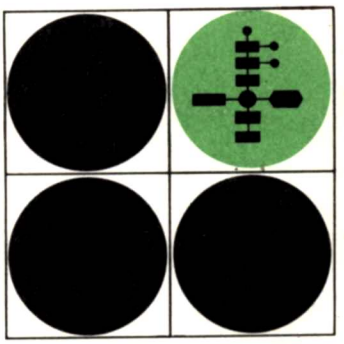


SDS MONARCH

REFERENCE MANUAL

SCIENTIFIC DATA SYSTEMS

SDS MONARCH REFERENCE MANUAL
900 SERIES COMPUTERS

November 1964

SDS

SCIENTIFIC DATA SYSTEMS/1649 Seventeenth Street/Santa Monica, California

Printed in U.S.A.

CONTENTS

Section		Page
I.	GENERAL DESCRIPTION	1
	Introduction	1
	How MONARCH Performs Its Functions	2
	Salient MONARCH Features	2
II.	THE MONARCH MONITOR	3
	Introduction	3
	Standard System Routines	3
	Communication with the Monitor — The MONARCH Control Message	5
	Termination of a Run	7
	Operating Environment for MONARCH	8
	Summary of Operating Instructions	8
III.	THE MONARCH LOADER	11
	Introduction	11
	The MONARCH "LOAD" Control Message	11
	Principles of Operation	13
IV.	MONARCH UPDATE ROUTINE	21
	Introduction	21
	The MONARCH "UPDATE" Control Message	21
	Controlling an Update Run — The Update File	22
	Copy Messages	23
	Contents of a Typical MONARCH System Tape	26
	Examples	26
	Error "Halts" During Update Runs	27
Appendices		
A.	THE MONARCH UNIT ASSIGNMENT TABLE (UAT)	29
B.	THE MONARCH SYSTEM TAPE	31
C.	CURRENT MONARCH FUNCTIONS AND CONTROL MESSAGES	35
D.	UPDATING META-SYMBOL ON MONARCH SYSTEM TAPES	55
E.	SDS STANDARD BINARY LANGUAGE	57

I. GENERAL DESCRIPTION

INTRODUCTION

MONARCH performs automatic execution of a sequence of independent or related programs without requiring operator intervention; yet it is simple in both construction and operation. Its modular construction not only contributes to ease of modification, but also results in a system that uses minimum core memory space during program execution.

The operating system is a basic, program execution package that provides the following functions under control of the MONARCH Monitor:

Loading and execution of standard system routines.

Examples

a. FORTRAN compilation
b. META-SYMBOL assembly
c. Punched card-to-magnetic tape conversion
d. MONARCH system updating

Loading and execution of previously compiled or assembled programs for checkout or production runs.

Examples

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

Combined assembly, loading and execution of programs for checkout or production runs.

Examples

a. Compile-and-go execution of FORTRAN programs
b. Assemble-and-go execution of symbolic programs

Combinations of one or more of the functions described above based on a "batch" of control information and programs accessible through one of the input media. This provides for execution of a mixed "batch" of programs to be executed under control of the monitor routine. The phrases "job stack" or "batched job stack" used below refer to the collection of control information, programs and data which are to be processed under control of the monitor routine.

Examples

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 the following functions be carried out in an arbitrary sequence determined by their order on the batched job stack:

   (1) Compilations
   (2) Compile-and-go execution of FORTRAN programs
   (3) Encoding of symbolic language programs
   (4) Encoding and assembly of symbolic programs
   (5) Encoding, assembly and execution of symbolic programs
   (6) Card-to-tape conversion using a standard system routine
   (7) Tape-to-printer conversion (listing) using a standard system routine

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.

Examples

a. The monitor routine can be requested to load standard input/output routines from the system tape prior to loading a program from the batched job stack for a production run of that program.

b. The monitor routine can be requested to load a dynamic (snap-shot) memory dump routine from the MONARCH library subsequent to loading a program from the batched job stack for a checkout run of that program.

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 MONARCH transfers control to it. These functions are communicated to MONARCH by means of control messages.

When requested to load a program and transfer control to it, MONARCH loads the program and relinquishes control over the computer and its associated peripheral equipment. The only way MONARCH can regain control of the computer is by reloading the MONARCH System from the system tape. This is done manually, by the console operator, or under program control by the program being executed.

The resident portion of MONARCH (i.e., those MONARCH instructions and data which remain in memory during program execution) consists of:

a. A routine to reload the MONARCH Operating System when control is transferred to it

b. The MONARCH Unit Assignment Table that preserves standard unit assignments from one program run to the next

c. The system processor error and job switches, contained in a word in memory, that are used to indicate that a subsequent program execution run is to be deferred due to errors encountered during assembly or compilation of that program

MONARCH is not an executive system. For purposes of this discussion, an executive system is one which performs some or all of the following functions during the execution of a program:

a. Handles the initial processing of all internal and external interrupts during execution of a program

b. Controls, and monitors, transmission of data to and from the computer memory and input/output units. Although requests for data transmission originate with the program being executed, the actual data transmission is controlled by an input/output routine operating as part of the execution system.

c. Controls, and monitors, communication with the console operator during the execution of a program. Typically, the medium for such communication is the console typewriter.

d. Automatically usurps control of the computer during execution of a program whenever certain abnormal conditions arise (e.g., memory parity error, nonrecoverable input/output error)

e. Automatically dumps memory and/or the contents of registers when a program error occurs

MONARCH performs none of the above functions. For example, during execution of the META-SYMBOL Assembly Program, subroutines which are part of the assembly program perform all input/output functions and routines within the assembly program handle all communication with the console operator.

SALIENT MONARCH FEATURES

The salient features of MONARCH are:

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

Core memory requirements for the monitor routine are minimized during program execution. That is, the monitor performs its functions between program execution, and MONARCH holds only those instructions and data required for continuity of operation and error-recovery procedures in core memory during program execution.

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

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

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

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, using SYMBOL or META-SYMBOL, can be incorporated into this operating system as a standard system routine.

II. THE MONARCH MONITOR

INTRODUCTION

The monitor is the highest level of authority in MONARCH. The monitor 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 functions between jobs and does not exercise control over the execution of a program once that program has been loaded and the monitor has transferred control to it.

One of the basic subroutines in the monitor is the system tape search subroutine. This subroutine searches the system tape by routine name when requested by the monitor. Its function is to locate (but not load or execute) a routine on the system tape.

Another basic subroutine analyzes, and interprets the contents of, the control messages which convey control information to the monitor. It also converts the parameters in control messages to a standard internal form.

Other subroutines, called "action" subroutines, perform the functions associated with specific control messages. For example, one action subroutine, associated with the "ASSIGN" message, modifies the contents of the MONARCH Unit Assignment Table based on the values of the parameters in "ASSIGN" messages (i.e., performs the new unit assignments). 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 can read control messages from the input devices from which control messages can be accepted.

That portion of the monitor which remains in core memory during program execution consists of the Monitor Bootstrap Routine and the Unit Assignment Table. In normal operation, the monitor does not initiate a rewind of the system tape before transferring control to the program to be executed. The Bootstrap is preserved in core memory during program execution, but since its function is limited to reading in the MONARCH loader and the MONARCH Monitor at the termination of the current job, it occupies only a few words of memory. The memory space occupied by the remainder of the subroutines comprising the monitor routine, and the space used by the MONARCH Loader, is available for use by the program being executed.

STANDARD SYSTEM ROUTINES

Standard system routines are those that exist on a MONARCH system tape and 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.

a. The Monitor. This routine is the heart of the operating system and is described in this section.

b. 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. See Section 3 (below) for a description of the MONARCH Loader.

c. 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.

See below for further discussion of the Monitor Bootstrap, the MONARCH Bootstrap Loader, and their relationship to each other.

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

a. The MONARCH Update Routine. With this routine, the user can create new MONARCH system tapes or update existing system tapes. This routine is described in Section IV.

b. The Standard Input/Output Subroutines. These subroutines enable other system routines to use them to perform all input/output functions required by those system routines. These I/O subroutines can be selectively loaded on an "as needed" basis. These subroutines are:

   Card Read/Punch Subroutine (CDRP)

   Magnetic Tape Input/Output Subroutine (MTAPE)

   Paper Tape/Typewriter Input/Output Subroutine (PTYIO)

   Line Printer Output Subroutine (PRINT)

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 which represent input/output functions to be performed and, finally, direct the loading of those, and only those, I/O subroutines needed to perform the indicated input/output functions. The MONARCH Update Routine relies on this feature to provide the Input/Output Subroutine needed to perform a specific update run.

c. The META-SYMBOL Assembly System. Presence of this routine provides a powerful and flexible assembly language and its processor.

d. The FORTRAN II System. Presence of these routines enables the MONARCH user to use the full capabilities of the SDS 900 Series FORTRAN II Compiler, Loader and Run-time Package.

See Section III, Automatic Selective Loading - Monitor Library, for a description of another "optional" MONARCH feature (the MONARCH Library) which can contribute greatly to the usefulness and efficiency of operation of a MONARCH System.

MONARCH is designed and constructed to facilitate the incorporation of additional system routines. The user can include in a MONARCH System any routine which meets the following requirements:

a. The routine must exist (on cards or paper tape) in SDS 900 Series Universal Binary Format.

b. 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.

c. 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.

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

b. 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 below).

c. The routine should be written to obtain unit and channel assignments for <u>all</u> of its input/

output functions from the MONARCH Unit Assignment Table.

COMMUNICATION WITH THE MONITOR-THE MONARCH CONTROL MESSAGE

The Control Message

When the MONARCH system is loaded, the monitor gains control of the computer and obtains the first item of control information from the input device indicated by the Unit Assignment Table entry for control messages. This item may be any legal control message.

Control messages can be obtained from 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 that message prior to the execution of the function requested by that control message. 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. Monitor tells the operator of the completion of a given function by typing out 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 control messages is the same. This format is as follows:

$$\Delta\, f.\quad \text{or}\quad \Delta f_{\wedge} p1,\ p2,\ \ldots\ .$$

where:

Δ (1 character) indicates beginning of message

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

pi (i=1, . . . , 24) is a symbolic, numeric or literal parameter which provides necessary control information related to the control function code (f). For example, a request for the system loader to load a program must indicate the initial load relocation bias for the program which is to be loaded. A maximum of 24 parameters can be specified in one control message.

. (1 character) indicates logical end of message. Physical end of message is indicated by end of record in the case of magnetic tape or cards, or carriage return in the case of paper tape or typewriter. The logical end of message is required only to introduce comments.

$\wedge$ indicates the minimum number of spaces (1) separating the function code from the first parameter.

, A separator "=", ">", "[", "<", "$", "*", "/", "(" or ")" can be substituted.

Regardless of the length of the record containing a control message, the routine which scans control messages will examine only the first 18 words (i.e., only the first 72 characters) of the record. The period indicating the end of the message must, therefore, occur prior to the 73rd character of the record.

