Computation Department

M-026

LTSS Livermore Time-Sharing System

Part III: PROBLEM PROGRAM PRODUCTION

Chapter 204: THE CHIPPEWA COMPILER

Edna Carpenter, et al.

August 31, 1968 Edition - 1

LAWRENCE
LIVERMORE
LABORATORY
University of California

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

AUTHORS: EDNA CARPENTER, STAN SOLBECK

PURPOSE: THIS CHAPTER OF 'LIVERMORE TIME SHARING SYSTEM' DESCRIBES THE 6600 CHIPPEWA COMPILER.

HISTORY: THIS CHAPTER OF 'LIVERMORE TIME SHARING SYSTEM' SUPERCEDES MOST OF VOLUME I OF CIC MANUAL-H, ENTITLED: 'THE CHIPPEWA COMPILER.'

```
******************************************************************************
*                                                                            *
*        IF YOU WISH TO BE PLACED ON A MAILING LIST FOR CORRECTIONS TO       *
*        THIS CHAPTER OF LTSS, PLEASE SEND YOUR NAME AND L-CODE TO:          *
*                                                                            *
*                                 LTSS LIST                                  *
*                        COMPUTER INFORMATION CENTER                         *
*                                   L-61                                     *
*                                                                            *
******************************************************************************
```

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

NOTATION: IT IS FREQUENTLY NECESSARY TO SET WORDS AND PHRASES OFF FROM THE BODY OF THE TEXT. UNFORTUNATELY, SINCE THE ENTIRE TEXT IS WRITTEN IN CAPITAL LETTERS, WE ARE UNABLE TO USE CASE AS A DISTINGUISHING CHARACTERISTIC. THEREFORE, VARIOUS PUNCTUATION MARKS ARE USED.

A. LANGUAGE STATEMENTS AND MESSAGE STRINGS ARE SEPARATED FROM TEXT BY QUOTATION MARKS. FOR EXAMPLE: THE SYSTEM SENDS A MESSAGE 'PROBLEM ERROR 0200' TO THE TELETYPE ...

B. TO DEFINE A WORD OR PHRASE, THE TERM IS INCLUDED BETWEEN BRACKETS < .... >. FOR EXAMPLE: A <TELETYPE> IS A MACHINE ...

C. VARIOUS LEVELS OF NAMES ARE INCLUDED IN THE FOLLOWING TABLE:

LEVEL	PUNCTUATION	EXAMPLES
SYSTEM NAME	NONE	THE FROST SYSTEM...
FILE NAME	/..../	PUBLIC FILE /LRLLIB/ IS...
PROGRAM NAME	-....-	SUBROUTINE -ALTER- WILL...
VARIABLE NAME	'....'	SET 'ABC' EQUAL TO

D. IT IS ALSO FREQUENTLY NECESSARY TO INDICATE CONTENTS AND LOCATIONS. THIS IS DONE BY THE FOLLOWING:

	PUNCTUATION	EXAMPLES
CONTENTS OF	[....]	[ABC] = 4.
LOCATION OF	(....)	(ABC) = 10472 (LRLLIB)= DISC ADDRESSES [07-1123]

THIS NOTATION ALLOWS US TO DISTINGUISH BETWEEN DIFFERENT ENTITIES THAT HAVE THE SAME NAMES. FOR EXAMPLE, 'PUBLIC FILE /OUT/ CONTAINS PROGRAM -OUT-'. PUNCTUATION IS OFTEN OMITTED IF THE TERM IS ON A LINE OF ITS OWN, IS IN A TABLE, OR IS IN A LIST.

CROSS REFERENCES:

1. CROSS REFERENCES WITHIN CHAPTER 204 ARE INDICATED BY CHAPTER.SECTION.SUBSECTION AND PAGE NUMBER: (SEE 204.7.1, PAGE 20).

2. CROSS REFERENCES TO OTHER CHAPTERS ARE INDICATED IN THE SAME WAY, BUT WITHOUT PAGE NUMBERS: (SEE 4.7.3 AND 5.8).

-TABLE OF CONTENTS-

204.1. INTRODUCTION

THE CHIPPEWA FORTRAN COMPILER OPERATES AS AN INDEPENDENT PROGRAM UNDER THE CONTROL OF THE 6600 TIME SHARING SYSTEM. THIS COMPILER WILL ACCEPT FORTRAN II, FORTRAN IV, THE CHIPPEWA ASSEMBLY LANGUAGE (CLASS), AND A SUBSET OF ASCENTF.

THE CONTROL ROUTINE FOR THE COMPILER IS IN PUBLIC FILE /CHIP/. THE COMPILER, CALLED -RUN-, AND SUPPORTING LIBRARY ROUTINES ARE CONTAINED IN THE PUBLIC FILE /CLIB/. FOR DESCRIPTIONS OF THE LIBRARY ROUTINES AVAILABLE, SEE CHAPTERS 301 THROUGH 307.

204.2. PROGRAM CONTROL INFORMATION

204.2.1. HEADER CARDS

(1) PROGRAM CARD

THE FIRST CARD OF ANY SOURCE DECK, WHOSE RESULTANT OBJECT PROGRAM IS TO BE IN EXECUTABLE FORM, MUST BE A PROGRAM CARD OF THE FOLLOWING TYPE.

```
COL.7
    PROGRAM NAME (F1, F2, ...)               [IMPLIES FORTRAN IV]
    FORTRAN IV PROGRAM NAME (F1, F2, ...)
    FORTRAN II PROGRAM NAME (F1, F2, ...)
    MACHINE PROGRAM NAME (F1, F2, ...)            [IMPLIES CLASS]
    ASCENTF PROGRAM NAME (F1, F2, ...)
```

'NAME' MUST BE SEVEN OR FEWER CHARACTERS (AND DIFFERENT FROM THE NAME OF THE FILE CONTAINING THE SOURCE DECK). THIS WILL BE THE NAME OF THE BINARY EXECUTABLE FILE.

'F1' AND 'F2' ARE ARGUMENTS ASSOCIATED WITH I/O DEVICES REQUIRED FOR THE PROGRAM AND ITS SUBROUTINES. *(1)*.

(2) SUBROUTINE OR FUNCTION CARD

SUBROUTINES AND FUNCTIONS MAY FOLLOW THE END CARD OF THE MAIN PROGRAM OR BE COMPILED SEPARATELY. THE FIRST CARD OF A SUBROUTINE OR FUNCTION MUST BE OF THE FOLLOWING FORM.

```
COL.7
    SUBROUTINE NAME (A, B, ...)
    FORTRAN II SUBROUTINE NAME (A, B, ...)
    FUNCTION NAME (A, B, ...)
    TYPE FUNCTION NAME (A, B, ...)
    MACHINE SUBROUTINE NAME (A, B, ...)
    ASCENTF SUBROUTINE NAME (A, B, ...)
```

SUBROUTINES AND FUNCTION SUBPROGRAMS ARE COMPILED IN THE SAME MODE (THAT IS, FORTRAN IV, FORTRAN II, ASCENTF, OR MACHINE LANGUAGE) AS THE MAIN CODE, UNLESS SPECIFICATION IS MADE ON THE SUBROUTINE CARD.

'NAME' MUST BE SEVEN OR LESS CHARACTERS.
'A', 'B', ... ARE SUBROUTINE ARGUMENTS.

(1) SEE SECTION 204.2.2, PAGE 6.

(3) SEGMENT CARD

THE FIRST CARD OF A LINK FOR A CHAIN JOB MUST BE OF THE FOLLOWING FORM.

```
              COL.7
                  SEGMENT NAME (F1, F2, ...)
```

(SEE CHAPTER 307 FOR MORE INFORMATION ON CHAIN JOBS.)

204.2.2. I/O DEVICE ASSIGNMENTS

THE ARGUMENTS F1, F2, ... MENTIONED UNDER THE DESCRIPTION OF THE PROGRAM CARD ASSOCIATE TAPE DESIGNATIONS (OR LOGICAL I/O UNIT NUMBERS) WITH DISC FILES, TAPES, TELETYPE, OR THE dd80C. EACH TAPE DESIGNATION (EXCEPT FOR 59 AND 100) THAT APPEARS IN THE PROGRAM MUST BE MENTIONED ON THE PROGRAM CARD, PRECEDED BY THE CHARACTERS 'TAPE', AND IS ASSIGNED A BUFFER WHICH WILL BE USED BY ITS CORRESPONDING I/O DEVICES.

(1) DISC FILES

WHEN A TAPE DESIGNATION IS TO REFER TO A DISC FILE, THE DISC FILE NAME MAY APPEAR ON THE PROGRAM CARD AND THE TAPE NUMBER IS EQUATED TO THE DISC FILE NAME.

```
EXAMPLE:        PROGRAM SAMPLE (HICCUP, TAPE1=HICCUP)
                CALL DEVICE (6HCREATE, 6HHICCUP, 50000)
                .
                .
                .
                WRITE (1, 5) A, B, C
              5 FORMAT (3E10.2)
```

204.2. PROGRAM CONTROL INFORMATION

IF THE DISC FILE NAME IS NOT DECLARED ON THE PROGRAM CARD THEN IT MUST BE ASSIGNED IN THE PROGRAM *(2)*. IF THIS OPTION IS USED, THEN THE DISC FILE MUST NEVER BE REFERRED TO BY ITS NAME BUT ONLY BY THE TAPE NUMBER.

```
EXAMPLE:         PROGRAM SAMPLE (TAPE3)
                 CALL DEVICE (6HCREATE, 6HHICCUP, 50000)
                 CALL ASSIGN (3, 6HHICCUP)
                 .
                 .
                 .
                 WRITE (3, 5) A, B, C
               5 FORMAT (3E10.2)
```

NOTE: BEFORE PROGRAM EXECUTION, INPUT DISC FILES FOR USE WITH A 'READ' STATEMENT CAN BE CREATED BY ANOTHER PROGRAM, VIA THE CARD READER, OR BY USING -NAB-.

(2) TAPES

WHEN A TAPE DESIGNATION REFERS TO A PHYSICAL TAPE, THEN THE TAPE VAULT NUMBER MUST BE DECLARED ON THE PROGRAM CARD AND THE TAPE NUMBER EQUATED TO THE VAULT NUMBER. THE EQUAL SIGN (=) MAY BE SURROUNDED BY BLANKS IF DESIRED.

```
EXAMPLE:         PROGRAM AA (ZZ999, TAPE9=ZZ999)
                 CALL DEVICE (4HTAPE, 5HZZ999)
                 READ TAPE 9, A
```

----
(2) SEE CHAPTER 303.

204.2. PROGRAM CONTROL INFORMATION

(3) TELETYPE AND dd80C

IF THE TAPE NUMBER IS 59, INFORMATION WILL BE READ FROM OR WRITTEN ON THE TELETYPE. IF THE TAPE NUMBER IS 100, INFORMATION WILL BE SENT TO THE dd80C VIA THE -CRTBCD- ROUTINE *(3)*.

WHEN THE TAPE NUMBERS 59 OR 100 APPEAR IN THE PROGRAM THEN 'TAPE59' OR 'TAPE100' MUST APPEAR ON THE PROGRAM CARD. WHEN VARIABLES ARE USED WHICH HAVE BEEN PRESET TO 59 OR 100 THE NAME 'TAPE59' OR 'TAPE100' NEED NOT APPEAR ON THE PROGRAM CARD AND THE 2001 WORD BUFFER WILL NOT BE INCLUDED WITH THE CODE. THIS BUFFER IS NOT USED FOR 'TAPE59' OR 'TAPE100'.

```
EXAMPLE:          PROGRAM EXAMPLE (TAPE100)
                  N = 59
                  WRITE (N, 4)
                4 FORMAT (18HTYPE IN INPUT DATA)
                  .
                  .
                  .
                  CALL CRTID (2HEA, 1, 1)
                  CALL SETCH (1., 64., 1, 0, 0, 0, 0)
                  WRITE (100, 5)
                5 FORMAT (15HTITLE FOR CRTID)
```

----

(3) SEE CHAPTER 304.

204.2. PROGRAM CONTROL INFORMATION

(4) READ AND PRINT STATEMENTS

THE STATEMENTS 'READ' AND 'PRINT' MAY REFER TO THE TELETYPE. IF THE STATEMENTS APPEAR IN THE PROGRAM THEN THE NAMES 'INPUT' OR 'OUTPUT' OR BOTH MUST APPEAR IN THE PROGRAM CARD.

```
EXAMPLE:          PROGRAM TEST (INPUT, OUTPUT)
                  PRINT 6
                6 FORMAT (15HTYPE INPUT LINE)
                  READ 7, A
                7 FORMAT (E10.2)
```

IF THE NAMES 'INPUT' OR 'OUTPUT' ARE EQUATED TO UNUSED TAPE NUMBERS, THEN THE STATEMENTS 'READ' OR 'PRINT' WILL REFER TO DISC FILES BY THE NAME /INPUT/ AND /OUTPUT/.

```
EXAMPLE:          PROGRAM TEST2 (INPUT, TAPE20=INPUT, OUTPUT, TAPE21=OUTPUT)
                  READ 7, A
                7 FORMAT (1E10.2)
                  PRINT 6, A
                6 FORMAT (15H THE VALUE OF A=E10.2)
```

CAUTION: THE CHIP OUTPUT FILE /OUTPUT/ *(4)* MUST BE GIVEN AWAY (SAY, BY MEANS OF UTILITY ROUTINE -OUT-) BEFORE THE FILE /OUTPUT/ IS USED FOR PROBLEM ANSWERS.

(5) BUFFER SIZE

THE BUFFER SIZE MAY BE SPECIFIED ON THE -CHIP- INPUT LINE, *(5)* ASSIGNED BY THE COMPILER (2001 OCTAL WORDS), OR ASSIGNED ON THE PROGRAM CARD BY EQUATING THE DISC FILE NAME OR THE TAPE VAULT NUMBER TO THE SIZE. THE MINIMUM BUFFER SIZE FOR NON-BUFFERED I/O IS 1001. THE MINIMUM FOR BUFFERED I/O IS ZERO. IF THE BUFFER SIZE IS SPECIFIED BOTH ON THE PROGRAM CARD AND ON THE CHIP INPUT LINE THEN THE VALUE ON THE PROGRAM CARD TAKES PREFERENCE.

```
EXAMPLES:       PROGRAM TEST1 (FILE1=1001, TAPE1=FILE1)
                     TAPE 1 WILL HAVE A BUFFER SIZE OF 1001

                PROGRAM TEST2 (HICCUP=1, TAPE1=HICCUP, TAPE2)
                     TAPE 1 WILL HAVE A BUFFER SIZE OF 1 AND MAY BE USED ONLY WITH BUFFERED
                     I/O.  TAPE 2 WILL HAVE A BUFFER SIZE OF 2001 UNLESS A SMALLER VALUE IS
                     SPECIFIED ON THE CHIP INPUT LINE.
```

----

(4) SEE SECOND ARGUMENT ON -CHIP- INPUT LINE, SECTION 204.3.1, PAGE 10.

(5) SEE SIXTH ARGUMENT ON -CHIP- INPUT LINE, SECTION 204.3.1, PAGE 10.

204.3. COMPILATION FROM A TELETYPE

204.3.1. GENERAL OPERATING INSTRUCTIONS

TO COMPILE A CODE FROM TELETYPE, THE USER TYPES:

```
CHIP A B C D E F G H / T Y
```

WHERE 'A', 'B', 'C', 'D', 'E', 'F', 'G', AND 'H' ARE ARGUMENTS USED BY THE COMPILER. THESE ARGUMENTS MAY BE SEPARATED BY BLANKS OR COMMAS, BUT NOT BOTH. DROPOUT IS ALLOWED. COMMA-SEPARATION MUST BE USED FOR INTERNAL DROPOUT. FOR EXAMPLE:

```
CHIP A B C D E / T Y
```

AND

```
CHIP A B,,D,,,,H / T Y
```

ARE LEGITIMATE EXECUTE LINES FOR -CHIP-, BUT THE FOLLOWING IS NOT:

```
CHIP A B  D   H / T Y
```

THE ARGUMENTS ARE AS FOLLOWS:

A IS THE NAME OF THE ASCII INPUT FILE TO BE COMPILED. IF 'A' IS OMITTED, THE NAME IS ASSUMED TO BE 'INPUT'.

