Price: $3.00

MONARCH REFERENCE MANUAL

for

900 SERIES/9300 COMPUTERS

90 05 66D

December 1969

XDS

Xerox Data Systems/701 South Aviation Boulevard/El Segundo, California 90245

© 1964, 1965, 1966, 1967, 1968, 1969, Xerox Data Systems, Inc.

Printed in U.S.A.

REVISION

This publication, XDS 90 05 66D, is a minor revision of the MONARCH Reference Manual, 90 05 66C (dated July 1967). A change in text from that of the previous manual is indicated by a vertical line in the margin of the page.

RELATED PUBLICATIONS

Title	Publication No.
XDS 92 Integrated Computer Reference Manual	90 05 05
XDS 910 Computer Reference Manual	90 00 08
XDS 920 Computer Reference Manual	90 00 09
XDS 925 Computer Reference Manual	90 00 99
XDS 930 Computer Reference Manual	90 00 64
XDS 940 Computer Reference Manual	90 06 40
XDS 9300 Computer Reference Manual	90 00 50
XDS 900 Series/9300 Monarch Technical Manual	90 06 16
XDS 900 Series FORTRAN II Reference Manual	90 00 03
XDS 900 Series FORTRAN II Operations Manual	90 05 87
XDS 900 Series/9300 ALGOL 60 Reference Manual	90 06 99
XDS 900 Series/9300 Symbol and Meta-Symbol Reference Manual	90 05 06
XDS 900 Series/9300 Project Management System Reference Manual	90 08 18
XDS 900 Series/9300 Extended Project Management System Reference Manual	90 15 04
XDS 900 Series/9300 Sort/Merge Reference Manual	90 09 97
XDS 900 Series/9300 Business Language Reference Manual	90 10 22
XDS 900 Series/9300 Manage Reference Manual	90 10 46
XDS 900 Series/9300/Sigma ADAPT Part Programming Reference Manual	90 10 45

NOTICE

The specifications of the software system described in this publication are subject to change without notice. The availability or performance of some features may depend on a specific configuration of equipment such as additional tape units or larger memory. Customers should consult their XDS sales representative for details.

CONTENTS

1. INTRODUCTION 1

Operating Environment 1
MONARCH 1
META-SYMBOL/SYMBOL 2
FORTRAN II 2
ALGOL 2
Functions of MONARCH 2
How MONARCH Performs Its Functions 2
Salient MONARCH Features 3

2. MONARCH CONTROL MESSAGES AND FUNCTIONS 4

Syntax of Control Messages 4
Control Message Parameters 4
Numeric Parameters 4
Literal Parameters 5
Symbolic Parameters 5
Control Message Formats 5
System Control 6
JOB 6
ENDJOB 6
ASSIGN (ONLINE) 6
Processor Control 7
METAXXXX 7
SYMBOL 8
FORTRAN 8
RTFTRAN 8
ALGOL 9
Input Control 9
FILLSYS 9
LOAD 9
FORTLOAD 10
RTFTLOAD 11
FORTLINK 11
ALGOLOAD 11
Utility Functions 11
C 11
SET 12
LABEL 12
DISPLAY (SHOW) 12
POSITION 13
REWIND 13
SKIPFILE 13
SKIPREC 13
BACKFILE 14
BACKREC 14
WEOF 14
BOOTLOAD 14
CARDTAPE 15
EOF 15
System Maintenance 15
UPDATE 15

3. THE MONARCH SYSTEM 16

Monitor 16
Standard System Routines 16
Termination of a Run 17
Loader 17
Unit Assignment Requirements 18
Storage Allocation 18
The Loading Process 19
Automatic Selective Loading from the MONARCH library 19

4. PROGRAMMING FOR MONARCH 21

Octal Dump Routine 21
Symbol Table Typeout Routine 21
Octal Correction Routine 21
Loader Routine 22
System Update Routine 23
The Update Control Message 23
Controlling an Update Run 24
COPY Messages 25
Contents of a Typical MONARCH System Tape 27
System Tape 27
Examples 28
Error Halts during Update Runs 29

5. PREPARING PROGRAM DECKS 30

META-SYMBOL Assembly and Execution 30
FORTRAN Compilation and Execution 31
ALGOL Compilation and Execution 34

6. OPERATING PROCEDURES 36

Loading the MONARCH System 36
Furnishing Control Messages 36
Program Halts and Recovery Procedures 38
System Output 38
MONARCH Loader's Symbol Table 38
FORTRAN Loader's Output 38

APPENDIXES

A. MONARCH UNIT ASSIGNMENT TABLE 40
B. MONARCH SYSTEM TAPE 43
C. LOADER OPERATIONS 49
D. UPDATING META-SYMBOL ON MONARCH TAPES 54
E. FORTRAN LINKAGE 55
F. XDS STANDARD BINARY LANGUAGE 60
G. PROCESSOR DIAGNOSTICS 66
H. RAD MONARCH 70
I. XDS CHARACTER CODES 74
J. FORTRAN II RUN-TIME MAGNETIC TAPE OPERATIONS 75
INDEX 79

ILLUSTRATIONS

1. MONARCH Program Halts and Recovery Procedures 37
2. FORTRAN Loader Output 39
3. MONARCH Loader Symbol Table 39
4. FORTRAN Run-time Errors 76

1. INTRODUCTION

MONARCH is a batch-oriented operating system that allows batched assemblies, compilations, and executions. The MONARCH system is available to users of XDS 9300 Computers and of XDS 900 Series Computers, and offers three distinct advantages:

1. Reduced operator intervention that increases operational reliability.
2. All control messages recorded at the typewriter for visual verification and permanent logging.
3. Batch processing capability that reduces throughput time.

MONARCH allows batch processing to proceed without the operator having to set up processing parameters or select input/output devices. Use of appropriate control cards preceding the program permits intermixing and uninterrupted processing of assemblies, compilations, and executions. Printouts of control and error messages are made available during processing, and the operator is concerned only with setting up of tapes, loading of cards, etc. If a program fails, the operator inspects the hard copy of control information and makes necessary adjustments in input/output assignments, tape designations, etc.

A certain portion of MONARCH – called the Resident – remains in main memory at all times. The resident includes the MONARCH Bootstrap (to load the MONARCH operating system), the Unit Assignment Table (UAT), error and job switches, memory dump routines, and for RAD MONARCH, the RAD file Management Package. The MONARCH operating system does not remain in core during the execution of programs; only the resident portion is there. MONARCH is reloaded as needed between jobs.

The major portion of the MONARCH operating system is the Monitor (see Section 3). The monitor accepts control information from many input media, assigns peripheral equipment as requested, and loads and executes specified standard system routines. The control messages must precede the program to be processed. In this manner, batch processing proceeds free from operator intervention and may involve the consecutive processing of programs from different media.

During all operations, a portion of MONARCH resident in main memory retains a record of input/output assignments and contains the bootstrap. When called, the bootstrap loads a short program (Bootstrap Loader) which in turn loads the MONARCH loader. The MONARCH loader can bring any of the following routines into core from the system tape:

1. The META-SYMBOL Assembler (900 Series only).
2. The SYMBOL Assembler.
3. The META-SYMBOL Loader (loads binary object programs produced by META-SYMBOL and SYMBOL).
4. The FORTRAN II Compiler (900 Series only).
5. The FORTRAN Loader (loads object program produced by the FORTRAN compiler, necessary subroutines from the FORTRAN Library, and the FORTRAN Run-Time package (900 Series only).
6. The ALGOL Compiler.
7. The ALGOL Loader (loads object programs produced by ALGOL).
8. The Update Routine (allows modifications and updating of the MONARCH system tape).
9. The SYSGEN Routine (copies a RAD MONARCH system tape onto the RAD).

The library and utility routines, provided with the system or added by the user to the system tape, are also brought into core by the loader.

In the remainder of this document, the phrase "system tape" refers to the medium on which the MONARCH system is stored: a magnetic tape for TAPE MONARCH, or a RAD File for RAD MONARCH.

OPERATING ENVIRONMENTS

The operating environment in which this system is to function is given below. The appropriate interrupt and data transmission facilities are assumed.

MONARCH

The MONARCH system requires for its use the following minimum configuration of computer equipment:

1. An XDS 9300 Computer or an XDS 900 Series Computer system with at least 8192 words of core memory.

2. A console typewriter to be used by the system to communicate with the console operator.

3. Two[†] or more on-line magnetic tape units. The system tape is on a magnetic tape unit.

4. For RAD MONARCH, one RAD File.

5. One or more of the following:

   card reader/punch
   paper tape reader/punch
   line printer

META-SYMBOL/SYMBOL

META-SYMBOL requires 12,288 words of core memory; its requirements for input/output devices are the same as those for MONARCH. SYMBOL operates on the minimum configuration required by MONARCH.

If both the encoded and symbolic input are on the same device, an additional magnetic tape or MAGPAK is required by META-SYMBOL. The same requirement is true if either the symbolic or encoded input is to be read from magnetic tape.

FORTRAN II

FORTRAN II operates on the minimum configuration required by MONARCH.

ALGOL

In addition to the minimum MONARCH configuration, ALGOL requires one output device; that is, ALGOL must have a device for binary output and one for output listing (a line printer).

FUNCTIONS OF MONARCH

The operating system is a basic program execution package which provides the following functions:

1. Loading and execution of standard system routines. For example:

   a. FORTRAN compilation.

   b. META-SYMBOL assembly.

   c. Punched card-to-magnetic tape conversion.

   d. MONARCH system updating.

2. Loading and execution of previously compiled or assembled programs for checkout or production runs. For example:

   a. Run a previously compiled ALGOL program.

   b. Run a program consisting of several previously compiled subprograms and a previously compiled FORTRAN main program.

3. Combined assembly, loading, and execution of programs for checkout or production runs. For example:

   a. Compile-and-go execution of FORTRAN programs.

   b. Assemble-and-go execution of symbolic programs.

4. Combinations of the above functions. In the following examples the phrases "job stack" and "batched job stack" refer to the collection of control information, programs, and data which are to be processed under control of the MONARCH monitor routine. For example:

   a. A series of META-SYMBOL assemblies.

   b. Several META-SYMBOL assemblies intermixed with one or more META-SYMBOL object programs to be assembled and then executed.

   c. A mixed batch requiring that any or all MONARCH functions be carried out in an arbitrary sequence determined by their order in the batched job stack.

5. Loading of standard input/output routines prior to loading and executing previously assembled programs, so that these standard routines can be executed upon request from the program being run. For example:

   a. Loading standard input/output routines from the system tape.

   b. Loading a conversion routine or trigonometric function routine from the MONARCH library.

HOW MONARCH PERFORMS ITS FUNCTIONS

The MONARCH operating system performs its functions between jobs and does not exercise control over the execution of a program once that program has been loaded and control has been transferred to it. The functions are indicated to MONARCH via control messages.

Upon request, MONARCH loads a program and then relinquishes control of the computer and its associated peripheral equipment to the program. The only possible way MONARCH can regain control of the computer is if MONARCH is reloaded from the system tape. This may

[†]A MAGPAK may be used in place of two magnetic tape units.

be done manually by the console operator or under program control by the program being executed.

SALIENT MONARCH FEATURES

The salient features of MONARCH are:

1. The system minimizes the amount of manual intervention required to execute a succession of independent or related programs on the computer.

2. Core memory requirements for the monitor routine are minimized during program execution; that is, the monitor performs its functions between program executions, and MONARCH holds only those instructions and data required for continuity of operation during program execution.

3. The amount of control information that must be furnished to the monitor and the system routines is held to a minimum.

4. The control information for all system functions is presented in a consistent and straightforward manner.

5. Insertion and deletion of routines from the system are accomplished via a simple update routine.

6. Routines to be added to the system are introduced in the standard format used for assembly program output. That is, almost any program which can be assembled by SYMBOL or META-SYMBOL can be incorporated into this operating system as a standard system routine.

2. MONARCH CONTROL MESSAGES AND FUNCTIONS

When the MONARCH system is loaded, the monitor takes control of the computer and obtains the first item of control information from the console typewriter. This item may be any legal control message. With a C control message, the operator may specify that future control messages are to be obtained from other input media. Control messages may be entered through the following input media:

console typewriter

punched cards via an on-line card reader

punched paper tape via a paper tape reader

magnetic tape via a magnetic tape unit (other than the one on which the system tape is mounted)

When the monitor obtains a control message from a medium other than the typewriter, it types the message before executing the function requested. (The operator may direct the monitor to print the control message on an on-line printer. See "Operating Procedures," Section 6.) In this way the monitor informs the console operator of the functions being performed under its control and maintains a written record of such functions. The monitor tells the operator that a given function is completed by typing the next control message or by requesting the next one from the typewriter.

SYNTAX OF CONTROL MESSAGES

Regardless of which device the monitor accesses for control information, the format of the messages is the same:

$\Delta f.$

or

$\Delta f_{\wedge} p_1, p_2, \ldots, p_i.$

Δ (1 character) indicates the beginning of a message.

f (1 to 8 alphanumeric characters) is a mnemonic control function code.

$\wedge$ indicates a space. These symbols are used to indicate the minimum number of spaces which must separate the function code and the first parameter.

p_1 ($1 \leq i \leq 24$) is a symbolic, numeric, or literal parameter that provides necessary control information related to the control function (f). For example, a request for the system loader to load a program must indicate the intitial load relocation bias for the program that is to be loaded. A maximum of 24 parameters may be specified in one control message.

, A separator. Acceptable separators are

, = > < $ [ ] * / ( ) ∧ ' @

(1 character) indicates the logical end of message. The physical end of message is indicated by the end of record in the case of magnetic tape and cards or by a carriage return in the case of paper tape and typewriter. The logical end of message is required only when comments are included.

Regardless of the length of the record containing a control message, the routine that scans control messages examines only the first 72 characters (18 words) of the record. Therefore, the period indicating the end of the message must occur prior to the 73rd character of the record.

The first character of a control message is a delta (Δ). This character and the function code may be contiguous or may be separated by one or more spaces. When the function does not require a parameter list, the function code is followed by a period; otherwise, a space separates the parameter list from the function code.

Various control messages require different numbers of parameters. If more than one parameter is required, all but the last must be followed by a separator; the last one is followed by a period.

CONTROL MESSAGE PARAMETERS

The monitor converts parameters in a MONARCH control message into a standard internal form. Except for literals (see below), it represents parameters internally as single-precision, 24-bit, binary values. Hence, the "value" of a parameter is its internal representation as a binary quantity or bit pattern, and it is the "value" of a parameter that is ultimately examined by the subroutine in the monitor which processes the specific function code. This method of parameter conversion usually allows many ways of representing a given value externally. For example, the control message parameters '000A' (literal parameter) and MT1W (symbolic parameter) have the following internal representations or values:

'000A' = 00000021
MT1W = 00203611

which could be entered as numeric parameters.

NUMERIC PARAMETERS

A numeric parameter may be written as:

1. An octal integer, consisting of up to nine octal digits, the first of which must be zero. An octal integer may be preceded by an algebraic sign:

   047, +062, -0, 0, 077777777, -032154767

2. A decimal integer, consisting of up to eight decimal digits the first of which must not be zero. A decimal integer may be preceded by an algebraic sign:

   9, +532, -0, +21657899, -31579988

The first digit of an octal integer must be zero in order for MONARCH to distinguish it from a decimal integer.

Unsigned octal parameters must have values less than 2^{24}; signed octal parameters and all decimal parameters must be less than 2^{23}. If the number of digits in a number exceeds the applicable limit, the least significant digits are truncated.

Regardless of the form used, the parameter is represented internally as a single-precision, signed, binary integer.

LITERAL PARAMETERS

A literal parameter consists of up to eight alphanumeric characters enclosed in single quotation marks (SDS internal code of 14).[†] Any legal character (see Appendix H) except a quotation mark may be written as a literal:

1. A single-precision literal consists of up to 4 characters:

   'A' 'A$_{\Delta}$B' 'RATE' 'A.3,'

2. A double-precision literal consists of up to 8 characters:

   'ABC' 'LITERAL' 'ALPHA777' 'START:3.'

A literal parameter is represented internally as a left-justified string of internal character codes (six bits each). A single-precision literal is stored in one computer word. A double-precision literal is stored in two words:

'ABCDEFGH' is stored in α and $\alpha + 1$ as

(α) = 21222324
(α+1) = 25262730

Spaces (internal code of 60) are used to fill any unspecified character positions. For example:

'ABC' is represented internally as 21222360

'ABCDE' is represented internally as 21222324
25606060

[†]The single quotation mark is not present on the typewriter; however, its internal code (14) is the same as that of the typewriter symbol @ (upper case 8). Therefore, whenever input is from the typewriter, the symbol @ must be used in place of the single quotation mark.

Double-precision literals are frequently used to designate program names for MONARCH search functions. (See LOAD control message.)

SYMBOLIC PARAMETERS

Symbolic parameters are symbolic representations of parameters associated with many of the control messages explained later in this section.

Symbolic parameters consist of up to four characters, the first of which must be alphabetic and the remaining either alphabetic or numeric. Each symbolic parameter has a predetermined value (24 bits) which is stored in MONARCH's table of symbolic parameters. Use of a symbol not defined in this table causes the control message in which the undefined symbol appeared to be ignored and an error message to be typed.

A octal parameter with the same value may be substituted for any symbolic parameter in a control message.

Listed below are the symbolic parameters used to specify input/output devices to the MONARCH system. In this list, h specifies the channel and is actually written as W or Y for 900 Series Computers or A, B, C, or D for 9300 Computers. Throughout this manual, references are made to the "W" and "Y" buffers; users of the 9300 Computer should substitute the appropriate channel letters in these places. If h is omitted from a parameter, channel W (or A) is assumed. The n specifies the unit number.

Parameter	Definition
CR	designates the card reader where there is only one.
CRnh	designates card reader n on channel h.
CPnh	designates card punch n on channel h.
PRnh	designates paper tape reader n on channel h.
PPnh	designates paper tape punch n on channel h.
TY	designates the console typewriter (i.e., typewriter 1 on channel W).
MTnh	designates magnetic tape unit n on channel h.
DFnh	designates RAD File n on channel h.
LP	designates the line printer (i.e., line printer 1 on channel W).
S	designates the system tape (i.e., magnetic tape unit 0 on channel W).

CONTROL MESSAGE FORMATS

The user directs the operation of the MONARCH system via control messages which may be input from the typewriter, punched paper tape, or punched cards. Most frequently, the control messages are on punched cards preceding the user's input card decks.

System Control	Utility Functions
JOB	C
ENDJOB	SET
ASSIGN (ONLINE)	LABEL
	DISPLAY (SHOW)
Processor Control	POSITION
	REWIND
METAXXXX[t]	SKIPFILE
SYMBOL	SKIPREC
FORTRAN	BACKFILE
RTFTRAN	BACKREC
ALGOL	WEOF
Input Control	BOOTLOAD
	CARDTAPE
FILLSYS	EOF
LOAD	
FORTLOAD	System Maintenance
RTFTLOAD	
FORTLINK	UPDATE
ALGOLOAD	

SYSTEM CONTROL

JOB The JOB control message specifies the system is to be in "job mode".

△JOB.

When the system is set to job mode, it resets the processor error switch. If an error occurs while a processor (such as FORTRAN or META-SYMBOL) is being executed, the processor error switch is set. Then, if the operation is a load-and-go (i.e., compile-and-execute or assemble-and-execute), the "load" function is not honored because of the processor error. If no error occurs during such a load-and-go operation, the "load" function is honored.

The MONARCH system remains in job mode until an ENDJOB control message is encountered. Therefore, whenever a job is preceded by a JOB message, it should be followed by an ENDJOB (see below) as a courtesy to the next user who may not wish to assemble (compile) his program in job mode.

END JOB This control message specifies that the system is not in job mode.

△ENDJOB

[t]XXXX indicates the name of a set of system PROCs that will be used to interpret the program mnemonics during the META-SYMBOL assembly (e.g., META920, META910, META9300, METASPEC, etc.). META910 will assemble on any machine and will produce binary output for 910. The set of PROCs is a △2 record within the scope of the META-SYMBOL logical file (see "System Update Routine" in Section 4 for an explanation of △2 records).

When an ENDJOB control message is received, MONARCH resets the processor error switch and terminates job mode. If a processor is being executed in a load-and-go operation not in job mode, MONARCH will honor the "load" function even if processor errors have occurred.

The system will not return to job mode until it receives a JOB control message.

ASSIGN
ON LINE ASSIGN (or ONLINE) enables the user to specify the input and output media to be used during the current job.

△ASSIGN $L=P_1, L=P_2, \ldots, L=P_9$.

△ONLINE $L=P_1, L=P_2, \ldots, L=P_9$.

L is a system label.

P is a symbolic parameter designating the specific device (see Symbolic Parameters).

The labels for the standard unit assignments are:

Label	Reference
SI	Symbolic Input
SO	Symbolic Output
BI	Binary Input
BO	Binary Output
EI	Encoded Input
EO	Encoded Output
LO	List Output
UI	Update Input
X1	System Scratch
X2[t]	System Intermediate Output Scratch
X3	System Scratch (magnetic tape)
S	System (magnetic tape)

Labels for Logical unit assignments are:

Label	Reference
L0	Magnetic Tape Logical Unit 0
L1	Magnetic Tape Logical Unit 1
L2	Magnetic Tape Logical Unit 2
L3	Magnetic Tape Logical Unit 3
L4	Magnetic Tape Logical Unit 4
L5	Magnetic Tape Logical Unit 5
L6	Magnetic Tape Logical Unit 6
L7	Magnetic Tape Logical Unit 7
LCR	Card Reader
LCP	Card Punch
LLP	Line Printer

Note: Ln may be assigned to any physical tape unit; i.e., L0=MT2W, L2=MT7W, etc.

The logical unit assignments are used in the Business Language and in FORTRAN II RAD MONARCH to permit reassignment of unit numbers.

[t]X2 must be assigned to magnetic tape unit 2 (MT2) under MAGPAK environment when using META-SYMBOL.

At least one pair of parameters must be given, and a maximum of nine pairs is allowed per control message. The value of the first parameter must be a label specifying a unit assignment entry; e.g., SI, LO, etc. The value of the second parameter must be a legal peripheral device designation on an existing channel; e.g., PR1W, MT3Y, etc. The symbolic parameter associated with the peripheral device should be consistent with the flow of information; that is, it would be illegal to assign BI=CP1W (binary input to be entered from the card punch).

Once a unit assignment has been made, it remains in effect until a new assignment for that label is made.

When a logical tape is assigned to the RAD for execution of a FORTRAN program, it should be assigned to X1 or X2, where X1 or X2 has been assigned to the RAD. This permits up to two such files to be simultaneously active.

Examples:

△ASSIGN BI=CR1W, BO=CP1W.

This message assigns card reader 1 on the W buffer as the binary input device and card punch 1 on the W buffer as the binary output device.

△ONLINE LCR=CR1W, LLP=LP1W, L2=MT1W.

This message assigns card reader 1 on the W buffer as the card input device, printer 1 on the W buffer as the on-line printer, and a second magnetic tape (L2) as magnetic tape physical unit 1 on the W buffer. This ONLINE statement assigns LCR, LLP, and L2 in BAT (Business Language Assignment Table, which is described in Appendix A).

Note that magnetic tape units are numbered 0 through 7; all other devices are numbered from 1.

△ASSIGN BI=CR1W, S=MT0W, LCR=CR1W.
△ASSIGN LLP=LP1W, L2=MT1W.

These messages assign: (1) card reader 1 on the W buffer as both the binary input device and the card reader for a Business Language program; (2) magnetic tape unit 0 on the W buffer as the system tape; (3) line printer 1 on the W buffer as the on-line printer for a Business Language program; and (4) the magnetic tape physical unit 1 on the W buffer as the magnetic tape logical unit 2 of a Business Language program.

PROCESSOR CONTROL

METAXXXX (900 Series only) This control message directs MONARCH to load and transfer control to the META-SYMBOL assembly system.

△METAXXXX P_1, P_2, C, CONC, EXCP, SET.

XXXX specifies which procedure-oriented library MONARCH is to load prior to the assembly. Thus, the control message may be written as:

△META920 for 920 procedure-oriented library.

△META910 for 910 procedure-oriented library.

△META9300 for 9300 procedure-oriented library.

△METAB910
△METAB920
△METAB93H
} for 900 Series special-purpose procedure-oriented library for business data processing.

The user may provide his own procedure-oriented library on the system tape. It must be identified by a unique, 4-character name. That name is then used in place of XXXX in the METAXXXX control message.

P_1 specifies type of input:

Parameter	Type of Input
SI	Symbolic Input
EI	Encoded Input

P_2 specifies type of output:

Parameter	Type of Output
SO	Symbolic (Source) Output
EO	Encoded Output
BO	Binary Output
LO	List Output

C (optional) specifies that compatibility mode translation of symbolic input is desired. Use of this parameter enables the user to translate a SYMBOL-4 or SYMBOL-8 source program into META-SYMBOL source form.

CONC (optional) specifies that a concordance listing[†] is to be produced by META-SYMBOL.

EXCP (optional) specifies that exceptions are to be made to the concordance listing[†] as designated on META-SYMBOL control cards INCLUDE and EXCLUDE. If EXCP is present, CONC is not specified.

SET (optional) specifies that a larger table should be reserved for use by the META-SYMBOL preassembler to accomplish translation of standard system procedures to the user's program format. This parameter is not necessary when the control message is META920, META910, or META9300, but should be used with a call for the XDS Business Language. SET may also be necessary for future higher order languages implemented in META-SYMBOL.

[†] "Concordance listing" refers to a listing of the symbols appearing in the META-SYMBOL source program, along with a reference to the instructions in which the symbols appeared. INCLUDE control cards may be used to limit concordance listing to specific symbols only; EXCLUDE control cards enumerate specific symbols which are to be omitted from the concordance listing.

The parameters may be listed in any order. One input and one output specification must be given. Multiple outputs may be requested.

META-SYMBOL assumes that the necessary input/output units have been assigned and that all tape units, except scratch tapes, are correctly positioned before MONARCH relinquishes control to it.

Examples:

△ASSIGN SI=CR, LO=LP, S=MT0W, X1=MT1W.
△META920 SI, LO.

This message sequence requests META-SYMBOL to assemble a symbolic source program and produce an assembly listing as the only output.

△ASSIGN S=MT0W, SI=CR, BO=PP1W, LO=LP.
△ASSIGN X1=MT1W.
△META9300 SI, BO, LO, CONC.

This sequence requests META-SYMBOL to assemble a symbolic source program from cards and to produce a binary output on paper tape and an assembly listing and concordance listing on the line printer.

SYMBOL The SYMBOL control message directs MONARCH to load and transfer control to the SYMBOL assembly system.

△SYMBOL P_1, P_2.

P_1 specifies which mnemonic table is to be used during the assembly:

Parameter	Mnemonic Table
910	910
920	920
9300	9300

P_2 specifies output data from SYMBOL:

Parameter	Type of Output
BO	Binary Output
LO	List Output

The parameters may appear in any order. Only one mnemonic table (P_1) may be specified; at least one output specification parameter (P_2) must be present. Symbolic input is assumed; therefore, SI should not be present as a parameter, but must be ASSIGNed.

Under MONARCH, SYMBOL has no initial halt to ready input, in contrast to previous bootstrap versions. Therefore, it is particularly important in the case of symbolic input from the paper tape reader (a device which has no device ready test) that the paper tape be ready at the time SYMBOL is loaded.

Each SYMBOL control message should be preceded by an ASSIGN control message, establishing the desired unit assignments. The ASSIGN card is indicative of device only; i.e., it supplants the typewriter control message of bootstrap versions and causes SYMBOL to load its own preset I/O package.

Example:

△ASSIGN S=MT0W, SI=CR, LO=LP.
△ASSIGN BO=PP1W, X1=MT1W.
△SYMBOL 9300, LO, BO.

This sequence of messages requests a SYMBOL assembly from cards, using the 9300 mnemonic table. The output from the assembly is to be a program listing on the line printer and an object program on punched paper tape.

FORTRAN (900 Series only) This control message causes MONARCH to load and relinquish control to the FORTRAN II compiler.

△FORTRAN P_1, P_2, P_3.

P_i specify type of input and output:

Parameter	Type of Transmission
SI	Symbolic Input
BO	Binary Output
LO	List Output

Any or all of the parameters may be omitted. Symbolic input is always assumed. The presence of the BO parameter causes an object program to be generated. The LO parameter causes an output listing to be produced. If no I/O unit assignment has been made to the BO or LO device, that parameter is ignored.

Examples:

△FORTRAN.

This message requests FORTRAN to compile a source program; no listing or object program is produced; only the program allocation, diagnostics, and any erroneous source line(s) will be listed.

△ASSIGN S=MT0W, SI=CR, LO=LP, X1=MT1W.
△FORTRAN BO, LO.

This sequence of messages requests FORTRAN to compile a source program read from cards (SI is assumed if it is not present) and to produce an output listing. No object program is produced since no BO unit assignment was made.

RTFTRAN (900 Series only) This control message causes MONARCH to load and relinquish control to the Real-time FORTRAN II compiler.

△RTFTRAN P_1, P_2, P_3.

P_i specify type of input and output consistent with the format of the above mentioned FORTRAN control message.

ALGOL The ALGOL control message causes MONARCH to load and transfer control to the ALGOL compiler.

ΔALGOL P_1, P_2, P_3.

P_i specify the input/output devices to be used by ALGOL.

Parameter	Type of Transmission
LS	List Source
LO	List Object Code
BO	Binary Output

The parameters may appear in any sequence, and any or all may be omitted. ALGOL always reads source inputs from the device previously assigned to SI. When a listing of the source program is requested (LS parameter), the listing is produced on the LO device. The LO device must be a line printer. When requested to list object code (LO parameter), ALGOL produces the list on the LO device. The BO parameter specifies that ALGOL is to produce a binary object program on the BO device.

If no output device is specified and an error occurs during compilation, an error message is produced on the console typewriter.

Note: ALGOL must have a scratch tape available to it and will automatically use the magnetic tape previously assigned to X1.

Example:

```
△ASSIGN S=MT0W, SI=CR, LO=LP, BO=PP1W.
△ASSIGN X1=MT1W.
△ALGOL LS, LO, BO.
```

This sequence of control messages requests the ALGOL compiler to read a source program from cards, compile it, listing the object and source programs on the line printer, and output the binary object program on punched paper tape. The magnetic tape unit 1 on the W buffer is to be the compiler's scratch tape.

Δ ADAPT[†] (900 Series only) No parameters. This control message directs MONARCH to load and transfer control to the ADAPT numerical control compiler. LINKZERO is loaded and control is transferred to it. This in turn reads the next control record which must contain the link number of the next link to be executed in column 4.

Example:

```
△ADAPT.
^^^1
```

This message sequence loads LINKZERO which in turn loads link 1 of the ADAPT compiler. Each link thereafter calls the next link until compilation is terminated.

Δ MANAGE[†] (900 Series only) No parameters. This control message causes MONARCH to load and transfer control to the MANAGE executive routine.

Δ DICTIONA[†] (900 Series only) No parameters. This control message is used in conjunction with MANAGE. It causes control to be transferred from the Executive routine to the Dictionary generator.

INPUT CONTROL

FILLSYS This control message transfers control to the monitor's bootstrap routine which will reload the MONARCH system.

△FILLSYS.

There are no parameters. The message is equivalent to executing an unconditional branch to memory location 00001.

LOAD The LOAD message directs the MONARCH loader to load one or more binary object programs.

△LOAD P_1, P_2, P_3.

P_1 is the load relocation bias, expressed as a positive octal or decimal integer, for the first (or only) program to be loaded. For programs whose load addresses and data words are not relocatable (i.e., absolute programs), the load relocation bias is ignored.

P_2 is the loader options parameter. The options are:

Parameter	Interpretation
STOP	Stop after each program is loaded (i.e., after each end record is read); no symbol table output and no return to the routine that called the loader. Octal equivalent of the symbolic parameter STOP is 10000000.

[†]The ADAPT compiler is not part of the standard MONARCH system, but is available on request for the tape MONARCH system only. When the ADAPT FORTRAN II MONARCH system tape is generated, only the following programs should be included on the system tape:

1. MONARCH control
2. I/O handler
3. FORTRAN II linking system
4. FORTRAN II compiler
5. FORTRAN II loader
6. FORTRAN II library
7. MONARCH tape update routines
8. SYSEND

[†]The MANAGE system is not part of the standard MONARCH system, but is available on request for the tape MONARCH system only. The MANAGE system tape must be generated with the Dictionary generator. The form and contents of this tape are specialized and will be supplied to the user upon request.

Parameter	Interpretation
GO	No halt after processing end record with transfer address; no symbol table output and no return to the routine which called the loader. Octal equivalent of the symbolic parameter GO is 40000000.
TSTP	Same as STOP except that the loader's symbol table is output. Octal equivalent of the symbolic parameter TSTP is 20000001.
TGO	Same as GO except that the loader's symbol table is output. Octal equivalent of the symbolic parameter TGO is 60000001.

