SPECTRA 70

RADIO CORPORATION OF AMERICA · ELECTRONIC DATA PROCESSING

SYSTEM

70 15

PROGRAMMING SYSTEM INFORMATION MANUAL

RADIO CORPORATION OF AMERICA

70-15-501

November 1965

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

First Printing: November, 1965

CONTENTS

Page

INTRODUCTION 1

ASSEMBLY SYSTEM
Functional Description 2
Input/Output Description 3
Equipment Requirements 4
Memory Requirements 4
Related Programming Components 5
Accuracy Control 5
Timing 5

INPUT/OUTPUT CONTROL SYSTEM (IOCS)
Functional Description 6
Input/Output Description 7
Equipment Requirements 9
Memory Requirements 9
Related Programming Components 9
Accuracy Control 9
Timing 9

REPORT PROGRAM GENERATOR (RPG)
Functional Description 10
Input/Output Description 11
Equipment Requirements 11
Memory Requirements 11
Related Programming Components 11
Accuracy Control 11
Timing 11

LOADERS
Functional Description 12
Input/Output Description 12
Equipment Requirements 15
Memory Requirements 15
Related Programming Components 15
Accuracy Control 15
Timing 15

CARD TO TAPE
Functional Description 16
Input/Output Description 16
Equipment Requirements 17
Memory Requirements 17
Related Programming Components 17
Accuracy Control 17
Timing 17

TAPE TO PUNCH
Functional Description 18
Input/Output Description 18
Equipment Requirements 18
Memory Requirements 18
Related Programming Components 18
Accuracy Control 18
Timing 18

CONTENTS (Continued)

		Page
TAPE TO PRINTER	Functional Description	19
	Input/Output Description	19
	Equipment Requirements	19
	Memory Requirements	19
	Related Programming Components	20
	Accuracy Control	20
	Timing	20
SINGLE-PHASE MEMORY DUMP/ SNAPSHOT	Functional Description	21
	Input/Output Description	21
	Equipment Requirements	22
	Memory Requirements	22
	Related Programming Components	22
	Accuracy Control	22
	Timing	22
DUAL-PHASE MEMORY DUMP/ SNAPSHOT	Functional Description	23
	Input/Output Description	23
	Equipment Requirements	24
	Memory Requirements	24
	Related Programming Components	25
	Accuracy Control	25
	Timing	25
TAPE EDIT	Functional Description	26
	Input/Output Description	26
	Equipment Requirements	27
	Memory Requirements	27
	Related Programming Components	27
	Accuracy Control	27
	Timing	27
PROGRAM BINDER	Functional Description	28
	Input/Output Description	29
	Equipment Requirements	30
	Memory Requirements	30
	Related Programming Components	30
	Accuracy Control	30
	Timing	30
PLT UPDATE	Functional Description	31
	Input/Output Description	31
	Equipment Requirements	32
	Memory Requirements	32
	Related Programming Components	32
	Accuracy Control	32
	Timing	32

CONTENTS (Continued)

Page

SINGLE-CHANNEL COMMUNICATIONS CONTROL SYSTEM

Functional Description 34
Input/Output Description 35
Equipment Requirements 36
Memory Requirements 37
Related Programming Components 37
Accuracy Control 37

SORT/MERGE GENERATOR

Functional Description 38
Input/Output Description 38
Equipment Requirements 39
Memory Requirements 39
Accuracy Control 39
Timing 39

APPENDIX A 70/15 PROGRAMMING SYSTEM STANDARDS

General 40
Library Standards 40
HSM Layout 47
Programming Standards 50
Operator/Machine Communication 53
Implementation Language Standards 53

PROGRAMMING SYSTEM

INTRODUCTION

◆ The RCA 70/15 Programming System contains a set of interrelated programming components that enhances the inherent capabilities of the Spectra 70/15 Processor and associated peripheral devices. The system provides a programming foundation that not only accommodates a broad range of applications but also affords the additional advantage of complete device interchangeability. Except for the Sort/Merge Generator, all routines can be operated on a basic 70/15 configuration consisting of 4K-byte processor, card reader, printer, and card punch. The Sort/Merge Generator requires an 8K-byte processor, three magnetic tapes, card reader, printer, and card punch.

A basic card-oriented library is provided. This library may be further supplemented by the addition of magnetic tapes to provide greater operating versatility. Provision is also made for adapting the library to programs stored in card image on magnetic tape.

The 70/15 Programming System converts symbolic language to machine language, assists in running segmented programs, and provides standard operational routines. Parameters initiate system calls and designate device assignments. By stacking the parameter cards, a sequential set of production programs may be executed.

This system anticipates automatic programming requirements from the inception of processing to its termination, from program assembly to report generation. An Input/Output Control System (IOCS), which may be assembled with the program or linked to it by the loader portion of the system, affords complete data exchange between the processor and on-line peripheral devices. In addition, a system maintenance feature provides for the updating of program or data files and the combining of subprograms or independent programs into a common system.

To simplify the testing of production programs, a complete set of diagnostic routines is available. These routines consist of a variety of memory dumps that print the contents, or selected areas, of memory during or after program testing. Several utility routines are also provided that perform functions such as card-to-tape, card-to-punch, and tape-to-printer. These utility routines are so designed, that any two may concurrently share the processor and be accessed during the same object run.

UTILITY ROUTINE CO-SHARING

◆ The 70/15 Card-to-Tape, Tape-to-Punch, and Tape-to-Printer routines allow, after binding, concurrent processing of any two routines. This feature also reduces program set-up and take-down time and eliminates reloading of the routines. These routines are co-shared as follows:

When two routines are referred to in the same program, the operator inserts a nonzero character into a standard memory location before the production run. After the first utility routine has been initialized, this character is interrogated. When a nonzero character is sensed, a halt occurs followed by a branch to the Program Loader routine. This gives the operator the opportunity to insert an End card which transfers control to the second utility routine. When entering into the shared routine, linkage is established between the two routines by moving the entry address of the first routine to the exit address of the second routine, and the entry address of the second routine to the exit address of the first. Co-sharing now exists for the remainder of the operation.

ASSEMBLY SYSTEM

FUNCTIONAL DESCRIPTION

◆ The RCA 70/15 Assembly System is a machine-oriented, automatic Assembler that simplifies and expedites the writing of programs for the RCA Spectra 70/15 System. The Assembler translates symbolic source-language statements into computer-recognizable object coding.

The Assembler is a basic two-pass card system that permits device interchangeability so that magnetic tape can be substituted at load time for a card reader, card punch, or printer. Provision is also made to process stacked programs sequentially when magnetic tapes are used. An assembly can then be made on a minimum source configuration for a maximum object configuration.

The Assembler consists of two main program segments; one to process each of the two passes of the source program. A description of each pass is as follows:

1st Pass — A table of name-address assignments is created in memory from the source card input.

2nd Pass — The original source card deck is also used as input to the second pass. The operands and operations are defined in this pass and the object machine-code program deck is generated on cards. An assembly listing is also printed.

Figure 1 illustrates the basic operation of the Assembler.

Note: If a magnetic tape is available, it can be used as input to the second pass in lieu of reloading the source deck. A second magnetic tape may also be used as the object-program output medium.

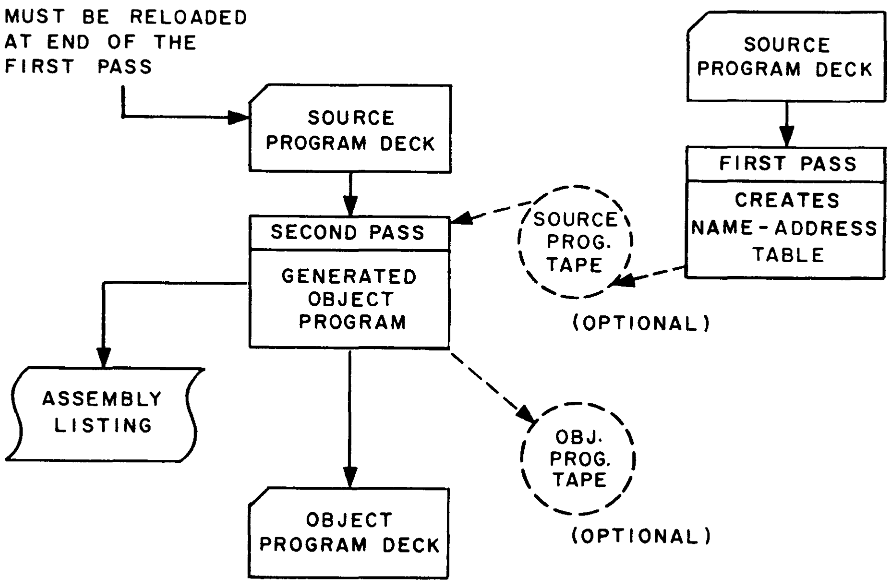


Figure 1. Assembler Operation

Some features of the Assembler are summarized below.

1. *Operation Code Mnemonics* — Each machine instruction is assigned a unique mnemonic operation code as specified in the assembly language. The programmer uses these mnemonics to specify the desired instructions.
2. *Symbolic Addressing* — Every memory location is available for assignment as a symbolic name for program reference. This allows

FUNCTIONAL DESCRIPTION *(Cont'd)*

reference to branch points, tables, constants, and storage areas without requiring knowledge of absolute memory addresses. The absolute memory addresses are assigned by the Assembler.

3. *Expressions* — Provide the ability to combine symbols and numeric values to form desired addresses.
4. *External References* — Enable a program to refer to data or control information outside its boundaries. This is provided by the use of a unique program symbolic name that represents the desired reference. Thus, separately assembled programs may be linked together at execution time by using the Relocatable Loader routine. The Binder routine can also be used to bind these separately assembled programs into a single program which can be loaded by any Absolute Loader.

All input/output operations are controlled by the 70/15 Input/Output Control System (IOCS). IOCS may be assembled with the program, linked in the binding run, or linked with the program object time.

The 70/15 Assembler is described in detail in the 70/15 Assembly System Manual, No. 70-15-602.

INPUT/OUTPUT DESCRIPTION

Input

◆ The input to the Assembler consists of source program card decks. Each card is composed of the following fields:

Fields	*Columns*
Operation	1–4
Operand and Comments	12–16
Identification — Sequence	73–80

Name Field — The Name field identifies a particular statement. Any other statement can refer to the statement by using that name. The Name field is represented by a symbolic expression.

Operation Field — A machine or Assembler mnemonic instruction is written in the Operation field. Whenever an invalid mnemonic is specified the Assembler generates a halt and an unconditional branch to the next instruction. This field may also be used to specify Assembler control instructions (e.g., START, END, ENTRY, and EXTRN) that supplement the machine instructions.

Operand and Comments Field — The Operand field defines the locations, data, or devices that are used by the Operation field. The field is divided into operands and the number of operands depends on the instruction format. Commas are used to separate a string of operands. Descriptive information can also be written in this field. At least one column to the right of the last operand must be skipped before writing any comments. In addition, the entire statement line can be written as comments when an asterisk (*) is written in column 1.

Identification — Sequence Field — This field is used for both program identification and statement sequencing.

Output

◆ The output of the Assembler consists of:

Object-program Output
Assembly Listing

Object-Program Output

◆ The object-program output consists of five types of cards.

Program Card — This card is the first card of every program and contains the name of the first source statement. It also contains the address of where the program is to be loaded. This card corresponds to the START source statement.

Text Cards — These cards contain the generated object coding.

ENTRY Cards — These cards correspond to the source ENTRY statements. They contain the name and address of the program entry points.

EXTRN Cards — These cards correspond to the source EXTRN statements. They contain the name and address of the last program reference to the external name.

END Card — This card corresponds to the source END card. It contains an address of the first logical instruction in the program.

Assembly Listings

◆ The Assembly listings consist of the source program statements plus the generated object coding. The fields are as follows:

Fields	*Columns*
Program Name and Error Flags	1–4
Location Counter	7–10
Generated Object Coding	13–17
Source Statements	33–112
Object Deck Reference Number	117–120

EQUIPMENT REQUIREMENTS

◆ The Assembler requires the equipment listed below and also makes use of additional magnetic tapes and an extra 4K bytes of memory if they are available. Additional tapes can be used as substitutes for the Assembler program, source input, object-program listing, and object-program deck in the Tape Assembly System.

Minimum Equipment

◆ Processor:	(70/15 A* or B**)
Card Reader:	(70/237 or 70/251 with Card Read Feature)
Card Punch:	(70/234 or 70/236)
Printer:	(70/242, 70/243, or 70/248)

Optional Equipment

◆ Magnetic Tape Device: (70/432, 70/442, or 70/445)**

MEMORY REQUIREMENTS

◆ The Assembler requires 4K bytes of memory. Depending on whether 4K or 8K bytes of memory are available, the number of names in a program can vary from 90 names for 4K to 700 names for 8K in the Card Assembler. The Tape Assembler permits a maximum of 550 names. The memory map of the Card Assembler is as follows:

Bytes	Content
0–899	Standard Memory, Loader, and IOCS areas
900–3499	Assembler coding, constants, and working storage
3500–Top of Memory	Name table

* Card Assembly System.

** Tape Assembly System.

MEMORY REQUIREMENTS *(Cont'd)*

The memory map of the Tape Assembler is as follows:

Bytes	Content
0–1499	Standard Memory, Loader, and IOC area
1500–4699	Assembler coding, constants, and working storage
4700–8192	Name table

RELATED PROGRAMMING COMPONENTS

Loader Routines — It is necessary for the Assembler to be loaded into memory by way of one of the following standard 70/15 Loaders:

Absolute Card Loader
Absolute PLT Loader

IOCS — The Assembler program uses the Input/Output Control System incorporated within the Assembler.

ACCURACY CONTROL

◆ The Asssembler observes the standard set of 70/15 program halts and accuracy controls. In addition, the Assembler performs appropriate error checking of source programs with associated warning flags.

TIMING

◆ The time to assemble an average size program of 500 statements is approximately 3.5 to 4 minutes. This timing estimate is based on a 70/15 system consisting of the following equipment:

70/15 Processor
70/237 Card Reader — (source-program input)
70/234 Card Punch — (object-program output)
70/242 Printer
70/432 Tape Units

INPUT/OUTPUT CONTROL SYSTEM (IOCS)

FUNCTIONAL DESCRIPTION

◆ The RCA 70/15 Input/Output Control System (IOCS) consists of a set of routines that facilitates the use of peripheral devices within the Spectra 70/15 System. The IOCS represents an integrated network of read, write, and control functions that relieve the programmer of substantial input/output programming. The system is also capable of error detection and recovery when such actions are appropriate and possible. Simultaneous processing capabilities are an additional aspect of the IOCS. When specified by the programmer, full advantage is taken of the Read Auxiliary instruction and the buffered output devices to provide this facility. For example, the functions of card reading, card punching, and printing may be executed concurrently with computing.

In order to effectively use the memory of the 70/15 Processor, the IOCS is provided in two versions. The only difference between the two versions is that one can control magnetic tape equipment while the second cannot. Both versions control other 70/15 peripheral devices. The nontape version requires less memory than the tape version. In addition, two more versions (tape and nontape) are provided for object program compatibility with the 70/25. These versions require slightly more memory than the standard versions.

