UNIVAC

DATA PROCESSING DIVISION

1005

SYSTEM

REPORT PROGRAM GENERATOR 80/90

PROGRAMMERS REFERENCE

This manual is published by the UNIVAC Division of Sperry Rand Corporation in loose leaf format as a rapid and complete means of keeping recipients apprised of UNIVAC ® Systems developments. The UNIVAC Division will issue updating packages, utilizing primarily a page-for-page or unit replacement technique. Such issuance will provide notification of hardware and/or software changes and refinements. The UNIVAC Division reserves the right to make such additions, corrections, and/or deletions as in the judgment of the UNIVAC Division, are required by the development of its respective Systems.

PRINTED IN U.S.A.

CONTENTS

Page

CONTENTS 1 to 2

1. INTRODUCTION 1-1 to 1-1

2. GENERAL DESCRIPTION 2-1 to 2-3

2.1. OPERATION MNEMONIC 2-1

2.2. FIELDS 2-2

2.3. LABEL 2-2

2.4. COMMENTS 2-2

2.5. INCREMENT FIELDS 2-3

2.6. SYSTEM REFERENCES 2-3

3. SPECIFICATION OF FIELDS AND DATA 3-1 to 3-14

3.1. INTERNAL 3-1

3.2. INPUT (CARD) 3-6

3.3. OUTPUT 3-9

4. PROCESSING DATA 4-1 to 4-15

4.1. ARITHMETIC OPERATIONS 4-1

4.2. INTERNAL DATA TRANSFERS AND EDITING 4-6

5. INPUT/OUTPUT 5-1 to 5-5

5.1. READING CARDS 5-1

5.2. PRINTING 5-2

	Page
5.3. SPACING FORMS	5-2
5.4. SKIPPING FORMS	5-3
5.5. PUNCHING CARDS	5-4
6. PROGRAM CONTROL	6-1 to 6-24
6.1. PROGRAM START	6-1
6.2. PROGRAM HALT	6-2
6.3. SETTING CONDITIONS	6-3
6.4. SEQUENCE CONTROL	6-5
6.5. LOOP CONTROL	6-20
6.6. SUBROUTINES	6-22
7. COMMENTS	7-1 to 7-1
8. COPY SOURCE DECK	8-1 to 8-1
9. PROGRAM ORGANIZATION	9-1 to 9-1
10. OPERATING PROCEDURES	10-1 to 10-1
APPENDICES	
A. SYSTEM LABELS	A-1 to A-3
B. SYSTEM SWITCHES	B-1 to B-2
C. LEVEL BREAKS (SAMPLE)	C-1 to C-6
D. USE AND DEFINITION OF EDIT MASKS	D-1 to D-3

1. INTRODUCTION

The language structure for the UNIVAC 1005 Report Program Generator is almost identical in the 80 and 90 column systems; however, where there is a difference between the two systems it will be indicated. The UNIVAC 1005 Report Program Generator is a problem oriented, programming system designed to reduce substantially the time and effort necessary to translate general data processing requirements into detailed computer instructions. Little knowledge of computer programming is required other than the basic rules for writing programs in the UNIVAC 1005 Assembly Language. The 1005 Report Program Generator, on the basis of a series of statements provided by the user, produces a computer program which will prepare the desired reports. The user's statements, punched into cards, provide:

- The formats of the input (card) files--these files contain the information from which the report is to be prepared.

- The formats of the desired output reports--printed documents, punched summary cards, or both.

- The sequence of operations to be performed on the input files--arithmetic operations, input/output, data movements, controls.

The UNIVAC 1005 Report Program Generator provides a printed listing of both the user's input statements and the generated assembly language code. After the assembly phase, this generated code is an efficient ready-to-run object program.

2. GENERAL DESCRIPTION

The UNIVAC 1005 Report Program Generator translates source input statements into 1005 assembly language. Source input statements in 1005 assembly language are permissable. Each input statement consists of one operation mnemonic, optionally one or more operands, optionally one label, and optionally one comment. One or more assembly language statements are generated for each source input line, and the printed output of the Report Program Generator alternates between the source code and its generated code. Source input statements in 1005 assembly language are passed on to the assembler for further processing.

Source input code for the Report Program Generator is prepared using 1005 Assembly Language coding forms. The information on the forms is then punched into cards.

2.1. OPERATION MNEMONIC

The operation mnemonic (referred to as "the macro" below) is coded in columns 6 thru 10 of the source input; the first character of the macro is coded in column 6, and the remaining characters must follow with no intervening blanks. As an example, of the eight configurations shown below, only the first and the fifth are permitted.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
1 2 3 4 5	6 7 8 9 10													
	READ		correct											
	READ		not permitted											
	REA D		not permitted											
	R EAD		not permitted											
	SET		correct											
	SET		not permitted											
	SET		not permitted											
	S E T		not permitted											

When a source input statement is found to contain an invalid operation mnemonic, the statement is punched without alteration into the output deck and is printed with the message "NO MACRO" appended at the right of the printed line (in columns 82 thru 89). A "NO MACRO" printout will appear with all assembler mnemonic operations but should be ignored where an assembler mnemonic was intentionally used.

2.2. FIELDS

Fields A, B and C of the source input code (when required) are coded in columns 11 thru 20, 21 thru 30, and 31 thru 40, respectively. Columns 11 and 21 are reserved for indirect addressing designations (excepting comments and constants) and are otherwise unused. Indirect addressing is permitted in only those fields were specifically so stated in the macro descriptions of sections 4 and 6. Except in certain obvious cases, it is expected that labels will be coded in Fields A, B and C of the source input code.

2.3. LABEL

If a label is present on a source input statement, it will be present in the label field of the first assembly language statement generated by that macro; its value is then determined in the normal fashion by the assembler. This ensures that when transferring program control within a report program, a programmer need only specify (as the operand of his "JUMP") the label of the desired transfer point.

The label field is five characters, of which only the first three are used--the fourth and fifth are ignored. Thus AGE and AGENT are both interpreted as AGE: COL7 and COL8 are both interpreted as COL. Rules for construction of labels are the same as those for the 1005 Assembly Language.

2.4. COMMENTS

Comments may be specified by using a comment source input card and may be coded in columns 57 thru 80 of any source input card or columns 57 thru 90 for 90 column card. It is suggested that columns 62 through 66 contain a card sequence number. Comment source cards are retained throughout the assembly process; comments beyond column 61 are lost during pass 1 of the assembly for DC, comma (,), and comment operations. Comments on all other source cards begin in column 41 and may extend to column 56. If a label is present on a comment source card, its value will be the address of the next available location of memory, as determined by the 1005 Assembly Language Processor. This feature allows the definition of more than one label for any processing step.

2.5. INCREMENT FIELDS

The increment fields (columns 17-20, 27-30, and 37-40) should be coded with great care. Incrementation is always counted with respect to the left-hand value (MSL) of a label, and is not normally required, except for the TEST CHARACTER and MOVE CHARACTER macros. In the macro descriptions of sections 4, 5, and 6, whether Field A, B or C may be incremented is indicated for each field.

2.6. SYSTEM REFERENCES

System references, as used in this manual, are source input in any of the following six forms:

¤nnnn	decimal address
#aabb	octal representation of any pair of characters
$RRCCBk	row/column/bank (decimal)
RC	row/column/bank (internal machine format)
#yy	system switch
+LABEL	right-hand value of LABEL (the LSL)

Each of these forms is fully described in the UNIVAC 1005 Assembly Language manual. A list of system labels and switches appears in Appendix 1. System references are permitted as operands only where specifically so stated in the macro descriptions of sections 4, 5, and 6. When system references are not permitted, a field <u>must</u> be a label (either user-defined or system label).

With each macro description in sections 4, 5, and 6 is a summary table of operand characteristics for each required field. The three columns in the table refer respectively to:

- IA: indirect addressing to define Field A, B and C.

- SR: system references in addition to labels coded in Fields A, B and C.

- INC: using the increment field of Field A, B and C.
  YES means the feature is permitted and NO means it is not.

The three rows in the table refer respectively from top to bottom to Field A, Field B (when required) and Field C (when required).

3. SPECIFICATION OF FIELDS AND DATA

3.1. INTERNAL

3.1.1. Constants

Constants are specified by furnishing the name, the length, and the content of each, on a source statement with the operation mnemonic "DC". The name of the constant must satisfy the rules for constructing labels, and is coded in the label field.

For an 80 column system, the length of the constant does not include the character, if any, used to furnish a sign for the constant. If the sign is not furnished, it is assumed to be a plus. For a 90 column system, the length of the constant includes the character, if any, used to furnish a sign for the constant. The last character of the constant must be a zero for a negative constant and a blank for positive constants. The length of a negative constant for either an 80 or 90 column system must not exceed 25 characters.

If a constant of a greater length than 44 is desired, the excess of 44 is coded on the next sequential source statement with an operation mnemonic of comma (,). Additional characters beyond 88 are coded on additional comma-cards to a maximum of 961 characters (22 cards including the DC). The comma-cards may have a name of their own coded in their label fields, whether or not the entire constant has a name; the name on a comma-card refers to the characters on that card only, but the name on a DC-card refers to the entire constant.

Constants may be defined anywhere within a program without interfering with program sequence control; they are not loaded into the instruction area. Each constant may comprise any characters in the character set including blanks and algebraic signs (the algebraic sign of the constant is not considered a character of the constant for an 80 column system).

3.1.1.1. Define Constant (DC)

- Function:

  Enter a constant into 1005 storage.

- Where:

Operation = a two character mnemonic operation code (DC)

Operand 1
Field A = The number of characters in the constant.
Column 17 = the sign of the constant (80 column system), unused for a 90 column system.
INC = the characters of the constant

Operand 2 = additional constant characters starting in column 18 and extending to column 61. Negative constants extend only to column 42.

- Operand Characteristics:

		IA	SR	INC
Operand 1 -	Field A	NO	NO	NO
Operand 2 -	Field B	NO	NO	NO
	Field C	NO	NO	NO

- Examples:

(1) Enter the constant + 7 (80 column system)

(2) Enter the constant -10 (80 column system)

(3) Enter an alphabetic constant.

LABEL 1	OPERATION 6	OPERAND 1				OPERAND 2								COM
		I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
SEVEN	DC		1	+	7									
MTEN	DC		2	–	10									
MESAG	DC		29		MOV	E	THIS		TO	$	PR AN	D	PR	INT IT

3.1.1.2. CONTINUE CONSTANT (COMMA -,)

- Function:

Continue a constant that overflows for a previous Define Constant or Continue Constant instruction.