The Control Message Function Code

The function code is separated from the beginning of message character (Δ) by zero or more spaces, and is followed by a space or a period. See Appendix C for a list of current function code mnemonics.

A period follows or can follow the function whenever one of the following is true:

a. The function code represents a control message for which the maximum number of parameters is zero, or

b. The function code represents a control message for which the minimum number of parameters is zero.

In all other cases, at least one parameter must be supplied in the message.

The MONARCH Monitor contains a table of function code entries each of which defines a function code mnemonic and a function (or action). The Monitor performs the function whenever it processes a control message containing that function code. The meaning of the parameters furnished in a control message is defined by a MONARCH "action" subroutine whose address is stored in the Function Code Table entry.

Control Message Parameters

The Monitor converts parameters in a MONARCH control message into a standard internal form. Except for double precision literals (see below), it represents parameters internally as single precision, 24-bit, binary values.

The Monitor converts the parameters in a control message to their internal representations in the same manner irrespective of which function code occurs in the control message. Hence, the "value" of a parameter is its internal representation as a 1 (or 2) word binary quantity or bit pattern, and it is the "value" of a parameter which is ultimately examined by the "action" subroutine associated with a specific function code. This method of parameter conversion usually allows many ways of representing a given value externally.

For example, the following control message parameters have identical internal representations or values:

'000A' = +17 = +021 = 00000021

MT1W = 00203611

See Appendix C for a description of the parameters associated with specific control messages as well as their interpretation by the associated MONARCH action subroutine. The several ways of representing parameters externally in control messages are described in the paragraphs which follow.

Numeric Parameters

A numeric parameter can be introduced in any one of the following forms:

a. An unsigned octal integer (1 to 9 characters), e.g., 047, 0, 077777777

b. An unsigned decimal integer (1 to 8 characters), e.g., 9, 0, 5283

c. A signed octal integer (2 to 10 characters), e.g., -062, -0, +023614

d. A signed decimal integer (2 to 9 characters), e.g., +9, -0, -58694319

Regardless of the form used, the parameter is represented internally as a single precision signed binary integer. Numeric parameters must have values less than 2^{24} if unsigned and less than 2^{23} if signed or decimal.

Literal Parameters

A literal parameter consists of from 1 to 8 alphanumeric characters enclosed in single quotation marks (SDS internal code of 14) and can be introduced as:

a. A single precision literal (1 to 4 characters), e.g., 'A', 'AB', 'ABC', 'ABCD' 'A.3,'

b. A double precision literal (1 to 8 characters), e.g., 'ABCDE', 'ABCDEF', '1234567', '12345678'

Single precision literals are represented internally as a left justified string of internal character codes (6 bits each) stored in one computer word.

Double precision literals are represented internally as a left justified string of internal character codes (6 bits each) stored in two computer words. For example:

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

as (α) = 21222324
($\alpha+1$) = 25262730

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

'ABC' is represented internally as:
021222360

Double precision literals are frequently used to designate program names for MONARCH search functions. (See the description of the MONARCH Loader for an example).

Symbolic Parameters

Symbolic parameters consist of from one 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 known to the MONARCH Operating System. Except for recognizing the correspondence between the symbol and its predetermined value, MONARCH does not interpret symbolic parameters (i.e., does not examine the parameter character by character to "compute" its value).

Use of a symbol not defined in MONARCH's table of symbolic parameters causes the control message in which that symbol occurred to be ignored and an error message to be typed.

The value (e.g., an octal parameter with the same value) can be substituted for any symbolic parameter in a control message.

The mnemonics given below are intended to be illustrative of the flexibility which can be obtained. If more than one parameter is required, all but the last must be followed by a comma. The first parameter is separated from the function code by a space.

CR	to designate the card reader when there is only one.
CRnh	to designate card reader #n on channel h.
CPnh	to designate card punch #n on channel h.
PRnh	to designate paper tape reader #n on channel h.
PPnh	to designate paper tape punch #n on channel h.
TY	to designate the console typewriter (#1 on the W buffer).
MTnh	to designate magnetic tape unit #n on channel h.
LP	to designate the line printer (#1 on the W buffer).
S	to designate the system tape.
IPnh	to designate incremental plotter #n on channel h.
MDnh	to designate magnetic drum unit #n on channel h.
DFnh	to designate disc file #n on channel h.

where:

h = W or Y
n = 0, 1, ..., 7

TERMINATION OF A RUN

When a program being executed under MONARCH reaches a normal conclusion, it should transfer control back to the monitor bootstrap in core memory 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 that minimum portion of the monitor routine that remains in memory during program execution.

When a program being executed under MONARCH detects a program or computer error which 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 which 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.

OPERATING ENVIRONMENT FOR MONARCH

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

a. An SDS 900 Series Computer system with at least 4096 words of core memory.

b. A Console typewriter to be used by the system to communicate information to the console operator.

c. One or more on-line magnetic tape units. The system tape is on a magnetic tape unit.

d. One or more of the following*:
   Card reader
   Paper tape reader

e. One or more of the following*:
   Card punch
   Paper tape punch
   Line printer
   Magnetic drum
   Magnetic disc

*May be replaced by magnetic tape equivalent.

SUMMARY OF OPERATING INSTRUCTIONS

Loading the MONARCH System

a. Mount MONARCH system tape on unit 0, W buffer; unit must be ready and at load point.

b. For SDS 910/920 Computers:
   (1) Set registers X, C and P and memory cell 1 as follows:

(00001)	=	03200002	(WIM 2)
(X)	=	77777771	(-7)
(P)	=	0	
(C)	=	00203610	(EOM 03610)

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

c. For SDS 925/930 Computers:
   Execute the magnetic tape FILL procedure

MONARCH loader and control system are then read into memory and MONARCH attempts to obtain the first control message either from the device indicated by (QMSG) = (0X7766*) or else from the console typewriter (unit 01 on the W buffer).

The following rule determines which device will be used:

If (QMSG) contains a legal unit, channel and I/O subroutine address, the first control message is taken from that device; otherwise, MONARCH sets (QMSG) to indicate the console typewriter (unit 01 on the W buffer) and attempts to read the first message from that unit.

Reloading the MONARCH System

To reload the system once it has been loaded:

a. Set (C) = 0 01 00001
b. Set RUN-IDLE-STEP to RUN

If this fails then:

a. Set (C) = 001 0X7751* (BRU QBOOT)
b. Set RUN-IDLE-STEP to RUN

*where

X=0 for 4k computer
X=1 for 8k computer
X=2 for 12k computer
X=3 for 16k computer

If this fails, employ load procedure described above in paragraph "Loading the MONARCH System".

Specifying the Control Message Medium

For changing the control message medium from the device currently in use to another input device, a control message (mnemonic function code "C") is provided. See above for a discussion of the manner in which the initial setting is determined. See description of "C" in Appendix C for details regarding the use of the "C" control message.

Example: to change the control message medium to card reader 1 on the W buffer:

"$\Delta C_{\wedge}$ CR1W."

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

Control messages can be supplied on punched cards, paper tape, magnetic tape, or manually via an on-line typewriter.

Furnishing Control Messages

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:

a. Console typewriter, the console operator should be prepared to furnish a control message, via that typewriter, whenever the input light for that typewriter is lit.

b. Paper tape reader, the console operator should make certain that a paper tape containing a control message is inserted in that paper tape reader and that the paper tape reader is in operation.

c. Card reader, the console operator should make certain that a card containing a control message is in that card reader's input hopper and that the card reader is "ready". MONARCH reads cards in binary mode and converts the card image to SDS internal code before analyzing the message.

d. Magnetic tape, the console operator should make certain that a reel containing a physical record which itself contains a control message is mounted on that tape unit and that the tape 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 (b, c, or d above), MONARCH reads successive records from that 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.

III. THE MONARCH LOADER

INTRODUCTION

In addition to serving as a standard system loader for user object programs, the MONARCH Monitor uses this routine to load from the system tape standard system routines such as the MONARCH Update Routine, META-SYMBOL, etc. The loader occupies upper memory, regardless of actual memory size (approximately $1600_8 \approx 900_{10}$ locations).

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 can be absolute or relocatable and can contain:

(a) External label references and/or definitions.

(b) External Programmed Operator (POP) references and/or definitions.

(c) Blank COMMON references and a definition.

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

(a) One or more data records (Record Type 0), and/or

(b) one or more external references or definition records (Record Type 1), and/or

(c) one or more Programmed Operator references or definition records (Record Type 2), and

(d) termination by an end record (Record Type 3) with or without a transfer address.

The last, or only program, in a series of programs to be loaded should have an end record (Type 3) with a transfer address and all of the programs preceding it should have end records (Type 3) 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 selective loading of appropriate subroutines from the MONARCH library. If this fails, the loader indicates that an error condition exists.

See Appendix E for a description of the data formats accepted by this loader. Note, however, that the MONARCH Loader does not accept labeled common definitions or references (Record Type 2, Item Types 1 and 3) and will treat labeled common references as format errors.

Programs to be loaded can be introduced on punched cards, magnetic tape, or paper tape, on units attached to either the W or Y buffer. The input/output subroutines within the MONARCH loader use neither interrupts nor interlace.

THE MONARCH "LOAD" CONTROL MESSAGE

Unit Assignment Requirements

When a LOAD message is issued to MONARCH, the Unit Assignment Table is assumed to contain the following information:

(a) QMSG contains the unit and channel designation for the peripheral device which is to furnish MONARCH control messages.

(b) QBINI contains the unit and channel designation for the peripheral device which is to furnish input (program(s)) to the loader. The unit must be a card reader, a magnetic tape unit, or a paper tape reader.

The following sequence of MONARCH control messages illustrates one means of setting up the Unit Assignment Table and requesting the MONARCH Loader to load one or more programs:

"△C TY1W."

"△ASSIGN BI = CR1W."

"△LOAD 010000, GO".

Parameters specified in "LOAD" Control Messages

Load Relocation Bias

The value of the first parameter in a LOAD message is interpreted by the loader as the load relocation bias 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. Normally this parameter will be expressed as a positive octal or decimal integer.

Loader Options

The loader interprets the value of the second parameter in a LOAD message as representing a 24-bit binary word, four (4) of whose bits represent ON/OFF switches with the remaining 20 bits not interpreted. The switch settings and the loader options they control are:

Bit No.	Status	Corresponding Loader Option
0	0	Halt after processing End Record with no transfer address. C register = 0 20 22222. The A register contains the load relocation bias K* computed for the next program to be loaded.
	1	No halt after processing End Record with no transfer address. The loader proceeds to load the next program using K* as load relocation bias.
1	1	Type the loader's symbol table on typewriter 1 on the W buffer.
	0	Loader's symbol table is not typed.
22	1	Return control (Via BRR) to program which calls the loader at the point when an End Record with no transfer address has been processed. The B register contains the load relocation bias K* computed for the next program to be loaded.
	0	Control is not returned to program which called the loader.
23	1	Halt immediately before executing "BRU" to transfer address in End Record. The C register contains the "BRU" instruction.
	0	No halt prior to executing "BRU" to transfer address in End Record.

