Form F28-8043

Commercial Translator Addenda Bulletin
Form J28-8072

Reference Manual: 709/7090 Input/Output Control System
Form C28-6100-1

IBM 7090 Operating Systems: Basic Monitor (IBSYS)
Form J28-8086-0.

Address comments regarding this publication to
IBM Applied Programming Publications, 1271 Avenue of the Americas, New York 20, N. Y.

TABLE OF CONTENTS

SECTION 01

DOCUMENTATION

INTRODUCTION

This section contains notation conventions used in this document and a table listing the distribution date, the distribution number, and the current pages of each section.

NOTATION CONVENTIONS

Throughout this document the following notation is used:

1. Material in square brackets [ ] represents an option which may be included or omitted at the user's choice.
2. Material in braces { } indicates that a choice of the contents is to be made.
3. Upper case letters must be present in the form indicated, if used.
4. Lower case words represent generic quantities whose value must be supplied by the user. These may be names, integer amounts or alphameric literals.
5. The order in which options are exercised on the various cards is not critical unless otherwise specified.

01.02 CURRENT PAGES

PAGES	DISTRIBUTION *	PAGES	DISTRIBUTION *
00.00.01	2	02.06.10	2
00.00.02	2	02.06.11	2
01.00.00	1	02.06.12	2
01.01.01	1	02.06.13	1
01.02.01	2	02.06.14	1
01.02.02	2	02.06.15	2
01.02.03	2	02.06.16	2
02.00.00	1	02.06.17	1
02.01.01	2	02.07.01	1
02.01.02	2	02.07.02	1
02.02.01	2	02.07.03	2
02.03.01	1	02.07.04	2
02.03.02	1	02.07.05	2
02.03.03	2	02.07.06	1
02.03.03.01	2	02.07.07	1
02.04.01	1	02.07.08	1
02.04.02	2	02.07.09	2
02.04.02.01	2	02.07.10	1
02.04.03	2	02.07.11	2
02.04.04	2	02.07.12	2
02.04.05	2	02.07.13	2
02.04.05.01	2	02.07.14	1
02.04.06	2	02.08.01	1
02.04.07	2	02.08.02	1
02.04.07.01	2	02.08.03	1
02.05.01	2	03.00.00	1
02.05.02	2	03.01.01	2
02.05.03	1	03.01.02	2
02.05.04	2	03.02.01	2
02.05.05	2	03.02.02	2
02.05.06	2	03.02.03	1
02.05.07	2	03.02.04	1
02.06.01.01	2	03.02.05	2
02.06.01.02	1	03.02.06	1
02.06.02	1	03.02.07	1
02.06.03	2	03.02.08	1
02.06.04	2	03.02.09	2
02.06.05	2	03.03.01	2
02.06.06	2	03.03.02	1
02.06.07	1	03.03.03	1
02.06.08	1	04.00.00	2
02.06.09	1	04.01.01	2

* (1 is dated 10/61; 2 is dated 01/62)

01.02 CURRENT PAGES (Cont'd)

PAGES	DISTRIBUTION *	PAGES	DISTRIBUTION *
04.01.02	1	06.04.05	1
04.01.03	2	06.04.06	1
04.01.04	1	06.04.07	1
04.01.05	1	06.04.08	1
04.01.06	1	06.04.09	1
04.01.07	2	90.01.00	1
04.01.08	1	90.01.01	2
04.02.01	1	90.01.02	2
04.02.02	2	90.01.03	2
04.02.03	2	90.01.04	2
05.00.00	2	90.01.05	2
05.01.01	2	90.02.00	2
05.02.01	2	90.02.01	1
05.03.01	2	90.02.02	1
05.03.02	2	90.02.03	1
05.03.03	1	90.02.04	1
05.04.01	1	90.02.05	1
05.05.01	2	90.02.06	1
05.06.01	1	90.02.07	1
05.06.02	2	90.02.08	1
05.06.03	1	90.02.09	1
05.06.04	2	90.02.10	1
05.06.05	2	90.02.11	2
05.07.01	1	90.02.11.01	2
06.00.00	2	90.02.12	1
06.01.01	1	90.02.13	2
06.02.01	2	90.02.14	1
06.02.02	2	90.02.15	1
06.02.03	2	90.02.16	1
06.02.04	2	90.02.17	1
06.02.05	1	90.02.18	1
06.02.06	1	90.02.19	1
06.03.01	2	90.02.20	1
06.03.02	2	90.02.21	1
06.03.03	2	90.02.22	1
06.03.04	2	90.02.23	1
06.03.05	2	90.02.24	1
06.04.01	2	90.02.25	1
06.04.02	2	90.02.26	1
06.04.03	2	90.02.27	1
06.04.04	1	90.02.28	1
		90.02.29	1

* (1 is dated 10/61; 2 is dated 01/62

01.02 CURRENT PAGES (Cont'd)

PAGES	DISTRIBUTION *
90.02.30	1
90.02.31	1
90.02.32	1
90.02.33	1
90.03.00	1
90.03.01	1
90.03.02	1
90.03.03	1
90.03.04	1
90.03.05	1
90.03.06	1
90.04.00	2
90.04.01	2
90.04.01.01	2
90.04.01.02	2
90.04.01.03	2
90.04.01.04	2
90.04.02	2
90.05.00	1
90.05.01	1
90.05.02	1
90.05.03	1
90.05.04	1
90.05.05	1
90.05.06	1
90.05.07	1
90.05.08	1
90.05.09	1
90.05.10	1
90.05.11	1
90.05.12	1
90.05.13	1
90.05.14	1
90.05.15	1
90.05.16	1
90.05.17	1
90.05.18	1
90.05.19	1
90.05.20	1
90.05.21	1
90.05.22	1
90.05.23	1
90.05.24	1
90.05.25	1
90.05.26	1
90.05.27	1
90.05.28	1
90.05.29	1
90.05.30	1
90.08.00	2
90.08.01	2
90.08.02	2

* (1 is dated 10/61; 2 is dated 01/62)

SECTION 02.00

COMPILER

INTRODUCTION

The 709/7090 Commercial Translator Compiler under the direction of the Commercial Translator Supervisor analyzes Commercial Translator Data, Procedure and Environment Descriptions and CRYPT symbolic machine language instructions and produces a relative binary object program deck. Description of the control cards which direct compilation, discussion of compiler output, amplification of source language rules, and a general discussion of input/output facilities are presented in this section. Discussion of compiler operation appears in section 05 and a detail description of object code generated by the Compiler appears in Appendix 90.02. Current compiler restrictions, limitations and/or deferred features are found in Appendix 90.01.

02.01 Compiler Control Cards

A. $CMPLE Card

A $CMPLE card must precede each source language program. It is recognized by the Commercial Translator Monitor (CTM) for purposes of operational control and interpreted by the Commercial Translator compiler to initiate processing of source language statements. The general form of the $CMPLE card is:

1 - 6	8 - 13	16			55 - 72
$CMPLE	deck.name	[NODECK]	[,LIST]	[,DICT]	[secondary.identifier]
		[,LOAD]	[,LOGIC]	[,FILES]	
		[,MAP]	[,NOGO]		

where

deck.name — is the primary deck identifier composed of six or less characters chosen from the set appropriate for use in CT names. The name may begin in any of the positions but must not include imbedded blanks. Leading blanks are ignored by the compiler.

The complete deck.name is punched in columns 1-6 of all Loader symbolic control cards in the generated object deck for use in cross referencing when multiple decks are combined at load time.

The options used must not be separated by blanks as the first blank terminates the list of options. The options must be separated only by commas.

[,NODECK] — instructs the Compiler to omit punching of an object deck. Normally, an output deck is produced except when a severe error has been encountered during compilation. The severity value (code) is used to determine whether a deck will be produced when NODECK is not specified. If the severity code is 5, a deck will not be produced.

[,LIST] — instructs the compiler to produce a symbolic listing of the generated instructions. Normal compiler action produces no symbolic list.

[,DICT] — instructs the compiler to list the dictionary giving the relative locations of all names used in the object program. Normally a dictionary is not listed.

[,LOAD]	instructs the compiler to write the object program on a system utility tape and turn control over to the Loader for loading the program after compilation. A map will be produced if the LOGIC option has been specified. The program will be executed unless the NOGO option has been taken, or the Compiler has encountered a source program error with a severity code greater than 1 or an undefined symbol in the code which it has generated.
[,LOGIC]	instructs the Loader to list the origin and extent of all program sections, system subroutines required for execution (including IOCS) and buffer assignments. The LOAD option is automatically initiated by LOGIC and unless the NOGO option is specified the program will also be executed.
[,FILES]	requests that a list of the I/O unit assignments made by the Loader be output on SYSOU1 whether or not the program is to be executed.
[,MAP]	is an option yet to be specified for obtaining additional load-time information.
[,NOGO]	instructs the Loader not to allow execution of the program after loading it. This does not inhibit the LOGIC and FILES options.
[secondary.identifier]	provides the Compiler with identifying information to be printed in heading of the output listing prepared on SYSOU1. It is also punched in the *CTEND Loader control card. Any character may be used in any of the positions, columns 55 through 72.

B. *FINISH card

The *FINISH card delimits the extent of the source language statements to be compiled. The form of the card is:

```
7
*FINISH
```

Note that this card must be followed by an end of file (see 04.02 and 05.03).

02.02 Compiler Output

The normal expected Compiler output is:

A. A relocatable column binary deck (see Appendix 90.03 for detailed format) unless the NODECK option has been selected or a severe error has been encountered during compilation.

1. Symbolic Control cards
2. *CTEXT
3. Relative Binary Program Deck

    a) Control Break Table
    b) File Check Table
    c) Text

4. *CTEND

B. List Tape containing

1. Source Program
   The source program is listed with certain additions.

    a) First column - source program card sequence numbers.
    b) Second column - statement number assigned by the Compiler to the Procedure sentence, Data Description entry or Environment card. It is of the form xxxxx,00.
    c) Third column - body of source program cards (columns 7-72). In some cases, names generated by the Compiler, i.e., GN)nnn, are shown in the name field of Procedure statements or Data entries unnamed by the programmer.

2. Error Messages
   A complete list of all error messages output by the Compiler appears in Appendix 90.04. Cross referencing between the source program and the error message list is made throught the statement numbers assigned by the Compiler. There are from three to six digits in the statement numbers. The last two are separated from the preceding digit(s) by a comma and they tell which clause is being referenced. The digit(s) preceding the comma tell which line is being referenced. The statement number 9999,99 is an exception. It is used to reference errors which are not confined to a single source statement.

3. File list
4. Assembly listing if LIST was selected and no severe error was encountered during compilation. (see Appendix 90.02 for guide to interpretation of this listing).

C. An intermediate tape containing a deck of the form described in (A) above will be produced if the object deck is to be loaded immediately after compilation.

02.03 General Programming Considerations

A. Use of Coding Forms

1. Card serial numbers in columns 1-6 of source decks are not sequence checked by the compiler.

2. Continuation of Specifications on Multiple Lines

   a) General Rule

   Data and Environment entries using more than a single card for complete specification have a non-blank character in column 72 to indicate that the following card is part of the same entry. Continuation of Procedure text on successive lines is determined by the nature of the text; no continuation character is used.

   b) Data and Environment Names

   All imbedded and leading blanks in the name fields of cards associated with an entry are eliminated and the non-blank characters are compressed to form a single name.

   c) Determination of Break Points

   In all sections each word or literal must be complete upon a line since the processor assumes a blank following column 72 of Procedure lines and replaces the contents of column 72 with a blank in Data and Environment lines. The only exception to this rule is introduced by the manner in which the 7090 processor handles literals in the Data Description. Literals in this section which are continued on multiple lines in violation of the rules given on page 83 of the General Information Manual are handled correctly. Use of this characteristic of the 7090 processor should not be made if compatibility with other processors is desired.

   d) Continuation of Data Description Entries

   The name and description fields only of Data Description entries may be continued. All other specifications to be made for an entry must be made on the first card of that entry since these fields are not scanned on continuation cards.

3. Termination of Statements and Entries

   a) Procedure statements are terminated by the first period (.) followed by a blank. Any information following this period blank is considered to be commentary.

b) Data and Environment entries are considered complete when column 72 is blank. The period (.) must not be used to signal completion. In these sections, a period is considered part of the previous word, thus creating an undefined name.

B. Key Words

Many of the reserved Commercial Translator words are usable as Procedure, Data and Environment names in certain situations as shown in the following key word lists. However, it is strongly recommended that the entire group be avoided altogether.

1. The following words are always interpreted as Key words and may not be used as programmer names in any division.

BEGIN	LOW.VALUES
FILE	ON
FOR	RECORD
HIGH.VALUE	WHEN
HIGH.VALUES	ZERO
IN	ZEROS
LOW.VALUE	

2. The following words may not be used as Data or Procedure names:

ABS	END	INCLUDE	QUANTITY
ADD	ENTER	IS	RUN
ALL	EQUAL	LESS	SECTION
AND	EQUALS	LIBRARY	SET
AT	EXACTLY	LOAD	STOP
BLANK	FILES	LT	THAN
BLANKS	FROM	MOVE	THEN
CALL	GET	NOT	TIMES
CLOSE	GIVING	NOTE	TO
COMMERCIAL	GO	OPEN	TR
CORRESPONDING	GREATER	OR	TRANSLATOR
CRYPT	GT	OTHERWISE	TRUNCATED
DISPLAY	HERE	OVERFLOW	USING
DO	IF	OVERLAP	WITH

3. The following key words may be used as Procedure and Data names providing it is not necessary to reference the procedure or data items in the Environment Division:

ACTIVITY	COM	LOW	SEQ
BCD	CONSERVE	MULTI	SERIAL
BINARY	CONTROL	NO	SPACE
BLOCK	DEFER	OPENCOUNT	SPANS
BLOCKSIZE	ERROR	OPENF	TAPE
BUFFERCOUNT	FIND	OPENW	THROUGH
CARD	HIGH	OUTPUT	TIME
CHECKC	HOLD	PLACE	UNIT1
CHECKF	INPUT	PRIMARY	UNIT2
CHECKPOINT	KEYS	REEL	WORD
CKSUMS	LABEL	RETAIN	
CLOSER	LABELN		
CLOSEW	LABELS		
COLLATE	LENGTH		

C. Name Qualification

Qualified names may not be used in the Environment Description or in CRYPT instructions. Data and Procedure names used in these sections must be of one word only. The verb CALL enables the programmer to reduce a qualified name to a one-word name to fulfill this requirement.

D. Effect of Data Storage Mode on Arithmetic Efficiency

Arithmetic operations are performed only in the internal (binary) mode. For this reason it is advantageous to have fields used in arithmetic operations in the internal mode when the programmer has a choice. If the fields are in the external mode and reoccur in different arithmetic statements, it is generally more efficient to MOVE this data to an area which has been defined with the mode as internal. This area would be referenced when the fields are used in arithmetic statements.

Consider the following examples:

Assume the following Data Description

A	01	999
B	01	99
C	01	99
D	01	999
E	01	99
X	01	IR999

1) The sequence

```
SET A = B+C
SET D = A+E
```

could be made arithmetically more efficient by the sequence

```
SET X = B+C
SET D = X+E
```

This assumes of course, that A is not to be used elsewhere in its external form.

2) The sequence

```
SET A = B+C
SET D = B+E
```

can be improved by the sequence

```
MOVE B to X
SET A = X+C
SET D = X+E
```

02.04 Procedure Description Clarification and Amplification

A. Constants and Literals

1. Figurative Constants

a) Values of HIGH.VALUE and LOW.VALUE

HIGH.VALUE will be considered to be the left parenthesis, (, and LOW.VALUE the zero,0, unless the Commercial collating sequence (COM) is specified in the Environment Description. The Commercial HIGH.VALUE is 9 and the LOW.VALUE is blank.

b) Figurative constants used in comparisons

ZERO may be compared to either numeric or alphameric fields.

HIGH.VALUE, LOW.VALUE and BLANK may be compared to alphameric fields only.

c) Use of figurative constants with MOVE and SET verbs.

Figurative constants may be used as source fields by both MOVE and SET. Two restrictions on the length of the target fields should be noted.

i. Figurative constants may not be moved to variable length arrays, i.e., to fields with which QUANTITY IN is associated signalling that the field length is determined at execution-time. However, figurative constants may be moved to a particular element of a variable length array.

For example, for the following data description entries:

	Lev	Quant	Description
ARRAY	01		
FIELD	02	100	A(6) QUANTITY IN NNN

the sentence MOVE BLANKS TO ARRAY is illegal, whereas MOVE BLANKS TO FIELD(3) is proper.

ii. Figurative constants may not be moved to fields which are longer than $2^{15} - 1$ characters.

The following chart shows the result of MOVEing and SETting figurative constants into variously defined target fields. (See 02.05 for description of the target fields).

Figurative Constant \ Target Field	Alpha-meric	External Decimal	Internal Decimal	Edited Field	Floating Point	Scientific Decimal
BLANK or BLANKS	blanks	blanks*	0's*	blanks*	0's*	blanks*
ZERO or ZEROS	0's	0's	0's	0's Edited	0's	0's Edited
LOW.VALUE or LOW.VALUES	0's# or blanks	0's #* or blanks	Illegal*	0's #* or blanks	Illegal*	0's #* or blanks
HIGH.VALUE or HIGH.VALUES	('s # or 9's	('s #* or 9's	Illegal*	('s #* or 9's	Illegal*	('s #* or 9's

* An error message is given for each doubtful or illegal usage.
\# Value dependent upon collating sequence order specified (Commercial or 709).

2. Literals.

The form of floating point literals used in Procedure statements is the standard scientific decimal form:

fraction F±exponent

The following rules apply to this form:

a) Both fraction and exponent may be signed or unsigned (a positive value is assumed in the absence of specification).
b) A decimal point must be present in the fraction (20.F+01 is interpreted as a floating point literal, but 20F+01 is interpreted as an arithmetic expression).
c) It must be followed by at least one digit which may be signed or unsigned (20.F0).
d) F indicates that the literal is to be converted to a single precision floating point number, and FF signals double precision floating point.

B. Verbs

1. ENTER

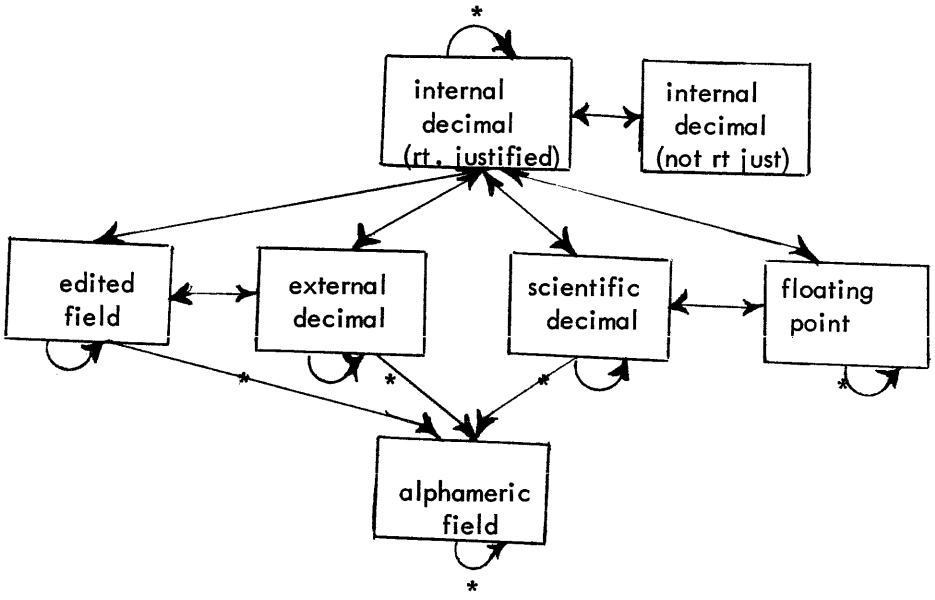
There are only two forms which this command may take: ENTER CRYPT and ENTER COMMERCIAL TRANSLATOR. The first form signals the processor that the following instructions are to be processed by CRYPT, the SCAT-like machine symbolic assembler of Commercial Translator. The second form of the command terminates the processing of CRYPT instructions and signals that Commercial Translator statements follow.

2. DISPLAY

The display device for the 709/7090 processor is the on-line 716 printer. Note that the portion of the material to be displayed which is written enclosed in quotation marks actually is an alphameric literal, which may not exceed fifty characters in length and must be complete upon a single line.
A check is made to determine whether the internal form of the sentence to be displayed is more than 255 words of storage. If it is, the entire literal will be printed but it will be printed as more than one physical record.

3. MOVE

The following chart presents a schematic of the steps involved in data movement, and intermediate forms which the CT processor causes data to assume automatically in the process. (See section 02.05 for description of field types).



a) When source and target fields have the same format no intermediate form is necessary (except for edited fields and internal decimal fields not right justified) and transmission is direct as indicated by the self-directed arrows shown with each type of field.

b) When source and target fields of unlike format are involved the conversion steps are defined by the diagram arrows where the route requiring least intermediate steps is used, e.g.,

i Movement of data from a floating point field to an external decimal field requires an intermediate conversion to internal decimal form.

ii Movement of floating point data to an alphameric field requires conversion to scientific decimal form prior to transmission to the alphameric target field.

iii Movement of internal decimal data to an alphameric field requires a conversion to external decimal form.

c) Note, that all types of data may be moved into an alphameric field but the contents of an alphameric field may only be moved to another alphameric field.

d) Subroutines are normally used to perform the requisite conversion and transmission except where arrows have associated an *. These moves are normally performed by in-line instructions.

e) For the source fields of the scientific decimal type, a free form of data is allowed within the limits of the field. For example, a field with the pictorial -99V-99 may contain the value 1 in any of the following ways, (b represents a space):

b.01b2b	(i.e. 01×10^2)
1bb+01b	(note above scale applies when no point)
.001bb3	
b1bbbbb	(note above scale applies when no point)
1000.-3	
etc.	

4. CORRESPONDING Option with MOVE and ADD

Correspondence is determined at the lowest possible level on the basis of name only. All qualifiers must be present and identical through the level of the name itself to meet the criteria of correspondence. Note, however, that the resultant MOVE or ADD may be affected since the rules governing these commands are the same whether or not the CORRESPONDING option is exercised. Consider the command

MOVE CORRESPONDING DATA.1 to DATA.2.

a) Where the data with associated levels are

DATA.1	01	DATA.2	04
GROUP	02	GROUP	05
ITEM	03	ITEM	06
FIELD.1	04	FIELD.1	08
FIELD.2	04	FIELD.2	08

correspondence would be found at FIELD.1 and FIELD.2 level, and instructions would be generated to

MOVE DATA.1 GROUP ITEM FIELD.1 TO DATA.2 GROUP ITEM FIELD.1.
MOVE DATA.1 GROUP ITEM FIELD.2 TO DATA.2 GROUP ITEM FIELD.2.

b) Where the data with associated levels are

DATA.1	01	DATA.2	01
GROUP	02		
ITEM	03	ITEM	03
FIELD.1	04	FIELD.1	04
FIELD.2	04	FIELD.2	04

no fields would be found to correspond due to the absence of the qualifier GROUP in DATA.2.

c) Where the data with associated levels are

DATA.1	05	DATA.2	01
GROUP	06	GROUP	02
ITEM	07	ITEM	03
FIELD.1	08		
FIELD.2	08		

correspondence would be determined at the ITEM level, and an attempt would be made to generate the instructions to

MOVE DATA.1 GROUP ITEM TO DATA.2 GROUP ITEM.

Since the field DATA.1 GROUP ITEM has sub-fields, it has no format characteristics of its own and is assumed to be alphameric. If DATA.2 GROUP ITEM is a field into which alphameric information may not be legally moved, an error will be noted.

Data items referenced in a MOVE or ADD CORRESPONDING clause may be subscripted. The compiler simply appends the designated subscript to the generated instructions. For example, if in example a) above the command had been

MOVE CORRESPONDING DATA.1 TO DATA.2 (I)

and appropriate quantity values had been supplied in the Data Description the generated instructions would now be

MOVE DATA.1 GROUP ITEM FIELD.1 TO DATA.2 GROUP ITEM FIELD.1(I)

MOVE DATA.1 GROUP ITEM FIELD.2 TO DATA.2 GROUP ITEM FIELD.2 (I)

5. CALL

The (old.name) in a CALL statement must be unique and may not be subscripted. This requirement is met if the (old.name) appears only once in the Data Description of if sufficient qualifiers are used to identify it uniquely. The use of record.names should be avoided in CALL statements.

6. SET

The operands of a SET instruction which are used in an arithmetic expression may be fields of any format except alphameric (the result field may be alphameric, however, and a statement of the form SET alpha.field.1 = alpha.field.2 is allowed since no arithmetic expression is specified). Appropriate conversion is performed in all cases although arithmetic operations are considerably less efficient when performed on fields having dissimilar formats. Generally, fields frequently referenced in arithmetic expressions should be converted to internal form right justified for most efficient operation.

a) Mode of Expression Evaluation

The 709/7090 Commercial Translator System allows single or double precision arithmetic in either the floating of fixed point modes. Floating point arithmetic is the 709/7090 floating binary arithmetic. Fixed point arithmetic is a binary integer arithmetic with decimal scaling.

If a floating point or double precision number is encountered in the evaluation of an arithmetic expression, all remaining operations in the expression will be performed in floating point or double precision mode, as appropriate. Once the mode of evaluation is thus changes, the appearance of a number in the other of these modes will force subsequent evaluation to be performed in double precision-floating point mode.

b) Order of Expression Evaluation

When the hierarchy of arithmetic operations in an expression is not explicitly specified by the use of parentheses, operators are honored in the following order:

TR or ABS or - (Negation)
**
* or /
+ or -

For example, the expression

A+B/C+D**E*F

will be taken to mean

(A+(B/C)) + ((D**E)*F)

When there are two operators of the same hierarchy, ordering proceeds from left to right

For example, the expression

A*B*C

will be taken to mean

(A*B)*C

Parentheses may be used to specify an exact ordering of operations within an arithmetic expression.

For example, the expression

A*(B*C)

will be taken to mean

(A*(B*C))

7. GET and FILE

For discussion of these two verbs, refer to section 02.07.

8. CLOSE ALL FILES

This command causes each open file to be closed in accordance with the close options exercised on the Loader *SPEC card. Section 03.02 describes the options as they appear on the *SPEC card which may be supplied by the programmer or generated by the Compiler from an Environment SPECIF card (Section 02.07). In the absence of specification, closing includes a rewind and unload of the unit.

Files closed by this form of the CLOSE command may not be subsequently reopened by the program.

9. STOP RUN

A STOP RUN instruction must be included in each program to provide for transfer of control to the CT Supervisor at conclusion of execution of the object program. All open files are closed prior to this transfer of control as if a CLOSE ALL FILES had been supplied.

C. Conditional Statements

1. Comparisons may not be made between numeric and alphameric fields. If ABC is described as three position alphameric field (AAA) and DEF as a three position numeric field (999) the following IF statement represents an invalid comparison:

IF ABC = DEF THEN ---

The implied or explicit specification of DEF as an external decimal field does not affect the validity of the comparison.

2. Comparisons of fields of unequal length

   a) Numeric fields are compared arithmetically without regard to length.

   b) Alphameric fields of unequal length are treated on the basis of the operator used in the comparison.

      i In tests for an = or NOT = condition the fields will always be found to be unequal. In the following example where fields A and B are of alphameric type with unequal length the only code generated will be a transfer to D.

      IF A = B THEN GO TO C OTHERWISE GO TO D.

      IF A NOT = B THEN GO TO D OTHERWISE GO TO C.

      ii In tests for NOT greater or less conditions the lengths are made equal by right truncation of the longer field.

3. For purposes of comparison edited fields (see 02.05 for definition) are converted to pure numeric fields. They may not be compared to alphameric fields.

4. All non-format fields (see 02.05 for definition) are compared alphamerically, subject to rules stated in 2b above.

5. Variable length fields may not be compared. (A variable length field is a field which uses the QUANTITY IN option).

6. For comparison purposes zero is considered an unsigned number, even though computational sequences generated by the compiler may produce negative or positive zeros.

D. Subscripting and Indexing

A source program may refer to array elements through use of subscripted names such as A(J, K). Each unique subscripted name appearing in one or more Procedure statements causes the Compiler to generate a positional indicator which will serve to locate the array element when the program is executed. For example, each of the following subscripted names would cause the Compiler to generate a positional indicator although they refer to elements in the same array: A(J, K), A(J, K+1), A(2,3), A(2, 1). Only one positional indicator will be generated for each, regardless of the number of times each is referenced.

A positional indicator contains the location (word and byte) of the particular element being referred to by the present values of the subscripts. The value of the positional indicator is a function of the location of the array, the dimensions of the array, and the values of the subscripts. A positional indicator with constant subscripts such as A(2,3) may be a load-time constant or may change in value if the dimensions or base location of the array vary during the execution of the object program.

In general, positional indicators are evaluated when their subscripts change value. Execution of the code generated for SET K=K+2 would cause the positional indicators for 3 of the subscripted names shown above to be updated:

A(J, K), A(2, K), and A(J, K+1)

The initial value of an array is A(1) and not A(0).

02.05 Data Description Amplification and Clarification

The name of each data field referenced in Procedure statements or Environment Descriptions must appear in the name field of a Data Description entry. Neither storage allocation nor assignment of data characteristics is made for data fields not so named.

A. Level

1. Any numbers 01-99 may be used to specify the level of fields. Leading zeros are optional and all level numbers less than 10 are right justified by the compiler.

2. The highest level in a program need not be 01. All data fields of level equal to the highest level in a source program will be left justified unless explicitly specified as right justified. Consider a source program in which only the following description of data appears:

Name	Level	Type	Description
A	05	RECORD	
B	10		A(20)
C	10		9(20)
D	05		9(2)
E	05		A(3)

Fields A, D, and E will each be left justified (although no justification is specified and previous fields may not complete full words) since 05 is the highest level used in the program.

B. Type Codes.

1. RECORD

a) A record is a data hierarchy which may not be part of any data organization except a file or a section. The "Quantity columns" are used to specify the number of times a format of data is repeated within a hierarchy; the use of these columns with a data hierarchy, such as RECORD, is unacceptable. When the type code RECORD is recognized the previous data organization is always terminated. Record definition is terminated by specification of a level number equal to or less than the one associated with RECORD.

b) A record must have length, and redefinition of a record area does not give it length.

The compiler considers that no length has been specified for REC1 in the following example:

Name	Level	Type	Description
REC1	01	RECORD	
		REDEF	REC1
AREA	01		A(40)

The length of a record is normally determined on the basis of fields defined within it. Length may also be given to a record through use of a pictorial with the RECORD entry itself.

2. COND

a) Normally this type code is associated with several entries naming and describing the values a particular field may have. These entries must be preceded by a higher level entry defining the format of the fields and must all have the same level number and include the COND type code.

3. REDEF (see iii under Data Description on page 90.01.03 for limitation).

a) When the REDEF type code is used, it should appear on a line with no additional coding except a serial number and the name of the item being redefined. The first item appearing thereafter must have the same level number as the item referenced by the REDEF.

b) When a REDEF is first encountered, the contents of the storage assignment counter are saved by the compiler, and assignment proceeds according the REDEF. The effect of a REDEF is terminated upon encountering an item of a level above or equal to the level of the item referenced by the REDEF or upon encountering another REDEF. Termination of the former type causes the storage assignment counter to be restored and assignment proceeds in the normal manner.

c) Compound names are formed without regard to REDEFs (or other type codes) and as a function of the level and position within the program of the simple names only. Consider the following examples, in which the completely qualified name of H is A G H in both instances.

EXAMPLE 1 -

Data Name	Level	Type	Description
A	01		
B	02		
C	03		
D	03		
E	02		
F	03		
		REDEF	B
G	02		
		REDEF	C
H	03		

EXAMPLE 2 -

Data Name	Level	Type	Description
A	01		
B	02		
C	03		
D	03		
E	02		
F	03		
		REDEF	E
G	02		
		REDEF	C
H	03		

4. RCDMRK

The RCDMRK type code causes insertion of a single character record mark constant in the indicated position. No description field specification is required since the compiler automatically provides a pictorial with a single A. Information may be freely moved into or out of this position subject only to the restriction imposed by its single character alphameric format.

5. LABEL

The type code LABEL allows the programmer to redefine with the indicated data description entries, the single 14 word label area in the Input/Output Control System from which all labels for output files are wirtten and into which all input labels are read. This area may only be processed by the Commercial Translator programmer by using the FOR LABEL option on the Environment FILE cards. Use of the FOR LABEL option with a file causes IOCS to transfer control to the procedure.name specified with the option when the file is opened or closed, or when a reel switch occurs. Discussion of the implementation of this procedure may be found in Appendix 90.07. The type code LABEL and the FOR LABEL option provide the programmer with an IOCS oriented method of checking and forming non-standard labels. The restrictions limiting the length of labels processed in this manner to 14 words and dictating that portions of the FOR LABEL coding must be done in CRYPT may make an alternate method desireable; i.e., defining labels as records and processing with the GET and FILE verbs.

6. PARAM and FUNCT

These two type codes described in the General Information Manual are no longer in the language.

C. Quantity Field

An entry in this field is used by the compiler to reserve storage for sequential data of the same format. If no value is specified, 1 is assumed. No check against this value is made at execute time. The maximum Quantity which may be specified is $2^{15} - 1$. Since Quantity numbers are used to specify sequences of data descriptions for use of lists and tables, and since data in tables is referred to by the use of subscripted names, quantity numbers should not be assigned to data items not having names, unless these items include named items at a lower level.

D. Mode and Justify Fields

1. The MODE column of the data description coding form merits careful attention. Internal mode (I) designates that the associated numeric field is to contain data in binary form. External mode (E) designates that the stored form is BCD.

2. A numeric internal mode field designated as right justified (R) appears by itself in the low order positions of a full word (two if double precision) with the sign value in the sign bit of the word. Numeric external fields to be involved in extensive computation should be converted to this internal right justified by a preliminary SET or MOVE in order to prevent repetitive unpacking and converting.

   The length of a numeric internal mode fixed point field designated as left justified or without justification specification is the least multiple of 6 bits sufficient to contain the number and its sign (leftmost bit of the field so designated). Left justification (L), as with external fields, reserves storage beginning with the leftmost bit of a new word and extending through as many words and bits as necessary. The no justification mode (blank) differs only in that storage reservation begins immediately to the right (within the same word if possible) of the preceding storage reservation.

   For a right justified external mode field, storage is reserved beginning in a new word so that the field's last character is also the rightmost character of its final word.

3. Specification of right justification is effective only for dara items with explicitly described formats. Left justification is always effective.

E. Description Field

1. Pictorials
   a) Definition of Allowable Data Field Types
      The chart on the following page defines the types of formats which data fields may assume. The determining characteristics, allowable format description characters and proper field contents are shown for the various field types.
   b) The pictorial characters A and X are considered to be synonymous by the 7090 CT Compiler.

Type of Field	Characterized By	Legitimate Format Characters in Description	Legitimate Information in Field
Alphameric	A or X in format field	A X (n)	all characters
External Decimal (1) (3)	E in Mode Column	9 (n) S V $\bar{9}$ or $\overset{+}{9}$ in rightmost character	digits and leading blanks an overpunch with the rightmost digit
Internal Decimal	I in Mode Column	9 (n) V S	binary
Edited Field (2)	8 * . , $ + - in format field or BLANK WHEN ZERO clause	9 8 * . , $ + - S V (n) $\bar{8}$ or $\bar{9}$ or $\overset{+}{8}$ or $\overset{+}{9}$ in rightmost character	digits and leading blanks an overpunch with the rightmost digit
Floating Point	I in Mode Column and F or FF in format field	F is single precision FF if double precision	floating binary
Scientific Decimal (4)	E in Mode Column and F in format field	9 (n) F . V + -	digits . + -

Notes -

(1) The only sign specification which may be used for an external decimal field is an overpunched + or - in the rightmost position of the field. A + and - may not appear in the character by itself.

(2) An edited field may have a character position reserved for the sign or may have the sign over the rightmost digit.

(3) Numeric external fields may contain leading blanks which are treated as leading zeros.

(4) Scientific decimal is the edited form of the floating point. The maximum fractional portion of a scientific decimal field is 16 digits.

c) A field for which no pictorial is given is treated as alphameric with length equal to the length of its subfields.

No subfields may be specified for the field whose format is described. The only entry which may legitimately appear at a lower level is a COND definition.

In the following example A and D are treated as alphameric fields with lengths of 12 characters each.

Name	Level	Mode	Justify	Description
A	01			
B	02	I	R	99
C	02	I	R	99
D	01			
E	02			A(6)
F	02			A(6)

d) Fixed point double precision numbers are denoted in the Data Description by formats representing more than 10 digits. Double precision internal floating point numbers are specified by "FF" in the Data Description.

e) Any non-format character appearing in the pictorial will result in the pictorial being interpreted as a name. An error message will be produced if the pictorial is not a data, key, or a procedure name.

2. Constants

a) Constants cannot be defined.

i in a field whose pictorial characterizes it as an edited field.

ii as a part of a located input area. (see section 02.07).

iii following a variable length field.

iv as part or all of the redefinition of an area.

b) No format specification is required in the definition of alphameric constants. The length of the literal is assumed as the length of the field. However, if a pictorial is furnished and the length given by the pictorial is:

i greater than the length of the constant the constant is left-justified in the field and the remaining positions of the field filled with blanks.

ii less than the length of the constant the characters of the constant starting with the left end are placed in the field until the field is full; at which time the remaining characters are discarded. Also, an error message is produced.

c) If a field is defined as external decimal the length specified by the pictorial must be exactly equal to the length of the constant. Also, the constant must utilize the sign convention given in the pictorial e.g.,

If the pictorial is 999^{+}, the number 123^{+} would be correct whereas 123 would not be correct

d) If a field is defined as internal decimal the length of the field specified in the pictorial may be larger than the constant. In this case the constant is right justified in the field. If the constant is larger, the constant is truncated at the left-end to the size specified by the pictorial, converted and stored. Also, an error message is produced.

The appropriate sign conventions for internal decimal constants are:

i leading + or -
ii trailing + or -
iii no sign

3. QUANTITY IN data.name

Use of this option indicates that the associated entry(s) is to be repeated a variable number of times as determined at execute time by the value in the indicated 'data.name'.

The compiler reserves sufficient storage for the maximum number of times the field may be present on the basis of the value entered in the Quantity Field.

4. BLANK WHEN ZERO

When associated with an output field this clause indicates that the field is to be replaced with blanks when it becomes zero. Since leading blanks in numeric input fields are automatically treated as zeros, the use of this clause is redundant. In some cases it may even result in decreased efficiency in the object program as these fields are treated as edited types.

02.06 Environment Description

A. Introduction

1. The Environment Description of the Commercial Translator language allows the programmer to specify the external physical factors which relate to the compilation and execution of the program. These factors may usually be changed without affecting the logical description of the problem, as contained in the Procedure and Data Description parts of the language. The Environment Description of a source program specifies and describes those factors which allow the processor and object program to deal with information as it exists in the "external world", including physical characteristics of data, machine configurations, and programmer-elected features.

2. In this chapter, a general discussion of the Environment Description is followed by a detailed explanation of the various specifications. Appendix 90.08 describes the effect of the various Environment specifications on the Loader symbolic control cards generated by the Compiler.

B. Environment Types

1. General Rules for Use of Coding Form
Figure 1 is a sample Environment Description coding form on which seven general types of specifications may be made. The information associated with each type is entered on one or more cards. The first card of each specification group may contain an appropriate name in the Name field (columns 7-22) and must contain one of the following seven type codes in the Type field (columns 25-30):

a) FILE
b) SPECIF
c) POOL
d) GROUP
e) CONTRL
f) OPTION
g) COND

When multiple cards are required continuation of the options (columns 31-71) or Name on subsequent cards is indicated by a non-blank character in the Continuation field (column 72). The absence of a character in column 72 signals that the next card initiates a new specification; if the next card does not contain a type specification the card is deleted and an error message given. If a card containing the continuation signal is followed by one which contains a type specification, the type is ignored and the options specified are considered to be part of the previous specification.

IBM

709/7090 COMMERCIAL TRANSLATOR ENVIRONMENT DESCRIPTION

CTL. 1 3	PROGRAM	SYSTEM	PAGE OF
	PROGRAMMER	DATE	IDENT. 73 80

SERIAL 4 6	NAME 7 22	23 24	TYPE 25 30	DESCRIPTION 31 71	CONT. 72

Figure 1

A period (.) must not be used to signal the end of a specification as it will be treated as part of the previous word thus creating an undefined symbol.

2. General Function of Specification Types

FILE cards	The FILE environment card must be included to describe the file characteristics which affect compilation, such as use for input or output and the type of blocking utilized.
SPECIF cards	The SPECIF card describes such file characteristics as symbolic input/output unit assignments, density, relative activity, labeling conventions and other programmer choices which are to be punched during compilation into load-time control cards. SPECIF cards may be omitted if the programmer prefers to provide his own cards containing the appropriate information at load-time.
POOL cards	POOL environment cards may be used by the programmer to direct the processor in the allocation of buffer storage for several files. In the absence of POOL cards, the processor will automatically make pool and buffer assignments.
GROUP cards	GROUP cards may be used by the programmer to specify buffer allocation for files belonging to a particular buffer POOL.
CONTRL cards	If several separately compiled programs are to be combined and executed as one program, any joint procedure or data description areas must be defined for the compiler by CONTRL environment cards.
OPTION cards	OPTION cards are used when the programmer wants the processor to depart from its standard use of 709/7090 collating sequence, or when he wishes special emphasis placed on minimizing either storage or running time requirements in his object program.
COND cards	The COND card names and defines a computer console key setting which the programmer may want to test by procedure statements in his program.

C. FILE Environment Card

FILE cards are used to supply the compiler with the characteristics of the file which are necessary at compile time. This specification must be made for each file processed by the program. The formats of the FILE cards and options available for input, output and checkpoint files are shown below:

1. Input Files

Name	Type	Description
file.name	FILE	INPUT [,{BCD / BINARY}] [,{CARD / TAPE}]
		,BLOCKSIZE nn
		[,ON ERROR statement.name.1]
		[,FOR LABEL statement.name.2]
		[,{HOLD / SPANS}] [,BEGIN]
		,record.name.1 nj
		[,{BLOCK CONTROL / FIND LENGTH IN data.name.1}]
		[,PLACE LENGTH IN data.name.2]
		[record.name.2 . . .]

2. Output Files

Name	Type	Description
file.name	FILE	OUTPUT [,{BCD / BINARY}] [,{CARD / TAPE}]
		,BLOCKSIZE nn
		[,FOR LABEL statement.name.2]
		[,BEGIN] [,SPANS]
		,record.name.1
		[,FIND LENGTH IN data.name.1]
		[,PLACE LENGTH IN data.name.2]
		[,PRIMARY] [,NO CONTROL WORD]
		[,record.name.2 . . .]

3. Checkpoint Files

Name	Type	Description
file.name	FILE	CHECKPOINT

4. Description of File Options and Requirements

a) The order in which options are exercised on the FILE card is critical only in that the options exercised for a particular record must be listed following the record.name and prior to the introduction of another record.name.

b) Detail Description of FILE options and requirements

{INPUT / OUTPUT / CHECKPOINT}

The description of the file must specify one of these three types. If a file is designated as CHECKPOINT it may have no other usage.

[, {BCD / BINARY}]

The specification states the mode in which an input file is to be read or an output file written. If no specification is made the file is assumed to be BCD.

[, {CARD / TAPE}]

CARD is specified if the on-line card reader or card punch is the processing unit. When CARD is specified BEGIN is assumed, i.e., each record starts at the beginning of a physical block. With this option only columns 1 through 72 are read. For the most efficient handling of card input a GROUP card should be used when more than two cards constitute a record. If no specification is made TAPE processing is assumed.

,BLOCKSIZE nn

nn is an integer representing the size of the largest block to be output or the maximum number of words to be input from an input block. This specification must be made. All input card files must have a block size of at lease 24 words. Maximum blocksize is 9999 words.

