SPIREL

SPIREL

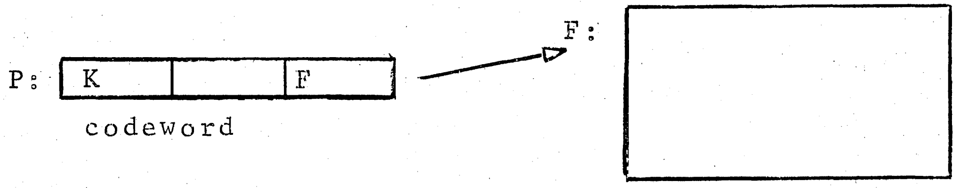
CONCEPTS

A *computer* serves a user by decoding *instructions* and performing the operations specified. In an analagous manner, the *SPIREL system* serves a user by decoding *control words* and performing the operations specified. In fact, once in the machine SPIREL may be thought of as an extension of the Rice Computer.

As instructions are used to dictate computer operations on single *words* in the memory, so are control words used to dictate SPIREL operations on *blocks*. Physically, a block is a set of contiguous words in memory. Associated with each block is a one-word "label" called a codeword. A block is a logical unit from the point of view of the user; it may contain a program, a vector of data, or codewords which in turn label other blocks. In general, an *array* is a logical structure which consists of a codeword which labels a block of codewords which label blocks, and so on until on the lowest level are blocks which do not contain labels. The *depth* or *dimension* of an array is just the number of codeword levels in the array. A *program* is a single block with one codeword, a one-dimensional array. A data *vector* is a single block with one codeword, a one-dimensional array. A *matrix* of data is a vector of data vectors, a two-dimensional array. Collections of programs and data vectors may be logically grouped to form program and data arrays of any depth deemed organizationally useful to the SPIREL user.

An array is uniquely associated with its single highest codeword, the *primary codeword*. In most applications all addressing of information in arrays (contents in lowest level blocks) is done *through* the primary codeword. Thus, access to information in an array depends on only one address, the *codeword address* for the array. The physical location of blocks is irrelevant to the user, so allocation of storage for blocks is performed by SPIREL, and addressing through levels of codewords constructed by SPIREL is accomplished by the indirect addressing of the hardware. A fixed region of the memory, locations 200 through 277 (octal), is by convention reserved for primary codewords, and allocation in this area is the responsibility of the individual user. The unique correspondence of a primary codeword address to its array provides an informative "name" for the array. A program with codeword address 225 may be called program *225, and a matrix with primary codeword at 271 may be called matrix *271; the '*' symbolizes indirect addressing in the assembly language, and here serves to emphasize this operation in connection with codewords.

A program *P of length K may occupy a block beginning at machine address F. Then program *P is represented in the machine as



In programming, control is passed to this program by the code

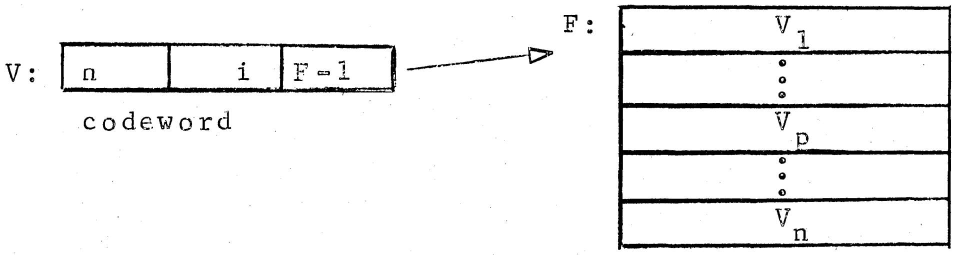
TSR *P

which becomes

TSR F

when the hardware indirect addressing is carried out. The address formed is that of the first word of the program.

A vector *V of n data elements V_1, V_2, ..., V_n may occupy a block beginning at machine address F. Then vector *V is represented in the machine as



In programming, the data element V_p is addressed by the code

p → index register i

then

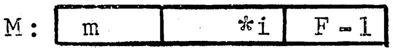
operation *V

which becomes

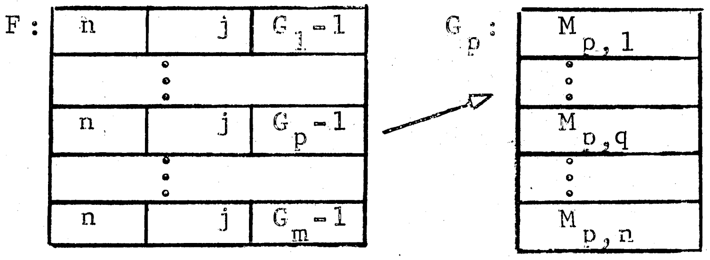
operation $p+F-1$

when the hardware indirect addressing is carried out. The address formed is that of V_p, as desired. The standard vector form uses index register B1 for element addressing. Non-standard forms permit variability of indexing and even allow the first word of the block containing the vector elements to be addressed as V_K where K is any integer.

A matrix *M of m rows by n columns of data $M_{1,1},\ldots,M_{m,n}$ is stored one row per block and is represented in the machine as



primary codeword



secondary codewords

In programming the data element $M_{p,q}$ is addressed by the code

p → index register i

q → index register j

then

operation *M

which becomes

operation $*p+F-1$

and then

operation $q+G_p-1$

when two levels of hardware indirect addressing are carried out. The address formed is that of $M_{p,q}$, as desired. This addressing in no way depends on the size of the matrix *M. The standard matrix form is for a rectangular matrix, using B1 for row specification and B2 for column specification. Non-standard forms permit variability of indexing and non-rectangular structures and even allow the first element to be addressed as $M_{K,L}$ for K and L any integers.

SPIREL operations on arrays are dictated by control words which specify the primary codeword address for the array. Such operations are:

- to take space for a program, vector, or standard matrix
- to print the lowest level blocks in an array
- to punch an array
- to execute a program
- to free the space occupied by an array
- and many others

Of particular importance is the fact that arrays may be dynamically created and erased or changed in size and structure so that only immediately pertinent arrays occupy space at any time during the run of a user's system.

SPIREL control words are 18-place octal configurations, i.e., one machine word in length. With the SPIREL system in the machine, control words may be transmitted to the system in two ways:

- <u>internally</u> under program control by the user --

  control word → T7

  and TSR *126

- <u>externally</u> to the SPIREL communication routine from paper tape --

  control word preceded by 'carriage return' punch on paper tape in the reader

  or from the typewriter --

  control word $\xrightarrow{\text{type}}$ U-register.

In all these cases the control word is in fact transmitted to XCWD (Execute Control Word), program *126 in the SPIREL system. This program is the nucleus of SPIREL; it interprets each control word and may use other programs in the system to carry out specified operations. SPIREL is, then, a collection of programs, and any of these may be utilized directly by the user of the system.

Details about codewords, system organization, control word decoding and formats, storage control, and the SPIREL programs are given in the succeeding sections.

CODEWORDS

With every block of memory is uniquely associated a codeword which has two primary functions:

- description of the block, including current length and location, current type of block content, and printing format for the block
- indirect addressing portion appropriate for programmed addressing through the codeword into the block.

The format for a codeword at address C, which labels a block beginning at address F=f+i or F=f+i-1 is:

	1 15	16 27	28	29 30	31	32 39	40 54
C:	n	i	a	y	*	m	f

indirect addressing portion

where

n=<u>length</u> of the block.

i=<u>initial index</u> of the block in 1's complement format, where zero i is denoted by i=7777; standard SPIREL provision is for i=1.

For a block with B-mods in its codeword (a vector), the first word is addressed as element C_i.

For a block with no B-mods in its codeword (a program), the first word of the block is addressed as word 1 of C; if i<1, the words preceding word 1 are cross reference words.

a=1 if block contains codewords; empty (0) otherwise.

y=printing format to be used for output of block if none given in print control word

0: octal, 4 words per line (standard SPIREL provision)

1: hexad, 108 characters per line

2: octal, 1 word per line in program layout

3: decimal, 5 words per line

*,m=indirect addressing and B-modification bits, effective in addressing indirectly through the codeword.

f=F-i if block has
B-mods in codeword
F-i+1 if block has no
B-mods in codeword

F=the address of the first word of the block

4 → Physically, but index [illegible] when <1 !

A codeword is completely formed by SPIREL at the time the corresponding block is created. This creation is the result of a control word to take space (structure formed and lowest level blocks filled with zeroes) or one to read a block (structure formed and lowest level blocks filled with words read from paper tape). The most frequently used codeword forms are illustrated below.

- For a program *P of length k, no cross references:

	1 15	16 27	28 39	40 54
P:	k	0001	0000	F

octal digits

where the first word in the block and the first word of code in the program coincide at address F.

For a program *P with two cross references prior to code:

	1 15	16 27	28 39	40 54
P:	k	7776	0000	F+2

octal digits

where the first word of the block is located at address F, and the first word of code in the program is located at address F+2.

In both cases the block occupied by the program is k words in length, and control is passed to the program by the code

```
TSR       *P
```

- For a standard vector *V of length n:

	1 15	16 27	28 39	40 54
V:	n	0001	0002	F-1

octal digits

where the vector element V_1 is located at address F, the initial index=1, and a B1 modifier is used. The element V_p is addressed by the code

```
SB1       p

CLA       *V
```

- For a standard matrix *M of m rows by n columns:

primary codeword

	1 15	16 27	28 39	40 54
M:	m	0001	4402	S-1

octal digits

secondary codewords

S:	n	0001	0004	R_1-1
		:		
S+p-1:	n	0001	0004	R_p-1
		:		
S+m-1:	n	0001	0004	R_m-1

octal digits

where each row is stored in a separate block and the matrix element $M_{p,1}$ is located at address R_p for each p, the initial row and column indices=1, a B1 modifier is used for row specification, a B2 modifier is used for column specification, the primary codeword contains an a-bit and a *-bit. The element $M_{p,q}$ is addressed by the code

```
SB1     p
SB2     q
CLA     *M
```

An array whose primary codeword address is utilized in code is a numbered array. There are also named arrays, whose names are stored in a system vector called the Symbol Table with primary codewords stored in the parallel Value Table. A named array is addressed in code indirectly through a cross reference word which contains the name of the array. All cross reference words for a program are located within the program before the code. The code is then one level removed from the primary codeword for a named array, and the linkage from cross reference word to the codeword in the Value Table is maintained by SPIREL.

The user specifies the primary codeword address or the name for each array, but the location of blocks in the memory is left to the "discretion" of SPIREL. Any given block may be located variously from run to run, and may be moved even during a run if the STEX storage control mechanism in SPIREL is active and such manipulation is necessary for another requested allocation.

- Memory Utilization

The SPIREL system itself is a collection of programs, tables, and individual constants. System conventions provide memory utilization as follows:

octal addresses	use
00000-00007	machine full-length fast registers Z,U,R,S, T4,T5.T6.T7
00010	used as codeword address by library routines
00011-00020	machine trap locations
00021-00022	not used
00023-00024	link to SPIREL console communication routine
00025-00026	~~not used~~ used by disc operations
00027-00036	entry to SPIREL program *120, diagnostic dump
00037	console entry to magnetic tape system (explained in MAGNETIC TAPE SYSTEM section)
00040-00077	used by SPIREL program *120, diagnostic dump, and by magnetic tape system programs
00100-00177	system codewords and individual constants
00200-00277	region for primary codewords of numbered arrays or numbered constants of the system user
00300-[E-400]*	storage of blocks labelled by system or user codewords, each block containing a program, data, or codewords which in turn label other blocks
[E-377]-[E-100]*	main magnetic tape system program
[E-77]-E*	magnetic tape system communication program

*E represents the end of memory:

E=17777 for 8K
37777 for 16K
57777 for 24K
77777 for 32K

- B6-list

The *working push-down storage* area is addressed by index register B6 and is commonly called the B6-list. SPIREL programs use the B6-list; programs of the system user may similarly use the B6-list; and index register B6 may be used for other purposes only if the working storage setting is maintained for those programs which depend on it.

Conventional use of the B6-list depends on one fact: that B6 contains the address of the first word of a block of storage not in use. Therefore, if one word of temporary storage is required, it is taken at B6 and B6 is incremented by one; if the last word stored on the B6-list is retrieved from the address B6-1 and is in fact no longer resident on the list, B6 is decremented by 1, and the storage location may be reused.

The B6-list exists in memory as program block *112. The initial system setting of B6 is to the first word address of this block (given in codeword). The standard length of the block is 200 (octal) locations.

A frequent application of the B6-list is for temporary storage of fast registers to be used by a subroutine, but to appear undisturbed to the program using the subroutine. A program wishing use but preserve T4, T5, and T6 might use the B6-list as follows:

- upon entry

```
T4      STO     B6,B6+1
T5      STO     B6,B6+1
T6      STO     B6,B6+1
```

- computation with private use of T4, T5, T6 and any desired use of the B6-list
- prior to exit

```
        LT6     B6-1,B6-1
        LT5     B6-1,B6-1
        LT4     B6-1,B6-1
```

- exit with B6 setting same as that upon entry.

- System Components

The programs, tables, and individual constants which comprise the SPIREL system are listed below. The programs are fully explained in later sections, and the diagram of SPIREL component linkage shows how the various components are functionally interconnected.

INDIVIDUAL CONSTANTS

Address	Name	Function
100	STORAG	describes available storage
101	FIRSTEX	first word address of storage exchange domain
102	LASTEX	last word address +1 of storage domain
107	NUMBER	relative Symbol Table address of SPIREL operand
114	PRCT	current active length of ADDR
115	ACWD	used by PUNCH
117	STPNT	gives index of last active entry in ST and VT
121	FWA	first word address of last program tagged (used by TRACE and TAGSET)
124	NAME	symbolic name (if any) of block currently being operated on by SPIREL

VECTORS, PRINT MATRIX, B6-LIST

Codeword Address	Name	Use	Length_8
112	LISTB6	Push-Down Storage Area	200
113	ST	Symbol Table	400
116	PM	Print Matrix	200
122	VT	Value Table	400
125	ADDR	Base Address Vector	6
174	TEXT	Console Input Text	14

PROGRAMS

Codeword Address	Name	Use
13	TRACE	Trace
14	ARITH	Arithmetic Error Monitor
20	CHECK	Check Block Bounds
110	HDPR	Print Control Word
111	MATRX	Process Matrix
120	DIADMP	Diagnostic Dump
126	XCWD	Execute from Control Word
127	SETPM	Set Up Print Matrix
130	SMNAM	Find Symbolic Name
131	DATIME	Print Date and Time
132	CLOCK	Decode Clock
133	PCNTRL	Punch Control Word
135	STEX	Storage Exchange
136	SAVE	Save Fast Registers
137	UNSAVE	Unsave Fast Registers
140	DELETE	Insert or Delete Space
141	CHINDX	Change Initial Index
142	TAGSET	Tagset
143	CONVRT	Convert from Decimal
144	PRINT	Print
145	PUNCH	Punch
146	XCWSQ	Execute Control Word Sequence
147	PFTR	PF Trace
150	MAP	Map STEX Domain
151	PRSYM	Print Symbol and Value Tables
152	PWRTN	Conversion of Powers of Ten
153	MRDDC	Multiple Read Decimal
155	BINDC	Binary to Decimal Conversion
156	RDCHK	Read with Checksum
157	PUNCHK	Punch with Checksum
170	PLOT	Plot Character on Scope
171	SAMPLE	Sample Typewriter for Console Input
172	ERPR	Print Error Messages
173	CONSOL	Interpret Console Input
175	CNTXT	Determine Context
176	TLU	Table Look-Up

Program *126, XCWD

The nucleus of the SPIREL system is the program *126, XCWD (Execute from Control Word). This routine interprets control words received in T7 and carries out the specified operations. The work of *126 may in most cases be described in the following steps:

1) address determination -- consists of determining the address of the first word to be operated upon and the number of words to be operated upon
2) operation determination -- consists of determining the operation to be performed
3) operation execution -- consists of performing the operation or using another SPIREL program to carry out the operation

XCWD accepts a control word in T7 and disturbs no other fast registers. Two sense lights affect the behavior of *126:

SL14 off causes XCWD to print one line for each control word it executes. The information printed includes the control word, the operation, the name and relative symbol table address of the block operated on, the location and number of words operated on, the number of free words of storage remaining and a notation if the storage exchange system is active. This SPIREL monitoring provides useful load records and may help in debugging. A system which has been checked out would probably run with SL14 on to suppress monitoring.

SL15 on causes "reading" and "correcting" operations of SPIREL to bypass storage of what is read and instead compare it with what is currently in the locations where storage would otherwise take place.

o Control Word Format

A control word is divided into seven fields: N,w,x,y,z, R, and F. These are arranged as follows:

bits	1 15	16 27	28 39	40 54
	nnnnn	wxyz	rrrr	fffff
	N		R	F

where each lower case letter represents one octal digit. In general, N,x,R, and F concern address specification; w,y, and z concern operation specification. x indicates the shape of the operand F

x = 0 → absolute address

x = 1 → relative (single level) address

x = 2 → relative with B1 modifier (single level)

x = 4 → relative (all levels) address

x = 5 → relative on symbol table

x = 3, 6, and 7 are given under More SPIREL Operations, p.29,19,20.

- Address Specification

General rules for address specification may be stated; exceptions exist and are noted in the list of control words later in this section.