*K = M+R

where M=2 plus the maximum value attained by the location counter as computed by the assembler or other translator which generated the program, and R = the value of the load relocation bias when the loader started loading the program.

The following symbolic parameters are frequently employed to designate the indicated parameter combinations:

Symbolic Parameter	Octal Equivalent	Corresponding Loader Options
STOP	01	Stops after each program is loaded, no symbol table typeout, and no return to routine which called the loader.

Symbolic Parameter	Octal Equivalent	Corresponding Loader Options
GO	040000000	No planned stops on End Records, no symbol table typeout and no return to routine which called loader.
TSTP	020000001	Same as STOP but loader's symbol table is typed.
TGO	060000001	Same as GO but loader's symbol table is typed.

Program Name (Search Key)

The third parameter in a load message may be omitted. If present it is treated as a 1- to 8-character program ID (name) which identifies a particular program to be loaded from the medium specified by QBINI. Normally this feature is used for selective loading from magnetic tape, but the function is equally valid for cards or paper tape.

When this parameter is present, its value is converted to a left adjusted, space filled, 8-character search key. Then the MONARCH Monitor causes records to be read from the unit assigned for binary input (QBINI) 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., 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 which processes the input as specified by the first two parameters. In the second instance, a "search-for-specified-program-failed" message is typed and the next MONARCH control message is requested.

PRINCIPLES OF OPERATION

Storage Allocation for the Loader and Its Symbol Table

When the MONARCH system is loaded, the MONARCH Loader is itself loaded into upper memory. This is true regardless of the memory size of the computer (with the bounds of 8 through 16K). The MONARCH Loader occupies approximately $1600_8 \approx 900_{10}$ memory locations in the range $X6200_8$ through $X7750_8$ (where X=1, 2, or 3).

At the time a request to load a user's program is initiated, the loader's symbol table contains a dozen external label definition entries which allow external references to locations within the resident portion of the MONARCH Operating System. The "resident portion" of MONARCH consists of: 1) a routine for reloading the MONARCH system (12 words), 2) the MONARCH Unit Assignment Table (currently 10 words), and 3) the Processor Error Switch and "JOB" Mode Switch (1 word). The resident portion of MONARCH occupies locations $X7751_8$ through $X7777_8$ (where X=1, 2 or 3).

The loader's symbol table (external label definition entries) initially occupies memory from about cell X6244 "down" to X6200. As each additional external symbol is inserted in the symbol table, it occupies the three (3) memory locations immediately below the last symbol table entry. Thus the loader and its symbol table occupy upper memory in that amount required by the size of the loader routine itself and the number of external symbol entries.

The loader gives an appropriate error indication whenever a new entry is to be made in the symbol table and this entry would "overlay" programs or data which have already been stored in memory by the loader.

Selection of the Appropriate Binary Input Subroutine

Based on parameters furnished by the MONARCH Monitor, the loader initializes one of 3 binary input subroutines (card, magnetic tape or paper tape) to read binary records from the unit and channel (W or Y buffer) indicated. The loader refers to a standard record area, in upper memory, to obtain a record from the selected binary input subroutine. Since the location of this area is the only interface between the loader and the input

subroutines, the loader itself is completely media independent. When searching and loading from the MONARCH Library the loader is communicating with the magnetic tape input subroutine via a communication cell, so that even this process is effectively media independent.

The Loading Process

The MONARCH Loader can be executed via the appropriate MONARCH control messages or directly, as a closed subroutine, by executing the instruction:

BRM QSYSLDR

where "QSYSLDR" is the externally defined label for the entry point to the loader. 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.

The loader commences execution by reading a record from the previously designated 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 (bits 0 through 2) of the control word is used to "point" to 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 for processing 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 on the value of Loader Option Parameter, does one of three things:

(a) Halt 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

(b) Return control to the program which called the loader (by executing a BRR QSYSLDR), with (B) = initial load relocation bias plus program length, and (A) = Loader Option Parameter.

(c) Set the load relocation bias equal to its previous value plus the length of the current program (as specified in the End Record) and continue loading records.

When an End Record with a transfer address is encountered, any indicated relocation is performed on the single data word in this record and the loader, depending on the value of the Loader Option Parameter, does one of two things:

(a) Halt with: (C) = the data word (i.e., transfer word) as modified by any relocation indicative present in the End Record
(A) = Loader Option Parameter
(B) = indeterminate

(b) Execute the data word (i.e., transfer word) after performing any indicated relocation of the address field. Normally, the data 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:

(a) Those within the (approximately) 1600_8 locations occupied by the loader and its input subroutines.

(b) Those locations pre-empted by the loader for its symbol table.

(c) Locations in which the loader is explicitly directed to store instructions or constants (i.e., data words supplied to the loader in Data Records).

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 which contains the load address also contains an indicator which specifies 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).

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 a data word is stored in the memory location specified by the effective load address, relocation indicators in the Data Record are examined to see if the data word is one whose binary value is to be modified in one or more of the ways described below.

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

$k = (i+l) \text{ modulo } 2^m$

$m =$ 14 for SDS 900 Series Computers

$l =$ current value of load relocation bias

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

$k = (l+c) \text{ modulo } 2^m$

$m =$ 14 for SDS 900 Series Computers

$c =$ current value of COMMON relocation bias

Programmed Operator Relocation. If the POP relocation indicator is "set" for a given data word, then the initial contents, j, of bits 3 through 8 of that data word are replaced by p where:

$0 \leqslant p =$ operation code from POP table $\#_j \leqslant 77_8$

Note: J 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, then the following modifications are performed:

(a) The rightmost m bits of d are replaced with k and the result is stored in α.

(b) Bit 18 of the contents of $\alpha-1$ is replaced with δ and the result stored in $\alpha-1$.

where:

$d =$ initial value of the data word

$\alpha =$ effective load address of d

$i =$ initial value of rightmost m bits of d

$l =$ current value of load relocation bias

$m =$ 14 for SDS 900 series computers

$k = (i+1)$ modulo 2^m

$\delta = 1$ if $(i+1) \geqslant 2^m$ and 0 if $(i+1) < 2^m$

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 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 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, 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, then 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) then it inserts the external definition item in the symbol table.

An external reference item contains a label and the relative address of the data word containing the last reference to that label in the program containing the reference item. For a given label in a given program, if A_1 were the address of the data word which contained the last reference to the label, then the external reference item will contain that address. The value, A_2, of the address field of the data word at A_1 will be either zero, meaning that it is also the first reference to the label, or non-zero, meaning that it (A_2) is the address of the next-to-last reference to the label. Within a program, each reference to the external label is "chained" to the previous reference in this fashion, with the data word containing the first reference being the "end" of the chain. The "end" of the chain is indicated by a zero address field in the data word. The chain "begins" with the address (A_1) in the external reference item, and is linked, by addresses, with the "end" of the chain. As a consequence of this method of "chaining" references and of the manner in which the loader "satisfies" the references in a chain, the Binary Records (Type 1) containing external references should always follow the last Data Record containing a reference (i.e., should always follow the Data Record containing the data word at A_1). "Satisfying" the reference in a chain consists of starting at A_1 and replacing the address field of each "link" in the chain with the value supplied by the corresponding external definition.

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, then 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) then 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:

(a) Satisfy external and internal POP references.

(b) Maintain external POP reference and definition items in the loader's symbol table.

(c) Construct a Programmed Operator transfer table in cells 0100_8 through 0177_8.

An "internal" POP definition is one which 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) operation code 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 operation code. The loader assigns a new operation code (X, $0 \leqslant X \leqslant 77_8$) 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 operation codes is to avoid possible conflicts with operation codes assigned in other, separately assembled (or compiled), programs which are also being loaded. It should be noted that the method depends on the assignment of sequential numbers, beginning with zero, to each 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 well 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 is 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 $X+100_8$.

Automatic Selective Loading from the MONARCH Library

Provision is made for referring to 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 then will 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 which have not been satisfied when the "END" record is encountered and then if there is a transfer address (indicating that it is the last, or only, program in a series of programs), 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 definitions in each library program in succession. When it encounters a library program which satisfies at least one such reference, it loads this program, and 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, then its definitions are "erased" 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, then 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.)

SUMMARY OF OPERATING INSTRUCTIONS

Load the MONARCH system. (See MONARCH OPERATING SYSTEM - Summary of Operating Instructions.)

"ASSIGN" a card reader, or a magnetic tape unit, or a paper tape reader as the "binary input" device (BI).

"△ASSIGN BI = CR1W" or
"△ASSIGN BI = PR1W" or
"△ASSIGN BI = MT1W" or
"△ASSIGN BI = MT3W"

Furnish a "LOAD" message to MONARCH

"△LOAD 010000, STOP" or
"△LOAD 0, GO" or
"△LOAD 0, 40000001, 'APROGRAM' "

The first example causes MONARCH to transfer control to the loader with instructions to use 010000_8 as load relocation bias, and to stop after each program is loaded. The second example specifies 0 as load relocation bias and specifies no "planned" stops during the loading process. The third example specifies 0 as load relocation bias and a "planned" stop after loading the program whose end record contains a transfer address. In this case, loading begins with the record following the level one MONARCH ID Record with "APROGRAM" in characters 9 through 16.

Planned Halts During Loading

(C) = 0 20 22222 when end record with no transfer address is encountered.

(C) = 0 01 xxxxx when end record with effective transfer address of xxxxx is encountered.

These halts occur only when requested (i.e., only when the value of the second parameter of the "LOAD" message so specifies).

Place RUN-IDLE-STEP to RUN to continue loading.

Halts for Abnormal Conditions

(C) = 0 20 00001 Buffer error (RUN to try again).

(C) = 0 20 00003 Illegal input format (RUN to ignore record).

(C) = 0 20 00004 Checksum error (RUN to ignore error).

(C) = 0 20 00006 LOADER symbol table overflow (no recovery).

(C) = 0 20 00007 Unsatisfied external label or POP reference after library search (no recovery). RUN to ignore error.

(C) = 0 20 00010 Duplicate external label or POP definitions (RUN to ignore - first definition encountered will be used).

Symbol Table Item Format

Each item in the Loader's symbol table consists of a two-word symbol followed by a one-word value.

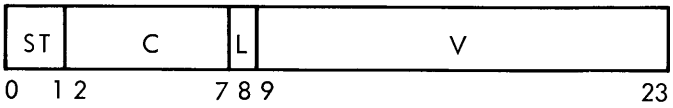
Symbol Format

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).