[, ON ERROR statement.name.1]

Statement.name.1 references a procedure statement name in the Commercial Translator program to which transfer will be made if the system is unable to recover from an input error. (See Section 02.07).

[, FOR LABEL statement.name.2]

FOR LABEL option provides transfer of control to statement.name.2 whenever a file is opened or closed or whenever a reel switch occurs. See Appendix 90.07 for application of this option in processing non-standard labels.

record.name

All records associated with the file must be named on the FILE card.

[, {HOLD / SPANS}]

Specification of either the HOLD or SPANS option

i For an input file: forces data to be processed in the transmit mode. This is required if records in the file overrun the boundaries of a block (SPANS); when it is required that each named record of the file be available until another of the same name is input (HOLD).

ii For an output file: specifies that output records are to be written in blocks of the specified length. This allows the existence of partial records in blocks for the sake of compactness (SPANS). Files written in this manner must be processed in the transmit mode when input.

The compiler does not differentiate between the words HOLD and SPANS. They produce the same effect and are both included in the vocabulary for mnemonic convenience.

If neither HOLDS or SPANS is selected an input file will be processed in the locate mode; an output file will be created with all records complete within blocks.

[, BEGIN]

The BEGIN option specifies that each record to be read or written by the program starts at the beginning of a physical block.

[, {BLOCK CONTROL / FIND LENGTH IN data.name.1}]

BLOCK CONTROL specifies that the input record is of variable length but has no standard control word preceding it; the logical record length is equal to the length of the block.
FIND LENGTH IN data.name.1 signals that the length of an input or output record is to be obtained from data.name.1 rather than as specified by the data description. If neither option is selected it is assumed that the variable length record is preceded on tape by the standard control word.

If both options are specified for a record, only BLOCK CONTROL will be effective.

[, PLACE LENGTH IN data.name.2]

Use of this option on input makes available in data.name.2 the length of the current record. On output the length of the current record will be placed in data.name.2 prior to filing the record.

[,PRIMARY]

The PRIMARY option enables the user to file selectively a record which is associated with more than one output file. The 'FILE record.name' command will file records belonging to more than one output file only in those files where PRIMARY has been specified for that record.name. If PRIMARY is not used with the record.name in any file, the record will be filed in all output files with which it is associated.

[,NO CONTROL WORD]

This option notifies the system not to produce the standard control word for the variable length output record.

D. SPECIF Environment Card

1. The SPECIF card is used to describe load time I/O options for the designated file. SPECIF cards are not necessary for correct compilation of a Commercial Translator program; the user may elect to punch the required load time cards himself rather than allowing the compiler to do so on the basis of SPECIF cards. The SPECIF card(s) may appear anywhere within the environment division. The general form of the SPECIF card is:

Name	Type	Description
	SPECIF	file.name [,UNIT1 'unit.1'] [,UNIT2 'unit.2'] [, {LOW / HIGH}] [, DEFER] [, OPENW] [, OPENF] [, {CLOSER / CLOSEW}] [, ACTIVITY nn] [, {CHECKC / CHECKF}] [, MULTI] [, SEQ] [, CKSUMS] [, {LABELS / LABELN}] [, SERIAL 'ser.no'] [, REEL 'reel.no'] [, RETAIN days] [, {HIGH / LOW}]]

file.name

File.name must be the first item of the SPECIF card description field. It must be identical with the file.name in the FILE card.

[, UNIT1 'unit.1'] [, UNIT2 'unit.2']

Note that the Quote Marks are mandatory. The alphameric literals 'unit.1' and 'unit.2' specify symbolically the I/O units (card or tape) to which the file is attached. The available unit specification includes relative unit assignment, symbolic channels, and tape transport model type. The secondary unit specification UNIT2 'unit.2', allows for the assignment of the secondary reel of a multi-reel file on another or the same unit which will be automatically referenced upon a reel switch. If UNIT2 is left blank, the secondary unit assignment will be the same as the primary unit. To explain these specifications the following notation is employed:

- X denotes one of the real channels A,B,...,H
- P denotes a symbolic (unspecified physical) channel S,T,...,Z
- k denotes one of the numbers 1,...,9,0
- m denotes a tape transport model number, II or IV

The allowable unit specifications are:

a) No specification - when the option is not exercised any available unit is to be assigned to the file.

b) 'm' - any available unit of this model type is to be assigned to the file.

c) 'X' -any available unit on this physical channel is to be assigned to the file.

d) 'P' - all files in the job having this symbolic channel designation are to be assigned to the same channel.

e) 'X(k)' - the kth available unit on the specified channel is to be assigned to the file; note that the parenthesis are required.

f) 'Pm' - any available unit of this model type on the symbolic channel specified is to be assigned to the file.

g) 'Pkm' - an available unit on the symbolic channel, having this model number, is to be assigned to the file. k in this use without parenthesis indicates the order of preference for the channel so that if the number of available units on the channel is less than the total requested for the channel those with lower numbers are to be assigned to the same channel.

h) System units may be assigned by:

i 'IN' - the current system Input unit is to contain the file.

ii 'OU' - the current system Output unit is to be used for a printed output file.

iii 'PP' - the current system Peripheral Punch unit is to be used for a punch output file.

iv 'UTk' - System utility tape k (non-parenthesized k = 1-4) is to be used for the file.

i) Card equipment is assigned for a file by

1) 'RDX' - card reader, channel X

2) 'PRX' - printer, channel X

3) 'PUX' - card punch, channel X

j) As a special option, an '*' in the UNIT2 field indicates that the secondary unit of a file is to be any unit on the same channel and of the same model type, which is available after all other assignments have been made. At load-time if units are available, but of a different model, one will be assigned; if no units are available on the channel, no secondary unit is assigned prior to execution of the job.

Examples

A tape to be assigned to the first available unit on channel A is specified in the SPECIF card as ,UNIT1 'A(1)'
If a programmer desires to use the system output tape he would specify ,UNIT1 'OU'
Note, that in this case (also for 'IN') included options for density, opening and closing are non-effective.
Additional and more complete information concerning unit assignment will be found in section 03.

[, {LOW / HIGH}]

LOW and HIGH specify the density of the tape. Density is assumed as HIGH if no specification is made.

[, DEFER]

This option may be exercised for files which are not required when processing is begun. Unless DEFER is specified the file must be mounted before program processing commences.

[, OPENW]

OPENW specifies that the file is not to be rewound before opening. If OPENW is not specified the file will be rewound prior to being opened.

[, OPENF]

OPENF specifies that this standard labeled file is to be found on a multi-file reel by searching forward until labeling information indicates that the proper file has been found. If this option is selected without the selection of the OPENW option the tape is rewound before searching. When OPENW and OPENF are both selected, the programmer must insure two things:

1) the tape is positioned before a header label; and
2) the labeled file to be searched for is further down the tape.

[, {CLOSER / CLOSEW}]

If the user desires that a file not be rewound and unloaded when it is closed he may exercise one of these options. CLOSER specifies that the file is only to be rewound upon closing. CLOSEW specifies that the file is not to be rewound upon closing.

[, ACTIVITY nn]

nn is a number between 1 and 99 that represents the relative activity (or usage) of this file with respect to other files. The loading-time program uses this information in the allocation of buffer areas to produce a more efficient object program.

[, CHECKC]
[, CHECKF]

CHECKC specifies that a checkpoint will be written on the checkpoint file upon reel switch of this file. CHECKF specifies that a checkpoint will be written on this file upon reel switch; the file must be a labeled output file for this option to be operative. No check point will be written if neither option is exercised.

[, MULTI]

MULTI specifies a tape file which is contained in more than one tape reel. A single reel file is assumed if MULTI is not specified.

[, SEQ]

SEQ specifies that each block in the file contains a block sequence number which is to be checked by the I/O system. (see IOCS)

[, CKSUMS]

CKSUMS specifies that each block in the file carries a checksum of data in the block which is to be checked by the I/O system. (see IOCS).

[, { LABELS }] [, SERIAL 'ser.no'] [, REEL 'reel.no'] [, RETAIN days] [, { HIGH }]
[{ LABELN }] [{ LOW }]

LABELS specifies a file with a standard label, i.e., labels to be checked or written automatically by IOCS.
LABELN indicates the use of a non-standard label, i.e., a label of 14 words or less which is to be checked by the programmer using a linkage supplied by the FOR LABEL option of the Environment FILE card. Either LABELS or LABELN must be explicitly stated if labels are to be recognized by IOCS and acted upon in either a standard or non-standard fashion. If neither option is exercised the file is considered to be unlabeled.

The SERIAL, REEL, and RETAIN options may be specified for files with either standard or non-standard labels; this information is kept in the IOCS file block and is printed on-line in certain error situations. It is used by IOCS in checking and preparing standard labels (but not non-standard labels) and may be used by the programmer in checking and preparing non-standard labels.

Note: careful study of the IOCS manual and Appendix 90.07 of this bulletin is essential for understanding standard and non-standard labels.

[, SERIAL 'ser.no']

The alphameric literal 'ser.no' is composed of 5 or less characters. Standard input labels will be checked against this serial number if a non-blank literal is specified. Standard output lables written by IOCS for this file will contain this serial number only if the REEL option specifies a 'reel.no' greater than 1; output serial numbers are normally taken from the label already present on the tape on which the first reel of the file is written.

[, REEL 'reel.no']

The alphameric literal 'reel.no' is composed of 4 or less numeric characters. Use of this option specifies the reel sequence number of the first reel of a file. When the option is not exercised this sequence number is assumed to be 1. Reel sequence number is adjusted at object time to reflect reel switching, and is checked in standard input labels.

[, RETAIN days]

The numeric literal, days, is composed of 3 or less numeric characters which specify the number of days a tape is to be retained from the date it was written.
An attempt to write a labeled file on this tape before the end of the period has expired will result in an on-line error message from IOCS. Unless a labeled file is to be written on a tape, the system assumes that no label is present on the tape.

[, {HIGH / LOW}]

LOW and HIGH appearing after LABELS or LABELN indicate the density of the label. Label dinsity is assumed to be that of the file if no specification is made.

E. POOL Environment Card

1. The POOL card enables the user to achieve a more efficient object program by specifying that certain files are to share a common buffer area. These cards are optional. If the programmer so desires, he may enter the information (normally provided in the POOL card) on an object time loading card. If POOL specifications are not provided at all, files will be grouped automatically by IOCS.

2. The general form of the POOL card is:

Name	Type	Description
pool.name	POOL	file.name.1 [file.name.2.....]
		[, BUFFERCOUNT nn]
		[, BLOCKSIZE nn]

3. Description of POOL options

[BUFFERCOUNT nn]

nn buffers will be assigned to the files in the pool by the loading program. nn must be equal to or greater than the number of files in the pool. (A further grouping of files is possible. This is described in the following section on GROUPS. When the GROUP option is used, nn must be equal to or greater than the total number of buffers assigned to groups within the pool). If BUFFERCOUNT is not specified, it will be assigned automatically by the compiler.

[,BLOCKSIZE nn]

nn is the number of words to be allocated to each of the buffers in the POOL. The POOL BLOCKSIZE should be equal to or greater than the largest BLOCKSIZE specified for any file in the pool. If BLOCKSIZE nn is not entered on the POOL card, the POOL BLOCKSIZE used will be the same as the BLOCKSIZE of the largest file in the POOL. (see FILE BLOCKSIZE)

F. GROUP Environment card

1. The GROUP card is used in conjunction with a POOL card in allocating buffer areas to be shared by files. The function of the GROUP cards is

to specify how buffers within a POOL are to be shared by the various files in the POOL. These cards are optional. The information (normally provided in the GROUP card) may be punched into a load-time control card. If GROUP specifications are not made at all, the compiler will attempt to assign at least 2 buffers to each file.

2. The general form of the GROUP card is:

Name	Type	Description
	GROUP	[pool.name]
		[,OPENCOUNT nn]
		[,BUFFERCOUNT nn]
		,file.name.1 [,file.name.2.....]

3. Description of GROUP Options

[pool.name]

The POOL to which a particular GROUP of files belongs must be specified by listing the pool.name as the first item in the variable field.

[, OPENCOUNT nn]

nn is the maximum number of files within the group which may be open concurrently during execution of the program. This count determines the minimum buffer count necessary for processing the GROUP of files. If no OPENCOUNT is given, the OPENCOUNT will be assumed equal to the number of files in the GROUP.

[, BUFFERCOUNT nn]

nn is the number of buffers which are to be assigned by the loader to this GROUP. The BUFFERCOUNT of a GROUP must be equal to or greater than the OPENCOUNT of the GROUP. If no BUFFERCOUNT is given for the GROUP, the loader will attempt to assign at least twice the OPENCOUNT number of buffers to the GROUP.

If the POOL BUFFERCOUNT or storage limitations prevent such assignment, buffers (in addition to the minimum necessary) are allocated to the GROUP on the basis of the activity of the files in the GROUP.

G. CONTRL Environment Card

1. The CONTRL cards define procedure and/or data areas common to two or more Commercial Translator programs. These programs may be compiled and checked out independently and then merged at object loading time into one running program. The general format is:

Name	Type	Description
loadnm	CONTRL	{ section.name sentence.name.1 TO sentence.name.2 record.name }

loadnm is a name of 6 characters or less which is used at object load time to equate areas in various programs defined by the environment CONTRL cards. This load time reference name must have been specified on a CONTRL card in each program referring to the area. The body of the CONTRL area may contain entirely different names in the various programs being combined.

In the form sentence.name.1 to sentence.name.2 the area defined will not include any of sentence.name.2.

2. The following restrictions apply to CONTRL areas.

a) CONTRL areas which are defined by the Data Description must begin with left justification.

b) References outside of CONTRL areas to names inside CONTRL areas are valid only if:

i The name referenced has the same bit displacement from the beginning of the CONTRL area in all programs to be merged.

ii The name referenced has the same format in all programs to be merged. Therefore, the following general rules apply to CONTROL areas:

CONTROL areas which are defined by the Data Description should be identical in organization and format up to and including the last externally referenced item.

CONTRL areas which are defined by procedure sections or sentences should be identical up to and including the last externally referenced procedure statement name. If there are no external references made to names inside the areas in any of the programs to be merged (i.e., if references are made to the section name or beginning sentence name only), no correspondence of organization is necessary inside the CONTRL area.

c) Indices inside a CONTRL area must not be referenced or modified outside of that area in any of the programs to be merged.

H. OPTION Environment Cards

1. OPTION cards may be used to set compilation mode switches within the compiler. The lowest to the highest value in the collating sequence of the 709/7090 is:

0 through 9 = ' + A through I $\bar{0}$.) - J through R $\bar{0}$ $ * blank / S through Z ≠ , (

The compiler normally generates comparison type instructions based on the 709/7090 collating sequence and an object program which is conservative of time rather than space. The OPTION card allows the programmer to indicate a selective preference for the Commercial (705) collating sequence and a time or space conservative object program. The lowest to the highest value in the commercial (705) collating sequence is:

Blank . ¤ ≠ & $ * - / , % # @ $\bar{0}$ A through I $\bar{0}$ J through R ‡ S through Z 0 through 9

The general form of the OPTION card is:

Name	Type	Description
	OPTION	[COLLATE COM [IN section.name]]
		[, CONSERVE {SPACE / TIME} [IN section.name]]

2. Description of OPTION features and requirements

[COLLATE COM]

Use of this option instructs the compiler to generate comparison type instructions on the basis of the Commercial collating sequence.

[, CONSERVE {SPACE / TIME}]

CONSERVE SPACE requests that the generated object program use as few storage locations as possible. In this mode subroutines are generated once for operations such as scaling, double precision arithmetic, etc., in-line calling sequences to the subroutines are used to perform the operations.

CONSERVE TIME requests an object program which will be executed in a minimum amount of time; this is achieved by the generation of in-line instructions to perform the operations. This is the normal mode of operation.

[IN section.name]

This option may be used with either CONSERVE or COLLATE and enables the programmer to limit the modal specification to a particular section. When this option is used with CONSERVE or COLLATE the mode of generation reverts to the normal mode at the end of the specified section. There is no restriction on the number of times these modes may be altered.

I. COND Environment Card

1. The COND card defines a computer console entry keys setting which may be interrogated by procedure statements. The general form of the COND card is:

Name	Type	Description
condition. name	COND	KEYS 'nn'

The alphameric literal 'nn' is composed of 12 octal digits representing "on" settings of the 36 console entry keys. When the condition.name is tested in a procedure statement only the keys specified in 'nn' are tested for "on" setting. The specified console keys must all be "on" for the condition to be true. No "off" or "on" test is made for the non-specified keys.

Note that Data Description condition.names are tested differently, i.e., for absolute equivalence to the specified value.

02.07 Input/Output Facilities of the 709/7090 Commercial Translator

A. Introduction

Input/Output operations are handled within the Commercial Translator system by the 709/7090 Input/Output Control System.

This section is intended to explain how data may be organized and the commands used for efficient input/output. Examples are given to illustrate usage, and appropriate references are made to the General Information Manual and to other parts of this publication.

B. Definitions

1. The Record

A record is a logical grouping of fields. (See the Data Description section of the Commercial Translator General Information manual). A record is made available for processing from an input file by the GET command. The data so obtained may then be operated on by internal commands, such as SET and MOVE. A record is placed in an output file by the FILE command.

2. The File

A Commercial Translator file is a collection of logically related records stored in an external medium, such as tape or cards. Characteristics of a file are specified in the Environment Description (02.06).

A file is referenced by the commands OPEN and CLOSE to initiate and complete file activity. (See Commercial Translator General Information manual).

3. The Block

Physical segments of data in a file are called blocks. A block may consist of any combination of records or parts of records. Four factors determine block and record relations: specified block size, whether records span the boundaries of the block, whether every record of the file begins a block, and whether each record is the block on tape (see SPANS, HOLD, BEGIN, BLOCK CONTROL and BLOCKSIZE in the Environment Description section, 02.06).

The greater the length of a block, the greater will be the percentage of useful information that can be stored on a length of tape. However, large blocks do require large storage allocations which if too large, may not result in the most efficient balance between input/output and internal processing (computing). Blocks of 200 to 500 words will probably provide for the optimum processing of master records within the Commercial Translator system.

4. Buffers

The Loader reserves a portion of core storage for use by the I/O system as operating storage. This storage area is subdivided into buffers into which all input blocks of data are read and from which all output blocks of data are written. Since the IOCS routine attempts dynamic optimization of buffer assignments, data blocks of a particular file may be located within any of the buffers associated with this or several files.

5. Locate and Transmit

In the 709/7090 Commercial Translator system it is possible to process the records of an input file in the buffers or in an area(s) reserved for them outside the buffer area; all records in a file must be processed in the same manner.

a) Locate Mode of Operation

The GET command when dealing with a record to be processed in the buffer area

i. 'locates' the next record in the file, i.e., determines the position of the record within the buffer area, and

ii. adjusts all program references to data within the record to reflect the new base reference address of the record within the buffer area.

b) Transmit Mode of Operation

The GET command when dealing with a record to be processed in an area assigned outside of the buffer area

i. locates the next record in the file, and

ii. 'transmits' the record to its assigned area. As the area assignment is fixed no adjustment of references to data within the record is necessary.

The 'transmit' mode is triggered by the selection of the SPANS, HOLD or CARD options on the Environment FILE card. 'Locate' is assumed when none of these options have been selected.

6. Record Types

a) Definition
There are two record types, fixed length and variable length.

i A fixed length record cannot vary from the length specified by the data description. When it is read or written, a fixed number or words are always made available or output from the buffer.

ii The number of words in a variable length record is defined at object tim be use of the QUANTITY IN option. When it is read or written the current value of the data.name(s) specified by the QUANTITY IN option(s) defines the number of words to be processed. Storage, if the record is transmitted, is always allocated on the basis of the maximum size of the record. The QUANTITY IN clause, by designating that the length of a field within a record is variable in length, identifies the record as being a variable length record.

b) Standard Variable Length Record Format

i The standard variable length record when contained in an external storage medium or in a buffer area is always immediately preceded by a control word containing the record length. Although this control word is not described as part of the record in the Data Description and may not be addressed by the programmer, it must be considered in specifying block size for a file.

The control word is automatically supplied for standard variable length records generated by the system, and must be supplied for those generated outside the system. The form of the control word for standard variable length binary records is:

```
!OCTN 0,,length.of.record.in.words.
```

Standard variable length BCD records must be immediately preceded by six (6) BCD characters specifying the length of the record in words. Facilities for handling non-standard variable length records are describe later in this section.

ii Caution if a variable length record is to be processed in the buffer area (located) it cannot be expanded unless each record in the file begins a new block. No check is made for violation of this rule either at compile time or at execute time.

7. The GET Command

The GET command makes available for processing the next record of an open file. If the file is not open and a GET command is given, the end of file exit is taken. No error message is given. The Commercial Translator language contains two forms of the verb:

GET RECORD FROM file.name
GET record.name

The verb is described incorrectly in the Commercial Translator General Information manual and the definition given in the last paragraph on page 39 of the manual should be replaced with:

"The first form of the command assumes that the system has at its disposal sufficient information to obtain the next record. When using a GET of the second form the programmer supplies the system with the name of the next record in the file. Regardless of what record is next in the file, it will be obtained by the GET in accordance with the characteristics associated with the specified 'record.name'. The 'record.name' must be associated with only one input file in the Environment Description."

In order to use the form 'GET RECORD FROM file.name' in 709/7090 Commercial Translator programs at least one of the following conditions must exist:

a) All records in the file are of a fixed and equal length.

b) The beginning of each record in the file corresponds with the beginning of a tape block (indicated by the BEGIN option of the Environment Description FILE card).

c) All records in the file are in the standard variable length form.

d) Each tape block in the file consists of one complete record whose length is equal to the length of the block (signalled by the Environment BLOCK CONTROL).

e) All records in the file which will be obtained by this form of the command are included in a PATTERN in the Environment Description.

C. Usage of the GET Command

1. Factors Affecting Choice and Use of Locate or Transmit Mode

a) File characteristics and processing requirements

A GET command instructs the system to make available for processing

a record which is in a buffer area. In the 709/7090 system, records may be either located or transmitted. Transmission occurs when either the HOLD or SPANS option is specified for the file in the Environment Description, SPANS being designated when records in the file may span blocks and HOLD when records of different names must be available from a file at the same time.

b) Relative Efficiency

The locate mode allows for more efficient utilization of memory, as no additional space outside of the buffer area is required for record storage.

The locate mode allows for more efficient processing of records whose fields are referenced only infrequently in arithmetic, MOVE, or indexing operations. In general, a record should be transmitted if the number of words in the record is less than three times the total number of arithmetic and MOVE references to fields of the record.

c) Miscellaneous Requirements of Locate and Transmit Modes

i If the locate mode is being used, no references can be made to the record or its fields until after execution of a GET command for that record. All references to fields of a located record initially specify location zero. Upon a GET these references are adjusted to reflect the location of the data within the buffer area. If references are made to located fields prior to the first automatic GET adjustment the low order portions of memory (the monitor) will be irreparably damaged.

ii When through use of the type code REDEF a record is specified to share an area with data other than records within the same file, all records of the file are automatically 'transmitted'. If the records of the file were originally to be located a message is printed indicating this change.

iii All records of a file (whether located or transmitted) which are related through use of the Data Description type code REDEF are made available for processing when any one of the records is requested with the GET command. When records of a file processed in the locate mode are obtained using the GET RECORD FROM file.name form of the verb the result is as described above.

2. End of File Processing

The AT END clause which may be used with the GET command may consist of a single imperative statement only. The clause is used to specify the action

to be taken when the end of a file is reached. A programmer may omit the clause when it is felt no end of file condition will occur with a particular GET.

However, when no alternative action is specified through use of the AT END clause, the discovery of an end condition in attempting to GET a record will cause immediate termination of execution of the object program; an error message is printed on-line indicating the cause.

3. Input Error Processing

IOCS automatically attempts to correct most input errors discovered in processing.

a) Input Errors Uncorrectable by IOCS

i Unrecoverable Redundancy Errors

These are errors typically caused by permanent tape defects and and attempts to read tapes in the wrong mode.

ii Block Checksum Errors

A block checksum error occurs when the checksum of the block written with it on tape does not agree with the checksum calculated at the time the tape is read. Block checksums are not checked unless the Environment SPECIF card option CKSUMS is selected (see section 02.06).

iii Block Sequence Errors

Block sequence errors occur when an out of sequence block number is discovered. If block sequence checking is desired the Environment SPECIF card option SEQ must be selected (see section 02.06).

iv Record Length Errors

Record length errors (referred to as EOB errors in the IOCS manual) generally arise when information in a block does not conform with the blocking conventions described by the programmer in the Environment Description. (Records of an input file which span block boundaries cannot be processed unless the SPANS option is specified).

Record length errors are totally unrecoverable by IOCS or the programmer and cause immediate termination of object program execution. An error message is given on the on-line printer.

b) Facilities for Programmer Error Recovery

IOCS cannot continue processing after discovering an unrecoverable redundancy error, a block checksum error or a block sequence error without direction from the programmer. The ON ERROR option of the Environment FILE card provides for communication between IOCS and the programmer in these three error situations.

In certain simple error situations such as an unrecoverable redundancy error discovered in a file whose records are complete within the block, the programmer may elect to return directly to the GET command ignoring the error(s) in the unreadable record. This is accomplished by specifying in the ON ERROR clause of the FILE card the procedure.name associated with the GET command.

Recovery from most errors, however, is much more complex. The nature of the error discovered by IOCS must be obtained and analyzed by the programmer and then recovery attempted with or without IOCS routines. A detailed explanation of IOCS facilities available to the Commercial Translator programmer is given in the IOCS manual (primarily pages 17 and 18).

c) System Action When Programmer Recovery not Attempted

If the programmer does not provide for error recovery through the ON ERROR clause, when an IOCS-unrecoverable error is discovered during execution of the object program, the system prints an on-line message describing the type of error and the name and certain characteristics of the offending file. Control is then returned to the Commercial Translator Supervisor for processing the next job.

D. Usage of the FILE Command

The FILE command transmits a record from a reserved memory area or an input buffer to an output buffer area for automatic processing onto tape or cards.

1. Forms of the Command

a) FILE record.name

The command "FILE record.name" causes a record to be filed in the output file(s) with which it has been associated through the Environment Description. A record so filed is still available for further processing. However, if the record carries the option PRIMARY in one or more of the output files to which it is associated with in the Environment Description, it is only filed in those files wherein it is so classified.

b) FILE record.name IN file.name

This form of the verb provides a means of filing a record in a specific file when the record.name is associated with several output files.

2. Miscellaneous Considerations

a) In using either form of the FILE command the record.name must be associated in the Environment Description with the name(s) of the file(s) into which it is to be filed.

b) If a file has not been OPENed or has been CLOSEd when a FILE command is encountered at execution time, the command acts as a NOP. No error message is given.

E. Variable Length Record Options

The options described for manipulation of variable length records are specified on the Environment FILE card for each record to which they pertain. They may be elected selectively when only the GET record.name form of the GET command is used to obtain records in the file; the options must be specified for all records if the GET RECORD FROM file.name form is used.

1. Non-standard Variable Length Input Records

Facilities are provided for handling variable length input records not preceded on tape by the standard control word in either one of two ways:

a) By use of the FIND LENGTH IN data.name option

If the length of a non-standard variable length record is already available in storage at the time the record is to be obtained, its location in storage may be indicated to the I/O system through use of this option.

b) By use of the BLOCK CONTROL Option

If the length of a non-standard variable length record is always equal to the size of the block in which it appears, this characteristic is specified by the BLOCK CONTROL option.

The PLACE LENGTH IN data.name option requests that the length of the record obtained be placed in the specified data.name after completion of the GET command. The data.name is not required to be in the record referenced. This option may be used with either of the above two options.

2. Variable Length Output Records

a) NO CONTROL WORD

This option when associated with the name of a variable length record on the Environment FILE card, instructs the system to suppress generation and output of the standard control word for the record.

b) FIND LENGTH IN data.name

Specification of this option notifies the system that when the record is filed its length is to be taken from the data.name specified rather than calculated on the basis of the length of the fields of the record. The data.name specified need not necessarily be in the referenced record. This option may be specified whether or not standard control words are to be prepared.

c) PLACE LENGTH IN data.name

This option instructs the system that prior to filing the record referenced, the length of the record is to be placed in the location, data.name, which is not required to be in the record.

3. Use of the Options with Fixed Length Records

For special purposes, the programmer may employ in connection with a fixed length record any of the variable length record options explained above. NO CONTROL WORD will produce no effect.

F. Examples of Usage

1. Example 1

a) Statement of problem

i File consists of records of three fixed lengths:

REC1 - 64 words
REC2 - 128 words
REC3 - 192 words

ii The exact order in which records appear on tape is unknown.

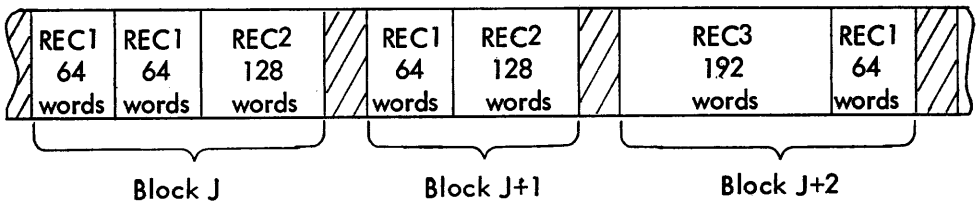
iii Block size is specified as 256 words (BLOCKSIZE 256).

iv Records do not span block boundaries nor is holding required. (SPANS or HOLD option not taken).

v Each record start does not coincide with the beginning of a new block (BEGIN option not taken).

b) Illustration of Tape

A portion of the tape might be:



c) Problem Explanation

Blocks J and J+2 are full 256 word blocks. Block J+1 is only 192 words long because REC3 could not be placed in Block J+1 without exceeding the 256 word limit on block size, nor could part of the record be put in Block J+1 and J+2 since records of this file are not allowed to span blocks.

When input these records will be processed in the "locate" mode (condition iv above). They may be obtained only with the GET record.name form of GET since the records have differing fixed lengths and no pattern has been supplied.

If the records are presented for output in accordance with the stated specifications and in the order shown the system will automatically prepare the tape as illustrated.

2. Example 2

a) Statement of problem

i File consists of records of three fixed lengths.

REC1- 64 words
REC2 - 128 words
REC3 - 192 words

ii The exact order in which records appear on tape is unknown.

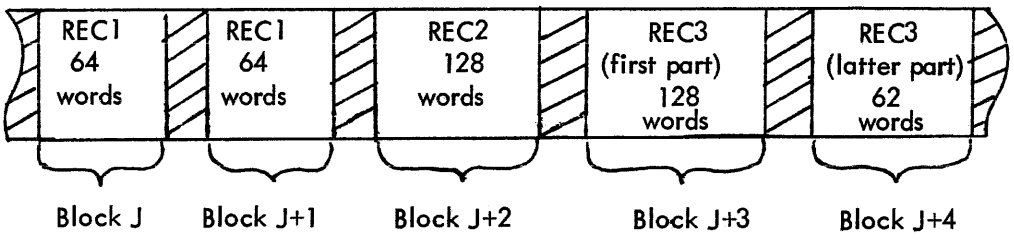
iii Block size is specified as 128 words (BLOCKSIZE 128).

iv Records may span block boundaries (SPANS option taken).

v Record always start at the beginning of new blocks (BEGIN option taken).

b) Illustration of Tape

A portion of the tape might be:



c) Problem Explanation

Blocks J, J+1, J+3, and J+4 are truncated in order to satisfy the BEGIN option REC3 exists in Blocks J+3 and J+4 since it is too large to be contained in a single block and records are allowed to span blocks in this file.

When input these records will be processed in the transmit mode since they may span block boundaries. The GET record.name form must also be used with this file for reasons outlined in Example 1.

. 3. Example 3

a) Statement of problem

i Records composed of 85 characters of significant information have been put on tape using standard card-to-tape equipment. The information for each record is contained on two cards:

Card 1 - 72 characters
Card 2 - 13 characters

ii The records of the file are all of the same fixed length and the same record.name.

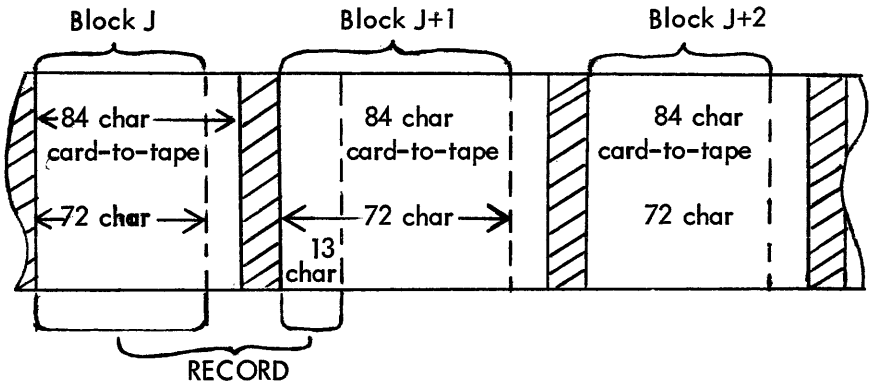
iii Block size is specified as 12 words (BLOCKSIZE12).

iv The records span block boundaries since they must exist in two blocks (SPANS option exercised).

v Records always start at the beginning of a new block (BEGIN option exercised).

b) Illustration of Tape

A portion of the tape might be:



c) Explanation of Problem

In this file hardware characteristics force a somewhat artificial record and block structure. The two parts of the 85 character records do not appear on tape contiguously, being separated by a record gap and by extraneous information. Additionally, useless information appears on tape between consecutive records.

The block size of 12 directs the system to bring only 12 words from

each tape block into internal buffer areas, thus discarding the last 2 words of each block. Since the SPANS option is exercised, the first 12 words of BLOCK J and the first 3 words of Block J + 1 will be transmitted to a 15 word memory area reserved for the record. The remaining portion of Block J + 1 is discarded due to the use of the BEGIN option, which directs system attention to a new block when a GET command is issued.

Either form of the GET command may be used to obtain records from this file.

4. Example 4

a) Statement of Problem

i File consists of standard fixed length records prepared on card-to-tape equipment.

REC1 - 80 characters
REC2 - 80 characters

ii The exact order in which records appear on tape is unknown.

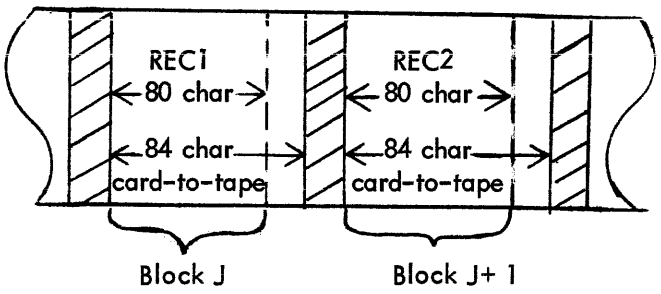
iii Block size is specified as 14 words (BLOCKSIZE 14) as it is desired to make all 80 characters available.

iv Records do not span block boundaries; however, it is desired to retain each record until the next record of the same name is obtained (HOLD option taken).

v Each record begins a new block (BEGIN option taken).

b) Illustration of Tape

A portion of the tape might be:



c) Explanation of Problem

Blocks J and J + 1 each contain 84 characters (14 words); 80 characters of the record plus 4 characters of unwanted data which the card-to-tape equipment supplied to make the block a multiple of six characters.

Upon the command 'GET REC1' (assuming that REC1 is the next record to be read) 14 words are transmitted (because of the HOLD option) from the buffer to the area associated with REC1. Now if the command 'GET REC2' is given, 14 words are transmitted to the area belonging to REC2. The information at REC1 is still available after this second command and will continue to be available until replaced by another 'GET REC1'.

Note, other Commercial Translator operations, such as the MOVE, will be based upon the 80 characters specified in the Data Description of the record rather than the 84 characters appearing on tape.

02.08 The 709/7090 Machine Symbolic Language - CRYPT

A. Introduction

The 709/7090 Commercial Translator System provides the programmer with the ability to include symbolic machine language instructions and data description at any logical point in his program. The procedure command, 'ENTER CRYPT' signals the System that the ensuing source statements are in symbolic machine language form. The object instructions generated from these statements will be located immediately following the last word generated from the preceding Commercial Translator statement.

The symbolic machine language acceptable to the system is similar to SCAT, with several restrictions and some additional flexibility. The standard SHARE 709/7090 coding form is used for CRYPT statements.

B. CRYPT Rules

The rules of the SCAT system are modified in the following way for CRYPT.

1. Since the special characters (see chapter 2 of the General Information manual) are used for special purposes in the source language of Commercial Translator, these characters cannot be used as part of symbols. The same rules which are applied to Commercial Translator names are applied to CRYPT names.

   Examples of illegal or specialized usage are:

L(5)	Will be interpreted to mean the fifth of the array elements, L.
.AB.	A period may not be used as the first or last character of a symbol.
A'B)$	The characters, "Quotation Mark," "Left Parenthesis," "Right Parentheis," "Dollar Sign" may not be used as part of a symbol.

2. No SCAT macros are acceptable.

3. The only pseudo ops which may be used are:

OCT	BSS
PZE, etc. (both MZE and FOR sets)	BES
EQU	BCI

4. The following pseudo ops may not be used:

   HEAD
   VFD
   DEC
   DUP
   ORG
   SYN
   END

5. Literals are not presently permitted in machine language.

C. Flexibility Above that of SCAT

In using the CRYPT language, advantage may be taken of several Commercial Translator features which have been added to SCAT facilities:

1. Names which are defined in the Commercial Translator sections of the program may be referenced in the variable field of the machine language sections. Thus, names in the variable field may contain up to 30 characters. Compound names (qualified) may not, however, be used in the machine language section. If such a qualified name is to be used, it should be renamed as a single name, using the CALL verb outside a CRYPT section.

2. The Commercial Translator system for the 709/7090 addresses array elements by the use of "Symbolic Registers". The Symbolic Register for the array, A(I,J), always contains the address of the element corresponding to the present values of the component variables, I and J. The system will replace a symbolic instruction of the form CLA A(I,J) with the following instruction: CLA* SRAIJ. It should be noted that the system will not check for changes in component variables while in the machine language mode. Therefore, although the programmer may change the values of I or J, the Symbolic Register for A(I,J) will not be updated by the System. Also, indirect addressing is not permitted on the original operation.

3. Arithmetic expressions may contain parenthesis. For example, the following is a legitimate usage: AXT A*(B+C),4.

4. The Commercial Translator Data Description form provides a flexible and effective means of describing data and entering constants.

D. Machine Language Instructions Generated from CRYPT Language

There are several instances where the System develops instructions which do not agree with those specified by the symbolic source language on a one-to-one basis. If DATE is specified in the Environment and Data Description sections as a field of record to be processed in locate mode, the system will generate for CLA DATE the machine language instructions of the form:

```
LAC    BASE.LOCATOR.OF.DATE,4
TXL    SYS)294,4,0
CLA    DATE.DISP,4
```

where:

BASE.LOCATOR.OF.DATE contains the current origin of the record in which DATE appears.

SYS)294 is an error message routine that prints the fact that BASE.LOCATOR.OF.DATE has not been set if such is the case.

DATE.DISP is the displacement of the DATE field from the base location of the record in which it appears.

Note that the contents of index register 4 are destroyed, and that instructions have been inserted in the program. Care must be taken to avoid incorrect referencing when relative addressing using * to signify the location counter is used.

E. Return to Commercial Translator Language

Processing of CRYPT instructions is terminated by an ENTER COMMERCIAL TRANSLATOR instruction. ENTER is punched as an operation code and COMMERCIAL TRANSLATOR as a variable field entry beginning in column 16. The Compiler expects to find Commercial Translator source statements after this instruction.

SECTION 03.00

LOADER

INTRODUCTION

The Loader, under the direction of the Commercial Translator Supervisor creates a machine language program from the relative binary deck(s) produced by the Compiler and normally turns control over to the program for execution. As part of the loading procedure, separately compiled program segments are combined; storage is allocated for program instructions, data and buffers; object program, system subroutines and specified IOCS modules are loaded; and control is transferred to the object program starting point.

A detailed description of the control cards which direct the Loader, and a discussion of the Loader input and output appear in this section.

Current Loader Deferred Features, Restrictions, and/or Limitations are found in Appendix 90.01.

03.01 Composition of the Commercial Translator Deck to be Loaded

A. Deck Composition

A Commercial Translator deck to be Loaded is composed of the following types of cards and is processed in the order indicated.

1. $LOAD
2. Symbolic Control Cards (in any order)
   - *FILE
   - *SPEC
   - *POOL
   - *GROUP
   - *RETAINS
   - *DELETES
   - *FILEQU
   - *START
3. *CTEXT
4. Relative Binary Program Deck (see section 90.03 for deck details and specifications).

   a) Control Break Table

   The control break table defines procedure and/or data areas in this deck which may be deleted; or replaced or referenced by other program segments, which have been separately compiled (see CONTROL in section 02.06).

   b) File Check Table (see section 90.03 for specifications)

   The file check table is used to validate the contents of the generated file block (see IOCS manual) and the assignment of each file to a buffer pool. It transmits the location of any non-standard label routines to the IOCS System.

   c) Text

   Compiler produced machine language instructions and data references.

5. *CTEND
6. Special end-of-file card (or physical end-of-file)

B. Combination of Separately Compiled Decks (Not currently available).

This feature allows cross-referencing of common data fields and procedure sections. The data description for a common field must be included within each compilation which references this field. The allocation of a unique storage area to a common data field or section may be specified by loader control cards.

If more than one deck is to be run as a single job, the symbolic control cards required for the combined program are loaded first, followed by the Relative Binary Program decks (each consisting of parts 3, 4 and 5); and, finally, the end-of-file. The symbolic control cards of the individual segments may be retained with their Relative Binary Program cards. However, all symbolic control cards appearing after the initial Relative Binary Program deck will be ignored by the Loader.

C. Requirements for Loading a Single Deck

To load a single deck, the <u>required</u> cards are

1. $LOAD
2. Symbolic Control Cards
   *FILE
   *SPEC
3. *CTEXT
4. Relative Binary Deck
5. *CTEND
6. End-of-file

The Compiler produces all of these except the $LOAD and the end-of-file card.

03.02 Loader Control Cards

A. Introduction

Loader control cards direct and control the Loader in such functions as combining separately compiled programs and allocation of storage for buffers. Also, they describe the physical file characteristics required by the 709/7090 Input/Output Control System. Most control cards are supplied by the compiler. However, the programmer might wish to supplement or replace them in some cases. Loader control cards must be placed at the beginning of the program deck. If placed otherwise, they are not interpreted.

B. $LOAD Card

1. The Loader is called when the Commercial Translator Supervisor recognizes the $LOAD card of the form:

```
1 - 6    8  -  13   16 -  54
$LOAD    deck.name  [MIMIMUM]     [,LOGIC]  [,MAP]
                    [BASIC  ]
                    [LABELS ]     [,NOGO ]  [,FILES]
                    [IOEX   ]
                     55  -        72
                    [secondary.identifier]
```

The options in the variable field must be separated by commas and may appear in any order. Any combination of options may be selected with no blanks separating them.

2. The options are defined as follows:

[MINIMUM / BASIC / LABELS / IOEX] — MIMIMUM requests that the minimum IOCS package be made available, BASIC requests that the basic IOCS be made available, LABELS request that the full IOCS with labeling features be made available, and IOEX requests that the Input/ Output Executor be made available. (see 709/7090 IOCS manual for definition of these forms). If no option is selected the MINIMUM IOCS will be made available unless checkpoint is specified. In this case BASIC IOCS will be available. However, the full (LABELS) IOCS will be made available if labels are to be handled.

[,LOGIC] — Instructs the Loader to list the origin and extent of all program sections, system subroutines required for execution (including IOCS) and buffer assignments. This option may be specified on the $CMPLE as a 'carry through' feature when a 'compile and load' run is to be made.

[MAP] This option has not yet been implemented. It will provide for obtaining load-time information in addition to that supplied by LOGIC.