If the loader is not requested to output the symbol table and unsatisfied Programmed Operator references or definitions occur, an error message and the unsatisfied references and/or definitions are typed, and MONARCH halts. If these unsatisfied references/definitions will not affect the operation of the program, the operator can clear the halt and the program will be executed. Otherwise, he can take appropriate action. When the loader is requested to output the symbol table, it produces the table on the line printer if Breakpoint 1 is set or on the typewriter if Breakpoint 1 is reset.

P_3 is an optional parameter that is interpreted as a program identification label assumed to occur in characters 9 through 16 of a level 1 MONARCH ID record on the current binary input unit (see Appendix B for a description of record formats).

At least one (P_1) and three parameters at most must be given for the LOAD control message. When parameter P_3 is present, its value is converted to a left adjusted, space-filled, 8-character search key. The monitor causes records to be read from the unit assigned for binary input (BI) until (1) a level 1 MONARCH ID record, with the same name in characters 9 through 16, is obtained or (2) the last file has been scanned (i.e., until a level 1 MONARCH ID record with "SYSEND$\wedge\wedge$" in characters 9 through 16 is encountered). In the first case control is relinquished to the MONARCH loader that processes the input as specified by the first two parameters. In the second instance, a message SEARCH FOR SPECIFIED ROUTINE FAILED is typed, and the next MONARCH control message is requested. A detailed description of the MONARCH loader is given in Section 3.

Prior to processing a load function, the monitor interrogates the processor error switch and the job mode switch (see JOB control message for an explanation of these switches). If both switches are set, the requested load function is aborted. An appropriate error message (PROCESSOR ERROR...) is printed, and typewriter 1 on the W buffer is selected for input of a control message.

Examples:

ΔLOAD 010000, STOP.

This message causes the MONARCH loader to load one or more programs, beginning in location 10000_8. Input is from the current BI device, and the loader stops (halts) after each program is loaded (i.e., after each end record is read).

ΔLOAD 2048, TSTP.

This message causes the MONARCH loader to load one or more programs, beginning in location 4000_8 (2048_{10}). Input is from the current BI device, and the loader halts after each program is loaded (i.e., after each end record is read). After loading is completed and prior to program execution, the symbol table is output.

ΔLOAD 0, TGO, 'FILENAME'.

This message (input from cards) causes the MONARCH loader to find program FILENAME (as a level 1 ID record) on the specified BI unit and to load the program with 0 relocation (i.e., as an absolute program). The loader's symbol table is output prior to program execution. If input is from the typewriter, this message would appear as

ΔLOAD 0, TGO, @FILENAME@.

(See discussion on "Literal Parameters.")

ΔLOAD GO.

This message forces a load relocation bias of 0 and may be used to load absolute programs.

FORTLOAD (900 Series only) The FORTLOAD control message causes MONARCH to load and transfer control to the FORTRAN loader. The parameters in the control message specify the mode in which the FORTRAN loader is to operate and the input devices from which it is to read.

ΔFORTLOAD $P_1, P_2, P_3, \ldots, P_8$.

P_i consists of up to eight parameters that may be given. The first three specify the mode in which the FORTRAN loader is to operate:

MAP	Produce a storage map[†] of the program on the console typewriter.
LMAP	Produce a label map[†] on the console typewriter.
LTRA	Produce a label trace[†] at execution time.

[†]The output resulting from the use of this parameter is described in Section 6.

These three parameters may appear in any order or may be omitted entirely. If they appear, they must be the first in the parameter string. The other parameters specify which input devices are to be read (e.g., X1, BI); at least one input device must be specified. Information is loaded from the devices in the order they appear in the parameter list. These devices must have been assigned and correctly positioned before the FORTRAN loader is called.

The FORTRAN loader automatically loads the previously compiled program, which must be on the first input device specified by the parameters. When additional input devices are specified (i.e., in addition to the unit from which the program is read), the loader reads from these devices only routines that are necessary because of unsatisfied references/definitions. However, if the user wishes to have the loader load from the additional devices unconditionally (i.e., regardless of whether or not the program references any of the routines read from that device), he places the letter U after the appropriate parameter.

Example:

```
△ASSIGN  BI=MT1W,X1=CR1W.
△FORTLOAD  MAP,LTRA,BI,X1U.
```

The FORTRAN loader will read a previously compiled FORTRAN program from magnetic tape unit 1 and will read, unconditionally, from the card reader. It will produce a storage map of the program and a label trace as the program is executed.

RTFLOAD (900 Series only). This control message causes MONARCH to load and transfer control to the Real-time FORTRAN loader. The parameters specify the mode in which the loader operates and the input devices from which it reads.

△RTFLOAD $P_1, P_2, P_3, \ldots, P_8$.

P_i consists of up to eight parameters that may be given. The first three specify the mode in which the FORTRAN loader operates; the narrative is consistent with that of FORTLOAD.

FORTLINK The FORTLINK control message causes MONARCH to load and transfer control to the FORTRAN loader. This message is used only when a link tape is to be generated. Linking is discussed in Appendix E.

△FORTLINK $P_1, P_2, P_3, \ldots, P_9$.

P_1 is the identification number to be assigned to the link about to be written on magnetic tape; it may be any three decimal digits.

P_{2-9} same as P_i for FORTLOAD.

ALGOLOAD This control message causes MONARCH to load and transfer control to the ALGOL loader. The parameter is optional.

△ALGOLOAD BI

BI (optional) specifies binary object program input. The ALGOL loader always reads binary object programs from the device previously assigned to BI.

After loading an ALGOL-compiled object program, the ALGOL loader searches the system tape (magnetic tape unit 0 on the W buffer) for any referenced library programs.

Examples:

```
△ASSIGN  BI=CR.
△ALGOLOAD.
```

This sequence of control messages causes the ALGOL loader to load a binary object program from the card reader, to load any referenced library programs from magnetic tape unit 0, and to transfer control to the object program.

```
△ASSIGN  BI=MT2W.
△ALGOLOAD  BI.
```

These messages cause the ALGOL loader to read a binary object program from magnetic tape unit 2. Then, the loader reads the system tape, loads the required library programs, and transfers control to the object program.

UTILITY FUNCTIONS

C The C control message directs MONARCH to accept future control messages from a specific input device.

△C P_1.

P_1 must be a legal input unit assigned to an existing buffer.

Parameter	Definition
CRnh	Designates card reader n on buffer h.
MTnh	Designates magnetic tape unit n on buffer h.
PRnh	Designates paper tape reader n on buffer h.
TYnh	Designates typewriter n on buffer h.

Unless a C control message directs otherwise, MONARCH automatically accepts control messages from the console typewriter (1 on the W buffer).

Once a C message has been processed, MONARCH immediately attempts to read a control message from the newly assigned device.

Example:

```
△C  PR1W.
```

This message assigns paper tape reader 1 on the W buffer as the control message input device.

SET This control message enables the user to set the contents of a specified memory location to a given value and is operative only if the MONARCH monitor is in control.

△SET A=V

A is any legitimate memory address.

V is the value to be stored in location A. If the value exceeds $2^{23}-1$, the most significant digits are stored.

A and V may be expressed as either octal or decimal numeric parameters.

Examples:

△SET 017=-59.

This message will cause the contents of memory location 00017_8 to be set to 77777705 ($-59_{10}=-73_8$).

△SET 64=077777.

This message will cause the contents of cell $100_8(64_{10})$ to be set to 00077777.

△SET 0235=001000114.

This message will cause the instruction BRU 00114 to be stored in location 00235_8.

LABEL The LABEL control message enables the user to write a level 1 or level 2 MONARCH ID record on a magnetic tape or on the RAD (see Section 4 "System Update Routine" for a discussion of MONARCH ID records).

△LABEL P_1,P_2,P_3.

P_1 is the value 1 to indicate a level 1 ID record or the value 2 to indicate a level 2 ID record.

P_2 is the unit on which the ID record is to be written. The value of this parameter must be either a legal logical or physical magnetic tape unit designation on an existing buffer (e.g., MT3W to specify magnetic tape unit 3 on the W buffer) or "BO", if the RAD file is to be labeled.

P_3 is a double- or single-precision literal, used to construct an 8-character name. If fewer than eight characters are given, the name field will contain (trailing) spaces in the right-most character positions.

A level 1 or level 2 MONARCH ID record (indicated by the first parameter) is constructed with the name field (characters 9 through 16) containing the identifier specified as the third parameter. The ID record is then written on the magnetic tape designated by the second parameter. A magnetic tape ID record consists of 40 characters (characters 17 through 40 are blanks) written in binary ("odd" parity) mode.

Example:

△LABEL 2,MT3W,'FILENAME'.

This message will cause MONARCH to write a level 2 MONARCH ID record on magnetic tape unit 3 on the W buffer. Characters 9 through 16 of this record will contain FILENAME.

Normally, the only labels of which RAD MONARCH is cognizant are those in the Directory (D file) which were entered at the time the RAD system was generated (see Appendix H). Since, however, the BO file immediately follows the S file, additional binary programs may be added to the system (S file) by issuing a LABEL control message before each such program is generated. After the last program, the message

△LABEL 1,BO,'SYSEND'

should be issued to cause the end of the system to be redefined as following the last location in the BO file.

DISPLAY **SHOW** DISPLAY (or SHOW) allows the user to produce the contents of one or more memory locations on the console typewriter and is operative only if the MONARCH monitor is in control.

△DISPLAY P_1 THRU P_2.

△SHOW P_1 THRU P_2.

P_1 (required) must be a legitimate memory address. If it is the only parameter given, it designates the one location whose contents are to be displayed. If three parameters (P_1, THRU, and P_2) are given, P_1 is the beginning address of the sequential memory locations whose contents are to be displayed.

THRU (optional). When the contents of more than one memory location are to be displayed, the second parameter of the control message must be the word THRU.

P_2 (optional). When present, this parameter must be a legitimate memory address which is equal to or greater than the value of P_1. P_2 specifies the ending address of the sequential memory locations whose contents are to be displayed.

After interpreting the parameters, MONARCH converts the contents of each designated memory location to octal and types each value, together with its octal address on typewriter 1 on the W buffer.

Examples:

If location 037777 contains zero, the message

△DISPLAY 037777.

will cause the following to be typed:

037777 = 00000000

The message

△SHOW 0164 THRU 0174.

will cause the address and contents of each of the 9 locations specified to be typed.

POSITION This control message enables a user to position a magnetic tape or the RAD at a given logical file (identified by a MONARCH level 1 ID record only – not a level 2 record; ID records are described under "System Update Routine" in Section 4).

△POSITION P_1,P_2.

P_1 must be a legal magnetic tape unit or logical unit designation on an existing buffer; e.g., MT3W specifies magnetic tape unit 3 on the W buffer, BI specifies the tape or RAD-assigned BI file.

P_2 is a literal consisting of up to eight alphanumeric characters. Trailing blanks (60_8) are supplied if fewer than eight characters are given. The value of this parameter is used as the search key.

To position the specified magnetic tape or logical unit at the desired logical file, the MONARCH Search subroutine reads successive records (in a forward direction) until a level 1 MONARCH ID record is found that contains, in characters 9 through 16, the given file identification (P_2). The tape is read in binary ("odd" parity) mode, and the maximum ID record length is assumed to be 40 words (160 characters). The search is terminated as follows:

1. If characters 9 through 16 of a level 1 ID record contain the logical file ID specified as the second parameter, control is returned to MONARCH to obtain the next control message. The tape will be positioned in the interrecord gap which follows the ID record.
2. If characters 9 through 16 of a level 1 ID record contains SYSEND$_{\wedge\wedge}$, a message is typed indicating that the specified logical file was not found. Then control is returned to MONARCH to obtain the next control message.
3. If characters 9 through 16 of a level 1 ID record contain neither SYSEND$_{\wedge\wedge}$ nor the specified logical file ID, the search is continued until either condition 1 or condition 2 is satisfied, or until the computer operator intervenes.

Access to RAD-assigned S or BI files is random, since all level 1 names are retained in a directory.

Examples:

△POSITION BI,'FILEIDEN'.

△POSITION MT2W,'FILEIDEN'.

Input from cards, the latter message will cause MONARCH to position magnetic tape unit 2 on the W buffer in front of the first record, following the level 1 MONARCH ID record that contains FILEIDEN in characters 9 through 16. If input is from the typewriter, this message would appear as

△POSITION MT2W,@FILEIDEN@.

(See discussion on "Literal Parameters".)

REWIND The REWIND control message causes MONARCH to rewind a specified magnetic tape unit or perform a pseudo-rewind on a specified disc file.

△REWIND P_1

P_1 must be a legal magnetic tape unit or logical unit designation on an existing buffer; e.g., MT3W or BO. If the logical unit specified has been assigned to a disc file, MONARCH performs a pseudo-rewind so that a subsequent read or write operation will start at the beginning of the file.

A disc file may be specified in a REWIND message only as a logical unit, such as SI or BO; it is not permissible to specify the disc file directly with a symbolic parameter such as DF1B.

Example:

△REWIND MT0W.

will cause MONARCH to rewind magnetic tape unit 0 on the W buffer.

SKIPFILE[†] **SKIPREC** These control messages cause MONARCH to skip files or records in a forward direction on a specified magnetic tape unit (see also, BACKFILE and BACKREC). The magnetic tape unit and number of files or records to be skipped are specified by the control message parameters.

△SKIPFILE P_1,P_2. (skip files)

△SKIPREC P_1,P_2. (skip records)

P_1 must be a legal magnetic tape unit designation on an existing buffer; e.g., MT2W designates magnetic tape unit 2 on the W buffer.

P_2 specifies the number of files or records to be skipped.

After interpreting the parameters, MONARCH moves the specified mangetic tape forward the indicated number of files or records. If an EOF mark is encountered during a skip record process, the tape will stop. Thus, the tape will be positioned immediately after the EOF.

Examples:

△SKIPFILE MT1W,5.

This message causes MONARCH to skip forward 5 files on magnetic tape unit 1 on the W buffer.

[†]SKIPFILE, BACKFILE and WEOF descriptions have not been altered, therefore, they still apply only to magtape.

△SKIPREC MT3Y,10.

This message causes MONARCH to skip forward 10 (12_8) records on magnetic tape unit 3 on the Y buffer.

BACKFILE **BACKREC** These control messages have a function similar to that of SKIPFILE and SKIPREC; however, with BACKFILE and BACKREC the magnetic tape is moved in a backward direction.

△BACKFILE P_1,P_2.

△BACKREC P_1,P_2.

P_1 and P_2 have the same interpretation as for SKIPFILE and SKIPREC.

After interpreting the parameters, MONARCH moves the specified magnetic tape backward the indicated number of files or records. If an EOF mark is encountered during a skip record process, the tape will stop. Thus, the tape will be positioned before the EOF mark. During a BACKFILE operation the user must backfile one plus the number of files to be skipped to return to a particular file of interest.

Examples:

△BACKFILE MT0Y,12.

This message causes MONARCH to skip backward 12_{10} (14_8) files on magnetic tape unit 0 on the Y buffer. When the 12th file is skipped MONARCH will read forward over the 12th EOF mark so that the tape will be positioned to read the first record of the next file.

△BACKREC MT2W,3.

This message causes MONARCH to skip backward 3 records on magnetic tape unit 2 on the W buffer.

WEOF The WEOF control message directs MONARCH to write an end-of-file (EOF) mark on the specified tape.

△WEOF P_1.

P_1 must be a legal magnetic tape unit designation on an existing buffer or a legal paper tape punch unit designation on an existing buffer. That is, P_1 may take the form MTub or PPxb, where u must be within the range $0 \leq u \leq 7$, b is W or Y, and x is 1 or 2.

Examples:

△WEOF MT3W.

This message causes MONARCH to write an end-of-file mark (17000000) on magnetic tape unit 3 on the W buffer.

△WEOF PP1Y.

This message causes a special end-of-file mark (17170000) to be punched on paper tape unit 1 on the Y buffer. (This is to facilitate the loading of a FORTRAN-compiled program into the FORTRAN library on the system tape (see Section 4, "System Update Routine").

BOOTLOAD This control message directs MONARCH to produce an absolute or relocatable bootstrap[†] on paper tape or magnetic tape as specified.

△BOOTLOAD P_1,P_2.

P_1 must be ABS for absolute or REL for relocatable.

P_2 must specify the magnetic tape unit or paper tape unit on which the bootstrap is to be produced; e.g., MT2W, PP1W for 900 series; or MT2A, PP1A for 9300.

MONARCH interprets the control message and produces the requested bootstrap on the specified tape. These bootstraps can load programs assembled by SYMBOL or META-SYMBOL. Although the bootstrap may be produced on any paper tape or magnetic tape unit, it can be read from only paper tape unit 1 on the W buffer or magnetic tape unit 0 on the W buffer, on 900 Series Computers. Substitute corresponding A channel for 9300 Computers.

Examples:

△BOOTLOAD ABS,PP1W.

This control message directs MONARCH to punch an absolute bootstrap on paper tape punch 1 on the W buffer.

△BOOTLOAD REL,PP2W.

This message causes MONARCH to punch a relocatable bootstrap on paper tape punch 2 on the W buffer.

△BOOTLOAD ABS,MT1W.

This message directs MONARCH to write an absolute bootstrap on magnetic tape unit 1 on the W buffer. To load the object program, dial the tape unit number to zero and execute a fill from magnetic tape:

```
Set (X1) = -7 (77777771)
EOM  03610   (0 02 03610)
WIM  2       (0 32 00002)
BRU  1       (0 01 00001)
```

[†]Descriptions of the bootstrap routines are available from the XDS Program Library: 900 Series Paper Tape Absolute Bootstrap, catalog number 020020; 900 Series Paper Tape Relocatable Bootstrap, catalog number 000019; 9300 Paper Tape Relocatable Bootstrap, catalog number 600001. The magnetic tape bootstrap routines are modified versions of the MONARCH boostrap loader.

Set the contents of register A to 0 32 00002.

Set the contents of register C to 0 35 00001.

Set the RUN-IDLE-STEP switch to STEP.

Press START.

Press FILL switch, which sets (X1) to -7. If program is relocatable, set (A) = relocation bias.

Set the contents of register C to 0 02 03610.

Set the RUN-IDLE-STEP switch to RUN.

Note: To load the 925/930/9300 magnetic tape bootstrap, execute a magnetic tape FILL procedure. For 9300 computers, use appropriate channel label (i.e., A, B, C, or D).

CARDTAPE The CARDTAPE control message causes MONARCH to select the designated card reader, to read cards in symbolic, encoded, binary, MONARCH identification, and control message formats and to write them on the magnetic tape specified.

△CARDTAPE P_1, P_2.

P_1 must be a legal card reader designation on an existing buffer; e.g., CR1W. This parameter specifies the card unit from which the cards are to be read.

P_2 must be a legal magnetic tape unit designation on an existing buffer; e.g., MT2W. This parameter specifies the tape unit on which the information is to be written.

Cards are read from the card reader specified and are written on the designated magnetic tape. Binary, encoded, MONARCH identification, and control cards are written in binary; all other cards are written in binary-coded decimal (BCD). When a △EOF card is read or a card reader end-of-file is detected, an end-of-file (EOF) mark is written on the magnetic tape, and control is returned to MONARCH. If successive files are to be written on tape, each file must be preceded by a CARDTAPE control message, including the necessary parameters.

Example:

The control message

△CARDTAPE CR1W,MT3W.

directs MONARCH to read cards from card reader 1 on the W buffer and to write them on magnetic tape unit 3 on the W buffer. When the read is completed, an EOF mark will be written on the magnetic tape.

EOF The EOF control message signifies the end of a logical file and transfers control to MONARCH.

△EOF.

There are no parameters for this message. It is recognized by the "action" routine that processes the CARDTAPE control message (see Section 3 for explanation of action routines). The EOF message is also recognized by the FORTRAN and META-SYMBOL processors.

SYSTEM MAINTENANCE

UPDATE This control message causes MONARCH to load the System Update Routine and to transfer control to it.

△UPDATE P_1.

P_1 (optional). When present, this parameter indicates blocking mode operation and must be within the range

$$41_{10} \leq P_1 \leq 256_{10}$$

The absence of the parameter indicates normal mode operation.

See Section 4 for a description of the update routine, its operating modes, and the control messages required for its use.

3. THE MONARCH SYSTEM

MONITOR

The major portion of the MONARCH system is the monitor routine. This routine accepts control information which, among other things, may include a request to load and execute a specified standard system routine. The monitor performs its function between jobs and does not exercise control over the execution of a program once the program has been loaded and control has been transferred to it.

The monitor consists of a number of subroutines. One of these subroutines is the system tape search routine. This is the subroutine that searches the system tape for a given routine name (see POSITION control message). Another monitor subroutine analyzes and interprets the contents of the control messages that convey control information to the monitor. It also converts the parameters in control messages to standard internal form.

Other subroutines, called "action" subroutines, perform the functions associated with specific control messages. For example, the action subroutine associated with the ASSIGN message modifies the contents of MONARCH's unit assignment table, based on the values of the parameters in the ASSIGN message. Another action subroutine, associated with the LOAD message, controls the searching of tape files for specified object programs and calls on the MONARCH loader to load these object programs. Additional subroutines employed by the monitor include those which perform input/output for MONARCH.

Part of the monitor, called the resident, remains in core memory during program execution. The resident consists of the monitor bootstrap routine (QBOOT), the unit assignment table, the error and job switches, the octal dump routine (QDUMP), and the symbol table dump driver (see Appendix B for a complete description of memory layout). The resident occupies the last 132_8 locations in memory. In RAD MONARCH, the resident also includes a RAD file management routine of approximately 500 memory locations, which handles all RAD I/O communications in a file-oriented manner. Memory space occupied by the remaining subroutines comprising the monitor and by other routines in the MONARCH system (such as the MONARCH loader) is available for use by the program being executed. The last available location in core is one cell below QDUMP (however, the term QDUMP-1 is illegal in META-SYMBOL language), or one below QBOOT for users who do not require the dump.

STANDARD SYSTEM ROUTINES

Standard system routines are those that exist on a MONARCH system tape and that can be loaded and executed by supplying an appropriate control message to the MONARCH monitor. Some existing system routines, as well as the necessary and desirable characteristics of potential system routines, are described below.

Certain of the standard system routines must be present on any MONARCH system tape. These programs comprise the minimum operable MONARCH system:

1. The monitor. This routine is the heart of the operating system.
2. The MONARCH loader. The monitor uses this routine to load standard system routines from the system tape and to load previously assembled programs presented by the MONARCH user. The MONARCH loader is described later in this section.
3. The MONARCH bootstrap loader. This routine performs the function of loading the MONARCH loader and the MONARCH monitor and precedes all other system routines on a MONARCH system tape. This is the routine that is called in for execution by the monitor bootstrap (QBOOT).

Certain system routines, while not essential for a minimum MONARCH system, enhance the usefulness and flexibility of any MONARCH system.

1. The MONARCH system update routine. With this routine, the user can create new MONARCH system tapes or update existing system tapes. This routine is described in Section 4.

2. The standard input/output subroutines. These subroutines are used by other system routines to perform required input/output functions. These I/O subroutines, which can be selectively loaded on an "as needed" basis, are

   Line Printer Output Subroutine (PRINT)

   Magnetic Tape Input/Output Subroutine (MTAPE)

   Card Read/Punch Subroutine (CDRP)

   Paper Tape/Typewriter Input/Output Subroutine (PTYIO)

   The action subroutines for a given system routine examine the parameters of the control message and the unit address codes of those MONARCH unit assignment table entries that represent input/output functions to be performed and, finally, direct the loading of the I/O subroutines needed. The MONARCH update routine relies on this feature to provice the input/output subroutines needed to perform a specific update run.

3. The META-SYMBOL assembly system. Presence of this routine provides a powerful and flexible assembly language and processor.

4. The FORTRAN II system. Presence of these routines enables the MONARCH user to use the full capabilities of the XDS 900 Series FORTRAN II Compiler, Loader, and Run-Time Package.

5. The ALGOL system. This system, which is available on request, operates on both XDS 900 Series and the 9300 Computers. It includes the ALGOL Compiler, Loader, and Run-Time package.

See "Automatic Selective Loading from the MONARCH Library" at the end of this section for a description of the MONARCH library, another optional MONARCH feature that can contribute greatly to the usefulness and efficiency of a MONARCH system.

MONARCH is designed to facilitate the incorporation of additional system routines as needed. The user can include in a MONARCH system any routine that meets the following requirements:

1. The routine must exist (on cards or paper tape) in XDS standard binary language.

2. Its memory space requirements must be such that it (or a special loader which precedes it on the system tape) can be loaded by the MONARCH loader.

3. It must be written in a manner that is consistent with run termination as described in the paragraph, "Termination of a Run," below.

Certain other characteristics, while not essential, ease the job of incorporating new system routines and render these routines more useful in the MONARCH environment:

1. The routine should be one that can be assembled as a series of one or more relocatable programs by SYMBOL or META-SYMBOL.

2. It should be written in such a way that any "parameters" required for its initialization can be easily supplied in the form of MONARCH control message parameters (see "Control Message Parameters" in Section 2).

3. The routine should be written to obtain unit and channel assignments for all its input/output functions from the MONARCH unit assignment table.

TERMINATION OF A RUN

When a program being executed under MONARCH reaches a normal conclusion, it should transfer control to the monitor bootstrap in core memory (location 1) rather than execute a HALT instruction; the monitor bootstrap initiates the reloading of the MONARCH loader and the MONARCH monitor. The monitor then attempts to read a new control message from the current control medium and in this way proceeds to the next job without the necessity for manual intervention. The monitor bootstrap is part of the MONARCH resident.

When a program being executed under MONARCH detects a program or computer error that makes it inadvisable to continue program execution, it should give whatever error indication is suitable and transfer control to the monitor bootstrap. This routine initiates the reloading of the MONARCH loader and MONARCH monitor, and then the console operator can decide whether or not to continue with the next job or function in a batched job stack or to take some alternative action.

When the console operator decides that a program being executed has halted inadvertently or is otherwise malfunctioning, he can stop the program, clear the registers and restart by manually transferring control to a restart location in the monitor bootstrap. The monitor bootstrap initiates the reloading of the system, and then the monitor attempts to obtain the next control message. At this point, the operator can decide whether or not to continue with the next job in the batched job stack or to execute some other system function.

The normal restart procedure is to execute a branch to location 1. Location 1 normally contains an unconditional branch to the monitor bootstrap in upper memory. The routine that is loaded by the monitor bootstrap is the MONARCH bootstrap loader, which precedes all other routines on the system tape. The MONARCH bootstrap loader in turn loads the MONARCH loader and the MONARCH monitor.

LOADER

The primary function of this routine is to load the user's object programs. It is also called upon by the MONARCH monitor to load from the system tape standard system routines such as META-SYMBOL, the system update routine, etc. The loader (including QDUMP, QBOOT, and UAT) occupies upper core.

The loader is capable of loading binary object programs in the format produced by SYMBOL and META-SYMBOL. A series of programs to be loaded may be absolute or relocatable and may contain:

1. External label references and/or definitions.

2. External Programmed Operator (POP) references and/or definitions[†].

3. Blank COMMON references and a definition.

Blank COMMON references should be preceded by a blank COMMON definition, but external references

[†]The capability of handling POP items is not included in 9300 MONARCH loader since the 9300 does not have Programmed Operators. All other capabilities of the 900 Series MONARCH loader are included in 9300 MONARCH loader.

and definitions (label or POP) need not be supplied in any particular order.

The term "program" in this description of the MONARCH loader means a sequence of:

1. One or more data records (record type 0) and/or
2. One or more external references or definition records (record type 1) and/or
3. One or more Programmed Operator references or definition records (record type 2) and
4. An end record (record type 3) with or without a transfer address.

See Appendix F, "XDS Standard Binary Language," for a description of the record formats accepted by this loader. Note that the MONARCH loader does not accept labeled COMMON definitions or references (record type 2, item types 1 and 3) and treats labeled COMMON references as format errors.

The last (or only) program in a series of programs to be loaded must have an end record (type 3) with a transfer address, and all programs preceding it must have end records without transfer addresses.

If there are unsatisfied label or POP references at the time the end record with a transfer address is encountered, the loader attempts to satisfy these by selectively loading the appropriate subroutines from the MONARCH library. If this is unsuccessful, the loader automatically outputs (on typewriter 1 on the W buffer if Breakpoint 1 is reset or on line printer 1 on the W buffer if Breakpoint 1 is set) the unsatisfied labels or POP references. Following this information, the loader outputs the symbol table if requested to do so (see LOAD control message in Section 2). Then the computer halts. After determining whether the missing definitions will affect the run, the user may elect to execute the program by simply clearing the halt (i.e., move the RUN-IDLE-STEP switch from RUN to IDLE to RUN) or to abort the run by transferring manually to the bootstrap (i.e., to location 1).

Programs may be loaded from punched card, magnetic tape, or paper tape units attached to either the W or Y buffer. In RAD MONARCH, programs may also be loaded from RAD files. The input/output subroutines within the 900 Series MONARCH loader use neither interrupts nor interlace. The 9300 I/O handlers use interlace and interrupts for all I/O operations. Any I/O operation performed by 9300 MONARCH that does not use the I/O handlers does not use interrupts. Reading and searching of the binary input medium by the 9300 MONARCH loader uses interlace but not interrupts. The symbol table typeout routine, the line printer octal dump routine, and the punching of the absolute bootstrap on paper tape do not use interrupts or interlace.

UNIT ASSIGNMENT REQUIREMENTS

When a LOAD control message (see Section 2 for a detailed description of this message) is issued to MONARCH, the unit assignment table[t] is assumed to contain the following information:

1. QMSG contains the unit and channel designation for the peripheral device that is to furnish MONARCH control messages. (QMSG is set by the C control message.)
2. QBINI contains the unit and channel designation for the peripheral device that is to furnish input (programs) to the loader. The unit must be a card reader, a magnetic tape unit, a paper tape reader, or the RAD file.

The following sequence of MONARCH control messages illustrates one means of setting the unit assignment table and requesting the MONARCH loader to load one or more programs:

```
△C TY1W.
△ASSIGN BI=CR1W.
△ENDJOB.
△LOAD 010000, GO.
```

The control messages are to be input from typewriter 1 on the W buffer, and the binary input is to be read from card reader 1 on the W buffer. The loader is to load the first (or only) program with a relocation bias of 10000_8 and is to transfer control to the location specified on the END record of the last program without stopping and without a symbol table printout. If any references are unsatisfied, a list of the unsatisfied references is typed on typewriter 1 on the W buffer. Then the loader halts. To continue, the operator clears the halt.

STORAGE ALLOCATION

When the MONARCH system is loaded, the MONARCH loader is stored in upper core, occupying locations $X5441_8$[tt] through $X7777_8$ (X=1, 2, or 3). The loader's symbol table (external label definition entries) initially occupies memory from $X5440_8$[tt] through $X5276_8$[tt]. As each additional external symbol is inserted in the symbol table, it occupies the three memory locations immediately below the last symbol table entry. Thus, the loader and its symbol table occupy that amount of upper core required by the loader routine itself and the external symbol entries.

At the time a request to load a user's program is initiated, the loader symbol table contains external table definition entries that allow external references to locations

[t] The external labels mentioned here are discussed in Appendix A, "The MONARCH Unit Assignment Table".

[tt] Approximately 1000_8 less in RAD MONARCH

within the resident portion of MONARCH. Those entries are defined in Appendix A.

The loader gives an appropriate error indication whenever a new entry is to be made in the symbol table that would "overlay" programs or data already stored in memory by the loader. This conditions is also referred to as symbol table overflow. See Section 6 "Operating Procedures."

THE LOADING PROCESS

Relocation and Data Records

A data record (record type 0) contains instructions and/or data to be stored in memory by the loader. Each data record contains a load address that is either the relative or absolute memory location in which the first data word (an instruction or a constant) is to be stored. The word in the data record containing the load address also contains an indicator specifying whether or not the current load relocation bias is to be added to this load address to obtain an effective load address (i.e., whether or not the data record contains "relocatable" data words) for the program.

The effective load address determines the location in which the first data word is stored, and successive data words in a data record are stored in consecutive memory locations following the first data word.

Before each data word in a program is stored its binary value may be modified as required (e.g., by load relocation modifier word; see Appendix F "XDS Standard Binary Language.")

External Label References and Definitions

The loader is capable of handling (resolving) symbolic cross-references between separately assembled and/or compiled programs. External reference and definition items in type 1 binary records provide the loader with the information needed to "link" together two or more separately assembled or compiled programs.

During the loading process, the loader maintains a symbol table of external label definitions and unsatisfied external references. There is no restriction on the order in which the definition of a label and the references to it appear in the input to the loader. The definition of a label may precede, or follow, some or all of the references to it. Note that it is permissible for any number of programs to contain references to a given label, provided that one program being loaded contains an external definition item for that label.

