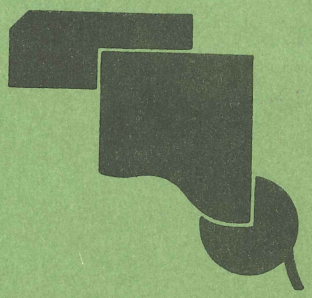
GENERAL ELECTRIC
COMPUTERS

GE-200 Series General Assembly Program II



GENERAL ELECTRIC

CPB-1180

GE-200 SERIES
GENERAL ASSEMBLY
PROGRAM II

REFERENCE MANUAL

Program Numbers

CD225F1.006
CD225F1.007
CD225F1.008
CD225F1.009
CD225F1.010
CD225F1.011
CD225F1.012
CD225F1.013
CD225F1.014

May 1963

Rev. January 1966

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

CPB-1180

PREFACE

This manual has been prepared by the General Electric Computer Equipment Department for the programmer and operator of the GE-200 Series Information Processing Systems. It is one of the many programming manuals available to users of General Electric systems. The manual describes and tells how to use the General Assembly Program II.

The programs described in this manual are available from the Program Library according to the user's configuration of equipment as follows:

CD225F1.006/7 - This is an 8k version of the General Assembly Program II and contains processing facilities for DATANET-15*, Document Handler (1200dpm), GE-235 Auxiliary Arithmetic Unit, and Multiple Tape Lister instructions.

CD225F1.008 - This is an 8k BRIDGE II-compatible version of the General Assembly Program II, and it contains processing facilities for DATANET-15, Document Handler (1200dpm), GE-235 Auxiliary Arithmetic Unit, and Multiple Tape Lister instructions.

CD225F1.009/10 - This is a 4k version of the General Assembly Program II and does not contain processing facilities for DATANET-15, Document Handler (1200dpm), GE-235 Auxiliary Arithmetic Unit, and Multiple Tape Lister instructions.

CD225F1.011/12 - This is an 8k version of the General Assembly Program II which contains processing facilities for the 300 lpm printer.

CD225F1.013/14 - This is a 4k version of the General Assembly Program II which contains processing facilities for the 300 lpm printer.

This publication supersedes the General Assembly Program II manual dated January 1966.

Suggestions and criticisms relative to form, content, purpose, or use of this manual are invited. Comments may be sent on the Document Review Sheet in the back of this manual or may be addressed directly to Documentation Standards and Publications, B-90, Computer Equipment Department, General Electric Company, 13430 North Black Canyon Highway, Phoenix, Arizona, 85029.

*DATANET, Reg. Trademark of the General Electric Company.

CONTENTS

I. THE GENERAL ASSEMBLY PROGRAM 1

How the Assembly Works 1
The Source Programs 3
The Assembly Passes 3
Pass 0 3
Pass 1 3
Pass 2 3
Programming the Source Program 4
Coding Sheet 5
Symbol Field 5
Opr (operation) Field 6
Instruction Line 6
Assembly Control Line 6
Constant Line 6
Operand Field 7
Mnemonic 7
Decimal 7
Symbol 8
X Field 8
Remarks Field 9
Sequence Field 9
Pseudo Instructions 10
Constant Line 10
Control Line 19
Additional Pseudo Instructions 27
Relocatable Assembly Instructions 27
Relative Addressing 29
Relative Addressing Using Plus or Minus 30
Relative Addressing Using ORG 30
Relative Addressing Using Asterisk 31
Multiple Relative Addressing 31
General Assembly Program--Detected Coding Errors 32
Pass 0 Detected Coding Errors 32
Pass 1 Detected Coding Errors 32
Pass 2 Detected Coding Errors 33

II. ASSEMBLY OPERATIONS 36

System Configurations 36
A-Register Input 38
Switch Settings 38
Card-to-Card Operations with Minimum Card Equipment 41
Pass 0 Card-to-Card 41
A Table of Special Symbols 42

Pass 0 Messages 43
Recovery of a Card Read Error. 43
Other Halts or Loops. 44
Pass 1 Card-to-Card 44
A Sorted Symbol Table (ST2) 45
Pass 1 Messages 45
Pass 2 Card-to-Card 46
Object Program 47
Pass 2 Messages 49
Card Operations with Magnetic Tape and Printer Equipment 49
Pass 0 Cards with Magnetic Tape and Printer 50
Pass 0 Messages 50
Pass 1 Cards with Magnetic Tape and Printer 51
Sorted Symbol Table (ST2) 51
Pass 1 Messages 51
Pass 2 Cards with Magnetic Tape and Printer 51
Object Program 51
Pass 2 Messages 52
Alternate Assembly Configuration 52
Pass 0 Alternate Assembly 52
Pass 1 Alternate Assembly 52
Pass 2 Alternate Assembly 53
Systems Tapes 53
Operating with Systems Tape 53
Master Deck 53
Tape Format 54
Instructions for Generating Systems Tape 54
Systems Tape Operations. 55
Pass 0 Systems Tape 55
Pass 1 Systems Tape 55
Pass 2 Systems Tape 55
Addition of Service Routines. 56
Addition of Symbolic Subroutines 56
Subroutines Are Called 56
Multiple Assemblies 58
When an Absolute Object Program 58
When a Relocatable Object Program 58
End-of-Tape Card 58
Bridge II Compatible Systems 58
Systems Tape Setup 58
Symbolic Subroutines 59
Assembly 60
Modifications to General Assembly Program II 60
Symbol Table Length 60
Symbol Table 1 61
Symbol Table 2 61
Priority Control Channel 61
Vacuum Pocket Retrofit 63
System Tape Controller Modification 63
Relocatable Object Programs 63
General 63
Calculation of Checksum 64
Perforated Tape Assembly 66
General 66
Perforated Tape Input/Output Conversion Tables 66

ILLUSTRATIONS

1. Diagram of the General Assembly Program II 2
2. Sample General Assembly Program Coding Sheet 4
3. Symbol Fields 5
4. Instruction Lines 6

5. Control Lines 6
6. Constant Lines 7
7. Operand Field Instruction Line 7
8. Operand Field Control and Constant Lines 8

9. X(Index) Field 9
10. Remarks and Sequence Field 9
11. Pseudo-Instructions for Constant Lines 10
12. Control Lines for Pseudo-Instruction 19

13. Transfer Card End Instruction 26
14. Printer Listing from Pass 0 Coding Errors 32
15. Diagram of General Assembly Program II Programs 37
16. Pass 0--Symbolic Program Deck 42

17. Special Symbol Table (ST1) 42
18. Printer Listing of Symbol Errors, Pass 0 43
19. Pass 0 Messages 44
20. Sorted Symbol Table (ST2) 45

21. Pass 1 Messages 46
22. Object Program Card Deck 47
23. Object Program Assembly, Pass 2 48
24. Pass 2 Messages 49

25. Pass 0 Magnetic Tape Messages 51
26. Relocatable Instruction Card 65
27. Relocatable Assembly Listing Format 65
28. Perforated Tape Character Set 8-Channel, Friden Flexowriter Model SPD 68

I THE GENERAL ASSEMBLY PROGRAM II

The General Assembly Program has been constantly improved to provide more features and greater flexibility. General Assembly Program II is the latest version to which several new operation and pseudo-operation codes have been added. The system transforms the symbolic mnemonic codes used by the programmer in coding the source program into a ready-to-execute machine language (or object) program. The features of the system are:

1. Memory addresses may be assigned either by using decimal or octal numeric notation or by using symbolic notation, whichever results in maximum convenience to the programmer.

2. Additions to and/or deletions from the coding are easily made when the program is being written.

3. During assembly, either of the following output formats may be selected:

   a. Absolute numeric coding which is executed only in a predetermined fixed area of the computer memory

   b. Relocatable numeric coding which is executable in any area of the computer memory (Its ultimate position in memory is decided by its loader routine.)

4. Many types of clerical or language errors are detected and listed, thus reducing the amount of machine debugging time needed in new programs.

5. A listing of the assembled program is printed out, including error indications, symbolic listings, and assigned memory addresses.

6. The system is completely compatible with the BRIDGE II, Operating Service System providing retrieval and execution of programs stored on magnetic tapes.

General Assembly Program II is a tool that provides accurate, well documented programs, which can be quickly debugged and placed in operation, resulting in significant savings to the user.

HOW THE ASSEMBLY WORKS

The General Assembly Program produces an object program by successively processing the symbolic coding of the source program. The source program is processed with the assembly programs through the computer in three passes. The output produced by each pass forms part of the input for the next pass. A flow chart of a typical General Assembly Program is shown in Figure 1. Note that the outputs from both pass 0 and pass 1 are used as inputs to pass 2.

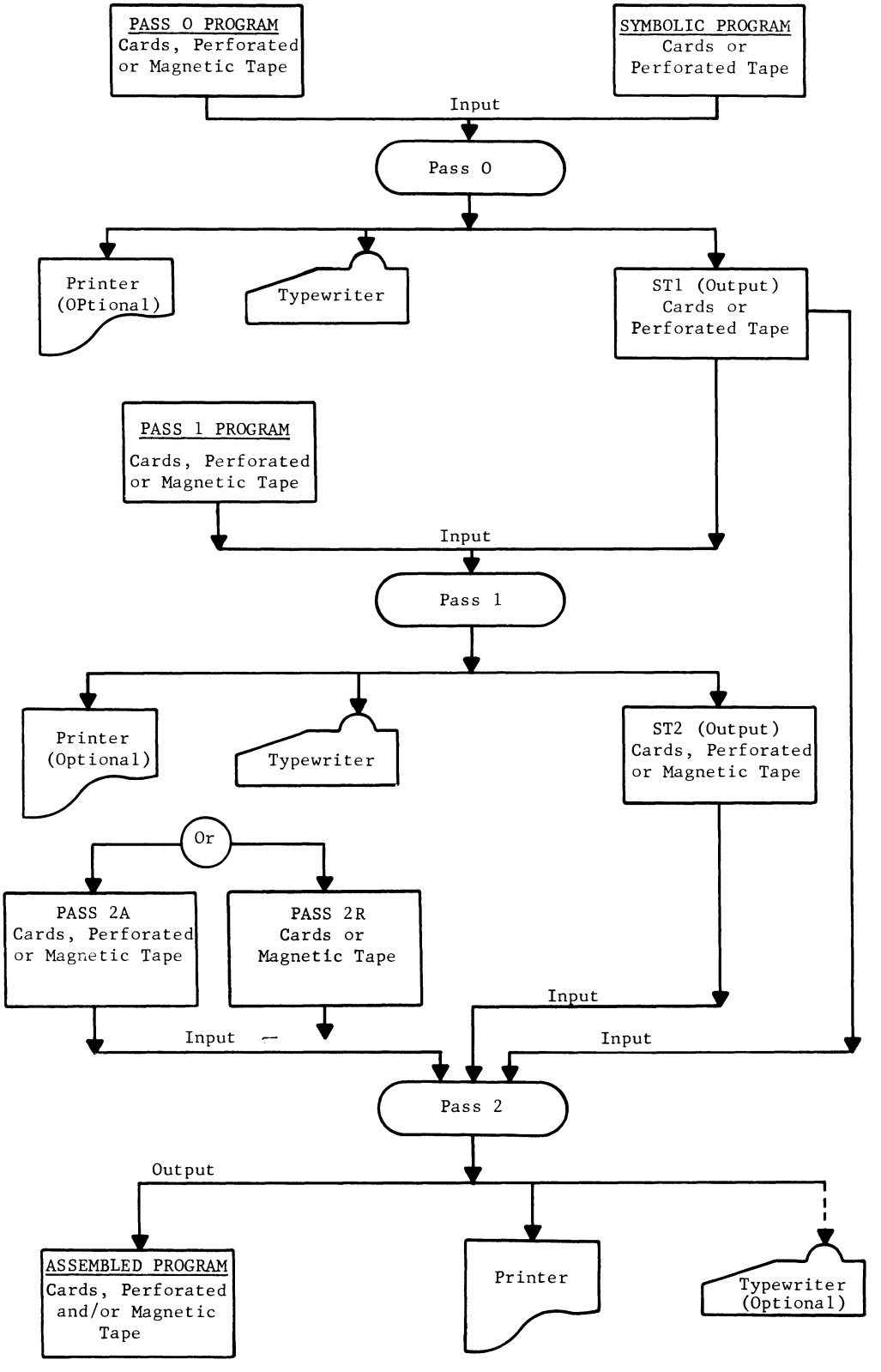


Figure 1. Diagram of the General Assembly Program II

The program produces as its final output an object program in any of the following media or combinations of them:

1. A program deck of punched cards in binary or octal notation
2. A program magnetic tape
3. A program perforated tape
4. A printed listing of the object program

The Source Program

The source program is written using pseudo-instructions, symbolic addresses and mnemonics of operations to be performed. Cards punched for each line of the programmer's coding sheet become the symbolic source deck used in each of the assembly passes.

The Assemby Passes

Each of the three passes of the assembly performs certain functions which are designed to provide the programmer with maximum information concerning the object program.

PASS 0 accomplishes 3 operations:

1. It prepares the source program for subsequent use in other assembly passes.
2. A table of special symbols is formed containing all symbolic operands appearing in the Input/Output (I/O) instructions. This symbol table is referred to as ST1.
3. It checks all symbolic names in the input deck and identifies undefined symbols, multiple symbols and those with no reference.

Pass 0 further provides a listing on the printer, or if desired, the listing can be printed on the console typewriter.

The output from Pass 0 will be a packed symbolic program deck starting with sequence number 20000 and a special symbolic table (ST1) deck with sequence numbers starting at 10000. The program decks will be punched into cards, written on magnetic tape or punched on perforated tape depending upon the configuration of the system.

PASS 1 uses the output of pass 0 to assign memory locations to all symbols in the input source deck. It forms a sorted table of these symbols and the numeric values assigned. This table is referred to as ST2 and is listed by the printer or on the console typewriter.

The output from pass 1 (ST2) can be punched into cards, written on magnetic tape, or punched on perforated tape, again depending upon the configuration of the user's system.

PASS 2 uses the outputs of pass 0 (Packed 20000 deck, ST1 symbols table) and pass 1 (ST2 symbol table) as input to the final pass. Pass 2 does the complete assembly of each instruction as specified by the source program.

Pass 2 can be either absolute (2A) or relocatable (2R), since both are not used in the same assembly. However their end result is the same. The output of pass 2 is the assembly listing with indicated errors and the object program itself on punched cards, perforated tape, or magnetic tape.

PROGRAMMING THE SOURCE PROGRAM

The General Assembly Program coding sheet (CK 34), illustrated in Figure 2, contains representative pseudo-operation codes, symbolic addresses, and mnemonics.

GENERAL ELECTRIC
COMPUTER EQUIPMENT DEPARTMENT
PHOENIX, ARIZONA

225 GENERAL ASSEMBLY PROGRAM CODING SHEET

PROGRAMMER: GE Coder | PROGRAM: Manufacturing Cost - Run #1 | DATE: | PAGE OF:

	Symbol	Opr	Operand	X	REMARKS	Sequence
1		ORG	1000			5
2	ROUND	DEC	50		ROUNDING CONSTANT	10
3	RATE	DEC	1025		PROCESS COST PER ITEM	15
4	MASK	OCT	37764300		MASKING CONSTANT FOR MOD ROUTINE	20
5	FIVE	DEC	5			25
6	CON#1	DEC	-34			30
7	START	LDX	ZERO	2	ZERO INDEX REGISTERS 2 & 3	35
8		LDX	ZERO	3		40
9		SPB	CRDIN	1	CARD READ SUBROUTINE	45
10		BRU	CRDEOF		CARD END-OF-FILE RETURN	50
11		SPB	STRIP	1	BCD-BINARY CONVERSION ROUTINE	55
12		DEC	CRD		CARD IMAGE ORIGIN	60
13		DEC	1		BEGINNING OF FIELD	65
14		DEC	4		FIELD SIZE	70
15		STA	AMT#1		ITEMS PREVIOUSLY PROCESSED	75
16		SPB	STRIP	1		80
17		DEC	CRD			85
18		DEC	16			90
19		DEC	4			95
20		STA	AMT#2		ITEMS CURRENTLY PROCESSED	100
21		ADD	AMT#1			105
22		STA	SUM		TOTAL ITEMS PROCESSED	110
23		MAQ				115
24		MPY	RATE		PROCESS COST PER ITEM	120
25		DAD	ROUND		ROUND PROCESS COST	125

CK 34 (4-67)

Figure 2. Sample General Assembly Program Coding Sheet

Each of the fields is designated from left to right as:

Symbol
Opr (abbreviation for operation)
Operand
X (abbreviation for index)
Remarks
Sequence

indicates to the assembly program that certain operations are to be performed.