[NOGO] Instructs the Loader to inhibit execution of the program after loading. It does not affect the LOGIC option.

[FILES] A list of files showing the I/O unit assignments made by the Loader, and any mounting instructions, is always prepared for the operator. If this option is specified, a duplicate of the file list is written on SYSOU1.

[secondary.identifier]
The information in these columns will appear in the page headings of all output prepared on SYSOU1.

C. *FILE Card

1. Format

The format of the *FILE card is, with two exceptions, the same as that specified in the 709/7090 IOCS reference manual. That is,

1-6	7-11	14-15	17	18-21	22-25	27 - 35	38 - 41
deck.name	*FILE	file.no	M	unit1	unit2	controls	reel.seq
		44-48		51 - 53	55 - 72		
		file.serial		ret.days	File.name		

The two exceptions are:

a) The file number (file.no) is assigned to each file by the Compiler and appears with the file name on the listing provided by the Compiler. In this card the field does not occupy column 13.
b) The unit (unit 1 and unit2) specification has been modified to include relative unit assignment, symbolic channels, and tape transport model type. The card fields have been enlarged from three to four columns: columns 18-21 for unit1 and columns 22-25 for unit2.

Appendix 90.08 contains a chart showing how the options of the Environment FILE and SPECIF cards are used by the Compiler in preparing the *FILE card.

2. Unit specifications (unit1, unit2)
The notation employed in explaining these options is:

X	denotes one of the real channels A, B, ...,H
P	denotes a symbolic (unspecified physical) channel S, T, ..., Z
k	denotes one of the unit numbers 1, ..., 9, 0
m	denotes a tape transport model number, II or IV.

The unit specification may be made in any of the following ways:

a) No specification - when the option is not exercised any available unit is to be assigned to the file.

b) m - any available unit of this model type is to be assigned to the file.

c) X - any available unit on this physical channel is to be assigned to the file.

d) P - all files in the job having this symbolic channel designation are to be assigned to the same channel.

e) X(k) - the kth available unit on the specified channel is to be assigned to the file; note that the parenthesis are required.

f) Pm - any available unit of this model type on the symbolic channel specified is to be assigned to the file.

g) Pkm - an available unit on the symbolic channel, having this model number, is to be assigned to the file. k in this use without parenthesis indicates the order of preference for the channel so that if the number of available units on the channel is less than the total requested for the channel those with lower numbers are to be assigned to the same channel.

h) System units may be assigned by:

i IN - the current system Input unit is to contain the file.

ii OU - the current system Output unit is to be used for a printed output file.

iii PP - the current system Peripheral Punch unit is to be used for a punch output file.

iv UTk - System utility tape k (non-parenthesized k= 1-4) is to be used for the file.

i) Card equipment is assigned for a file by:

1) RDX - card reader, channel X
2) PRX - printer, channel X
3) PUX - card punch, channel X

j) As a special option, an * in the UNIT2 field indicates that the secondary unit of a file is to be any unit on the same channel and of the same model type, which is available after all other assignments have been made. At load-time if units are available, but of a different model, one will be assigned; if no units are available on the channel, no secondary unit is assigned prior to execution of the job.

3. Unit Assignment

Units are assigned in the following order:

1. System units
2. Card units
3. Units on specified channels
4. Units on symbolic channels
5. Units with model only specification
6. Unit with no (blank) specification
7. Secondary units designation with *

It should be emphasized that relative unit specification is not the same as the physical setting of the tape unit. The use of A(3), for example, will result in the assignment of the third available unit on channel A; and not necessarily the physical unit A3. (An available unit is one presently attached to the machine, and not assigned for system use.) However, the assignment procedure is such that designation of a given relative unit on a given (physical) channel will always result in the same choice of unit, provided that the unit sequencing is not disturbed between jobs by use of the Basic Monitor $ATTACH - [$AS]control cards.

When consistent tape assignments are required (e.g., when a file is used by two successive jobs or when a file is written and then read in the same job) the unit specifications should be identical in the FILE cards. The units should be specified as system utilities (UTk) or by means of specific channels (X) with low unit availability designations (k) in order to guarantee use of the same drives.

Care should be taken in assigning system utility units (SYSUTk) to object program files. In particular, they should not be assigned to files which will specify in the printed file list the immediate mounting of tapes by the operator, since these tapes will still be in use after this list is printed.

D. *SPEC Card

The *SPEC card supplements the *FILE card in supplying file information. The format of the *SPEC card is:

1-6	7-11	14-15	17-20	22-23	25	27
deck.name	*SPEC	file.no	blocksize	activity	open	close

The 'deck.name' and 'file.no' are used to identify the file as the same file described on a *FILE card. Unless the same 'deck.name' and 'file.no' appear on a *FILE in the deck the *SPEC Card will be ignored.

'blocksize' is normally a number (0-999) but may be left blank. A number in this field specifies the maximum number of words to be input or output in a single block. It may be desirable to change the blocksize in an attempt to obtain better input/output processing speeds or tape utilization. Conflict with file specifications entered in the Commercial Translator Environment section must be avoided. The loader will check most normal situations in accordance with information supplied to it by the Compiler.

'activity' may be specified as a number 0-99 or may be left blank. A number in this field specifies the relative activity of the file in respect to other files and is used by the Loader in the allocation of buffer areas.

The 'open' options are:

1.	N	-No rewind
2.	R or blank	-Rewind

The 'close' options are:

1.	U	-Rewind and unload
2.	R or blank	-Rewind
3.	N	-No Rewind
4.	S	-No file mark or trailers, no rewind

Appendix 90.08 contains a chart showing the source of the fields of the *SPEC card when it is generated by the Compiler.

E. *POOL Card

The *POOL card designates which files are to share common buffer areas. The format of this card is:

1-6	7-11	14-15	17-20	23-25	27
deck.name	*POOL	pool.no	blocksize	buffer.cnt	file1, file2...filen

The 'blocksize and buffer count' specifications are described in the Environment Description POOL card (section 02.06).

1. If the file referenced is described in the deck named on the *POOL card, specification of the file number shown in the compiler file list adequately

identifies the file. Its format is

file.no

2. If the file reference is to a file in a deck other than the one named on the *POOL card, the deck.name of the deck in which the file is described must be specified in addition to the file number. Its format with required parentheses is

deck.name (file.no)

The file references must be separated by commas and no blanks may be embedded in the field.

If the variable field information extends beyond a card additional cards may be used as needed. However, the 'deck.name' and a 'pool.no' must be the same on all cards for the same pool. All other remaining fields need appear in only one of the cards; if on more than one, the first (non-blank) value encountered is used.

F. *GROUP Card

1. Format

The information of the *GROUP card corresponds to that on the Commercial Translator Environment GROUP card. The format of the *GROUP card is:

1-6	7-12	14-15	17-18	20-21	23-25
deck.name	*GROUP	group. no	pool.no	Opn.cnt	Buffer.cnt

27
file1,file2,...,filen

2. Specification

File references and variable field overflow, follow the same rules as the *POOL card. The deck.name, if used, must correspond to that of the *POOL card with which it is associated through the pool number. The file references on the *GROUP card need not be duplicated on the *POOL card in order to be included in the pool. However, the *POOL card may have file references other than those on associated *GROUP cards.

All *POOL cards are processed first, in the order read, then *GROUP cards. If there is a conflict in pool or group assignments (e.g., a file directly assigned to some pool, then to a group belonging to another pool, or a file assigned to two different groups or pools) the first assignment is used.

In the absence of any *POOL or *GROUP cards, the Loader will make its own assignments. These cards enable the user to make a more sophisticated assignment, based on knowledge of file use and activity.

G. *FILEQU Card

When separately compiled decks are combined, the loader must be told which files are common so that they can be processed as a single file. The names associated with a file at compile time are not available at load time; rather, files are identified by deck.name and file number. This identifying information is used on the *FILEQU to specify file equivalence. The form of the *FILEQU card is

1-6	7-13	16-72
deck.name	*FILEQU	file1,file2,...,filen

All files referenced on a *FILEQU card take on the processing characteristics of the first file referenced whose characteristics are defined by a *FILE card. File references may be of two forms:

1. deck.name (file.no)
'deck.name' is the name of the referenced program segment in which the file appears (see deck.name in *FILE card) and 'file.no' is the identifying number of the file within the referenced deck.

2. file.no
'file.no' identifies a file whose deck identification is that specified in columns 1-6 of the *FILEQU card. Note that all blanks in columns 1-6 of the *FILEQU card may be used as a legitimate qualifier for each specified 'file.no'.

As all files referenced on a *FILEQU card take the same processing characteristics as the first file mentioned, those processing characteristics may be altered by altering the content of the *FILE and *SPEC cards for that first file.

In case all of the equivalent files cannot be specified on a single card, additional cards may be used. At least one of the file references on the first card must appear on the subsequent cards.

H. *RETAINS Card

When several separately compiled programs are to be run as a single job, certain areas of procedure or data, referenced in the individual compilations must be identical. These areas must be specified in the Environment Description of each compiled job on CONTRL cards (see section 02.06). The extent and nature of the area is entered into the Control Break table of the compiled program together with the external reference name (ex.nm) as given on the CONTRL card. The Loader identifies the equivalence of these areas by means of the external reference names. In the absence of any other information, the area is assumed to be located in the first program segment containing the 'ex.nm' in its Control

Break table. All instructions and data corresponding to this area in the other segments are deleted by the Loader, and all references in these programs are changed to reflect the new location.

Since it is not always possible to arrange the order of the deck so that a CONTRL area is located in the first segment, the *RETAINS card provides a means by which the location of such an area can be established as desired. The format of this card is:

1-6	7-15	16-72
deck.name	*RETAINS	ex.nm1,ex.nm2,...,ex.nmk

'deck.name' is the name of the program segment in which the CONTRL areas ex.nm1,ex.nm2,...,ex.nmk are to be retained. These named areas will be deleted from all other program segments. If ex.nmj does not appear as an entry in the Control Break table of program 'deck.name', it cannot be retained by the program, and will have no effect on the loading procedure.

The variable field entries, ex.nmj, are separated by commas, and there must be no embedded blanks in the field.

I. *DELETES Card

The *DELETES card eliminates procedure and data areas defined by an Environment CONTRL card. For example, to delete sections of a program that were originally intended as an aid in source language debugging. The format of the *DELETES card is:

1-6	7-14	16-72
deck.name	*DELETES	ex.nm1,ex.nm2,...,ex.nmk

where ex.nmj is the external reference name appearing in the CONTRL card which defined the area. If a 'deck.name' is given, the CONTRL area(s) will be deleted only from the named program. If no 'deck.name' is given then the CONTRL area(s) will be deleted from all programs in which they appear.

J. *START Card

The Loader will normally use the starting point of the first program of combined segments as the starting point for the combined deck. The *START card permits this to be varied. The format of the *START card is:

7	16
*START	deck.name

Execution of the combined program will begin at the starting point of the deck named.

K. *CTEXT and *CTEND Cards

The *CTEXT and the *CTEND cards, supplied by the compiler mark the beginning and end of the relative binary deck (control dictionary, file check table, and text) produced on a single compilation. No modification or reordering of the binary deck should be attempted. The formats of these cards are:

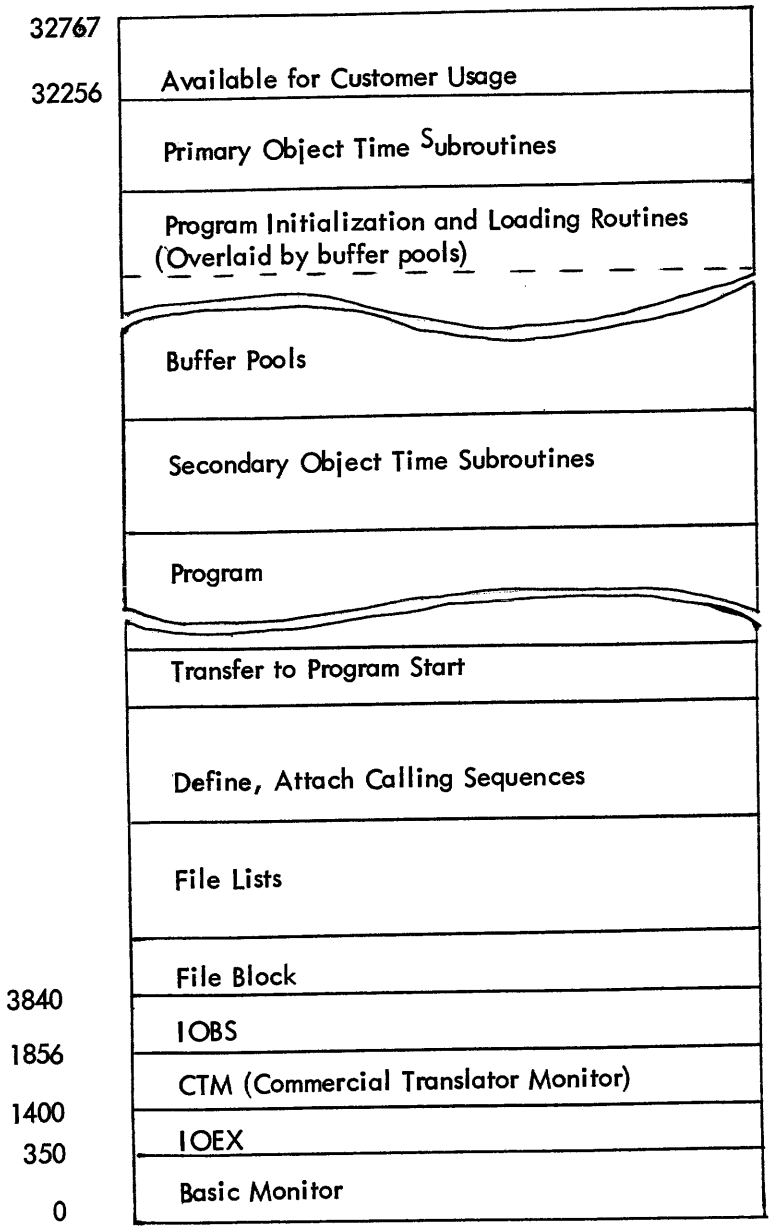
1-6	7-12	26-54	55-72
deck.name	*CTEXT	date.and.time	secondary.identifier
deck.name	*CTEND	date.and.time	secondary.identifier

'deck.name' and 'secondary.identifier' are those supplied on the $CMPLE card and 'date.and.time' are supplied by the Compiler to specify when the deck was produced.

03.03 Loader Output

A. Program for Execution

The binary object program deck, together with required subroutines, appropriate I/O routines and tables and a portion of the Basic Monitor, is placed in core storage by the Loader. If the program is to be executed, control is transferred to it at the conclusion of the loading process. The following chart illustrates the general layout of core storage at execution time.



Note: The core addresses shown above are approximate.

The program elements of the chart are described as follows:

File Block - this is a 12-word File Control Block for each file required by the program.

File Lists - these are generated for each buffer pool, showing the group structure and the files assigned to the pool.

Define and Attach Calling Sequences -
each buffer pool must be defined by a reference to the IOCS subroutine, DEFINE. Files are attached to this pool by reference to the ATTACH subroutine. These calling sequences are generated by the Loader and are the first instructions executed.

Transfer to Program Start -
the instruction following the last ATTACH calling sequence is a transfer to the start point of the program.

Secondary and Primary Subroutines -
three types of subroutines are used, Secondary, Primary, and Fixed.

Secondary subroutines -
such subroutines, if called by the object program, are treated as program sections and are loaded following the end of the program in core storage.

Primary subroutines -
certain object time subroutines are expected to have a large number of references in the object program. To avoid placing all instructions containing such references in the 2TEXT# file, primary subroutines, which are required to have a fixed length not modified by the loading process, are assigned core storage locations at the upper end of memory. This assignment is made at the time the first reference to the subroutine is encountered.

Fixed subroutines -
all program references to points in IOEX, CTM and IOBS are defined by fixed subroutines, each of which has an assigned origin. These fixed

subroutines are nothing more than BSS "maps" of the corresponding communication regions.

- 2TEXT is a file generated by the loader during the first reading of program text, including subroutines. In this file are placed all instructions containing references which cannot be defined until the first text pass is complete (e.g., references to secondary subroutines, references to control break sections in a program not yet loaded).

B. Printed Messages

1. Off-line on SYSOU1

a) Error messages
b) The file list if the FILES option was selected
c) A storage map of the program, subroutines and buffers ready for program execution, if the LOGIC option was selected.

2. On-line Printer

A file list is printed showing input/output unit assignments and the tape mounting instructions for the operator.

SECTION 04.

SUPERVISORY SYSTEM

INTRODUCTION

The Commercial Translator Processor operates as one of a group of processors which are connected and controlled by a system supervisor called the 709/7090 Basic Monitor. The flow of jobs through the Commercial Translator Processor is controlled by the Commercial Translator Supervisor (CTM). This section explains the functions of:

1. Basic Monitor
2. Commercial Translator Supervisor

An attempt has been made to provide sufficient operating detail to enable an installation to exploit fully the capabilities of the Basic Monitor, without obscuring the simple methods of ordinary job running.

For a more detailed explanation of the Basic Monitor refer to the manual, IBM 7090 Operating Systems: Basic Monitor (IBSYS). Form J28-8086-0.

For input/output, checkpoint and restart facilities refer to the 709/7090 Input/Output Control System. The form number of the manual is C28-6100-2.

04.01 FUNCTION OF THE BASIC MONITOR

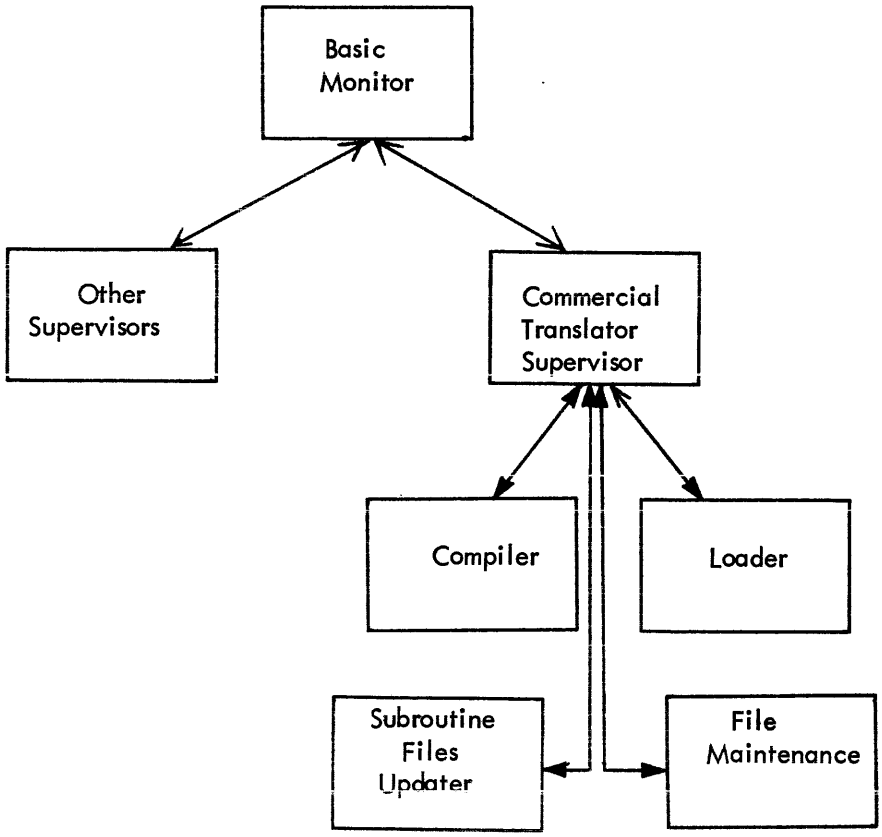
The primary function of the Basic Monitor is to provide continuous automatic operation of a sequence of (stacked) jobs which might involve different processors. For this purpose, it is divided into two parts:

1. A Nucleus, consisting principally of tables, which remain in core at all times, occupying a minumum amount of space. The Nucleus provides the communication necessary for inter-job and inter-processor continuity.

2. The Monitor proper, which is brought into core between jobs. By means of control cards, the monitor routines are directed, among other things, to provide specified assignment of various I/O units, perform certain accounting functions and, most important, pass control to a participating processor to handle the next job.

A. PROCESSING A JOB

The flow of control between the Basic Monitor and its participating processors is illustrated in the diagram below:



The sequence of operations within the framework is:

1. The Basic Monitor is called into core from the system tape. For the first job, this is done by the initial start procedure; for subsequent jobs, this is done by a routine in the Nucleus upon return from the processor which handled the previous jobs.

2. The Monitor functions as directed by control cards. For example, the sequence (the exact format of the control cards will be specified later)

   ```
   $ATTACH     A6
   $AS         SYSPP1
   ```

   would result in an entry in the Unit Function table of the Nucleus, assigning physical unit A6 to the function of System Peripheral Punch tape until changed be some subsequent assignment. It should be emphasized that such an assignment is needed only to deviate from the normal one in use at a particular installation.

   Finally, upon recognition of

   ```
   $EXECUTE    Processor-name
   ```

   (Processor-name = CT for Commercial Translator), the Basic Monitor will position the System Library tape to the named processor, read in the first record, and relinquish control to it.

3. The processor involved examines the Unit Function and Unit Availability tables within the Nucleus; and from them determines which I/O units to use for the job. It then proceeds to process the current job.

4. When the processor encounters the control card $IBSYS, the availability table is restored (if necessary) and control is turned back to the Basic Monitor.

5. Steps 1 through 4 are repeated until the stack of jobs has been completely processed.

The Unit Function and Availability Tables

In order to achieve the maximum flexibility in the use of I/O units attached to the machine, it is necessary to know the number and status of these units at all times. For this purpose, the nucleus contains an assembled table, the Unit Availability table, which describes the physical configuration of the machine, the number of channels, and the number and type of units on each channel.
Programs operating within the system use this table to make their unit assignments,

and are responsible for maintaining it. In addition, changes in configuration, such as removal of a tape transport for maintenance can be reflected in the table by means of Basic Monitor control cards.

Besides the problem of general unit availability, programs operating within an automatic system are forced to use certain units to carry out specific functions. For example, the units to be used for Library tape, the input (job) tape, and the output (list) tape, cannot be chosen arbitrarily by a processor without doing violence to the notion of continuous operation. In order to provide a uniform way to reference this class of units, they have been assigned symbolic names indicative of their function. The particular unit currently assigned to a given function is found in the Unit Function table of the Nucleus. This table, together with the Unit Availability table, provides complete information on the current status of the I/O units available within the system.

The symbolic function names are alwyas used on Basic Monitor control cards, in order to effect changes in status for specific unit assignment for this class of units. The functional units of the current system are:

Symbolic Name	Function
SYSLB1	System
SYSLB2	Library
SYSLB3	and
SYSLB4	Alternates
SYSCRD	System Card Reader
SYSPRT	System Printer
SYSPCH	System Punch
SYSIN1	System Input
SYSIN2	and Alternate
SYSOU1	System Output
SYSOU2	and Alternate
SYSPP1	System Peripheral Punch
SYSPP2	and Alternate
SYSCK1	System Checkpoint Tape
SYSCK2	and Alternate
SYSUT1	System
SYSUT2	Utility
SYSUT3	Tapes
SYSUT4	

The participating processors of the system do not necessarily use all of these units; nor do they use the Utility tapes for the same functions. In order to take full advantage of the tape handling facilities of the Basic Monitor, the operator should take the time to learn the unit requirements and usages for the various processors. The table of Section 06.02 provides such information of the CT Compiler, CT Loader, CT Maintenance, and Subroutine Files Updater.

B. BASIC MONITOR CONTROL CARDS

The notation employed in discussing these control cards (and others in the Bulletin) is as follows:

1. Words in square brackets [ ] represent an option which may be included or omitted at the user's choice.
2. Words in curly brackets { } indicate that a choice is to be made from the contents.
3. Upper case words, and all punctuation, are to be used as is.
4. Lower case words represent a substitution to be made.
5. A number over the first letter of a field indicates the initial card column of the field; a second number, if it appears, is the final column of the field.

$DATE CARD

The first card in any operating stack should be:

1	16
$DATE	mmddyy

which enters the date into a communication cell. Mmddyy would be written 061561 for June 15, 1961.

$ATTACH and $AS CARDS

The two cards

1	16
$ATTACH	Unit [,II]
$AS	Unit-function-name [,H]

are used together, in the order shown, to assign unit 'unit' to the functional unit 'unit-function-name'. This assignment remains in force until changed by another, or until the Basic Monitor is refreshed by a $RESTORE card (see below).

'Unit' is written for tapes in the usual way, such as B9 or C1, while for card equipment,

RDX	for Card Reader, Channel X
PRX	for Printer, Channel X
PUX	for Card Punch, Channel X

is used. The drive type is assumed to be IV unless II appears, and the density is assumed low unless H appears.

$RELEASE CARD

A card of the format

1	16
$RELEASE	Unit-function-name

releases the unit assigned to the functional unit 'unit-function-name' from system use, and makes it generally available. Thus, it is possible for object programs to make use of almost all units on the machine, if necessary. For obvious reasons, the system library tape should not be treated in this cavalier fashion.

$EXECUTE CARD

Recognition of the card

1	16
$EXECUTE	Processor-name

causes the Basic Monitor to position the System Library tape to the named processor, read in the first record, and relinquish control to it.

$IBSYS CARD

A card of the form

1
$IBSYS

returns System control to the Basic Monitor, and prepares it to interpret control cards. It is the one control card which must be universally recognized by all participating processors in the system, in order to effect return of control to the Basic Monitor.

$RESTORE CARD

A card of the form

1
$RESTORE

restores the Basic Monitor nucleus to its pristine state, as well as preparing the monitor to interpret control cards. The $RESTORE card is used primarily to restore the functional units to their normal (assembled) installation assignment.

$CARDS and $TAPE CARDS

The Basic Monitor normally reads control cards from the system input unit, SYSIN1. Upon recognition of the card

```
1
$CARDS
```

however, the next control card will be read from the system card reader (SYSCRD).

Return to control card reading from SYSIN1 is effected by the control card

```
1
$TAPE
```

$UNITS CARD

For operating convenience, the card

```
1
$UNITS
```

causes an on-line print of the physical units currently assigned to the Functional units.

$PAUSE CARD

The card

```
1
$PAUSE
```

provides a means for temporary halt of the machine, in order to allow some required operator intervention.

$SWITCH CARD

A card of the form

```
1                16
$SWITCH          Unit-function-name-1,Unit-function-name-2
```

causes the units assigned to the functional units 'unit-function-name-1' and 'unit-function-name-2' to be interchanged.

Since the actual physical units involved need not be designated, this provides a particularly simple way of:

1. Switching the system input, output, or peripheral punch to its alternate unit;

2. Providing continuity of unit assignment between programs.

For example, the system tape updating program, (IBEDT) produces the updated system tape on SYSUT1.

The card

```
1                16
$SWITCH          SYSLB1, SYSUT1
```

will then cause operation to continue using the updated system in place of the old. Note that density specifications are not transferred in the switching process.

Tape Operation Cards

A card of the form:

```
1                16
$ENDREEL
$REWIND          unit.function.name
$REMOVE
```

requests that the specified operation be performed upon the Functional Unit unit.function.name.

The operations are:

1. $ENDFILE — Write an end of file on the tape assigned to the specified unit.function.name.

2. $REWIND — Rewind the tape assigned to the specified unit.function.name.

3. $REMOVE — Rewind and unload the tape assigned to the specified unit.function.name.

$DETACH Card

And I/O unit which for any reason is disabled may be made unavailable for any use by a card of the form:

```
1                16
$DETACH          unit.name
```

The unit named remains unavailable until referenced by a $ATTACH card or until the Nucleus is refreshed.

$STOP CARD

A card of the form

```
1
$STOP
```

will cause printing of the message 'END OF JOBS, CANNOT PROCEED' and halt the machine.

$* CARD

All Basic Monitor control cards are printed on-line in order to keep the operator informed of the developments as they occur. Remarks or instructions can be included by means of the card

```
1  3
$* any-text
```

which is printed when encountered, but has no other effect.

04.02 FUNCTION OF THE COMMERCIAL TRANSLATOR SUPERVISOR - CTM

The primary function of CTM is to provide continuous automatic operation upon a sequence of (stacked) Commercial Translator Processor jobs. This supervisor gains control from the Basic Monitor as the result of a Basic Monitor Control card of the form:

```
$EXECUTE        CT
```

CTM then controls the processing of the input on SYSIN until the Basic Monitor control card

```
$IBSYS
```

is encountered, at which time return is made to the Basic Monitor.

The format of that part of SYSIN which is concerned with Commercial Translator Processing is as follows:

```
$EXECUTE        CT
        CTM Control Card for JOB1
        JOB1
        End-of-file
        CTM Control Card for JOB2
        JOB2
        End-of-file
        Input for JOB2
        End-of-file
          .
          .
        Etc.
          .
          .
$IBSYS
```

*Note each Commercial Translator job must be followed by an end-of-file. (See 05.03)

In accomplishing automatic operation upon a stacked job input-tape, CTM performs the following functions:

1. CTM reads control cards for subsystem branching.
2. CTM provides communication, transfer, and return points for CTM subsystems.
3. CTM provides system tape positioning and reading of CTM subsystems and IOCS packages.
4. CTM provides for switching of the Basic Monitor Unit Function Table entries for SYSOU's, SYSIN's, and SYSPP's when any CTM subsystem causes reel switching of the corresponding table entries.

CTM recognizes 8 different control cards. All of these are in the format of the Basic Monitor Control cards (Operation begins in column 1).

1. $IBSYS
2. $ID
3. $ENDREEL
4. $CMPLE
5. $LOAD
6. $SUBUP
7. $MAIN
8. $PAUSE

$IBSYS cards cause a return to the Basic Monitor.

$ID cards cause a transfer to an accounting routine.

$ENDREEL cards, when read by CTM, cause a reel switch involving SYSIN1 and SYSIN2.

$ENDREEL cards must always be preceded by an associated end-of-file on SYSIN.

The $ENDREEL may occur in the middle of a job (Example 1) or between jobs (Example 2).

Example 1

SYSIN1

CTM Control Card for JOB1
--- JOB1

End-of-file associated with $ENDREEL
$ENDREEL

SYSIN2

--- Remainder of JOB1

End-of-file associated with JOB1
CTM Control Card for JOB2
ETC.

Example 2

SYSIN1

CTM Control Card for JOB1

End-of-file
$ENDREEL

SYSIN2

CTM Control Card for JOB2
Etc.

Four of the remaining five control cards cause CTM to read the corresponding CTM subsystem from the system tape and transfer control to the subsystem. The $PAUSE control card, does not cause control to be turned over to a subsystem. It prints the $PAUSE card on the on-line printer. This information should inform the operator of the reason for the pause. The message PRESS START TO CONTINUE is also printed. On the other four control cards CTM scans only the CTM subsystem which gets control from CTM.

The Operations which direct the CT Supervisor are:

Operation	Function
$CMPLE	Perform compilation on following source deck.
$LOAD	Perform loading of following object deck.
$SUBUP	Perform Subroutine File updating.
$MAIN	Perform File Maintenance.
$PAUSE	Prints $PAUSE then pauses for operator intervention.

Accounting

A sample accounting routine is provided on the system tape. The sample accounting routine is not intended to work on 7090s (other than the one at WDPC - UCLA). If an installation chooses to insert a different accounting routine, it is recommended that the listing of the sample accounting routine be read. The listing contains the information necessary concerning communication and a description of how to replace the sample accounting routine on the system tape.

SECTION 05.

SYSTEMS OPERATIONS

INTRODUCTION

The information in this section is intended to orient a machine operator with the physical operation of the Commercial Translator Processor. The information includes:

1. Peripheral Equipment Used
2. Peripheral Equipment Assignment
3. System Set-Up Procedure
4. System Restart Procedure
5. Output From System
6. Program Flow and Operator Messages
7. Object-Time Tape Assignment

Further clarification of systems operation may be obtained by reading Section 04, SUPERVISORY SYSTEM; and Section 06, SYSTEMS MAINTENANCE.

05.01 PERIPHERAL EQUIPMENT USED

The Commercial Translator Compiler uses the following peripheral components:

1. One on-line printer.
2. A minimum of 5 tapes:
   a. One System tape (binary).
   b. One listing output tape.
   c. Three utility (scratch) tapes.
3. One tape or card reader for input.
4. One tape or punch for punch output.

The above represents the minimum machine configuration for compilation of a Commercial Translator source program.

The Loading of a CT object program (excluding the tapes used by the Object program) requires:

1. One on-line printer.
2. Possibly three tapes:
   a. One system tape (binary).
   b. Possibly two utility tapes (Used temporarily during the loading of large programs. Small programs will not require these two tapes).
3. One tape or card reader for input.

File Maintenance on the symbolic system tapes(s) requires:

1. One on-line printer.
2. One system tape (binary).
3. One tape or card reader for input.
4. Symbolic system tapes to be maintained and enough blank tapes to contain the new symbolic system.
5. Possibly several blank tapes to contain output from this maintenance procedure which will be used as input to an assembly routine, also on the binary system tape.

05.02 PERIPHERAL EQUIPMENT ASSIGNMENT

In describing the assignment of peripheral equipment, a symbolic notation is used. All I/O units have been given symbolic names. The names chosen for units by the CT Processor are all six characters in length and the first three characters are always SYS. In this manner the following names were chosen to represent I/O units that are referenced by the CT Processor.

Name	Usage by CT Processor
SYSLB1	System Library Tape Number 1
SYSLB2	System Library Tape Number 2
SYSLB3	System Library Tape Number 3
SYSLB4	System Library Tape Number 4
SYSOU1	System Listing Output - First Tape Unit
SYSOU2	System Listing Output - Second Tape Unit
SYSIN1	System Input - First Unit, Card Reader or Tape
SYSIN2	System Input - Second Unit, Card Reader or Tape
SYSPP1	System Punch - First Unit, Punch or Tape
SYSPP2	System Punch - Second Unit, Punch or Tape
SYSUT1	System Utility Tape Number 1
SYSUT2	System Utility Tape Number 2
SYSUT3	System Utility Tape Number 3
SYSUT4	System Utility Tape Number 4

(See the table in Section 06.02 showing the references to these SYStem units by the various parts of the CT Processor).

The physical units assigned to each of the SYStem units will vary from installation to installation. It may well be, for instance, that a particular installation does not want to have two different units assigned for punching purposes. In this case, both SYSPP1 and SYSPP2 would be assigned the same physical unit.

The assignment of physical units to SYStem units is made through a table which is part of the CT Processor which remains in core during all CT Processing.

Temporary changes to the table may be made through the use of Basic Monitor Control Cards. These control cards precede a particular job or group of jobs, for which a 'non-standard' SYStem units assignment is desired. This method of changing the table is described in Section 04.01.

05.03 SYSTEM SET-UP PROCEDURE

A. Preparing the system input -- SYSIN1 and SYSIN2.

Since Subroutine updating and File Maintenance will be performed rarely, the input to the Commercial Translator Processor will usually consist of one or more of the following job types:

1. Compile
2. Load and run
3. Compile, load, and run.

Each job on SYSIN (1 or 2) must be preceded by a control card which specifies the action to be performed on the following deck. (For format of job control cards see Section 04.02). Also, each job must be separated from other jobs by an end-of-file. The end-of-file separation should be obtained by placing an end-of-file card in back of each job deck. This card when read by an off-line 714 card-to-tape converter causes an end-of-file to be written on the tape. The same card is recognized as an end-of-file by the CT Processor whenever the input is read through the on-line card reader. This end-of-file card then, should be thought of as an integral part of every job deck. The format of the end-of-file card is:

Columns	Punches in rows
1 and 2	8 and 7
3 and 4	7, 4, 1, and 12

If there are input cards associated with a particular object program (i.e. if the object program references SYSIN), the input cards must be placed in a separate file immediately following the job to be run.

The end-of-file mark used with a 1401 card to tape set-up may have a different punch combination. Refer to the utility program write up when this set-up is being used.

The group of stacked Commercial Translator jobs must be preceded and followed by Basic Monitor Control Cards. (See Section 04.01 for format and function of Basic Monitor Control Cards). An example of the format of stacked input to be run on-line might be:

Cards		Explanation
$DATE	mmddyy	Provides current date for system use
$ATTACH	RDA	Temporarily resets standard SYSIN assignment
$AS	SYSIN1	
$EXECUTE	CT	Control turned over to CT Processor
$ID	JOB1	Accounting card is optional
$CMPLE		
	BCD	
	Source	
	Program	
	Cards	
*FINISH		This card precedes end-of-file on COMPILE jobs.
End-of-file-card		
$ID	JOB2	Accounting card is optional
$LOAD		
	Object	
	Deck	
End-of-file card		
	Object-time	
	Input cards	
End-of-file card		
$IBSYS		Return control to Basic Monitor
$ENDFILE	SYSOU1	Write end-of-file on list tape
$REMOVE	SYSOU1	Rewind and unload list tape
$ENDFILE	SYSPP1	Write end-of-file on punch tape
$REMOVE	SYSPP1	Rewind and unload punch tape
$PAUSE		Temporary Halt

Normally, only two Basic Monitor Control Cards will precede the stacked jobs:

$DATE	mmddyy
$EXECUTE	CT

B. Starting Procedure

After a job run is started, no action is required on the part of the operator except to follow specific instructions displayed on the printer. To start initially, the following procedure is used:

1. Mount the System Library SYSLB1 on some unit. It should be rewound and set for high density.

2. Mount blank tapes on SYSUT1, SYSUT2, SYSUT3. Ready SYSIN, SYSOU, and SYSPP. (It is not necessary to set the density on any of these units on 7090's. They will set by the CT Processor).

3. Clear the entry keys and set sense switch 1. This switch determines where the Basic Monitor expects to find its first control card. If up. the system input units, (SYSIN1) is used; if down, the system card reader, (SYSCRD) is used.

4. If SYSLB1 is on unit A1, depress the load tape button. If it is on any other unit, load tape on that unit must be simulated by some self-loading program. Normal processing should begin.

5. If it does not, check all units for ready status, and the unit on the System Library for correct density setting.

05.04 SYSTEM RESTART PROCEDURE

If for any reason the computer should loop or come to a complete stop during processing of a CT job, the following action should be taken:

1. Enter the instruction STR (500000000000) in the Entry Keys
2. Set to manual and press the Enter Instruction button
3. Set to automatic and press the Start button

The above procedure will serve to get to the next job whenever the program in core which accomplishes this function has not been destroyed. When the above procedure results in another computer loop or stop, the action taken should be similar to the initial starting procedure:

1. Rewind SYSLB1 (A1)
2. Place sense switch 1 down
3. Place a 'RESTART' deck in the on-line card reader of the form:

```
$DATE       mmddyy
$EXECUTE    CT
```

   If there were Basic Monitor Control Cards causing a temporary SYStem units change as part of the stacked jobs, the same cards should be placed between the $DATE and $EXECUTE cards to effect the same SYStem configuration as was in operation when the program malfunctioned.
4. Clear and depress the Load Tape button.

The above procedure will cause the CT Processor to process the next job on SYSIN.

05.05 OUTPUT FROM SYSTEM

A. Compiler

The CT Compiler output consists of:

1. A list of the source program and error messages concerning the source program is written on the SYSOU tape. In addition, the programmer may obtain on SYSOU a listing of the object program generated by the Compiler. SYSOU must be a tape unit. Attaching the on-line printer as SYSOU will not work.

2. Punch output is placed on SYSPP. SYSPP may be a tape for off-line punching or the on-line punch.

3. Operator messages are written on-line printer. End-of-file marks are put on SYSOU and SYSPP as part of the reel switching operation and through the use of Basic Monitor Control Cards. The separation of decks from off-line punching must be accomplished by means of the deck identification punched in columns 73-77, and a card sequence number punched in columns 78-80. The deck identification comes from columns 8 through 12 of the $CMPLE card.

B. Loader

The CT Loader during the loading operation may write on SYSOU. No other output will occur except operator messages on the on-line printer.

C. Subroutine Files Updater

The Subroutine Files Updater produces new subroutine files on SYSUT2 and may produce a map of these subroutine files on SYSOU, and/or messages on the on-line printer.

D. File Maintenance

The CT File Maintenance produces a new BCD system master tape on SYSLB4, an input tape for the assembler on SYSUT4 and SYSUT3, and writes error messages on the on-line printer.

*NOTE The output written on SYSOU from the Compiler and Loader is grouped 5 lines per physical record with record marks after each of the first 4 lines. If a 720 printer is used to list the tape, the group switch must be set to 5.

05.06 PROGRAM FLOW AND OPERATOR MESSAGES

The field test version of the CT Processor will write on-line messages concerning the progress of compilation, loading, and execution, of a CT program, through the various phases. In addition, tape switching and redundancy information may be issued by IOCS.

A. Compiler
The ability to interpret the significance of various conditions is helpful to the operator in determining if a compilation is progressing satisfactorily. For this reason, a brief resume of compiler action is as follows:

1. Phase 1 translates from the external statements to a condensed internal form.

    a. The first pass scans for data and procedure names, entering them and their descriptions into a dictionary.

    b. The second pass completes conversion of the text, determining the exact meaning of qualified names, and builds other tables of information necessary for the generation of instructions.

    The letters CTC are printed on-line when this phase is entered.

2. Phase 2 consists of the generation of instructions, and the initial pass of their assembly. The letters CTD are printed on-line when this phase is entered.

3. Phase 3 completes the assembly process, defining the relative locations of instructions and data, the listing of error messages, and the punching of the object deck. The letters CTE are printed on-line when this phase is entered.

Finally the word DONE is printed on-line when control is returned to CTM from the Compiler.

Unless a catastrophic error occurs (e.g., the omission of a division header), compilation will be completed regardless of the number of errors encountered. If a catastrophic error occurs, a standard end-of-job message will be printed and control will be passed to the CTM supervisor.

The Compiler uses the functional units SYSIN, SYSOU, SYSPP, SYSUT1, SYSUT2, and SYSUT3. It is itself located on SYSLB1; and when the COPY or LIBRARY options are available, it will use SYSLB2. Tape movement during compilation can be followed by becoming familiar with the table below.

Compilation Phase	Unit Function	Action	Usage
CTC- Pass 1	SYSLB1	Read	Input CTC
	SYSIN	Read	Input of source text
	SYSUT1	Write	Partial Internal text
	SYSUT3	Write*	Internal form of error messages
	SYSOU	Write	Partial List text
	SYSUT1	Rewind	End of Pass 1
CTC-Pass 2	SYSUT1	Read	Input Partial text
	SYSUT2	Write	Output final Internal text and External Dictionary
	SYSUT3	Write*	Internal form of error messages and symbolic instructions
	SYSUT2	Rewind	End of Pass 2
CTD	SYSLB1	Read	Input CTD
	SYSUT2	Read	Internal text
	SYSUT3	Write*	Internal form of error messages and symbolic instructions
CTE	SYSLB1	Read	Input CTE
	SYSUT2	Read	External dictionary
	SYSUT3	Read	Internal form of error messages and symbolic instructions
	SYSPP1	Write	Punch output
	SYSOU	Write	Completion of List output
	SYSUT1	Write ǂ	Tape for immediate LOAD

*SYSUT3 is an overflow tape, and may not be called into use for a very short program.

ǂ SYSUT1 is not written unless an immediate run is requested on the CMPLE card.

B. Loader

The successful loading of an object program will be accompanied by the following messages:

1. A list of the object-time tape assignments with appropriate mounting instructions will be printed on-line.

2. Just prior to control being turned over to the object program, a message will be printed and a computer stop will occur to insure that the necessary object-time tapes have been mounted. The operator should press the Start button when the mounting instructions have been carried out. (This procedure will undoubtedly change, but the field test version of the CT Processor will operate in this manner).

3. A message should now print saying that control has been turned over to the object program.

In addition, I/O messages concerning tape switching and redundancy may be printed on the on-line printer.