When the loader encounters an external definition item, it searches the symbol table for a previous definition of that label in the table; if there is one, the loader discards the new definition. If the search reveals that the label is already in the table as an unsatisfied reference, the loader uses the definition to satisfy all the references to that label and replaces the unsatisfied reference item in the table with the definition item. However, if that label does not occur in the symbol table (as a reference or as a definition), the loader inserts the external definition item in the symbol table.

When the loader encounters an external reference item, it searches the symbol table to see if it already contains an external reference item for that label; if so, the new external reference is associated with the existing table entry. If the search reveals that the label is already included in the table as an external definition the loader uses the definition to satisfy all the references to that label. However, if that label does not occur in the symbol table (as a reference or as a definition), the external reference item is inserted in the symbol table.

External Programmed Operator References and Definitions

The loader is capable of satisfying references to internal and external Programmed Operator (POP) definitions. External POP definition items, external reference items, and internal POP definition items provide the loader with the information needed to:

1. Satisfy external and internal POP references.
2. Maintain external POP reference and definition items in the loader's symbol table.
3. Construct a Programmed Operator transfer table in cells 0100_8 through 0177_8.

An "internal" POP definition is one that is recognized only within the scope of the program in which it occurs. No entries are made in the loader's symbol table for internal POP definitions or references.

Many of the loader functions performed in the processing of external POP references and definitions are also performed (by the same loader subroutines) for external label references and definitions. In particular, the functions of insertion and replacement of symbol table entries and the handling of duplicate definitions are the same for both external label and external POP items.

AUTOMATIC SELECTIVE LOADING FROM THE MONARCH LIBRARY

Provision is made for automatic search of the MONARCH library when an end record with a transfer address is encountered and unsatisfied label or POP references exist. This library normally consists of a collection of frequently used closed subroutines and Programmed Operator subroutines. The loader automatically loads any such subroutines when it encounters an external reference in a program (or group of programs) being loaded. This relieves the programmer of the burden of including such

subroutines in the program decks (or tapes) he furnishes to the loader. For example, the programmer may wish to employ certain input/output subroutines available on the program library and refer to them symbolically in his main program. Note that the loader first attempts to satisfy all external references from the definitions supplied in the program decks (or tapes) furnished by the programmer, and only when this attempt is unsuccessful does it attempt to satisfy these references by loading programs from the program library. The following paragraphs describe the procedures employed to access programs in the program library.

When the loader is loading a previously assembled program and there are external references that have not been satisfied when the end record with a transfer address is encountered, the loader causes the monitor to locate the MONARCH library on the system tape. The loader then enters a special mode in which it searches the external definition in each library program in succession. When it encounters a library program which satisfies at least one such reference, it loads this program; then, if there are still some unsatisfied references, it continues to search the program library. To avoid "backtracking" when switching from "search" to "load" mode, the definitions from each library program being examined are temporarily added to the table of external definitions and references maintained by the loader. Note that the records containing external label definitions and external Programmed Operator definitions must precede all other information in a binary object program; hence, only these definitions have to be saved in memory to enable the loader to switch from "search" to "load" mode without rereading records from the system tape.

If a given library program does not contain a definition for any of the unsatisfied references, its definitions are removed from the table and the next library program is examined. If there are still unsatisfied external references when the end of the program library is encountered, the loader indicates that an error condition exists.

The loader employs an entirely similar method in attempting to obtain definitions for any unsatisfied Programmed Operator references. If these references cannot be satisfied from the Programmed Operator definitions on the system tape, the loader indicates that an error condition exists. The library search for Programmed Operators is concurrent with the search for external definitions of labels (i.e., the Programmed Operator definitions are part of the program library). Since MONARCH makes only one pass through the library, no routine on the library can call a routine preceding it.

4. PROGRAMMING WITH MONARCH

This section describes MONARCH subroutines that can be referenced from the user's program. Also discussed here is the MONARCH System Update Routine which is used to create new MONARCH system tapes and to update existing system tapes.

OCTAL DUMP ROUTINE

A Line Printer Octal Dump Routine with zero suppression is incorporated in the MONARCH loader. This routine resides in the MONARCH resident portion and may be referenced internally or from the console. When the dump routine is to be referenced in a program, the following calling sequence must be assembled as part of the user's program:

α	BRM	QDUMP
$\alpha + 1$	PZE	P_1
$\alpha + 2$	PZE	P_2
$\alpha + 3$	return	

α represents any location. When the dump is completed, control is returned to the user's program at $\alpha + 3$.

QDUMP is the externally defined label for the entry point of the routine and must be an externally defined symbol in the user's program.

P_1 specifies the beginning address of the sequential memory locations whose contents are to be printed.

P_2 designates the ending address of the sequential memory locations whose contents are to be printed. The address represented by P_2 must be equal to or greater than that of P_1.

P_1 and P_2 may be numeric or symbolic (external).

To reference the dump routine from the console, set the contents of the registers as follows:

A = Beginning address (see P_1 above)
B = Ending address (see P_2 above)
C = For 900 Series Computers:

RAD		TAPE
$0\ 01\ 17751_8$	8K memory	0 01 17650
$0\ 01\ 23751_8$	10K memory	0 01 23650
$0\ 01\ 27751_8$	12K memory	0 01 27650
$0\ 01\ 33751_8$	14K memory	0 01 33650
$0\ 01\ 37751_8$	16K memory	0 01 37650

For 9300 Computers:

	TAPE
8K memory	0 01 17646
12K memory	0 01 27646
16K memory	0 01 37646

Position the RUN-IDLE-STEP switch to RUN.

To continue dumping when the computer halts, reset the A and B registers to the desired addresses and clear the halt.

Note: On 900 Series Computers the octal dump routine operates only on machines with memory of 16K or less.

SYMBOL TABLE TYPEOUT ROUTINE

The Symbol Table Typeout Routine produces a list of all symbols and the location to which each is assigned. If Breakpoint 1 is reset, the symbol table is output on typewriter 1 on the W buffer; if Breakpoint 1 is set, the table is output on line printer 1 on the W buffer.

To reference the symbol table typeout routine, set the contents of the 900/9300 C register to BRU TYPSY5:

RAD		TAPE
C = $0\ 01\ 17750_8$	8K memory	0 01 17734
= $0\ 01\ 23750_8$	10K memory	0 01 23734
= $0\ 01\ 27750_8$	12K memory	0 01 27734
= $0\ 01\ 33750_8$	14K memory	0 01 33734
= $0\ 01\ 37750_8$	16K memory	0 01 37734

Position the RUN-IDLE-STEP switch to RUN. The computer will halt after the typeout. The user may clear the halt, causing a transfer to QBOOT which reloads the MONARCH system, or he may transfer manually to any location (i.e., insert in the C register a BRU to the desired location).

The output produced by this routine consists of two columns; the first contains the symbol, left justified with trailing blanks, and the second contains the memory location to which the symbol was assigned. (See Appendix C for the method used in forming the addresses.)

It is suggested that, upon receiving the MONARCH system of programs, the user have a symbol table typeout produced. The listing produced will be useful when reference to a specific address within the MONARCH system of programs is required. A sample of such a listing appears in the discussion "Monarch Loader's Symbol Table" in Section 6.

OCTAL CORRECTION ROUTINE

Corrections may be made to a program at load time via octal correction cards. Such cards are placed just before the end card in the binary deck. The format of correction cards is

$P_1 P_2 P_3 \ldots P_n$.

P_1 is an address, consisting of up to five octal digits, that specifies the location of the first correction. P_1 may start in any column. If it does not start in column 1, a character string (P_2) may precede P_1.

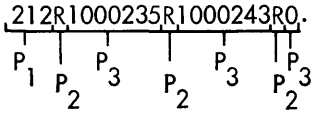
P_2 is any character string (one or more characters) not containing an octal digit. If the first character of the string is an R, the preceding octal number is assumed to be relocatable. If the first character is any other character, the preceding octal number is assumed to be nonrelocatable.

P_3 is the octal correction (one or more octal digits). If more than eight digits appear, only the last eight digits read (i.e., the eight low-order digits) are accepted.

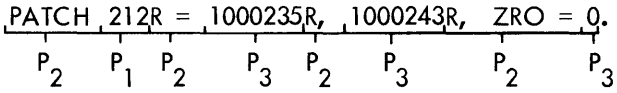
Succeeding octal numbers are stored in consecutive locations relative to location P_1. Continuation from one card to the next is not permitted. A period may serve as a terminator but is not required. If a period is used, any information following the period is treated as comments and is not processed.

Examples:

Assume that a program is to be loaded and that relative locations 212, 213, and 214 are to be changed to contain BRU 00235, BRU 00243, and HLT 00000, respectively. The necessary corrections could be written as



The same changes could also be written as



If the program is loaded with a relocation bias of 1000_8, these locations will contain:

Location	Contents
1212	01000235
1213	01000243
1214	00000000

LOADER ROUTINE

The MONARCH loader can be executed either via the appropriate MONARCH control messages or directly as a closed subroutine. The user's program can transfer to the loader by executing the instruction

```
BRM   QSYLDR
```

QSYLDR is the externally defined label for the entry point to the loader and must be an externally defined symbol in the user's program. The loader assumes that the A register contains the load relocation bias to be used and that the B register contains the binary value of the loader option parameter (see LOAD control message in Section 2).

The loader commences execution by reading a record from the previously designated binary input medium and checking the first word (control word) of the record to see whether or not it is a valid binary record. Next, the record type code (see Appendix F) of the control word is used to indicate the appropriate subroutine within the loader for processing that type of record.

When the loader has processed a record, it continues by reading in the next record unless the record just processed is an end record (record type code of 3). When an end record without a transfer address has been processed, the loader, depending upon the value of the loader option parameter, does one of three things:

1. Halts with:

   C = 0 20 22222

   A = load relocation bias to be used for loading the next program (unless changed manually by the console operator)

   B = indeterminate

2. Returns control to the program that called the loader (by executing a BRR QSYLDR), with:

   B = initial load relocation bias plus program length

   A = loader option parameter

3. Sets the load relocation bias equal to its previous value plus the length of the current program (as specified in the end record) and continues loading records.

When an end record with a transfer address is encountered, a library search is made to satisfy references. Then any indicated relocation is performed on the single data word in the end record, and the loader, depending on the value of the loader option parameter, does one of two things:

1. Halts with:

   C = transfer word as modified by any relocation indicative present in the end record

   A = loader option parameter

   B = load relocation bias

2. Executes the transfer word after performing any indicated relocation of the address field. Normally, the transfer word is a BRANCH UNCONDITIONAL instruction (BRU), whose address is determined by the value of the expression in the operand field of a SYMBOL or META-SYMBOL END line.

The loader does not "initialize" unused memory locations with "background" values (e.g., halt instruction). The only memory locations modified by the loader are

1. Those within the locations occupied by the loader and its input subroutines.
2. Those locations pre-empted by the loader for its symbol table.
3. Locations in which the loader is explicitly directed to store instructions or constants (i.e., data words supplied to the loader in data records).

SYSTEM UPDATE ROUTINE

This routine is used to create new MONARCH system tapes and to update existing system tapes. The functions of insertion and deletion of both system programs and data files (including the MONARCH monitor and the MONARCH loader) are provided.

Since each routine on the system tape (except the bootstrap loader and the MONARCH loader) is preceded by an identifier (a MONARCH ID record), insertions and deletions are indicated to the update routine in terms of those identifiers. The MONARCH loader has the identifier LOAD associated with it even though no MONARCH ID record actually precedes that routine on the system tape. The bootstrap loader is automatically recorded on a new system tape as the first record on the tape.

It may be necessary to include, as standard system routines, programs whose memory space requirements preclude the use of the MONARCH loader to load them at execution time. Such programs should be preceded on the system tape by a special purpose loader that is capable of loading the system routine in question from the system tape. It is this special purpose loader which is loaded, and executed, under control of the MONARCH loader when a MONARCH control message calls for execution of the system routine in question.

All programs on a MONARCH system tape, with the exception of the bootstrap loader, must be in either XDS standard binary language or FORTRAN binary language. Data files to be recorded on a MONARCH system tape must be presented to the update routine in either of these formats or else in XDS encoded symbolic format. The only other form of information permissible on a MONARCH system tape is MONARCH ID records. Such routines and data files must be presented to the update routine on punched cards, paper tape, or magnetic tape.

It is also noted that a standard system routine (e.g., the META-SYMBOL assembler) may itself consist of several independently assembled subprograms and only the first of these is preceded on the system tape by a level 1 MONARCH ID record. Hence, the MONARCH loader automatically loads any subprograms following the first subprogram until it encounters either the next level 1 MONARCH ID record or a binary end record (type 3) with a transfer address. The MONARCH update routine acknowledges this type of program structure when performing insertion and deletion functions in the course of writing a new system tape. Any or all of the subprograms of a standard system routine may be preceded on the system tape by a level 2 MONARCH ID record to permit insertion or deletion of individual subprograms by the MONARCH update routine. These level 2 MONARCH ID records are ignored by the MONARCH loader when loading a standard system routine for execution.

The update routine produces a typewriter or line printer listing of the MONARCH ID records (level 1 and level 2) associated with all routines and all data files written on a new system tape. These ID records appear on the listing in the order in which they exist on the new system tape. This listing should be preserved for use as the basis for constructing update control messages for the next system update run. (See examples later in this section.)

Routines to be inserted by the MONARCH update routine must be preceded by a level 1 and/or level 2 MONARCH ID record and must be presented in the order in which they appear on the new system tape. COPY messages (i.e., control messages to the update routine) must be presented in the order in which they are to be executed. No reordering of update input is performed.

THE UPDATE CONTROL MESSAGE

When the update routine is loaded for execution by the MONARCH loader, the MONARCH unit assignment table is assumed to contain the following information:

1. QMSG (control message input unit) contains the unit and channel designation for the peripheral device that is to furnish MONARCH control messages and update control messages. The unit must be a card reader or a typewriter.
2. QSYSU (the update input unit UI) contains the unit and channel designation for the peripheral device that is to furnish any programs or data files to be inserted in the new system. The unit must be a card reader, a paper tape reader, or a magnetic tape unit.
3. QSYST (system scratch tape X1) contains the unit (magnetic tape only) and channel designation for the peripheral device upon which the new system tape will be written.

4. QSYS (sytem tape S) is assumed to specify magnetic tape unit 0 on channel W and the old system tape is assumed to be mounted on that unit. However, if both QSYS and QSYST contain the same unit and channel designation, the update routine assumes that no old system tape is present.

5. QSYMO (list output unit LO) is assumed to specify whether MONARCH ID records will be listed on the line printer (LO=LP) or on the typewriter (LO=TY).

It should be noted that the update control message input unit (QMSG) and update input unit (UI) assignments may differ. For example, QMSG may be assigned to the card reader (CR) and update input (UI) may be assigned to a paper tape, magnetic tape, or card reader.

The following sequence of MONARCH control messages illustrates the means of setting up the unit assignment table and calling in the update routine for execution.

```
△C          CR1W.
△ASSIGN     S=MT0W, X1=MT1W, UI=CR1W.
△ASSIGN     LO=LP.
△UPDATE     256.
```

The MONARCH update routine has two modes of operation: the normal mode and the blocking mode. In the normal mode all records are written as 40-word records. In the blocking mode all records (MONARCH ID records excluded) are written as blocked records. The maximum length of a blocked record is determined by the blocking number which is a parameter of the update control message ($41_{10} \leq$ blocking number $\leq 256_{10}$). If the blocking number is less than 41, 41 is automatically used. If the blocking number is greater than 256, 256 is automatically used.

A blocked record consists of a 1-word block sentinel (defined below) followed by one or more "logical" records. A "logical" record is one of the following:

1. A MONARCH ID record (40 words).
2. A binary record (average length fewer than 28 words, maximum length 31 words).
3. An encoded symbolic record (40 words maximum).

An unblocked record consists of a single logical record; however, all unblocked records are written as 40-word records even if the logical record contains fewer than 40 words.

The block sentinel word has the following format:

Bits	Contents
0 - 8	0
9 - 11	3
12 - 23	Number of words in physical record (including the block sentinel word)

The blocking number serves a twofold purpose: it specifies the maximum number of words per record and indicates whether the update routine is to operate in the blocking mode on the selected segments. If no blocking factor is specified, the update routine operates on all segments in the normal mode.

Which segments are to be blocked when the update routine is operating in the blocking mode is determined by the level 1 or level 2 MONARCH ID record preceding that segment. If a level 1 or level 2 MONARCH ID record contains a B in character position 22 and blanks in character positions 21, 23, and 24, everything within the scope of that level 1 or level 2 ID record is blocked. Since the MONARCH loader is not preceded by a MONARCH ID record, it is automatically blocked whenever a blocking mode is specified.

CONTROLLING AN UPDATE RUN— THE UPDATE FILE

Normally, two logical "files" are presented to the update routine to enable it to create a new system tape. One of these files is the old system tape, and it is an optional input. The other file is the update file; it is never optional, although its form and content may vary considerably.

In the general case, the update file consists of an ordered sequence of COPY messages, MONARCH ID records, binary records, and encoded symbolic records. In a particular instance, an update file may consist entirely of COPY messages, in which case only the functions of selective duplicating and selective deleting are performed. Alternatively, a given update file may consist entirely of MONARCH ID records, binary records, and encoded records, in which case only the functions of selective insertion and, by the absence of COPY messages, blanket deletion of all information on the old system tape are performed. In the latter case, the absence of COPY messages removes the requirement for providing an old system tape for the update run.

Physically, the update file can exist entirely on magnetic tape, on punched cards, or (although highly unlikely) entirely on paper tape. Alternatively, all COPY messages and MONARCH ID records in the update file can be presented as typewriter messages while any programs to be inserted are presented on punched cards, paper tape, or magnetic tape. (Samples of update file listings are included at the end of this section.)

Insertion

Insertion is controlled by presenting the update routine with a MONARCH ID record via the control message medium (QMSG) which may also be the update medium (QSYSU), and one or more programs (or data files) via the update medium (QSYSU). The MONARCH ID record

is the first record written on the new system tape. The update routine then copies records from the update medium onto the new system tape until:

1. An end-of-file condition (not △EOF) is detected. The update routine will then request a control message.
2. A possible COPY message is encountered; i.e., a record other than a binary, encoded, or MONARCH ID record. (The update routine proceeds to analyze it as if it were a control message.)
3. If the update medium is paper tape and a binary or encoded end record (type 3) is encountered, a halt is executed. (Set (A) = 0 to continue insertion, or set (A) ≠ 0 to stop insertion and cause the update routine to request a control message next; then set the RUN-IDLE-STEP switch to RUN.)
4. A level 1 MONARCH ID record with SYSEND$_{\wedge\wedge}$ in characters 9 through 16 was written on the new system tape. (Both old and new system tapes are rewound, and the monitor is loaded from the system tape on unit 0 of the W buffer.)

Note: When an insertion is under control of a level 2 MONARCH ID record and the insertion is to become, or replace, the first subdivision of a major division of the system tape, the level 1 MONARCH ID record for the major division must precede the level 2 MONARCH ID record in the update file. (See examples at the end of this section.)

Programs to be inserted must be in XDS standard binary language. Data files must be in this format or else in XDS 9300/900 Series encoded symbolic format.

All binary and encoded records inserted in a new system tape will have their checksums validated by the update routine.

Deletion

Deletion of programs or data files from an old system tape is accomplished by simply excluding those programs or data files from the scope of a COPY message. In other words, failure to COPY a program results in its being deleted from the new system tape.

Replacement

Replacement of programs or data files is accomplished by deleting (not COPYing) the existing program or file and by inserting a new version of that program or file.

Retention (COPY function)

Retention of programs or data files is accomplished by including those programs or data files in the scope of a COPY message. Retention must be made explicit; the only program implicitly "retained" from an old system tape to a new system tape is the bootstrap loader, but this program is not "copied" from the old tape by duplicating the first record on the old tape. It is in core at the time the update routine is executed; therefore, the update routine writes the bootstrap loader from core onto tape as its first operation (i.e., immediately after control is transferred to it and before any COPY messages are read). All binary and encoded records written on the new system tape will have their checksums validated by the update routine.

COPY MESSAGES

The purpose of a COPY message is to obtain programs or data files from the old system tape and record them on the new system tape. The COPY message is used in lieu of placing the indicated programs in the update file. COPY messages refer to records (e.g., binary programs) by using the program names that appear in character positions 9 through 16 of the MONARCH ID records on the old system tape.

Major divisions of a MONARCH system are preceded, on the system tape, by a level 1 MONARCH ID record:

$$\triangle 1_{\wedge\wedge\wedge\wedge\wedge\wedge}\text{LIBRARY}_{\wedge}\ldots$$

Minor divisions of a MONARCH system are preceded, on the system tape, by a level 2 MONARCH ID record:

$$\triangle 2_{\wedge\wedge\wedge\wedge\wedge\wedge}\text{COSINE}_{\wedge\wedge}\ldots$$

Minor divisions of a MONARCH system are arbitrary subdivisions of a program or a data file recognized by the update routine (see description of the MONARCH Loader in Section 3 for mention of another use of level 2 MONARCH ID records in connection with automatic library searching).

Each MONARCH ID record must have a unique label. Labels may not contain separators (see Section 2). In other words, each label must be explicit; a label such as

$$\triangle 1_{\wedge\wedge\wedge\wedge\wedge\wedge}\text{SIN, COS}_{\wedge}\ldots$$

is illegal because of the comma.

If an argument of a COPY message consists of one program name, the name is assumed to occur in characters 9 through 16 of a level 1 MONARCH ID record on the old system tape. If an argument of a COPY message consists of two program names (the second may be enclosed in parentheses), the first (leftmost) is assumed to occur in characters 9 through 16 of a level 1 MONARCH ID record on the old system tape, while the second program name is assumed to occur in characters 9 through 16 of a level 2 MONARCH ID record which occurs

subsequent to the level 1 record. In other words, the second program name is assumed to refer to a subdivision of the major division of the old system tape that was identified by the first program name. For example,

$\Delta\wedge\wedge\wedge$COPY LIBRARY (COSINE).

LIBRARY is a level 1 ID record, and COSINE is a level 2 ID record within the scope of LIBRARY.

Execution of a COPY message by the MONARCH update routine involves copying the MONARCH ID record(s) and any binary or encoded records that are in the scope of the MONARCH ID records named in the COPY message.

The term "in the scope of" is defined as follows:

If "A" and "B" are distinct program names in level 1 ID records, and "X" and "Y" are distinct program names in level 2 ID records, then:

1. A binary or encoded record (r) is "in the scope of" A provided that no other level 1 ID record occurs between A and r on the system tape.
2. A binary or encoded record (r) is "in the scope of" X provided that no other level 2 ID record occurs between X and r on the system tape.
3. X is "in the scope of" A provided that no other level 1 ID record occurs between A and X on the system tape.
4. A binary or encoded record (r) is "in the scope of" both X and A if rules 1, 2, and 3 apply.
5. If a binary or encoded record (r) is "in the scope of" X, it is not "in the scope of" Y.
6. If a binary or encoded record (r) is "in the scope of" A, it is not "in the scope of" B.

The Syntax of COPY Messages

A valid COPY message is an instance of one of the following:

$\Delta\wedge\wedge\wedge$COPY$\wedge$a.

$\Delta\wedge\wedge\wedge$COPY$\wedge$a$\wedge$(b).

$\Delta\wedge\wedge\wedge$COPY$\wedge$a$\wedge$THRU$\wedge$b.

$\Delta\wedge\wedge\wedge$COPY$\wedge$a$\wedge$(c)$\wedge$THRU$\wedge$b.

$\Delta\wedge\wedge\wedge$COPY$\wedge$a$\wedge$THRU$\wedge$b$\wedge$(d).

$\Delta\wedge\wedge\wedge$COPY$\wedge$a$\wedge$(c)$\wedge$THRU$\wedge$b$\wedge$(d).

where a, b, c, and d represent program names (MONARCH ID labels). The first character of a program name must be alphabetic and each remaining character must be either alphabetic or numeric. Each name may consist of up to eight characters. The message must be terminated by a period.

Parentheses may be omitted, their only purpose being to enhance readability.

The caret ($\wedge$) is used to indicate the minimum number of spaces that must separate words in a COPY message.

COPY messages without the word THRU are said to have one argument. COPY messages with the word THRU are said to contain two arguments. Each argument consists of either one or two program names.

COPY Messages with One Argument

A COPY message with one argument consisting of one program name causes the update routine to read all records in the scope of the level 1 ID record with the same name from the old system tape and write them on the new system tape. For example, when

$\Delta\wedge\wedge\wedge$COPY$\wedge$LIBRARY.

is encountered, the update routine bypasses any records on the old system tape preceding the level 1 MONARCH ID record with LIBRARY$\wedge$ in characters 9 through 16. This is the first record to be written on the new system tape in response to this COPY message. The update routine then copies all records following that ID record until the next level 1 MONARCH ID record is encountered on the old system tape. It is this "next" level 1 record that terminates the copying of records from the old system tape; it is not copied onto the new system tape as a result of this COPY message, but it is the first "old system tape" record to be examined when the next update control message is processed.

A COPY message with one argument consisting of two program names causes the update routine to read all records in the scope of the level 2 MONARCH ID record corresponding to the second program name and write them on the new system tape. For example, when

$\Delta\wedge\wedge\wedge$COPY$\wedge$LIBRARY$\wedge$(COSINE).

is encountered, the update routine bypasses any records on the old system tape preceding the level 1 MONARCH ID record with LIBRARY$\wedge$ in characters 9 through 16; that is, unless the old system tape is already positioned at, or beyond (but still within the scope of), that level 1 ID record. In either case, the update routine searches, within the scope of LIBRARY, for a level 2 MONARCH ID record with COSINE$\wedge\wedge$ in characters 9 through 16. If the level 1 MONARCH ID record for LIBRARY has not already been written on the new system tape, it is the first record written on the new system tape in response to this COPY message. In either case the update routine writes, on the new system tape, the level 2 MONARCH ID record with the name COSINE. The update routine then copies all records following that ID record until the next MONARCH ID record (either level 1 or level 2) is encountered on the old system tape. It is this "next" MONARCH ID record that terminates the copying of records from the old system tape. It is not copied onto the new system tape as a result of this COPY message, but it is the first "old system tape" record

to be examined when the next update control message is processed.

COPY Messages with Two Arguments

A COPY message with two arguments is equivalent to a series of "one argument" COPY messages. The update routine performs the necessary copying indicated by the first argument exactly as in the case of a 1-argument COPY message; but, in addition, it copies all records following those included in the scope of the first argument until the MONARCH ID record whose name matches the second (or only) program name[†] of the second argument is encountered. At this point, the update routine performs the necessary copying indicated by the second argument exactly as in the case of a 1-argument COPY message.

Special tests are made to detect cases in which the first and second arguments are identical. When this occurs, the COPY message is reduced to the equivalent 1-argument COPY message.

The following sets of COPY messages are equivalent if "A", "B", and "C" occur (in that order) as program names in consecutive level 1 MONARCH ID records on a system tape:

Set 1: △^^^COPY A THRU A.
△^^^COPY B THRU B.
△^^^COPY C THRU C.

Set 2: △^^^COPY A.
△^^^COPY B.
△^^^COPY C.

Set 3: △^^^COPY A THRU A.
△^^^COPY B THRU C.

Set 4: △^^^COPY A THRU B.
△^^^COPY C THRU C.

Set 5: △^^^COPY A THRU C.

Thus, the use of THRU, in a COPY message with two arguments, provides an alternative to using a series of 1-argument COPY messages.

Termination of an Update Run

COPY messages of the form:

△^^^COPY^SYSEND^^.

or

△^^^COPY^a^(b)^THRU^SYSEND^^.

cause the indicated COPY function to be performed, the update process to be terminated, the new system tape to be rewound, and control to be returned to the MONARCH monitor. In this case, the MONARCH monitor in question is "bootstrapped" from tape 0 on the W buffer.

[†]Matching of the second program name is inhibited until a level 1 MONARCH ID record whose name matches the first program name is encountered.

CONTENTS OF A TYPICAL MONARCH SYSTEM TAPE

1 record containing MONARCH bootstrap and LOADER

first record of MONARCH loader
.
.
.
Last record of MONARCH loader

△1 MONITOR
first record of MONARCH monitor
.
.
.
Last record of MONARCH monitor

△1 PRINT
first record of PRINT subroutine
.
.
.
END record of PRINT subroutine

△1 MTAPE
first record of MTAPE subroutine
.
.
.
END record of MTAPE subroutine

△1 CDRP
first record of CDRP subroutine
.
.
.
END record of CDRP subroutine

△1 PTYIO
first record of PTYIO subroutine
.
.
.
END record of PTYIO subroutine

△1 LIBRARY
△2 SINE
first record of SINE subroutine
.
.
.
END record of SINE subroutine

△2 COSINE
.
.
.
△1 SYSEND

EXAMPLES

Facsimile of a typical listing of MONARCH ID records resulting from a MONARCH update run:

```
△1      LOAD.
△1      MONITOR.
    △2      CONTROL.
    △2      TABLES.
    △2      QMSGRD.
    △2      LDIOSR.
    △2      CARD.
    △2      MTYIO.
    △2      MAGTP.
    △2      TFMONRCH.
△1      PRINT.
△1      MTAPE.
△1      CDRP.
△1      PTYIO.
△1      CDR.
△1      LIBRARY.
    △2      CDRP.
    △2      CDR.
    △2      PRINT.
△1      META920.
    △2      ENCODER.
    △2      MON1.
    △2      MSCONTRL.
    △2      PREASSEM.
    △2      ASSEMBLR.
△1      UPDATE.
    △2      BOOTSTRAP.
    △2      UPDATERT.
△1      SYSEND.
```

Examples of Program Sequences for Update Runs

1. To duplicate an existing MONARCH system tape:

```
△C TY1W.
△ASSIGN S=MT0W, X1=MT1W, UI=CR1W, LO=LP.
△UPDATE.
△^^^COPY LOAD THRU SYSEND^^.
```

2. To insert a system routine ("RN") between existing system routines "R1" and "R2":

```
△C CR1W.
△ASSIGN S=MT0W, X1=MT1W, UI=CR1W, LO=LP.
△UPDATE.
△^^^COPY LOAD THRU R1.
△1^^^^^^RN^^^^^^.
"first binary record of RN"
        .
        .
        .                          } binary deck for RN
"last binary record of RN (end record)"
△^^^COPY R2 THRU SYSEND^^.
```

3. To delete a system routine ("R7") that appears on the old system tape between system routines "R6" and "R8":

```
△C CR1W.
△ASSIGN S=MT0W, X1=MT1W, UI=CR1W, LO=LP.
△UPDATE.
△^^^COPY LOAD THRU R6.
△^^^COPY R8 THRU SYSEND^^.
```

4. To replace a system routine ("R7"), appearing on the old system tape between "R6" and "R8", with a new version of "R7":

```
△C CR1W.
△ASSIGN S=MT0W, X1=MT1W, UI=CR1W, LO=LP.
△UPDATE.
△^^^COPY LOAD THRU R6.
△1^^^^^^R7^^^^^^.

"first binary record on new version of R7"
        .
        .
        .
"last binary record of new version of R7"

△^^^COPY R8 THRU SYSEND^^.
```

5. To insert a new subroutine ("NEW") as the first subdivision under "LIBRARY", where "LIBRARY" is the name in a level 1 MONARCH ID record on the old system tape, "CDR" is the name in the level 1 MONARCH ID record immediately preceding "LIBRARY", and "CDRP" is the name in the first level 2 MONARCH ID record under "LIBRARY" on the old system tape.

```
△C CR1W.
△ASSIGN S=MT0W, X1=MT1W, UI=CR1W, LO=LP.
△UPDATE.
△^^^COPY LOAD THRU CDR.
△1^^^^^^LIBRARY^.
△2^^^^^^NEW^^^^^.

"first binary record of NEW"
        .
        .
        .
"last binary record of NEW"

△^^^COPY LIBRARY (CDRP) THRU SYSEND^^.
```

6. To re-order the system tape. Re-ordering of the system tape is accomplished by initializing a rewind following a series of COPY messages and/or inserts, then reading and executing a new series of COPY messages. This allows for repositioning logical files on the system tape without having to include the binary decks.

The control card for the rewind is recognized by the update routine. The format of the card is △REWIND. No blanks are allowed. (In the update routine △REWIND implicitly rewinds MT0.)

Example:

Old system tape order is the following sequence of level 1 ID records:

A B C D SYSEND

New order of tape requires:

A C D B SYSEND

The series of COPY messages should be:

△$_{\wedge\wedge\wedge}$COPY A.

△$_{\wedge\wedge\wedge}$COPY C THRU D.

△REWIND.

△$_{\wedge\wedge\wedge}$COPY B.

△$_{\wedge\wedge\wedge}$COPY SYSEND$_{\wedge\wedge}$.

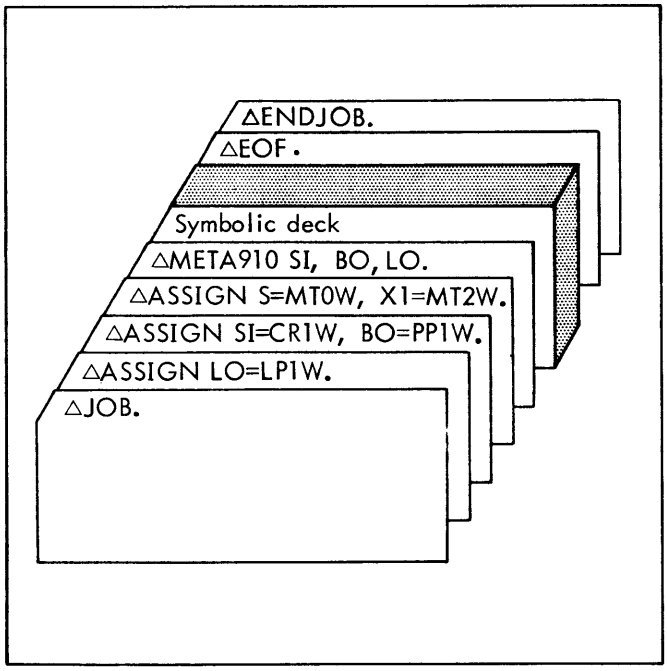
ERROR HALTS DURING UPDATE RUNS

Certain error conditions occurring during an update run cause an error message to be typed and the computer to halt. These error conditions are self explanatory and include the corrective action needed. The term OST refers to the old system tape (S), the term NST refers to the new system tape (X1), and the term UPD refers to the update input medium (UI), in the texts of the error messages.

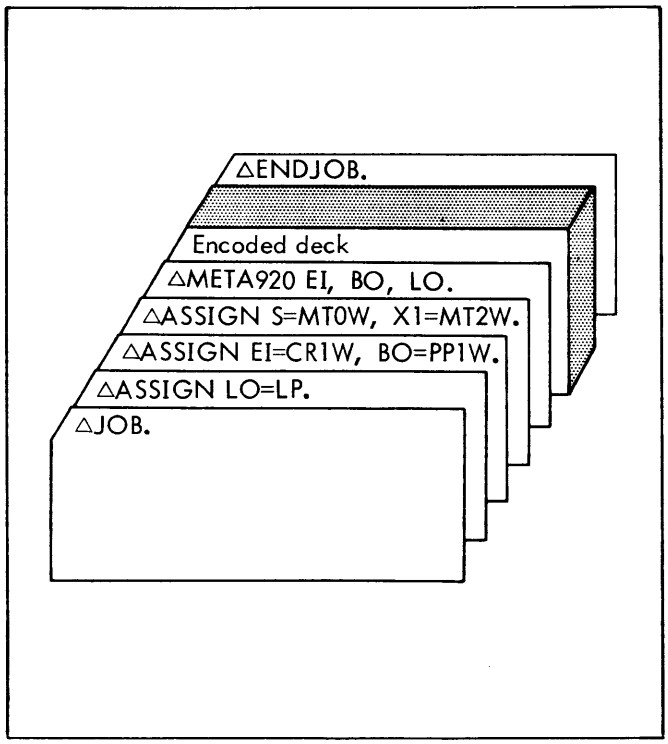
5. PREPARING PROGRAM DECKS

META-SYMBOL ASSEMBLY AND EXECUTION

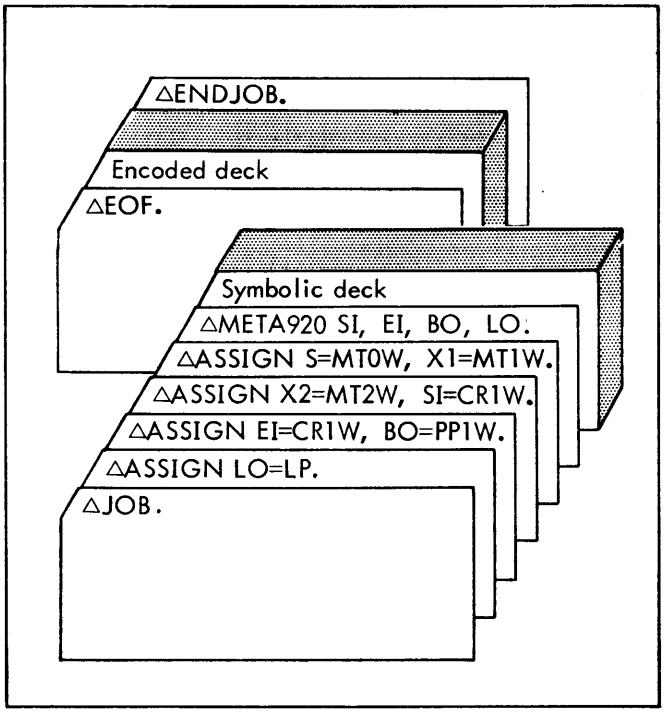
Assemble a META-SYMBOL symbolic program to produce an object program for a 910/925.



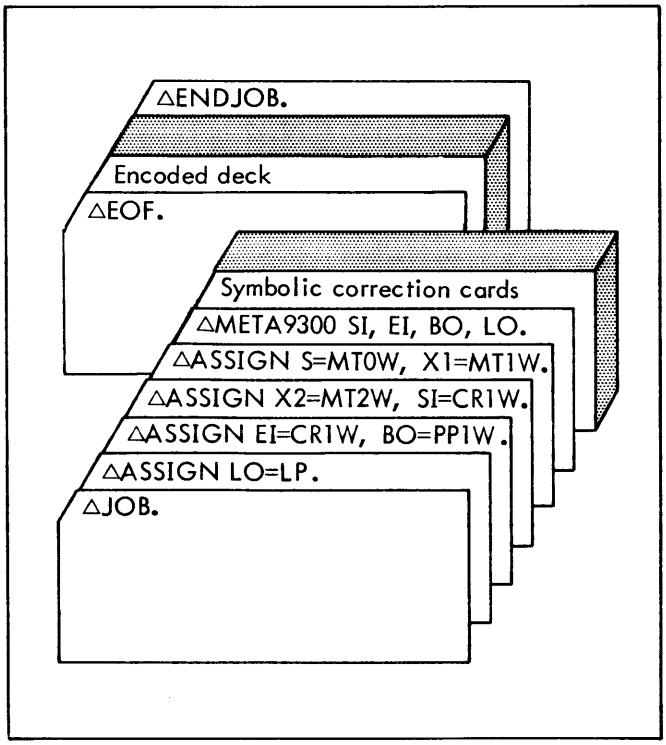
Assemble a META-SYMBOL encoded deck to produce an object program for 920/930. (Note that encoded deck requires no EOF indication.)



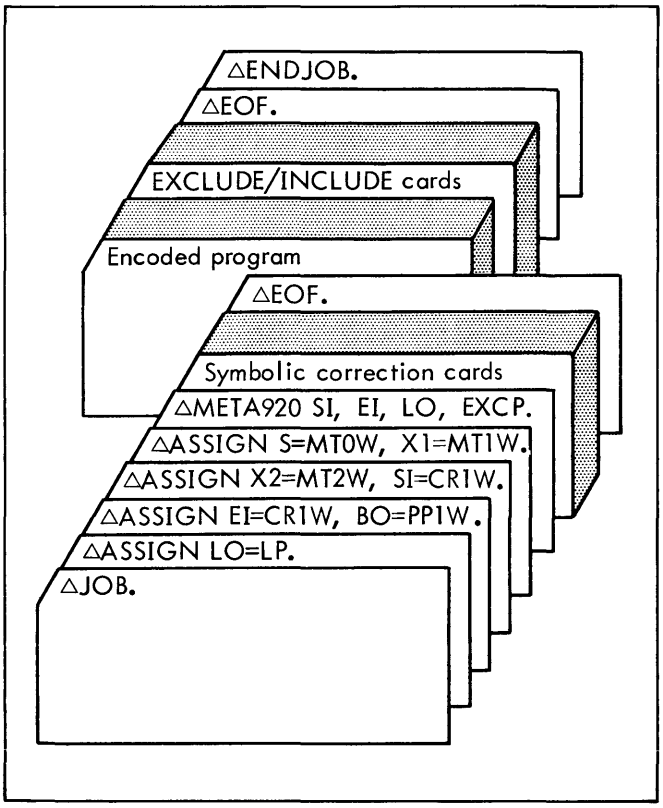
Assemble META-SYMBOL symbolic and encoded input from card reader. (Note that △ EOF indicates termination of SI.)



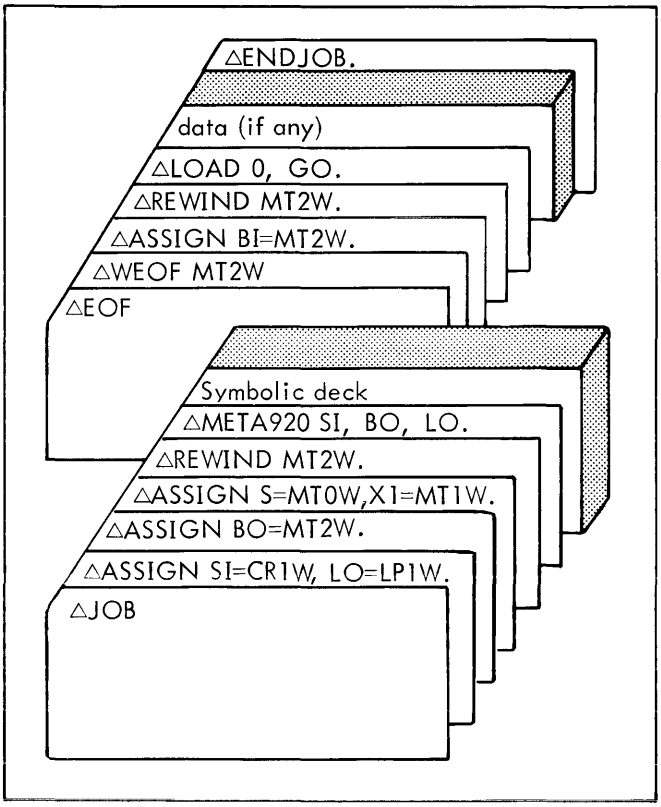
Assemble META-SYMBOL encoded deck with symbolic corrections on a 900 Series Computer to produce an object program to run on 9300.



Assemble a META-SYMBOL encoded deck with symbolic corrections, requesting a concordance listing and specifying symbols to be included and/or excluded.



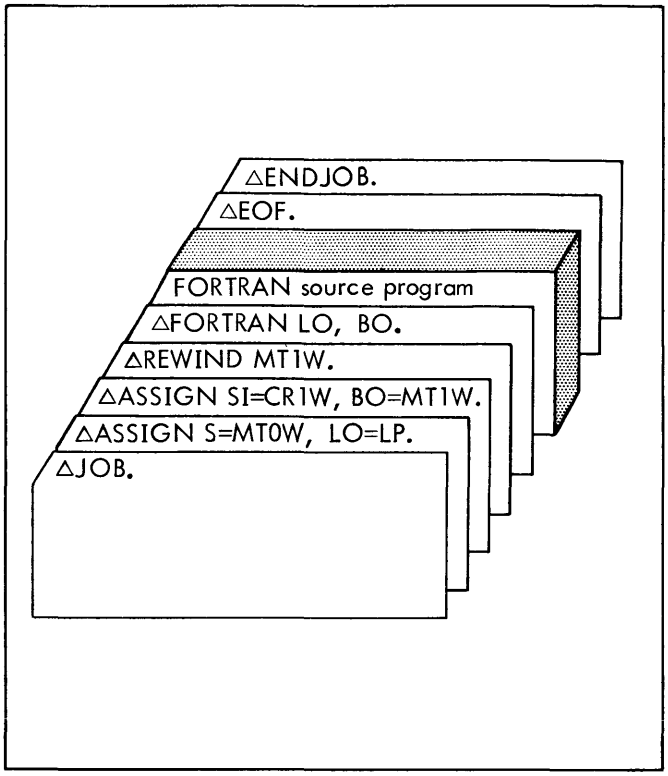
Assemble and execute a META-SYMBOL source program.



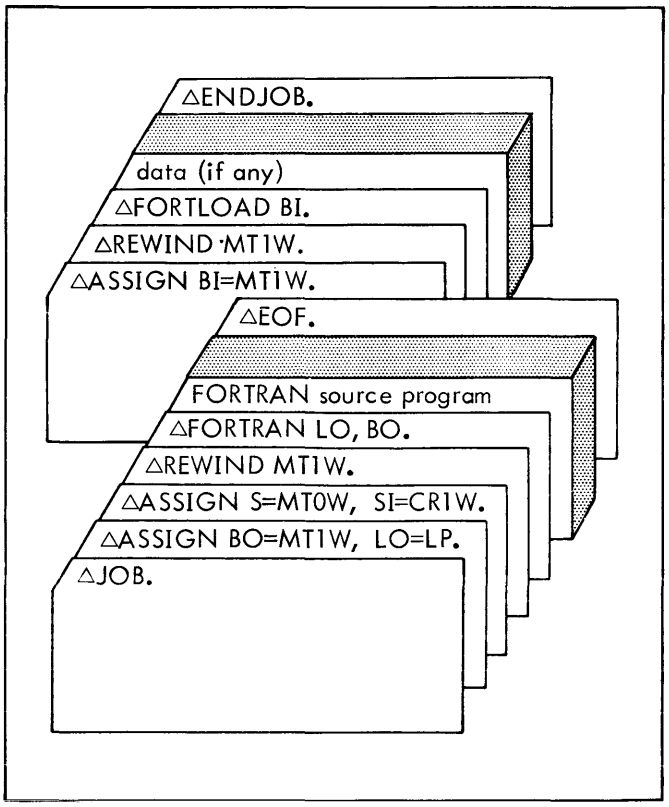
FORTRAN COMPILATION AND EXECUTION

(900 Series Only)

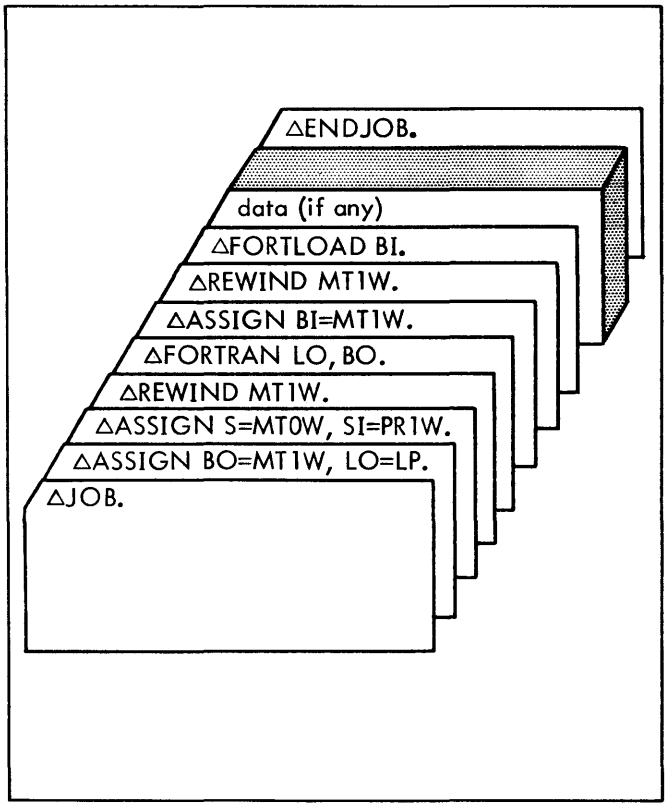
Compile a FORTRAN source program.



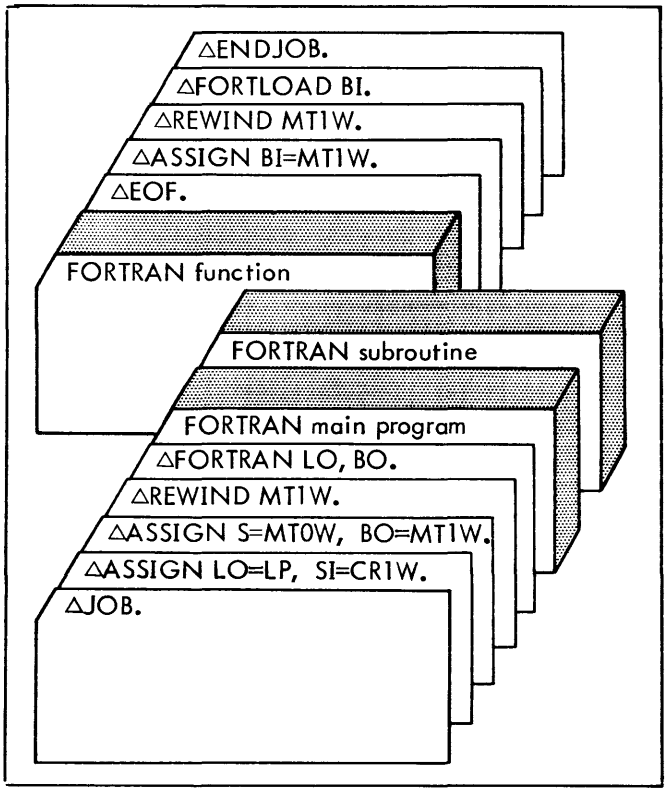
Compile and execute a FORTRAN source program.



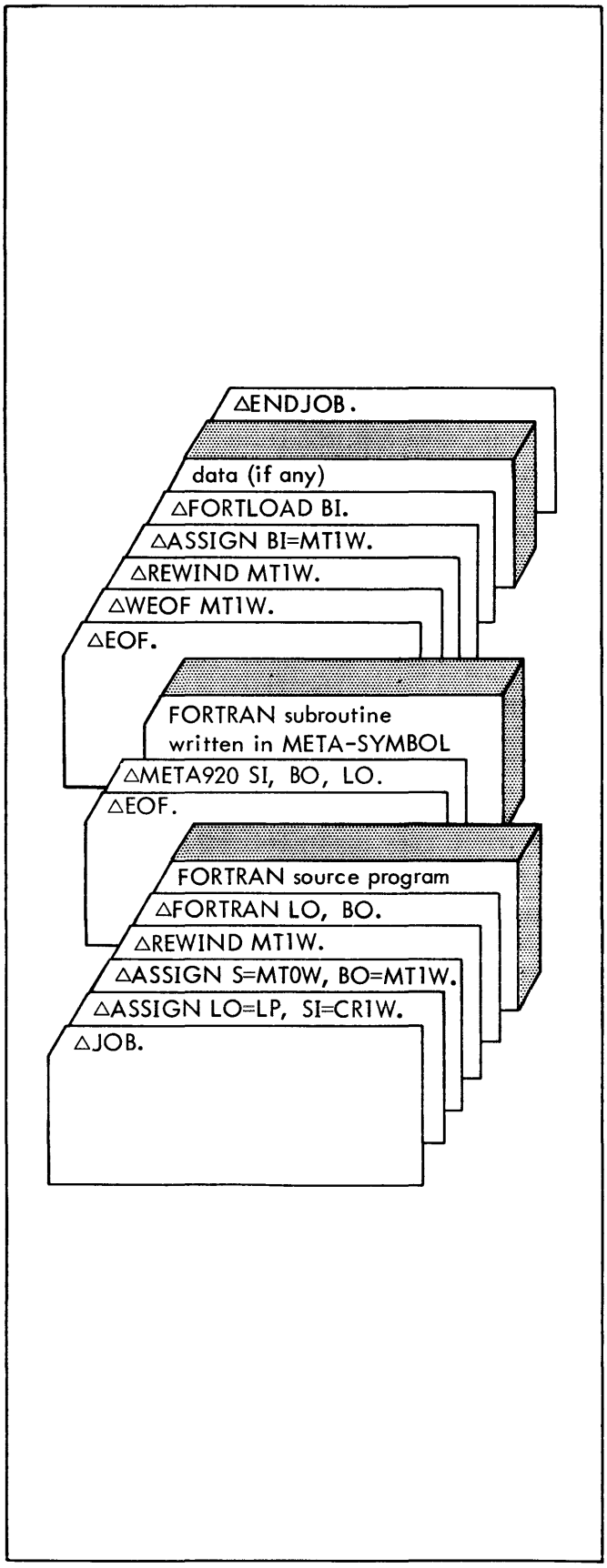
Compile and execute a FORTRAN program input from paper tape. If the source tape does not end with a ΔEOF, control must be transferred to MONARCH manually.



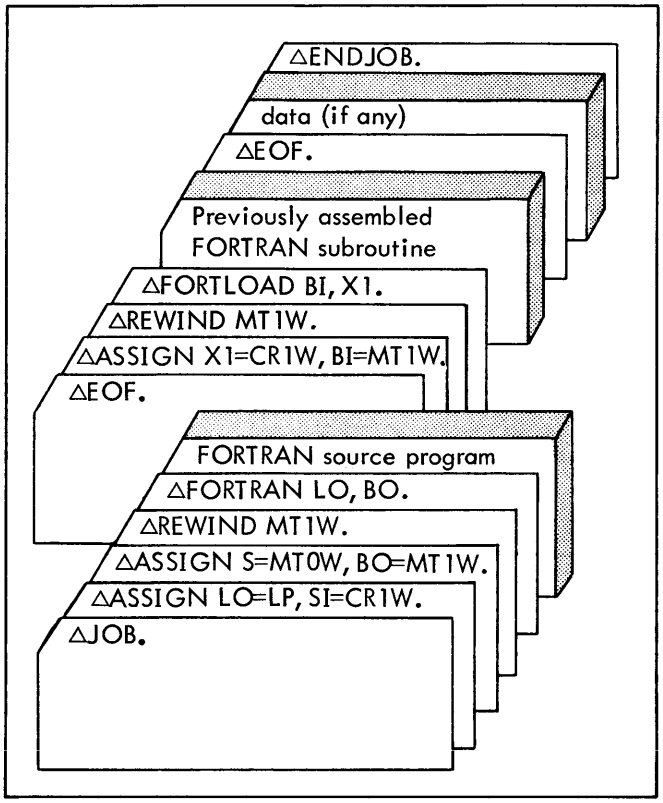
Compile and execute a FORTRAN program which includes a FORTRAN subroutine and function.



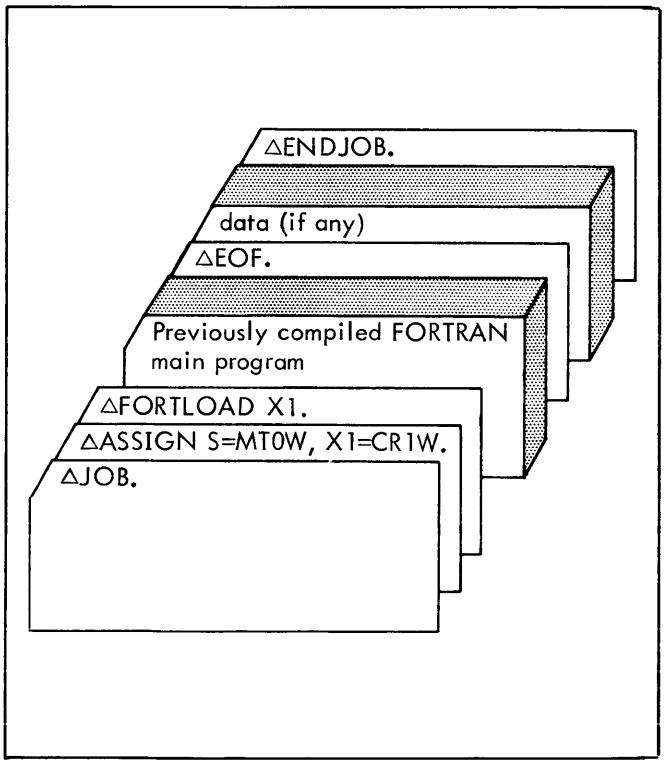
Compile and execute a FORTRAN program that uses a subroutine, written in META-SYMBOL language, which must be assembled.



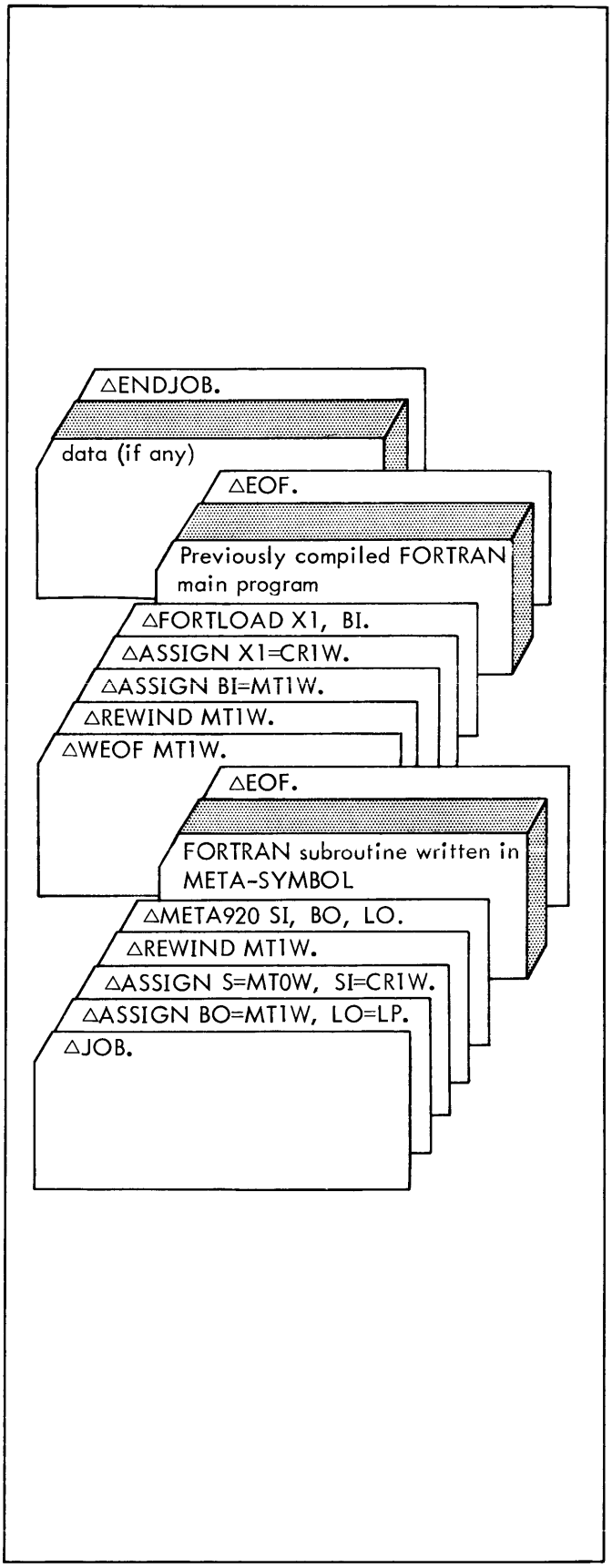
Compile and execute a FORTRAN program which uses a previously assembled (or compiled) FORTRAN subroutine on cards.



Execute a previously compiled FORTRAN program.

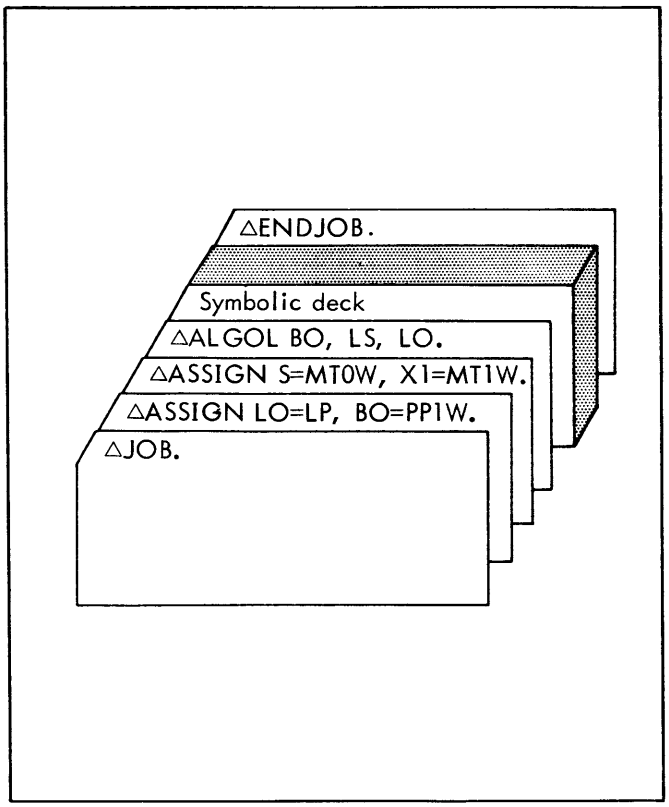


Execute a previously compiled FORTRAN program that uses a subroutine, written in META-SYMBOL language, which must be assembled.

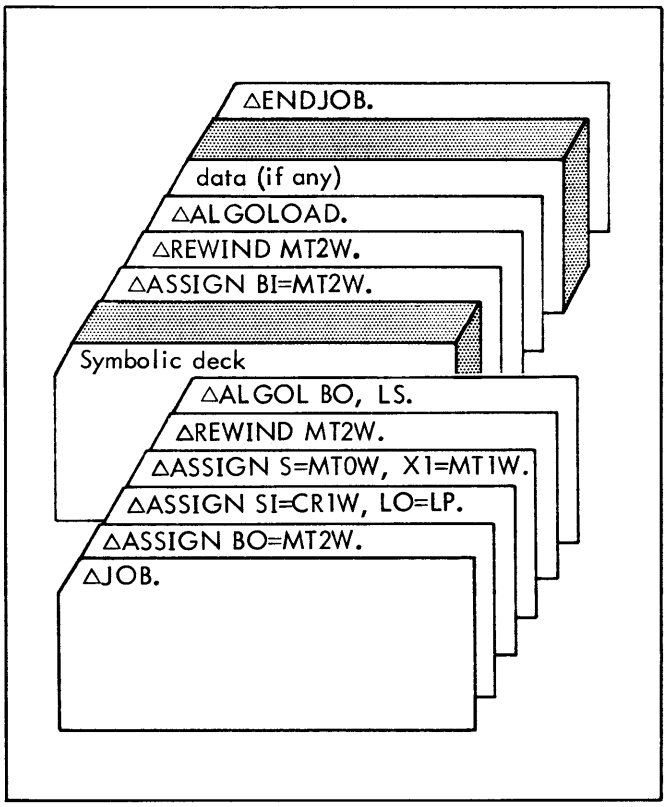


ALGOL COMPILATION AND EXECUTION

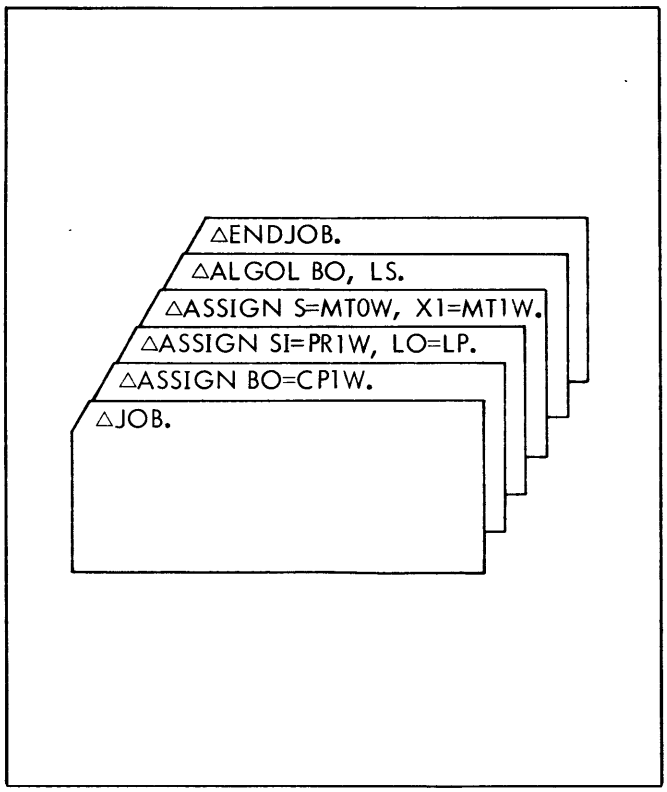
Compile an ALGOL source program input from cards.



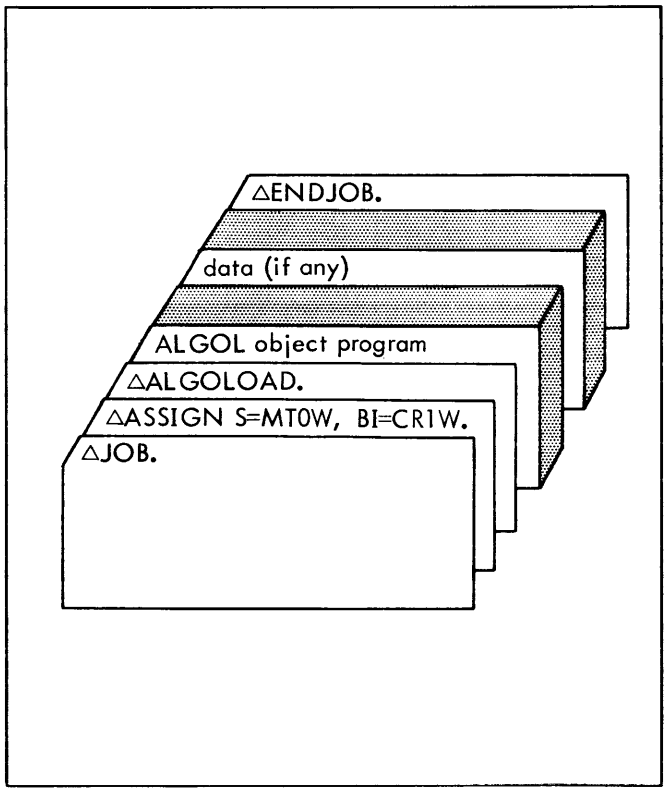
Compile and execute an ALGOL source program input from cards.



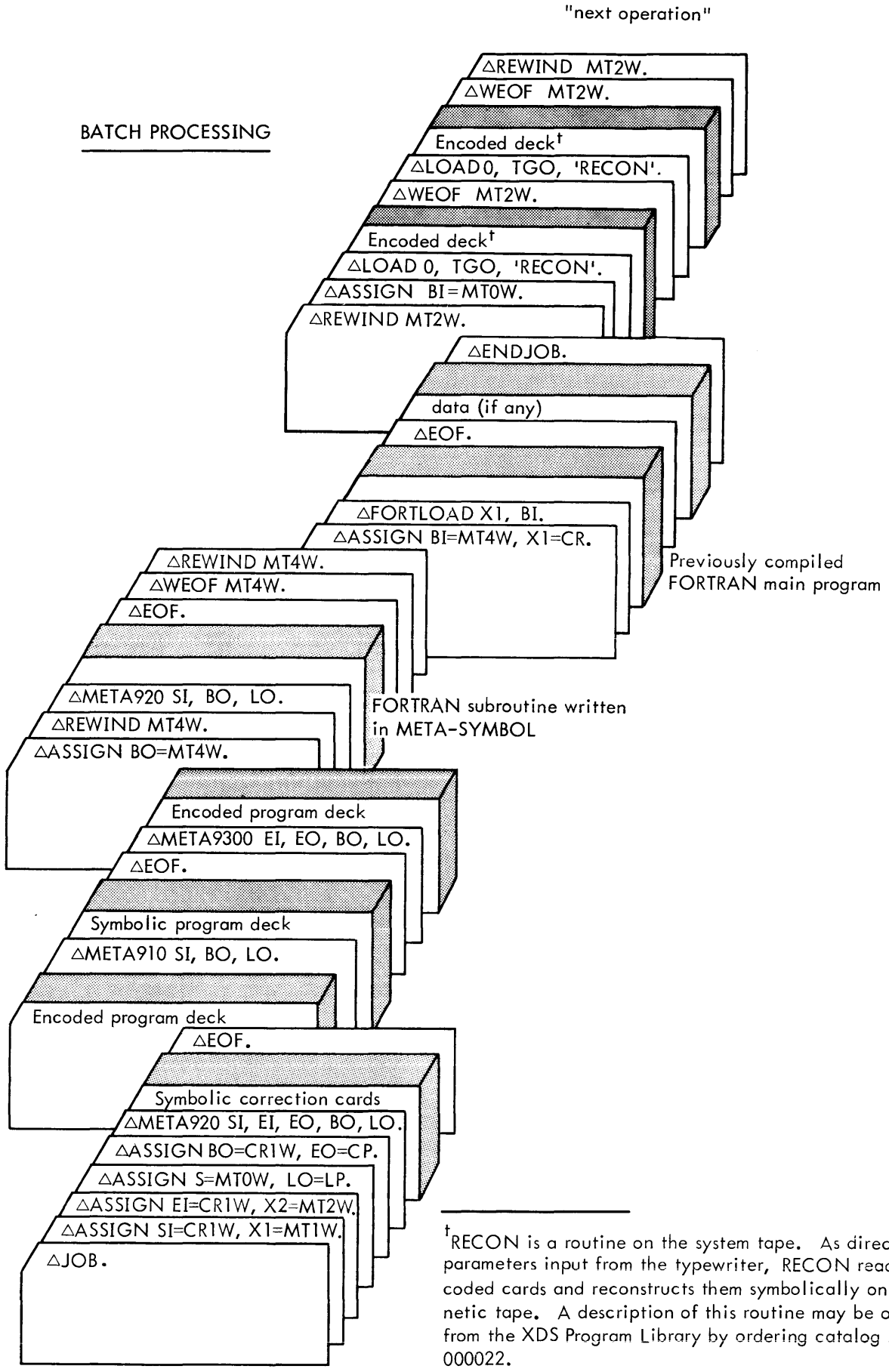
Compile an ALGOL source program input from paper tape.



Execute a previously compiled ALGOL program.



BATCH PROCESSING



†RECON is a routine on the system tape. As directed by parameters input from the typewriter, RECON reads encoded cards and reconstructs them symbolically on magnetic tape. A description of this routine may be obtained from the XDS Program Library by ordering catalog number 000022.

6. OPERATING PROCEDURES

LOADING THE MONARCH SYSTEM

1. Mount the MONARCH system tape on magnetic tape unit 0 on the W buffer; the unit must be ready for operation and the tape positioned at load point.
2. To load the system initially, proceed as follows:
   a. For XDS 910/920 Computers:
      (1) Set registers X, C, and P and memory cell 1 as follows:

```
00001 = 0 32 00002    WIM  2
X     = 77777771      -7
P     = 0
C     = 0 02 03610    EOM  03610
```

      (2) Set the RUN-IDLE-STEP switch to RUN. (Do not STEP first.)

   b. For XDS 925/930 Computers:

      Execute the magnetic tape FILL procedure.

   c. For XDS 9300 Computers:
      (1) Press RESET.
      (2) Execute a magnetic tape FILL.
3. To reload the system once it has been loaded:
   a. For 900 Series Computers:
      (1) Set C = 0 01 00001 BRU 00001
      (2) Set RUN-IDLE-STEP switch to RUN.

      If this procedure fails:

      (1) Set C = 0 01 x7736 BRU QBOOT

          X = 1 for an 8K computer
            = 2 for a 12K computer
            = 3 for a 16K computer

      (2) Set RUN-IDLE-STEP switch to RUN.

      If this also fails, execute the loading procedure described in paragraphs 1 and 2 above.

   b. For 9300 Computers:
      (1) Press RESET.
      (2) Press STEP.
      (3) Press RUN.

      If this procedure fails, execute the loading procedure described in paragraphs 1 and 2 above.

FURNISHING CONTROL MESSAGES

When the MONARCH system is initially loaded, the monitor attempts to obtain a control message from the console typewriter, i.e., the device indicated by the contents of QMSG in the unit assignment table. The control message medium may be changed from the device currently in use to another input device with a C control message (described in Section 2). For example, to change the control message medium to card reader 1 on the W buffer, use the control message

```
ΔC   CR1W.
```

After processing a C control message, MONARCH immediately attempts to read a control message from the newly assigned device. Control messages may be supplied on punched cards, paper tape, magnetic tape, or manually via an on-line typewriter.

At any given time while the MONARCH monitor has control of the computer, it expects to be able to obtain the next control message from the control message medium currently assigned. This imposes the following requirements on the console operator. If the medium is a:

1. Console typewriter, the console operator should be prepared to furnish a control message, via the typewriter, whenever its input light is lit.
2. Paper tape reader, the console operator should make certain that a paper tape containing a control message is inserted in the paper tape reader before the control request is made and that the paper tape reader is in operation.
3. Card reader, the console operator should make certain that a card containing a control message is in the card reader's input hopper and that the device is ready for operation. MONARCH reads cards in binary and converts the card image to XDS internal code before analyzing the message.
4. Magnetic tape, the console operator should make certain that a reel containing a physical record with a control message in it is mounted on the tape unit and that the unit is in ready status. MONARCH reads the tape in binary (odd parity) mode and assumes the maximum record length is 40 words.

If programs or data precede the next control message on the current control message medium (2, 3, or 4 above), MONARCH reads successive records from the unit until a control message record is encountered or an end-of-file condition occurs (cards and magnetic tape only). If an end-of-file is encountered before a control message is read, MONARCH types an appropriate message and requests the next control message from typewriter 1 on the W buffer.

Contents of Register C	Program	Normal, Planned, or Error Halt	Explanation of Halt	Recovery Procedure	Action
001xxxxx (xxxxx = transfer address from end card)	Loader	Planned	Computer halts when an end card with a transfer address is encountered. This halt occurs only when the value of the second parameter of the LOAD control message so specifies, i.e. STOP, TSTP.	Set RUN-IDLE-STEP switch to RUN to execute the program.	Transfers to object program for execution of that program.
001xy650 (See "Octal Dump Routine" in Section 4 for values of x and y.)	Loader	Normal Planned	Stops in the octal dump console driver routine.	Reset the A and B registers to continue the dump operation or set the C register to 00100355 (BRU RDMSGR) to have the system read a control message. Set the RUN-IDLE-STEP switch to RUN.	Dumps the next requested area to line printer 1 on the W buffer or selects the control message medium for input.
02000000	Update	Normal	Normal halt when update medium is paper tape.	Set (A) > 0 if the program just read was the last paper tape record and clear the halt. Set (A) = 0 if the program just read was not the last one. Insert next program tape into the paper tape reader and clear the halt.	Continues update process.
02000001	Loader or Bootstrap Loader (MT0 only)	Error	This halt indicates a buffer error.	1. For card input, replace the misread card in the hopper. Set the RUN-IDLE-STEP switch to RUN. 2. For paper tape input, reposition the record for re-read. Set the RUN-IDLE-STEP switch to RUN. 3. For magnetic tape unit 0, the program automatically tries ten times to read the record. Setting the RUN-IDLE-STEP switch to RUN causes it to try once more.	1. Not applicable. 2. Not applicable. 3. Not applicable.
02000003	Loader	Error	This halt indicates an illegal input format.	Clear the halt to cause the program to ignore the record and continue.	Not applicable.
02000004	Loader or Bootstrap Loader	Error	A checksum error has occurred.	Clear the halt to cause the program to ignore the error and continue.	Not applicable.
02000006	Loader	Error	This halt indicates symbol table overflow.	No recovery.	Not applicable.
02000007	Loader	Error	Unsatisfied external label or POP reference remains after library search. Unsatisfied references and/or missing definitions are automatically output to the typewriter or line printer.	1. No recovery; i.e., no way to satisfy the references or definitions. 2. Clear the halt to ignore the unsatisfied labels or references and continue.	1. Not applicable. 2. Error is ignored.
02000010	Loader	Error	Duplicate external labels or POP definitions encountered.	Clear the halt to continue.	Error is ignored. The first definition encountered will be used.
02022222	Loader	Planned	The computer halts when an end record with no transfer address is encountered. This halt occurs only when requested (i.e., only when the value of the second parameter of the LOAD control message so specifies).	Set (A) = load relocation bias of next program, and set RUN-IDLE-STEP switch to RUN to continue loading.	Reads next program unit.
04010410	Loader	Normal Planned	Stops in the symbol table typeout driver routine.	Clear the halt to continue.	Reloads MONARCH system into core.

Figure 1. MONARCH Program Halts and Recovery Procedures

PROGRAM HALTS AND RECOVERY PROCEDURES

All error messages are self-explanatory and include recovery procedures, whenever recovery is possible. These messages are listed on typewriter 1 on the W buffer if Breakpoint 1 is reset or on line printer 1 on the W buffer if Breakpoint 1 is set.

As stated previously, control messages may be input from cards, paper tape, typewriter, or magnetic tape. However, if an error is detected in a control message, typewriter 1 on the W buffer is selected for input (after an appropriate error message is output), and MONARCH waits for a new control message to be submitted by the operator.

Figure 1 describes normal, planned, and error halts for which no messages are produced. The errors are identified by codes placed in the C register.

SYSTEM OUTPUT

Parameters of various control messages enable the user to specify that certain information be produced during or following the operation of portions of the MONARCH system. For example, with a LOAD control message parameter the user may request a printout of the loader's symbol table; via the FORTLOAD control message he may specify that a label map, a storage map, and/or a trace of the object program be printed.

MONARCH LOADER'S SYMBOL TABLE

The MONARCH loader constructs a symbol table during the loading of a program. If the parameter TSTP or TGO appeared in the LOAD control message, the loader will output this symbol table after the program has been loaded. The table is produced on typewriter 1 on the W buffer if Breakpoint 1 is reset or on line printer 1 on the W buffer if Breakpoint 1 is set. For a 900 Series computer with a 12K memory, the printout has the format shown below. A complete symbol table list is given in Figure 3 at the end of this section. "Symbol Table Typeout" in Section 4 explains the procedure for producing a symbol table for computers with other memory sizes.

```
QLPBbbbb    40027764
QCPBbbbb    40027763
QCRBbbbb    40027762
QL7Bbbbb    40027761

QSYSTPbb    40027244
ACCUM1bb    40000206
WDTYPbbb    40000212
EOMbbbbb    40000213
```

The first column contains the 8-character symbols (trailing blanks are supplied for symbols having fewer than eight characters). The first group of symbols, each beginning with Q, are the loader's external label definitions that allow the user's program to reference locations within the resident portion of MONARCH. Following these are the symbols from the user's program.

The second column lists the numeric codes specifying the types of symbols (external label definitions, external POP references, etc.) and the locations to which the symbols are assigned (the last five octal digits). If the first digit (counting from the left) of the numeric code is equal to or greater than 4 (i.e., bit position 0 of the computer word contains a 1 bit), the reference is satisfied; if the first digit is less than 4 (i.e., bit position 0 of the computer word is a 0 bit), the reference is unsatisfied. If the fourth digit (counting from the left) is equal to or greater than 4 (i.e., bit position 9 of the computer word is a 1 bit), the symbol in the first column has a duplicate definition, in which case the first definition encountered is used and the subsequent duplicate definition is ignored. See Appendixes C and F for a detailed description of the various types of symbols and how they are designated.

FORTRAN LOADER'S OUTPUT

During the loading of an object program, the FORTRAN loader outputs the headings NAME, ENTRY, ORIGIN, LAST, SIZE/10, COMMON, and BASE (on typewriter 1 on the W buffer if Breakpoint 1 is reset or on line printer 1 on the W buffer if Breakpoint 1 is set).

If neither a label map nor a storage map is requested, the next output shows the program name, the entry location, the origin of the program, the last location of the program, the number (in decimal notation) of locations occupied by the program, and (when applicable) the number (in octal notation) of COMMON locations. If a label is requested, it is output immediately below the headings. If a storage map is also requested, it is output following the label map. An abbreviated example of the FORTRAN loader's output is given in Figure 2.

The entry, *PROGRAM, is always printed; it identifies the line containing the total program storage information. $$$$$$$ is the compiler-assigned identification for the main program.

The value in the column BASE is used to determine the exact location of variables. Variables are compiled to be stored immediately following the program in which they are used. At compilation time, variables are assigned locations relative to the end of the program; it is these relative locations which are printed as "Program Allocation" by the compiler. To determine the absolute location of a given variable, add its relative location to the value listed by the FORTRAN loader as "base" for the program containing that variable. For example, assume the variable J was assigned relative location 55. Using the base shown in Figure 2, the absolute location of J is determined by adding 04643 and 55, which results in 04720.

NAME	ENTRY	ORIGIN	LAST	SIZE/10	COMMON	BASE
= 7	03522					
= 2	03525					
= 232	03547					
= 7232	03553					
.						
.						
.						
$$$$$$$$	03462	03452	05224	875		04643
ABSF	05226	05225	05242	14		
203SYS	05244	05243	05254	10		
.						
.						
.						
*PROGRAM	03462	03452	06101	1304		

Figure 2. FORTRAN Loader Output

QLP3bbbb	40027764	WDTYPbbb	40000212	MTYIObbb	40004023
QCPBbbbb	40027763	EOMbbbbb	40000213	MPRNTbbb	40005241
QCRBbbbb	40027762	IORELCbb	40000231	CODESbbb	40002426
QL7Bbbbb	40027761	PRMCTRbb	40000237	PARAMSbb	40002650
QL6Bbbbb	40027760	METPRCbb	40000242	OMSGRDbb	40003277
QL5Bbbbb	40027757	CMFDTbbb	40000250	CHARbbbb	40003177
QL4Bbbbb	40027756	CADDRbbb	40000255	BTLDXbbb	40007100
QL3Bbbbb	40027755	MSGbbbbb	40000256	LDI2Xbbb	40007065
QL2Bbbbb	40027754	RDMSGbbb	40000326	CTFDTbbb	40003465
QL1Bbbbb	40027753	RDMSGRbb	40000355	CTBUFbbb	40003474
QLØBbbbb	40027752	MSGRSTbb	40000576	HOLBCDbb	40003333
QSYSPbbb	40027775	GETWRDbb	40000617	LDIOSRbb	40005540
QBINIbbb	40027774	TYPMbbbb	40001231	SYMBARbb	40006753
OSYMObbb	40027773	TYPOUTbb	40001231	METAbbbb	40006335
QBINObbb	40027772	SETbbbbb	40001365	FLPTbbbb	40006225
QSYSTbbb	40027771	DISPLYbb	40001410	FORTLAbb	40006044
QSYMIbbb	40027770	ASSIGNbb	40001470	FCPTbbbb	40006236
QSYSbbbb	40027767	Lbbbbbbb	40001535	FORTCAbb	40005775
OSYSIbbb	40027765	BTLDbbbb	40001607	FKPTbbbb	40006224
QSYSUbbb	40027776	ABSbbbbb	40001617	FORTKAbb	40006040
QPESWbbb	40027777	RELbbbbb	40001631	ALGOLAbb	40006262
OSYLDRbb	40025431	MESSAGbb	40001713	POSNbbbb	40006617
OBOOTbbb	40027736	MESSPRbb	40001734	LABELbbb	40006572
QDUMPbbb	40027646	CRDTPbbb	40001735	REWINDbb	40006534
QETBLbbb	40026476	LOADbbbb	40002077	WEOFbbbb	40006542
QMSGbbbb	40027766	FILLSYbb	40002216	BKFILEbb	40006422
QSRCHbbb	40027104	JOBMSGbb	40002232	SKFILEbb	40006500
QCWbbbbb	40027246	ENDJOBbb	40002236	BKRECbbb	40006504
QTAPEbbb	40026760	UPDATEbb	40002242	SKRECbbb	40006511
QPAPERbb	40026737	GSYSPbbb	40002276	FORTBIAS	40006672
QCARDbbb	40026714	OMTEbbbb	40002311	LDI2bbbb	40006334
OSYSINbb	40027243	SRLDSYbb	40002313	QENDMNbb	40007455
QSYSTPbb	40027244	CARDbbbb	40003560		
ACCUM1bb	40000206	MAGTPbbb	40004501		

Figure 3. MONARCH Loader Symbol Table (900 Series Computer, 12K Memory)

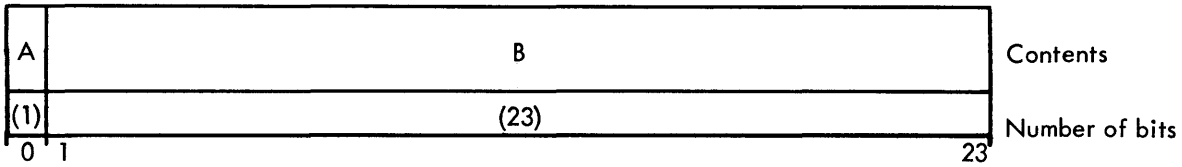
APPENDIX A. THE MONARCH UNIT ASSIGNMENT TABLE

To allow the use of the same input/output device for the same function throughout a series of runs, MONARCH maintains a table of standard unit assignments in upper memory. Each entry represents, by convention, a particular input or output function. For example, in a batch of runs consisting of assemblies and compilations, it is desirable to be able to designate a particular output unit (e.g., a card punch) as the unit on which all object programs are to be written. In the MONARCH system, this unit is referred to as the binary output unit (BO) and would be assigned, in this case, BO =CP1W.

Twenty-one such input/output functions have been designated in the MONARCH system; ten standard UAT entries and eleven special purpose Business Language package I/O entries. Additional functions may be added at a later date. The format of the unit assignment table entries and a description of the functions currently provided are given below.

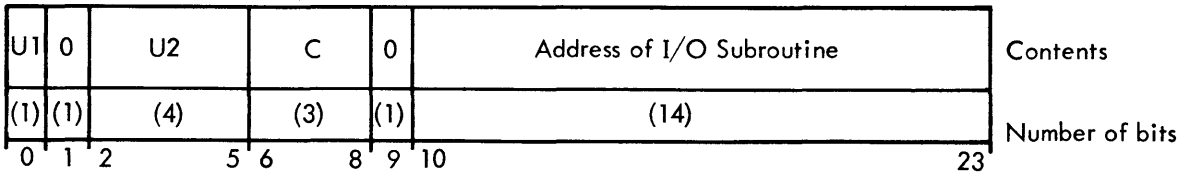
Standard XDS I/O subroutines are constructed so that they can make use of the MONARCH unit assignment table to obtain unit and channel codes for their operation. The reader should consult the description of these subroutines for additional information regarding the use of the MONARCH unit assignment table. (The program description catalog numbers for these subroutines are shown on the tape listing in Appendix B.)

At load time the unit assignment table is automatically allocated to the top of core in relocatable form. The last word of the unit assignment table is $QPESW, a 1-word entry defined as the job and processor error switch (see JOB and ENDJOB control messages in Section 2). The format of the words in the UAT is illustrated below. The upper portion of each diagram contains identifying symbols which, along with their definitions, describe the contents of the word. The lower portion of each diagram shows the number of bits reserved to each of these elements.



A = Job mode indicator:
0 = not in job mode
1 = in job mode
B = Processor error count

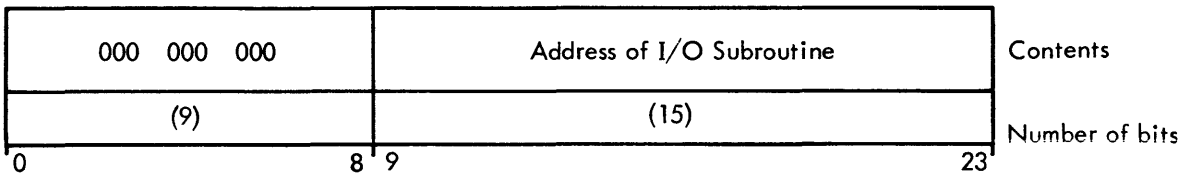
Format of the 1-word unit assignment table entries for 900 Series MONARCH:



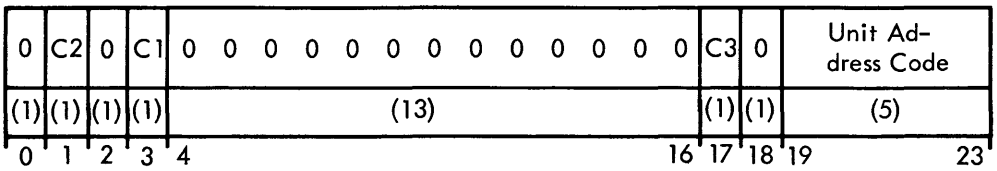
U1 U2 = Unit Address Code (5 low-order bits of the 6-bit unit address code)
C = Channel designator:
0 = W buffer
1 = Y buffer

Format of the 2-word unit assignment table entries for 9300 MONARCH:

Word 1



Word 2



C1 = high-order bit of the 3-bit channel code
C2 = second highest order bit of the 3-bit channel code
C3 = low-order bit of the 3-bit channel code

The high speed printers are designated, in UAT entries, by unit address codes of 20_8 (number 1) and 21_8 (number 2).

STANDARD UNIT ASSIGNMENT ENTRIES

Function	External Label	MONARCH Symbolic Parameter
Control message input	$QMSG	--
System (MONARCH magnetic tape)	$QSYS	S
System scratch	$QSYST	X1
System intermediate output scratch (magnetic tape)	$QSYSI	X2[†]
System scratch (magnetic tape)	$QSYSP	X3
Encoded output (META-SYMBOL)	$QSYSP	EO
Symbolic input (e.g., card reader)	$QSYMI	SI
Symbolic output, Update input	$QSYSU	SO, UI
Binary input (loader uses this)	$QBINI	BI
Encoded input (META-SYMBOL)	$QBINI	EI
Binary output (e.g., card punch)	$QBINO	BO
List output (e.g., printer)	$QSYMO	LO

BUSINESS LANGUAGE UNIT ASSIGNMENT ENTRIES

Function	External Label	MONARCH Symbolic Parameter
Magnetic Tape Zero	$QL0B	L0
Magnetic Tape One	$QL1B	L1
Magnetic Tape Two	$QL2B	L2
Magnetic Tape Three	$QL3B	L3
Magnetic Tape Four	$QL4B	L4
Magnetic Tape Five	$QL5B	L5
Magnetic Tape Six	$QL6B	L6
Magnetic Tape Seven	$QL7B	L7
Card Reader	$QCRB	LCR
Card Punch	$QCPB	LCP
Line Printer	$QLPB	LLP

When a standard processor such as FORTRAN or META-SYMBOL is loaded, MONARCH selectively loads any standard I/O subroutines required for I/O functions which the processor is expected to perform. The address of each I/O subroutine loaded is stored in the UAT entries whose unit address codes correspond to that subroutine. For example, if the magnetic tape I/O subroutine is selectively loaded, its address is stored in each UAT entry whose unit address code specifies a magnetic tape unit.

[†]X2 must be assigned to magnetic tape unit 2 (MT2) under MAGPAK environment when using META-SYMBOL.

MONARCH maintains a list of standard I/O subroutines required for each system action routine, in addition to the initial loading address for the first I/O subroutine to be loaded for operation with the system action routine.

MONARCH provides external label definitions for unit assignment table entries which correspond to I/O subroutines selectively loaded by the MONARCH loader.

The I/O subroutines are referred to indirectly through the unit assignment table in upper memory. Unit assignments can be made externally through ASSIGN messages. Note that the ASSIGN message does not set up I/O subroutine addresses (bits 10 through 14) in UAT entries.

The Business Language user's program is linked to the Business Language I/O handlers via external references and definitions at load time.

APPENDIX B. THE MONARCH SYSTEM TAPE

The MONARCH system tape consists of an ordered collection of programs and data files preceded by a special bootstrap loader which itself can be loaded under control of the FILL switch on the XDS 910/925 Computers, the magnetic tape FILL switch on the XDS 925/930 Computers, and the magnetic tape LOAD switch on the XDS 9300 Computer. The bootstrap loader, in turn, loads the MONARCH loader and the MONARCH monitor. All programs (or processors) on the system tape except the bootstrap loader and the MONARCH loader are preceded by MONARCH ID records. The MONARCH monitor, the MONARCH loader and the MONARCH update routine all make use of MONARCH ID records to locate programs or data files on a MONARCH system tape. Only the first 16 characters of a MONARCH ID record are interpreted by the MONARCH operating system. MONARCH ID records have the following format:

Character Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	70	71	72
Contents	△	n	ƀ	ƀ	ƀ	ƀ	ƀ	ƀ	a	c	c	c	c	c	c	c	e	e	e	e	e	e

n = 1 or 2
ƀ = space
a = any alphabetic character
c = any alphanumeric character or trailing space (i.e., space not followed by another character)
e = any character

Major divisions of a MONARCH system are preceded, on the system tape, by a level 1 MONARCH ID record:

△1^^^^^^PROGNAME . . .

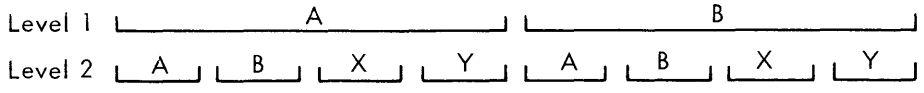
Minor divisions are preceded by a level 2 MONARCH ID record:

△2^^^^^^SUBRNAME . . .

Minor divisions of a MONARCH system are arbitrary subdivisions of a program or of a data file that are recognized by the MONARCH update routine. Normally, these subdivisions serve only to enable the user to update an old system tape at the subdivision level, i.e., to insert, delete or replace one or more subdivisions of a program without affecting the remaining subdivisions. Individual subroutines on the MONARCH library are also separated by level 2 records both for the MONARCH loader and to make it possible to insert, delete, and replace the subroutines individually.

Program names occurring in level 1 MONARCH ID records must be unique within a given MONARCH system tape. Program names occurring in level 2 MONARCH ID records need be unique only within a given major subdivision of a MONARCH system tape. The following additional rules apply to the program names on a given MONARCH system tape:

If A and B are unique program names occurring in level 1 MONARCH ID records and X and Y are unique program names occurring in level 2 MONARCH ID records, then A, B, X, and Y may be used as program names in level 2 ID records of A and/or B. Symbolically this may be represented as



The last record on a MONARCH system tape is a level 1 MONARCH ID record with the program name SYSEND in characters 9 through 14 (15 and 16 must be blank).

The system tape contains the monitor, the MONARCH loader, the system tape update routine, and other standard system routines required by the particular installation using the system. The system tape unit must be assigned as unit 0 on the W buffer for the XDS 910/920, unit 0 on channel W for the XDS 925/930, and unit 0 on channel A for the XDS 9300. A sample listing of the contents of a system tape for a 920 computer appears at the end of this appendix.

MONARCH SYSTEM TAPE RECORDS

Format of first words of valid system tape records:

XDS STANDARD BINARY OBJECT PROGRAM RECORD (first word)

Record Type (T)		Word Count (C)	Mode (Binary)	Folded Checksum (FC)	
(3)	(1)	(5)	101	(12)	
0 2	3	4 8	9 11	12 23	0

META-SYMBOL ENCODED PROGRAM RECORD (first word)

Record Type (T)	Word Count (C)	Mode (Binary)	Folded Checksum	
(3)	(6)	111	(12)	
0 2	3 8	9 11	12 23	0

MONARCH ID RECORD (first word)

△ Character	Zero	Mode (IDL)	Space Character	Space Character	
101 111	000	0 0/1 0/1	110 000	110 000	
0 5	6 8	9 11	12 17	18 23	0

IDL = 001, Major Division - Level 1 ID Record
IDL = 010, Minor Division - Level 2 ID Record

MONARCH BOOTSTRAP LOADER (first word = WIM 012,2)

X	OPERATION	I	Address
0 1 0	0 1 1 0 1 0	0	0 0 0 0 0 0 0 0 0 0 1 0 1 0

MEMORY ALLOCATION

The amount of normal core allocation can be determined with the formula[†]

QENDMN + Loader + Symbol Table = size
(starting address) (length) (length)

	octal	decimal
QENDMN	7455	3885
Loader	2512	1354
Symbol Table[††]	454	300
Total	12643	5539

[†]The formula is valid for 9300 Computers; however, the amounts given are for a 900 Series Computer.

[††]The symbol table requires three words per definition.

The amount of core allocation during an update run can be determined with the formula:[†]

LDIOSR	+ Loader	+ Bootstrap Loader	+ Update	+ Symbol Table = size
(starting address)	(length)	(length)	(length)	(length)

	octal	decimal
LDIOSR	5540	2912
Loader	2512	1354
Bootstrap Loader	260	176
Update	4136	2142
Symbol Table[††]	454	300
Total	15344	6884

The following is a sample[†††] tape listing for a 920 Computer. The circled numbers refer to notes at the end of the listing.

```
                                  (2)                          (3)
    Δ1      LOAD      .
(1) Δ1      MONITOR       B       06/24/65      --LABEL--      042012 ....
        Δ2      CONTROL           06/24/65      1174           042004
        Δ2      TABLES            06/30/65      0425           042005
        Δ2      GMSGRD            04/01/65      0177           042006
        Δ2      CARD              04/01/65      0162           030004
        Δ2      MTYIO             04/01/65      0225           020019
        Δ2      MAGTP             04/30/65      0352           040004
        Δ2      MPRNT             06/14/65      0191           060005B
        Δ2      LDIOSR            05/20/65      0157           042007
        Δ2      FORTACT           04/05/65      0181           042014
        Δ2      FORTBIAS          04/01/65      0000           042015
        Δ2      ALGOLA            04/01/65      0041           042017
        Δ2      LDI2              06/30/65      0592           042030
        Δ2      TFMONRCH          04/01/65      0002           042008
(4)     Δ1      PRINT         B   06/14/65      0191           060005B....
        Δ1      MTAPE         B   04/30/65      0352           040004 ....
        Δ1      CDRP          B   04/01/65      0211           030005
        Δ1      PTYIO         B   04/01/65      0225           020019
    Δ1      META920           06/30/65                                 ....
        Δ2      ENCODER       B   06/30/65
        Δ2      MON1          B   05/20/65
        Δ2      MSCONTRL      B   04/01/65      0640
        Δ2      PREASSEM          06/30/65
        Δ2      PROC910           04/27/65
        Δ2      PROC920           06/17/65
        Δ2      PROC93JO          05/02/65
        Δ2      PROCB920          05/12/65
        Δ2      SHRINK            06/30/65
        Δ2      ASSEMBLR          06/30/65
        Δ2      PAS2              06/30/65
        Δ2      FINISH            06/30/65
        Δ2      CONCRD            06/12/65
        Δ2      CON2              06/12/65
    Δ1      LINKINIT      B       04/22/65      0665                ....
    Δ1      LINKZERO      B       04/22/65      2496
(5) Δ1      FORTRAN       B                     --LABEL---     202004H....
        Δ2      FC1               04/01/65      3568
        Δ2      FC2               04/01/65      0449
        Δ2      FC3               05/14/65      0449
```

[†]The formula is valid for 9300 Computers; however, the amounts given are for a 900 Series Computer.

[††]The symbol table requires three words per definition.

[†††]This sample is not valid for RAD MONARCH.

```
Δ1      FORTLOAD     B                    --LABEL---          012015
    Δ2      FL1              04/01/65     0231
    Δ2      FL2              05/24/65     0253
    Δ2      FL3              05/12/65     0025
Δ1      FORTLIB                           --LABEL---          202006H
    Δ2      SYS230           04/01/65     0108
    Δ2      ALOG             04/01/65     0138
    Δ2      EXP              04/01/65     0144
    Δ2      COS              04/01/65     0203
    Δ2      SQRT             04/01/65     0083
    Δ2      ATAN             04/01/65     0256
    Δ2      ABS              04/01/65     0013
    Δ2      IABS             04/01/65     0013
    Δ2      FLOAT            04/01/65     0004
    Δ2      IFIX             04/01/65     0008
    Δ2      SIGN             04/01/65     0021
    Δ2      ISIGN            04/01/65     0020
    Δ2      AMOD             04/01/65     0013
    Δ2      MOD              04/01/65     0009
    Δ2      AMIN             04/01/65     0065
    Δ2      DIM              04/01/65     0010
    Δ2      IDIM             04/01/65     0010
    Δ2      LOCF             04/01/65     0004
    Δ2      IF               04/01/65     0025
    Δ2      EXIT             04/01/65     0010
    Δ2      LINKING          04/01/65     0010
    Δ2      SYS160           04/01/65     0013
    Δ2      SYS201           04/01/65     0004
    Δ2      SYS202           04/01/65     0008
    Δ2      SYS203           04/01/65     0009
    Δ2      SYS204           04/01/65     0019
    Δ2      SYS205           04/01/65     0021
    Δ2      SYS206           04/01/65     0011
    Δ2      SYS207           04/01/65     0021
    Δ2      SYS210           04/01/65     0010
    Δ2      SYS211           04/01/65     0053
    Δ2      SYS212           04/01/65     0033
    Δ2      SYS213           04/01/65     0009
    Δ2      SYS214           04/01/65     0009
    Δ2      SYS215           04/01/65     0006
    Δ2      SYS216           04/01/65     0036
    Δ2      SYS217           04/01/65     0005
    Δ2      SYS220           04/01/65     0074
    Δ2      SYS221           04/01/65     0005
    Δ2      SYS222           04/01/65     0057
    Δ2      SYS223           04/01/65     0007
    Δ2      SYS224           04/01/65     0006
    Δ2      SYS225           04/01/65     0046
    Δ2      SYS226           04/01/65     0026
    Δ2      SYS227           04/01/65     0019
    Δ2      SYS231           04/01/65     0003
    Δ2      SYS232           04/01/65     0003
    Δ2      SYS233           04/01/65     0028
    Δ2      SYS235           04/01/65     0036
    Δ2      SYS236           04/01/65     0230
    Δ2      SYS241           04/01/65     0349
    Δ2      SYS242           04/01/65     0057
    Δ2      SYS243           04/01/65     0046
    Δ2      SYS244           04/01/65     0018
    Δ2      SYS245           04/01/65     0445
    Δ2      SYS776           04/22/65     1122
    Δ2      SYS777           04/01/65     1472                202005H
```

FORTRAN Library

```
⑥ Δ1      LIBRARY      B                    --LABEL---          202003D....
    Δ2        CDRP            04/01/65       0211                030005
    Δ2        CDR             04/01/65       0162                030004
    Δ2        PTYIO           04/01/65       0225                020019
    Δ2        PRINT           06/14/65       0191                    060005B
    Δ2        MTAPE           04/01/65       0351                040004
    Δ2        BDD             07/01/65       0184                203037B
    Δ2        BFS             07/01/65       0075                203014B
    Δ2        BDF             07/01/65       0171                203039B
    Δ2        DBD             07/01/65       0178                203036B
    Δ2        DFS             07/01/65       0052                203015B
    Δ2        DBF             07/01/65       0169                203038B
    Δ2        BID             07/01/65       0097                203012B
    Δ2        DIB             07/01/65       0083                203013B
    Δ2        LOG             07/01/65       0048                203009B
    Δ2        LGF             07/01/65       0152                203024C
    Δ2        EXP             07/01/65       0062                203008B
    Δ2        EXF             07/01/65       0159                203025C
    Δ2        ATN             07/01/65       0057                203007B
    Δ2        ATD             07/01/65       0158                203032B
⑦   Δ2        ATF             07/01/65       0249                203026C
    Δ2        CSD             07/01/65       0148                203034B,033
    Δ2        CSF             07/01/65       0217                203028C,027
    Δ2        COS             07/01/65       0032                203018B,006
    Δ2        DSQ             07/01/65       0082                203035B
    Δ2        FSQ             07/01/65       0071                203029B
    Δ2        SQR             07/01/65       0084                203019B
    Δ2        FFF             07/01/65       0082                203011C
    Δ2        FSN             07/01/65       0124                203010B
    Δ2        FLD             07/01/65       0208                203023C
    Δ2        DPN             07/01/65       0010                203022B
    Δ2        DPD             07/01/65       0076        203022B,040B,017B,016B
    Δ2        LDP             07/01/65       0006                203022B
    Δ2        STD             07/01/65       0006                203022B
    Δ2        LTP             07/01/65       0008                203020B
    Δ2        STP             07/01/65       0009                203020B
    Δ2        LQP             07/01/65       0010                203021B
    Δ2        STQ             07/01/65       0011                203021B
    Δ2        TRACE        B  04/01/65       0544                260003B
Δ1      SYMBOL          05/06/65                                      ....
    Δ2        LOADER       B  06/20/65
    Δ2        PSI             05/06/65
    Δ2        CSI             05/06/65
    Δ2        MSI             05/06/65
    Δ2        PBO             05/06/65
    Δ2        CBO             05/06/65
    Δ2        MBO             05/06/65
    Δ2        TLO             05/06/65
    Δ2        LLO             05/06/65
    Δ2        MLO             05/06/65
    Δ2        S1              06/27/65
    Δ2        S2              05/06/65
    Δ2        S3              05/06/65
    Δ2        M910            05/06/65
    Δ2        M920            05/06/65
    Δ2        M9300           05/06/65
Δ1      MEDIA        B        05/12/65       2408                000017C....
Δ1      TRACE        B        04/01/65       0544                260003B....
Δ1      RECON        B        05/18/65       1773                000022A....
Δ1      PRINTDGN     B     10/09/64.
Δ1      UPDATE       B                       --LABEL---
    Δ2        BOOTSTRAP       04/01/65       0141                042009
    Δ2        UPDATERT        04/22/65       1519                042011
⑧ Δ1      ALGOL        B        04/01/65       --LABEL---          242008
    Δ2        ALGOL1          04/01/65       4582                042018
    Δ2        ALGOLX          04/01/65       0501                042019
```

META-
SYMBOL
Library

```
Δ1      ALGOLOAD              04/01/65      --LABEL---
    Δ2      RL            B   04/01/65      0883              242009
    Δ2      INPUT             04/01/65      0036              042020
    Δ2      OUTPUT            04/01/65      0036              042021
    Δ2      ENDIO             04/01/65      0007              012019
    Δ2      ENDIOL            04/01/65      0006              012020
    Δ2      ACCTAP            04/01/65      0014              012021
    Δ2      PNCHTP            04/01/65      0008              012022
    Δ2      ACCEPT            04/01/65      0020              012023
    Δ2      TYPE              04/01/65      0008              012024
    Δ2      READ              04/01/65      0035              032002
    Δ2      PUNCH             04/01/65      0045              032003
    Δ2      PRINT             04/01/65      0074              062003
    Δ2      READIT            04/01/65      0073              042022
    Δ2      WRITOT            04/01/65      0056              042023
    Δ2      REWRTP            04/01/65      0360              042025
    Δ2      REWIND            04/01/65      0012              042024
    Δ2      SETIOT            04/01/65      0056              042026
    Δ2      TSTWRT            04/01/65      0046              042027
    Δ2      TREADY            04/01/65      0017              042028
    Δ2      INITFS            04/01/65      0029              012025
    Δ2      ABS               04/01/65      0014              012026
    Δ2      SQRT              04/01/65      0083              212005
    Δ2      1EXP              04/01/65      0154              212006
    Δ2      LN                04/01/65      0011              012028
    Δ2      EXP               04/01/65      0011              012027
    Δ2      LGF               04/01/65      0145              212018
    Δ2      EXF               04/01/65      0151              212007
    Δ2      SIGN              04/01/65      0022              012029
    Δ2      SINCOS            04/01/65      0200              212009
    Δ2      ARCTAN            04/01/65      0261              212010
    Δ2      MIN               04/01/65      0015              012030
    Δ2      MAX               04/01/65      0015              012031
    Δ2      MOD               04/01/65      0023              012030
Δ1      ALGORUN       B       04/01/65      --LABEL---
    Δ2      EXEC              04/01/65      1602              242011
    Δ2      FSCAN                04/01/65   1371              212012,13,1
    Δ2      LIST              04/01/65      0355              242010
Δ1      SYSEND
```

ALGOL Library

(9)

Notes:

1. If CONTROL routine is replaced (see "System Update Routine" in Section 4), the Δ1 MONITOR label card must precede the Δ2 CONTROL card and the binary deck; this rule is generally true for the first program in each logical file.
2. As a general rule, any record read by the MONARCH loader can be blocked (see "System Update Routine"); a blocked record is identified by the letter B in column 22. Columns 21, 23 and 24 must be blank.
3. Columns 25 through 72 may contain comments. In this listing, the comments are program approval date, core allocation, and catalog number. The statement -- LABEL -- in the core allocation column indicates the beginning of a logical file.
4. The four I/O handlers, PRINT, MTAPE, CDRP, and PTYIO, must appear in this order on the system tape and must follow immediately after TFMONRCH; i.e., no insertions may be made between any two of these subroutines or between TFMONRCH and PRINT.
5. FORTRAN II is available only on 900 Series Computers.
6. Any routine or processor that is to go on the system tape should be written prior to the library if it has any references to be satisfied from the library.
7. No programs within the scope of a Δ1 library may have an END card with a transfer address as the last card in a binary record.
8. The ALGOL system is supplied only on specific request.
9. A double EOF mark is written after SYSEND.

APPENDIX C. LOADER OPERATIONS

A general description of the MONARCH loader is given in Section 3. This appendix explains the main features of the loader in greater detail.

RELOCATION AND DATA RECORDS

A data record (record type 0) contains instructions and/or data to be stored in memory by the loader. Each data record contains a load address which is either the relative or absolute memory location in which the first data word (an instruction or a constant) is to be stored. The word in the data record containing the load address also contains an indicator that specifies whether or not the current load relocation bias is to be added to the given load address to obtain an effective load address. In other words the indicator specifies whether or not the data record contains relocatable words.

The effective load address determines the location in which the first data word is stored; successive data words are then stored in consecutive memory locations following the first word.

Relocation is performed according to the type of record being loaded. Four types of relocation are possible; these are described below. Record types are explained in Appendix F, "XDS Standard Binary Language."

LOAD RELOCATION

If the load relocation indicator is "set" for a given data word, the initial contents (i) of the rightmost m bits in that data word are replaced with k where:

$k = (i + b)$ modulo 2^m
$m = 14$ for XDS 900 Series Computers
$= 15$ for XDS 9300 Computers
$b =$ current value of load relocation bias

COMMON RELOCATION

If the blank COMMON relocation indicator is "set" for a given word, the initial contents of the rightmost m bits in that data word are replaced by k where:

$k = (b + c)$ modulo 2^m
$m = 14$ for XDS 900 Series Computers
$= 15$ for XDS 9300 Computers
$c =$ current value of COMMON relocation bias
$b =$ current value of load relocation bias

PROGRAMMED OPERATOR RELOCATION

If the POP relocation indicator is "set" for a given data word, the initial contents (n) of bits 3 through 8 of that data word are replaced by p where:

$p =$ operation code from POP table entry number n
$(p \geq 0)$ $(0 \leq n \leq 77_8)$

Note: n is the "relative" POP operation code and p is the effective POP operation code computed by the loader.

SPECIAL I/O RELOCATION

If the special I/O relocation indicator is "set" for a given data word, the following modifications are performed:

1. The rightmost m bits of d are replaced with k and the result is stored in α.
2. Bit 18 of the contents of α -b is replaced with δ and the result stored in α -b.

where:

d = initial value of the data word
α = effective load address of d
i = initial value of rightmost m bits of d
b = current value of load relocation bias
m = 14 for XDS 900 Series Computers
  = 15 for XDS 9300 Computers
k = (i + b) modulo 2^m
δ = b if (i + b) $\geq 2^m$ or 0 if (i + b) $< 2^m$

EXTERNAL LABEL REFERENCES AND DEFINITIONS

The loader is capable of handling (resolving) symbolic cross-references between separately assembled and/or compiled programs. External reference and definition items in binary records (type 1 records) provide the loader with the information needed to link together two or more separately assembled or compiled programs.

During the loading process, the loader maintains a (symbol) table of external label definitions and unsatisfied external references. There is no restriction on the order in which the definition of a label and the reference(s) to it appear in the input to the loader. The definition of a label may precede, or follow, some or all of the references to it. Note that it is permissible for any number of programs to contain references to a given label, provided that one program being loaded contains an external definition item for that label.

When the loader encounters an external definition item, it searches the symbol table for a previous definition of that label in the table; if there is one, the loader increments the duplicate definition counter and discards the new definition. If the search reveals that the label is already in the table as an unsatisfied reference, the loader uses the definition to satisfy all the references to that label and replaces the unsatisfied reference item in the table with the definition item. However, if that label does not occur in the symbol table (as a reference or as a definition), the loader inserts the external definition item in the symbol table.

The operand field of an instruction which references an external label requires special consideration. At the time of assembly, the operand field in this case will contain either zero or the relative address (in the same program) of the previous instruction which referenced that same external label.

A typical assembly containing references to an externally defined label EXLABL might appear:

```
                .
                .
                .
*01002    07600000          LDA     EXLABL
                .
                .
                .
*01172    23501002          STA     EXLABL, 2
                .
                .
                .
*01205    07701172          EAX     EXLABL
                            END
EXLABL    1205
```

At load time the loader uses the relative address (1205) of the last instruction containing a reference to EXLABL to down-chain (i.e., chain back through) the program (to 1172, to 1002), thus determining all instructions that reference EXLABL. The zero address portion of the instruction at 1002 indicates to the loader the end of the chain in that program.

Use of EXLABL + 2 in a source program could cause the loader to chain back to the wrong instruction, and for that reason external labels cannot be modified in this way at assembly time. However, since indexing and indirect addressing are modifications occurring at execution time, they are legal with externally defined labels. For example, to access the locations EXLABL and EXLABL + 1 (EXLABL externally defined), the following technique might be used.

The code:

```
        EAX     EXLABL
        LDA     0,2
        STA     TEMP
        LDA     1,2
        STA     TEMP + 1
TEMP    RES     2
         .
         .
         .
        END
```

would be equivalent to:

```
        LDA     EXLABL
        STA     TEMP
        LDA     EXLABL + 1
        STA     TEMP + 1
```

When the loader encounters an external reference item, it searches the symbol table to see if it already contains an external reference item for that label; if so, the external reference chain associated with the new external reference item is "linked" to the external reference chain associated with the existing table entry and the new external reference item is discarded. If the search reveals that the label is already included in the table as an external definition, the loader uses the definition to satisfy all the references to that label and then discards the external reference item. However, if that label does not occur in the symbol table (as a reference or as a definition), the external reference item is inserted in the symbol table; to be satisfied by a later definition.

EXTERNAL PROGRAMMED OPERATOR REFERENCES AND DEFINITIONS

The loader is capable of satisfying references to internal and external Programmed Operator (POP) definitions. External POP definition items, external reference items, and internal POP definition items provide the loader with the information needed to:

1. Satisfy external and internal POP references.
2. Maintain external POP reference and definition items in the loader's symbol table.
3. Construct a Programmed Operator transfer table in cells 0100_8 through 0177_8.

An "internal" POP definition is one that is recognized only within the scope of the program in which it occurs. No entries are made in the loader's symbol table for internal POP definitions or references.

Many of the loader functions performed in the processing of external POP references and definitions are also performed (by the same loader subroutines) for external label references and definitions. In particular, the functions of insertion and replacement of symbol table entries and the handling of duplicate definitions are the same both for external label and external POP items.

An internal POP definition supplies the loader with the (relative) sequence number that appears in bits 3 through 8 of data words referencing that POP and the address of the origin of the POP subroutine which corresponds to that sequence number. The loader assigns a new sequence number X ($0 \leq X \leq 77_8$) which it will use to replace bits 3 through 8 of all data words containing references to that POP definition. The loader also stores the address of the POP subroutine in the address field of cell X + 100_8. The reason for assigning new sequence numbers is to avoid possible conflicts with sequence numbers assigned in other, separately assembled (or compiled) programs that are also being loaded. A given POP

mnemonic (e.g., FLA) will be given a unique sequence number during loading, so any reference to FLA will "quote" this sequence number. It should be noted that the method depends on the assignment of sequential numbers, beginning with zero, to each different POP reference or definition in a given program.

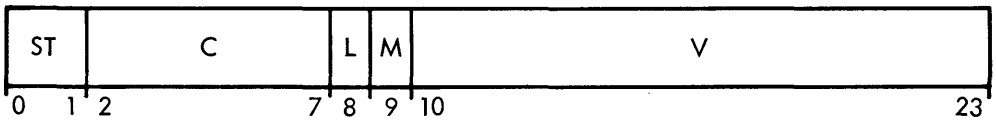
The primary difference in the treatment of internal and external POP definitions is that the external POP definition is represented in the loader's symbol table and hence it is recognized as a definition in all programs being loaded, not just the one in which it occurred. All of the remarks in the preceding paragraph relating to internal POP definitions apply equally to external POP definitions.

External POP reference items are inserted in the symbol table if no matching definition is found as a result of the symbol table search. When a matching external definition is supplied, the operation code assigned by the loader and the POP subroutine address are used by the loader to satisfy the reference. The POP operation code (X) replaces bits 3 through 8 of all data words containing references to that POP, and the POP subroutine address replaces the contents of the address field of cell X + 100_8. (See Programmed Operator Technical Manuals.)

SYMBOL TABLE ITEM FORMAT

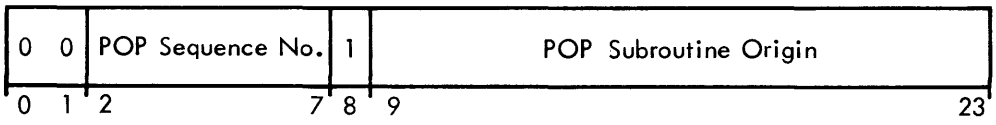
Each item in the loader's symbol table consists of a two-word symbol followed by a one-word value. The symbolic portion of a symbol table item consists of from one to eight alphanumeric characters, left justified within two computer words. Unoccupied character positions contain blanks (060).

The value portion of a symbol table item may be one of seven types. In each case, the left-most 9 bits identify the type and the right-most 15 bits contain the value.

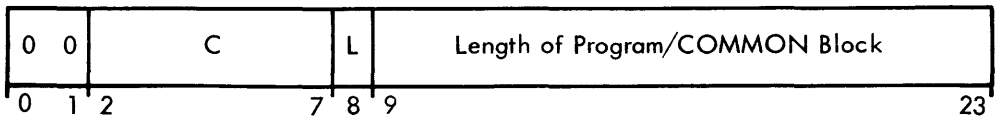


ST = Subtype
C = Code
L = 0 for label items
= 1 for POP items
M = 0 no doubly defined symbol
= 1 doubly defined symbol
V = Value

Internal POP definition

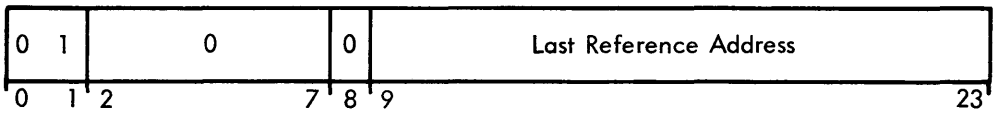


Common or Program Length

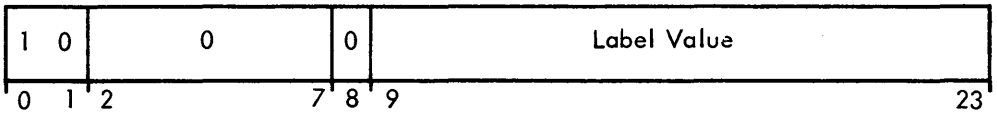


C = 1 if V contains program length
L = 1 if V contains length of labeled Common

External Label Reference

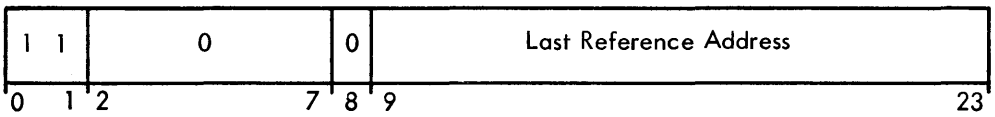


External Label Definition

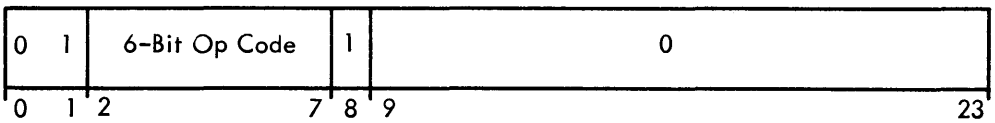


In the 900 Series MONARCH loader, bit 9 of the value word for a given entry in the symbol table is set to a 1 bit if the external definition associated with that entry has a duplicate definition. In the case of the 9300 MONARCH loader, bit 5 is of the value word is set to a 1 bit.

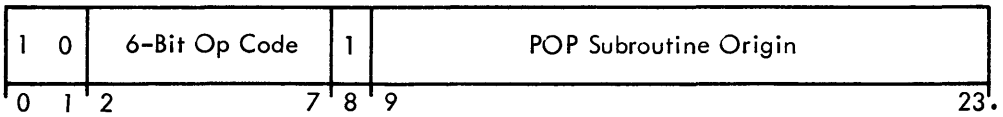
Labeled Common Reference



External POP Reference



External POP Definition



In the 900 Series MONARCH loader, bit 9 of the value word for a given entry in the symbol table is set to a 1 bit if the external definition associated with that entry has a duplicate definition.

Note: Items whose subtype is 00 are not entered in the table. POP items whose subtype is 11 are not entered in the table.

The origin of the POP subroutine is stored in the address field of the actual POP transfer table entry, at $X + 100_8$, when a POP definition is encountered. The actual 6-bit POP address (X) replaces the sequence number when the item is inserted in the symbol table.

Zero is stored in the address field of the actual POP transfer table entry ($X + 100_8$) when a POP reference item is inserted in the symbol table. The actual operation code replaces the sequence number.

The actual 6-bit POP operation code is also stored in the instruction code field of the POP transfer table entry whose address is obtained by adding 100_8 to the sequence number.

APPENDIX D. UPDATING META-SYMBOL ON MONARCH TAPES

Any portion of META-SYMBOL may be updated using the standard MONARCH ASSIGN, UPDATE, and COPY control cards. However, two sections of the system contain more than one deck, and during an update all portions of the labeled segment must be updated. These sections are (1) ENCODER (includes ENCODER, the proper POP deck, and S4B), and (2) MON1 (includes MON1 followed by the absolute loader).

When modified through reassembly, PREASSEM, SHRINK, ASSEMBLR, and FINISH must be converted to absolute form before being placed on the system tape. If the changes are by means of binary patches, the patches are inserted at the end of the absolute deck, just preceding the end card.

If the S4B portion of the ENCODER increases in size by more than a few words, the origin of the tables generated by the ENCODER must be changed. To move these tables, reassemble the ENCODER modifying the symbol TABLES defined at the end of the ENCODER by an EQU directive.

If the size of ASSEMBLER increases in size, the constant DTAB in the preassembler must be changed. This can be done by inserting a binary patch redefining this constant as needed. DTAB is the origin of the encoded dictionary.

If the size of the MSCONTROL program is increased, it is necessary to reassemble and move everything following it including the ENCODER, parts 1 and 2 of PREASSEM, and part 1 of ASSEMBLER. If this becomes necessary, the constants CPO, BPO, HED, CSEQ, and CORG must be appropriately redefined in both parts of PREASSEM, and the constants LITAB and PACKL must be redefined in part 1 of ASSEMBLER. The origins of these programs will also change.

A more thorough discussion of modification procedures may be found in the META-SYMBOL Technical Manual, XDS 900827. It is assumed that anyone attempting source level modification of META-SYMBOL will be familiar with that document.

APPENDIX E. FORTRAN LINKING

FORTRAN linking is available only when FORTRAN II is part of the MONARCH operating system. This operating system provides a modified Run-Time package, an initialization routine (LINKINIT), an additional subroutine (LINKING) for the FORTRAN library, and an additional control message (FORTLINK) for MONARCH.

FORTRAN linking allows the segmenting of FORTRAN programs and the loading and executing of these segments or "links" selectively under program control. A link is made up of a FORTRAN main program, subprograms, and FORTRAN library subroutines and functions. The loading of any link automatically erases the previous link. Only COMMON, modified FORTRAN Run-Time, and the MONARCH resident remain undisturbed during loading of a link.

Briefly, the FORTRAN linking procedure creates a link by loading a compiled FORTRAN program into core with necessary library subroutines and writing this program as an absolute dump preceded by a numerical identification (ID) on tape. The run-time package is not written on the tape. After all links have been written in this manner, each with its own unique ID, LINKZERO is loaded from the system tape and scans the linking tape, builds a table of ID's in the order of their appearance on the linking tape, inputs an initial list of the linking ID's to be followed at execution time, and executes the first specified link. The sequence of links to be executed is determined from CALL statements in the FORTRAN program (e.g., CALL LINK (N), CALL NEXT LINK, etc.).

Links may be written onto a separate magentic tape or onto the MONARCH system tape itself. Links may also be written on the X2 disc file instead of tape, in which case the ID's will be placed in the Directory disc file.

LINK PROCESS

Each of the various links in a chained program is a complete FORTRAN main program using any FORTRAN subprograms and library subroutines required. When control is passed from one link to another, the new link completely destroys the old one and execution begins at the first executable statement of the new main program. Only variables in COMMON are passed from one link to another. Care should be exercised that no link is large enough to overlay the COMMON from a previous link that may be needed by a following one. The easiest way to assure this is to reserve the same amount of COMMON in each link.

After all the desired links have been stored on magnetic tape or on the X2 disc file, a chaining sequence is initiated by requesting LINKZERO (i.e., via a LOAD 0, GO, 'LINKZERO' message). LINKZERO is loaded complete with a modified Run-Time package and the linking routine. The modified Run-Time package remains in memory to be used by succeeding links; the other links do not have Run-Time associated with them. LINKZERO determines the initial sequence of links to be performed and calls the first one into memory. The linking routine contains a push-down list of link numbers. Statements are provided for adding and removing links from the list and for calling them into memory. Any link may use the following statements:

1. CALL LINK (3)
   Call in link number 3. (The ID number for a link may be any three decimal digits.) The actual procedure here is to put 3 at the top of the push-down list and then call that link, which removes the number from the list.

2. CALL LINK (integer variable) or CALL LINK (integer expression).

3. CALL LINK (expression 1, expression 2, ..., expression n).
   Compute the n expressions (where n has a maximum value of 30) and place the resulting numbers at the top of the push-down list so that they will be called in the order 1, 2, ..., n before calling whatever was on the list previously. Note that 2, above, is a special case of this. Zero is a legitimate link number and, when called, causes a return to MONARCH.

4. CALL LINK (-2)
   Remove the top two items from the push-down list. Do not call any link but proceed to the statement following the call.

5. CALL LINK (negative expression 1, expression 2, expression 3, ..., expression n)
   If the first number is negative, remove the appropriate number of links from the push-down list before proceeding to enter the following numbers. Then call the link specified by expression 2. Only the first number may be negative. An error message (NEG. ARG) will result if any others are negative. Note that 4, above, is a special case of this.

6. CALL NEXT LINK
   Call the link specified by the top number in the push-down list and remove that number from the list. If the list runs out of numbers, control is returned to MONARCH.

   No parameters are used following CALL NEXT LINK. If present, they are ignored.

7. CALL FILL LINK (expression 1, expression 2, ..., expression n)
   Compute the n expressions, place the resulting numbers at the top of the push-down list, and return to the calling program.

When a link is called, it always begins at the first executable statement. However, one can effectively make it start at any number of places by providing a transfer instruction as the first executable statement:

1. Label the appropriate statements (e.g., 16, 2, 19).

2. As the first statement in the link, write a computed GO TO statement which references a location in COMMON; e.g., GO TO (16, 2, 19), I where I is in COMMON.

3. In the calling program prior to the CALL LINK statement, set the COMMON location (i.e., I) equal to the value of the desired label (in this example, I would be set to 1, 2, or 3).

GENERATING A LINK TAPE / LINK FILE ON DISC

LINKINIT is used to initialize a link file (X2). When X2 follows the system, which is the case when X2 is assigned to the same device (MTO for Tape MONARCH and DF for RAD MONARCH) calling LINKINIT will result in the removal of any previous links from the link file.

The MONARCH control message FORTLINK has the format

$\triangle$FORTLINK $P_1, P_2 \ldots, P_o$.

P_1 The identification number to be assigned to the link about to be written on magnetic tape or disc, may be any three decimal digits.

P_{2-9} Same as P_i for FORTLOAD.

The FORTLINK control message causes MONARCH to load and transfer control to the FORTRAN loader, which in turn loads a FORTRAN-compiled program and produces a storage map and/or label map as specified by the parameters P_2 through P_9. Then that FORTRAN-compiled program is written onto the link tape or disc as link number P_1. The links do not have to be written numerical sequence.

The program always uses X2 as the link tape; therefore, X2 must be assigned to a magnetic tape unit before MONARCH encounters the FORTLINK control message. X2 may be assigned to any tape unit including the MONARCH system tape. Alternately, X2 may be assigned to the disc.

The links themselves consist of two records. The first record is a 10-word record containing the link number. The second record is the core dump of the FORTRAN program. The core dump does not include the FORTRAN Run-Time package.

If the links are written on the X2 disc file, the first "record" consists of just 2 words: the program origin and the number of program locations (from origin to beginning of erasable storage).

When completed, the linking tape will consist of: an end-of-file mark, a short record and a long record for the first link, a short record and a long record for the second link, etc., and then finally another

end-of-file mark. It makes no difference whether the link tape is the system tape or a separate scratch tape. As far as the operation and execution of a linked program is concerned, the links are bracketed by end-of-file marks.

If the links are written on the disc, the link ID entries, automatically placed in the disc directory, render end-of-record and end-of-file marks unnecessary.

EXAMPLES

Example A:

```
△ASSIGN X2=MT3W, BI=MT0W.
△LOAD 0, GO, 'LINKINIT'.
△ASSIGN X1=PR1W.
△FORTLINK 29, MAP, X1.
△FORTLINK 16, X1, X1.
```

The link tape is to be on magnetic tape unit 3. The binary input is from the system tape. The LOAD control message causes the linkage initialization routine LINKINIT to be loaded. Magnetic tape 3 is rewound, two end-of-file marks are written on it, and the tape is rewound again. Control is then transferred to MONARCH, which makes the new unit assignment for X1 and loads the FORTRAN loader.

The FORTRAN loader loads the previously compiled FORTRAN program from the paper tape reader and produces a storage map of the program. The library is loaded from the system tape and the message "LOADING COMPLETE ..." is typed. Next, magnetic tape 3 (i.e., the link tape X2) is scanned forward to the second end-of-file mark; the tape is then backspaced over this end-of-file mark. The FORTRAN program, which was just loaded, is written onto the tape as link number 29, another end-of-file mark is written, and the tape is rewound. The message "LINK WRITTEN ON TAPE" is typed and control is returned to MONARCH.

The effect of the second FORTLINK control message is similar to that just described except that no map is produced and the previously compiled programs are on two separate pieces of tape (possibly a main program and a function). After the programs are loaded and the message "LOADING COMPLETE ..." is typed, magnetic tape 3 is again scanned forward to the second end-of-file mark; the tape is backspaced over this end-of-file mark. Then the FORTRAN program just loaded is written onto the tape as link number 16, another end-of-file mark is written, and the tape is rewound.

This process can continue until all the links have been written on tape. There is no restriction on the links except that no two links may have the same identification number.

Example B:

```
△ASSIGN X2=MT0W, BI=MT0W.
△LOAD 0, GO, 'LINKINIT'.
△ASSIGN X1=PR1W.
△FORTLINK 17, MAP, LMAP, X1.
△FORTLINK 169, X1, X1U.
```

In this example the system tape is to be used as the link tape. After the linkage initialization routine is loaded, the system tape is scanned forward to the first end-of-file mark, and a second end-of-file mark is written. Then the system tape is rewound, and control is transferred to MONARCH, which makes the new unit assignment for X1 and loads the FORTRAN loader.

The FORTRAN loader loads the previously compiled FORTRAN program from the paper tape reader, loads the library routines from the system tape, and produces a storage map and a label map. After the message "LOADING COMPLETE ..." is typed, the system tape is scanned forward to the second end-of-file mark and backspaced over this end-of-file mark. The just-loaded FORTRAN program is written onto the system tape as link number 17, another end-of-file mark is written, and the system tape is rewound. The message "LINK WRITTEN ON TAPE" is typed, and control is transferred to MONARCH.

The effect of the second FORTLINK control message is similar to that just described except that the previously compiled programs are on two separate pieces of paper tape. The second piece of tape is to be read unconditionally. After the programs and library subroutines are loaded, the message "LOADING COMPLETE..." is typed. The system tape is again scanned forward to the second end-of-file mark and backspaced over it. Then, the just-loaded FORTRAN program is written onto the system tape as link number 169, another end-of-file mark is written, and the system tape is rewound. The message "LINK WRITTEN ON TAPE" is typed, and control is transferred to MONARCH.

This process may continue until all the links have been written on the system tape.

Example C:

```
△ASSIGN X2=MT2W, BI=MT0W.
△LOAD 0, GO, 'LINKINIT'.
△ASSIGN SI=PR1W, BO=MT1W, LO=LP1W.
△REWIND MT1W.
△FORTRAN LO, BO.
△REWIND MT1W.
△ASSIGN BI=MT1W.
△FORTLINK 2, BI.
△REWIND MT1W.
△FORTRAN LO, BO.
△REWIND MT1W.
△FORTLINK 4, MAP, LMAP, BI.
```

In this example magnetic tape 2 is the link tape. The binary input is from the system tape to enable the loader to load the link initialization routine. The link tape is rewound, two end-of-file marks are written, and the tape is rewound again. Then control is transferred to MONARCH, which makes the unit assignments for the SI, BO, and LO units and rewinds magnetic tape 1.

Next the FORTRAN compiler is brought into core. The compiler accepts the source input from the paper tape reader, compiles the program, produces a listing on the line printer, and writes the compiled program onto magnetic tape 1. Then control is returned to MONARCH, which rewinds magnetic tape 1 and makes a new unit assignment for the binary input unit.

The FORTRAN loader is brought into core, and it loads the previously compiled program from magnetic tape 1. After the program and any necessary library subroutines have been loaded, the message "LOADING COMPLETE..." is typed. Next, the link tape is scanned forward to the second end-of-file mark and backspaced over it. Then, the just-loaded program is written onto tape as link number 2, another end-of-file mark is written, and the tape is rewound. Control is transferred to MONARCH, which rewinds magnetic tape 1 and calls in the FORTRAN compiler again.

The compiler accepts the second source program from the paper tape reader, compiles it onto magnetic tape 1, and produces a listing on the line printer. Then control is returned to MONARCH which rewinds magnetic tape 1 and calls in the FORTRAN loader.

The FORTRAN loader loads the compiled program from magnetic tape 1. The link tape is scanned forward to the second end-of-file mark and backspaced over this mark. Then the just-loaded program is written onto the link tape as link number 4, another end-of-file mark is written, and the tape is rewound.

EXECUTING A LINKED PROGRAM

Execution of a linked program is initiated when the routine LINKZERO is called into core via a LOAD control message. The LINKZERO routine

1. Scans the link tape to determine how many links are present and the order in which they appear on the tape and to record the identification numbers (link numbers) of the first thirty links.

2. Positions the tape between the two end-of-file marks that define the beginning and end of the link tape.

3. Stores information indicating the order in which the links are to be executed.

4. Locates the link to be executed first, loads it into core, and transfers control to it.

During the execution of the user's program, the links are located on the link tape by one of two methods:

1. If the link is one of the first thirty on the tape, its location will be known, and it can be located and read into core directly.

2. If the link is not one of the first thirty, its location is unknown; therefore, the link tape is positioned in front of the first link and is scanned forward until the desired link or an end-of-file mark is found.

If the link cannot be found, an error message (EOF STOP) is typed, and the computer halts. The link number for which the search was unsuccessful is displayed in the A register.

Example:

```
△ASSIGN X2=MT2W, BI=MT0W, SI=TY1W.
△LOAD 0, GO, 'LINKZERO'.
```

After LINKZERO is loaded into core, the message "LINKZERO LOADED" is typed. The link tape X2 is scanned forward to the first end-of-file mark. The tape is then read to determine the order of the first thirty links. This information is stored in a 30-word table. (More than thirty links may be written on the tape, but only the first thirty identification numbers are stored in the table.) After this information is stored, the tape is positioned in front of the first link on the tape. From this point on, the tape remains positioned between the two end-of-file marks that delimit the links.

LINKZERO then accesses whatever symbolic input device has been assigned—in this example, the typewriter. The user enters the list of link numbers in the order in which the links are to be executed. (Links may be executed any number of times and in any sequence.) This information is stored in a second 30-word table. The list of link numbers is entered under FORTRAN FORMAT (I4). The user must enter at least one link number; a maximum of thirty may be entered. (This feature enables the user to specify the first link and then have the links executed under program control; i.e., the first link calls another link which calls another, etc.; or the user may specify, via the input device at execution time, the order in which the links are to be executed, and each link is written to call the "next" link rather than a specific one.) The first link number entered by the user will be the first link executed. LINKZERO locates this link on the link tape, loads it into core, and transfers control to it.

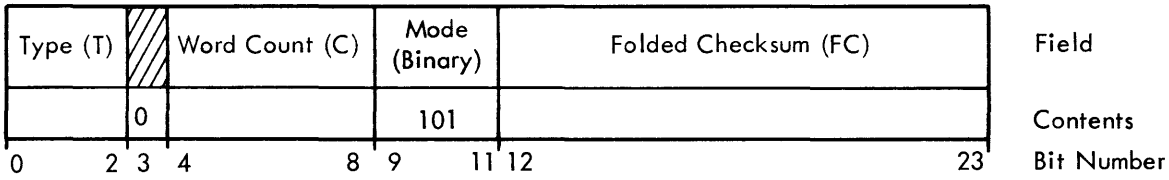
If the links are written on the disc, they decrease available disc storage. To free this storage space, another △LOAD 0, GO, 'LINKINIT' control message should be used, at the completion of the link job.

APPENDIX F. XDS STANDARD BINARY LANGUAGE

The following description specifies a standard binary language for XDS 900 Series and 9300 Computers. The intention has been that this language be both computer-independent and medium-independent. Thus, the language provides for handling Programmed Operator definitions and references even though the 9300 does not have this hardware feature; similarly, there is a provision for relocation relative to blank COMMON, even though this requirement is not present in XDS 900 Series FORTRAN II.

In the following description of the language, a file is the total binary output from the assembly/compilation of one program or subprogram. A file is both a physical and a logical entity since it can be subdivided physically into unit records and logically into information blocks. While a unit record (in the case of cards) may contain more than one record, a logical record may not overflow from one unit record to another.

1. CONTROL WORD – first word in each type of record



T	Record Type
000	Data record (text)
001	External references and definitions, block and program lengths
010	Programmed Operator references and definitions
011	End record (program or subroutine end)
100 ⋮ 111	Not assigned

C = total number of words in record, including Control Word

Note that the first word contains sufficient information for handling these records by routines other than the loader (that is, tape or card duplicate routines). The format is also medium-independent, but preserves the Mode indicator positions desirable for off-line card-handling.

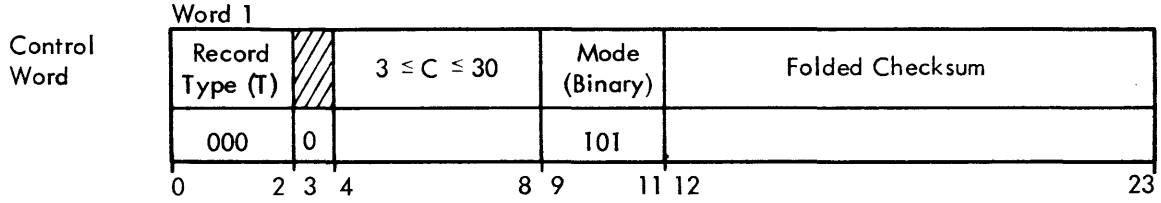
An exclusive OR checksum is used. If the symbol -- is used to denote exclusive OR, and W_i denotes the i-th word in the record ($1 \leq i \leq C$), then

$$FC = (W_1)_{0-11} \text{ -- } (S)_{0-11} \text{ -- } (S)_{12-23} \text{ -- } 07777$$

where

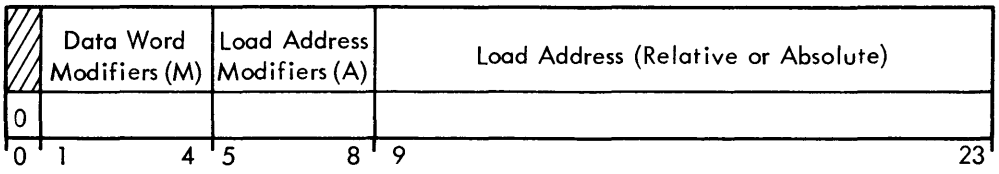
$$S = W_2 \text{ -- } W_3 \text{ -- } \ldots \text{ -- } W_c$$

2. DATA RECORD FORMAT (T=0)



Load Address Word

Word 2

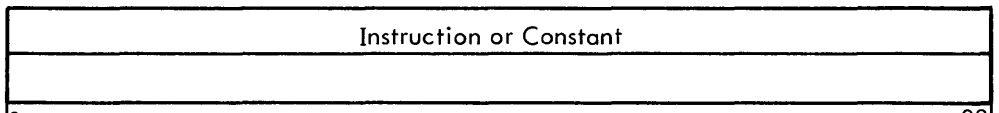


The presence of bits in field M indicates the presence of words n + 3, n + 4, n + 5, and n + 6 (shown below):

If bit position 4 contains a 1, word n + 3 (load relocation) is present.
If bit position 3 contains a 1, word n + 4 (common relocation) is present.
If bit position 2 contains a 1, word n + 5 (POP relocation) is present.
If bit position 1 contains a 1, word n + 6 (special I/O relocation) is present.

Data Word 1

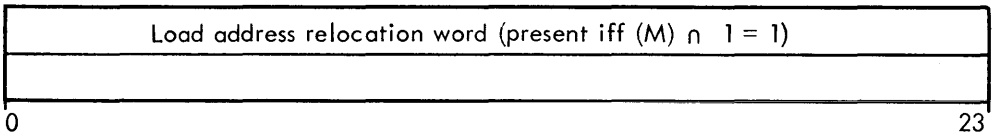
Word 3



Words 3 through n+2 contain instructions or constants (where $1 \leq n \leq 24$)

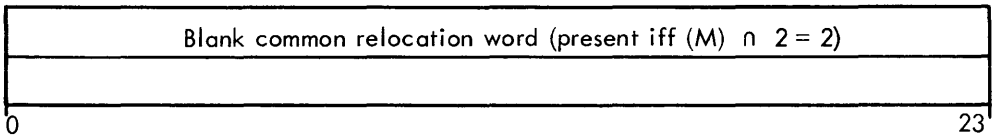
Load Relocation

Word n + 3



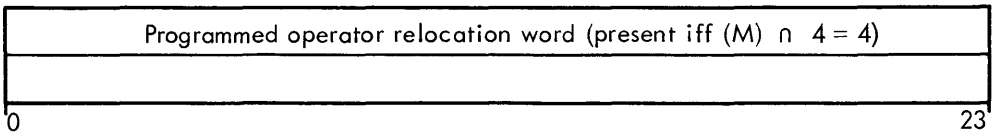
Common Relocation

Word n + 4



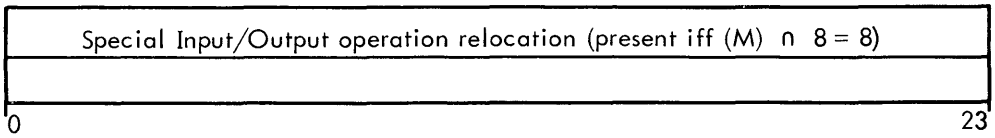
Programmed Operator Relocation

Word n + 5



Special I/O Relocation

Word n + 6



Words n + 3 through n + 6 are modifier words. Each bit in each of these words corresponds to a data word; that is, bits 0 through 23 of each modifier word correspond to data words 3 through n + 2 (where $1 \leq n \leq 24$). A bit set to 1 in a modifier word indicates that the specified data word requires modification by the loader. There are four types of modification (and hence four possible modifier words) which are indicated in data records. Presence of a modifier word in a data record is indicated by the M (data word modifier) field in the load address word.

The load address is subject to modification as indicated by the A field of the load address word as follows:

(A) = 0, absolute.
(A) ∩ 1 = 1, current load relocation bias is added to load address.
(A) ∩ 2 = 2, current common relocation bias is added to load address; the remaining bits of A are unassigned.
(A) = 3, illegal.

3. EXTERNAL REFERENCES AND DEFINITIONS, BLOCK AND PROGRAM LENGTHS (T = 1)
(Includes labeled COMMON, blank COMMON and program lengths)

Control Word

Word 1

Record Type (T)		4 ≤ C ≤ 31*	Mode (Binary)	Folded Checksum
001	0		101	
0 2	3	4 8	9 11	12 23

* From 1 to 10 items per record

Common Length or Program Length Item

1- to 8-Character Label

C1	C2	C3	C4
C5	C6	C7	C8
0 5	6 11	12 17	18 23

Length Word

Item Type		B	C	Length of Program or Common Block (L)
00	00000			
0 1	2 6	7	8	9 23

B = 1 if (L) is program length
C = 1 if (L) is length of a labeled common block

External Reference Item

1- to 8-Character Label

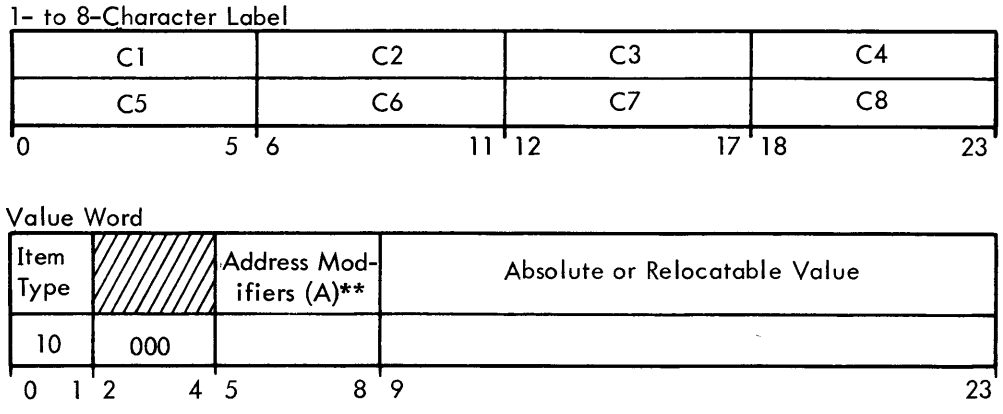
C1	C2	C3	C4
C5	C6	C7	C8
0 5	6 11	12 17	18 23

Chain Word

Item Type		Address Modifiers (A)**	Address of Last Reference
01	000		
0 1	2 4	5 8	9 23

** See data record, load address word, for interpretation.

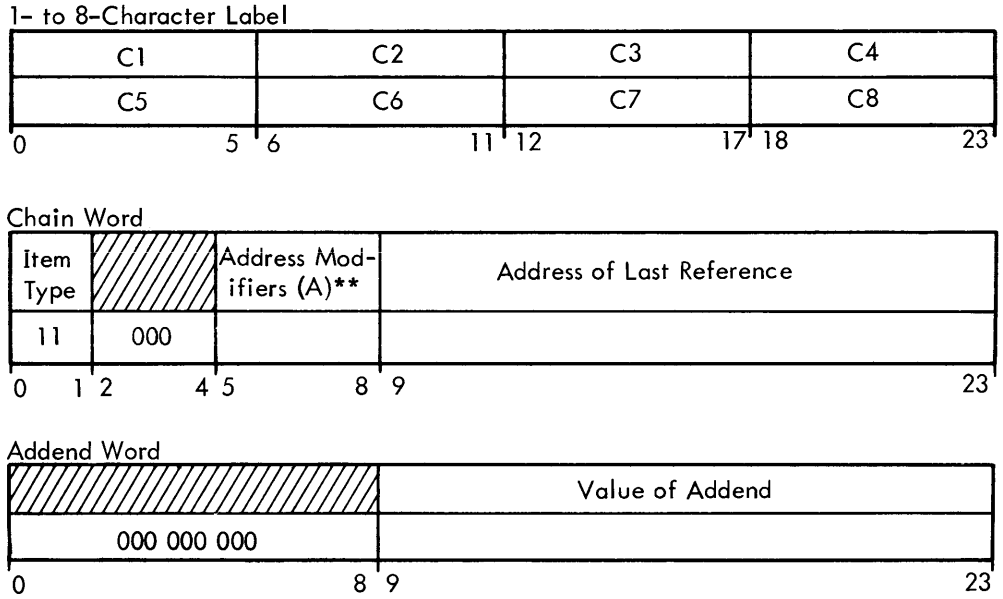
External Definition Item



** See data record, load address word, for interpretation

External symbolic definitions include subroutine "identification" as a subset and require no special treatment of subroutines with multiple names.

External Reference with Addend Word*



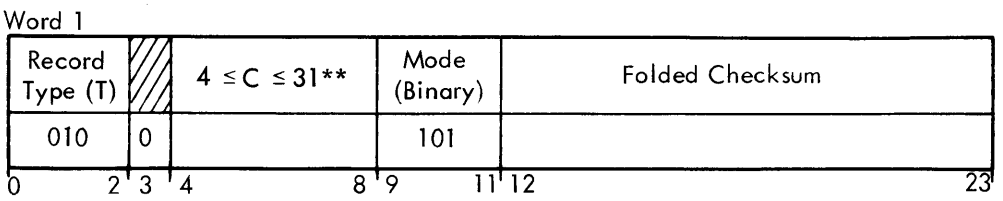
* One of these items for each unique reference; e.g., each of the following references is represented by a separate item:

A + 5, B + 5, B + 6, C + 2, C + 5

** See data record, load address word, for interpretation.

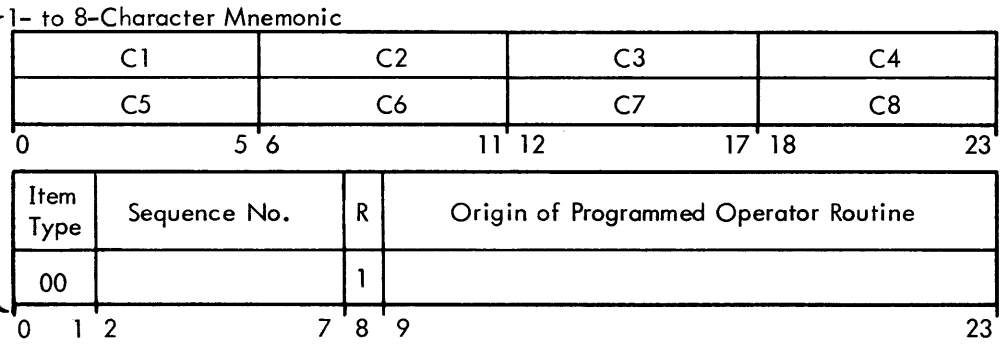
4. PROGRAMMED OPERATOR REFERENCES AND DEFINITIONS (T = 2)

Control Word

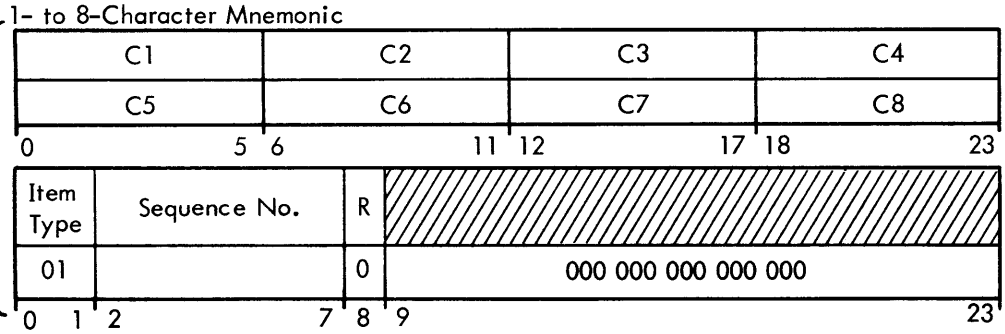


** From 1 to 10 items per record

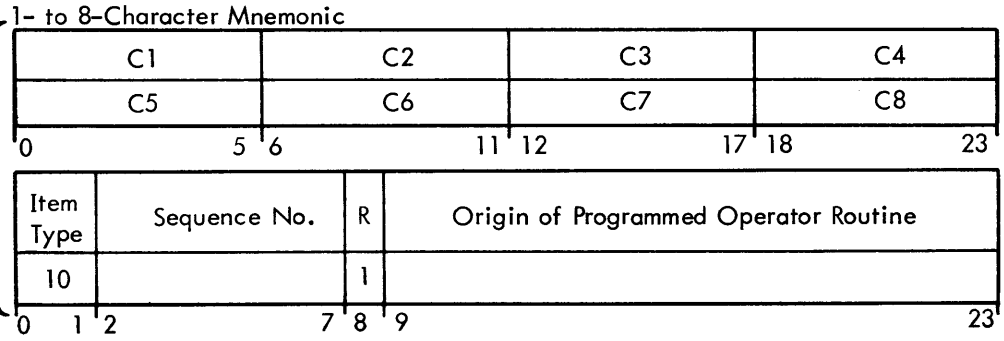
Internal Programmed Operator



Programmed Operator Reference



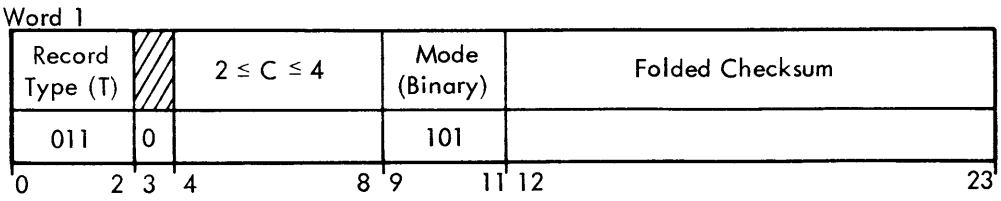
External Programmed Operator Definition



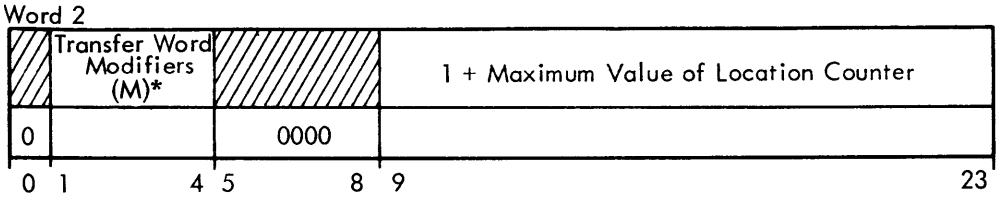
R = 1 if origin of Programmed Operator Routine is relocatable.
The sequence number indicates the order in which the definitions or references occurred in the source program.

5. END RECORD (T = 3)

Control Word

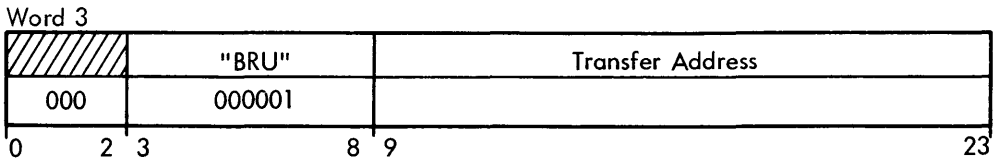


Length of Program



** See data record description for interpretation.

Transfer Word



This may be followed by a relocation word as described above in "Data Record Format," paragraph 2.

APPENDIX G. PROCESSOR DIAGNOSTICS

This summary of processor errors is provided for convenience of reference. The user should refer to the applicable processor reference manual for a more complete discussion.

META-SYMBOL (900 Series Only)

The standard abort message is

META-SYMBOL ERROR xx

where xx has the following values:

xx	Interpretation
01	Insufficient space to complete encoding of input.
02	Corrections to encoded deck but encoded input file is empty.
03	End of file detected while reading encoded input.
04	Insufficient space to complete preassembly operations.
05	Insufficient space to complete the assembly.
06	Data error. META-SYMBOL does not recognize the data as anything meaningful.
07	Requested output on a device which is not available.
08	Corrections out of sequence.
09	End of file detected by ENCODER when trying to read intermediate tape X1.
10	Request for non-existent system procedures.
11	Byte larger than dictionary (bad encoded deck).
12	Not encoded deck.
13	Checksum error reading system tape.
14	Preassembler overflow (ETAB).
15	Not used.
16	Data error causing META-SYMBOL to attempt to process procedure sample beyond end of table.
18	Improperly formatted or missing PROC deck series-specification card.
24	Shrink overflow.

Errors 05, 06, and 16 are accompanied by a printout that shows the value of certain internal parameters at the time of the abort:

LINE NUMBER	yyyyy	
BREAK 1	yyyyy	
LOCATION COUNTER	yyyyy	
UPPER	yyyyy	
LOWER	yyyyy	
BREAK	yyyyy	
SMPWRD	yyyyy	
LTBE	yyyyy	second pass only
LTBL	yyyyy	second pass only

(yyyyy represents the value of the particular item). The last six of these are useful in determining the nature of the assembler overflow.

After the appropriate message has been typed, control is transferred to MONARCH.

I/O ERROR MESSAGES AND HALTS

When an I/O error is detected, a simple message is typed and the computer halts. The message consists of a 2-letter indication of the type of error and a 2-digit indication of the I/O device. The letter indicators are defined below; the 2-digit number is the unit address number used in EOM selects (see applicable computer reference manual). The action taken if the halt is cleared depends upon the type of error and the device involved. There are three types of error.

BUFFER ERROR (BE)

1. Examples:

   BE11 buffer error while reading magnetic tape 1.
   BE52 buffer error while writing magnetic tape 2.

2. Action upon clearing the halt:

   a. Magnetic tape input - since ten attempts are made to read the record before the halt occurs, continuing causes META-SYMBOL to accept the bad record.

   b. Paper tape or card input - try again.

   c. Magnetic tape output - try again.

   d. Output other than magnetic tape - continues.

CHECKSUM ERROR (CS)

1. Examples:

   CS06 checksum error card reader.
   CS11 checksum error reading magnetic tape 1.

2. Action upon clearing the halt:

   Accepts bad record.

WRITE ERROR (FP)

1. Example:

   FP12 magnetic tape 2 file protected

2. Action upon clearing the halt:

   Checks again.

SYMBOL

Input/output errors during a SYMBOL assembly result in a typed error message stating the logical file (e.g. SI) and the cause for alarm (ER or FP). Halts will occur after the message has been typed only in the cases listed below. An erroneous magnetic tape input record is accepted by SYMBOL.

1. File-protected magnetic tape of RAD file (output) – Clearing the halt will result in another attempt to write the record.

2. Paper tape reader symbolic input error – To reread the record one must reposition the paper tape and clear the halt; otherwise, the erroneous record is lost.

3. Card reader input error – clear the halt. Unless the card is repositioned in the reader the record is lost.

RAD saturation can occur when the disc is unable to accept either a symbolic input record during SYMBOL's pass one or a binary output record during pass two. If overflow occurs, the file management routine types an appropriate message and control is returned to MONARCH.

FORTRAN II (900 Series Only)

Both the FORTRAN II compiler's input/output error messages and the FORTRAN loader's error messages are listed here.

I/O ERROR MESSAGES

For input and output, FORTRAN uses MONARCH's I/O handlers. If an error occurs during input or output, the compiler produces an error message of the form

FORTRAN I/O ERROR x

x	Interpretation
1	An error has been detected during listing. Compilation continues.
2	An error has been detected while reading magnetic tape. The message is output after ten unsuccessful read attempts. Compilation continues using the result of the last read.
3	An error has been detected while punching or writing the object program. Output is suppressed, and compilation continues.
4	No input device has been assigned. This is an irrecoverable error, and control is transferred to MONARCH.
5	RAD saturation has occurred. Compilation continues after the processor error switch is set.

These messages are printed on whatever listing device has been assigned. If no listing device has been assigned, the messages are not printed, but the indicated action is still performed.

There are three halts in the compiler. All display a flagged NOP in the C register with either a 7_8 or 70_8 as an address.

Address	Interpretation and Action
7	The computer will halt before reading paper tape for the first time, when a stop code has been read, or when the tape gap has been detected. Clearing the halt will allow compilation to continue.
70	An error has been detected while reading cards or paper tape. Clearing the halt will allow compilation to continue.

FORTRAN II LOADER ERROR MESSAGES

The error messages of the FORTRAN II loader running under MONARCH are as follows:

Message	Interpretation and Action
ERROR SWITCH SET	MONARCH is in the job mode (see JOB control message in Section 2), and the processor error switch has been set. Control is returned to MONARCH, which then attempts to read another control message.

Message	Interpretation and Action
PROGRAM TOO BIG	The program being loaded exceeds available memory. Loading continues, assuming an infinite memory. When loading is finished, the program size is typed out, and control is transferred to the MONARCH bootstrap.
READ ERROR, RELOAD LAST RECORD	The message is typed out, and the computer halts. If the input device is the RAD, control returns to MONARCH. Otherwise, clearing the halt allows loading to continue. This message is caused by: 1. Card read error. 2. Paper tape read error. 3. Magnetic tape read error (ten attempts have been made to read the record). 4. Disc file read error. 5. Checksum error.
ILLEGAL INPUT, RELOAD PROGRAM	This message is typed out, and control is transferred to the MONARCH bootstrap. This message can be caused by: 1. I2 interrupt while reading paper tape. 2. Input that is not a legal subprogram. 3. FORTRAN program heading improperly blocked. 4. Number less than 200_8 or greater than 244_8 assigned to a system subroutine reference. 5. Error in checking the sequential block count. 6. Labeled COMMON which is not accepted by the FORTRAN loader. 7. First word in a record not being a control word. 8. Type 0 records for POP relocation and special I/O relocation, type 2 records (POP references and definitions), illegal records (types 4 through 7), and external references with addend items which are not accepted by the FORTRAN loader.
EOF STOP or NO LINK (disc case)	Can occur only when FORTRAN linking is used (see Appendix E). The requested link cannot be found on the link tape or on the disc. The computer halts with the link number for which the search was unsuccessful displayed in the A register. Clearing the halt will cause the search to be repeated.
ILLEGAL INPUT FILE ASSIGNED TO DISC	The message is typed out, and control is transferred to MONARCH. This message is caused by the specification of a logical file other than BI to be loaded from the RAD.

APPENDIX H. RAD MONARCH

INTRODUCTION

RAD MONARCH is a modified version of Tape MONARCH, which is oriented to an XDS 925/930 computer with a 9367 RAD unit. The concepts are those of Tape MONARCH; i.e., to maximize the efficiency of the XDS 900 Series through minimization of interjob overhead, a factor which is in proportion to the ratio of peripheral I/O rates to memory cycle time (as job execution time diminishes in proportion to memory cycle time, interjob overhead plays an increasingly important role).

In Tape MONARCH, interjob overhead corresponds roughly to rewinding the system tape and reloading the non-resident operating system. In RAD MONARCH, no actual rewind is necessary, since access to the operating system is on a random basis.

USE OF RAD MONARCH

The generation of a RAD MONARCH system is accomplished in two phases:

1. The Tape MONARCH system is converted to a RAD MONARCH system by an UPDATE operation (this step is unnecessary if begun from a tape-situated RAD MONARCH system).

2. The resultant NST (New System Tape) is input to a SYSGEN operation that copies the RAD MONARCH system onto the RAD.

At this point, the Tape and RAD MONARCH systems are completely analogous since they are non-resident and bootstrapped into memory in between jobs from the medium on which they are resident, and control cards are identical except for the unit assignment of logical files to the RAD. For example:

Tape MONARCH

```
△ASSIGN  S = MT0, X1 = MT1, X2 = MT2, SI = CR, LO = CP, BO = MT3.
△META910 SI, LO, BO
```

RAD MONARCH

```
△ASSIGN  S = DF1A, X1 = DF1A, X2 = DF1A, SI = CR, LO = LP, BO = DF1A.
△META910 SI, LO, BO.
```

The sole difference is that the system (S) is assigned to the RAD and that X1, X2, and BO may be optionally assigned to the RAD. In all other respects the cards controlling a job are identical within the two systems. Since the X1, X2 and BO (BI) files are treated as sequential, this includes the use of REWIND cards wherever they are necessary in Tape MONARCH.

DIFFERENCES BETWEEN TAPE AND RAD MONARCH

	Tape MONARCH	RAD MONARCH
Residency	≃42 locations (approx.)	≃653 octal locations (approx.)
File Access	sequential	random
No. Files	= no. tapes	4:S, X1, X2, BO/BI
Sequential Access Time (avg.)	5 ms./record	17.5 ms./sector
Transfer rate	kc.	558.6 kc.

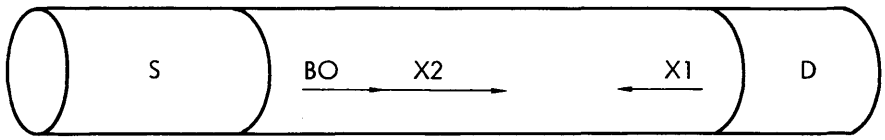
The primary difference between Tape and Disc MONARCH is that tapes permit variable length recording, whereas the RAD is read/written by sector. Therefore, a file management package is resident in

RAD MONARCH that "buffers" RAD I/O transfers. As a result, the sequential access time to the RAD is actually dependent on the ratio of the average record length to the sector size. Moreover, RAD MONARCH permits certain files to be accessed on a random basis. In this manner, loading time for system processors is reduced to a minimum.

Organization of RAD Storage

Example:

RAD



CORE X7777	UAT (11)	Unit Assignment Table
	BAT (11)	Business Assignment Table
	BRU DUMP (1)	Linkage to Memory Dump
	BRU TYPSM (1)	Linkage to Symbol Tbl. Dump
	CAT (11)	READ EOM/SKS Table
	FILE POINTERS (12)	File Maintenance Pointers
	RAD PACKAGE	RAD File Mgt. Package
	BUFFERS	RAD I/O Buffers (3)
X6564	QBOOT	Linkage to Bootstrap
	BRU QBOOT (1)	
00000	ERASABLE (1)	

RAD storage allocation is based on the concept of two concurrently active files. When the system is generated, these are S (the System File itself) and D (the Directory). After the system is generated, S and D are closed, and the remaining storage is used for the X1 and X2 files. Since X2 and BO (BI) are not simultaneously active, they both share the same file; X2 is defined as beginning immediately after the BO file.

GENERATION OF RAD MONARCH

RAD MONARCH is generated by first bootstrapping the RAD MONARCH tape, and issuing the following control messages:

```
△ASSIGN S = MTO, X1 = DFuc, SO = X.
△SYSGEN N
```

This sequence of messages results in the system being read from magnetic tape 0 on channel A and written on RAD unit "u", on channel "c", beginning with sector O. N is the number of files dedicated to the RAD MONARCH system. A "directory" of level 4 (△1) files is maintained from sector N downward. The original level 1 records are output on LO (LP or TY).

Example:

```
△ASSIGN S = MTO, X1 = DF1Y, LO = LP.
△SYSGEN O, 4095.
```

The X2 and X1 files are then begun following the S and D files. Level 1 identifiers are discarded from the output (X1) file during SYSGEN; level 2 records are abbreviated to the first four words.

When the end of the Old System Tape (OST) is reached, the S and D files are closed, the OST is rewound, and control is returned to MONARCH.

SYSGEN uses the resident RAD package in performing all RAD I/O operations.

COMMUNICATION WITH THE RAD FILE MANAGEMENT PACKAGE

The RAD File Management Package is a file-oriented Input/Output Control System (IOCS) that provides sequential access up to three concurrently active files (S, X1, and X2). The operations provided are Rewind, Open, Close, Read, Write, Write with Verification and Position.

The calling sequence consists of a BRM followed by a pointer to a file description table:

```
        BRM       *QSYS
        OP        FDT
        RETURN    ((A) = "NEXT" RAD ADDRESS)
          .
          .
          .
FDT     RES       1
        PZE       RECORD ORIGIN (CORE ADDRESS)
        DATA      RECORD LENGTH
        DATA      LOGICAL FILE NO.
```

OP is a pseudo instruction whose value indicates the appropriate file operation.

0:	Rewind
1:	Open
2:	Close
3:	Read
4:	Write
5:	Write with Verification (S and D only)
6:	Position

The first word of the file description table is set by the file management package to indicate abnormal conditions as follows:

Bit No.	Interpretation
1	I/O Transmission error
2	EOF (read only)
3	Storage overflow (write only)
5	File protect (write only)

The correspondence between physical and logical files is as follows:

File No.	File
0	S
1	X1
2	X2
3	BO/BI
4	D

The file management package behaves as though the RAD were word addressable. For each active file, it maintains the "current sector" in core until it has been fully utilized, at which time the buffer is emptied/refilled in preparation for the next sector. Since the sectors are ordinarily contiguous, the p-th word is located at word r in sector q, where

$$p = q * 64 + r, \quad 0 \leq q \leq 2047, \quad 0 \leq r \leq 63.$$

X2 and BO are actually the same file, but are distinguishable from the standpoint of rewind. RAD files are similar to tape files and are destructible when rewound, therefore, the same control cards can be used for either Tape or RAD MONARCH. SKIPFILE, SKIPREC, BACKFILE and BACKREC are, however, not provided for RAD files.

Each write on X1 and X2 is checked for infringement upon the other file (overflow of storage space). Both files are rewound between JOBs.

Since RAD logical files would be expected to begin at a sector boundary only in rare instances, it is necessary to open the file prior to recording or writing. For example, if the X1 file begins with the fourth word in sector 04053_8, the file management routine must first transfer sector 04053_8 to the core buffer before reading or writing can commence.

Similarly, all output files should be closed after the last write operation. The file management package automatically opens a file after a rewind operation.

The write with verification operation is used during SYSGEN to verify, by post-read, every RAD write operation. Since the X2 file is not active during SYSGEN, its buffer is used for the post-read operation and is compared word-by-word with the contents of the S or D buffer. The user is cautioned not to use write with verification, since he might inadvertently affect X2 or BO in doing so.

The position operation is used by MONARCH to position S randomly, and like the rewind, it is automatically followed by an open operation. To utilize this option, the contents of A must contain the RAD address to which positioning is requested when the file management routine is entered.

Additional technical information on the File Management Routine may be found in the MONARCH Technical Manual 90 06 16.

APPENDIX I. XDS CHARACTER CODES

Characters Typewriter	Characters Printer	Internal XDS Code	Card Code	BCD Code on Magnetic Tape	Characters Typewriter	Characters Printer	Internal XDS Code	Card Code	BCD Code on Magnetic Tape
Ø	0	00	0	12	-	-	40	11	40
1	1	01	1	01	J	J	41	11-1	41
2	2	02	2	02	K	K	42	11-2	42
3	3	03	3	03	L	L	43	11-3	43
4	4	04	4	04	M	M	44	11-4	44
5	5	05	5	05	N	N	45	11-5	45
6	6	06	6	06	O	O	46	11-6	46
7	7	07	7	07	P	P	47	11-7	47
8	8	10	8	10	Q	Q	50	11-8	50
9	9	11	9	11	R	R	51	11-9	51
Space	Blank	12	8-2	12[c]	Car. Ret. ![a]	![e]	52	11-0[d]	52
# or =	=	13	8-3	13	S	S	53	11-8-3	53
@ or	'	14	8-4	14	*	*	54	11-8-4	54
:	:	15	8-5	15	]	]	55	11-8-5	55
>	>	16	8-6	16	;	;	56	11-8-6	56
√	√	17	8-7	17	△	△	57	11-8-7	57
& or +	+	20	12	60	b	Blank	60	Blank	20
A	A	21	12-1	61	/	/	61	0-1	21
B	B	22	12-2	62	S	S	62	0-2	22
C	C	23	12-3	63	T	T	63	0-3	23
D	D	24	12-4	64	U	U	64	0-4	24
E	E	25	12-5	65	V	V	65	0-5	25
F	F	26	12-6	66	W	W	66	0-6	26
G	G	27	12-7	67	X	X	67	0-7	27
H	H	30	12-8	70	Y	Y	70	0-8	30
I	I	31	12-9	71	Z	Z	71	0-9	31
Backspace ?[a]	?[e]	32	12-0[d]	72	Tab ‡[a]	‡[e]	72	0-8-2	32
.	.	33	12-8-3	73	,	,	73	0-8-3	33
¤ or)	)	34	12-8-4	74	% or (	(	74	0-8-4	34
[	[	35	12-8-5	75	¬	¬[e]	75	0-8-5	35
<	<	36	12-8-6	76	\	\	76	0-8-6	36
‡ Stop	‡[e]	37[b]	12-8-7	77	** Delete	**[e]	77[b]	0-8-7	37

NOTES:

(a) The characters ? ! and ‡ are for input only. The functions Backspace, Carriage Return, or Tab always occur on output.

(b) On the off-line paper tape preparation unit, 37 serves as a stop code and 77 as a code delete.

(c) The internal code 12 is written on tape as a 12 in BCD. When read, this code is always converted to 00.

(d) The codes 12-0 and 11-0 are generated by the card punch; however, the card reader will also accept 12-8-2 for 32 and 11-8-2 for 52 to maintain compatibility with earlier systems.

(e) For the 64-character printers only.

APPENDIX J. FORTRAN II RUN-TIME MAGNETIC TAPE OPERATIONS

Magnetic tape operation statements such as

READ INPUT TAPE expression, m, list

and

REWIND expression

will refer to logical unit K, where K is the (truncated) value of the expression.

Before execution of a FORTRAN II program, the logical unit numbers must be assigned to physical devices. This is done with ASSIGN control messages as follows: For magnetic tapes,

△ASSIGN LK = P,

where

K = 0, 1, . . . , or 7

and

P = MT0, MT1, . . . , MT7, MT0W, . . . , MT0Y, . . . , or MT7Y.

For the X1 and X2 files on the

△ASSIGN X1 = DF, X2 = DF, L6 = X1, L7 = X2.

Note that the order of assignment is from left to right; that is, X1 and X2 are dummy variables, and are used only to distinguish between two RAD files.

The message

△ASSIGN LK = DF

is always taken as an abbreviation for

△ASSIGN X1 = DF, LK = X1,

where

K = 0, 1, . . . , or 7; i.e., without a dummy variable, X1 is assumed.

Thus the above example can be shortened to

△ASSIGN L6 = DF, X2 = DF, L7 = X2.

The value of K in the system label, LK, corresponds precisely to the value of K in the FORTRAN II magnetic tape operation statement such as

WRITE TAPE K, list

even when "tape" K has been assigned to one of the disc files. Only the X1 and X2 files of the disc are available for this application.

ERROR CONDITIONS

Figure 4 on the following pages illustrates run-time errors for 900 series FORTRAN II with and without RAD, and indicates, for each of the errors,

message typed

whether a halt occurs

cause of the error

contents of register at time of halt, if such information may be useful or if it may be changed before proceeding

result if program is allowed to continue.

Message	Halt	Explanation
AGTO	X	Assigned GO TO – Variable never assigned. Variable displayed in X. Result: Branch to (effective address determined by variable) + 1.
ARGM	X	Argument Mode – Argument of wrong mode given to FORTRAN subprogram. Proper mode is fixed if A = 0, floating if A = 01000000. Dummy address of argument displayed in X. Result: Argument used as if its mode were correct.
ARGN		Argument Number – Wrong number of arguments given to FORTRAN subprogram. Result: If too many, extra ones ignored. If too few, whatever arguments remain in erasable storage will be used.
CARD	X	Card "READ CHECK" or "FEED CHECK" error – If "READ CHECK" light is on, the last card read was in error. Place it back in the hopper. If "FEED CHECK" light is on, the offending card is still in the hopper. It probably has a wrinkled leading edge. Result: Try to read the card again.
CGTO		Computer GO TO – Value outside allowable range. Result: Go to first statement number in list.
CRDS		Card Reader Not Ready – Program has waited 15 seconds for reader. Place cards in reader and press start. Result: Program continues to wait for reader. Typeouts occur in 5-minute intervals.
EFIA		E, F, I, or A Needed in FORMAT – Unable to output variables. Result: Proceeds without outputting variables.
EIOL	X	End Input/Output List encountered without prior initialization. Result: Proceeds without taking any I/O action.
EXP		Exponential Function – Argument greater than 176. Result: Answer set to maximum floating-point value.
FCHt	X	FORMAT Character Illegal – The illegal character is displayed as the fourth character (t) in the message typed out. Result: Begins scan for next specification; i.e., treats character as if it were a comma.
FORL	X	FORMAT Label Error – The scalar variable referenced by an I/O statement has not been assigned a FORMAT statement label. Result: The contents of the (effective address determined by the variable) + 1 is used as the address of the start of the FORMAT statement.
FORM	X	FORMAT Missing – I/O statement references something else. X = address of supposed FORMAT. A = first word of supposed FORMAT. Result: Scans supposed FORMAT.
FORP	X	FORMAT Pointer Error – The address in the I/O list pointing to the FORMAT statement is not in an acceptable form; i.e., HLT, BRU or AGT (112 or 113). X = address of pointer. A = bad pointer. Result: Pointer at address specified by X is treated as if it had form specified by a HLT.
FXIO	X	Floating or Fixed Data requested for I/O without prior initialization. Result: Proceeds without taking any I/O action
ICHt	X	Input Character Illegal – The illegal character is displayed as the fourth character (t) in the message typed out. Result: Begins scan for next field; i.e., treats character as if it were a comma.

Figure 4. FORTRAN Run-time Errors

Message	Halt	Explanation
IFSL		If Sense Light – Value not 1-24. Result: Assume sense light off.
IFSS		If Sense Switch – Value not 1-4. Result: Assume sense switch off.
INOV		Integer Overflow – Input value of integer quantity exceeds 8,388,607. Result: Number truncated to the least significant 24 bits.
LABL	X	Label Undefined – Result: Computer will not proceed.
LCRD	X	Last Card Read in Error – May or may not be caused by a validity check. Place the last card back in the hopper. Result: Try to read the card again.
LOG		Logarithm Function – Argument negative or zero. Result: Answer set to zero.
NO[	X	No left parenthesis in FORMAT statement – Result: Computer will not proceed.
N**F		Negative Number Raised or Nonintegral Power. Result: Computes (|N|**F).
OCTt		Non-Octal Character (t) encountered during input under octal FORMAT specification. Result: Character is truncated to 3 least significant bits.
PNCH		Card Punch Not Ready – Program has waited 15 seconds for punch. Make the punch ready. Result: Continues to wait. Typeouts occur in 5-minute intervals.
PRNT		Printer Not Ready – Program has waited 15 seconds for printer. Make the printer ready. Result: Continues to wait. Typeouts occur in 5-minute intervals.
PRTY	X	Parity Error During Input – Result: Processing continues using incorrect character.
REP[		Repeat Count Precedes Outermost [in FORMAT – Result: Where applicable, group repeat count is applied to entire FORMAT specification.
REP$		Repeat Count Precedes $ in FORMAT – Result: Repeat count is ignored.
SIZE	X	Size of Erasable Storage Exceeded – There is no unused memory in which to transfer arguments to subroutines. Result: Erasable storage will run into COMMON, if any, or out of memory.
SNLT		Sense Light – Value not 0-24. Result: Statement has no effect.
SQRT		Square Root Function – Argument negative. Result: Square root of absolute value of argument.
XPOV		Exponent Overflow on Input Datum – Result: List item set to positive maximum (approximately $.579 \times 10^{77}$).
XPUN		Exponent Underflow on Input Datum – Result: List item set to zero.
O**N		Zero Raised to Nonpositive Power – Result: (0**0) will be 1 or 1.0, and (0**NEGATIVE) will be the maximum possible integer or floating number, as the case may be.
[OVF	X	Nesting Level Exceeded – Limit on number of parenthesized groups of FORMAT specifications is normally 4 levels. Result: Higher levels of nesting are disregarded.
TPNO	X	Logical Unit Number Not 0,7 – Logical unit number displayed in A. Result: Number will be truncated and the low-order octal digit (0-7) will be used.

Figure 4. FORTRAN Run-time Errors (cont'd)

Message	Halt	Explanation
		For the remainder of the tape errors, the logical unit in error will be indicated as the fourth character (#) of the message typed out.
BKS#	X	Backspace – Failed 10 Times. Result: Proceed as if backspace has successfully taken place.
EOF#	X	End of File Reached During Reading – Result: Continue to read past end of file.
ETR#	X	End of Tape While Reading – Remove the finished tape and replace with next reel. Result: Continue reading.
ETW#	X	End of Tape While Writing – Remove the finished tape and replace with next reel. Result: Continue writing. This, in conjunction with ETR, facilitates writing and reading of multiple reels.
FPT#	X	File Protect – Attempted to write on tape which is file protected. Result: Check again.
LRR#	X	Long Record Read – READ TAPE (binary) has read a logical record which contains more information than is required by the I/O list. Result: The remainder of the record is skipped.
RDT#	X	Read Tape Error – Failed to read 10 times. Result: Proceed assuming read to have been satisfactory.
SRR#	X	Short Record Read – READ TAPE (binary) attempting to read more information from a logical record than is present. Result: Remaining items in the I/O list are supplied with words of zero.
TNR#		Tape Not Ready – Program has waited 3 minutes, 17 seconds for tape unit. Ready the tape unit. Result: Program continues to wait for tape unit. Typeouts occur in 3-minute, 17-second intervals.
WEF#	X	Write End of File Error – Result: Try again.
WRT#	X	Write Tape Error – Failed to write 5 times. Result: Proceed, assuming write to have been satisfactory.
DNR#	X	Attempted to read a nonexistent record on X1 or X2 disc file – Result: Return to MONARCH.
DWC#	X	Invalid word count in second word of a disc file record – Result: Return to MONARCH.
DTE#	X	Disc Transmission Error – $(A)_1 = 1$ implies read/write error. $(A)_3 = 1$ implies an unsatisfied request to write due to lack of any further available disc storage. $(A)_5 = 1$ implies write operation was attempted into a file protected area. Result: If rewind or open operation, try again. If read or write operation, proceed, assuming operation to have been satisfactory.

Figure 4. FORTRAN Run-time Errors

INDEX

A

Action subroutines, 16
ALGOL, 6, 9
ALGOL compiler, 1, 9, 17
  equipment configuration, 2
  source deck structure, 34
ALGOL loader, 11
ALGOLOAD, 6, 11
ASSIGN, 5, 16, 42

B

BACKFILE, 6, 13, 14
BACKREC, 6, 13, 14
Batch processing, 1, 35
Blank COMMON references and definition, 17
Blocking mode, 24
BOOTLOAD, 6, 14
Bootstrap, 1, 9, 14, 16, 17
Bootstrap loader, 1, 16, 17
Business Language Assignment Table (BAT), 6, 40, 41, 42

C

C, 4, 6, 11, 36
Card read/punch subroutine, 16
CARDTAPE, 6, 15
CDRP, 15
Character Set, 69
COMMON relocation, 49
Control messages, 4, 5
  ALGOL, 6, 9
  ALGOLOAD, 6, 11
  ASSIGN, 6, 16, 42
  BACKFILE, 6, 13, 14
  BACKREC, 6, 14
  BOOTLOAD, 6, 14, 15
  C, 4, 6, 11, 36
  CARDTAPE, 6, 15
  DISPLAY, 6, 12
  ENDJOB, 6
  EOF, 6, 14, 15
  FILLSYS, 5, 9
  FORTLINK, 6, 11, 55, 56, 57, 58
  FORTLOAD, 6, 10, 11
  FORTRAN, 6, 8
  JOB, 6
  LABEL, 6, 12
  LOAD, 6, 9, 10
  METAXXXX, 6, 7
  ONLINE, (see ASSIGN)
  POSITION, 6, 13
  REWIND, 6, 13
  RTFTLOAD, 6, 11
  RTFTRAN, 6, 8
Control messages (cont.)
  SET, 6, 12
  SHOW (see DISPLAY)
  SKIPFILE, 6, 13, 14
  SKIPREC, 6, 13, 14
  SYMBOL, 6, 7
  UPDATE, 6, 15
  WEOF, 6, 14
COPY, 23-29

D

Data records, 18, 48, 49, 60
Decimal integer, 5
Diagnostics, 66
  FORTRAN II, 68
  META-SYMBOL, 66, 67
  MONARCH, 38
  SYMBOL, 67, 68
DISPLAY, 6, 12

E

ENDJOB, 6
End-of-file (EOF) mark, 13, 15, 56, 57
End record, 64, 65
EOF, 6, 13, 15
Equipment configuration, 1, 2
Error switch, 6, 10, 16, 40 (see JOB, ENDJOB)
External label references/definitions, 17, 18, 19, 20, 50, 51, 60, 62
External POP references/definitions, 17, 18, 19, 20, 49, 51, 52, 53, 60, 62, 63, 64

F

FILLSYS, 5, 9
Format of control messages, 4
FORTLINK, 6, 11, 55, 56, 57, 58
FORTLOAD, 6, 10, 11
FORTRAN, 6, 8
FORTRAN II compiler, 1, 9, 17
  equipment configuration, 2
  linking, 9, 55-59
  source deck-structure, 31
FORTRAN loader, 9, 38, 39
Functions of MONARCH, 2
Furnishing control messages, 36

H

Halts (MONARCH program), 38

I

Input control messages, 9, 10, 11
  ALGOLOAD, 6, 11

Input control messages (cont.)
FILLSYS, 5, 9
FORTLINK, 6, 11, 56, 57, 58
FORTLOAD, 6, 10, 11
LOAD, 6, 9, 10
I/O device specification, 5
I/O subroutines (standard), 16, 17, 40, 41, 42

J

JOB, 6
Job mode, 6
Job switch, 1, 10, 16, 40

L

LABEL, 6, 12, 13
Level 1/2 ID records, 10, 12, 13, 23, 24, 25, 26, 27, 28
Library (MONARCH), 17
Loading from, 18, 19, 20
Line printer output subroutine, 16, 18
Linking (FORTRAN), 9, 55
Linking process (FORTRAN), 55, 56
Literal parameters, 5
LOAD, 6, 9, 10
Loader, 1, 9, 16, 17, 21, 22, 23
Loading the MONARCH system, 36, 43, 48
Load relocation, 49

M

Magnetic tape I/O subroutine, 16
Memory allocation, 17, 18, 44
Memory dump routine (see Octal Dump Routine)
META-SYMBOL assembler, 1, 2, 7, 17
equipment configuration, 2
error messages, 66
source deck structure, 31
updating of, 54
METAXXXX, 6, 7
MONARCH
bootstrap, 1, 8, 9, 14, 16, 17, 18, 45
ID records 10, 12, 13, 23, 24, 25, 26, 27, 28, 42, 43, 44
library, 17, 18, 19, 20
loader, 1, 9, 10, 16, 17, 18, 20, 48
system, 16, 43, 44
Monitor, 1, 2, 3, 16, 17
MTAPE, 16, 48

N

Numeric parameters, 4

O

Octal correction routine, 21, 22
Octal dump routine, 16, 21
Octal integer, 4
ONLINE, 6
Operating environment, 1
Operating procedures, 36

P

Paper tape/typewriter I/O subroutine, 16
Parameters of control messages, 3
literal, 5
numeric, 4
symbolic, 5
POP relocation, 49
POSITION, 6, 13
Preparing program decks, 30
ALGOL, 34
batch processing, 35
FORTRAN II, 31
META-SYMBOL, 30
PRINT, 16
Program, 18
Processor control messages, 6, 7
ALGOL, 6, 9
FORTRAN, 6, 8
METAXXXX, 6, 7
SYMBOL, 6, 8
Processor error switch (see Error Switch)
PTYIO, 16, 48

Q

QBINI, 18, 41
QBINO, 41
QBOOT, 17
QDUMP, 16, 21
QMSG, 18, 23, 41
QPESW, 40
QSYLDR, 22
QSYMO, 24, 41
QSYS, 24, 41
QSYSI, 41
QSYSP, 41
QSYST, 23, 41
QSYSU, 23, 41

R

Recovery procedure, 38
Relocation and data records, 19, 49, 50
Resident, 1, 16, 17
Restart procedure, 17
REWIND, 6, 13

S

Search subroutine, 13, 16
Separators, 4
SET, 6, 12
SHOW, 6, 12
SKIPFILE, 6, 13
SKIPREC, 6, 13
Special I/O relocation, 50
Standard binary language, 17, 60
Standard system routines, 16
Storage allocation, 18, 44

SYMBOL, 6, 8
SYMBOL assembler, 1, 8, 17
  equipment configuration, 1
  error messages, 67, 68
Symbolic parameters, 5
Symbol table, 8, 17, 38, 52
Symbol table typeout routine, 15, 18, 19, 20
Syntax of control messages, 4
System control messages 6
  ASSIGN, 6, 16, 42
  ENDJOB, 6
  JOB, 6
  ONLINE, 6
  SYSEND, 12, 28, 29, 43, 48
System maintenance messages
  UPDATE, 6, 15
System output, 38

T

Tape search routine, 13, 16
Termination of a run, 17
TYPSY5, 21

U

Unit assignment table (UAT), 1, 16, 17, 18, 40
UPDATE, 6, 15
Update routine, 1, 15, 16, 23
  blocking/normal mode, 24
  COPY message, 24, 25, 26, 27
Update routine (cont.)
  deletion, 25
  insertion, 24, 25
  replacement, 25
  retention, 25
  UPDATE, 6, 14
  update file, 23, 24
  updating META-SYMBOL, 54
Utility functions messages, 6
  BACKFILE, 6, 14
  BACKREC, 6, 14
  BOOTLOAD, 6, 14
  C, 4, 6, 7, 11, 36
  CARDTAPE, 6, 15
  DISPLAY, 6, 12
  EOF, 6, 14
  LABEL, 6
  POSITION, 6, 13
  REWIND, 6, 13
  SET, 6, 12
  SHOW, 6, 12
  SKIPFILE, 6, 13
  SKIPREC, 6, 13
  WEOF, 6, 14

V

Value of a parameter, 4

W

WEOF, 6, 14

XDS

Xerox Data Systems 701 South Aviation Blvd. / El Segundo, California 90245 (213) 772-4511 / Cable SCIDATA / Telex 674839 / TWX 910-325-6908

EASTERN TECHNOLOGY CENTER
12150 Parklawn Drive
Rockville, Maryland 20852
(301) 933-5900

PRINTED CIRCUITS DEPT.
600 East Bonita Avenue
Pomona, Calif. 91767
(714) 624-8011

TECHNICAL TRAINING
5250 West Century Blvd.
Los Angeles, Calif. 90045
(213) 772-4511

INTERNATIONAL MANUFACTURING SUBSIDIARY

Scientific Data Systems Israel, Ltd.
P.O. Box 5101
Haifa, Israel
04-530253, 04-64589
Telex 922 4474

SALES OFFICES

Western Region

Building Arts Bldg.
Suite G100
5045 N. 12th St.
Phoenix, Arizona 85014
(602) 264-9324

1360 So. Anaheim Blvd.
Anaheim, Calif. 92805
(714) 774-0461

5250 West Century Blvd.
Los Angeles, Calif. 90045
(213) 772-4511

Vista Del Lago Office Center
122 Saratoga Avenue
Santa Clara, Calif. 95050
(408) 246-8330

13701 Riverside Drive
Sherman Oaks, Calif. 91403
(213) 986-5510

*Wells Fargo Building
Suite 410
Del Amo Financial Center
21535 Hawthorne Blvd.
Torrance, Calif. 90503
(213) 542-5561

3333 South Bannock
Suite 400
Englewood, Colo. 80110
(303) 761-2645

320 Ward Avenue
Honolulu, Hawaii 96814
(808) 531-8257

Fountain Professional Bldg.
9004 Menaul Blvd., N.E.
Albuquerque, N.M. 87112
(505) 298-7683

El Paso Natural Gas Bldg.
Suite 201
315 E. 2nd South Street
Salt Lake City, Utah 84111
(801) 322-0501

400 Building
Suite 415
400 108th Avenue N.E.
Bellevue, Wash. 98004
(206) 454-3991

Midwestern Region

*International Towers Building
6th Floor
8550 West Bryn Mawr Avenue
Des Plaines, Illinois 60018
(312) 693-6060

Clausen Bldg., Suite 310
16000 W. Nine Mile Road
Southfield, Michigan 48075
(313) 353-7360

4410 Woodson Road
Suite 111
St. Louis, Missouri 63134
(314) 423-6200

One Erieview Plaza
13th Floor
Cleveland, Ohio 44114
(216) 522-1850

Seven Parkway Center
Suite 238
Pittsburgh, Pa. 15220
(412) 921-3640

Southern Region

State National Bank Bldg.
Suite 620
200 W. Court Square
Huntsville, Alabama 35801
(205) 539-5131

Orlando Executive Center
1080 Woodcock Road
Orlando, Florida 32803
(305) 841-6371

2964 Peachtree Road, N.W.
Suite 350
Atlanta, Georgia 30305
(404) 261-5323

Jefferson Bank Bldg.
Suite 720
3525 N. Causeway Blvd.
Metairie, Louisiana 70002
(504) 837-1515

4920 S. Lewis Avenue
Suite 103
Tulsa, Oklahoma 74105
(918) 743-7753

One Turtle Creek Village
Dallas, Texas 75219
(214) 528-6580

*2300 West Loop South
Suite 150
Houston, Texas 77027
(713) 623-0510

Eastern Region

10227 Wincopin Circle
Suite 716
Columbia, Maryland 21043
(301) 730-4900

20 Walnut Street
Wellesley Hills, Mass. 02181
(617) 237-2300

Brearley Office Building
190 Moore Street
Hackensack, N. J. 07601
(201) 489-0100

The Fortune Building
280 North Central Avenue
Hartsdale, New York 10530
(914) 948-2929

*1301 Avenue of the Americas
New York City, N.Y. 10019
(212) 765-1230

673 Panorama Trail West
Rochester, New York 14625
(716) 586-1500

P.O. Box 168
535 Pennsylvania Ave.
Ft. Washington Industrial Park
Ft. Washington, Pa. 19034
(215) 643-4250

Kogerama Building
Suite 212
No. 1 Tidewater
Executive Center
Norfolk, Virginia 23502
(703) 497-6811

Washington (D.C.) Operations

*2351 Research Blvd.
Rockville, Maryland 20850
(301) 948-8190

Canada

864 Lady Ellen Place
Ottawa 3, Ontario
(613) 722-8387

1009 7th Avenue, S.W.
Calgary 2, Alberta
(403) 265-8134

280 Belfield Road
Rexdale 605, Ontario
(416) 677-8422

1901 North Service Road
Trans-Canada Highway
Dorval, P.Q.
(514) 683-3755

INTERNATIONAL OFFICES & REPRESENTATIVES

European/African Headquarters

Scientific Data Systems
York House, Empire Way
Wembley, Middlesex
HA 9 OQB
England
(01) 903-2511, Telex 27992

Sweden

Nordisk Elektronik AB
Stureplan 3
Stockholm 7
(08) 24 83 40

Denmark

A/S Nordisk Elektronik
Danasvej 2
Copenhagen V
EVA 8285/EVA 8238

Norway

Nordisk Elektronik (Norge) A/S
Middelthunsgt. 27
Oslo 3
(2) 60 25 90

France

Compagnie Internationale
pour l'Informatique, C.I.I.
66, Route de Versailles
78-Louveciennes
Yvelines
951 86 00 (Paris area)

Israel

Elbit Computers Ltd.
Subsidiary of Elron
Electronic Industries Ltd.
88 Hagiborim Street
Haifa
6 4613

*Regional Headquarters