THE INPUT FILE MAY BE CREATED THROUGH THE CARD READER OR BY USING -NAB-. IF THE CARD READER IS USED, THE INPUT DECK MUST BE PRECEDED BY AN ID CARD OF THE FORM:

```
ID 777777 FILNAME
```

WHERE:

COLS. 1-2 CONTAIN THE LETTERS 'ID'.
COL. 3 IS BLANK .
COLS. 4-9 CONTAIN A SIX DIGIT USER NUMBER.
COL. 10 IS BLANK.
COLS. 11-17 CONTAIN A 1-7 CHARACTER ALPHANUMERIC INPUT FILE NAME.
COLS. 18-80 ARE BLANK.

THE INPUT DECK MAY CONTAIN A FORTRAN, ASCENTF OR MACHINE LANGUAGE (CLASS) MAIN PROGRAM WITH SUBROUTINES AND FUNCTIONS OR ANY COMBINATION THEREOF. NO DATA OR BINARY CARDS ARE ALLOWED AS PART OF THE INPUT DECK.

B IS THE FILE NAME FOR THE PRINTABLE OUTPUT. IF 'B' IS OMITTED, THE NAME IS ASSUMED TO BE 'OUTPUT'. TO SEE THIS LISTING ONE MAY USE -OUT- OR -NAB-.

IF 'B' IS 'HSP' OR 'OLP', THE FILE WILL BE WRITTEN ON THE HIGH SPEED (RADIATION) PRINTER TAPE OR PRINTED ON THE ON-LINE PRINTER, RESPECTIVELY. IF THE FIRST CARD IN THE INPUT FILE HAS THE CHARACTERS '*ID' IN COLUMNS 1-3, THE PRINTER IDENTIFICATION WILL BE TAKEN FROM COLUMNS 47-72 OF THIS '*ID' CARD. OTHERWISE, THE COMPILER WILL ASK FOR THE 'ID LINE' FROM THE TELETYPE. THE LAST SIX CHARACTERS SHOULD BE 'BOXNNN', WHERE 'NNN' IS THE USER'S OUTPUT BOX NUMBER.

C IS A LETTER OR NUMBER USED TO SPECIFY CERTAIN OPTIONS AT COMPILE TIME. IF 'C' IS OMITTED, IT IS ASSUMED TO BE 'S'.

IF 'C' IS 'L', A COMPLETE OCTAL LIST OF ALL THE BINARY INSTRUCTIONS IS INCLUDED WITH THE PROGRAM IN THE OUTPUT LISTING (LONG LIST OPTION).

IF 'C' IS 'S', A SHORT LIST OF THE PROGRAM IS OBTAINED WITH THE OCTAL ADDRESS OF EACH FORTRAN STATEMENT. IN EITHER CASE ('L' OR 'S'), A MEMORY MAP AND ERROR COMMENTS ARE LISTED.

IF 'C' IS 'I', THE INCOMPLETE MODE OPTION IS GIVEN. THE I-MODE MAY BE USED WHEN IT IS DESIRED TO COMPILE A GROUP OF SUBROUTINES AND/OR FUNCTIONS AND OBTAIN A BINARY FILE FOR EACH. EACH FILE CONSISTS OF BINARY INSTRUCTIONS PLUS RELOCATION INFORMATION (THE FILENAME IS TAKEN FROM EACH HEADER CARD AND IS PRECEDED BY AN ASTERISK).

ANY SUBROUTINE OR FUNCTION, WHEN COMPILED WITHOUT A MAIN CODE, OR GROUPS OF SUCH WHICH CALL OTHER SUBROUTINES OR FUNCTIONS OR CONTAIN LABELED COMMON, MUST BE COMPILED IN THE I-MODE. THE I-MODE IS MOST GENERALLY USED FOR FILES WHICH ARE TO BE INSERTED IN A PRIVATE LIBRARY. *(6)*

IF 'C' IS '4', THE 400 FROST WORD OPTION IS GIVEN *(7)*. THIS IS TO BE USED ONLY WHEN COMPILING SEGMENTS. SEGMENTS ARE NORMALLY COMPILED WITHOUT 400 FROST 'MINUS WORDS' (AS LINKS IN A CHAIN JOB). IF THE '4' IS NOT USED, THE SEGMENT WILL NOT HAVE THE ADDITIONAL 400 WORDS. IF THE '4' IS USED, 400 MINUS WORDS ARE ADDED TO THE BINARY FILE. SEGMENTS COMPILED IN THIS MANNER MAY BE EXECUTED AS PROGRAMS.

D IS THE OCTAL SIZE FOR THE COMPILER (CONTAINED IN FILE /+RUN/ AT COMPILE TIME). IF 'D' IS OMITTED, IT IS ASSIGNED A SIZE OF 46,000 (OCTAL), A GOOD SIZE FOR SMALL CODES.

----

(6) SEE SECTION 204.3.2, PAGE 15, FOR MORE DETAILS.

(7) SEE CHAPTER 2 FOR A DESCRIPTION OF THESE 400 'MINUS WORDS'.

204.3. COMPILATION FROM A TELETYPE

THE COMPILER LENGTH MUST BE LARGE ENOUGH, AT COMPILE TIME, TO ACCOMMODATE THE COMPILER, ALL BINARY INSTRUCTIONS GENERATED BY THE COMPILER, AND ALL LABELED COMMON BLOCKS. AT LOAD TIME, IT MUST ACCOMMODATE THE LOADER AND THE ENTIRE BINARY FILE (EXCLUDING BLANK AND NUMBERED COMMON AND BUFFERS). ONE SIZE IS USED FOR BOTH SITUATIONS.

TO ILLUSTRATE THIS, ASSUME THE SIZE OF AN OBJECT PROGRAM IS 45,000 WORDS (EXCLUDING BLANK AND NUMBERED COMMON AND BUFFERS). ALSO, ASSUME THAT THE PART TO BE LOADED FROM THE LIBRARY AT LOAD TIME IS 25,000 WORDS. THE COMPILER SIZE AT COMPILE TIME = 40,000 WORDS (COMPILER EXCLUDING TABLES) + 20,000 WORDS (BINARY CODE GENERATED BY THE COMPILER INCLUDING LABELED COMMON) = 60,000 WORDS. THE SIZE REQUIRED AT LOAD TIME = 10,000 WORDS (LOADER) + 45,000 WORDS (ENTIRE BINARY FILE, EXCLUDING BLANK AND NUMBERED COMMON AND BUFFERS) = 55,000 WORDS. IN THIS CASE 'D' SHOULD BE APPROXIMATELY 65,000 WORDS (GREATER THAN THE SIZE REQUIRED AT COMPILE OR LOAD TIME).

E IS THE OCTAL LENGTH OF THE OBJECT PROGRAM FILE. IF 'E' IS OMITTED, IT WILL BE THE SAME AS 'D'.

THE FILE NAME GIVEN TO THE BINARY CODE IS TAKEN FROM THE PROGRAM OR SEGMENT CARD *(8)*. FILE NAMES GENERATED FROM I-MODE COMPILATIONS ARE DESCRIBED UNDER C, ABOVE.

THE FILE LENGTH IS EQUAL TO THE LENGTH OF THE COMPILED CODE, PLUS THE LENGTH OF THE NEEDED LIBRARY ROUTINES, PLUS THE LENGTH OF BLANK AND NUMBERED COMMON, PLUS THE LENGTH OF ALL I/O BUFFERS. IN THE ILLUSTRATION UNDER 'D', ABOVE, ASSUME BLANK AND NUMBERED COMMON TAKE 10,000 WORDS AND TOTAL I/O BUFFER LENGTH IS 4022 WORDS. THEN E = 20,000 + 25,000 + 10,000 + 4022 = 60,000 OCTAL WORDS (APPROXIMATELY). FOR THE I-MODE, THE PROGRAM LENGTH NEED BE ONLY AS LONG AS THE LONGEST SUBROUTINE.

F IS THE OBJECT PROGRAM I/O BUFFER LENGTHS (OCTAL). IF 'F' IS OMITTED, IT IS SET TO 2001, THE OPTIMUM SIZE FOR EFFICIENT I/O. TO SAVE SPACE, THE USER MAY REDUCE THIS TO 1001, THE MINIMUM SIZE. IF AN 'F' OF LESS THAN 1001 IS REQUESTED, 1001 (OCTAL) WORDS ARE ASSIGNED.

G IS THE OBJECT PROGRAM (BLANK AND NUMBERED) COMMON LENGTH (OCTAL). IF 'G' IS OMITTED, IT IS SET EQUAL TO THE AMOUNT REQUIRED FOR THE MAIN PROGRAM BEING COMPILED. THIS DOES NOT AFFECT LABELED COMMON.

----

(8) CAUTION: THIS NAME MUST NOT BE THE SAME AS THE NAME IN 'A'.

204.3. COMPILATION FROM A TELETYPE

H IS THE LINE LIMIT (OCTAL) FOR THE PRINTABLE OUTPUT FILE. IF 'H' IS OMITTED, THE LINE LIMIT IS SET TO 4500 (OCTAL) AND THE OUTPUT FILE WILL BE 71,400 OCTAL WORDS (4500*14 + 2000).

FOR INSTRUCTIONS ON COMPILING WITH A PRIVATE LIBRARY, SEE SECTION 204.3.3, PAGE 16.

204.3. COMPILATION FROM A TELETYPE

```
EXAMPLES:   CHIP INP / T V

                INPUT FILE IS NAMED 'INP'.
                OUTPUT FILE IS NAMED 'OUTPUT'.
                LENGTH OF COMPILER IN CORE IS 46,000 OCTAL.
                LENGTH OF BINARY EXECUTABLE OUTPUT FILE IS 46,000 OCTAL.

            CHIP INPX,LIST,L,55000,30000,1001,,6000 / T V

                INPUT FILE IS NAMED 'INPX'
                OUTPUT FILE IS NAMED 'LIST', AND  WILL CONTAIN
                   A LISTING OF ALL INSTRUCTIONS IN OCTAL.
                LENGTH OF COMPILER IS 55,000 OCTAL.
                LENGTH OF BINARY EXECUTABLE OUTPUT FILE IS 30,000 OCTAL.
                BCD BUFFER LENGTH IS 1001 OCTAL.
                OUTPUT FILE SIZE IS 6000 OCTAL LINES.
```

-RUN- WILL TELL THE USER HOW MUCH COMPILER SPACE AND HOW MUCH JOB SPACE HAS UNUSED. THIS WILL ENABLE THE USER TO MINIMIZE THE REQUIREMENTS IN FUTURE COMPILATIONS. AN ***MO*** DIAGNOSTIC OR 'PROBLEM ERROR 200' USUALLY MEANS THAT NOT ENOUGH SPACE WAS ALLOWED FOR THE COMPILER -RUN- (THAT IS, FILE /+RUN/). AN ***SF*** DIAGNOSTIC MEANS NOT ENOUGH SPACE ALLOWED FOR OBJECT PROGRAM. *(9)*

NOTE: EXECUTION OF THE BINARY FILE MAY BE ACCOMPLISHED BY STARTING THE BINARY FILE AS A CONTROLLEE FROM THE TELETYPE. TO EXECUTE THE FILE, IT SHOULD BE COPIED BEFORE ITS FIRST EXECUTION, OR -CHANGE- *(10)* SHOULD BE CALLED AS THE FIRST EXECUTABLE STATEMENT IN THE PROGRAM. EITHER PROCEDURE WILL RESERVE THE ORIGINAL STATUS OF THE BINARY FILE.

----

(9) SEE SECTION 204.4.2, PAGE 19.

(10) SEE CHAPTER 306.

204.3.2.   RELOCATABLE SUBROUTINES

SUBROUTINES AND FUNCTIONS WHICH ARE REPEATEDLY COMPILED, BUT NOT CHANGED, MAY BE COMPILED WITHOUT THE MAIN PROGRAM AND KEPT IN A PRIVATE LIBRARY *(11)* TO AVOID RECOMPILATIONS EACH TIME A NEW EXECUTABLE BINARY FILE IS GENERATED. NOTE THAT THE CHIPPEWA COMPILER DOES NOT PRODUCE RELOCATABLE BINARY CARDS.

INPUT TO THE COMPILER IS FROM AN ASCII DISC FILE CONTAINING ONE OR MORE SUBROUTINES AND/OR FUNCTIONS TO BE COMPILED. (THE FIRST ROUTINE CANNOT BE A FUNCTION.) SUBROUTINES COMPILED IN THIS MANNER MUST BE COMPILED IN THE INCOMPLETE (I) MODE (THAT IS, THE THIRD ARGUMENT ON THE -CHIP- EXECUTE LINE MUST BE AN 'I') IF THEY REFER TO OTHER SUBROUTINES OR FUNCTIONS, IF THEY HAVE LABELED COMMON, OR IF THERE IS MORE THAN ONE SUBROUTINE IN THE FILE.

A PRINTABLE LISTING OF ALL THE SUBROUTINES IS WRITTEN IN ONE OUTPUT FILE. EACH SUBROUTINE COMPILED IN THE I-MODE WILL BE WRITTEN INTO A SEPARATE RELOCATABLE BINARY FILE. THE NAME OF EACH OF THESE FILES WILL BE THE SUBROUTINE NAME PRECEDED BY AN ASTERISK. THE SUBROUTINE MUST BE PUT INTO THE PRIVATE LIBRARY FILE UNDER THIS NAME.

ALL SUBROUTINES WHICH CALL OTHER SUBROUTINES OR USE OTHER FUNCTIONS MUST BE COMPILED IN THE I-MODE, UNLESS EVERYTHING IS COMPILED TOGETHER, INCLUDING THE MAIN CODE. IF THEY ARE NOT, A ***BI*** ERROR MAY RESULT WHEN TRYING TO USE THEM. *(12)*

EXAMPLE: CONSIDER THE INPUT FILE 'INP' CONTAINING THE FOLLOWING:

```
SUBROUTINE A (I)
.
END
SUBROUTINE X (A, B)
.
END
FUNCTION BY (J)
.
END
```

----

(11) SEE CHAPTER 301 FOR A DESCRIPTION OF PRIVATE LIBRARY USAGE.

(12) SEE SECTION 204.4.2, PAGE 19.

204.3. COMPILATION FROM A TELETYPE

THE CHIP INPUT LINE IS:

```
    CHIP INP,,I / I I
```

```
WHERE:
          INP    IS THE NAME OF THE ASCII INPUT FILE CONTAINING A,X,BY
          OUTPUT IS THE NAME OF THE LISTABLE OUTPUT FILE
          I      MEANS COMPILE IN INCOMPLETE MODE
RESULTS:
          *A     IS THE NAME OF THE BINARY RELOCATABLE FILE FOR SUBROUTINE A
          *X     IS THE NAME OF THE BINARY RELOCATABLE FILE FOR SUBROUTINE X
          *BY    IS THE NAME OF THE BINARY RELOCATABLE FILE FOR FUNCTION BY
```

204.3.3. COMPILING WITH A LIBRARY

WHEN COMPILING A NEW CODE WHICH USES SUBROUTINES IN A PRIVATE LIBRARY, THE NAME OF THE LIBRARY IS ENCLOSED IN PARENTHESIS AND IS THE FIRST ARGUMENT IN THE CHIP EXECUTE LINE, PRECEDING ARGUMENT 'A', DESCRIBED ON PAGE 10.

EXAMPLE:

```
    CHIP (LIBPRV) INP,,,70000,30000,1000 / T Y
```

THE NAME OF THE PRIVATE LIBRARY IS 'LIBPRV'.
THE NAME OF THE INPUT FILE IS 'INP'.
THE COMPILER SIZE IS 70,000 OCTAL.
THE PROGRAM SIZE IS 30,000 OCTAL.
THE BCD BUFFER SIZE IS 1001 OCTAL.