The Coding Sheet

The following simple rules ensure that correct results will be obtained if followed by the programmer in his source program. The purpose of each field and its applicable rules are as follows:

SYMBOL field is an address. The following rules apply to its use:

Rule 1. Symbols can vary from one to six characters in length and be any combination of alphabetic and numerics.

Rule 2. Plus (+) or minus (-) characters are not allowed in the symbol field due to the principle of relative addressing.

Rule 3. Symbols used must contain at least one nonnumeric character.

Rule 4. Symbols may start at any point within the field because leading and inserted blanks are ignored by the General Assembly Program.

The assembly program assigns a Symbol field entry, along with its associated information, a specific memory location. Thus, the programmer need not know the actual address but can refer to the symbolic address when the information is needed. Figure 3 shows both proper and improper use of symbols. The symbols shown are for illustrative purposes only.

	PROGRAMMER			
	Symbol (1 2 3 4 5 6)	Opr (8 9 10)	Operand (12 13 14 15 16 17 18 19)	X (20)	PROGRAM REMARKS (31)
1	CON#1	DEC	5		
2	ZERO	DEC	0		
3	TOTALS	BSS	30		
4	AB4	OCT	21765 31		
5	1.10	DEC	110		
6					
7	IMPROPER USE OF SYMBOLS				
8	CON+1	DEC	5		
9	110	DEC	110		
10	A-B4	OCT	21765 31		
11	A B4	DEC	4		
12					
13					

Figure 3. Symbol Fields

Entry Lines 1, 2, and 3 are proper entries because they follow Rule 1.
Entry Line 4 is proper because it follows Rule 3 and Rule 4.
Entry Lines 8 and 10 are improper because they violate Rule 2.
Entry Line 9 is improper because it violates Rule 3.
Entry Line 11 is proper because it follows Rule 4, however the assembly program would interpret A B4 as AB4. If the symbol AB4 were defined and used elsewhere in the same program, errors would occur, and the entries would be recognized as multiple symbols.

THE OPR (operation) field of the coding sheet uses a three-character mnemonic to specify the required function of the line. These mnemonics indicate to the assembly program the type of line--that is, an instruction, a control, or a constant line. The latter two types of lines involve pseudo-instructions which are discussed in detail later.

An "Instruction Line" contains a mnemonic indicating the desired computer operation. Usually this line is handled by the assembly program as one computer machine operation for each instruction mnemonic. Typical instruction lines are shown by Figure 4.

PROGRAMMER

Symbol						Opr			Operand									X
1	2	3	4	5	6	8	9	10	12	13	14	15	16	17	18	19		20
						L	D	A	A	M	T							
						A	D	D	A	M	T	2						
						S	T	A	S	U	M							
						B	Z	E										
						B	R	U	C	H	E	C	K					
						D	L	D	T	C	T	A	L					

Figure 4. Instruction Lines

A "Control Line" is interpreted and used by the program for internal assembly operations and does not become part of the assembled program. However, a control line in certain applications can cause additional words to be reserved in the assembled program. Figure 5 illustrates typical control lines.

Symbol						Opr			Operand									X
1	2	3	4	5	6	8	9	10	12	13	14	15	16	17	18	19		20
						O	R	G	1	0	0							
S	U	M				B	S	S	2	0								
C	R	D	I	N		E	Q	U	2	5	6							

Figure 5. Control Lines

A "Constant Line" indicates to the assembly program the type of constant required by the user. The assembly program then assembles the constant in the correct form. Figure 6 contains constant lines.

Symbol						Opr			Operand								X
1	2	3	4	5	6	8	9	10	12	13	14	15	16	17	18	19	20
F	I	V	E			D	E	C	5								
						D	E	C	3	1	4	1	7				
C	O	N	#	1		D	D	C	6	2	5	8	9	2			
						O	C	T	3	7	7	7	7	6	6		
						F	D	C	1	.	0	B	1				

Figure 6. Constant Lines

OPERAND field. The content of the Operand field depends upon the type of line--whether it is instruction, control, or constant. If an instruction line is involved, the operand may be:

A Mnemonic specifying an operation to be performed as shown by line 1 of Figure 7.

Symbol						Opr			Operand								X
1	2	3	4	5	6	8	9	10	12	13	14	15	16	17	18	19	20
						B	C	S	B	P	N						6
						B	R	U	*	-	1						
						L	D	A	P	R	I	N	T	+	3	9	
						C	H	S									
						S	T	A	P	R	I	N	T	+	3	9	
						B	C	N									
						B	R	U	*	-	1						
						R	C	D	2	5	6						
						H	C	R									
						L	D	A	S	Y	N	C					

Figure 7. Operand Field Instruction Line

A Decimal number which is the address portion of a computer instruction. This decimal address will be converted to binary by the General Assembly Program. For an example, see line 8 of Figure 7.

A Symbol representing some address or number within the program. This symbol in conjunction with plus (+) or minus (-) can be used for relative addressing. The combination of symbol and sum (+) or difference (-) must not exceed 8 characters. Also an asterisk (*) can be used to denote reference to the line itself. Figure 7 contains illustrative examples. The asterisk and relative addressing involving arithmetic expressions can be used to reduce the total number of program symbols used, if necessary. Additional uses of the asterisk symbol are explained under "Relative Addressing."

When a control line is specified, the Operand field of the line contains information required by the General Assembly Program. Figure 8 contains illustrative examples.

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75
	ORG	1000		
CARD	BSS	34		
STORE	BSS	40		
TOTAL	EQU	500		
	REM			CONSTANTS
CONST	DEC	28		
CON#1	DDC	628154		
	OCT	3777766		

Figure 8. Operand Field Control and Constant Lines

When the Operand field is part of a constant line, the operand must specify the constant.

THE X (index) field (column 20) specifies address modification before execution of the assembled instruction, or it may provide additional information to the assembly program, such as priority control channel or tape handler number. If the X field is part of an instruction line, it may be blank or contain either a number or an alphabetic.

Figure 9 shows examples of how the X field is used. Address modification and use of the X-registers are explained in detail in the PROGRAM REFERENCE MANUAL for the COMPATIBLES/200. An "A" is used in the X field for certain auxiliary arithmetic unit instructions.

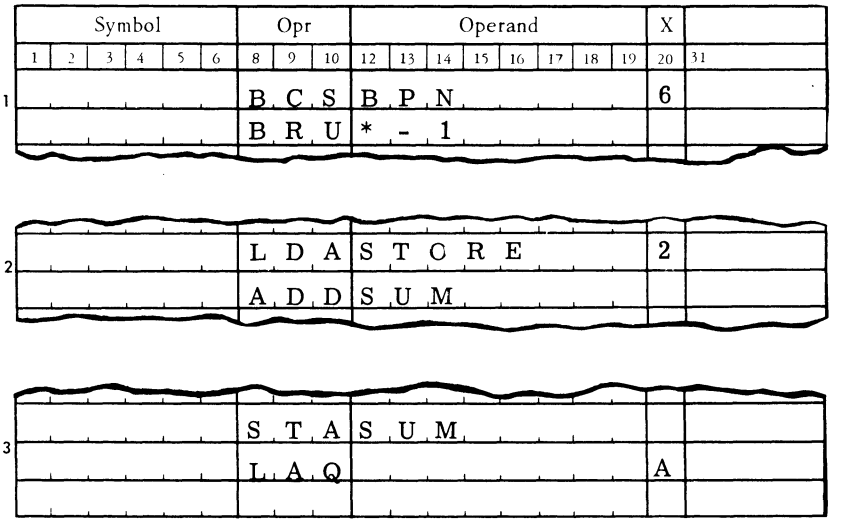


Figure 9. X (Index) Field

THE REMARKS field is a helpful programming aid in that it can be used by the coder to explain or describe the actions of each program line. The Remarks field does not require memory locations within the assembled program, nor does it affect the assembly process. It should be used extensively by programmers for adequate documentation. Refer to Figure 10 for examples.

THE SEQUENCE field specifies the order of the lines to be assembled. This is strictly a programmer's convenience and is checked only when specifically requested by use of an SEQ pseudo-instruction (on magnetic tape General Assembly Program only). Cards for the source program should be sequenced to prevent accidental mixing going unnoticed. Normally, the General Assembly Program does not check the card sequence; it does however show the sequence on the object program listing. Figure 10 shows sequenced lines.

Opr	Operand	X	REMARKS	Sequence
8 9 10	12 13 14 15 16 17 18 19	20	31 75	76 77 78 79 80
R E M			SUBROUTINE TO CHECK VOID DATE	5
D L D	V O I D		VOID DATE DAY/MONTH/YEAR	1 0
S U B	C O N # 1		VOID CONSTANT	1 5
B N Z	O K		DATA NOT VOID SO PROCESS	2 0
S P B	C L O S E	1	CLOSE FILE DATA VOID	2 5

Figure 10. Remarks and Sequence Field

Pseudo-Instructions

A symbolic program written for General Assembly Program II has both pseudo- and machine-instructions. Pseudo-instructions are symbols representing information needed by the assembly program for proper assembly. These instructions, along with the machine instructions, are included in the object program listing. Pseudo-instructions are not executed by the Processor, but are used to generate constants, control the assembly, and provide information on the program listing.

CONSTANT-LINE pseudo-instructions for the General Assembly Program and the type of information assembled for each is given in Figure 11. This is followed by explanations of the individual instructions.

CONTENTS OF OPR FIELD	TYPE OF ASSEMBLED INFORMATION	NUMBER OF CHARACTERS SPECIFIED
ALF/NAL	One BCD Word	3 Alphanumeric
MAL	One to fifteen consecutive BCD Words	3 Alphanumeric per word
PAL	One to fifteen consecutive BCD Words with sign bit on in last words	3 Alphanumeric per word
DEC	One Fixed Point Binary Number	
DDC	One Double Length (2 word) Fixed Point Binary Number	
FDC	One Floating Point (2 word) Binary Number	
OCT	One Binary Word	Up to 7 Octal Characters
Z(XX)	One Binary Word	Up to 7 Octal Characters

Figure 11. Pseudo-Instructions for Constant Lines

In the following text each instruction is introduced in the standard format, name of instruction (operation to be performed), and General Assembly Program mnemonic operation code.

ALPHANUMERIC

ALF

This instruction causes alphanumeric constants (three alphabetic or numeric characters) to be entered in the object program. The first three characters in the operand are converted to BCD and placed in a memory location determined by the assembly program. Blanks or spaces, where desired, must be indicated. Only columns 12, 13, and 14 of the Operand field can contain the data for the instruction.

Example: The words, PLANT CODE NOT IN TABLE, constitute a program typewriter message and may be entered in the object or assembled program by using the ALF instruction.

Symbol	Opr	Operand	
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17	
			Appears in Memory
T Y P E	A L F	P L A	0474321
	A L F	N T	0456360
	A L F	C O D	0234624
	A L F	E N	0256045
	A L F	O T	0466360
	A L F	I N	0314560
	A L F	T A B	0632122
	A L F	L E	0432560

Note that a space is indicated by leaving the column blank, which results in an octal 60 being placed in the memory location by the assembly program. If the desired data is not left-justified, starting with column 12, incorrect constants will result.

Example:

Symbol	Opr	Operand	
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17	
			Appears in Memory
	A L F	P L A	0604743
	A L F	N T	0606045

The constant in line 3 results in the loss of the character A, and line 4 contains two blanks (octal 60) and the character N only, losing the character T.

It should also be noted that an ALF command is required for each line containing the desired alphanumeric constant.

NEGATIVE ALPHANUMERIC

NAL

This pseudo-instruction is used to enter the 2's complement of an alphanumeric constant in the object program. The assembly program applies the same requirements to this instruction as it does to ALF.

Example: The 2's complement of the codes A14, AB2, ABF are to be placed in the object program.

Symbol	Opr	Operand	
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17	
C O D E	N A L	A 1 4	Appears in Memory
	N A L	A B 2	3567674
	N A L	A B F	3565576
			3565552

MULTIPLE ALPHANUMERIC

MAL

This pseudo-instruction will enter alphanumeric data into as many as fifteen consecutive memory locations. The number of words to be filled must be specified by a numeric in columns 12 and 13 and the data must be placed in the Remarks field.

Example:

Symbol	Opr	Operand	X	
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31
T Y P E	M A L	0 8		PLANT CODE NOT IN TABLE

This data is placed in memory as follows:

Data	Memory
PLA	0474321
NT	0456360
COD	0234624
E N	0256045
OT	0466360
IN	0314560
TAB	0632122
LE	0432560

(see also note following example of PAL pseudo-instruction.)

MULTIPLE ALPHANUMERIC FOR PRINTER WITH PRINT LINE INDICATOR

PAL

This pseudo-instruction is similar to the MAL instruction with the exception of entering a minus sign in the last word of the alphanumeric data. (The minus sign signifies an end-of-line during the printer operation.)

Example:

Symbol						Opr			Operand								X	REMARKS
1	2	3	4	5	6	8	9	10	12	13	14	15	16	17	18	19	20	31
						P	A	L	4									PLANT CODE

The data enters memory as shown below. Note that memory word 4 of the data contains a 1 in the sign bit, or 0 position of the word.

	Data	Memory
1.	PLA	0474321
2.	NT	0456360
3.	COD	0234624
4.	E	2256060

Note: Pseudo-instructions MAL and PAL cannot be used when a symbolic General Assembly Program is entered from magnetic tape during the assembly operation.

DECIMAL
DEC

This instruction places the binary equivalent of a decimal constant in the object program. The constant is assigned a memory location as determined by the assembly program. The operand portion of the constant can be symbolic or decimal. If symbolic, at least one character must be used other than 0-9, plus (+), minus (-), decimal point (.), B, or E. If no sign is present, the number is assumed to be plus (+). A minus sign, specifying a negative number, results in the 2's complement of the number being placed in memory.

Examples of Positive numbers:

Opr			Operand						
8	9	10	12	13	14	15	16	17	
D	E	C	5						
D	E	C	1	2	8				
D	E	C	7	3	7	3	8		
D	E	C	9	2	8				
D	E	C	1	2					
D	E	C	+	1	7	5	0		

Appears in
Memory
0000005
0000200
0220012
0001640
0000014
0003326

Examples of Negative numbers:

Opr	Operand	Appears in Memory
8 9 10	12 13 14 15 16 17	
D E C	- 5	3777773
D E C	- 1 2 8	3777600
D E C	- 7 3 7 3 8	3557766
D E C	- 6 3 9 7 8	3603026
D E C	- 1	3777777

The character B can be used to specify a binary scale for either positive or negative numbers. The number following B is used to position the binary point for the decimal constant preceding the B in the Operand field. If no scale is specified, the assembly program assumes a binary scale of 19.

Examples using the B character:

Opr	Operand	Appears in Memory
8 9 10	12 13 14 15 16 17 18 19	
D E C	5 B 1 6	0000050
D E C	- 5 B 1 6	3777730
D E C	7 3 7 3 8 B 1 8	0440024
D E C	- 4 B 3	3000000
D E C	+ 4 B 3	1000000

The characters decimal point (.) and E can be used to specify decimal scales or decimal exponents. Normally, the . indicates a mixed number (lines 3-6 in the following example), while E specifies the power of 10 by which the constant is multiplied. For example, E-2 indicates 10-2 and E2 indicates 10^2. If the character B is not used with decimal point (.) or E, only the integral portion of the number will be converted by the program. If the number of characters used to specify a decimal constant exceeds eight, the OPR (operation) field of the next line is left blank and the constant is continued in the operand field, using two lines for one entry (see lines 7 and 8, 9 and 10, 11 and 12). Only one binary scale and one decimal scale can be indicated for a single decimal constant.