IOCS may be assembled with the object program, assembled separately and linked in a binding pass, or loaded with the program into memory at "run" time. The system has seven entry points (four for the nontape version), each of which is accessed from a calling sequence in the program. Based on the entry point, calling sequence, and device parameters supplied by the programmer, the IOCS executes the desired function and returns control to the program at a return address specified in the calling sequence. A detailed description of the device parameters, calling sequences, and routine entry points is provided in the 70/15 Assembly System manual, No. 70-15-602. The entry points are defined as follows:

1. IN — The IN calling sequence transfers data from input devices (such as magnetic tape, card reader, or paper tape) into memory.
2. OUT — The OUT calling sequence transfers data from memory to output devices (such as magnetic tape, card punch, printer, or the paper tape punch).
3. CHK — The CHK calling sequence senses and stores status information relative to a particular device. The standard device byte and the sense byte are received, stored, and analyzed.
4. RWD — The RWD calling sequence rewinds magnetic tapes to BT.
5. RWDA — The RWDA calling sequence rewinds and disconnects magnetic tapes.
6. TMRK — The TMRK calling sequence writes a tape mark on 7- or 9-channel magnetic tape.
7. CTRL — The CTRL calling sequence performs control functions such as stacker selection and printer paper advance.

The programmer refers to the peripheral device in his program on a symbolic basis by means of logical device numbers. These are replaced by actual device numbers at object time by the Loader routine.

INPUT/OUTPUT DESCRIPTION

Input

◆ To use the IOCS the programmer is required to:

1. Define a parameter area for each device used by the program.
2. Code the appropriate calling sequence for the I/O function (IN, OUT, etc.) to be executed.
3. Assemble the program and the IOCS.
4. Incorporate I/O Define cards with the assembled program at object time so that the Loader routine can set up a Device Correspondence Table for conversion of logical to actual device numbers.

Device Parameter Area

◆ This storage area contains information required by the IOCS to control the peripheral device. At assembly time one area must be supplied for each device used by the program. This area is shown below and defined in table 1.

Logical Device No.	Simo Indicator	Starting Address	Ending Address	Abnormal Return Address	Alarm Return Address
+0 +1	+2 +3	+4 +5	+6 +7	+8 +9	+10 +11

A-final Address	Standard Device Byte	I/O Sense Byte	Rollback and Error Recovery Area					
			Cyclic Parity Char.	OP Code	Trunk Device	D1 Address	+D2 Address	Write Control Char.
+12 +13	+14	+15	+16	+17	+18	+19 +20	+21 +22	+23

Table 1. Device Parameter Area

Bytes	Description
0 — 1	A constant (00–09) defining the logical device number. An actual device number is assigned at load time by means of an I/O Define card.
2 — 3	A constant specifying simultaneous or nonsimultaneous processing for the device. The leftmost byte is used as a device pending indicator.
4 — 5	An initial address of the I/O storage area.
6 — 7	A terminal address of the I/O storage area.
8 — 9	An address of the subroutine in the program to which control is to be transferred when certain abnormal conditions are encountered.
10 — 11	An initial address of the subroutine in the program to which control is to be transferred when an alarm condition is encountered.
12 — 15	Area where A-final address, standard device byte, and I/O sense byte are to be stored upon device termination.
16 — 23	Area where a previously issued I/O instruction for magnetic-tape rollback and error recovery is to be stored. The programmer is not required to provide this area when using the nontape version of the IOCS. This area is also used to store a cyclic parity character and a control byte for Write Control instructions.

Calling Sequences

◆ To execute an IN, OUT, RWD, RWDA, TMRK, or CHK function the programmer must move the appropriate address of the device parameter area and his return address to the standard area $P, which is used for communication between a program and subroutine. Subsequent to this, an unconditional branch to the appropriate IOCS entry point must be performed.

To execute a CTRL function the programmer must also include in the calling sequence the hexadecimal representation for the desired control function.

Device Correspondence Table (DCT)

◆ The purpose of this table is to store logical device numbers with their actual device numbers. The DCT is created by the loader according to the information on the I/O Define cards. The loader has a limit of 10 entries (00-09). The DCT contains three bytes containing the following information for each device:

Table 2. Device Correspondence Table (DCT)

Byte 1		Byte 2		Byte 3
4 Bits	4 Bits	4 Bits	4 Bits	8 Bits*
Trunk No.	Device No.	Alternate Trunk No.	Device Type	Control Information

where: Trunk No. is the number of the desired trunk (0-5).
Device No. is the number of the device desired (0-F).
Device Type may be any of the following:
1 = Magnetic Tape
2 = Card Reader or Videoscan Document Reader
3 = Card Punch
4 = Paper Tape Reader
5 = Paper Tape Punch
6 = Printer or Bill Feed Printer
7 = Input/Output Typewriter
8 = Card Punch with Reader Feature

Output

◆ When the IOCS is entered, the following occurs:

1. The entry corresponding to the logical device number is retrieved from the DCT.
2. The trunk corresponding to this device is checked and, if necessary, serviced and stored. This includes posting of the A-final address, standard device byte, and sense byte to the device parameter area.
3. The device pending indicator in the parameter area is checked to see if the last reference to this device was serviced. If necessary, the standard device byte and sense byte are checked. When other than a normal condition is detected, an attempt is made to re-execute the instruction or a return is made to the alarm or abnormal return address, whichever is appropriate.

* Control information of byte 3 refers to packing density and is only applicable to 7-channel magnetic tape.

Output
(Cont'd)

4. If neither the device pending indicator nor an abnormal or alarm condition is detected, the IOCS determines if a CHK is requested. If it is a CHK, control is transferred to the normal return address of the program. If it is not a CHK entry, control is given to the issue portion of the IOCS for further processing.
5. The requested I/O operation is performed.
6. If nonsimultaneous processing was specified, a CHK is performed.
7. Control is then returned to the program.

EQUIPMENT REQUIREMENTS

Minimum Equipment

◆ Processor: (70/15 A or B)

Optional Equipment

◆ Magnetic Tape Device:	(70/432, 70/442, or 70/445)
Card Reader:	(70/237)
Document Reader:	(70/251 — Demand Feed Only)
Card Punch:	(70/234 or 70/236)
Printer:	(70/242 or 70/243)
Bill Feed Printer:	(70/248 — Continuous Forms Only)
Input/Output Typewriter:	(70/216)
Paper Tape Reader/Punch:	(70/221)

MEMORY REQUIREMENTS

◆ The tape IOCS requires approximately 950 bytes of memory. The nontape version requires approximately 450 bytes. This does not include the parameter areas defined by the program. The 70/25 compatibility versions require an additional 32 or 82 bytes for the nontape and tape versions, respectively.

RELATED PROGRAMMING COMPONENTS

◆ The 70/15 Assembler is used to assemble the IOCS. In order to load the assembled IOCS and establish a Device Correspondence Table, any of the 70/15 tape or card loaders may be utilized. The 70/15 Binder routine may be used to bind the assembled IOCS with assembled decks.

ACCURACY CONTROL

◆ When an inoperable-condition code setting is detected following the attempted execution of an I/O instruction, the IOCS stores the trunk and device numbers in standard location, \$P + 5, displays an $(8F)_{16}$ in the M register, and halts. When the IOCS detects an alarm or abnormal condition, it transfers control to either the alarm or abnormal address.

If a parity error is encountered when reading or writing magnetic tape, the IOCS re-reads or re-writes that portion of the tape 10 times. If after 10 times the error is not corrected, a branch to the alarm address takes place.

Any time the IOCS transfers control to either the alarm or abnormal address, the address of the parameter area of the device causing the return and the normal return address are retained in standard location \$P.

TIMING

◆ The approximate time required to initiate an I/O function by way of the IOCS is 0.3 millisecond.

REPORT PROGRAM GENERATOR (RPG)

FUNCTIONAL DESCRIPTION

◆ The Report Program Generator (RPG) produces an object report program from a procedure-oriented source language. Common report features such as input-data selection, editing, calculating, summarizing, and control breaks are provided by the generator.

The source program is the input to the Report Program Generator. This input describes information concerning the input-data format, operations to be performed on the data, and the output format of the report. The generator interprets this information and generates the machine coding required to perform the requested functions.

Some of the features of the report program produced by the generator are as follows:

1. A procedure-oriented language with columnar format for ease of use.
2. Output listing showing source, object coding generated, and errors.
3. A data description section for describing input fields.
4. Reports that will process fixed-size records in variable or fixed-size batches or unbatched variable-size records.
5. Up to nine control breaks.
6. Variable heading information.
7. Variable spacing between lines of print.
8. Data selection and arithmetic calculations.
9. Truncation and rounding of data.
10. Any number of records may be combined to form one print line and vice versa.
11. Actual machine code (own code) may be interspersed in the source program.
12. Input data fields may be split.
13. Editing by a mask.
14. Totals printed at any given line.
15. Headings printed at top of page.
16. Multireel magnetic tape file.

The RPG is organized into two passes. The source program is passed once and is interpreted and processed in the first pass. The first pass consists of six phases, each of which deal with a different section of the source program. The functions are as follows:

Phase	Function
1	Interprets Environment Division information.
2	Generates input/output calls for routines required to process all files needed by the report program.
3	Interprets data descriptions described in the Data Division.
4	Interprets format descriptions described in the Data Division.
5	Interprets all statements written in the Procedure Division.
6	Generates object code.

The second pass of the RPG binds together the generated object code and other components of the 70/15 Programming System such as IOCS into a standard 70/15 object program.

INPUT/OUTPUT DESCRIPTION

Input

◆ Input to the RPG consists of source language cards. The source language is composed of three divisions described below.

Environment Division

◆ The Environment Division contains information such as program name, label procedures and identification, input and output media of the generator, and records and batching configurations. Also included is the report program output information such as size of report and number of lines per page spacing.

Data Division

◆ The Data Division contains a description of the input data to the report program, and the working storages and constants that the report program will use.

Procedure Division

◆ The Procedure Division contains statements of the operations to be performed on the input data and the output commands to be executed.

Output

◆ The output of the RPG consists of an object program of standard 70/15 load cards and a listing of source-language statements and generated coding.

EQUIPMENT REQUIREMENTS

Minimum Equipment

◆ Processor:	(70/15 A)
Card Reader*:	(70/237 or 70/251 with Card Read Feature)
Card Punch*:	(70/234 or 70/236)
Printer*:	(70/242, 70/243, or 248)

Optional Equipment

◆ Processor:	(70/15 B)
Magnetic Tape Device:	(70/432, 70/442, or 70/445)
Paper Tape Reader:	(70/221 or 70/222)

MEMORY REQUIREMENTS

◆ The RPG can use all of the memory that is available, whether it be 4K or 8K. The RPG provides for a maximum of 80 tags for a 4K processor and 400 tags for an 8K processor.

RELATED PROGRAMMING COMPONENTS

◆ The RPG and any generated object program may be loaded into memory by way of any of the standard 70/15 loaders except the 4K RPG card version which includes a special loader.

ACCURACY CONTROL

◆ The RPG observes the standard set of 70/15 program halts and accuracy controls. In addition, all erroneous source-language statements are flagged on the program listing.

TIMING

◆ The appropriate compiling time for a card system will be 1 minute for each 100 statements. This timing estimate is based on the following equipment complement:

Model 70/15 A Processor	Model 70/236 Card Punch
Model 70/237 Card Reader	Model 70/243-1 Printer

* Magnetic Tape is an acceptable substitute for the Card Reader, Card Punch or Printer when the total system complex includes a Magnetic Tape device.

LOADERS

FUNCTIONAL DESCRIPTION

◆ The 70/15 Programming System Loader routines accept object programs from cards or magnetic tape and load them into memory. In addition these routines also provide for:

1. Linking common references within subprograms during the loading of programs for execution.
2. Calling in program overlay segments.
3. Transferring to the starting location of the program after it has been loaded.
4. Assigning actual devices to the logical devices specified by the programmer.
5. Executing instructions outside the program area during the loading process.

The 70/15 system has five loading routines that provide the programmer with complete flexibility of operation. These routines are described below.

Relocatable Card Loader

◆ Loads any program card deck into a predesignated location of memory. This loader is the standard loader for card program processing. It occupies the least amount of memory and performs all of the load functions normally required for tested and bound program decks.

Absolute/Patch Card Loader

◆ In addition to loading program card decks into memory, the Absolute/Patch Loader provides the programmer with the facility to apply program modifications (by way of patch cards) during the loading process.

Absolute Card Loader

◆ Loads into memory any set of programs from a card reader. The address references can be relocated relative from their originally assembled assignments. Also, inter-program references (ENTRY's and EXTRN's) undefined before loading are satisfied through the use of this loader. This loader is used during the loading of relocatable card programs and therefore has patch facilities.

Absolute PLT Loader

◆ Loads into memory any absolute program from the Program Library Tape (PLT). This loader is the standard loader for programs on a PLT and loads 80-character card images from tape. The library search for called programs is facilitated through an SLC/CALL card (see table 3) which calls upon the loader to scan, locate, and load a program from the PLT.

Batched Absolute PLT Loader

◆ Loads into memory from the Program Library Tape (PLT) programs formatted in batched (five-per-block) card images. The reduction of the number of inter-record gaps in a batched program results in programs being loaded in about one-third of the time it takes to load an unbatched program.

INPUT/OUTPUT DESCRIPTION

Input

◆ Input to the loader routines consists of load cards read directly from a card reader or, indirectly, in the form of card images on a Program Library Tape.

Input *(Cont'd)*

Table 3 indicates the 10 load card types and the specific loaders that use them.

Table 3. Load Card Types

Load Card Types	Absolute Card Loader	Absolute Patch Card Loader	Absolute PLT Loader	Absolute Batched PLT Loader	Relocatable Card Loader
I/O DEFINE — Defines execution time for peripheral device linkage.	X	X	X	X	X
SLC/CALL — Identifies programs to be loaded and sets location counter to address of where program is to be loaded.			X	X	X
EXECUTE — Set of instructions to be performed in card image area (punched in EBCDIC).	X	X	X	X	X
PROGRAM — First card of an object program.			X	X	X
ENTRY — Used for external reference linkage.					X
TEXT — Contains the generated object program.	X	X	X	X	X
PATCH — Used for patching programs with certain loaders.		X			X
EXTERNAL — Used for external reference linkage.					X
HEXADECIMAL EXECUTE — Set of instructions to be performed in card image area (punched in hexadecimal).		X			X
END — The last card of a program.	X	X	X	X	X

Figures 2 through 4 show examples of card and tape program composition.