AFTER THE MAIN PROGRAM AND OTHER ROUTINES IN THE INPUT FILE HAVE BEEN COMPILED AND LOADED, THE PRIVATE LIBRARY WILL BE SEARCHED FOR NEEDED ROUTINES. FINALLY, ANY REMAINING ROUTINES WHICH ARE STILL NEEDED WILL BE SEARCHED FOR IN THE PUBLIC FILE /CLIB/ (SEE CHAPTER 301).

204.4. RESULTS OF COMPILATION

204.4.1. LISTING

(1) FORTRAN STATEMENTS

IN THE CASE OF A SHORT LIST, EACH FORTRAN STATEMENT IS LISTED WITH A NUMBER ON THE LEFT ASSOCIATED WITH EACH STATEMENT. THIS NUMBER IS THE APPROXIMATE OCTAL LOCATION IN THE BINARY OUTPUT FILE OF THE FIRST INSTRUCTION GENERATED BY THAT PARTICULAR FORTRAN STATEMENT. *(13)*

IN THE CASE OF A LONG LIST, EACH FORTRAN STATEMENT IS LISTED AND BENEATH IT THE OCTAL INSTRUCTIONS FOR THAT STATEMENT.

(2) FUNCTION ASSIGNMENTS

A LIST OF EACH ARITHMETIC STATEMENT FUNCTION AND ITS APPROXIMATE LOCATION IN THE BINARY OUTPUT FILE IS GIVEN.

(3) STATEMENT AND VARIABLE ASSIGNMENTS

FOLLOWING EACH SUBROUTINE OR MAIN CODE IS A LIST OF STATEMENT ASSIGNMENTS AND VARIABLE ASSIGNMENTS. THESE ARE THE BEGINNING LOCATIONS FOR EACH NUMBERED STATEMENT AND THE LOCATION OF EACH VARIABLE.

(4) SUBROUTINE ASSIGNMENTS

THIS LIST GIVES THE ENTRY POINT FOR EACH SUBROUTINE CALLED FROM THE PUBLIC LIBRARY /CLIB/ OR FROM YOUR PRIVATE LIBRARY. WHEN A SUBROUTINE IS ENTERED BY THE PROGRAM, A RETURN ADDRESS IS STORED IN THE ENTRY LOCATION. THUS ONE CAN DETERMINE THE LAST POINT IN THE PROGRAM FROM WHICH THE SUBROUTINE WAS CALLED.

(5) BLOCK ASSIGNMENTS

THIS GIVES THE LOCATION OF THE FIRST WORD OF EACH COMMON BLOCK. THE OTHER VARIABLES WITHIN THE COMMON BLOCK ARE IN ASCENDING LOCATIONS.

(13) A SAMPLE OUTPUT LISTING WITH THE SHORT LIST OPTION IS GIVEN IN APPENDIX B, PAGE 62.

204.4. RESULTS OF COMPILATION

(6) BUFFER ASSIGNMENTS

THIS GIVES THE STARTING LOCATION FOR EACH BUFFER. EACH DIFFERENT TAPE ASSIGNMENT REQUIRES A BUFFER.

(7) LOCAL LENGTH (OCTAL)

THIS IS THE LENGTH OF THE BINARY CODE EXCLUDING BLANK AND NUMBERED COMMON AND BUFFER ASSIGNMENTS.

(8) COMMON LENGTH (OCTAL)

THIS IS THE LENGTH OF BLANK AND NUMBERED COMMON.

(9) BUFFER LENGTH (OCTAL)

THIS IS THE TOTAL LENGTH OF ALL BUFFERS.

(10) UNUSED SPACE (OCTAL)

THIS GIVES THE AMOUNT THE COMPILER SIZE MAY BE REDUCED AND THE AMOUNT THE BINARY PROGRAM LENGTH MAY BE REDUCED ON THE -CHIP- INPUT LINE.

(11) VARIABLE NAME MAPS

THE COMPILER WILL GENERATE VARIABLE NAME MAPS FOR SUBROUTINE OR FUNCTIONS LOADED FROM A PRIVATE LIBRARY. TO GET VARIABLE NAME MAPS FOR ROUTINES IN PUBLIC FILE /CLIB/, COMPILE WITH LIBRARY NAME /CLIB/ *(14)*.

----

(14) SEE SECTION 204.3.3, PAGE 16.

204.4.2. ERROR COMMENTS

IN THE OUTPUT FILE PRODUCED DURING COMPILATION, TWO-CHARACTER FORTRAN ERROR PRINTOUTS IN THE FORM ***AC*** (WHERE AC INDICATES THE TYPE OF ERROR) FOLLOW STATEMENTS WHICH ARE INCORRECT. NOTE THAT ANY ERROR SO FLAGGED IS CONSIDERED FATAL, AND NO OBJECT CODE WILL BE PRODUCED.

MISSING STATEMENT NUMBERS ARE LISTED AT THE END OF THE MAIN PROGRAM OR SUBPROGRAM WITHIN WHICH THEY OCCUR AND ARE FOLLOWED BY ***MS***.

MISSING SUBROUTINES ARE LISTED ON THE TELETYPE, AND USUALLY CAUSE A MEMORY OVERFLOW ERROR ***MO***. (THIS MAY BE THE RESULT OF NOT DIMENSIONING A VARIABLE.)

-NAB- MAY BE USED TO LIST ERRORS ON THE TELETYPE.

EXAMPLE:

```
NAB OUTPUT / T V
F^'^***'
        WHERE ^ IS A SPACE, FOLLOWED BY 'T ALL', WILL GIVE THE FORTRAN
        LINES CONTAINING THE ERRORS.

T N-1 2 WILL LIST THE FORTRAN STATEMENT IN  ERROR  AND  THE  TWO-CHARACTER
        DIAGNOSTIC ('N' IS THE LOCATION OF ONE OF THE *** FIELDS,  AS
        OBTAINED ABOVE).
```

THE UNUSED COMPILER SPACE AND THE UNUSED PROGRAM SPACE ARE LISTED ON THE TELETYPE. IT IS THEREFORE POSSIBLE ON FUTURE COMPILATIONS TO KEEP THE COMPILER SIZE AND THE OBJECT CODE SIZE TO A MINIMUM.

THE CHIPPEWA COMPILER DOES NOT SCAN FORMAT STATEMENTS FOR ERRORS. MISCOUNTS IN HOLLERITH TEXT WILL NOT BE FOUND DURING COMPILATION. THE DIAGNOSTIC IN THIS CASE WILL USUALLY OCCUR AT EXECUTE TIME.

THE FOLLOWING IS A LIST OF THE TWO-CHARACTER ERROR COMMENTS AND THEIR MEANINGS. FOR THE MOST PART, THE WORD 'FORMAT' WITHIN THE EXPLANATION OF AN ERROR DIAGNOSTIC REFERS TO THE FORM (OR ARRANGEMENT) OF A PARTICULAR FORTRAN EXPRESSION, AND NOT TO A FORMAT STATEMENT AS SUCH. FOR EXAMPLE,

```
IF(A) 3, ,4
```

IS AN EXPRESSION FORMAT ERROR, SINCE DROP-THROUGH 'IF'S ARE NOT ALLOWED.

AC NUMBER OF ARGUMENTS IN A SUBROUTINE REFERENCE DIFFERS FROM A PREVIOUS REFERENCE.

AL FORMAT ERROR IN A LIST OF ARGUMENTS.

AS FORMAT ERROR IN AN 'ASSIGN' STATEMENT.

BC FORMAT ERROR IN THE DESIGNATION OF A BOOLEAN CONSTANT.

BI A BINARY RELOCATABLE SUBROUTINE HAS THE WRONG FORMAT.

BO A LABELED BLOCK OF COMMON EXCEEDS THE BLOCK LENGTH ALREADY ESTABLISHED.

BX FORMAT ERROR IN A B-TYPE BOOLEAN STATEMENT.

CD VARIABLE NAME IS DUPLICATED IN COMMON.

CE COMMON-EQUIVALENCE ERROR.

CL FORMAT ERROR IN A 'CALL' STATEMENT.

CM FORMAT ERROR IN A 'COMMON' STATEMENT.

CN MORE THAN NINETEEN CONTINUATION CARDS, OR ONE CARD APPEARS IN AN ILLOGICAL SEQUENCE.

CO COMMON OVERFLOW. A SUBROUTINE HAS MORE COMMON THAN THE MAIN PROGRAM. (IF ***CO*** OCCURS FOR EVERY SUBROUTINE AND ONCE MORE AT THE END OF THE CODE, THIS USUALLY MEANS THAT TOO SMALL A COMPILER SIZE WAS REQUESTED ON THE -CHIP- INPUT LINE.)

CT MISSING STATEMENT NUMBER ON A 'CONTINUE' STATEMENT.

DA DUPLICATE DUMMY ARGUMENTS APPEAR IN A FUNCTION-DEFINITION STATEMENT.

DC FORMAT ERROR IN THE EXPRESSION OF A DECIMAL CONSTANT.

DD DUPLICATE NAME IN A 'DIMENSION' STATEMENT.

DF A FUNCTION NAME HAS OCCURRED AS THE NAME OF ANOTHER FUNCTION.

DM FORMAT ERROR IN A 'DIMENSION' STATEMENT.

DO FORMAT ERROR IN A 'DO' STATEMENT.

204.4. RESULTS OF COMPILATION

DP DUPLICATE STATEMENT NUMBER.

DR DATA RANGE ERROR -- A DATA STATEMENT MAY NOT BE USED TO READ DATA INTO BLANK OR NUMBERED COMMON.

DS MISSING 'DO' STATEMENT NUMBER.

DT FORMAT ERROR IN A 'DATA' STATEMENT.

EC EQUIVALENCE CONTRADICTION ERROR.

EF END OF FILE IS READ BEFORE LAST 'END' CARD.

EM IMPROPER MODE OF THE BASE OR EXPONENT OF AN INDICATED EXPONENTIATION.

EQ EQUIVALENCE

TATEMENT.

FM CANNOT DETERMINE TYPE OF STATEMENT.

FN MISSING STATEMENT NUMBER ON A 'FORMAT' STATEMENT.

FS FORMAT SPECIFICATION ERROR.

FT ERROR IN A TYPE STATEMENT.

GO FORMAT ERROR IN A 'GO TO' STATEMENT.

ID ILLEGAL 'DO'.

IF ERROR IN 'IF' STATEMENT (DROP-THROUGH IS NOT ALLOWED).

IL ERROR IN INDEXED LIST OF AN I/O STATEMENT.

IT ILLEGAL TRANSFER.

LN 'NAMELIST' ERROR.

LR MORE ARGUMENTS ARE REFERENCED THAN A STANDARD LIBRARY SUBROUTINE USES.

LS ERROR IN AN I/O LIST.

MA AN ARGUMENT OF A SUBROUTINE OR FUNCTION HAS BEEN MISUSED IN AN EQUIVALENCE STATEMENT.

MC MACHINE CONSTANT ERROR.

MD MACHINE DUPLICATE TAG.

MF MACHINE FORMAT ERROR.

ML MACHINE LOCATION TAG ERROR.

MJ COMPILER FIELD LENGTH, AS TYPED IN ON THE -CHIP- INPUT LINE, IS TOO SHORT.

MR MISSING SUBROUTINE.

MS MISSING STATEMENT NUMBERS.

MT MACHINE TAG DEFINITION ERROR.

NC NAME CONFLICT OF A SUBROUTINE OR FUNCTION.

NM ERROR IN FORMAT ON THE PROGRAM CARD.

OD REFERENCE TO AN ARRAY OCCURS BEFORE ITS 'DIMENSION' STATEMENT.

PN UNEQUAL NUMBER OF OPEN AND CLOSED PARENTHESES.

RN FORMAT ERROR IN A 'RETURN' STATEMENT.

SB FORMAT ERROR IN A SUBSCRIPT OF AN ARRAY.

SE FORMAT ERROR IN A 'SENSE' STATEMENT.

SF THE REQUIRED LENGTH OF THE OBJECT CODE, AS SPECIFIED ON THE -CHIP- INPUT LINE, IS TOO SHORT.

SL THE COMPILER FIELD LENGTH OR THE OBJECT CODE FIELD LENGTH, AS TYPED IN ON THE -CHIP- INPUT LINE, IS NOT LARGE ENOUGH TO LOAD ALL THE NEEDED LIBRARY SUBROUTINES.

SM ERROR IN THE STATEMENT-LABEL FIELD.

SN ERROR WHERE A STATEMENT NUMBER SHOULD APPEAR.

SY FORTRAN SYSTEM ERROR.

204.4. RESULTS OF COMPILATION

TM MORE THAN SIXTY ARGUMENTS IN A SUBROUTINE LIST OR A SUBROUTINE REFERENCE.

TY ERROR IN A TYPE STATEMENT.

UA UNIDENTIFIED-ARRAY ERROR.

UE A REFERENCE TO AN I/O FILE OR TAPE THAT HAS NOT LISTED IN THE PROGRAM CARD.

US UNREFERENCED BINARY SUBROUTINE.

VC VARIABLE NAME CONFLICT.

VD VARIABLE DIMENSIONED ARRAY ERROR.

XF ERROR IN AN EXPRESSION.

204.4. RESULTS OF COMPILATION

THE FOLLOWING ERROR COMMENTS MAY BE PRINTED ON THE TELETYPE DURING EXECUTION:

COMPILER FIELD LENGTH TOO SMALL FOR LOADING. RESTART CHIP

CAN'T GIVE (FILENAME) TO USER 999999. USE OUT OR GIVE.

> -RUN- GIVES THIS MESSAGE WHEN THE OLP OR HSP OPTION IS USED AND THE FILE CANNOT BE GIVEN TO USER 999999.

CAN'T OPEN LIB FILE (FILENAME)

> -RUN- GIVES THIS MESSAGE IF /CLIB/ OR PRIVATE LIBRARY FILE CANNOT BE OPENED.

CIO ARGUMENT ERROR.

> -RUN- GIVES THIS MESSAGE WHEN ONE OF THE COMPILER'S I/O BUFFER LIMITS IS EXCEEDED. THE USER SHOULD RESTART.

CANNOT CREATE FILE (FILENAME).

> -RUN- GIVES THIS MESSAGE IF THE BINARY FILE CANNOT BE CREATED. -RUN- TRIES TO OPEN, DESTROY AND RECREATE IF THE FIRST 'CREATE' CALL FAILS.

CANNOT OPEN FILE (FILENAME).

> -RUN- GIVES THIS MESSAGE IF ANY FILE TRYING TO BE ACCESSED BY -RUN- CANNOT BE OPENED.

NO MORE MINUS WORDS.

> -RUN- GIVES THIS MESSAGE IF THE COMPILER USES ALL AVAILABLE MINUS WORDS WHILE CREATING BINARY FILES.

BAD ASCII FILE. NO END OF FILE.

> -RUN- GIVES THIS MESSAGE IF AN ERROR OCCURS WHILE READING FROM THE ASCII INPUT FILE.

GOB IO ERROR NUMBER (NO.) WHILE WRITING IN (FILENAME).

> -RUN- GIVES THIS MESSAGE IF IT HAS TROUBLE WRITING EITHER THE BINARY OR OUTPUT FILE.

204.5. AVAILABLE FORTRAN LANGUAGE

204.5.1. FORTRAN STATEMENT FORMAT

THERE ARE THREE TYPES OF CODING LINES:

TYPE	COL.	CONTENT
STATEMENT	1-5	STATEMENT NUMBER (ALPHANUMERIC STATEMENT LABELS ARE NOT ALLOWED).
	1	D, I, B, F [FORTRAN II].
	6	BLANK OR ZERO.
	7-72	FORTRAN STATEMENT.
	73-80	IDENTIFICATION (IGNORED BY THE COMPILER).
CONTINUATION	1-5	BLANK.
	6	FORTRAN CHARACTER OTHER THAN BLANK OR ZERO.
	7-72	CONTINUED FORTRAN STATEMENT.
	73-80	IDENTIFICATION FIELD.
COMMENT	1	CHARACTER C, *, OR $.