- Where:

Operation = a one character mnemonic operation code (,)
Operands 1 and 2 = consecutive character positions, beginning at column 18, and ending at column 61.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

NOTE: 90 Column Systems does not utilize column 17 for sign control. To achieve the same results as shown in line one of the example, column 12 would be coded with A 2, column 17 would be blank, and a blank in column 19 would be interpreted as a plus sign.

Line 2 of the example would be coded with a 3 in column 12, column 17 would be blank, and columns 18-20 would be coded 100 respectively. The zero in column 20 would be interpreted as a minus sign.

- Examples:

(1) Enter the first 44 characters of a 132 character constant.

(2) Enter the next 44 characters of the constant.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
LONG	DC		132		THI	S	CONS	T	ANT		INCLU	D	ES	THE
	,				COL	U	MNS A	T	TH	E	RIGH	T	UP	TO
THIRD	,				AND		INCLU	D	ING		COLUM	N	61	.

3.1.2. Work Areas

Work areas are specified by furnishing the name and length of each, on a source statement with the operation mnemonic "DA".

The name of the area is coded in the label field and must satisfy the rules for constructing labels. The area may be any length less than 962.

Work areas may be defined anywhere within a program without interfering with program sequence control; and these work areas are not reserved in the instruction area.

Work areas are not automatically cleared when the object program is loaded.

3.1.2.1. Define Work Area (DA)

- Function:

  Define a work area.

- Where:

  Operation = a two character mnemonic operation code (DA).

  Operand 1
  Field A = The number of character positions required in the work area.

  Operand 2 = Not used.

- Operand Characteristics:

		IA	SR	INC
Operand 1 -	Field A	NO	NO	NO
Operand 2 -	Field B	NO	NO	NO
	Field C	NO	NO	NO

- Examples:

  (1) Define an eight character work area named "TEMP".

  (2) Define a 12-character work area named "T2".

  (3) Define an 80 character work area named "CARD".

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
TEMP	DA		8											
T2	DA		12											
CARD	DA		80											

3.1.3. Accumulators

Accumulators are specified by furnishing the name and length of each, on a source statement with operation mnemonic "DA".

The name of the accumulator is coded in the label field and must satisfy the rules for constructing labels. The accumulator may not exceed 31 characters in length. Accumulators may be defined anywhere within a program without interfering with program sequence control; they are not reserved in the instruction area.

Accumulators are not automatically set to zero when the object program is loaded.

3.1.3.1. Define Accumulator (DA)

- Function:

  Define an accumulator.

- Where:

  Operation = a two character mnemonic operation code (DA)

  Operand 1
  Field A = The number of positions required in the accumulator.

  Operand 2 = Not used.

- Operand Characteristics:

		IA	SR	INC
Operand 1 -	Field A	NO	NO	NO
Operand 2 -	Field B	NO	NO	NO
	Field C	NO	NO	NO

- Examples:

  (1) Define a 19 digit accumulator named A19.

  (2) Define a 6 digit accumulator named A2.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
A19	DA		19											
A2	DA		6											

3.1.4. Edit Masks

Edit masks are specified by furnishing the name, the length, and the content of each, on a source statement with operation mnemonic "DC".

The name of the mask is coded in the label field and must satisfy the rules for constructing labels. A mask must not exceed 31 characters in length. If a field to be edited is larger than 31 characters; it must be edited in segments not exceeding 31 characters.

Edit masks may be defined anywhere within a program without interfering with program sequence control; they are not loaded into the instruction area.

For rules governing the use and definition of edit masks, see Appendix D.

3.1.4.1. Define Edit Mask (DC)

- Function:

  Define an Edit Mask.

- Where:

  Operation = a two character mnemonic operation code (DC)

  Operand 1
  Field A = number of positions in the mask.

  Operand 2
  Field B = The characters of the mask beginning in column 18
  and not extending beyond column 61.

- Examples:

  (1) Edit a 7-digit field into a 16-character dollars and cents field suppressing leading zeros and commas.

  (2) Same as above but insert asterisks for suppressed characters.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
THOU	DC		17		TOT	A	L $Δ¤	,	¤¤¤	.	¤¤≠			
MASK	DC		17		TOT	A	L $\¤	,	¤¤¤	.	¤¤≠			

3.2. INPUT (CARD)

An input card file is described by a set of source statements which must be supplied to the Report Program Generator in a group. Each distinct card file requires its own group, and any number of card files is permitted.

The first card of each group is a "DA" whose first (and only) operand is a systems label. (See Appendix A.). A label in columns 1 thru 5 is not permitted. References to the entire card input area for an 80 column system are made through use of the label "$R1" which is the system label of the card input area. References to the card input area for a 90 column system are made through use of the labels "$R1" which defines the first 45 columns of the card input area and "$R2" which defines the second 45 columns of the card input area.

The remaining cards of the group are field definitions. A field definition is a source statement with

(a) operation mnemonic dash (-),

(b) the name of the field coded in the label field, and

(c) decimal numbers coded in Fields A and B.

Field B is the length (number of characters) of the field being defined; Field A is the column number of the rightmost character of the field as it appears in the card. Every field of the input card <u>must</u> have a name. A field name may appear in the descriptions of more than one input file if the respective fields agree in position and length.

Every column of the input card file need not appear in a defined field.

3.2.1. DEFINE INPUT FIELD (-)

- Function:

  Define a field in the input file.

- Where:

  Operation = a one character operation code (-).

  Operand 1
  Field A = A number indicating the rightmost character of the field.

  Operand 2
  Field B = The number of card columns in the field. This number must not be higher than the number entered in Field A.

  Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples For an 80 Column System:

```
(1)  TYPE  in column   1        (length 1)
(2)  DATE  in columns  2 to  6  (length 5)
(3)  AGENT in columns  7 to 10  (length 4)
(4)  AMT   in columns 11 to 18  (length 8)
(5)  INFO  in columns  1 to 18  (length 18)
(6)  ITEM  in columns 51 to 80  (length 30)
(7)  XYƵ   in columns  1 to 80  (length 80)
```

NOTE: "XYƵ" may be used as the "name" of the card image area, as an alternative to using "$R1".

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	DA		$R1											
TYPE	–		1				1							
DATE	–		6				5							
AGENT	–		10				4							
AMT	–		18				8							
INFO	–		18				18							
ITEM	–		80				30							
XYZ	–		80				80							

- Examples For 90 Column System:

```
(1)  TYPE  in column   1        (length 1)
(2)  DATA  in columns  2 to  6  (length 5)
(3)  AGENT in columns  7 to 10  (length 4)
(4)  AMT   in columns 46 to 50  (length 5)
(5)  CITY  in columns 81 to 90  (length 10)
```

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	DA		$R1											
TYPE	–		1				1							
DATE	–		6				5							
AGENT	–		10				4							
	DA		$R2											
AMT	–		5				5							
CITY	–		45				10							

If a Transfer Descending (TD) is executed after a 90 column card is read so that column 46 is adjacent to column 45 the following card input file definition would apply:

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	DA		$RC											
TYPE	–		1				1							
DATE	–		6				5							
AGENT	–		10				4							
AMT	–		50				5							
CITY	–		90				10							

3.3. OUTPUT

3.3.1. Printing

3.3.1.1. Detail Lines

Each card of the input file (s) may be printed as a detail line by transferring the contents of the card input area to the left-most 80 (or 90) positions of the print output area and specifying a print operation. The MVALF and PRINT macros, which accomplish this action, are described in sections 4.2.1. and 5.2. To print a detail line for an 80 column system the following coding is required:

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
(optional)	MVALF		$R1				$80							
(optional)	PRINT													

In general, programmed clearing of the print output area prior to using is not necessary, the PRINT operation automatically clears all 132 positions to spaces.

3.3.1.2. Nondetail Lines

A nondetail line is described by a set of source statements which must be supplied to the Report Program Generator in a group. Each distinct nondetail line requires its own group, and any number of nondetail lines is permitted.

The first card of each group is a "DA" whose first (and only) operand is "$PR". (See next example.) A line label is not permitted; references to the entire print area are made through use of the label "$PR," which is the system label of the print output area.

The remaining cards of the group are field definitions. A field definition is a source statement with:

(a) operation mnemonic dash (-),

(b) the name of the field coded in the label field, and

(c) decimal numbers coded in Fields A and B.

Field B is the length (number of characters) of the field being defined; Field A is the print position number (from 1 to 132) of the rightmost character of the field. Every field must have a name. A field name may appear in the descriptions of more than one nondetail line if the respective fields agree in position and length.

When a line has been printed the entire contents of the print output area, "$PR," will automatically be cleared to blanks; information required following the printing must be specifically saved by moving it to other areas.

Constants that are to appear in the printed line must be transferred to the appropriate field each instance of printing, and must be defined as constants through use of the "DC" macro.

The sum of the lengths of the fields specified in the field definitions of any one group (excluding field overlapping) must not exceed 132, but may be any smaller number. In general, only the first line printed during execution need specify the contents of all 132 positions; automatic clearing of the print output area forces all otherwise unspecified print positions to be blank thereafter.

3.3.1.3. DEFINE PRINT FIELD (-)

- Function:

  Define a field in the print area.

- Where:

  Operation = a one character operation code (-).

  Operand 1
  Field A = The rightmost position of the field. The number entered must not exceed 132.

  Operand 2
  Field B = The number of characters in the field. This number must not be higher than the number entered in Label A.

  Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

(1) 15 positions for HEADR constant in 1 to 15
(2) 5 positions for DATE field in 16 to 20
(3) 20 positions for blanks (automatic if printing occurred previously)
(4) 25 positions for PROD field in 41 to 65
(5) 20 positions for blanks (automatic if printing occurred previously)
(6) 6 positions for CODER field in 86 to 91
(7) 41 positions for blanks (automatic if printing occurred previously)

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	DA		$PR											
HEADR	-		15				15							
DATE	-		20				5							
PROD			65				25							
CODER			91				6							

3.3.1.4. Printing Summary Cards

Each summary card may be printed by transferring the contents of the punch output area to the leftmost positions of the print output area and specifying a print operation. Transferring of the print image must occur <u>before</u> punching in this case, due to the automatic clearing of the punch output area following the actual punching. To print a summary card for an 80 column system the following code is required:

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
(optional)	MVALF		$P1				$80							
(optional)	PRINT													

3.3.2. Punching

3.3.2.1. Detail Reproducing

Each card of the input file (s) may be punched as part of the output file by transferring the contents of the card input area to the card output area and specifying a punch operation. The MVALF and PUNCH macros, which accomplish this action, are described in sections 4.2.1. and 5.5. To punch a detail line for an 80 column system the following coding is required:

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
(optional)	MVALF		$R1				$P1							
(optional)	PUNCH													

In general, programmed clearing of the punch output area prior to using is not necessary, as the PUNCH operation automatically clears all positions to spaces.