The control word fields N,x,R, and F provide information which determines

$\mathcal{F}$, the address of the first word to be operated on

and η, the number of words to be operated on.

To specify SPIREL operation on a set of locations whose absolute machine locations are known:

x=0
F= $\mathcal{F}$
N= η
R irrelevant

For example, it might be useful to have SPIREL print the user's codeword region, 100 (octal) words starting at location 200 (octal).

To specify SPIREL operation on all of or a portion of a block labelled by a numbered codeword:

x=1 relative
F=codeword address

If $\mathcal{F}$ is to be the first word of the block or blocks operated on:

R=0

R≠0 specifies the relative program word or vector element, counting from the initial index of the block in either case, where initial index=1 for programs containing no cross references prior to code and for standard vectors. Note that the 0^{th} word of a program or 0^{th} element of a vector is specified by R=7777.

If η is to be the current length of the block or blocks operated on:

N=0

otherwise:

N= η

To specify SPIREL operation on the entire contents of each of the lowest level blocks of an array which is labelled by a codeword:

x=4
N=0
R=0

(More general application of x=4 is explained in the section on Recursive Application of SPIREL.)

To specify SPIREL operation on a named scalar or block, the name is given after the control word on paper tape or in R for internal control. Then the control word to address a named scalar or block contains:

x=0 for a scalar, 1 otherwise
F=0

and the control word to address all of the lowest level blocks of a named array contains:

x=4
F=0

To specify SPIREL operation, other than read, on a named scalar or all of the lowest level blocks of a named array, the relative Symbol Table (ST) address of the name may be used in typed external communication from the console. The typed control word then contains:

x=5
F=relative ST address of name

Addressing of named quantities is discussed in detail in the section on Symbolic Addressing. In the sections which follow, with the exception of the read operation, a control word which utilizes

x=0
F=location

or

x=1
F=codeword address

may also take the form

x=5

F=relative ST address of name

if the operation can be meaningfully applied to a named scalar or all of the lowest level blocks of a named array.

- Basic SPIREL Operations

The most basic SPIREL operations and the corresponding control word forms are described in this section. This set is sufficient for initial understanding and use of the system.

Read

Read operations are specified by control words with w=0 and x=0,1,2,4, or 5. The y field specifies the read mode:

y=0, octal from paper tape -- 18 octal digits per word, each preceded by a carriage return

=1, hexad from paper tape -- 9 hexads per word, each preceded by a carriage return

=2, zeroes generated by SPIREL

=3, decimal from paper tape, each word followed by a carriage return and in the form discussed in the section on paper tape input formats under Use of SPIREL

=4, hexad with tags and checksum from paper tape in the form punched by SPIREL

For y=0,1 or 2 words are stored with the tag given by z. As words are read (except SPIREL-generated zeroes, specified by y=2), they are checked for proper storage in memory. If the check fails, the word as stored and the location of the word are typed in octal on the console typewriter. The location given is an absolute machine address if x=0 in the control word; it is the relative location in the block if x≠0 in the control word. If SL15 is on, the words read (except SPIREL-generated zeroes, specified by y=2) are not stored but are compared to the contents of the memory location where storage would normally take place. If the comparison fails the word actually stored in memory and its location are typed on the console typewriter as above.

Read control word forms are as follows:

nnnnn 00yz 0000 fffff Read N words in mode y and store with tag z (if y=0,1, or 2) beginning at location F.

nnnnn 01yz rrrr fffff. If codeword at F addresses an array and STEX storage control is active, free that array. Create a block of length N and form its codeword at F. Place octal configuration given by R in the corresponding bit positions of codeword at F. (The last three triads of R specify the B-modification and indirect-addressing bits to be used. The first triad describes the contents of the block being read; its interpretation is described under Codewords.) Set initial index=1. Read N words in mode y and store with tag z (if y=0,1, or 2) into the block labelled by the codeword at F.

nnnnn 02yz 0000 fffff Exactly equivalent to the control word nnnnn 01yz 0002 fffff, used to read a standard vector of data.

B2

nnnnn 04yz rrrr fffff If codeword at F addresses an array and STEX storage control is active, free that array. Create the structure of a standard matrix with N rows and R columns with primary codeword at F. Read N×R words, successive complete rows, in mode y with tag z (if y=0,1, or 2) into the matrix with primary codeword at F.

B1

- Basic SPIREL Operations (continued)

The most basic SPIREL operations and the corresponding control word forms are described in this section. This set is sufficient for initial understanding and use of the system.

Correct

The correct operation is specified by a control word with w=1 and x=0,1,2,4, or 5. Correction implies that some part of an existing unit is to be replaced with new information without completely recreating the unit. Therefore, a correct is just a read into an existing block over the previous contents. The fields y and z specify mode and tag as explained for read operations, and again SL15 on causes comparison instead of storage of what is read.

The correct control word form is as follows:

nnnnn 11yz rrrr fffff Read N words into the block labelled by the codeword at F, where N and R are specified according to the standard rules for address specification. Read in mode y with tag z (if y=0,1, or 2).

● Basic SPIREL Operations (continued)

The most basic SPIREL operations and the corresponding control word forms are described in this section. This set is sufficient for initial understanding and use of the system.

Tagset

Tagset operations are specified by control words with w=2 and x=0,1,2,4, or 5. The purpose is to set tags on words in the memory, and the tag to be set is given by z=0,1,2, or 3 (tag 0 meaning no tag). If y=3, all words in the address range specified are tagged. Otherwise, all words in the address range specified are considered instructions and tagging is selective, with a word being tagged only if its "class" triad = y. (The class of an instruction is given in the third triad from the left.)

The tagset operation is most often used to set tag 3 on instructions in programs to be executed. Then if execution is carried out in the trapping mode, the trace program in SPIREL monitors on the printer the execution of the tagged instruction. This trace output is explained in detail in the section on Use of SPIREL.

The tagset control word forms are as follows:

<u>nnnnn 20yz 0000 fffff</u> Set tag z on all words of class y from location F to location F+N-1, inclusive.

<u>nnnnn 21yz rrrr fffff</u> Set tag z on all words of class y in the block labelled by the codeword at F, where N and R are specified according to the standard rules for address specification.

00023 21 33 0000 00202

Basic SPIREL Operations (continued)

The most basic SPIREL operations and the corresponding control word forms are described in this section. This set is sufficient for initial understanding and use of the system.

Execute

The exeucte operation is specified by wxyz=3100 or 3500 and is designed so that SPIREL will, in essence, transfer control to the address specified as the entry to a closed subroutine. This operation is usually employed as an external directive to SPIREL. Primary control is then with SPIREL; successive programs may be executed, with other SPIREL operations interspersed as desired.

The form of the execute control word is as follows:

00000 3100 rrrr fffff Transfer control to word R of the program which comprises the block labelled by the codeword at F. As a special case, if R=0, transfer control to word 1, or the first word of executable code in the program as determined by the initial index. At entry to the specified program, all fast registers except PF and T7 (and T4 if a named program is specified) are set to the values they had at the time the control word was given to SPIREL. The program executed should be written as a closed subroutine, i.e., it should exit to the address contained in PF upon entry.

The first execute will be used to start a system running after loading as necessary into a fresh SPIREL. This first execute has special effect if STEX has not been activated. All memory in use is consolidated so that items previously loaded may be moved to fill gaps of free storage. All free storage is then available for further allocations by TAKE. In effect, STEX is deactivated prior to the execution called for. See section on storage control for more details.

o Basic SPIREL Operations (continued)

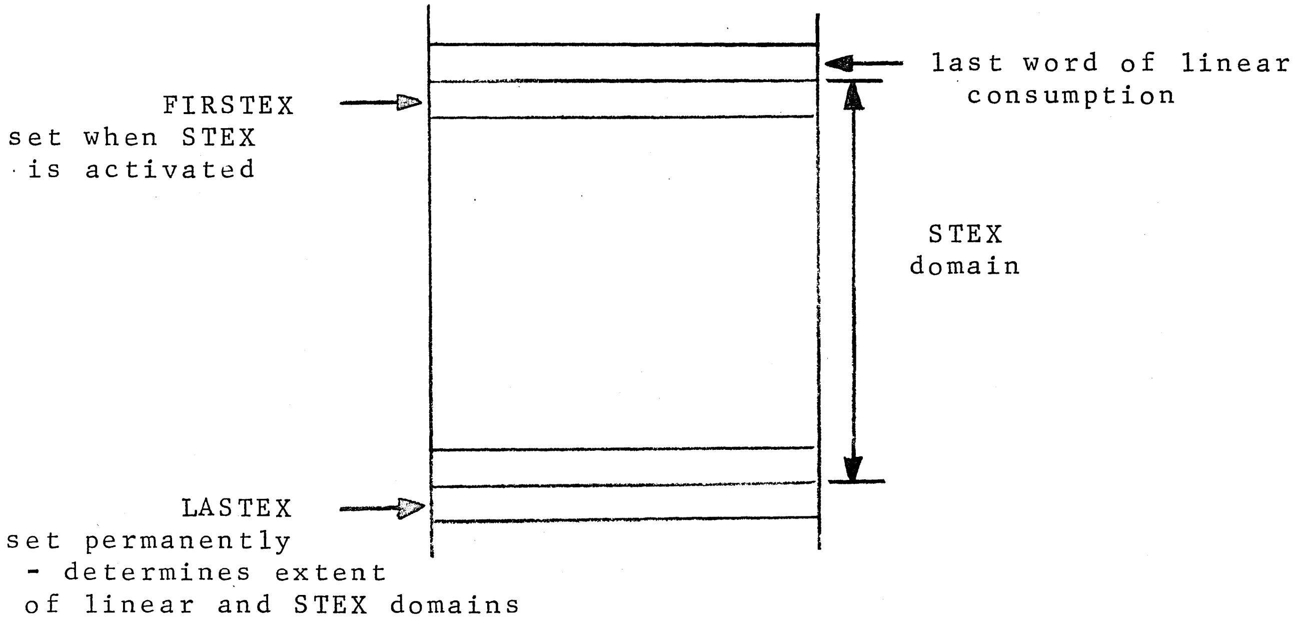
The most basic SPIREL operations and the corresponding control word forms are described in this section. This set is sufficient for initial understanding and use of the system.

Activate STEX

The simple storage control algorithm in SPIREL operates on a principle of linear consumption of space in memory. If STEX, the storage exchange program, is active at the time blocks are created, later redefinition of these blocks will result in the space previously occupied being returned to the system for re-use. Thus, at any given time only space which is currently labelled by a codeword is in use. Activation of STEX causes the STEX domain to be defined

- o from the word beyond the extent of linear consumption, this address being stored at FIRSTEX, location 101
- o through the word before the address stored at LASTEX, location 102.

This definition is illustrated by:


last word of linear consumption
FIRSTEX
set when STEX is activated
STEX domain
LASTEX
set permanently
- determines extent of linear and STEX domains

Any block created after STEX is activated is said to be loaded under STEX control and will be loaded in the STEX domain. Any block may be recreated under STEX control, but old space will be available for re-use only if it was orginally taken in the STEX domain.

The STEX storage control system is described in detail in another section.

The control word which causes STEX to be activated is

00000 3120 0000 00135

It belongs to the execute class of control words. If STEX is activated, subsequent activations are meaningless but harmless.

The first activation of STEX, which may be used after some loading into a fresh SPIREL, has special effect if it precedes the use of an execute control word. All memory in use is consolidated so that items previously loaded may be moved to fill gaps of free storage. The last word of memory in use is then taken as the last word of linear consumption and STEX is activated as described above. In effect, STEX is deactivated prior to the activation called for. See section on storage control for more details.

- Basic SPIREL Operations (continued)

The most basic SPIREL operations and the corresponding control word forms are described in this section. This set is sufficient for initial understanding and use of the system.

Print

Print operations are specified by control words with w=4 and x=0,1,2,4, or 5. The format of output is given by y as follows:

y	format	words/line
0	octal	4
1	hexad	12 (108 chars.)
2	octal, program format	1
3	decimal	5
4	octal, for 8½x11" pages	3
5	octal, with tag	1
6	decimal, with tag	1
7	decimal, for 8½x11" pages	3

For all options except y=1 (hexad), the value of z controls printing at the left of each line the location of the first word on the line:

z = 0, to print location in octal
1, not used
2, to not print location
3, to print location in decimal

The number printed is the relative location in a block for x=1 or 4 (relative) or the absolute machine address for x=0 (absolute).

The format for each decimal number printed is:

floating point

$$\underset{\text{if number is negative}}{-}\underbrace{d.ddddddddddd}_{\text{12 decimal digits}}e\pm\underbrace{dd}_{\text{exponent, 2 decimal digits}}$$

fixed point integer

$$\underset{\text{if number is negative}}{-}\underbrace{ddddddd}_{\text{1-15 decimal digits}}$$

The SPIREL monitoring provided if SL14 is off provides a printed identifier with each block printed.

The print control words are as follows:

nnnnn 40yz 0000 fffff Print in format y N words beginning at location F, with location format given by z.

nnnnn 41yz rrrr fffff Print in format y from the block labelled by codeword at F, where N and R are specified according to the standard rules for address specification and the location format is given by z.

00000 44yz 0000 fffff Print in format y all of the lowest level blocks in the array labelled by codeword at F, with location format given by z.

- Basic SPIREL Operations (continued)

The most basic SPIREL operations and the corresponding control word forms are described in this section. This set is sufficient for initial understanding and use of the system.

Punch

Punch operations are specified by control words with w=5 and x=0,1,2,4, or 5. The punch format is given by y and corresponds to the mode in which the information punched will later be read:

y=0, octal
 =1, hexad
 =2, no information (zeroes to be generated internally at read time to correspond to information specified in punch control word)
 =3, decimal format
 =4, hexad with tags and checksum
 =5, high density hexads with tags, parity and checksum

If z is even, every item punched is preceded and followed by control words sufficient to cause the information punched to be read at a later time into logical position identical to that at time of punching; this is the usual procedure. If z is odd, no control words accompany the information punched.

If z is even and a complete array is punched, the control words punched cause recreation of an identical array when the punched tape is later read. If z is even and only part of an array is punched, the control words punched assume that an identical structure exists when the punched tape is later read, and the punched information is corrected into the structure.

Any tape which is punched with the SPIREL punch operations may then be read under SPIREL control with SL15 on to effect a validation of the punched tape by comparison with the information that was to be punched.

The punch control word forms are as follows:

nnnnn 50y0 0000 fffff Punch in mode y N words beginning at location F.

nnnnn 51y0 rrrr fffff Punch in mode y from the block labelled by the codeword at F, where N and R are specified according to the standard rules for address specification.

00000 54y0 0000 fffff Punch in mode y all of the array labelled by the codeword at F.

o More SPIREL Operations

The SPIREL operations described in this section are not necessary for initial understanding and use of the system.

Address Base Manipulation

SPIREL operations may be applied to blocks and arrays when the pertinent codeword address is known. The primary codeword address of any array is fixed and known to the user, and is in the range 200-277 (octal). When the F fields of control words with x=1 are interpreted as machine addresses, the SPIREL address base is said to be set to zero. For the purpose of applying SPIREL operations to sub-arrays, the SPIREL address base may be set so that the F field of control words is interpreted relative to the 0^{th} element of a block of codewords. Each base change in a sequence of base changes will progress one level into an array, and such a sequence may go to depth 4.

Consider an array of three dimensions with primary codeword at address A and B-modification on each level. Assuming that A exists in the machine, the following series of SPIREL operations illustrates the use of address base manipulations:

operation	effect
----	address base initially set to zero
set address base to A 00000 0600 0000 aaaaa	address base at A_0
print I in octal 00000 4400 0000 iiiii	print two-dimensional array A_I in octal format
set address base to J 00000 0600 0000 jjjjj	address base at $A_{J,0}$
print M in octal, one word per line, with tag 00000 4150 0000 mmmmm	print block $A_{J,M}$ in the format one word per line, octal with tag

operation	effect
set address base back one level 00000 0700 0000 00001	address base at A_0
print K in decimal 00000 4430 0000 kkkkk	print two-dimensional array A_K in decimal format
set address base to N 00000 0600 0000 nnnnn	address base at $A_{N,0}$
print from R,P words at Q in octal ppppp 4100 qqqq rrrrr	print words $A_{N,R,Q},\ldots,A_{N,R,Q+P}$ in octal format
set address base to zero 00000 0700 0000 00000	return to initial address base setting

Note that the 0th element of a block is denoted by 77777, and negative element addresses are specified in one's complement form.

The address base manipulation control word forms are as follows:

00000 06y0 rrrr fffff If y=0 and address base is set to zero, set address base down to F if R is null, to F_R if R is not null. If y=0 and address base is set down to A, set address base down one level to A_F if R is null, two levels to A_{F_R} if R is not null. If y=1 and address base is set down to A, increment base to A+F on same level, then set down to $(A+F)_R$ if R is not null.

00000 0700 0000 fffff Set address base back F levels. If F=0, set address base back to zero.

SPIREL execution of these control words causes no monitoring on the printer, but monitoring of SPIREL operations performed with the address base set to other than zero reflects the successive levels of address base settings in effect.

- More SPIREL Operations (continued)

The SPIREL operations described in this section are not necessary for initial understanding and use of the system.