204.5.2. LIST OF AVAILABLE FORTRAN STATEMENTS

AS NOTED IN THE INTRODUCTION, -CHIP- WILL COMPILE EITHER FORTRAN II OR FORTRAN IV PROGRAMS. THE USER SHOULD REFER TO [1] FOR DETAILED LANGUAGE DESCRIPTIONS. THIS SECTION MERELY LISTS THE AVAILABLE LANGUAGE STATEMENTS.

204.5. AVAILABLE FORTRAN LANGUAGE

(1) SUBPROGRAM STATEMENTS:

```
          ENTRY POINTS           SEGMENT NAME (F1,F2,...)
                                 PROGRAM NAME (F1,...,FN)
                                 FORTRAN IV PROGRAM NAME (F1,...,FN)
                                 FORTRAN II PROGRAM NAME (F1,...,FN)
                                 MACHINE PROGRAM NAME (F1,...,FN)
                                 ASCENTF PROGRAM NAME (F1,...,FN)
                                 SUBROUTINE NAME (P1,...,PN)
                                 FORTRAN IV SUBROUTINE NAME (P1,...,PN)
                                 FORTRAN II SUBROUTINE NAME (P1,...,PN)
                                 MACHINE SUBROUTINE NAME (P1,...,PN)
                                 ASCENTF SUBROUTINE NAME (P1,...,PN)
                                 FUNCTION NAME (P1,...,PN)
                                 TYPE FUNCTION NAME (P1,...,PN)
                                 FORTRAN IV FUNCTION NAME (P1,...,PN)
                                 FORTRAN II FUNCTION NAME (P1,...,PN)
                                 FORTRAN IV TYPE FUNCTION NAME (P1,...,PN)
                                 FORTRAN II TYPE FUNCTION NAME (P1,...,PN)

          INTERSUBROUTINE        EXTERNAL NAME1,NAME2,...
                           F     NAME1,NAME2,...          [FORTRAN II MODE]

          TRANSFER STATEMENTS    CALL NAME
                                 CALL NAME (P1,...,PN)
                                 RETURN
```

(2) DATA DECLARATION AND STORAGE ALLOCATIONS:

```
          TYPE DECLARATION       COMPLEX LIST
                                 DOUBLE PRECISION LIST
                                 DOUBLE LIST
                                 REAL LIST
                                 INTEGER LIST
                                 LOGICAL LIST

          STORAGE ALLOCATION     DIMENSION V1,V2,...,VN
                                 COMMON /NAME/ LIST
                                 EQUIVALENCE (A,B,...),(A1,B1,...)...
                                 DATA V1/LIST/,V2/LIST/,
                                 BLOCK DATA
```

204.5. AVAILABLE FORTRAN LANGUAGE

```
(3) ARITHMETIC STATEMENT FUNCTIONS:

                                   NAME (P1,P2,...PN) = EXPRESSION

(4) SYMBOL MANIPULATION AND CONTROL:

       REPLACEMENT             A=E    ARITHMETIC
                                      D      A=E        [FORTRAN II MODE]
                                      I      A=E        [FORTRAN II MODE]
                               L=E    LOGICAL/RELATIONAL
                               M=E    MASKING
                                      B      M=E        [FORTRAN II MODE]

       INTRAPROGRAM            GO TO L                  ['L' IS A LABEL]
        TRANSFERS              GO TO M                  ['M' IS A VARIABLE]
                               GO TO M, (N,...NM)
                               GO TO (N1,...,NM),I
                               IF (A) N1,N2,N3
                               IF (L) N1,N2
                               IF (L) <STATEMENT>
                               IF DIVIDE CHECK N1,N2
                               IF (ENDFILE I)N1,N2
                               IF (EOF,I)N1,N2
                               IF ACCUMULATOR OVERFLOW N1,N2
                               IF QUOTIENT OVERFLOW N1,N2

(5) LOOP CONTROL:
                               DO N I = M1,M2,M3

(6) MISCELLANEOUS PROGRAM CONTROLS:
                               ASSIGN S TO M
                               CONTINUE
                               PAUSE
                               PAUSE N
                               STOP
                               STOP N

(7) I/O FORMAT:
                               FORMAT (SPEC1,SPEC2,...)
```

(8) I/O STATEMENTS:

```
                                READ N, L
                                PRINT N, L
                                READ (I, N) L
                                READ INPUT TAPE I, N, L
                                WRITE (I, N) L
                                WRITE OUTPUT TAPE I, N, L
                                READ (I) L
                                READ TAPE I, L
                                WRITE (I) L
                                WRITE TAPE I, L
                                READ (I, X)
                                WRITE (I, X)

       I/O TAPE HANDLING        END FILE I
                                REWIND I
                                BACKSPACE I

       BUFFERED I/O             BUFFER IN (I,M)(A(K1),B(K2))
                                BUFFER OUT (I,M)(A(K1),B(K2))
                                IF(UNIT,I) N1,N2,N3,N4
                                IF(IOCHECK,I) N1,N2
                                K=LENGTH(I)
```

(9) PROGRAM AND SUBPROGRAM TERMINATION:

```
                                END
```

204.5.3. INPUT/OUTPUT STATEMENTS

IN THIS SECTION A NUMBER OF THE I/O STATEMENTS LISTED ABOVE ARE DESCRIBED IN DETAIL. PLEASE REFER ALSO TO CHAPTER 303.

204.5. AVAILABLE FORTRAN LANGUAGE

READ -- TELETYPE OR DISC INPUT

SUMMARY: TO READ INPUT FROM THE TELETYPE OR FROM A DISC FILE.

FORM: READ N, L

WHERE: N FORMAT NUMBER.
L LIST

RESTRICTIONS: IF THIS STATEMENT IS USED, THE NAME 'INPUT' MUST APPEAR ON THE PROGRAM CARD *(15)*. IF THE NAME 'INPUT' IS NOT EQUATED TO A TAPE NUMBER, THEN THE INPUT WILL BE FROM TELETYPE.

PRINT -- TELETYPE OR DISC OUTPUT

SUMMARY: TO OUTPUT ON THE TELETYPE OR INTO A DISC FILE.

FORM: PRINT N, L

WHERE: N FORMAT NUMBER
L LIST

RESTRICTIONS: IF THIS STATEMENT IS USED, THE NAME 'OUTPUT' MUST APPEAR ON THE PROGRAM CARD. IF THE NAME 'OUTPUT' IS NOT EQUATED TO A TAPE NUMBER, THEN THE OUTPUT WILL BE TO TELETYPE.

EXAMPLE:

```
      PROGRAM EXAMPLE (INPUT, OUTPUT)
      PRINT 1
1     FORMAT (18HTYPE INITIAL VALUE)
      READ 2, A
2     FORMAT (F15.8)
       .
       .
```

---

(15) SEE SECTION 204.2.2, PAGE 6.

204.5. AVAILABLE FORTRAN LANGUAGE

READ/WRITE -- FORMATTED BCD I/O

SUMMARY: TO READ OR WRITE BCD TAPES, ASCII DISC FILES, OR TO COMMUNICATE WITH THE TELETYPE.

FORM:

```
READ INPUT TAPE I, N, L
READ (I, N) L

WRITE OUTPUT TAPE I, N, L
WRITE (I,N) L
```

WHERE:

I TAPE UNIT NUMBER
N FORMAT NUMBER
L LIST

REMARKS:

1. ASCII DISC FILES.
   THE FILE NAME MAY BE DECLARED ON THE PROGRAM CARD AND THE TAPE NUMBER EQUATED TO THE FILE NAME. OTHERWISE A 'CALL ASSIGN' MAY BE USED TO EQUATE THE TAPE NUMBER AND DISC FILE NAME (SEE CHAPTER 303).

2. BCD TAPES.
   THE TAPE VAULT NUMBER MUST BE DECLARED ON THE PROGRAM CARD, AND THE TAPE NUMBER MUST BE EQUATED TO THE TAPE VAULT NUMBER.

3. TELETYPE
   IF I=59, INFORMATION WILL BE READ FROM OR WRITTEN ON THE TELETYPE. IF I IS THE INTEGER 59, THEN 'TAPE59' MUST APPEAR ON THE PROGRAM CARD. IF I IS A VARIABLE WHICH HAS BEEN SET TO 59, 'TAPE59' DOES NOT HAVE TO APPEAR ON THE PROGRAM CARD, AND THE PROGRAM WILL BE 2001 WORDS SHORTER.

4. dd80
   IF I=100, INFORMATION WILL BE WRITTEN ON THE dd80C VIA THE -CRTBCD- ROUTINE (SEE CHAPTER 304). IF I IS THE INTEGER 100, THEN 'TAPE100' MUST APPEAR ON THE PROGRAM CARD. IF I IS A VARIABLE WHICH HAS BEEN SET TO 100 'TAPE100' DOES NOT HAVE TO APPEAR ON THE PROGRAM CARD AND THE 2001 WORD BUFFER WILL BE ELIMINATED.

EXAMPLE:

```
PROGRAM EXAMPLE (HICCUP, TAPE1=HICCUP, TAPE59)
READ (1, 5) A, B, C
WRITE (59, 3) A, B, C
```

THIS READS THE DISC FILE HICCUP AND PRINTS A, B, C ON THE TELETYPE.

EXAMPLE:

```
PROGRAM EXAMPLE (HICCUP, TAPE1=HICCUP)
READ (1, 5) A, B, C
N=59
WRITE (N, 3) A, B, C
```

THIS DOES THE SAME AS THE ABOVE EXAMPLE. HOWEVER, THE 2001 WORD BUFFER FOR TAPE59 IS DELETED.

READ/WRITE -- BINARY I/O

SUMMARY: TO READ OR WRITE BINARY TAPES OR DISC FILES.

FORM:

```
READ TAPE I, L

WRITE TAPE I, L
```

WHERE: I TAPE UNIT NUMBER
L LIST

REMARKS: 1. BINARY FILES.

THE FILE NAME MAY BE DECLARED ON THE PROGRAM CARD AND THE TAPE NUMBER EQUATED TO THE FILE NAME. A 'CALL ASSIGN' MAY BE USED TO EQUATE THE TAPE NUMBER TO A DISC FILE.

2. BINARY TAPES

THE TAPE VAULT NUMBER MUST BE DECLARED ON THE PROGRAM CARD AND THE TAPE NUMBER MUST BE EQUATED TO THE TAPE VAULT NUMBER.

EXAMPLE:

```
PROGRAM EX (AE222, TAPE2=AE222, XX, TAPE4=XX)
.
.
N=2
READ TAPE N, A, B, C
WRITE TAPE 4, A, B, C
```

THIS READS TAPE 'AE222' AND WRITES IN THE DISC FILE 'XX'.

END FILE -- WRITE END-OF-FILE

SUMMARY: TO WRITE AN END-OF-FILE MARK ON TAPE OR DISC.

FORM: END FILE I

WHERE: I TAPE UNIT NUMBER

REWIND -- REWIND TAPE

SUMMARY: TO REWIND A TAPE OR SET THE FIRST WORD ADDRESS OF THE DISC BACK TO ZERO.

FORM: REWIND I

WHERE: I TAPE UNIT NUMBER

REMARKS: REWIND TURNS OFF ALL END-OF-FILE SIGNALS.

BACKSPACE -- BACKSPACE RECORD

SUMMARY: TO BACKSPACE ONE RECORD OF BINARY INFORMATION ON TAPE OR DISC, OR BACKSPACE ONE RECORD OF BCD INFORMATION ON TAPE.

FORM: BACKSPACE I

WHERE: I TAPE UNIT NUMBER

BUFFER IN/OUT -- BUFFERED TAPE OR DISC I/O

SUMMARY: TO READ INFORMATION FROM OR WRITE INFORMATION TO DISC OR TAPE.

FORM:

```
BUFFER IN (I,M) (A(K1), B(K2))

BUFFER OUT (I,M) (A(K1), B(K2))
```

WHERE:

```
I      TAPE UNIT NUMBER
M      MODE.     0 IS BCD.  1 IS BINARY.
A(K1) FIRST WORD ADDRESS
B(K2) LAST WORD ADDRESS
```

REMARKS:

1. THE COMPILER USES DISPLAY CODE *(16)* INTERNALLY, NOT ASCII. BUFFER IN, IN THE BCD MODE, DOES NOT CONVERT ASCII TO DISPLAY, AND BUFFER OUT IN THE BCD MODE DOES NOT CONVERT DISPLAY TO ASCII. 'CALL SWITCH' MAY BE USED TO DO THIS (SEE CHAPTER 307).

2. THE TAPE NUMBER MUST BE EQUATED TO A DISC FILE OR A TAPE EITHER ON THE PROGRAM CARD OR WITH A 'CALL ASSIGN' STATEMENT. IF DONE ON THE PROGRAM CARD, A BUFFER IS NOT NEEDED. THE DISC FILE NAME OR TAPE NAME MAY BE SET EQUAL TO ZERO TO ELIMINATE A 2001 WORD BUFFER. FOR EXAMPLE:

```
PROGRAM SET(XX=0, TAPE1=XX)
DIMENSION A(100)
BUFFER IN (1, 0) (A(1), A(100))
```

---

(16) SEE APPENDIX A, PAGE 61.

204.5. AVAILABLE FORTRAN LANGUAGE

IF (UNIT) -- TEST FOR COMPLETION OF BUFFERED I/O

SUMMARY: TO CHECK THE STATUS OF A PREVIOUSLY INITIATED 'BUFFER IN' OR 'BUFFER OUT'.

FORM: IF (UNIT, I) N1, N2, N3, N4

WHERE:
I TAPE UNIT NUMBER
N1 NEXT STATEMENT IF BUFFERED I/O NOT FINISHED.
N2 NEXT STATEMENT IF BUFFERED I/O FINISHED WITH NO ERRORS.
N3 NEXT STATEMENT IF BUFFERED I/O FINISHED WITH END-OF-FILE OR END-OF-TAPE. (USE 'IF (EOF)' TO DISTINGUISH THESE TWO CASES.)
N4 NEXT STATEMENT IF BUFFERED I/O FINISHED WITH PARITY OR RECORD LENGTH ERROR. (USE 'IF (IOCHECK)' TO DISTINGUISH THESE TWO CASES.)

REMARKS:
1. A RECORD LENGTH ERROR OCCURS (TAPE ONLY) IF THERE ARE MORE WORDS IN THE RECORD THEN THE NUMBER OF WORDS REQUESTED.
2. IF N1 BRANCHES BACK TO THE UNIT TEST, A DELAY UNTIL I/O IS FINISHED OCCURS.

IF (EOF) -- TEST FOR END-OF-FILE

SUMMARY: TO DETERMINE IF AN END-OF-FILE HAS BEEN READ FROM DISC OR TAPE.

FORM:
IF (ENDFILE I) N1, N2
IF (EOF, I) N1, N2

WHERE:
I TAPE UNIT NUMBER
N1 NEXT STATEMENT IF END-OF-FILE FOUND
N2 NEXT STATEMENT IF NO END-OF-FILE FOUND

IF (IOCHECK) -- TEST FOR PARITY ERROR

```
  SUMMARY:  TO DETERMINE IF A PARITY ERROR EXISTS.

     FORM:       IF (IOCHECK, I) N1, N2

    WHERE:  I    TAPE UNIT NUMBER
            N1   NEXT STATEMENT IF PARITY ERROR.
            N2   NEXT STATEMENT IF NO PARITY ERROR.
```

LENGTH -- DETERMINE RECORD LENGTH

```
  SUMMARY:  TO DETERMINE THE NUMBER OF COMPUTER WORDS TRANSMITTED IN THE  LAST  BUFFERED
            OPERATION ON UNIT 'I'.

     FORM:       K = LENGTH (I)

    WHERE:  I    TAPE UNIT NUMBER.
            K    NUMBER OF WORDS TRANSMITTED IN THE LAST BUFFERED I/O ON UNIT 'I'.
```

Errata 1
March 9, 1970

204.5.4. BUILT-IN FUNCTIONS

NOTE: SEE CHAPTERS 302-307 FOR AVAILABLE LIBRARY ROUTINES.