3.3.2.2. Nondetail (Summary) Punching

A nondetail (summary) card is described by a set of source statements which must be supplied to the Report Program Generator in a group. Each distinct nondetail card requires its own group, and any number of nondetail cards is permitted.

The first card of each group is a "DA" with $P1 in Field A. (See next example.) A label is not permitted. References to the entire card output area for an 80 column system are made through the use of the label "$P1," which is the system label of the card output area. References to the card output area for a 90 column system are made through use of the labels "$P1" and "$P2".

The remaining cards of the group are field definitions. A field definition is a source statement with

(a) operation mnemonic dash (-),

(b) the name of the field coded in the label field, and

(c) decimal numbers coded in Fields A and B.

Field B is the length (number of characters) of the field being defined; Field A is the column number of the rightmost character of the field as it will appear in the punched card. Every field must have a name, but not every column of the output card need appear in a defined field. A field name may appear in the descriptions of more than one output file if the respective fields agree in position and length.

3.3.2.3. DEFINE PUNCH FIELD (-)

- Function:

  Define a field in an output card.

- Where:

  Operation = a one character operation code (-).

  Operand 1
  Field A = The rightmost character of the field.

  Operand 2
  Field B = The number of characters in the field. This number must not be higher than the number entered in Field A.

  Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

```
(1)  18 positions for SLSMN field in positions  1 to 18
(2)   3 positions for BRNCH field in positions 20 to 22
(3)  11 positions for YRVLM field in positions 30 to 40
(4)  10 positions for NET   field in positions 50 to 59
(5)   7 positions for COMM field in positions 64 to 70
(6)   2 positions for CON constant in positions 79 to 80
```

Subfields for 80 Column System:

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	DA		$P1											
SLSMN	–		18				18							
BRNCH	–		22				3							
YRVLM	–		40				11							
NET	–		59				10							
COMM	–		70				7							
CON	–		80				2							

Subfields for 90 Column System:

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	DA		$P1											
SLSMN	–		18				18							
BRNCH	–		22				3							
YRVLM	–		40				11							
	DA		$P2											
NET	–		14				10							
COMM	–		25				7							
CON	–		35				2							

4. PROCESSING DATA

4.1. ARITHMETIC OPERATIONS

Five Macro Instructions are provided for arithmetic operations.

4.1.1. ADD (ADD)

■ Function:

Algebraically add a field or accumulator to a second field or accumulator. Both fields are assumend to be signed.

NOTES: (1) The maximum length of each operand is 31 locations. They need not be of the same length.

(2) The contents of Operand 1 are not affected by this instruction. The result is stored in Operand 2.

(3) One of three sign indicators (#AP, #AZ, #AM) will be set to reflect the resulting condition. The indicator set will remain set until the next arithmetic or round instruction is given.

(4) Arithmetic overflow will cause indicator #AF to be set.

■ Where:

Operation = a mnemonic operation code (ADD).

Operand 1
Field A = The label address of the first field or accumulator.

Operand 2
Field B = The label address of the second field or accumulator.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	YES
Operand 2 - Field B	YES	NO	YES
Field C	NO	NO	NO

- Examples:

(1) Add Field A to Accumulator 1

(2) Add Field TAX to field DEDCT

(3) Add Accumulator 3 to Accumulator 5

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	ADD		A				A1							
	ADD		TAX				DEDCT							
	ADD		A3				A5							

4.1.2. SUBTRACT (SUB)

- Function:

Algebraically subtract one field or accumulator from a second field or accumulator. Both fields are assumed to be signed.

NOTES: (1) The maximum length of each operand is 31 locations. They need not be of the same length.

(2) The contents of Operand 1 is not affected by this instruction. The result is stored in Operand 2.

(3) One of three sign indicators (#AZ, #AP, #AM) will be set to reflect the resulting condition. The indicator set will remain set until the next arithmetic or round instruction is given.

- Where:

Operation = a mnemonic operation code (SUB).

Operand 1
Field A = The label address of the first field or accumulator.

Operand 2
Field B = The label address of the second field or accumulator.

Field C = Not used.

- Operand Characteristics:

		IA	SR	INC
Operand 1 -	Field A	NO	NO	YES
Operand 2 -	Field B	YES	NO	YES
	Field C	NO	NO	NO

- Examples:

(1) Subtract NET from GROSS.

(2) Subtract Accumulator 2 from PAY.

(3) Subtract ABC from Accumulator 6.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	SUB		NET				GROSS							
	SUB		A2				PAY							
	SUB		ABC				A6							

4.1.3. MULTIPLY (MPY)

- Function:

Multiply a field or accumulator by a second field or accumulator, storing the result in a third area.

NOTES: (1) The signs of both operands are ignored and assumed to be positive for an 80 column system. The operand addresses should exclude the sign location (if any) for a 90 column system.

(2) The contents of the fields specified in Fields A and B will not be disturbed unless overlapped by the field specified in Field C.

- Where:

Operation = a mnemonic operation code (MPY).

Operand 1
Field A = The label address of a four (4) digit multipicand.

Operand 2
Field B = The label address of a six (6) digit multiplier.

Field C = The label address of a ten (10) digit product area.

- Operand Characteristics:

		IA	SR	INC
Operand 1 -	Field A	NO	NO	NO
Operand 2 -	Field B	NO	YES	YES
	Field C	NO	NO	NO

- Examples:

(1) Multiply A by B and store the result in C.

(2) Multiply PAY by RATE and store the result in Accumulator 4.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	MPY		A				B				C			
	MPY		PAY				RATE				A4			

4.1.4. MULTIPLY LONG (MPYL)

- Function:

Multiply a field or accumulator by a second field or accumulator, storing the result in a third area.

NOTES: (1) The signs of both operands are ignored and assumed to be positive for an 80 column system. The operand addresses should exclude the sign location (if any) for a 90 column system.

(2) The contents of the fields specified in Fields A and B will not be disturbed unless overlapped by the field specified in Field C.

- Where:

Operation = a mnemonic operation code (MPYL).

Operand 1
Field A = The label address of a nine (9) digit multiplicand.

Operand 2
Field B = The label address of an eleven (11) digit multiplier.

Field C = The label address of a twenty (20) digit product area.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	YES	YES
Field C	NO	NO	NO

- Examples:

(1) Multiply (long)CENTS by RATIO and store the result in LIRA.

(2) Multiply (long) Accumulator 2 by AMT and store the result in COST.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	MPYL		CENTS				RATIO				LIRA			
	MPYL		A2				AMT				COST			

4.1.5. DIVIDE (DIV)

- Function:

Divide a field or accumulator by a second field or accumulator, storing the result in a third area.

NOTES: (1) The signs of both operands are ignored and assumed to be positive for an 80 column system. The operand addresses should exclude the sign location (if any) for a 90 column system.

(2) The contents of the fields specified in Fields A and B will not be disturbed unless overlapped by the field specified in Field C.

- Where:

Operation = a mnemonic operation code (DIV).

Operand 1
Field A = The label address of a six (6) digit divisor.

Operand 2
Field B = The label address of an eight (8) digit dividend.

Field C = The label address of an eight digit area, to contain the eight (8) digit quotient.

- Operand Characteristics:

		IA	SR	INC
Operand 1 -	Field A	NO	NO	NO
Operand 2 -	Field B	NO	YES	YES
	Field C	NO	NO	NO

- Examples:

(1) Divide TOTAL by WEEKS and store the result in TEMP.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	DIV		WEEKS				TOTAL				TEMP			

4.2. INTERNAL DATA TRANSFERS AND EDITING

Nine Macro instructions are provided to transfer, edit, and modify data.

4.2.1. Data Transfers (alphanumeric and numeric)

4.2.1.1. MOVE ALPHANUMERIC (MVALF)

- Function:

Move an alphanumeric field or accumulator into a second field or accumulator.

- Where:

Operation = a mnemonic operation code (MVALF).

Operand 1
Field A = The label-address of the field or accumulator to be moved. The data stored in Operand 1 will not be altered by the instruction.

Operand 2
Field B = The label address of the field or accumulator to receive the data moved. Operand 2 must not include more than 961 storage locations, or more locations than are specified by Operand 1.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	YES	NO	YES
Field C	NO	NO	NO

- Examples:

(1) Move Accumulator 1 to GROSS.

(2) Move SALES to INCOM.

(3) Move DAY to Accumulator 3.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	MVALF		A1				GROSS							
	MVALF		SALES				INCOM							
	MVALF		DAY				A3							

4.2.1.2. MOVE NUMERIC (MVNUM)

- Function:

Move a field or accumulator into a second field or accumulator. Deletes all zone and sign bits for an 80 column system. Deletes all zero bits for a 90 column system.

- Where:

Operation = a mnemonic operation code (MVNUM).

Operand 1
Field A = The label address of the field or accumulator to be moved. The data stored in Operand 1 will not be altered by the instruction.

Operand 2
Field B = The label address of the field or accumulator to receive the data moved. Operand 2 must not include more than 961 storage locations or more locations than are specified by Operand 1.

Field C = Not used.

- Operand Characteristics:

		IA	SR	INC
Operand 1 -	Field A	NO	NO	NO
Operand 2 -	Field B	YES	NO	YES
	Field C	NO	NO	NO

- Examples:

(1) Move the decimal field VALUE to Accumulator 1.

(2) Replace NET by the absolute value of NET.

(3) Move the field INPUT to Accumulator 2, removing overpunches.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I.A.*	FIELD A 12	±	INC. 18	I.A.*	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	MVNUM		VALUE				A1							
	MVNUM		NET				NET							
	MVNUM		INPUT				A2							

4.2.2. Data Transfer with Edit Feature

4.2.2.1. MOVE WITH EDIT (MVEDT)

- Function:

Move an alphanumeric field or accumulator to a second field or accumulator, modifying the data transferred by a specified mask.

- Where:

Operation = a mnemonic operation code (MVEDT).