The CT Loader references the units SYSLB1, SYSIN's, SYSOU's, SYSUT1, SYSUT2, and SYSUT3. SYSUT1 is needed only if the run is of the type 'Compile and Load', in which case SYSUT1 contains the job to be loaded. Tape movement during loading consists of the following:

Loading Phase	Unit Function	Action	Action Description
First pass over text	SYSIN1 or SYSUT1	Read	Source of input
	SYSLB1	Read	Source of Loader and object subroutines
	SYSUT2	Write	Write text needing further processing
	SYSUT3	Write	Write text needing no further processing
	SYSUT2	Rewind	
Second pass over text which referred to locations undefinable during first pass	SYSUT2	Read	Read text needing further processing
	SYSUT3	Write	Write text defined in this pass
	SYSUT2	Rewind	
	SYSUT3	Rewind	
Read in object instructions and transfer to program	SYSUT3	Read	Read final program into core
	SYSUT3	Rewind	

A large section of core is reserved during the first two passes for text information (i.e. program instructions). It is only when this section cannot contain all the text information, that overflow text is written on SYSUT2 and SYSUT3. Also, it is more unlikely that SYSUT2 will be written upon than SYSUT3 For short programs, therefore, the only tape movement may occur on SYSLB1, SYSOU and the input tape (SYSIN or SYSUT1); and the loading process will appear to be a single pass operation.

C. Object-program execution

During the execution of an object-program, four different types of messages may appear on the on-line printer.

1. IOCS tape switching and redundancy information.

2. Items which are printed as the result of a source language 'DISPLAY' statement. The programmer may use this device to follow the flow of the object-program.

3. 'STOP' statements.
The STOP messages may be of two forms:
   a. STOP nnnnnn where nnnnnn is any number 6 digits or less. The computer will stop, and hitting the START key will cause the object program to continue in execution. (It is hoped that the programmer uses this instruction sparingly, if at all).
   b. STOP RUN
   This message means that object-time processing of the job is completed and control has returned to the CTM supervisor.
The STOP messages will be accompanied by the source language statement number at which the STOP occurred, i.e. AT xxxxx,yy STOP nnnnnn where xxxxx,yy is the statement number.

4. Object program error messages, usually concerning I/O errors. Object-time processing will terminate and control will revert back to the CT Monitor.

The Field Test version of the CT Processor will use the on-line printer for the DISPLAY, STOP statements, and object program error messages. (not SYSOU)

D. Subroutine Files Updater

The messages written on the on-line printer will be of two types:

1. Messages concerning any I/O redundancy and tape switching information.

2. A message signaling the successful or unsuccesful completion of the updating process.

The CT Subroutine Files Updater references the CT system tape, SYSUT2 and SYSUT3 (utility). The CT system tape is the input tape and SYSUT2 is the output tape.

E. File Maintenance

Messages written on the on-line printer will be of two types:

1. If errors occur during the File Maintenance procedure, a message of the form:

   (MESSAGE) LAST CHANGE AT Namennnn PRESS START TO CONTINUE

   is printed, and a pause occurs to allow operator action. Each (MESSAGE) which may print is described under Section 06.04.

2. At the end of the maintenance procedure a list of all files on the new BCD master tape in order of appearance is printed on-line.

The File Maintenance routine contains references to SYSLB3, SYSLB4, SYSIN, SYSUT3 and SYSUT4. Tape movement consists of reading the two input tapes SYSLB3 and SYSIN and writing the two (or three) output tapes SYSLB4, SYSUT4, and SYSUT3. At conclusion both SYSLB3 and SYSLB4 will rewind and unload, and SYSUT's will rewind.

05.07 OBJECT-TIME TAPE ASSIGNMENT

The assignment of units by the Loader for use by the object program depends on three variables:

1. The channel and unit arrangement at each installation.
2. The configuration of the SYStem units table at the time of the object-time assignment.
3. The object-programs' requested assignment.

The procedure by which the Loader assigns specific units to the various object-time files based upon the above variables is discussed in Section 06.02. Without prior knowledge of how the assignment is made for each job, the machine operator will have to wait for the tape mounting instructions given by the Loader before he can mount the tapes to be used by the object program. However, prior knowledge may be obtained by one of the following:

1. Repeated running of the object program without changing any of the above variables.
2. Understanding the method by which the Loader makes unit assignments. The method is simple and can be easily learned.
3. If a 1401 is available, a program can be written for the 1401 that will give the operator tape mounting and running instructions for the batch of jobs being card-to-taped on SYSIN.

SECTION 06

SYSTEMS MAINTENANCE

INTRODUCTION

The information in this section is intended for the person or persons at each installation responsible for installing and maintaining the Commercial Translator Processing System. This section contains information concerning:

1. Mimimum Machine Requirements
2. System Units and Object-time Tape Assignment
3. Updating Subroutine Files of System Tape
4. Symbolic Tape Maintenance

It is assumed in this section, that the reader is familiar with the contents of Section 04 and Section 05.

06.01 MINIMUM MACHINE REQUIREMENTS

The minimum machine requirements for 709/7090 Commercial Translator processing are:

1. 32,768 words of core storage.
2. Data-Channel-Trap
3. One on-line printer (Channel A)
4. Five tape units
   - A. One Binary System tape
   - B. One Listing Output tape
   - C. Three utility tapes
5. Two additional units
   - A. One on-line card punch or one more tape unit
   - B. One on-line card reader or one more tape unit

06.02 SYSTEM UNITS AND OBJECT-TIME TAPE ASSIGNMENT

A. System Units

The following table lists a set of possible installation 'standard' unit assignments for the SYStem units table and shows the usage of these units by the CT Compiler, CT Loader, CT Subroutine Files Updater, and CT Maintenance.

Symbolic Designation	Possible Configuration			Usage By			
	4 Channel	3 Channel	2 Channel	Compiler	Loader	Sbr. Files Updater	Maintenance
SYSLB1	A1	A1	A1	System	System	Old System Tape	System Tape
SYSLB2	None or A3	None or A3	None or A6	Source Language Library			
SYSLB3	None or A4	None or A5	None or A7				Old BCD Master
SYSLB4	None or B4	None or B4	None or B7				New BCD Master
SYSCRD	RDA	RDA	RDA				
SYSPRT	PRA	PRA	PRA				
SYSPCH	PUA	PUA	PUA				
SYSOU1	B1	B1	B1	First List Output	First List Output	First List Output	
SYSOU2	B1 or	B1 or	B1 or	Second List Output	Second List Output	Second List Output	
SYSIN1	C1 or RDA	C1 or RDA	A2 or RDA	First System Input	First System Input	First System Input	
SYSIN2	C1 or RDA or C3	C1 or RDA or C3	A2 or RDA or A5	Second System Input	Second System Input	Second System Input	
SYSPP1	PUA or D1	PUA or A4	PUA or B2	First Punch			
SYSPP2	PUA or D1 or D3	PUA or A4 or A6	PUA or B2 or B6	Second Punch			
SYSCK1	None	None	None				
SYSCK2	None	None	None				
SYSUT1	A2	A2	A3	Utility Input / Compile and Load			

table continued on next page

System Units Table Continued.

Symbolic Designation	Possible Configuration			Usage By			
	4 Channel	3 Channel	2 Channel	Compiler	Loader	Sbr. Files Updater	Maintenance
SYSUT2	B2	B2	B3	Utility	Utility	New CT Sbr. Files	
SYSUT3	C2	C2	A4	Utility	Utility	Utility	Second Output for Assembly
SYSUT4	None or D2	None or C4	None or B4				First Output for Assembly

The usage of the words 'first' and 'second' denotes a multi-reel file with alternating units on which tape switching is performed automatically by that part of the CT Processor which uses the file. If a single physical tape unit is assigned to such files, the operator must perform the necessary mounting of a new tape whenever a reel switch occurs (the first tape will rewind and unload and the unit will be selected again as the second or alternating reel).

The distributed system tape has the following assignment assembled into the system units table.

SYMBOLIC UNIT	PHYSICAL ASSIGNMENT
SYSLB1	A1
SYSLB2	(No unit provided)
SYSLB3	(No unit provided)
SYSLB4	(No unit provided)
SYSCRD	Card Reader Channel A
SYSPRT	Printer Channel A
SYSPCH	Punch Channel A
SYSOU1	B1
SYSOU2	B1
SYSIN1	A2
SYSIN2	A2
SYSPP1	B2
SYSPP2	B2
SYSCK1	(No unit provided)
SYSCK2	(No unit provided)
SYSUT1	A3
SYSUT2	B3
SYSUT3	A4
SYSUT4	B4 (Not used by CT SYSTEM)

The above assignment will suffice for Compiling and Loading Commercial Translator programs. The assignment was made to leave as many tapes free for object-time program tape assignment as possible. If units are assigned to SYSLB's the units assigned are not available for object-time assignment. Since the above configuration does not include unit assignments for SYSLB2, SYSLB4, SYSCK1, and SYSCK2, these SYStem units will be assigned by Basic Monitor Control Cards whenever corrections are sent to the installation involving these SYStem units.

B. Object-Time Tape Assignment

The assignment of physical units to the various files in the object program is performed by the Loader by processing the symbolic 'unit' columns (18-21 and 22-25) in the FILE cards which precede the binary object deck. For referencing purposes, the following definition of permissible symbolic units which may appear on the FILE cards is repeated. In the definition of permissible symbolic units requested by the programmer, the notation is as follows:

X	is a true channel request and may be A, B, C, D, E, F, G, H.
Y	is a symbolic channel request and may be S, T, U, V, W, X, Y, Z.
k	is a relative unit request and may be 1, 2, 3, 4, 5, 6, 7, 8, 9, 0.
M	is a true tape model request and may be II or IV

The permitted unit requests are:

a.	blank	
b.	Model only	M
c.	True channel	X
d.	Symbolic channel	Y
e.	True channel, Model	XM
f.	True channel, Relative unit	X(k)
g.	Symbolic channel, Model	YM
h.	Symbolic channel, Relative unit	YK or Y(k)
i.	Symbolic channel, Relative unit, Model	YKM
j.	For secondary unit assignment after all other requests have been fulfilled	*

In addition, the programmer may request the following SYStem units and card equipment.

IN, IN1, IN2	System input units
OU, OU1, OU2	System output units
PP, PP1, PP2	System punch units
UT1, UT2, UT3, UT4	System Utility tapes
RDX	Card reader, channel X
PRX	Printer, channel X
PUX	Punch, channel X

It should be noted that no true unit assignments may be requested. This follows from the fact that any given unit may be performing a SYStem function, and therefore not available for assignment as an object-time unit. However, the programmer is allowed to request relative units on channels. In explaining the meaning of relative units, the following example is given.

> In sorting problems, a programmer may use many tapes, alternately reading several and then writing several. From the logical flow the program, the programmer may decide that 3 files (for example) should logically be on one channel and 3 other files should logically be on another channel for best operating efficiency. The programmer might then use the symbolic unit designations A(1), A(2), and A(3) to refer to three of the files and B(1), B(2), and B(3), to refer to the other three files. The numbers (1), (2), and (3), are the programmers' indicies or relative unit assignment. The loader tape-assignment-routine will try to find three available units on channels A and B. Available,used in this sense, means units attached to the computer and not referenced in the SYStem units table (i.e. not used by the system). Using the distributed version of the SYStem units table as an example, the following assignment would be made:

SYMBOLIC	ASSIGNMENT MADE BY LOADER
A(1)	A5
A(2)	A6
A(3)	A7
B(1)	B4
B(2)	B5
B(3)	B6

> (This assumes there are at least 7 units attached on Channel A, and at least 6 units attached on Channel B).

The logical process by which symbolic unit requests are converted to physical unit assignments in the order of priority is:

1. All card units, SYStem units, and true-channel relative units are assigned first. The symbolic units OU, OU1, OU2, IN, IN1, IN2, PP, PP1, and PP2 when assigned to SYStem units take on the characteristics of the SYStem units. They are considered multi-reel alternating units with no labels, and any programmer rewind options are ignored. In this respect, for example, OU, OU1, OU2, are synonomous. The symbolic units UT1, UT2, UT3 should use the 'DEFER' mounting option as SYSUT1, SYSUT2, SYSUT3 are used by the loader. (Mounting instructions occur at object-time OPEN).

2. True channel, no specified units are assigned next.

   If, during steps 1 ana 2, there are insufficient units on the requested channels, a message prints to the effect that object-time tape assignment could not be completed because there were not enough units on the specified channels.

3. Symbolic channel assignment is now made on the basis of those symbolic channels with the largest requirement being handled first. Starting with the highest true channel, the channels are processed to see if they will handle the symbolic channel requirements. When a true channel is found that contains sufficient available units to handle the symbolic channel, that true channel is chosen. If there is no true channel which will handle the symbolic channel requirements, a true channel is chosen to assign as many of the symbolic units to the channel as possible. The remaining symbolic units are now assigned by repeating the same process (i.e. finding the symbolic channel with the greatest requirement etc.).

4. Model only specifications are now processed. Units are chosen starting at the highest channel number, first available unit.

5. Symbolic units with no specification in the UNIT1 columns on the file card are now assigned.

   If during steps 3, 4, or 5 there are insufficient tapes available for assignment a message prints saying there are not enough units on the system, and tape assignment could not be completed.

6. * Secondary units are assigned last and on the same channel as the first unit where this is possible.

When the UNIT2 columns on the file card are blank the unit assigned is always the same as the unit assigned to UNIT1.

06.03 Updating Subroutine Files of System Tape

The Subroutine Files will be updated by the SUBUP routine. The input tape is the tape from which SUBUP is called. The output tape is SYSUT2. SYSUT3 is used as a scratch tape for the updating of the subroutine files. Control will be given to the routine which updates the subroutine files when CTM recognizes a card of the form:

```
1          16
$SUBUP  [MAP]  [,nn]
```

This card is interpreted as follows:

[MAP] The MAP option produces on SYSOU1 a list containing pertinent facts concerning the Commercial Translator object-time subroutines.

[nn] This option causes a test to be made to insure that the proper number of processing cards were read. This is a numeric field containing the count of the number of update cards. This count does not include the $SUBUP card. The maximum number of cards which can be handled is 32,767. If the option is not used no check is made.

The Commercial Translator object-time subroutines consist of two files; the first containing a dictionary, and the second the subroutines themselves. The records in these files have the general form:

```
IORXN     K
BCI       1, Name
Record
```

All records have the same maximum blocksize. The X in the I/O word is P for all records of the same subroutine (i.e. same name) except the last, for which X = T. K is the number of the subroutine starting with zero for the dictionary, which has the name 'SRDICT'.

Control Cards

The functions available in the maintenance process are described under the various control cards.

*INSERT Card

The format of this card is

```
7           16
*INSERT     SR ( subroutine.name )   [,type]
```

A subroutine composed of the cards following the *INSERT control card is output onto SYSUT2.

*REPLACE Card

A card of the form

```
7           16
*REPLACE    SR ( subroutine.name )   [,type]
```

replaces the entire subroutine names 'subroutine.name' on the old system tape by a new subroutine formed entirely from the alteration cards following the *REPLACE card.

*INSERT and *REPLACE permit a second variable field parameter, 'Type', which can have the following values:

'SECONDARY'	This is to be treated as a Secondary subroutine during loading. This value is assumed if the field is left blank.
'PRIMARY'	This is to be treated as a Primary subroutine during loading.
Octal Location	This is to be treated as a Fixed subroutine and will begin at the given fixed octal location when loaded.

Secondary, Primary, and Fixed, subroutines are discussed in Section 03.

*REMOVE Card

If the card

```
7           16
*REMOVE     SR(subroutine.name )
```

is used, the subroutine named is omitted in the output on SYSUT2.

*AFTER Card

The card

```
7          16
*AFTER     SR ( subroutine.name )
```

suspends control card reading until the record named has been copied on the new system tape.

*REMARK Card

A card of the form

```
7          16
*REMARK    SR ( subroutine.name )
```

causes the characters in columns 16-72 of this card to be printed on-line at the time the card is encountered.

The opening parenthesis, subroutine.name, and closing parenthesis must immediately follow the 'SR' in all of the control cards described above.

Subroutine control cards must occur in the correct position in the modification deck and in the proper order. The subroutine deck which follows an *INSERT or *REPLACE control card must be a complete CT Relative binary deck, preceded by a *CTEXT card followed by a *CTEND card.

A scratch tape may be required for subroutine modification and SYSUT3 is used for this purpose. (Not currently required).

When the last subroutine modification has been processed, the library tape is positioned beyond the file mark which terminates the old subroutine file.

Updating Procedure

The following procedure is used in preparing a new system tape:

1. Prepare the input as described. A typical sequence might be:

```
1          7              16
-------------------------------------------------------------
$DATE                     mmddyy          any number of unit reassignments
$ATTACH                   unit            if and as needed by installation
$AS                       function.name
$EXECUTE                  CT
$SUBUP
    subroutine decks
```

This updates the subroutine files and puts them on SYSUT2. They are placed on the new system tape by the following sequence:

```
$IBSYS
$IBEDT
        *EDIT       options
        *AFTER      LOAD
        *DUP        SYSUT2,SYSUT1,2

end-of-file card

$IBSYS
```

The control cards must be so ordered that their record name references are in the same sequence as the records on the tape. If a record is referenced out of sequence in the update deck, a message will be printed, and its effect will be nullified.

2. Load and set tape units in accordance with the standart (or altered) settings of the SYStem units table.

3. Follow Basic Monitor operating instructions. Refer to the manual, IBM 7090 Operating Systems: Basic Monitor (IBSYS) Form J28-8086-0.

A map of the system tape may be obtained on SYSOU by specifying MAP on the *EDIT card to IBEDT. CT may be located any place on any of the system library tapes. The output map of CT is as follows:

File	Name	Purpose
n	CTM	Comm. Trans. Supvsr:
n + 1	BASIC	Basic IOCS
n + 2	CTB	CT Compiler Service
	CTC	CT Compiler, Phase 1
	CTD	CT Compiler, Phase 2
	CTE	CT Compiler, Phase 3
n + 3	IOCS	Label IOCS
n + 4	LOAD	Loader
n + 5	SRDICT	CT Subr. Dictionary
n + 6	CTSUBT	CT Subroutine Texts
n + 7	IOBB	Basic IOCS Overlay
	IOBM	Minimum IOCS Overlay
	NOBS	No IOCS Overlay
n + 8	SUBUP	Subroutine Updater
n + 9	MAIN	Symbolic Maintenance

06.04 SYMBOLIC TAPE MAINTENANCE

The Symbolic Tape Maintenance program will update a master tape of card files, optionally producing a new master tape, and selectively producing a second output tape intended for assembly or job input. Information, as directed, is taken from the master tape (SYSLB3) or from control and alteration cards on the input tape (SYSIN1). If a new master is created, it is written on (SYSLB4); while the second output is written on SYSUT4, and , as an alternate unit, SYSUT3. SYSLB3, and SYSLB4 must be single-reel files. No provision is made for end-of-reel tape switching on SYSLB3.

Control is given to the Symbolic Maintenance program when CTM recognizes a control card of the form:

```
1       16
$MAIN  [M]   [,B]  [,C]
```

The interpretation applied to the variable field of this card is:

[M]	Create a new master tape.
[B]	The second output tape is to be blocked 10 records per block.
[C]	This specifies that the second output tape is going to serve as input to the Commercial Translator Compiler. The B option should not be used when the tape is intended for input to the Compiler.

Tape Formats

Each file on the master tape consists of 14 word card records, the first of which may have either of the following formats:

```
1       7       15
*       *FILE   Name      or
        8
Name    *FILE
```

Only the latter form will be written on the new master tape where 'Name' is a four character (or less) identifier for the file. The identifier is used to reference a given file, and forms part of the sequence numbers attached to the cards of the file.
The sequence numbers are created by the routines when this file is first placed on the master tape. Their form is:

Namennn

and they occupy positions corresponding to columns 73-80 of the card. Individual cards of a file are referenced by these numbers, in order to indicate the nature and extent of any changes which are to be made. Enumeration begins with 1, on the first card following the *FILE card, and is in unbroken sequence. Since there are only four digits, numbers from 10,000 to 10,999 use + as the first digit, numbers from 11,000

to 11,999 use A as the first digit, numbers from 12,000 to 12,999 use B as the first digit, etc. A single file is limited to 99,999 cards. When a file is modified, new sequence numbers are automatically created reflecting the re-ordering necessary to account for any insertion and deletions. The *FILE card is never written on the second output tape.

The last file of a master tape is exceptional. It must consist of a single card which may have either of the following formats:

```
1       7
*       *END            or
        8
        *END
```

which serves to delimit the extent of the tape. Only the latter form will be written on the new master.

An end-of-file mark must follow the last card of every file on the master tape.

A master tape created by the program is alwyas blocked 10 records per block, and may be listed on the 720A with the group switch set in the "10" position. If the blocking option B is chosen for the second output tape, however, it cannot be listed on the 720A. The input tapes, both master and alteration, will process correctly with any blocking factor of 10 or less.

If the C option is used on the $MAIN card, the second output tape, SYSUT4, if of the following form:

1. The first record on SYSUT4 is $EXECUTE CT
2. Three additional records are placed after the last end-of-file on the SYSUT4 tape:

```
$IBSYS
$*MAIN JOB STACK COMPLETED
$SWITCH         SYSIN1, SYSUT4
```

The first record of each file on SYSUT4 after the $EXECUTE card should be a CTM control card (usually $CMPLE card). By this method, it is possible to maintain source programs on BCD master tapes; and, as desired, change, compile, and execute particular programs in one operation. The form of SYSIN for this type of use of the Commercial Translator Processor might be:

```
$DATE          mmddyy              Current Date
$ATTACH        D2
$AS            SYSUT4              (Low Density)
$EXECUTE       CT
    *MAIN      C                   Go to Maintenance Routine
    *FILE      A                   Changes to source program on master tape
changes
.
End-of-file card
$IBSYS                             Return to Basic Monitor
$SWITCH        SYSIN1,SYSUT4       Switch input units and process the source program
$PAUSE
```

Control Cards

The control cards used to direct the Symbolic Maintenance program fall into two categories.

1. Cards used to initiate action on a given file are called major control cards. Their format follows this general pattern:

   ```
   1       8          16
   Name  {operation} [option]
   ```

   Changes to the file 'Name' can be made only if it is referenced by one of these cards. If not, the file is copied, exactly as it stands, onto the new master tape: or, if there is no new master, it is simply ignored.

2. Modifications within a file are described by the minor control cards. The format pattern of these cards is:

   ```
   1       8          16
   Name  {operation}  mmmm,nnnn
   ```

   A card of this form following a major control card which references file 'Name indicates that modification is to occur from sequence numbers mmmm to nnnn inclusive. Cards following the monor control card are used for the modification; their extent being delimited by the occurrence of another control card.

Any number of non-control cards may follow a minor control card; and any number of minor control card sets may follow a major control card. Ordering of these cards, however, is essential. Major control cards must refer to the files in their order of appearance on the master tape. Minor control cards must be ordered by sequence within a file; and the sequence numbers cannot overlap. Cards which occur out of order result in an error message (see below), and are ignored if updating continues.

It should be noted that a card with an * in column 1 is not recognized as a control card, irrespective of the content of the remainder of the card. This permits commentary to pass through the system without accidentally being treated as a control card. A consequence of this, or course, is that file identifiers cannot begin with the character *; they are otherwise unrestricted, however.

*FILE CARD

The major control card

1	8	16
Name	*FILE	[{A} {M}]

initiates action of the file 'Name.' If in the varable field:

1. A is chosen, the updated file is written onto the second output tape, SYSUT4.

2. M is chosen,the updated file is written onto the second output tape, SYSUT4, and an end-of-file mark is placed behind it.

If neither option is exercised, the updated file is not written onto the second output tape.

If the *FILE card is not followed by minor control cards, the result is simply to produce sequence numbers on the cards of the file. In this way, a master tape prepared by some non-system program or method (card-to-tape, for example) can be easily converted to the standard format.

*RENAME CARD

A file may be renamed at the same time it is modified by means of the major control card:

1	8	16	36
Nam1	*RENAME	[{A} {M}]	Nam2

"Nam2" replaces "Nam1" as the name of the modified file. In all other respects the *RENAME card functions exactly like the *FILE card.

*NEW CARD

A file may be added to the master tape by means of the major control card:

1	8	16
Name	*NEW	[{A / M}]

'Name' becomes the file identifier, and the options are the same as described under the *FILE card.

When the *NEW card is encountered, a master tape identifier (*FILE) card is generated and placed on the output tape(s). Cards following the *NEW card become the content of the new file, their extent being determined by the occurrence of another(major) control card. A minor control will also terminate the file, but should not be used in as much as the maintenance program assumes that the new file is not sequenced and, consequently, not subject to modification.

*DELETE CARD

An entire file, including the *FILE identifier card, is removed from the tape by use of the major control card:

1	8
Name	*DELETE

A file which is being deleted is not written on any output tape, and is not subject to any modification.

*ALTER CARD

Changes on a given file are described most often using the minor control card:

1	8	16
Name	*ALTER	mmmm [,nnnn]

When this *ALTER card is recognized, the modification process is as follows:

1. Cards up to, but not including, sequence numbers mmmm are placed on the output tape(s), with sequence number adjustment if required.

2. If nnnn appears, cards sequenced from mmmm through nnnn inclusive are deleted, and the cards which follow the *ALTER card are put in their place, until the occurrence of another control card. The number of cards inserted need not be the same as the number deleted.

3. If nnnn does not appear, no deletion occurs, and the cards are simply inserted before the card sequenced mmmm.

'Name' must agree with the file identifier or the card is not effective. Leading zeros may be suppressed in the variable field for both mmmm and nnnn, and a blank may be used in place of the comma. For convenience, the operation may also be written *A instead of *ALTER.

*INSAS CARD

The minor control card

1	8	16
Name	*INSAS	mmmm

is used when an insertion is to be made only on the second output tape. Cards following the *INSAS card are inserted on this tape prior to the card sequenced mmmm. The sequence number is not advanced during the insertion, so that the sequence numbers of cards which appear on both the master and second output tapes always agree. For convenience the operation may also be written *IA instead of *INSAS.

This type of insertion is useful if the second output tape is intended for assembly input, since LIST, UNLIST, END and other special cards can be placed where desired, without affecting the file kept on the master tape.

Another very important use is to create *MAIN type control cards when the second output tape is intended for use as the input of another symbolic update. A control card cannot normally be placed on an output tape, since recognition of such a card terminates any modification sequence and begins another. For this reason, a card under the scope of an *IA card which has the character combination *+* in columns 1-3 is special. For such a card, columns 1-6 are replaced by columns 25-30, and columns 25-30 are blanked out before the card is placed on the second output tape.

*NOTES CARD

When modifying a file of symbolic instruction cards, it is sometimes convenient to replace only the commentary field of certain cards. The minor control card

1	8	16
Name	*NOTES	mmmm [,nnnn]

is for this purpose, and functions as follows:

1. Cards up to, but not including, sequence mmmm are placed on the output tape(s).

2. If nnnn appears, the cards sequenced from mmmm to nnnn inclusive are deleted. Cards which follow the *NOTES card are then matched one-for-one with cards on the master file following sequence number nnnn. Columns 36-72 of these cards replace columns 36-72 of the corresponding master file card. Columns 1-35 of the card are not disturbed.

3. If nnnn does not appear, the commentary replacement begins with the card sequenced mmmm.

4. The process terminates with occurrence of another control card, or when the file 'name' is exhausted of cards.

Care should be exercised in using the *NOTES card in order to avoid intrusion into the scope of a minor control card which follows. For, since the *NOTES card results in passing cards of the master tape beyond the deletion (if any). the first sequence number which can enter into the next change is:

p+mmmm or p+nnnn+1

where p is the number of cards within the scope of the *NOTES card. For convenience, the operation may be written *N instead of *NOTES.

*END CARD

Upon recognition of the major control card:

```
8
*END
```

the changes applicable to the last file referenced by a major control are completed, the remainder of the master tape is copied onto the new master (is any) and control is returned to CTM. An *END card is simulated if an end-of-file condition is encountered in the input.

CHANGE STATUS INDICATORS

For convenience in comparing the updated tape with the old, the change status of every card in the file referenced by a major control card is indicated in column 83 on both the new master and second putput tapes. If this column contains:

1. A blank, the card was not changed;
2. An asterisk (*), the card was inserted;
3. A dollar sign ($), the card was commented;
4. A plus sign (+), the card is an insert on the second output tape only;
5. A slash (/), the card is the first one following a deletion.

MESSAGES

Just prior to return to CTM, if a new master tape was created, a list of the files in order of appearance on the tape is printed on-line. If an error is detected, a message of the form:

(MESSAGE) LAST CHANGE AT Name Nnnn PRESS START TO CONTINUE

is printed, and a pause occurs to allow operator action. The messages which may print are:

1. CANNOT FIND FILE 'name'

   Either 'name' is mis-spelled or the *FILE card is out of order. In either case, if the start button is depressed, the program continues as if an *END card had been encountered.

2. CHANGE READ ERROR

   Permanent read redundancy on the input tape. The program will continue, accepting the input as it stands, if the start button is depressed.

3. OLD MASTER READ ERROR

   Permanent read redundancy on the input tape. The program will continue, accepting the input as it stands, if the start button is depressed.

4. *FILE CARD MISSING

   The first card of a file on the master tape is not a *FILE identifier or an *END card. The program will continue normal operation if the start button is depressed. The master tape should be replaced, however, since modifications to the file with the missing identifier are not possible.

5. *ALTER OUT OF SEQUENCE

   If the start button is depressed, the out of sequence set is ignored and normal operation continues.

6. MAJOR CONTROL MISSING

   The first card of the change sequence is not a major control card. If the start button is depressed, the program continues normally, ignoring all changes cards prior to the first major control card. This occurrence is normal if the input tape was prepared as the second output of a previous update for which the C option was not used, since some *FILE identifier card must alwyas be first on such a tape.

7. MINOR CONTROL MISSING

The first card following a major control card is not a minor control card. If the start button is depressed, the program continues normally, ignoring all cards on the input tape until the next control card.

8. MINOR CONTROL ERROR

The identifier 'name' on the minor control card does not agree with the identifier of the file. If the start button is depressed, the program continues normally, ignoring all cards on the input tape until the next control card.

9. SEQ ERROR ON MASTER FILE

A gap in the sequence numbers was found while positioning a file to the first card involved in some modification set. If the start button is depressed, the program continues normally, ignoring all cards on the input tape until the next control card. The sequence number gap will be repaired. Such an error can only occur on a master tape which was not prepared by the system.

10. UNEXPECTED END OF FILE

The master tape contains some end-of-file mark other than the normal file separators. If the start button is depressed, operation will be continued, although recovery is probably not possible. The master tape should be replaced.

APPENDIX 90.01

DEFERRED FEATURES, RESTRICTIONS, AND LIMITATIONS

90.01 Deferred Features, Restrictions and Limitations

A. Compiler

1. Language

a) Procedure

i CALL verb

Subscripts may not be used in specifying the (old.name). Both of the following usages are erroneous:

```
CALL    (A(J))B.
CALL    (A(3))B.
```

ii CRYPT

Qualified names may not be referenced by CRYPT instructions. The following usage is incorrect:

```
CLA     MASTER     NAME
```

The verb CALL may be used to reduce a qualified name to a single unique name.

iii DISPLAY verb

Data names not intended to be used as qualifiers must be separated by commas; otherwise all but the last will be disregarded.

iv GET verb

Card files processed on-line may only be in BCD and fixed length for field test.

All input records containing arrays will be processed in the transmit mode by the field test processor. This is true for both fixed and variable length records.

Records from different files which have been REDEF'd together will not be automatically transmitted by the field test processor (see 02.07.05 c-ii for description of the feature). SPANS or HOLD must be used.

v. INCLUDE verb

Mechanization of the INCLUDE verb has been deferred, and consequently no library facilities are currently available.

vi Indexing (including the DO verb)

A repeated usage of a subscript name should be avoided when possible. When a subscript changes value, all positional indicators containing that subscript will be re-evaluated. Use of the same subscript names for multiple purposes generally causes unnecessary calculations in the object program, but it will not be incorrect.

The dimensions of an array must be set before calculation of subscript values associated with the array is performed. Presently, failure to observe this restriction causes invalid object code.

For example, the following sequence of instructions is incorrect:

```
SET I = 3, SET J = 5.
SET Q = 10.          (where the data description shows Q
                      occurrences of A)
MOVE A (I, J) TO R.
```

In this case, the positional indicator for A(I, J) is computed when I and J are set and not updated when Q is set. The system presently causes all records containing arrays to be transmitted. However, care should be taken when an array follows a variable length array. In this case, the positional indicator will not be updated when the base is determined. Therefore, the subscripts must be set prior to making a subscripted reference to the array.

A DO section will always be performed at least once regardless of the values of the loop control variables.

No object time check is made to insure that subscript references conform to the limits specified by the array dimensions in the Data Description.

vii MOVE verb

Figurative constants may not be moved to variable length arrays; they may, however, be moved to fixed length arrays or to array elements.

viii LOAD and OVERLAP verbs
Implementation of these verbs has been deferred.

ix Miscellaneous
Procedure.names not followed by a period (.) and blank are handled properly; no diagnostic message is given.

A period (.) followed by a blank in a procedure statement terminates analysis of the statement except when they appear within an alphabetic literal; any information in the same card which follows the period is assumed to be commentary. No diagnostic comment is made.

b) Data Description

i COPY type code
Implementation of COPY has been deferred.

ii RECORD type code
A name associated with the RECORD type code must be unique. If the record is defined within a section the section.name may not be used as a qualifier of the record.name.

Since a quantity specification may not be made at the record level, record.names may not be subscripted.

iii REDEF type code
Whenever an input record containing an array is involved in a REDEF, the record containing the array should precede the REDEF in order to get optimum results. An area involved in a REDEF must not contain subscript variables or a QUANTITY.IN value.

iv Pictorials
A field may not be described as a mixed numeric and alphameric field, i.e., both A's and 9's in a pictorial. if this specification is made the field is treated as alphameric.

v Miscellaneous
Neither a data item (literal) nor a condition.name may be subscripted. However, a conditional variable may be subscripted.

Fields may not be defined following a variable length array as part of the same data hierarchy.

c) Environment Description

i CONTRL card

Mechanization of multiple deck combination has been deferred. CONTRL specifications will have no effect on the object deck produced by the Compiler, i.e., no control break table is punched.

ii FILE card

A maximum of 63 files may be described.

No file check table is produced in the object deck.

CARD, BINARY and locate/transmit mode restrictions are discussed with the GET verb (A. 1. a. iv. above).

iii SPECIF card

Absolute tape assignment cannot be made; symbolic and relative assignment techniques are used (see section 02.06).

Unless system utility tapes (UTk) are explicitly designated the drives assigned to them are not used at execution time as object program units.

iv Miscellaneous

Qualified names may not be used in the Environment Description.

2. Internal Table Size Limitations

		Appox-Max Size
a)	Internal dictionary including all program names whether defined by the programmer or generated by the Compiler.	3500
b)	Number of SECTIONS.	35
c)	Number of different edited field formats.	35
d)	Number of base locators (for the field test version this is the number of located records).	127
e)	Number of QUANTITY IN specifications.	25
f)	Depth of nested sections.	18
g)	Number of index expressions (a * VARIABLE ± b)	50
h)	Number of positional indicators (each unique combination of array and subscript notation, e.g., A(I), A(I + 1) B(I) requires 3 positional indicators.	90
i)	Number of array dimensions (explicit or implicit Quantity specifications).	85
j)	Number of levels in a data hierarchy.	23
k)	Number of generated constants in the constant pool -CP)+NN	500

B. Loader

1. The control break table is not processed by the loader. This implies that multiple program loading with cross references by means of control breaks is not available.

2. Deferred implementation of the LOAD-OVERLAP verbs implies that programs may not be segmented into separate memory loads at this time.

3. No debugging facilities are currently available.

4. Normally the MINIMUM module of IOCS is used with the object program. If checkpoints are desired or specified BASIC IOCS is used. If labeling exists the LABELS version of IOCS is necessary.

5. The Compiler accepts without comment a deck.name containing imbedded blanks and punches it in Loader symbolic control cards. A deck.name of this form is not acceptable to the Loader and will prevent execution of the object program.

APPENDIX 90.02

GENERATED CODE OF THE COMMERCIAL TRANSLATOR

INTRODUCTION

The normal listing output of the Commercial Translator consists of the original source statements augmented by a sentence number, an indication of errors detected during compilation, and a file list.

In addition, an assembly-like listing of the instructions generated by the compiler may be obtained by including the word LIST as one of the parameters on the $CMPLE card. This appendix explains the format and content of the instruction listing.

SYMBOLIC LISTING

The symbolic listing is composed of four columns under the headings:

A. LOC
B. OCTAL
C. CNTRL
D. SYMBOLIC

A. LOC

Each octal number under the heading LOC represents the displacement of the associated word from the first word of the object program (not necessarily PROGRAM.START). A blank space in this column accompanied by USE in the symbolic part of the listing represents a location counter discontinuity. The location counter discontinuity results from the compiler's use of three different 'pseudo' location counters during compilation. The three 'pseudo' location counters have been given the names:

Location Counter 0
Location Counter 1
Location Counter 2

Location Counter 0 is used by the compiler for the generation of in-line instructions, i.e., those instructions whose location is determined by the relative position of items in the source deck. Instructions generated under Location Counter 0 represent:

1. The main body of procedure text
2. Storage reservation for fixed location Data areas, i.e., data not located in an input/output buffer.

Location Counter 1 is used by the compiler for the generation of out-of-line subroutines and working storage areas which are used by the main body of the program. The subroutines generated under Location Counter 1 are, in general, routines which update 'pointer words'. A 'pointer word' is a word that points to (or addresses) a particular data item. If the location of the data item changes, the word pointer must also change. Data items which may change locations include:

1. Items located in buffer areas
2. Items within an array or following an array

Location Counter 2 is used by the compiler in initializing 'pointer words' with predetermined constants.

References to the three 'pseudo' location counters occur at LOC discontinuities in the form of:

USE N

where N is 0, 1, or 2. It might be noted in the listing that the first instruction generated when Location Counter 0 is in use has the initial Displacement of 0 in the LOC field. Also, the first instruction generated when Location Counter 1 is in use will have, in the LOC field, a value one greater than the highest value obtained under Location Counter 0.

In other words, Location Counter 1 begins where Location Counter 0 ends.

B.-C. OCTAL CNTRL

Each 12 digit octal number under the heading OCTAL represents a word punched in the object deck. Each 5 digit binary number under the heading CNTRL specifies additional information about the OCTAL word. The 5 digit number is also punched in the object deck and controls the loading of the OCTAL word. A detailed explanation of the object-deck format and the CNTRL and OCTAL words is given in Appendix 90.03.

D. SYMBOLIC

The symbolic part of the listing is a 'SAP' -like listing with a few modifications, namely:

1. Printing of names up to 30 characters in length.
   If a statement name exceeds 15 characters the associated symbolic instruction is printed on the following line.
2. Printing of multiple names.
   If more than one name is associated with an instruction, each name is printed, one name per line.
3. Printing of identical names.
   If the programmer makes use of the name qualification feature in the source program, as in the following example:

Name	Level	Description
*DATA		
A	01	
C	02	A
B	01	
C	02	A
*PROCEDURE		
Move A C to B C.		

The reference to A C will appear in the symbolic listing as 1)C and the reference to B C will appear as 2)C. This results from the fact that the compiler sequentially numbers each identical name it encounters in the source program and it is this number which accompanies each reference to the name in the symbolic listing. (Note: In the reservation of storage for *DATA areas, only those names which represent the lowest level number in the data area will be printed on the left side. For the above example, the data part of the listing would appear as:

*DATA	BSS	0
A	BSS	1
B	BSS	1

4. Compiler generated names

The symbolic listing will contain many references to names not found in the source language. These compiler generated names have the same form; a two or three letter symbol, followed by a right parenthesis, followed by a number, i.e.:

SYM)NNN

The symbol part, SYM, may be one of the following:

	Symbol	Abbreviation For
I.	RS	Result Storage
II.	TS	Temporary Storage
III.	BL	Base Locator
IV.	PI	Positional Indicator
V.	CP	Constant Pool
VI.	GN	Generated Name
VII.	SYS	System Subroutine
VIII.	IOC	I/O Subroutine or Communication

The number NNN specifies a particular item in one of the 8 groups. In the case of constants (CP references), the designation CP)+NN is used.

1. RS - Result Storage

Result Storage cells are reserved by the compiler to temporarily store the results of a CT statement. Unique result storage cells are reserved for each section of the program as they are needed. Examples of instructions which refer to Result Storages might be:

```
SLW     1.RS)0
LAS     2.RS)2
```

The number preceding the decimal point is a section-number and the number following the right parenthesis is a particular result storage within the section. Section numbers begin with 0 (main body of program) and increase by 1 for each 'Begin Section' in the source program. Near the end of the symbolic listing (after the last instruction generated by the Procedure Division) the following instructions will appear:

```
        USE     1
RS)     BSS     N(Omitted if no Result Storages are required)
```

These instructions reserve the necessary cells for all Result Storage. N is the sum of maximum Result Storage used in each section.

II. TS - Temporary Storage

Temporary Storage cells are generated by the Compiler to temporarily store an item. The difference between Temporary Storage cells and Result Storage cells is that Temporary Cells are not section-qualified. An example of an instruction referring to Temporary Storages might be:

```
ANA     TS)1
```

Temporary Storage cells are reserved immediately after Result Storage cells by the instruction.

```
TS)     BSS     N
```

where N is the total number of cells to be reserved.

III. BL - Base Locator

Base Locator cells are a type of 'pointer word'. These pointer words address one of two types of data, namely:

1. Records located in a buffer.
2. Data located after a variable length array.

References to Base Locators occur in the symbolic listing whenever the pointer word must be changed. This type of reference to a Base Locator is usually part of a 'GET' calling sequence to the Input/Output system. An example of a sequence that changes the value of a Base Locator due to locating data within an input buffer might be:

```
TSX     IOC)8,4
PZE     FILENAME,,SYS)260
PZE     END-OF-FILE-PROCEDURE,,ERROR-PROCEDURE
IOCDN*  BL)2,,14
```

This sequence of instructions fills in the cell BL)2 with the location of the first word of a 14 word record which is in an input buffer.

Following the updating of a Base Locator further references to BL's may occur in the form:

```
LAC     BL)2,4
CAL     DATANAME,4
```

References to data located by means of a Base Locator is usually done in this manner, i.e., by first loading an index register with the 2's complement of the location of the first word (the base) of the data block and then referencing any data item within that block by using the relative word position or displacement from the beginning of the block.

If, in the above example, the Data organization for DATANAME was:

NAME	LEVEL	TYPE	FORMAT
*DATA			
ARECORD	01	RECORD	
ITEM	02		A(6)
DATANAME	02		A(6)

and ARECORD is located in an input buffer, the relative word position of DATANAME would be 1 and the symbolic instruction

```
CAL     DATANAME,4
```

would generate the machine instruction:

```
CAL     1,4
```

The form of a Base Locator word is:

```
PZE     LOC,,BYTE
```

where LOC is the word address of the first word of the data organization, and BYTE is the position in that word (0-5) of the first data item. By definition, a 'simple' Base Locator is one with BYTE always 0. Input records are located by 'simple' Base Locators. A 'complex' Base Locator, by definition, is one that locates items after a variable length array. The BYTE of a 'complex' Base Locator may range from 0 through 5 depending on the byte length of the array.

IV. PI - Positional Indicator

Positional Indicator cells are a type of 'pointer word'. These pointer words locate items within an array. When a CT operation such as SET causes a subscript to change value, the corresponding positional indicator is modified. A modification of this type might appear in the listing as:

```
CLA     CP)+1
ADD     PI)6
STO     PI)6
```

Storage for Positional Indicators is reserved immediately following the Base Locators by the instruction:

```
PI)     BSS     N
```

N is the total number of Positional Indicators.