Insert and Delete Words in Blocks

The insert operation provides a facility for lengthening and shortening blocks labelled by codewords. The insert control word form is

nnnnn 1150 rrrr fffff

The Rth word of the block labelled by codeword at F (counting from the initial index) is addressed, and N words are inserted at that point in the block. N is interpreted in one's complement arithmetic. If N>0, N words containing zeros are inserted beginning at the Rth word, and the former Rth word becomes the R+Nth word; the length of the block is increased by N. If N<0, N words are deleted beginning at the Rth word, and former R+Nth word becomes the Rth word; the length of the block is decreased by N. If R is empty, N words are added to the end of the block. If N is empty, the Rth word and all following are deleted.

The insert operation requires that space for the new form of the block be available while the old form still exists. If STEX is active, the space occupied by the old form is freed when the new form is complete.

The delete operation generates the new form of the block on top of the old form and copies only the segment after that deleted. If STEX is active, the space deleted is freed. If codewords are deleted and freed, the space labelled by the codewords is also freed.

In addition to x=1, x=2,4, or 5 are also allowed.

- More SPIREL Operations (continued)

The SPIREL operations described in this section are not necessary for initial understanding and use of the system.

Change Initial Index

The initial index of the block labelled by the codeword at F is set to N by the control word

nnnnn 1160 0000 fffff

N is specified in one's complement form, and an initial index of zero is designated by N=77777.

If the codeword at F contains B-mods, the first word of the block is subsequently addressed as vector element F_N.

If the codeword at F contains no B-mods, the words preceding word 1 in program F are understood to be symbolic cross references and are immediately loaded. Programs which are written to refer to named quantities, in particular Genie generated programs, require these cross references to scalars and codewords in the Value Table (VT, SPIREL system vector *122) in positions parallel to the positions of the names of the quantities in the Symbol Table (ST, SPIREL system vector *113). The cross references must be loaded, linked to the current VT, each time the program is loaded and prior to its execution.

In addition to x=1, x=2,4, or 5 are also allowed.

- More SPIREL Operations (continued)

The SPIREL Operations described in this section are not necessary for initial understanding and use of the system.

Inactivate Storage

The inactivate operation may be applied to any array by using the control word

00000 1170 0000 fffff

If the array labelled by the codeword at F is in the STEX domain, all storage for the array is freed and F is cleared.

In addition to x=1, x=2,4, or 5 are also allowed.

- More SPIREL Operations (continued)

The SPIREL operations described in this section are not necessary for initial understanding and use of the system.

Monitor

If the STEX storage control system is active, blocks in the STEX domain are subject to being physically moved when it is necessary to concentrate free space. At the console, the user may wish to obtain information about the location of a particular block or the number of words of free storage. The control word

00000 3110 rrrr fffff

causes SPIREL monitoring to occur if SL14 is off. R is interpreted as in standard address specification as the word in the block labelled by the codeword at F for which monitoring is desired. No SPIREL operation is performed on the block.

◎ More SPIREL Operations (continued)

The SPIREL operations described in this section are not necessary for initial understanding and use of the system.

Reorganize

If STEX, the storage exchange program, is active, all free space in memory may be collected into one contiguous block by the control word

00000 3130 0000 00135

This operation is called reorganization, and the word REORGANIZATION is provided on the printer if SL 14 is off. If STEX is not active, this control word has no effect.

◎ More SPIREL Operations (continued)

The SPIREL operations described in this section are not necessary for initial understanding and use of the system.

Execute Control Word Sequence

A block may contain SPIREL control words. The execute control word sequence operation, when applied to the block, will cause SPIREL to interpret the words addressed as control words and carry out the specified operations in order. If in the sequence a control word specifies a named block by F=0, then the next word in the sequence must contain the name of the block (five printer hexads left-adjusted in the format discussed in the section on symbolic addressing. Therefore, a control word sequence N words in length which contains $\mathfrak{m}$ names then contains N-$\mathfrak{m}$ control words.

The control word forms which instruct SPIREL to execute a control word sequence are as follows:

nnnnn 3040 0000 fffff   Execute control word sequence N words in length stored in memory from location F to location F+N-1.

nnnnn 3140 rrrr fffff   Execute control word sequence in the block labelled by codeword at F, where N and R are specified according to the standard rules for addressing.

- More SPIREL Operations (continued)

The SPIREL operations described in this section are not necessary for initial understanding and use of the system.

Map STEX Domain

The structure, i.e. codewords, for arrays in the STEX domain is printed by use of the control word

00000 3150 0000 00135

If a codeword outside the STEX domain addresses a block inside the STEX domain, it is taken as the primary codeword for an array. The structure of the array is then printed. If z is odd, all inactive and active block headers are printed. e.g.,

00000 3151 0000 00135

⊗ More SPIREL Operations (continued)

The SPIREL operations described in this section are not necessary for initial understanding and use of the system.

Deactivate STEX

The control word which causes STEX to be deactivated is

00000 3160 0000 00135

It belongs to the execute class of control words.

The operation involves reorganization of the STEX domain and recourse to linear storage consumption by TAKE in the inactive area of storage. More details are given in the section on Storage Control.

If STEX is not active, deactivation is meaningless but harmless. Activation of STEX after deactivation causes creation of a new empty STEX domain.

- More SPIREL Operations (continued)

The SPIREL operations described in this section are not necessary for initial understanding and use of the system.

Obtain Date and Time

The date and time of day are available in the computer. SPIREL will format this information (14 positions in length) for printing when the control word

00000 4300 rrrr 00131

is executed. The next line actually printed will contain the date and time beginning at print position R. SPIREL will format and print the date and time beginning at print position R when the control word

00000 4310 rrrr 00131

is executed. The print positions are numbered 1-108 (decimal) from left to right across the page. In both cases, if R is empty, the print position is set by SPIREL to 48 so that the date and time will appear at the right side of a page 8½ inches wide.

● More SPIREL Operations (continued)

The SPIREL operations described in this section are not necessary for initial understanding and use of the system.

Print or Punch Symbol and Value Tables

System elements may have names which correspond to single word storage addresses and codeword addresses for arrays. When loaded under SPIREL control, such elements have their names in the SPIREL system vector *113, the Symbol Table (ST). The parallel SPIREL system vector *122, the Value Table (VT), contains the corresponding single word or primary codeword. The index of the last active ST-VT entry is maintained within the SPIREL system as a constant at 117, STPNT.

These tables may be printed with basic SPIREL control words, but a special printing format appropriate to the contents of the tables is provided when the following control word is used:

nnnnn 45y0 rrrr 00000

N words of ST and VT are printed, beginning at ST_R and VT_R. If N and R are empty, the range for printing is implied by the value of y:

y=0 or 3	all entries in the currently active ST-VT
y=4 or 7	all entries with positive indices in the currently active ST-VT

In any case, y=3 or 7 causes only the entries in context (discussed below) in the range specified or implied to be printed out.

Some or all of the quantities with names in the Symbol Table may be punched in checksum format for symbolic loading. To consider for punching N items beginning with the Rth in the ST-VT the following control word is used:

nnnnn 55y0 rrrr 00000

If N and R are empty, the range of entries considered for punching is implied by the value of y:

y=0,1,2, or 3	all entries in the currently active ST-VT
y=4,5,6, or 7	all entries with positive indices in the currently active ST-VT

In any case, the value of y specifies which items to punch:

y=0 or 4 all programs, then all data* in the range specified or implied

y=1 or 5 programs only in the range specified or implied

y=2 or 6 data* only in the range specified or implied

y=3 or 7 all programs, then all data* in context in the range specified or implied

*data consists of scalars in VT and arrays with codewords in VT; data on tape is preceded by a control word to activate STEX.

As a special case, any ST-VT can be printed with the following control word:

```
nnnnn 46y0 rrrr fffff
```

with fffff being the codeword address of the desired ST. The codeword for the corresponding VT must be in the memory location immediately following the ST codeword. N, R and Y are used as described for the system ST-VT.

Prior to the printing or punching called for by these control words, context is automatically determined. This means that the ST entries for all items with negative ST indices which are necessary for support of the programs currently loaded are "marked" with a tag 0; all others will have a tag 1. All items with positive ST-VT indices are automatically taken to be in context, whatever their tags. This assumes that all "library" items are loaded with negative ST-VT indices and that all ST entries with negative indices are given a tag 1 prior to loading of any "private" programs numerically or with positive ST-VT indices; this is the case with SPIREL and the standard library. The same situation may be generated by any user with SPIREL, his own library, and his own private routines.

- Recursive Application of SPIREL

In general, SPIREL control words with x=1 cause the specified operation to be applied to the block labelled by codeword at F. If meaningful, x=1 may be replaced by x=4 and the operation will be applied through the array labelled by codeword at F. This recursive application is accomplished by the use of SPIREL by SPIREL. In other words, when the SPIREL program *126 (XCWD) encounters a control word C with x=4 (except in the case of read, w=0), and F labels a block of codewords, the address base is set down to F, SPIREL is applied to the first N blocks on the next level with N control words C' in which N'=R and R'=0, and the address is set back up one level. If a control word with x=4 is applied to a block which does not contain codewords, the behavior of *126 is as if x=1, and the recursion is terminated. Thus the depth of the recursion is determined by the structure or depth of the array addressed.

As an example, consider the control word to print in decimal

```
                    00003 4430 0002 00200
```

where the array *200 is a standard data matrix. Since the block labelled by the codeword at 200 contains codewords, these control words are generated and delivered for SPIREL execution:

```
                    00000 0600 0000 00200
                         to set address base to 200
                    00002 4430 0000 00001
                         to print the first two words of row 1
                    00002 4430 0000 00002
                         to print the first two words of row 2
                    00002 4430 0000 00003
                         to print the first two words of row 3
                    00000 0700 0000 00001
                         to set address base back one level, to
                         zero.
```

If row 2 in the array *200 had contained codewords for 4 blocks of data, the control word

00002 4430 0000 00002

in the above sequence would have caused further generation of the control words:

00000 0600 0000 00002

to set address base to 200_2

00000 4430 0000 00001

to print all of block $200_{2,1}$

00000 4430 0000 00002

to print all of block $200_{2,2}$

00000 0700 0000 00001

to set address base back one level,
to 200

- Symbolic Addressing

SPIREL provides facilities for addressing scalars, programs, vectors, and matrices by name. A control word with a null F field will cause program *126 (XCWD) to read what follows on paper tape as a 5-hexad name preceded by a cr punch. The name is added to the Symbol Table (ST,*113) if it is not already present. Then the F field is assigned the address in the Value Table (VT,*122) which parallels the name in ST. Under program control a control word with null F may be given in T7, a 5-hexad name left-adjusted in R, and entry made to the second order of *126 with the order

TSR *126, CC+1

Names are represented by 5 printer hexads and are formed by the following rules:

- upper case letters A,B,...,Z are represented by the hexads 40,41,...,71
- lower case letters a,b,...,z are represented by the hexads 40,41,...,71
- the first in a sequence of lower case letters is preceded by a '26' hexad (backspace punch on flexowriter)
- the numerals 0,1,...,9 are represented by the hexads 00,01,...,11
- a name containing less than 5 hexads is filled to 5 hexads by '25' hexads (tab punch on flexowriter) on the right

Examples of 5-hexad names are

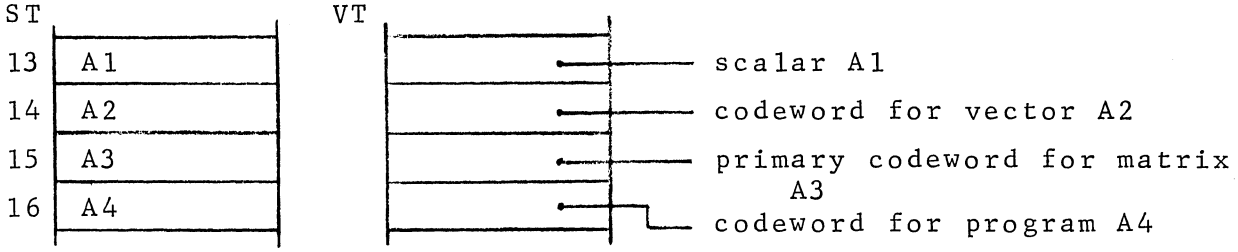
54 40 55 25 25	for	MAN
54 26 40 55 25	for	Man
54 40 55 01 25	for	MAN1
26 54 25 25 25	for	m

The printed load record for a run gives the name of any quantity loaded or referenced symbolically. To the right of each name appears a number F which is the relative Symbol Table address of that name. A SPIREL control word of the form

nnnnn w5yz rrrr fffff

is equivalent to symbolic reference to the Fth Symbol Table entry for all operations except READ. This form is easily typed for console communication to SPIREL.

Consider the ST-VT configuration



The control word with symbol

```
00001 0030 0000 00000 cr 40 01 25 25 25
```

will cause the scalar A1 in decimal form to be read into A1's VT entry.

The control word with symbol

```
00000 4130 0000 00000 cr 40 02 25 25 25
```

or the control word

```
00000 4530 0000 00014
```

will cause the vector A2 with codeword in A2's VT entry to be printed in decimal form.

The control word with symbol

```
00000 5440 0000 00000 cr 40 03 25 25 25
```

or the control word

```
00000 5540 0000 00015
```

will cause the matrix A3 with primary codeword in A3's VT entry to be punched with symbol. The tape punched will load at a later time, creating a matrix with primary codeword in A3's VT entry, even if this entry is not in exactly the same relative VT location.

The control word with symbol

```
00004 0420 0003 00000 cr 40 03 25 25 25
```

will cause the space currently addressed by the codeword in A3's VT entry to be freed. Then a 4 by 3 matrix of zeroes to be created and addressed by the codeword in A3's VT entry.

The control word with symbol

```
00000 4100 0000 00000 cr 40 04 25 25 25
```

or the control word

```
00000 4500 0000 00016
```

will cause the program A4 with codeword in A4's VT entry to be printed out in octal.

The control word with symbol

```
00001 4030 0000 00000 cr 40 01 25 25 25
```

or the control word

```
00001 4530 0000 00013
```

will cause the scalar A1, stored in A1's VT entry, to be printed out in decimal.

The name of a double operand (such as a complex scalar or non-scalar) is attached to the first component of the pair. The second component is named "ditto" which is printed '←←←←←' and is represented by the hexad string

```
75 75 75 75 75       for ←←←←←  ("ditto")
```

For any double operand, its name and "ditto" appear consecutively on the Symbol Table. If K is a double operand, given the name K, SPIREL will operate on the first component; then given the name "ditto", SPIREL will operate on the second component. To operate on the second component of K independent of the first, SPIREL must be given the name K before the name "ditto" with a control word. This may be accomplished by use of the monitor control word which designates no operation to SPIREL:

```
00000 3110 0000 00000 cr 52 25 25 25 25
nnnnn wxyz rrrr 00000 cr 75 75 75 75 75
```

If K has ST relative address 31, then "ditto" for K is at 32. The first component is addressed by the control word

```
nnnnn w5yz rrrr 00031
```

and the second component by

```
nnnnn w5yz rrrr 00032
```

Summary of Control Words

y \ w	0 read	1 correct	2 tagset	3 execute	4 print	5 punch
0	octal	octal	class 0	execute program	octal 4 wds/line	octal
1	hexad	hexad	class 1	on-line control word print only	hexad, 108 chars/line	hexad
2	zeroes	zeroes	class 2	activate STEX	octal-program format 1 wd/line	zeroes
3	decimal	decimal	all classes	reorganize	decimal, 5 wds/line	decimal
4	hexad + tag + checksum	hexad + tag + checksum	class 4	execute control word sequence	octal 3 wds/line 8½x11"	hexad +tag + checksum
5		insert/ delete space	class 5	map codewords in STEX domain	octal with tag, 1 wd/line	high density hexad + tag + parity + checksum
6		change initial index	class 6	deactivate STEX	decimal with tag, 1 wd/line	
7		free storage	class 7	initialize STEX	decimal 3 wds/line 8½x11"	

o Input through the Console Typewriter

When a SPIREL system comes off magnetic tape, control is in the console communication loop, at PAUSE -- so named because the message *PAUSE* appears on the display scope. The blue light on the console typewriter will be on.

At any time, PAUSE may be obtained by going to location 23 or 24.

At PAUSE the console typewriter is used to input commands to the SPIREL SYSTEM. As text is input, it is displayed on the scope. The 'bs' (backspace) key causes the last character entered to be erased. Typing a question mark '?' causes the line to be erased. A carriage return or semi-colon ';' causes the accumulated command to be interpreted and printed.

To simply have text transmitted to the printer, '*' should be used as the first character.

The system interprets and obeys a variety of commands which may be input at the console:

a) Control commands to halt or have SPIREL read control words from paper tape;

b) SPIREL commands to have control words formed and passed to XCWD;

c) REGISTER commands to cause machine registers to be loaded;

d) MAGNETIC TAPE commands to search or read the system tape or pass control to manual mag-tape system;

e) ARITHMETIC commands to perform execution of arithmetic statements. See LIBRARY -- ←IFE.

It is important to understand that while in the console communication loop, at PAUSE or with unterminated text displayed, the only means of communication is through the console typewriter; field switches may not be used. Direct manual control may be exercised at 'HALT', obtained by issuing the 'halt' CONTROL command. At HALT the machine stops; all registers are set as directed by the user.

o Control Commands

Control commands are designated by single characters or keys. In all cases, the command is displayed on the scope and printed.

The control commands are:

halt --
(uc) H(cr)
The machine stops with 'HALT' displayed in U. Lights and registers (except U,R,S,T7,P2,B6) are set as at the last halt, or as changed by intervening executions and register commands. B6 is set to the top of the SPIREL B6-list (*112).[†] Lights and registers may be reset manually at the halt. If a control word is typed into U, pushing CONTINUE causes the typed control word to be passed on to XCWD. If 'HALT' is left in U and there is paper tape in the reader, pushing CONTINUE causes one control word from tape to be passed on to XCWD and FETCHing causes control words on the tape to be processed until the end of tape or a null control word is encountered. Pushing CONTINUE with 'HALT' in U and no paper tape in the reader causes return to the PAUSE.

fetch --
(uc) F(cr)
or
"index"
Control words are read from paper tape and passed to XCWD. Control returns to the console communications loop when the end of tape or a null control word is encountered. The fetch command is equivalent to halt and FETCH with the field switch.

cont --
(uc) C(cr)
A single control word is read from paper tape and passed to XCWD. Control then returns to the console communications loop. The continue command is equivalent to halt and CONTINUE with the field switch.

[†] This occurs only if Console (*173) was entered at the 1st instruction or from location 23. If entry is at the 2nd instruction or from location 24, B6 is left as it was at entry.

o SPIREL Commands

SPIREL commands provide a convenient expression of control word input to XCWD. Each command is input as text followed by 't'[1] which causes the command to be interpreted and the corresponding control word to be passed to XCWD. The form of a SPIREL command text is a two-letter key followed by qualification information in which

n denotes name or octal codeword address
i,j,k denote octal numbers, complemented by minus sign
s denotes type or selected portion[2]
o denotes octal absolute address
f denotes format specifier
d denotes string of data words separated by commas[2]

The SPIREL commands are:

key	function	commands	
AS	activate STEX	AS	activate STEX
BD	base down	BD n s BD n↓i,...j s	set address base down to n set address base down to $n_{i,\ldots,j}$
BU	base up	BU BU i	set address base up one level set address base up i levels
BZ	base to zero	BZ	set address base to zero
CB	create a block	CB i AT n s	create a block (no B-mods) of zeroes i long with codeword at n
CH	check bounds	CH n s	set tag to check bounds on n
CM	create a matrix	CM i BY j AT n s	create a standard matrix of zeroes i rows by j columns with codeword at n
CO	correct	CO n↓i,...,j s = d CO o s = d	correct data words into array n, starting at word $n_{i,\ldots,j}$ correct data words into locations o, o+1,...

[1] carriage return or ';'
[2] Form explained in later section

key	function	commands	
CV	create a vector	CV i AT n s	create a standard vector of zeroes i long with codeword at n
CW	control word	CW i j k n s	send control word with N field=i, WXYZ field=j R field=k, F field=n or null with name n to XCWD
DS	deactivate STEX	DS	deactivate STEX
ER	erase	ER n s	erase array with codeword at n
EX	execute	EX n	execute program with codeword at n
		EX n↓i	execute n starting at word i
ID	insert/delete	ID i AT n↓j s	insert (i>0) or delete (i<0) |i| words at n_j
		ID i AT n s	insert (i 0) i words at end of n
		ID n↓i s	delete word n_i and all following
IN	initial index	IN i AT n s	set initial index of n to i
MO	monitor	MO n s	monitor all of n
		MO n↓i,...,j s	monitor word $n_{i,\ldots,j}$
		MO i AT n↓j,...,k s	monitor i words in array n, starting at word $n_{j,\ldots,k}$
		MO o A	monitor one word at location o
		MO i AT o	monitor i words starting at location o
MP	map	MP	map STEX domain
PR	print	PR n f s	print all of n
		PR n↓i,...,j f s	print word $n_{i,\ldots,j}$
		PR i AT n↓j,...,k f s	print i words in array n, starting at $n_{j,\ldots,k}$
		PR o A f s	print one word at location location o
		PR i AT o f	print i words starting at location o

key	function	commands	
		where f blank causes printing in decimal f = O causes printing in octal H causes printing in hexad P causes printing in program format	
PU	punch	PU n f s	punch all of n
		PU n↓i,...,j f s	punch word $n_{i,\ldots,j}$
		PU i AT n↓j,...,k f s	punch i words in array n, starting at word $n_{j,\ldots,k}$
		PU o A f s	punch one word at location o
		PU i AT o f	punch i words starting at location o
		where f blank causes punching in hexads with checksum f = O causes punching in octal H causes punching in hexads Z causes punching of space-taking control words only	
RE	reorganize	RE	reorganize STEX domain
ST	print ST	ST	print all of ST-VT
		ST U	print user's (positive) portion of ST-VT
		ST i AT j	print i words of ST-VT starting at relative location j
T0	set tag 0	T* n s	set tag on all of n
T1	set tag 1	T* n↓i,...,j s	set tag on word $n_{i,\ldots,j}$
T2	set tag 2	T* i AT n↓j,...,k s	set tag on i words in array n, starting at word $n_{j,\ldots,k}$
T3	set tag 3		
TR	set tag to trace		
		T* o A s	set tag on location o
		T* i AT o	set tag on i words starting at location o
		where * = 0,1,2,3, or R	

⊙ Register Commands

Register commands cause machine registers to be loaded. Each command is input as text followed by 't' which causes the command to be interpreted and the specified register set. The form of a register command is

```
R = d
```

where R is the register name and d denotes a single data word (form explained in later section).

The registers which may be loaded and R for them are:

index registers   CC(effects transfer),B1,B2,B3,B4,B5

⊙ Special Options

(1) Any SPIREL command of the form:

```
XX i AT n (↓u....w)
```

where XX is any valid command, may be written:

```
XX n (↓u...v) FROM j TO k
```

where j and k are program order numbers, etc. Thus the need to know the actual number of words (in octal) being operated on is eliminated. Note that j must be strictly less than k. For example:

```
PR 5 AT V↓2    would become
PR V FROM 2 TO 7
```

a more natural form.

(2) If a GENIE program is compiled with the SL[12] option, any internal variable name or statement label may be used as a subscript in a SPIREL command. However if a GENIE program was begun with RSEQ only statement labels may be used.

For example: if A is an internal constant in program F, one might say

```
PR F↓A
```

which will print the value of A. Or to trace the second FOR loop of program F:

```
TR F FROM ←FOR2 TO ←RPT2
```

thus eliminating the necessity to know either order number or how many words the loop contains.

◎ Magnetic Tape Commands

Magnetic tape commands provide communication with the magnetic tape system with manual control or for reading of the system tape and tape searching. (See separate section for details on MT System). Each command is input as text followed by 't' which causes the command to be interpreted and obeyed.

The magnetic tape commands are:

command	function
PL	read nearest PLACER
MT	go to MT 'arrow' halt
MT i	read block i (octal number) from MT system tape
SP	read nearest SPIREL
TS i	search (overlapped with computation) to block i (octal numbers) on MT system tape

- Data Input Formats

Data may be input from the console into registers (by the register commands) and into absolute locations and arrays (by the SPIREL command to correct).[†] Each data word specifies the content of one computer word and may be input in decimal, octal, or alphabetic form.

Decimal integers, as

```
1968, -29
```

may contain up to 18 characters and none may be '.' or '*' or space.

Decimal floating point numbers, as

```
.1, -2.95678, -0.5*6, 91.7*-9, 5*3
```

may contain up to 18 characters, no spaces, and either '.' or '*' or both must appear.

Octal numbers, such as

```
+252525, +01.01000.00.4001.77765, +10, +3120.0000.00135
```

may contain up to 18 digits after the '+' (which designates octal) and not including any '.' which may be used as a spacer. Leading zeroes are assumed if fewer than 18 digits are given.

Alphabetic words, such as

```
*A BC DEF*,* 123X YZ*
```

may contain up to 9 characters after the initial '*' (which designates alphabetic) and not including the final '*'. The final '*' may be omitted if the word ends the command. Trailing spaces are assumed if fewer than 9 characters are given.

[†] The number data words is found by actual count and any other number is ignored. For complex arrays or locations the data string is split in half, (the first half real and the second half imaginary), and the number of complex pairs is half the number of single words.

- Type Format (s)

The s type format specifier may be blank, R, I, or C. Blank means to determine the type, real or complex (double), and perform the operation accordingly. (Complex can only be implied if a name is on the symbol table.) R (real) and I (imaginary) mean perform the operation on that half of the complex named location or array. C (complex) means imply complex type on the name which causes both real and imaginary parts to be operated on. When a CO (correct) is used and complex may be implied, refer to page 8, Data Input Formats, for details.

- Mode Format (f)

A — absolute. If referring directly to a core address and not a name on the symbol table the A specifier must be used if the memory location is not to be taken as the location of a codeword, from which an array would be printed.

†H — hexad (BCD or literal)

†O — octal

†D — decimal

P — program (only for print)

- Errors

(1) REFERENCE MADE TO SYMBOLIC NAME NOT ON SYMBOL TABLE

†Meaningful only with print and punch command.

- Normal Running

The procedure for running with SPIREL is usually as follows:

1) load SPIREL from magnetic tape
2) load private programs and any data which may be outside the STEX domain from paper tape
3) if data is to be loaded before execution, activate STEX with the control word

   00000 3120 0000 00135

   to pack all items previously loaded and relegate all free storage to dynamic allocation by STEX -- then load data
4) position "run tape" which contains control word to start execution, any data to be read under program control, and perhaps control words for further SPIREL operations
5) CONTINUE to start system running; if STEX has not been activated, first execution will do so -- pack all items previously loaded and relegate all free storage to dynamic allocation by STEX

- Input Paper Tape Formats

The SPIREL read control words designate the read mode to be employed in reading the pertinent data from paper tape. For each read mode there is an appropriate punch format for the data on paper tape.

The octal format (y=0) prescribes that each word consist of exactly 18 octal digits and that each word be preceded by a "spill character", usually a 'carriage return' punch. Octal tapes may be punched manually on the flexowriter. Also, a punch control word with y=0 produces octal format on paper tape, but this is inefficient use of paper tape since only three channels are utilized.

The hexad format (y=1) prescribes that each word consist of exactly 9 hexads and that each be preceded by a "spill character", usually a 'carriage return' or 'tab' punch. The hexad format utilizes all six data channels on paper tape and is equivalent to the octal format at twice the density. Hexad tapes are usually not punched manually on the flexowriter but are easily produced through SPIREL with a punch control word in which y=1.

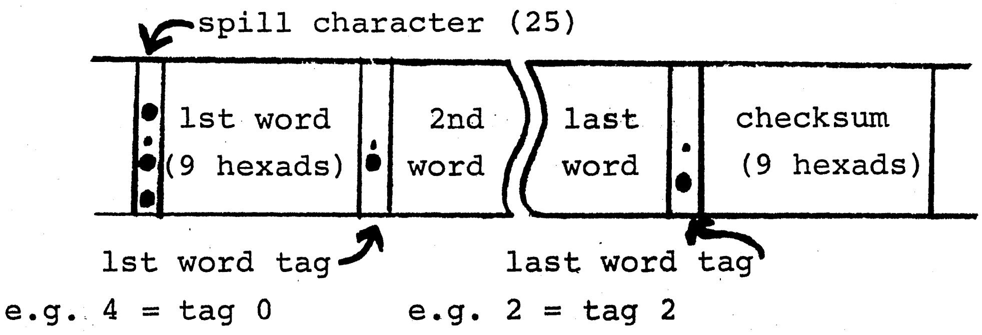
The hexad with tag and checksum format (y=4) is produced only by a SPIREL punch control word with y=4. For example, output tapes from the assembly program and the compiler are in this form. The advantages of this format over the plain hexad format are implied by the name:

- Tags on words are represented on paper tape and reproduced when the tape is read.
- A sum over all words is formed as the tape is punched and represented on the tape. This sum is recomputed when the tape is read and the computed sum is compared to that punched on the tape. This provides a check on both punching and reading.

The format for one word consists of 9 hexads and one tag triad per word, where tag representations are

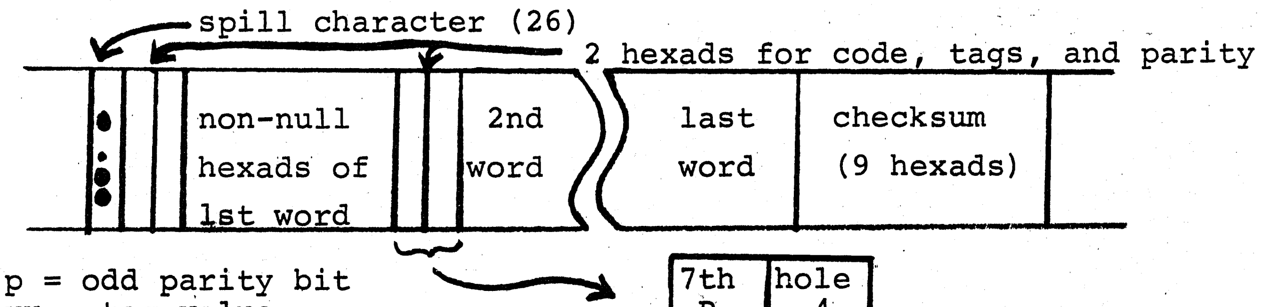
1 for tag 1
2 for tag 2
3 for tag 3
4 for no tag

A spill character (25) is always punched before the data. The physical format of the punched tape is shown below.



The high density hexad with tag, parity, and checksum format (y=5) is produced only by a SPIREL punch control word with y=5. The high density is obatined by not punching null hexads and is preceeded by two hexads of code, tags, and parity. Errors can be more readily detected since the total odd parity on the word, its tag, and the code is punched with each word. A spill character (26) is always punched before the data which is followed by the usual checksum. Tag 0 is punched as 0 and is added as a 0 to the checksum.

The code in the first two hexads of every word show the position of the non-null hexads in the original word in order to read the data back in. The 0's in the nine bit code show the position of the null hexads. Thus the number of hexads punched is the number of 1's in the code. The tapes punched in this format are virtually impossible to read by hand. The physical format of the high density punched tape is shown below.



p = odd parity bit
xx = tag value
(0 ≤ xx ≤ 3)
1 thru 9 = code bits

7th	hole
P	4
x	5
x	6
1	7
2	8
3	9

The words with tag are not separated by any "spill characters" but are immediately adjacent to each other. A set of words punched with a single punch control word is preceded by a single "spill character' and followed by the checksum, one hexad word which is the fixed point sum of right half-word plus left half-word plus tag representation for all words in the set.

The decimal format (y=3) for a single word depends on the internal representation desired for the number. Tapes in decimal format may not be punched by SPIREL, but this is the format most frequently utilized for manually prepared data input tapes. A set of decimal words to be read due to the execution of a single read control word with y=3 is begun with a 'lower case' punch. Spaces and case punches are then ignored, and a punch other than one of

0 1 2 3 4 5 6 7 8 9 . + - e f t *

terminates a number which is being read. The character punched after each number to terminate it is most frequently a 'carriage return' punch. In the particular formats which follow, the letter d stands for a decimal digit, one of

0 1 2 3 4 5 6 7 8 9

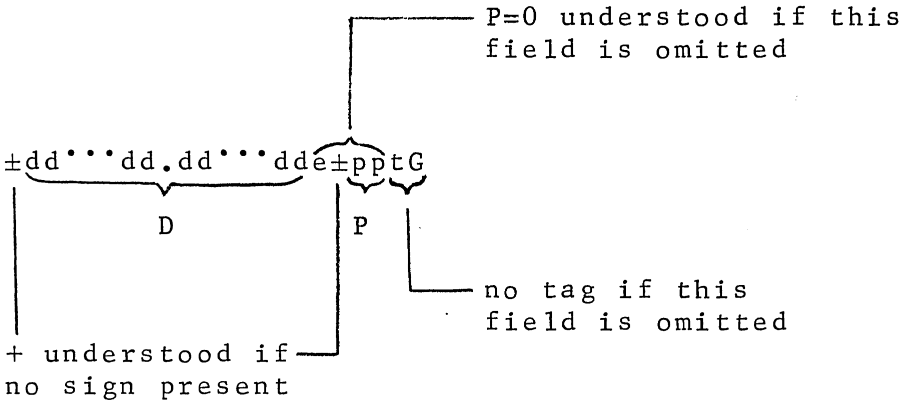
The punches 'e' and '*' may be used interchangeably. If a decimal point is punched in representing a number, it may begin or end the number. The particular decimal formats are as follows:

integer ±D of no more than 14 decimal digits, with tag G=1,2,3,4. G=1,2,3 causes the number to be stored with tag 1,2,3. G=4 causes the number to be stored without changing the tag in memory.

$$\underset{\text{+ understood if no sign present}}{\pm}\underbrace{dd\cdots ddd}_{D}\underbrace{tG}_{\text{no tag if this field omitted}}$$

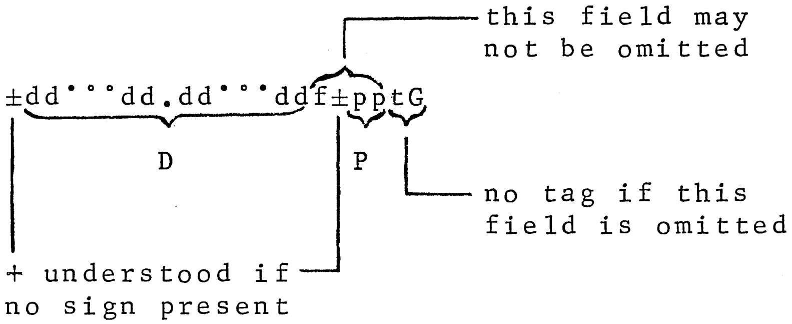
The decimal point is assumed to be to the right of the least significant digit punched and at the right end of the machine word. Integer arithmetic is to be employed.

floating point $\pm D\times10^{\pm P}$, in absolute value between 10^{-78} and 10^{74}, with tag G=1,2,3,4


P=0 understood if this field is omitted
±dd···dd.dd···dde±pptG
D
P
no tag if this field is omitted
+ understood if no sign present

Decimal digits in D beyond the 14th are ignored. Floating point arithmetic is to be employed. The floating point form is recognized only if D contains a decimal point or if the field e±pp (or *±pp) appears.

fixed point fraction $\pm D\times10^{\pm P}$ in absolute value $\geq 2^{-47}$ and<1, with tag G=1,2,3,4


this field may not be omitted
±dd···dd.dd···ddf±pptG
D
P
no tag if this field is omitted
+ understood if no sign present

Decimal digits in D beyond the 14th are ignored. The representation in the machine assumes that the decimal point is at the left end of the mantissa. Fixed point arithmetic is to be employed.

Tracing

Tracing is a means of observing the execution of instructions. This facility is provided in the SPIREL system. The instructions to be traced must bear a tag 3. Mode light 3 and trapping light 6 must be turned on prior to a run in which tracing is to occur; this is the normal ML, TL configuration when SPIREL is initially loaded. The trace is provided by the SPIREL system program *13 (TRACE).

Trace options exist to obtain arithmetic register content (U,R,S) in either octal or decimal and B-register content (B1,...,B6,PF) in octal. For each tagged instruction the first five fields of trace output are as follows:

(CC): address of the instruction relative to the first word of the last program tagged with a tagset control word

(CC'): address of the next instruction to be executed relative to the first word of the last program tagged with a tagset control word; printed only for non-sequential transfer

(M'): the final address formed as a result of decoding field 4 of the instruction

(ATR): contents of the arithmetic tag register (1,2, or 3) after field 4, <u>not</u> after execution of the instruction; blank if the tag is zero

(TF'): T-flags, if they are on; printed as two octal digits, the rightmost bit for T7, next for T6, etc.

(I): the instruction executed, formatted into fields

The arithmetic register trace then provides in octal or decimal:

(S): the contents of S as a result of execution of field 4 of the instruction, before the operation in field 2 is carried out

(U'): contents of U after execution of the instruction

(R'): contents of R after execution of the instruction

And the B-register trace provides (B1'), (B2'), (B3'), (B4'), (B5'), (B6'), (PF') --- the contents of the seven B-registers after execution of the instruction.

Mode Lights may be used at the console for selections of trace options.

To erase the tag from each instruction traced, turn on ML15.

To produce trace output only if a branch of control occurs, turn on ML14.

To obtain B-register trace only, turn on ML13.

To obtain arithmetic trace in decimal, turn on ML10; this option is effective only if T-flags are not present on T4, T5, and T6; this option uses the SPIREL print matrix (*116) and should not be used on programs which are doing set-ups for printing.

To obtain both arithmetic and B-register trace (two lines per instruction traced) turn on ML9 with ML13 off.

The trace procedure saves all machine registers, even the T-flags, except S and P2. If the contents of S is to be preserved from one instruction to the next, neither may be traced. Orders which set P2 or assume that P2 in unchanged may not be traced --- with one exception, that an order which sets P2 to be used only as a transfer address may be traced, but subsequent orders which assume an unchanged P2 may not be traced. This exception allows transfers to the SPIREL programs *136(SAVE) and *137(UNSAVE) to be traced, but these routines themselves may not be traced.

An order to be executed in the repeat mode may be tagged for tracing only if the order which causes entry into the repeat mode is also traced and is of the form

```
                    MLN
      or            SBi,ERM
      or            ABi,ERM
```

and the instruction executed immediately after the repeated instruction is also traced. The trace output for the traced repeated instruction consists of one printed line in which (CC), (ATR), (I), and (S) pertain to the first execution and (CC'), (TF'), (M'), (U'), (R'), and (Bi') pertain to the last execution.

- Arithmetic Error Monitor

While running with SPIREL the following arithmetic error conditions may be monitored: mantissa overflow, exponent overflow, and improper division. To monitor a condition the corresponding trap light must be set on: respectively MOV, EOV, and ÷. If an error which is being monitored occurs, a message is printed. Information provided includes the error made, the location of the instruction generating the error, and the name or codeword address of the program containing that instruction.

- <u>Block Bounds Check</u>

While running with SPIREL, addressing into arrays may be monitored to check that references are not made outside the bounds of the array. This is accomplished by placing a tag 1 on the primary codeword for the array and running with trapping light 13 on, as when SPIREL is initially loaded. Checking is provided by the SPIREL program *20(CHECK).

If a tag 1 is encountered in indirect addressing in field 4 of an instruction, the program CHECK gets control through a hardware trap. The index value k at each level of indirect addressing is then checked to see that

$$i \le k < i+n$$

where i is the initial index and n is the length in the codeword for that level. If an error is detected, information is printed which includes the array name, index values at each level, the location of the offending instruction, and the name or codeword address or the program containing that instruction. Storing out of bounds is inhibited; otherwise, the checked instruction is executed normally.

To successfully check bounds on an array, the tag on the codeword must be preserved during execution. All SPIREL and library routines and all Genie-generated code preserve tags on codewords. Preservation in private code is the responsibility of the user.

The program CHECK uses control trapping on tag 1 after execution of the checked instruction, but this trapping facility may simultaneously be applied to non-checked instructions by the user.

- Diagnostic Dump

If a program stops unexpectedly, the contents of machine registers, the codeword region, the B6-list, and a map of the STEX domain may be printed by using the diagnostic dump, SPIREL system program *120, with entry prefix in machine locations 00027-00036 (octal). The procedure for obtaining this output is as follows:

--Record, mentally or otherwise, the contents of CC as displayed on the console if the instruction at which the stop occurred is of interest.

--Type 00027 (octal) into CC and FETCH to pass control to the diagnostic dump routine.

--A programmed halt occurs within the diagnostic dump routine. Type the recorded (CC) into U and CONTINUE; or simply CONTINUE if (CC) was not recorded.

--Diagnostic dump output is provided on the printer:

Register contents, (CC) and (I) given as 0 if nothing was typed into U at the halt.

Codeword region, with SPIREL address base shown in heading output.

B6-list.

STEX domain map.

--Control is returned to the loop for console communication with SPIREL. Registers are restored as follows: T-registers without flags except T7, B-registers except CC and B6 and PF, special purpose registers except P2, all lights. The SPIREL address base is set to zero, and B6 is set to the first word of the B6-list. The user may continue to use SPIREL.

⊙ High-Speed Memory Dump

If a program fails in such a way that the SPIREL system cannot be reached, a printed record of the memory configuration at the time of the failure is occasionally of assistance in debugging. For this purpose a self-loading High-Speed Dump tape is available at the console.

To load the tape, position the Dump tape in the reader, depress RESET, then LOAD. Do not CLEAR. The program loads at 57400. The contents of machine registers are printed out, and a halt occurs with

(U): 57400 0000 0000 0010

Pushing CONTINUE causes dumping of the contents of memory, from location 10 to location 57400. To change this dump range type into U at the halt type the new upper bound in the first five traids and the new lower bound in the last five traids.

The dump output is printed with four full words per line and the address of the first word at the left of the line. Each full word is split into five fields, corresponding to the fields of the machine instruction. If a word is tagged, an a, b, or c (corresponding to tag 1, tag 2, or tag 3) is printed immediately to the right of the tagged word.

- Error Messages in SPIREL

There are no error halts in the SPIREL system. Nearly all error conditions result in an error message being printed and a return through location 24 to the console communication loop. The printing results in a minimal disturbance to the system so that the source of the error can better be determined.

The message is printed by ERPR *172. Where possible ERPR determines the program and order number where the error originated and prints this information with the message. The possible error messages are listed below and are self explanatory:

1. CHECKING ERROR ON PAPER TAPE INPUT,
2. INSUFFICIENT SPACE IN MEMORY FOR READ,
3. INSUFFICIENT SPACE IN MEMORY FOR INSERT,
4. IMPROPERLY FORMATTED DECIMAL NUMBER,
5. ORDER TRACED WHICH REQUIRED P2 SAVED,
6. ATTEMPT TO SET ADDRESS BASE TO FIFTH LEVEL,
7. ATTEMPT TO SET ADDRESS BASE BACK TOO MANY LEVELS,
8. ATTEMPT TO SET ADDRESS BASE BACK WHEN SET TO ZERO,
9. ATTEMPT TO TAKE BLOCK OF ZERO LENGTH,
10. ATTEMPT TO EXECUTE NON-EXISTENT PROGRAM,
11. ATTEMPT TO SET ADDRESS BASE TO NULL CODEWORD,
12. ATTEMPT TO SET ADDRESS BASE PAST LAST CODEWORD LEVEL,
13. REFERENCE MADE TO SYMBOLIC NAME NOT ON SYMBOL TABLE,
14. SYMBOL TABLE-VALUE TABLE IS FULL,
15. IMPROPER MEMORY CONFIGURATION.

It is not suggested to use the system after any of the above error conditions occur without first rectifying the error.

o <u>Symbol Table-Value Table Print Format</u>

The SPIREL control word

nnnnn 45y0 rrrr 00000

provides parallel printing of corresponding Symbol Table (ST) and Value Table (VT) entries. These "tables" are SPIREL system standard vectors *113 and *122 respectively. The output is in seven fields as follows:

(1) relative address in vector, i.e., index

(2) symbol from bits 1-30 of ST entry

(3) bits 31-39 of ST entry in octal:
000 if VT entry contains a scalar
400 if VT contains a codeword

(4) address field of ST entry, giving the absolute address of the corresponding VT entry in octal

(5) tag on ST entry: 0 if item is in context, 1 otherwise

(6) VT entry: decimal value associated with name in ST entry if it is a single-word quantity; codeword (in octal) for array associated with name in ST, empty if the array does not currently exist.

(7) tag on VT entry

nnnnn 46y0 rrrr fffff

provides parallel printing of the ST-VT whose ST codeword is at fffff. The VT codeword must be in the memory location immediately following the ST codeword.

- SPIREL System on Magnetic Tape

Copies of the SPIREL system for 24K memory are located on the MT System magnetic tape. When one of these copies is read into the memory, control is in the console communications loop. The Mode and Trapping Lights are set to permit tracing of tagged instructions and block bounds checking. Loading of "private" blocks will begin at about location 10000 (octal) and may extend to location 57400 (octal).

● Linear Consumption by TAKE

The simple storage control algorithm in the SPIREL system is called TAKE (in program *135). TAKE is given all of inactive storage as its domain when STEX is deactivated. TAKE operates on the principle of linear consumption of memory. A pointer to the first inactive word of storage, address L, is maintained. A request for M words is satisfied by an allocation of M words at L, and L is incremented by M. This is an irreversible procedure in that space, once allocated, may not be reclaimed for use in later allocations. L is stored in the address field of STORAG, location 100 (octal).

The upper bound on allocatable storage is specified by the contents of location 102 (octal):

LASTEX=last allocatable address +1

In the standard SPIREL only the magnetic tape system communication routine is above allocatable storage. In producing a SPIREL (see System Duplicator section) any upper bound may be specified.

TAKE may be utilized "privately" to obtain blocks of memory in a way compatible with allocation by the SPIREL system. On entry to program *135, (B2)=number of words desired. On exit (B1)=address to first word of block allocated. If the request for a space cannot be satisfied, (B1)=0 on exit.

- Activation of STEX and its Domain

The SPIREL Storage Exchange algorithm STEX (in *135) is activated by the control word

00000 3120 0000 00135

If L is the address of the first word of inactive storage at the time STEX is activated, the STEX domain is [L,(LASTEX)-1], all inactive storage. The address of the first word in the STEX domain is stored in the address field of FIRSTEX, location 101 (octal). Deactivation of STEX for recourse to the TAKE system is explained in a later section. STEX provides optimal use of storage because

- blocks may be freed explicitly, making such space as becomes logically unnecessary available for reassignment to blocks logically required;
- blocks whose codewords are re-used to label new blocks are automatically freed for re-use;
- if the total free space in memory is sufficient to satisfy a request for a block but the "first" free block (explained later) is not large enough, free space is automatically concentrated and the allocation is made.

Any block may be freed at any time. Only if STEX is active and the block is in the STEX domain will the space be available for re-use in later allocations.

The first STEX activation in a fresh SPIREL has special effect if it precedes the use of an execute control word. All memory in use is first concentrated outside the STEX domain which is defined as all free memory. Items loaded prior to this first STEX activation may be moved to fill gaps of free space.

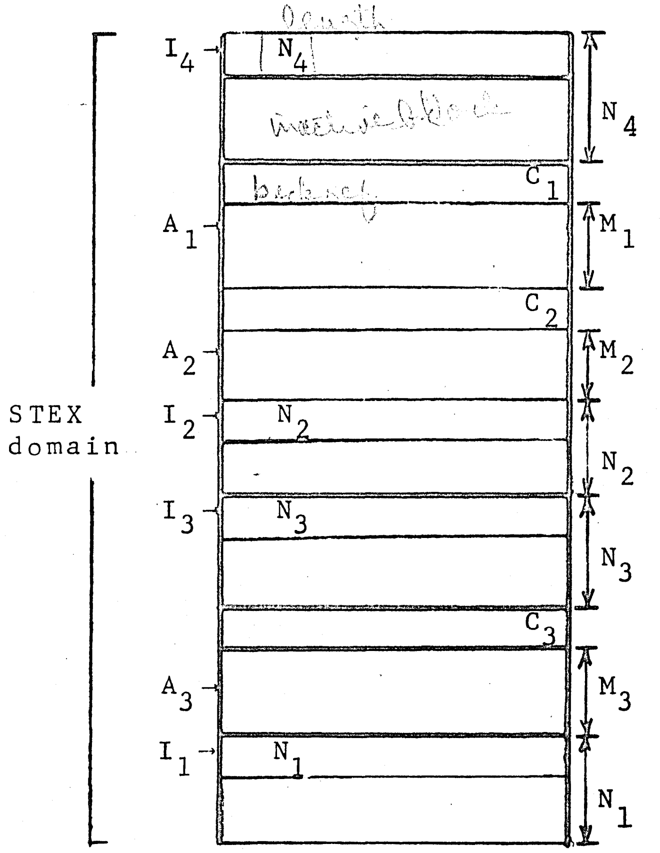
FIRSTEX indicates whether or not STEX is active:

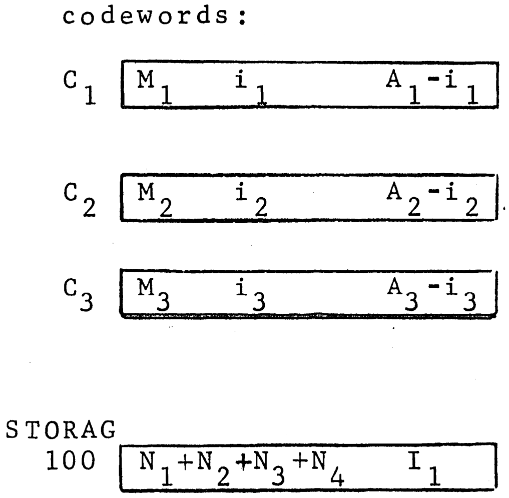
(FIRSTEX) null if STEX is not active
(FIRSTEX) not null if STEX is active
with exponent not null prior to first execute or activate STEX control word
with exponent null after user activation

• Memory Configuration Generated by STEX

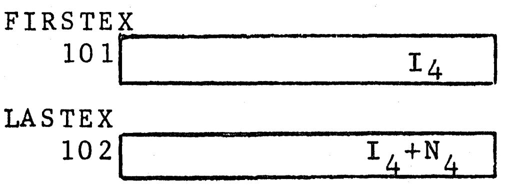
The domain of STEX is divided into active and inactive areas. Each area is further segmented into blocks, not necessarily adjacent. Each active block is labelled by a codeword. The first word of an active block is not a part of the block from the user's point of view; it is called the back-reference for the block and contains in its address field the codeword address of the block. One inactive block may be found from location 100, which contains the total number of inactive locations in the memory in its first 15 bits and the first word address of this inactive block in its last 15 bits. Each inactive block contains in its first word the length of the block it heads in the first 15 bits. The first word of an inactive block is inactive in the sense that it may be activated just as any other word in the block.

An illustration of the STEX memory configuration is given below:





STEX domain illustrated is of length $N_1+N_2+N_3+N_4+M_1+M_2+M_3+3$



In general, the length of the STEX domain is given by

$$K = \sum_{i=1}^{n} N_i + \sum_{j=1}^{m} M_j + m$$

where the domain is divided into n inactive blocks and m active blocks. Note that K is a constant, determined at the time STEX is activated, and K = (LASTEX)-L, where L is the address of the first word of inactive storage at the time STEX is activated.

The codewords that address the active blocks may be located anywhere in the memory, but the blocks of every array must lie wholly inside or outside the domain of STEX.

A duplicate codeword may exist for any array. Its existence is indicated by a duplicate back-reference stored in the same word as the normal back-reference but in the 15 bits adjacent to the address field. Inactivating the duplicate will leave the array unchanged and clear the duplicate. Inactivating the original codeword will have the additional effect of clearing the duplicate.

If it is necessary to move a codeword for an active block, the back-reference for the block must be appropriately altered. For instance, when the block addressed through codeword A is activated the back-reference for the block contains the address A; if (A) is stored at B, back-reference for the block must be changed to B.

STEX offers many advantages for data storage control, but programs may also be loaded into the STEX domain with a few restrictions. Since pathfinder settings are absolute and return is not made through codewords, programs should not be moved in a memory reorganization which STEX may have to perform. This possibility is eliminated if programs are loaded before any space is taken for data that may ever be inactivated. This rule should always be followed.

- Use of STEX

Once activated, STEX may be used directly by the coder with entry parameters

(B1) = codeword address of block on which STEX is to operate,

(B2) = length of block.

STEX first tests the word addressed by (B1). If this codeword is not null, the storage addressed through this codeword to all sub-levels is inactivated by STEX and all codewords are made null. If the codeword is null, no inactivation occurs. Then (B2) is tested. If (B2) ≠ 0, a block of storage of length (B2) is activated with back-reference to the address (B1), the codeword at (B1) remains null.

If the old codeword at (B1) had no a-bit and if sufficient inactive storage is available in the high order locations adjacent to the old first word address then the new FWA will be the same as the old. Note that in the case of a new array of size less than or equal to the old array addressed through the appropriate codeword address, the user is assured that the freed space will be reused. In any case (B1) is set to the FWA of the block activated.

To take N locations to be addressed through codeword C, set (B1) = C, (B2) = N and enter STEX. All space formerly occupied by array C will be inactivated and all associated codewords cleared. Exit will be made with (B1) = FWA of a new block to be addressed through C and (B2) unchanged. The back-reference for the new block is supplied by STEX. If the inactive area is not sufficiently large to meet a request for space, exit is made with (B1) = 0.

To simply inactivate memory addressed through C, enter STEX with (B1) = C and (B2) = 0.

Freeing a block is accomplished by inserting an inactive header in place of the back-reference word. Activating a block is accomplished by obtaining space from the inactive block addressed by 100. If the address portion of 100 is null, then a reorganization will occur before activation. In this case all active memory is written to the low address end of the STEX domain, leaving one inactive block at the high address end. Space ıs then obtained from this new (and only) inactive block. Reorganization will also occur if the block addressed by 100 is not large enough to accommodate the request for storage.

If reorganization occurs and SL14 is off, the message REORGANIZATION is printed. If (B1) = 0 on entry to STEX, reorganization is performed and no space is allocated. A reorganization may also be forced by the control word

00000 3130 0000 00135

When STEX is used directly, the coder must generate his own codewords. The alternative of taking space with a "read" control word provides generation of codewords for the coder.

- Deactivation of STEX

The SPIREL Storage Exchange algorithm STEX may be deactivated by the control word

```
00000 3160 0000 00135
```

If STEX is not active, this control word has no effect. If STEX is active, the following procedure is carried out:

--The STEX domain is reorganized so that all inactive space is collected in a single block at the high end of the domain.

--The TAKE algorithm is reinstated.

--STORAG is set so that linear consumption will commence from the beginning of currently inactive storage, and FIRSTEX is cleared to indicate that there is no STEX domain established.

Two points should be carefully noted:

--Deactivation of STEX involves reorganization, so absolute addresses in the STEX domain may become meaningless as a result of this operation. In particular, a program in the STEX domain should not ask for deactivation of STEX.

--Items in the STEX domain at the time of deactivation will not be in the domain if STEX is subsequently reactivated. Each activation creates a new empty domain beyond all storage in use.

It is difficult to imagine an application of STEX deactivation while a system is running. But a very useful application is in maintenance of a system of programs. A collection of system items may be loaded with STEX active and kept on magnetic tape as a master. For running, STEX may be deactivated prior to loading further. But items may be deleted, added, and changed in the master without any wasted space.

● Vectors, Print Matrix, B6-List

Any component which is of use to the individual programmer is denoted by Δ in the margin next to its name.

Δ *112, B6-List, LISTB6

Length: 200 (octal)

Function: This block is not B-modified. The area is used for working push-down storage, called the B6-list. Index register B6 is initially set to point to the first word in the block. The B6 setting is maintained dynamically as a pointer to the next word in the block which may be used for push-down storage.

*113, Symbol Table, ST

Length: 400 (octal)

Function: This is a standard B1-modified vector. Each entry contains the name and descriptive parameter for an item in the total system being run, an item which is identified symbolically rather than by its address or codeword address. The parallel entry in the Value Table, *122, contains the item or codeword corresponding to the item name in the Symbol Table. The index of the last active entry in the Symbol Table is dynamically maintained at location 117.

Δ *116, Print Matrix, PM

Length: 200 (octal)

Function: This block is not B-modified. The address of the first word of *116 is used as the address field in all SPIREL print orders, except in octal tracing. The print matrix is always cleared immediately after SPIREL-controlled printing.

*122, Value Table, VT

Length: 400 (octal)

Function: This is a standard B1-modified vector. Each entry contains the value of or the codeword for an item in the total system being run, an item which is identified symbolically rather than by address or codeword address. The parallel entry in the Symbol Table, *113, contains the name and descriptive parameter for the item. The index of the last active entry in the Value Table is dynamically maintained at location 117.

*125, Base Address Vector, ADDR

Length: 6

Function: This block is not B-modified. It is used by SPIREL to dynamically maintain a record of all levels through which the base address has been set down and to compute effective addresses from those specified in control words.

*174, Text Vector, TEXT

Length: 14 (octal)

Function: This is a standard B1-modified vector. It contains the current text supplied by *171, SAMPLE, which *173, CONSOLE operates on. The text is in the form of printer hexads.

- Programs

Since programs *135(STEX), *136(SAVE), and *137(UNSAVE) are necessary components of the SPIREL system, they are not included in the lists of supporting routines.

*13, Trace, TRACE

Length: See SYSTEM DUPLICATOR, P. 4.

Function: The SPIREL trace program receives control through hardware trapping due to tag 3 on instructions, both before and after execution of the instruction. Information for each line of trace output is derived, formatted, and printed by this program.

Registers Not Preserved: P2,S

Supporting Routines: ARITH(*14), SETPM(*127), PM(*116) for decimal option only

*14, Arithmetic Error Monitor, ARITH

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program receives control through hardware trapping due to exponent overflow, mantissa overflow, or improper division. The type and source of error are printed.

Registers Not Preserved: P2,S

Supporting Routines: SETPM(*127), PFTR(*147)

*20, Check Block Bounds, CHECK

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program receives control through hardware trapping due to tag 1 on codewords as they are used in indirectly addressing. The values of indices on each level of indirect addressing are checked for being legal. If an error is detected, the source of the error, the array addressed, and index values are printed.

Registers Not Preserved: P2,S

Supporting Routines: TRACE(*13), SETPM(*127), PFTR(*147)

*110, Print Control Word, HDPR

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for online control word monitoring when SL14 is off.

Supporting Routines: SETPM(*127)

*111, Process Matrix, MATRX

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the recursive application of SPIREL as explained elsewhere.

Supporting Routines: all SPIREL components, but only those necessary to perform the specified operation on any particular utilization; see section on SPIREL Component Linkages.

*120, Diagnostic Dump, DIADMP

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system when control is passed to location 00027 (octal). The diagnostic dump formats and prints the contents of the fast registers as explained in the section on Use of SPIREL.

*126, Execute Control Word, XCWD

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is the nucleus of the SPIREL system. It interprets control words and may use other system programs to carry out specified operations. External communication to XCWD from paper tape and from the console is provided within the system and is explained elsewhere. For internal communication, control should be passed to the second word of *126 if the SPIREL operation specified is to be performed on a named item; otherwise control is given to the first word of *126.

Input: (T7)=SPIREL control word to be executed.

(R)=5 left-adjusted printer hexads (with '25' fill if necessary) for name of item to be operated on if control is given to *126 at the second word; in this case F in the control word in T7 is empty.

Registers Not Preserved: none (and SPIREL cannot operate on fast registers)

Supporting Routines: all SPIREL components, but only those necessary to perform specified the operation on any particular utilization; see section on SPIREL Component Linkages.

*127, Set Up Print Matrix, SETPM

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program will format a single word into the print matrix for printing at a specified position on a line if appropriately instructed. It will print the contents of the print matrix and clear the print matrix if appropriately instructed. Common usage of SETPM for the printing of a single line consists of an entry for each word of information to be printed and an entry to have the collection printed and the print matrix cleared. Thus, SETPM provides a facility for composition and output of lines on the printer.

Input: (T7)=information to be set up in print matrix, if any

(B1)=parameter which controls operation of SETPM

(B3)=print position, 1-108 (decimal) at which field set-up should begin, if relevant (print position ≤ 0 is permitted, in which case nothing set up to the left of position 1 is printed)

Operation: on the basis of (B1) on entry:

†(B1)<0, octal format of last 5 triads of (T7) at print position (B3), and increment (B3) by 5

entry (B3) ↓ ↓ exit (B3)

$\underbrace{\text{ooooo}}_{\text{1-5 octal digits, filled with leading blanks}}$

†(B1)=0, octal format of (T7) in 18-position field at print position (B3), and increment (B3) by 18

entry (B3) ↓ exit (B3) ↓

$\underbrace{\text{oooooooooooooooooo}}_{\text{1-18 octal digits, filled with leading blanks}}$

†(B1)=1, hexad format of (T7) in 9-position field at print position (B3), and increment (B3) by 9

entry (B3) ↓ ↓ exit (B3)

$\underbrace{\text{hhhhhhhhh}}_{\text{9 hexads}}$

(B1)=2, print and clear PM, (B3) set to one and (T7) meaningless

†(B1)=3, general long decimal format of (T7) in 18-position field at print position (B3), and increment (B3) by 18

floating point --

$$\overset{\text{entry (B3)}}{\downarrow}\underbrace{(-)}_{\substack{\text{sign of number,}\\ \text{blank if positive}}}\underbrace{d.ddddddddddd}_{\substack{\text{12 decimal}\\ \text{digits}}}e\underbrace{\pm}_{\text{sign of exponent}}\underbrace{dd}_{\substack{\text{exponent,}\\ \text{2 decimal digits}}}\overset{\text{exit (B3)}}{\downarrow}$$

fixed point --

$$\overset{\text{entry (B3)}}{\downarrow}\underbrace{\wedge\wedge}_{\substack{\text{2-16}\\ \text{blanks}}}\underbrace{(-)}_{\substack{\text{sign of number,}\\ \text{precedes leading digit,}\\ \text{blank if positive}}}\underbrace{ddddddddddddddd}_{\text{1-15 decimal digits}}\overset{\text{exit (B3)}}{\downarrow}$$

†(B1)=4, short decimal integer format of mantissa of (T7) in 6-position field at print position (B3), and increment (B3) by 6

$$\overset{\text{entry (B3)}}{\downarrow}\underbrace{(-)}_{\substack{\text{sign of number,}\\ \text{precedes leading digit,}\\ \text{blank if positive}}}\underbrace{ddddd}_{\substack{\text{1-5 decimal digits,}\\ \text{filled with leading blanks}}}\overset{\text{exit (B3)}}{\downarrow}$$

†(B1)=5, general short decimal format of (T7) in 12-position field at print position (B3), and increment (B3) by 12

floating point --

$$\overset{\text{entry (B3)}}{\downarrow}\underbrace{(-)}_{\substack{\text{sign of number,}\\ \text{blank if positive}}}\underbrace{d.ddddd}_{\substack{\text{6 decimal}\\ \text{digits}}}*\underbrace{\pm}_{\text{sign of exponent}}\underbrace{dd}_{\substack{\text{exponent,}\\ \text{2 decimal digits}}}\overset{\text{exit (B3)}}{\downarrow}$$

fixed point --

```
  entry (B3)↓     exit (B3)↓
          (-)ddddddddddd
           /  └────┬────┘
          /   1-11 decimal digits,
         /    filled with leading blanks
   sign of number,
   precedes leading digit,
   blank if positive
```

†(B1)=6, octal instruction format of (T7) at print position (B3), and increment (B3) by 22

```
  entry (B3)┐             exit (B3)↓
            ↓
            oo ooooo oo oooo ooooo
            └─────────┬──────────┘
              18 octal digits
              with 4 blanks
```

†(B1)=10 (octal), format of (T7) on basis of (B2) at print position (B3):

(B2) < 0	as for (B1) < 0, short octal but with leading zeroes printed (no leading blanks)
(B2) = 0	as for (B1) = 0, long octal but with leading zeroes printed (no leading blanks)
(B2) = 1	as for (B1) = 1, hexad but with leading zeroes deleted (filled with leading blanks)

+print and clear PM after set-up if SETPM entered at second word.

Registers Not Preserved: T7

Supporting Routines: BINDC(*155) when decimal formatting is used.

*<u>*130, Find Symbolic Name, SMNAM</u>*

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for operations which are performed on items with symbolic names.

Supporting Routines: TLU(*176)

Δ <u>*131, Print Date and Time, DATIME</u>

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program reads the digital clock through

*132 (CLOCK), sets up the 14-character date and time in the print matrix, and prints the contents of the print matrix if requested to do so. It is used by XCWD to perform "obtain date and time" operations; it may also be used directly.

Input: T7=00000 00p0 rrrr 00000,

where p=0 causes set-up only

p=1 causes set-up, print, and clear

r specifies the print position of the date

r=0 causes set-up for 8 1/2 x 11" pages
(first character in print position 48)

Registers Not Preserved: T7

Supporting Routines: CLOCK(*132), SETPM(*127)

*132, Decode Clock, CLOCK

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program interrogates the digital clock and calendar in the machine and translates the coded time and date into printer hexads. The time is based on a 24-hour clock. CLOCK is used by DATIME(*131).

Input: none

Output: Printer hexads in T5, T6. For example, if CLOCK were executed at 9:45 pm on May 17, 1964, the CLOCK output would be:

```
T5=05/17/64_
T6=21.45_ _ _ _
```

where _ denotes "space".

Registers Not Preserved: B2,T4,T5,T6

Supporting Routines: none

*133, Punch Control Word, PCNTRL

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for any punch operation executed with z even.

Supporting Routines: none

Δ *135, Storage Exchange, STEX

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program performs storage allocation as explained in detail in the section on Storage Control.

Input: (B1)=codeword address of array or block which is to be freed or for which space is to be allocated; 0 if reorganization is desired.

(B2)=length of block to be allocated; 0 to only free space.

Output: (B1)=address of first word of block allocated; 0 if allocation requested and insufficient space available; same as entry if no allocation is requested.

(B2) same as on entry.

Registers Not Preserved: none

Supporting Routines: SETPM(*127)

Δ *136, Save Fast Registers, SAVE

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program uses 12(octal) words on the B6-list for storage of all fast registers and a word denoting the registers to be saved.

Input: (R), bits 46-54, to specify registers to be saved:

45	46								54
	T4	T5	T6	B1	B2	B3	B4	B5	PF

not meaningful to SAVE

The registers are stored on the B6-list in the order shown from right to left (i.e., PF saved first), and then (R) is itself stored on the B6-list. Notice that (R)=-Z on entry causes all nine registers to be saved.

Use: Control should be passed to SAVE by a TRA (not a TSR) instruction which may be traced.

Registers Not Preserved: none

Supporting Routines: none

Δ *137, Unsave Fast Registers, UNSAVE

Length: See SYSETM DUPLICATOR, P. 4.

Function: This program complements SAVE. It obtains from the B6-list the (R) stored by the complementary execution of SAVE, restores all fast registers designated to be saved, and decrements B6 by 12(octal). The T-flags are cleared by UNSAVE.

Use: Control should be passed to UNSAVE by a TRA (not a TSR) instruction. UNSAVE returns via (P2) on entry, and the instruction

```
TRA      *137
```

may be traced.

Registers Not Preserved: none

Supporting Routines: none

*140, Insert or Delete Space, DELETE

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for the operation of inserting or deleting words in blocks.

Supporting Routines: TAKE or STEX(*135) for insert

*141, Change Initial Index, CHINDX

Length: See SYSTEM DUPLICATOR P. 4.

Function: This program is used in the SPIREL system for the operations of changing the initial index of a block and loading cross reference words of programs.

Supporting Routines: TLU(*176) for programs with cross references

*142, Tagset, TAGSET

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for the operation of tag setting.

Supporting Routines: none

*143, Convert from decimal, CNVRT

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used to convert from a general decimally formatted number to binary.

Input: T6, T7 contain printer hexads representing decimal number. R = number of hexads. Hexads are assumed to be left justified. (No + sign in T6, T7)

Error Message: IMPROPERLY FORMATTED DECIMAL NUMBER

Output: Binary number in T7.

Resisters Not Preserved: X, T7.

Supporting Routines: PWRTN(*152)

*144, Print, PRINT

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for all print operations.

Supporting Routines: SETPM (*127)

*145, Punch, PUNCH

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for all punch operations.

Supporting Routines: CLOCK(*132); PCNTRL(*133) for punching tapes with control words (i.e., with z even); PUNCHK(*157) for punching tapes in the hexad with tag and checksum format; BINDEC (*155) for punching decimal tapes.

*146, Execute Control Word Sequence, XCWSQ

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for the operation of executing a control word sequence.

Supporting Routines: all SPIREL components, but only those necessary to perform the operations specified by the control words in the sequence being executed; see section on SPIREL Component Linkages.

*147, PF Trace, PFTR

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used to determine relative value of PF in a program with named or numbered codeword.

Supporting Routines: none

*150, Map STEX Domain, MAP

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program prints the structure of all arrays in the STEX domain, i.e., codewords.

Supporting Routines: XCWD(*126),SETPM(*127).

*151, Print Symbol and Value Tables, PRSYM

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for the printing and punching of ST and VT in the special format described elsewhere.

Supporting Routines: SETPM(*127), CNTXT(*175)

Δ *152, Conversion of Powers of Ten, PWRTN

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for reading of decimal numbers from paper tape. Given the floating point number P and the integer Q, this program computes the floating point number $N=P\times 10^Q$.

Input: (T5)=signed floating point number P
(B2)=integer Q in one's complement form, less than 75 (decimal) in absolute value

Output: $(T5)=P\times 10^Q$, $(PF)=0$ if $|Q|<75$
$(T5)=0$, $(PF)=1$ if $|Q|\geq 75$

Registers Not Preserved: B1, T4

Supporting Routines: none

Δ *153, Multiple Read Decimal, MRDDC

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for the reading of data in decimal input formats from paper tape. It may be used directly to read decimal numbers, convert them as explained elsewhere, and store them.

Input: (B1)=address at which to begin storing the numbers read.
(B2)=number of numbers to read

Output: (B1) same as on input
(B2)=0

Error Message: IMPROPERLY FORMATTED DECIMAL NUMBER occurs if an improper decimal number is read, one which is out of the range permitted for the format used.

Registers Not Preserved: none

Supporting Routines: PWRTN(*152)

Δ <u>*155, Binary to Decimal Conversion, BINDC</u>

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for conversion of a number from its internal binary representation to a decimal representation in printer hexads. It may be used directly for the same purpose.

Input: (T4) = number to be converted, fixed point integer if exponent empty, floating point otherwise.

Output: (T4), (T5) = number in decimal printer hexad form, 18 hexads:

floating point

$$\underbrace{\pm d.ddddddddddd}_{\text{12 decimal digits}}e\underbrace{\pm dd}_{\text{exponent, 2 decimal digits}}$$

fixed point integer

$$\underbrace{\cdots}_{\text{2-16 blanks}}\ \underbrace{\pm dd\cdots dd}_{\text{1-15 decimal digits}}$$

Registers Not Preserved: T6,T7,B1,B2,B3

Supporting Routines: none

<u>*156, Read with Checksum, RDCHK</u>

Length: 57 (octal)

Function: This program is used in the SPIREL system for reading tapes in the hexad with tag and checksum format or high density hexad with tag, parity and checksum format explained in the section on Use of SPIREL. It may be used for the same purpose by an individual.

Input: (B1) = address at which to store first word read.
(B2) = number of words to read.

Error Message: CHECKSUM ERROR ON ABSOLUTE TAPE occurs if the checksum computed while reading does not agree with that read from paper tape.

Registers Not Preserved: T4,T5,T6,T7,B2,B3,B4,B5 ((B2)=0 on exit)

Supporting Routines: none

Note: RDCHK determines the hexad format by the value of the spill character punched by PUNCHK(*157). A 25 is for normal hexads and a 26 is for high density hexads.

Δ <u>*157, Punch with Checksum, PUNCHK</u>

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system for punching tapes in the hexad with tag and checksum format at the normal entry or high density hexad, with parity, tag, and checksum format at the second instruction. The "spill character" (one hexad) that precedes a checksummed sequence is also provided by this program. It may be used for the same purpose by an individual. See the section on Use of SPIREL.

Input: (B1) = address of first word to be punched.
(B2) = number of words to be punched.

Registers Not Preserved: T4,T5,T6,T7,B2,B3,B4,B5 ((B2)=0 on exit)

Supporting Routines: none

Δ <u>*170, Plot Character on Scope, PLOT</u>

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This routine plots one character on the display scope.

Input: (B2) = character as hexad in printer code.
(B5) = position, $0 \leq (B5) < 216_{10}$, for maximum of eight lines, 27 characters per line.

Output: (B5) incremented by 1.

Registers Not Preserved: B2,T4,T5,T6,T7

Supporting Routines: none

*171, Sample Typewriter for Console Input, SAMPLE

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This routine takes input from the console typewriter, maintains the input text on the scope and in memory, and exits when control input is received.

Output: return to PF if 'carriage return' character received, otherwise to PF+1 with (B1)=0 for 'halt', typed 'H'
1 for 'fetch', 'F' or
2 for 'cont', 'C' 'index'

Every command is displayed on the scope and printed before exiting. When SAMPLE is ready to receive new input from the console typewriter, the message *PAUSE* appears on the scope.

Registers Not Preserved: all B5 = # levels in TEXT
#174

Supporting Routines: SETPM(*127), PLOT(*170)

*172, Error Printer, ERPR

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program prints error messages for the SPIREL system programs. See USE OF SPIREL, P. 9, for list of messages and details.

Registers Not Preserved: none

Supporting Routines: SETPM(*127), PFTR(*147)

*173, Interpret Console Input, CONSOL

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This routine interprets commands received through the console typewriter. It will pass control words to XCWD, communicate with the magnetic tape system, set registers, and allow the user to communicate with XCWD from paper tape or the typewriter.

Input: (CC)=23 gives control to CONSOL, text to be interpreted is in vector TEXT(*174).

Output: One or more control words passed on to XCWD in T7.

Registers Not Preserved: T7

Supporting Routines: XCWD(*126, CNVRT(*143), SAMPLE(*171)

*175, Context Determination, CNTXT

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program determines all named programs with negative ST indices that are necessary for the support of all numbered programs and named programs with positive ST indices. Library routines may be loaded with negative indices to -0 and ST entries bearing tag 1. Then all private routines are loaded. CNTXT will operate on the ST entries with negative indices, leaving tag 1 on those not necessary for support and clearing the tag on all those "in context".

Supporting Routines: none

Δ *176, Table Look-Up, TLU

Length: See SYSTEM DUPLICATOR, P. 4.

Function: This program is used in the SPIREL system to determine the index of the Symbol Table entry for a name. If the name "looked for" does not appear on the Symbol Table, TLU adds it and increments the current last active index at location 117 by one.

Input: (T4) = left-adjusted 5 printer hexad representation of the name to be "looked for" on Symbol Table, *113; the rest of (T4) is irrelevant to TLU. Entry at CC+1 will cause the name looked for not to be placed on the Symbol Table if it was not found.

Output: (B1) = index of ST entry for the name "looked for".
(PF) = 1 if the entry was added to the active portion of the Symbol Table; 0 otherwise.

Registers Not Preserved: B1, PF

Supporting Routines: none

• Purpose of the Duplicator

When a system of programs has been debugged and extensive production running is contemplated or when memory space becomes critical, it may be advantageous to produce a self-loading paper tape bearing necessary SPIREL components, library routines used, and the coder's private system elements. The system duplicator SPIDUP, program *10, is designed for this purpose. This program is not itself a SPIREL component. SPIDUP has codeword address 10 (octal) and this will not conflict with normal codeword locations.

If the amount of space allowed for the SPIREL system is known, one may determine if the desired SPIREL system can be accommodated. This is simply done by adding up the lengths of all programs and vectors needed plus 300 (octal), the normal loading address of SPIREL. Since the system is being continually modified, the current lengths of all its components are listed on Page 4 of this section.

The dump tape for the entire SPIREL system contains the control word to execute *10 plus the "dump words".

- Use of the Duplicator

A dump tape is used to tell program *10 what system elements are to be punched. The dump tape consists of a series of dump words, each preceded by a 'carriage return' punch and consisting of 18 octal digits. The dump word forms are as follows:

00000 0100 0000 fffff for all of the block (program or vector) with codeword at F.

nnnnn 0000 0000 fffff for N words beginning at machine address F.

nnnnn 0120 rrrr fffff for a control word which will cause a block of N zeros to be created with codeword at F and R in the dump word at the corresponding position in the codeword. If N in the dump word is empty, N and R will be obtained from the codeword at F at time of punching.

nnnnn 1100 rrrr fffff for N words beginning at the Rth element of the block with codeword at F. A "correct" control word is punched which is meaningful at later reading only if the block has been previously created.

A null dump word (18 '0' punches preceded by a 'carriage return') terminates the list of system components to be punched. Hexads on the dump tape after the null dump word will be reproduced onto the tape punched. Except for blocks of zeros, all components are punched in the hexad with tag and checksum format.

The complete system from which components are to be punched to produce a self-loading system tape should be in the machine. STEX must not be active. Then execution of program *10 results in the following steps:

- A leader is punched. It contains the SPIREL loader and RDCHK, program *156.
- The programmed stop

  Z HTR CC

  occurs, and the dump tape should be readied in the reader.
- CONTINUEing causes the dump tape to be read and the specified system components to be punched, until the null dump word is read.

- The SPIREL system tail is punched.
- Any information on the dump tape beyond the null dump word is duplicated, hexad for hexad.

● SPIREL Generation

A complete SPIREL system is produced by use of a dump tape as follows:

	dump words		components and length (octal)	
	00001 0000 0000 00102	†°	LASTEX	
	00000 0120 0000 00112	†	LISTB6	200
no	00000 0120 0000 00113		ST	400
	00000 0120 0000 00116	†	PM	200
no	00000 0120 0000 00122		VT	400
no	00000 0120 0000 00174	†	TEXT	14
	00000 0100 0000 00013		TRACE	335
	00000 0100 0000 00014		ARITH	125
no	00000 0100 0000 00020		CHECK	263
	00000 0100 0000 00110		HDPR	106
no	00000 0100 0000 00111		MATRX	213
no	00000 0100 0000 00120		DIADMP	130
	00000 0100 0000 00125	†	ADDR	6
	00000 0100 0000 00126	†	XCWD	433
	00000 0100 0000 00127	†	SETPM	155
	00000 0100 0000 00130		SMNAM	57
no	00000 0100 0000 00131		DATIME	14
no	00000 0100 0000 00132		CLOCK	55
no	00000 0100 0000 00133		PCNTRL	~~14~~ ← CSTO 32
	00000 0100 0000 00134 00135	†	STEX	364
	00000 0100 0000 00136	†	SAVE	7
	00000 0100 0000 00137	†	UNSAVE	36
no	00000 0100 0000 00140		DELETE	277
	00000 0100 0000 00141		CHINDX	52
	00000 0100 0000 00142		TAGSET	15
no	00000 0100 0000 00143		CNVRT	53
	00000 0100 0000 00144		PRINT	113
no	00000 0100 0000 00145		PUNCH	160
no	00000 0100 0000 00146		XCWSQ	27
no	00000 0100 0000 00147		PFTR	46
no	00000 0100 0000 00150		MAP	141
	00000 0100 0000 00151		PRSYM	152
no	00000 0100 0000 00152		PWRTN	41
no	00000 0100 0000 00153		MRDDC	154
no	00000 0100 0000 00155		BINDC	103
no	00000 0100 0000 00157		PUNCHK	56
no	00000 0100 0000 00170	† X	PLOT	141
no	00000 0100 0000 00171	† X	SAMPLE	203
no	00000 0100 0000 00172		ERPR	225
	00000 0100 0000 00173	† X	CONSOL	1035
	00000 0100 0000 00175		CNTXT	54
	00000 0100 0000 00176		TLU	43
	00000 0000 0000 00000		null word to end dump	
no	77777 1160 0000 00172		correct index of ERPR	
no	77776 1160 0000 00173		correct index of CONSOL	

†components which must be included in minimal SPIREL.

°The dump word for LASTEX may be omitted; the end of allocatable memory will then be set to E-377 (where E is the last word in memory) to exclude only the magnetic tape system programs. A small setting will provide private absolute storage if such is desired.

For determination of a sufficient set of SPIREL components to satisfy the requirements of a private system, reference should be made to the diagram of SPIREL component linkages. If HDPR, program *110, is to be eliminated, all control words must be executed with SL14 on unless location 110 routes control immediately to the program using *110. Set

```
          (110):    00000 0000 0001 00000
```

before execution of *10 and include the dump word

```
                    00001 0000 0000 00110
```

in the dump tape.
To alter the initial SPIREL loading address (normally octal 300) or the message printed upon completion of paper tape reading, refer to the symbolic listing of SPIDUP in the SPIREL reference notebook.

- Purpose

Backing storage can be used as an extension of effective memory space or as a fast alternative to paper tape for input and output. For such applications the SPIREL-compatible backing storage system (SBSS) is provided. Data created and processed by SPIREL in memory may be output onto magnetic tape by SBSS; data input from magnetic tape by SBSS may be processed by SPIREL.

The SPIREL backing storage system uses codeword addresses 160-167 and 177. It may be loaded with any SPIREL. STEX must be active for use of SBSS.

- Use

The SPIREL backing storage system, SBSS, provides a data transmission link between memory and a device. For transmission to occur the logical device (numbered 0-7) must be attached to a tape unit (numbered 0-3). A unit must be detached before it is manipulated other than under SBSS control, by programs or manually.

The logical unit of information on a device is a file. While a unit is in use for output an output file is open on the device; while it is in use for input an input file is open. Physically, a file consists of a sequence of records of uniform length. For an open file there exists a buffer area in memory which contains the last record read or the accumulation for the next record to be written. An output file must be closed to assure that all data output to the file is actually out of the buffer.

Output files are written sequentially on a device and are numbered 1 to 252_{10}; files may not be overwritten. A device may be positioned at any file. For input, the file at which the device is positioned is the one read.

SBSS maintains status and control information about each device in a device status block.

There are three levels of SBSS usage:

1) On the array level the user may read and write SPIREL arrays by simply specifying the codeword address and the device number to the program ARRAY(*166). All buffering is handled automatically on input and output. Storage allocation for arrays is provided on input.

2) On the buffered level the user may read and write any memory area by specifying the first word address, the number of words, and the device number to the programs GET(*160) for input and PUT(*161) for output. All buffering is handled automatically.

3) On the direct level the user may read and write buffer loads (records) by specifying the device number to the programs READ(*162) for input and WRITE(*163) for output.

On all levels, the program CONTRL(*167) is used for logical (non-transmission) operations:

to attached and detached devices;

to open and close input and output files;

to specify buffer length for a device;

to position a device at a specified file, by rewinding, by backspacing one file or record, or near the end of information on the device;

to obtain status information about a device;

to set control options for a device;

to reposition after restart

Various comments on the use of SBSS are appropriate. Exercise of the system will no doubt suggest more.

- Files need not be explicitly open by the user. They will be opened when data transmission occurs.
- Output files should be closed as soon as possible after they are completed. While an output file is open, there is constant danger of transport misbehavior causing a file to become unreadable.
- When attaching a device to a tape unit, the exact number of files on the tape must be given or all sorts of bad things can happen.
- Never have more than one device attached to the same unit. It leads to immense confusion and almost certain file destruction.

- Tape Format

Marker allocation on an SBSS written magnetic tape is as follows:

marker 0 -- unused

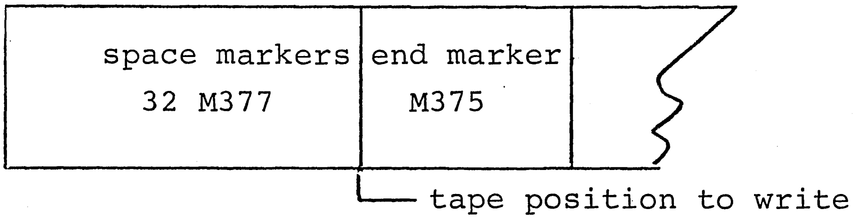
markers 1-374_8 -- file markers so that 252_{10} files are allowed

marker 375 -- end-of-information marker

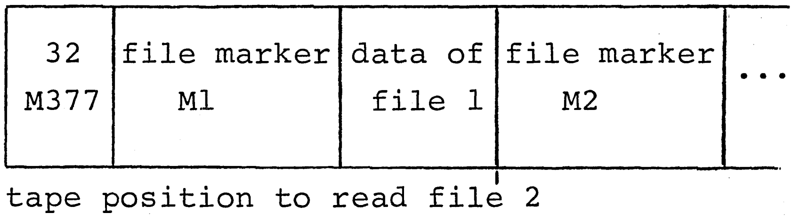
marker 376 -- record marker

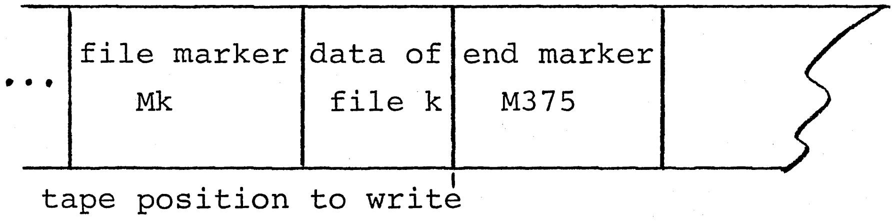
marker 377 -- space marker

A new reel of tape containing no files, but ready to be written, bears only a leader and an end-of-information marker.



A reel containing k files bears k file markers and an end-of-information marker.





The k^{th} file on a reel begins with a file marker and contains any number of record markers and records, all of the same length.

file marker Mk	record marker M376	data of first record	record marker M376	...

...	record marker M376	data of last record	file marker Mk+1 or end marker M375	

The i^{th} record in the k^{th} file on a reel begins with a record marker and contains n words of which the first two describe the record.

record marker M376	word 1 of record	word 2 of record	words 3 to n containing data of record

Word 1 contains

n(15 bits), length of the record;

k(9 bits), file number;

number of data words used (15 bits), null if this is not the last record in file k so that n-2 words are used;

i(15 bits), record sequence number.

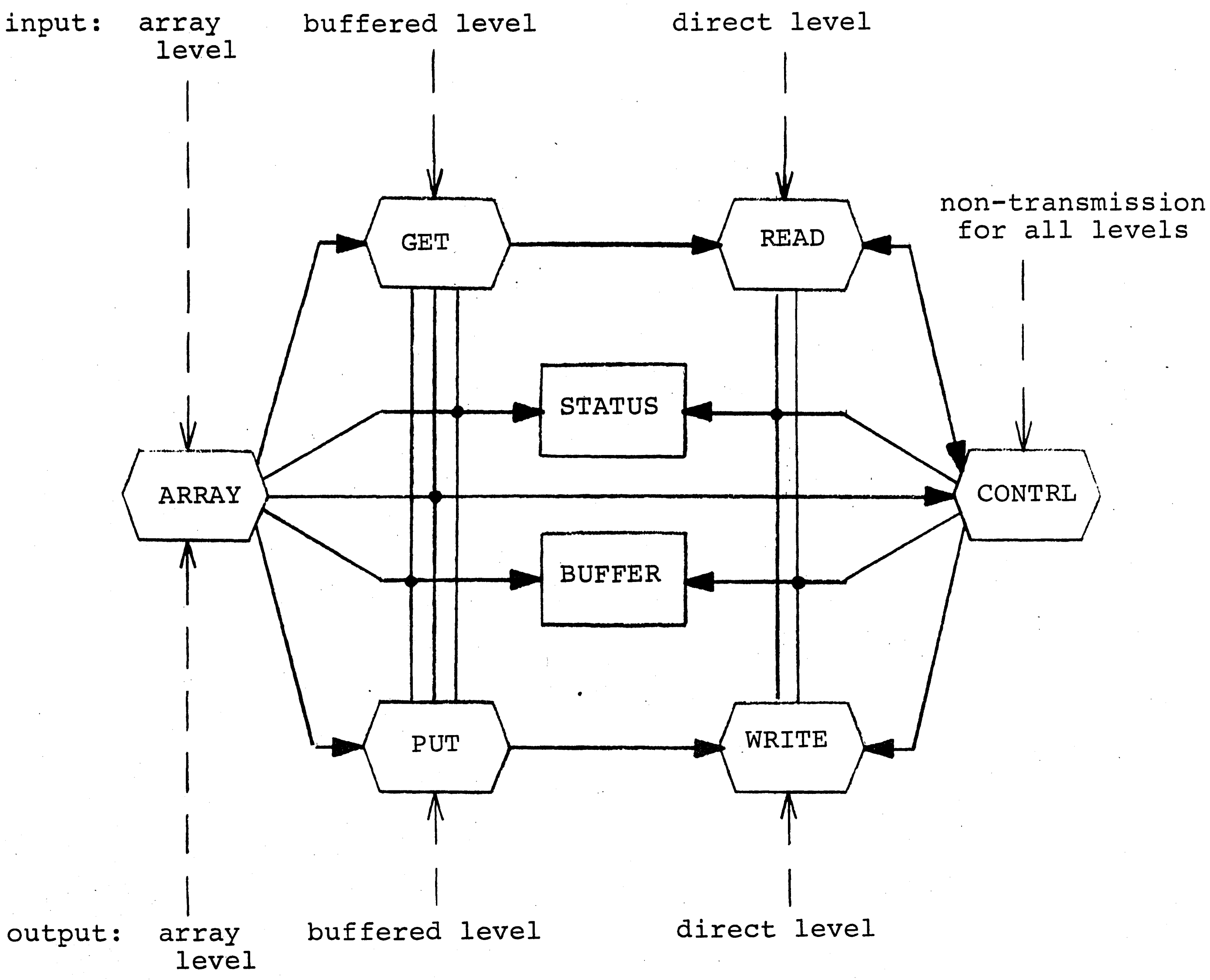
Word 2 contains a symmetric difference checksum for the record.

Organization

The SPIREL backing storage system, SBSS, uses codewords as follows:

160	GET	read (B2) words to memory starting at address (B1) from device (B3)
161	PUT	write (B2) words from memory starting at address (B1) to device (B3)
162	READ	read one record from device (B3) into buffer for that device
163	WRITE	write buffer for device (B3) as one record on that device
164	BUFFER	buffer matrix with row i as the buffer for device i (i=0,1,...,7) and row -1 as the check buffer for all open output files needing one
165	STATUS	device status matrix with row i containing status for device i (i=0,1,...,7)
166	ARRAY	read or write array with codeword address (B1), relative to SPIREL address base setting, to or from device (B3)
167	CONTRL	for device (B3), perform non-transmission operation specified by (B1), with parameters specified in B2 and B4
177	ERRT	tape error routine

The levels of SBSS usage and interconnection of components is shown in the following diagram.



- SBSS Programs

All programs preserve registers except T7 which are not used for input or output.

*160, Input from Device to Memory, GET

Length 63 (octal)

Function: This routine transmits data from a device buffer to the user's memory locations, opening an input file if necessary and causing the buffer to be filled from the device as required.

Input: (B1)=address at which to begin storing the data read
(B2)=number of words to read
(B3)=device number

Special entry: at second order to simply skip over data; transmission suppressed and input (B1) irrelevent.

Output: (T7)=number of words transmitted (or skipped)

Return: to PF+1 normally
to PF if end-of-file encountered before specified number of words transmitted (or skipped) or input (B2)=0 and end-of-file encountered

Errors: none

Supporting Routines: READ(*162), CONTRL(*167)

*161, Output from Memory to Device, PUT

Length: 42 (octal)

Function: This routine transmits data from the user's memory locations to a device buffer, opening an output file on the device if necessary and causing the buffer to be emptied to the device as required.

Input: (B1)=address at which to begin obtaining data to write

(B2)=number of words to write

(B3)=device number

Return: to PF

Errors: none

Supporting Routines: WRITE(*163), CONTRL(*167)

*162, Input from Device to Buffer, READ

Length: 77 (octal)

Function: This routine reads the next record of the current input file on a device into the buffer for that device, opening an input file if necessary.

Input: (B3)=device number

Special entry: at second order to obtain first word of next record as output in T7 but remain positioned at same next record.

Return: to PF if no errors

Errors: 01 record sequence error detected

02 cannot read record in 8 attempts

Supporting Routines: CONTRL(*167), ERRT(*177)

*163, Output from Buffer to device, WRITE

Length: 103 (octal)

Function: This routine writes the buffer for a device into the current output file for that device and checks what is written. An output file must be open on the device.

Input: (B3)=device number

Return: to PF if no errors

Errors: 03 no open output file
04 UME in memory
05 cannot write record in 8 attempts

Supporting Routines: ERRT(*177)

*166, Transmit Array to or from Device, ARRAY

Length: 204 (octal)

Function: This routine transmits a SPIREL array, with tags, between memory and a device, opening an input or output file as necessary.

Write

Entry: at first order

Input: (B1)=codeword address of array to be written, relative to SPIREL address base setting
(B3)=device number

Return: to PF

Errors: none

Read

Entries: at second order to read an array
at third order to skip an array

Input: (B1)=+0 to read array with same codeword address (or name) as when written
=codeword address (≠+0) at which to read array, relative to SPIREL address base setting

Return: to PF+1 normally
to PF if there is no data to be read from the file upon entry

Errors: 06 end-of-file encountered before read (or skip) complete

Write and Read

On Tape:

primary codeword

length	initial index	modifiers	codeword address
name in BCD			

zero if no name

zero if there is a name

subsidiary codeword

length	initial index	modifiers	relative codeword address

runs from 1 to length

Supporting Routines: XCWD(*126), GET(*160), PUT(*161), CNTRL(*167), ERRT(*177)

*167, Control for Tape System, CONTRL

Length: 430 (octal)

Function: This routine performs non-transmission functions in the SPIREL backing storage system; file control, logical operations, and positioning.

Input: (B1) to specify operation
(B3)=device number
(B2), (B4) may contain parameters

Output: may be in T7

Return: to PF if no error

Operations: on the basis of $(B1)_8$ on entry

(B1)=0, open input file

If device (B3) is not attached to a unit, no operation is performed.

Any open file is first closed

If device (B3) is positioned at the end of information, an input file cannot be opened and error 07 occurs.

If all is well, the next input file on device (B3) is opened and a buffer is created if necessary. If there is not memory space for the buffer, error 13 occurs.

(B1)=1, close input file

If device (B3) is not attached to a unit or there is no input file open on the device, no operation is performed.

Otherwise, the current input file on device (B3) is closed; the associated buffer is freed if the switch in the device status block dictates.

(B1)=2, open output file

If device (B3) is not attached to a unit, no operation is performed.

Any open file is first closed.

If the tape is full (252_{10} files), error 10 occurs.

If the switch in the device status block indicates that the device is write-protected, error 11 occurs.

If all is well, a new output file is opened on device (B3) and a buffer is created if necessary. If there is not memory space for the buffer, error 13 occurs. On exit (T7)=number of file opened.

(B1)=3, close output file

If device (B3) is not attached to a unit or there is no output file open on the device, no operation is performed.

Otherwise, the buffer for device (B3) is emptied if necessary and the current output file is closed. The buffer is freed if the switch in the device status block dictates, and the check buffer may be freed also. On exit (T7)=number of file closed.

(B1)=4, attach device

and (B4)=logical tape unit, 0-3

(B2)=number of files on the tape, 0 for a new output tape

If device (B3) is already attached to a unit, no operation is performed.

If a non-existent unit is specified in B4, error 12 occurs.

Otherwise, device (B3) is attached to unit (B4). The tape is rewound and a leader is written if input (B2)=0 (new tape).

(B1)=5, set buffer length

and (B2)=length, number of words

If a file is open on device (B3), no operation is performed. Buffer length may be set only when no file is open, but a buffer is not created until it is needed for transmission.

If no buffer is specified, devices 0-3 have buffer length 400_8, devices 4-7 have buffer length 1000_8.

(B1)=6, detach device

If device (B3) is not attached to a unit, no operation occurs.

Any open file is first closed.

Device (B3) is detached from its unit; the tape is rewound, and the device is left positioned at file 1.

(B1)=7, rewind

If device (B3) is not attached to a unit, no operation is performed.

Any open file is first closed.

If there are files on device (B3), the tape is rewound and left positioned at file 1.

(B1)=10, backspace file

If device (B3) is not attached to a unit, no operation is performed.

Any open file is first closed.

Then the tape is positioned at the beginning of a file.

On exit

(T7)=-1 if tape was at beginning of file 1 and could not be backspaced

=0 if tape was positioned within a file and was backspaced to the beginning of that file, or was backspaced to the beginning of the preceding file.

(B1)=11, backspace record

On exit

(T7)=-1 if device (B3) is not attached to a unit or no file is open on the device and no positioning is done

=0 if tape was positioned at the beginning of a file and could not be backspaced within the file

=1 if tape was backspaced one record in the current file

(B1)=12, position at file

and (B2)=file number

On exit

(T7)=-1 if device (B3) is not attached to a unit or a file is open on the device and no positioning is done

=0 if tape is positioned at the beginning of file (B2)

=1 if file (B2) does not exist on the device and no positioning is done

(B1)=13, position near end-of-information

On exit

(T7)=-1 if device (B3) is not attached to a unit or a file is open on the device and no positioning is done

=0 if the tape is positioned at the last file

(B1)=14, get field from device status block

and $(B2)_8$ specifies the field, as explained in the section on device status:

(B2)=1, LEFT	(B2)=4, FILE NO.	(B2)=7, FILES
=2, NEXT	=5, LAST	=10, UNIT
=3, LENGTH	=6, REC NO.	=11, SWITCH

On exit, (T7)=field value for device (B3), 0 if (B2) is not meaningful.

(B1)=15, set switches

and (B4) specifies switches to turn off
(B2) specifies switches to turn on

Switches, as explained in the section on device status, for device (B3) are turned on for 1's in (B4), then off for 1's in (B2). System switches are immune from setting by this operation.

(B1)=16, re-position after restart

If device (B3) is not attached to a unit, no operation is performed.

Otherwise, the tape is rewound and then positioned as designated in the device status block for device (B3).

Errors: 07 cannot open input file
10 cannot open output file
11 attempt to output on protected device
12 attempt to attach device to non-existent unit
13 insufficient space for buffer

Supporting Routines: STEX(*135), READ(*162), WRITE(*163) ERRT(*177)

- <u>Device Status and Control</u>

The device status matrix, STATUS(*165), contains one device status block per row, row i for device i, i=0,1,...,7. B3 is the row modifier, and there is no column modifier.

Each device status block is three words long and contains information about the device and its buffer:

DEVICE STATUS BLOCK

<table>
<tr><td>word 1</td><td colspan="5">LEFT</td><td colspan="8"></td><td colspan="5">NEXT</td></tr>
<tr><td>word 2</td><td colspan="5">LENGTH</td><td colspan="3">FILE NO.</td><td colspan="5">LAST</td><td colspan="5">REC NO.</td></tr>
<tr><td>word 3</td><td colspan="5">FILES</td><td colspan="5">ERROR</td><td colspan="4"></td><td colspan="4">SWITCH</td></tr>
</table>

Word 1 describes buffer status:

LEFT -- number of words in the buffer yet to be obtained for input or supplied for output

NEXT -- relative location of the next word in the buffer to be obtained for input or supplied for output

Word 2 duplicates the first word of a record on tape:

LENGTH* -- number of words in the buffer, and number of words in a record (data words +2)

FILE NO. -- number of file at which currently positioned

LAST -- non-zero only for the last record of a file and then gives number of data words in the record

REC NO. -- number of the record at which currently positioned

*If not specified through use of CONTRL(*167), buffer length is 400_8 for devices 0-3, 1000_8 for devices 4-7.

Word 3 contains control information:

FILES -- number of files currently on the tape

UNIT -- logical tape unit number to which the device is attached, if at all

ERROR -- codeword address for user's error routine -- not used now

SWITCH -- set of binary switches to describe status and prescribe user options -- see below for details

The twelve bits of SWITCH are shown in detail:

		KPBF	WRIT
	RDCK	LNFT	READ
WRPT	KPCK		ATCH

word 3 of device status block

User option switches -- all off initially

WRPT, write protect -- if on, do not permit output to this unit

RDCK, read check -- if off, check output by read without store
if on, check output by word-for-word comparison in check buffer

KPCK, keep check buffer -- if on, do not free check buffer when output file is closed

KPBF, keep data buffer -- if on, do not free buffer for device when file is closed

LNFT, length from tape -- if off, use LENGTH in status block as buffer length for input
if on, get buffer length from tape on input

System status switches -- may not be set by user

WRIT, write -- if on, output file open on device

READ, read -- if on, input file open on device

ATCH, attach -- if on, device is attached to unit

- <u>Buffers</u>

The buffer matrix, BUFFER(*164), contains one device buffer per row, row i for device i, i=0,1,...,7. Row -1 is used as a check buffer for all devices requiring one. BUFFER is modified by B3=-1,0,...,7 for rows and B5=1,... for columns.

A device buffer is created, if necessary, when a file is opened on a device. It is freed when a file is closed, unless the device status block specifies to keep the buffer.

The check buffer is created or lengthened, if necessary, when an output file is opened on a device for which read checking is specified in the device status block. The check buffer is freed when an output file is closed on a device for which the device status block specifies not to keep the check buffer <u>and</u> no other device is using it.

A device buffer is just a copy of a record on tape. The first two words are for control; the first is the same as word 2 of the device status block, the second the symmetric difference checksum. The remaining words contain user data.

DEVICE BUFFER

word 1	LENGTH	FILE NO.	LAST	REC NO.
word 2	CHECKSUM			
	data			

- Errors

SBSS errors occur when conditions arise due to physical tape problems or logical programming errors from which the system cannot recover. An error message is printed giving the error number, device, and information from the device status block including the unit and current tape position. Then a halt occurs with the error number and 'TAPE' in U.

Error	Program	Condition
01	162,READ	record sequence numbers out of order - probably missed a marker
02	162,READ	tape cannot be read - tried to read record 8 times
03	163,WRITE	trying to write without output file open - can occur only on direct access level
04	163,WRITE	WMTE light came on while writing, signifying UME in memory
05	163,WRITE	tape cannot be written - tried writing twice in each of 8 spots
06	166,ARRAY	end-of-file encountered while reading an array
07	167,CONTRL	cannot open input file because positioned at end of information
10	167,CONTRL	cannot open output file because maximum number (252) exist already
11	167,CONTRL	cannot open output file because device is write-protected
12	167,CONTRL	cannot attach device to non-existent unit
13	167,CONTRL	insufficient space in memory for buffer