Operand 1
Field A = The label address of the field or accumulator to be moved. The data stored in Operand 1 will not be altered by the instruction.

Operand 2
Field B = The label address of the field or accumulator to receive the data moved.

Field C = The label address of the edit mask.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	YES	YES	YES
Operand 2 - Field B	YES	NO	YES
Field C	YES	YES	YES

NOTE: A description of edit masks and editing features is presented in Appendix D.

- Examples:

(1) Move TAX to OUTPT, editing with DOLLR.

(2) Move Accumulator 5 to SAVE, editing with ZERO.

(3) Move IN to TEMP, editing with Accumulator 4.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	MVEDT		TAX				OUTPT				DOLLR			
	MVEDT		A5				SAVE				ZERO			
	MVEDT		IN				TEMP				A4			

4.2.3. Filling - Work Areas

4.2.3.1. FILL AREA (FILL)

- Function:

Fill a field or accumulator with a specified character.

- Where:

Operation = a mnemonic operation code (FILL).

Operand 1
Field A = The label address of the area to be filled. Operand 1 must not exceed 961 characters in length. Fill begins in the leftmost position specified.

Operand 2
Field B = The character with which the area specified by Operand 1 is to be filled. This character is always entered in both columns 22 and 23.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	YES	NO	YES
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

(1) Fill TOTAL with zeroes.

(2) Fill Accumulator 6 with asterisks.

(3) Fill all but the two leftmost characters of HEADR with dashes.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	FILL		TOTAL				00							
	FILL		A6				**							
	FILL		HEADR	+	2		--							

4.2.4. Clearing-Work Areas

4.2.4.1. CLEAR AREA (CLEAR)

- Function:

Clear one, two, or three fields or accumulators to spaces.

- Where:

Operation = a mnemonic operation code (CLEAR).

Operand 1
Field A = The label address of a field or accumulator to be cleared. The maximum number of characters in this operand is 961. Clearing begins at the leftmost position specified.

Operand 2
Field B = The label address of a second field or accumulator to be cleared. The maximum number of characters in this operand is 961. Clearing begins at the leftmost position specified.

Field C = The label address of a third field or accumulator to be cleared. The maximum number of characters in this operand is 961. Clearing begins at the leftmost position specified.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	YES	NO	YES
Operand 2 - Field B	YES	NO	YES
Field C	YES	NO	YES

- Examples:

(1) Clear Accumulators 3 and 7 and field MM2

(2) Clear fields SALES, NET, and MONTH

(3) Clear field OUT and all but the four leftmost characters of MASK

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	CLEAR		A3				A7				MM2			
	CLEAR		SALES				NET				MONTH			
	CLEAR		OUT				MASK	+	4					

4.2.5. Moving a Single Character

4.2.5.1. MOVE CHARACTER (MVCHR)

- Function:

Move a character, contained in the instruction, to a single storage location.

- Where:

Operation = a mnemonic operation code (MVCHR)

Operand 1
Field A = The character to be moved. It is coded into columns 12 and 13.

Operand 2
Field B = The label address of the location to receive the specified character. The location specified may be part of a larger field or accumulator.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	YES	YES	YES
Field C	NO	NO	NO

- Examples:

(1) Move a 7 to the fourth character of COST

(2) Move a blank to the first character of field B32

(3) Move a - to the last character of Accumulator 1.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	MVCHR		77				COST	+	3					
	MVCHR						B32							
	MVCHR		--				+A1							

4.2.6. Rounding Arithmetic Results

4.2.6.1. ROUND (ROUND)

- Function:

Round a decimal value by half adjusting in the rightmost position of the area specified. This instruction will cause the value five (5) to be added in the rightmost position of the field. The result is then shifted one position to the right, dropping the rounded position and filling the leftmost position with a space code.

NOTES: (1) Rounding can be applied to positive numbers only.

(2) The ROUND instruction affects the sign indicators (#AP, #AZ and #AM).

- Where:

Operation = a mnemonic operation code (ROUND).

Operand 1
Field A = The label address of the field or accumulator to be rounded.

Operand 2
Field B = Not used.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

(1) Round Accumulator 2 one place

(2) Round LEVEL one place

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	ROUND		A2											
	ROUND		LEVEL											

4.2.7. Shifting Arithmetic Results

4.2.7.1. SHIFT FIELD (SHIFT)

- Function:

Shift the contents of a field or accumulator to the right a specified number of positions, filling the opened positions to the left with spaces. Caution must be used when shifting signed Fields.

- Where:

Operation = a mnemonic operation code (SHIFT).

Operand 1
Field A = The number of positions the field or accumulator is to be shifted. The maximum shift is 961 locations.

Operand 2
Field B = The label address of the field or accumulator to be shifted.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

(1) Shift accumulator 5 right 3 places.

(2) Round QOTNT 7 places.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	SHIFT		3				A5							
	SHIFT		6				QOTNT							
	ROUND		QOTNT											

4.2.8. Transfer of Sign

4.2.8.1. TRANSFER SIGN (SIGN)

- Function:

Transfer the algebraic sign of a field or accumulator to a second field or accumulator.

NOTES: (1) The sign is located in the zone bits of the rightmost character of a field, for an 80 column system. The rightmost character of a field contains only the sign for a 90 column system.

(2) Only the sign bits of the receiving field are altered for an 80 column system while the entire character is altered for a 90 column system.

- Where:

Operation = a mnemonic operation code (SIGN).

Operand 1
Field A = The label address of the field containing the sign to be transferred.

Field B = The label address of the field to receive the sign.

Field C = Not used.

- Operand Characteristics:

		IA	SR	INC
Operand 1 -	Field A	NO	NO	NO
Operand 2 -	Field B	NO	NO	NO
	Field C	NO	NO	NO

- Examples:

(1) Move the sign of YRNET to WKNET.

(2) Move the sign of Accumulator 2 to SIGN and make Accumulator 2 positive.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	SIGN		YRNET				WKNET							
	SIGN		A2				SIGN							
	MVNUM		A2				A2							

5. INPUT / OUTPUT

5.1. READING CARDS

5.1.1. READ A CARD (READ)

- Function:

Read the next card from the input file.

NOTE: Card images are always read into the following storage locations:

80 column - positions 1 - 80
90 column - positions 1 - 45 and 63 - 107

- Where:

Operation = a mnemonic operation code (READ).

Operand 1 = Not used.

Operand 2 = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:
  (1) Read the next card from the input file.
  (2) Move the 80 column card image to field CARD.
  (3) Move columns 41 thru 80 of an 80 column card image to HALF.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	READ													
	MVALF		$R1				CARD							
	MVALF		$R1				HALF							

5.2. PRINTING

5.2.1. PRINT (PRINT)

- Function:

Print a line, space the form one line and clear Print Storage. Print Storage is located at positions 161-292.

- Where:

Operation = a mnemonic operation code (PRINT).

Operand 1 = not used.

Operand 2 = not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

(1) Print the contents of the print area.

(2) Print the contents of the 132 character field HEADR.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	PRINT													
	MVALF		HEADR				$PR							
	PRINT													

5.3. SPACING FORMS

5.3.1. SPACE (SPACE)

- Function:

Advance the form in the printer one space without printing.

- Where:

  Operation = a mnemonic operation code (SPACE).

  Operand 1 = not used.

  Operand 2 = not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

  (1) Advance the carriage three (blank) lines.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	SPACE													
	SPACE													
	SPACE													

5.4. SKIPPING FORMS

5.4.1. SKIP (SKIP)

- Function:

  Skip the form in the printer to a specified line. The seven (7) code on the Form Control Tape is reserved as the "Home Paper" code.

- Where:

  Operation = a mnemonic operation code (SKIP).

  Operand 1
  Field A = The decimal equivalents of the bit configurations on the Form Control Tape. The number is entered into column 12.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
FIELD C	NO	NO	NO

- Examples:

(1) Skip to control tape configuration 5

(2) Print the contents of the print area as a header.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	SKIP		5											
	SKIP		7											
	PRINT													

5.5. PUNCHING CARDS

5.5.1. PUNCH (PUNCH)

- Function:

Punch the contents of the punch storage area, clearing Punch Storage to spaces.

NOTE: The Punch Storage area is always located in the following locations:

80 column - positions 293-372
90 column - positions 293-337 and 383-427

- Where:

Operation = a mnemonic operation code (PUNCH).

Operand 1 = not used.

Operand 2 = not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

(1) Punch a card

(2) Then punch the contents of field SUMRY.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	PUNCH													
	MVALF		SUMRY				$P1							
	PUNCH													

NOTE: 80 characters would be transferred in an 80 column system because "$P1" defines 80 characters. 45 characters would be transferred in a 90 column system because "$P1" defines 45 characters.

6. PROGRAM CONTROL

Twenty two macro instructions are available for controlling the flow of processing. They are necessary for starting and halting a program, for setting and testing conditions, for operations with subroutines, and for altering the execution sequence of program.

The program is normally executed in the order in which the processing statements are presented to the Report Program Generator. However, numerous macro instructions are provided to allow the programmer to alter the sequence under a variety of conditions.

A control statement is not required to indicate the end of source input code statements. The Report Program Generator halts when the last input card is read.

6.1. PROGRAM START

6.1.1. END PROGRAM LOAD (END)

- Function:

  Terminate loading of the program and begin execution. This instruction is identical to the END directive of the 1005 Assembler and is not part of a loaded program. It has no function during compiling or assembling and does not terminate the reading of source input cards.

- Where:

  Operation = a mnemonic operation code (END).

  Operand 1
  Field A = The label address of the first processing instruction to be executed.

  Operand 2 = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	YES	YES
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

(1) Terminate loading and start the program at STEP.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	END		STEP											

6.2. PROGRAM HALT

6.2.1. HALT PROCESSING (HALT)

- Function:

Stop program execution and light a Halt Indicator. This instruction becomes part of the object program. When the RUN key on the UNIVAC 1005 Console is depressed following a HALT, processing will continue at the source statement immediately following the source HALT statement.

- Where:

Operation = a mnemonic operation code (HALT).

Operand 1
Field A = The code for the desired System Switch. A listing of System Switches is provided in Appendix B.

Operand 2 = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	YES	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

(1) Halt execution and turn on HALT Indicator 2. Continue to RESTT if RUN button is pressed.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	HALT		#H2											
	GOTO		RESTT											

6.3. SETTING CONDITIONS

6.3.1. SET CONDITION (SET)

- Function:

Set or reset one, two or three System Switches.