V. CP - Constant Pool

Constants needed by the object program are pooled together in a block of storage immediately following the Positional Indicators. An example of a reference to Constant Pool cells might be:

```
CAL     CP)+11
ANA     CP)+22
```

VI. GN - Generated Name

During compilation it is necessary to give certain instructions names in order to refer to these instructions from a remote part of the program. Names are generated and placed on such instructions and might appear in the listing as:

```
        TSX     GN)019,4
GN)019  CLA     PI)3
```

VII. SYS - System Subroutines

There will be many references in the listing to subroutines which are not part of the binary deck produced by the compiler. Typical references to such subroutines might appear in the listing as:

```
STI     SYS)133
TSX     SYS)182,4
TXI     SYS)243,1,84
```

These System Subroutines are contained on the System Tape and are brought into core, as needed, by the CT LOADER. The function of each of these routines and an explanation of the calling sequences that appear in the listing are explained in the latter part of this Appendix.

VIII. IOC - Input Output Subroutines and Communication

References to Input/Output subroutines and communication cells appear in the listing in the form IOC)NNN. Typical references to IO routines or communication cells are:

```
TSX     IOC)9,4
MZE     IOC)17
PZE     IOC)29
```

Most of these are routines that are a part of the CT monitoring system which is in core at all times. The function of each routine is explained in the latter part of this Appendix.

CT SYSTEM SUBROUTINES AND COMMUNICATION CELLS

The Commercial Translator object deck makes references to two types of routines not contained in the deck itself. These two types of subroutines are:

Type 1. Subroutines contained on the system tape which are placed in core as part of the CT Monitoring System and remain in core to be used by any part of the CT System, Compiler, Loader, Object Program, Updater, File Maintenance or Accounting Routine. These subroutines, for the most part, are concerned with Input/Output functions.

Type 2. Subroutines contained on the system tape which are placed in core by the CT Loader as required by the particular object program being loaded.

In addition to System Subroutines, there are individual cells and groups of cells which contain information which must be accessible to various parts of the CT system. These Communication Cells are categorized in the same manner as System Subroutines, i.e., those communication cells which are part of the CT Monitoring System are Type 1, and those communication cells which are a part of a particular object program are Type 2.

Each of these System Subroutines and Communication Cells has been assigned a number. Type 1 entries received a number between 1 and 127 and Type 2 entries received a number greater than 127. In the symbolic listing of a compiled program, references to Type 1 entries will be of the form:

IOC)NNN

where NNN is the assigned number between 1 and 127. Type 2 entries will appear in the symbolic listing as:

SYS)NNN

where NNN is the assigned number greater than 127.

The following pages list the CT Subroutines and Communication Cells which may appear in the symbolic listing of a CT program.

IOC REFERENCE NUMBERS

A knowledge of the 7090 IOCS is necessary in understanding most of the IOC Numbers.

SYSTEM REFERENCE	FORMAT and/or FUNCTION
IOC)1	PZE L,,N A cell in the CT Monitor communications area which locates (L) a list of files, and designates the number (N) of files in the list. This List is used in Opening and Closing files.
IOC)2	PZE L,,12*N A cell in the CT Monitor communication area which locates (L) a number (N) of IOCS file blocks. I/O information is kept in each of the 12 word blocks for every file in the CT program.
IOC)3	The entry point to the IOCS DEFINE subroutine.
IOC)4	The entry point to the IOCS JOIN subroutine.
IOC)5	The entry point to the IOCS ATTACH subroutine.
IOC)6	The entry point to the IOCS CLOSE subroutine.
IOC)7	The entry point to the IOCS OPEN subroutine.
IOC)8	The entry point to the IOCS READ subroutine.
IOC)9	The entry point to the IOCS WRITE subroutine.
IOC)10	The entry point to the IOCS COPY subroutine.
IOC)11	The entry point to the IOCS REWIND subroutine.
IOC)12	The entry point to the IOCS WRITE-END-OF-FILE subroutine.
IOC)13	The entry point to the IOCS BACKSPACE-RECORD subroutine.
IOC)14	The entry point to the IOCS BACKSPACE-FILE subroutine.
IOC)15	The entry point to the IOCS CHECKPOINT subroutine.
IOC)16	The entry point to the IOCS STASH subroutine.
IOC)17	The entry point to the IOCS MESSAGE-WRITER subroutine.
IOC)29	This is a 14 word area within IOCS which is used to process all labels.

IOC)40 This is the end of job return point in the CT Monitor communication area for all CT jobs.

IOC)46 This is the first cell of the CT Monitor communication area.

IOC)53 This is a cell in the CT Monitor communication area which contains the current date in BCD in the form:

MM DD YY

MM is the month
DD is the day
YY is the year

IOC)54 This is a cell in the CT Monitor communication area which contains the current time in BCD. (Subject to each installation providing an accounting routine which fills in this cell).

SYS REFERENCE NUMBERS

Many of the SYS Reference numbers are concerned with subroutines to move fields at object time. The interaction of various of these 'MOVE' subroutine made it desirable to package several of them together into one subroutine (called MOVPAK).

In discussing the various types of moves the following abbreviations and definitions are used.

1. Alphameric fields
   - AF: Alphameric Field (pictorial contains only A's and X's)
2. Fixed Point Numeric fields
   - XD: External Decimal (pictorial contains only 9's, S's, V and $\overset{+}{9}$ or $\overset{-}{9}$)
   - ID: Internal Decimal (pictorial contains only 9's, S's, V; IR given as mode and justification).
   - IDnj: Internal Decimal not justified (same as ID but justification not given as R)
   - EF: Edited Field (pictorial contains one or more of the following characters: 8 + - . , * or the clause Blank When Zero
3. Numeric field with exponent
   - SD: Scientific Decimal (pictorial represents signed mantissa followed by signed exponent)
   - FP: Floating Point (pictorial is F or FF)
4. Figurative constants
   - BL: Blanks
   - ZE: Zeros
   - HV: High values
   - LV: Low values
5. Literals
   Literals are classed as AF, ID, or FP.

SYSTEM REFERENCE	FORMAT and/or FUNCTION
SYS)128 SYS)129	These two cells serve as storage for multi precision arithmetic operations.
SYS)130	This cell is set non-zero whenever any one of the numeric move or convert subroutines of MOVPAK detects the truncation of significant high order values (i.e. overflow).
SYS)131	This cell is set non-zero whenever any ohe of the numeric move or convert subroutines of MOVPAK detects an improper data condition.
SYS)132	PZE LOC,,BYTE This cell points to the first word address (LOC) and first BYTE (0-5) of the source field involved in a Move. The cell is set by one of the following: 1. It is set automatically by calls upon subroutines SYS)179 or SYS)181 (see calling sequences).

2. It is set by in-line coding preceding calls upon subroutines SYS)180 or SYS)182 (see calling sequences). The in-line coding which sets the cell will be one of three forms:

Case 1. When the data item is working storage

```
LDI   CP)+NN
STI   SYS)132
```

where CP)+NN will contain the location and byte of the data item.

Case 2. When the data item is located by means of a 'simple' Base Locator. (A simple base locator is defined as one that always has byte+0).

```
CAL   BL)NN
ACL   CP)+NN
SLW   SYS)132
```

where CP)+NN is a constant of the form:

```
PZE WORD-DISPLACEMENT,,BYTE
```

Case 3. When the data item is located by means of a 'complex' Base Locator. (A complex base locator is defined as one that has Byte 0 to 5).

```
CAL   BL)NN
ACL   CP)+NN1      Displacement Constant
PDX   0,4
TXL   *2,4,5
ACL   CP)+NN2      OCT 777772000000
SLW   SYS)132
```

SYS)133

```
PZE  LOC,,BYTE
```

This cell points to the first word address (LOC), and the first BYTE(0-5) of the target field involved in a Move. The cell is set by calls upon subroutines SYS)179 or SYS)180 or by means of in-line coding of the same form as described under SYS)132.

SYS)134 This cell is set non zero whenever a floating point underflow results from a Move.

SYS)155 This floating point exponential routine raises the double precision number in the AC-MQ to the single precision power located in SYS)128. The double precision result is left in the AC-MQ.

SYS)156 This floating point exponential routine raises the double precision number in the AC-MQ to the double precision power located SYS)128 - SYS)129. The double precision result is left in the AC-MQ.

SYS)160 TSX SYS)160,4

This is a subroutine for sign adjustment for double precision fixed point numbers. The routine is entered with the number in the AC-MQ and the result is left in the AC-MQ.

SYS)161 This is a conversion table used in converting from 709 to 705 collating sequence.

SYS)162

```
TSX  SYS)162,4
OP   SYS)161
PZE  LOC(1),T(1), LOCATOR(1)
PZE  LENGTH(1),,6*BYTE(1)
PZE  LOC(2),T(2),LOCATOR(2)
PZE  LENGTH(2),,6*BYTE(2)
HIGH RETURN from comparison
EQUAL RETURN from comparison
LOW RETURN from comparison
```

This subroutine performs an alphabetic comparison on two fields. OP is a CVR or NOP depending of the need to adjust the collating sequence before the comparison. If T(J) is 0, LOC(J) is the location of the field. If T(J) is not zero, the field is located b the 'pointer' word LOCATOR (J) and LOC(J) is the word displacement from the base. LENGTH(J) is the length of the field in characters and BYTE(J) is the beginning byte position of the field.

SYS)163

```
TSX  SYS)163,4
PZE  CP)+NN
```

This routine upscales the single precision AC by 10**10 and then upscales by the constant located at CP)+NN.

SYS)164

```
TSX  SYS)164,4
PZE  CP)+NN
```

This routine upscales the number in the MQ by 10**10 and then upscales by the constant located at CP)+NN.

SYS)165

```
TSX  SYS)165,4
PZE  CP)+NN
```

This routine upscales the double precision AC-MQ by the constant located at CP)+NN.

SYS)166

```
TSX  SYS)166,4
PZE  CP)+NN
```

This routine upscales the double precision AC-MQ by the constant located at CP)+NN. On entry to the routine, the high order part of the number is in the MQ and the low order in the AC.

SYS)167
```
TSX   SYS)167,4
PZE   CP)+NN
```

This routine downscales the double precision AC-MQ by the constant located at CP)+NN and leaves the result in the AC-MQ.

SYS)168
```
TSX   SYS)168,4
PZE   CP)+NN
```

This routine downscales the double precision AC-MQ by 10**10 and then downscales by the constant located at CP)+NN leaving the result in the MQ.

SYS)169
```
TSX   SYS)169,4
```

This routine divides the double precision fixed point number in SYS)128 and SYS)129 by the AC-MQ. The result is left in the AC-MQ.

SYS)170
```
TSX   SYS)170,4
PZE   CP)+NN
```

This routine multiplies the double precision AC-MQ by SYS)128 SYS)129, scales the product down by the constant located at CP)+NN and leaves the result in the AC-MQ.

SYS)171
```
TSX   SYS)171,4
PZE   CP)+NN
```

This routine multiplies the double precision AC-MQ by SYS)128 SYS)129, scales the product down by 10**10, and then downscales by the constant located at CP)+NN. The result is left in the AC-MQ.

SYS)172
```
TSX   SYS)172,4
```

This floating point exponential routine raises the single precision number in the AC to the single precision power located in SYS)128. The result is left in the AC.

SYS)173
```
TSX   SYS)173,4
```

This floating point exponential routine raises the single precision number in the AC to the double precision power located in SYS)128 - SYS)129. The double precision result is left in the AC-MQ.

SYS)174
```
TSX   SYS)174,4
PZE   FILENAME
```

This routine opens the file designated FILENAME.

SYS)175
```
TSX  SYS)175,4
PZE  IOC)1
```

This routine opens all files in the file list located by IOC)1.

SYS)176
```
TSX  SYS)176,4
PZE  FILENAME
```

This routine closes the file designated FILENAME.

SYS)177
```
TSX  SYS)177,4
PZE  IOC)1
```

This routine closes all files in the file list located by IOC)1.

SYS)178
```
TSX  SYS)178,4
PZE  CP)+NN1,,CP)+NN2
PZE  CP)+NN3,,CP)+NN4
```

This routine displays a message concerning a STOP verb. The CP (Constant Pool) entries contain the Statement Number of the Stop (in BCD), and the type of STOP (STOP NNN or STOP RUN).

SYS)179
```
TSX  SYS)179,4
SOURCE-ADDRESS-REFERENCE
TARGET-ADDRESS-REFERENCE
Begin specific Move subroutine call
```

This is one entry point to MOVPAK. This entry uses the information in the calling sequence to set up the Move Source Pointer, SYS)132, and the Move Target Pointer, SYS)133. After these data pointers have been set, the instructions beginning at 3,4 are executed to actually perform the moving operation.
The ADDRESS-REFERENCES in the calling sequence are of the following form:

Case 1. For Working Storage data items.

```
PZE LOC,,BYTE
```

where LOC is the first word address and BYTE is the first byte of either the source or target field.

Case 2. For data items located by a Base Locator.

```
MZE BL)NN,,CP)+NN
```

where BL)NNN is the location of the Base Locator and CP)+NN is a constant that is the displacement distance of the data item from the base.

Case 3. For Data items located by a Positional Indicator.

MON PI)NN,,O

where PI)NN is the Positional Indicator which locates the data item.

SYS)180 TSX SYS)180,4
TARGET-ADDRESS-REFERENCE
Begin specific subroutine call

This is an entry to MOVPAK in which either the source address of the data item has been previously stored in SYS)132, or the source item is in a machine register (AC or MQ). In either case, the Target Pointer, SYS)133, is set up using the information in the calling sequence in the same manner as SYS)179, and the specific MOVE instructions beginning at 2,4 are executed.

SYS)181 TSX SYS)181,4
SOURCE-ADDRESS-REFERENCE
Begin specific subroutine call

This is an entry to MOVPAK in which the target address of the data item has been previously stored in SYS)133 or the target address is a machine register (AC or MQ). In either case, the Source Pointer is set up using the information in the calling sequence in the same manner as SYS)179, and the specific Move instructions beginning at 2,4 are executed.

SYS)182 TSX SYS)182,4
Begin specific subroutine call

This is an entry to MOVPAK in which both the Source Pointer, SYS)132 and the Target Pointer SYS)133, have been preset by inline instructions; or a machine register is used for either or both source and target. The specific Move instructions are executed beginning at 1,4.

SYS)183 TXI SYS)183,1,TARGET-SIGN-CONVENTION

This MOVPAK subroutine moves external decimal fields to external decimal fields. The TARGET-SIGN-CONVENTION is as follows:

If 0	no sign
If 1	overpunch minus
If 2	overpunch plus

Immediately following the TXI instruction will be two or more of the following instructions:

```
TXI  SYS)191, 1, NUMBER-OF-CHARACTERS-TO-MOVE
TXI  SYS)199, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
TXI  SYS)205, 1, NUMBER OF-CHARACTERS-TO-BYPASS
TXI  SYS)209, 1, NUMBER OF TRAILING-ZEROS-TO-INSERT
TXI  SYS)210, 1, NUMBER-OF-LEADING-ZEROS-TO-INSERT
TRA  SYS)219     Round current character
TXI  SYS)223,1,TARGET-NUMERIC-LENGTH
TXI  SYS)227, 1, NUMBER-OF-CHARACTERS-TO-MOVE  *Note 1.
TXI  SYS)231, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
                                               *Note 1.
TXI  SYS)235, 1, NUMBER-OF-CHARACTERS-TO-BYPASS
                                               *Note 1.
```

*Note 1. The last character processed under control of this instruction is examined for source field sign.

This type of MOVPAK call is always terminated by the instruction

```
TXI  SYS)223, 1, TARGET-NUMERIC-LENGTH
```

An example of a Move of the type XD to XD might appear in the listing as:

```
LDI  CP)+NN ı                Load Source Pointer
STI  SYS)132
LDI  CP) +NN..               Load Target Pointer
STI  SYS)133
TSX  SYS)182, 4              Begin moving field
TXI  SYS)183, 1, ı           XD to XD
TXI  SYS)191, 1, 10          10 characters to a 10 character field
TXI  SYS)223, 1, 10
```

SYS)184

```
TXI  SYS)184, 1, NUMBER-OF-CHARACTERS-TO-CONVERT
```

This is a MOVPAK subroutine which converts from external decimal to internal decimal leaving results in the AC or AC-MQ. The sign is assumed over the low order digit.

SYS)185

```
TXI  SYS)185, 1, TARGET-EDIT-CONTROL  *Note 1.
OCT              TARGET-CONTROL-WORD-BITS
```

SYS)185 is a MOVPAK subroutine which moves an external decimal field to an edited field.

The first word is followed by a TARGET CONTROL-WORD *Note 2, and two or more of the following instructions:

```
TXI  SYS)193,1,NUMBER-OF CHARACTERS-TO-MOVE
TXI  SYS)201,1,NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
TXI  SYS)206,1,NUMBER-OF-CHARACTERS-TO-BYPASS
TXI  SYS)211,1,NUMBER-OF-TRAILING-ZEROS-TO-INSERT
TXI  SYS)212,1,NUMBER-OF-LEADING-ZEROS-TO-INSERT
TRA  SYS)220   Round current character
TXI  SYS)225,1,TARGET-NUMERIC-LENGTH  (End of call
                 sequence)
TXI  SYS)228,1,NUMBER-OF-CHARACTERS-TO-MOVE  *Note 3.
TXI  SYS)232,1,NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
                                                *Note 3.
TXI  SYS)236,1,NUMBER-OF-CHARACTERS-TO-BYPASS
                                                *Note 3.
```

*Note 1. TARGET-EDIT-CONTROL
Bits are placed in the decrement which indicate the following characteristics of the target field.

Octal digit	Meaning
00001	asterisks
00002	comma(s)
00004	decimal point
00010	dollar sign
00020	Blank When Zero

*Note 2. Additional specific information about the target field is given in the TARGET-CONTROL-WORD which has the form:

Prefix = 0 if no target field commas; otherwise = number of digits to the left of first comma.

Address = number of leading *'s, or 8's in pictorial.

Decrement = number of 8's, 9's, *'s, to the left of the real or implied decimal point in target pictorial.

Tag = TARGET-SIGN-CONVENTION
= 0 if no sign
= 1 if overpunch minus
= 2 if overpunch plus
= 3 if right minus
= 4 if right plus
= 5 if left minus
= 6 if left plus

*Note 3. The last character processed under control this instruction is examined for source field sign.

SYS)186 TXI SYS)186,1,NUMBER-OF CHARACTERS-TO-DEVELOP

This is a MOVPAK subroutine which converts from internal decimal in the AC or AC-MQ to unsigned external decimal.

SYS)187 TXI SYS)187,1,NUMBER-OF-CHARACTERS-TO-DEVELOP

This is a MOVPAK subroutine which converts from internal decimal in the AC or AC-MQ to external decimal with overpunch minus sign convention.

SYS)188 TXI SYS)188,1,NUMBER-OF-CHARACTERS-TO-DEVELOP

This is a MOVPAK subroutine which converts from internal decimal in AC or AC-MQ to external decimal with overpunch plus sign convention.

SYS)189 TXI SYS)189,1,TARGET SIGN CONVENTION *Note 1.

This is a MOVPAK subroutine which moves an edited field to an external decimal field. This word is followed by two or more of the following instructions:

```
TXI  SYS)192,1,NUMBER-OF-CHARACTERS-TO-MOVE
TXI  SYS)195,1,NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
TXI  SYS)197,1,NUMBER-OF-CHARACTERS TO-MOVE
TXI  SYS)200,1,NUMBER-OF-CHARACTERS-TO-BYPASS
                                          *Note 2.
TXI  SYS)203,1,NUMBER OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
TXI  SYS)207,1,NUMBER-OF-CHARACTERS-TO-BYPASS
TXI  SYS)213,1,NUMBER-OF-LEADING-ZEROS-TO-INSERT
TXI  SYS)215,1,NUMBER-OF-TRAILING-ZEROS-TO-INSERT
TXI  SYS)217,1,NUMBER-OF-CHARACTERS-TO-SCAN-FOR-SIGN
TRA  SYS)221   Round current characters
TXI  SYS)224,1,TARGET-NUMERIC-LENGTH  (End of call sequence)
TXI  SYS)229,1,NUMBER-OF-CHARACTERS-TO-MOVE
                                          *Note 3.
TXI  SYS)233,1,NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
                                          *Note 3.
TXI  SYS)237,1,NUMBER-OF-CHARACTERS-TO-BYPASS
                                          *Note 3.
```

*Note 1. TARGET-SIGN-CONVENTION

If 0	No sign
If 1	Overpunch minus
If 2	Overpunch plus

*Note 2. The first character processed under control of this instruction is examined for source field sign.

*Note 3. The last character processed under control of this instruction is examined for source field sign.

SYS)190

```
TXI  SYS)190, 1, TARGET-EDIT-CONTROL      *Note 1.
OCT  TARGET-CONTROL WORD BITS
```

This is a MOVPAK subroutine which moves an edited field to an edited field. The first instruction is followed by a TARGET CONTROL WORD, *NOTE 1, and two or more of the following instructions:

```
TXI  SYS)194, 1, NUMBER-OF-CHARACTERS-TO-MOVE
TXI  SYS)196, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
TXI  SYS)198, 1, NUMBER-OF-CHARACTERS-TO-MOVE
TXI  SYS)202, 1, NUMBER-OF-CHARACTERS-TO-BYPASS
                                          *Note 2.
TXI  SYS)204, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
TXI  SYS)208, 1, NUMBER-OF-CHARACTERS-TO-BYPASS
TXI  SYS)214, 1, NUMBER-OF-LEADING-ZEROS-TO-INSERT
TXI  SYS)216, 1, NUMBER-OF-TRAILING-ZEROS-TO-INSERT
TXI  SYS)218, 1, NUMBER-OF-CHARACTERS-TO-SCAN-FOR-SIGN
TRA  SYS)222     Round current character
TXI  SYS)226, 1, TARGET-NUMERIC-LENGTH (End of call sequence)
TXI  SYS)230, 1, NUMBER-OF-CHARACTERS-TO-MOVE
                                          *Note 3.
TXI  SYS)234, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
                                          *Note 3.
TXI  SYS)238, 1, NUMBERS-OF-CHARACTERS-TO-BYPASS
                                          *Note 3.
```

*Note 1. The TARGET-EDIT-CONTROL and the TARGET-CONTROL-WORD are the same format as described under SYS)185.

*Note 2. The first character processed under control of this instruction is examined for source field sign.

*Note 3. The last character processed under control of this instruction is examined for source field sign.

SYS)191

```
TXI  SYS)191, 1, NUMBER-OF-CHARACTERS-TO-MOVE
```

This is a MOVPAK subroutine used in conjunction with SYS)183 in moving external decimal fields to external decimal fields.

SYS)192 TXI SYS)192, 1, NUMBER-OF-CHARACTERS-TO-MOVE

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)193 TXI SYS)193, 1, NUMBER-OF-CHARACTERS-TO-MOVE

This MOVPAK subroutine is used in conjunction with SYS)185 to move external decimal fields to edited fields.

SYS)194 TXI SYS)194, 1, NUMBERS-OF-CHARACTERS-TO-MOVE
This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)195 TXI SYS)195, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)196 TXI SYS)196, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW

This MOVPAK subroutine is used in conjuction with SYS)190 to move edited fields to edited fields.

SYS)197 TXI SYS)197, 1, NUMBER-OF-CHARACTERS-TO-MOVE

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)198 TXI SYS)198, 1, NUMBER-OF-CHARACTERS-TO-MOVE

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)199 TXI SYS)199, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW

This MOVPAK subroutine is used in conjunction with SYS)183 to move external decimal fields to external decimal fields.

SYS)200 TXI SYS)200, 1, NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)201 TXI SYS)201,1,NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW

This MOVPAK subroutine is used in conjunction with SYS)185 to move external decimal fields to edited fields.

SYS)202 TXI SYS)202,1,NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)203 TXI SYS)203,1,NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)204 TXI SYS)204,1,NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)205 TXI SYS)205,1,NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)183 to move external decimal fields to external decimal fields.

SYS)206 TXI SYS)206 ,1,NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)185 to move external decimal fields to edited fields.

SYS)207 TXI SYS)207,1,NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)208 TXI SYS)208,1,NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)209 TXI SYS)209,1,NUMBER-OF-TRAILING-ZEROS-TO-INSERT

This MOVPAK subroutine is used in conjunction with SYS)183 to move external decimal fields to external decimal fields.

SYS)210 TXI SYS)210,1,NUMBER-OF-LEADING-ZEROS-TO-INSERT

This MOVPAK subroutine is used in conjunction with SYS)183 to move external decimal fields to external decimal fields.

SYS)211 TXI SYS)211,1,NUMBER OF-TRAILING-ZEROS-TO-INSERT

This MOVPAK subroutine is used in conjunction with SYS)185 to move external decimal fields to edited fields.

SYS)212 TXI SYS)212,1,NUMBER-OF-LEADING-ZEROS-TO-INSERT

This MOVPAK subroutine is used in conjunction with SYS)185 to move external decimal fields to edited fields.

SYS)213 TXI SYS)213,1,NUMBER-OF-LEADING-ZEROS-TO-INSERT

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)214 TXI SYS)214,1,NUMBER-OF-LEADING-ZEROS-TO-INSERT

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)215 TXI SYS)215,1,NUMBER-OF-TRAILING-ZEROS-TO-INSERT

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal.

SYS)216 TXI SYS)216,1,NUMBER-OF-TRAILING-ZEROS-TO-INSERT.

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)217 TXI SYS)217,1,NUMBER-OF-CHARACTERS-TO-SCAN-FOR-SIGN

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)218 TXI SYS)218,1,NUMBER-OF-CHARACTERS-TO-SCAN-FOR-SIGN

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)219 TRA SYS)219 Round current character

This MOVPAK subroutine is used in conjunction with SYS)183 to move external decimal fields to external decimal fields.

SYS)220 TRA SYS)220 Round current character

This MOVPAK subroutine is used in conjunction with SYS)185 to move external decimal fields to edited fields.

SYS)221 TRA SYS)221 Round current character

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)222 TRA SYS)222 Round current character

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)223 TXI SYS)223, 1, TARGET-NUMERIC-LENGTH

This MOVPAK subroutine is used in conjunction with SYS)183 to move external decimal fields to external decimal fields.

SYS)224 TXI SYS)224, 1, TARGET-NUMERIC-LENGTH

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)225 TXI SYS)225, 1, TARGET-NUMERIC-LENGTH

This MOVPAK subroutine is used in conjunction with SYS)185 to move external decimal fields to edited fields.

SYS)226 TXI SYS)226, 1, TARGET-NUMERIC-LENGTH

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)227 TXI SYS)227, 1, NUMBER-OF-CHARACTERS-TO-MOVE

This MOVPAK subroutine is used in conjunction with SYS)183 to move external decimal fields to external decimal fields.

SYS)228 TXI SYS)228, 1, NUMBER-OF-CHARACTERS-TO-MOVE

This MOVPAK subroutine is used in conjunction with SYS)185 to move external decimal fields to edited fields.

SYS)229 TXI SYS)229, 1, NUMBER-OF-CHARACTERS-TO-MOVE

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)230 TXI SYS)230, 1, NUMBER-OF-CHARACTERS-TO-MOVE

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)231 TXI SYS)231, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERPUNCH

This MOVPAK subroutine is used in conjunction with SYS)183 to move external decimal fields to external decimal fields.

SYS)232 TXI SYS)232, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERPUNCH

This MOVPAK subroutine is used in conjunction with SYS)185 to move external decimal fields to edited fields.

SYS)233 TXI SYS)233, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERPUNCH

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)234 TXI SYS)234, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERPUNCH

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)235 TXI SYS)235, 1, NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)183 to move external decimal fields to external decimal fields.

SYS)236 TXI SYS)236, 1, NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)185 to move external decimal fields to edited fields.

SYS)237 TXI SYS)237,1,NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)189 to move edited fields to external decimal fields.

SYS)238 TXI SYS)238,1,NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)190 to move edited fields to edited fields.

SYS)239 TXI SYS)239,1,NUMBER-OF-CHARACTERS-TO-MOVE

This MOVPAK subroutine moves alphabetic fields to alphabetic fields.

SYS)240 TXI SYS)240,1,NUMBER-OF-CHARACTERS-TO-MOVE

This MOVPAK subroutine moves alphabetic fields to alphabetic fields with additional blank insertion.

SYS)241 TXI SYS)241,1,NUMBER-OF-BLANKS-TO-INSERT

This MOVPAK subroutine is used in conjunction with SYS)240 to move alphabetic fields to alphabetic fields.

SYS)242 TXI SYS)242,1,ALPHABETIC-CONTROL *Note 1.
PZE CONTROL1,,CONTROL2

This MOVPAK subroutine moves alphabetic fields to alphabetic fields where some information was unknown at compile time.

*Note 1. ALPHABETIC CONTROL
Bits are placed in the decrement which indicate the following about CONTROL1 and CONTROL2.

OCTAL DIGIT	MEANING
00001	Target fields's length is in words (not characters)
00002	Source field's length is in words (not characters)
00004	CONTROL2 is the location of the target field's length (not length itself)
00010	CONTROL1 is the location of the source field's length (not length itself)

SYS)243 TXI SYS)243,1,NUMBER-OF-BLANKS-TO-INSERT

This MOVPAK subroutine moves blanks to an alphabetic field.

SYS)244 TXI SYS)244, 1, NUMBER-OF-ZEROS-TO-INSERT

This MOVPAK subroutine moves zeros to an alphabetic field.

SYS)245 TXI SYS)245, 1, NUMBER-OF-CHARACTERS-TO-INSERT
OCT CHARACTERS

This MOVPAK subroutine moves characters to an alphabetic field. The second word contains 6 characters of the type to be moved.

SYS)246 TXI SYS)246, 1, NUMBER-OF-CHARACTERS-IN-TARGET-AREA

This MOVPAK subroutine moves the internal decimal right justified field in the AC or AC-MQ to an internal decimal field not justified.

SYS)247 TXI SYS)247, 1, NUMBER-OF-CHARACTERS-IN-SOURCE-AREA

This MOVPAK subroutine moves from an internal decimal field not justified to internal decimal right justified in the AC or AC-MQ.

SYS)248 TRA SYS)248
TARGET-CONTROL-WORD-TYPE-SD *Note 1.

This MOVPAK subroutine converts the floating point AC to scientific decimal.

*Note 1. The CONTROL-WORD-TYPE-SD has the following form:

Prefix = MZE if decimal in pictorial.
= PZE if no decimal in pictorial.

Address = Scale applied to mantissa in pictorial.

Tag is Sign Convention.
= 0 is mantissa and exponent sign conventions are both minus.
= 1 if mantissa sign convention is minus and exponent sign convention is plus.
= 2 if mantissa sign convention is plus and exponent sign convention is minus.
= 3 if both mantissa and exponent sign conventions are plus.

Decrement = Total length in characters, of the field.

SYS)249 TRA SYS)249
SOURCE-CONTROL-WORD-TYPE-SD

This MOVPAK subroutine converts from scientific decimal to floating point leaving the results in the AC. The SOURCE-CONTROL-WORD-TYPE-SD has the same form as described under SYS)248.

SYS)250 TSX SYS)250,4
TARGET-CONTROL-WORD-TYPE-HD *Note 1.

This MOVPAK subroutine converts the floating point value in the AC to internal decimal and leaves the results in the AC or AC-MQ.

*Note 1 The form of the CONTROL-WORD-TYPE-HD is as follows:

Prefix is the sign of the scale.
= PZE for plus.
= MZE for minus.

Address = Scale applied to internal decimal value.

Decrement = Numeric length of value.

SYS)251 TSX SYS)251,4
SOURCE-CONTROL-WORD-TYPE-HD

This MOVPAK subroutine converts the internal decimal value in the AC or AC-MQ to floating point leaving the results in the AC. The SOURCE-CONTROL-WORD-TYPE-HD has the same form as described under SYS)250.

SYS)252 TRA SYS)252
SOURCE-CONTROL-WORD-TYPE-SD
TARGET-CONTROL-WORD-TYPE-SD

This MOVPAK subroutine moves a scientific decimal field to a scientific decimal field. The CONTROL-WORD-TYPE-SD for both source and target have the same form as described under SYS)248.

SYS)253 TRA SYS)253
SOURCE-CONTROL-WORD-TYPE-SD
TARGET-CONTROL-WORD-TYPE-HD

This MOVPAK subroutine converts a scientific decimal field to internal decimal leaving the results in the AC or AC-MQ. CONTROL-WORD-TYPE-SD has the same format as described under SYS)248 and CONTROL-WORD-TYPE-HD has the same format as described under SYS)250.

SYS)254

```
TRA  SYS)254
SOURCE-CONTROL-WORD-TYPE-ID
TARGET CONTROL-WORD TYPE-SD
```

This MOVPAK subroutine converts from internal decimal in the AC or AC-MQ to scientific decimal. CONTROL-WORD-TYPE-ID has the same format as described under SYS)250 and CONTROL-WORD-TYPE-SD has the same format as described under SYS)248.

SYS)255

```
TRA  SYS)255
TARGET-CONTROL-WORD-TYPE-SD
```

This MOVPAK subroutine converts from double precision floating point in the AC-MQ to scientific decimal. The CONTROL-WORD-TYPE-SD has the same form as described under SYS)248.

SYS)256

```
TRA  SYS)256
SOURCE-CONTROL-WORD-TYPE-SD
```

This MOVPAK subroutine converts from scientific decimal to double precision floating point leaving the results in the AC-MQ. CONTROL-WORD-TYPE-SD has the same form as described under SYS)248.

SYS)257

```
TSX  SYS)257,4
TARGET-CONTROL-WORD-TYPE-ID
```

This MOVPAK subroutine converts from double precision floating point in the AC-MQ to internal decimal leaving the results in the AC or AC-MQ. TARGET-CONTROL-WORD-TYPE-ID has the same form as described under SYS)250.

SYS)258

```
TSX  SYS)258,4
SOURCE-CONTROL-WORD-TYPE-ID
```

This MOVPAK subroutine converts from internal decimal in the AC or AC-MQ to double precision floating point leaving the results in the AC-MQ. SOURCE-CONTROL-WORD-TYPE-ID has the same form as described under SYS)250.

SYS)260 This SYS number may appear in a GET calling sequence to the IOCS Read routine:

```
TSX     IOC)8,4
PZE     FILENAME,,SYS)260
PZE     SYS)265,,SYS)283
IOCDN*  BL)NN,,14
```

The SYS)260 subroutine prints an error message indicating processing terminated due to record length error.

SYS)261

```
TSX  SYS)261,4
TSX  SYS)263,6
(Normal Return)
```

Subroutine SYS)261 converts the logical accumulator from a BCD number to binary, checking for non-numeric characters and/or imbedded or trailing blanks; and leaves the result in the decrement of the AC. This routine is used in conjunction with getting a variable length BCD record from tape where the first word of the record gives the remaining length of the record. If an error is detected during conversion, return is 1,4 to routine SYS)263 which prints an error message.

SYS)262

```
TSX  SYS)262,4
```

This subroutine converts the binary AC address to a 6 character (with leading blanks) BCD word to be used in 'Filing' a variable length BCD record. The BCD word is left in the logical AC.

SYS)263

```
TSX  SYS)263,4
```

This subroutine prints an error message in conjunction with SYS)261 upon GET error condition (see SYS)261) and exits to the CT monitor.

SYS)264

```
TXL  *+5,1,BLOCKSIZE-1
TSX  SYS)264,4
PZE  FILENAME
OCT  STATEMENT-NUMBER
OCT  SUB-STATEMENT-NUMBER
```

When the record size (in IR1) exceeds the Blocksize for the file, SYS)264 prints a message and exits to the CT Monitor.

SYS)265 This SYS number appears as part of the GET calling sequence to the IOCS Read routine whenever the 'AT END' option is not used with the GET verb.

```
TSX      IOC)8,4
PZE      FILENAME,,SYS)260
PZE      SYS)265,,SYS)283
IOCDN*   BL)NN,,14
```

SYS)265 prints a message concerning the unexpected end-of-file and exits to the CT Monitor.

SYS)266

```
TRA  SYS)266
   or
TXI  SYS)266,0,0
```

This routine performs a 'panic' Close-All-Files and exits to the CT Monitor.

SYS)267

```
TXI  SYS)267,1,TARGET-EDIT-CONTROL
OCT  TARGET-CONTROL-WORD-BITS
AXT  NUMBER-OF-DIGITS-TO-CONVERT,1
```

This MOVPAK subroutine converts from internal decimal in the AC-MQ to form an edited field. The TARGET-EDIT-CONTROL and TARGET-CONTROL-WORD-BITS have the same form as described under SYS)185.

SYS)268

```
TXI  SYS)268,1,1
```

This MOVPAK subroutine converts an edited field to internal decimal leaving the results in the AC-MQ. This instruction is followed by two or more of the following instructions:

```
TXI  SYS)269,1,NUMBER-OF-CHARACTERS-TO-CONVERT
TXI  SYS)270,1,NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
TXI  SYS)271.1,NUMBER-OF-CHARACTERS-TO-BYPASS
TXI  SYS)272,1,NUMBER-OF-DECIMAL-ZEROS-TO-DEVELOP
TXI  SYS)273,1,NUMBER-OF-CHARACTERS-TO-SCAN-FOR-SIGN
TRA  SYS)274   Round Current Character
TXI  SYS)275,1,TARGET-DECIMAL-NUMERIC-LENGTH (end of call
                                                sequence)
TXI  SYS)276,1,NUMBER-OF-CHARACTERS-TO-CONVERT    *Note 2
TXI  SYS)277,1,NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
                                                  *Note 2
TXI  SYS)278,1,NUMBER-OF-CHARACTERS-TO-BYPASS     *Note 2
TXI  SYS)279,1,NUMBER-OF-LEADING-DECIMAL-ZEROS-TO-DEVELOP
TXI  SYS)280,1,NUMBER-OF-CHARACTERS-TO-CONVERT
TXI  SYS)281,1,NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
TXI  SYS)282,1,NUMBER-OF-CHARACTERS-TO-BYPASS     *Note 1
```

*Note 1 The first character processed under control of this instruction is examined for source field sign.

*Note 2 The last character processed under control of this instruction is examined for source field sign.

SYS)269

```
TXI  SYS)269,1,NUMBER-OF-CHARACTERS-TO-CONVERT
```

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)270

```
TXI  SYS)270,1,NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW
```

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)271 TXI SYS)271, 1, NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)272 TXI SYS)272, 1, NUMBER-OF-DECIMAL-ZEROS-TO-DEVELOP

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)273 TXI SYS)273, 1, NUMBER-OF-CHARACTERS-TO-SCAN-FOR-SIGN

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)274 TRA SYS)274 Round Current Character

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)275 TXI SYS)275, 1, TARGET-DECIMAL-NUMERIC-LENGTH

This MOVPAK suvroutine is used in conj unction with SYS)268 to convert an edited field to internal decimal.

SYS)276 TXI SYS)276, 1, NUMBER-OF-CHARACTERS-TO-CONVERT

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)277 TXI SYS)277, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)278 TXI SYS)278, 1, NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)279 TXI SYS)279, 1, NUMBER-OF-LEADING-DECIMAL-ZEROS-TO-DEVELOP

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)280 TXI SYS)280, 1, NUMBER-OF-CHARACTERS-TO-CONVERT

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)281 TXI SYS)281, 1, NUMBER-OF-CHARACTERS-TO-TEST-FOR-OVERFLOW

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)282 TXI SYS)282, 1, NUMBER-OF-CHARACTERS-TO-BYPASS

This MOVPAK subroutine is used in conjunction with SYS)268 to convert an edited field to internal decimal.

SYS)283 This SYS number appears as part of a GET calling sequence to the IOCS Read routine whenever the 'ON ERROR' option in not used in the Environment description of the file.

```
TSX     IOC)8,4
PZE     FILENAME,,SYS)260
PZE     SYS)265,,SYS)283
IOCDN*  BL)NN,,14
```

SYS)283 prints a message concerning the GET error and exits to the CT Monitor.

SYS)286 TSX SYS)286, 4

This subroutine performs a hollerith to BCD conversion on the record located by locator word SYS)287 and places the BCD record in SYS)288.

SYS)287 This SYS number appears as part of the calling sequence to GET a record from the on-line card reader and convert the record to BCD.

```
TSX     IOC)8,4
PZE     FILENAME,,SYS)260
PZE     SYS)265,,SYS)283
IOCDN*  SYS)287,,24
```

SYS)287 is a pointer word which serves to locate the record for the hollerith to BCD conversion routine, SYS)286.

SYS)288 This is a 12 word area used in conjunction with SYS)286 in converting a hollerith image to BCD.

SYS)291 This SYS number may appear as a part of the 'Filing' sequence to the IOCS Write routine.

```
TSX       IOC)9,4
PZE       FILENAME,,SYS)291
IOCDN*    *+1,,24
```

SYS)291 appears only in Write sequences to File a record on card equipment (Locating Mode). The routine, when entered, prints an error message and exits to the CT Monitor.

SYS)292

```
TSX  SYS)292,4
PZE  RECORD LOC,,RECORD-LENGTH
(Partial Conversion Exit)
(Total Conversion Exit)
```

This subroutine converts a record located at RECORD-LOC from BCD to hollerith for 'Filing' on card equipment.

SYS)294

```
LAC  BL)NN,N
TXL  SYS)294,N,0
```

This subroutine prints an error message whenever a reference is made to a Base Locator before the locator has been loaded, and exits back to the CT Monitor.

SYS)296 This SYS number appears as part of the calling sequence in 'Filing' a record on card equipment.

```
TSX       IOC)9,4
PZE       FILENAME
IOCDN*    *+1,,24
AXT       0,4
SXA       SYS)296,4
```

SYS)296 is a pointer word within subroutine SYS)292 which locates the buffer to be used in the BCD to hollerith conversion.

APPENDIX 90.03

RELATIVE BINARY PROGRAM DECK

INTRODUCTION

This section defines the deck order and format of the debugging dictionary, control break table, file check table, and text. Additional information pertaining to these items will be found in the section on the Loader (03.00)

A. General Description of Binary Load Decks

1. Binary Card Format

The 22 word columnar binary card form is used.

Word 1	S, 1	11 (relative deck indicator)
	2	checksum control bit 0 = verify checksum 1 = do not verify checksum
	3	1 (Commercial Translator)
	4	0
	5-7	deck type
	8-12	01010
	13-17	word count (beginning with word 3)
	18-20	000
	21-35	card sequence number

Word 2 logical sum of word 1 and all data words on the card.

Words 3-24 data

2. Binary Decks

Each section of the binary deck (e.g., Control Break Table, Text) is sequenced independently, beginning with sequence number 0. Any section except the text can be missing under certain conditions. Within any section, the cards must be in proper sequence, and the sections themselves, if present, must be in order by deck type.

The deck type codes, which appear in word 1, bits 5-7, are as follows:

001	debugging dictionary
010	control break table
011	file check table
100	text

The debugging dictionary is not implemented in the initial release.

The binary portion of the program deck is limited by the *CTEXT and *CTEND cards.

B. FILE CHECK Table

1. Introduction

The File Check Table, if present, is used to validate the contents of

the generated File Block and the assignment of each file to a Buffer Pool and to transmit the location of any non-standard label routines to the IOCS System. File type, mode, allowable I/O units and blocking requirements are among those items which may be assumed as unchanging by the generated object program and hence are recorded in File Check Table to insure that the user does not change these via modification of *FILE cards and *SPEC cards.

2. Card Format

Word 1 - standard 9L format text type 011.
Word 2 - checksum
3-4 - file check entry i
5-6 - file check entry i + 1
. etc.
. etc.
. etc.

3. File Check Entry Specifications

Two words of text are required for each file referenced in a program. Their format is:

	Bit	Content
1st word of file check entry	S	If = 1, mixed mode file
	1-5	0
	6	If = 1, binary mode = 0, BCD mode
	7-8	If = 00, Input = 01, PTO = 10, TTO = 11, Checkpoint
	9-17	0
	18-20	If = 001 Card equipment only 010 Tape drive only 011 Either tape or card allowable
	21-35	File reference number
2nd word of file check entry	S	If = 1, this is last entry of the File Check Table
	1-2	0
	3-17	0 or relative loc non-standard label routine
	18-20	If = 000 N = blocksize If = 001 N ≤ blocksize If = 010 N = blocksize a multiple of N
	21-35	N

C. Control Break Table

1. Introduction

One entry is generated by each CONTRL entry in the Environment section. An entry occupies a pair of words, the first pair on a card beginning with word 3. Up to eleven entries can be placed on a card.