FORM	ALTERNATE FORM	DEFINITION	ACTUAL PARAMETER TYPE	MODE OF RESULT
ABS(X)	ABSF(X)	ABSOLUTE VALUE	REAL	REAL
AIMAG(C)		OBTAIN THE IMAGINARY PART OF A COMPLEX ARGUMENT	COMPLEX	REAL
AINT(X)	INTF(X)	TRUNCATION, INTEGER	REAL	REAL
AMAX0(I1,I2,..)	MAX0F(I1,I2,..)	DETERMINE MAXIMUM ARGUMENT	INTEGER	REAL
AMAX1(X1,X2,..)	MAX1F(X1,X2,..)	DETERMINE MAXIMUM ARGUMENT	REAL	REAL
AMIN0(I1,I2,..)	MIN0F(I1,I2,..)	DETERMINE MINIMUM ARGUMENT	INTEGER	REAL
AMIN1(X1,X2,..)	MIN1F(X1,X2,..)	DETERMINE MINIMUM ARGUMENT	REAL	REAL
AMOD(X1,X2)	MODF(X1,X2)	X1 MODULO X2	REAL	REAL
CMPLX(X1,X2)		CONVERT REAL TO COMPLEX (X1 + X2 i)	REAL	COMPLEX
CONJG(C)		CONJUGATE OF C	COMPLEX	COMPLEX

DIM(X1,X2)	DIMF(X1,X2)	IF X1 GREATER THAN X2: X1 - X2
		IF X1 LESS THAN OR EQUAL TO X2: 0
			REAL	REAL

DMAX1(D1,D2,..)		DETERMINE MAXIMUM ARGUMENT	DOUBLE	DOUBLE

Errata 1
March 9, 1970

DMIN1(D1,D2,..)		DETERMINE MINIMUM ARGUMENT	DOUBLE	DOUBLE
FLOAT(I)	FLOATF(I)	INTEGER OF REAL CONVERSION	INTEGER	REAL
IABS(I)	XABSF(I)	ABSOLUTE VALUE	INTEGER	INTEGER

IDIM(I1,I2)	XDIMF(I1,I2)	IF I1 GREATER THAN I2:
		I1 - I2
		IF I1 LESS THAN OR EQUAL
		TO I2: 0
			INTEGER	INTEGER

IFIX(X)	XFIXF(X)	REAL-TO-INTEGER CONVERSION	REAL	INTEGER
INT(X)	XINTF(X)	TRUNCATION INTEGER	REAL	INTEGER
ISIGN(I1,I2)	XSIGNF(I1,I2)	SIGN OF I2 TIMES I1	INTEGER	INTEGER
MAX0(I1,I2,..)	XMAX0F(I1,I2,..)	DETERMINE MAXIMUM ARGUMENT	INTEGER	INTEGER
MAX1(X1,X2,..)	XMAX1F(X1,X2,..)	DETERMINE MAXIMUM ARGUMENT	REAL	INTEGER
MIN0(I1,I2,..)	XMIN0F(I1,I2,..)	DETERMINE MINIMUM ARGUMENT	INTEGER	INTEGER
MIN1(X1,X2,..)	XMIN1F(X1,X2,..)	DETERMINE MINIMUM ARGUMENT	REAL	INTEGER
MOD(I1,I2)	XMODF(I1,I2)	I1 MODULO I2	INTEGER	INTEGER
REAL(C)		OBTAIN THE REAL PART OF A COMPLEX ARGUMENT	COMPLEX	REAL
SIGN(X1,X2)	SIGNF(X1,X2)	SIGN OF X2 TIMES X1	REAL	REAL

204.6. COMPILER AND LIBRARY ANOMALIES

1. ONLY A SIMPLE INTEGER ARITHMETIC EXPRESSION CAN BE USED AS A SUBSCRIPT, WHERE 'SIMPLE' MEANS CONTAINING NO EXPONENTIATION OR PARENTHESIS (EXCEPT FOR REFERENCES TO BUILT-IN FUNCTIONS).

2. FORTRAN II STATEMENTS MAY BE USED IN FORTRAN IV PROGRAMS, EXCEPT WHERE COMMON IS REORDERED BY EQUIVALENCE.

3. MORE THAN ONE STATEMENT MAY BE PUT ON A LINE. THE DOLLAR SIGN ($) SEPARATES THESE STATEMENTS. THE USE OF THE DOLLAR SIGN IS LIKE STARTING IN COL. 7; THAT IS, NO STATEMENT NUMBER IS ALLOWED ON THE SECOND STATEMENT.

4. BLANK CARDS ARE NOT ALLOWED BEFORE PROGRAM, FUNCTION OR SUBROUTINE CARDS. BLANKS ARE ALLOWED BETWEEN THE HEADER CARD AND 'END' CARDS.

5. MULTIPLE REPLACEMENT STATEMENTS ARE ALLOWED, I.E., A=B=C=0.0

6. ONE- AND TWO-BRANCH LOGICAL 'IF' STATEMENTS ARE ALLOWED:

```
   IF(L) S
   IF(L) N1,N2
```

'L' IS A LOGICAL EXPRESSION.
'S' IS A STATEMENT.
     IF 'L' IS TRUE (NON ZERO) STATEMENT 'S' IS EXECUTED.
     OTHERWISE THE PROGRAM CONTINUES WITH THE NEXT STATEMENT.
'N1' AND 'N2' ARE STATEMENT NUMBERS.
     THE PROGRAM GOES TO STATEMENT 'N1' IF 'L' IS TRUE (NON ZERO).
     IF 'L' IS FALSE (ZERO) TRANSFER IS TO STATEMENT 'N2'.

7. THE DIVIDE (/) IN FORTRAN II BOOLEAN STATEMENTS ('B' IN COL. 1) IS AN 'EXCLUSIVE OR' MASKING OPERATION.

8. 'DATA' STATEMENTS MAY BE WRITTEN AS FOLLOWS:

```
1.  DATA ( (GIB(I),I=1,10) = 1., 2., 3., 4(4.32) )
2.  DATA (GIB(I),I=1,10) /1., 2., 3., 4=4.32/
3.  DATA GIB /1., 2., 3., 4*4.32/
4.  DATA (GIB = 1., 2., 3., 4(4.32))
```

THESE STATEMENTS ALL SET THE FIRST SEVEN LOCATIONS OF ARRAY -GIB- TO THE FOLLOWING VALUES:

     1., 2., 3., 4.32, 4.32, 4.32, 4.32

204.6. COMPILER AND LIBRARY ANOMALIES

TYPES 1 AND 2 MAY NOT BE USED FOR DOUBLY OR TRIPLY SUBSCRIPTED ARRAYS. TYPES 3 AND 4 WORK FOR ANY TYPE ARRAY.

'DATA' STATEMENTS MAY NOT BE USED TO SET VALUES IN BLANK OR NUMBERED COMMON.

9. 'H' AND 'R' FIELDS ARE AVAILABLE. AN 'H' FIELD IS LEFT ADJUSTED WITH BLANK FILL. AN 'R' FIELD IS RIGHT ADJUSTED WITH ZERO FILL. HOLLERITH CONSTANTS ARE ALWAYS TYPE 'INTEGER'. FOR EXAMPLE:

```
     4HCHIP     GENERATES MACHINE WORD     03101120555555555555 .   *(17)*
     4RCHIP     GENERATES MACHINE WORD     00000000000003101120 .   *(17)*
```

10. DOUBLE PRECISION CONSTANTS -- THE LOW ORDER PART IS ALWAYS SET TO ZERO BY THE COMPILER. THE INPUT ROUTINE WILL CORRECTLY CONVERT A DOUBLE PRECISION CONSTANT IF A 'D' FORMAT IS USED.

11. THE STANDARD LIBRARY ROUTINES DO NOT CHECK FOR INDEFINITE OR OUT OF RANGE RESULTS.

12. IF DIVIDE CHECK N1, N2   [NO PARENTHESES]
CHECKS REGISTERS X6 AND X7 FOR AN OUT OF RANGE OR INDEFINITE CONDITION. IF EITHER OF THESE CONDITIONS EXIST, CONTROL IS TRANSFERRED TO STATEMENT 'N1'. OTHERWISE CONTROL IS TRANSFERRED TO 'N2'.

IF ACCUMULATOR OVERFLOW N1,N2   [NO PARENTHESES]
AND
IF QUOTIENT OVERFLOW N1,N2   [NO PARENTHESES]
BOTH CHECK REGISTERS X6 AND X7 FOR AN OUT OF RANGE CONDITION. IF THIS CONDITION EXISTS, CONTROL IS TRANSFERRED TO 'N1'. OTHERWISE, CONTROL IS TRANSFERRED TO STATEMENT 'N2'.

13. STATEMENT LABELS MAY BE USED AS SUBROUTINE ARGUMENTS. E.G., CALL BYIN (A,10S) PUTS THE ADDRESS OF 'A' IN REGISTER B1 AND THE ADDRESS OF STATEMENT 10 INTO REGISTER B2.
HOWEVER, A STATEMENT LABEL CANNOT BE USED WITH THE LOC FUNCTION. V = LOC(96S) WILL NOT COMPILE.

14. 'CONTINUE' STATEMENTS MUST HAVE STATEMENT NUMBERS.

15. USING BLANK AND NUMBERED COMMON REDUCES THE AMOUNT OF CORE STORAGE REQUIRED BY THE COMPILER.

16. 'PARAMETER' AND 'CLICHE' STATEMENTS ARE NOT AVAILABLE. MULTIPLE ENTRY POINTS TO SUBROUTINES ARE NOT ALLOWED. 'NAMELIST' AND 'PUNCH' STATEMENTS HAVE NOT BEEN IMPLEMENTED.

----

(17) IN DISPLAY CODE. (SEE APPENDIX A, PAGE 61.)

204.6. COMPILER AND LIBRARY ANOMALIES

17. THE STATEMENTS WHICH TEST AND SET SENSE LIGHTS OR SENSE SWITCHES SHOULD NOT BE USED, SINCE THEY STORE BITS AND TEST BITS IN WORD 0.

18. THE 'ENCODE' AND 'DECODE' STATEMENTS DO NOT WORK. HOWEVER, READ BCD AND WRITE BCD (WHEN USED WITH DISC FILES) MAY BE USED TO ACHIEVE THE SAME EFFECT. THE FOLLOWING EXAMPLE ILLUSTRATES THE CODING.

```
   REWIND 6
   WRITE (6,1) ALPHA
1  FORMAT (1X,A10)
   READ (6,2) (CHARS(K),K=1,10)
2  FORMAT(1X,10A1)
```

THE NUMBER OF WORDS WRITTEN MUST NOT EXCEED 512, BECAUSE THE BUFFER IS AUTOMATICALLY EMPTIED WHEN 512 WORDS HAVE BEEN WRITTEN. SINCE THE INFORMATION NEVER GOES TO DISC, ONLY A ONE WORD DISC FILE NEED BE CREATED AND ASSOCIATED WITH THE TAPE NUMBER USED FOR FORMAT CONVERSION.

THERE SHOULD ALWAYS BE A 1X AT THE BEGINNING OF EACH FORMAT STATEMENT, SINCE A '1' OR '0' AS THE FIRST CHARACTER CAUSES A PAGE RESTORE OR A LINE FEED TO BE INSERTED BEFORE THE INFORMATION.

NOTE 1: IT IS NOT POSSIBLE TO REREAD THIS INFORMATION A SECOND TIME.

NOTE 2: IF A 'REWIND' IS INSERTED BETWEEN THE 'WRITE' AND THE 'READ' THE INPUT ROUTINE WILL SEND INFORMATION FROM THE DISC FILE WHICH CAUSES INCORRECT RESULTS.

19. -XLOCF- DOES NOT WORK. USE -LOC-.

20. FILE NAMES AND VARIABLE NAMES MAY NOT BE MORE THAN SEVEN CHARACTERS.

21. 'PAUSE', 'PAUSE N', 'STOP' OR 'STOP N' WILL STOP A PROGRAM, TYPE OUT THE MESSAGE:

```
     STOP OR PAUSE.  HIT LINEFEED TO CONTINUE.
```

AND WAIT FOR A LINEFEED TO CONTINUE.

22. FOR LOGICAL EXPRESSIONS, A MINUS ZERO OR A NONZERO QUANTITY IS CONSIDERED TRUE, AND ONLY A PLUS ZERO IS CONSIDERED FALSE. THE EXPRESSION (I.LT.0) IS EVALUATED AS TRUE IF I = -0.

23. WHEN WRITING WITH AN 'E' OR 'F' TYPE FORMAT, INFINITE OR OUT OF RANGE NUMBERS ARE PRINTED AS 'RRRR'; INDEFINITE NUMBERS ARE PRINTED AS 'IIII'.

204.6. COMPILER AND LIBRARY ANOMALIES

24. NUMERICAL FIXED POINT ('I' SPECIFICATION) OR FLOATING POINT ('F' SPECIFICATION) OUTPUT HAVING AN ASTERISK (*) AS THE LEADING CHARACTER SIGNIFIES THAT THE ACTUAL VALUE OF THE DATA WAS TOO LARGE FOR THE SPECIFIED FIELD WIDTH.

25. HOLLERITH DATA MAY BE PRINTED FROM A FORMAT USING '*' RATHER THAN 'H'.
FOR EXAMPLE:

```
     FORMAT (*PRINT OUT THIS COMMENT*)
```

WILL PRINT OUT THE CHARACTERS BETWEEN THE *'S AS HOLLERITH DATA.

26. THE CHIPPEWA COMPILER USES DISPLAY CODE INTERNALLY, RATHER THAN ASCII. SEE APPENDIX A, PAGE 61.

204.7. DESCRIPTION OF PUBLIC FILE /CHIP/

204.7.1. OPERATIONAL DETAILS

THE PUBLIC FILE /CHIP/ PROVIDES A MEANS OF PASSING INFORMATION FROM THE USER TO THE CHIPPEWA COMPILER -RUN- (WHICH RESIDES IN FILE /+RUN/ AT COMPILE TIME). THIS INFORMATION IS USED BY -RUN- IN THE COMPILING OR ASSEMBLING OF CODES WRITTEN FOR THE CHIPPEWA COMPILER.