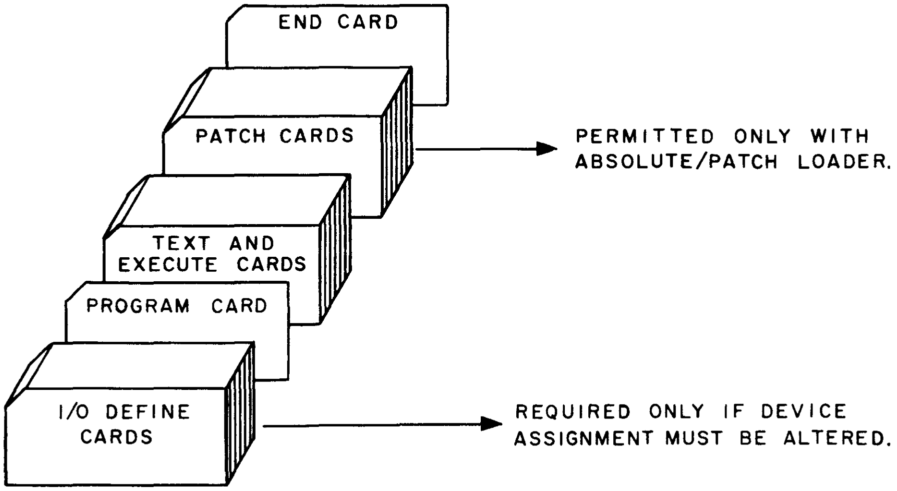

END CARD
PATCH CARDS
PERMITTED ONLY WITH ABSOLUTE/PATCH LOADER.
TEXT AND EXECUTE CARDS
PROGRAM CARD
I/O DEFINE CARDS
REQUIRED ONLY IF DEVICE ASSIGNMENT MUST BE ALTERED.

Figure 2. Object Card Program for Standard Machine Loading

Input
(Cont'd)

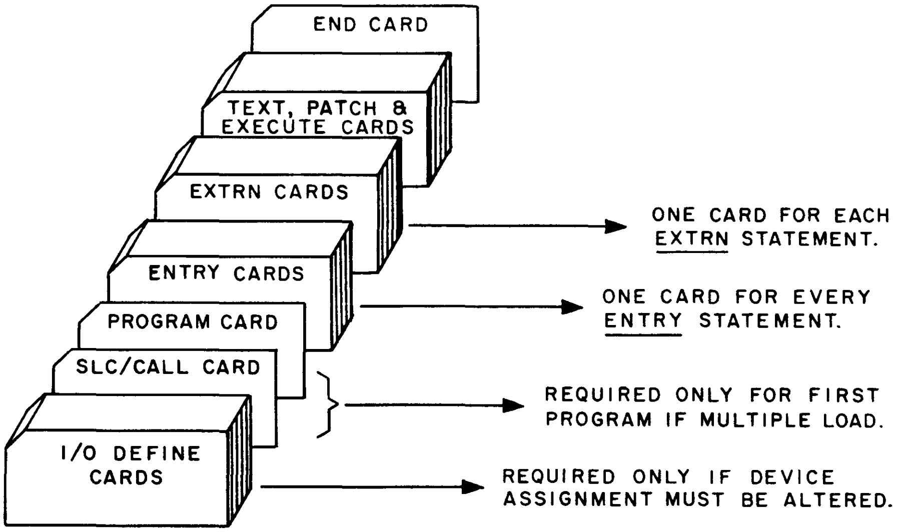

END CARD
TEXT, PATCH & EXECUTE CARDS
EXTRN CARDS
ENTRY CARDS
PROGRAM CARD
SLC/CALL CARD
I/O DEFINE CARDS
ONE CARD FOR EACH EXTRN STATEMENT.
ONE CARD FOR EVERY ENTRY STATEMENT.
REQUIRED ONLY FOR FIRST PROGRAM IF MULTIPLE LOAD.
REQUIRED ONLY IF DEVICE ASSIGNMENT MUST BE ALTERED.

Figure 3. Object Card Program for Relocatable Machine Loading

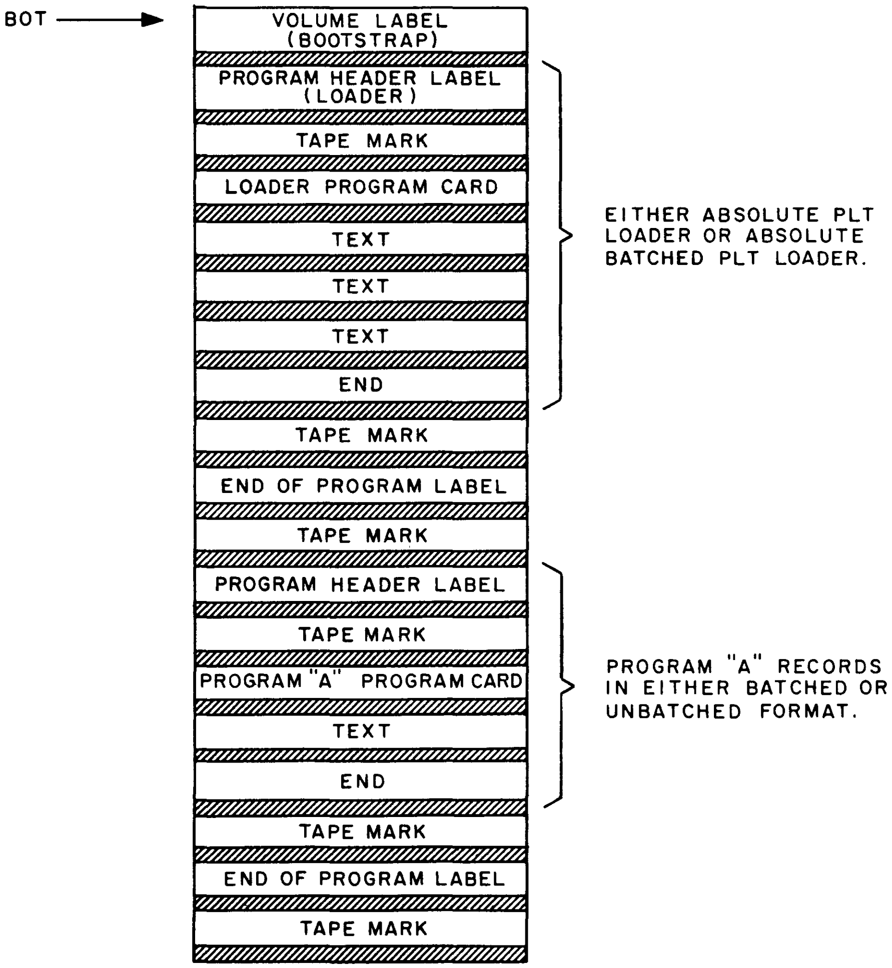


Figure 4. PLT Used by Absolute and Batched Absolute PLT Loaders

Output

◆ The output of all loaders consists of a program loaded into memory

EQUIPMENT REQUIREMENTS

Minimum Equipment

◆ Processor: (70/15 A or B)
Card Reader: (70/237 or 70/251 with Card Read Feature)

Optional Equipment

◆ Magnetic Tapes: (70/432, 70/442, or 70/445)

MEMORY REQUIREMENTS

◆ The approximate memory requirement of each loader is as follows:

Absolute Card — 200 bytes
Absolute/Patch Card — 400 bytes
Relocatable Card — 1300 bytes
Absolute PLT — 600 bytes
Batched Absolute PLT — 700 bytes

The memory map for the loaders is as follows:

```
  Byte 0–199 Reserved
Byte 200–417 (Absolute)
        –613 (Absolute/Patch)
        –787 Absolute/PLT)
        –851 (Batched Absolute PLT)
       –1543 (Relocatable)
ENTRY-EXTRN table (only with Relocatable Loader)
PLT Batch Read Area (only with Batched Absolute PLT Loader)
                    (Top of Memory)
```

RELATED PROGRAMMING COMPONENTS

◆ The loaders operate independently. However, input/output commands within the Assembler are coded with logical device numbers and linked to actual devices only at execution time.

ACCURACY CONTROL

◆ The Loaders observe the standard 70/15 accuracy control procedures.

TIMING

◆ The time to load a program is dependent upon the loading device; however, the loaders operate all devices at their rated speeds.

CARD TO TAPE*

FUNCTIONAL DESCRIPTION

◆ The Card-to-Tape routine transcribes information from standard 80-column punched cards to magnetic tape. The output tape file is in standard 70/15 magnetic tape format, batched or unbatched, labeled or unlabeled. This routine also checks standard tape labels before writing information. Multifile and multivolume files can be produced. Two output tapes may be assigned to this routine to facilitate tape swapping of the output.

Optional input may consist of variable-length paper tape records of 80 characters or less. This routine expands with spaces any paper tape record less than 80 characters to a full 80-character record before transcription to magnetic tape.

Five general parameters, described below, are recognized by this routine. These parameters determine the tape label form and content, and the output-batching factor for the output data blocks.

Own-coding written by the programmer can be conveniently incorporated into the routine through the use of the Binder routine. Certain symbolic entry points in the routine are available to the programmer. These entry points are as follows:

CTCD — most significant character of card image area.

CTRT — most significant character of instruction following branch to the program.

CTUT — most significant character of stored trailer label of the program.

CTFL — most significant character of stored file label.

The above entry points provide access to the card image and label areas to facilitate nonstandard label processing and input data editing. When own-code is bound with the Card-to-Tape routine, control is given to the programmer immediately following each physical read.

INPUT/OUTPUT DESCRIPTION

Input

◆ Input to the Card-to-Tape routine consists of the following five parameter cards and the programmer's data cards:

Volume Parameter — Directs the routine to either generate the volume label, or not to generate the volume label, *or* to retain present volume label. This parameter must be the first record.

File Parameter — Directs the routine to generate standard 70/15 Header and End-of-File labels or to generate nonlabeled files. This parameter defines the start of a file and also contains the data batching factor.

Header Parameter (Optional) — Specifies the Header label for the program.

Trailer Parameter (Optional) — Specifies the Trailer label for the program.

End of Data Parameter — Defines the end of the file and the end of data. This parameter is the last record of the transcription deck. Figure 5 shows the composition of an input deck.

* For co-sharing operation, see Utility Routine Co-Sharing Description, page 1.

Output

◆ The output of this routine consists of data blocked on tape according to the input parameters.

EQUIPMENT REQUIREMENTS

Minimum Equipment

◆ Processor:	(70/15 A or B)
Card Reader:	(70/237 or 70/251 with Punch Card Read Feature)
Magnetic Tape Device:	(70/432, 70/442, or 70/445)

Optional Equipment

◆ Paper Tape Reader/Punch: (70/221)

MEMORY REQUIREMENTS

◆ This routine requires approximately 2,400 bytes of memory, excluding the input area.

RELATED PROGRAMMING COMPONENTS

◆ The Card-to-Tape routine must be loaded into memory by way of one of the following 70/15 loaders:

Absolute Loader
Absolute PLT Loader

ACCURACY CONTROL

◆ This routine performs error checking of input parameter records and observes the standard 70/15 program halts and accuracy controls.

TIMING

◆ Approximately 4.5 to 5 minutes are required to process an input of 2,000 cards from a 70/237 Card Reader, using a 70/432 Tape Unit as the output device.

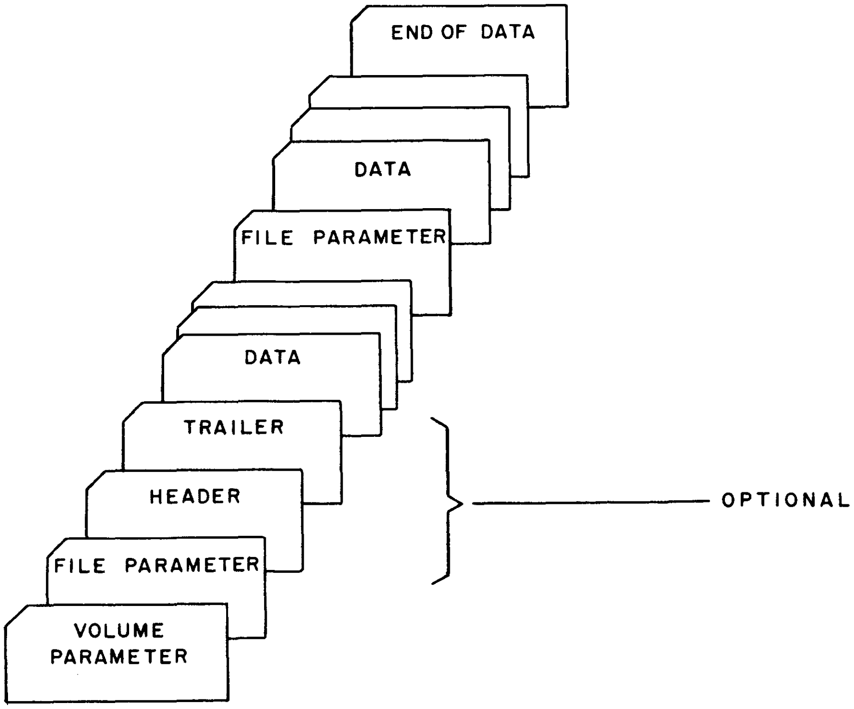


Figure 5. Composition of an Input Deck for Card-To-Tape Routine

TAPE TO PUNCH*

FUNCTIONAL DESCRIPTION

◆ This routine transcribes information from a standard 70/15 labeled magnetic tape file to standard 80-column punched cards. The input tape file contains fixed-length 80-character records which may be batched. Optionally, the routine can accept the following input media when magnetic tapes are not used:

1. Standard 80-column punched cards.
2. Fixed-length, 80-character paper tape records.
3. The magnetic tape file may be multifile or multivolume. Standard 70/15 tape labels can be optionally chosen to be punched. Also, own-coding can be conveniently incorporated into the routine through the use of the Binder routine.

Two symbolic entry points in the routine are available for access to the card punch area:

CPCD — most significant character of card punch area.
CPRT — most significant character of instruction following branch to the program.

Transfer to own-code is performed after each card (including labels) is ready to be punched. In order for own-coding to check labels, the parameter must request that labels be reproduced in the output card deck.

INPUT/OUTPUT DESCRIPTION

Input

◆ Input consists of a program parameter card and card-image data from either card decks or tape files.

Output

◆ The output of this routine consists of card records punched in EBCDIC card code. No tape block input smaller than 80 characters will be reproduced on the output.

EQUIPMENT REQUIREMENTS

Minimum Equipment

◆ Processor:	(70/15 A or B)
Card Reader:	(70/237 or 70/251 with Punch Card Read Feature)
Magnetic Tape Device:	(70/432, 70/442, or 70/445)

MEMORY REQUIREMENTS

◆ This routine requires approximately 2,000 bytes of memory, excluding the input area.

RELATED PROGRAMMING COMPONENTS

◆ The Tape-to-Punch routine must be loaded into memory by way of one of the following 70/15 Loaders:

Absolute Loader
Absolute PLT Loader

ACCURACY CONTROL

◆ This routine performs error checking of input parameter records and observes the standard 70/15 program halts and accuracy controls.

TIMING

◆ Approximately 7 minutes are required to process an input of 2,000 cards with a 70/232 Card Reader and a 70/236 Card Punch.

* For co-sharing operations, see Utility Routine Co-Sharing Description, page 1.

TAPE TO PRINTER*

FUNCTIONAL DESCRIPTION

◆ This routine produces print listings of a standard, labeled file with batched fixed-length records or unbatched variable-length records, on either multifile volume or multivolume files.