2. Format

```
Word 1   BCI   1,Name
Word 2   P     L,,N
```

where P = PZE real control break
= PTW virtual control break

L = relative location of the beginning of the control break
N = length of the control break in words (may be 0).

All cards except the last must be full. The sign of the second word of the last entry is negative (P = MZE or MTW).

Control break table entries are ordered first by increasing relative location, and within this ordering by increasing length.

D. Text

1. Format

Words 3, 4 and 5 of a text card are used for up to 19 5-bit control groups, one for each following data word on the card. The sign bits of these control words are not used, and the control groups are given in sequence, up to 7 groups per word.

The information in the Text deck corresponds exactly to that shown on the assembly listing under the columns "OCTAL" and "CNTRL".

The first bit of a control group distinguishes between two basic word types:

1 standard data word
0 special entry

2. Special Entries

00000 end of card (no data word is associated with this entry)

01111 end of text; the address of the corresponding data word contains the relative program entry point.

00010 system reference point; the address of the corresponding data word is the system reference number. Such an entry is generated by the appearance of a SYS pseudo-op in CRYPT, and is used to define reference points in object time subroutines.

00001 location counter control entry; the data word is of the form

	OP	A
OP =	PZE	A is an absolute origin
=	PTW	fixed length BSS of length A.
=	PTH	variable length BSS; the variable field follows as a complex expression. A is the length assumed for assembly.
=	MON	A is a relative origin.

01110 the corresponding data word is an immediate operator. Such an entry occurs only in 2TEXT.

Codes 00011 through 01101 are not assigned.

3. Standard Word

The control group is of the form 1 AB CD, where

AB = 00 constant decrement
= 01 relative decrement
= 10 the decrement is a system reference
= 11 the decrement is represented by a complex expression

CD has the corresponding code values describing the address. The prefix and tag of a standard word are constant.

4. System References:

Each of these references is a 15-bit code. The high-order bits are used to distinguish between types of reference, while the low-order part of the field gives the reference number.

The reference type codes are:

0000 system reference (11-bit system reference number follows).
0001 file reference (11-bit file index number follows).
0010, 0011, 10 these codes are not assigned.
11 external name table reference. This occurs only in 2TEXT, and is followed by 13 bits giving the relative location of an entry in the External Name Table during loading.

5. The decrement or address, or both, of a data word can be represented by a complex expression. This expression consists of a string of one or more words, each with a corresponding control group, following the data word. If both the decrement and address are complex, the decrement occurs first.

Each word of a complex expression has the form

OP A,B,C

where

OP = PZE +
= PON -
= PTW *
= PTH /

The control group has the same form as that for a standard word, with the 11 code changed to mean a result storage cell. Six such result storages (0,1,...,6) are used for intermediate results during the evaluation of a complex expression.

A word in a complex expression is to be interpreted as

A OP C $\longrightarrow$ B

where B is the result storage cell into which the result is to be placed. The complex expression is terminated by a word for which B = 7.

6. Immediate Operator

In 2TEXT, during loading, an immediate operator word can precede a standard word or a word in a complex expression. The immediate operator has the form

P A,T,D

A and D are constants. P and T have the form ± OP, where OP represents any of the operator codes for a complex expression. The sign determines the order of operands. The meaning of this operator can best be described by an example:

S_1 OP_1 D S_2 OP_2 A Operator

P' D' T' A' Data Word

will cause a word to be formed (assuming S_1 and S_2 are +):

P' D OP_1 D' T' A OP A'

If S_1 or S_2 or both are minus, the corresponding operands are reversed.

APPENDIX 90.04

ERROR MESSAGES AND SEVERITY CODES

A. Error Messages

In the following list of error messages the code number is '0' because the value may vary. One of the severity values 1 through 5 will actually be printed with the error message.

```
   DATE 01/31/62  TIME

THE FOLLOWING ERRORS WERE DETECTED DURING COMPILATION-

 NUMBER  CODE  MESSAGE

   0,00    0   ERROR MESSAGE NOT YET IN FILE.
   1,00    0   -FILE- CARD LACKS NAME IN COLUMNS 7 THROUGH 22.
   2,00    0   -RUN- DELETED. ITS USE IS RESTRICTED TO PROCESSOR.
   3,00    0   -OPTION- CARD FORMAT ERROR.
   4,00    0   -COND- CARD FORMAT ERROR.
   5,00    0   RECORD LENGTH 24 OF 'NAME.2' EXCEEDS 'NAME.1' -BLOCKSIZE- 12.
                  -FILE- CARD MUST HAVE -SPANS-.
   6,00    0   -COND- CARD KEY SETTING EXCEEDS 12 DIGITS. RIGHTMOST 12 DIGITS USED.
   7,00    0   -COND- CARD KEY SETTING MUST BE OCTAL. KEY SETTING '1' USED.
   8,00    0   'NAME.1' IS NEITHER A RECORD NOR A FILE. CHECK DATA DESCRIPTION OR NAME.
   9,00    0   RECORD 'NAME.2' MUST BE ON A -FILE- CARD.
                  CHECK ENVIRONMENT DESCRIPTION.
  10,00    0   RECORD 'NAME.2' MUST BE ON AN INPUT -FILE- CARD.
                  CHECK ENVIRONMENT DESCRIPTION.
  11,00    0   RECORD 'NAME.2' CANNOT BE ON MORE THAN ONE INPUT -FILE- CARD.
                  CHECK ENVIRONMENT DESCRIPTION.
  12,00    0   INCORRECT USE OF -GET RECORD FROM-. CANNOT DETERMINE RECORD LENGTH.
                  CHECK ENVIRONMENT DESCRIPTION.
  13,00    0   'NAME.1' FILE LACKS RECORD NAME IN -FILE- CARD
                  OR NO CORRESPONDING RECORD NAME IS IN DATA DESCRIPTION.
  14,00    0   INCORRECT USE OF -GET RECORD FROM-. 'NAME.1' IS NOT AN INPUT FILE.
                  CHECK ENVIRONMENT DESCRIPTION.
  15,00    0   'NAME.1' FILE LACKS RECORD NAME IN -FILE- CARD
                  OR NO CORRESPONDING RECORD NAME IS IN DATA DESCRIPTION.
  16,00    0   'NAME.2' IS NOT A RECORD. CHECK DATA DESCRIPTION.
  17,00    0   'NAME.2' MUST BE RECORD NAME IN -FILE- CARD.
  18,00    0   ILLEGAL INTERNAL CONDITION. NOTHING DONE. POSSIBLE COMPILER ERROR.
  19,00    0   RECORD 'NAME.2' MUST BE ON AN OUTPUT -FILE- CARD.
                  CHECK ENVIRONMENT DESCRIPTION OR NAME.
  20,00    0   BINARY DATA CANNOT BE OUTPUT ON BCD TAPE.
  21,00    0   'NAME.1' IS NOT A FILE. CHECK ENVIRONMENT DESCRIPTION.
  22,00    0   'NAME.1' IS NOT AN OUTPUT FILE. CHECK ENVIRONMENT DESCRIPTION.
  23,00    0   INCORRECT USE OF -GET RECORD FROM-. CHECK DATA DESCRIPTION OR ENVIRONMENT DESCRIPTION.
  24,00    0   ILLEGAL INTERNAL CONDITION. NOTHING DONE. POSSIBLE COMPILER ERROR.
  25,00    0   OPERATION IGNORED BECAUSE 'NAME.1' HAS IMPROPER DATA FORMAT
                  ( E  A(2) ) FOR THIS USE.
  26,00    0   OPERATION IGNORED BECAUSE 'NAME.1' IS ILLEGAL TRANSFER ADDRESS.
  27,00    0   DOWNSCALE GENERATED WHICH LOSES ALL SIGNIFICANT FIGURES.
  28,00    0   ATTEMPTED DIVISION BY ZERO BYPASSED. RESULT TAKEN TO BE ZERO.
  29,00    0   ILLEGAL INTERNAL CONDITION. NOTHING DONE. POSSIBLE COMPILER ERROR.
  30,00    0   FUNCTION 'NAME.1' LACKS EXPLICIT SPECIFICATION OF ALL ARGUMENTS.
  31,00    0   SUBSCRIPT VARIABLE 'NAME.1' MUST BE AN INTEGER.
  32,00    0   MODE AND DATA DESCRIPTION CONFLICT. 'NAME.1' FORMAT USED.
  33,00    0   ILLEGAL COMBINATION OF FORMAT CHARACTERS. CHECK DATA DESCRIPTION.
                  'NAME.1' FORMAT USED.
  34,00    0   MAXIMUM FORMAT CHARACTER COUNT EXCEEDED. 'NAME.1' FORMAT USED.
  35,00    0   MAXIMUM NUMERIC LENGTH EXCEEDED. 'NAME.1' FORMAT USED.
  36,00    0   DATA ON FORMAT DESCRIPTION LEVEL CANNOT HAVE SUB-ORGANIZATION.
                  CHECK 'NAME.1'.
  37,00    0   FORMAT ERROR FOR CONDITIONAL VARIABLE. 'NAME.1' FORMAT USED.
```

```
38,00   0   CONDITIONAL VARIABLE CANNOT HAVE -QUANTITY-.
39,00   0   DATA NOT ON FORMAT DESCRIPTION LEVEL CANNOT BE SPECIFIED AS RIGHT JUSTIFIED.
40,00   0   -REDEF- TO 'NAME.1' CANNOT OCCUR BEFORE DEFINITION.
41,00   0   NAME ASSOCIATED WITH -REDEF- OR -QUANTITY IN- IS UNDEFINED.
42,00   0   DATA ITEM WITHOUT LENGTH. CHECK 'NAME.1'.
43,00   0   CONSTANT CANNOT BE ASSOCIATED WITH -REDEF- OR INPUT RECORD, OR PRECEDED BY VARIABLE LENGTH FIELD.
44,00   0   UNSPECIFIED MAXIMUM QUANTITY ASSUMED TO BE 1 FOR STORAGE ALLOCATION.
45,00   0   'NAME.1' IS -COND- TYPE AND CANNOT BE
              ASSOCIATED WITH -REDEF- OR -QUANTITY IN-.
46,00   0   'NAME.1' IS NOT DATA NAME AND CANNOT BE
              ASSOCIATED WITH -REDEF- OR -QUANTITY IN-.
47,00   0   DATA NOT ON FORMAT DESCRIPTION LEVEL CANNOT BE ASSOCIATED WITH -QUANTITY IN-.
48,00   0   NO RECORDS SPECIFIED IN -PATTERN- ON -FILE- CARD FOR 'NAME.1'.
49,00   0   SINGLE RECORD IN THE -PATTERN- ON -FILE- CARD
              FOR 'NAME.1'. INEFFICIENT PROGRAM PRODUCED.
50,00   0   NUMBER OF RECORDS IN -PATTERN- CANNOT EXCEED 16.
51,00   0   CONFLICT BETWEEN LENGTH OF CONSTANT AND PICTORIAL OF 'NAME.1'.
52,00   0   MAXIMUM NUMERIC LENGTH EXCEEDED FOR CONSTANT OR LITERAL.
53,00   0   INCORRECT USAGE OF PERIOD, SIGN, OR F FOR CONSTANT OR LITERAL.
54,00   0   ILLEGAL CHARACTER FOR CONSTANT OR LITERAL.
55,00   0   FLOATING POINT OVERFLOW IN CONVERTING CONSTANT OR LITERAL.
56,00   0   FLOATING POINT UNDERFLOW IN CONVERTING CONSTANT OR LITERAL.
57,00   0   CONSTANT CANNOT BE GIVEN FOR EDITED TYPE FIELD.
58,00   0   CONSTANT OF EXTERNAL DECIMAL TYPE IN ERROR. CHECK 'NAME.1'.
59,00   0   CONFLICT BETWEEN LENGTH OF ALPHABETIC CONSTANT AND PICTORIAL.
60,00   0   ZERO COUNT IN PICTORIAL REPLACED BY ONE.
61,00   0   OPERATION DEFINED AS NAME OR FOUND IN NAME FIELD.
62,00   0   PREVIOUS CARD NOT PROPERLY TERMINATED. PERIOD ASSUMED.
63,00   0   NEITHER -ADD- NOR -MOVE- PRECEDES -CORRESPONDING-
64,00   0   CANNOT -END- SECTION WHEN NONE ARE OPEN.
65,00   0   CANNOT -END- SECTION 'NAME.1'
              BEFORE SECTION 'NAME.2'.
66,00   0   ONE OR MORE SECTIONS NOT CLOSED.
67,00   0   THERE IS AN ILLEGAL NON-NUMERIC CHARACTER IN THE NUMERIC FIELD.
68,00   0   EVALUATION IGNORED FOR FUNCTION 'NAME.1'.
              TOO MANY ARGUMENTS SPECIFIED.
69,00   0   ILLEGAL INTERNAL CODE 'NAME.1' SENT TO ASSEMBLY.  POSSIBLE COMPILER ERROR.
70,00   0   CHECK ARRAY OF ELEMENTS, 'NAME.1', FOR NUMBER OF DIMENSIONS.
71,00   0   INVALID FORMAT FOR SUBSCRIPT VARIABLE IN ARRAY OF ELEMENTS, 'NAME.1'.
72,00   0   TOO MANY -USING- PARAMETERS IN -DO- STATEMENT.
73,00   0   TOO FEW -USING- PARAMETERS IN -DO- STATEMENT.
74,00   0   TOO MANY -GIVING-  PARAMETERS IN -DO- STATEMENT
75,00   0   TOO FEW -GIVING-  PARAMETERS IN -DO- STATEMENT
76,00   0   FORMAT ERROR FOR LOOP CONTROL VARIABLE 'NAME.1'.
77,00   0   FORMAT ERROR FOR PARAMETER 'NAME.1' OF LOOP CONTROL VARIABLE.
78,00   0   FORMAT ERROR FOR LITERAL PARAMETER OF LOOP CONTROL VARIABLE 'NAME.1'.
79,00   0   SUBSCRIPT VARIABLE 'NAME.1' MUST BE OF NUMERIC TYPE.
80,00   0   CONFLICT BETWEEN JUSTIFICATION AS GIVEN BY ORIGINAL DEFINITION AND  -REDEF-.
81,00   0   CONFLICT BETWEEN LEVEL AS GIVEN BY ORIGINAL DEFINITION AND  -REDEF-.
82,00   0   INCORRECT USAGE OF FIGURATIVE CONSTANT.
83,00   0   INVALID FORM OF -DO- STATEMENT.
84,00   0   ILLEGAL MOVE  - FROM 'NAME.1' ( E  A(2) )
              TO 'NAME.2' ( IR  999 ). NOTHING DONE .
85,00   0   PERMANENT READ ERROR IN PHASE 2. COMPILATION SUSPECT.
86,00   0   DIFFICULT TO PROGRAM KEY SETTING. CHECK ENVIRONMENT DESCRIPTION.
```

```
 87,00    0    PROBABLE PROGRAM CONTINUITY ERROR. PROGRAM FLOWS INTO  *DATA.
 88,00    0    -COND- CARD LACKS NAME IN COLUMNS 7 THROUGH 22.
 89,00    0    -FILE- CARD FORMAT ERROR.
 90,00    0    THIS ENVIRONMENT TYPE NOT YET PROCESSED BY COMPILER.
 91,00    0    NUMERIC INTEGER MUST FOLLOW -BLOCKSIZE- IN THE -FILE- CARD.
 92,00    0    STATEMENT OR SECTION NAME MUST FOLLOW -ONERROR- IN THE -FILE- CARD.
 93,00    0    STATEMENT OR SECTION NAME MUST FOLLOW -FORLABEL- IN THE -FILE- CARD.
 94,00    0    DATA NAME MUST FOLLOW -PLACE LENGTH IN- IN THE -FILE- CARD.
 95,00    0    DATA NAME MUST FOLLOW -FIND LENGTH IN- IN THE -FILE- CARD.
 96,00    0    THERE IS AN ILLEGAL WORD IN THE -FILE- CARD.
 97,00    0    INVALID -CORRESPONDING- STATEMENT.
 98,00    0    CHECK DATA DESCRIPTION OF ARRAY OF ELEMENTS, 'NAME.1'.
 99,00    0    PROBABLE PROGRAM CONTINUITY ERROR. PROGRAM FLOWS INTO STATEMENT OR SECTION
                  'NAME.1' ADDRESSED BY A -DO-.
100,00    0    DATA DESCRIPTION CONTAINS PICTORIAL WHICH EXCEEDS LEGAL LIMIT OF 30 CHARACTERS.
101,00    0    'NAME.1' IS AN IMPROPERLY QUALIFIED NAME.
102,00    0    NAME OF NUMERIC TYPE MUST FOLLOW -QUANTITY IN-.
                  'NAME.1' IS ALPHABETIC TYPE.
103,00    0    DATA NAME HAVING -RECORD-, -COND-, -REDEF- OR -LABEL- CANNOT BE -QUANTITY- ITEM.
104,00    0    -REDEF- OR -LABEL- OCCURRING BETWEEN NONFORMAT AND FORMAT DESCRIBED
                  LEVELS MAY AFFECT POSITIONING ADVERSELY.
105,00    0    -QUANTITY- ITEM 'NAME.1' CANNOT FOLLOW VARIABLE FIELD WHICH
                  DEPENDS ON THE -QUANTITY- ITEM ITSELF.
106,00    0    STATEMENT OR SECTION NAME MUST FOLLOW -AT END-. CHECK 'NAME.1'.
107,00    0    ILLEGAL COMPARISON STRUCTURE.
108,00    0    'NAME.1' IS AN UNDEFINED SYMBOL.
109,00    0    PROCESSOR UNABLE TO FIND VARIABLE USED AS SUBSCRIPT. POSSIBLE COMPILER ERROR.
110,00    0    -COPY- AND -LIBRARY- ARE NOT YET HANDLED BY SYSTEM.
111,00    0    'NAME.1' HAS IMPROPER DATA FORMAT FOR THIS USE IN THE
                  -FIND LENGTH IN- OPTION.
112,00    0    'NAME.1' HAS IMPROPER DATA FORMAT FOR THIS USE IN THE
                  -PLACE LENGTH IN- OPTION.
113,00    0    REDUNDANT RIGHT PARENTHESIS ELIMINATED .
114,00    0    REDUNDANT LEFT PARENTHESIS.
115,00    0    INVALID MACHINE OPERATION.
116,00    0    MISSING OPERAND ASSUMED TO BE ZERO.
117,00    0    -FIND LENGTH IN- OPTION USED WITH SOME BUT NOT ALL OF THE RECORDS BELONGING TO THE FILE
                  'NAME.1' REFERENCED BY A -GET RECORD FROM-.
118,00    0    -PLACE LENGTH IN- OPTION USED WITH SOME BUT NOT ALL OF THE RECORDS BELONGING TO THE FILE
                  'NAME.1' REFERENCED BY A -GET RECORD FROM-.
119,00    0    INCOMPLETE -MOVE- EXPRESSION.
120,00    0    'NAME.1' ELIMINATED FROM ADD BECAUSE OF IMPROPER DATA FORMAT ( E  A(2) ).
121,00    0    -BLOCK CONTROL- OPTION USED WITH SOME BUT NOT ALL OF THE RECORDS BELONGING TO THE FILE
                  'NAME.1' REFERENCED BY A -GET RECORD FROM-.
122,00    0    INCOMPLETE STATEMENT DELETED FROM TEXT.
123,00    0    CANNOT USE VARIABLE LENGTH ITEMS FOR COMPARISON.
124,00    0    ILLEGAL INTERNAL CONDITION. NOTHING DONE. POSSIBLE COMPILER ERROR.
125,00    0    STATEMENT WITHOUT PROPER VERB DELETED FROM TEXT.
126,00    0    STATEMENT WITH MORE THAN ONE VERB DELETED FROM TEXT.
127,00    0    TRANSFER BYPASSED BECAUSE 'NAME.1' IS NOT A STATEMENT OR SECTION NAME.
128,00    0    TRANSFER BYPASSED BECAUSE 'NAME.1' IS A
                  STATEMENT OR SECTION NAME ADDRESSED BY A -DO-.
129,00    0    FORMAT ERROR FOR TRANSFER INDEX. NOTHING DONE.
130,00    0    TRANSFER INDEX NOT AN INTEGER. INTEGRAL PART TAKEN AS VALUE.
131,00    0    INVALID DISPLAY STATEMENT.
```

132,00	0	END OF FILE ON JOB TAPE WITHOUT *FINISH CARD.
133,00	0	NO RIGHT PARENTHESIS IN FORMAT PICTORIAL.
134,00	0	ILLEGAL CHARACTER REPLACED IN INTERNAL TEXT BY 0, AND IN EXTERNAL TEXT BY $.
135,00	0	PERMANENT READ ERROR FOR INPUT. DUBIOUS COMPILATION.
136,00	0	REDUNDANCY WHILE WRITING EXTERNAL DICTIONARY. DUBIOUS COMPILATION.
137,00	0	REDUNDANCY WHILE READING EXTERNAL DICTIONARY. DUBIOUS COMPILATION.
138,00	0	FILE NAME SHOULD FOLLOW -CLOSE-.
139,00	0	FILE NAME SHOULD FOLLOW -OPEN-.
140,00	0	INTERNAL TEXT SYNCHRONIZATION FAILURE. DUBIOUS COMPILATION.
141,00	0	MORE THAN ONE -PROGRAM.START-. FIRST USED.
142,00	0	-PROGRAM.START- MUST BE A STATEMENT OR SECTION NAME.
143,00	0	-PROGRAM.START- CANNOT BE A STATEMENT OR SECTION NAME ADDRESSED BY A -DO-.
144,00	0	ILLEGAL ENVIRONMENT CARD TYPE.
145,00	0	MISSING ADDRESS.
146,00	0	MISSING TAG.
147,00	0	MISSING DECREMENT.
148,00	0	LENGTH OF ALPHABETIC CONSTANT EXCEEDS INTERNAL TABLE CAPACITY AND SHOULD BE SUBDIVIDED.
149,00	0	NUMBER OF SECTIONS IN PROGRAM EXCEEDS INTERNAL TABLE CAPACITY.
150,00	0	ALPHABETIC LITERAL EXCEEDS 50 CHARACTERS.
151,00	0	VFD IS NOT YET HANDLED BY SYSTEM.
152,00	0	'NAME.1' SHOULD NOT BE USED AS DATA NAME.
153,00	0	-SPECIF- CARD FORMAT ERROR.
154,00	0	FILE NAME MUST BE FIRST ITEM IN VARIABLE FIELD.
155,00	0	ALPHAMERIC LITERAL MUST FOLLOW -UNIT-.
156,00	0	ALPHAMERIC LITERAL MUST FOLLOW -SERIAL-.
157,00	0	ALPHAMERIC LITERAL MUST FOLLOW -REEL-.
158,00	0	NUMERIC INTEGER MUST FOLLOW -RETAIN-.
159,00	0	NUMERIC INTEGER MUST FOLLOW -ACTIVITY-.
160,00	0	ALPHABETIC LITERAL FOLLOWING KEY WORD CANNOT EXCEED 6 CHARACTERS.
161,00	0	-POOL- CARD FORMAT ERROR.
162,00	0	NUMERIC INTEGER MUST FOLLOW -BLOCKSIZE- ON -POOL- CARD.
163,00	0	NUMERIC INTEGER MUST FOLLOW -BUFFERCOUNT-.
164,00	0	-GROUP- CARD FORMAT ERROR.
165,00	0	NUMERIC INTEGER MUST FOLLOW -OPENCOUNT-.
166,00	0	'NAME.1' IS NOT UNIQUE IN THIS SECTION.
167,00	0	SECOND QUOTE MARK MISSING.
168,00	0	ALPHABETIC LITERAL EXTENDS ACROSS CARDS.
169,00	0	PROBABLE PROGRAM CONTINUITY ERROR. PROGRAM FLOWS INTO GENERATED CONSTANTS.
170,00	0	-WHEN- SUBSTITUTED FOR -IF- BECAUSE OF IMPROPER USE.
171,00	0	NUMBER OF OPERATORS IN THIS SENTENCE EXCEEDS MAXIMUM OF 60. SENTENCE DELETED FROM TEXT.
172,00	0	CONSTANT POOL OVERFLOW.
173,00	0	REFERENCE MADE TO NON-EXISTENT SYSTEM GENERATED NAME.
174,00	0	REFERENCE TO SYSTEM SUBROUTINE LACKS PROPER NUMBER. ZERO ASSUMED.
175,00	0	NO -STOP RUN- IN PROGRAM.
176,00	0	-CONTRL- CARD FORMAT ERROR.
177,00	0	THIS SENTENCE EXCEEDS INTERNAL TABLE CAPACITY. SENTENCE DELETED FROM TEXT.
178,00	0	PROCEDURE KEY WORD USED IN DATA OR ENVIRONMENT, INTERPRETED AS A DATA NAME.
179,00	0	-END- SECTION MUST BE THE ONLY CLAUSE IN THE SENTENCE.
180,00	0	MOVE OF FIGURATIVE CONSTANT TO VARIABLE LENGTH FIELD NOT YET HANDLED BY SYSTEM.
181,00	0	MOVE OF FIGURATIVE CONSTANT TO FIELD LONGER THAN 32766 CHARACTERS NOT YET HANDLED BY SYSTEM.
182,00	0	IMPROPER DATA FORMAT.
183,00	0	NUMBER OF SUBSCRIPT EXPRESSIONS USED EXCEEDS INTERNAL TABLE CAPACITY.
184,00	0	NUMBER OF SUBSCRIPTED NAMES USED EXCEEDS INTERNAL TABLE CAPACITY.
185,00	0	DATA DESCRIPTION PICTORIAL ERROR.
186,00	0	DATA OR ENVIRONMENT FIXED FIELD INFORMATION SHOULD BE PUNCHED IN ONLY THE FIRST CARD

```
                OF A MULTIPLE CARD GROUP. POSSIBLE CONTINUATION CHARACTER ERROR.
187,00    0     CONDITIONAL EXPRESSION TEST CAPACITY EXCEEDED.
                REWRITE AS TWO OR MORE SEPARATE EXPRESSIONS, EACH WITH ILLEGAL SENTENCE STRUCTURE NOTHING DONE.
188,00    0     'NAME.1', ADDRESSED BY A -DO-, IS NEITHER A STATEMENT NOR A SECTION NAME.
189,00    0     EXTERNAL MODE SUBSTITUTED FOR ILLEGAL MODE CHARACTER.
190,00    0     FIELD IS NOT JUSTIFIED BECAUSE OF ILLEGAL JUSTIFICATION CHARACTER.
191,00    0     'NAME.1' IS NOT PROPERLY DEFINED.
192,00    0     SENTENCE STRUCTURE ERROR. POSSIBLE ILLEGAL USE OF A KEY WORD.
193,00    0     LIMIT OF 63 FILES EXCEEDED.
194,00    0     DATA NAME LACKS LEVEL.
195,00    0     CANNOT FILE RECORD 'NAME.2' IN THIS FILE,
                   'NAME.1', BECAUSE ENVIRONMENT FILE CARD LACKS THIS RECORD NAME.
196,00    0     ILLEGAL SENTENCE STRUCTURE NOTHING DONE.
197,00    0     RECORD NAME MUST PRECEDE DESCRIPTION OF RECORD.
198,00    0     NO RECORDS PROCESSED IN FILE 'NAME.1'
199,00    0     UPSCALE MAY CAUSE HIGH ORDER TRUNCATION FOR STORE INTO 'NAME.1'
200,00    0     NUMBER OF VARIABLE LENGTH FIELDS EXCEEDS INTERNAL TABLE CAPACITY.
201,00    0     NUMBER OF NESTED LEVELS IN 'NAME.1' EXCEEDS INTERNAL TABLE CAPACITY.
202,00    0     NUMBER OF DATA GROUPS ASSOCIATED WITH BASE LOCATOR EXCEEDS INTERNAL TABLE CAPACITY.
203,00    0     NUMBER OF ARRAYS EXCEEDS INTERNAL TABLE CAPACITY.
204,00    0     NUMBER OF DIFFERENT EDIT FIELDS EXCEEDS INTERNAL TABLE CAPACITY.
205,00    0     INTERNAL TABLE OVERFLOW. REDUCE DUPLICATE USAGE OF SUBSCRIPT VARIABLE NAMES.
206,00    0     'NAME.1' HAS INEFFICIENT FORMAT FOR SUBSCRIPT VARIABLE.
207,00    0     -CONTRL- NAME MUST BE UNIQUE AND 6 CHARACTERS OR LESS.
208,00    0     SENTENCE CANNOT START WITH -OTHERWISE-
209,00    0     'NAME.1' HAS INSUFFICIENT BLOCKSIZE. BLOCKSIZE USED IS

SEVERITY LIMIT WAS NOT REACHED
```

B. Severity Codes

The Severity Codes (values) are numbered 1 through 5. The least severe errors have a code of 1. The most severe errors have a code of 5. The ascending sequence, 2, 3, and 4 indicates the degree of severity between the least and most severe errors, 1 and 5.

An error severity code of 1 does not prevent the running of the object program immediately after compilation. Any code above 1 does prevent running of the object program immediately after compilation. That is, the compiler will not compile and go.

An error severity code of 5 causes the compiler to stop compiling. It then proceeds to the next job.

APPENDIX 90.05

SAMPLE PROBLEM

INTRODUCTION

This section defines a sample problem and gives actual results from compilation and execution of the problem.

90.05 Sample Problem

A. Introduction

The following sample problem is provided to illustrate various features and characteristics of the Commercial Translator system. The problem presented here is essentially the example described in the Commercial Translator General Information Manual. Modifications of two types have been made: those which were necessary for operation on the 7090, and those which improved the organization and efficiency of the problem.

B. General Problem Statement

The problem presented here is a simple payroll application. A master file and a detail file containing records for each employee are input and matched. A check, an updated master record, and possibly a bond order are output for each master-detail match. An error notice is given for unmatched masters or details. In addition to individual pay records, the payroll register produced contains departmental totals and a final summation of the run.

C. The MASTER File

For each employee on the payroll there is in the master file a record containing indicative information such as name, rate of pay, and number of exemptions. Accumulated year-to-date totals are also carried in the record.

1. Data Description

Data Description entries detail the structure of individual records and the formats of component fields. The fields which are used in arithmetic expressions are maintained in internal format; right justification is specified for increased operating efficiency.

The smallest unit of information which may be described by a Data Description entry and referenced by a procedure statement is the character composed of six bits. Individual bits may not be described or referenced in the language of Commercial Translator. The 9's which describe a field in internal mode inform the processor of the largest magnitude a field may attain and of the number of words (when R is specified) or the number of characters (when L or no justification is specified) required to contain the field.

Master records on tape or in storage occupy 709/7090 words in the following manner:

Word 1	EMPLOYEE.NUMBER
Word 2	first 6 characters of NAME
Word 3	second 6 characters of NAME
Word 4	last 3 characters of NAME
Word 5	RATE
Word 6	DATE
Word 7	EXEMPTIONS
Word 8	GROSS
Word 9	RETIREMENT
Word 10	INSURANCE
Word 11	FICA
Word 12	WHT
Word 13	BONDEDUCTION
Word 14	BONDACCUMULATION
Word 15	BONDENOMINATION

2. Environment for MASTER Records

The master file is a standard binary tape file containing only MASTER records. Twenty 15-word records are grouped to form 300-word blocks on tape, with each record complete within a block, since it is desired to process input records in the buffer area, in the locate mode. The FILE and SPECIF cards for INPUTMASTER and OUTPUTMASTER, provide the system with these characteristics.

D. The DETAIL File

The detail input file, DETAILFILE, consists of records containing a date, total hours worked during a pay period, and the employee number of an employee on the payroll.

1. Data Description

Data Description entries specify that all fields of the detail records are in external mode since they are generated on a card punch. HOURS is the only field upon which arithmetic operations are performed, and consequently, is the only field specified as numeric (9's in description). Detail records consist of the following 709/7090 words:

Word 1	EMPLOYEE NUMBER
Word 2	DATE
Word 3	HOURS in the first three characters 3 blanks (supplied automatically)

2. Environment Description

The detail input file is a non-labeled, ungrouped, BCD tape file produced on card-to-tape equipment. On the input tape a 3-word record occupies the first portion of tape blocks 14 words long.

Only the 3-word record is to be brought into storage and processed, and the specification BLOCKSIZE 3 on the DETAILFILE FILE cards accomplishes this aim. Alternatively, BEGIN and BLOCKSIZE 14 might have been specified on the FILE card which would have enabled correct processing but allowed the full 14-word block to enter core.

E. Output Report Files

In this problem four output files are prepared for listing: CHECKFILE, BONDORDERFILE, PAYFILE, ERRORFILE.

1. The CHECKFILE

a) Data Description

The Data Description of CHECK records in the CHECKFILE specifies a 2-line check to be printed under carriage control on a 720 printer. A standard carriage control tape is assumed with Channel 1 associated with the first check line and Channel 2 four spaces away for the second line. The two check lines are grouped in one record, separated by a record mark (signalled by type RCDMRK) for printer control.

The CHECK records would be composed of:

Word 1	carriage control character
	2 characters for MONTH
	a dash
	2 characters for DAY
Word 2	a dash
	2 characters for YEAR
	3 blanks
Word 3	6 blanks
Word 4	6 blanks
Word 5	6 blanks
Word 6	6 blanks
Word 7	3 blanks
	first 3 characters of EMPLOYEE.NUMBER
Word 8	last 3 characters of EMPLOYEE.NUMBER
	record mark
	carriage control character

	first character of NAME
Word 9	next 6 characters of NAME
Word 10	next 6 characters of NAME
Word 11	last 5 characters of NAME
	1 blank
Word 12	6 blanks
Word 13	6 blanks
Word 14	6 blanks
Word 15	1 blank
	first 5 characters of AMOUNT
Word 16	last 3 characters of AMOUNT
	3 blanks (supplied automatically)

b) Environment

The CHECKFILE is a BCD output tape file. A single record composed of the two lines is output as a block as specified by BLOCKSIZE 20 on the FILE card.

2. The PAYFILE

a) Data Description

Two types of records appear in the PAYFILE: an individual PAYRECORD which corresponds with each check issued, and DEPARTMENT.TOTAL summations. Department totals are accumulated in the internally formatted working area, INTERNAL.TOTALS, until a change in department number occurs; the fields of DEPARTMENT.TOTAL are external edited versions of INTERNAL.TOTALS.

In output report records blanks may be specified as a constant. If the format specification indicates the length of the field, it is unnecessary to provide the proper number of blanks between quote marks.

b) Environment Description

The FILE card for the PAYFILE describes an output, BCD tape file which may contain records PAYRECORD and DEPARTMENT.TOTAL. A BLOCKSIZE of 20 is specified to accommodate PAYRECORD. The shorter records, DEPARTMENT.TOTAL, will be written with proper length, and will always begin a new buffer. If this short record was only 10 words long, it would be necessary to specify BEGIN in the File Card to avoid grouping of the short records.

COMPILATION OF SAMPLE PROBLEM

DATE 10/18/61 TIME 2.45 ACCOUNT ID. CT PUBLICATIONS PAGE 1

```
*COMPILE LIST                                   CT PUBLICATIONS
CTC
                    *DATA
          1,00  71175 MASTER          1RECORD     L
          2,00  71177   DAT          2
          3,00          EMPLOYEE.NUM 3
                71200           BER
          4,00  71201     DEPARTMENT 4            AA
          5,00  71202     EMPLOYEE   4            AAAA
          6,00  71203   NAME         3            A(15)
          7,00  71204 TRIGGERS       2            AAA
          8,00  71205 RATE           2            IR99V999
          9,00  71206 DATE           2
         10,00  71207   MONTH        3            AA
         11,00  71210   DAY          3            AA
         12,00  71211   YEAR         3            AA
         13,00  71212 EXEMPTIONS     2            IR99
         14,00  71213 TOTALS         2
         15,00  71214   GROSS        3            IR9(4)V99
         16,00  71215   RETIREMENT   3            IR999V99
         17,00  71216   INSURANCE    3            IR999V99
         18,00  71217 FICA           2            IR999V99
         19,00  71220 WHT            2            IR9(4)V99
         20,00  71221 BONDEDUCTION   2            IR99V99
         21,00        BONDACCUMULATI 2            IR999V99
                71222           ON
         22,00        BONDENOMINATIO 2            IR999V99
                71223           N
         23,00  71224 DETAIL        01RECORD
         24,00  71226 EMPLOYEE.NUMBER02
         25,00  71227   DEPARTMENT   03           AA
         26,00  71230   EMPLOYEE     03           AAAA
         27,00  71231 DATE          02
         28,00  71232   MONTH        03           AA
         29,00  71233   DAY          03           AA
         30,00  71234   YEAR         03           AA
         31,00  71235 HOURS          02           99V9
         32,00  71236 CHECK          01RECORD
         33,00  71240 CNTRLCHAR      02           A '1'
         34,00  71241 DATE           02
         35,00  71242   MONTH        03           AA
         36,00        GN)001         03           A '-'
         37,00  71244   DAY          03           AA
         38,00        GN)002         03           A '-'
         39,00  71246   YEAR         03           AA
         40,00        GN)003         03           A(30) ' '
         41,00  71250 EMPLOYEE.NUMBER02           A(6)
         42,00  71251 ENDFRSTLINE    02RCDMRK     A
         43,00  71252 CNTRLCHARSECLIN02           A '2'
         44,00  71253 NAME           02           A(18)
         45,00        GN)004         02           A(20) ' '
         46,00  71255 AMOUNT         02           $8889.99
         47,00  71256 PAYRECORD      1RECORD
```

DATE 10/18/61 TIME 2.45 ACCOUNT ID. CT PUBLICATIONS PAGE 2

	48,00	71260	EMPLOYEE.NUMBER	2	
	49,00	71261	DEPARTMENT	3	AA
	50,00		GN)005	3	A ' '
	51,00	71263	EMPLOYEE	3	AAAA
	52,00		GN)006	3	A ' '
	53,00	71265	NAME	2	A(15)
	54,00		GN)007	2	A ' '
	55,00	71267	DATE	2	
	56,00	71270	MONTH	3	AA
	57,00		GN)008	3	A '-'
	58,00	71272	DAY	3	AA
	59,00		GN)009	3	A '-'
	60,00	71274	YEAR	3	AA
	61,00		GN)010	2	AAA ' '
	62,00	71276	HRS	2	8889.9
	63,00		GN)011	2	AA ' '
	64,00	71300	GROSS	2	88889.99
	65,00		GN)012	2	AA ' '
	66,00	71302	WHT	2	88889.99
	67,00		GN)013	2	AA ' '
	68,00	71304	FICA	2	8889.99
	69,00		GN)014	2	AA ' '
	70,00	71306	BONDEDUCTION	2	8889.99
	71,00		GN)015	2	AA ' '
	72,00	71310	INS.PREM	2	8889.99
	73,00		GN)016	2	AA ' '
	74,00	71312	RET.PREM	2	8889.99
	75,00		GN)017	2	AAA ' '
	76,00	71314	NETPAY	2	8889.99
	77,00		GN)018	2	AA ' '
	78,00	71316	BONDENCMINATION	2	88889.99
	79,00	71317	DEPARTMENT.TOTAL	1RECORD	
	80,00		GN)019	2	A(21) ' DEPARTMENT '
	81,00	71322	CURRENT.DEPT	2	AA
	82,00		GN)020	2	A(12) ' TOTALS '
12	83,00	71324	HOURS	2	8889.9
	84,00		GN)021	2	AA ' '
11	85,00	71326	GROSS	2	88889.99
	86,00		GN)022	2	AA ' '
10	87,00	71330	WHT	2	88889.99
	88,00		GN)023	2	AA ' '
9	89,00	71332	FICA	2	8889.99
	90,00		GN)024	2	AA ' '
8	91,00	71334	BONDEDUCTION	2	8889.99
	92,00		GN)025	2	AA ' '
7	93,00	71336	INSURANCE	2	8889.99
	94,00		GN)026	2	AA ' '
6	95,00	71340	RETIREMENT	2	8889.99
	96,00		GN)027	2	AA ' '
5	97,00	71342	NETPAY	2	88889.99
	98,00		GN)028	2	AA ' '
4	99,00	71344	BONDPURCHASES	2	88889.99
3	100,00	71345	BONDORDER	1RECORD	
	101,00		GN)029	2	A ' '
2	102,00	71350	DAT	2	

```
103,00          EMPLOYEE.NUM 3          A(6)
        71351           BER
104,00      GNJ030          3           A ' '
105,00  71353   NAME        3           A(15)
106,00      GNJ031          3           A ' '
107,00      BONDENOMINATIO  2           899V99
        71355           N
108,00      GNJ032          2           A ' '
109,00  71357   DATE        2           A(6)
110,00  71360 ERROROUT      1RECORD
111,00  71362   ERRORTYPE   2           A
112,00  71363   INFO        2           A(23)
113,00  71364 WORKING       1
114,00  71365   GROSS       2           IR9(5)V99
115,00  71366   RETIREMENT  2           IR9(4)V99
116,00  71367   INSURANCE   2           IR9(4)V99
117,00  71370   FICA        2           IR9(4)V99
118,00  71371   WHT         2           IR9(5)V99
119,00  71372   NETPAY      2           IR9(5)V99
120,00  71373   HOURS       2           IR99V9
121,00  71374   INDEX       2           IR99
122,00  71375   POS         2           IR99
123,00  71376 INTERNAL.TOTALS 1
124,00  71377   HOURS       2           IR9999V9
125,00  71400   GROSS       2           IR9(5)V99
126,00  71401   WHT         2           IR9(5)V99
127,00  71402   FICA        2           IR9(4)V99
128,00  71403   BONDEDUCTION 2          IR9(4)V99
129,00  71404   INSURANCE   2           IR9(4)V99
130,00  71405   RETIREMENT  2           IR9(4)V99
131,00  71406   NETPAY      2           IR9(5)V99
132,00  71407   BONDPURCHASES 2         IR9(5)V99
133,00  71410 GRAND.TOTALS  1
134,00  71411   HOURS       2           IR9(4)V9
135,00  71412   GROSS       2           IR9(5)V99
136,00  71413   WHT         2           IR9(5)V99
137,00  71414   FICA        2           IR9(4)V99
138,00  71415   BONDEDUCTION 2          IR9(4)V99
139,00  71416   INSURANCE   2           IR9(4)V99
140,00  71417   RETIREMENT  2           IR9(4)V99
141,00  71420   NETPAY      2           IR9(5)V99
142,00  71421   BONDPURCHASES 2         IR9(5)V99
143,00  71422 TABLE         1
144,00      GNJ033          2           IR9(5) '00999'
145,00      GNJ034          2             '080060'
146,00      GNJ035          2           IR9(5) '01499'
147,00      GNJ036          2             '100060'
148,00      GNJ037          2           IR9(5) '01999'
149,00      GNJ038          2             '120090'
150,00      GNJ039          2           IR9(5) '02499'
151,00      GNJ040          2             '150090'
152,00      GNJ041          2           IR9(5) '02999'
153,00      GNJ042          2             '150120'
154,00      GNJ043          2           IR9(5) '03499'
155,00      GNJ044          2             '200120'
```

```
156,00          GN)045          2          IR9(5) '03999'
157,00          GN)046          2            '200150'
158,00          GN)047          2          IR9(5) '04499'
159,00          GN)048          2            '250150'
160,00          GN)049          2          IR9(5) '04999'
161,00          GN)050          2            '300180'
162,00          GN)051          2          IR9(5) '06499'
163,00          GN)052          2            '300250'
164,00          GN)053          2          IR9(5) '07999'
165,00          GN)054          2            '300350'
166,00          GN)055          2          IR9(5) '99999'
167,00          GN)056          2            '300500'
168,00          GN)057        REDEF          TABLE
169,00  71454 TABLE.ITEM        1          12
170,00  71455 RATE              2            IR99V999
171,00  71456 INSPREM           2            9V99
172,00  71457 RETPREM           2            9V99

                *ENVIRONMENT
173,00          INPUTMASTER     FILE  INPUT,BINARY,TAPE,MASTER,BLOCKSIZE 300
174,00                          SPECIFINPUTMASTER, UNIT1 'D1',OPENW,CLOSER
175,00          OUTPUTMASTER    FILE  OUTPUT,BINARY,TAPE,MASTER,BLOCKSIZE 300
176,00                          SPECIFOUTPUTMASTER, UNIT1 'C1',OPENW,CLOSER
177,00          DETAILFILE      FILE  INPUT,BCD,TAPE,DETAIL,BLOCKSIZE 3
178,00                          SPECIFDETAILFILE,UNIT1 'C2',OPENW,CLOSER,LOW
179,00          CHECKFILE       FILE  OUTPUT,BCD,TAPE,CHECK,BLOCKSIZE 20
180,00                          SPECIFCHECKFILE,UNIT1 'D2',OPENW,CLOSER,LOW
181,00          PAYFILE         FILE  OUTPUT,BCD,TAPE,PAYRECORD,
                                      DEPARTMENT.TOTAL,BLOCKSIZE 20
182,00                          SPECIFPAYFILE, UNIT1 'D3',OPENW, CLOSER,LOW
183,00          BONDORDERFILE   FILE  OUTPUT,BCD,TAPE,BONDORDER,BLOCKSIZE 6
184,00                          SPECIFBONDORDERFILE,UNIT1 'C3',OPENW,CLOSER,LOW
185,00          ERRORFILE       FILE  OUTPUT,BCD,TAPE,ERROROUT,BLOCKSIZE 6
186,00                          SPECIFERRORFILE,UNIT1 'D4',OPENW,CLOSER,LOW

                *PROCEDURE
187,00  71461 GN)000CALL (MASTER EMPLOYEE.NUMBER)   M.EMP.NO,
        71462            (DETAIL EMPLOYEE.NUMBER)   D.EMP.NO,
        71463            (DETAIL DEPARTMENT)        D.DEPT,
        71464            (MASTER BONDEDUCTION)      M.BND.DED,
        71465            (MASTER BONDACCUMULATION)  M.BND.ACC.
188,00  71466 START.          OPEN ALL FILES,
                        MOVE ZEROS TO INTERNAL.TOTALS, GRAND.TOTALS,
                        GET MASTER, AT END DO END.OF.MASTERS.
189,00                  MOVE MASTER DEPARTMENT TO CURRENT.DEPT,  GO TO GET.DETAIL.
190,00  71471 GET.MASTER.  GET MASTER, AT END DO END.OF.MASTERS.
191,00  71474 GET.DETAIL.  GET DETAIL, AT END GO TO END.OF.DETAILS.
192,00  71477 COMPARE.EMPLOYEE.NUMBERS.  GO TO CHECK.NEW.DEPT WHEN D.EMP.NO =
                        M.EMP.NO,  LOW.DETAIL WHEN D.EMP.NO LT M.EMP.NO.
193,00  71500 HIGH.DETAIL.  MOVE 'M' TO ERRORTYPE,  MOVE MASTER DAT  TO
                        ERROROUT INFO,
                        FILE ERROROUT.
194,00                  GET MASTER, AT END DO END.OF.MASTERS.
195,00                  GO TO COMPARE.EMPLOYEE.NUMBERS.
196,00  71503 LOW.DETAIL. MOVE DETAIL TO ERROROUT INFO, MOVE 'D' TO ERRORTYPE,
```

```
                                    FILE ERROKOUT,
                                    GO TO GET.DETAIL.
           197,00   71504 END.OF.MASTERS.  IF D.EMP.NO = HIGH.VALUE THEN GO TO END.OF.RUN
                                    OTHERWISE SET M.EMP.NO = HIGH.VALUE.
           198,00   71507 END.OF.DETAILS.  IF M.EMP.NO = HIGH.VALUE THEN GO TO END.OF.RUN
                                    OTHERWISE SET D.EMP.NO = HIGH.VALUE,  GO TO HIGH.DETAIL.
           199,00   71512 END.OF.RUN.  DO DEPARTMENT.END,
                                    MOVE CORRESPONDING GRAND.TOTALS TO PAYRECORD,
                                    MOVE GRAND.TOTALS HOURS TO PAYRECORD HRS,
                                    MOVE GRAND.TOTALS INSURANCE TO PAYRECORD INS.PREM,
                                    MOVE GRAND.TOTALS RETIREMENT TO PAYRECORD RET.PREM,
                                    MOVE 'GT' TO PAYRECORD DEPARTMENT,
                                    MOVE  BLANKS TO PAYRECORD NAME, PAYRECORD EMPLOYEE,
                                    PAYRECORD MONTH, PAYRECORD DAY, PAYRECORD YEAR,
                                    MOVE GRAND.TOTALS BONDPURCHASES TO PAYRECORD
                                    BONDENOMINATION,  FILE PAYRECORD,
                                    CLOSE ALL FILES,  STOP RUN.
           200,00   71513 CHECK.NEW.DEPT.  IF CURRENT.DEPT IS NOT EQUAL TO D.DEPT THEN
                                    DO DEPARTMENT.END OTHERWISE GO TO COMPUTE.PAY.
           201,00                   MOVE ZEROS TO INTERNAL.TOTALS,
                                    MOVE D.DEPT TO CURRENT.DEPT.
           202,00   71516 COMPUTE.PAY.  MOVE DETAIL DATE TO MASTER DATE,  MOVE DETAIL
                                    HOURS TO WORKING HOURS, PAYRECORD HRS.
           203,00                   IF WORKING HOURS GT 40.0 THEN SET WORKING GROSS = (WORKING
                                    HOURS * 1.5 -20) * MASTER RATE OTHERWISE SET WORKING
                                    GROSS = WORKING HOURS * MASTER RATE.
           204,00                   DO FICA.ROUTINE,
                                    DO WITHOLDING.TAX.ROUTINE.
           205,00                   IF M.BND.DED IS NOT EQUAL TO ZERO THEN DO BOND.ROUTINE
                                    OTHERWISE MOVE BLANKS TO PAYRECORD BONDEDUCTION,PAYRECORD
                                    BONDENOMINATION.
           206,00                   DO SEARCH FOR INDEX = 1(1)12.
           207,00   71523 NET.      SET WORKING NETPAY = WORKING GROSS - WORKING FICA - WORKING
                                    WHT - WORKING RETIREMENT - WORKING INSURANCE - M.BND.DED.
           208,00                   ADD CORRESPONDING WORKING TO MASTER TOTALS, INTERNAL.TOTALS,
                                    MOVE CORRESPONDING DETAIL TO PAYRECORD, CHECK,
                                    MOVE CORRESPONDING WORKING TO PAYRECORD,
                                    MOVE MASTER NAME TO PAYRECORD NAME, CHECK NAME,
                                    MOVE WORKING NETPAY TO CHECK AMOUNT,
                                    FILE CHECK,
                                    FILE PAYRECORD,
                                    FILE MASTER.
           209,00                   GO TO GET.MASTER.
           210,00   71524 FICA.ROUTINE.  BEGIN SECTION.
           211,00                   SET WORKING FICA = .03 * WORKING GROSS,
                                    ADD WORKING FICA TO MASTER FICA.
           212,00                   IF MASTER FICA GT 144.00 THEN SET WORKING FICA = WORKING
                                    FICA - (MASTER FICA - 144.00), SET MASTER FICA = 144.00.
           213,00         GN)077END FICA.ROUTINE.
           214,00   71527 WITHOLDING.TAX.ROUTINE.  BEGIN SECTION.
           215,00                   SET WORKING WHT = 0.18 * (WORKING GROSS - 13 * MASTER
                                    EXEMPTIONS),
                                    SET WORKING WHT = WORKING WHT * IR(WORKING WHT GT 0),
                                    ADD WORKING WHT TO MASTER WHT.
           216,00         GN)078END WITHOLDING.TAX.ROUTINE.
```

```
217,00   71531 BOND.ROUTINE.  BEGIN SECTION.
218,00         ADD M.BND.DED TO M.BND.ACC,INTERNAL.TOTALS BONDEDUCTION.
219,00         MOVE M.BND.DED TO PAYRECORD BONDEDUCTION.
220,00         IF MASTER BONDENOMINATION NOT GT M.BND.ACC THEN SET
               M.BND.ACC = M.BND.ACC - MASTER BONDENOMINATION OTHERWISE
               MOVE BLANKS TO PAYRECORD BONDENOMINATION,  GO TO BOND.END.
221,00         MOVE CORRESPONDING MASTER TO BONDORDER,  ADD BONDORDER
               BONDENOMINATION TO INTERNAL.TOTALS BONDPURCHASES,MOVE
               BONDORDER BONDENOMINATION TO PAYRECORD BONDENOMINATION.
222,00         FILE BONDORDER.
223,00   71534 BOND.END.  END BOND.ROUTINE.
224,00   71535 SEARCH.  BEGIN SECTION.
225,00         IF MASTER RATE GT TABLE.ITEM RATE (INDEX) THEN GO TO
               SEARCH.END OTHERWISE MOVE INDEX TO POS,
               MOVE INSPREM (POS) TO INS.PREM, WORKING INSURANCE,
               MOVE RETPREM (POS) TO RET.PREM, WORKING RETIREMENT,
               GO TO NET.
226,00   71540 SEARCH.END.  END SEARCH.
227,00   71541 DEPARTMENT.END.  BEGIN SECTION.
228,00         MOVE CORRESPONDING INTERNAL.TOTALS TO DEPARTMENT.TOTAL,
               FILE DEPARTMENT.TOTAL,
               ADD CORRESPONDING INTERNAL.TOTALS TO GRAND.TOTALS.
229,00       GN1083END DEPARTMENT.END.

CTD
CTE

NO ERRORS WERE DETECTED DURING COMPILATION
```

THE FOLLOWING LOADER CONTROL CARDS PRECEDE THE BINARY DECK.

```
*FILE  01 *D1       I HB                INPUTMASTER          1
*SPEC  01  300    N R                                        2
*FILE  02 *C1       P HB                OUTPUTMASTER         3
*SPEC  02  300    N R                                        4
*FILE  03 *C2       I LD                DETAILFILE           5
*SPEC  03    3    N R                                        6
*FILE  04 *D2       P LD                CHECKFILE            7
*SPEC  04   20    N R                                        8
*FILE  05 *D3       P LD                PAYFILE              9
*SPEC  05   20    N R                                       10
*FILE  06 *C3       P LD                BONDORDERFILE       11
*SPEC  05    6    N R                                       12
*FILE  07 *D4       P LD                ERRORFILE           13
*SPEC  07    6    N R                                       14
*CTEXT          DATE 101861 TIME  2.45  CT PUBLICATIONS     15
```