Examples using the characters . and E:

Opr	Operand	Appears in Memory
DEC	5.52	0000005
DEC	5.52B18	0000013
DEC	.52B18	0000001
DEC	5.5B18E2	0002114
DEC	.7373 8E2	0000223
	B18	
DEC	-.5625B1	
	0	3777340
DEC	-.1875B1	3777640
	0	
DEC	.3217B0	0511327

In using the DEC instruction it is possible to express the decimal constant in the Operand field in symbolic notation. However, unless there is some specific advantage, doing this results in having to add an EQU instruction immediately following the DEC that is otherwise not required.

DOUBLE LENGTH DECIMAL

DDC

DDC, like DEC, is used to enter a decimal constant in the object program. This constant is assigned two sequential memory locations starting with the first even-numbered location available. If no binary scale is specified, the assembly program assumes a binary scale of 38.

Example of DDC:

Opr	Operand	Appears in Memory
8 9 10	12 13 14 15 16 17 18 19	31
DDC	12	0000000
		0000014
DDC	7864321	0000001
		1000000
DDC	-24	3777777
		3777750
DDC	.0796896	0024315
	7928B2	0127545
DDC	24B36	0000000
		0000140
DDC	1.5707961	0622077
	3184 7B2	0651010
DDC	1018B0E-	0000000
	10	0066516
DDC	15255 27B0	0000007
	E-10	1774566

FLOATING POINT DECIMAL

FDC

This instruction is used to enter a floating point decimal constant in the object program. The use of FDC is the same as the DDC--that is, two sequential memory locations, starting with an even-numbered location, are assigned by the assembly program. After conversion, the constant is in normalized form, if the specified binary scale is minimum; otherwise the constant is unnormalized. If no binary scale is specified, the assembly program determines the binary scale and a normalized floating point number results.

A detailed description of floating point decimal formats and operations appears in the General Electric reference manual covering the auxiliary arithmetic unit subsystem.

Examples of FDC:

Opr	Operand	
8 9 10	12 13 14 15 16 17 18 19	31
		Appears in
		Memory
		0006000
FDC	1B1	0000000
		0006705
FDC	1.442695	0507312
	041B1	
		0013244
FDC	3.321928	1517022
	094B2	0022400
FDC	10B4	0000000
		Both give
FDC	1E40	same results:
	and	1027530
FDC	1E40B133	1451743
		Result unnor-
		malized
		1031654
FDC	1E40B134	0624761

OCTAL

OCT

The entering of octal constants in the object program is accomplished with the OCT pseudo-instruction. The octal number specified in the Operand is right-justified and assigned one memory location designated by the assembly program. Leading zeros in the Operand field are ignored. A leading minus (-) in the operand sets the sign bit of the constant to 1. Octal constants are used primarily for establishing particular bit configurations in memory.

Example of OCT:

Opr	Operand	
8 9 10	12 13 14 15 16 17 18 19	31
		Appears in
		Memory
OCT	2777766	2777766
OCT	377	0000377
OCT	-377	2000377
OCT	27600	0027600
OCT	3777746	3777746
OCT	-1777746	3777746

OCTAL OPERATION CODE

Z (XX)

The Z pseudo-instruction is used to set the operation bits of the assembled instruction to any desired configuration. The Operand can be decimal or symbolic, and indexing is optional. In use, a Z is placed in column 8 with the two octal digits desired as an operation code in columns 9 and 10.

Sample Coding:

Opr	Operand	X	REMARKS
8 9 10	12 13 14 15 16 17 18 19	20	31 75
Z00	TEMP	2	EQUIVALENT TO LDA TEMP 2
Z04	-100	2	EQUIVALENT TO BXL 100 2
Z20	0		EQUIVALENT TO WORD 1 of RWD

Listing:

Memory Location (Octal)	Memory Contents (Octal)	Assembly Program		
01764	0000000	TEMP	DDC	0
01765	0000000			
⟆ ⟆	⟆ ⟆	⟆ ⟆		
0770	0041764		Z00	TEMP 2
0771	0457634		Z04	-100
0772	2000000		Z20	0

CONTROL-LINE PSEUDO-INSTRUCTIONS. Control line instructions provide information which controls the internal operations of the General Assembly Program, although this information does not become part of the assembled program. In certain cases, additional words may be reserved in, or added to the assembled program. Figure 12 is a list of control instructions. Following it are explanations of the individual instructions.

	Contents of Opr Field	Operand Field	Type of Assembled Information
	SBR	Symbol of subroutine	Calls subroutine
	ORG	Decimal or symbolic	Assigns memory address
	LOC	Octal	Assigns memory address
Optional	EQU	Decimal or symbolic	Equates memory address
Optional	EQO	Octal	Equates memory address
Optional	BSS	Decimal or symbolic	Reserves memory blocks
	TCD	Decimal or symbolic	Transfers control
	END	Decimal or symbolic	Terminates assembly

Figure 12. Control Lines for Pseudo-Instructions

SUBROUTINE CALL
SBR

This pseudo-instruction can be used only if the General Assembly Program is called from the master assembly tape. An SBR instruction is used to instruct pass 0 to obtain the specified subroutine from the master program tape. An error indication results if the specified subroutine is not present. The library subroutines furnished for the GE-225 are in symbolic form and not in binary format.

The subroutine calls are saved until the user's END card is encountered. The specified subroutines are then placed behind the user's coding.

The SBR pseudo-instruction cannot be used if a symbolic General Assembly Program is entered from magnetic tape during the assembly operation.

Note: The subroutines are assigned memory locations following the last instruction of the user's coding. If the user desires, the subroutines can be placed in reserved memory locations, as shown in the following examples.

Example:

Opr	Operand	X	REMARKS
8 9 10	12 13 14 15 16 17 18 19	20	31 75
S B R	S T R I P		BINARY - BCD CONVERSION ROUTINE
E N D	S T A R T		

Listing:

```
                02623          SBR STRIP
                                 REM
02623  2514003  STRIP    BOV
02624  0000000  =1102    LDA    0
02625  0000000           LDA    1
```

This subroutine can also be called in the following manner.

Example:

Symbol	Opr	Operand	X
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20
	L D A	T E M P	
	A D D	T O T A L	
	S P B	N P R I B D	1
N P R I B D	S B R		
Z E R O	D D C	O	
	E N D	S T A R T	

In both instances the STRIP and NPRIBD subroutines could be assigned memory locations following the last instruction (ZERO DDCO) of the coding.

If the programmer desires to place the subroutine in a specific location other than at the end of his program, he must reserve a sufficient number of words in memory by using a BSS instruction and by placing an ORG card with the origin of the reserved area immediately preceding the END card.

Example:

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75
SUBR#1	BSS	76		RESERVE 76 LOCATIONS FOR ROUTINE
	LDA	TEMP		
	SPB	NPRIBD	1	
	SBR	NPRIBA		
	BRU	*		
	ORG	SUBR#1		
	END	START		
	END			
	SBR	NPRIBD		

The NPRIBD subroutine is listed at the end of the object program, but the ORG SUBR#1 instruction causes General Assembly Program to assign memory locations starting at SUBR#1 rather than locations following the BRU*.

Pass 0 of the General Assembly Program will report as "possible undefined" symbols any reference to subroutine symbols made by the main program, because the subroutines are not obtained until after the undefined check.

ORIGIN

ORG

This pseudo-instruction controls the memory assignments performed by the general assembly program. When an ORG instruction is encountered, the assembly program uses the contents of the Operand field to reset an internal counter in the assembly program referred to as the memory allocation register (MAR). Normally, the MAR is increased by 1 for each instruction encountered.

If the Operand is a decimal, it is converted to binary by the program before being used. If the Operand is symbolic, the symbol(s) must be predefined before being used. A symbol is defined by placing its name in the Symbol field (columns 1-6) once, and only once, in a given program. The General Assembly Program ignores all but the Operand field on an ORG instruction. When no ORG is used, the program assigns an origin of memory location 00000.

Examples:

Opr	Operand	X	REMARKS
8 9 10	12 13 14 15 16 17 18 19	20	31 75
O R G	1 2 8		THE MAR IS SET TO 200(128_{10}) AND NEXT
			INSTRUCTION STARTS AT AN OCTAL 200

Opr	Operand	X	REMARKS
8 9 10	12 13 14 15 16 17 18 19	20	31 75
O R G	B E G I N		THE SYMBOL "BEGIN" MUST BE PREDEFINED
			AND THE MAR IS SET TO THE ASSIGNED VALUE
			IF BEGIN IS NOT PREDEFINED THE MAR IS SET
			TO ZERO

Example: Use an ORG to assemble an object program at memory location 1000 (decimal).

Symbol	Opr	Operand	X
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20
	O R G	1 0 0 0	
T W O	D E C	2	
T E N	D E C	1 0	

LOCATION IN OCTAL

LOC

This operation performs the same functions as an ORG; however, the contents of the Operand field must be an octal number. The assembly program will ignore leading zeros.

Example:

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75
	LOC	1750		MAR IS SET TO OCTAL 1750
TWO	DEC	2		
TEN	DEC	10		

EQUALS

EQU

This instruction equates a new symbol to some memory location already known to the assembly program. The Operand (decimal or symbolic) indicates the specific memory location to be used. This instruction does not affect the memory allocation register; thus it may be used as often as necessary and at any point within the source or symbolic program without disturbing the memory assignment sequence. If the operand is symbolic, the symbol must be predefined. A decimal operand is converted to binary before being utilized.

Example:

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75
CRD	EQU	256		
AREA	EQU	CRD		CRD MUST HAVE BEEN PREDEFINED
AREA2	EQU	CRD+40		

EQUALS OCTAL

EQO

The EQO instruction is the same as the EQU except that the Operand field must be in octal form. Leading zeros in the Operand field are ignored.

Example:

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75
CRD	EQO	400		
AREA	EQU	CRD		CRD MUST HAVE BEEN PREDEFINED

BLOCK STARTED BY SYMBOL

BSS

A BSS causes the assembly program to increase the memory allocation register (MAR) by the number in the Operand field. This instruction is used to reserve a block of memory locations in the object program. The Operand field may be decimal or symbolic. If symbolic, the symbol used must be predefined; if decimal, the operand is converted to binary by the assembly program before use. The BSS can be used as often as needed.

Example:

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75
	ORG	0256		
CRD	BSS	28		MAR IS INCREASED BY 28
PRINT	BSS	40		MAR IS INCREASED BY 40
INDEX	BSS	3		MAR IS INCREASED BY 3
STORE	BSS	SAVE		SAVE MUST BE PREDEFINED

The BSS instruction of line 2 of the example will reserve 28 consecutive memory locations starting at location 256. The other BSS commands reserve additional blocks of memory.

A negative decimal operand can be used to reduce the MAR, in effect equating a block of memory to another block already defined.

PUNCH TRANSER CARD

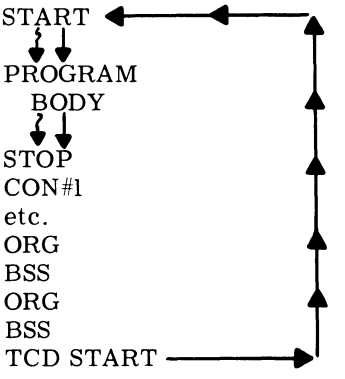
TCD

A TCD generates an instruction that will cause the loader to transfer control to the location specified by the Operand field at execution. In the relocatable format this is a type 5 card. The operand may be decimal or symbolic. A TCD (transfer control data card) may be used as often as necessary in a source program, because this instruction does not affect the memory allocation register. A symbol in the operand must be predefined.

Example:

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75
	T C D	2 9 0 0		
	T C D	S T A R T # 1		START #1 MUST BE PREDEFINED

The transfer card is the last card of the object program; when the program is loaded for execution, the transfer card directs the central processor to the location of the initial program instruction:



The TCD should not be used in place of an END instruction at the end of a source program. Information on END follows.

END OF PROGRAM

END

This pseudo-instruction causes the assembly program to generate a transfer card to transfer control to the initial program location (specified in the Operand field) when the object program is loaded into memory for execution. In the relocatable format, this is a type 3 card. The Operand field may be decimal or symbolic; if symbolic, the symbol must be predefined. The END instruction indicates the end-of-program and terminates assembly. It must be used only once and must be the last instruction of the source program.

The TCD instruction previously described should be used where a transfer control card is to be generated before the end of the assembly program.

Failure to use the END instruction results in a typewriter message so indicating. Assembly continues, but no transfer card is generated. Thus, this instruction should appear in the program.

Example:

Opr			Operand							X	REMARKS
8	9	10	12	13	14	15	16	17	18	19	20	31	75
E	N	D	1	0	0	0							
E	N	D	S	T	A	R	T					START MUST BE PREDEFINED	
E	N	D	C	O	N	S	T	+	3			CONST MUST BE PREDEFINED	

The END instruction of line 1 would result in the punching of a transfer card as shown in Figure 13.

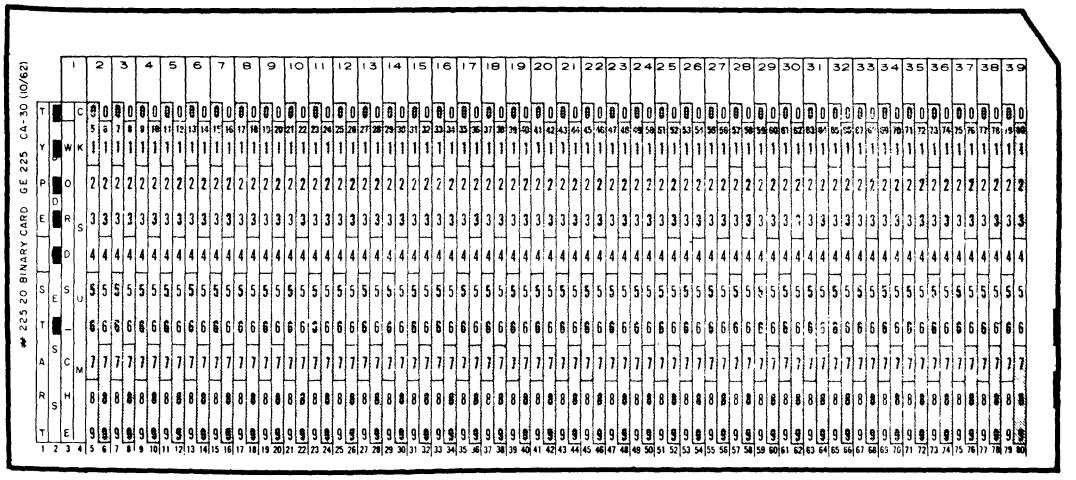


Figure 13. Transfer Card END Instruction

REMARKS

REM

The REM mnemonic in the Opr field is used to provide additional information on the object program listing as given in the Remarks and Sequence fields (columns 31 through 80). All other fields in the instruction line are ignored.

Example of REM:

Opr	Operand	X	REMARKS	Sequence
8 9 10	12 13 14 15 16 17 18 19	20	31 75	76 77 78 79 80
R E M			SUBROUTINE TO CHECK VOID DATE	V 0 0 5

EJECT PRINTER PAPER

EJT

Normally the assembly program prints 54 lines per page and then ejects to the top of the next page. When the EJT command is encountered, the line count is reset and the paper is immediately ejected to the top of the next page.

Example:

Opr	Operand	X	REMARKS
8 9 10	12 13 14 15 16 17 18 19	20	31 75
D E C	6 2 1 5		
R E M			END ADJUSTMENT CONSTANTS
E J T			CONTINUE LISTING ON NEXT PAGE

CHECK SOURCE PROGRAM CARD SEQUENCE NUMBERS

SEQ

Except for listings, sequence numbers are normally ignored. However, if the SEQ command is used, the assembly program checks to see that the card sequence numbers are in ascending order (tape General Assembly Program only). Blanks are ignored. Numbers less than or equal to the preceding number are errors and are flagged with an S in the right margin of the listing. However, the assembly process continues.