Optionally, the routine can accept the following input media when magnetic tapes are not used:

1. Standard 80-column cards.
2. Variable-length (12 to 161 bytes) paper tape records.
3. A record cannot be processed that is less than 12 bytes or greater than 161 bytes.

As an option, each data record can contain listing control information governing page changing, line spacing, etc. A single parameter card informs the routine as to which file is to be processed and how its records are formatted. Standard 70/15 output tape labels can be optionally generated.

Own-coding can be conveniently incorporated into the routine through the use of the Binder routine. Two symbolic entry points in the routine are available for access to the print image area.

TPRD — most significant character of print record.

TPRT — most significant character of instruction following branch to user.

Control is transferred to the own-code section before the printing of each line.

INPUT/OUTPUT DESCRIPTION

Input

◆ Input to this routine consists of a program parameter card and print data derived from magnetic tape, paper tape, or card decks.

Output

◆ The output of this routine consists of print lines equal to the size of the printer buffer. If control information is supplied with each print record, it will control the spacing and page changing. If no control information is supplied, single spacing is provided and the program will head each page as follows:

File Name   Tape/Card to Printer   Date   Page No.

EQUIPMENT REQUIREMENTS

Minimum Equipment

◆ Processor:	(70/15 A or B)
Card Reader:	(70/237 or 70/251 with Punch Card Read Feature)
Magnetic Tape Device:	(70/432, 70/442, or 70/445)
Printer:	(70/242 or 70/243)

MEMORY REQUIREMENTS

◆ This routine requires approximately 2,000 bytes of memory, excluding the input area.

* For co-sharing operations, see Utility Routine Co-Sharing Description, page 1.

RELATED PROGRAMMING COMPONENTS

◆ The Tape-to-Printer routine must be loaded into memory by way of one of the following 70/15 Loaders:

Absolute Loader
Absolute PLT Loader

ACCURACY CONTROL

◆ This routine performs error checking of input parameter records and observes the standard 70/15 program halts and accuracy controls.

TIMING

◆ Approximately 3.5 to 4 minutes are required to process an input of 2,000 80-character records with a 70/432 Tape Unit and a 70/242 Printer.

SINGLE-PHASE MEMORY DUMP/ SNAPSHOT

FUNCTIONAL DESCRIPTION

◆ The Single-Phase Memory Dump/Snapshot routine is a program testing aid that prints the contents of specified memory areas at defined points in a program cycle and upon termination of a program.

This routine is assembled with, or linked to, the program being tested. The programmer determines the points within his program at which he desires a printer listing of memory, and inserts a calling sequence at these points. When program control transfers to a calling sequence, this routine prints the contents of the memory area defined by the calling sequence and returns control to the program.

If the program comes to a halt unexpectedly, the programmer can activate the Memory Dump routine at the console.

INPUT/OUTPUT DESCRIPTION

Input

◆ Input to this routine consists of:

1. The contents of designated memory locations at specified times.
2. Parameters describing the boundaries of the memory areas to be printed, and the address to which control is to be transferred after the printing is completed. These parameters are stored in the standard $P area by the console operator or by way of the calling sequence shown below.

The normal method of activating this routine is by a calling sequence executed as part of the program. The programmer may insert as many calling sequences as needed. The format for each sequence is as follows:

Name	Operation	Operand	Comments
	MVC	$P(10),*+10	Move parameters to $P area
	B	SNAP	Branch to dump routine
	DC	A(*+10)	Return address
	DC	A(LLLL)	Left-hand address
	DC	A(RRRR)	Right-hand address
	DC	C'NAME'	Dump identifier

If the program comes to an unexpected halt, the operator may activate this routine at the console by inserting the parameter information into the $P area.

Notes

◆ 1. The left- and right-hand addresses may be expressed either as symbolic addresses or hexadecimal addresses.

2. If this routine is not assembled with the program, the symbol SNAP must be defined as an external symbol (EXTRN).

Output

◆ Output from this routine is a printer listing of specified memory areas, delivered directly to the printer or to magnetic tape for subsequent printing.

Each 48-byte group of input data is listed on the printer as two print lines. The first line shows the input as EBCDIC graphics, grouped into twelve 4-byte sets. (An asterisk symbol appears after the fourth and eighth

Output *(Cont'd)*

set to make the displayed data convenient for reading.) The second print line shows the hexadecimal equivalent of the 48 bytes of the first line.

Each pair of print lines is identified by showing the memory location (in hexadecimal) of the leftmost byte.

SAMPLE OUTPUT

```
NAME  RTN:XXXX  RCA 70/15 MEMORY DUMP  HSM LLLL/RRRR  MM/DD/YY

0000 J O H N  S M I T  H 1 2 3  A N Y S *T R E E  T...   *U S A .....
     D1D6C8D5 E2D4C9E3 C8F1F2F3 C1D5E8EZ E3D9C5C5 E3       E4EZC1

0030 B I L L  J O N E  S 4 5 6   M A I N *A V E N  U E.. *U S A ....
     C2C9D3D3 D1D6D5C5 E2F4F5F6  E4C0C9D5  C1S5C5D5 E5C5.. * F4E2C1

0060
  .
  .
  .
0FFF
```

where: MM/DD/YY = date as it appears in the standard date area.
XXXX = Return address.
LLLL = LHE of Memory Area that was dumped.
RRRR = RHE of Memory Area that was dumped.
NAME = Any four printable graphics desired by the programmer.

Note: The location counter is incremented by $(48)_{10}$ for each line set of EBCDIC graphics and hexadecimal equivalents.

EQUIPMENT REQUIREMENTS

Minimum Equipment

◆ Processor: (70/15 A or B)
Printer: (70/242, 70/243, or 70/248)
Card Reader: (70/237)

Optional Equipment

◆ A Videoscan Document Reader (70/251) with the punch card read feature may be substituted for the Card Reader.

A Magnetic Tape Device (70/432, 70/442, or 70/445) may be substituted for the printer.

MEMORY REQUIREMENTS

◆ This routine requires approximately 900 bytes.

RELATED PROGRAMMING COMPONENTS

◆ This routine may be loaded into memory using any of the standard 70/15 loaders. The Program Binder routine must be used if this routine is to be bound to the program.

ACCURACY CONTROL

◆ This routine observes the standard set of 70/15 accuracy controls. In addition, a test is made of the memory limits to insure that the right-hand end is equal to or larger than the left-hand end, and that both the left-hand end and the right-hand end are within the size of memory.

TIMING

◆ The speed of the output will be approximately the speed of the on-line printer or magnetic tape station.

DUAL-PHASE MEMORY DUMP/ SNAPSHOT

FUNCTIONAL DESCRIPTION

◆ The Dual-Phase Memory Dump/Snapshot routine is a program testing aid that prints the contents of specified memory areas at defined points in a program cycle and upon termination of a program. It is used whenever the memory requirements of the program do not allow the Single-Phase Memory Dump/Snapshot routine to reside in core concurrently with the program.

This routine consists of two phases. Phase One is assembled with, or linked to, the program being tested. The programmer determines the points within his program at which he desires a printer listing of memory, and inserts a calling sequence at these points. When program control transfers to these points, this routine writes to an output device the contents of the memory area defined by the calling sequence, and returns control to the program.

If the program comes to a halt unexpectedly, the programmer can activate Phase One at the console.

Phase Two, which is executed independently at a later time, edits the output of Phase One and produces a printer listing of the contents of memory as it existed when each calling sequence activated Phase One.

INPUT/OUTPUT DESCRIPTION

Input

◆ Input to Phase One consists of:

1. The contents of designated memory locations at specified times.
2. Parameters describing the boundaries of the memory areas to be printed and the address to which control is to be transferred after the output operation is completed. These parameters are stored in the standard $P area by the console operator or by way of the calling sequence shown below.

The normal method of activating this routine is by a calling sequence executed as part of the program. The programmer may insert as many calling sequences as needed. The format for each sequence is as follows:

Name	Operation	Operand	Comments
	MVC	$P(10),*+10	Move parameters to $P area
	B	SNAP	Branch to dump routine
	DC	A(*+10)	Return address
	DC	A(LLLL)	Left-hand address
	DC	A(RRRR)	Right-hand address
	DC	C'NAME'	Dump identifier

If the program comes to an unexpected halt, the operator may activate this routine at the console by inserting the parameter information into the $P area.

Notes

◆ 1. The left- and right-hand addresses may be expressed either as symbolic addresses or hexadecimal addresses.

2. If Phase One is not assembled with the program, the symbol SNAP must be defined as an external symbol (EXTRN).

Input to Phase Two consists of the output produced by Phase One. Parameters are not required for Phase Two.

Output

◆ Output from Phase One consists of a magnetic tape or a card file, on which is recorded the calling sequence parameters and the unedited contents of memory as it existed when this phase was activated.

Output from Phase Two is a printer listing of the specified memory areas.

Each forty-eight (48) byte group of input data is listed on the printer as two print lines. The first line shows the input as EBCDIC graphics, grouped into twelve 4-byte sets. (An asterisk symbol appears after the fourth and eighth set to make the displayed data convenient for reading.) The second print line shows the hexadecimal equivalent of the 48 bytes of the first line.

Each pair of print lines is identified by showing the memory location (in hexadecimal) of the leftmost byte.

SAMPLE OUTPUT

```
NAME  RTN:XXXX  RCA 70/15 MEMORY DUMP  HSM LLLL/RRRR  MM/DD/YY

0000 J O H N  S M I T  H 1 2 3  A N Y S *T R E E  T...  *U S A.....
     D1D6C8D5 E2D4C9E3 C8F1F2F3 C1D5E8EZ E3D9C5C5 E3     E4EZC1

0030 B I L L  J O N E  S 4 5 6  M A I N *A V E N  U E.. *U S A.....
     C2C9D3D3 D1D6D5C5 E2F4F5F6 E4C0C9D5  C1E5C5D5 E5C5..*E4E2C1

0060
  .
  .
  .
```

where: MM/DD/YY = date as it appears in the standard date area.

XXXX = Return address.

LLLL = LHE of Memory Area that was dumped.

RRRR = RHE of Memory Area that was dumped.

NAME = Any four printable graphics desired by the programmer.

Note: The location counter is incremented by $(48)_{10}$ for each line set of EBCDIC graphics and hexadecimal equivalents.

EQUIPMENT REQUIREMENTS

Minimum Equipment

◆ Processor:	(70/15 A or B)
Magnetic Tape Device:	(70/432, 70/442, or 70/445)
Card Reader:	(70/237)
Printer:	(70/242, 70/243, or 70/248)

Optional Equipment

◆ A Videoscan Document Reader (70/251) with the punch card read feature may be substituted for the Card Reader.

A Card Punch (70/234 or 70/236) may be substituted for the printer.

MEMORY REQUIREMENTS

◆ This routine uses approximately 300 bytes for phase one and 1,300 bytes for phase two.

RELATED PROGRAMMING COMPONENTS

◆ This routine may be loaded into memory using any of the standard 70/15 loaders. The Program Binder routine must be used if phase one is to be bound to the program.

ACCURACY CONTROL

◆ This routine observes the standard set of accuracy controls. In addition, a test is made of the memory limits to ensure that the right-hand end is equal to or larger than the left-hand end, and that both the left-hand end and the right-hand end are within the size of memory.

TIMING

◆ The speed of the output will be approximately the speed of the on-line printer or card punch unit.

TAPE EDIT

FUNCTIONAL DESCRIPTION

◆ The Tape Edit routine provides printer output of selected portions of magnetic tape containing EBCDIC coded information. It will handle multifiles or multivolume files. The routine has been designed as an object-program deck to be run independent of a program. Through the use of preset and selectable options, designated blocks, files, or programs on tape may be printed. The parameters for the Tape Edit are entered by way of the 70/216 Input/Output Typewriter or by a parameter card.

The routine is preset to rewind to BTC and print to end of data (double file mark) ; however, the programmer has other options that can be exercised at run time.

INPUT/OUTPUT DESCRIPTION

Input

◆ The input to the Tape Edit routine takes two forms:

1. The contents of a magnetic tape written in EBCDIC characters.
2. Parameters supplied by the programmer to specify the desired tape print options.

The Tape Edit parameter area contains the following:

1. *Return* — The return address to end-of-job halt.
2. *Option Number* — Option to be executed.
3. *NNN Value* — Number of blocks or files to be printed.

The *options* that can be executed at run time are as follows:

(Preset) 1 = Rewind to BTC and print to double file mark.

2 = Rewind to BTC and print NNN blocks.

3 = Rewind to BTC and print NNN files.

4 = Back space NNN blocks and print to position.

5 = Print next NNN blocks and reposition.

6 = Unwind NNN blocks and print.

7 = Unwind NNN files and print.

The maximum input-data block size is 800 bytes for a 4K memory and 4,800 bytes for an 8K memory. Blocks exceeding the allowable sizes are truncated.

Output

◆ The output of the *Tape Edit* routine is the printed copy of the information contained on a magnetic tape. All tape information is printed in block format. The block number is printed in the left margin beside the first line of each block. The title of the routine, the date, page number, and option are printed at the top of the first page.

Printout Example

```
            RCA 70/15 TAPE EDIT                              DATE
            OPTION NO.______                                PAGE NO.

BBBB CCCC G  G  G  G    G  G  G  G    G  G  G  G    G  G  G  G    ***** G  G  G  G

            HHHHHHHH    HHHHHHHH    HHHHHHHH    HHHHHHHH    ***** HHHHHHHH
```

BBBB — Block number.

CCCC — Character count within block.

GG — EBCDIC graphics.

HHHH — Hexadecimal equivalents.

EQUIPMENT REQUIREMENTS

◆ Processor:	(70/15 A or B)
Printer:	(70/242, 70/243, or 70/248)
Magnetic Tapes:	(70/432, 70/442, or 70/445)
Card Reader:	(70/237 or 70/251 with Card Read Feature)

MEMORY REQUIREMENTS

◆ The Tape Edit routine requires approximately 2,770 bytes of memory, excluding the input area.

RELATED PROGRAMMING COMPONENTS

◆ The Tape Edit routine can be loaded into memory using any of the standard 70/15 loaders.

ACCURACY CONTROL

◆ The Tape Edit observes the standard set of 70/15 accuracy controls. Also, the routine checks to see if the block is greater than the allowable block size, and if so, a flag is set in the print area to indicate truncation.

TIMING

◆ The speed of output will be approximately the speed of the on-line printer.

PROGRAM BINDER

FUNCTIONAL DESCRIPTION

◆ The Program Binder routine binds into a single 70/15 program any set of 70/15 programs derived either from card decks or 70/15 program library tapes. The output can also be either card decks or program library tapes. The address references can be relocated relative to their originally assembled locations. This routine also performs a check for inter-program reference ENTRYs and EXTRNs undefined at assembly time.

The Program Binder routine consolidates a group of programs that would not otherwise fit into memory due to the size of the relocatable loader and the reserved memory of the ENTRY-EXTRN table.