- Where:

Operation = A mnemonic operation code (SET).

Operand 1
Field A = The code for the System Switch to be set or reset. A listing of System Switches is provided in Appendix B.

Operand 2
Field B = The code for a second System Switch to be set or reset.

Field C = The code for a third System Switch to be set or reset.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	YES	NO
Operand 2 - Field B	NO	YES	NO
Field C	NO	YES	NO

- Examples:

(1) Set even parity for tape operation.

(2) Set sense switches one and two and tape unit No. 1.

(3) Reset sense switch one and set tape unit No. 2.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	SET		#EP											
	SET		#+2				#+1				#S1			
	SET		#-1				#S2							

6.4. SEQUENCE CONTROL

6.4.1. Testing for Conditions

6.4.1.1. TEST CONDITION (TEST)

- Function:

  Test a System Switch for a specified setting. Transfer program sequence control if the condition is present.

- Where:

  Operation = A mnemonic operation code (TEST).

  Operand 1
  Field A = The letters "COND".

  Operand 2
  Field B = The code for the desired System Switch. A listing of System Switches is provided in Appendix B.

  Field C = The label address of the next instruction to be executed if the condition is present. If the condition tested is not met, control is transferred to the next instruction in sequence.

- Operand Characteristics:

	IA	SR	INC
Operand - Field A	NO	NO	NO
Operand - Field B	NO	YES	NO
Field C	NO	YES	YES

- Examples:

  (1) Test for arithmetic overflow; if set go to OVER routine.
  (2) Test for and reset parity error and if set go to PAPER.
  (3) Test for alteration switch No. 2 set and if set go to ON2.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	TEST		COND				#AF				OVER			
	TEST		COND				#PE				PAPER			
	TEST		COND				#-2				ON2			

6.4.1.2. TEST NEGATIVE (TEST)

- Fuction:

  Test the contents of a single storage location for a negative sign. A field is identified as negative by the presence of an X bit in the rightmost character position for an 80 column system. The rightmost character is tested for a 90 column system.

- Where:

  Operation = A mnemonic operation code (TEST).

  Operand 1
  Field A = The letters "NEG".

  Operand 2
  Field B = The label address of the character location to be tested.

  Field C = The label address of the next instruction to be executed if the location tested is negative. If the location is not negative, control is transferred to the next instruction in sequence.

- Operand Characteristics:

		IA	SR	INC
Operand 1 -	Field A	NO	NO	NO
Operand 2 -	Field B	NO	YES	YES
	Field C	NO	YES	YES

- Examples:

  (1) If field PRFT is <u>negative</u>, transfer control to BOMB.
  (2) If the third character of INPUT is <u>positive</u>, go to MAN.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	TEST		NEG				+PRFT				BOMB			
	TEST		NEG				INPUT	+	2		NEXT			
	GOTO		MAN											
NEXT														

6.4.1.3. TEST CHARACTER (TEST)

- Function:

Test a storage location for the presence of a specific character.

- Where:

Operation = A mnemonic operation code (TEST).

Operand 1
Field a = The specific character for which the test is being made, not the address of the test character. This character is entered into column 12 and 13.

Operand 2
Field B = The label address of the storage location being tested.

Field C = The label address of the next instruction to be executed if the location tested contains the character specified in Operand 1. If the character is not present, control is transferred to the next instruction in sequence.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	YES	NO
Operand 2 - Field B	YES	YES	YES
Field C	NO	YES	YES

- Examples:

(1) If the last character of UNIT is not a 7, go to NO7.

(2) If the second character of UNIT is *, go to SPECL.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	TEST		77				+UNIT				NEXT			
	GOTO		NO7											
NEXT	TEST		**				UNIT	+	1		SPECL			

6.4.2. Comparing for Conditions

6.4.2.1. COMPARE ALPHANUMERIC (COMPA)

■ Function:

Perform an alphanumeric comparison on two fields or accumulators. This comparison is made on a match/non-match basis. Either the EQUAL or NOT-EQUAL indicator is set as a result of this instruction. These indicators remain set until the next compare instruction. They may be tested by the IFEQ and IFNE instructions.

■ Where:

Operation = A mnemonic operation code (COMPA).

Operand 1
Field A = The label address of a field or accumulator.

Operand 2
Field B = The label address of a field or accumulator to be compared with Operand 1.

Field C = Not used.

■ Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	YES	NO	YES
Field C	NO	NO	NO

■ Examples:

(1) Compare INPUT with NAME

(2) Compare Accumulator 1 with INPUT

(3) Compare the rightmost 6 characters of Accumulator 2 (9 characters) with SCALE

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	COMPA		INPUT				NAME							
	COMPA		A1				INPUT							
	COMPA		SCALE				A2	+	3					

6.4.2.2. TRANSFER EQUAL (IFEQ)

- Function:

If a previous comparision set the EQUAL indicator, transfer program control.

NOTES: (1) The IFEQ instruction can be used in conjunction with any of the compare instructions.

(2) Indicator settings are not affected by the IFEQ instructions.

- Where:

Operation = A mnemonic operation code (IFEQ).

Operand 1
Field A = The label address of the next instruction to be executed, if the EQUAL indicator is set. If the EQUAL indicator is not set, the next instruction in sequence is executed.

Operand 2
Field B = Not used.

Field C = Not used.

- Operand Characteristics:

		IA	SR	INC
Operand 1 -	Field A	NO	YES	YES
Operand 2 -	Field B	NO	NO	NO
	Field C	NO	NO	NO

- Examples:

(1) If the previous compare set the EQUAL indicator, transfer program control to step EQU.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	IFEQ		EQU											

6.4.2.3. TRANSFER NOT-EQUAL (IFNE)

- Function:

If a previous comparison set the NOT-EQUAL indicator, transfer program control.

NOTE: Indicator settings are not affected by the IFNE instruction.

- Where:

Operation = A mnemonic operation code (IFNE).

Operand 1
Field A = The label address of the next instruction to be executed, if the NOT-EQUAL indicator is set. If the indicator is not set, the next instruction in sequence is executed.

Operand 2
Field B = Not used.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	YES	YES
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

(1) If the previous compare set the NOT-EQUAL indicator, transfer program control to PHASE.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	IFNE		PHASE											

6.4.2.4. COMPARE NUMERIC (COMPN)

- Function:

Perform an algebraic comparison on two numeric fields or accumulators. This instruction always results in the setting of one of three indicators. These indicators are:

(1) EQUAL,

(2) LESS-THAN, and

(3) GREATER-THAN.

If either the LESS-THAN or GREATER-THAN indicator is set, the NOT-EQUAL indicator will also be set. All of these indicators will remain set until the next compare instruction is given. They may be tested by any IF instruction. The fields being compared must be signed fields.

- Where:

Operation = A mnemonic operation code (COMPN).

Operand 1
Field A = The label address of a numeric field or accumulator.

Operand 2
Field B = The label address of the field or accumulator to be compared with Operand 1.

NOTES: (1) Operands 1 and 2 must be of equal length.

(2) Resulting indicator settings will be made with respect to Operand 1; e.g., if Operand 1 is greater, the GREATER-THAN indicator will be set.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	YES	NO	YES
Field C	NO	NO	NO

- Examples:

  (1) Compare SUM with TOTAL.

  (2) Compare Accumulator 3 with SUM.

  (3) Compare HYTE with all but the first digit of TOP.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	COMPN		SUM				TOTAL							
	COMPN		A3				SUM							
	COMPN		HYTE				TOP	+	1					

6.4.2.5. TRANSFER CONTROL IF LESS THAN (IFLT)

- Function:

  If the previous comparison set the LESS-THAN indicator, transfer program control.

  NOTES: (1) Indicator settings are not affected by the IFLT instruction.

  (2) IFLT instruction may be used only in conjunction with the COMPN and COMPM instructions.

- Where:

  Operation = A mnemonic operation code (IFLT).

  Operand 1
  Field A = The label address of the next instruction to be executed, if the LESS THAN indicator is set. If the indicator is not set, the next instruction in sequence is executed.

  Operand 2
  Field B = Not used.

  Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	YES	YES
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Example:

(1) If the previous compare set the LESS-THAN indicator, transfer program control to step ENUF.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	IFLT		ENUF											

6.4.2.6. TRANSFER CONTROL IF GREATER THAN (IFGT)

- Function:

If the previous comparison set the GREATER-THAN indicator, transfer program control.

NOTES: (1) Indicator settings are not affected by the IFGT instruction.

(2) The IFGT instruction may be used only in conjunction with the COMPN and COMPM instructions.

- Where:

Operation = A mnemonic operation code (IFGT).

Operand 1
Field A = The label address of the next instruction to be executed, if the GREATER-THAN indicator is set. If the indicator is not set, the next instruction in sequence is executed.

Operand 2
Field B = Not used.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 = Field A	NO	YES	YES
Operand 2 = Field B	NO	NO	NO
Field C	NO	NO	NO

- Example:

  (1) If the previous compare set the GREATER-THAN indicator, transfer control to step PRICE.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	IFGT		PRICE											

6.4.2.7. COMPARE MAGNITUDE (COMPM)

- Function:

  Perform an absolute magnitude compare on two numeric fields or accumulators. This instruction will always result in the setting of one of three indicators. These indicators are:

  (1) EQUAL,

  (2) LESS-THAN and

  (3) GREATER-THAN.

  If either the LESS-THAN or the GREATER-THAN indicator is set, the NOT-EQUAL indicator will also be set. These indicator settings will not be altered until the next compare instruction is given. All of these indicators may be tested by any IF instruction. The sign (if any) is excluded from consideration and has no effect on the result of the compare magnitude.

- Where:

  Operation = A mnemonic operation code (COMPM).

  Operand 1
  Field A = The label address of a field or accumulator.

  Operand 2
  Field B = The label address of a field or accumulator to be compared to Operand 1.

  NOTE: Indicator settings are made with respect to Operand 1; e.g., if Operand 1 is greater than Operand 2, the GREATER-THAN indicator will be set.

  Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	YES	NO	YES
Field C	NO	NO	NO

- Examples:

(1) Compare the magnitudes of Accumulator 2 and AMT.

(2) Compare the magnitudes of TIME and Accumulator 3.

(3) Compare the magnitudes of the field FIFTY and all but the left three characters of AMT.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	COMPM		A2				AMT							
	COMPM		TIME				A3							
	COMPM		FIFTY				AMT	+	3					

6.4.3. Explicit Sequence Change

6.4.3.1. TRANSFER CONTROL (GOTO)

- Function:

Unconditionally transfer program control to the instruction specified by Operand 1.

- Where:

Operation = A mnemonic operation code (GOTO).

Operand 1
Field A = The label address of the next instruction to be executed.

Operand 2
Field A = Not used.

Field B = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	YES	YES
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Example:

(1) Transfer program control to program step TAX.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	GOTO		TAX											

6.4.4. Implicit Sequence Change (Level Breaks)

Level break operations (LEVLA, LEVLN, LEVLM) are used to conditionally transfer program control to a specified subroutine when consecutive values of a field in an input file differ. The condition is that the ALLOW switch is ON.

The ALLOW switch is an internal switch set to either ON or OFF, and is set to ON automatically when an object program is loaded. Whenever a level break occurs (that is, consecutive values of a field differ), the ALLOW switch is set to OFF before program control is transferred to the appropriate subroutine. The only means of returning the switch to the ON position is the use of the ALLOW BREAK macro instruction.

The first time thru the level break operation, no break can occur; processing is limited to saving the first value of the designated field. The second time and thereafter, testing for differing consecutive values occurs and a break is possible. In all cases, the current value of the tested field is saved for the next comparison.

When a level break occurs and program control transfers to the specified subroutine, the EXIT operation of that subroutine is automatically set to return program control to the operation sequentially following the level break operation (LEVLA, LEVLN, or LEVLM).

Level break operations should be coded in order of decreasing priority of fields tested. Appendix C is an example of the normal scheme. An ALLOW BREAK operation precedes the first level break operation, but there may be intervening operations between any two level break operations or between any LEVL operation and ALLOW BREAK. More than one ALLOW BREAK may appear in a program.

6.4.4.1. LEVEL BREAK ALPHANUMERIC (LEVLA)

- Function:

  Transfer program control if a control break (level break) occurs on an alphanumeric field. Comparison is based on all six bits of each character.

- Where:

  Operation = A mnemonic operation code (LEVLA).

  Operand 1
  Field A = The label address of a field to be tested for a level break.

  Operand 2
  Field B = The label address of a BEGIN statement of a subroutine.

  Field C = The number of characters in the field specified by Operand 1.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	YES
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

  (1) Alphanumeric level breaks in FIELD (8 characters) are to be processed by COST.

  (2) Alphanumeric level breaks in the seven rightmost characters of FIELD are to be processed by TIME.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	LEVLA		FIELD				COST				8			
	LEVLA		FIELD	+	1		TIME				7			

6.4.4.2. LEVEL BREAK NUMERIC (LEVLN)

- Function:

  Transfer Program Control if a control break (level break) occurs on a numeric field. The comparison of fields is algebraic. Rules governing the comparison of the control field are the same as the compare numeric statement (see Section 6.4.2.4).

- Where:

  Operation = A mnemonic operation code (LEVLN).

  Operand 1
  Field A = The label address of a field to be tested for a level break.

  Operand 2
  Field B = The label address of a BEGIN statement of a subroutine.

  Field C = The number of characters in the field specified by Operand 1.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	YES
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

  (1) Numeric level breaks in VALUE (11 characters) are to be processed by WHLSL.

  (2) Numeric level breaks in the four rightmost characters of VALUE are to be processed by XYZ.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	LEVLN		VALUE				WHLSL				11			
	LEVLN		VALUE	+	7		XYZ				4			

6.4.4.3. LEVEL BREAK MAGNITUDE (LEVLM)

- Function:

  Transfer program control if a control break (level break) occurs on a numeric field. The comparison of fields is numeric. Rules governing the comparison of the control field are the same as the compare magnitude statement (see Section 6.4.2.7).

- Where:

  Operation = A mnemonic operation code (LEVLM).

  Operand 1
  Field A = The label address of a field to be tested for a level break.

  Operand 2
  Field B = The label address of a BEGIN statement of a subroutine.

  Field C = The number of characters in the field specified by Operand 1.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	YES
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

  (1) Magnitude level breaks in A3 (12 characters) are to be processed by TOTAL.

  (2) Magnitude level breaks in the two rightmost characters of A3 are to be processed by MAJOR.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	LEVLM		A3				TOTAL				12			
	LEVLM		A3	+	10		MAJOR				2			

6.4.4.4. ALLOW BREAK (ALLOW)

Function:

Turn on the internal switch that permits the occurrence of level breaks.

- Where:

Operation = A mnemonic operation code (ALLOW).

Operand 1
Field A = The letters BREAK.

Operand 2
Field B = Not used.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Example:

(1) Turn on the internal switch which permits the occurrence of level breaks.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	ALLOW		BREAK											

6.5. LOOP CONTROL

6.5.1. LOOP (LOOP)

- Function:

Repeat the execution of a group of instructions a specified number of times.

NOTES: (1) Each execution of the LOOP statement will automatically decrement the value in Field A by 01. In an 80 column system control will be transferred to the label specified in Field B if the value of Field A is positive or zero. When the value in Field A becomes negative, control is transferred to the next instruction in sequence. In a 90 column system control will be transferred to the next instruction in sequence when the value in Field A is reduced to zero.

(2) When the execution has been repeated the desired number of times, control is transferred to the next instruction in sequence, and the loop operation is automatically reset.

- Where:

Operation = A mnemonic operation code (LOOP).

Operand 1
Field A = A two digit number (01 to 99) representing the number of times the loop is to be executed.

Operand 2
Field B = The label address of the first instruction of the group.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	YES	YES
Field C	NO	NO	NO

- Examples:

(1) Repeat the group of operations starting with MANY 6 times from below (80 column system).

(2) When the execution has been repeated the desired number of

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
MANY														
	LOOP		05				MANY							

NOTE: The number specified in Field A in a 90 column system would be 6 to execute the group 6 times.

6.6. SUBROUTINES

A subroutine, as here defined, is a group of operations coded sequentially, whose first operation is BEGIN (a labeled operation) and whose last operation is EXIT (also labeled). A subroutine is characterized by the following property: When Field B of any LEVL macro is the label of a BEGIN, the return exit address is automatically inserted into the corresponding EXIT whenever a level break occurs to that BEGIN.

In all other cases of transferring control to a BEGIN (by "GOTO", "TEST", "IF", "SBRTN" and "LOOP"), the address in EXIT can and must be set by the programmer. This is most easily accomplished through use of the "SBRTN" operation, described below. Nesting of subroutines is not allowed between any pair of BEGIN's and at least one EXIT must appear.

6.6.1. BEGIN A SUBROUTINE (BEGIN)

■ Function:

Define the beginning of a subroutine. Begin must be labeled. When executed as a result of a LEVL instruction it sets up the return address in the corresponding exit.

■ Where:

Operation = A mnemonic operation code (BEGIN).

Operand 1
Field A = Not used.

Operand 2
Field B = Not used.

Field C = Not used.

■ Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

■ Example:

(1) Begin a subroutine.

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	BEGIN													

6.6.2. EXIT FROM A SUBROUTINE (EXIT)

- Function:

Provide a variable GOTO of which the operand is set by the program during execution.

NOTE: The exit address can be specified by the programmer in either of two ways, each of which requires that the EXIT operation be labeled. They are:

(1) SBRTN (described below) and,

(2) the use of an alphanumeric move from the operand of a D1 to the modified label (see Example 2).

- Where:

Operation = A mnemonic operation code (EXIT).

Operand 1
Field A = Not used.

Operand 2
Field B = Not used.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

(1) Set up a variable exit.

(2) Move the address of RTRN to the EXIT of Example (1).

LABEL	OPERATION	OPERAND 1				OPERAND 2								
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
LABEL	EXIT													
	MVALF		NAME				LABEL	+	3					
NAME	D1		RTRN											

6.6.3. ENTER A SUBROUTINE (SBRTN)

- Function:

Transfer program control to a subroutine and set an exit address.

- Where:

Operation = A mnemonic operation code (SBRTN).

Operand 1
Field A = The label address of the first step of the subroutine.

Field B = The label address of an EXIT or GOTO operation, which is to contain the return address. The address of this operand is not incremented.

Field C = The address to which the subroutine will return when it executes the EXIT or GOTO named in Field B. If no entry is made in Field C, return from the subroutine is made automatically to the next sequential step following the SBRTN operation.

- Operand Characteristics:

		IA	SR	INC
Operand 1 -	Field A	NO	YES	YES
Operand 2 -	Field B	NO	NO	NO
	Field C	NO	YES	YES

- Examples:

(1) Transfer control to TAX and set the exit to normal return.

(2) Transfer control to TAX and set the exit to CMPUT (the EXIT corresponding to BEGIN labeled TAX is labeled EX1).

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	SBRTN		TAX				EX1							
	SBRTN		TAX				EX1				CMPUT			

7. COMMENTS

7.1. . (PERIOD)

- Function:

Comment cards will be printed and punched.

Comments may start in column 8 and should not extend beyond column 61.

- Example:

LABEL	OPERATION	OPERAND 1				OPERAND 2								
1	6	I. A. *	FIELD A 12	±	INC. 18	I. A. *	FIELD B 22	±	INC. 28		FIELD C 32	±	INC. 38	41
	.		THESE		CAR	D	S ARE		MER	E	LY RE	P	ROD	UCED
	.		AND P	R	INT	E	D.							

8. COPY SOURCE DECK

8.1. COPY (COPY)

- Function:

Begin reproducing the source deck as comments in the output deck to be assembled, or terminate reproducing. When the copy off feature is used, the volume of cards to be sent through the assembler is reduced but the source card which activates the generation of assembler code will not appear on the final listing.

- Where:

Operation = A mnemonic operation code (COPY).

Operand 1
Field A = The word ON or the word OFF to turn the feature on or off.

Operand 2
Field B = Not used.

Field C = Not used.

- Operand Characteristics:

	IA	SR	INC
Operand 1 - Field A	NO	NO	NO
Operand 2 - Field B	NO	NO	NO
Field C	NO	NO	NO

- Examples:

(1) Turn COPY on.
(2) Turn COPY off.

LABEL	OPERATION	OPERAND 1				OPERAND 2									
		I. A. *	FIELD A	±	INC.	I. A. *	FIELD B	±	INC.		FIELD C	±	INC.	
1	6		12		18		22		28		32		38	41
	COPY		ON											
	COPY		OFF											

9. PROGRAM ORGANIZATION

Before writing a program, it is advisable to prepare a complete description of the problem with particular attention to input and output layout. With this done, it is a simple task to assign names to the various fields and to write field definitions for the input and output areas.

Having prepared all of the field descriptions required, a list of constants, edit masks, and accumulators should be prepared. The input/output layouts should be consulted to be certain each accumulator has been defined with a sufficient length to handle the maximum possible size.

The sequence of operations in the object program is determined by the sequence in which they are written and may be altered as directed by program control directives.

Below are some conventions which should be followed to assure correct and efficient object coding.

- A page overflow routine should be executed at the very beginning to assure proper initial positioning of paper before processing begins.

- Normally, the reading of an input item will be immediately followed by a test for the end of the run.

- The LEVL directives normally occur before any further processing is specified, since a control break indicates that the last card of a control group has already been processed. After the LEVL operations have been written in their proper sequence, they should be followed by the processing which is to be done when no control break has occurred. This will conserve storage and result in a more efficient program.

- Sequence control directives should be preceded by those operations which are to be performed regardless of the result of the transfer. This will conserve storage and result in a more efficient program.

- End of job processing, to which control is transferred as a result of the test mentioned in second convention (listed above), should include execution of the highest priority level break subroutine (assuring execution of all lower ones) and the page overflow routine.

10. OPERATING PROCEDURES

The Report Program Generator produces a 1005 Assembly Language intermediate output deck from the user's source statements; the intermediate deck is then assembled (as it stands) and a final object code deck produced.

The program is run by placing the Report Program Generator in the card read hopper, adding the source statements to the hopper, and pressing START, CLEAR, FEED, and RUN.

Each source statement will be printed and the assembly language code generated, if any, will be printed. The generated code is always punched, and the source code will or will not be punched according to the most recent setting of the COPY switch (ON is set if not otherwise specified).

The assembly phase should be executed as described in the 1005 Assembly Language Manual.

APPENDIX A. SYSTEM LABELS

1.1. BOTH 80 AND 90 COLUMN SYSTEMS

Standard Label	Predesignated Area	Decimal Address	Row/Column Address
$IR	Instruction Register	None	R32/C1-R32/C7,B1
$XR	Register X	None	R32/C1-R32/C31,B2
$TR	Translate Table Area	1828-1891 3750-3813	R28/C30-R30/C31,B2 or R28/C30-R30/C31,B4
$AR	Arithmetic Register	None	R32/C1-R32/C31,B1
$CC	Instruction Control Counter	None	R32/C8-R32/C9,B1
$PR	Print Area (132 Chars.)	161-292	R6/C6-R10/C13,B1
$X2	Fill Register	None	R2/C32,B1

1.2. 80 COLUMN SYSTEMS LABELS ONLY

$R1	80 Col. Read Area	1-80	R1/C1-R3/C18,B1
$R2	80 Character Area	81-160	R3/C19-R6/C5,B1
$RC	80-Col. Read Code Image	1-160	R1/C1-R6/C5,B1
$P1	80 Col. Punch Area	293-372	R10/C14-R12/C31,B1
	(Also 80 Col. Read-Punch Read Area)		
$P2	80-Col. Read-Punch Punch Area	373-452	R13/C1-R15/C18,B1
$PC	80-Col. Punch Code Image	293-452	R10/C14-R15/C18,B1
$Z1	80-Col. Read-Punch Code Image Read	293-452	R10/C14-R15/C18,B1

$Ƶ2	80 Col. Read-Punch Code Image Punch	453-612	R15/C19-R20/C23,B1
$80	FIRST (leftmost) 80 Chars. of Print Output Area	161-240	R6/C6-R8/C23,B1
$BM	First Character beyond Read-Punch Punch Output Area	453	R15/C19,B1

1.3. 90 COLUMN SYSTEM ONLY

$R1	First 45 Columns Read Input	1-45	R1/C1-R2/C14,B1
$R2	Second 45 Columns Read Input	63-107	R3/C1-R4/C14,B1
$P1	First 45 Columns Punch	293-337	R10/C14-R11/C27,B1
$P1	First 45 Columns Read-Punch Read	293-337	R10/C14-R11/C27,B1
$P2	Second 45 Columns Punch	383-427	R13/C11-R14/C24,B1
$P2	Second 45 Columns Read-Punch Read	383-427	R13/C11-R14/C24,B1
$Ƶ1	First 45 Columns Read-Punch Punch	338-382	R11/C28-R13/C10,B1
$Ƶ2	Second 45 Columns Read-Punch Punch	428-472	R14/C25-R16/C7,B1
$BM	First location after Read-Punch Punch	473	R16/C8,B1
$R3	45 locations after Column 45 of Read Input	46-90	R2/C15-R3/C28,B1
$RC	First 90 locations of Bank 1	1-90	R1/C1-R3/C28,B1
$Ƶ1	45 locations after Columns 45 of Punch	338-382	R11/C28-R13/C10,B1
$PC	90 locations beginning at Column 1 of Punch	293-382	R10/C14-R13/C10,B1
$ƵC	90 locations beginning at Column 1 of Read-Punch Punch	338-427	R11/C28-R14/C24,B1

$90	First 90 locations of Print Storage	161-250	R6/C6-R9/C2,B1

APPENDIX B. SYSTEM SWITCHES

Two Characters with a single bit (numbered 1 to 12)

SWITCH	BIT	JUMP ON CONDITION	SET CONDITION
#FF	1 (X)	Form Overflow	
#AF	2 (Y)	Arithmetic Overflow	
#+2	3 (8)	Sense #2 Set	Sense #2
#+1	4 (4)	Sense #1 Set	Sense #1
#-2	5 (2)	Alternate #2 On	Reset Sense #2
#-1	6 (1)	Alternate #1 On	Reset Sense #1
#IN	7 (X)	Interrupt	
#UA	8 (Y)	Unit Alert	
#PE	9 (8)	Parity Error Always (Resets)	
#AP	10 (4)	Arithmetic Plus	
#AƵ	11 (2)	Arithmetic Ƶero	
#AM	12 (1)	Arithmetic Minus	
#SO	1 (X)		Odd Parity
#SE	2 (Y)		Even Parity
	7 (X)		Reset PPT for channel 8 punching
	8 (Y)		Set PPT for channel 8 punching
#S2	9 (8)		Servo #2
#S1	10 (4)		Servo #1
#H2	11 (2)		Indicator #2 and Halt

SWITCH	BIT	SET CONDITION
#H1	12 (1)	Indicator #1 and Halt
#H3	both 11 (2) & 12 (1)	Indicators #1 & #2 & Halt
#ET	both 1 (x) & 2 (y)	End of Tape Reached

APPENDIX C. LEVEL BREAKS

1.1. TYPICAL APPROACH TO LEVEL BREAKS

Level Breaks (sometimes called Control Breaks with Rolling Totals) are programmed to recognize changes in control fields and to enter subroutines that output totals. The convention established for checking multiple levels of control fields is to check the most important control field (major) first because a break in a more important control field forces breaks in all control fields of lesser importance (intermediate and minor).

A program that was to break controls for a Major Total on the State Field, an Intermediate Total on the County Field and a Minor Total on the City Field would check the State Field first. When the control break for State occurs, the programmer knows that a County and City break has been forced and three (3) total subroutines have to be executed.

The programmer will usually perform the following steps (not necessarily in this sequence) when he recognizes a Major Control Break:

- Condition the Exit Statement of the Minor Total Subroutine to enter the Intermediate Total Subroutine.

- Condition the Exit Statement of the Intermediate Total Subroutine to enter the Major Total Subroutine.

- Execute the Minor Total Subroutine.

- Exit to the Intermediate Total Routine.

- Execute the Intermediate Total Routine.

- Exit to the Major Total Subroutine.

- Execute the Major Total Subroutine.

- Replace the control field storage areas with the data from the card that caused the control break.

- Jump to the subroutine that processes the First Card of the Group or execute the instruction following the Minor Compare for level break.

The program segment shown in the example follows these conventions, taking advantage of the instruction generations offered by the RPG. The example can be used to familiarize the programmer with the techniques used for Rolling Totals regardless of the number of control fields or control breaks.

1.2. LEVEL BREAK

Flow of Processing

1.2.1. ALLOW BREAK

This statement is the first instruction of the Level Break series. It sets the ALLOW BREAK SWITCH to the ON position.

1.2.2. NO BREAK

The program will pass through the LEVEL statements in the order given if no change in control fields is encountered. Detail processing will follow.

1.2.3. MINOR BREAKS

When the program recognizes a change in the minor control field only, the following events will occur:

- The ALLOW BREAK SWITCH will be set to the OFF position.

- The address of the detail statement which sequentially follows the minor LEVEL statement is transferred to the EXIT statement of the minor total subroutine.

- The program executes the minor total subroutine, transfers control to detail processing via the EXIT statement of the minor subroutine.

1.2.4. INTERMEDIATE BREAKS

When the program recognizes a change in the intermediate control field, the following events will occur:

- The ALLOW BREAK SWITCH will be set to the OFF position.

- The address of the minor LEVEL statement is transferred to the EXIT statement of the intermediate subroutine.

- The program executes the SBRT statement of the intermediate total subroutine. This statement transfers the address of the first processing step of the intermediate total subroutine (INT + 7 in the example) to the EXIT statement of the minor total subroutine.

- The program executes the minor total subroutine and transfers control to the first processing step of the intermediate total subroutine via the EXIT statement of the minor total subroutine.

- The program executes the intermediate total subroutine and transfers control to the minor LEVEL statement via the EXIT statement of the intermediate total subroutine.

- The program will execute the minor LEVEL statement. No break will occur, because the intermediate break set the ALLOW BREAK SWITCH to the OFF position.

- The program executes the minor LEVEL statement so that the new minor compare field can be stored.

- The program advances to the next sequential processing step following the minor LEVEL statement.

1.2.5. MAJOR BREAKS

When the program recognizes a change in the major control field, the following events will occur:

- The ALLOW BREAK SWITCH will be set to the OFF position.

- The address of the intermediate LEVEL statement is transferred to the EXIT statement of the major total subroutine.

- The program executes the SBRT statement of the major total subroutine. This statement transfers the address of the first processing step of the major total subroutine (MAJOR + 7 in the example) to the EXIT statement of the intermediate total subroutine.

- The program executes the SBRT statement of the intermediate total subroutine. This statement transfers the address of the first processing step of the intermediate total subroutine (INT + 7 in the example) to the EXIT statement of the minor total subroutine.

- The program executes the minor total subroutine and transfers control to the first processing step of the intermediate total subroutine via the EXIT statement of the minor total subroutine.

- The program executes the intermediate total subroutine and returns to the first processing step of the major total subroutine.

- The program executes the major total subroutine and transfers control to the intermediate LEVEL statement via the EXIT statement of the major total subroutine.

- The program will execute the intermediate and minor LEVEL statements. No break will occur, because the major break set the ALLOW BREAK SWITCH to the OFF position.

- The program executes these LEVEL statements, so that the new intermediate and minor compare fields can be stored.

- The program advances to the next sequential processing step following the minor LEVEL statement.

```
LABEL OPERATION OPERAND 1                OPERAND 2                                    COMMENTS
                I.A.* FIELD A ± INC.     I.A.* FIELD B ± INC.  FIELD C ± INC.
1    6          12              18       22              28    32          38         41             56 57

     . EXPLANATION APPEARS IN COLS 41-56***EXPLANATION***
     . DETAIL INSTRUCTIONS              INITIALIZE PROG.
X    ALLOW BREAK                        LEVEL RTN      1
A    LEVLA STATE     MAJOR     4        CHECK MAJOR    2
B    LEVLA COUNT     INT       6        CHECK INTER    3
C    LEVLA CITY      MINOR     6        CHECK MINOR    4
D    . SEE BLOCK D;COULD BE USED TO     UPDATE THE     -
     . UPDATE MINOR TOTAL(S), READ, GO TO XMINOR TOTAL.
MAJORBEGIN                              MAJOR BEGINS NOW
     SBRTN INT       EX2                PUT"MAJ+7" INTO-
     .                                  EX2, GO TO INT .
     . MAJOR TOTAL ROUTINE STARTS HERE, OUTPUT MAJOR
     . USUALLY ADD TO FINAL,OUTPUT,CLEARTOTALS,
EX1  EXIT                               LEAVE FROM MAJ.
INT  BEGIN                              INT BEGINS NOW
     SBRTN MINOR     EX3                PUT INT+7 IN EX3
     .                                  GO TO MIN
     . ADD TO MAJOR,OUTPUT, CLEAR INT   OUTPUT INT TOTAL
EX2  EXIT                               LEAVE FROM INT
MINORBEGIN                              MINOR BEGINS NOW
     . ADD TO INT,OUTPUT,CLEAR MINOR    UPDATE AND OUTPT
EX3  EXIT                               GO TO"D" IF NO
     .                                  INTERMEDIATE,OR
     .                                  GO TO INT+7 IF
     .                                  SET BY SBRTN
```

Figure C-1. Sample Coding

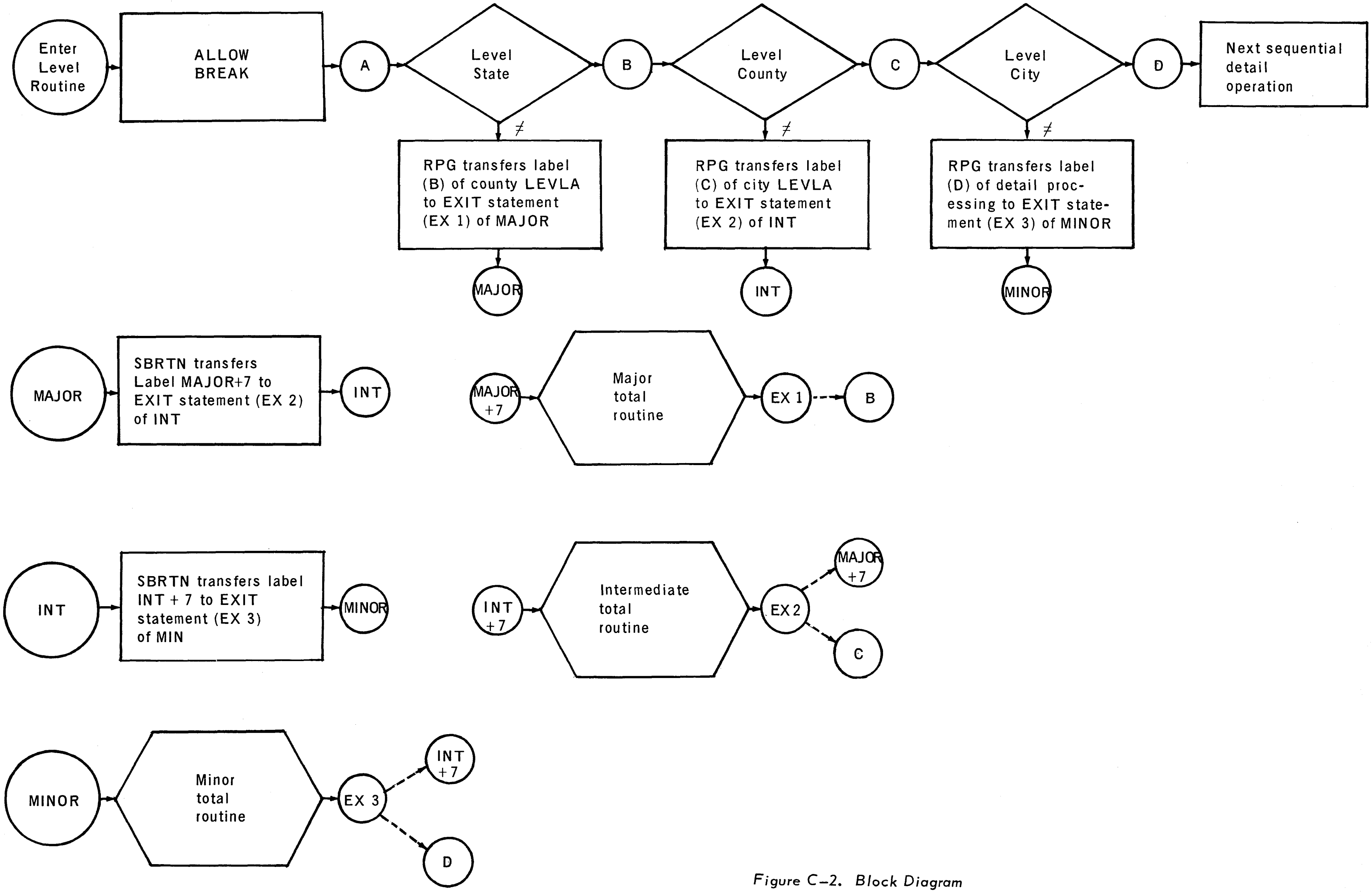
Enter Level Routine
ALLOW BREAK
A
Level State
B
Level County
C
Level City
D
Next sequential detail operation
≠
RPG transfers label (B) of county LEVLA to EXIT statement (EX 1) of MAJOR
≠
RPG transfers label (C) of city LEVLA to EXIT statement (EX 2) of INT
≠
RPG transfers label (D) of detail processing to EXIT statement (EX 3) of MINOR
MAJOR
INT
MINOR
MAJOR
SBRTN transfers Label MAJOR+7 to EXIT statement (EX 2) of INT
INT
MAJOR +7
Major total routine
EX 1
B
INT
SBRTN transfers label INT + 7 to EXIT statement (EX 3) of MIN
MINOR
INT +7
Intermediate total routine
EX 2
MAJOR +7
C
MINOR
Minor total routine
EX 3
INT +7
D

Figure C–2. Block Diagram

APPENDIX D. USE AND DEFINITION OF EDIT MASKS

1.1. USE AND DEFINITION OF EDIT MASKS

Edit masks are normally used to insert specified characters into, and/or delete certain leading characters from, numeric fields. Insertions may also be made into alphanumeric fields. One field is "edited" into a second through use of the Move-with-edit macro, MVEDT. An edit mask may not exceed a length of 31 characters.

Internal counters examine the contents of successive locations of the "sending" field, the "receiving" field, and the edit mask, beginning with the "leftmost" character (the MSL) of each. The editing process is terminated when either the last character of the edit mask has been reached, or case (2) described below occurs. The receiving field is not automatically cleared to blanks by the editing process. During editing, the characters comprising the edit mask have the following meaning:

All characters except unequal, lozenge, left-slash, and delta: Send the current character of the edit mask to the current character of the receiving field and increment by one position the edit mask and receiving field counters.

Unequal (≠): Terminate the editing process and do not send a character to the current character of the receiving field.

Lozenge (¤): Move the current character of the sending field to the current character of the receiving field, and advance all three counters by one position.

Delta (△): Turn on the zero-suppress feature (blank fill), do not increment any counters, and then continue as the lozenge (¤). The zero-suppress feature is turned off by the next "current character" of either the sending field or edit mask which is neither a zero, space, nor a comma. While the feature is in effect, zeros and commas sent from either the sending field or edit mask are changed to blanks during transmission to the receiving field.

Left-slash (\): Turn on the zero-suppress feature (asterisk fill), do not increment any counters, and then continue as the lozenge (¤). The zero-suppress feature is turned off by the next "current character" of either the sending field or edit mask which is neither a zero, space, nor a comma. While the feature is in effect, zeros, spaces, and commas sent from either the sending field or edit mask are changed to asterisks during transmission to the receiving field.

1.2. EXAMPLES

```
A   DC   18   TOTAL b̸ b̸ $¤¤,¤¤¤.¤¤≠
B   DC   18   TOTAL b̸ b̸ $Δ¤,¤¤¤.¤¤≠
C   DC   18   TOTAL b̸ b̸ $\¤,¤¤¤.¤¤≠
D   DC   18   TOTAL b̸ b̸ $Δ¤,¤\¤.¤¤≠
E   DC    7   1234567
F   DC    7   0000002
G   DC    7   0034567
H   DC    7   0000567
```

```
A on E produces    TOTAL $12,345.67
B on E    "        TOTAL $12,345.67
C on E    "        TOTAL $12,345.67
D on E    "        TOTAL $12,345.67

A on F produces    TOTAL $00,000.02
B on F    "        TOTAL $      .02
C on F    "        TOTAL $******.02
D on F    "        TOTAL $    **.02

A on G produces    TOTAL $00,345.67
B on G    "        TOTAL $   345.67
C on G    "        TOTAL $***345.67
D on G    "        TOTAL $   345.67

A on H produces    TOTAL $00,005.67
```

```
B on H produces         TOTAL $     5.67

C on H     "            TOTAL $*****5.67

D on H     "            TOTAL $    *5.67
```

UNIVAC

DIVISION OF SPERRY RAND CORPORATION