Example:

Opr	Operand	X	REMARKS
8 9 10	12 13 14 15 16 17 18 19	20	31 75
S E Q			CHECK SEQUENCE NOS. *05
L D X	Z E R O	2	10
D L D	T E M P	2	15

PRINT NAME OR TITLE ON EACH PAGE

NAM

A page number is printed at the top of each page of a listing. When NAM is used with a name or title, which would appear in the Remarks field of the coding sheet (columns 31 to 75), the name is printed at the top of each page. The name may be changed by issuing a new NAM instruction. The name can contain a maximum of 45 characters.

Example:

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75
	N A M	E D I T # 3		ROUTINE TITLE
	O R G	1 0 0 0		
Z E R O	D E C	0		

NO LIST
NLS

General Assembly Program normally lists the object program on the high-speed printer. The NLS pseudo-instruction may be used at any point in the source program to inhibit the printer listing.

LIST
LST

Where the assembled program listing has been stopped by an NLS, the listing can be resumed with an LST instruction.

Example:

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75
	NLS			ELIMINATE LISTING OF
PUN	DDC	0		NPRIBD SUBROUTINE
PUN1	BSS	4		
PUN2	BSS	2		
NPRIBD	EQU	PUN+12		
	LST			RESUME LISTING
TEMP	BSS	20		

RELATIVE ADDRESSING

The General Assembly Program II provides the facility for assigning addresses relative to some starting point or some symbolic memory location. This is termed "relative addressing" and is a useful programming aid. Relative addressing can be accomplished in several ways as the following examples show.

Relative Addressing Using + (plus) or - (minus) Symbols

Example Coding:

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75
A M T	E Q U	2 0 0		THIS INSTRUCTION EQUATES THE SYMBOL AMT
				TO MEMORY LOCATION 200 (DECIMAL)
	L D A	A M T + 6		THIS INSTRUCTION ILLUSTRATES RELATIVE
				ADDRESSING . + 6 REFERS TO MEMORY
				LOCATION 206
	L D A	A M T - 2		AMT - 2 IS MEMORY LOCATION 198

Since the General Assembly Program has relative addressing capabilities, it will assign the correct memory addressed to AMT+6 and AMT-2.

Relative Addressing Using ORG Instruction

Example Coding:

Symbol	Opr	Operand	X	REMARKS
1 2 3 4 5 6	8 9 10	12 13 14 15 16 17 18 19	20	31 75
C O N	E Q U	5 0 0		
	O R G	C O N		THIS ORG WILL START AT MEMORY LOCATION
				500 AS ESTABLISHED BY USE OF EQU FOR THE
				SYMBOL
	O R G	C O N + 5 0		STARTING MEMORY LOCATION HERE IS 550
				DUE TO RELATIVE ADDRESSING

The starting locations of subdivisions within a program can be determined by a relative address as defined with an ORG instruction.

Relative Addressing Using * (asterisk) Symbols

Example Coding:

Opr	Operand	X	REMARKS
B C N			IF THESE INSTRUCTIONS START AT MEMORY
B R U	* - 1		LOCATION 1000, THE ASTERISK IN THE OPERAND
R C D	C R D		FIELD OF THE SECOND INSTRUCTION IS INTER-
H C R			PRETED BY GAP AS BEING 1001. THE ADDRESS
L D A	* + 6		IN THE OPERAND FIELD IS 1001-1 or 1000.
B M I			
B R U	*		THE INSTRUCTION ON LINE 7 CAUSES THE COM-
S U B	C O N 4		PUTER TO ENTER A CONTINUOUS LOOP SINCE
B R U	* + 8		THE ASTERISK IS INTERPRETED BY GAP AS THE
			ADDRESS OF THE INSTRUCTION ITSELF. THE
			MACHINE THEN EXECUTES THE SAME INSTRUC-
			TION CONTINUOUSLY.

A convenient method of relative addressing that reduces the number of symbols required in the use of the asterisk (*) character. An asterisk in the Operand field of an instruction is interpreted by the assembly program as the address of the instruction itself. As shown in line 7 of the example, the use of the asterisk is equivalent to writing the same symbol in both the Symbol and Operand fields of the same line (that is, STOP, BRU, STOP).

Multiple Relative Addressing

Multiple relative addressing is permitted by which combinations of symbols, numbers, and asterisks can be added or subtracted in any given Operand field. The order of symbols, numbers, and asterisks in the operand is not restricted.

Example Coding:

Symbol	Opr	Operand	X
	B X L	S U M - B + 8	2
	B S S	* + B - C + 4	
	B R U	9 + 4 + 3 + G	
	L D A	8 + * + *	
	S T A	1 2 8 + 5 + 2 4	

Line 1 illustrates a convenient way to define the size of a memory area to be manipulated under the control of a BXL instruction. The area is 8 locations larger than the memory area between the SUM and B.

Line 5 leaves to the assembly program simple arithmetic that often causes error and documents the programmer's intention to store a quantity in the 14th location of the second field of a card image starting a location 128.

The other lines singly illustrate valid combinations about which the programmer should know, no matter how infrequently they are used.

GENERAL ASSEMBLY PROGRAM-DETECTED CODING ERRORS

As an aid to the programmer, the General Assembly Program detects certain types of coding errors and lists them on either the typewriter or high-speed printer.

Pass 0 Detected Coding Errors

Pass 0 provides a listing of possible undefined symbols; multiple symbols; symbols with no reference; and, if the tape assembly program is used, an indication of any tape errors. Figure 14 is a sample printout from pass 0.

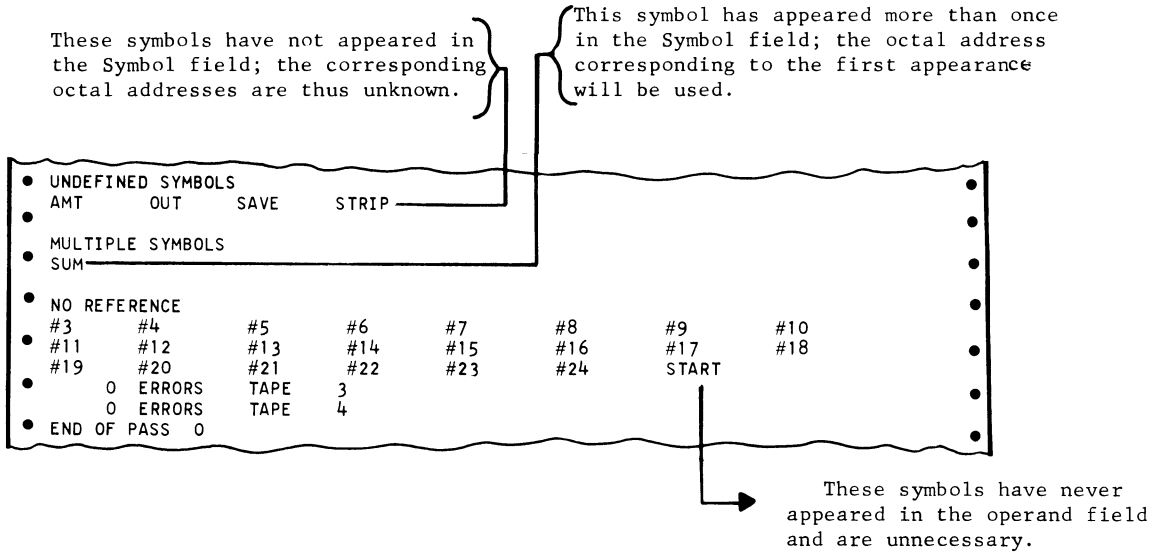
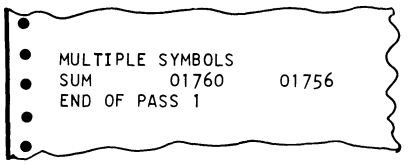


Figure 14. Printer Listing from Pass 0 Coding Errors

Pass 1 Detected Coding Errors

Pass 1 provides a multiple symbol list that gives the memory locations referring to the symbol.

This printout can be done on the high-speed printer or, if no printer is available, the console typewriter. A printout from the printer appears below.

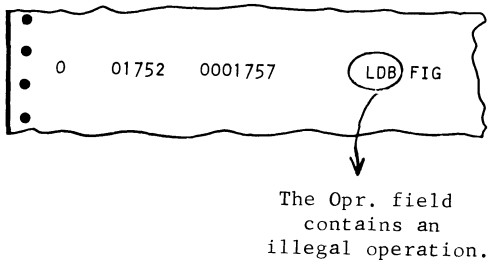


Pass 2 Detected Coding Errors

Pass 2 lists the assembled program along with codes which indicate an error or a suspected error in the program coding. Six symbols (O, U, M, A, T, and S) are used as error codes. These error codes are printed to the left of each line on which they occur. A brief description of these codes follows:

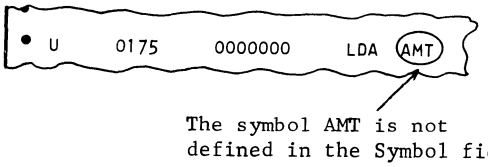
Symbol	Meaning
O	Illegal Mnemonic Operation. This mnemonic is unknown to the General Assembly Program. The program generates a 00 octal operation code as a substitute.

Example:



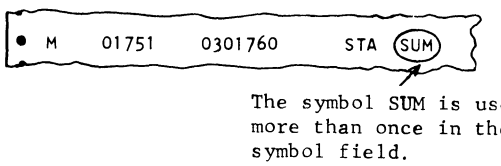
Symbol	Meaning
U	Undefined Symbol. A symbolic name appearing in the Operand field does not appear in the Symbol field of any instruction or constant line. The address assigned to this symbol is 0000.

Example:



Symbol	Meaning
M	Multiple Defined Symbol. A symbolic name in this line appears in the Symbol field more than once in the program. The symbol in question is given an octal address corresponding to the address of the instruction in which it first appeared in the Symbol field.

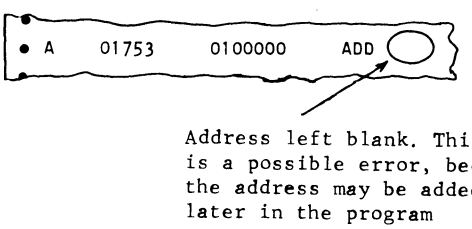
Example:



Symbol	Meaning
A	Error or Suspected Error in the Operand Address.

1. Blank Operand field in a line normally requiring an address.
2. An entry in the Operand field of a line which normally should be blank.
3. Numeric value of the operand not meeting the requirements of the line in which it was used.

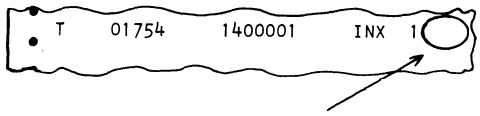
Example:



Symbol	Meaning
T	Error or Suspected Error in X-Field.

1. The X-field is blank in a line normally requiring an entry.
2. The X-field contains an entry in an instruction line which ordinarily does not require address modification.
3. The numeric value of the entry in the X-field violates the requirements of the line in which it appears.

Example:

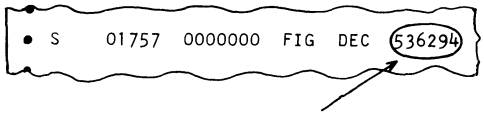


The X-field is blank on an instruction which requires an entry.

Symbol	Meaning
S	Erroneous Scale Factors in DEC, DD, FDC.

1. The specified binary and decimal scales are incompatible.
2. Two decimal or binary scales have been specified in the constant line.
3. The specified or implied binary scale causes the constant to exceed 1 binary word (overflow).

Example:



The constant in the Operand field is larger than the scale permits (assumed by the assembly program to be binary 19) This constant must be double length.

II ASSEMBLY OPERATIONS

The General Assembly Program II consists of four separate programs: Pass 0, Pass 1, Pass 2A (absolute), and Pass 2R (relocatable for 8k or larger memory systems only). Operating instructions depend upon whether a card, paper tape, or magnetic tape subsystem is used for assembly and upon the configuration of the system.

A flow diagram of the 3 passes is shown in Figure 15. Pass 2 can be either 2A (absolute) or 2R (relocatable) since both are not used in the same assembly. The specifications of the input/output media can be changed during assembly, so long as the output from one pass is acceptable as input to the subsequent passes. Thus, the operating instructions and console switch settings vary with different systems' hardware configurations.

SYSTEM CONFIGURATIONS

The hardware requirements for the operation of General Assembly Program II (225F1.006/7) with all possible media are:

Card Reader subsystem
Card Punch subsystem
Magnetic Tape Controller and a minimum of 4 Magnetic Tape Handlers
Printer subsystem (900 or 450 lpm)*
Typewriter
4k Memory or 8k (minimum for relocatable).

*Refer to 225E6.001/2/3 for 300 lpm simulators.

The minimum hardware requirements for the operation of the General Assembly Program II (CD225F1.009/010) using punched cards as input/output are:

Card Reader subsystem
Card Punch subsystem
Typewriter
4k Memory

The minimum hardware requirements using paper tape as input/output are:

Card Reader subsystem
Perforated Tape Reader and Punch subsystems
Typewriter
4k Memory

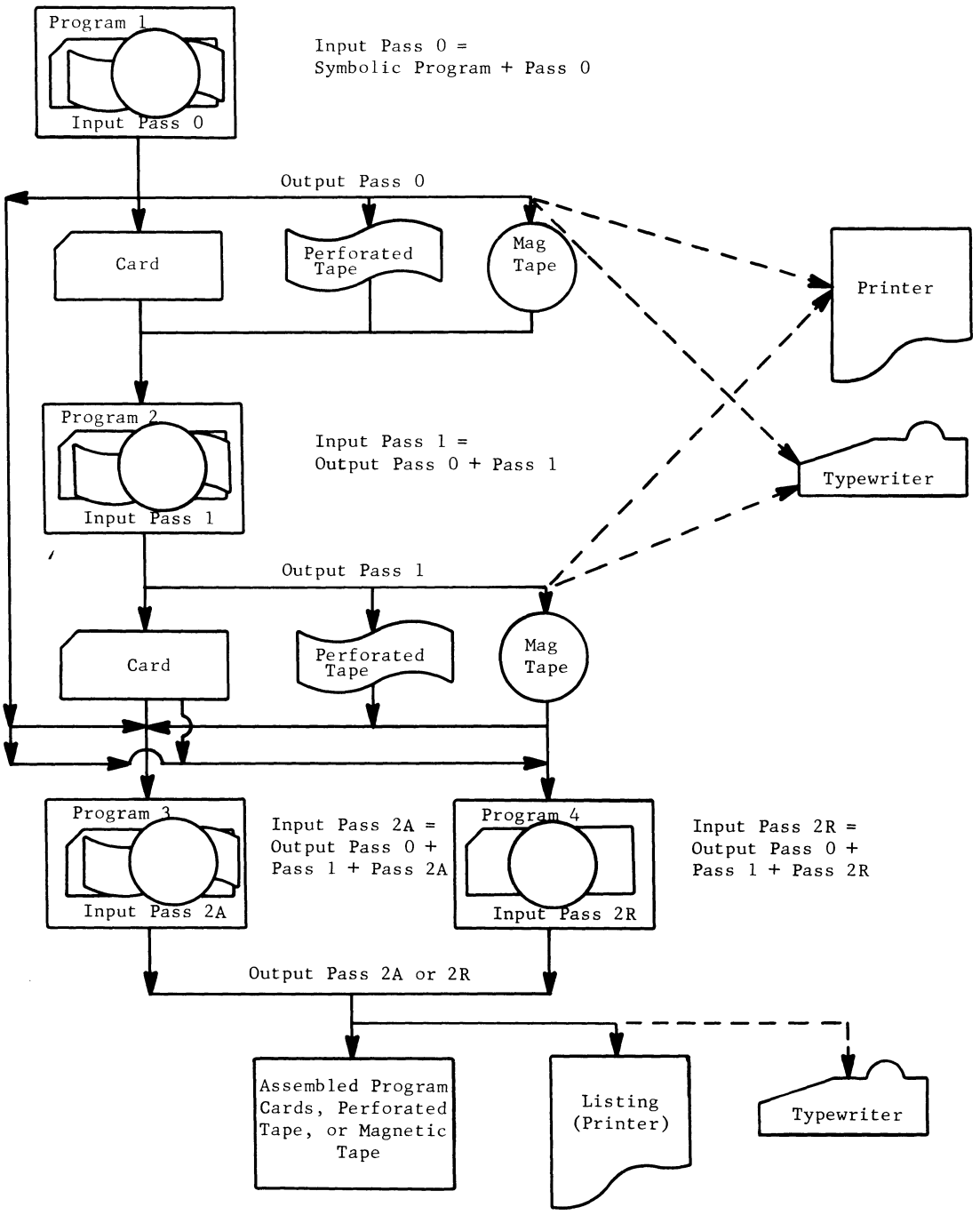


Figure 15. Diagram of General Assembly Program II Programs

Regardless of the hardware configuration used during assembly of the source program, General Assembly Program II will assemble object programs for any hardware configuration.

A-REGISTER INPUT SWITCHES

The A-register input switches are used to indicate the peripheral configuration available to the General Assembly Program II program as well as other modifications that may be employed while assembling. The setting of the console switches may be altered between passes, but care must be exercised to maintain peripheral compatibility between passes.

Example: It is possible to specify magnetic tape, card, or paper tape as input to and only magnetic tape as output from pass 0. At the end of pass 0, the switches must be altered to specify magnetic tape input/output to pass 1. At the end of pass 1, the switches may be altered to specify magnetic tape input and paper tape as output to pass 2. Figure 15 diagrams the input/output relationships. The individual switch settings for various conditions are given below.

Switch Settings

Switch 1

Normal: Absolute output

Down: Relocatable output.

Switch 2

Normal: Printer is on line.

Down: No on-line printer. An octal deck is punched instead of binary cards.

Switch 3

Normal: Tape 3 is used to obtain comments on the pass 2 program listing.

Down: Tape 3 is not available, and no comments will appear on the listing. (If switch 4 is down, switch 3 is ignored.)

Switch 4

Normal: Tape 4 is used as output by pass 0 and input by pass 1 and pass 2. Tape 5 is used as output by pass 1 and input/output by pass 2.

Down: No magnetic tapes available, all input/output via cards or perforated tape (switch 4 overrides switches 3 and 6).

Switch 5

Not used.

Switch 6

This switch is used only by pass 2.

Normal: No tape 6 available. Punched output on cards or paper tape.

Down: Binary program output from pass 2 is written on tape 6 (if switch 4 is down, switch 6 is ignored).

Switch 7

Not used.

Switch 8

Not used.

Switch 9

Normal: Card punch on line.

Down: No on-line card punch (if switch 4 is down and/or switch 6 is normal, switch 9 is ignored).

Switch 10

Not used.

Switch 11

Normal: Card or magnetic tape input.

Down: Perforated tape input.

Switch 12

Normal: Card or magnetic tape output. Switch 13 is ignored.

Down: Perforated tape output. Switch 13 is interrogated.

Switch 13

Switch 13 affects only pass 2.

Normal: If switch 2 is normal, printer listing and perforated tape program. If switch 2 is down, perforated tape listing.

Down: Typewriter listing and perforated tape program (if switch 2 is normal, switch 13 is ignored).

Switch 14

Normal: No packed symbolic listing.

Down: Print packed symbolic listing (if switch 2 is down, switch 14 is ignored).

Switch 15

This switch is used only by pass 0.

Normal: Read input from card or perforated tape and process concurrently.

Down: Read input from card or perforated tape and write tape 3. Alter pass 0 to read input from tape 3 and process.

Switch 16

This switch is used only by pass 0.

Normal: Input to pass 0 is on cards or perforated tape. Switch 15 is interrogated.

Down: Input to pass 0 is on magnetic tape 3. (If switch 3 or 4 is down, switch 16 is ignored; if switch 16 is down, switch 15 is ignored.)

Switch 17

Not used.

Switch 18

Normal: "No reference" symbols are typed or printed after pass 0.

Down: Suppresses or terminates the typing or printing of the "no reference" symbols.

Switch 19

Except for machine malfunctions, the central processor will halt during assembly under three circumstances only:

1. The number of special symbolic operands (symbol table 1) exceeds 250, the maximum size allowed by the symbol table.
2. The total number of symbols (symbol table 2) exceeds the size of the table.
3. During the final phase of assembly, a name appearing in the Symbol field cannot be found in the symbol table.

When these errors occur, an indicative typeout results and the central processor goes into a programmed loop. Toggling switch 19 bypasses the "symbol table overflow" halt during passes 0 and 1 and the "symbol lost" halt during pass 2.

CARD-TO-CARD OPERATIONS WITH MINIMUM CARD EQUIPMENT

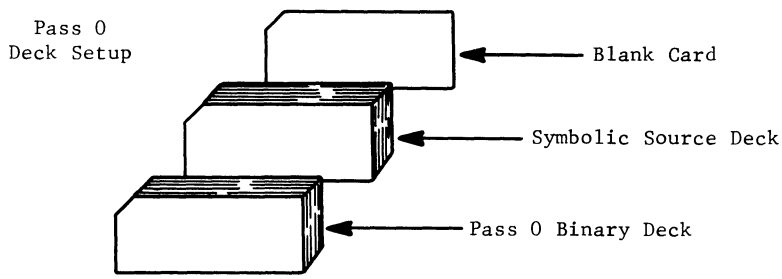
The following instructions are for loading General Assembly Program II (CD225F1.006) from cards for card-to-card operations with this equipment configuration:

Card Reader
Card Punch
Typewriter
Printer (optional)

Pass 0 Card-to-Card Operations

When operating a minimum card system, pass 0 must be processed first, as follows:

1. Place the symbolic program followed by one blank card behind the binary pass 0 deck.



2. Load card deck into card reader.

3. Set A-register input switches as desired for end result.

Switch	Setting	Result
4	Down	no magnetic tapes
2	Normal	on-line printer
2	Down	no printer on line

All other switches normal.

4. Start pass 0.

Pass 0 prepares the source program for subsequent use in other assembly passes by packing the symbolic input program deck, four instructions to one card, as shown by Figure 16. This packed deck is produced in the form of punched cards. The assembly program creates pass 0 sequence numbers for its own use (20000 series equals packed symbolic numbers), as shown in the upper right corner of the three cards in Figure 16.

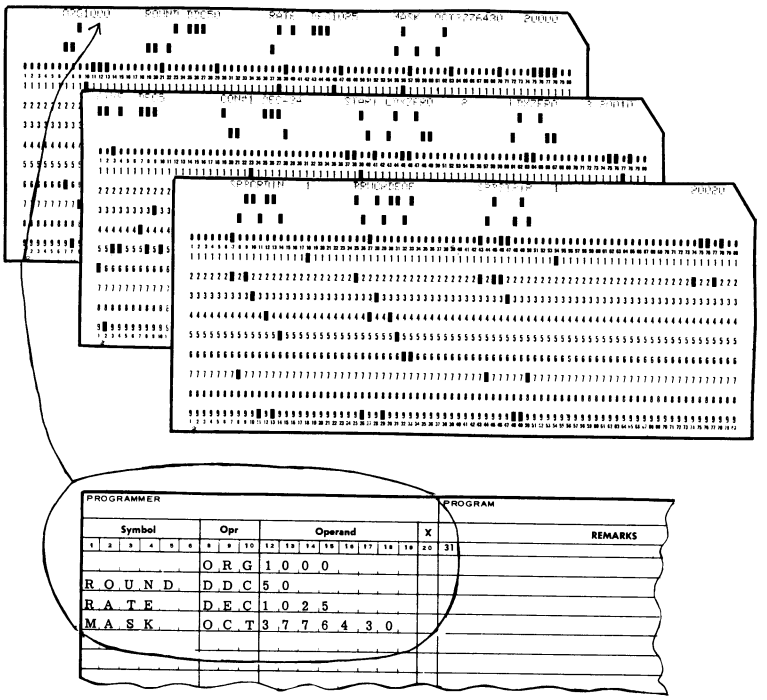
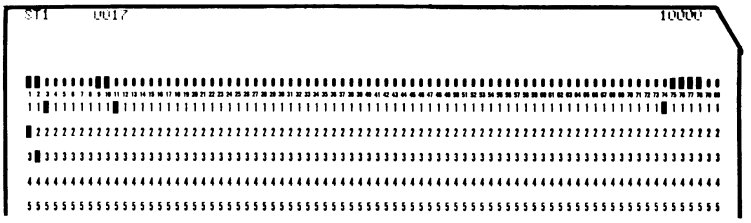


Figure 16. Pass O -- Symbolic Program Deck

A TABLE OF SPECIAL SYMBOLS is formed containing all symbolic operands appearing in the card input/output, double length, floating point, or document handler instructions. This symbol table, referred to as symbol table 1 (ST1), can be recorded on magnetic tape, perforated tape, or punched cards. Figure 17 shows a symbol table 1 card and a printer listing of the table.

Columns 1-3 on the symbol card contain the identifying letters ST1. In columns 9-12, octal 0017 (15_{10}) is a control number used by the assembly program. It equals 3 times the number of symbols in ST1 (5). Columns 74-78 contain the assembly-program-created sequence number of the card in the pass 0 packed deck.



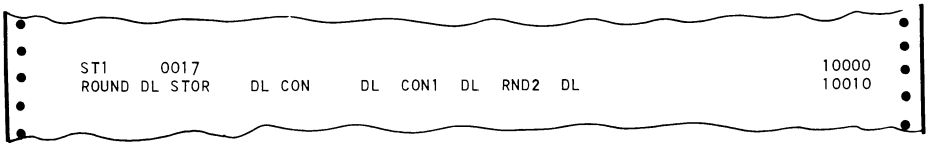


Figure 17. Special Symbol Table (ST1)

For any succeeding pass, the cards are arranged according to the sequence numbers--that is, ST1 (10000 series) cards before packed symbolic program cards (20000 series) and ST2 (00000 series) cards before both of these. In the printer listing shown in Figure 17, the letters DL (double length) between ROUND and STOR is a special category symbol.

Pass 0 checks all symbolic names in the input deck for no reference, possible undefined, or multiple-defined usage. If any one of these conditions exists, pass 0 provides a listing on the printer; or if desired, the listing can be printed on the console typewriter. Figure 18 is a sample printout by pass 0 on the printer showing these symbolic names.

```
UNDEFINED SYMBOLS
AMT#1    AMT#2     CRDIN     CRDEOF     STRIP     SUM     STOR     ZERO

NO REFERENCE
BEGIN     CON#1     FIVE      MASK      START
     0   ERRORS     TAPE      3
     0   ERRORS     TAPE      4
END OF PASS 0
```

Figure 18. Printer Listing of Symbol Errors, Pass 0

PASS 0 MESSAGES, at the conclusion of or during pass 0, on the typewriter or printer indicate operating conditions. Some of these messages require immediate intervention in the form of switch settings or rereading of input cards. These messages for pass 0 are described in Figure 19.

Recovery of a Card Read Error is as follows. If, during the loading of the input source deck, a card read error occurs, pass 0 types the message CARD READ ERROR.

For the 400 cpm card reader:

1. Depress MANUAL on the GE-225 operator's console.
2. Remove the card deck from the input stacker; remove the card from the read platform and place it in front of the deck; load the card previously read incorrectly into the read platform; replace deck in the input stacker.
3. Press A→I on console.
4. Press AUTO and then START.

For the 1000 cpm card reader, follow essentially the same procedures, except for step 2. Step 2 for the 1000 cpm card reader requires returning the last card in the output stacker to the front of the deck in the input hopper and then performing steps 3 and 4.

MESSAGES	MEANING	RESULT
NO END CARD	Indicates that the symbolic deck does not terminate with and END card.	Assembly will continue to the normal end of job. After assembling, the transfer card may be punched manually and added at the end of the object deck.
END OF PASS 0	Signifies the end of the pass 0 run.	
ST1 OVERFLOW	Indicates that the number of special symbolic operands exceeds 250.	Program goes into loop which may be overridden by setting switch 19. This causes pass 0 to continue, but the special symbolic operands encountered after the error halt are not entered in symbol table 1. This may result in the improper assignment of a memory address to these symbols in the following passes.
CARD READ ERROR	Indicates that the last card read by the card reader may either have been mis-read or have a punch present in column 7 or 11. The card in error will be the last card fed into the output hopper.	If the card has a punch in either column 7 or 11, repunch the card correctly. Examine the card for off-center or physical damage (causes for mis-read). To recover and restart program follow the procedure for Recovery of Card Read Error which follows.

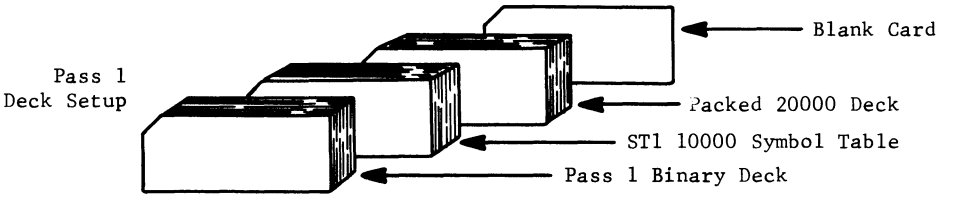
Figure 19. Pass 0 Messages

Other Halts or Loops are recovered by placing the central processor in manual mode. Check the ready status of all input/output devices. If any are in a not-ready status, take the necessary actions to ready these. Press AUTO then START.

If the CARD FEED light on the card reader is on, check the card deck for damaged cards. Replace such cards as necessary and reload pass 0 from the beginning.

Pass 1 Card-to-Card Operations

1. The output from pass 0 is input to pass 1, but it must be rearranged. The cards that have sequence numbers beginning with 10000 (columns 74-78) should be placed in front of cards starting with 20000 (columns 74-78). The cards are rearranged with binary pass 1 deck followed by output from pass 0, and one blank card.



2. Load cards into card reader.
3. Set console switches as desired for each result.
4. Process pass 1.

SORTED SYMBOL TABLE (ST2) is the output from pass 1 and gives the equivalent locations punched in the cards and, if the printer is on line, listed as they are punched. In addition, a list of all multiple-defined symbols, together with all of the equivalent values associated with each symbol, is printed (or typed, if no printer is available).

Errors or possible errors detected in the Operand field of a BSS, EQU, or ORG instruction are printed or typed with the present setting of the memory allocation register, the card type, and the error code. The error codes are:

U - an undefined symbol
A - a possible error in an address.

Figure 20 shows a listing of Symbol Table 2 and a representative ST2 card.

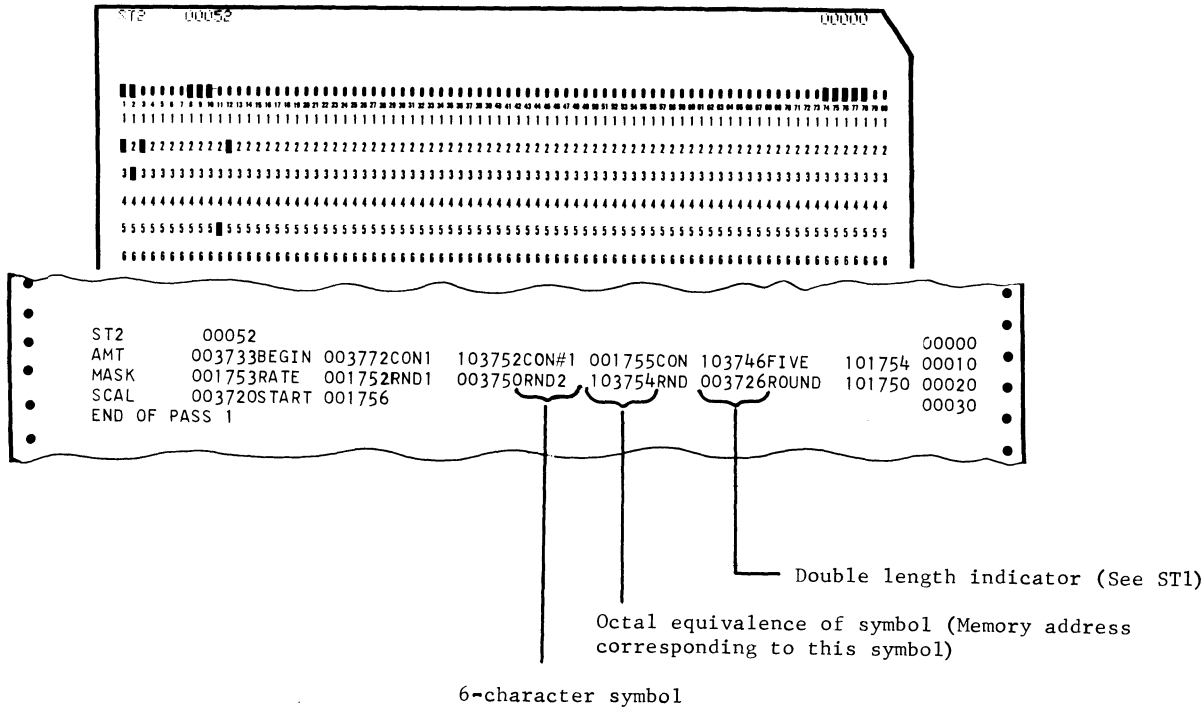


Figure 20. Sorted Symbol Table 2 (ST2)

In columns 8-12 of the card and the first line of the listing, the numerals 00052 are an octal control number (42_{10}) equal to 3 times the number of symbols in ST2 (14). In columns 74-78 of the card, and at the right side of the listing, are the assembly-program-created pass 1 sequence numbers (00000 series equals ST2). Other symbols on the printer listing are explained in Figure 20.

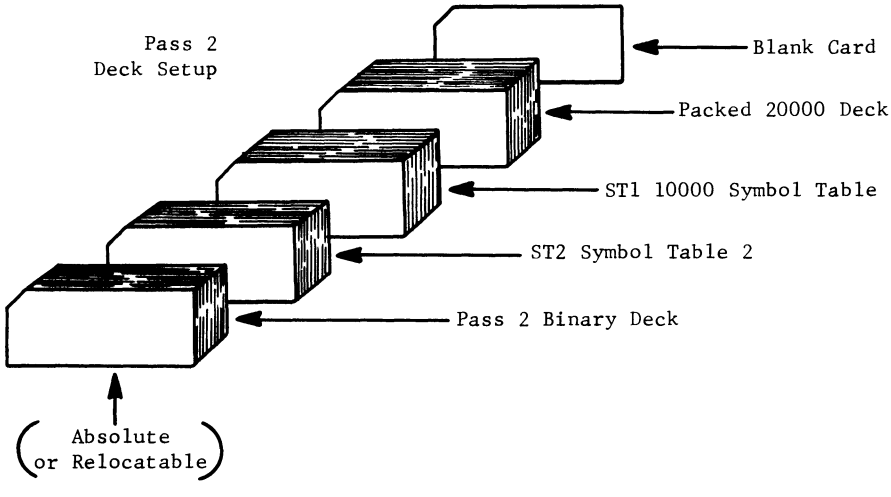
PASS 1 MESSAGES that can occur during this pass are shown in Figure 21.

MESSAGES	MEANING	RESULT
NO END CARD	Indicates that the symbolic deck does not terminate with an END card.	Assembly will continue to the normal end of job.
END OF PASS 1	Signifies the end of pass 1 run.	
ST2 OVERFLOW	Indicates that the number of special symbolic operands exceeds 250.	Same as for pass 0.
CARD READ ERROR	Card improperly read. Same as indicated for pass 0.	Same as for pass 0
ERROR IN DECK SETUP	Input card for pass 1 are not arranged properly.	Check arrangement, correct and reload pass 1.

Figure 21. Pass 1 Messages

Pass 2 Card-to-Card Operations

1. The input for pass 2A or pass 2R is the output from pass 1 and pass 0. Set up the input deck as follows: the output from pass 0 and 1 followed by one blank card:



2. Load cards in card reader.
3. Set console switches for desired end result.
4. Process pass 2.

THE OBJECT PROGRAM on cards is the output of pass 2. If a printer is available the output will include an assembly listing with indicated errors (O, U, A, M, S, T). If no printer is on line for pass 2 the cards will contain the octal memory location assigned to the instruction, symbolic instruction in octal, and the codes for read or suspected errors. Figure 22 shows the object program's octal deck of cards for execution with an octal loader or this card deck can be converted to a binary deck which can be loaded with a binary loader.

If a printer is on line, the output deck will be a binary deck. Figure 23 illustrates a binary card and the assembly listing of the object program as assembled from the instructions of the program coding sheet shown in Figure 2.

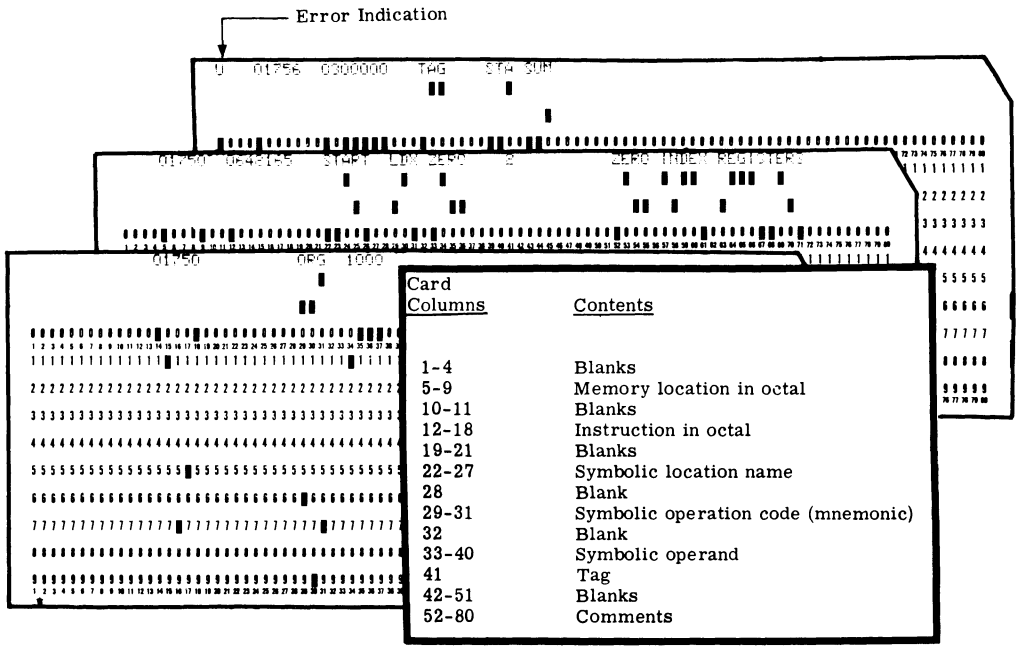
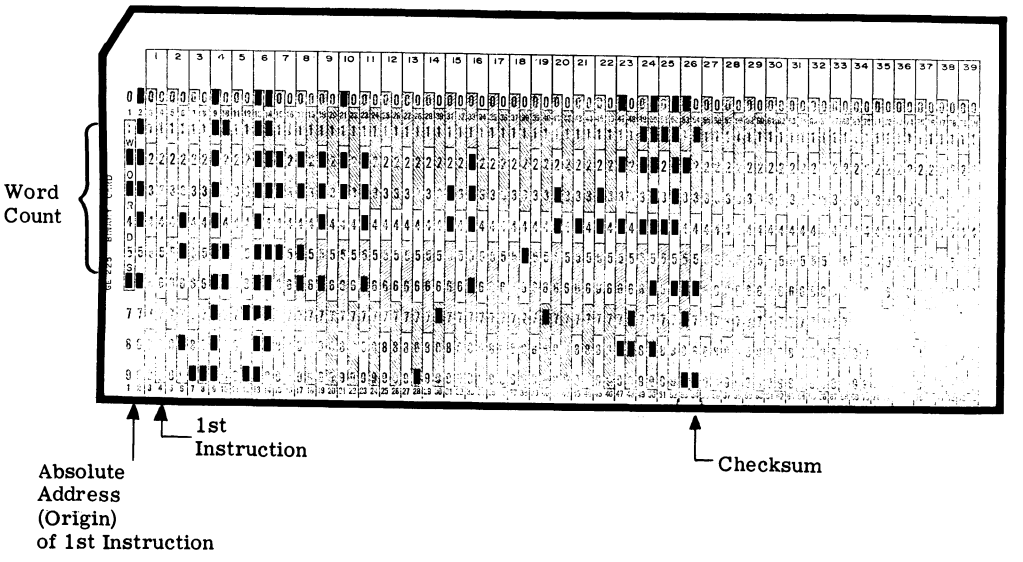


Figure 22. Object Program Card Deck




```
                01750                ORG 1000                                                    00005
01750           0000000   ROUND      DDC 50                 ROUNDING CONSTANT                00010
01751           0000062
01752           0002001   RATE       DEC 1025               PROCESS COST PER ITEM             00015
01753           3776430   MASK       OCT 3776430            MASKING CONSTANT FOR MOD ROUTINE  00020
01754           0000005   FIVE       DEC 5                                                    00025
01755           3777736   CON#1      DEC -34                                                  00030
01756           0640000   START      LDX ZERO    2          ZERO INDEX REGISTERS 2+3         00035
01757           0660000              LDX ZERO    3                                           00040
01760           0720000              SPB CRDIN   1          CARD READ SUBROUTINE             00045
01761           2600000              BRU CRDEOF             CARD END-OF-FILE RETURN          00050
01762           0720000              SPB STRIP   1          BCD-BINARY CONVERSION ROUTINE    00055
01763           0000000              DEC CRD                CARD IMAGE ORIGIN                00060
01764           0000001              DEC  1                 BEGINNING OF FIELD               00065
01765           0000004              DEC  4                 FIELD SIZE                       00070
01766           0300000              STA AMT#1              ITEMS PREVIOUSLY PROCESSED       00075
01767           0720000              SPB STRIP   1                                           00080
01770           0000000              DEC CRD                                                  00085
01771           0000020              DEC  16                                                  00090
01772           0000004              DEC  4                                                   00095
01773           0300000              STA AMT#2              ITEMS CURRENTLY PROCESSED        00100
01774           0100000              ADD AMT#1                                               00105
01775           0300000              STA SUM                TOTAL ITEMS PROCESSED            00110
01776           2504006              MAQ                                                     00115
01777           1501752              MPY RATE               PROCESS COST PER ITEM            00120
02000           1101750              DAD ROUND              ROUND PROCESS COST               00125
```

Figure 23. Object Program Assembly, Pass 2

PASS 2 MESSAGES are shown in Figure 24.

MESSAGES	MEANING	RESULT
ERROR	Indicates presence of a real or suspected programming error.	
NO ERRORS	Indicates no real or suspected programming errors were found.	
END OF PASS 2	Signifies the end of the pass 2 run.	
SYMBOL LOST	Indicates a symbol appearing in the Symbol field cannot be found in the symbol table.	This may result from a mispunched symbol table from pass 1, a bad read of the symbol table by pass 2, or a bad read of the present instruction card. The program will go into a loop. Whenever a symbolic name appears in the Symbol field, pass 2 must search the symbol table formed by pass 1 to insure that the assembled program is in phase with the memory assignment made in pass 1. Toggle switch 19 to finish assembly or restart assembly at pass 0.
	It can also be the result from overriding a symbol table overflow message in pass 1.	If this message is overridden, the symbol which was lost will not be found when it is used in the Operand field. This may result in a number of undefined references which must be corrected by the user.
ERROR IN DECK SETUP	Input cards for pass 2 are not arranged properly.	Correct the deck and reload pass 2.

Figure 24. Pass 2 Messages

CARD OPERATIONS WITH MAGNETIC TAPE AND PRINTER EQUIPMENT

The following operating instructions are for loading General Assembly Program II from cards for the following configuration:

Card Reader
Card Punch
2 Magnetic Tape Handlers
Typewriter
Printer (900 or 450 lpm)
2 additional Magnetic Tape Handlers (optional).

Pass 0 Cards With Magnetic Tape And Printer

1. Set up cards as for card-to-card pass 0.

2. Load cards into the card reader.

3. Set A-register input switches as desired for end results.

Switch	Setting	Result
2	Normal	Printer on line.
2	Down	No on-line printer.
3	Normal	1 additional magnetic tape available--for comments.
3	Down	Only two magnetic tapes available for assembly--no comments will appear on listing.
6	Normal	2 or 3 magnetic tapes available as specified by switch 3.
6	Down	1 additional magnetic tape available to write assembled binary program on tape 6. 3 or 4 tapes available as specified by switch 3.
16	Optional	

All other switches normal.

4. Start pass 0.

As the symbolic cards are read by pass 0, these will be written in card image records (27 words) on tape 3, if tape 3 is available. The packed symbolic output from pass 0 will be written on tape 4. The special symbols table (ST1) will be left in memory ready for pass 1.

PASS 0 MESSAGES will be the same as for card-to-card pass 0 plus the following conditions applicable to magnetic tape as shown in Figure 25 and the following:

1. If switch 3 is up, and the symbolic program is on tape 3 in decimal records 27 words long, switch 16 may be set down. This will cause pass 0 to read the symbolic program from tape 3 instead of from cards. If a read error occurs while reading from tape 3, the message, READ ERROR TAPE 3 RESTART PASS 0, will be typed and the program will halt. This means that either tape 3 must be corrected before trying again, or that assembly must be restarted from the original symbolic cards.

2. Whenever pass 0 detects a write error while writing tape 3 or tape 4, the assembly will rewrite the record until a successful write is performed.

   A count of the number of rewrites necessary during pass 0 is typed and printed, if the printer is on line. If these error counts are not zero, it does not necessarily mean that the assembly should be restarted. It is merely an indication of the number of times pass 0 was required to rewrite the tapes, in order to get a good tape.

MESSAGE	MEANING
XXX ERRORS TAPE 3	If tape 3 is used for comments, this typeout signifies the number of bad spots on tape 3.
XXX ERRORS TAPE 4	Signifies the number of bad spots on tape 4.
READ ERROR TAPE 3 RESTART PASS 0	Indicates a read error occurred while reading from tape 3.

Figure 25. Pass O Magnetic Tape Messages

Pass 1 Cards With Magnetic Tape and Printer

The procedure for processing pass 1 and the results of the processing are as follows:

1. The input to pass 1 is in memory and on tape 4.
2. Place two blank cards behind the pass 1 binary deck.
3. Load cards in the card reader.
4. Start pass 1.

SORTED SYMBOLIC TABLE (ST2) and equivalent values are the output from pass 1. This is written on tape 5 and printed, if the printer is on line. The remaining outputs are the same as for card-to-card pass 1.

PASS 1 MESSAGES and recovery procedures are the same as card-to-card pass 1.

Pass 2 Cards With Magnetic Tape And Printer

The procedure for processing pass 2 and the results of the processing are as follows:

1. The input to pass 2 is the output from pass 0 on tape 4 and on tape 3 (if present), and the output from pass 1 on tape 5.
2. Place two blank cards behind the pass 2A or 2R binary deck.
3. Load cards in the card reader.
4. Start pass 2.

THE OBJECT PROGRAM as the output from pass 2 is the same as the output from card-to-card pass 2. In addition, the program listing is written on tape 5, which may be used to obtain multiple listings. If tape 6 is available, the binary program will be written on tape 6 as it is being punched.

PASS 2 MESSAGES and recovery procedures are the same as card-to-card pass 2 plus the following:

ERROR TAPE 5 indicates that tape 5 was read incorrectly. Reload pass 2. If the error message is repeated, restart assembly from pass 0 with switch 16 down.

For all other errors, restart assembly from pass 0 with switch 16 down.

ALTERNATE ASSEMBLY CONFIGURATIONS

A user may wish to specify the input/output devices in a mixed fashion--for example, symbolic input from punched cards, the use of magnetic tapes as intermediate storage, and output on the printer and perforated tape. This can be attained by properly altering the console switch settings between passes. The minimum configuration required is:

Card Reader
Two Magnetic Tape Handlers
Typewriter
Printer
Perforated Tape Punch

Pass 0 Alternate Assembly

1. Arrange cards as for card-to-card pass 0.

2. Set console switches.

Switch	Setting	Result
2	Normal	Printer on line
3	Down	No magnetic tape 3
4	Normal	Tapes 4 and 5 used in assembly

3. Load pass 0 cards.

The output from pass 0 will be written on tape 4. The special symbol table will be left in memory for pass 1.

Pass 1 Alternate Assembly

1. No change in switches.
2. Load pass 1 cards.

The output from pass 1 will be written on tape 5.

Pass 2 Alternate Assembly

1. Set switch 12 down.
2. Load pass 2 cards.

The output from pass 2 will be a printed listing and paper tape program.

SYSTEMS TAPE

The assembly program can be operated from a systems tape by installations having magnetic tape capability. The tape assembly program (CD225F1.007) requires less assembly time than the card assembly program (CD225F1.006) because less card reading and card punching is necessary. The systems tape format is described in the following section.

Operating With Systems Tape

The description that follows is not for the BRIDGE II Compatible General Assembly Program system. Those installations utilizing the BRIDGE II Compatible systems must follow the special instruction provided with the program (CD225F1.008).

The General Assembly Program II master deck to produce the systems tape is made up of the components listed below, including service routines for the system such as memory dumps, tape dumps, etc. The user may add service routines to the assembly program master deck as desired. It is possible for the user to insert subroutines in the master deck, and up to 10 subroutines can be added to the symbolic deck at assembly time by use of the SBR pseudo-instruction.

THE MASTER DECK is formed as follows:

1. Tape writer
2. Define controller and handler
3. PCLLDR
   Utility programs + 1 blank
   (additional routines may be inserted)
4. WEFPCLLDR
   Rewind tape 3 and load test program + 1 blank
5. RCDWTB
   Run ID+EOT + 1 blank
6. LDR
   Test for ID and end of tape + 1 blank
7. RCDWTB
   Run pass 0 + 1 blank
8. LDR
   Pass 0 + 1 blank
9. WEFRCDWTD
   Subroutines + 1 blank (user option)
10.]]]]]] BSSO
11. WEFRCDWTB
    Run pass 1 + 1 blank
12. LDR
    Pass 1 + 1 blank
13. RCDWTB
    Run pass 2A + 1 blank
14. LDR
    Pass 2A + 1 blank

15. RCDWTB
    Run tape 31D + 1 blank

16. LDR
    Test tape 3 for ID record + 1 blank

17. RCDWTB
    Run pass 2R + 1 blank

18. LDR
    Pass 2R + 1 blank

19. RCDWTB
    Run RELOAD ID TEST + 1 blank

20. LDR
    Reload ID + 1 blank

21. WEFRWD + 2 blanks

THE TAPE FORMAT as produced by the master deck is as follows:

1. Tape loader
2. Memory dump program
3. Tape loader
4. Tape dump program
5. End of file
6. Tape loader
7. Rewind tape 3 for General Assembly Program II
8. Identification record
9. Tape loader
10. Test for program identification and end-of-tape record
11. Identification record
12. Tape loader
13. Pass 0
14. End of file (User's subroutine library inserted here.)
15. End of file
16. Identification record
17. Tape loader
18. Pass 1
19. Identification record
20. Tape loader
21. Pass 2A (Absolute)
22. Identification record
23. Tape loader
24. Reposition tape 3
25. Identification record
26. Tape loader
27. Pass 2R (Relocatable)
28. Identification record
29. Tape loader
30. Reload test program, blocks 9 and 10
31. End of file

INSTRUCTIONS FOR GENERATING THE SYSTEMS TAPE are as follows:

1. Remove the REM cards (00001-00005) from the front of the card deck for the General Assembly Program II (CD225F1.007).

2. Mount a blank magnetic tape on tape handler 1, channel 1. This will become the General Assembly Program II systems tape.

3. Load the card deck in the card reader and depress the console START switch.

While the deck is being read and the information is being written on magnetic tape, six cards will be punched by the card punch. Cards 1, 3, and 5 are in BCD mode and describe cards 2, 4, and 6, which are in binary mode. These cards are as follows:

BCD card	1	Program call card - file 1, record 1
Binary card	2	Memory dump call card
BCD card	3	Program call card - file 1, record 2
Binary card	4	Tape dump call card
BCD card	5	Program call card - file 2, record 1
Binary card	6	General Assembly Program II call card.

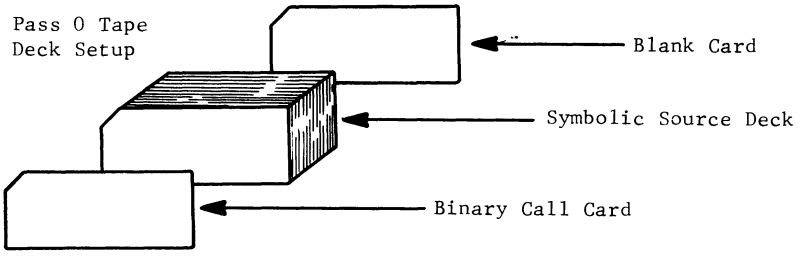
The binary call cards (cards 2, 4 and 6), when loaded through the card reader, locate and load the binary tape loader (generated with the systems tape) associated with the desired program. The appropriate tape loader then loads the program.

Upon successful completion of loading the information from the cards onto magnetic tape, the typewriter prints out END OF RUN and the tape rewinds.

SYSTEMS TAPE OPERATIONS are as follows:

Pass 0 Systems Tape

1. Set console switches as desired.
2. Place symbolic input program behind the assembly program call card:



3. Load cards in the card reader.

The output from pass 0 will be as specified by the switches. At the end of pass 0, pass 1 will automatically be loaded into memory, and will go into execution phase.

Pass 1 Systems Tape. The output from pass 1 will be as specified by the pass 1 switch settings previously described. At the end of pass 1, pass 2 will automatically be loaded into memory, and will go into execution phase.

Pass 2 Systems Tape. The output from pass 2 will be as specified by the switches. At the end of pass 2, the systems tape will rewind and the processor will halt.

If a tape or checksum error is detected during the actual loading of the assembly, the computer will halt at location 44. Restart the assembly from the beginning. If the error halt occurs again, use the General Assembly Program II master deck to rewrite the system tape and try again.

Switch settings may be altered during the assembly. Each pass reads the control switches once at the beginning of each run. After each pass begins reading the input, the switches can be altered as desired without affecting the execution of the current phase. Duplicate listings of the program can be obtained by printing tape 5.

Addition Of Service Routines

The user may add service routines to the master deck by removing any card loaders from his service routine deck. The deck consisting of the binary program cards and the program transfer card is inserted in the master deck between blocks 3 and 4 in the master deck format. Use the altered deck to write a new systems tape. During generation of the system tape, a call card is punched for each service routine in the master deck.

Addition Of Symbolic Subroutines

It is also possible for symbolic subroutines to be placed onto the systems tape by modifying the assembly program master deck. The subroutines placed on the tape are normally the ones most frequently used. A typical arrangement might be:

Name	Description	Name	Description
CHOOSE	Least Key Finder	SORT	Internal Memory Sort
CRDIN	Card Read Routine	PRINT 1	Typewriter Print Routine
IDBNPR	Internal BCD-to-Binary	DMPY	Double Precision Multiply
NPRIBD	Internal Binary-to-BCD	TRACE	Trace Routine
TPI/O	Tape Input and Output		

Although more subroutines can be stored on the tape, no more than ten can be called for during one assembly.

The advantages of having the subroutines on tape are:

1. Reduces card reading time
2. Reduces card handling by operators and programmers
3. Makes changing and maintaining the routines easier.

THE SUBROUTINES ARE CALLED FOR as needed, by the programmer, using the SBR pseudo-instruction.

To prepare a subroutine for addition to the deck, perform the following steps:

1. Obtain a deck of the symbolic program cards for the desired subroutine.

2. Punch two symbolic cards to be placed at the front of the deck.

Card 1	columns	1-6	Subroutine name
		8-10	BSS
		12	0
Card 2	columns	8-10	REM

3. Punch one symbolic card to be placed at the end of the deck.

Columns	8-10	END
	12	0

4. Run this modified deck through pass 0 with switch 4 down to obtain a deck of punched cards. This deck will be as follows:

Card 1	columns 1-6	Subroutine name
	7-9	BSS
	10	0
	74-78	20000
Card 2	columns 7-9	REM
	74-78	20010
Card 3	Packed symbolic subroutine	
	columns 74-78	20020
Card N-1		
Card N	columns 7-9	END
	10	0
Card N+1	columns 1-3	ST1
	9-12	Number of special symbols
	74-78	10000
Card N+2	Special symbol table (These will not appear if there are no special symbols.)	
Card N+M		

5. Insert cards N + 1 and N + 2 through N + M (if any) between cards 1 and 2.

6. Insert this deck in block 9 in the assembly program master deck. If two or more subroutines are in the library, they must be in ascending order according to the binary value of columns 1-6, card 1 contains the symbolic name of the subroutine.

7. Use the altered deck to write a new systems tape.

Any modifications to the General Assembly Program II must be inserted in the appropriate program (blocks 8, 11, 13, and 17) in the master deck.

Where a subroutine contains a name in the symbol field of its first instruction, it is a simple matter to adopt this symbol as the SBR call name. Most of the programming routines have this symbol--FLIP, STRIP, etc. Where a subroutine does not contain a name in the symbol field of the first instruction, the user could insert the symbol he has chosen for his SBR call in this field. This ensures that the General Assembly Program will link the SBR name with a symbol in the subroutine.

When a subroutine which is not on tape is called for, a printout occurs in pass 0. This printout gives the subroutine call symbol followed by SBR OP--for example:

```
CRDIN  SBR  OP
```

Multiple Assemblies

WHEN AN ABSOLUTE OBJECT PROGRAM IS DESIRED, using the systems tape to assemble, the General Assembly Program II assembles one program.

WHEN A RELOCATABLE OBJECT PROGRAM IS DESIRED, using the system tape to assemble, it is possible to write several programs on tape 3 and to assemble these consecutively. If any one of the pseudo-instructions SBR, MAL or PAL appears in the source program contained on tape, it will not be possible to call the source program from tape 3 for assembly. The three instructions, if encountered on tape 3, will cause the assembler to write on tape and in so doing destroy some of the user's coding. SBR, for example, will cause the requested subroutine to overlay the END card image. MAL and PAL will cause the requested alphanumerics to be written over the adjacent sequence of instructions.

Each program on tape 3 should be preceded by an identification card containing an asterisk in column 1. Columns 2-80 may contain any BCD information desired. There must be an end-of-file record after each symbolic program on tape 3. In addition, an end-of-tape record must be written on tape 3 after the last program to be assembled.

End-of-Tape Card

The format of the end-of-tape card is as follows:

Columns	Contents
1	Asterisk
4 - 6	END
7	Blank
8 and 9	OF
10	Blank
11 - 14	TAPE

At the end of pass 0, and in preparation for pass 1, tape 3 is positioned at the beginning of the source program that was just read. To accomplish this repositioning, the assembler backspaces the tape until it finds the beginning of the program. The repositioning is not accomplished by executing the rewind command, because the tape may contain more than one symbolic program. Thus, the beginning of tape may not necessarily be the beginning of the program being assembled.

Forward Sort/Merge Generator (CD225G1.002), ZOOM-A (CD225F1.002) and pass 0 of the assembler will build tape 3 into 27-word records. The assembler expects to find, and checks for, 27-word records. If an input tape is built by a user's program with a record size not equal to 27, the tape will be searched until a 27-word record is found. This search could continue to the end of tape.

BRIDGE II COMPATIBLE SYSTEM

BRIDGE II is used to build and maintain computer programs on magnetic tapes in a standardized format. This format includes the presence of a Next Run Locator program between each of the user's programs on tape. BRIDGE II is used to prepare program tapes for execution, but does not itself participate in their execution. The Next Run Locators handle the finding and starting of each program in its turn.

General Assembly Program II has been made compatible with BRIDGE II so that it may exist on a system tape with other systems such as FORWARD, GECOM, and ZOOM-A that are compatible and can be maintained by BRIDGE II. All input, output, and the instruction repertoire are the same. The only requirements are that the minimum system configurations be:

Card Reader
Card Punch
3 Magnetic Tapes (minimum)
Typewriter
8k Memory.

As optional equipment, a printer and 2 magnetic tapes may be added.

System Tape Setup

The systems tape (CD225F1.008) provided by General Electric's Program Library will contain the following programs:

BRIDGE II (CD225J1.001)
GAP II (BRIDGE Compatible version) (CD225F1.008)
COBOL (CD225H5.000)
GECOM II (GECOM Compatible version) (CD225H1.005)
FORWARD SORT/MERGE (CD225G1.006)
FORTRAN (CD225H6.001)

The user can copy from the tape any program or group of programs or, if it is desired, the tape can be used in its entirety. The General Assembly Program II (BRIDGE Compatible version) portion of the systems tape is generated from the following parts in the sequence outlines below.

1. Start Card
2. SCC RUNBRIDGE II
3. DOP
4. LBLBTL001GAPSYSTEM
5. PRG RUNGAPII PO
6. LBLRUNGAPII PO ABS
7. Loader
8. GAP Pass 0
9. SYM SYMSUBRUTINE
10. Users Symbolic Subroutines Prepared as for GAP II
11. ENDO
12. END
13. PRG RUNGAPII P1
14. LBLRUNGAPII P1 ABS
15. Loader
16. GAP Pass 1
17. PRG RUNGAP II P2A
18. LBLRUNGAPII P2AABS
19. Loader
20. GAP Pass 2A
21. PRG RUNGAPII P2R
22. LBLRUNGAPII P2AABS
23. Loader
24. GAP Pass 2R
25. CPT
26. One Blank Card
27. EBS
28. Two Blank Cards

Symbolic Subroutines

Subroutines may be added to the General Assembly Program II portion of the systems tape following pass 0. These subroutines are in packed symbolic form and are inserted following the

SYM SYMSUBRUTINE

card that immediately follows pass 0 in the deck. The following configuration of cards defines the sequence necessary to copy General Assembly Program II and insert the desired subroutines.

1. Start Card (BRIDGE Control) RUNBRIDGEII
2. SCC
3. DOP
4. THA
5. LBLBTL001 (optional)
6. COP BTL MASTERRUN RUNGAPII PO RUNGAPII PO
7. SYM SYMSUBRUTINE
8. Users Packed Symbolic Subroutines
9. COP BTL MASTERRUN RUNGAPII P1 RUNGAPII P2R
10. CPT
11. One Blank Card
12. EBS
13. Two Blank Cards

Detailed information for the copying process together with blocking, labels, etc. will be found in the BRIDGE II Operating Service System manual (Publication No. CPB-1039).

Assembly

START and SCC (RUNGAPII PO) cards are sent out with the systems tape for use with BRIDGE Compatible General Assembly Program II. When assembling a program using BRIDGE Compatible General Assembly Program II these cards are used at the beginning of the card deck in place of the General Assembly Program call card. Assembly follows as previously described for General Assembly Program II Assembly.

MODIFICATIONS TO GENERAL ASSEMBLY PROGRAM II

Modifications to General Assembly Program II are accomplished by changing the binary program decks with binary correction cards. The addresses to be altered to accomplish modifications are a function of each particular General Assembly Program II deck. Each General Assembly Program II deck that is issued by the General Electric Program Library is accompanied with a sheet that details the necessary addresses.

Symbol Table Length and 8K Modifications

General Assembly Program II requires a minimum of 4096 words of core storage. Each program is packed at the beginning of memory. All of the available memory following each program is used for working storage. If a larger memory is available, the following corrections to pass 0 and pass 1 serve to increase the available working storage. This allows the assembly program to form a larger symbol table but has no effect on the internal functions.

Program	Octal Location	Correction for 8192 words
Pass 0	XXXXX	XXXXXXX
Pass 1	XXXXX	XXXXXXX

Note: X = Information contained on the information sheet issued with the program deck by the Program Library.

These alterations specify to pass 0 and pass 1 that the constant in the specified locations is the address of the last memory location that may be used. This may be set to any desired constant beyond the programs at the user's convenience. For example, if the user desires to reserve the last 256_{10} locations of an 8192 memory, these corrections should be xxxxxxx (contained on the information sheet which is issued with the program deck by the program library). The size of the symbol table which may be held by each pass is included in the following lists of symbol table characteristics:

SYMBOL TABLE 1:

1. Built-in pass 0 only
2. Consists of DL, I/O, FLP, and document handler symbols
3. Maximum size is - - - decimal
4. Printed out at end of pass 0
5. Error message SYMBOL TABLE OVERFLOW 1 when table capacity is exceeded.

SYMBOL TABLE 2

1. Built-in pass 0 and pass 1

2. Consists of all symbols used in a program

3. Maximum size is indicated on assembly listing in decimal form immediately following the words GAP 0 and GAP 1

4. Table size variable if the number of instructions contained in pass 0 or pass 1 is changed. (The General Assembly Program itself determines the number of symbols each pass will hold. The table size is printed out at the beginning of each assembly)

5. Printed out only at end of pass 1

6. Error messages:

   Pass 0--SYMBOL TABLE OVERFLOW 2 when table size exceeds the number indicated on the listing at beginning of the pass

   Pass 1--SYMBOL TABLE OVERFLOW when table size exceeds the number indicated on the listing at beginning of the pass. (This number will not be the same in both pass 0 and pass 1.)

Priority Control Channel Assignment Modification

Without modification, the tape controller is on priority control channel 1 and the printer controller is on channel 6.

Modification of channel assignments is accomplished by inserting binary correction cards just ahead of the branch card of each deck.

The octal format of the controller number assignment word is: 000000P.

Program	Octal Location of Printer Channel No.	Octal Location of Tape Channel No.
Pass 0	XXXXX	XXXXX
Pass 1	XXXXX	XXXXX
Pass 2A, Absolute	XXXXX	XXXXX
Pass 2R, Relocatable	XXXXX	XXXXX

Note: X = Information contained on the information sheet which is issued with the program deck by the program library.

Caution: All tape channel numbers or all printer channel numbers for all four programs should be changed at one time.

For example, to change the tape controller from channel 1 to channel 2:

1. Make up octal correction cards.

	Card Columns 5-9	12-18
For Pass 0 Card 1	XXXXX	0000002
For Pass 1 Card 2	XXXXX	0000002
For Pass 2A Card 3	XXXXX	0000002
For Pass 2R Card 4	XXXXX	0000002

Note: X = Information contained on the information sheet issued by the program library.

2. Use an Octal-to-Binary Card Converter to convert the octal correction cards to binary correction cards with checksum and origin. (Use CD225C3.002.)

3. Insert the respective binary correction cards just before the transfer cards of passes 0, 1, 2A, and 2R.

Vacuum Pocket Retrofit Modification

The General Assembly Program II assumes that magnetic tapes handlers have vacuum pockets. If the handlers do not have vacuum pockets, the following locations in each program should be altered by inserting binary correction cards just ahead of the branch card.

Program	Octal Location	Octal Instruction
Pass 0	XXXXX	XXXXXXX
Pass 1	XXXXX	XXXXXXX
Pass 2A,	XXXXX	XXXXXXX
Absolute	XXXXX	XXXXXXX
Pass 2R,	XXXXX	XXXXXXX
Relocatable	XXXXX	XXXXXXX

Note: X = Information contained on the information sheet issued with the program deck by the program library.

Modifications to assembly program card decks are easily made by punching the octal correction cards, converting them to binary, and inserting the binary card before the respective transfer cards of each assembly program deck.

System Tape Controller Channel Modification

To change the tape controller channel number for the system tape, it is necessary to change the systems tape definition card.

Punch a card as follows:

Columns	Contents
1-3	CON
9	Controller Channel Number
15	1

Tape loaders will be altered and written as required on the master tape. In addition, new call cards for the programs in block 3 and in block 4 will be punched for the specified tape. The call card punched for block 4 (file 2; program 1) is the required assembly program call card for this tape. This individual pass must be modified by inserting binary correction cards just ahead of the branch card.

RELOCATABLE OBJECT PROGRAMS

General

General Assembly Program II can provide the user with relocatable object programs. The binary cards produced are in a format acceptable to the Multi-Capability Modular Loader (MCML II). For additional information, refer to the MCML II (CD225B1.006R) documentation.

To conform to the requirements of MCML, a TCD card causes a type 5 card containing the appropriate address to be punched, and an END card causes a type 3 card containing the transfer address to be punched.

Figure 26 shows the format for relocatable instruction cards. Figure 27 shows the format of assembly listings produced with the relocatable instruction cards. The assembly listing is augmented with a relocate indicator that appears in column 20 (between the octal display of the instruction and the Symbol field). The relocate indicator takes on three values:

0--absolute, no modification requested
1--develops positive address relative to the relocation constant
2--develops negative address relative to relocation constant

The number indicates what relocation key is associated with the instruction word as it appears in the binary card image. The number thereby indicates how the relocatable loading process will modify the operand portion of the instruction.

The relocate flag also appears in the octal card output of General Assembly Program II in column 20. The octal card output can be converted to a relocatable binary deck by using the Octal to Relocatable Binary Converter routine (CD225C3.004R).

Calculation of Checksum

Figure 26 illustrates the relocatable instruction card format. The checksum is punched in bit positions 7-9 of column 3 and bit positions 0-9 of column 4. The checksum is the sum of all words in the load string with the exception of the checksum itself.* Because there is the possibility of overflow there must be a test for overflow after each addition and a 1 must be added to the sum whenever overflow does occur. The 20-bit calculated checksum must be punched in the 13 bits allowed for it in the relocatable card format. For this reason, bits 0-6 of the calculated checksum must be added to bits 7-19 of the checksum. There is the possibility of overflow as the result of this addition and, in the case of overflow, a 1 must be added to the 13-bit checksum. The following coding illustrates how the checksum may be calculated for information punched on a card in the relocatable format. This example assumes that the overflow indicator has been cleared, index word zero contains the word count (WDCT), and X-word 1 contains the word currently being added to the checksum.

* A load string consists of one or more sequential instructions to be consecutively loaded into memory immediately preceded by two words which contain the origin, the checksum, and other information (see Figure 27).

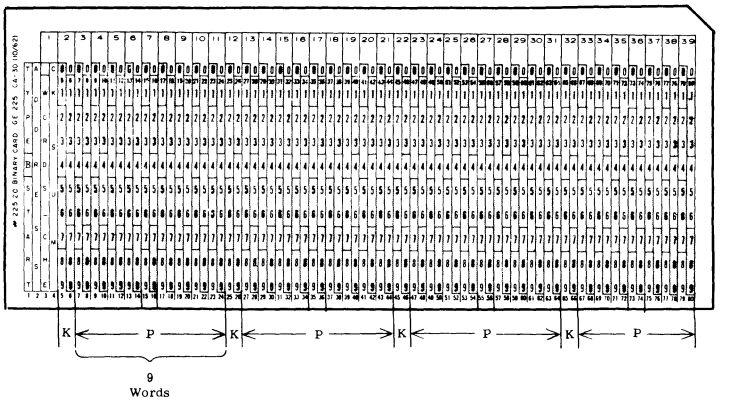


Figure 26. Relocatable Instruction Card

Field	Column	Rows	Description
TYPE	1	0-3	Type indicator: a 03 indicates a standard relocatable instruction card; a 05 indicates a mark transfer point; a 11 indicates a loader card.
B	1	4	Checksum override indicator. 0 = card checksummed during loading. 1 = checksum ignored. Usually the card will contain a checksum.
START ADDRESS	1 2	5-9 0-9	Location of the first instruction relative to the program origin.
D	3	0	Not used
WORDS	3	1-6	Number of program words on the card.
CK SUM	3 4	7-9 0-9	Checksum.
K	as shown		Relocation key or control word for the next sequence of one to nine program words. This key contains a relocation indicator for each of the following program words (P). The K field consists of nine two-bit fields each of which contains a code which applies only to the operand portion of the program word. This code is: Code (Binary) — Operand Address is 00 — Absolute 01 — Positive address relative to the relocation constant 10 — Negative address relative to relocation constant Control Word Bits 0 1 2 3 4 5 6 7 8 9 ⟷ 18 19 not used — 1st 2nd ⟷ 9th Program Word
P	as shown		Program words to be stored in consecutive memory locations relative to the location given in field.

Figure 27. Relocatable Assembly Listing Format

```
            LDX     WDCT        0
            LDX     ZERO        1
            LDA     START               First two words
            ADD     START + 1
ADD         ADD     START + 2   1
            BOV
            ADO                         Check A-register
            INX     -1          0       Compute 20-bit checksum
            INX      1          1
            BXH      1          0
            BRU     ADD                 Save 13 bits of
            STO     CKSM                checksum
            SRA     13                  Position bits 0
            EXT     MSK                 through 6 of checksum
            ADD     CKSM
            SLA     6
            SRA     6                   Test for overflow
            BOV
            ADO
            STO     START + 1

MSK         OCT     3777600
CKSM        DEC     0
ZERO        DEC     0
WDCT        DEC     0                   Word computed by assembly program
```

PERFORATED TAPE ASSEMBLY

General

General Assembly Program II accepts perforated tape as an input medium. The output can be in the same form, if desired.

The perforated tape input is punched from the regular assembly program coding sheet using the standard Friden Flexowriter* SPD character set. The character set is shown in Figure 28. The first character punched must be a carriage return (CR) followed by the 80 columns from the sheet. Each source line of coding punched from the coding sheet must be separated by a CR. However, perforated tape codes prepared by off-line devices may vary widely and the user may wish to modify the perforated code used by General Assembly Program II. This is easily accomplished by changing the conversion tables as described below.

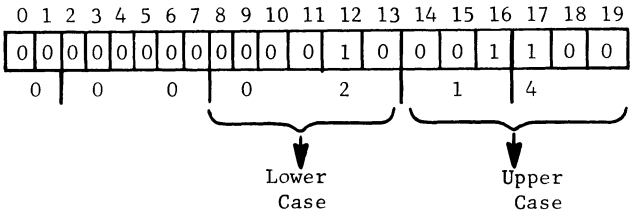
Perforated Tape Input/Output Conversion Tables

Pass 0 reads and converts perforated tape codes to internal GE-225 BCD from a conversion table. This table can be modified by correction cards to conform to the user's requirements.

Each entry in the table of characters carries two BCD configurations: one corresponding to an upper case tape code and one to a lower case. For example, the second entry in the memory word shown below contains the BCD character corresponding to a lower case perforated tape, code for the numeral 2 in bit positions 8-13 and the BCD character corresponding to an upper case perforated tape code for the numeral 2 in bit positions 14-19.

* Trademark of Friden, Inc.

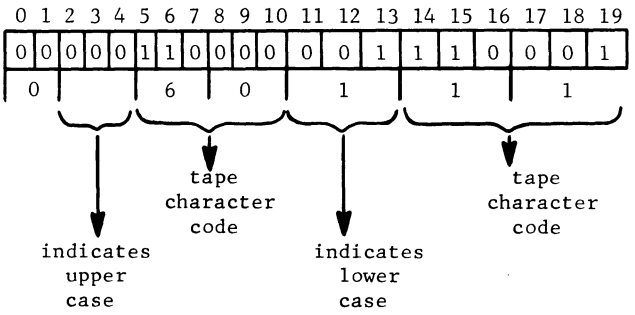
Memory Word



As pass 0 and pass 1 prepare input tapes for succeeding passes, no conversion is required, These intermediate tapes are punched in codes corresponding to internal GE-225 BCD.

The output from pass 2 may be listed on off-line devices which require a character conversion. This is accomplished by another conversion table. A typical memory word appears below.

Memory Word



In pass 2A (absolute) this table occupies locations as are specified in the information sheet issued with the program deck. Each word in the table carries the perforated tape codes for two internal BCD characters, the first code in bit positions 2-10, the second in 11-19. In each 9-bit configuration, the first octal digit is a 0 if the corresponding tape code is in upper case and a 1 if it is in lower case. The next two octal digits contain the tape code itself.

Note that, in the character set shown in Figure 28, several of the Flexowriter codes are meaningless (with the two exceptions given below) to the GE-225 and are interpreted as spaces (memory octal 60). Also, any other of the Flexowriter codes that are not listed are meaningless and interpreted as spaces. The two exceptions are LOWER CASE and UPPER CASE. These are not meaningless, as they are necessary in the interpretation of two identically-punched codes that may be punched from the same Flexowriter key, such as 3 and #.

Tape Character	Lower Case	Upper Case	Paper Tape Code Channel Numbers									Memory Code (Octal)
			8	7	6	5	4	S	3	2	1	
0	x				●			.				00
1	x							.			●	01
2	x							.		●		02
3	x					●		.		●	●	03
4	x							.	●			04
5	x					●		.	●		●	05
6	x					●		.	●	●		06
7	x							.	●	●	●	07
8	x						●	.				10
9	x					●	●	.			●	11
A	x	x		●	●			.			●	21
B	x	x		●	●			.		●		22
C	x	x		●	●	●		.		●	●	23
D	x	x		●	●			.	●			24
E	x	x		●	●	●		.	●		●	25
F	x	x		●	●	●		.	●	●		26
G	x	x		●	●			.	●	●	●	27
H	x	x		●	●		●	.				30
I	x	x		●	●	●	●	.			●	31
J	x	x		●		●		.			●	41
K	x	x		●		●		.		●		42
L	x	x		●				.		●	●	43
M	x	x		●		●		.	●			44
N	x	x		●				.	●		●	45
O	x	x		●				.	●	●		46
P	x	x		●		●		.	●	●	●	47
Q	x	x		●		●	●	.				50
R	x	x		●			●	.			●	51
S	x	x			●	●		.		●		62
T	x	x			●			.		●	●	63
U	x	x			●	●		.	●			64
	x	x			●			.	●		●	65
W	x	x			●			.	●	●		66
X	x	x			●	●		.	●	●	●	67
Y	x	x			●	●	●	.				70
Z	x	x			●		●	.			●	71
SPACE	x	x				●		.				60
@		x						.		●		14
#		x				●		.		●	●	13
$		x						.	●			53
=		x				●		.	●		●	16
*		x					●	.				54
(		x				●	●	.			●	75
)		x			●			.				76
/	x				●	●		.			●	61
:		x			●	●		.			●	60
-	x			●				.				40
"		x		●				.				60
%	x			●		●	●	.		●	●	74
—		x		●		●	●	.		●	●	60
+	x			●	●	●		.				20
:		x		●	●	●		.				60
.	x	x		●	●		●	.		●	●	33
,		x						.			●	60
LOWER CASE				●	●	●	●	.		●		60
UPPER CASE				●	●	●	●	.	●			60
CARRIAGE RETURN	x	x		●				.				

Figure 28. Perforated Tape Character Set 8-Channel, Friden Flexowriter Model SPD

DOCUMENT REVIEW SHEET

TITLE: GE-200 Series General Assembly Program II

CPB #: 1180

FROM:

Name: _____

Position: _____

Address: _____

_____

Comments concerning this publication are solicited for use in improving future editions. Please provide any recommended additions, deletions, corrections, or other information you deem necessary for improving this manual. The following space is provided for your comments.

COMMENTS: _____

Please cut along this line

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
Fold on two lines shown on reverse
side, staple, and mail.

STAPLE STAPLE

FOLD

FIRST CLASS
PERMIT, No. 4332
PHOENIX, ARIZONA

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

GENERAL ELECTRIC COMPANY
COMPUTER EQUIPMENT DEPARTMENT
13430 NORTH BLACK CANYON HIGHWAY
PHOENIX, ARIZONA - 85029

ATTENTION: DOCUMENTATION STANDARDS AND PUBLICATIONS B-90

FOLD

Progress Is Our Most Important Product

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

LITHO U.S.A.