The Binder has three table areas in memory defined as follows:

1. *ENTRY Table* — This table contains the name of the entrance point and its relocated address in the program. The maximum number of ENTRY statements that the table can contain is 40.
2. *EXTRN Table* — This table contains the names of the external program tags and the relocated address of the last reference to them in the program. The maximum number of EXTRN statements that the table can accommodate is 50.
3. *LINK Table* — This table supplies the Binder with information as to where, in the program to be relocated and bound, undefined external references exist. The maximum number of external references that the table can contain is 59.

The following card descriptions indicate how the Binder interprets the various loader card types.

1. *SLC/CALL Card* — If an address is given on the card, it is used as the origin of the next program. If blank, the incoming subprogram is relocated to the base of the previous subprogram.

   If a name is given on the card, the routine looks on the program tape to find a like-named program as the next subprogram to be bound.

   If the card is blank, the next subprogram is found in the card reader. Also, the routine will either retain the present table of ENTRY-EXTRN definitions or erase the table, depending on parameter card information.
2. *Program Card* — On detection of this card, the Binder calculates the float factor of the program by taking the difference between the location counter and the address at which the program originally was assembled. Only the first program card following the bind parameter actuates the generation of a program card in the output.
3. *ENTRY* Card — This card directs the Binder to float the address on the card and to transfer the ENTRY card name and address to the ENTRY table.
4. *Text Card* — The Binder applies the float factor to all the relocatable addresses on this card and then writes it out. External references are also placed in the LINK table.
5. *EXTRN Card* — This card directs the Binder to float the address on the card and add it along with the EXTRN card name to the EXTRN table.

FUNCTIONAL DESCRIPTION *(Cont'd)*

6. *End Card* — This is the last card of an individual program and its detection directs the Binder to process the ENTRY-EXTRN-LINK tables if this is the last program loaded. If not, the next program is processed.
7. *Execute Card* — This card causes the Binder to float all relocatable addresses on the card.

Certain load cards are considered illegal by the Binder. These are the I/O Define, the Hexadecimal Execute, and the Patch load cards. If one appears in the input, the invalid card is displayed on the printer and an error halt occurs.

INPUT/OUTPUT DESCRIPTION

Input

◆ The input to the Binder consists of the Binder parameters and programs on either cards or a program library tape.

The Binder parameters are as follows:

1. *Bind Parameter* — Contains the parameter name identifier, the new name for the bound program, sequence number, increment, and date.
2. *End of Job Parameter* — Signifies end of job.

Input Deck Composition

◆ A set of card programs to be bound are composed according to the diagram shown in figure 6.

Output

◆ The output of the Binder consists of a program in load format either on cards or a program library tape.

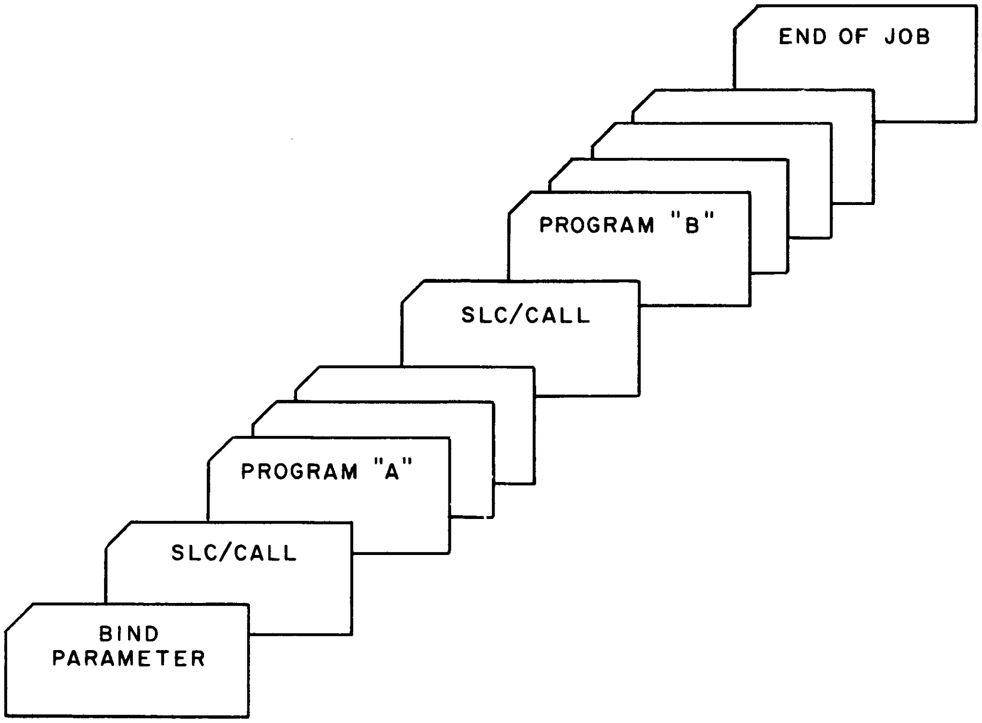


Figure 6. Composition of an Input Deck for Program Binder Routine

EQUIPMENT REQUIREMENTS

Minimum Equipment

◆ Processor:	(70/15 A or B)
Card Reader:	(70/237 or 70/251 with Card Read Feature)
Card Punch:	(70/234 or 70/236)
Printer:	(70/242 or 70/243)

Optional Equipment

◆ Magnetic Tapes: (70/432, 70/442, or 70/445).

MEMORY REQUIREMENTS

◆ The memory requirement for this program is 3,646 bytes.

RELATED PROGRAMMING COMPONENTS

◆ Object programs produced by the Binder routine are nonrelocatable and must be read into memory by means of the Absolute Loader or Absolute/Patch Loader.

ACCURACY CONTROL

◆ This program observes the standard set of 70/15 program halts and accuracy controls. In addition, the Binder flags any errors detected during the program binding process.

TIMING

◆ The time to bind a program consisting of 1,000 cards is approximately 5 minutes. This timing estimate is based on the following peripheral equipment:

70/237 Card Reader (input)
70/236 Card Punch (output)

PLT UPDATE

FUNCTIONAL DESCRIPTION

◆ The PLT Update is a program tape maintenance routine that has the facility to perform insert, replace, delete, and extract operations on a standard 70/15 unlabeled, single-volume, single-file tape. The records on the file must be fixed 80-characters and may be batched. The update options that can be performed are as follows:

1. Extract programs from a PLT to an output tape.
2. Insert programs from card decks or magnetic tape onto an existing PLT.
3. Replace programs on a PLT with programs from card decks or tape.
4. Delete programs from a PLT in the process of generating a new PLT.
5. Modify individual programs through the insertion of text cards which have the effect of patches.

An input deck of parameter cards and text cards directs the routine to perform these update options. Each program to be affected can have only one parameter option executed in any single run. However, many text card inserts can be performed on an individual program within a run.

INPUT/OUTPUT DESCRIPTION

Input

◆ The input to this routine consists of:

1. A standard unlabeled, single-file, single-volume program tape.
2. An optional merge source tape in above format.
3. Update parameter cards.
4. Optional modification text cards.

A description of each update parameter card is as follows:

Extract Program Parameter — Directs the routine to extract programs from the input volume and to copy them to an output volume.

Insert Program Parameter — Directs the routine to copy the named program from the merge source or cards onto the output volume and to copy all other files or programs from the master input source to the output volume.

Replace Program Parameter — Directs the routine to follow the same steps as for the Insert Program Parameter except that a program with the same name is on the input volume and is replaced by the new program.

Delete Program Parameter — Directs the routine to delete the named program in this generation of a new volume.

Modify Program Parameter — Directs the routine to copy all other programs and to process the one named in the parameter. The cards following this card must be text cards, that allow the programmer to replace, insert, delete, or alter records within that program.

End of Job Parameter — Denotes the end of the parameter input.

Output

◆ The output of the PLT Update routine consists of a new master PLT tape. The output master tape is in the same format as the input.

EQUIPMENT REQUIREMENTS

Minimum Equipment

◆ Processor:	(70/15 A or B)
Card Reader:	(70/237 or 70/251 with Card Read Feature)
Magnetic Tapes:	(70/432, 70/442, or 70/445)

MEMORY REQUIREMENTS

◆ The size of the update is approximately 3,200 bytes. This is excluding input/output areas.

RELATED PROGRAMMING COMPONENTS

◆ This routine is loaded into memory by way of one of the following standard 70/15 loaders:

Absolute Loader
Absolute PLT Loader

ACCURACY CONTROL

◆ The PLT Update observes the standard set of 70/15 accuracy controls. In addition, the routine flags any errors that are a result of incorrect parameter records.

TIMING

◆ The time is approximately 1 minute for updating a master program library tape, containing 20 programs, by inserting five new programs from a merge tape. This timing is based on the following peripheral equipment:

70/432 Tape Unit
70/237 Card Reader
70/242 Printer

A sample program update is shown in figure 7.

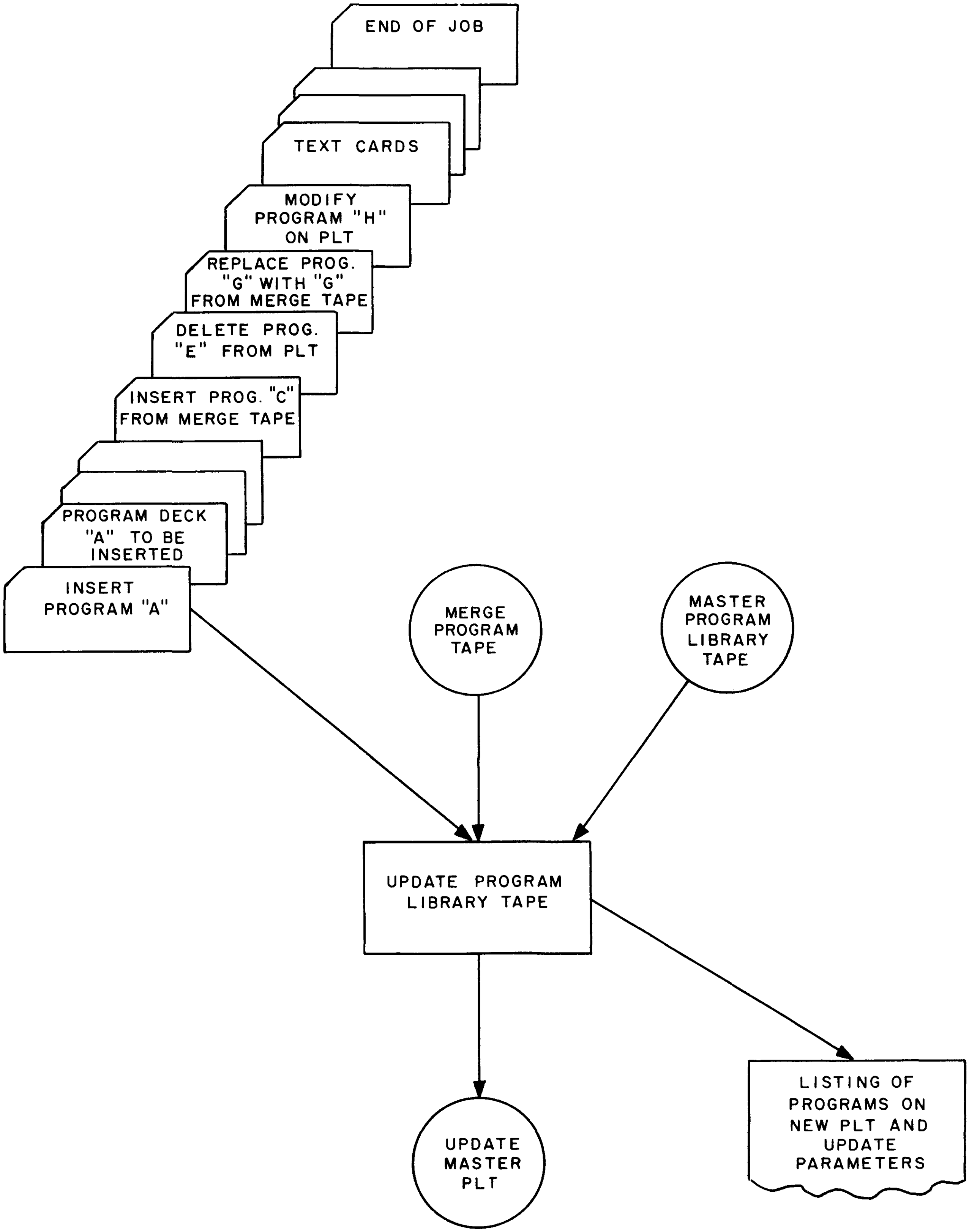

END OF JOB
TEXT CARDS
MODIFY PROGRAM "H" ON PLT
REPLACE PROG. "G" WITH "G" FROM MERGE TAPE
DELETE PROG. "E" FROM PLT
INSERT PROG. "C" FROM MERGE TAPE
PROGRAM DECK "A" TO BE INSERTED
INSERT PROGRAM "A"
MERGE PROGRAM TAPE
MASTER PROGRAM LIBRARY TAPE
UPDATE PROGRAM LIBRARY TAPE
UPDATE MASTER PLT
LISTING OF PROGRAMS ON NEW PLT AND UPDATE PARAMETERS

Figure 7. Sample Program Update

SINGLE-CHANNEL COMMUNICATIONS CONTROL SYSTEM

FUNCTIONAL DESCRIPTION

◆ The 70/15 Single-Channel Communications Control System is a set of routines that facilitate the reception and transmission of data between a Spectra 70/15 Processor, equipped with a 70/652 Communication Control, and another RCA processor, also equipped with the appropriate communication control or buffer.

The basic system consists of a combined Receive/Transmit routine. Optional routines are available and are selectable by the programmer to suit his operational needs. The optional routines are:

1. Automatic Dialing routine
2. Code Translation routine
3. Input/Output Typewriter/DXC routine

The Receive/Transmit routine provides the basic logic for controlling the reception and transmission of data. The receive section of this routine is responsible for:

1. Accepting and recognizing control characters used for coordination and synchronization of the communication control, line, and data sets.
2. Receiving and assembling data messages.
3. Recognizing and indicating error conditions and equipment malfunctions.
4. Acknowledging the receipt of error free messages.
5. Sending an invitation to transmit data to a location that has requested permission to transmit.
6. Sending of the terminate sequence to the transmitter when the programmer desires to end communications and disconnect the line.
7. Transferring control to the appropriate routines at points that require intervention, such as when a complete error free message has been received and assembled.

The transmit section of this routine is responsible for:

1. Transmitting control character sequences for coordination and synchronization.
2. Initiating and transmitting data messages.
3. Recognizing control characters sent by the receiving location in acknowledgement of a valid transmission, invitations to transmit, terminate sequence, etc.
4. Recognizing and indicating error conditions and equipment malfunctions peculiar to the transmit mode.
5. Retransmitting of errored messages.
6. Transferring control to routines at points which require intervention.
7. Transmitting an invitation to transmit to the receiving location when all the messages have been sent.
8. Transmitting the terminate sequence when the transmitter desires to end communication and disconnect the line.

The Automatic Dialing routine is required for systems using the 70/652-26 Communication Control and the Bell Automatic Calling Unit. This routine is responsible for the transmission of the telephone number

FUNCTIONAL DESCRIPTION *(Cont'd)*

to the Automatic Calling Unit. When the line connection is established, this routine transfers control to the transmit routine. If a line connection cannot be established on the first attempt, this routine will retry to call the remote location a specified number of times until either the connection is established or it is determined to abandon the call based on a programmer decision.

The Code Translation routine is optional and provides the capability of translating messages to and from the following codes:

1. RCA 301/3301 code to EBCDIC.
2. EBCDIC to RCA 301/3301 code.
3. ASCII code (7-level) to EBCDIC.
4. EBCDIC to ASCII code (7-level).

Messages are translated to the RCA 301/3301 code and ASCII code prior to transmission. The above codes are translated to EBCDIC as they are received and assembled in the program input area. The translation requirements are specified at assembly time and included in the object program. This routine relieves the programmer of the responsibility of translating messages to and from the transmission line code of the remote location.

The Input/Output Typewriter/DXC routine is optional but is required if either or both of these devices are included in the equipment configuration. This routine must be assembled with the Receive/Transmit routine and provides the necessary logic to determine whether an interrupt was caused by the Input/Output Typewriter, DXC, or the Communications Control. When an interrupt occurs due to a service request from either the Input/Output Typewriter or the DXC, control will be transferred to the programmer's routine responsible for servicing these devices.

INPUT/OUTPUT DESCRIPTION

Input

◆ The input to the Single-Channel Communications Control System has two separate phases. The first phase consists of parameter cards that define entry points to the programmer's routines and control information. This data is supplied once, prior to assembly, and is assembled and included in the object communication control routine. The following data is to be supplied in symbolic form by the parameter cards:

1. Trunk address assigned to the 70/652.
2. Byte configuration assigned to each of the selectable control characters:

   DD1
   DD2
   TERM
   ACK
3. Maximum size of the input area required (number of characters).
4. Symbolic starting location of the input area.
5. Maximum size of the output area required (number of characters).
6. Symbolic starting location of the output area. The input area can be used if desired.

Input *(Cont'd)*

7. Symbolic address of the programmer's routine to receive control when a Request to Transmit has been received.
8. Symbolic address of the programmer's routine to receive control when a complete, error-free message has been received and placed in the program input area.
9. Symbolic address of the programmer's routine to receive control when the transmitting location invites the receiver to transmit.
10. Symbolic address of the programmer's routine to receive control when a Go Ahead signal (DD1) is received.
11. Retry count — the number of times to retransmit a Request to Transmit that has not been acknowledged.
12. Count — the number of times to retransmit a message that is not acknowledged.
13. Symbolic address of the error identifier byte.
14. Symbolic address of the programmer's routine to receive control when an error condition is recognized and the error identifier byte is stored in the designated memory area.
15. Area codes, telephone numbers, and the number of dialing digits for systems with the Auto Call feature.
16. Dummy character to be substituted for untranslatable characters in systems using the code translation option.
17. Trunk address assigned to the DXC if included in the system configuration.
18. Trunk address assigned to the Input/Output Typewriter if included in the system configuration.
19. Symbolic address of the programmer's routine to receive control when a DXC causes an interrupt.
20. Symbolic address of the programmer's routine to receive control when an Input/Output Typewriter causes an interrupt.

Input to the second phase, which is the operating communication routine (object program), is in the form of data messages the programmer desires to transmit. Message length and control characters are defined by the parameters supplied by the programmer at assembly time.

Output

◆ Output from the Single-Channel Communication Control System are the data messages received from a remote processor. The data messages are assembled in the area designated by the programmer.

EQUIPMENT REQUIREMENTS

Minimum Equipment

◆ Processor:	(70/15 A or B)
Communications Control:	(70/652-25)
Card Reader:	(70/237 or 70/251 with Card Read Feature)

Optional Equipment

◆ Communications Control:	(70/652-26)
Data Exchange Control:	(70/627)
Input/Output Typewriter:	(70/216)

Optional Equipment *(Cont'd)*

Other RCA standard peripheral devices may be selected to fit processing requirements. The control and operation of these devices are the responsibility of the program or other RCA control systems which may be selected.

MEMORY REQUIREMENTS

◆ The Single-Channel Communication Control System requires approximately 2,500 bytes of memory. This does not include the programmer's related routines or input/output areas.

RELATED PROGRAMMING COMPONENTS

◆ In order to load the Single-Channel Communication System, any of the standard RCA 70/15 loaders may be used.

ACCURACY CONTROL

◆ The Single-Channel Communication Control System indicates the error conditions which are detected by the Communication Control. Error detection is performed by the Communication Control equipment and an indication of an error is given to the control system by way of the sense byte. Interpretation of the sense byte and the coding of the error identifier byte is a function of the control system. The programmer will be given the error identifier byte in a designated memory area. The programmer's routine, at this point, can determine what effect the error has on his particular system and what action is to be taken as a result of the type of error.

SORT/MERGE GENERATOR

FUNCTIONAL DESCRIPTION

◆ The Sort/Merge Generator produces sort or merge programs based on control statements supplied by the programmer. The generated object sort and merge programs are punched on cards or written on tape in standard load card format.

Object sort programs produced by the Generator enable the programmer to sort files of random records into one sequential file; object merge programs enable the programmer to merge multiple files of sequenced records into one sequential file. Sequencing is performed on as many as twelve keys in an input record. Records can be sorted or merged into ascending or descending sequence, and the programmer can specify an individual ordering sequence for each key.

Own-code facilities allow the programmer to insert, replace, and delete records during sort first pass, sort last pass, and merge processing.

A summary of the features which characterize the object sort and merge programs is listed below:

1. Up to a 7-way sort or merge is provided.
2. Standard Spectra 70 label processing is provided.
3. Input files may be labeled or unlabeled; input records may be fixed or variable in length, blocked or unblocked.
4. Checkpoints are taken at the end of each pass to allow for restarts.
5. Tape alternation may be specified for sort input files, merge input files, and merge output files.
6. Certain input/output work tape duplication is permitted.
7. Own-code exits are provided to allow for additional input and output label processing, as well as sort first pass, sort last pass, and merge record processing.

INPUT/OUTPUT DESCRIPTION

Input

◆ Input to the Sort/Merge Generator consists of control statements which may be followed (optional) by an object deck of own-coding.

Input to a generated sort program consists of one or more reels containing homogeneous records in random sequence. Maximum input block size is 2,048 bytes; maximum input record size is 1,024 bytes. The number of records that can be sorted is determined by the number of records that can be written onto one work tape during string generation, based on the internal sort blocking factor.

Input to a generated merge program consists of two or more files (single-reel or multireel) containing homogeneous records ordered in the same sequence as the desired output sequence. There is no limit on the number of records that can be merged. Maximum input block size is:

$$\frac{3072 - 2S}{W}$$

where: S is the output block size,
W is the way of the merge.

The maximum input record size is 1,024 bytes.

Output

◆ Output from the Sort/Merge Generator is an object sort or merge program in standard load card format. If own-code is specified, the own-coding is bound to the output object program by the Generator.

Output from an object sort or merge program consists of a single, sequenced file containing the sorted or merged records. The maximum output block size for an object sort or merge is:

$$\frac{3072 - (TR)}{2}$$

where: T is the number of work tapes,
R is the record size.

Note: Tape alternation on output is allowed for object merges.

EQUIPMENT REQUIREMENTS*

Sort/Merge Generator

◆ Processor:	(70/15 B)
Card Reader:	(70/237 or 70/251 with Punch Card Read Feature)
3 Magnetic Tape Devices:	(70/432, 70/442, or 70/445)
Card Punch:	(70/234 or 70/236) A magnetic tape device may be substituted if the object sort or merge is to be written to tape.

Object Sorts

◆ Processor:	(70/15 B)
Card Reader:	(70/237 or 70/251 with Punch Card Read Feature)
Printer:	(70/242, 70/243, or 70/248)
3 Magnetic Tape Devices:	(70/432, 70/442, or 70/445) Five additional magnetic tape devices may be utilized to increase the efficiency of the sort.

Object Merges

◆ Processor:	(70/15 B)
Card Reader:	(70/237 or 70/251 with Punch Card Read Feature)
Printer:	(70/242, 70/243, or 70/248)
Magnetic Tape Devices:	As required (For an n-way merge, a minimum of n + 1 magnetic tape devices are required.)

*

MEMORY REQUIREMENTS

◆ The Sort/Merge Generator makes use of the full 8K memory capacity. The following are estimates of the memory requirements of generated sort and merge programs. These estimates include reserved memory, but do not provide for own-code or input/output areas:

Sort first pass	— 4,500 bytes
Sort last pass	— 4,200 bytes
Merge	— 4,700 bytes

ACCURACY CONTROL

◆ Parameter cards, supplied at generation time or at object execution time, are validated for proper format.

TIMING

◆ The Sort/Merge Generator requires approximately 3 to 5 minutes to generate an object sort or merge. Preliminary timing formulas for object sorts may be found in the Spectra 70 Marketing Guide; object merges operate at the rated speed of the magnetic tape devices.

~~* 9 level tapes, or 7 level tapes with the pack-unpack feature.~~

* THE SORT/MERGE GENERATOR REQUIRES A MINIMUM OF ONE 9-LEVEL TAPE; IF A MAG TAPE IS SUBSTITUTED FOR THE CARD PUNCH, THAT TAPE MUST BE 9-LEVEL.
OBJECT SORTS REQUIRE THAT A MINIMUM OF THREE 9-LEVEL TAPES BE ASSIGNED AS WORK TAPES.
INPUT AND OUTPUT FILES PROCESSED BY AN OBJECT MERGE MAY BE 9-LEVEL OR 7-LEVEL TAPE WITH THE PACK/UNPACK FEATURE.

APPENDIX A 70/15 PROGRAMMING SYSTEM STANDARDS

GENERAL

◆ System Standards for the 70/15 adhere to the Spectra 70 System Standards in that data standards, with respect to data formats and conventions, are the same.

Label formats are also the same but label processing in the 70/15 is a subset of the Spectra 70 label processing functions.

This section deals with the standards that are applicable to 70/15 Programming Standards. Compliance with the standards described herein is a necessary requirement for the proper operation of the 70/15 Programming System.

The standard elements are listed as follows:

1. Library Standards
2. Standard HSM Layout
3. Programming Standards
4. Operator/Machine Communication

Library standards have been established with respect to loading programs into memory and program organization. Program organization is oriented basically towards a card library system but program library tape organization is also provided.

Elements of the 70/15 Programming System are designed to use standard HSM locations in order to achieve efficient utilization of memory.

Programming standards for parameters and program tag assignments have been established in order to simplify programming. In the area of operator/machine communication, certain standard halts and error recovery procedures have also been established for ease of operation.

LIBRARY STANDARDS

◆ The organization of programs in the 70/15 is oriented basically towards a card library system. The unit of program loading is the load card, read directly from a card reader or indirectly in the form of card images on magnetic tape.

Loaders

◆ The following Loader routines are available for 70/15 program loading:

Absolute Loader
Absolute/Patch Loader
Relocatable Loader
Absolute PLT Loader
Batched Absolute PLT Loader

Load Card Formats

◆ In the format legends, a "V" in column 1 denotes a loader card while the numeric (0-9) in column 2 denotes the particular type. The use of lowercase type denotes variable-character content and upper-case letters and numerics denote the use of that particular character constant. The load cards and their formats are described below.

I/O Define

◆ This card defines execution-time, peripheral-device linkage and is used with every loader routine when loading a program.

I/O Define (Cont'd)

Card Column	1	2	3	4	5	6	7	8	9	10	11	12	13	80
Legend	V	0	L	x	x	A	t	u	a	d	h	h		Ignored

where: xx — the logical device number to which the actual device is assigned. The range of xx is 00 to 09.

tu — the actual device. The range for the actual device trunk, t, is 0 to 5 and its unit, u, from 0 to F.

L,A — mnemonics denoting logical and actual device, respectively.

a — the alternate trunk (0-5) from which the device may also be accessed.

d — the device type being defined. The permitted content and meaning are as follows:

0 = DXC
1 = Magnetic Tape
2 = Card Reader or Videoscan Document Reader
3 = Card Punch
4 = Paper Tape Reader
5 = Paper Tape Punch
6 = On-Line Printer
7 = Input/Output Typewriter
8 = Card Reader/Punch
9 = Single Channel Communications

hh — the hexadecimal representation of the control byte to be issued to a 7-channel magnetic tape station to set its mode of processing.

The various bit configurations, their meanings, and their hexadecimal representations are shown in table 4.

Table 4. Hexadecimal Representation of Control Byte

Binary	Meaning	Hexadecimal
1111 0000	800, odd, pack/unpack on, translator off	F0
1011 0000	556, odd, pack/unpack on, translator off	B0
0111 0000	200, odd, pack/unpack on, translator off	70
1110 1000	800, odd, pack/unpack off, translator on	E8
1010 1000	556, odd, pack/unpack off, translator on	A8
0110 1000	200, odd, pack/unpack off, translator on	68
1110 0000	800, odd, pack/unpack off, translator off	E0
1010 0000	556, odd, pack/unpack off, translator off	A0
0110 0000	200, odd, pack/unpack off, translator off	60
1100 1000	800, even, pack/unpack off, translator on	C8
1000 1000	556, even, pack/unpack off, translator on	88
0100 1000	200, even, pack/unpack off, translator on	48
1100 0000	800, even, pack/unpack off, translator off	C0
1000 0000	556, even, pack/unpack off, translator off	80
0100 0000	200, even, pack/unpack off, translator off	40

These are the only logical bit configurations for this control byte. A logical combination is defined as one that would not cause an error. Trying to set both odd and even parity would be considered an error and not logical.

If 00 (or blanks) are specified, a 9-level tape station is assumed.

SLC/CALL

◆ This card defines the origin of the program load and identifies the program to be loaded from tape. It is used with the Relocatable Loader, PLT Loader, or the Batched Absolute PLT Loader.

Card Column	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 80
Legend	V	I	n	a	m	e		h	h	h	h	R			Ignored

where: name — the program name to be loaded next from the program source.

hhhh — the hexadecimal address to where the program is to be loaded.

R — A blank implies erasure and generation of a new ENTRY-EXTRN Table. When nonblank signifies that the loader should not erase the ENTRY-EXTRN Table but continue to make use of it.

EXECUTE

◆ This card provides for a set of instructions to be performed in the card image area. It can be used with any of the Loader routines.

Card Column	1	2	3	4	5	6	7	10	11 71	72 80
Legend	V	2	f	f	f	f			instructions and constants	Ignored

where: ffff — four EBCDIC characters representing 32 float factors to be applied to the instructions and constants. Each two bytes of text correspond to a float bit. If the bit is 1 the float factor will be added to the two bytes before execution. The most-significant bit corresponds to the most-significant pair of bytes.

Program

◆ This card is the first card of an object program. Although it appears in every program, it is only used by the PLT Loaders and the Relocatable Loader.

Card Column	1	2	3	4	5	6	7	8	9	10	11 21	22 80
Legend	V	3	n	a	m	e	a	a	b	b	date	Ignored

where: name — the program name.

aa — the EBCDIC address to where the program is assembled.

bb — the EBCDIC address of the highest byte + 1 of this program.

date — the contents of the date area of memory at assembly time.

ENTRY

◆ This card is used for external reference linkage. It corresponds to the ENTRY source statement and is used only by the Relocatable Loader.

Card Column	1	2	3	4	5	6	7	8	9 80
Legend	V	4	n	a	m	e	a	a	Ignored

where: name — the name of the ENTRY point being defined.

aa — the EBCDIC address where it is assembled in the program.

Text

◆ This card contains the generated object program and is processed by all loaders.

Card Column	1	2	3	4	5	6	7	8	9	10	11 72	73 80
Legend	V	5	f	f	f	f	n	a	a		text	Ignored

where: ffff — four EBCDIC characters corresponding to 32 float bit factors to be applied to the text. Each two bytes of text correspond to a float bit. If the bit is 1, the float factor is added to the two bytes before transferring them to memory. The most-significant bit corresponds to the most-significant pair of bytes to be assembled to an even memory location. If the first byte of text is assembled to an odd location, it is not floated.

n — is the EBCDIC number (1-62) of bytes of text to be transferred.

aa — the address to where the first byte of text is assembled.

PATCH

◆ This card is used basically for patching programs and is used with the Absolute/Patch Loader and the Relocatable Loader.

Card Column	1	2	3	4	5	6	7 70	71	72	73 80
Legend	V	6	h	h	h	h	patch		N	Ignored

where: hhhh — the hexadecimal address to where the patch area was originally assembled.

patch — a hexadecimal representation of program text. An EBCDIC "R" punched in the patch text will mean that the float factor is to be added after conversion to the four hexadecimal characters preceding it. The patch is terminated with the first blank.

N — when present, the address in columns 3-6 is not floated.

EXTRN

◆ This card is used for external reference linkage. It corresponds to the EXTRN source statement and is processed by the Relocatable Loader.

Card Column	1	2	3	4	5	6	7	8	9 80
Legend	V	7	n	a	m	e	a	a	Ignored

where: name — the name of the external reference.

aa — the EBCDIC address of the assembled location of the last reference to external name in the program.

Hexadecimal Execute

◆ This card performs the same function as the Execute Card but is normally used for executes with Patch or Relocatable Loaders.

Card Column	1	2	3 6	7 70	71 80
Legend	V	8		hex execute	Ignored

where: hex execute — the hexadecimal representation of the execute. An EBCDIC "R" punched in the hexadecimal

Hexadecimal Execute (Cont'd)

execute field will mean that the float factor is to be added after conversion to the four hexadecimal characters preceding it. The execute is terminated with the first blank.

END

◆ This card is the last card of an object program and is processed by all loaders.

Card Column	1	2	3	4	5	6	7	8	9	80
Legend	V	9	n	a	m	e	a	a	Ignored	

where: name — the name of the entry point to which the Relocatable Loader is to branch.

aa — the EBCDIC address of the assembled location to which the loader is to branch. Blanks in this field direct the Absolute and Absolute/Patch Loaders to issue a read of the parameter source.

Program Organization

Cards

◆ Card program decks ready for machine loading can be either an absolute card-deck load or a relocatable card-deck load. (See figures 8 and 9.)

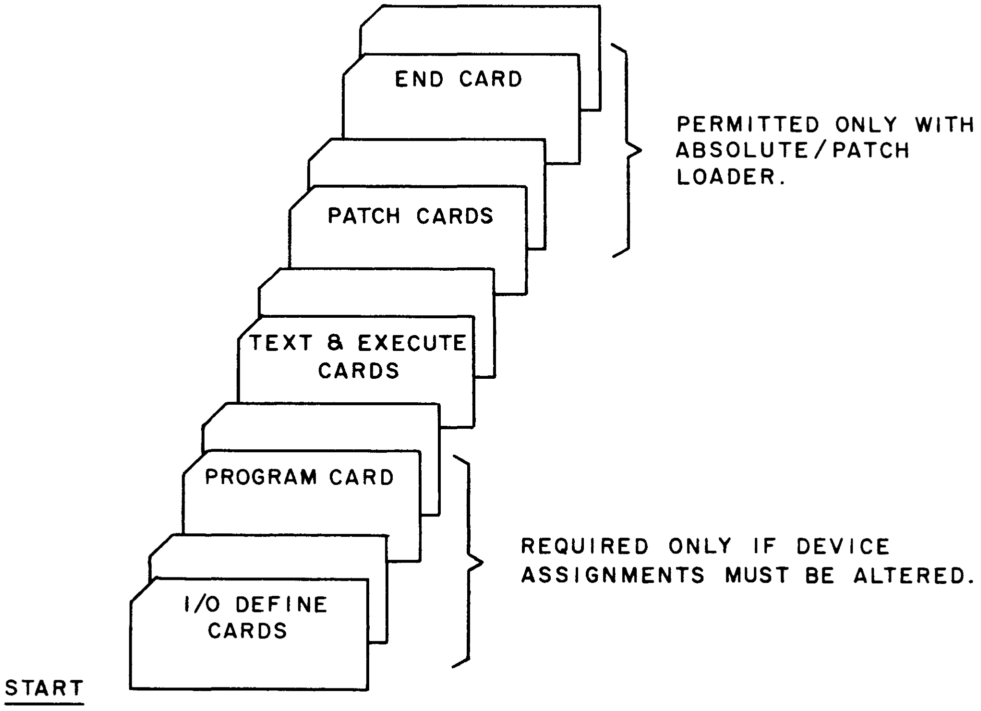

END CARD
PERMITTED ONLY WITH ABSOLUTE/PATCH LOADER.
PATCH CARDS
TEXT & EXECUTE CARDS
PROGRAM CARD
REQUIRED ONLY IF DEVICE ASSIGNMENTS MUST BE ALTERED.
I/O DEFINE CARDS
START

Figure 8. Absolute Card-Deck Load

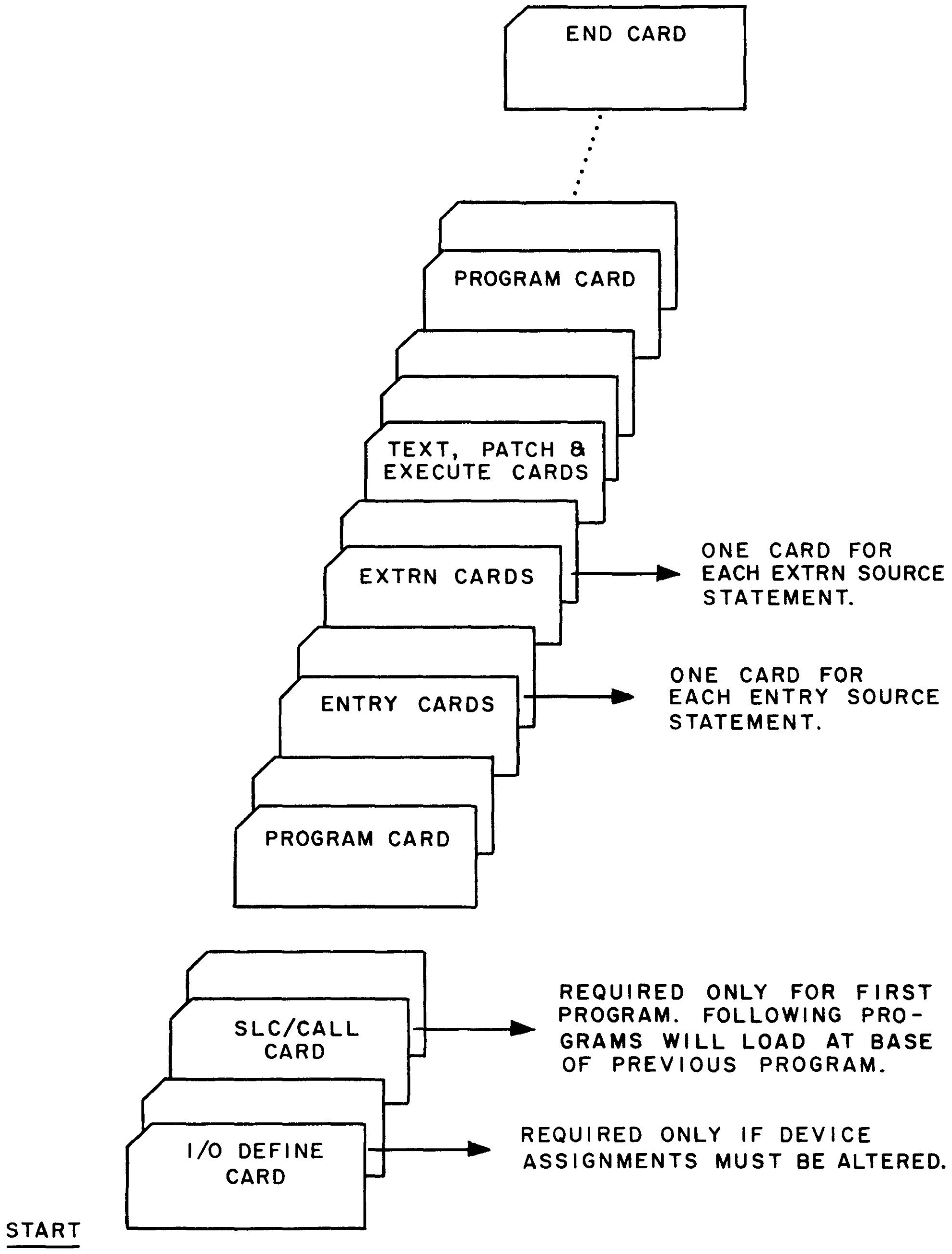

END CARD
PROGRAM CARD
TEXT, PATCH &
EXECUTE CARDS
EXTRN CARDS
ONE CARD FOR EACH EXTRN SOURCE STATEMENT.
ENTRY CARDS
ONE CARD FOR EACH ENTRY SOURCE STATEMENT.
PROGRAM CARD
SLC/CALL CARD
REQUIRED ONLY FOR FIRST PROGRAM. FOLLOWING PROGRAMS WILL LOAD AT BASE OF PREVIOUS PROGRAM.
I/O DEFINE CARD
REQUIRED ONLY IF DEVICE ASSIGNMENTS MUST BE ALTERED.
START

Figure 9. Relocatable Card-Deck Load

Program Library Tape

◆ Programs on a Program Library Tape (PLT) are in the form of 80-column card images on tape. These card images can be batched (five per batch) or nonbatched.

The PLT is designed basically the same as any standard 70/15 single volume, unlabeled file and all programs must be in ascending alphanumeric sequence.

For loading of programs from the PLT, the Absolute PLT Loader or Batched Absolute PLT Loader must appear as the first program on the tape.

Figure 10 shows the Program Library Tape Load Format.

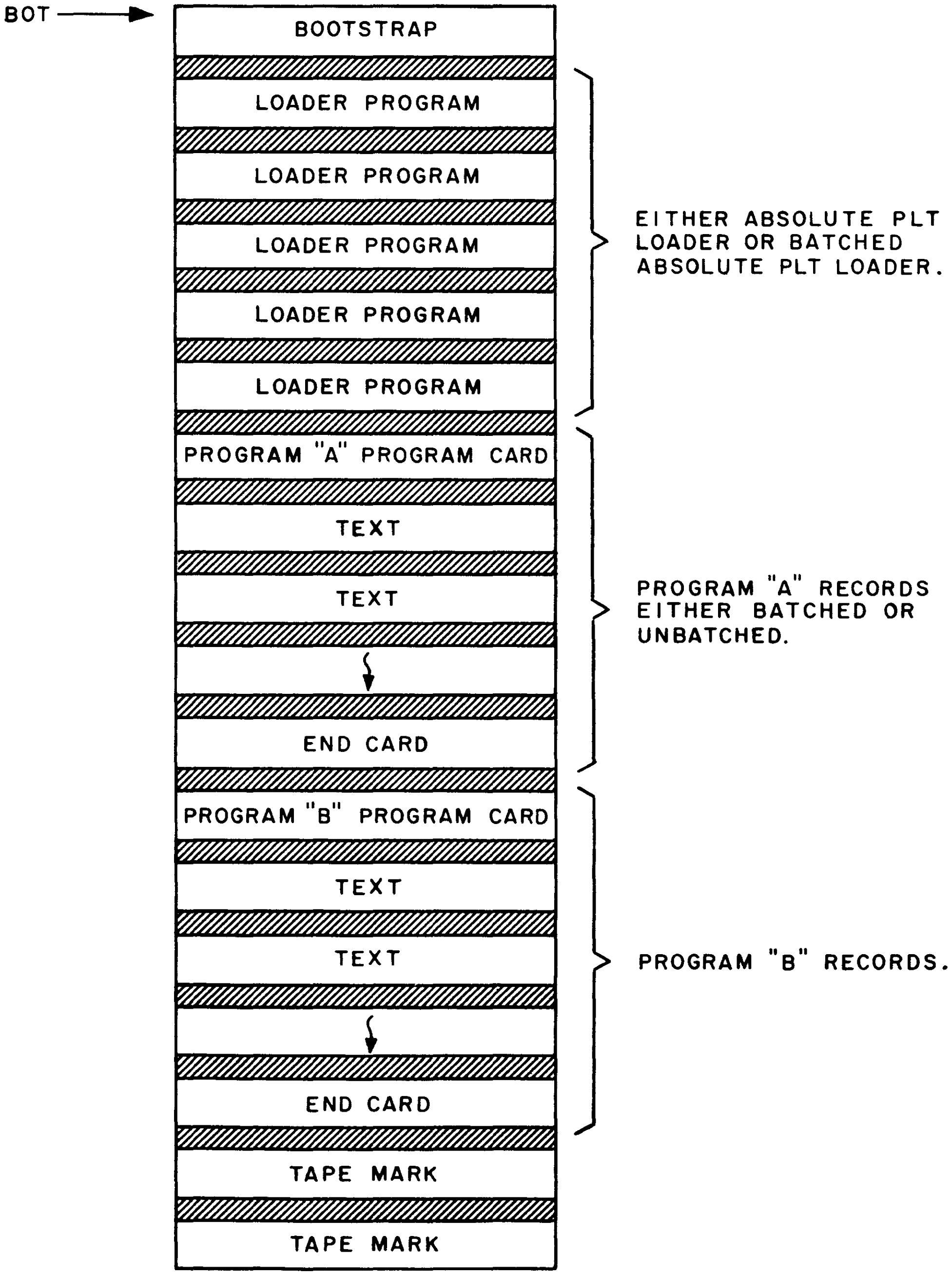
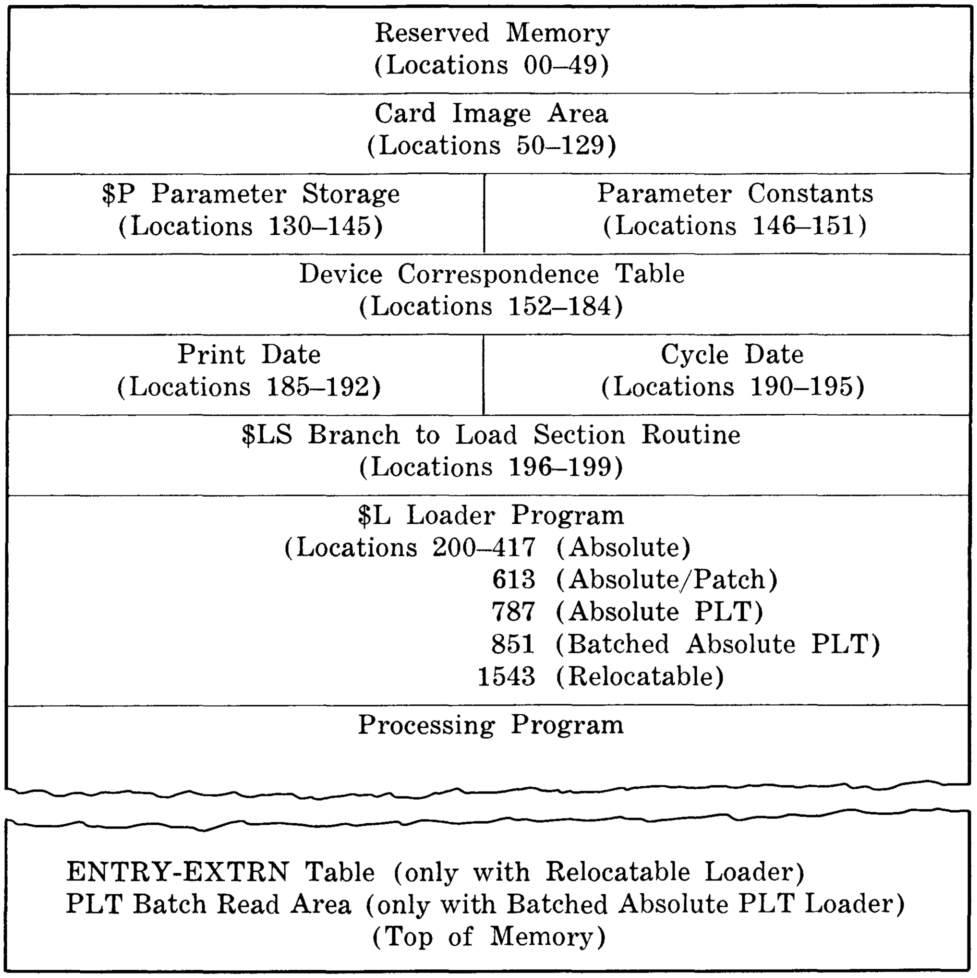


Figure 10. Program Library Tape Load Format

HSM LAYOUT

◆ Standard elements of the 70/15 Programming System are mapped into high-speed memory according to the following standard memory locations:



Reserved Memory (Locations 00–49)	
Card Image Area (Locations 50–129)	
$P Parameter Storage (Locations 130–145)	Parameter Constants (Locations 146–151)
Device Correspondence Table (Locations 152–184)	
Print Date (Locations 185–192)	Cycle Date (Locations 190–195)
$LS Branch to Load Section Routine (Locations 196–199)	
$L Loader Program (Locations 200–417 (Absolute) 613 (Absolute/Patch) 787 (Absolute PLT) 851 (Batched Absolute PLT) 1543 (Relocatable)	
Processing Program	
ENTRY-EXTRN Table (only with Relocatable Loader) PLT Batch Read Area (only with Batched Absolute PLT Loader) (Top of Memory)	

A description of these standard memory areas follows.

Card Image Area
Locations 50–129

◆ The card image area is the read-in area used by the loader when loading a program from cards or magnetic tape. The loader processes the card image in this read-in area.

$P (Parameter Storage)
Locations 130–145

◆ The $P area is used for temporary storage of input parameters for program subroutines. Each subroutine utilizes this area differently but it is conventional to place the return address in the first two bytes of the $P area.

Parameter Constants
Locations 146–151

◆ These special constants are used by the media-to-media utility routines and the loader. The constants are as follows:

L	S	U	M	0	0

where: L — indicates to the media-to-media routines whether a second job is ready for concurrent processing.

$L \neq 0$ — indicates no other job is ready.

$L = 0$ — indicates another job is ready.

Parameter Constants Locations 146–151 *(Cont'd)*

It is the responsibility of the operator to set L to a nonzero character. The loader and the media-to-media routines will reset it to zero whenever a *not-ready* condition exists.

S — informs the media-to-media routines about the status of concurrent job processing.

S = 0 — means no jobs are being processed.

S = 1 — means one job is being processed.

S = 2 — means two jobs are concurrently being processed.

UM — a two-byte address that tells the loader the highest legal address in memory, 4095 or 8191.

"UM" is interrogated by programs which adjust their memory requirements. The contents of UM are loaded in the process of bootstrapping the loader. An EXECUTE card containing all the parameter constants is the means of this loading.

00 — two binary zero bytes reserved for future use.

Device Correspondence Table Locations 152–184

◆ The loader places an entry in the Device Correspondence Table (DCT) by use of an I/O Define card. The DCT is a table from which an actual peripheral device is related to a logical device number. When a logical device is indicated in the device parameters, the 70/15 I/O Control System uses the logical device number to locate and access the correct actual device.

The DCT allows a maximum of 10 devices (0–9) to be tabled at any one time. There are three bytes per device. The table and the meaning of the bytes are shown below.

Logical Device	0	1	2	9
Actual Device	A B C	A B C	A B C	A B C

Bytes ABC are broken down as follows:

A		B		C
4 bits	4 bits	4 bits	4 bits	8 bits
TK#	U#	ATK#	Dev. Type	Control Info.

where: TK# — the number of the device trunk (0–5).

U# — the number of the device unit 0–F).

ATK# — the number of the alternate trunk by which the device can be accessed (0–5).

Dev. Type — is the device type as follows:

0 = DXC
1 = Magnetic Tape
2 = Card Reader or Videoscan Document Reader
3 = Card Punch
4 = Paper Tape Reader
5 = Paper Tape Punch
6 = On-Line Printer
7 = Input/Output Typewriter
8 = Card Reader/Punch
9 = Single-Channel Communications

Device Correspondence Table Locations 152–184 *(Cont'd)*

Control Info. — is the control byte required to prime a 7-level tape station. It is described under the I/O Define card format in the program library organization section.

Print Date Locations 185–192

◆ The Print Date area is used to store any date described by the programmer in the following format.

MM/DD/YY

where MM — two-byte month (01–12)
DD — two-byte day (01–31)
YY — two-byte year (00–99)

The date may be set up by an EXECUTE card, console operation, or program instruction. Normally the current date is placed in the date area at the beginning of the work day and extracted from there for individual program needs.

Cycle Date Locations 191–195

◆ The Cycle Date is used by tape-label processing programs to determine the security state of an interrogated tape. If the expiration date (see Tape Label Formats) of the tape file is less than that of the contents of the Cycle Date area, the tape may be utilized as an output tape for the program. The Cycle Date is stored in the form:

YYDDD

where: YY — the year of the century
DDD — the date of the year

$LS Branch Locations 196–199

◆ This four-byte area contains an unconditional branch to the first instruction immediately following the loader read. This permits the program to generate a control parameter in the card image area and then to branch to below the read as though a parameter read had occurred.

ENTRY-EXTRN Table

◆ This table is of variable size. It is constructed from the top of memory down. Each element corresponds to a unique ENTRY or EXTRN reference in the program and is six bytes long. A diagram of an element follows:

4 bytes	2 bytes
name	aa

where: name — a four-byte name of the ENTRY or EXTRN card reference.
aa — the two-byte address derived from the same cards.

If the most-significant bit of aa is 1, the element corresponds to an EXTRN element; if a zero, it corresponds to that of an ENTRY.

The address of an EXTRN element is the location of the last program reference to the named external.

The address of the ENTRY element is the address where that referenced entity is located in the program and where external reference to it are linked.

PROGRAMMING STANDARDS

◆ Programming standards followed in the 70/15 Programming System are described below.

Program Tag Assignments

◆ To avoid possible multiplicity of symbolic-name usage when components of the 70/15 Programming System are assembled with the production programs, the format of the names or tags has been restricted in each 70/15 System component. Names of entry points within the program may have a different mnemonic value.

The programming tag or name assignments and the components to which they apply are as follows. The "xx" may be any alphanumeric pair of characters.

Program Tag Assignments

70/15 Programming System	Name Format
Loaders	LDxx
Assembler	ASxx
Input-Output Control	IOxx
Card-to-Tape	CTxx
Tape-to-Punch	CPxx
Tape-to-Printer	TPxx
Memory Dumps/Snapshots	MDxx
Tape Edit	TPxx
Program Binder	PBxx
PLT Update	FUxx
Sort/Merge	SMxx
Report Program Generator	RPxx
Communication Control	CCxx

Parameters

◆ All input parameters to the 70/15 Programming System contain a "$" in the first location of the record. Whenever an end of file or end of job parameter is required, the record must contain $EOF in its first four locations.

Each 70/15 component compares the first four characters of every record with the first four characters of its own parameter name set. If there is no match, the record is assumed not to be a parameter, but a data card.

Label Processing

Introduction

◆ The 70/15 Programming System permits processing the following classification of files:

1. Standard labeled files
2. Combined labeled files
3. Nonstandard labeled files
4. Unlabeled files

Label processing of standard label files only will be provided in the system. It is, therefore, the responsibility of each programmer to provide label own-coding to process combined labeled files, nonstandard labeled files, and unlabeled files.

Label Functions

◆ 1. *Data Protection* — The 70/15 tape label processing is designed to perform data protection for all tapes that are to be written to. This consists of checking the expiration date in the label against the Cycle Date to see if the tape may be reused. The programmer has the option to bypass this check if he so desires.

Label Functions (Cont'd)

2. *Certification* — Verification that the correct input data is mounted is also provided for. This verification is based on checking certain items in labels such as file identification, and volume sequence number.
3. *Audit Control* — Audit Control is provided to verify that the correct number of data blocks has been read by checking the block count.

Label functions are performed as follows in regards to the 70/15 Programming System components:

Assembly, RPG, Sort/Merge — Perform data protection for all tapes to be written to

IOCS — Responsibility of the programmer to provide label processing coding in his program

Utility routines — Perform data protection for all tapes to be written to. A label own-code entrance is provided for other than standard label processing.

Test routines — Labels accepted but no label functions performed by these routines

System Maintenance routines — Perform data protection for all tapes to be written to

Information concerning label formats will be in a Spectra 70 Systems Standards publication.

Input/Output

Peripheral Device Definitions

◆ Except for the loaders, the input/output command within each 70/15 System component do not assume the use of actual peripheral device trunk and unit numbers. Instead, they are coded with logical device numbers and are linked to actual devices only at execution time. A standard function of the loader performs these linkage assignments.

A set of I/O Define cards (one for each logical device required), when passed through the loader, sets up a Device Correspondence Table (memory locations 152–184), that serves the I/O needs of the routine. These device assignments must be made before the loader transfers control to the routine.

Each component in the 70/15 Programming System observes the following standard logical device assignments:

Logical Device No.	Actual Device
00	Card Reader (or substitute)
01	1st (7th) Magnetic Tape input (output)
02	2nd (6th) Magnetic Tape input (output)
03	3rd (5th) Magnetic Tape input (output)
04	4th (4th) Magnetic Tape input (output)
05	5th (3rd) Magnetic Tape input (output)
06	6th (2nd) Magnetic Tape input (output)
07	7th (1st) Magnetic Tape input (output) (or substitutes)
08	Card Punch (or substitute)
09	Printer (or substitute)

Own-Code

◆ Every 70/15 System component that contains an own-code option will transfer to the own code through the use of an external reference (see Assembler) to an instruction named "USER". If own code, containing an instruction and entry point named "USER", is bound to the system component, a transfer will be effected to it whenever appropriate for the given program.

If a parameter requesting transfer to own code appears when no external linkage has been set up, the program will come to an F2 error halt.

Compatibility From 70/15 to 70/25

70/15 Programming Systems

◆ Without modification, every component in the 70/15 Programming System can be executed on a 70/25 Processor whose peripheral devices are connected to *selector channels.* Any standard 70/15 loader, with a compatibility card added, is used to load the 70/15 program into the 70/25.

The 70/15 Program Loader includes a provision by which certain instructions can be added to the loader and executed at load time to conditions, masks, etc., on the 70/25 for the 70/15 program.

The load procedure loads appropriate multiples of 4,096 into the 70/25 Base Address registers for proper program relocation.

The load procedure inhibits all interrupts before control is given to the 70/15 program or 70/15 component. The 70/15 coding must explicitly allow only those interrupts that are legal in the 70/15 and for which the programmer or programming system has provided interrupt-handling code.

All 70/15 programming components required by a program must adhere to the rules for compatibility as defined for programs below.

Programs

◆ The programmer must explicitly allow those interrupts that are expected on the 70/15. Any other interrupt condition must be left inhibited in the Interrupt Mask.

The 70/15 Program logic cannot be "time dependent".

The first 50 bytes of memory must be reserved for hardware use and cannot be used for temporary storage, etc.

The 70/15 Program cannot include core wrap-around techniques.

The 70/15 Program must not assume an I/O operation to be complete until it has executed either a Post Status or Sense instruction to the device.

Unused fields of 70/15 construction must contain only binary zeros.

Intentional OP Code Traps must be used carefully so that illegal 70/15 op codes which correspond to legal 70/25 op codes are not used unless they are specifically intended.

The 70/15 Programs must appear in the standard 70/15 program loader format.

The 70/15 Read Auxiliary and Write Auxiliary instructions must have a correct D2 address specified.

Any 70/15 Program that can legally get I/O Request Interrupts (Input/Output Typewriter) must allow all interrupts for the associated channel. In order for such a program to run on the 70/25, it must include interrupt code which explicitly verifies that an I/O interrupt is an I/O Request and not an I/O termination. If an I/O termination interrupt is encountered, the 70/15 interrupt code must return to the interrupted program and effectively ignore the interrupt.

OPERATOR/MACHINE COMMUNICATION

◆ The 70/15 Programming System permits continuous operation of the system with minimal dependence on the operator. For those instances where the programming system must return control to the program, a series of standard error halts and message indicators has been designed.

Standard Halts

◆ Each display byte appearing in the "M" register or from the staticizing of a Halt and Branch instruction has a unique meaning in the 70/15 Programming System. By observing the display character, the operator can differentiate between the various meanings and take interpretive action.

Error Recovery

◆ Standard error recovery is provided for the programmer by the 70/15 Programming System only in the case of magnetic tape operations. The Input/Output Control System will automatically perform rereading or erasing in the event of magnetic tape reads and/or write errors for the specified number of times. In the event that recovery is not possible, control will be released to the programmer for appropriate action.

In the case of all other peripherals, standard error correction or recovery is not performed and control is immediately given to the programmer who will supply own coding routines to effect recovery.

Programming System

◆ Standard error recovery procedures are provided in each system component for the applicable devices.

IMPLEMENTATION (LANGUAGE) STANDARDS

◆ With the exception of the bootstrap portion (first two cards) of the loaders, all programs in the 70/15 System are written in the RCA 70/15 Assembly language.