LOC	OCTAL	CNTRL			SYMBOLIC
	5 00000 0 01621	00001		USE	1
01671				BGN	2,PI11
	5 00000 0 00000	00001		USE	0
00000	2 00000 0 00000	00001	*DATA	BSS	0
00000	010000400000	10000	CHECK	OCT	010000400000
00001	400000606060	10000	+1	OCT	400000606060
00002	606060606060	10000	+2	OCT	606060606060
00003	606060606060	10000	+3	OCT	606060606060
00004	606060606060	10000	+4	OCT	606060606060
00005	606060606060	10000	+5	OCT	606060606060
00006	606060000000	10000	+6	OCT	606060000000
00007	000000720200	10000	+7	OCT	000000720200
00010	2 00000 0 00002	00001		BSS	2
00012	000000000060	10000	+1	OCT	000000000060
00013	606060606060	10000	+2	OCT	606060606060
00014	606060606060	10000	+3	OCT	606060606060
00015	606060606060	10000	+4	OCT	606060606060
00016	600000000000	10000	+5	OCT	600000000000
00017	000000606060	10000	+6	OCT	000000606060
00020	000060000000	10000	PAYRECORD	OCT	000060000000
00021	006000000000	10000	+1	OCT	006000000000
00022	2 00000 0 00001	00001		BSS	1
00023	000000000060	10000	+1	OCT	000000000060
00024	000040000040	10000	+2	OCT	000040000040
00025	000060606000	10000	+3	OCT	000060606000
00026	000000000060	10000	+4	OCT	000000000060
00027	600000000000	10000	+5	OCT	600000000000
00030	000000606000	10000	+6	OCT	000000606000
00031	2 00000 0 00001	00001		BSS	1
00032	006060000000	10000	+1	OCT	006060000000
00033	000000006060	10000	+2	OCT	000000006060
00034	2 00000 0 00001	00001		BSS	1
00035	006060000000	10000	+1	OCT	006060000000
00036	000000006060	10000	+2	OCT	000000006060
00037	2 00000 0 00001	00001		BSS	1
00040	006060600000	10000	+1	OCT	006060600000
00041	000000000060	10000	+2	OCT	000000000060
00042	600000000000	10000	+3	OCT	600000000000
00043	000000606060	10000	+4	OCT	000000606060
00044	606060606060	10000	DEPARTMENT.TOTAL	OCT	606060606060
00045	606060602425	10000	+1	OCT	606060602425
00046	472151634425	10000	+2	OCT	472151634425
00047	456360000060	10000	+3	OCT	456360000060
00050	634663214362	10000	+4	OCT	634663214362
00051	606060606000	10000	+5	OCT	606060606000
00052	000000000060	10000	+6	OCT	000000000060
00053	600000000000	10000	+7	OCT	600000000000
00054	000000606000	10000	+8	OCT	000000606000
00055	2 00000 0 00001	00001		BSS	1
00056	006060000000	10000	+1	OCT	006060000000
00057	000000006060	10000	+2	OCT	000000006060

```
00060  2 00000 0 00001   00001                    BSS     1
00061  006060000000      10000          +1       OCT     006060000000
00062  000000006060      10000          +2       OCT     000000006060
00063  2 00000 0 00001   00001                    BSS     1
00064  006060000000      10000          +1       OCT     006060000000
00065  000000000060      10000          +2       OCT     000000000060
00066  600000000000      10000          +3       OCT     600000000000
00067  000000606060      10000          +4       OCT     000000606060
00070  600000000000      10000  BONDORDER        OCT     600000000000
00071  006000000000      10000          +1       OCT     006000000000
00072  2 00000 0 00001   00001                    BSS     1
00073  000000000060      10000          +1       OCT     000000000060
00074  000000000060      10000          +2       OCT     000000000060
00075  2 00000 0 00001   00001                    BSS     1
00076  2 00000 0 00004   00001  ERROROUT          BSS     4
00102  2 00000 0 00011   00001  WORKING           BSS     9
00113  2 00000 0 00011   00001  INTERNAL.TOTALS
                                                  BSS     9
00124  2 00000 0 00011   00001  GRAND.TOTALS      BSS     9
00135  000000001747      10000  TABLE             OCT     000000001747
00136  001000000600      10000          +1       OCT     001000000600
00137  000000002733      10000          +2       OCT     000000002733
00140  010000000600      10000          +3       OCT     010000000600
00141  000000003717      10000          +4       OCT     000000003717
00142  010200001100      10000          +5       OCT     010200001100
00143  000000004703      10000          +6       OCT     000000004703
00144  010500001100      10000          +7       OCT     010500001100
00145  000000005667      10000          +8       OCT     000000005667
00146  010500010200      10000          +9       OCT     010500010200
00147  000000006653      10000         +10       OCT     000000006653
00150  020000010200      10000         +11       OCT     020000010200
00151  000000007637      10000         +12       OCT     000000007637
00152  020000010500      10000         +13       OCT     020000010500
00153  000000010623      10000         +14       OCT     000000010623
00154  020500010500      10000         +15       OCT     020500010500
00155  000000011607      10000         +16       OCT     000000011607
00156  030000011000      10000         +17       OCT     030000011000
00157  000000014543      10000         +18       OCT     000000014543
00160  030000020500      10000         +19       OCT     030000020500
00161  000000017477      10000         +20       OCT     000000017477
00162  030000030500      10000         +21       OCT     030000030500
00163  000000303237      10000         +22       OCT     000000303237
00164  030000050000      10000         +23       OCT     030000050000
00165                                  GN)000
00165  0074 00 4 00257   10010  START             TSX     SYS)175,4
00166  0 00000 0 00001   10010          +1       PZE     IOC)1
00167  0441 00 0 01744   10001          +2       LDI     CP)+40
00170  0604 00 0 00205   10010          +3       STI     SYS)133
00171  0074 00 4 00266   10010          +4       TSX     SYS)182,4
00172  1 00066 1 00364   10010          +5       TXI     SYS)244,1,54
00173  0441 00 0 01745   10001          +6       LDI     CP)+41
00174  0604 00 0 00205   10010          +7       STI     SYS)133
00175  0074 00 4 00266   10010          +8       TSX     SYS)182,4
00176  1 00066 1 00364   10010          +9       TXI     SYS)244,1,54
00177  3 01713 0 01712   10101         +10       TXH     CP)+14,0,CP)+15
```

```
00200  0074 00 4 00010  10010          +11    TSX   IOC)8,4
00201  0 00404 0 04001  11010          +12    PZE   INPUTMASTER,,SYS)260
00202  0 00433 0 00205  11001          +13    PZE   GN)058,,SYS)283
00203  5 00017 6 01667  10001          +14    IOCTN* BL)2,,15
00204  0020 00 0 00210  10001          +15    TRA   GN)059
00205  0774 00 7 00210  10001  GN)058          AXT   *+3,7
00206  0634 00 4 00331  10001          +1     SXA   END.OF.MASTERS,4
00207  0020 00 0 00332  10001          +2     TRA   END.OF.MASTERS+1
00210  4500 00 0 01714  10001  GN)059          CAL   CP)+16
00211  0320 00 0 00047  10001          +1     ANS   CURRENT.DEPT
00212  0535 00 1 01667  10001          +2     LAC   BL)2,1
00213  7 00000 1 00446  10010          +3     TXL   SYS)294,1,0
00214  4500 00 1 00000  10000          +4     CAL   1)DEPARTMENT,1
00215  0771 00 0 00022  10000          +5     ARS   18
00216  4320 00 0 01715  10001          +6     ANA   CP)+17
00217  4602 00 0 00047  10001          +7     ORS   CURRENT.DEPT
00220  0020 00 0 00232  10001          +8     TRA   GET.DETAIL
00221  3 01717 0 01716  10101  GET.MASTER      TXH   CP)+18,0,CP)+19
00222  0074 00 4 00010  10010          +1     TSX   IOC)8,4
00223  0 00404 0 04001  11010          +2     PZE   INPUTMASTER,,SYS)260
00224  0 00433 0 00227  11001          +3     PZE   GN)060,,SYS)283
00225  5 00017 6 01667  10001          +4     IOCTN* BL)2,,15
00226  0020 00 0 00232  10001          +5     TRA   GN)061
00227  0774 00 7 00232  10001  GN)060          AXT   *+3,7
00230  0634 00 4 00331  10001          +1     SXA   END.OF.MASTERS,4
00231  0020 00 0 00332  10001          +2     TRA   END.OF.MASTERS+1
00232                           GN)061
00232  3 01717 0 01720  10101  GET.DETAIL      TXH   CP)+20,0,CP)+19
00233  0074 00 4 00010  10010          +1     TSX   IOC)8,4
00234  0 00404 0 04003  11010          +2     PZE   DETAILFILE,,SYS)260
00235  0 00433 0 00240  11001          +3     PZE   GN)062,,SYS)283
00236  5 00003 6 01670  10001          +4     IOCTN* BL)3,,3
00237  0020 00 0 00241  10001          +5     TRA   GN)063
00240  0020 00 0 00351  10001  GN)062          TRA   END.OF.DETAILS
00241                           GN)063
00241  0535 00 1 01670  10001  COMPARE.EMPLOYEE.NUMBERS
                                        LAC   BL)3,1
00242  7 00000 1 00446  10010          +1     TXL   SYS)294,1,0
00243  4500 00 1 00000  10000          +2     CAL   2)EMPLOYEE.NUMBER,1
00244  0535 00 2 01667  10001          +3     LAC   BL)2,2
00245  7 00000 2 00446  10010          +4     TXL   SYS)294,2,0
00246  4340 00 2 00000  10000          +5     LAS   1)EMPLOYEE.NUMBER,2
00247  0020 00 0 00251  10001          +6     TRA   *+2
00250  0020 00 0 00527  10001          +7     TRA   CHECK.NEW.DEPT
00251  4500 00 1 00000  10000          +8     CAL   2)EMPLOYEE.NUMBER,1
00252  4340 00 2 00000  10000          +9     LAS   1)EMPLOYEE.NUMBER,2
00253  0020 00 0 00256  10001          +10    TRA   *+3
00254  0020 00 0 00256  10001          +11    TRA   *+2
00255  0020 00 0 00310  10001          +12    TRA   LOW.DETAIL
00256  4500 00 0 01721  10001  HIGH.DETAIL     CAL   CP)+21
00257  0320 00 0 00076  10001          +1     ANS   ERRORTYPE
00260  0760 00 0 00006  10000          +2     COM
00261  4320 00 0 01676  10001          +3     ANA   CP)+2
00262  4602 00 0 00076  10001          +4     ORS   ERRORTYPE
00263  0441 00 0 01746  10001          +5     LDI   CP)+42
```

```
00264  0604 00 0 00205  10010          +6      STI    SYS)133
00265  4500 00 0 01667  10001          +7      CAL    BL)2
00266  0361 00 0 01747  10001          +8      ACL    CP)+43
00267  0602 00 0 00204  10010          +9      SLW    SYS)132
00270  0074 00 4 00266  10010         +10      TSX    SYS)182,4
00271  1 00025 1 00360  10010         +11      TXI    SYS)240,1,21
00272  1 00002 1 00361  10010         +12      TXI    SYS)241,1,2
00273  0074 00 4 00011  10010         +13      TSX    IOC)9,4
00274  0 00000 0 04007  10010         +14      PZE    ERRORFILE,,0
00275  7 00004 0 00076  10001         +15      IOST   ERROROUT,,4
00276  3 01717 0 01722  10101         +16      TXH    CP)+22,0,CP)+19
00277  0074 00 4 00010  10010         +17      TSX    IOC)8,4
00300  0 00404 0 04001  11010         +18      PZE    INPUTMASTER,,SYS)260
00301  0 00433 0 00304  11001         +19      PZE    GN)064,,SYS)283
00302  5 00017 6 01667  10001         +20      IOCTN* BL)2,,15
00303  0020 00 0 00307  10001         +21      TRA    GN)065
00304  0774 00 7 00307  10001  GN)064          AXT    **3,7
00305  0634 00 4 00331  10001          +1      SXA    END.OF.MASTERS,4
00306  0020 00 0 00332  10001          +2      TRA    END.OF.MASTERS+1
00307  0020 00 0 00241  10001  GN)065          TRA    COMPARE.EMPLOYEE.NUMBERS
00310  0441 00 0 01746  10001  LOW.DETAIL      LDI    CP)+42
00311  0604 00 0 00205  10010          +1      STI    SYS)133
00312  4500 00 0 01670  10001          +2      CAL    BL)3
00313  0361 00 0 01750  10001          +3      ACL    CP)+44
00314  0602 00 0 00204  10010          +4      SLW    SYS)132
00315  0074 00 4 00266  10010          +5      TSX    SYS)182,4
00316  1 00017 1 00360  10010          +6      TXI    SYS)240,1,15
00317  1 00010 1 00361  10010          +7      TXI    SYS)241,1,8
00320  4500 00 0 01721  10001          +8      CAL    CP)+21
00321  0320 00 0 00076  10001          +9      ANS    ERRORTYPE
00322  0760 00 0 00006  10000         +10      COM
00323  4320 00 0 01677  10001         +11      ANA    CP)+3
00324  4602 00 0 00076  10001         +12      ORS    ERRORTYPE
00325  0074 00 4 00011  10010         +13      TSX    IOC)9,4
00326  0 00000 0 04007  10010         +14      PZE    ERRORFILE,,0
00327  7 00004 0 00076  10001         +15      IOST   ERROROUT,,4
00330  0020 00 0 00232  10001         +16      TRA    GET.DETAIL
00331  0774 00 0 00000  10000  END.OF.MASTERS  AXT    0
00332  4500 00 0 01723  10001          +1      CAL    CP)+23
00333  0535 00 1 01670  10001          +2      LAC    BL)3,1
00334  7 00000 1 00446  10010          +3      TXL    SYS)294,1,0
00335  4340 00 1 00000  10000          +4      LAS    2)EMPLOYEE.NUMBER,1
00336  0020 00 0 00342  10001          +5      TRA    GN)066
00337  0020 00 0 00341  10001          +6      TRA    **2
00340  0020 00 0 00342  10001          +7      TRA    GN)066
00341  0020 00 0 00370  10001          +8      TRA    END.OF.RUN
00342  4500 00 0 01667  10001  GN)066          CAL    BL)2
00343  0361 00 0 01751  10001          +1      ACL    CP)+45
00344  0602 00 0 00205  10010          +2      SLW    SYS)133
00345  0074 00 4 00266  10010          +3      TSX    SYS)182,4
00346  1 00006 1 00365  10010          +4      TXI    SYS)245,1,6
00347  747474747474     10000          +5      OCT    747474747474
00350  0020 60 0 00331  10001  GN)067          TRA*   END.OF.MASTERS
00351  4500 00 0 01723  10001  END.OF.DETAILS  CAL    CP)+23
00352  0535 00 1 01667  10001          +1      LAC    BL)2,1
```

```
00353  7 00000 1 00446   10010          +2    TXL   SYS)294,1,0
00354  4340 00 1 00000   10000          +3    LAS   1)EMPLOYEE.NUMBER,1
00355  0020 00 0 00361   10001          +4    TRA   GN)068
00356  0020 00 0 00360   10001          +5    TRA   *+2
00357  0020 00 0 00361   10001          +6    TRA   GN)068
00360  0020 00 0 00370   10001          +7    TRA   END.OF.RUN
00361  4500 00 0 01670   10001  GN)068        CAL   8L)3
00362  0361 00 0 01752   10001          +1    ACL   CP)+46
00363  0602 00 0 00205   10010          +2    SLW   SYS)133
00364  0074 00 4 00266   10010          +3    TSX   SYS)182,4
00365  1 00006 1 00365   10010          +4    TXI   SYS)245,1,6
00366  747474747474      10000          +5    OCT   747474747474
00367  0020 00 0 00256   10001          +6    TRA   HIGH.DETAIL
00370                          GN)069
00370  0774 00 7 00373   10001  END.OF.RUN    AXT   *+3,7
00371  0634 00 4 01473   10001          +1    SXA   DEPARTMENT.END,4
00372  0020 00 0 01474   10001          +2    TRA   DEPARTMENT.END+1
00373  0500 00 0 00125   10001          +3    CLA   6)GROSS
00374  0074 00 4 00264   10010          +4    TSX   SYS)180,4
00375  0 00001 0 00027   10001          +5    PZE   2)GROSS,,1
00376  1 00004 1 00413   10010          +6    TXI   SYS)267,1,4
00377  000005000004      10000          +7    OCT   000005000004
00400  0774 00 1 00007   10000          +8    AXT   7,1
00401  0500 00 0 00126   10001          +9    CLA   6)WHT
00402  0074 00 4 00264   10010         +10    TSX   SYS)180,4
00403  0 00005 0 00030   10001         +11    PZE   2)WHT,,5
00404  1 00004 1 00413   10010         +12    TXI   SYS)267,1,4
00405  000005000004      10000         +13    OCT   000005000004
00406  0774 00 1 00007   10000         +14    AXT   7,1
00407  0500 00 0 00127   10001         +15    CLA   6)FICA
00410  0074 00 4 00264   10010         +16    TSX   SYS)180,4
00411  0 00003 0 00032   10001         +17    PZE   2)FICA,,3
00412  1 00004 1 00413   10010         +18    TXI   SYS)267,1,4
00413  000004000003      10000         +19    OCT   000004000003
00414  0774 00 1 00006   10000         +20    AXT   6,1
00415  0500 00 0 00130   10001         +21    CLA   5)BONDEDUCTION
00416  0074 00 4 00264   10010         +22    TSX   SYS)180,4
00417  0 00000 0 00034   10001         +23    PZE   2)BONDEDUCTION,,0
00420  1 00004 1 00413   10010         +24    TXI   SYS)267,1,4
00421  000004000003      10000         +25    OCT   000004000003
00422  0774 00 1 00006   10000         +26    AXT   6,1
00423  0500 00 0 00133   10001         +27    CLA   5)NETPAY
00424  0765 00 0 00043   10000         +28    LRS   35
00425  0221 00 0 01724   10001         +29    DVP   CP)+24
00426  0074 00 4 00264   10010         +30    TSX   SYS)180,4
00427  0 00004 0 00040   10001         +31    PZE   1)NETPAY,,4
00430  1 00004 1 00413   10010         +32    TXI   SYS)267,1,4
00431  000004000003      10000         +33    OCT   000004000003
00432  0774 00 1 00006   10000         +34    AXT   6,1
00433  0500 00 0 00124   10001         +35    CLA   5)HOURS
00434  0074 00 4 00264   10010         +36    TSX   SYS)180,4
00435  0 00005 0 00025   10001         +37    PZE   HRS,,5
00436  1 00004 1 00413   10010         +38    TXI   SYS)267,1,4
00437  000004000003      10000         +39    OCT   000004000003
00440  0774 00 1 00005   10000         +40    AXT   5,1
```

```
00441  0500 00 0 00131  10001          +41   CLA   5)INSURANCE
00442  0074 00 4 00264  10010          +42   TSX   SYS)180,4
00443  0 00003 0 00035  10001          +43   PZE   INS.PREM,,3
00444  1 00004 1 00413  10010          +44   TXI   SYS)267,1,4
00445  000004000003     10000          +45   OCT   000004000003
00446  0774 00 1 00006  10000          +46   AXT   6,1
00447  0500 00 0 00132  10001          +47   CLA   5)RETIREMENT
00450  0074 00 4 00264  10010          +48   TSX   SYS)180,4
00451  0 00000 0 00037  10001          +49   PZE   RET.PREM,,0
00452  1 00004 1 00413  10010          +50   TXI   SYS)267,1,4
00453  000004000003     10000          +51   OCT   000004000003
00454  0774 00 1 00006  10000          +52   AXT   6,1
00455  4500 00 0 01725  10001          +53   CAL   CP)+25
00456  0320 00 0 00020  10001          +54   ANS   3)DEPARTMENT
00457  0760 00 0 00006  10000          +55   COM
00460  4320 00 0 01700  10001          +56   ANA   CP)+4
00461  4602 00 0 00020  10001          +57   ORS   3)DEPARTMENT
00462  0441 00 0 01753  10001          +58   LDI   CP)+47
00463  0604 00 0 00205  10010          +59   STI   SYS)133
00464  0074 00 4 00266  10010          +60   TSX   SYS)182,4
00465  1 00017 1 00363  10010          +61   TXI   SYS)243,1,15
00466  0441 00 0 01754  10001          +62   LDI   CP)+48
00467  0604 00 0 00205  10010          +63   STI   SYS)133
00470  0074 00 4 00266  10010          +64   TSX   SYS)182,4
00471  1 00004 1 00363  10010          +65   TXI   SYS)243,1,4
00472  0441 00 0 01755  10001          +66   LDI   CP)+49
00473  0604 00 0 00205  10010          +67   STI   SYS)133
00474  0074 00 4 00266  10010          +68   TSX   SYS)182,4
00475  1 00002 1 00363  10010          +69   TXI   SYS)243,1,2
00476  0441 00 0 01756  10001          +70   LDI   CP)+50
00477  0604 00 0 00205  10010          +71   STI   SYS)133
00500  0074 00 4 00266  10010          +72   TSX   SYS)182,4
00501  1 00002 1 00363  10010          +73   TXI   SYS)243,1,2
00502  0441 00 0 01757  10001          +74   LDI   CP)+51
00503  0604 00 0 00205  10010          +75   STI   SYS)133
00504  0074 00 4 00266  10010          +76   TSX   SYS)182,4
00505  1 00002 1 00363  10010          +77   TXI   SYS)243,1,2
00506  0500 00 0 00134  10001          +78   CLA   3)BONDPURCHASES
00507  0074 00 4 00264  10010          +79   TSX   SYS)180,4
00510  0 00001 0 00042  10001          +80   PZE   2)BONDENOMINATION,,1
00511  1 00004 1 00413  10010          +81   TXI   SYS)267,1,4
00512  000005000004     10000          +82   OCT   000005000004
00513  0774 00 1 00007  10000          +83   AXT   7,1
00514  0074 00 4 00011  10010          +84   TSX   IOC)9,4
00515  0 00000 0 04005  10010          +85   PZE   PAYFILE,,0
00516  7 00024 0 00020  10001          +86   IOST  PAYRECORD,,20
00517  0074 00 4 00261  10010          +87   TSX   SYS)177,4
00520  0 00000 0 00001  10010          +88   PZE   IOC)1
00521  0074 00 4 00262  10010          +89   TSX   SYS)178,4
00522  0 01727 0 01726  10101          +90   PZE   CP)+26,,CP)+27
00523  0 01731 0 01730  10101          +91   PZE   CP)+28,,CP)+29
00524  0074 00 4 00261  10010          +92   TSX   SYS)177,4
00525  0 00000 0 00001  10010          +93   PZE   IOC)1
00526  1 00000 0 00050  10010          +94   TXI   IOC)40,0
00527  4500 00 0 00047  10001  CHECK.NEW.DEPT CAL  CURRENT.DEPT
```

```
00530  4763 00 0 00022  10000          +1    LGL   18
00531  4320 00 0 01732  10001          +2    ANA   CP)+30
00532  0602 00 0 01621  10001          +3    SLW   RS)0
00533  0535 00 1 01670  10001          +4    LAC   BL)3,1
00534  7 00000 1 00446  10010          +5    TXL   SYS)294,1,0
00535  4500 00 1 00000  10000          +6    CAL   2)DEPARTMENT,1
00536  4320 00 0 01732  10001          +7    ANA   CP)+30
00537  4340 00 0 01621  10001          +8    LAS   RS)0
00540  0020 00 0 00542  10001          +9    TRA   *+2
00541  0020 00 0 00546  10001         +10    TRA   GN)070
00542  0774 00 7 00545  10001         +11    AXT   *+3,7
00543  0634 00 4 01473  10001         +12    SXA   DEPARTMENT.END,4
00544  0020 00 0 01474  10001         +13    TRA   DEPARTMENT.END+1
00545  0020 00 0 00547  10001         +14    TRA   GN)071
00546  0020 00 0 00563  10001  GN)070        TRA   COMPUTE.PAY
00547  0441 00 0 01744  10001  GN)071        LDI   CP)+40
00550  0604 00 0 00205  10010          +1    STI   SYS)133
00551  0074 00 4 00266  10010          +2    TSX   SYS)182,4
00552  1 00066 1 00364  10010          +3    TXI   SYS)244,1,54
00553  4500 00 0 01714  10001          +4    CAL   CP)+16
00554  0320 00 0 00047  10001          +5    ANS   CURRENT.DEPT
00555  0535 00 1 01670  10001          +6    LAC   BL)3,1
00556  7 00000 1 00446  10010          +7    TXL   SYS)294,1,0
00557  4500 00 1 00000  10000          +8    CAL   2)DEPARTMENT,1
00560  0771 00 0 00022  10000          +9    ARS   18
00561  4320 00 0 01715  10001         +10    ANA   CP)+17
00562  4602 00 0 00047  10001         +11    ORS   CURRENT.DEPT
00563  0535 00 1 01670  10001  COMPUTE.PAY   LAC   BL)3,1
00564  7 00000 1 00446  10010          +1    TXL   SYS)294,1,0
00565  4500 00 1 00001  10000          +2    CAL   2)DATE,1
00566  0535 00 2 01667  10001          +3    LAC   BL)2,2
00567  7 00000 2 00446  10010          +4    TXL   SYS)294,2,0
00570  0602 00 2 00005  10000          +5    SLW   1)DATE,2
00571  4500 00 0 01670  10001          +6    CAL   BL)3
00572  0361 00 0 01760  10001          +7    ACL   CP)+52
00573  0602 00 0 00204  10010          +8    SLW   SYS)132
00574  0074 00 4 00266  10010          +9    TSX   SYS)182,4
00575  1 00003 1 00270  10010         +10    TXI   SYS)184,1,3
00576  0601 00 0 00110  10001         +11    STO   3)HOURS
00577  0441 00 0 01761  10001         +12    LDI   CP)+53
00600  0604 00 0 00205  10010         +13    STI   SYS)133
00601  4500 00 0 01670  10001         +14    CAL   BL)3
00602  0361 00 0 01760  10001         +15    ACL   CP)+52
00603  0602 00 0 00204  10010         +16    SLW   SYS)132
00604  0074 00 4 00266  10010         +17    TSX   SYS)182,4
00605  1 00004 1 00271  10010         +18    TXI   SYS)185,1,4
00606  000004000003     10000         +19    OCT   000004000003
00607  1 00002 1 00324  10010         +20    TXI   SYS)212,1,2
00610  1 00003 1 00301  10010         +21    TXI   SYS)193,1,3
00611  1 00005 1 00341  10010         +22    TXI   SYS)225,1,5
00612  0500 00 0 00110  10001         +23    CLA   3)HOURS
00613  0340 00 0 01701  10001         +24    CAS   CP)+5
00614  0020 00 0 00617  10001         +25    TRA   *+3
00615  0020 00 0 00637  10001         +26    TRA   GN)072
00616  0020 00 0 00637  10001         +27    TRA   GN)072
```

```
00617  0560 00 0 01702  10001          +28     LDQ   CP)+6
00620  0200 00 0 00110  10001          +29     MPY   3)HOURS
00621  4600 00 0 01623  10001          +30     STQ   RS)1+0
00622  0560 00 0 01703  10001          +31     LDQ   CP)+7
00623  0200 00 0 01733  10001          +32     MPY   CP)+31
00624  4600 00 0 01621  10001          +33     STQ   RS)0
00625  0500 00 0 01623  10001          +34     CLA   RS)1
00626  0402 00 0 01621  10001          +35     SUB   RS)0
00627  0131 00 0 00000  10000          +36     XCA
00630  0200 00 2 00004  10000          +37     MPY   1)RATE,2
00631  0131 00 0 00000  10000          +38     XCA
00632  0361 00 0 01734  10001          +39     ACL   CP)+32
00633  0765 00 0 00043  10000          +40     LRS   35
00634  0221 00 0 01735  10001          +41     DVP   CP)+33
00635  4600 00 0 00102  10001          +42     STQ   4)GROSS
00636  0020 00 0 00650  10001          +43     TRA   GN)073
00637  0535 00 1 01667  10001  GN)072         LAC   BL)2,1
00640  7 00000 1 00446  10010           +1     TXL   SYS)294,1,0
00641  0560 00 1 00004  10000           +2     LDQ   1)RATE,1
00642  0200 00 0 00110  10001           +3     MPY   3)HOURS
00643  0131 00 0 00000  10000           +4     XCA
00644  0361 00 0 01736  10001           +5     ACL   CP)+34
00645  0765 00 0 00043  10000           +6     LRS   35
00646  0221 00 0 01733  10001           +7     DVP   CP)+31
00647  4600 00 0 00102  10001           +8     STQ   4)GROSS
00650  0774 00 7 00653  10001  GN)073         AXT   *+3,7
00651  0634 00 4 01165  10001           +1     SXA   FICA.ROUTINE,4
00652  0020 00 0 01166  10001           +2     TRA   FICA.ROUTINE+1
00653  0774 00 7 00656  10001           +3     AXT   *+3,7
00654  0634 00 4 01220  10001           +4     SXA   WITHOLDING.TAX.ROUTINE,4
00655  0020 00 0 01221  10001           +5     TRA   WITHOLDING.TAX.ROUTINE+1
00656  0560 00 0 01674  10001           +6     LDQ   CP)+0
00657  0200 00 0 01733  10001           +7     MPY   CP)+31
00660  0131 00 0 00000  10000           +8     XCA
00661  0535 00 1 01667  10001           +9     LAC   BL)2,1
00662  7 00000 1 00446  10010          +10     TXL   SYS)294,1,0
00663  0340 00 1 00014  10000          +11     CAS   1)BONDEDUCTION,1
00664  0020 00 0 00666  10001          +12     TRA   *+2
00665  0020 00 0 00672  10001          +13     TRA   GN)074
00666  0774 00 7 00671  10001          +14     AXT   *+3,7
00667  0634 00 4 01262  10001          +15     SXA   BOND.ROUTINE,4
00670  0020 00 0 01263  10001          +16     TRA   BOND.ROUTINE+1
00671  0020 00 0 00702  10001          +17     TRA   GN)075
00672  0441 00 0 01762  10001  GN)074         LDI   CP)+54
00673  0604 00 0 00205  10010           +1     STI   SYS)133
00674  0074 00 4 00266  10010           +2     TSX   SYS)182,4
00675  1 00007 1 00363  10010           +3     TXI   SYS)243,1,7
00676  0441 00 0 01763  10001           +4     LDI   CP)+55
00677  0604 00 0 00205  10010           +5     STI   SYS)133
00700  0074 00 4 00266  10010           +6     TSX   SYS)182,4
00701  1 00010 1 00363  10010           +7     TXI   SYS)243,1,8
00702                           GN)075
01741                           GN)088         EQU   CP)+37
00702  0774 00 4 00711  10001           +1     AXT   GN)086,4
00703  0634 00 4 01404  10001           +2     SXA   SEARCH,4
```

```
00704  0500 00 0 01675  10001          +3    CLA  CP)+1
00705  0601 00 0 00111  10001          +4    STO  INDEX
00706  0500 00 0 01741  10001          +5    CLA  GN)088
00707  0601 00 0 01671  10001          +6    STO  PI)1
01405                          GN)085       EQU  SEARCH+1
00710  0020 00 0 01405  10001          +1    TRA  SEARCH+1
00711  0500 00 0 00111  10001  GN)086       CLA  INDEX
00712  0400 00 0 01675  10001          +1    ADD  CP)+1
00713  0601 00 0 00111  10001          +2    STO  INDEX
00714  0500 00 0 01711  10001          +3    CLA  CP)+13
00715  0400 00 0 01671  10001          +4    ADD  PI)1
00716  0601 00 0 01671  10001          +5    STO  PI)1
00717  0500 00 0 01704  10001          +6    CLA  CP)+8
00720  0402 00 0 00111  10001          +7    SUB  INDEX
00721  0120 00 0 01405  10001          +8    TPL  GN)085
00722  0500 00 0 00102  10001  NET           CLA  4)GROSS
00723  0402 00 0 00105  10001          +1    SUB  4)FICA
00724  0402 00 0 00106  10001          +2    SUB  4)WHT
00725  0402 00 0 00103  10001          +3    SUB  3)RETIREMENT
00726  0402 00 0 00104  10001          +4    SUB  3)INSURANCE
00727  0535 00 1 01667  10001          +5    LAC  BL)2,1
00730  7 00000 1 00446  10010          +6    TXL  SYS)294,1,0
00731  0402 00 1 00014  10000          +7    SUB  1)BONDEDUCTION,1
00732  0601 00 0 00107  10001          +8    STO  3)NETPAY
00733  0500 00 0 00114  10001          +9    CLA  5)GROSS
00734  0400 00 0 00102  10001         +10    ADD  4)GROSS
00735  0601 00 0 00114  10001         +11    STO  5)GROSS
00736  0500 00 0 00121  10001         +12    CLA  4)RETIREMENT
00737  0400 00 0 00103  10001         +13    ADD  3)RETIREMENT
00740  0601 00 0 00121  10001         +14    STO  4)RETIREMENT
00741  0500 00 0 00120  10001         +15    CLA  4)INSURANCE
00742  0400 00 0 00104  10001         +16    ADD  3)INSURANCE
00743  0601 00 0 00120  10001         +17    STO  4)INSURANCE
00744  0500 00 0 00116  10001         +18    CLA  5)FICA
00745  0400 00 0 00105  10001         +19    ADD  4)FICA
00746  0601 00 0 00116  10001         +20    STO  5)FICA
00747  0500 00 0 00115  10001         +21    CLA  5)WHT
00750  0400 00 0 00106  10001         +22    ADD  4)WHT
00751  0601 00 0 00115  10001         +23    STO  5)WHT
00752  0500 00 0 00122  10001         +24    CLA  4)NETPAY
00753  0400 00 0 00107  10001         +25    ADD  3)NETPAY
00754  0601 00 0 00122  10001         +26    STO  4)NETPAY
00755  0500 00 0 00113  10001         +27    CLA  4)HOURS
00756  0400 00 0 00110  10001         +28    ADD  3)HOURS
00757  0601 00 0 00113  10001         +29    STO  4)HOURS
00760  0500 00 1 00007  10000         +30    CLA  1)GROSS,1
00761  0400 00 0 00102  10001         +31    ADD  4)GROSS
00762  0601 00 1 00007  10000         +32    STO  1)GROSS,1
00763  0500 00 1 00010  10000         +33    CLA  1)RETIREMENT,1
00764  0400 00 0 00103  10001         +34    ADD  3)RETIREMENT
00765  0601 00 1 00010  10000         +35    STO  1)RETIREMENT,1
00766  0500 00 1 00011  10000         +36    CLA  1)INSURANCE,1
00767  0400 00 0 00104  10001         +37    ADD  3)INSURANCE
00770  0601 00 1 00011  10000         +38    STO  1)INSURANCE,1
00771  4500 00 0 00006  10001         +39    CAL  3)EMPLOYEE.NUMBER
```

```
00772  0535 00 2 01670  10001   +40   LAC   BL)3,2
00773  7 00000 2 00446  10010   +41   TXL   SYS)294,2,0
00774  0560 00 2 00000  10000   +42   LDQ   2)EMPLOYEE.NUMBER,2
00775  4773 00 0 00000  10000   +43   RQL   0
00776  0771 00 0 00022  10000   +44   ARS   18
00777  4763 00 0 00022  10000   +45   LGL   18
01000  0602 00 0 00006  10001   +46   SLW   3)EMPLOYEE.NUMBER
01001  4763 00 0 00022  10000   +47   LGL   18
01002  0560 00 0 00007  10001   +48   LDQ   3)EMPLOYEE.NUMBER+1
01003  4773 00 0 00022  10000   +49   RQL   18
01004  4763 00 0 00022  10000   +50   LGL   18
01005  0602 00 0 00007  10001   +51   SLW   3)EMPLOYEE.NUMBER+1
01006  4500 00 0 01725  10001   +52   CAL   CP)+25
01007  0320 00 0 00024  10001   +53   ANS   4)MONTH
01010  0760 00 0 00006  10000   +54   COM
01011  4320 00 2 00001  10000   +55   ANA   2)MONTH,2
01012  4602 00 0 00024  10001   +56   ORS   4)MONTH
01013  4500 00 0 01714  10001   +57   CAL   CP)+16
01014  0320 00 0 00024  10001   +58   ANS   4)DAY
01015  4500 00 2 00001  10000   +59   CAL   2)DAY,2
01016  0771 00 0 00006  10000   +60   ARS   6
01017  4320 00 0 01715  10001   +61   ANA   CP)+17
01020  4602 00 0 00024  10001   +62   ORS   4)DAY
01021  4500 00 0 00025  10001   +63   CAL   4)YEAR
01022  4765 00 0 00030  10000   +64   LGR   24
01023  4500 00 2 00001  10000   +65   CAL   2)YEAR,2
01024  4763 00 0 00030  10000   +66   LGL   24
01025  0602 00 0 00025  10001   +67   SLW   4)YEAR
01026  4500 00 0 01725  10001   +68   CAL   CP)+25
01027  0320 00 0 00020  10001   +69   ANS   3)DEPARTMENT
01030  0760 00 0 00006  10000   +70   COM
01031  4320 00 2 00000  10000   +71   ANA   2)DEPARTMENT,2
01032  4602 00 0 00020  10001   +72   ORS   3)DEPARTMENT
01033  4500 00 0 00020  10001   +73   CAL   3)EMPLOYEE
01034  0560 00 2 00000  10000   +74   LDQ   2)EMPLOYEE,2
01035  4773 00 0 00014  10000   +75   RQL   12
01036  0771 00 0 00022  10000   +76   ARS   18
01037  4763 00 0 00022  10000   +77   LGL   18
01040  0602 00 0 00020  10001   +78   SLW   3)EMPLOYEE
01041  4763 00 0 00006  10000   +79   LGL   6
01042  0560 00 0 00021  10001   +80   LDQ   3)EMPLOYEE+1
01043  4773 00 0 00006  10000   +81   RQL   6
01044  4763 00 0 00036  10000   +82   LGL   30
01045  0602 00 0 00021  10001   +83   SLW   3)EMPLOYEE+1
01046  4500 00 0 01737  10001   +84   CAL   CP)+35
01047  0320 00 0 00000  10001   +85   ANS   3)MONTH
01050  4500 00 2 00001  10000   +86   CAL   2)MONTH,2
01051  0771 00 0 00006  10000   +87   ARS   6
01052  4320 00 0 01740  10001   +88   ANA   CP)+36
01053  4602 00 0 00000  10001   +89   ORS   3)MONTH
01054  0560 00 0 00000  10001   +90   LDQ   3)DAY
01055  4763 00 0 00030  10000   +91   LGL   24
01056  0560 00 2 00001  10000   +92   LDQ   2)DAY,2
01057  4773 00 0 00014  10000   +93   RQL   12
01060  4765 00 0 00030  10000   +94   LGR   24
```

```
01061  4600 00 0 00000  10001  +95  STQ  3)DAY
01062  4500 00 0 01737  10001  +96  CAL  CP)+35
01063  0320 00 0 00001  10001  +97  ANS  3)YEAR
01064  4500 00 2 00001  10000  +98  CAL  2)YEAR,2
01065  0767 00 0 00022  10000  +99  ALS  18
01066  4320 00 0 01740  10001 +100  ANA  CP)+36
01067  4602 00 0 00001  10001 +101  ORS  3)YEAR
01070  0500 00 0 00102  10001 +102  CLA  4)GROSS
01071  0074 00 4 00264  10010 +103  TSX  SYS)180,4
01072  0 00001 0 00027  10001 +104  PZE  2)GROSS,,1
01073  1 00004 1 00413  10010 +105  TXI  SYS)267,1,4
01074  000005000004     10000 +106  OCT  000005000004
01075  0774 00 1 00007  10000 +107  AXT  7,1
01076  0500 00 0 00105  10001 +108  CLA  4)FICA
01077  0074 00 4 00264  10010 +109  TSX  SYS)180,4
01100  0 00003 0 00032  10001 +110  PZE  2)FICA,,3
01101  1 00004 1 00413  10010 +111  TXI  SYS)267,1,4
01102  000004000003     10000 +112  OCT  000004000003
01103  0774 00 1 00006  10000 +113  AXT  6,1
01104  0500 00 0 00106  10001 +114  CLA  4)WHT
01105  0074 00 4 00264  10010 +115  TSX  SYS)180,4
01106  0 00005 0 00030  10001 +116  PZE  2)WHT,,5
01107  1 00004 1 00413  10010 +117  TXI  SYS)267,1,4
01110  000005000004     10000 +118  OCT  000005000004
01111  0774 00 1 00007  10000 +119  AXT  7,1
01112  0500 00 0 00107  10001 +120  CLA  3)NETPAY
01113  0765 00 0 00043  10000 +121  LRS  35
01114  0221 00 0 01724  10001 +122  DVP  CP)+24
01115  0074 00 4 00264  10010 +123  TSX  SYS)180,4
01116  0 00004 0 00040  10001 +124  PZE  1)NETPAY,,4
01117  1 00004 1 00413  10010 +125  TXI  SYS)267,1,4
01120  000004000003     10000 +126  OCT  000004000003
01121  0774 00 1 00006  10000 +127  AXT  6,1
01122  0441 00 0 01753  10001 +128  LDI  CP)+47
01123  0604 00 0 00205  10010 +129  STI  SYS)133
01124  4500 00 0 01667  10001 +130  CAL  BL)2
01125  0361 00 0 01764  10001 +131  ACL  CP)+56
01126  0602 00 0 00204  10010 +132  SLW  SYS)132
01127  0074 00 4 00266  10010 +133  TSX  SYS)182,4
01130  1 00017 1 00357  10010 +134  TXI  SYS)239,1,15
01131  0441 00 0 01765  10001 +135  LDI  CP)+57
01132  0604 00 0 00205  10010 +136  STI  SYS)133
01133  4500 00 0 01667  10001 +137  CAL  BL)2
01134  0361 00 0 01764  10001 +138  ACL  CP)+56
01135  0602 00 0 00204  10010 +139  SLW  SYS)132
01136  0074 00 4 00266  10010 +140  TSX  SYS)182,4
01137  1 00017 1 00360  10010 +141  TXI  SYS)240,1,15
01140  1 00003 1 00361  10010 +142  TXI  SYS)241,1,3
01141  0500 00 0 00107  10001 +143  CLA  3)NETPAY
01142  0765 00 0 00043  10000 +144  LRS  35
01143  0221 00 0 01724  10001 +145  DVP  CP)+24
01144  0074 00 4 00264  10010 +146  TSX  SYS)180,4
01145  0 00001 0 00016  10001 +147  PZE  AMOUNT,,1
01146  1 00014 1 00413  10010 +148  TXI  SYS)267,1,12
01147  000004000003     10000 +149  OCT  000004000003
```

```
01150  0774 00 1 00006  10000          +150    AXT   6,1
01151  0074 00 4 00011  10010          +151    TSX   IOC)9,4
01152  0 00000 0 04004  10010          +152    PZE   CHECKFILE,,0
01153  7 00020 0 00000  10001          +153    IOST  CHECK,,16
01154  0074 00 4 00011  10010          +154    TSX   IOC)9,4
01155  0 00000 0 04005  10010          +155    PZE   PAYFILE,,0
01156  7 00024 0 00020  10001          +156    IOST  PAYRECORD,,20
01157  0534 00 4 01667  10001          +157    LXA   BL)2,4
01160  0634 00 4 01163  10001          +158    SXA   GN)089,4
01161  0074 00 4 00011  10010          +159    TSX   IOC)9,4
01162  0 00000 0 04002  10010          +160    PZE   OUTPUTMASTER,,0
01163  7 00017 0 00000  10001  GN)089          IOST  MASTER,,15
01164  0020 00 0 00221  10001            +1    TRA   GET.MASTER
01165  0774 00 0 00000  10000  FICA.ROUTINE    AXT   0
01166  0560 00 0 01705  10001            +1    LDQ   CP)+9
01167  0200 00 0 00102  10001            +2    MPY   4)GROSS
01170  0131 00 0 00000  10000            +3    XCA
01171  0361 00 0 01736  10001            +4    ACL   CP)+34
01172  0765 00 0 00043  10000            +5    LRS   35
01173  0221 00 0 01733  10001            +6    DVP   CP)+31
01174  4600 00 0 00105  10001            +7    STQ   4)FICA
01175  0535 00 1 01667  10001            +8    LAC   BL)2,1
01176  7 00000 1 00446  10010            +9    TXL   SYS)294,1,0
01177  0500 00 1 00012  10000           +10    CLA   1)FICA,1
01200  0400 00 0 00105  10001           +11    ADD   4)FICA
01201  0601 00 1 00012  10000           +12    STO   1)FICA,1
01202  0500 00 1 00012  10000           +13    CLA   1)FICA,1
01203  0340 00 0 01706  10001           +14    CAS   CP)+10
01204  0020 00 0 01207  10001           +15    TRA   **3
01205  0020 00 0 01217  10001           +16    TRA   GN)076
01206  0020 00 0 01217  10001           +17    TRA   GN)076
01207  0500 00 1 00012  10000           +18    CLA   1)FICA,1
01210  0402 00 0 01706  10001           +19    SUB   CP)+10
01211  0601 00 0 01627  10001           +20    STO   1.RS)0
01212  0500 00 0 00105  10001           +21    CLA   4)FICA
01213  0402 00 0 01627  10001           +22    SUB   1.RS)0
01214  0601 00 0 00105  10001           +23    STO   4)FICA
01215  0500 00 0 01706  10001           +24    CLA   CP)+10
01216  0601 00 1 00012  10000           +25    STO   1)FICA,1
01217                          GN)076
C1217  0020 60 0 01165  10001  GN)077          TRA*  FICA.ROUTINE
C1220  0774 00 0 00000  10000  WITHOLDING.TAX.ROUTINE
                                               AXT   0
01221  0560 00 0 01710  10001            +1    LDQ   CP)+12
01222  0535 00 1 01667  10001            +2    LAC   BL)2,1
01223  7 00000 1 00446  10010            +3    TXL   SYS)294,1,0
01224  0200 00 1 00006  10000            +4    MPY   EXEMPTIONS,1
01225  0200 00 0 01733  10001            +5    MPY   CP)+31
01226  4600 00 0 01633  10001            +6    STQ   2.RS)0
01227  0500 00 0 00102  10001            +7    CLA   4)GROSS
01230  0402 00 0 01633  10001            +8    SUB   2.RS)0
01231  0131 00 0 00000  10000            +9    XCA
01232  0200 00 0 01707  10001           +10    MPY   CP)+11
01233  0131 00 0 00000  10000           +11    XCA
01234  0361 00 0 01736  10001           +12    ACL   CP)+34
```

```
01235  0765 00 0 00043  10000          +13     LRS   35
01236  0221 00 0 01733  10001          +14     DVP   CP)+31
01237  4600 00 0 00106  10001          +15     STQ   4)WHT
01240  0057 00 777777   10000          +16     RIR   777777
01241  0560 00 0 01674  10001          +17     LDQ   CP)+0
01242  0200 00 0 01733  10001          +18     MPY   CP)+31
01243  0131 00 0 00000  10000          +19     XCA
01244  0340 00 0 00106  10001          +20     CAS   4)WHT
01245  0020 00 0 01250  10001          +21     TRA   *+3
01246  0020 00 0 01250  10001          +22     TRA   *+2
01247  0055 00 000001   10000          +23     SIR   000001
01250  0754 00 0 00000  10000          +24     PXA   0,0
01251  0054 00 000001   10000          +25     RFT   000001
01252  0500 00 0 01675  10001          +26     CLA   CP)+1
01253  0131 00 0 00000  10000          +27     XCA
01254  0200 00 0 00106  10001          +28     MPY   4)WHT
01255  4600 00 0 00106  10001          +29     STQ   4)WHT
01256  0500 00 1 00013  10000          +30     CLA   1)WHT,1
01257  0400 00 0 00106  10001          +31     ADD   4)WHT
01260  0601 00 1 00013  10000          +32     STO   1)WHT,1
01261  0020 60 0 01220  10001  GN)078         TRA*  WITHOLDING.TAX.ROUTINE
01262  0774 00 0 00000  10000  BOND.ROUTINE  AXT   0
01263  0535 00 1 01667  10001          +1      LAC   BL)2,1
01264  7 00000 1 00446  10010          +2      TXL   SYS)294,1,0
01265  0500 00 1 00015  10000          +3      CLA   BONDACCUMULATION,1
01266  0400 00 1 00014  10000          +4      ADD   1)BONDEDUCTION,1
01267  0601 00 1 00015  10000          +5      STO   BONDACCUMULATION,1
01270  0500 00 0 00117  10001          +6      CLA   4)BONDEDUCTION
01271  0400 00 1 00014  10000          +7      ADD   1)BONDEDUCTION,1
01272  0601 00 0 00117  10001          +8      STO   4)BONDEDUCTION
01273  0500 00 1 00014  10000          +9      CLA   1)BONDEDUCTION,1
01274  0074 00 4 00264  10010          +10     TSX   SYS)180,4
01275  0 00000 0 00034  10001          +11     PZE   2)BONDEDUCTION,,0
01276  1 00004 1 00413  10010          +12     TXI   SYS)267,1,4
01277  000004000003     10000          +13     OCT   000004000003
01300  0774 00 1 00006  10000          +14     AXT   6,1
01301  0535 00 1 01667  10001          +15     LAC   BL)2,1
01302  7 00000 1 00446  10010          +16     TXL   SYS)294,1,0
01303  0500 00 1 00016  10000          +17     CLA   1)BONDENOMINATION,1
01304  0340 00 1 00015  10000          +18     CAS   BONDACCUMULATION,1
01305  0020 00 0 01313  10001          +19     TRA   GN)079
01306  0020 00 0 01307  10001          +20     TRA   *+1
01307  0500 00 1 00015  10000          +21     CLA   BONDACCUMULATION,1
01310  0402 00 1 00016  10000          +22     SUB   1)BONDENOMINATION,1
01311  0601 00 1 00015  10000          +23     STO   BONDACCUMULATION,1
01312  0020 00 0 01320  10001          +24     TRA   GN)080
01313  0441 00 0 01763  10001  GN)079         LDI   CP)+55
01314  0604 00 0 00205  10010          +1      STI   SYS)133
01315  0074 00 4 00266  10010          +2      TSX   SYS)182,4
01316  1 00010 1 00363  10010          +3      TXI   SYS)243,1,8
01317  0020 00 0 01403  10001          +4      TRA   BOND.END
01320  0535 00 1 01667  10001  GN)080         LAC   BL)2,1
01321  7 00000 1 00446  10010          +1      TXL   SYS)294,1,0
01322  4500 00 1 00005  10000          +2      CAL   1)DATE,1
01323  0602 00 0 00075  10001          +3      SLW   5)DATE
```

```
01324  0500 00 1 00016   10000          +4    CLA   1)BONDENOMINATION,1
01325  0C74 00 4 00264   10010          +5    TSX   SYS)180,4
01326  0 00000 0 00074   10001          +6    PZE   3)BONDENOMINATION,,0
01327  0020 00 0 00413   10010          +7    TRA   SYS)267,0,0
01330  000003000001      10000          +8    OCT   000003000001
01331  0774 00 1 00005   10000          +9    AXT   5,1
01332  4500 00 0 00070   10001         +10    CAL   5)EMPLOYEE.NUMBER
01333  0560 00 1 00000   10000         +11    LDQ   1)EMPLOYEE.NUMBER,1
01334  4773 00 0 00000   10000         +12    RQL   0
01335  0771 00 0 00036   10000         +13    ARS   30
01336  4763 00 0 00036   10000         +14    LGL   30
01337  0602 00 0 00070   10001         +15    SLW   5)EMPLOYEE.NUMBER
01340  4763 00 0 00006   10000         +16    LGL   6
01341  0560 00 0 00071   10001         +17    LDQ   5)EMPLOYEE.NUMBER+1
01342  4773 00 0 00006   10000         +18    RQL   6
01343  4763 00 0 00036   10000         +19    LGL   30
01344  0602 00 0 00071   10001         +20    SLW   5)EMPLOYEE.NUMBER+1
01345  0441 00 0 01766   10001         +21    LDI   CP)+58
01346  0604 00 0 00205   10010         +22    STI   SYS)133
01347  4500 00 0 01667   10001         +23    CAL   BL)2
01350  0361 00 0 01764   10001         +24    ACL   CP)+56
01351  0602 00 0 00204   10010         +25    SLW   SYS)132
01352  0074 00 4 00266   10010         +26    TSX   SYS)182,4
01353  1 00017 1 00357   10010         +27    TXI   SYS)239,1,15
01354  0441 00 0 01767   10001         +28    LDI   CP)+59
01355  0604 00 0 00204   10010         +29    STI   SYS)132
01356  0074 00 4 00266   10010         +30    TSX   SYS)182,4
01357  1 00001 1 00414   10010         +31    TXI   SYS)268,1,1
01360  1 00005 1 00415   10010         +32    TXI   SYS)269,1,5
01361  1 00005 1 00423   10010         +33    TXI   SYS)275,1,5
01362  0601 00 0 01643   10001         +34    STO   3.RS)1
01363  0500 00 0 00123   10001         +35    CLA   2)BONDPURCHASES
01364  0400 00 0 01643   10001         +36    ADD   3.RS)1
01365  0601 00 0 00123   10001         +37    STO   2)BONDPURCHASES
01366  0441 00 0 01763   10001         +38    LDI   CP)+55
01367  0604 00 0 00205   10010         +39    STI   SYS)133
01370  0441 00 0 01767   10001         +40    LDI   CP)+59
01371  0604 00 0 00204   10010         +41    STI   SYS)132
01372  0074 00 4 00266   10010         +42    TSX   SYS)182,4
01373  1 00004 1 00276   10010         +43    TXI   SYS)190,1,4
01374  000005000004      10000         +44    OCT   000005000004
01375  1 00002 1 00326   10010         +45    TXI   SYS)214,1,2
01376  1 00005 1 00306   10010         +46    TXI   SYS)198,1,5
01377  1 00007 1 00342   10010         +47    TXI   SYS)226,1,7
01400  0074 00 4 00011   10010         +48    TSX   IOC)9,4
01401  0 00000 0 04006   10010         +49    PZE   BONDORDERFILE,,0
01402  7 00006 0 00070   10001         +50    IOST  BONDORDER,,6
01403  0020 60 0 01262   10001  BOND.END      TRA*  BOND.ROUTINE
01404  0774 00 0 00000   10000  SEARCH        AXT   0
01405  0535 00 1 01667   10001          +1    LAC   BL)2,1
01406  7 00000 1 00446   10010          +2    TXL   SYS)294,1,0
01407  0500 00 1 00004   10000          +3    CLA   1)RATE,1
01410  0535 00 2 01671   10001          +4    LAC   P()1,2
01411  7 00000 2 00446   10010          +5    TXL   SYS)294,2,0
01412  0340 00 2 00000   10000          +6    CAS   0,2
```

```
01413  0020 00 0 01416  10001          +7   TRA  *+3
01414  0020 00 0 01417  10001          +8   TRA  GN)081
01415  0020 00 0 01417  10001          +9   TRA  GN)081
01416  0020 00 0 01472  10001         +10   TRA  SEARCH.END
01417  0500 00 0 00111  10001  GN)081       CLA  INDEX
01420  0601 00 0 00112  10001          +1   STO  POS
01742                           GN)091      EQU  CP)+38
01743                           GN)093      EQU  CP)+39
01421  0560 00 0 00112  10001          +1   LDQ  POS
01422  0200 00 0 01711  10001          +2   MPY  CP)+13
01423  0131 00 0 00000  10000          +3   XCA
01424  0400 00 0 01742  10001          +4   ADD  GN)091
01425  0601 00 0 01673  10001          +5   STO  PI)3
01426  0560 00 0 00112  10001          +6   LDQ  POS
01427  0200 00 0 01711  10001          +7   MPY  CP)+13
01430  0131 00 0 00000  10000          +8   XCA
01431  0400 00 0 01743  10001          +9   ADD  GN)093
01432  0601 00 0 01672  10001         +10   STO  PI)2
01433  0441 00 0 01770  10001         +11   LDI  CP)+60
01434  0604 00 0 00205  10010         +12   STI  SYS)133
01435  0441 00 0 01672  10001         +13   LDI  PI)2
01436  0604 00 0 00204  10010         +14   STI  SYS)132
01437  0074 00 4 00266  10010         +15   TSX  SYS)182,4
01440  1 00004 1 00271  10010         +16   TXI  SYS)185,1,4
01441  000004000003     10000         +17   OCT  000004000003
01442  1 00003 1 00324  10010         +18   TXI  SYS)212,1,3
01443  1 00003 1 00301  10010         +19   TXI  SYS)193,1,3
01444  1 00006 1 00341  10010         +20   TXI  SYS)225,1,6
01445  0441 00 0 01672  10001         +21   LDI  PI)2
01446  0604 00 0 00204  10010         +22   STI  SYS)132
01447  0074 00 4 00266  10010         +23   TSX  SYS)182,4
01450  1 00003 1 00270  10010         +24   TXI  SYS)184,1,3
01451  0601 00 0 00104  10001         +25   STO  3)INSURANCE
01452  0441 00 0 01771  10001         +26   LDI  CP)+61
01453  0604 00 0 00205  10010         +27   STI  SYS)133
01454  0441 00 0 01673  10001         +28   LDI  PI)3
01455  0604 00 0 00204  10010         +29   STI  SYS)132
01456  0074 00 4 00266  10010         +30   TSX  SYS)182,4
01457  1 00004 1 00271  10010         +31   TXI  SYS)185,1,4
01460  000004000003     10000         +32   OCT  000004000003
01461  1 00003 1 00324  10010         +33   TXI  SYS)212,1,3
01462  1 00003 1 00301  10010         +34   TXI  SYS)193,1,3
01463  1 00006 1 00341  10010         +35   TXI  SYS)225,1,6
01464  0441 00 0 01673  10001         +36   LDI  PI)3
01465  0604 00 0 00204  10010         +37   STI  SYS)132
01466  0074 00 4 00266  10010         +38   TSX  SYS)182,4
01467  1 00003 1 00270  10010         +39   TXI  SYS)184,1,3
01470  0601 00 0 00103  10001         +40   STO  3)RETIREMENT
01471  0020 00 0 00722  10001         +41   TRA  NET
01472                           GN)082
01472  0020 60 0 01404  10001  SEARCH.END   TRA* SEARCH
01473  0774 00 0 00000  10000  DEPARTMENT.END AXT 0
01474  0500 00 0 00113  10001          +1   CLA  4)HOURS
01475  0074 00 4 00264  10010          +2   TSX  SYS)180,4
01476  0 00005 0 00051  10001          +3   PZE  2)HOURS,,5
```

```
01477  1 00004 1 00413   10010    +4    TXI   SYS)267,1,4
01500  000004000003      10000    +5    OCT   000004000003
01501  0774 00 1 00005   10000    +6    AXT   5,1
01502  0500 00 0 00114   10001    +7    CLA   5)GROSS
01503  0074 00 4 00264   10010    +8    TSX   SYS)180,4
01504  0 00001 0 00053   10001    +9    PZE   3)GROSS,,1
01505  1 00004 1 00413   10010   +10    TXI   SYS)267,1,4
01506  000005000004      10000   +11    OCT   000005000004
01507  0774 00 1 00007   10000   +12    AXT   7,1
01510  0500 00 0 00115   10001   +13    CLA   5)WHT
01511  0074 00 4 00264   10010   +14    TSX   SYS)180,4
01512  0 00005 0 00054   10001   +15    PZE   3)WHT,,5
01513  1 00004 1 00413   10010   +16    TXI   SYS)267,1,4
01514  000005000004      10000   +17    OCT   000005000004
01515  0774 00 1 00007   10000   +18    AXT   7,1
01516  0500 00 0 00116   10001   +19    CLA   5)FICA
01517  0074 00 4 00264   10010   +20    TSX   SYS)180,4
01520  0 00003 0 00056   10001   +21    PZE   3)FICA,,3
01521  1 00004 1 00413   10010   +22    TXI   SYS)267,1,4
01522  000004000003      10000   +23    OCT   000004000003
01523  0774 00 1 00006   10000   +24    AXT   6,1
01524  0500 00 0 00117   10001   +25    CLA   4)BONDEDUCTION
01525  0074 00 4 00264   10010   +26    TSX   SYS)180,4
01526  0 00000 0 00060   10001   +27    PZE   3)BONDEDUCTION,,0
01527  1 00004 1 00413   10010   +28    TXI   SYS)267,1,4
01530  000004000003      10000   +29    OCT   000004000003
01531  0774 00 1 00006   10000   +30    AXT   6,1
01532  0500 00 0 00120   10001   +31    CLA   4)INSURANCE
01533  0074 00 4 00264   10010   +32    TSX   SYS)180,4
01534  0 00003 0 00061   10001   +33    PZE   2)INSURANCE,,3
01535  1 00004 1 00413   10010   +34    TXI   SYS)267,1,4
01536  000004000003      10000   +35    OCT   000004000003
01537  0774 00 1 00006   10000   +36    AXT   6,1
01540  0500 00 0 00121   10001   +37    CLA   4)RETIREMENT
01541  0074 00 4 00264   10010   +38    TSX   SYS)180,4
01542  0 00000 0 00063   10001   +39    PZE   2)RETIREMENT,,0
01543  1 00004 1 00413   10010   +40    TXI   SYS)267,1,4
01544  000004000003      10000   +41    OCT   000004000003
01545  0774 00 1 00006   10000   +42    AXT   6,1
01546  0500 00 0 00122   10001   +43    CLA   4)NETPAY
01547  0074 00 4 00264   10010   +44    TSX   SYS)180,4
01550  0 00003 0 00064   10001   +45    PZE   2)NETPAY,,3
01551  1 00004 1 00413   10010   +46    TXI   SYS)267,1,4
01552  000005000004      10000   +47    OCT   000005000004
01553  0774 00 1 00007   10000   +48    AXT   7,1
01554  0500 00 0 00123   10001   +49    CLA   2)BONDPURCHASES
01555  0074 00 4 00264   10010   +50    TSX   SYS)180,4
01556  0 00001 0 00066   10001   +51    PZE   1)BONDPURCHASES,,1
01557  1 00004 1 00413   10010   +52    TXI   SYS)267,1,4
01560  000005000004      10000   +53    OCT   000005000004
01561  0774 00 1 00007   10000   +54    AXT   7,1
01562  0074 00 4 00011   10010   +55    TSX   IOC)9,4
01563  0 00000 0 04005   10010   +56    PZE   PAYFILE,,0
01564  7 00024 0 00044   10001   +57    IOST  DEPARTMENT.TOTAL,,20
01565  0500 00 0 00124   10001   +58    CLA   5)HOURS
```

```
01566  0400 00 0 00113  10001          +59     ADD   4)HOURS
01567  0601 00 0 00124  10001          +60     STO   5)HOURS
01570  0500 00 0 00125  10001          +61     CLA   6)GROSS
01571  0400 00 0 00114  10001          +62     ADD   5)GROSS
01572  0601 00 0 00125  10001          +63     STO   6)GROSS
01573  0500 00 0 00126  10001          +64     CLA   6)WHT
01574  0400 00 0 00115  10001          +65     ADD   5)WHT
01575  0601 00 0 00126  10001          +66     STO   6)WHT
01576  0500 00 0 00127  10001          +67     CLA   6)FICA
01577  0400 00 0 00116  10001          +68     ADD   5)FICA
01600  0601 00 0 00127  10001          +69     STO   6)FICA
01601  0500 00 0 00130  10001          +70     CLA   5)BONDEDUCTION
01602  0400 00 0 00117  10001          +71     ADD   4)BONDEDUCTION
01603  0601 00 0 00130  10001          +72     STO   5)BONDEDUCTION
01604  0500 00 0 00131  10001          +73     CLA   5)INSURANCE
01605  0400 00 0 00120  10001          +74     ADD   4)INSURANCE
01606  0601 00 0 00131  10001          +75     STO   5)INSURANCE
01607  0500 00 0 00132  10001          +76     CLA   5)RETIREMENT
01610  0400 00 0 00121  10001          +77     ADD   4)RETIREMENT
01611  0601 00 0 00132  10001          +78     STO   5)RETIREMENT
01612  0500 00 0 00133  10001          +79     CLA   5)NETPAY
01613  0400 00 0 00122  10001          +80     ADD   4)NETPAY
01614  0601 00 0 00133  10001          +81     STO   5)NETPAY
01615  0500 00 0 00134  10001          +82     CLA   3)BONDPURCHASES
01616  0400 00 0 00123  10001          +83     ADD   2)BONDPURCHASES
01617  0601 00 0 00134  10001          +84     STO   3)BONDPURCHASES
01620  0020 60 0 01473  10001  GN)083         TRA*  DEPARTMENT.END
       5 00000 0 01671  00001                USE   2
01666  5 00000 0 01666  00001                ORG   BL)1
01666  0 00000 0 00035  10010          +1      PZE   IOC)29
01667  0 00000 0 00000  10000          +2      PZE   0
01670  0 00000 0 00000  10000          +3      PZE   0
       5 00000 0 01621  00001                USE   1
01621  2 00000 0 00036  00001  RS)            BSS   30
01657  2 00000 0 00007  00001  TS)            BSS   7
01666  2 00000 0 00003  00001  BL)            BSS   3
01671  2 00000 0 00003  00001  PI)            BSS   3
01674  000000000000     10000  CP)            OCT   000000000000
01675  000000000001     10000          +1      OCT   000000000001
01676  446060606060     10000          +2      OCT   446060606060
01677  246060606060     10000          +3      OCT   246060606060
01700  276360606060     10000          +4      OCT   276360606060
01701  000000000620     10000          +5      OCT   000000000620
01702  000000000017     10000          +6      OCT   000000000017
01703  000000000024     10000          +7      OCT   000000000024
01704  000000000014     10000          +8      OCT   000000000014
01705  000000000003     10000          +9      OCT   000000000003
01706  000000034100     10000          +10     OCT   000000034100
01707  000000000022     10000          +11     OCT   000000000022
01710  000000000015     10000          +12     OCT   000000000015
01711  000000000002     10000          +13     OCT   000000000002
01712  606060011010     10000          +14     OCT   606060011010
01713  730002606060     10000          +15     OCT   730002606060
01714  777777000077     10000          +16     OCT   777777000077
01715  000000777700     10000          +17     OCT   000000777700
```

```
01716  6060600011100      10000      +18    OCT   6060600011100
01717  730000606060       10000      +19    OCT   730000606060
01720  6060600011101      10000      +20    OCT   6060600011101
01721  007777777777       10000      +21    OCT   007777777777
01722  6060600011104      10000      +22    OCT   6060600011104
01723  747474747474       10000      +23    OCT   747474747474
01724  000003641100       10000      +24    OCT   000003641100
01725  000077777777       10000      +25    OCT   000077777777
01726  6060600011111      10000      +26    OCT   6060600011111
01727  730104606060       10000      +27    OCT   730104606060
01730  606263464760       10000      +28    OCT   606263464760
01731  605164456060       10000      +29    OCT   605164456060
01732  777700000000       10000      +30    OCT   777700000000
01733  000000000144       10000      +31    OCT   000000000144
01734  000000000764       10000      +32    OCT   000000000764
01735  000000001750       10000      +33    OCT   000000001750
01736  000000000062       10000      +34    OCT   000000000062
01737  770000777777       10000      +35    OCT   770000777777
01740  007777000000       10000      +36    OCT   007777000000
01741  0 00000 0 00135    10001      +37    PZE   2)RATE+0
01742  0 00000 0 00134    10001      +38    PZE   RET.PREM-2
01743  0 00000 0 00134    10001      +39    PZE   INS.PREM-2
01744  0 00000 0 00113    10001      +40    PZE   INTERNAL.TOTALS,,0
01745  0 00000 0 00124    10001      +41    PZE   GRAND.TOTALS,,0
01746  0 00001 0 00076    10001      +42    PZE   INFO,,1
01747  0 00000 0 00000    10000      +43    PZE   1)DAT,,0
01750  0 00000 0 00000    10000      +44    PZE   DETAIL,,0
01751  0 00000 0 00000    10000      +45    PZE   1)EMPLOYEE.NUMBER,,0
01752  0 00000 0 00000    10000      +46    PZE   2)EMPLOYEE.NUMBER,,0
01753  0 00002 0 00021    10001      +47    PZE   3)NAME,,2
01754  0 00003 0 00020    10001      +48    PZE   3)EMPLOYEE,,3
01755  0 00000 0 00024    10001      +49    PZE   4)MONTH,,0
01756  0 00003 0 00024    10001      +50    PZE   4)DAY,,3
01757  0 00000 0 00025    10001      +51    PZE   4)YEAR,,0
01760  0 00000 0 00002    10000      +52    PZE   1)HOURS,,0
01761  0 00005 0 00025    10001      +53    PZE   HRS,,5
01762  0 00000 0 00034    10001      +54    PZE   2)BONDEDUCTION,,0
01763  0 00001 0 00042    10001      +55    PZE   2)BONDENOMINATION,,1
01764  0 00000 0 00001    10000      +56    PZE   1)NAME,,0
01765  0 00005 0 00007    10001      +57    PZE   2)NAME,,5
01766  0 00002 0 00071    10001      +58    PZE   4)NAME,,2
01767  0 00000 0 00074    10001      +59    PZE   3)BONDENOMINATION,,0
01770  0 00003 0 00035    10001      +60    PZE   INS.PREM,,3
01771  0 00000 0 00037    10001      +61    PZE   RET.PREM,,0
00165  500000000165       01111             START GN)000
```

THE LAST LOADER CONTROL CARD PUNCHED IS
*CTEND

REPORT OUTPUT FOR SAMPLE PROBLEM

PAYFILE REPORT

01 1010 BLUT H	10-06-61	40.0	136.00	19.80	0.00		2.00	2.00	112.20		
DEPARTMENT 01 TOTALS		40.0	136.00	19.80	0.00	0.00	2.00	2.00	112.20	0.00	
02 1021 CASPERIAN J	10-06-61	20.0	106.66	16.86	3.20		3.00	3.00	80.60		
DEPARTMENT 02 TOTALS		20.0	106.66	16.86	3.20	0.00	3.00	3.00	80.60	0.00	
03 1721 CATLETT H	10-06-61	31.0	217.00	36.72	0.00		3.00	3.00	174.28		
DEPARTMENT 03 TOTALS		31.0	217.00	36.72	0.00	0.00	3.00	3.00	174.28	0.00	
04 1723 DORR D	10-06-61	32.5	292.50	47.97	2.01		3.00	3.00	236.52		
DEPARTMENT 04 TOTALS		32.5	292.50	47.97	2.01	0.00	3.00	3.00	236.52	0.00	
06 1793 CABAN R	10-06-61	40.0	94.00	16.92	0.00		1.50	1.50	74.08		
DEPARTMENT 06 TOTALS		40.0	94.00	16.92	0.00	0.00	1.50	1.50	74.08	0.00	
07 1802 SMITH B	10-06-61	30.0	87.00	13.32	0.00		1.50	1.50	70.68		
07 1803 SOBEK W	10-06-61	40.0	120.00	16.92	0.00	4.00	2.00	2.00	95.08		
DEPARTMENT 07 TOTALS		70.0	207.00	30.24	0.00	4.00	3.50	3.50	165.76	0.00	
09 1924 REYNOLDS J	10-06-61	40.0	399.60	69.59	0.00	5.00	3.00	3.00	319.01		
09 1977 WILLIAMS P	10-06-61	37.0	369.63	59.51	0.00	10.00	3.00	3.00	294.12		
09 1978 RICHARD D	10-06-61	40.0	440.00	69.84	0.00		3.00	3.00	364.16		
09 1980 WOO J	10-06-61	39.0	468.00	81.90	0.00	17.00	3.00	3.00	363.10	37.50	
DEPARTMENT 09 TOTALS		156.0	1677.23	280.84	0.00	32.00	12.00	12.00	1340.39	37.50	
GT		389.5	2730.39	449.35	5.21	36.00	28.00	28.00	2180.63	37.50	

CHECKFILE REPORT (partial)

```
110-06-61
2WILLIAMS P
091977
$294.12
110-06-61
2RICHARD D
091978
$364.16
110-06-61
2WOO J
091980
$363.10
```

BONDORDERFILE REPORT

```
091980 WOO J 3750 100661
```

ERRORFILE REPORT

```
M011001AJAX T
M011011BLOODSOE F
M041722DICK M
M041791SMITH J
M051792JONES F
D061500100661400
M071794MANDELI A
M071801FUNGE T
D071899100661400
M081921CRAMER P
M081922BAKER B
M081923WICKIWICZ J
M091925MOORE J
M091926MOORE D
M091976WUFFE T
M091981ZUGEE W
D091983100661400
```

Appendix 90.08

Compiler Use of Environment Description Options in Generation of Loader Symbolic Control Cards.

A. The *FILE Card

The fields of Loader *FILE symbolic control cards when generated by the Compiler are filled on the basis of options exercised on Environment FILE and SPECIF cards.

Columns	Contents	Source of information
1-6	deck.name	$CMPLE
7-11	*FILE	Compiler emitted
14-15	file number	Compiler generated
17	tape mounting indicator	SPECIF - DEFER
	blank	option specified
	*	option not specified
18-21	unit1	SPECIF - UNIT1 'unit.1'
22-25	unit2	SPECIF - UNIT2 'unit.2'
27	File List Control (blank)	Compiler emitted
28	file type	FILE
	I - input	INPUT
	T - total block output	OUTPUT, SPANS
	P - partial block output	OUTPUT
29	reel control	SPECIF
	blank - single reel unlabeled	MULTI nor LABELS nor LABELN specif.
	blank - not to be searched for label	LABELS or LABELN bu no OPENF
	L - label to be searched for on labeled, open file only	OPENF and LABELS or LABELN
	M - multi-reel unlabeled file	MULTI, neither LABELS nor LABELN
30	File density	SPECIF
	H - high	HIGH
	L - low	HIGH
31	file mode	FILE
	D - BCD	BCD
	B - binary	BINARY
32	labeling conventions	SPECIF
	H - high	LABELS or LABELN and HIGH
	L - low	LABELS or LABELN and LOW
	S - same as file	LABELS or LABELN
33	block sequence numbers to be checked	SPECIF - SEQ
34	block checksums to be checked	SPECIF - CKSUMS
35	checkpoint conventions	FILE and SPECIF
	C - on checkpoint file	FILE CHECKPOINT AND SPECIF CHKS
	F - on specified file	FILE OUTPUT and SPECIF
		CHECKF and LABELS or LABELN
38-41	sequence number of first reel	SPECIF - REEL 'reel.no'
44-48	file serial number	SPECIF - SERIAL 'serial.no'
51-53	retention days	SPECIF - RETAIN days
54-72	file name	FILE - name field

B. The *SPEC Card

The fields of the Loader *SPEC symbolic control cards generated by the Compiler are filled on the basis of options exercised on Environment FILE and SPECIF cards.

Columns	Contents	Source of Information
1-6	deck.name	$CMPLE card
7-11	*SPEC	Compiler emitted
14 15	file number	Compiler generated
17-20	blocksize	FILE - BLOCKSIZE nn
22-23	activity	SPECIF - ACTIVITY nn
25	opening conventions	SPECIF
	N - without rewind	OPENW
	R or blank - with rewind	OPENW not specified
27	closing conventions	SPECIF
	N - without rewind	CLOSEW
	R or blank - with rewind	CLOSER
	U - with rewind and unload	neither CLOSEW nor CLOSER specified

PUBLICATIONS

Following is a list of IBM publications which may be of interest to the reader:

REFERENCE MANUALS

Form Number	Title
A22-6528-1	IBM 7090 Data Processing System
A22-6536	IBM 709 Data Processing System
C28-6054-2	709/7090 FORTRAN Programming System
C28-6066-3	709/7090 FORTRAN Operations
C28-6100-2	709/7090 Input/Output Control System

GENERAL INFORMATION MANUALS

D22-6508-2	IBM 709/7090 Data Processing System
F28-8043	IBM Commercial Translator
F28-8053-1	COBOL
F28-8074-1	FORTRAN

BULLETINS

G22-6505-1	IBM 7090 Data Processing System
J28-6098-1	FORTRAN Assembly Program (FAP) for the IBM 7090
J28-6186	FORTRAN Assembly Program (FAP) for the IBM 7090 Supplementary Information for the 32K Version
J28-6114-1	32K 709/7090 FORTRAN: Double-Precision and Complex Arithmetic
J28-6132	Advance Specifications: 7090 FORTRAN and FORTRAN Assembly Program (FAP)
J28-6133	32K 709/7090 FORTRAN: Source Language Debugging at Object Time
J28-6135	32K 709/7090 FORTRAN: Adding Built-In Functions
J28-6166	SHARE 7090 9PAC: Part 1 - Introduction and General Principles
J28-6167	SHARE 7090 9PAC: Part 2 - The File Processor
J28-6168	SHARE 7090 9PAC: Part 3 - The Reports Generator
J28-6173	IBM 7000/1400 Output Editing System-Preliminary Reference Manual
J28-6174	S-Program for the IBM 7090: Preliminary Specifications
J28-6184	IBM 7094 Programs and Programming Systems
J28-8072	Addenda to Commercial Translator
J28-8086	7090 Operating Systems: Basic Monitor (IBSYS)

IBM
International Business Machines Corporation
Data Processing Division, 112 East Post Road, White Plains, N. Y.

Printed in U. S. A. J28-6169-1