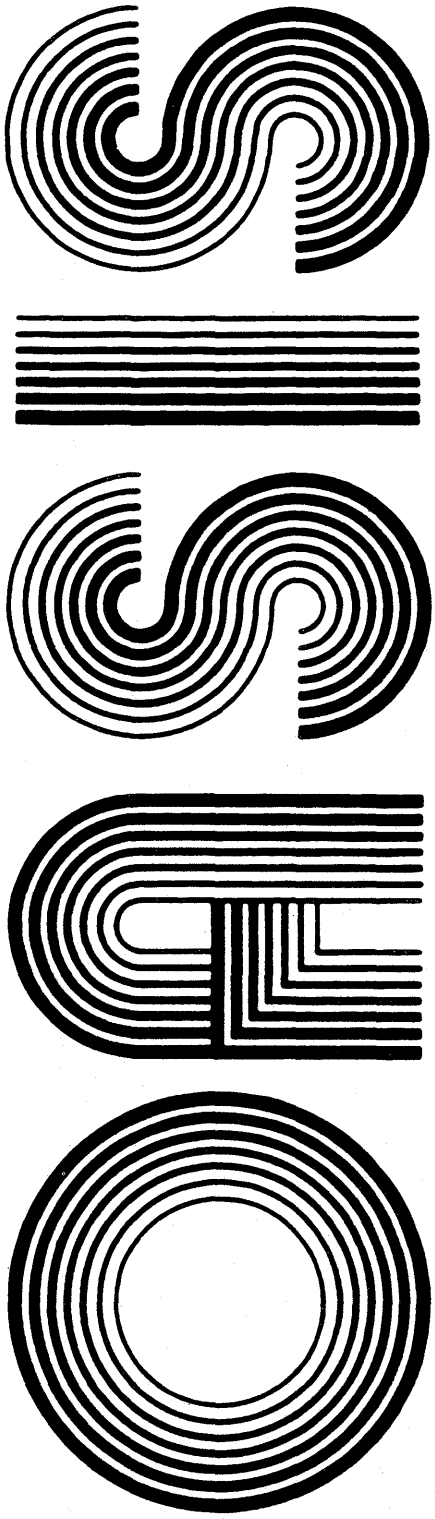
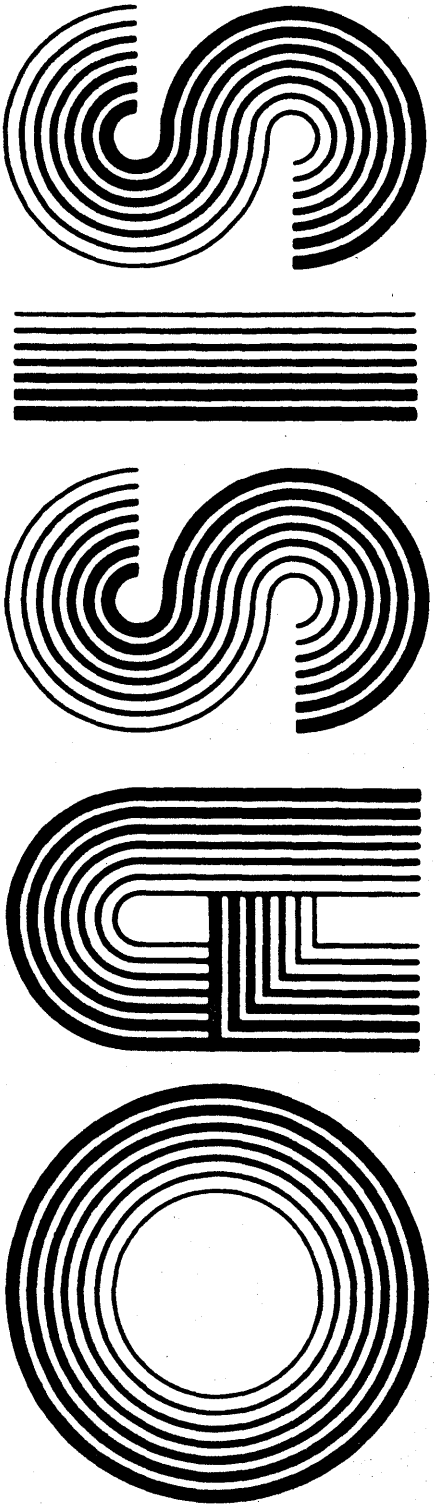


LINKAGE EDITOR REFERENCE MANUAL

OPERATING SYSTEM SOFTWARE

MAKES MICROS RUN LIKE MINIS

PHASE ONE SYSTEMS, INC.



LINKAGE EDITOR REFERENCE MANUAL

Second Edition

Revised

Documentation by: C. P. Williams
Software by: Timothy S. Williams

OPERATING SYSTEM SOFTWARE

MAKES MICROS RUN LIKE MINIS

PHASE ONE
SYSTEMS, INC.

7700 EDGEWATER DRIVE SUITE 830
OAKLAND, CALIFORNIA 94621 USA

P R E F A C E

This manual describes the Lingage Editor included in the development package available as an option with the OASIS Operating System.

This manual, named LINK , like all OASIS documentation manuals, has the manual name and revision number (if applicable) in the lower, inside corner of each page of the body of the manual. In most chapters of the manual the last primary subject being discussed on a page will be identified in the lower outside corner of the page.

Related Documentation

The following publications provide additional information useful in the use of the this program:

OASIS System Reference Manual

OASIS EXEC Language Reference Manual

OASIS Text Editor Reference Manual

OASIS MACRO Assembler Language Reference Manual

OASIS Dynamic Debugger Reference Manual

TABLE OF CONTENTS

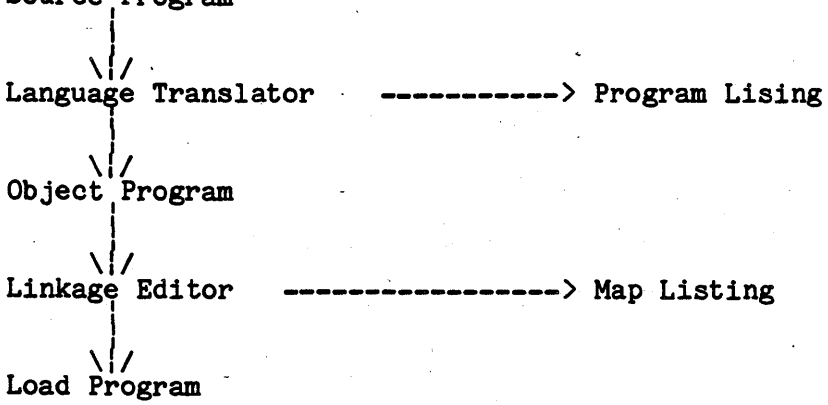
CHAPTER 1

INTRODUCTION

The OASIS Linkage Editor is a command program that is used to "link" together the output of an assembly or compilation process into an executable load module. This is a necessary step that follows the source program assembly or compilation of any problem program (except those programs written for the EXEC language processor, the OASIS BASIC compiler/interpreter, or the OASIS COBOL compilor).

Every program is designed to fulfill a particular purpose. To achive that purpose, the program can generally be divided into logical units that perform specific functions. A logical unit of coding that performs a function, or several related functions, is a module. Ordinarily, separate functions should be programmed into separate modules, a process called modular programming.

Each module is separately assembled or compiled by one of the language translators. The input to a language translator is a source module; the output from a language translator is an object module. Before an object module can be executed, it must be processed by the Linkage Editor. The output of the Linkage Editor is a load module.



Any module is composed of one or more program address blocks (PABS). A PAB is a unit of coding (instructions and data) that is, in itself, an entity. A PAB is the smallest separately relocatable unit of a program.

Each module in the input to the Linkage Editor may contain symbolic references to PABS in other modules; such references are called external references. The symbol referred to by an external reference must be either the name of a PAB or the name of an entry point in a PAB. PAB names and entry names are called external names. By matching an external reference with an external name, the Linkage Editor resolves references between modules.

The following chapters discuss the syntax of the LINK command, the use and requirements of the input files to the Linkage Editor, and the output of the Linkage Editor.

CHAPTER 2

LINK COMMAND

The OASIS Linkage Editor is invoked by executing the LINK command. The syntax of the command is:

LINK [<fn> [<ft> [<fd>]]] [(<option>...[])]

Where:

<fn> Indicates the file name of the object file to be linked or the file name of the specifications file (see FILE option).

<ft> Indicates the file type of the object file to be linked or the file type of the specifications file (see FILE option). A default file type of OBJECT will be used when option FILE is not specified and a default file type of LINK will be used when option FILE is specified.

<fd> Indicates the file disk of the object file to be linked or the file disk of the specifications file (see FILE option). When no <fd> is specified the default file search sequence will be used (see OASIS System Reference Manual).

LINK Options

BOOT Indicates that the output is a bootstrap loader. When this option is specified the first 256 bytes of the output of the linkage is written to sector 0 of the output drive.

DISK[=x] Indicates the drive that the output listing file is to be written to. When this option is specified the linkage parameters and map are written to a disk file with a file name equal to the <fn>, a file type of LINKMAP and the drive specified by this option.

DRIVE=x Indicates the drive that the output file is to reside on. When this option is not specified the output file will be on the same drive as the input file.

FILE Indicates that the file description specified is the file description of the file containing the LINK control parameters.

MAP Indicates that the linkage map is to be generated and output to the listing device. This is a defalut option.

NOMAP Indicates that the linkage map is not to be generated.

NOPRINT Indicates that the linkage parameters and map are not to be output to the printer. This is a default option.

NOTYPE Indicates that the linkage parameters and map are not to be displayed on the console.

NOXREF This option suppresses the cross reference table generation. This is a default option.

PRINTER[n] Indicates that the linkage parameters and map are to be output to the printer specified. If n is not specified then PRINTER1 is used.

SYSTEM Indicates that the output is a system file. For example: LINK CLASS2 (SYSTEM outputs the file SYSTEM.CLASS2:S.

TYPE Indicates that the linkage parameters and map are to be displayed on the console. This is a default option.

USR Indicates the the output is a BASICUSR file. For example: LINK UPPER (USR outputs the file UPPER.BASICUSR.

WORK=x Indicates the drive to be used for the linkage work files. When this option is not specified the system disk will be used.

XREF This option is not implemented as of this release.

When the LINK command is invoked with no file description specified the program will expect the specifications file to be entered from the console. In this mode the LINK prompt character (#) will be displayed when the LINK command is waiting for a command.

CHAPTER 3

LINK INPUT FILES

3.1 Specification File

The link specification file is the input file that controls that basic operations of the linkage process. This file may be a console file or a disk file. It is not necessary to use the specification file, in fact, the normal simple linkages don't use this file.

The specification file is normally used when two or more object modules are being linked together or when some parameters of the resulting load module need to be modified from the object code.

To use the console as a specification file do not specify a file description when invoking the Linkage Editor. For example: >LINK (PRINT.

To use a disk file as a specification file you must use the option FILE when invoking the Linkage Editor and there must be a file description specified. For example: >LINK TEST (FILE PRINT SYSTEM NOMAP.

In the following subsections the term <expression> refers to an arithmetic expression involving constants, previously defined symbols, and the operators +, -, *, and /. For example:

```
LABEL+23
LOC1+4*10H
1000H
```

An expression is evaluated in a left to right manner with no operator precedence. Numeric constants may be in decimal or hexadecimal (trailing H). String constants are specified with single quote characters surrounding them.

3.1.1 DEFINE Command

The DEFINE command allows you to assign a value to a symbol. The format of the command is:

DEFINE <symbol>=<expression>

Where:

<symbol> Specifies the symbol that is to be assigned a value. This symbol must have already been defined by one of the included object modules.

<expression> Specifies the value that is to be assigned to <symbol>.

The DEFINE command is normally used to resolve an unresolved reference.

3.1.2 END Command

The END command marks the end of the input specification file records and instructs the Linkage Editor to output the load module and the load map. The format of the command is:

END

When the END command is encountered the Linkage Editor attempts to resolve any unresolved references by searching all attached disks for object files with a file name the same as an unresolved reference. When a qualifying object file is found an INCLUDE is performed on that file. This process is repeated until the end of the table of unresolved references is reached. (Note: including a file in this manner may cause more unresolved references to be formed.)

When all references have been resolved that can be resolved in this manner and there still remains one or more unresolved references an implied LIST command is performed and the Linkage Editor returns to the console for further commands.

If there are no more unresolved references the load module is created on the specified or default disk, the load map is output to the list device and the Linkage Editor is exited.

3.1.3 ENTRY Command

The ENTRY command allows you to specify the execution entry point of the load module. The format of the command is:

ENTRY <expression>

Where:

<expression> Indicates the address of the entry point.

An ENTRY command has precedence over any end-of-file instructions that might specify an execution entry point.

3.1.4 INCLUDE Command

The INCLUDE command is the primary command of the input specification file. The INCLUDE command instructs the Linkage Editor to locate, analyze and assimilate an object file into the load module. The format of the command is:

INCLUDE <module name>[,<module name>]...

Where:

<module#name> Indicates the name of the object file to be included at this time. The file type of the object file must be OBJECT. More than one module name may be specified with one INCLUDE command by separating the module names with commas.

When the Linkage Editor receives an INCLUDE command it searches the attached disks for the module and includes the text and instructions of that module into the load module being built.

3.1.5 IGNORE Command

The IGNORE command allows you to create a load module that contains unresolved references by instrucing the Linkage Editor to ignore certain symbols. The format of the command is:

IGNORE <symbol>[,<symbol>]...

Where:

<symbol> Indicates the symbol that is to be ignored by the Linkage Editor. More than one symbol may be specified with one IGNORE command by separating the symbols with commas.

When a symbol is ignored by the Linkage Editor in this manner it is important to note that the reference to it is not actually taken out of the text of the load module--it merely references relative address zero. You should not ignore a symbol whose reference code is actually executed--the results will be undefined.

3.1.6 LIST Command

The LIST command allows you to see all of the currently unresolved references. The format of the command is:

LIST

When the LIST command is encountered the Linkage Editor displays all currently unresolved references on the list device.

3.1.7 NAME Command

The NAME command allows you to specify a program name for the load module that is different from the default. (The default name will be the name of the first included object module.) The format of the command is:

NAME <fn>[.<ft>][:<fd>]

Where:

<fn> Indicates the file or program name of the load module.

<ft> Indicates the file type of the load module. When this parameter is not specified the default file type will be used. (The default file type is dependent upon options used in the LINK command.)

<fd> Indicates the file disk of the load module. When this parameter is not specified the default file disk will be used. (The default file disk is dependent upon options used in the LINK command.)

3.1.8 ORIGIN Command

The ORIGIN command allows you to change a relocatable load module into an absolute load module. The format of the command is:

ORIGIN <expression>

Where:

<expression> Specifies the address that the load module is to be loaded at.

The ORIGIN command causes the relocation table to be used to change all relocatable references to absolute references and changes the load module into an absolute command module (the relocation table is not included in the load module).

3.1.9 OVERLAY Command

The OVERLAY command is not implemented in this version of LINK.

3.1.10 QUIT Command

The QUIT command allows you to abort the linkage process without creating a load module. The format of the command is:

QUIT

The QUIT command might be used when it is discovered that there are object modules required that have not been assembled yet or when the linkage is merely a test to determine unresolved references.

3.1.11 REPLACE Command

The REPLACE command allows you to change references from one, possibly undefined, symbol to another symbol. The format of the command is:

REPLACE <symbol1>=<symbol2>

Where:

<symbol1> Indicates the symbol that is to be replaced.

<symbol2> Indicates the symbol that is to replace <symbol1>.

The REPLACE command provides an easy means of linking an unfinished program. For example, the program might have calls to subroutines that are unwritten as yet. The REPLACE command could be used to change these references to a dummy subroutine that does exist without making a lot of changes to the source program just for test purposes.

Please note that symbols, as used by the Linkage Editor, are symbols defined as entry points, not just labels used in the assembly process.

3.1.12 SET Command

The SET command allows you to change the values in the load module text. The format of the command is:

SET <expression>=<data>[,<data>]...

Where:

<expression> Specifies the address, relative to the base address of the load module of the text to be changed.

<data> Is a list, separated by commas, of values that the text is to be changed to.

The SET command is normally used in, and is invaluable for, the modifications of parameters, defaults, etc., of a program without the modification of the source program.

3.1.13 Comments

Comments may be placed in the specifications file by using the semicolon (;) character. The Linkage Editor treats all characters in a record following the semicolon as a comment and will merely include them in any listing file that it may

create.

3.2 Object File

An OASIS object file is the primary output file from the MACRO assembler and the primary input file to the Linkage Editor. An object file is a binary stream, sequential format file of control and text records. Each record in an object file consists of a header section and a text section. The header section for each record contains three values:

<record length><record type><PAB number>

Where:

<record length> Specifies the number of bytes in the record, including the record length byte.

<record type> Specifies the type of record with one of the following codes:

01 PAB definition record (P)
03 Text record (T)
05 Relocation record (R)
07 Entry definition record (E)
09 External reference record (X)
0B PAB to PAB reference record (F)
0F End of file record (Z)

<PAB number> Indicates which PAB the data following refers to.

Following the header section of a record is the text section. This text section varies in structure from one record type to another. The following sub-sections describe the format of each record type. The letter in parentheses is the letter displayed by the OASIS LIST command for that record type.

3.2.1 PAB Definition Record (P)

The PAB definition record specifies the PAB name, type, base address, and length. The PAB number and length are relative to the current object file only.

<header><pab length><pab name><pab type><base address>

PAB types are coded as a number:

01 = absolute
02 = relocatable
04 = common relocatable

3.2.2 Text Record (T)

The text record is normally the most common type of record in an object file. It contains the assembled instructions and data constants as specified in the source program.

<header><start addr><data>[<data>]...

3.2.3 Relocation Record (R)

The relocation record specifies a list of addresses within a PAB that must have the load address of the PAB added to them to form accurate address references. Although relocation record(s) may appear anywhere in the object file before the end of file record it is normal for this type of record to immediately follow the text records affected (see examples).

<header><addr>[<addr>]...

3.2.4 Entry Definition Record (E)

The entry definition record specifies an address within a PAB that has been specified as an entry point with the ENTRY instruction in the source program. Along with the address the entry label is specified.

<header><entry addr><entry name>

When the Linkage Editor encounters an entry definition record it saves the entry point location and label and also searches its unresolved references table looking for any references that can be resolved by this definition.

3.2.5 External Reference Record (X)

The external reference record specifies an address within a PAB that is a reference to a label specified as an externally defined label with an EXTRN instruction in the source program. The address and label referenced is specified in this record.

<header><ref from addr><ref to name>

The Linkage Editor tries to resolve this external reference by matching it with its currently defined entry point locations (defined with the entry definition record). If no match is found the external reference data is saved in the unresolved references table.

3.2.6 PAB to PAB Reference Record (F)

The PAB to PAB reference record is a special type of external reference, similar to the external reference record. The main difference is that the referenced label was resolved by the assembler because the label was defined in another PAB of the same assembly.

<header><ref from addr><ref to pab>[<ref from addr><ref to pab>]...

The PAB to PAB reference record specifies a list of addresses within a PAB that are references to other PABs.

The Linkage Editor uses the information in this record by adding the referenced PAB's base address to the referencing address and also adds the referencing address to the relocation table.

3.2.7 End of File Record (Z)

This record indicates the logical end of file for the object program. Normally this record is only two bytes in length (count and record type). When the source program contained an END statement that specified a starting address this record will also note the starting address and its PAB number. In this latter case the record length will be five.

<header>[<start pab><start addr>]

3.2.8 Object File Record Examples

The following example object records are displayed as the LIST command displays them. This differs from the actual contents of the records in that the record type is displayed with the letter code instead of the numeric code and addresses are displayed in normalized mode instead of Z80 mode (low byte first).

10 P 00 04A2 MAP 02 0000	PAB definition, record length 16, relative PAB number 0, PAB length of 1186, PAB name is MAP, PAB type is relocatable, base address of 0.
08 T 00 00D9 7E19FF	Text record, record length 8, relative PAB number 0, addresses starting at 00D9, text is 7E 19 FF at locations 00D9, 00DA, and 00DB respectively.
05 R 01 000A 0012 003F	Relocation record, record length 5, relative PAB number 1, relative addresses 000A, 0012, and 003F in that PAB must have the load address added to them before execution.
0D E 00 0039 MAP	Entry definition record, record length 13, relative PAB number 0, address 0039 is the entry point named MAP (trailing spaces added).
0D X 00 0052 HELPMSG	External reference record, record length 13, relative PAB 0, address 0052 references external label HELPMSG (trailing space added).
09 F 00 004F 02 0086 01	PAB to PAB reference record, record length 9, relative PAB number 0, address 004F is a reference to relative PAB number 2, address 0086 is a reference to relative PAB number 1.
05 Z 00 0039	End of file record, record length 5, execution address is 0039 in PAB 0.

CHAPTER 4

LINK OUTPUT FILES

The output of the Linkage Editor generally includes two files: the load module and the listing file. The load module may be one of two forms dependant upon whether the load module is absolute or relocatable.

Load modules, when output, are always output to a disk file. The listing file, when output, may be output to a disk file, the console (default), or to one of the attached printer devices, dependant upon the options specified to the Linkage Editor.

4.1 Absolute Load Module File

An absolute load module output by the Linkage Editor is an exact image of the program to be executed. The directory entry for this type of a file specifies the load address and the load address is the execution address. An example of this type of a load module is the SYSTEM.NUCLEUS:S.

4.2 Relocatable Load Module File

A relocatable load module output by the Linkage Editor consists of two sections. The first section is an exact image of the program to be executed if it were loaded at address zero. The second section is the relocation table for the first section.

This relocation table consists of variable length records with the first byte specifying the word count. Following the word count byte are two byte entries (words) that specify relative addresses of the load module that need to have the relocation constant added to them before program execution begins. The relocation constant is the load address of the program. This load address is not known untill the program is actually loaded into memory by the system.

The directory entry of a relocatable load module contains a record count that includes the recount count of both sections. Records in the first section are always 256 bytes in length.

Most OASIS commands are distributed as relocatable load module files.

4.3 Map Listing File

The map listing file may be output to a disk file, the console or one of the printer devices, dependant upon the options specified to the LINK command. The DISK option will cause the listing file to be output to disk; the PRINTER option will cause the listing file to be output to one of the printers; the TYPE option will cause the listing file to be output to the console (default); the NOTYPE option will cause the listing file to be suppressed.

The map listing file consists of two sections. The first section is a listing of the input specifications file, including any comments.

The second section is a memory map of the load module created by the Linkage Editor. This map is a listing of the PABs used in the load module. Listed with each PAB is the memory region used by the PAB, the PAB type, and the entry points defined in the PAB in ascending address sequence.

At the end of the memory map the relative entry address is listed along with the total length of the load module.

(This page intentionally left blank)

APPENDIX A

LINK EXAMPLES

A.1 Example One: Simple, Single PAB

```
>MACRO CLASS6
SYSTEM.CLASS6 - Hazeltine 1400-1500                    04/28/80 15:09 Page 1
                                                       CLASS6.ASSEMBLE:SOURCE$$

Addr Obj-Code  Line *** Source Statement ***

                    2                   ; Operates with 1420, 1500, 1510, 1520
                    3                   ; Change leadin to ESC for 1410 and 1552
                    4           MACLIB  CLASS
                    5
0000                6           INIT
                   10+
0000 C33600        11+          JP      TRANIN    ; input vector
                   12+
                   13+; test if control
                   14+
0003               15+TRANOUT:
0003 FE20          16+          CP      20H       ; is it control?
0005 3804          17+          JR      C,CTL     ; brif is
0007 CF40          18+          SC      DEVOUT    ; else, display as is
0009 AF            19+          XOR     A         ; clear cy
000A C9            20+          RET
                   21+
                   22+; test if dca x,y
                   23+
000B               24+CTL:
000B FE10          25+          CP      DLE       ; is it 10H
000D CA9F00        26+          JP      Z,DCA     ; jump if is
                   27+
                   28+          IF      .NOT..NUL.
                   30+          ENDIF
                   31+
                   32+; point to proper entry
                   33+
0010 215F00        34+          LD      HL,TAB1-2 ; point to indirect table
0013 87            35+          ADD     A         ; code times two
0014 5F            36+          LD      E,A       ; move to de
0015 1600          37+          LD      D,0       ; 16 bits
0017 19            38+          ADD     HL,DE     ; point to correct slot
0018 5E            39+          LD      E,(HL)    ; get address in de
0019 23            40+          INC     HL
001A 56            41+          LD      D,(HL)
                   42+
                   43+; test for not available
                   44+
001B 7A            45+          LD      A,D       ; is address = zero?
001C B3            46+          OR      E
001D 37            47+          SCF               ; set cy just in case
001E C8            48+          RET     Z         ; return if is
                   49+
                   50+; put out codes until byte = 0FFH
                   51+
001F               52+WRITE:
001F 1A            53+          LD      A,(DE)    ; get byte
0020 13            54+          INC     DE        ; bump
0021 FE8C          55+          CP      8CH       ; ff delay code?
0023 2807          56+          JR      Z,WRFFDLY ; brif is
0025 4F            57+          LD      C,A       ; move to reg c
0026 3C            58+          INC     A         ; test for 0FFH
0027 C8            59+          RET     Z         ; return if is
0028 CF40          60+          SC      DEVOUT    ; else, write to console
002A 18F3          61+          JR      WRITE     ; loop
002C               62+WRFFDLY:
002C FD7E07        63+          LD      A,(IY+7)  ; get delay rate
002F B7            64+          OR      A         ; any?
0030 28ED          65+          JR      Z,WRITE   ; no, ignore
0032 CF4C          66+          SC      DELAY     ; else, pause
0034 18E9          67+          JR      WRITE     ; continue
                   68+
                   69+
```

```
                    70+; input char translate routine
                    71+
0036                72+TRANIN:
                    73+             IF      .NUL.
                    74+             IF      .NUL.
                    79+             ENDIF
                    80+             IF      .NUL.
                    85+             ENDIF
                    86+             IF      .NUL.
                    91+             ENDIF
                    92+             IF      .NUL.
                    97+             ENDIF
                    98+             IF      .NUL.
                   103+             ENDIF
                   104+             IF      .NUL.
                   109+             ENDIF
                   110+             IF      .NUL.
                   115+             ENDIF
                   116+             IF      .NUL.
                   121+             ENDIF
                   124+             ENDIF
0036 B7            125+             OR      A           ; clear cy
0037 FDCB057E      126+             BIT     7,(IY+5)    ; is this conin device?
003B C8            127+             RET     Z           ; no, return
003C FDCB0576      128+             BIT     6,(IY+5)    ; 2nd char of esc sequence?
0040 2013          129+             JR      NZ,ESC2     ; brif is
0042 FDE5          130+             PUSH    IY          ; save iy
0044 CF30          131+             SC      GETSCR      ; point to scr
0046 FDBE40        132+             CP      (IY+64)     ; is this an esc char
0049 FDE1          133+             POP     IY          ; restore iy
004B 2802          134+             JR      Z,ESC1      ; brif is
004D B7            135+             OR      A           ; turn off cy
004E C9            136+             RET                 ; return
004F               137+ESC1:
004F FDCB05F6      138+             SET     6,(IY+5)    ; turn on code
0053 37            139+             SCF                 ; set cy
0054 C9            140+             RET                 ; return
0055               141+ESC2:
0055 FDCB05B6      142+             RES     6,(IY+5)    ; turn off sw
0059 CBAF          143+             RES     5,A         ; fold the char
005B CF4E          144+             SC      CONESC      ; go translate
005D B7            145+             OR      A           ; test if any
005E C0            146+             RET     NZ          ; yes, return
005F 37            147+             SCF                 ; turn on cy
0060 C9            148+             RET                 ; return
                   149+
                   150+
                   200
009F               201              DCA     6
                   204+
009F               205+DCA:
                   282+;
                   283+; Hazeltine 1500
                   284+;
009F 0E7E          285+             LD      C,'~'       ; lead 1
00A1 CF40          286+             SC      DEVOUT
00A3 0E11          287+             LD      C,DC1       ; lead 2
00A5 CF40          288+             SC      DEVOUT
00A7 7C            289+             LD      A,H         ; col number
00A8 C660          290+             ADD     96          ; bias
00AA 4F            291+             LD      C,A
00AB CF40          292+             SC      DEVOUT      ; display
00AD 7D            293+             LD      A,L         ; line number
00AE C620          294+             ADD     32          ; bias
00B0 4F            295+             LD      C,A
00B1 CF40          296+             SC      DEVOUT      ; display
00B3 AF            297+             XOR     A           ; clear cy
00B4 C9            298+             RET
                   299+
                   300+
                   400
00B5               401              DEFINE HOME,~,DC2
00B8               524              DEFINE CLEAR,^,FS,8CH
00BC               654              DEFINE EOS,~,CAN,8CH
00C0               784              DEFINE EOL,~,SI,8CH
00C4               914              DEFINE LEFT,BS
```

```
00C6              1030          DEFINE RIGHT,DLE
00C8              1146          DEFINE UP,~,FF
00CB              1269          DEFINE DOWN,~,VT
00CE              1392          DEFINE IL,~,SUB
00D1              1515          DEFINE IC
00D2              1624          DEFINE DL,~,DC3
00D5              1747          DEFINE DC
00D6              1856          DEFINE EU,~,GS
00D9              1979          DEFINE PON,~,EM
00DC              2102          DEFINE POFF,~,US
00DF              2225          DEFINE FON
00E0              2334          DEFINE FOFF
00E1              2443          DEFINE BON,~,SOH
00E4              2566          DEFINE BOFF,~,US
00E7              2689          DEFINE ULON
00E8              2798          DEFINE ULOFF
00E9              2907          DEFINE RVON
00EA              3016          DEFINE RVOFF
                  3125
00EB              3126          END

No assembly errors.

>LIST CLASS6.OBJECT:A (OBJECT

10 P 00 00EB CODE     02 0000
07 T 00 0000 C336
80 T 00 0002 00FE203804CF40AFC9FE10CA9F00215F00875F1600195E23567AB337C8
             1A13FE8C28074F3CC8CF4018F3FD7E07B728EDCF4C18E9B7FDCB057EC8
             FDCB05762013FDE5CF30FDBE40FDE12802B7C9FDCB05F637C9FDCB05B6
             C6AFCF4EB7C037C900000000000000000000000000000000000000000000
             000000000000000
3D T 00 007D 000000000000000000000000000000000000000000000000000000000000000
             00000000000E7ECF400E11CF407CC6604FCF407DC6204FCF40AFC9
07 T 00 0061 B500
08 T 00 00B5 7E12FF
07 T 00 0077 B800
09 T 00 00B8 7E1C8CFF
07 T 00 008F BC00
09 T 00 00BC 7E188CFF
07 T 00 008D C000
09 T 00 00C0 7E0F8CFF
07 T 00 006F C400
07 T 00 00C4 08FF
07 T 00 006B C600
07 T 00 00C6 10FF
07 T 00 0093 C800
08 T 00 00C8 7E0CFF
07 T 00 0073 CB00
08 T 00 00CB 7E0BFF
07 T 00 0081 CE00
08 T 00 00CE 7E1AFF
07 T 00 0085 D100
06 T 00 00D1 FF
07 T 00 0083 D200
08 T 00 00D2 7E13FF
07 T 00 0087 D500
06 T 00 00D5 FF
07 T 00 0097 D600
08 T 00 00D6 7E1DFF
07 T 00 0067 D900
08 T 00 00D9 7E19FF
07 T 00 0069 DC00
08 T 00 00DC 7E1FFF
07 T 00 0063 DF00
06 T 00 00DF FF
07 T 00 0065 E000
06 T 00 00E0 FF
07 T 00 0099 E100
08 T 00 00E1 7E01FF
07 T 00 009B E400
08 T 00 00E4 7E1FFF
07 T 00 0075 E700
06 T 00 00E7 FF
07 T 00 008B E800
06 T 00 00E8 FF
```

```
07 T 00 007B E900
06 T 00 00E9 FF
07 T 00 007D EA00
06 T 00 00EA FF
37 R 00 0001 000E 0011 0061 0077 008F 008E 006F 006B 0093 0073 0081 0085
             0083 0087 0097 0067 0069 0063 0065 0099 009B 0075 008B 007B
             007D
02 Z

>LINK CLASS6 (SYSTEM

LINK version 5.4B                    CLASS6              04/28/80 14:06 Page 1

Memory map for SYSTEM.CLASS6:S

PAB-name     Low  High Length Type       Entry       Addr

CODE        0000  00EA  00EB  REL

Entry Address: 0000

Total Length:  00EB (235 decimal)

>DUMP SYSTEM CLASS6 S

Sector 1764  (Track=68, Sector=9)
0000: C33600FE 203804CF 40AFC9FE 10CA9F00 '.6.. 8..@.......'
0010: 215F0087 5F160019 5E23567A B337C81A '!_._...^#Vz.7..'
0020: 13FE8C28 074F3CC8 CF4018F3 FD7E07B7 '...(.O<..@...~..'
0030: 28EDCF4C 18E9B7FD CB057EC8 FDCB0576 '(..L......~....v'
0040: 2013FDE5 CF30FDBE 40FDE128 02B7C9FD ' ....0..@..(....'
0050: CB05F637 C9FDCB05 B6CBAFCF 4EB7C037 '...7........N..7'
0060: C9B500DF 00E000D9 00DC00C6 000000C4 '................'
0070: 000000CB 00E700B8 000000E9 00EA0000 '................'
0080: 00CE00D2 00D100D5 000000E8 00C000BC '................'
0090: 000000C8 000000D6 00E100E4 0000000E '................'
00A0: 7ECF400E 11CF407C C6604FCF 407DC620 '~.@...@|.`O.@}. '
00B0: 4FCF40AF C97E12FF 7E1C8CFF 7E188CFF 'O.@..~..~...~...'
00C0: 7E0F8CFF 08FF10FF 7E0CFF7E 0BFF7E1A '~.......~..~..~.'
00D0: FFFF7E13 FFFF7E1D FF7E19FF 7E1FFFFF '..~...~..~..~...'
00E0: FF7E01FF 7E1FFFFF FFFFFF00 00000000 '.~..~...........'
00F0: 00000000 00000000 00000000 00000000 '................'
Sector 1765  (Track=68, Sector=10)
0100: 1A01000E 00110061 0077008F 008D006F '.......a.w.....o'
0110: 006B0093 00730081 00850083 00870097 '.k...s..........'
0120: 00670069 00630065 0099009B 0075008B '.g.i.c.e.....u..'
0130: 007B007D 00000000 00000000 00000000 '.{.}............'
0140: 00000000 00000000 00000000 00000000 '................'
0150: 00000000 00000000 00000000 00000000 '................'
0160: 00000000 00000000 00000000 00000000 '................'
0170: 00000000 00000000 00000000 00000000 '................'
0180: 00000000 00000000 00000000 00000000 '................'
0190: 00000000 00000000 00000000 00000000 '................'
01A0: 00000000 00000000 00000000 00000000 '................'
01B0: 00000000 00000000 00000000 00000000 '................'
01C0: 00000000 00000000 00000000 00000000 '................'
01D0: 00000000 00000000 00000000 00000000 '................'
01E0: 00000000 00000000 00000000 00000000 '................'
01F0: 00000000 00000000 00000000 00000000 '................'
```

A.2 Example Two: Specification File

This example is a listing of the specification file used to link the OASIS NUCLEUS command. Note the aboundant use of comments and the modularity of the object modules. This makes maintenance of the program easier and is the recommended practice for all programs other than the simple, single module code.

The various DEFINEs, IGNOREs and SETs are used to customize various parameters to a specific configuration.

The file is named NUCLEUS.LINK and is used by entering the command:

```
>LINK NUCLEUS (FILE

NAME SYSTEM.NUCLEUS:A
;
INCLUDE N$BASE          ; Low memory assignments
INCLUDE N$MU            ; Multi user definitions
INCLUDE N$SC            ; System Call dispatch
INCLUDE N$DISKIO        ; Disk I/O interface
INCLUDE N$KEYIN         ; Console input line
INCLUDE N$DISPLA        ; DISPLAY, SYSDISP
INCLUDE N$CONIN         ; CONIN, SYSIN, CONST
INCLUDE N$CONOUT        ; CONOUT, SYSOUT
INCLUDE N$PRINT         ; PRTOUT, PRINT
INCLUDE N$DEVICE        ; DEVINIT,DEVIN,DEVST,DEVOUT,DEVUNINI,PUTDEV
INCLUDE N$NUMBER        ; NUMBER,HEXI,HEXO,DECI,DECO
INCLUDE N$MULDIV        ; 16 Bit MULTIPLY/DIVIDE
INCLUDE N$RECLOC        ; Sector/File Locks
INCLUDE N$SNU           ; Select-Next-User, Activate
INCLUDE N$BYTE          ; GETBYTE, PUTBYTE
INCLUDE N$EXCLUS        ; Lock maintence
INCLUDE N$PEEK          ; