LINKING EDITOR FOR RELOCATABLE MODULES
LINK
User's Guide

Version 2

May, 1977

Model Code No. 50167

DATAPOINT

LINKING EDITOR FOR RELOCATABLE MODULES
LINK

User's Guide

Version 2

May, 1977

Model Code No. 50167

NOTICE

Datapoint strongly recommends that its customers use Datapoint Customer supplies. These disks, diskettes, cassettes and ribbons are certified by Datapoint to meet all Datapoint hardware specifications for consistent optimum performance.

PREFACE

LINK is a program for producing absolute 'core-image' program segments from one or more relocatable program modules. Input to LINK is one or more libraries of one or more relocatable program modules each. LINK combines selected libraries, or specific modules of libraries, into absolute program segments, which are in a form acceptable to the DOS loader. Absolute program segments may be extremely simple or arbitrarily complex in structure.

LINK requires at least a 2200 DOS (16K) system for execution. Any addition memory will be used as table space.

LINK is released on a DMF cassette, and the MIN utility (see the chapter on MIN in the DOS User's Guide) is necessary for installation. Additionally, the system SORT utility is necessary if cross-reference listings are to be produced.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION AND DEFINITIONS

LINK is a program for the manipulation and transformation of relocatable programs into absolute programs. Each relocatable program which is input to LINK is called a "relocatable program module", and each absolute program which is output from LINK is called an "absolute program segment". One or more relocatable program modules are contained in a relocatable program "library". Relocatable program libraries may contain an arbitrary number of relocatable program modules. Libraries are manipulated by means of a companion program, LIB.

LINK error messages are listed in Appendix A. The format of relocatable program libraries is discussed in Appendix B. Example listings are discussed in Appendix C.

Each relocatable program module consists of separate blocks of relocatable code, called "program address blocks" (PABs). Each PAB has an eight character name. When a program segment is created, all relocatable code relative to a particular PAB is bound to consecutive locations irrespective of program modules. That is, if program module ALPHA (which defines two PABs named CAT and DOG) is LINKed together with program module BETA (which defines two PABs CAT and RAT) and GAMMA (which defines two PABs CAT and MOUSE), the program segment would have a total of 4 PABs assigned: CAT, DOG, RAT, and MOUSE. All code that was relative to PAB CAT would be contiguous, in consecutive memory locations, followed by code relative to the other PABs. If each PAB that contains a different type of code, e.g., volatile data, pure re-entrant code, self-modifying code, I/O control information and buffers, etc., has the same name throughout the system to be LINKed, each of the different types of code will be bound into a conjoined address space.

Each PAB may have one or more attributes associated with it. These attributes concern the organization of that PAB's memory space. The attributes are: 1) Page sensitive orientation, meaning that the PAB must reside in a physical memory page -- i.e., does not cross a page boundary (Page sensitive orientation is on a per relocatable program module basis, rather than a per absolute program segment basis. That is, a PAB may be page sensitive in several modules, and yet have a total length exceeding 255 bytes. However, in each module where it is page sensitive, it will not cross a page boundary.); 2) Top of page orientation, meaning that the PAB must begin at the top of a

physical memory page -- a 0 MOD 256 address (Top of page orientation too is on a per relocatable module basis rather than a per absolute program segment basis.); 3) COMMON PAB orientation, meaning that PAB lengths are not summed together throughout the modules, but rather that the length of the longest PAB of that name in the modules is used as the length for that PAB.

If a PAB has either page sensitive or top of page orientation in any one relocatable module, the initial starting address of that PAB is a 0 mod 256 memory location.

If a PAB having COMMON PAB orientation is not declared COMMON by use of the COMMON command (Section 3.5), it must be declared COMMON in every program module which references it.

PAB attributes are not mutually exclusive -- a PAB may have both page sensitive orientation and top of page orientation; or may have COMMON PAB orientation and page sensitive orientation.

A special kind of PAB is the PAB which has a fixed starting address assigned to it. This PAB is obviously not relocatable, since it has a fixed starting address. It is an "absolute" PAB. A PAB may not be both relocatable in one program module and absolute in another. A PAB must be absolute in only one program module, or consistently relocatable in all modules that reference it.

Information is passed between relocatable modules by means of "external symbols". External symbols are from one to eight characters in length, and are composed of the twenty-six alphabetic characters, the ten numeric characters, and the "$". External symbols must begin with an alphabetic character. An "external definition" is a module's way of informing other modules that a specific symbol is defined in that module. An external definition may be an entry point, or may be a module data address, or a subroutine, or may simply be a value which is to be made available to other modules. An "external reference" is a module's way of requesting another module to define a symbol. An external reference is an external symbol referenced in a relocatable module which is not defined in that module. The basic process of linking relocatable modules together is a "resolution" of external references. Resolution of an external reference is the association of an external reference with that reference's external definition. External references may be resolved within a single program library. Or, the resolution of an external reference may require one or more passes through those program libraries which are to be searched as libraries. The system relocatable library, RELOCLIB/SYS, is normally searched to resolve external references, although this search may be suppressed.

Other libraries may be made available in the resolution of external references, as indicated in Chapter 3. If the system library is searched, it is the last library searched.

CHAPTER 2. THE SIMPLEST CASE

An example of the simplest kind of LINK is:

```
LINK [filename</REL>][,<filename></ABS][,<filename></PRT>]<;options>
```

where information contained within matching "<>" pairs is the default value (except for the "options", which are discussed in Chapter 4). The command line consists of three filenames: the (optional) source relocatable library, the (optional) object segment name, and (optional) printfile name. The default conditions for these filenames are the following:

1) the extension of the library to be LINKed is /REL;

2) the filename of the segment produced is the same as the filename of the relocatable library;

3) the extension of the segment produced is /ABS;

4) the filename of the printfile produced is the same as the filename of the relocatable library;

5) the extension of the printfile produced is /PRT.

Of course, all of these assumptions may be overridden on the command line.

(A "printfile" is a printer image file that is directed to disk rather than directed to a lineprinter. The LINK printfile uses column 1 of the printer line image as a special carriage control column. See the chapter on LIST in the DOS version 2 User's Guide for more information on printfiles.)

If there are any external references in a library that are not defined in that library, the system relocatable library, RELOCLIB/SYS, is normally searched to resolve those external references, although an option is provided to suppress this automatic library search.

CHAPTER 3. THE MORE COMPLEX CASE

Complex LINKs are defined as those which are not the simplest case. Included in this class are:

1) segmentation of a program into more than one segment -- a main segment and one or more overlay segments, for instance;

2) presenting for the library search libraries in addition to, or instead of, the system library;

3) inclusion of program modules from more than one library as part of a segment;

4) definition and replacement of external symbols at LINK time;

5) location assignment and ordering of PAB names;

6) specifying an entry point for a segment;

7) changing the contents of absolute memory locations after all other LINKing has been performed.

Complex LINKs are accomplished by presenting to LINK one or more of the commands described in this chapter. These commands may be presented to LINK in either of two ways: through the keyboard, or through a command file. The presentation of commands through the keyboard is indicated by the absence of the first file name on the LINK command line. The presentation of commands by a command file is indicated by an option discussed in Chapter 4.

Commands presented to LINK are verified to see that they are syntactically correct. After verificatian they are placed in an internal command file, SCRATCH/TXT. During code generation, commands are retrieved and processed from this internal command file.

Comment lines (defined as a command line which has a decimal point "." or a slash "/" in column 1) may be placed at any point in an external command file or keyin commands (except after the terminating asterisk "*") and will be contained as part of the internal LINK command file.

In the discussions below of the individual statement types, a statement prototype is listed. In the prototype, fields set off

by pointed brackets "<>" are required fields; fields set off by square brackets "[]" are optional fields.

Several of the commands discussed below accept an expression as one of their parameters. In the discussion below, expressions which should not contain any relocatable symbols are preceded with an ampersand "&". An expression is evaluated strictly from left to right (no parentheses are allowed), and consists of the following items:

+-*/	Standard arithmetic operators, including unary minus. Note: all arithmetic is performed on 16 bit integers. No check is made for overflow or underflow. Attempt to divide by zero is checked.
<>	Left and right shift.
<name>	Any external symbol. Note: the 16 bit value for a relocatable external symbol is not its absolute value until late in the LINK.
Onnnnn	Octal constant, consisting of the digits from 0-7. Octal constants always begin with a "0".
nnnnnn	Integer decimal constant, consisting of the digits from 0-9. Decimal constants do <u>not</u> begin with a "0".

Literal character strings are not permitted in expressions.

Examples of expressions:

```
-77
START+01000
ADDRESS>8
CATS*3+DOGS*2/5
```

3.1 The SEGMENT Command

The SEGMENT command is the most significant LINK command. That is, other LINK commands occurring between two SEGMENT commands are associated with the first SEGMENT command. The first command presented to LINK must be a SEGMENT command. The SEGMENT command may have up to four parameters, separated from one another by commas. In most instances, any parameter may be null. The prototype of the segment command is:

```
SEGMENT <absfile>,[&startexp],[highadr],[lexfile]
```

Fields are positional; e.g., if "lexfile" is supplied but "startadr" and "highadr" are not, "lexfile" must be separated from "absfile" by three commas.

The first parameter on the SEGMENT command is the name of the disk file (in standard FILENAME/EXT:DRn format) into which the absolute code for this program segment is to be placed. Any names for relocatable and absolute files which may have appeared on the LINK command line are ignored. An object segment filename must be presented on the first SEGMENT command. If the object segment filename has no extension, it is assumed to be /ABS. An object segment filename may be presented on other SEGMENT commands, although if none is presented, the filename appearing on the first SEGMENT command is substituted.

The second parameter on the SEGMENT command contains an expression which specifies a starting loading address for that segment. This expression should contain no relocatable symbols. If this field is null on the first SEGMENT command, a starting load address of 010000 is assumed. If this field is null on all other SEGMENT commands, the starting load address is assumed to be the next available location after the first segment.

The third parameter on the SEGMENT command contains an external symbol which will have as a value the next available absolute location at which code could be generated. This field may be null. This symbol and its value are useful only if a /LEX file (see below) is requested. The symbol and its value are then placed in the /LEX file, and may be referenced by using the created /LEX file as a library in subsequent LINKs.

The fourth parameter on the SEGMENT command is a filename into which LINK is to write the resolved external definitions of the current segment. This file is created as a relocatable library file in a format readable by LINK, and may be used by other segments or other LINKs as a library reference. The default extension of this file is /LEX. This file is generated only if a name appears in the fourth field.

The /LEX file contains only one module (whose name is the same as the first parameter of the SEGMENT command) which consists of only a table of external definitions for the current segment -- no relocatable code or external references are included in the /LEX file.

Examples of the SEGMENT command:

```
SEGMENT ASM/CMD,,MAINEND,ASM
```

```
SEGMENT /OV1,,OV1END
SEGMENT /OV2,OV1END
SEGMENT /OV3
```

Note that the last SEGMENT command has the same meaning as:

```
SEGMENT /OV3,MAINEND
```

(assuming that MAINEND is a larger value than the ending location of segment ASM/OV2). Also, segment ASM/OV2 must include OV1/LEX as a library reference to resolve OV1END.

3.2 The INCLUDE Command

The prototype of the INCLUDE command is:

```
INCLUDE [-]<libname[.progname]>
```

If a "-" is present, a ".progname" is required. The INCLUDE command has three uses: first, to include a specific program module from a library in an absolute program segment; second, to include all program modules from a library in an absolute program segment; and third, to exclude specific modules from an entire library that has been included.

All modules within a library are included by giving the library name. A specific module within a library is included by giving the library name with extension (the default extension is /REL) and specific drive (unless all drives are to be searched) followed by a "." followed by the module name. A specific module from a library is excluded by placing a "-" before the library name.

Examples of INCLUDE:

```
INCLUDE PAYROLL:DR2
INCLUDE -PAYROLL/REL:DR2.MAIN
INCLUDE NEWPAY.MAIN
```

More than one name may be indicated on an INCLUDE command line; the above could also have been indicated by:

```
INCLUDE PAYROLL:DR2,-PAYROLL/REL:DR2.MAIN,NEWPAY.MAIN
```

If a program module is explicitly included and also excluded, a message will be printed and the module will not be included.

3.3 The LIBRARY Command

The prototype of the LIBRARY command is:

```
LIBRARY [-]<libname[.progname]>
```

If a "-" is present, a ".progname" is required. The LIBRARY command is used to indicate that either a module or an entire library is to be searched to resolve external references. A module of a relocatable library that is referenced by the LIBRARY command is included in a program segment only if there is a reference to an external definition of that library module. If a specific module of a library is to be excluded from a library search, that library name and program module reference is preceeded by a "-". If a program module is both explicitly included in library search and excluded from library search, a message will be printed and the module will not be included in the search. An option on the command line is provided to exclude the system library, RELOCLIB/SYS (see Chapter 4.). The format of the LIBRARY command is identical to the INCLUDE command, including the ability to specify more than one library on the LIBRARY command.

Examples of the LIBRARY command:

```
LIBRARY SUB
LIBRARY IO/REL:DR2
LIBRARY -IO/REL:DR2.KEYIN$
LIBRARY NEW.KEYIN$
```

3.4 The ORIGIN Command

The prototype of the ORIGIN command is:

```
ORIGIN <pabname>[,&startexp][,highadr]
```

The ORIGIN command performs one of two functions. The first function is to assign an ordering to relocatable PAB names in the current program segment. (Ordering in this context means assigning a starting memory location to a PAB. Obviously, absolute PABs use as a starting address the value assigned to them for a starting address. If no starting address has been specified on the SEGMENT statement, the default starting address for the relocatable PAB names in a segment is 010000. If any absolute PABs have been LINKed into a program segment, the starting address of the relocatable PABs is the next available location after the greatest absolute PAB, or the starting address, whichever is greater.) If no ordering has been assigned using ORIGIN

statements, PAB names will be ordered alphabetically based upon the first character of the PAB name. It is this alphabetical ordering which can be overridden by use of the ORIGIN command. All PAB names appearing on ORIGIN statements are ordered first, in the order given, and the remaining are ordered alphabetically based upon the first character of the PAB name.

The second function of the ORIGIN statement is to assign a specific starting memory address to a PAB name. A starting address expression may follow the PAB name on an ORIGIN command, separated from the name by a comma. The starting address expression should contain no relocatable symbols.

The ORIGIN command may also perform a third function. If the third parameter on the ORIGIN command is not null, it should contain an external symbol to which LINK will assign a value of the next available absolute location at which code could be generated.

Examples of the ORIGIN command:

```
ORIGIN PABONE
ORIGIN PABTWO,STARTEXP+01000
ORIGIN PAB3,START,END
```

3.5 The COMMON Command

The prototype of the COMMON command is:

```
COMMON <pabname>
```

The COMMON command is used to indicate that each specific PAB name is to be treated as a common PAB, in the FORTRAN sense of COMMON blocks. That is, the code in PABs defined as COMMON is not appended sequentially throughout all relocatable program modules as is the code in other PABs; rather, the length of the longest PAB of that name is used as the length of that PAB.

If more than one PAB name appears on a COMMON comand line, each of those PAB names are to be considered separate COMMON blocks, rather than as one COMMON block.

Examples of the COMMON command:

```
COMMON DATA
COMMON BLANCOMM,COMMON,DATA1
```

3.6 The ENTRY Command

The prototype of the ENTRY command is:

```
ENTRY <startexp>
```

The ENTRY command is used to specify a specific transfer address expression as the starting execution address of an absolute program segment. All transfer addresses indicated in any of the relocatable program modules are ignored, and the expression specified is used as the primary transfer address. The transfer address expression may contain relocatable symbols. Only the first ENTRY command is valid in any segment; other ENTRY commands are treated as errors. If no module in a program segment had a transfer address defined, and no ENTRY command has been specified, the transfer address is the starting address of the first PAB assigned.

Example of the ENTRY Command:

```
ENTRY START+3
```

3.7 The DEFINE Command

The prototype of the DEFINE command is:

```
DEFINE <symbol>=<&expression>
```

Only one definition may be specified on the DEFINE command line. The DEFINE command is used to define a value for an external symbol at LINK time. The defined expression should contain no relocatable symbols.

Example of the DEFINE command:

```
DEFINE PRINTADR=0303
```

3.8 The IGNORE Command

The prototype of the IGNORE command is:

```
IGNORE <symbol>
```

The IGNORE command is used to completely ignore one or more external symbols in the module to be LINKed. All references to IGNOREd symbols are treated as if they were non-existent (the value associated with IGNOREd symbols is zero). The IGNORE command may be used to prevent certain external references from attaching the modules in which they are defined to the segment being LINKed, without indicating that the references are undefined. The IGNORE command may also be used to indicate that the references are nowhere defined, and to ignore the error message which would be generated because of the otherwise undefined symbol.

Examples of the IGNORE command:

```
IGNORE UNDEFINE,NOTYET,ERROR
IGNORE DSPLY$,PAYROLL
```

3.9 The REPLACE Command

The prototype of the REPLACE command is:

```
REPLACE <symbol1>=<symbol2>
```

The REPLACE command is used to replace a specific external reference with a different external definition. All occurrences of <symbol1> are replaced by <symbol2>. Expressions are not allowed in the REPLACE command. Only one REPLACEd name may appear on a command line.

Examples of the REPLACE command:

```
REPLACE DSPLY$=USERDISP
REPLACE KEYIN$=USERKEY
```

3.10 The MEMORY Command

The prototype of the MEMORY command is:

```
MEMORY <exp>=<literal>
```

The MEMORY command is used to specifically define the contents of an absolute memory location (after all other LINKing is performed). No matter what the value of the location after normal LINKing, the MEMORY command has precedence. The address expression of the MEMORY command may contain relocatable symbols; the value(s) to be placed at that address (and subsequent addresses) may be any expression (only the low order eight bits of the expression's value are used to determine the byte's value), or multi-byte character strings. Character strings may make use of the "#" as a forcing character. Only one starting memory address is allowed on each MEMORY command.

Examples of the MEMORY command:

```
MEMORY DATE=061,061,040,0101,0125,040,067,065
MEMORY DATE='11'
MEMORY DATE+3='AU'
MEMORY DATE+6='75'
MEMORY STRING='USE OF FORCING CHARACTER','###''
```

3.11 The TITLE Command

The prototype of the TITLE command is:

```
TITLE <character string>
```

The TITLE command is used to reset the heading if a listing is generated. If a listing is not being generated the TITLE commands will be ignored. The TITLE command should follow the SEGMENT command for which it sets the title.

Example of the TITLE command:

```
TITLE THIS IS THE NEW HEADING
```

3.12 The KEYIN Command

The prototype of the KEYIN command is:

```
KEYIN [character string]
```

The KEYIN command is used to request one LINK command from the keyoard or chainfile. Its use will primarily be within a LINK command file (see section 4.1). The character string may be used to prompt the operator for the desired command.

Examples of the KEYIN command:

```
KEYIN
KEYIN  ENTER "MEMORY DATE='dddyy'"
```

CHAPTER 4. LINK CONTROLS

4.1 LINK Options

Options to do all of the following are accepted by LINK:

1) Read LINK commands from a disk file rather than from the keyboard;

2) List the loader tables on the screen or servo or local printer, or place this listing in a disk printfile;

3) Suppress generation of the object program if LINK errors are found;

4) Load and run the first segment of the LINKed program immediately after it has been LINKed;

5) Indicate that the system library RELOCLIB/SYS is not to be searched to resolve external references;

6) produce a sorted cross-reference listing of external symbols, their absolute value, and program modules which reference and define them;

7) produce an alphabetical listing of either all or only referenced entries in the external symbol table.

4.1.1 Command File Input - "F"

Complex LINKs were earlier defined as those which make use of the LINK commands. These commands may either be entered through the keyboard, or pre-defined in a file in the order in which they are to be presented to LINK. If the first file specification is missing, LINK assumes that commands are to be presented through the keyboard. If the first file specification is present, and the "F" option (for command file input) is indicated, then the first file specification is assumed to be the name of a command file which contains the commands to be used in the LINK. The assumed extension of this file is /TXC. Each logical record of this file contains a LINK command (see Chapter 3). In either of these complex LINK options, the input to LINK is terminated by a "*".

4.1.2 Listing The Loader Tables - "L","D"

The "L" option indicates that the loader tables listing is to be generated and either printed or stored on disk. The "D" option indicates that the loader table listing is to be generated and displayed on the screen.

Error messages are displayed on the screen in any event.

Pressing the DISPLAY key while a message is being displayed will temporarily suspend the LINK processing until the DISPLAY key is released.

4.1.3 Selection of a Print Medium - "P","Q","W"

When the "L" option (see Section 4.2) is specified the default print medium is a local printer. The user can specify an alternate printer medium by using the "P", "Q", or "W" options. The "P" option indicates that a disk print image file is to be produced instead of a printer listing. The third filename on the command line is the filename of the printfile produced. The "Q" option indicates that the LINK printfile is to be queued after any data in the indicated printfile. The "W" option indicates that the listing should be produced on a servo printer. If the options "P", "Q", or "W" are used, the "L" is not necessary. The "W" option may not be used in conjunction with "P" or "Q".

4.1.4 Error Termination - "E"

The "E" option indicates that the object file is not to be opened, and the LINK is to be terminated if any errors were discovered prior to attempting to open the object file.

4.1.5 Load And Run - "R"

The "R" option indicates that if the LINK had no errors, the first segment of the LINKed program is to be loaded into memory and executed immediately after the LINK is terminated.

4.1.6 Suppressing The Automatic Library Search - "N"

The "N" option indicates that the system library, RELOCLIB/SYS, is not to be searched to resolve external references. If a LIBRARY command with RELOCLIB/SYS indicated as a library is given to LINK, this option will be overridden in the segment which contains the LIBRARY command.

4.1.7 Generation of Cross-reference Listing - "X"

The "X" option indicates that a sorted cross-reference listing of all entries in the external symbol table is to be produced, one listing for each segment. LINK creates a temporary file of the external symbol, its value, and program names that reference and define the symbol. LINK then rolls out to the system SORT utility to sort the file, rolls itself back in to print the file, and produces the cross-reference listing.

4.1.8 Printing the symbol table - "S","T"

The "S" option sorts alphabetically and then prints the entries in the external symbol table, printing each symbol's name, value, and flags associated with it ("D" for defined, "I" for ignored, "P" for replaced, "R" for referenced). The "T" option indicates that the symbol table is to be printed in a tabular form, four entries per print line. Only one of these options may be used and they can only be used when "X" is not used. Also, printed output must be specified, using "L", "D", "P", "Q", or "W".

4.1.9 Abbreviating Cross-reference or symbol table listing - "A"

The "A" option requests an abbreviated symbol table or cross-reference listing, omitting all symbols that are defined but not referenced. It may be used in conjunction with the "X", "S", or "T" options.

4.1.10 Page Length

The maximum number of lines to be produced per page may be changed by including a 2 digit decimal number in the options string. The default value is defined as 57.

4.2 DISPLAY Key

The DISPLAY Key may be depressed at anytime to cause LINK to pause after displaying or printing the next line of data. Normal processing will resume when the DISPLAY key is released.

4.3 KEYBOARD Key

The KEYBOARD key may be depressed at anytime to cause LINK to terminate after displaying or printing the next line of data. Termination of a chain file will also result if one is running.

APPENDIX A. ERROR MESSAGES

The following is a list of error messages that will currently be generated by LINK Version 2. An F by the side of the message indicates that the execution of LINK will terminate. All others will cause the DOS ABTIF flag to be set.

```
  ABOVE PAB EXCEEDS A PAGE
  ABOVE PAB EXCEEDS TABLE OVERFLOW
  BAD DRIVE NUMBER
  BLANK REQUIRED
  CANNOT DELETE A LIBRARY
  CANNOT DUPLICATE ENTRY COMMAND
  COMMAND FILE NOT FOUND
  CONFLICT BETWEEN EXCLUDE AND INCLUDE
F DIVIDE BY ZERO ATTEMPTED
F ERRORS DISCOVERED -- NO CODE GENERATED
F EXCLUSION TABLE OVERFLOW
  EXTERNAL TABLE OVERFLOW
  EXTENSION REQUIRED
F FIRST COMMAND MUST BE SEGMENT COMMAND
  FIRST SEGMENT MUST BE NAMED
F FORMAT TRAP IN INPUT FILE
  ILLEGAL CHARACTER IN COMMAND FILE
  ILLEGAL COMMAND
  ILLEGAL CONSTANT
  ILLEGAL EXPRESSION
F ILLEGAL INSTRUCTION: ...
        An illegal relocatable op code was discovered.
  ILLEGAL NAME
  ILLEGAL OPERATOR
  ILLEGAL OPTION
  ILLEGAL RELOCATABLE INPUT AT LRN ... OF PFN ... ON
        DRIVE ...
  ILLEGAL SEPARATOR
F LIBRARY HAS IMPROPER ID STRUCTURE
  LIBRARY IS MISSING ID RECORD
  LIBRARY IS MISSING PROGRAM ID RECORD
  LIBRARY NOT FOUND
F LINK OVERLAY n IS MISSING
F LINK OVERLAY n IS UNLOADABLE
  MISSING EQUAL SIGN
F MUST GET A LISTING TO GET SYMBOL TABLES
```

```
  NO SEGMENT COMMAND FOUND
F NOT ENOUGH ROOM FOR TABLES IED
F ONLY ONE SYMBOL TABLE OPTION ALLOWED
F PAB NAME TABLE OVERFLOW
F PAB NAME ... INTERSECTS PAB NAME ...
  PROGRAM IS NOT IN LIBRARY
F PROGRAM TABLE OVERFLOW
  QUOTATION MARK REQUIRED
F RANGE TRAP IN INPUT FILE
F REFERENCE TABLE OVERFLOW
  SECONDARY STARTING ADDRESS FOR PAB NAME ... IGNORED
  SECONDARY TRANSFER ADDRESS IGNORED.
  SORT/CMD NOT LOADABLE
  TRANSFER ADDRESS OF nnnnnnn OVERRIDDEN
  UNDEFINED EXTERNAL REFERENCED
  UNDEFINED SYMBOLS ARE:
       A PAB name has more than one starting address.
  ... HAS A DUPLICATE DEFINITION
       NOTE:  If an external symbol is defined to the
       same absolute value in one or more relocatable
       modules, it does not have a duplicate definition.
```

APPENDIX B. RELOCATABLE CODE FORMATS

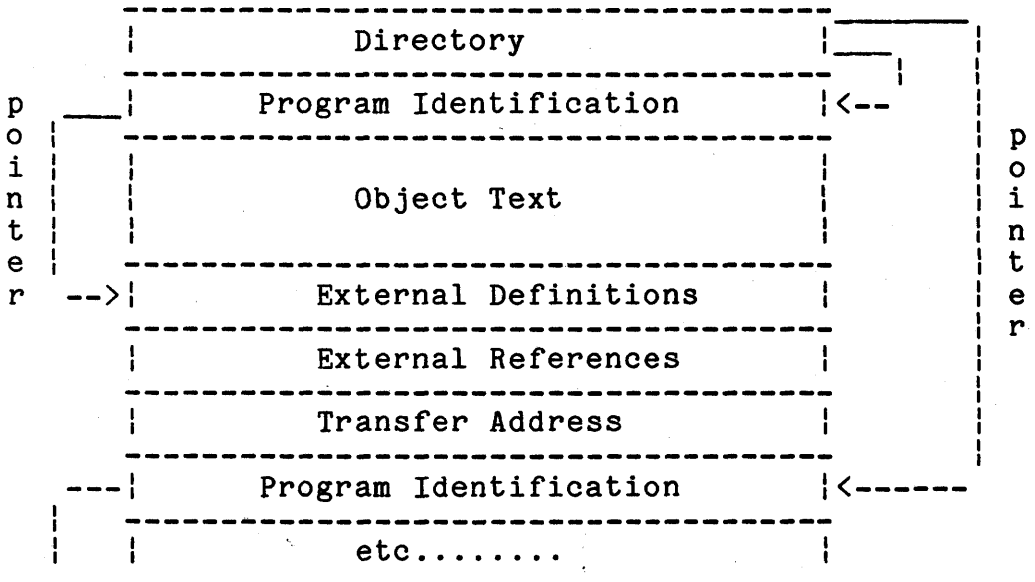
Relocatable object code is initially assumed to be starting at location 010000 until a "select new PAB" or "select new location" code is encountered.

Each sector containing relocatable code starts with a one byte header containing sector contents code. The relocatable code in each sector is followed by a byte containing binary zero.

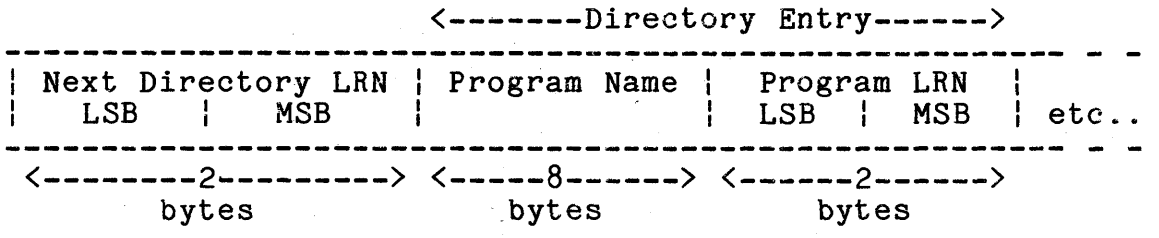
Sector contents codes are:

0200	Directory
0201	Program Identification
0202	Object Text
0203	External Definitions
0204	External References
0205	Transfer Address

Relocatable code files are in library form as follows:

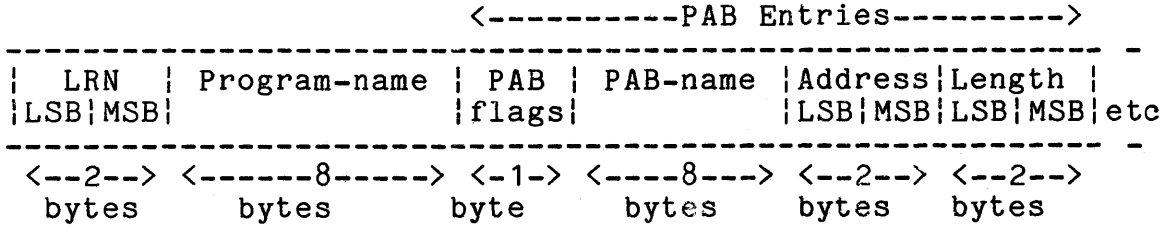


B.1 Directory



A directory entry is required for each object program in a library. The first sector of the object code library is reserved as a directory for the first twenty-four programs in the library. If the library contains more than twenty-four programs, a pointer is generated that points to the LRN of the next directory sector (the sector following the twenty-fourth object program). The last directory sector used has a pointer set to 0377, 0377.

B.2 Program Identification

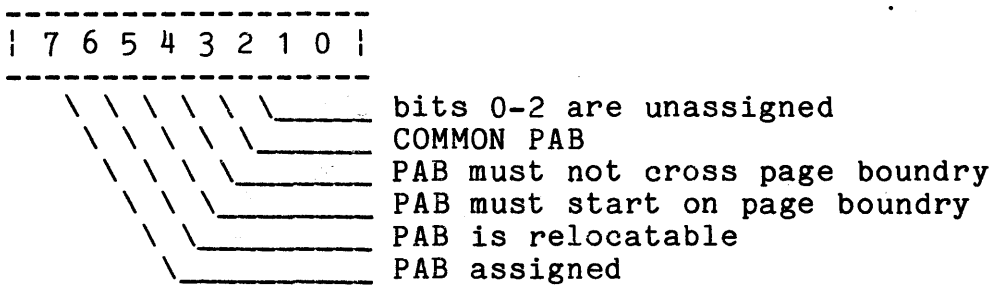


LRN is a pointer to the first sector following object text (the first external definition sector, or the first external reference sector, or the transfer address if there are no definitions or references).

The program name is an eight character name of the program, as reflected in the program id record.

Each PAB (program address block) defines a separate address counter used to assign memory locations. Up to fifteen PAB's can be defined for each program (PAB numbers 1-15). Flag bits are used to indicate relocatability and page sensitivity.

PAB flags:



B.3 Object Text

Relocatable object text is interspersed with control bytes used by the linkage editor in creating absolute code.

B.3.1 Memory Location

Codes 0160 and 0161 are used to define starting memory locations.

Select New PAB

0160	PAB

PAB defines the number of the Program Address Block to be used for the object code that follows. If the PAB is not in use, the new location will be zero.

Select New Location

0161	LSB	MSB

LSB and MSB define the new location in the current PAB of the next byte of object code.

B.3.2 Absolute Text

Codes 0001-0077 precede code and data that does not require relocation.

Absolute Text

1-0077	1-63 absolute text bytes

The code is a count of the number of absolute text bytes that follow.

B.3.3 Complex Relocatable References

Codes 0100-0157 are used to define operators and operands of complex expressions that are evaluated by the linkage editor during relocation. Complex expressions are in encoded Polish Postfix notation.

Push Relocatable Location on Logical Stack

0100+PAB	LSB	MSB

PAB, LSB and MSB define the assembled memory location.

Push External Reference on Logical Stack

0120+MSB	LSB

MSB and LSB are an index to an external reference entry (See section B.5).

Push Binary Value on Logical Stack

0140	LSB	MSB

LSB and MSB are a 16 bit binary integer.

Operators:

<	>	.OR.	.XOR.
0141	0142	0143	0144

.AND.	+	-	*
0145	0146	0147	0150

/	Negate	.MOD.
0151	0152	0153

Codes 0141-0153 are expression operators.

Pop Result of Evaluation from Logical Stack:

Pop LSB	Pop MSB	Pop LSB-MSB	Pop MSB-LSB
0154	0155	0156	0157

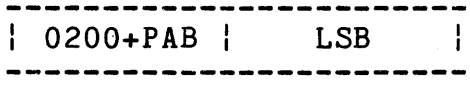
Codes 0154-0157 terminate evaluation of complex expressions and indicate the form of the absolute code to be generated.

B.3.4 Simple Relocatable References

Codes 0200-0377 are used for simple relocatable references consisting of a single relocatable symbol or relocatable symbol plus a non-relocatable displacement. Codes for simple relocation can be decoded as follows:

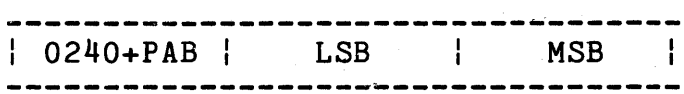
```
-------------------
| 7 6 5 4 3 2 1 0 |
-------------------
   \ \ \ \ \_ bits 0-3 are part of relocation definition
    \ \ \ \__ external reference
     \ \ \___ inverted address (MSB-LSB)
      \ \____ 16 bit address
       \_____ simple relocatable memory reference
```

LSB Reference



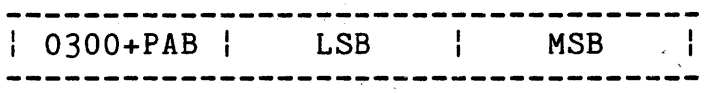
LSB defines the relocatable memory location.

MSB Reference



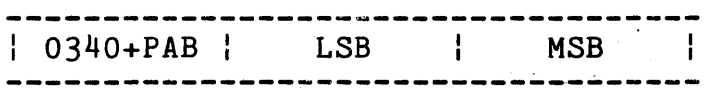
PAB, LSB and MSB define the relocatable memory location. A full sixteen bit address must be given in case a carry occurs between LSB and MSB during relocation.

LSB-MSB Reference



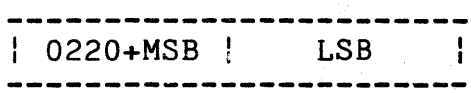
PAB, LSB and MSB define the relocatable memory location.

MSB-LSB Reference



PAB, MSB and LSB define the relocatable memory location.

LSB External Reference



MSB and LSB are an index to an external/forward reference entry table (See section B.5).

MSB External Reference

0260+MSB	LSB

MSB and LSB are an index to an external/forward reference entry table (See section B.5).

LSB-MSB External Reference

0320+MSB	LSB

MSB and LSB are an index to an external/forward reference entry table (See section B.5).

MSB-LSB External Reference

0360+MSB	LSB

MSB and LSB are an index to an external/forward reference entry table (See section B.5).

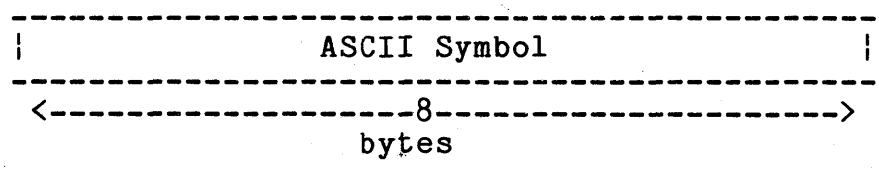
B.4 External Definitions

External name	PAB or 0200	LSB	MSB
<-----8-----> bytes	<---1---> byte	<-1-> byte	<-1-> byte

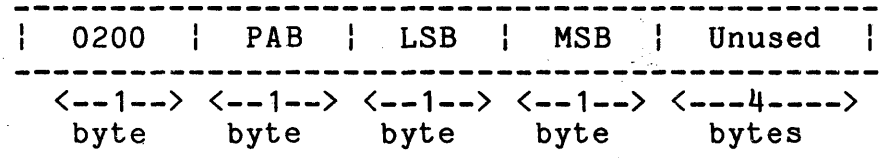
External definitions are external symbols made available to other relocatable modules. External references made by other relocatable modules are linked to external definitions as discussed in Chapter 1. The location of each relocatable external definition is defined by PAB, LSB and MSB. A flag (0200), LSB and MSB define non-relocatable external definition values. Up to twenty-two external definitions can be defined in each external definition sector. All external definition sectors for a given program must be contiguous, and not intermixed with external reference sectors.

B.5 External and Forward References (4096 maximum)

External Reference



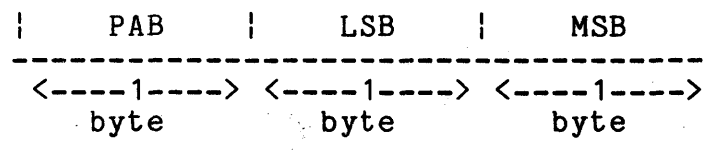
Forward reference



A forward reference is defined as a reference whose value is unknown at some given time in the relocatable module's creation, but whose value is known later, and then is plugged into the forward reference table.

All external reference/forward definition sectors must be contiguous.

B.6 Transfer Address



PAB, LSB and MSB define the starting location in the program. If PAB=0377, a starting location was not specified.

APPENDIX C. SAMPLE LISTINGS

The first page of a LINK listing indicates the number of symbols available and the options in effect for the current LINK. The first page also contains a listing of the commands for all segments in the current LINK - together with any errors which may have been discovered.

After this initial listing, the commands for each individual segment are again listed, starting on a new page. Following the command listing is a PAB utilization table, listing all PAB names in the segment, together with their starting and ending addresses, and their length. A line displaying the lowest/highest address for the segment is also printed. After the PAB utilization section is a table of all program names and their PAB's. This table lists starting addresses, lengths, and flags for all PAB's in each program module. After this table is a listing of all MEMORY commands (if present) and the transfer address.

The second part of each segment's listing is the optional symbol table print or cross reference listing. These tables also begin on a new page.

2275 EXTERNAL SYMBOLS AVAILABLE; OPTIONS IN EFFECT: L,P,S

```
COMMAND LISTING FOR THIS LINK:
SEGMENT SAMPLE01/ABS
INCLUDE SAMPLE01/REL
LIBRARY RELOCLIB/SYS
*
```

```
SEGMENT SAMPLE02/ABS,,MAINEND,SAMPLE02/LEX
INCLUDE SAMPLE02/REL
INCLUDE-SAMPLE02/REL.ALPHA
INCLUDE SAMPLE01/REL.ALPHA
LIBRARY SAMPLE03/REL
ORIGIN  LIBCODE$,020000
ORIGIN  RAT
ORIGIN  CAT
COMMON  MOUSE
ENTRY   START+3
DEFINE  PRINTADR=0303
IGNORE  SERVO
REPLACE START=ALPHA01
MEMORY  START=052
MEMORY  DATE='04 NOV 75'
LIBRARY RELOCLIB/SYS
```

PAB UTILIZATION IN PROGRAM SEGMENT SAMPLE02/ABS

PAB NAME	START	END	LENGTH
LIBCODE$	0020000	0020500	0000501
RAT	0020501	0020523	0000023
CAT	0021000	0021256	0000257
MOUSE	0021664	0022400	0000515
DOG	0021400	0021542	0000143
FISH	0021543	0021663	0000121

SEGMENT LOWEST/HIGHEST ADDRESS: 0020000/0022400

PROGRAM NAME	PAB NAME	START	LENGTH	FLAGS
DELTA	MOUSE	0021664	0000515	R,C
	RAT	0020501	0000023	R
ALPHA	CAT	0021000	0000257	R,P
	DOG	0021400	0000143	R,T
KAPPA	FISH	0021543	0000121	R
FILESCAN	LIBCODE$	0020000	0000372	R
INCPAIR	LIBCODE$	0020372	0000036	R
DECPAIR	LIBCODE$	0020430	0000036	R
INCSWP	LIBCODE$	0020466	0000013	R

SYMBOL TABLE PRINT:

ALPHA01	0010000	D,R
ALPHA02	0012400	D,R
BCFRHL	0013231	D,R
BETA01	0010257	D,R
BETA02	0013417	D,R
DECBC	0013173	D,R
DECDE	0013205	D
DECHL	0013217	D,R
DEDBC	0013175	D
DEDDE	0013207	D
DEDHL	0013221	D
DEFRHL	0013241	D
FILESCAN	0012546	D,R
GAMMA01	0012000	D,R
GAMMA02	0013417	D,R
HLFRHL	0013251	D
INCBC	0013161	D
INCDE	0013147	D,R
INCHL	0013135	D,R
INCSWP	0013260	D,R
INDBC	0013163	D
INDDE	0013151	D
INDHL	0013137	D
INDSWP	0013262	D

2275 EXTERNAL SYMBOLS AVAILABLE; OPTIONS IN EFFECT: D,F,L,P,X

COMMAND LISTING FOR THIS LINK:

```
SEGMENT SAMPLE02/ABS,,MAINEND,SAMPLE02/LEX
INCLUDE SAMPLE02/REL
INCLUDE-SAMPLE02/REL.ALPHA
INCLUDE SAMPLE01/REL.ALPHA
LIBRARY SAMPLE03/REL
ORIGIN  LIBCODE$,020000
ORIGIN  RAT
ORIGIN  CAT
COMMON  MOUSE
ENTRY   START+3
DEFINE  PRINTADR=0303
IGNORE  SERVO
REPLACE START=ALPHA01
MEMORY  START=052
MEMORY  DATE='04 NOV 75'
LIBRARY RELOCLIB/SYS
*
SEGMENT SAMPLE02/OV1,MAINEND+100,OV1END,SAMPLEOV/LEX
INCLUDE SAMPLE04/REL
LIBRARY SAMPLE02/LEX
LIBRARY RELOCLIB/SYS
*
SEGMENT SAMPLE02/OV2,OV1END
LIBRARY SAMPLEOV/LEX
INCLUDE SAMPLE05/REL
LIBRARY RELOCLIB/SYS
*
```

```
SEGMENT SAMPLE01/ABS
INCLUDE SAMPLE01/REL
LIBRARY RELOCLIB/SYS
```

PAB UTILIZATION IN PROGRAM SEGMENT SAMPLE01/ABS

PAB NAME	START	END	LENGTH
CAT	0010000	0012334	0002335
DOG	0012400	0012542	0000143
RAT	0013417	0013533	0000115
MOUSE	0013273	0013416	0000124
LIBCODE$	0012543	0013272	0000530

SEGMENT LOWEST/HIGHEST ADDRESS: 0010000/0013533

PROGRAM NAME	PAB NAME	START	LENGTH	FLAGS
ALPHA	CAT	0010000	0000257	R,P
	DOG	0012400	0000143	R,T
BETA	CAT	0010257	0001202	R
	RAT	0013417	0000115	R,P
GAMMA	CAT	0012000	0000335	R,P
	MOUSE	0013273	0000124	R,C
FILESCAN	LIBCODE$	0012543	0000372	R
INCPAIR	LIBCODE$	0013135	0000036	R
DECPAIR	LIBCODE$	0013173	0000036	R
LOADPAIR	LIBCODE$	0013231	0000027	R
INCSWP	LIBCODE$	0013260	0000013	R

TRANSFER ADDRESS FOR SEGMENT SAMPLE01/ABS:DRO IS 0013273

```
MEMORY LOCATION: 0021000 CHANGED TO: 0052
MEMORY LOCATION: 0020512 CHANGED TO: 0060
MEMORY LOCATION: 0020513 CHANGED TO: 0064
MEMORY LOCATION: 0020514 CHANGED TO: 0040
MEMORY LOCATION: 0020515 CHANGED TO: 0116
MEMORY LOCATION: 0020516 CHANGED TO: 0117
MEMORY LOCATION: 0020517 CHANGED TO: 0126
MEMORY LOCATION: 0020520 CHANGED TO: 0040
MEMORY LOCATION: 0020521 CHANGED TO: 0067
MEMORY LOCATION: 0020522 CHANGED TO: 0065

TRANSFER ADDRESS FOR SEGMENT SAMPLE02/ABS:DR0 IS 0021003
```

SORTED CROSS REFERENCE LISTING:

ALPHA01	0021000	*SAMPLE01/REL.ALPHA		
ALPHA02	0021400	*SAMPLE01/REL.ALPHA		
BETA03	0022401	*SAMPLE02/REL.DELTA		
DATE	0020512	*SAMPLE02/REL.DELTA		
DECBC	0020430	SAMPLE01/REL.ALPHA	SAMPLE03/REL.KAPPA	*RELOCLIB/SYS.DECPAIR
DECDE	0020442	*RELOCLIB/SYS.DECPAIR		
DECHL	0020454	RELOCLIB/SYS.FILESCAN	*RELOCLIB/SYS.DECPAIR	
DEDBC	0020432	*RELOCLIB/SYS.DECPAIR		
DEDDE	0020444	*RELOCLIB/SYS.DECPAIR		
DEDHL	0020456	*RELOCLIB/SYS.DECPAIR		
FILESCAN	0020003	SAMPLE01/REL.ALPHA	*RELOCLIB/SYS.FILESCAN	
INCBC	0020416	SAMPLE03/REL.KAPPA	*RELOCLIB/SYS.INCPAIR	
INCDE	0020404	SAMPLE01/REL.ALPHA	*RELOCLIB/SYS.INCPAIR	
INCHL	0020372	RELOCLIB/SYS.FILESCAN	*RELOCLIB/SYS.INCPAIR	
INCSWP	0020466	SAMPLE03/REL.KAPPA	*RELOCLIB/SYS.INCSWP	
INDBC	0020420	*RELOCLIB/SYS.INCPAIR		
INDDE	0020406	*RELOCLIB/SYS.INCPAIR		
INDHL	0020374	*RELOCLIB/SYS.INCPAIR		
INDSWP	0020470	*RELOCLIB/SYS.INCSWP		
LABEL	0021543	SAMPLE02/REL.DELTA	*SAMPLE03/REL.KAPPA	
MAINEND	0022401			
PRINTADR	0000303	SAMPLE02/REL.DELTA		
SERVO	*******	SAMPLE02/REL.DELTA		
START	0021000	*SAMPLE02/REL.DELTA		

```
SEGMENT SAMPLE02/OV1,MAINEND+100,OV1END,SAMPLEOV/LEX
INCLUDE SAMPLE04/REL
LIBRARY SAMPLE02/LEX
LIBRARY RELOCLIB/SYS
```

PAB UTILIZATION IN PROGRAM SEGMENT SAMPLE02/OV1

PAB NAME	START	END	LENGTH
SNAKE	0023144	0024152	0001007
FROG	0022545	0023143	0000377

SEGMENT LOWEST/HIGHEST ADDRESS: 0022545/0024152

PROGRAM NAME	PAB NAME	START	LENGTH	FLAGS
EPSILON	SNAKE	0023144	0001007	R
	FROG	0022545	0000377	R
SAMPLE02				

TRANSFER ADDRESS FOR SEGMENT SAMPLE02/OV1:DRO IS 0022545

SORTED CROSS REFERENCE LISTING:

ALPHA01	0021000	*SAMPLE02/LEX.SAMPLE02	
ALPHA02	0021400	*SAMPLE02/LEX.SAMPLE02	
BETA03	0022401	*SAMPLE02/LEX.SAMPLE02	
DATE	0020512	SAMPLE04/REL.EPSILON	*SAMPLE02/LEX.SAMPLE02
DECBC	0020430	*SAMPLE02/LEX.SAMPLE02	
DECDE	0020442	*SAMPLE02/LEX.SAMPLE02	
DECHL	0020454	*SAMPLE02/LEX.SAMPLE02	
DEDBC	0020432	*SAMPLE02/LEX.SAMPLE02	
DEDDE	0020444	*SAMPLE02/LEX.SAMPLE02	
DEDHL	0020456	*SAMPLE02/LEX.SAMPLE02	
FILESCAN	0020003	*SAMPLE02/LEX.SAMPLE02	
INCBC	0020416	*SAMPLE02/LEX.SAMPLE02	
INCDE	0020404	*SAMPLE02/LEX.SAMPLE02	
INCHL	0020372	*SAMPLE02/LEX.SAMPLE02	
INCSWP	0020466	*SAMPLE02/LEX.SAMPLE02	
INDBC	0020420	*SAMPLE02/LEX.SAMPLE02	
INDDE	0020406	*SAMPLE02/LEX.SAMPLE02	
INDHL	0020374	*SAMPLE02/LEX.SAMPLE02	
INDSWP	0020470	*SAMPLE02/LEX.SAMPLE02	
LABEL	0021543	*SAMPLE02/LEX.SAMPLE02	
MAINEND	0022401	*SAMPLE02/LEX.SAMPLE02	
OV1END	0024153		
PRINTADR	0000303	*SAMPLE02/LEX.SAMPLE02	
START	0021000	SAMPLE04/REL.EPSILON	*SAMPLE02/LEX.SAMPLE02

```
SEGMENT SAMPLE02/OV2,OV1END
LIBRARY SAMPLEOV/LEX
INCLUDE SAMPLE05/REL
LIBRARY RELOCLIB/SYS


PAB UTILIZATION IN PROGRAM SEGMENT SAMPLE02/OV2

PAB NAME   START      END      LENGTH
DUCK       0024153  0032046    0005674

SEGMENT LOWEST/HIGHEST ADDRESS: 0024153/0032046


PROGRAM NAME              PAB NAME    START    LENGTH   FLAGS

OMICRON                   DUCK        0024153  0005674  R

SAMPLE02



TRANSFER ADDRESS FOR SEGMENT SAMPLE02/OV2:DRO IS 0024153
```

SORTED CROSS REFERENCE LISTING:

ALPHA01	0021000	*SAMPLEOV/LEX.SAMPLE02
ALPHA02	0021400	*SAMPLEOV/LEX.SAMPLE02
BETA03	0022401	*SAMPLEOV/LEX.SAMPLE02
DATE	0020512	SAMPLE05/REL.OMICRON *SAMPLEOV/LEX.SAMPLE02
DECBC	0020430	*SAMPLEOV/LEX.SAMPLE02
DECDE	0020442	*SAMPLEOV/LEX.SAMPLE02
DECHL	0020454	*SAMPLEOV/LEX.SAMPLE02
DEDBC	0020432	*SAMPLEOV/LEX.SAMPLE02
DEDDE	0020444	*SAMPLEOV/LEX.SAMPLE02
DEDHL	0020456	*SAMPLEOV/LEX.SAMPLE02
FILESCAN	0020003	*SAMPLEOV/LEX.SAMPLE02
INCBC	0020416	*SAMPLEOV/LEX.SAMPLE02
INCDE	0020404	*SAMPLEOV/LEX.SAMPLE02
INCHL	0020372	*SAMPLEOV/LEX.SAMPLE02
INCSWP	0020466	*SAMPLEOV/LEX.SAMPLE02
INDBC	0020420	*SAMPLEOV/LEX.SAMPLE02
INDDE	0020406	*SAMPLEOV/LEX.SAMPLE02
INDHL	0020374	*SAMPLEOV/LEX.SAMPLE02
INDSWP	0020470	*SAMPLEOV/LEX.SAMPLE02
LABEL	0021543	*SAMPLEOV/LEX.SAMPLE02
MAINEND	0022401	*SAMPLEOV/LEX.SAMPLE02
OV1END	0024153	*SAMPLEOV/LEX.SAMPLE02
PRINTADR	0000303	*SAMPLEOV/LEX.SAMPLE02
START	0021000	*SAMPLEOV/LEX.SAMPLE02

Manual Name_____

Manual Number_____

READER'S COMMENTS

Did you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

_____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____

All comments and suggestions become the property of Datapoint.

Fold Here

Fold Here and Staple

FIRST CLASS
Permit
5774
San Antonio
Texas

BUSINESS REPLY MAIL
No Postage Necessary if mailed in the United States

Postage will be paid by:

DATAPOINT CORPORATION
DIRECTOR OF SOFTWARE SUPPORT
8550 Datapoint Drive, Mail Station# N60
San Antonio, Texas 78284