Value

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



ST: Subtype

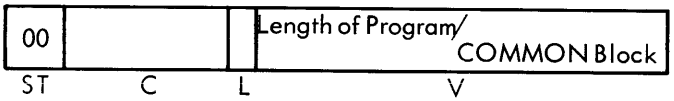
C: Code

L: 0 for label items, 1 for POP items

V: Value

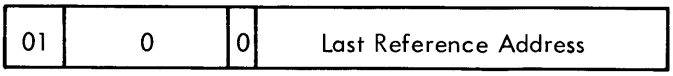
Internal POP definition

Common or Program Length

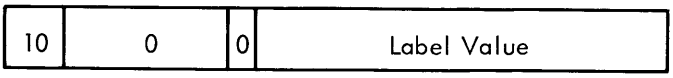


C = 1 if V contains program length
L = 1 if V contains labeled common length

External Label Reference

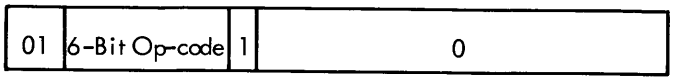


External Label Definition

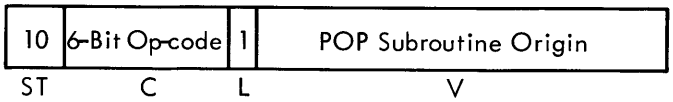


Labeled Common Reference

External POP Reference



External POP 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 actual6-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.

IV. MONARCH UPDATE ROUTINE

INTRODUCTION

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 these 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 Boostrap Loader is automatically recorded on a new system 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 these programs at execution time. Such programs should be preceded on the system tape by a special purpose loader which 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 SDS 900 Series Universal Binary Format. Data files to be recorded on a MONARCH System Tape must be presented to the Update Routine in this format or else in SDS 900 Series 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 either punched cards or paper 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). 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 listing of the MONARCH ID Records (Level 1 and Level 2) associated with <u>all</u> 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 must be presented in the order in which they are to be executed. No reordering of update input is performed.

THE MONARCH 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:

a. QMSG contains the unit and channel designation for the peripheral device which is to furnish MONARCH control messages and Update control messages. The unit must be a card reader or a typewriter.

b. QSYSU contains the unit and channel designation for the peripheral device which is to furnish any programs or data files which are to be inserted in the new system. The unit must be a card reader or a paper tape reader.

c. QSYST contains the unit (magnetic tape only) and channel designation for the peripheral device upon which the new system tape will be written.

d. QSYS 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, then the Update Routine assumes that no old system tape is present.

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, X2=MT1W, UI=CR1W"
"△UPDATE"
```

CONTROLLING AN UPDATE RUN-THE UPDATE FILE

Contents of an 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, and 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 can 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 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 either cards or on paper tape. (Samples of Update File listings are included in Example B 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) or 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 as a result. The Update Routine then copies records from the Update medium onto the new system tape until:

a. An end-of-file condition is detected (card reader only). (The Update Routine will then request a control message.)

b. A possible COPY message is encountered; i.e., a record which is neither binary nor encoded nor a MONARCH ID Record is encountered. (The Update Routine proceeds to analyze it as if it were a control message.)

c. 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 and set (A)≠ 0 to stop insertion and cause Update Routine to request a control message next, then set RUN-IDLE-STEP to RUN.)

d. A Level 1 MONARCH ID Record with "SYSEND$\wedge\wedge$" in characters 9 through 16 was written on the new system tape. [Both system tapes (old and new) are rewound and the MONARCH Monitor is loaded from the system tape on unit 0 of the W buffer.]

Note: When inserting under control of a Level 2 MONARCH ID Record and when the insertion is to become, or replace, the first subdivision of a major division of the system tape, then the Level 1 MONARCH ID Record for that major division must precede the Level 2 MONARCH ID Record in the Update File. (See Example B 5 at the end of this section.)

Programs to be inserted must be in SDS 900 Series Universal Binary Format. Data files must be in this format or else in SDS 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 not including those programs or data files in 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, and this program is not "copied" from the old system tape by duplicating the first record on the old system tape. All binary and encoded records written on the new system tape will have their checksums validated by the Update Routine.

COPY MESSAGES

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 1 to 8 character program names. The first character of a program name must be alphabetic and each remaining character must be either alphabetic or numeric. 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 which 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 1 or 2 program-names.

Relationship of COPY Messages to the Structure of MONARCH System Tapes

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 which 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:

e.g., "$\Delta 1_{\wedge\wedge\wedge\wedge\wedge\wedge}$LIBRARY$_{\wedge}$. . ."

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

e.g., "$\Delta 2_{\wedge\wedge\wedge\wedge\wedge\wedge}$COSINE$_{\wedge\wedge}$. . .".

"Minor divisions" of a MONARCH system are arbitrary subdivisions of a program or a data file which are recognized by the MONARCH Update Routine (see description of the MONARCH Loader for mention of another use of Level 2 MONARCH ID Records in connection with automatic library searching).

If an argument of a COPY message consists of one program-name, then that program-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 can be enclosed in parentheses), the first (leftmost) is assumed to occur in characters 9 through 16 of 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 MONARCH ID Record. In other words, the second program-name is assumed to refer to a subdivision of that major division of the old system tape which was identified by the first program-name.

Execution of a COPY message by the MONARCH Update Routine involves copying the MONARCH ID record(s) and any binary or encoded records which 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:

a. A binary or encoded record (b), is "in the scope of" A provided that no other Level 1 ID Record occurs between A and b on the system tape.

b. A binary or encoded record (b), is "in the scope of" X provided that no other Level 2 ID Record occurs between X and b on the system tape.

c. X is "in the scope of" A provided that no other Level 1 ID Record occurs between A and X on the system tape.

d. A binary or encoded record (b) is "in the scope of" both X and A if rules a, b, and c apply.

e. If a binary or encoded record is "in the scope of" X then it is not "in the scope of" Y.

f. If a binary or encoded record is "in the scope of" A then it is not "in the scope of" B.

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 which precede the Level 1 MONARCH ID Record with "LIBRARY$_{\wedge}$" in characters 9 through 16. This is the first record which is 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 MONARCH ID Record which 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 which consists 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 which precede 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 then 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, then it is the first record written on the new system tape in response to this COPY message. In either case the Update Routine then 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 which 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 "one 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 "one 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 "one 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:	Δ	COPY A THRU A.
	Δ	COPY B THRU B.
	Δ	COPY C THRU C.
Set 2:	Δ	COPY A.
	Δ	COPY B.
	Δ	COPY C.
Set 3:	Δ	COPY A THRU A.
	Δ	COPY B THRU C.
Set 4:	Δ	COPY A THRU B.
	Δ	COPY C THRU C.
Set 5:	Δ	COPY A THRU C.

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

*Matching of the second program name is inhibited until a Level 1 MONARCH ID Record whose name matches the first program name is encountered.

COPY Messages which will Terminate an Update Run

COPY messages of the form:

"△ COPY SYSEND." or "△ COPY a(b) THRU SYSEND."

causes the indicated COPY function to be performed, the update process to be terminated, and 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.

CONTENTS OF A TYPICAL MONARCH SYSTEM TAPE

1 record containing MONARCH
BOOTSTRAP and "LOADER" LOADER
1st record of MONARCH LOADER
⋮
Last record of MONARCH LOADER
△1 MONITOR
1st record of MONARCH
⋮
Last record of MONARCH
△1 PRINT
1st record of "PRINT" subr.
⋮
END record of "PRINT" subr.
△1 MTAPE
1st record of "MTAPE" subr.
⋮
END record of "MTAPE" subr.
△1 CDRP
1st record of "CDRP"
⋮
END record of "CDRP"
△1 PTYIO
1st record of "PTYIO"
⋮
END record of "PTYIO"
△1 LIBRARY
△2 SINE
1st record of "SINE" subr.
⋮
End record of "SINE" subr.
△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 9/1/64  .
    △2      TABLES          .
    △2      QMSGRD          .
    △2      LDIOSR 8/28/64  .
    △2      CARD            .
    △2      MTYIO           .
    △2      MAGTP           .
    △2      TFMONRCH        .
△1      PRINT     .
△1      MTAPE     .
△1      CDRP      .
△1      PTYIO     .
△1      CDR       .
△1      LIBRARY   .
    △2      CDRP 8/28/64    .
    △2      CDR 8/28/64     .
    △2      PRINT           .
△1      METASYM   .
    △2      ENCODER         .
    △2      MON1            .
    △2      MSCONTRL        .
    △2      PREASSEM        .
    △2      ASSEMBLR        .
△1      UPDATE    .
    △2      BOOTSTRAP       .
    △2      UPDATERT        .
△1      SYSEND    .
```

EXAMPLES (Continued)

EXAMPLES OF CONTROL MESSAGES AND PROGRAM SEQUENCES FOR UPDATE RUNS

1. To duplicate an existing MONARCH system tape:

```
"△C TY1W."
"△ASSIGN S=MT0W,X1=MT1W,UI=CR1W."
"△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."
"△UPDATE."
"△  COPY LOAD THRU R1."
"△1   RN    ".
"1st binary record of RN"
    :       :
"last binary record of RN (end record)"
"△  COPY R2 THRU SYSEND."
```

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

```
"△C CR1W."
"△ASSIGN S=MT0W,X1=MT1W,UI=CR1W."
"△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."
"△UPDATE."
"△  COPY LOAD THRU R6."
"△1    R7     ."
"1st binary record of 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 <u>first</u> 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."
"△UPDATE."
"△  COPY LOAD THRU CDR."
"△1    LIBRARY . . ."
"△2    NEW     . . ."
"1st binary record of NEW"
    :           :
"last binary record of NEW"
"△  COPY LIBRARY (CDRP) THRU SYSEND."
```

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 identified below along with the error message, the contents of the C register, and 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 (U1), in the texts of the error messages.

a. Message is: "UPDATE CONTROL MSG OR MONARCH ID RECORD NOT RECEIVED.".

The computer does not halt; instead, the Update Routine attempts to read a record from the control message medium.

The error condition results from failure to supply a COPY message (or a COPY message whose first 4 characters are not "△^^^"), or failure to supply a legal Level 1 or Level 2 MONARCH ID Record.

Corrective action: Since the erroneous record has already been read, and rejected, corrective action consists of supplying the missing (or corrected) COPY message or ID record via the control message medium (C).

b. Message is: "UNABLE TO POSITION NST AT LOAD POINT.".

(C) = 0 20 11111.

Corrective action: Manually position the tape at load point and clear the halt.

c. Message is: "UNABLE TO POSITION OST AT LOAD POINT. ".
(C) = 0 20 22222.
Corrective action: Same as b.

d. Message is: "ABNORMAL CONDITION, NST. ".
(C) = 0 20 11111.
Abnormal condition could be due to: buffer error, tape is file-protected, end-of-tape sensed, etc.
Corrective action: Manually ready the tape unit for writing and clear the halt.

e. Message is: "ABNORMAL CONDITION OST.".
(C) = 0 20 22222.
Abnormal condition could be due to buffer error, unit not ready, etc.
Corrective action: Manually ready the unit for reading and clear the halt.

f. Message is: "ABNORMAL CONDITION, UPDATE INPUT (UPD). ".
(C) = 0 20 33333.
See e above.

g. Message is: "ABNORMAL CONDITION, MESSAGE INPUT MEDIUM. ".
(C) = 0 20 04444.
See e above.

h. Message is: "ILLEGAL COPY MESSAGE, OR COPY MESSAGE REQUIRED. ".
(C) = 0 20 04444.
Corrective action: Supply the correct (or missing) COPY message via the control message medium and clear the halt (vice-versa if control message medium is an on-line typewriter).

i. Message is: "ILLEGAL RECORD FORMAT, OST. ".
(C) = 0 20 22222.
Corrective action: None. Clear the halt to ignore the error. See Appendix B for a description of legal OST records.

j. Message is: "SEARCH OF OST FOR 1ST ARGUMENT OF "COPY" FAILED. ".
(C) = 0 20 02222.
Corrective action: Correct COPY message(s) and rerun the update.

h. Message is: "CHECKSUM ERROR, OST OR UPD BINARY OR ENCODED RECORD. ".
(C) = 0 20 00004.
Corrective action: None. Clear the halt to ignore the error.

APPENDIX A

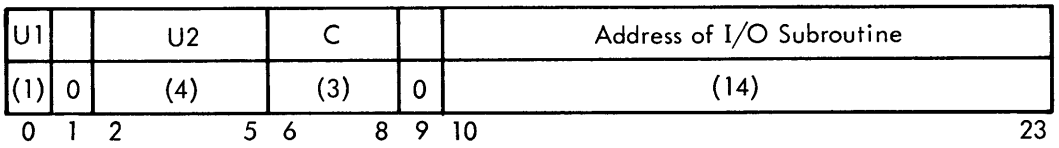
THE MONARCH UNIT ASSIGNMENT TABLE (UAT)

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).

Ten such input/output functions have been (initially) designated in the MONARCH System; additional functions may be added at a later date. The format of the Unit Assignment Table entries and a description of the ten currently provided functions are given below.

The standard SDS 900 Series I/O subroutines are constructed in such a way 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.

Format of 1-word unit assignment table entries:



C: Channel Designator

C = 0 : W buffer

C = 1 : Y buffer

U1 U2: Unit Address Code (5 low-order bits of the 6-bit unit address code)

The high speed printers are designated, in UAT entries, by 20_8 (No. 1) and 21_8 (No. 2).

Standard unit assignment entries:

	Function	External Label	MONARCH Symbolic Parameter
1.	Control message input	$QMSG	--
2.	System (MONARCH magnetic tape)	$QSYS	S
3.	System scratch	$QSYST	X1
4.	System intermediate output scratch (magnetic tape)	$QSYSI	X2
5.	System scratch (magnetic tape)	$QSYSP	X3
6.	Symbolic input (e.g., card reader)	$QSYMI	SI
7.	Symbolic output, Update input	$QSYSU	SO, UI
8.	Binary input (loader uses this)	$QBINI	BI
9.	Binary output (e.g., card punch)	$QBINO	BO
10.	List output (e.g., printer)	$QSYMO	LO

When a standard system routine 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, the MONARCH Update Routine makes use of this feature to obtain the I/O subroutines needed to perform its input/output functions.

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

MONARCH provides external label definitions for Unit Assignment Table entries which correspond to I/O routines selectively loaded by MONARCH, using 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.

APPENDIX B

THE MONARCH SYSTEM TAPE

A MONARCH system tape consists of an ordered collection of programs and data files and is preceded on the system tape by a special "bootstrap" loader which can be loaded under control of the magnetic tape "FILL" switch on the SDS 925 and 930 Computers. 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. Only the first 16 characters of a MONARCH ID Record are interpreted by the MONARCH Operating System. MONARCH ID Records have the following format:

Char. pos.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	70	71	72
Contents	Δ	n	b	b	b	b	b	b	a	c	c	c	c	c	c	c	e	e	e	e	e	e

n = 1 or 2, b = space, a = any alphabetic char., c = a or any digit or b, e = any character.

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

e.g., "Δ1$_{\wedge\wedge\wedge\wedge\wedge\wedge}$PROGNAME. . .".

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

e.g., "Δ2$_{\wedge\wedge\wedge\wedge\wedge\wedge}$SUBRNAME. . .".

Minor divisions of a MONARCH system are arbitrary subdivisions of a program or of a data file which 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. The Level 2 subdivisions of the MONARCH Library serve to separate the individual subroutines on the library for the MONARCH loader as well as making it possible to insert, delete and replace them individually.

The program names occurring in Level 1 MONARCH ID Records must be unique within a given MONARCH system tape. The program names occurring in Level 2 MONARCH ID Records must be unique 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 program names occurring in Level 1 MONARCH ID Records and "X" and "Y" are program names occurring in Level 2 MONARCH ID Records, then:

a. A and B must not be the same name.
b. X and Y can both occur as names of subdivisions of A provided that X and Y are not the same name.
c. "A" can be the name of a subdivision of A, i.e., A can occur in a Level 2 MONARCH ID Record which is in the scope of a Level 1 MONARCH ID Record with the name "A".
d. X can occur as the name of a subdivision of A and also as the name of a subdivision of B.
e. B can occur as the name of a subdivision of A.

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 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.

The system tape contains the Monitor, the MONARCH Loader, the system tape Update Routine, and such other standard system routines as shall be required by the particular installation using this operating system. The system tape unit will be unit 0 on the W buffer for the SDS 910/920 and unit 0 on channel W for the SDS 930.

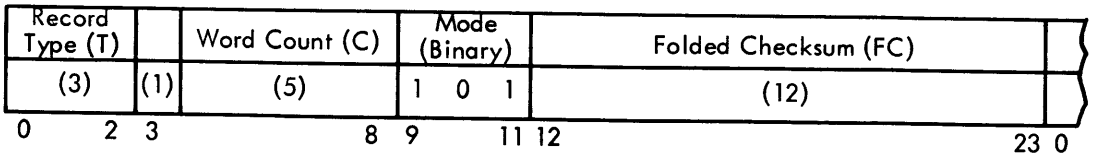
A typical system tape is described below in terms of the standard system routines it contains:

MONARCH Bootstrap Loader
MONARCH Loader
Monitor
Standard input/output subroutines
Program Library (including Programmed Operator subroutines)
SDS 920/930 META-SYMBOL Assembler System
    Translator
    META-SYMBOL Assembler
    Procedures
Card-to-tape conversion routine
Tape-to-card conversion routine
Tape-to-printer conversion routine
FORTRAN loader
FORTRAN compiler
FORTRAN library
System Tape Update Routine
End of system tape ("$\Delta 1_{\wedge\wedge\wedge\wedge\wedge\wedge}$SYSEND$_{\wedge\wedge}$")

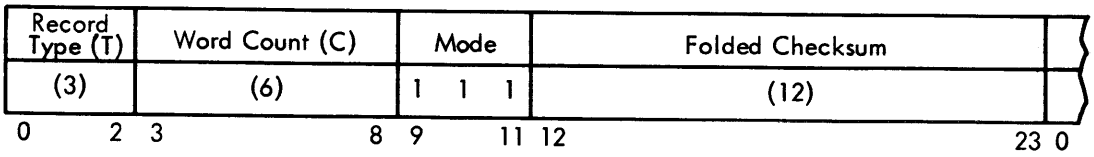
THE MONARCH SYSTEM TAPE

FORMAT OF FIRST WORDS OF VALID SYSTEM TAPE RECORDS

UNIVERSAL BINARY OBJECT PROGRAM RECORD (1st word)



META-SYMBOL ENCODED PROGRAM RECORD (1st word)



MONARCH ID RECORD (1st word)

△ Char.						Zero			Mode (IDL)			Space Char.						Space Char.						
1	0	1	1	1	1	0	0	0	0	0/1	0/1	1	1	0	0	0	0	1	1	0	0	0	0	
0					5	6					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 (1st word = WIM 012.2)

X			OP						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	

APPENDIX C

CURRENT MONARCH FUNCTIONS AND CONTROL MESSAGES

Functions	Page
FILLSYS	37
C	38
ASSIGN	39
ONLINE	40
DISPLAY	41
SHOW	42
SET	43
POSITION	44
REWIND	45
JOB	46
ENDJOB	47
LABEL	48
METASYM	49
LOAD	50
FORTRAN	51
FORTLOAD	52
UPDATE	53

FILLSYS

Purpose:	To transfer control to the Monitor Bootstrap Routine which will rewind the system tape and initiate the reloading of the MONARCH System. The message is equivalent to executing an unconditional branch to memory location 1.
Action:	The system tape is rewound and the MONARCH Bootstrap Loader is read from the system tape and executed.
Parameter:	None.
Example:	"△FILLSYS."

C

Purpose: To assign a specific input unit on a specific channel as the device from which MONARCH control messages are to be obtained.

Action: After interpreting and validating the parameter in the message, the unit and channel designations specified are stored in the UAT entry (QMSG) for MONARCH control messages.

Parameters: One, and only one, parameter must occur in the message.

The value of the parameter must be an EOM instruction (operation code of 02) with the channel designation in bit 17 and a legitimate unit code in bits 18 through 23. Specifically, the unit address code should be in the range 01_8 through 17_8.

Example: "△C PR1W."

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

ASSIGN

Purpose: To assign specific input/output units to functions represented by the MONARCH Unit Assignment Table (UAT).

Action: After interpreting and validating the parameters in the message, the unit and channel designations specified are stored in the UAT entries specified.

Parameters: At least one pair of parameters must be given, and a maximum of 9 pairs is allowed.

a. The value of the first parameter of a pair must be the memory address of the UAT entry to be set.

b. The value of the second parameter of a pair must be an EOM instruction (operation code of 02) with the channel designation in bit 17 and a legitimate unit code in bits 18 through 23.

Example: "△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

Purpose: To assign specific input/output units to functions represented by the MONARCH Unit Assignment Table (UAT).

Action: After interpreting and validating the parameters in the message, the unit and channel designations specified are stored in the UAT entries specified.

Parameters: At least one pair of parameters must be given, and a maximum of 9 pairs is allowed.

a. The value of the first parameter of a pair must be the memory address of the UAT entry to be set.

b. The value of the second parameter of a pair must be an EOM instruction (operation code of 02) with the channel designation in bit 17 and a legitimate unit code in bits 18 through 23.

Example: "△ ONLINE 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.

DISPLAY

Purpose: To provide a means of displaying the contents of one or more memory locations on the console typewriter.

Action: After interpreting and validating the parameter(s) the contents of each designated memory location is converted to octal and, together with its octal address, is typed on typewriter 1 on the W buffer.

Parameters: There must be either one or three parameters. The value of the first parameter must be a legitimate memory address. If three parameters are given, the value of the second must be "THRU" and the value of the third must be a memory address greater than or equal to the value of the first parameter.

Example:

a. One parameter:

"△DISPLAY 037777."

If we assume that location 037777 contains zero, then this message will cause the following to be typed:

"[037777] = 000000000"

b. Three parameters:

"△DISPLAY 0164 THRU 0174."

This message will cause the address and contents of each of the 11 locations specified to be typed.

SHOW

Purpose: To provide a means of displaying the contents of one or more memory locations on the console typewriter.

Action: After interpreting and validating the parameter(s) the contents of each designated memory location is converted to octal and, together with its octal address, is typed on typewriter 1 on the W buffer.

Parameters: There must be either one or three parameters. The value of the first parameter must be a legitimate memory address. If three parameters are given, the value of the second must be "THRU" and the value of the third must be a memory address greater than or equal to the value of the first parameter.

Example:

a. One parameter:

"△SHOW 037777."

If we assume that location 037777 contains zero, then this message will cause the following to be typed:

"[037777] = 000000000"

b. Three parameters:

"△SHOW 0164 THRU 0174."

This message will cause the address and contents of each of the 11 locations specified to be typed.

SET

Purpose: To set the contents of the specified memory location to the value indicated.

Action: After interpreting and validating the parameters the memory location specified by the first parameter is set equal to the value of the second parameter. MONARCH then attempts to read the next control message.

Parameters: Two, and only two, parameters must be supplied.

a. The value of the first parameter must be a legitimate memory address.

b. The value of the second parameter is stored in the location specified by the first.

Example: The message

"△SET 017 = -59."

will cause the contents of cell 017_8 to be set to -73_8.

POSITION

Purpose: To position a magnetic tape at a given file by reading successive records (in a forward direction) until a level 1 MONARCH ID record is found which contains, in characters 9-16, the File ID specified.

Action: Records are read, under control of the MONARCH Search Subroutine, from the tape unit specified by the first parameter until a level 1 MONARCH ID Record is read then:

a. If characters 9 through 16 contain the File ID specified by the second parameter, control is returned to MONARCH to obtain the next control message. The tape will be positioned in the inter-record gap which follows the ID record.

b. If characters 9 through 16 contain "SYSEND$\wedge\wedge$" then a message is typed indicating that the specified MONARCH ID Record was not found, and control is returned to MONARCH to obtain the next control message.

c. If characters 9 through 16 contain neither "SYSEND$\wedge\wedge$" nor the specified FILE ID, then the search is continued until either condition a or condition b is satisfied.

The tape is read in binary ("odd" parity) mode and the maximum record length is assumed to be 40 words (160 characters).

Parameters: Two parameters must be given.

a. The value of the first parameter must be an EOM instruction (operation code of 02_8) with the channel designation in bit 17 and a legitimate tape unit address in bits 18 through 23.

b. The value of the second parameter is used as the search key; it can be from 1 to 8 characters in length. Trailing spaces (60_8) are supplied if fewer than 8 characters are given.

Example: The message

"$\triangle$POSITION MT2, "FILEIDEN". " will cause MONARCH to position tape unit 2 on the W buffer in front of the first record following the level 1 MONARCH ID Record which contains "FILEIDEN" in characters 9 through 16.

REWIND

Purpose: To cause MONARCH to rewind the specified magnetic tape unit.

Action: The tape unit specified by the parameter is rewound.

Parameters: One, and only one, parameter must be supplied. Its value must be an EOM instruction (operation code of 02₈) with the channel designation in bit 17 and a legitimate tape unit address in bits 18 through 23.

Example: The message

"△REWIND MT3."

will cause MONARCH to rewind magnetic tape unit 3 on the W buffer.

JOB

Purpose: To set the JOB mode switch and reset the processor error switch.

When MONARCH encounters a "LOAD" control message while the JOB mode switch is set, the processor error switch is tested and:

a. If the processor error switch is set, a message is typed indicating that a processor error has occurred. MONARCH will ignore the "LOAD" message and attempt to read the next control message from the current control message medium.

b. If the processor error switch is reset, the "LOAD" message is processed regardless of the status of the processor error switch.

Action: The memory cell containing the JOB and the processor error switches is set to 40000000_8. Bit zero represents the JOB switch and bits 1 through 23 represents the processor error switch.

Parameters: None.

Example: "△JOB."

ENDJOB

Purpose:	To reset the JOB switch and the processor error switch. See JOB message description.
Action:	The memory cell containing the JOB and processor error switches is set to zero.
Parameters:	None.
Example:	"△ENDJOB."

LABEL

Purpose: To write a Level 1 MONARCH ID record on a magnetic tape.

See Appendix B for a definition of MONARCH ID Records.

Action: A Level 1 MONARCH ID Record is constructed with the name field (characters 9 through 16) containing the identifier specified as the second parameter. This ID record is then written on the magnetic tape specified by the first parameter. A 40-character record is written in binary ("odd" parity) mode.

Parameters: At least one parameter must be given.

a. The value of the first parameter must be an EOM instruction (operation code 02) with the channel designation in bit 17 and a legitimate magnetic tape unit address in bits 18 through 23.

b. The second parameter is optional. If not given, the ID record written will contain spaces (60_8) in the name field (characters 9 through 16). When present, this parameter, which can be given as a double precision literal, will be used to construct an 8-character name. If less than 8 characters are given, the name field will contain (trailing) spaces in the right-most character positions.

Example: "△LABEL MT3W, "FILENAME"."

This message will cause MONARCH to write a Level 1 MONARCH ID Record on magnetic tape unit 3 on the W buffer. Characters 9 through 16 of this record will contain "FILENAME".

METASYM

Purpose: To cause MONARCH to load the META-SYMBOL Assembly System and transfer control to it.

Action: The parameters are interpreted and validated and then MONARCH searches the system tape for the first program in the META-SYMBOL system routine and loads it and transfers control to it. MONARCH and META-SYMBOL assume that the necessary input/output units have been assigned and that all tape units, except scratch tapes, are correctly positioned.

Parameters: At least one parameter from each of the first two groups (a and b below) must be given. The parameters may occur in any order in the control message.

a. Parameters specifying input to META-SYMBOL:

Symbolic parameter	META-SYMBOL input
SI	symbolic input
EI	encoded input

b. Parameters specifying output data to be provided by META-SYMBOL:

Symbolic parameter	META-SYMBOL output
SO	symbolic (source) output
EO	encoded output
BO	binary output
LO	listable output

c. If compatibility mode translation of symbolic input is desired, the symbolic parameter C should be included in the control message.

Example: "△METASYM SI, LO."

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

LOAD

Purpose: To load one or more binary object programs using the MONARCH Loader.

See Section 3 for a description of the MONARCH Loader and a discussion of its capabilities.

Action: The parameters are interpreted and validated and then if a Program ID parameter was given, MONARCH will search for the Level 1 MONARCH ID Record containing that ID by reading records from the binary input device (BI) currently assigned. Whether or not a Program ID parameter was given, MONARCH will then transfer control to the MONARCH Loader.

Parameters: At least one and at most 3 parameters must be given.

a. The value of the first parameter is interpreted by the loader as the initial load relocation bias.

b. The value of the second parameter is interpreted by the loader as the Loader Options parameter.

c. The third parameter is optional. If present, its value is interpreted as a Program ID and this Program ID is assumed to occur in characters 9 through 16 of a Level 1 MONARCH ID Record on the current binary input unit.

Example: "△LOAD 010000, STOP."

This message causes the MONARCH Loader to load one or more programs from the current binary input unit (BI) and stops (halts) after each program is loaded.

FORTRAN

Purpose: To cause MONARCH to load the FORTRAN compiler.

Action: The parameters are interpreted and MONARCH then searches the system tape for the FORTRAN compiler, loads it, and transfers control to FORTRAN.

Parameters: Zero to three parameters may be given. The parameters may occur in any order.

a. SI, Symbolic Input. If this parameter is missing, it is assumed to be present.

b. BO, Binary Output. This will cause an Object program to be generated. If no I/O unit has been assigned, this parameter is ignored.

c. LO, List Output. This will cause an output listing to be produced. If no I/O unit has been assigned, this parameter is ignored.

Example: "△FORTRAN LO."

This message requests the FORTRAN compiler to compile and list the source program. An object program is not generated.

"△ FORTRAN."

This message requests the FORTRAN compiler to compile a source program; no listing or object programs are produced, only the program allocation and error, if any, will be listed.

FORTLOAD

Purpose: To cause MONARCH to load the FORTRAN Loader.

Action: The parameters are interpreted and MONARCH then searches the system tape for the FORTRAN Loader, loads it, and transfers control to it.

Parameters: One to eight parameters may be given. The first specify the mode in which the loader will operate. They are:

- MAP - Produces a map of the program on the console typewriter
- LMAP - Produces a Label map on the console typewriter.
- LTRA - Will cause a label trace at execution time.

These three parameters may appear in any order or may be omitted entirely. They must be first in the parameter string. The other parameters specify which input devices are to be read. They are read and loaded in the order that they appear in the parameter list. These devices must have been assigned and correctly positioned. To load the program unconditionally, a letter V will follow the parameter.

Example: △FORTLOAD MAP, LMAP, X1U, CRU, X2.

This message will cause the FORTRAN Loader to load, unconditionally, I/O device X1 until an end-of-file condition is encountered. The card reader is then read, unconditionally, until an △EOF card is read. Then I/O device X2 is read, but only loading the programs that are needed. Finally, the system tape is read, loading the needed library and runtime. A MAP and a Label MAP are produced on the console typewriter.

UPDATE

Purpose:	To cause MONARCH to load the system Update Routine and transfer control to it. See Section 4 for a description of the Update Routine and the control messages required for its use.
Action:	MONARCH searches the system tape for the Update Routine, loads it, and transfers control to it.
Parameters:	None.
Example:	"△UPDATE."

APPENDIX D

UPDATING META-SYMBOL ON MONARCH TAPES

To change the ENCODER, S4B, MON1, MSCONTRL, ABSOLUTE LOADER or the PROC deck, use the standard MONARCH ASSIGN, UPDATE, and COPY control cards. Insert in the update deck the binary (encoded for PROCS) decks to be changed and do a normal update.

The order of programs on the MONARCH system tape is:

△1 METASYM

- △2 ENCODER (includes ENCODER followed by S4B).
- △2 MON1 (includes MON1 followed by the absolute loader).
- △2 MSCONTRL.
- △2 PREASSEM (includes PREASM absolute followed by encoded system PROCS).
- △2 ASSEMBLER. META-SYMBOL absolute.

When updating a section of META-SYMBOL all portions of the labeled segment must be updated. For example, to insert a new PROC deck, one must also insert the PREASM absolute deck preceding it.

To update PREASSEM and ASSEMBLER, use the normal update procedure. If the changes are via binary patches, the patches are inserted at the end of the absolute binary deck just preceding the END card and the update performed.

If the ASSEMBLER or the PREASM are modified through reassembly, it is necessary to convert these programs to absolute before placing them on the updated system tape. After conversion to absolute and inserting the transfer card, update these portions of the system like any other portion.

To generate an absolute version of ASSEMBLER or PREASSEM, load the relocatable binary program using the MONARC LOAD procedure. It may be good practice to stop after each segment and record the relocation bias so that they are available for binary patching and debugging.

After loading the relocatable program, load the dump routine (deck to be supplied); leave all keys in the reset position while loading the dump. This deck has a 1-card loader preceding it; therefore, to load, do a bootstrap from cards.

When the dump program is loaded the program halts with a NOP to 012121 showing in the C register. At this point, enter into the A register the first location to punch and into B the last location of the program (the relocation bias of the last segment plus its length).

Place RUN-IDLE-STEP in RUN and the program is punched in absolute binary format onto cards. This simple dump routine punches full cards so that you may actually punch a few words following the end of the program. Also, the dump does not punch the transfer card (END card). If the starting location of the program changes the END card, it will have to be repunched manually using standard UBL format. Normally, the END card from the previous absolute deck can be used.

A few comments about increasing program sizes is in order here. 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.

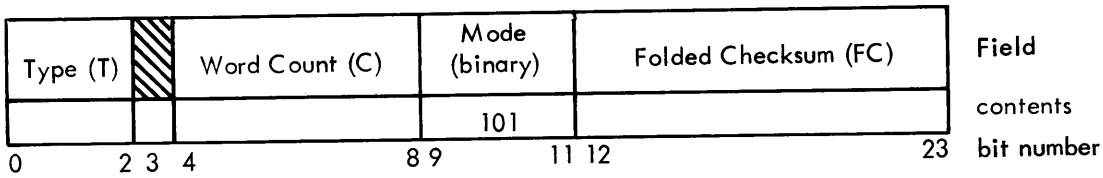
APPENDIX E

SDS STANDARD BINARY LANGUAGE

The following description specifies a standard binary language for the SDS 900 Series and 9300 Computers. The intention has been that this language be both computer-independent and medium-independent. Thus, there is provision 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 SDS 900 Series FORTRAN II.

In the following, 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 - 1st word in each type of record



T	RECORD TYPE
000	Data Record (text)
001	External References & Definitions, Block & Program Lengths
010	Programmed Operator References and Definitions
011	End Record (Program or Subroutine end)
100 thru 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

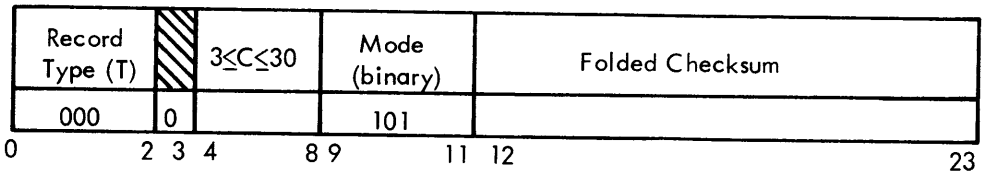
$$FC = (W_1)_{0-11} \text{ -- } (C)_{0-11} \text{ -- } (C)_{12-23} \text{ -- } 07777$$

where

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

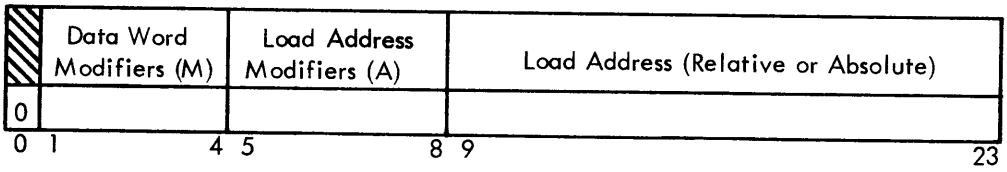
2. DATA RECORD FORMAT (T=0)

Control Word



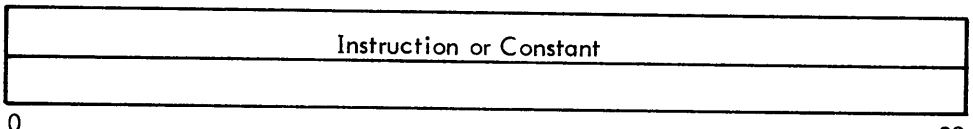
Word 1

Load Address Word



Word 2

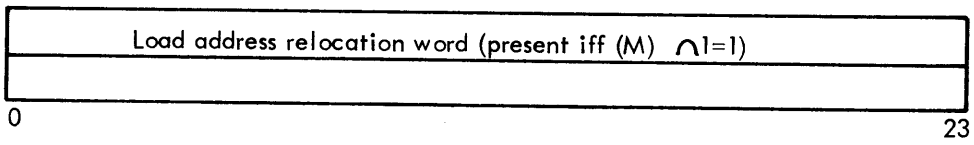
Data Word 1



Word 3

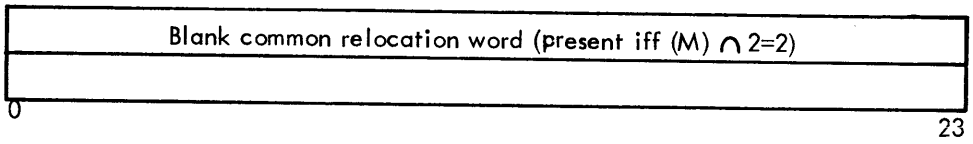
Words 3 thru n+2 contain instructions or constants, (where $1 \leq n \leq 24$)

Load Relocation



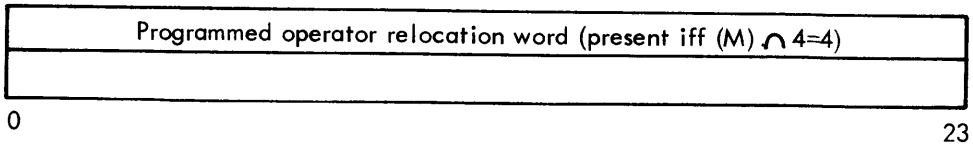
Word n+3

Common Relocation



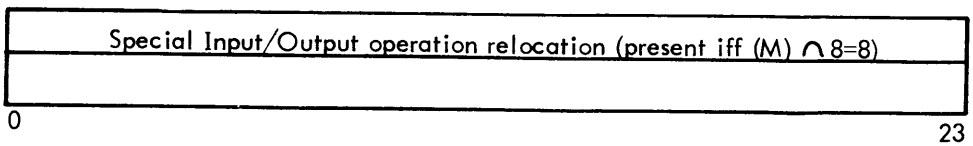
Word n+4

Programmed Operator Relocation



Word n+5

Special I/O Relocation



Word n+6

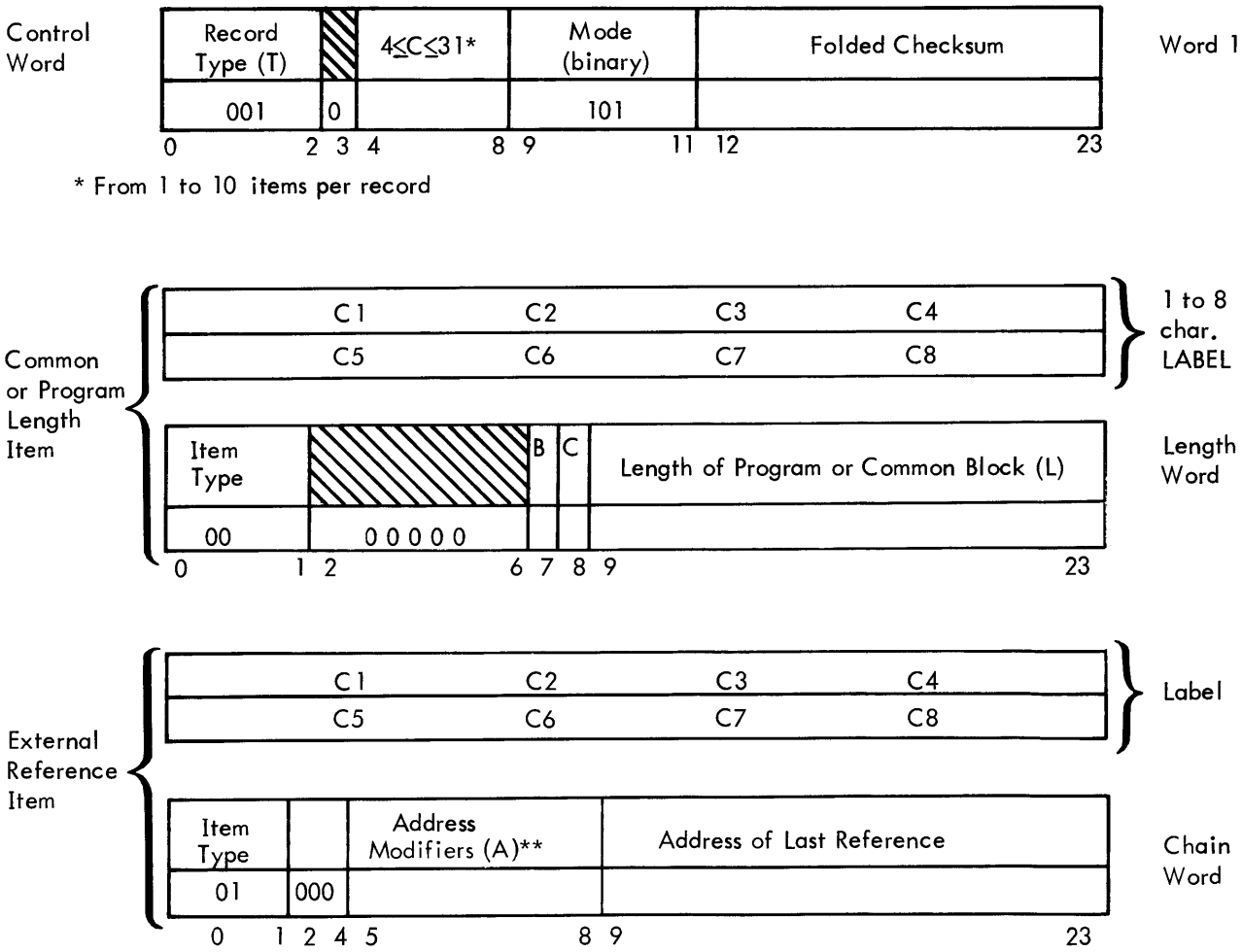
Words n+3 thru n+6 are modifier words. Each bit in each of these words corresponds to a data word (bits 0 thru 23 correspond to words 3 thru n+2, respectively). A bit set to one (1) indicates that the specified data word required modification by the loader. There are four (4) types of modification (and hence four possible modifier words) which are indicated in data records. Presence of a modifier word 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 means absolute):

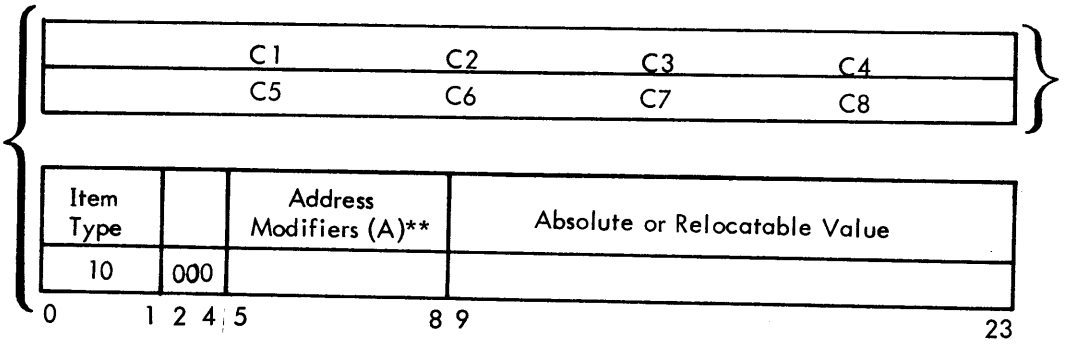
(A)∩1=1, current load relocation bias is added to load address
(A)∩2=2, current common relocation bias is added to load address
the remaining bits of A are unassigned. (A)=3 is illegal.