-CHIP- MAY BE STARTED IN ONE OF TWO MANNERS:

```
 1.     USER TYPES:      CHIP (I) A,B,C,E,F,G,H / T V

        WHERE:      I    PRIVATE LIBRARY NAME, IF ANY.
                    A    NAME OF ASCII INPUT FILE.
                    B    FILE NAME FOR PRINTABLE OUTPUT.
                    C    COMPILE MODE.
                    D    COMPILER FIELD LENGTH (OCTAL).
                    E    PROGRAM FIELD LENGTH (OCTAL).
                    F    PROGRAM I/O BUFFER LENGTHS (OCTAL).
                    G    PROGRAM COMMON LENGTH (OCTAL).
                    H    LINE LIMIT FOR OUTPUT FILE (OCTAL).
```

FURTHER INFORMATION ABOUT THE CHIP INPUT LINE IS GIVEN IN SECTION 204.3.1, PAGE 10.

```
 2.     USER TYPES:      CHIP / TV
        CHIP RESPONDS:   NOV VERSION. TYPE HELP OR INPUT LINE.
                         OK

        USER MAY NOW TYPE ONE OF THE FOLLOWING:
        A. USER TYPES:        LINEFEED
           CHIP RESPONDS:     NOV VERSION. TYPE HELP OR INPUT LINE.
                              OK

        B. USER TYPES:        HELP
           CHIP RESPONDS:     DESCRIPTION OF INPUT LINE.
                              OK

        C. USER TYPES:        END
           SYSTEM RESPONDS:   ALL DONE

        D. USER TYPES:        (I) A,B,C,D,E,F,G,H     [AS ABOVE]
```

204.7. DESCRIPTION OF PUBLIC FILE /CHIP/

ONCE -CHIP- HAS BEEN STARTED, IT PROCEEDS AS FOLLOWS:

1. GETS INPUT FROM THE EXECUTE LINE.
2. CREATES OR OPENS /+CHIP/.
3. CHANGES PROBLEM PROGRAM NAME TO /+CHIP/.
4. CHECKS FIRST WORD OF INPUT FOR A LINEFEED, 'END', OR 'HELP' AND RESPONDS AS DESCRIBED ABOVE. OTHERWISE IT ASSUMES THE INPUT LINE HAS BEEN GIVEN.
5. CHECKS INPUT FOR PRIVATE LIBRARY NAME AND CHECKS FOR DROP OUT OF ANY OF THE OTHER ARGUMENTS OF THE INPUT LINE.
6. CHECKS ARGUMENT 'B' FOR A FILE NAME OR FOR 'OLP' OR 'HSP'. IF NO FILE NAME IS GIVEN, THE OUTPUT FILE NAME IS /OUTPUT/. IF 'OLP' OR 'HSP' IS GIVEN, A FILE NAME IS GENERATED FROM A CLOCK READING AND A LEADING P OR H RESPECTIVELY. IF A FILE NAME IS GIVEN, -CHIP- ASSIGNS THAT NAME TO THE OUTPUT FILE.
7. CONVERTS ASCII ARGUMENTS TO DISPLAY CODE *(18)*.
8. CHECKS FOR DROPOUT OF COMPILER LENGTH. IF NOT GIVEN, THE COMPILER LENGTH IS ASSUMED TO BE 46000 OCTAL. IF GIVEN, IT IS CONVERTED FROM DISPLAY CODE TO BINARY.
9. ADDS 1130 WORDS TO THE COMPILER LENGTH IF THERE IS NO PRIVATE LIBRARY OR ADDS 2260 WORDS TO THE COMPILER LENGTH IF THERE IS A PRIVATE LIBRARY. THE EXTRA WORDS ARE USED IN THE LOADING OF THE FILE INDICES OF THE PRIVATE LIBRARY AND/OR THE PUBLIC LIBRARY /CLIB/ BY -RUN-.
10. CHECKS FOR DROPOUT OF PROGRAM FIELD LENGTH. IF NO LENGTH IS SPECIFIED, -CHIP- ASSIGNS THE COMPILER FIELD LENGTH. IF IT IS SPECIFIED, IT IS CONVERTED FROM DISPLAY CODE TO BINARY.
11. CHECKS FOR DROPOUT OF LINE LIMIT. IF NO LIMIT IS SPECIFIED, IT ASSIGNS A LINE LIMIT OF 4500 OCTAL. IF IT IS SPECIFIED, IT IS CONVERTED FROM DISPLAY CODE TO BINARY.
12. CREATES /+RUN/ AT A SIZE EQUAL TO THE COMPILER FIELD LENGTH SPECIFIED ON THE -CHIP- INPUT LINE PLUS 1130 OR 2260 WORDS (AS SPECIFIED IN 9) PLUS 400 WORDS FOR FROST. IF THE CREATE FAILS, -CHIP- OPENS AND DESTROYS ANY EXISTING FILE /+RUN/ AND CREATES /+RUN/ AT THE PROPER LENGTH.
13. COMPUTES THE SIZE OF THE OUTPUT FILE (OCTAL LENGTH = LINE LIMIT * 14 + 2000).
14. CREATES THE OUTPUT FILE. -CHIP- WILL TRY TO OPEN, DESTROY, AND RECREATE IF THE FIRST CREATE CALL FAILS.
15. OPENS PUBLIC FILE /CLIB/ AND STORES ARGUMENTS TO BE USED BY /+RUN/.
16. COPIES COMPILER FROM /CLIB/ TO /+RUN/. (THE COMPILER EXISTS IN THE FIRST 40000 WORDS OF /CLIB/.)
17. UPDATES -CHIP- USAGE COUNTER IN FILE /GRRR/.
18. INITIALIZES /+RUN/ AS A CONTROLLEE AND SENDS A MESSAGE TO START /+RUN/.
19. /+RUN/ OPENS AND DESTROYS /+CHIP/.

----

(18) SEE APPENDIX A, PAGE 61.

204.7.2. ERROR MESSAGES DURING -CHIP- INITIATION

-CHIP- MAY SEND THE FOLLOWING ERROR MESSAGES:

1. CANNOT CREATE +CHIP
   GIVEN IF -CHIP- CANNOT CREATE OR OPEN /+CHIP/ (USUALLY NOT ENOUGH DISK SPACE AVAILABLE) OR IF THE CHANGE NAME TO /+CHIP/ FAILS.

2. CANNOT CREATE +RUN
   GIVEN IF -CHIP- FAILS TO CREATE, OPEN, OR DESTROY /+RUN/.
   IF THE CREATE FAILS, -CHIP- TRIES TO OPEN AND DESTROY /+RUN/, THEN TRIES TO RECREATE IT AT PROPER LENGTH. THE ERROR MESSAGE IS GIVEN ONLY IF THIS RECOVERY FAILS.

3. CANNOT CREATE HSP OUTPUT FILE
   GIVEN IF -CHIP- FAILS TO CREATE THE LISTABLE OUTPUT FILE, WITH RECOVERY PROCEDURES DESCRIBED ABOVE.

4. CANNOT OPEN CLIB ...
   GIVEN IF -CHIP- CANNOT GAIN ACCESS TO /CLIB/.

5. DISC PARITY ERROR. WHILE COPYING RUN. RESTART ..

6. CANNOT INITIALIZE +RUN
   CONTROLLEE /+RUN/ CANNOT BE INITIALIZED OR THE SEND A MESSAGE CALL TO START THE CONTROLLEE FAILED.

7. MISSING RT PARENS IN TTY INPUT. RETYPE ENTIRE LINE
   GIVEN IF RIGHT PARENTHESIS IS NOT USED AFTER PRIVATE LIBRARY NAME.

THE USER SHOULD TRY TO RESTART -CHIP- IF ERRORS 1-6 OCCUR. IN CASE OF ERROR 1, /+CHIP/ SHOULD BE DESTROYED BEFORE RESTARTING. IN CASE OF ERROR 7, THE INPUT LINE SHOULD BE RETYPED (NOT NECESSARY TO RESTART -CHIP-).

204.8. CHIPPEWA ASSEMBLY LANGUAGE (CLASS)

THIS SECTION IS QUOTED, WITH PERMISSION, FROM [3], PAGES 62-71.

204.8.1. INTRODUCTION TO CLASS

THE FORTRAN COMPILER -RUN-, IS CAPABLE OF PROCESSING PROGRAMS OR SUBROUTINES WRITTEN IN ASSEMBLY LANGUAGE. SUCH PROGRAMS OR SUBROUTINES MAY BE INTERMIXED WITH REGULAR FORTRAN PROGRAMS AND SUBROUTINES. EACH MUST BE ORGANIZED AS FOLLOWS:

1. HEADER CARD.
2. FORTRAN CARDS, IF ANY.
3. DECLARATION CARDS, IF ANY.
4. INSTRUCTION CARDS.
5. CONSTANT CARDS.
6. END CARD.

IT IS NOTED THAT

A) THE INSTRUCTION PORTION OF THE DECK MUST BE PRECEDED BY '0' LINES CORRESPONDING TO CONTROL WORDS, ARGUMENTS, AND AN EXIT/ENTRY LINE.

B) THE CONSTANT PORTION OF THE DECK MUST BE SEPARATED FROM THE INSTRUCTION PORTION BY A CARD WITH TWO PERIODS (..) PUNCHED IN COLUMNS 7 AND 8.

C) THE END CARD IS PUNCHED AS IN FORTRAN, I.E., END IN COLUMNS 7-9 OF A CARD.

D) CONSTANTS MAY APPEAR IN THE INSTRUCTION PORTION OF THE DECK PROVIDED THEY ARE POSITIVE AND LESS THAN 2**54.

E) A CARD WITH AN ASTERISK (*) IN COLUMN 1 MAY APPEAR ANYWHERE IN THE DECK AND IS TREATED AS A REMARK CARD.

F) A CARD WITH A PERIOD (.) IN COLUMN 1 MAY APPEAR ANYWHERE IN THE DECK AND WILL CAUSE A PAGE EJECT AT THE TIME THE PROGRAM OR SUBROUTINE IS LISTED.

204.8.2. HEADER FORMATS

EACH PROGRAM OR SUBROUTINE CODED IN ASSEMBLY LANGUAGE MUST HAVE A HEADER CARD IN ONE OF THE FOLLOWING FORMATS:

```
MACHINE PROGRAM NAME
MACHINE PROGRAM NAME (A1, ..., AN)
MACHINE SUBROUTINE NAME
MACHINE SUBROUTINE NAME (A1, ..., AN)
```

IT IS NOTED THAT

A) THE HEADER INFORMATION MUST BE PUNCHED BETWEEN COLUMN 6 AND COLUMN 73 OF EACH CARD USED.

B) UP TO 19 CONTINUATION CARDS MAY BE USED IN ANY DECLARATION, BUT AN ASTERISK (*) MUST APPEAR IN COLUMN 6 OF EACH CONTINUATION CARD.

C) IN THE FOREPART OF THE PROGRAM OR SUBROUTINE TO BE ASSEMBLED THERE MUST BE THREE 'O' LINES PLUS ONE 'O' LINE FOR EACH ARGUMENT A1, ..., AN:

```
      0       CONTROL WORD ONE
      0       CONTROL WORD TWO
A1    0       ARGUMENT 1
:     :
.     .
AN    0       ARGUMENT N
      0       EXIT/ENTRY
```

THE FIRST TWO 'O' LINES CORRESPOND TO CONTROL INFORMATION FURNISHED BY THE COMPILER, AND THE LAST 'O' LINE IS UNUSED BY A PROGRAM BUT IS THE EXIT/ENTRY LINE FOR A SUBROUTINE. THE FIRST EXECUTABLE INSTRUCTION MUST FOLLOW THE EXIT/ENTRY LINE.

D) THE ARGUMENTS A1, ..., AN ARE TREATED AS DUMMY ARGUMENTS BY THE COMPILER IN THAT THEY ARE USED ONLY TO OBTAIN AN ARGUMENT COUNT TO INSERT IN THE SECOND CONTROL WORD.

E) IF AN ASSEMBLY-LANGUAGE SUBROUTINE IS TO BE REFERENCED BY A FORTRAN PROGRAM OR SUBROUTINE, THEN THE ASSEMBLY-LANGUAGE SUBROUTINE MUST BE WRITTEN ASSUMING THAT THE ADDRESSES OF THE FIRST SIX ARGUMENTS, A1-A6, WILL BE TRANSMITTED THROUGH INDEX REGISTERS B1-B6; ARGUMENTS BEYOND THE SIXTH, A7-AN, WILL BE TRANSMITTED INTO THE LOCATIONS CORRESPONDING TO A7-AN WITHIN THE ASSEMBLY-LANGUAGE SUBROUTINES; A RETURN JUMP WILL BE MADE TO THE LOCATION FOLLOWING AN.

F) FUNCTION ROUTINES, EITHER FORTRAN-CODED, ASSEMBLY-LANGUAGE CODED, OR FROM THE SYSTEM'S LIBRARY, LEAVE RESULTS IN X6 UPON EXITING.

204.8.3. FORTRAN FORMATS

FORTRAN STATEMENTS WHICH ARE ALLOWED IN AN ASSEMBLY-LANGUAGE PROGRAM OR SUBROUTINE ARE THE FOLLOWING:

COMMON
EQUIVALENCE
DIMENSION
EXTERNAL
DATA

IDENTIFIERS APPEARING IN THE ABOVE STATEMENTS MAY BE USED IN SUBSEQUENT SYMBOLIC INSTRUCTIONS. CONTINUATION CARDS MUST CONTAIN AN ASTERISK (*) IN COLUMN 6.

204.8.4. DECLARATION FORMATS

SIX TAG-ASSOCIATING DECLARATIONS ARE ALLOWED IN ASSEMBLY LANGUAGE. THESE PROVIDE FOR ASSOCIATING ALPHANUMERIC TAGS WITH CONSTANTS, WITH REGIONS RESERVED LOCALLY OR IN COMMON, AND WITH EXTERNAL SUBROUTINES WHICH ARE SUBJECT TO REFERENCE. EXAMPLES AND EXPLANATIONS OF THESE DECLARATIONS FOLLOW:

1. CON (C1=25, C2=777B, C3=-6.54E-2)

   CAUSES THE CONSTANTS ON THE RIGHT OF THE EQUALS RELATIONS TO BE ASSEMBLED INTO A STORAGE AREA AND TAGGED WITH THE IDENTIFIERS APPEARING ON THE LEFT.

2. HOL (H1=ABCDEFGHIJ, H2=1234567890)

   CAUSES THE TEN-CHARACTER GROUPS, INCLUDING SPACES, ON THE RIGHT OF THE EQUALS RELATIONS TO BE CONVERTED TO DISPLAY CODE, PLACED INTO A STORAGE AREA, AND TAGGED WITH THE IDENTIFIERS APPEARING ON THE LEFT.

3. ABS (JJ=100, KK=100B, LL=7777B)

   CAUSES THE UNSIGNED VALUES ON THE RIGHT OF THE EQUALS RELATIONS TO BE ASSEMBLED INTO INSTRUCTIONS CONTAINING THE TAGS ON THE LEFT IN THEIR ADDRESS FIELDS.

4. RES (K1=10, K2=100B, K3=1000)

   CAUSES LOCAL BLOCK RESERVATIONS, WHERE THE NUMBER OF WORDS RESERVED IN EACH BLOCK IS THE UNSIGNED NUMBER TO THE RIGHT OF THE EQUALS RELATION AND WHERE THE BEGINNING OF EACH BLOCK IS TAGGED WITH THE IDENTIFIER ON THE LEFT.

5. COM (B1=1, B2=300, B3=205B)

   CAUSES BLANK COMMON BLOCK RESERVATIONS, WHERE THE NUMBER OF WORDS RESERVED IN EACH BLOCK IS THE UNSIGNED NUMBER TO THE RIGHT OF THE EQUALS RELATION AND WHERE THE BEGINNING OF EACH BLOCK IS TAGGED WITH THE IDENTIFIER ON THE LEFT.

6. SUB (SI=SIN, LG=LOG, OUT=OUTPTC)

   CAUSES THE SUBROUTINES WHOSE NAMES APPEAR ON THE RIGHT OF THE EQUALS RELATIONS TO BE ASSEMBLED INTO MEMORY AND TAGGED WITH THE IDENTIFIERS APPEARING ON THE LEFT.

IT IS NOTED THAT

A) EACH TAG-ASSOCIATING DECLARATION IS PUNCHED BETWEEN COLUMN 6 AND 73 OF EACH CARD USED.

B) UP TO 19 CONTINUATION CARDS MAY BE USED IN ANY DECLARATION, BUT AN ASTERISK (*) MUST APPEAR IN COLUMN 6 OF EACH CONTINUATION CARD.

C) THE OPEN PARENTHESIS (( ) FOLLOWING CON, HOL, ABS, RES, COM, OR SUB MAY BE REPLACED BY ANY SEPARATOR IF THE FINAL CLOSING PARENTHESIS ( )) IS DROPPED.

204.8.5. INSTRUCTION FORMATS (CLASS)

IN THE ASSEMBLY LANGUAGE, OPERATIONAL REGISTERS ARE DESIGNATED BY SINGLE-CHARACTER NAMES AS FOLLOWS:

S = X0	0 = B0	A = A0
T = X1	I = B1	B = A1
U = X2	J = B2	C = A2
V = X3	K = B3	D = A3
W = X4	L = B4	E = A4
X = X5	M = B5	F = A5
Y = X6	N = B6	G = A6
Z = X7	O = B7	H = A7

THE LETTER 'R' IS USED TO SPECIFY A RETURN JUMP, AND THE LETTER 'P' IS USED TO SPECIFY ALL OTHER JUMPS.

LET 'S' REPRESENT ANY OF THE LETTERS S-Z, 'I' REPRESENT ANY OF THE LETTERS I-O OR THE DIGIT 0, AND 'A' REPRESENT ANY OF THE LETTERS A-H. LET 'Q' REPRESENT A POSITIVE INTEGER LESS THAN 216 OR AN ALPHANUMERIC TAG OF 2-6 CHARACTERS. THEN THE FORMS OF ASSEMBLY-LANGUAGE INSTRUCTIONS, GROUPED ACCORDING TO FUNCTIONAL UNITS REQUIRED FOR EXECUTION, ARE AS FOLLOWS:

SYMBOLIC FORM	MACHINE FORM	EXAMPLE
0	00XXX	0
R=Q	01XXK	R=TAG
P=Q+I	02iXK	P=TAG+N
P=Q,S=0	030jK	P=TAG,T=0
P=Q,S/0	031JK	P=TAG,U/0
P=Q,S)0	032JK	P=TAG,V)0
P=Q,S(0	033JK	P=TAG,W(0
P=Q,S.I	034JK	P=TAG,X.I
P=Q,S.O	035JK	P=TAG,Y.O
P=Q,S.D	036JK	P=TAG,Z.D
P=Q,S.N	037JK	P=TAG,S.N
P=Q,I=I	04iJK	P=TAG,J=K
P=Q,I/I	05iJK	P=TAG,L/M
P=Q,I)I	06iJK	P=TAG,N)O
P=Q,I(I	07iJK	P=TAG,I(O
S=S	10ijx	Y=V
S.L=S*S	11ijk	T.L=W*X
S.L=S+S	12ijk	V.L=Y+Z

204.8. CHIPPEWA ASSEMBLY LANGUAGE (CLASS)

S.L=S-S	13ijk	Y.L=U-Y
S=-S	14ijk	Z=-W
S.C=S*S	15ijk	W.C=T*U
S.C=S+S	16ijk	X.C=Y+W
S.C=S-S	17ijk	Y.C=S-Z
S=S(Q)	20ijk	T=T(24)
S=S(-Q)	21ijk	U=U(-10)
S=S(I)	22ijk	Y=X(N)
S=S(-I)	23ijk	W=W(-M)
S,I=S-	24ijk	X,J=W-
S,I=S+	25ijk	Y,K=Z+
S,I=S.	26ijk	Z,O=U.
S=I,S.	27ijk	T=K,Y.
S=*Q	43ijk	Y=*12
S.N=S+S	30ijk	T.N=U+Y
S.N=S-S	31ijk	U.N=X-Z
S.D=S+S	32ijk	Y.D=Y+Y
S.D=S-S	33ijk	W.D=S-U
S.R=S+S	34ijk	X.R=Y+Z
S.R=S-S	35ijk	Y.R=Z-T
S.I=S+S	36ijk	T.I=U+X
S.I=S-S	37ijk	Z.I=Y-Z
S.N=S*S	40ijk	S.N=X*X
S.R=S*S	41ijk	Y.R=Y*Y
S.D=S*S	42ijk	T.D=T*U
S.N=S/S	44ijk	Y.N=T/X
S.R=S/S	45ijk	Z.R=X/W
$	46XXX	$
S=*S	47ijk	T=*W
S=(A+Q)	50ijK	T=(C+TAG)
S=(I+Q)	51ijK	U=(J+100)
S=(S+Q)	52ijK	Z=(T+30B)
S=(S+I)	53ijk	T=(T+J)
S=(A+I)	54ijk	U=(B+K)
S=(A-I)	55ijk	Y=(C-N)
S=(I+I)	56ijk	W=(M+N)
S=(I-I)	57ijk	X=(L-K)
I=A+Q	60ijK	J=H+TAG
I=I+Q	61ijK	K=L+10
I=S+Q	62ijK	L=L+55B
I=S+I	63ijk	M=T+J

204.8. CHIPPEWA ASSEMBLY LANGUAGE (CLASS)

```
I=A+I          64ijk      N=G+K
I=A-I          65ijk      O=A-L
I=I+I          66ijk      I=I+J
I=I-I          67ijk      J=K-M
S=A+Q          70ijK      T=G+TAG
S=I+Q          71ijK      U=K+5
S=S+Q          72ijK      V=L+15B
S=S+I          73ijk      W=X+J
S=A+I          74ijk      X=A+K
S=A-I          75ijk      Y=C-L
S=I+I          76ijk      Z=M+I
S=I-I          77ijk      S=N-O
```

IT IS NOTED THAT

A) THE ARITHMETIC MODE INDICATORS L, C, N, D, R, AND I MAY IMMEDIATELY FOLLOW A RESULT REGISTER NAME, I.E., THE PERIOD (.) IN THESE CASES IS OPTIONAL.

```
TL=X*Y
UC=Y+Y
VN=T/W
WD=X+Y
XR=S-T
```

B) IN THE INSTRUCTIONS 02 AND 50-77 EITHER TERM MAY BE DROPPED, IN WHICH CASE A 0 DESIGNATION IS ASSEMBLED.

```
P=K
P=TAG
T=(J)
J=15B
U=K
```

C) IN THE INSTRUCTIONS 50-54, 60-64, AND 70-74 THE TERMS MAY BE INTERCHANGED UNLESS Q IS A CONSTANT.

```
T=(TAG+L)
M=K+B
U=L+X
```

D) IN THE INSTRUCTIONS 50-52, 60-62, AND 70-72 THE PLUS SIGN (+) MAY BE REPLACED BY A MINUS SIGN (-) IF Q IS A CONSTANT.

```
X=(I-30)
J=K-55B
Y=-1
```

204.8. CHIPPEWA ASSEMBLY LANGUAGE (CLASS)

E) IN THE INSTRUCTIONS 51, 61, AND 71 THE RIGHT MEMBER MAY BE AN INDICATED SUM OR DIFFERENCE OF A TAG AND A CONSTANT, IN WHICH CASE THE CONSTANT MUST FOLLOW THE TAG.

```
W=(TAG-35)
K=TAG+1
U=TAG+100B
```

F) IN THE INSTRUCTION 51 THE PARENTHIZED QUANTITY MAY BE A CONSTANT, REPRESENTED IN CONVENTIONAL FORTRAN FORM, ONLY IF THE RESULT REGISTER IS TO RECEIVE THE MACHINE VERSION OF THAT CONSTANT; IN THIS CASE THE ADDRESS OF THE CONVERTED NUMBER IS ASSEMBLED INTO THE INSTUCTION.

```
T=(-1.5E-6)
U=(4755051604554 7B)
```

G) IF IT IS DESIRED TO HAVE Q CORRESPOND TO AN OCTAL INTEGER, THEN THE DIGITS IN THE NUMBER MUST BE TRAILED BY A B.

H) ALTERNATE FORMS FOR CERTAIN INSTRUCTIONS ARE:

S=S.S	11ijk
S=S$S	12ijk
S=-S.S	15ijk
S=-S$S	16ijk
S=S+S	36ijk
S=S-S	37ijk
S=S*S	40ijk
S=S/S	44ijk
A=A+Q	50ijK
A=I+Q	51ijK
A=S+Q	52ijK
A=S+I	53ijk
A=A+I	54ijk
A=A-I	55ijk
A=I+I	56ijk
A=I-I	57ijk

I) EACH INSTRUCTION MUST BE PUNCHED BETWEEN COLUMN 6 AND COLUMN 73 OF A CARD; NO BLANKS ARE PERMITTED WITHIN THE INSTRUCTION CODE.

J) A COMMENT MAY FOLLOW ANY INSTRUCTION CODE, BUT A LEAST ONE BLANK MUST SEPARATE IT FROM THE INSTRUCTION.

K) AN ALPHANUMERIC LOCATION TAG OF 2-6 CHARACTERS MAY BE PUNCHED IN COLUMNS 1-6 OF A CARD CONTAINING AN INSTRUCTION; NO BLANKS ARE PERMITTED WITHIN THE TAG.

L) A PLUS SIGN (+) IN A LOCATION FIELD WILL FORCE THE CORRESPONDING INSTRUCTION TO THE HIGH ORDER POSITIONS OF A NEW WORD.

M) THE INSTRUCTION 00 IS ASSEMBLED AS A FULL ZERO WORD.

204.8.6. CONSTANT FORMATS

DECIMAL CONSTANTS IN STANDARD FORTRAN NOTATION MAY BE SPECIFIED IN ASSEMBLY-LANGUAGE. OCTAL CONSTANTS MAY BE SPECIFIED BY PLACING A 'B' AFTER THE DIGITS OF THE NUMBER.

IT IS NOTED THAT

A) A CONSTANT MUST BE PUNCHED BETWEEN COLUMN 6 AND COLUMN 73 OF A CARD; NO BLANKS ARE PERMITTED WITH THE CONSTANT SPECIFICATION.

```
            -100.0
            500
            100B
            +15.64E-3
```

B) AN ALPHANUMERIC LOCATION TAG OF 2-6 CHARACTERS MAY BE PUNCHED IN COLUMNS 1-6 OF A CARD CONTAINING A CONSTANT; NO BLANKS ARE PERMITTED WITHIN THE TAG.

```
CON1        -150.0
CON2        3.6E10
```

C) BLOCK RESERVATIONS OF ZERO WORDS, TAGGED OR UNTAGGED, MAY BE MADE BY ENCLOSING THE NUMBER OF WORDS TO BE RESERVED IN PARENTHESES; THE PARENTHESIZED QUANTITY MUST APPEAR BETWEEN COLUMN 6 AND COLUMN 73 OF A CARD; IF A LOCATION TAG APPEARS ON THE SAME CARD, THEN IT WILL BE ASSOCIATED WITH THE FIRST WORD OF THE BLOCK.

```
BK1         (100)
BK2         (200B)
```

204.9. ASCENTF ASSEMBLY LANGUAGE

THIS SECTION IS QUOTED, WITH PERMISSION, FROM [3], PAGES 72-73, 84-86.

204.9.1. CARD FORMATS

THE FORTRAN COMPILER, -RUN-, IS CAPABLE OF PROCESSING PROGRAMS OR SUBROUTINES WRITTEN IN A SUBSET OF ASCENT ASSEMBLY LANGUAGE. SUCH PROGRAMS OR SUBROUTINES MAY BE INTERMIXED WITH REGULAR FORTRAN PROGRAMS AND SUBROUTINES. EACH MUST BE ORGANIZED AS FOLLOWS:

1. HEADER CARDS.
2. FORTRAN CARDS, IF ANY.
3. INSTRUCTION CARDS.
4. CONSTANT CARDS.
5. END CARD.

IT IS NOTED THAT

A) THE INSTRUCTION PORTION OF THE DECK MUST BE PRECEDED BY LINES OF CODING WHICH PRODUCE '0' WORDS CORRESPONDING TO CONTROL WORDS, ARGUMENT WORDS, AND AN EXIT/ENTRY WORD.

B) THE INSTRUCTION PORTION OF THE DECK MAY CONTAIN BSS, BSSZ, AND EQU CARDS. THE ADDRESS FIELD OF ANY SUCH CARD MAY CONTAIN ONLY A SINGLE CONSTANT. BSS AND BSSZ CARDS PRODUCE ZERO REGIONS.

C) THE CONSTANT PORTION OF THE DECK MAY CONTAIN BSS, BSSZ, EQU, DPC, BCD, AND CON CARDS. THE ADDRESS FIELDS OF THESE CARDS MAY CONTAIN ONLY A SINGLE CONSTANT OR TEN-CHARACTER STRING OF THE FORM *ABCDEFGHIJ*. DPC AND BCD CHARACTER STRINGS ARE REDUCED TO DISPLAY CODE.

D) THE CONSTANT PORTION OF THE DECK MUST BE SEPARATED FROM THE INSTRUCTION PORTION BY A CARD WITH TWO PERIODS (..) PUNCHED IN COLUMNS 7 AND 8.

E) THE END CARD IS PUNCHED AS IN FORTRAN, I.E., 'END' APPEARS BETWEEN COLUMN 6 AND 73 OF THE CARD.

F) FORTRAN C-TYPE COMMENT CARDS ARE NOT PERMITTED.

204.9. ASCENTF ASSEMBLY LANGUAGE

G) FORTRAN CARDS MAY CONTAIN COMMON, EQUIVALENCE, DIMENSION, EXTERNAL, OR DATA STATEMENTS. IDENTIFIERS APPEARING IN THE STATEMENTS MAY BE USED IN SUBSEQUENT SYMBOLIC INSTRUCTIONS. CONTINUATION CARDS MUST CONTAIN AN ASTERISK (*) IN COLUMN 6.

H) THE HEADER CARD MUST BE IN ONE OF THE FOLLOWING FORMATS:

```
ASCENTF PROGRAM NAME
ASCENTF PROGRAM NAME (A1,...,AN)
ASCENTF SUBROUTINE NAME
ASCENTF SUBROUTINE NAME (A1,...,AN)
```

THE HEADER INFORMATION MUST BE PUNCHED BETWEEN COLUMN 6 AND COLUMN 73 OF EACH CARD USED. CONTINUATION CARDS MAY BE USED, BUT MUST BE DESIGNATED BY AN ASTERISK (*) IN COLUMN 6.

I) INSTRUCTION FORMATS ARE AS DESCRIBED IN ASCENT PROGRAMMING MANUALS, BUT WITH CERTAIN RESTRICTIONS. AN ADDRESS FIELD MAY CONTAIN AN INDICATED SUM OF A TAG AND CONSTANT, BUT NOT A SUM OR DIFFERENCE OF TWO TAGS. LOCATION TAGS MAY START IN COLUMN 1, BUT MAY NOT EXTEND BEYOND COLUMN 6. INSTRUCTIONS MAY START ANYWHERE BEYOND COLUMN 6, BUT NO CARD MAY CONTAIN MORE THAN ONE INSTRUCTION. THE PS INSTRUCTION CAUSES ASSEMBLY OF A FULL ZERO WORD.

J) LOCATION TAGS ASSOCIATED WITH PSEUDO-OPERATIONS MAY START IN COLUMN 1, BUT MAY NOT EXTEND BEYOND COLUMN 6.

K) DOUBLE-PRECISION AND COMPLEX 'LITERAL CONSTANTS' ARE NOT ACCEPTED.

L) A MINUS SIGN (-) IN A LOCATION FIELD IS NOT ALLOWED.

M) AN ASTERISK (*) IN AN ADDRESS FIELD IS NOT ALLOWED.

204.9. ASCENTF ASSEMBLY LANGUAGE

204.9.2. INSTRUCTION CODES (ASCENTF)

OP	MNEMONIC	ADDRESS	REMARKS
			BRANCH UNIT
00	PS		.PROGRAM STOP
01	RJ	K	.RETURN JUMP TO K
02	JP	Bi+ K	.JUMP TO Bi+K
030	ZR	Xi K	.JUMP TO K IF Xi=0
031	NZ	Xi K	.JUMP TO K IF Xi≠0
032	PL	Xi K	.JUMP TO K IF Xi=PLUS (POSITIVE)
033	NG	Xi K	.JUMP TO K IF Xi=NEGATIVE
034	IR	Xi K	.JUMP TO K IF Xi IS IN RANGE
035	OR	Xi K	.JUMP TO K IF Xi IS OUT OF RANGE
036	DF	Xi K	.JUMP TO K IF Xi IS DEFINITE
037	ID	Xi K	.JUMP TO K IF Xi IS INDEFINITE
04	EQ	BiBj K	.JUMP TO K IF Bi=Bj
04	ZR	Bi K	.JUMP TO K IF Bi=0 OR B0
05	NE	BiBj K	.JUMP TO K IF Bi≠Bj
05	NZ	Bi K	.JUMP TO K IF Bi≠B0
06	GE	BiBj K	.JUMP TO K IF Bi≥Bj
06	PL	Bi K	.JUMP TO K IF Bi≥B0
07	LT	BiBj K	.JUMP TO K IF Bi<Bj
07	NG	Bi K	.JUMP TO K IF Bi<B0
			BOOLEAN UNIT
10	BXi	Xj	.TRANSMIT Xj TO Xi
11	BXi	Xj*Xk	.LOGICAL PRODUCT OF Xj AND Xk TO Xi
12	BXi	Xj+Xk	.LOGICAL SUM OF Xj AND Xk TO Xi
13	BXi	Xj-Xk	.LOGICAL DIFFERENCE OF Xj AND Xk TO Xi
14	BXi	-Xk	.TRANSMIT THE COMP. OF Xj TO Xi
15	BXi	-Xk*Xj	.LOGICAL PRODUCT OF Xj AND Xk COMP. TO Xi
16	BXi	-Xk+Xj	.LOGICAL SUM OF Xj AND Xk COMP. TO Xi
17	BXi	-Xk-Xj	.LOGICAL DIFFERENCE OF Xj AND Xk COMP. TO Xi

204.9. ASCENTF ASSEMBLY LANGUAGE

SHIFT UNIT

20	LXi	jk	.LEFT SHIFT Xi, jk PLACES
21	AXi	jk	.ARITHMETIC RIGHT SHIFT Xi, jk PLACES
22	LXi	Bj,Xk	.LEFT SHIFT Xi NOMINALLY Bi PLACES
23	AXi	Bj,Xk	.ARITHMETIC RIGHT SHIFT Xi NOMINALLY Bi PLACES
24	NXi	Bj Xk	.NORMALIZE Xk IN Xi AND Bj
25	ZXi	Bj Xk	.ROUND AND NORMALIZE Xk IN Xi AND Bj
26	UXi	Bj Xk	.UNPACK Xk TO Xi AND Bj
27	PXi	Bj Xk	.PACKS Xi FROM Xk AND Bj
43	MXi	jk	.FORM MASK IN Xi,jk BITS

ADD UNIT

30	FXi	Xj+Xk	.FLOATING SUM OF Xj AND Xk TO Xi
31	FXi	Xj-Xk	.FLOATING DIFFERENCE Xj AND Xk TO Xi
32	DXi	Xj+Xk	.FLOATING DP SUM OF Xj AND Xk TO Xi
33	DXi	Xj-Xk	.FLOATING DP DIFFERENCE OF Xj AND Xk TO Xi
34	RXi	Xj+Xk	.ROUND FLOATING SUM OF Xj AND Xk TO Xi
35	RXi	Xj-Xk	.ROUND FLOATING DIFFERENCE OF Xj AND Xk TO Xi

LONG ADD UNIT

36	IXi	Xj+Xk	.INTEGER SUM OF Xj AND Xk TO Xi
37	IXi	Xj-Xk	.INTEGER DIFFERENCE OF Xj AND Xk TO Xi

MULTIPLY UNIT

40	FXi	Xj*Xk	.FLOATING PRODUCT OF Xj AND Xk TO Xi
41	RXi	Xj*Xk	.ROUND FLOATING PRODUCT OF Xj AND Xk TO Xi
42	DXi	Xj*Xk	.FLOATING DP PRODUCT OF Xj AND Xk TO Xi

204.9. ASCENTF ASSEMBLY LANGUAGE

DIVIDE UNIT

44	FXi	Xj/Xk	.FLOATING DIVIDE Xj BY Xk TO Xi
45	RXi	Xj/Xk	.ROUND FLOATING DIVIDE Xj BY Xk TO Xi
46	NO		.NO OPERATION
47	CXi	Xj	.COUNT THE NUMBER OF 1'S IN Xj TO Xi

INCREMENT UNIT

50	SAi	Aj+K	.SET Ai TO Aj+K
50	SAi	Aj-K	.SET Ai TO Aj+ COMP. OF K
51	SAi	Bj+K	.SET Ai TO Bj+K
51	SAi	Bj-K	.SET Ai TO Bj+ COMP. OF K
52	SAi	Xj+K	.SET Ai TO Xj+K
52	SAi	Xj-K	.SET Ai TO Xj+ COMP. OF K
53	SAi	Xj+Bk	.SET Ai TO Xj+Bk
54	SAi	Aj+Bk	.SET Ai TO Aj+Bk
55	SAi	Aj-Bk	.SET Ai TO Aj-Bk
56	SAi	Bj+Bk	.SET Ai TO Bj+Bk
57	SAi	Bj-Bk	.SET Ai TO Bj-Bk
60	SBi	Aj+K	.SET Bi TO Aj+K
60	SBi	Aj-K	.SET Bi TO Aj+ COMP. OF K
61	SBi	Bj+K	.SET Bi TO Bj+K
61	SBi	Bj-K	.SET Bi TO Bj+ COMP. OF K
62	SBi	Xj+K	.SET Bi TO Xj+K
62	SBi	Xj-K	.SET Bi TO Xj+ COMP. OF K
63	SBi	Xj+Bk	.SET Bi TO Xj+Bk
64	SBi	Aj+Bk	.SET Bi TO Aj+Bk
65	SBi	Aj-Bk	.SET Bi TO Aj-Bk
66	SBi	Bj+Bk	.SET Bi TO Bj+Bk
67	SBi	Bj-Bk	.SET Bi TO Bj-Bk
70	SXi	Aj+K	.SET Xi TO Aj+K
70	SXi	Aj-K	.SET Xi TO Aj+ COMP. OF K
71	SXi	Bj+K	.SET Xi TO Bj+K
71	SXi	Bj-K	.SET Xi TO Bj+ COMP. OF K
72	SXi	Xj+K	.SET Xi TO Xj+K
72	SXi	Xj-K	.SET Xi TO Xj+ COMP. OF K
73	SXi	Xj+Bk	.SET Xi TO Xj+Bk
74	SXi	Aj+Bk	.SET Xi TO Aj+Bk
75	SXi	Aj-Bk	.SET Xi TO Aj-Bk
76	SXi	Bj+Bk	.SET Xi TO Bj+Bk
77	SXi	Bj-Bk	.SET Xi TO Bj-Bk

204.9. ASCENTF ASSEMBLY LANGUAGE

204.9.3. PSEUDO-OPERATIONS (CLASS AND ASCENTF)

PROGRAM: DEFINES THE JOB TO BE A PROGRAM; THE SYMBOL IN THE ADDRESS FIELD IS THE NAME OF THE PROGRAM AS REFERENCED BY THE SYSTEM.

SUBROUTINE: DEFINES THE JOB TO BE A SUBROUTINE AND SETS RELOCATABLE BITS FOR LATER USE BY THE FORTRAN COMPILER OR MACHINE PROGRAM. THE SYMBOL IN THE ADDRESS FIELD IS THE NAME USED TO REFERENCE THE SUBROUTINE.

END: LAST CARD OF A PROGRAM OR SUBROUTINE.

BSS: ADDRESS FIELD DEFINES THE LENGTH OF THE BLOCK RESERVATION. ADDRESS FIELD MAY BE INTEGER CONSTANT OR SYMBOLIC CONSTANT.

EQU: EQUIVALENCES A SYMBOL TO ANOTHER SYMBOL OR A CONSTANT.

DPC: ALLOWS THE ENTRY OF CONSOLE DISPLAY CODES, THE LENGTH BEING SPECIFIED BY A 2 DIGIT INTEGER, OR THE CODES BETWEEN *'S ARE ENTERED.

BCD: CONVERTS THE CHARACTERS ENCLOSED BY THE ASTERISKS OR CONVERTS THE CHARACTERS SPECIFIED BY THE BEGINNING 2 DIGIT INTEGER.

CON: CONVERTS EACH TERM TO A 60 BIT CONSTANT.

EJECT: EJECTS THE LISTING TO THE TOP OF THE NEXT PAGE.

SPACE: SPACES THE NUMBER OF LINES SPECIFIED BY THE ADDRESS FIELD.

REFERENCES

[1] '6000 SERIES CHIPPEWA FORTRAN MANUAL.' CDC PUBLICATION NO. 60132700, 1965.

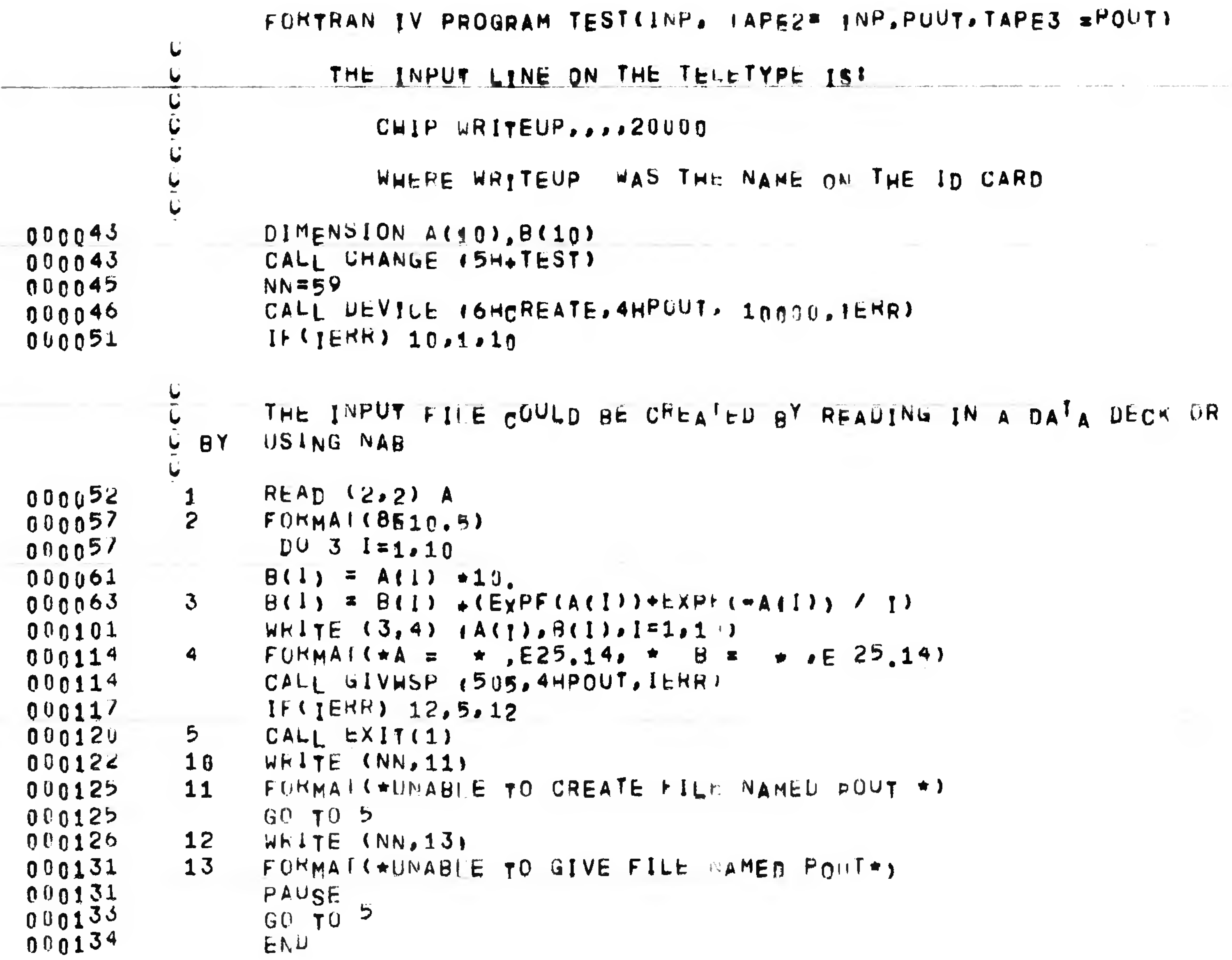
[2] '6000 SERIES CHIPPEWA SYSTEM MANUAL.' CDC PUBLICATION NO. 60134400, 1965.

[3] 'CONTROL DATA 6600 CHIPPEWA OPERATING SYSTEM.' CDC PUBLICATION NO. 60124500, 1965.

APPENDIX A. 6600 DISPLAY CODE

	0	1	2	3	4	5	6	7
0	EM	A	B	C	D	E	F	G
1	H	I	J	K	L	M	N	O
2	P	Q	R	S	T	U	V	W
3	X	Y	Z	0	1	2	3	4
4	5	6	7	8	9	+	-	*
5	/	(	)	$	=		,	.
6	AT	'	''	NBR	%	&	;	<
7	>	?	:	[	\	]	↑	←

6600 DISPLAY CODE



```
                    FORTRAN IV PROGRAM TEST(INP, TAPE2= INP,POUT,TAPE3 =POUT)
               C
               C        THE INPUT LINE ON THE TELETYPE IS:
               C
               C           CHIP WRITEUP,,,,20000
               C
               C           WHERE WRITEUP  WAS THE NAME ON THE ID CARD
               C
000043              DIMENSION A(10),B(10)
000043              CALL CHANGE (5H+TEST)
000045              NN=59
000046              CALL DEVICE (6HCREATE,4HPOUT, 10000,IERR)
000051              IF(IERR) 10,1,10
               C
               C    THE INPUT FILE COULD BE CREATED BY READING IN A DATA DECK OR
               C BY USING NAB
               C
000052          1   READ (2,2) A
000057          2   FORMAT(8E10.5)
000057               DO 3 I=1,10
000061              B(I) = A(I) *10.
000063          3   B(I) = B(I) +(EXPF(A(I))+EXPF(-A(I)) / I)
000101              WRITE (3,4) (A(I),B(I),I=1,10)
000114          4   FORMAT(*A =  * ,E25.14, *  B =  * ,E 25.14)
000114              CALL GIVHSP (505,4HPOUT,IERR)
000117              IF(IERR) 12,5,12
000120          5   CALL EXIT(1)
000122          10  WRITE (NN,11)
000125          11  FORMAT(*UNABLE TO CREATE FILE NAMED POUT *)
000125              GO TO 5
000126          12  WRITE (NN,13)
000131          13  FORMAT(*UNABLE TO GIVE FILE NAMED POUT*)
000131              PAUSE
000133              GO TO 5
000134              END
```

```
rTEST
nFUNCTION  ASSIGNMENTS
nSTATEMENT ASSIGNMENTS
 1      -  000053    10      =  000123    12      =  000127    5       -  000121
nVARIABLE  ASSIGNMENTS
 A      -  000224    B       =  000236    1       =  000252    IERR    -  000251
 NN     -  000250
nSTART OF CONSTANTS
 000137
nSTART OF TEMPORARIES
 000207
nSTART OF INDIRECTS
 000221
```

```
0SUBROUTINE ASSIGNMENTS
 CHANGE -  000257    DEVICE =  000472    EXP    =  001043    OUTPTC =  001143
 GIVHSP -  003454    GOB    =  004377    INPUTC =  004416    EXIT   =  006544
 PAUSE  -  006620    END    =  006652    LOC    =  006664    ISL    =  006672
 SWITCH -  006702    ISR    =  006776    EMPTY  =  007004    STO    =  007116
 DELAY  -  007125
0BLOCK  ASSIGNMENTS
 O1100  -  003445
0BUFFER ASSIGNMENTS
 INP    -  015767    TAPE2  =  015767    FOUT   =  013756    TAPE3  =  013756
0LOCAL  LENGTH
 007131
0COMMON LENGTH
 000000
0BUFFER LENGTH
 004022
0UNUSED PROGRAM  SPACE
 004500
0UNUSED COMPILER SPACE
 003500
```

DISTRIBUTION

RERUN-6-6-73

TID File	200

NOTICE

"This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Atomic Energy Commission, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights."