CS-1 COMPILING SYSTEM

OPERATING INSTRUCTIONS

AND

USE OF SPECIAL OPERATORS

MAY 68

Written By: R. J. Homyak
R. J. Homyak

Approved By: L. W. Buhr
L. W. Buhr

TABLE OF CONTENTS

INTRODUCTION 1

CS-1 COMPILING SYSTEMS (Illus.)..... 2

PREPARATIONS FOR A CS-1 COMPILE 3

INPUT SPECIFICATIONS 5

OPERATING THE COMPILER 12

SPECIAL CS-1 COMPILER STARTING ADDRESSES 14

OUTPUTS 15

SPECIAL CAPABILITIES 17

USE OF L1-MERGE OPERATOR 19

USE OF THE DATAX OPERATOR 20A

USE OF THE DATAF OPERATOR 20C

FORMAT II MNEMONIC STATEMENTS 21

APPENDIX A--DESCRIPTION OF 1230 FLOATING POINT AND SPECIAL OPERATORS 29

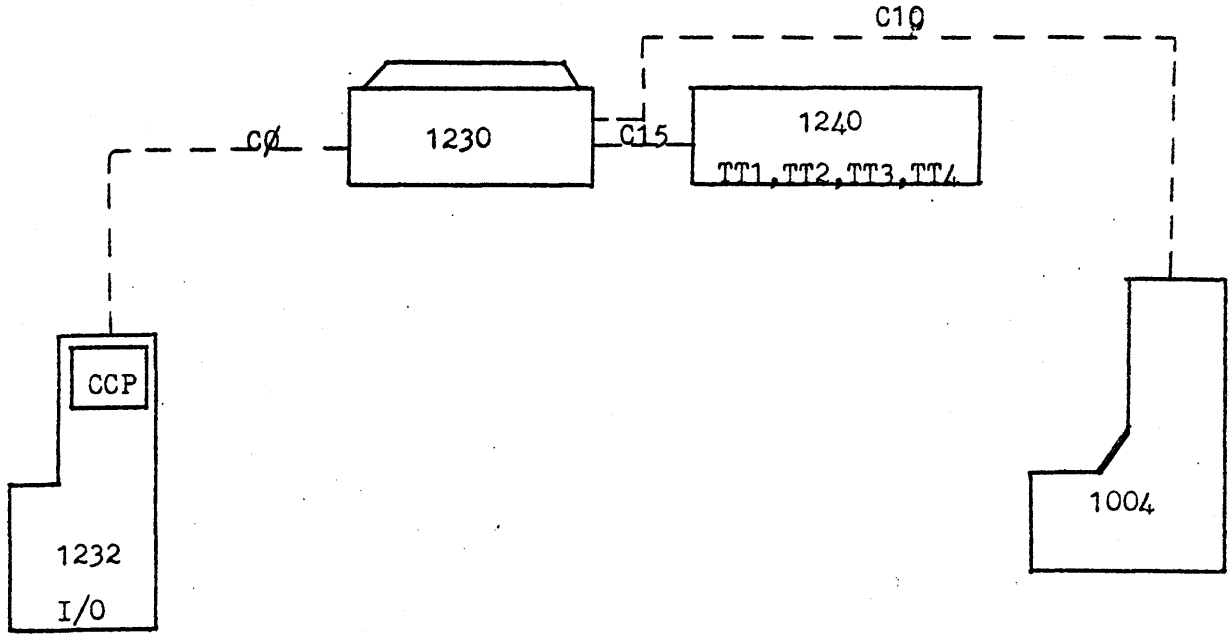
INTRODUCTION

The CS-1 compiler contains the following features:

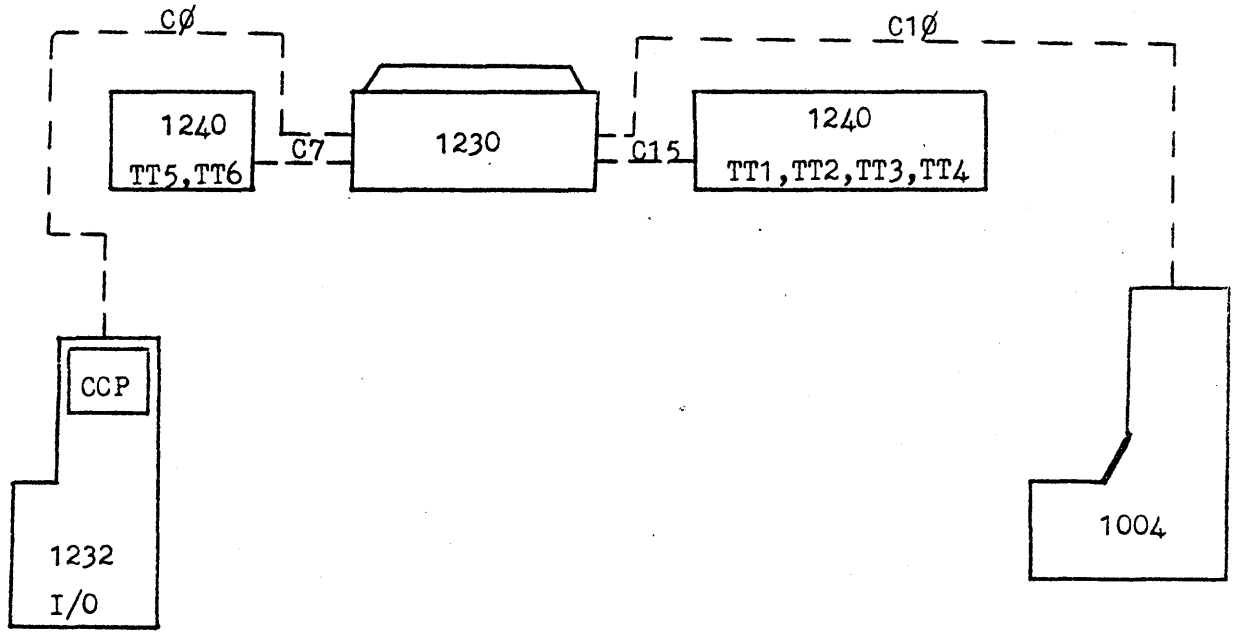
- Computer-oriented and problem-oriented language processing with refined generations.
- Large program capacity.
- Compiler tables on magnetic tape.
- Program correction upon input of a compile run (L1-Merge Operator).
- Control of all input specifications
- Generation for Floating Point, Double Arithmetic, Normalize, and Special RTC operators.

CS-1 COMPILING SYSTEMS

TWO SCRATCH
SYSTEM



THREE SCRATCH
SYSTEM



NOTE: 1299 SWITCH USED TO UTILIZE
1240 TWO HANDLER

PREPARATIONS FOR A CS-1 COMPILE

*TAPE TRANSPORT REQUIREMENTS:

The CS-1 Compiler program is stored on magnetic tape and requires a minimum of two additional scratch tapes to compile with a fourth scratch tape assigned for input and/or output. The CS-1 Compiler program will accept a maximum system of three additional scratch tapes to compile with a fourth and fifth scratch tape assigned for input and/or output. The two or three scratch tapes required are used by the CS-1 compiler for temporary table storage areas. The additional tape transports are required if the LØ program input is via mag tape, the printer output is off-line, or if a 40, 41, 226, or 46 output is requested.

The minimal compiling system is described as being a "Two Scratch System." When the two scratch system is utilized, a third scratch tape is available for input and/or output. Under the present input Control System, a two scratch compilation assumes that the compiler tape is on logical transport one, that two scratch tapes are available on logical transports two and three, and that any mag tape input (if used) is on logical transport four. Transport four may be used for any mag tape output after the compilation is complete.

The maximum compiler system available is described as being a "Three Scratch System." When the three scratch system is utilized, a fifth and sixth scratch tape is available for input and/or output. Under the present input control system, a three scratch compilation assumes that the compiler tape is on logical transport one, that three scratch tapes are available on logical transports two, three, and four, and that any mag tape input (if used) is on logical transport five. Transports four, five, and six may be used for any mag tape output after the compilation is complete.

* Transports five and six are actually transports one and two (channel seven) respectively of the 1240 two handler.

MOUNTING THE MAGNETIC TAPES:

Before a CS-1 Compile can be undertaken magnetic tapes must be mounted according to the following system requirements:

Two Scratch System:

Logical Transport 1 - CS-1 Compiler (write enabled)
Logical Transport 2 - Scratch tape (write enabled)
Logical Transport 3 - Scratch tape (write enabled)
Logical Transport 4 - Input tape (if used)

Three Scratch System:

Logical Transport 1 - CS-1 Compiler (no write enable)
Logical Transport 2 - Scratch tape (write enabled)
Logical Transport 3 - Scratch tape (write enabled)
Logical Transport 4 - Scratch tape (write enabled)
Logical Transport 5 - Input tape (if used)
Logical Transport 6 - Output tape (if used)

NOTE: Logical transports 5 and 6 are logical transports 1 and 2 of the 1240 two handler.

INPUT SPECIFICATIONS:

Input for a CS-1 compile may be on paper tape, magnetic tape, cards, or any combination of the three. All inputs to the CS-1 compiler program are specified by control cards. The control cards satisfy the type of Scratch Tape system to be utilized by the compiler program, the types of inputs to be processed in the current compilation, and the number of inputs to be processed concurrently. All control cards must adhere to the following format specifications:

Column	Punch Required	Number Designation
5	2	Two Scratch System
5	3	Three Scratch System
5	4	Three Scratch System with Printer Off-Line

When a four (4) is punched in column five (5) the CS-1 Compiler program assumes a Three Scratch configuration and assigns the Off-Line output to logical Transport Six (6). Printer Off-Line is not allowed on a Two Scratch system.

Column	Punch Required	Number Designation
10	1	Paper Tape (26) Input
10	2	Card Input
10	3	Mag Tape (46) Input
10	4	Mag Tape (226) Input
10	5	Mag Tape (8-1) Input

When a one (1), three (3), or four (4) is punched in column 10 the user should supplement his input with an Allocation card deck since the 26, 46, and 226 outputs do not contain any compiler control cards such as C-CONTROL and OUTPUTS or ALLOCATION cards.

Column	Punch Required	Number Designation
15	*	Indicates Multiple Input
80	*	Indicates Control Card

NOTE: A two (2), three (3), or four (4) need be punched in Column 5 on the initial control card only. The type of system specified will then be maintained until the compiler is reloaded or until P is set to 1403 for re-assignment of tapes.

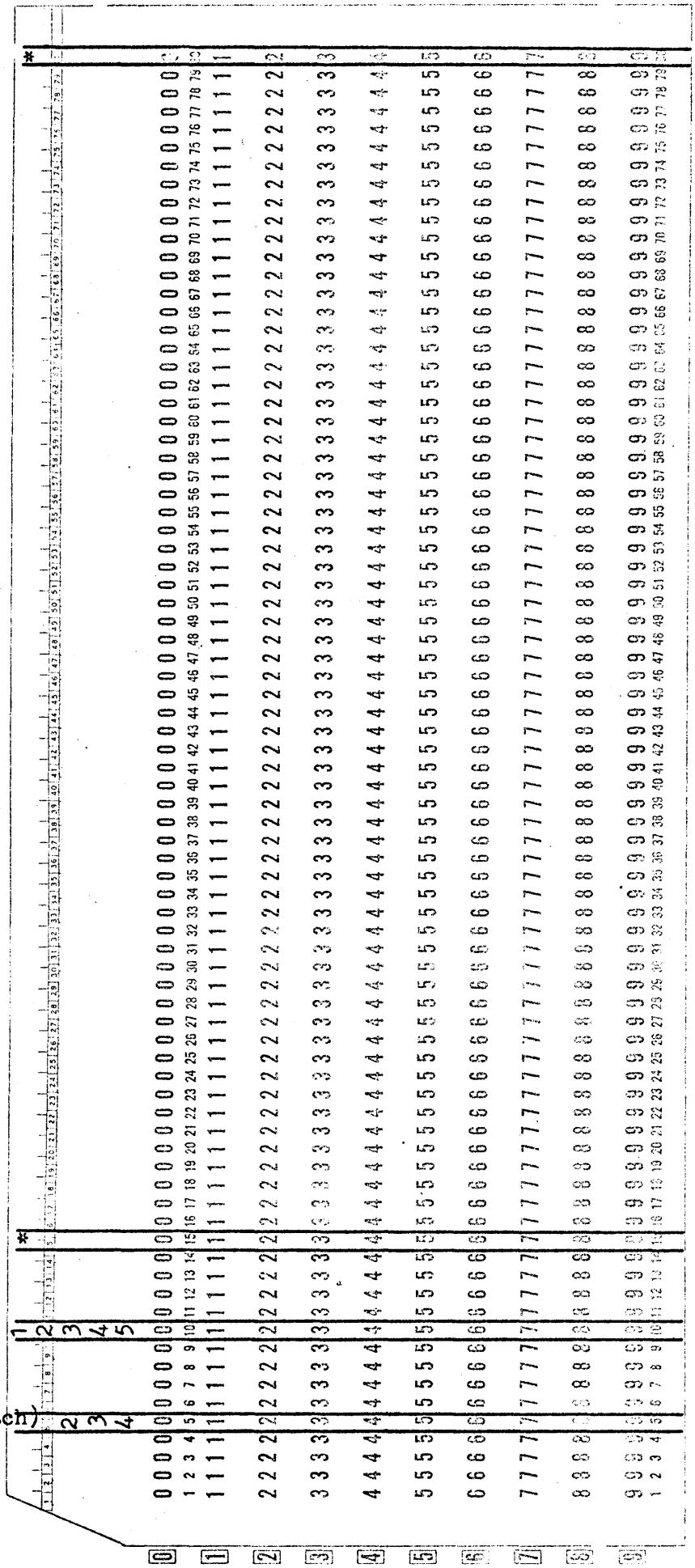
CARD LAYOUT FORM

Control Card I.D.

Multiple Input Indicator

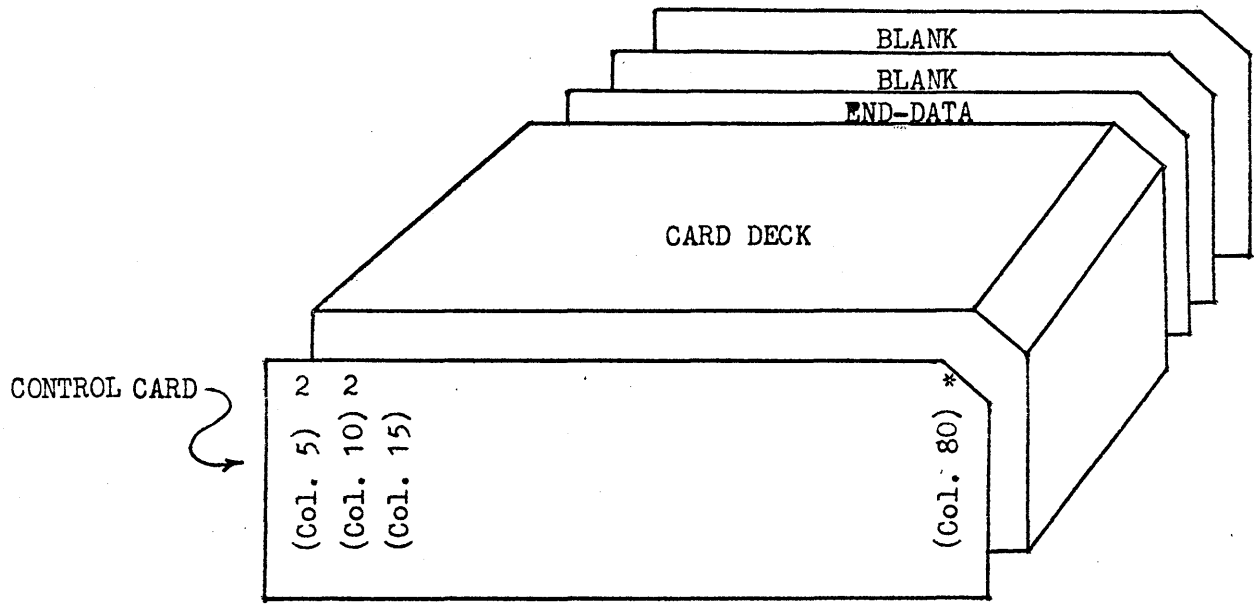
Input Mode

Type of System (2/3) Scratch



The control System is designed to facilitate input and tape assignment specifications to CS-1 for compiler operations. The following card deck layouts illustrate the use of control cards for both Two and Three Scratch System compiles as well as compiles with multiple inputs.

1. TWO SCRATCH SYSTEM - CARD INPUT ONLY

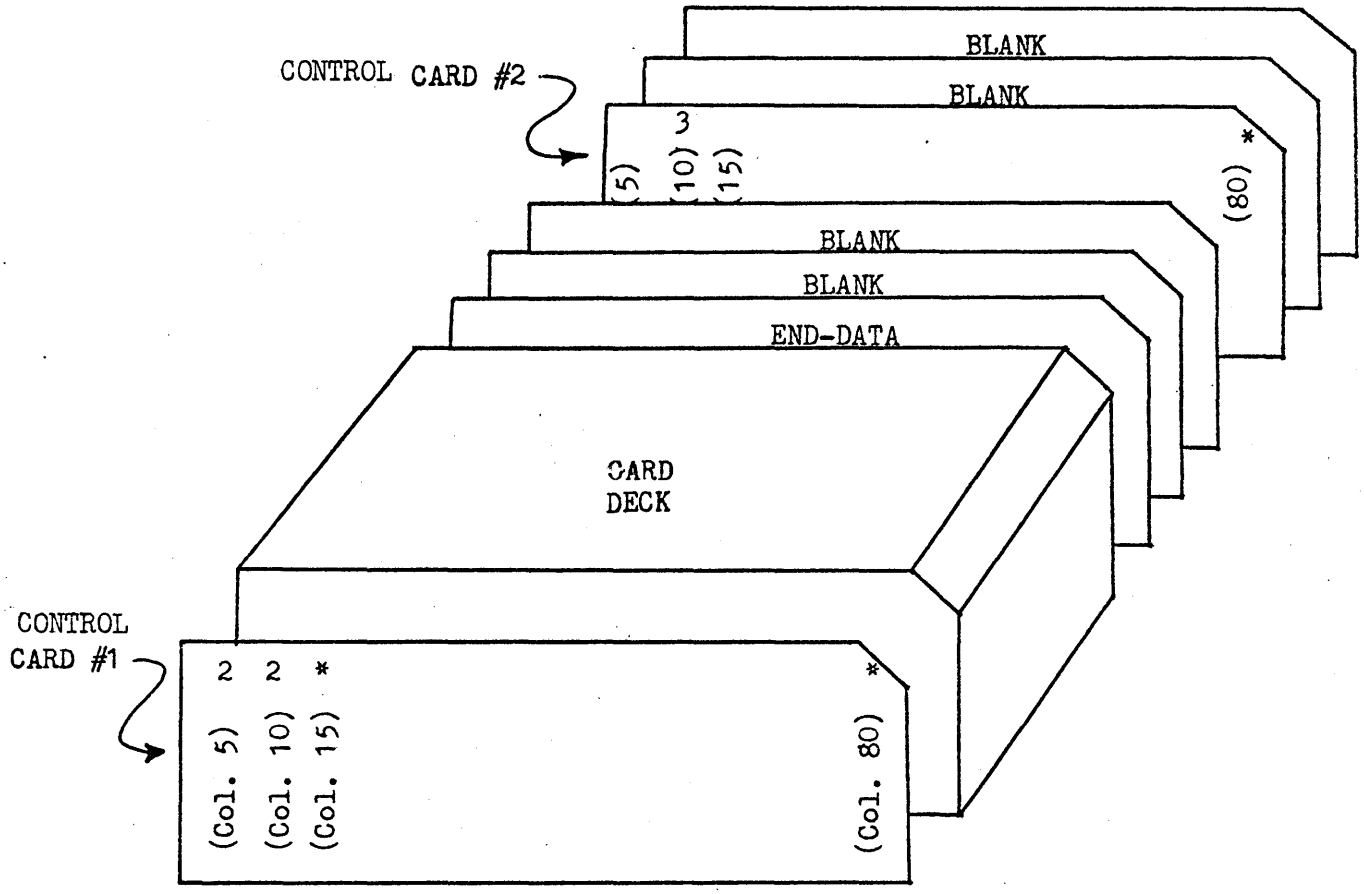


Control Card:

Column 5 - 2 Specifies Two Scratch System
Column 10 - 2 Specifies Card Input
Column 15 - Blank Specifies No More Input
Column 80 - * Specifies Control Card

NOTE: Card deck layout identical for a Three Scratch System with the exception of a 3 being punched in column 5 of the control card.

2. TWO SCRATCH SYSTEM - CARD AND 46 TAPE INPUT (MULTIPLE INPUTS)



Control Card #1:

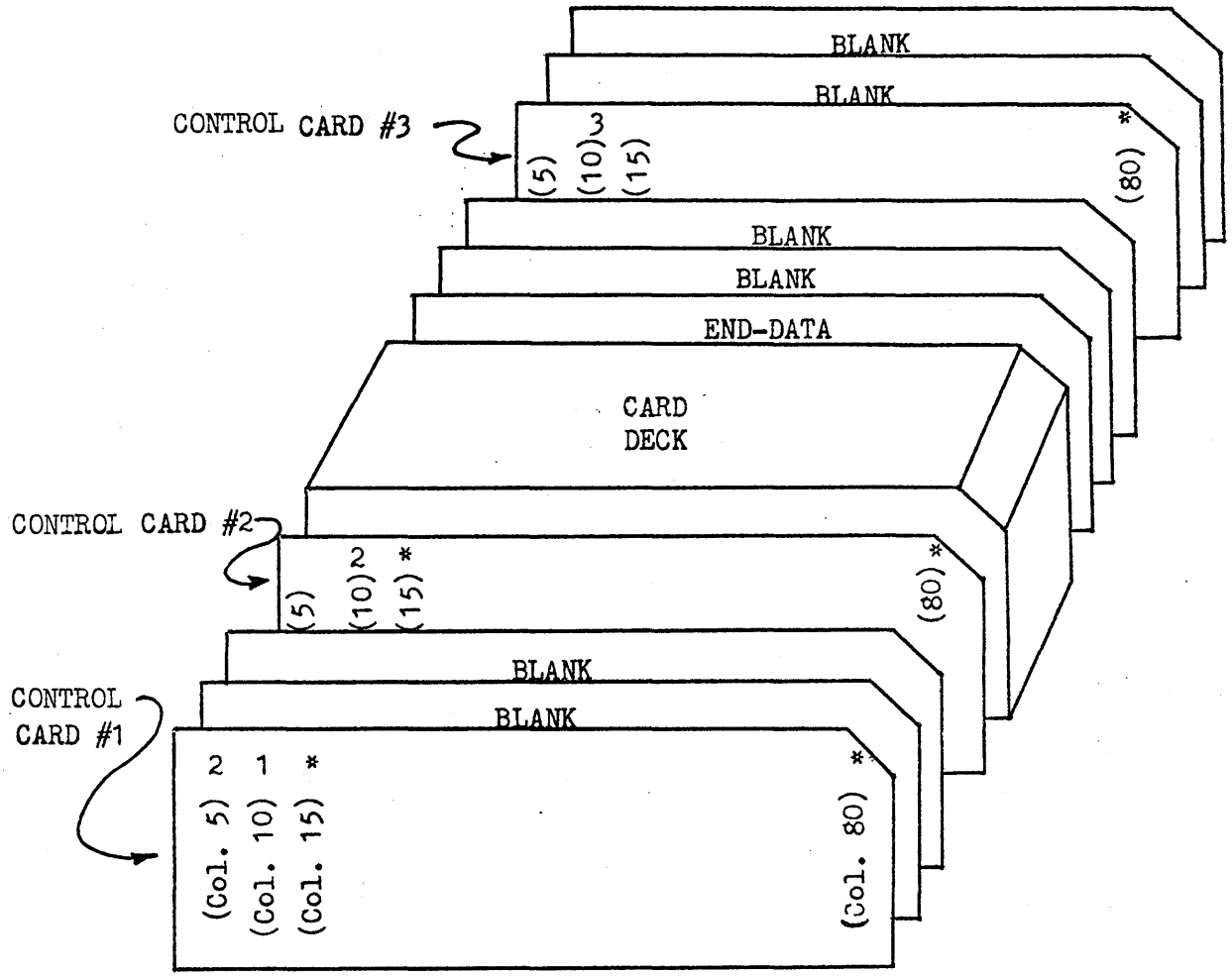
Column 5 - 2 Specifies Two Scratch System
Column 10 - 2 Specifies Card Input
Column 15 - * Specifies More Input
Column 80 - * Specifies

Control Card #2:

Column 5 - Blank (not needed)
Column 10 - 3 Specifies 46 Mag tape Input On TT4.
Column 15 - Blank Specifies No More Input
Column 80 - * Specifies Control Card

NOTE: Card deck layout identical for a Three Scratch System with two exceptions: firstly a 3 should be punched in Column 5 of control card #1 and, secondly, the 46 Mag Tape Input will then be on TT5.

3. TWO SCRATCH SYSTEM - PAPER TAPE (26), CARDS, AND MAG TAPE INPUT (226). (MULTIPLE INPUTS)



Control Card #1:

Column 5 - 2 Specifies Two Scratch System
Column 10 - 1 Specifies Paper Tape (26) Input
Column 15 - * Specifies More Input
Column 80 - * Specifies Control Card

Two Blank Cards must seperate Control cards.

Control Card #2:

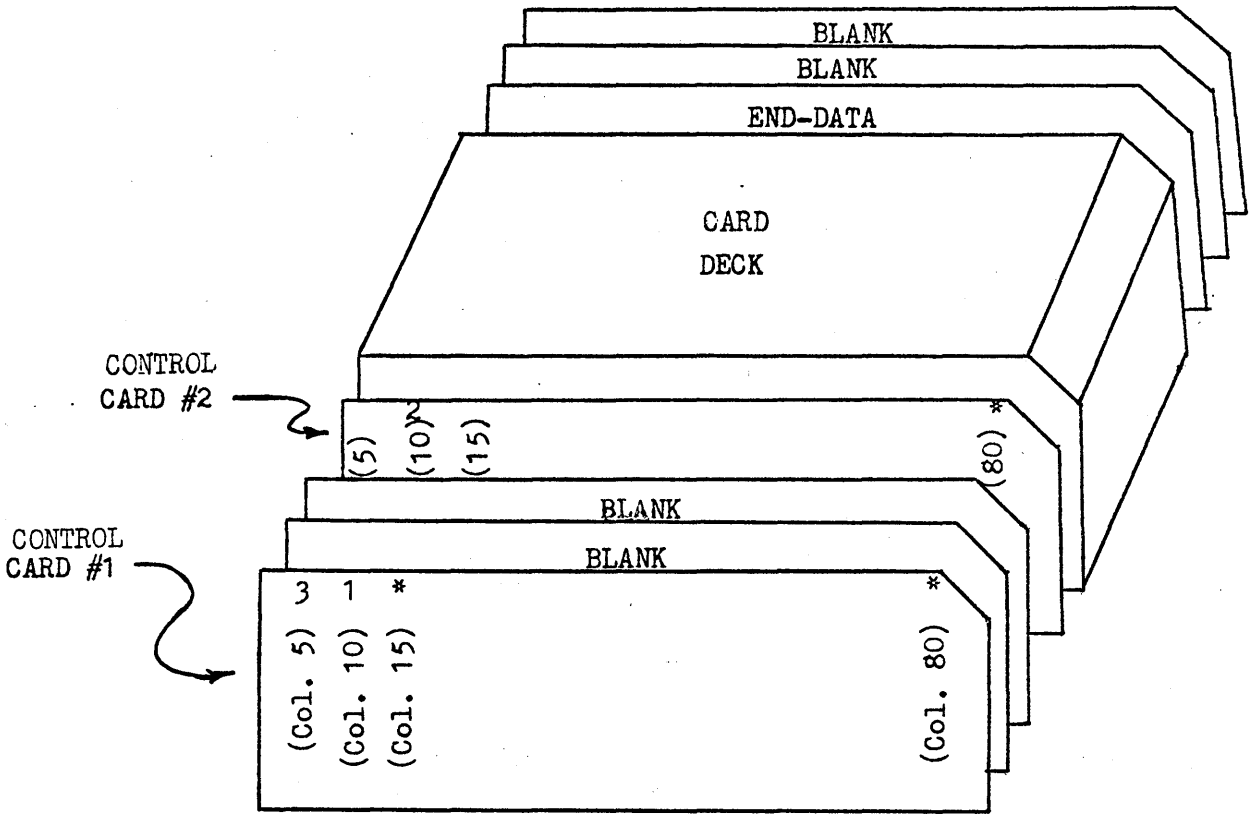
Column 5 - Blank (not needed)
Column 10 - 2 Specifies Card Input
Column 15 - * Specifies More Input
Column 80 - * Specifies Control Card

Control Card #3:

Column 5 - Blank (not needed)
Column 10 - 3 Specifies 46 Mag Tape Input on TT4
Column 15 - Blank Specifies No More Input
Column 80 - * Specifies Control Card

NOTE: Card deck layout identical for a Three Scratch system with two exceptions; firstly a 3 should be punched in Column 5 of Control Card #1 and secondly the 46 Mag Tape Input will then be on TT5.

4. THREE SCRATCH SYSTEM - PAPER TAPE (26) AND CARDS. (MULTIPLE INPUTS)



Control Card #1:

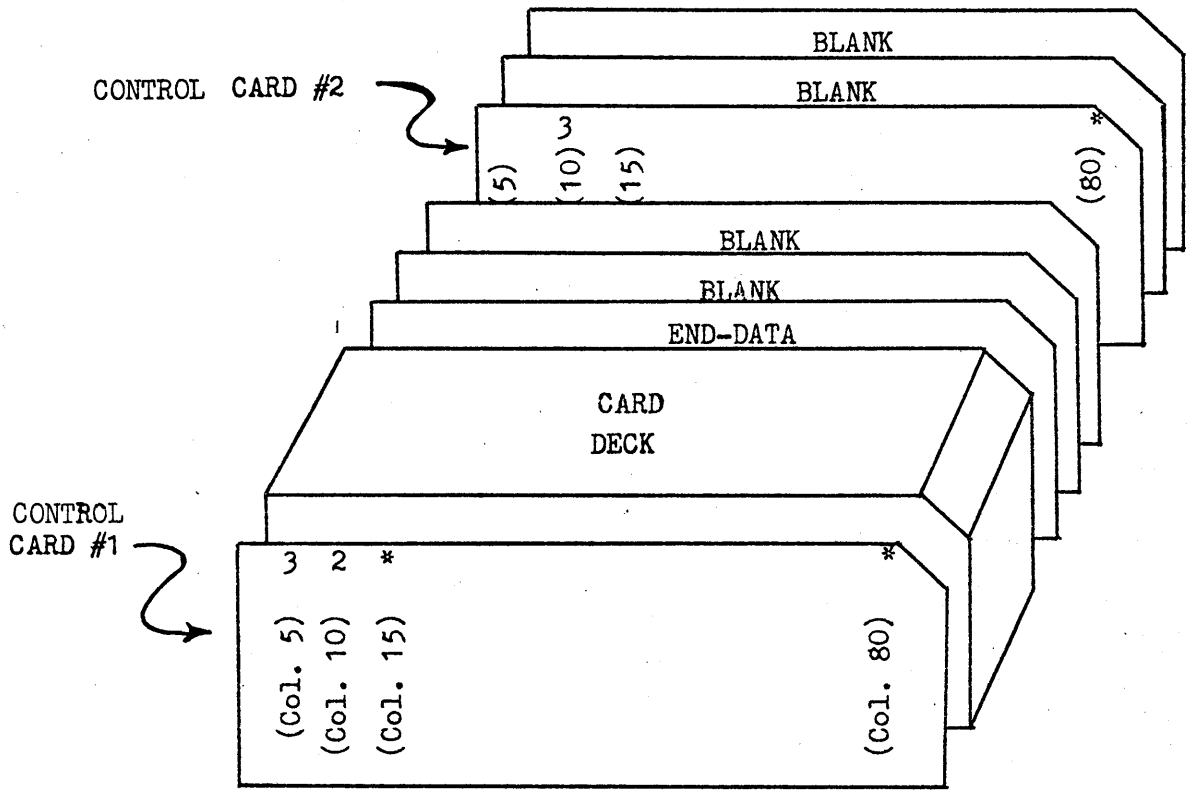
Column 5 - 3 Specifies Three Scratch System
Column 10 - 1 Specifies Paper Tape (26)
Column 15 - * Specifies More Input
Column 80 - * Specifies Control Card

Two Blank cards must separate Control Cards

Control Card #2:

Column 5 - Blank (not needed)
Column 10 - 2 Specifies Card Input
Column 15 - Blank Specifies No More Input
Column 80 - * Specifies Control Card

5. THREE SCRATCH SYSTEM - CARDS AND 46 MAG TAPE (MULTIPLE INPUTS)



Control Card #1:

Column 5 - 3 Specifies Three Scratch System
Column 10 - 2 Specifies Card Input
Column 15 - * Specifies More Input
Column 80 - * Specifies Control Card

Control Card #2:

Column 5 - Blank (not needed)
Column 10 - 3 Specifies 46 Mag Tape on TT5
Column 15 - Blank Specifies No More Input
Column 80 - * Specifies Control Card

IN THE PRECEDING ILLUSTRATIONS THE CARD DECKS MAY CONTAIN COMPILER CONTROL CARDS, OUTPUTS CARDS, ALLOCATION CARDS, PROGRAMS, AND L1-MERGE CARDS. IT SHOULD BE NOTED THAT TWO BLANK CARDS MUST SEPARATE TWO CONTROL CARDS OR AN END-DATA CARD AT THE NEXT CONTROL CARD. NO BLANK CARDS ARE NEEDED TO SEPARATE A CONTROL CARD AND AN ALLOCATION OR PROGRAM DECK.

OPERATING THE COMPILER

- Mount Compiler Tape on Logical Transport 1.
- Mount scratch tapes and input tapes (if needed) according to type of system selected and mode of input.
- Verify card deck adheres to Input Requirements and that proper Control cards are present along with appropriate blank cards.
- Place card deck in 1004 Card reader-feed one card.
- Master Clear The Computer
- Select Program Mode I - Mag tape bootstrap
- Depress LOAD
- Start

Normal typeouts following are RTD, RIN, and INPUT. If Key 6 is set, the compiler will 6 stop before the next or current input is processed.

Normal printouts during compiling are:

- RLIB
- INDENTIFY LIBRARIES HI-U(A) LO-L(A)
- Ø*Ø
- RT1
- TRAN 1 - possibly followed by generator errors on output tape or printer.
- RT2
- TRAN2 - error typeouts also possible here
- RA
- EQ1
- EQ2
- AA2 - all allocation errors are listed on the output magnetic tape or printer.
- RØ

- Each programmer selected output will be typed out by number as given the last being followed by SELECT OUTPUTS in A and Q and a 4-STOP. Additional outputs may be selected now by entering U(A), L(A), U(Q), and L(Q). If an output to be listed off-line exceeds one reel, set Key 2 before writing off the end of tape. A sentinel will be written on the tape, the tape rewound, and the typeout "MOUNT NEW OUTPUT TAPE." When a new tape has been set up, release Key 2 and normal operation will continue. This method can be used to terminate an output tape at any time.

- After all outputs have been taken a stop-code can be placed on the tape either by selecting an output 700 or by master clearing the console, setting P = 1404 and depressing START.

SPECIAL INSTRUCTIONS

- Stopping the compiler. The compiler may be stopped by setting Key 5. This capability is included so that the compiler may be stopped and minor operations such as inspect and change, core dumps, etc. accomplished at any point in a compiling run, with recovery possible. To continue compiling after a 5-STOP, master clear, set P to 1407, and depress START.

- Special LIID Stop. The capability to stop the compiling run at a predetermined LIID during the translation and allocation passes has been included. To exercise this option, 5-stop the computer at the beginning of the desired pass (TRAN1, TRAN2, RA, or AA2), and inspect and change the upper of cell Ø1376 to the desired LIID. The compiler will 4-stop when that LIID and each succeeding one are reached.

- Control information missing. If during input the compiler types out "MISSING CD-INSERT CARD AND RESTART," insert missing control card in deck, replace cards in 1004 reader, and depress Start. If the compiler types out "SET AL TO INPUT MODE," set AL to the next mode of input to be processed - Start.

- Restarting a compile. A new compile may be reinitiated by setting P = 1400 if the next compile adheres to the previous compiles scratch tape specifications. If a new assignment of scratch tapes and input tapes is necessary, set P = 1403 or reload the compiler. The card deck must be ready for processing prior to reinitiating a compile.

COMMENTS:

- The compiling process will go from start to finish without stopping unless a Control card missing or incorrect or an Outputs card is missing in which case the compiler will stop and type out "SELECT OUTPUTS IN A AND Q." At this time the user may select up to four outputs prior to compiling by setting A(U), A(L), Q(U), Q(L) to the desired outputs and Starting.

- All unallocated tags will automatically be allocated to zero (Ø) during the compile and will be printed after the RA segment. The unallocated tags and references will also be listed at the end of the 151 output if selected.

- If the first label in the program being compiled is unallocated (no base assigned), the compiler will automatically compile the program relatively (Base Ø).

- Tape unit assignments for input and scratch tapes used by the compiler are automatic according to the type of system designated (2 or 3 scratch) by the initial Control card.

- To change from a 2 scratch system to a 3 scratch system or vice-versa the user must insert the appropriate Control card in the beginning of his input deck and either reload the compiler or start at P = 1403. If a compile is restarted at P = 1400, the compiler will maintain the type of scratch tape system previously designated.

SPECIAL CS-1 COMPILER STARTING ADDRESSES:

1400 - Start A Compile - Same Scratch Tape System As Previously Designated.
1401 - Reselect Outputs Or Bypass Current Output.
1403 - Designate New Scratch Tape System and Compile.
1404 - Terminate Output Tape
1407 - TCS 5-STOP Recovery
1411 - Printer Core Dump - Set A to limits: A(L) Initial Limit-A(U) Terminal Limit

620 - Paper Tape Verify
620 - (Key 1) Paper Tape Load
620 - (Key 2) Bioctal Dump
621 - Inspect and Change
622 - Store Q

OUTPUTS

OUTPUTS AVAILABLE

Code Number	Description
	Paper Tape
10	Bioctal assembled program
12	Absolute assembled program (source code)
22	L1, L3 programs (L1 beside L4)
24	Relative Bioctal
26	Corrected LØ
27	Corrected LØ (Selective)
30	Labels and addresses
31	Significant labels and addresses
32	Numerically ordered addresses with labels
33	Alphabetically ordered labels with addresses
34	Relative Allocation
	Printer Listing
1Ø1	Identification
1Ø4	L1 and L2 programs and notes (Allocation pass unnecessary)
1Ø5	L1 program and notes (Allocation and Translation passess unnecessary)
114	L1, L3, programs (L1 beside L3) new page each major header
12Ø	L1, L3, and L3 programs
122	L1, L3, programs (L1 over L3)
125	L1, L3, programs (L1 beside L3)
126	LØ program before library additions
130	Labels and addresses
132	Sort of addresses and corresponding labels
133	Sort of labels and corresponding addresses
137	L1 in documentation format
151	Labels, addresses, and references
700	Terminate printer output tape
725 751	documentation size
	Magnetic Tape
4Ø	Absolute assembled program
41	Relative assembled program
226	LØ (with library additions) suitable for magnetic tape input or tape-to-card
46	Unappended (before library additions) L1 table on magnetic tape
	Card
326	L1 table converted to card (tape to card)

TAKING SPECIAL OUTPUTS

- Outputs 40, 41, 226, and 46 may be selected by means of an OUTPUTS card however they must also specify tape transport. The format to be used is XXYYY where XX = Transport, YYY = Output.

  Example: OUTPUTS*125*151*4040*5046

  In the above example an output 40 is taken on logical TT4 and and output 46 is taken on logical TT5.*

- Outputs 40 and 41 may be taken on one output tape. The user may request any number of 40 or 41 outputs on the same transport in any one compile or separate compiles.

- Outputs 226 and 46 must be taken on separate magnetic tapes. They may either be specified on the OUTPUTS card or selected manually.

- Output 46 is checkread after being written on magnetic tape. As it is being checkread a new listing is provided the user with L1ID's corresponding exactly to those on the 46 tape. This listing is a necessity in order to use the L1-Merge operator to update a 46 tape and compile if the user's program had library references i.e., TYPET, TYPEDEC. If no library references were made in the user's program the L1ID's on an Output 125 will correspond exactly to those on a 46 tape and may be used as a reference for L1-Merge operations. Key 5 may be set after the 46 tape is written and begins rewinding. When the 5 stop occurs, set P = 1401 and the chkread will be bypassed and the next output taken.

COMMENTS

- To recover if tape unit errors or checkread errors occur during any output, master clear, set P = 1401, and depress start. The current output will be bypassed and may be reselected when all other outputs are complete.

- Output 326 may be taken after any compile or directly from a 46 output tape. If the user wishes a new card deck of his program, he may reinitiate a compile with his updated 46 as input and select a 326 for his output. The 46 tape will be read, Table 1 built, and the output given without compiling.

  NOTE: If the users program (46 tape) had library references, the card decks of these routines will also be punched unless the user selects an Output 7 as well as the Output 326.

- Output 326 does not contain any compiler control cards i.e., C-CONTROL, OUTPUTS or any Allocation cards. The output 46 begins with the first Program or System Header card and ends with the End-Data card.

- Output 151 lists labels and references in the following order: Labels (alphabetically sorted), Indirect Allocations(in order they are processed), Unallocated Tags.

* NOTE: For a two scratch system only TT4 is available for magnetic tape output. For a three scratch system only TT4, TT5, and TT6 are available for magnetic tape output.

SPECIAL CAPABILITIES

PAPER TAPE CHECKREAD

- Description - Check reads paper tape
- Instructions - Master clear, set P = 620. Mount tape in reader, release all keys, depress START. If a checksum error occurs, CKSUM ERROR will be typed out, otherwise the computer will 4-STOP.

PAPER TAPE LOAD

- Description - Loads paper tape
- Instructions - Master clear, set P = 620. Mount tape in reader, set Key 1. If relative tape is being loaded, set L(A) to beginning address. Depress START. If a checksum error occurs, CKSUM ERROR will be typed out, otherwise the computer will 4-stop indicating satisfactory load.

BIOCTAL DUMP

- Description - Dumps contents of core onto paper tape in bioctal format with checksums.
- Instructions - Master clear, set P = 620. Set Key 2. Set L(A) to initial address of area desired and U(A) to terminal address. Depress START. After dump, tape may be checkread using Paper Tape Checkread.

INSPECT AND CHANGE

- Description - Allows any cell in core to be inspected and changed if desired, except ØØ166 (B6).
- Instructions - Master Clear, set P = 621, and depress START. I and C will be typed out and the computer will then accept an octal address keyed in from the 1232 I/O Console. To key in an address, the user may enter from one to five octal digits. If fewer than five digits are entered they must be followed by a period (.), the input digits will then be right justified and treated as a five digit address. When an address has been keyed in, the contents of that address will be typed out and one of the following options may be exercised:

  (/) To change the value of the location whose contents have just been typed out, type a slash (/) and then type in the desired value. Here again a period may be used if fewer than ten octal digits are typed. This value will be stored in memory and then allow a new control character to be typed.

  (SP) To inspect the contents of the next succeeding memory location, depress the space bar. The succeeding address and its contents will be typed, after which a new control character will be accepted.

  (,) To alter the current sequence, type a comma (,). A Carriage Return will be executed after which a new address may be typed in.

STORE Q

Description - Stores contents of Q at successive memory address between any two limits.

Instructions - Master clear, set P = 622. Set Q to the value to be stored; Set L(A) to initial address of storage area and U(A) to terminal address of area. Depress START.

USE OF THE L1 - MERGE OPERATOR

Output 46 -

Output 46 consists of the unappended (before library additions) L1 table on magnetic tape in a format suitable to loading as input to a succeeding compile. In conjunction with the L1-MERGE operation it allows a flexible system for correcting programs.

Note: No C-CONTROL type operations are included on the output 46 tape or listing.

The program correction operation to be used with output 46 is L1-Merge. The corrections that are accepted are:

1) delete
2) replace
3) insert

<u>The L1 identifiers (L1ID) on the L1-MERGE cards must be in ascending numerical order.</u>

L1-MERGE Operation

	V_0		V_1
L1-MERGE *	L1 ID to commence merging	*	Last L1 ID to be deleted

The L1-MERGE operation, a minor independent operator, specifies to the compiler that normal L_0 card input is to be combined with an output 46 magnetic tape to produce the L1 table. Statements to be merged follow the L1-MERGE header.

V_0 - The L1ID of the output 46 to be modified at which merging is to commence.

V_1 - If present, the L1ID of the last statement on the output 46 which is to be deleted. <u>If V_1 is absent, no deletions are made</u>.

Examples:

➡ L1-MERGE * 100

➡ SET * I * TO * 0

➡ END-DATA

This will cause the statement SET * I * 0 to be inserted following L1ID 100 when the output 46 is processed.

USE OF THE DATAX OPERATOR

```
   L        W                  V∅
(Label) ⇨ DATAX * (Decimal Constant, Exponent, Scaling)
```

The DATAX operation allows the programmer to specify a scaled octal constant which has the value represented by a decimal constant expressed in scientific notation with an exponent (10 to the exponent) and scaling.

L This is the label (optional) of the scaled octal constant.

W This is the operator DATAX.

$V_{\emptyset}$ This decimal constant represents the octal value to be generated. The format of the $V_{\emptyset}$ operand is as follows:

(±) X.YYYYYYYYYY, (±) EE, (±) SSS

Where:

1. "X.YYYYYYYYYY" is the decimal constant with a maximum of ten digits following the decimal point. The decimal point must follow the most significant digit. The most significant digit should be a digit other than zero (∅).

2. "EE" is the decimal exponent indicating 10 to the power EE. If the commas are present with no exponent, the exponent is assumed zero (∅). The range of EE is $-99 \leq EE \leq +99$.

3. "SSS" is the decimally expressed binary scaling. If the commas are present with no scaling, the scaling is assumed zero (∅).

4. The two commas specified in the DATAX format should be present. The single exception is that the number ∅ (zero) may be specified by: DATAX * ∅
DATAX * ∅ will cause proper generation however, format error messages will be printed out.

5. The use of the + sign is optional. If a sign is missing the value is assumed positive.

Warnings will be given for:

a. Underflow; SSS to small

b. Overflow; SSS to large

c. Commas missing

If any of the above errors occur, the generated operand will be zero (Ø).

The generated operand will be a whole word (thirty bit) constant significant to the last bit.

EXAMPLES:

		Generation	
LABEL	DATAX*3.9999999999,Ø,27	37777	77777
LABEL1	DATAX*-1.25,Ø,18	77765	77777
PI	DATAX*3.141592653,Ø,26	14441	76652

USE OF THE DATAF OPERATOR

```
  L          W                    Vø
(Label)  ➡  DATAF  *  (Decimal Constant, Exponent)
```

The DATAF operation allows the programmer to specify a floating point constant which has the value represented by the decimal constant expressed in scientific notation with exponent (10 to the exponent). The floating point words conform to the 1230 Hardware Floating Point format and occupy two consecutive memory words. The first word is the characteristic with sign extension and the second word is the mantissa. Refer to Appendix A for a more detailed description of the 1230 Hardware Floating Point format.

L This is the label (optional) assigned to the first word (characteristic) of the two consecutive floating point constant words.

W This is the operator DATAF.

$V_{\emptyset}$ This scientifically expressed decimal constant represents the value of the floating point constant to be generated. It is in the form:

(±) X.YYYYYYYYYY, (±) EE

Where:

1. "X.YYYYYYYYYY" is the decimal constant with a maximum of ten digits following the decimal point. The decimal point should follow the most significant digit which may be zero (Ø).

2. "EE" is the decimal exponent indicating 10 to the power EE. If the comma is present with no exponent, the exponent is assumed zero; proper generation will occur however an error message indicating "EXPONENT MISSING" will occur. The range of EE is $-99 \leq EE \leq +99$.

3. The comma and decimal point specified in the DATAF format should be present. Constants expressed as DATAF*Ø or DATAF*5 will cause proper generation however, format error messages will be printed out.

4. The use of the + sign is optional. If a sign is missing the value is assumed positive.

Warnings will be given for any deviation in the prescribed format and proper generation may or may not result depending on the type of error detected.

EXAMPLE:

```
                                                    Generation
LABEL ➡ DATAF*12,1                    LABEL ➡ 00000  00007
                                              36000  00000
```

(Error Printout)

```
FORMAT ERROR        12,1

DATAF*+/-X.YYYYYYYYYY,+/-EE

DECIMAL POINT MISSING

NO FRACTIONAL CHARS

CARD  0001
00036  LABEL  DATAF*12,1
```

The algorithm utilized to compute the floating point constant is designed to provide the programmer with extreme accuracy. Rounding of the mantissa takes place any time the bit to the right of the least significant bit is set.

Example: $.4_{10} = .3146314631463146_8\cdots$

```
                                          Generation
   DATAF*Ø.4,Ø                            77777  77776
                                          31463  14632
```

The generated operand will be two consecutive thirty bit words. The first word will contain the floating point <u>characteristic</u> with sign extension. The second word will contain the floating <u>mantissa</u>.

Examples:

```
                                          Generation
   LABEL ➡ DATAF*1.Ø,Ø                    00000  00001
                                          20000  00000
           DATAF*1.5,1                    00000  00004
                                          36000  00000
   PI    ➡ DATAF*3.141592653              00000  00002
                                          31103  75524
           DATAF*1.Ø,-63                  77777  77456
                                          32247  76234
           DATAF*-Ø.4,-1                  77777  77773
                                          53412  17267
```

```
        ➡  L1-MERGE * 102 * 104
        ➡  CL * W(BDFLG)
        ➡  RPL * Y+1 * W(FLFLG)
        ➡  RETURN
SETFLG  ➡  ENT * A * W(BDFLG)
        ➡  L1-MERGE * 155 * 165
        ➡  END-DATA
```

This input will delete L1ID's 102, 103, and 104 when the output 46 is processed, inserting the four statements between the L1-MERGE operations in their place. Additionally, L1-ID's 155 through 165 will be deleted.

Using the above rules, basic operations are accomplished as follows:

a. To delete one statement -- V_0 equals L1ID to be deleted; V_1 equals V_0; no statement between L1-MERGE operation and succeeding header:

```
L1-MERGE * 7 * 7
L1-MERGE * etc.
```

b. To replace one statement -- V_0 equals L1ID to be replaced; V_1 equals V_0; new statement follows L-1MERGE operation:

```
L1-MERGE * 12 * 12
ENT * B1 * 5
L1-MERGE * etc.
```

c. To insert one statement -- V_0 equals L1ID following which new statement is to be inserted; V_1 absent; new statement follows L1-MERGE operation:

```
L1-MERGE * 25
STR * Q * W(BDFLG)
L1-MERGE * etc.
```

To add statements at the end of the output 46, the L1ID of the last statement on the output 46 should be used as the V_0 operand of the L1-MERGE operation.

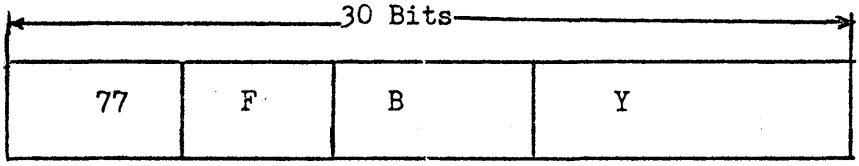
The L1-MERGE deck is terminated by an END-DATA card and must be placed at the end of the users Input deck.

FORMAT II MNEMONIC STATEMENTS

The inclusion of the Hardware Floating Point Package adds eight new commands to the repertoire of the UNIVAC 1230 Computer that afford the capacity for double-precision arithmetic in addition to floating point operations. Optional interrupt features are also available for more convenient use of the real-time clock (RTC).

FORMAT II INSTRUCTION LAYOUT

The added instructions are implemented in a manner to avoid interfering with existing 1230 programs by using previously illegal function codes of 77.



The function codes of the FORMAT II instructions are four digits (12 bits) long. The upper two digits of the function code are always 77 to denote a FORMAT II INSTRUCTION; thus f is described as being 12 bits in length (bits 29 through 18). The specific FORMAT II instruction is denoted by the lower two digits (bits 23 through 18) of the function code. The normal three bits are available for B register modification and the operand Y is 5 digits (bits 14 through 0) in length. Due to the FORMAT II instruction a function code of 77 will no longer generate a fault interrupt; however, the following three function codes will generate fault interrupts: ØØ, 77ØØ, 7777.

SPECIAL OPERATORS

- The following mnemonics will generate the appropriate absolute instruction codes when compiled with the new CS-1 compiler:

MNEMONIC	TYPE II FCT CODE	OPERATION PERFORMED
DENT(Double Enter)	7711	(Y+1,Y) ➡ A,Q
DSTR(Double Store)	7715	(A,Q) ➡ Y+1,Y
FADD(Flpt. Add)	7720	(A,Q) + (Y+1,Y) ➡ A,Q Flpt.
FSUB(Flpt. Subtract)	7721	(A,Q) - (Y+1,Y) ➡ A,Q Flpt.
FMUL(Flpt. Multiply)	7722	(A,Q) · (Y+1,Y) ➡ A,Q Flpt.
FDIV(Flpt. Divide)	7723	(A,Q) ÷ (Y+1,Y) ➡ A,Q Flpt.
DADD(Double Length Add)	7724	(A,Q) + (Y+1,Y) ➡ A,Q
DSUB(Double Length Subtract)	7725	(A,Q) - (Y+1,Y) ➡ A,Q
ERTC*OF	7706	Enable RTC Overflow Interrupt
ERTC*MONITOR	7726	Enable RTC Monitor Interrupt
NORM*AQ	7707	Shift AQ Left until bits $A_{29} \neq A_{28}$. Shift Count ➡ Y

- The above mnemonics with the exception of ERTC*OF and ERTC*MONITOR have a normal "read-y" operand with an automatic K designation of W (Whole word).

```
Example:  MNEMONIC        read-y
          DENT      *     W(DATA+B2-3)
          DENT      *     DATA+B2-3
```

Both of the above instructions will generate the same absolute coding since a K of W is assumed. The "read-y" operand always specifies the first word of two consecutive memory words.

- The ERTC instructions do not require a "read-y" operand.

- The NORM*AQ instruction's "read-y" operand specifies one memory word.

DESCRIPTION OF OPERATIONS PERFORMED

ERTC*OF — 77 06 - Enable Real-Time Clock Overflow Interrupt Request. This instruction enables the real time clock overflow interrupt when the real time clock overflows from 77777 77777 to 00000 00000.

NORM*AQ — 77 07 - Normalize. - Shift (AQ) left circularly until (A29) $\neq$ (A28) or until k = 0. The number of shifts that occurred is then stored in storage location Y lower six-bits.

DENT — 77 11 - Double Length Enter. - Clear the A and Q registers, then transmit the contents of address Y to the Q register and the contents of address Y + 1 to the A register.

DSTR — 77 15 - Double Length Store. - Store the contents of the Q register at address Y and the contents of the A register at address Y + 1.

FADD — 77 20 - Floating Point Add. - This instruction shall add a floating point number whose mantissa is in A and characteristic is in Q to a floating point number whose mantissa is in Y + 1 and characteristic is in Y. The final normalized mantissa shall be in A and the final characteristic shall be in Q.

FSUB — 77 21 - Floating Point Subtract. - This instruction shall subtract a floating point number whose mantissa is in Y + 1 and characteristic is in Y from a floating point number whose mantissa is in A and characteristic is in Q. The final normalized mantissa shall be in A and the final characteristic shall be in Q.

FMUL — 77 22 - Floating Point Multiply. - This instruction shall multiply a floating point number whose mantissa is in A and the characteristic is in Q by a floating point number whose mantissa is in Y + 1 and characteristic is in Y. The final normalized mantissa shall be in A and the final characteristic shall be in Q.

FDIV — 77 23 - Floating Point Divide. - This instruction shall divide a floating point number whose mantissa is in A and characteristic in Q by a floating point number whose mantissa is in Y + 1 and characteristic is in Y. The final normalized mantissa shall be in A and the final characteristic shall be in Q.

DADD — 77 24 - Double Length Add.-This instruction shall add a 60-bit number in Y + 1 and Y to a 60-bit number in AQ. The sum shall be in AQ.

DSUB — 77 25 - Double Length Subtract. - This instruction shall subtract a 60-bit number in Y + 1 and Y from a 60-bit number in AQ. The difference shall be in AQ.

ERTC*MONITOR — 77 26 - Enable Real-Time Clock-Monitor-Interrupt. - This instruction shall enable the real time clock monitor interrupt when the lower 17 bits of the real time clock are equal to the lower 17 bits of the Real Time Clock Monitor register. (Address 170)

Floating Point Format and a Discussion of Floating Point and RTC interrupts is contained in Appendix A.

The following are illustrations of generation which would result from the use of the prescribed mnemonics:

```
• MNEMONIC      read-y        GENERATION
    DENT    *   DATA+B2       77112*NNNNN

  (Example)     Addr.         Machine Instruction
                00162         00000 00000
                10000 DATA    76777 54377
                10001         00001 43332
                  .             .   .
                  .             .   .
                  .             .   .
                20000         77112 10000 ➪ DENT
```

After execution of the DENT instruction at addr. 20000, A and Q would look as follows:

```
                    A             Q
                00001 43332   76777 54377
```

```
• MNEMONIC      read-y        GENERATION
    DSTR    *   W(DATA+B2)    77152*NNNNN

  (Example)     Addr.         Machine Instruction
                00162         00000 00000
                10000 DATA    76777 54377
                10001         00001 43332
                  .             .   .
                  .             .   .
                  .             .   .
                20000         77152 10000 ➪ DSTR
```

After execution of the DSTR instruction at addr. 20000, DATA and DATA+1 would look as follows if A = 00176 47753 and Q = 77775 47776:

```
                10000 DATA    77775 47776
                10001         00176 47753
```

```
• MNEMONIC      read-y        GENERATION
    FADD    *   W(DATA+2)     77200*NNNNN

  (Example)     Addr.         Machine Instruction
                10000 DATA    00000 00004 ➪ Flpt. 12
                10001         24000 00000
                10002 DATA1   00000 00005 ➪ Flpt. 24
                10003         24000 00000
                  .             .   .
                  .             .   .
                  .             .   .
                20000         77110 10000 ➪ DENT
                20001         77200 10002 ➪ FADD
```

After execution of the DENT instruction at addr. 20000 and the FADD instruction at addr. 20001, A and Q would look as follows:

```
                    A             Q
                36000 00000   00000 00005 ➪ Flpt. 36
```

```
MNEMONIC      read-y       GENERATION
  FSUB    *   DATA-2       77210*NNNNN

(Example)     Addr.        Machine Instruction
              10000 DATA1  00000 00004 ➡ Flpt. 12
              10001        24000 00000
              10002 DATA   00000 00005 ➡ Flpt. 34
              10003        34000 00000
                :            :     :
                :            :     :
                :            :     :
              20000        77110 10002 ➡ DENT
              20001        77210 10000 ➡ FSUB
```

After execution of the DENT instruction at addr. 20000 and the FSUB instruction at addr. 20001, A and Q would look as follows:

```
                A              Q
              22000 00000    00000 00005 ➡ Flpt. 22
```

```
MNEMONIC      read-y       GENERATION
  FMUL    *   W(DATA)      77220*NNNNN

(Example)     Addr.        Machine Instruction
              10000 CONST  00000 00007 ➡ Flpt. 137
              10001        27600 00000
              10002 DATA   00000 00004 ➡ Flpt. 12
              10003        24000 00000
                :            :     :
                :            :     :
                :            :     :
              20000        77110 10000 ➡ DENT
              20001        77220 10002 ➡ FMUL
```

After execution of the DENT instruction at addr. 20000 and the FMUL instruction at addr. 20001, A and Q would look as follows:

```
                A              Q
              35540 00000    00000 00012 ➡ Flpt. 1666
```

```
MNEMONIC      read-y       GENERATION
  FDIV    *   DATA         77230*NNNNN

(Example)     Addr.        Machine Instruction
              10000 CONST  00000 00003 ➡ Flpt. 6
              10001        30000 00000
              10002 DATA   00000 00001 ➡ Flpt. 1.4
              10003        30000 00000
                :            :     :
                :            :     :
                :            :     :
              20000        77110 10000 ➡ DENT
              20001        77230 10002 ➡ FDIV
```

After execution of the DENT instruction at addr. 20000 and the FDIV instruction at addr. 20001, A and Q would look as follows:

```
                A              Q
              20000 00000    00000 00003 ➡ Flpt. 4
```

```
• MNEMONIC      read-y      GENERATION
  DADD    *     DATA1       77240*NNNNN

(Example)       Addr.       Machine Instruction
                10000 DATA1 77777 65476
                10001       00000 00000
                10002 DATA2 00000 22301
                10003       00000 00000
                  .           .     .
                  .           .     .
                  .           .     .
                20000       77110 10002 ➪ DENT
                20001       77240 10000 ➪ DADD
```

After execution of the DENT instruction at addr. 20000 and the DADD instruction at addr. 20001, A and Q would look as follows:

```
                A             Q
            00000 00001   00000 07777
```

```
• MNEMONIC      read-y      GENERATION
  DSUB    *     DATA        77250*NNNNN

(Example)       Addr.       Machine Instruction
                10000 DATA  77754 37776
                10001       00000 01234
                10002 DATA1 01322 77665
                10003       00001 13244
                  .           .     .
                  .           .     .
                  .           .     .
                20000       77110 10002 ➪ DENT
                20001       77250 10000 ➪ DSUB
```

After execution of the DENT instruction at addr. 20000 and the DSUB instruction at addr. 20001, A and Q would look as follows:

```
                A             Q
            00001 12007   01346 37667
```

```
• MNEMONIC      read-y      GENERATION
  ERTC*OF       None        77060*00000

• MNEMONIC      read-y      GENERATION
  ERTC*MONITOR  None        77260*00000
```

MNEMONIC	read-y	GENERATION
NORM*AQ *	COUNT	77070*NNNNN

(Example)

Addr.		Machine Instruction	
10000	DATA	00000	05541
10001		02111	00027
10002	COUNT	00000	00000
°		°	°
°		°	°
°		°	°
20000		77110	10000 ⇨ DENT
20001		77070	10002 ⇨ NORM

After execution of the DENT instruction at addr. 20000 and the NORM instruction at addr. 20001, memory word COUNT will look as follows:

10002 COUNT 00000 00003

NOTE: The following values in A and Q will all result in the same number of shifts if normalized.

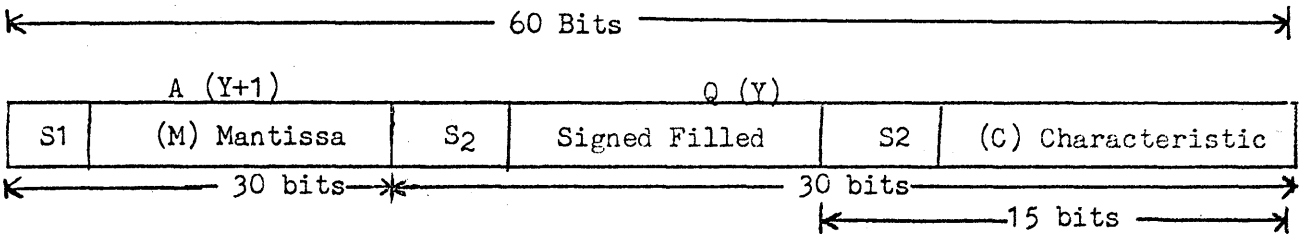
A	Q
00000 00000	00000 00000
77777 77777	77777 77777
00000 00000	00000 00001

Shift count will be 72_8 or 58_{10}.

APPENDIX A

1230 COMPUTER FLOATING POINT AND RTC INTERRUPT

FLOATING POINT OPERATIONS. - The floating point format is shown below with each operand occupying 45 bits of a possible 60. Consecutive memory locations are indicated by Y, Y+1.



S_1 = 1 bit algebraic sign

M = 29 bits mantissa

S_2 = 1 bit algebraic sign

C = 14 bits characteristic

The number notation used is one's complement. To obtain the algebraic complement of a floating point number, all 30 bits of the mantissa and algebraic word are complemented.

The characteristic as shown above is a 14-bit plus sign, one's complement, unbiased number held in the Q register or address Y. The characteristic values are from 2^{-16383} to 2^{+16383} with sign extended through the upper 15 bits of Q or Y.

The mantissa is a 29-bit plus sign, fractional one's complement number held in the A register or address Y+1. The mantissa is normalized to be greater than or equal to $\frac{1}{2}$ but less then 1. All floating point instructions normalize the mantissa to be within this range.

ZERO MANTISSA.

a) When the mantissa is zero, the characteristic is scaled to a binary zero ($+2^0$).

b) When doing a floating point add where one value is zero the sum will result in in the other value.

c) When doing a floating point subtract when the contents of A equals zero and 1/m 0, the result is the complement of the Y mantissa. 1/m = 0 also results in a zero difference.

d) When doing a floating point multiply with a zero mantissa the product is zero.

e) When doing a floating point divide by zero results in a floating point divide fault interrupt.

f) When doing a floating point divide into zero the result is zero.

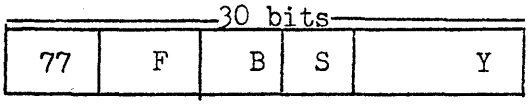
FLOATING POINT INTERRUPTS. - When any floating point operation results in any of the following conditions it sets the interrupt bit in the priority scan for program interruption. These interrupts cannot be locked out.

a) Characteristic overflow - This occurs when the decimal value of the characteristic exceeds +16,383.

b) Characteristic underflow - This occurs when the decimal value of the characteristic is less than -16,383.

c) Floating Point Divide Fault - This occurs whenever an attempt is made to divide a floating point number by zero in executing the floating point divide instruction.

FLOATING POINT INSTRUCTION FORMAT. - The floating point instructions are Format II instructions and are described below in Basic Operations.



F = 6-bits Format II function code, B = 3-bits index registers, S = 2-bits address extension used, Y = 13-bits contains relative address. Effective address Y = Y (extended as per S) + B.

BASIC OPERATIONS

X	N	10^m
88.0	.88	10^2
4.6	.46	10^1
.14	.14	10^0
.0705	.705	10^{-1}

When performing any arithmetic functions it is mandatory to properly position the decimal (or binary) point. The above chart shows a method of acaling the original number (X). N (.88) is the mantissa and the exponent (power of 10) is referred to as the characteristic. Now take the value 010.110010_2 and see what happens when we use the power of 2. The most insignificant zero in the binary number is the sign bit; if the binary point is shifted two binary places to the left, it becomes $0.\ 10\ 110\ 010 \times 2^2$. The number $.000\ 000\ 010_2$ could be stored as $0.10\ 000\ 000 \times 2^{-7}$. As can be seen, any binary number, no matter how large or small, can be expressed as some quantity times a power of 2. The number is stored in one location, and the power of 2 is stored in another location but is always identified with the number to which it belongs.

a) FLOATING POINT ADD/SUBTRACT. - The floating point addition and subtraction are identical except for complementing of Y in the subtraction; therefore, only the addition process will be explained.

Initially, A contains the augend in floating point format, and Y contains the addend in floating point format. The mantissa is the actual number to be added and the characteristic is the true binary exponent of the number, and the binary point is between S1 and M. The add instruction adds a floating point number whose mantissa is in Y+1 and characteristic is in Y. The final normalized mantissa is in A and the final characteristic is in Q.

To manually perform floating point add do the following:

1. Scale the first number.

2. Put the scaled numbers into AQ.

3. Set Format II F/F, put 15030 01000 in U register, clear sequencer, set repeat active, set up step mode, hit Start twice.

4. Take the number to be added and scale it.

5. Put scaled number manually into AQ.

6. Perform the following to do a floating point add: Set Format II F/F, put 20 030 01000 in the U register. Clear sequencer, set repeat active, set op step mode, hit start twice. A floating point add has been performed on the numbers in AQ plus the numbers in address Y(1000) & Y(1001)+1.

b) FLOATING POINT MULTIPLY. - Function code is F = 7722. When this is executed it will replace the contents of the AQ register with the normalized floating point product of AQ times (Y + 1, Y). The 29 most significant bits, after normalizing, plus sign of the result are found in the A register. See Section 5 for sequence timing.

NOTE

This instruction can cause a characteristic overflow or underflow interrupt.

c) FLOATING POINT DIVIDE. - Function code is F = 7723. When this is executed the contents of AQ register is replace with the normalized floating point quotient of AQ divided by (Y + 1, Y). See Section 5 for sequence tuning.

NOTE

This instruction can cause a characteristic overflow or underflow interrupt or a floating point divide fault interrupt.

REAL-TIME CLOCK. - When the real-time clock requests updating, the RTC flip-flop is set. A check is then made of priority to see if a function of equal priority has requested priority. If no equal priority has been franted, the priority will be granted to the clock, the I/O sequence will be initiated, and the count in the clock register will be advanced by one. After the clock has been advanced priority may then be granted to a function of equal or lower priority. The clock circuits are shown in figure 8-75. The clock input may either be internal or external depending on whether the 4010 card is placed in location 0J52C or 9J53C.

a) REAL TIME CLOCK INTERRUPT. - This feature provides two basic RTC interrupts: overflow and monitor (Delta Increment). These are programmable Real-Time Clock Interrupt features providing a wide range of internal, program-controlled timing/clocking of the computer programs and computer controlled related systems

for such application as Real-Time Executive control systems, Real-Time batch processing, and Real-Time Time-Shared operations.

b) REAL TIME CLOCK OVERFLOW INTERRUPT. - The RTC overflow interrupt occurs when the RTC switch is activated on the computer console and the following two conditions are satisfied:

1) The "Enable RTC overflow interrupt" instruction (F = 7706) has been executed by the computer (Fig. 8-316B).

2) The RTC goes from 77777 77777 to 00000 00000.

When the above two conditions are satisfied, an internal interrupt occurs. These two conditions must be repeated to allow another RTC overflow interrupt.

The RTC may be programmed by means of the store instruction consisting of any desired initial value so that the time-to-overflow, based on the RTC rate of 1024 cps, may be controlled accurately.

The maximum time between RTC overflow interrupts is $(2^{30}-1)$ cycles of the real-time clock at 1024 cps or approximately 12 days.

c) READ TIME CLOCK MONITOR INTERRUPT. - The RTC monitor interrupt will occur when the RTC switch is activated on the computer console and the following two conditions are satisfied:

1) The "Enable RTC monitor interrupt" instruction (F = 7726) has been executed by the computer (Fig. 8-316A and B).

2) The lower 17-bits of the RTC register (Fig. 8-316B) are equal to the lower 17-bits of the RTC monitor register (Fig. 8-316B).

When these two conditions are satisfied, an internal interrupt occurs. These two conditions must be repeated to allow another monitor interrupt to occur.

d) RTC MONITOR INTERRUPT FOR SPECIFIC DELTA TIME INCREMENT X. - The programmer performs the following:

1) Reads up the current contents of the RTC,

2) Adds the Delta Time Increment X,

3) Stores the resultant clock value in the RTC Monitor Register, and

4) Executes and "Enable RTC Monitor Interrupt" instruction F = 7726.

The RTC Monitor Interrupt will occur after the passage of X time, based on 1024 cycles per second.

Note the maximum time capable of being programmed between RTC Monitor Interrupts is $(2^{17} - 1)$ cycles of the RTC (at 1024 cps) of approximately 128 seconds.

Memory Address Allocations

Address (Octal)	Function
(00000-00077)	Main Memory address allocation
00000	Fault Entrance Register (with Automatic Recovery Switch in Neutral Position)
00001	Floating Point Overflow Interrupt Entrance Register
00002	Floating Point Underflow Interrupt Entrance Register
00003	Floating Point Divide Fault Interrupt Entrance Register
00004	Real Time Clock Monitor Interrupt Entrance Register
00006	Real Time Clock Overflow Interrupt Entrance Register
00007 through 00017	Memory Words
00020 through 00037	External Interrupt Entrance Registers for Channel 0 through 17
00040 through 00057	Input Monitor Interrupt Entrance Registers for Input Channels 0 through 17
00060 through 00077	Output Monitor Interrupt Entrance Registers for Output Channels 0 through 17
(00100-00277)	Control Memory address allocation
00100 through 00117	Input Buffer Control words for Channels 0 through 17
00120 through 00137	Output Buffer Control Words for Channels 0 through 17
00140 through 00157	External Function Buffer Control Words for Channels 0 through 17
00160	Real-Time Clock

00161 through 00167	B Registers
00170	Real Time Clock Monitor Register
00171 through 00177	Memory Storage
00200 through 00237	ESI Input Buffer Termination Word or CDM Reload Word storage for Channels 0 through 17
00240 through 00257	ESI External Function Buffer Termination Word Storage for Channels 0 through 17
00260 through	Memory Storage
00300 through 00477	Reserved for the optional additional 128 words of control memory
(00500-00537)	Main Memory address Allocation
00500 through 00517	External Function Monitor Interrupt Entrance Registers for Output Channels 0 through 17
00520 through 00537	External Interrupt Word Storage Registers for Input Channels 0 through 17
(00540-00577)	NDRO Bootstrap
(00600-77777)	Main Memory address allocation
00600 through 00617	Intercomputer Time-Out Entrance Register for Channels 0 through 17

SPERRY RAND

UNIVAC
FEDERAL SYSTEMS DIVISION

INTERCOMMUNICATION

TO: Distribution

FROM (NAME): M. L. Schwabel

LOCATION & DATE: VAFB 12 May 1970

DEPARTMENT: Systems Development

CARBONS:

SUBJECT: CS-1 Compiler

A feature has been added to the CS-1 system to allow for the resequencing of a #46 output tape. A CS-1 program on a #46 tape may now be given a new DECK ID and sequence numbers without the generation of a card deck.

OPERATING INSTRUCTIONS:

Bootstrap load the CS-1 system tape with KEY 2 set.

The program will now type out the following instructions:

46 on TT2, SCRATCH ON TT3

(SET KEY 3 FOR 2 COPIES, 2ND COPY ON TT4)

TYPE STARTING CARD NO.

Now type in 4 numeric characters for the beginning card number.

TYPE DECK ID

Type in 4 characters for a DECK ID. Any alphabetic or numeric character may be typed in (no special characters).

The resequenced #46 will be on TT3, and a second copy on TT4 if KEY 3 was set.

M. L. Schwabel
Systems Development

M. L. Schwabel

MLS:bg