3. EXTERNAL REFERENCES AND DEFINITIONS, BLOCK AND PROGRAM LENGTHS

(T=1) (Includes labeled common, blank common and program lengths)



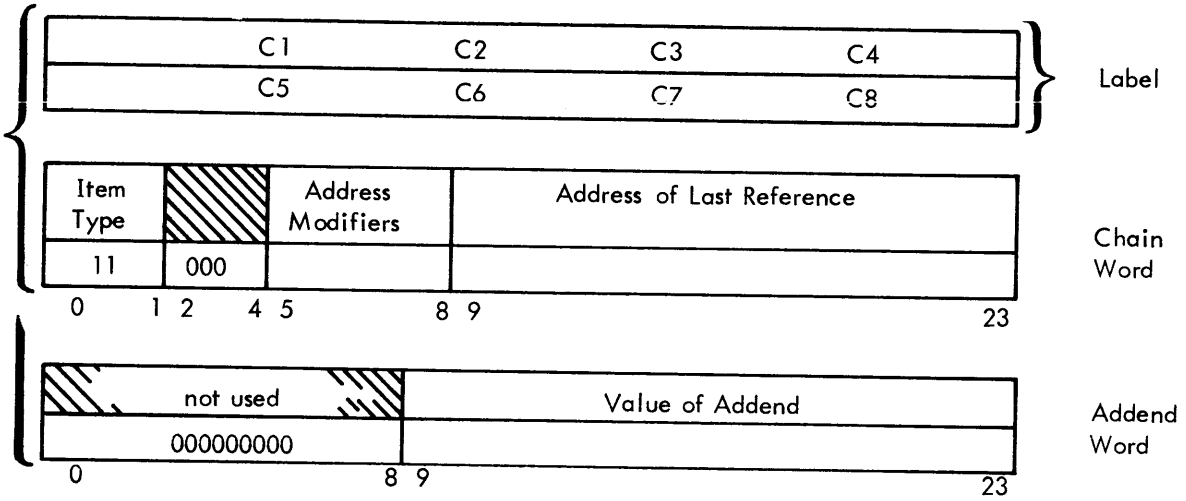
External Definition Item



External symbolic definitions include subroutine "identification" as a subset and require no special treatment of subroutines with multiple names. B = 1 if (L) is program length, C = 1 if (L) is length of a labeled common block.

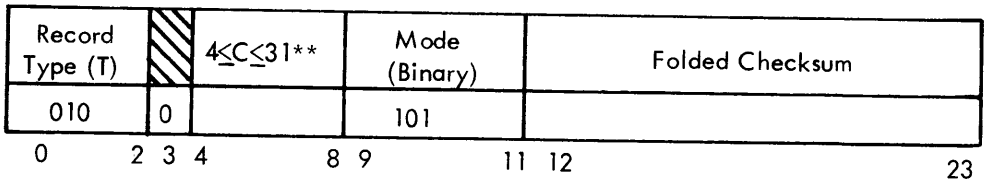
** See data record, load address word, for interpretation

Labelled Common Reference Item*



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

Control Word

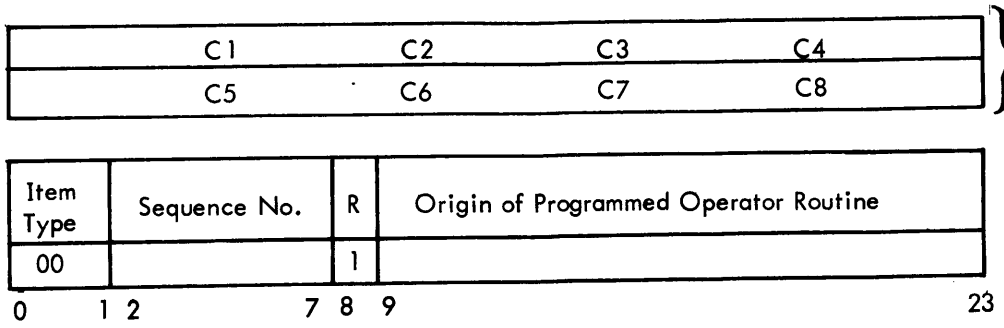


*One of these items for each unique reference; e.g., each of the following references is represented by a separate item:

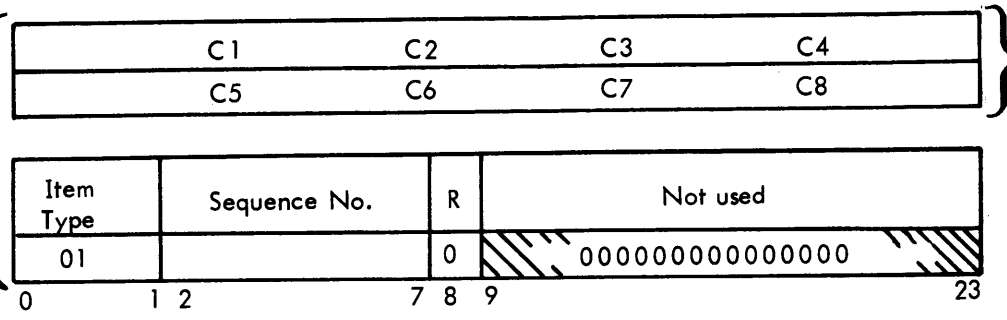
A+5, B+5, B+6, C+2, C+5

** From 1 to 10 items per record

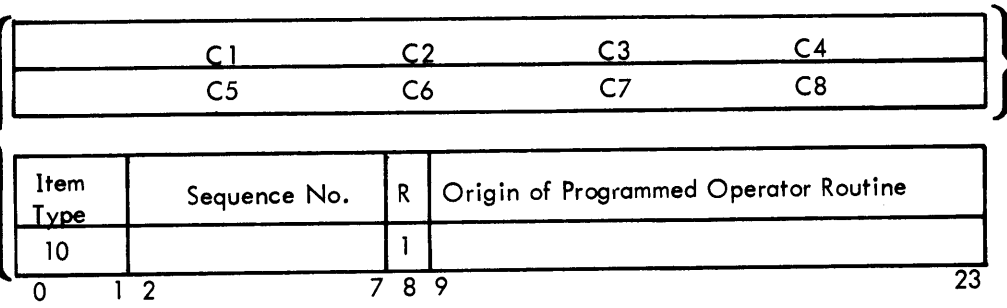
Internal Programmed Operator



Programmed Operator Reference



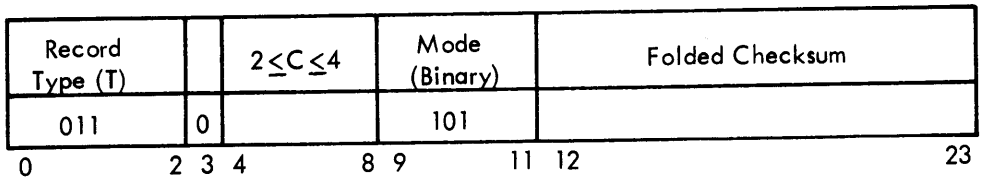
External Programmed Operator Definition



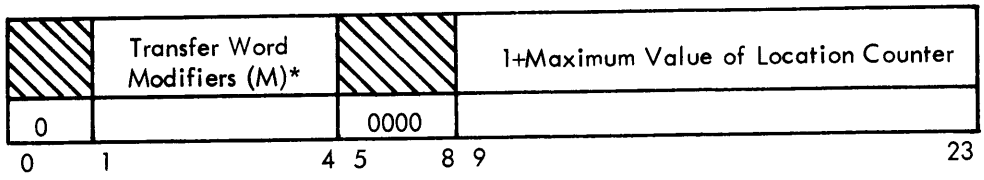
R=1 iff origin of Programmed Operator Routine is relocatable.
The sequence No. indicates the order in which the definitions or reference occurred in the source program.

5. END RECORD (T=3)

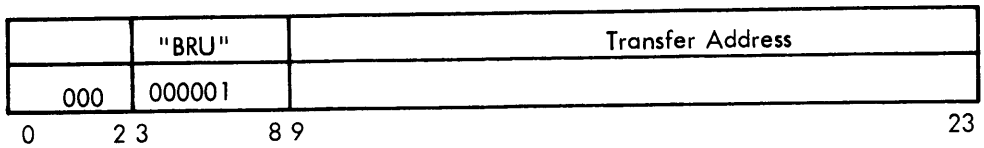
Control Word



Length of Program



Transfer Word



This may be followed by modifier words as described in Section 2.

*See data record description for interpretation

SDS SCIENTIFIC DATA SYSTEMS / 1649 Seventeenth Street / Santa Monica, California / Phone: (213) UP 0-5471

SALES OFFICES

EASTERN
125-10 Queens Blvd.
Kew Gardens, N.Y.
(212) LIggett 4-9898

69 Hickory Drive
Waltham, Mass.
(617) 899-4700

1145 Nineteenth St., N.W.
Washington, D.C.
(202) 337-6838

SOUTHERN
Holiday Office Center
Huntsville, Ala.
(205) 881-5746

16 South Bumby St.
Orlando, Florida
(305) 425-4611

3334 Richmond Ave.
Houston, Tex.
(713) JAckson 6-2693

MIDWEST
3150 Des Plaines Ave.
Des Plaines, Ill.
(312) 824-8147

One Parkway Center
875 Greentree Rd.
Pittsburgh, Pa.
(412) 921-3640

WESTERN
1649 Seventeenth St.
Santa Monica, Calif.
(213) UPton 0-5471

Fountain Professional Bldg.
9000 Menaul Blvd., N.E.
Albuquerque, N. Mex.
(505) 298-8009

Sunnyvale Office Center
505 West Olive Ave.
Sunnyvale, Calif.
(408) 736-9193

Carlson Bldg.
808 106th St., N.E.
Bellevue, Washington
(206) GL 4-3991

FOREIGN REPRESENTATIVES

EUROPE
CECIS
14 Rue de la Baume
Paris 8, France

CANADA
INSTRONICS, Ltd.
P.O. Box 100
Stittsville
Ontario, Canada

JAPAN
F. Kanematsu & Co. Inc.
Central P.O. Box 141
New Kaijo Bldg.
Marunouchi
Tokyo, Japan

AUSTRALIA
RACAL Pty. Ltd.
5 Ridge St.
N. Sydney NSW, Australia

900566A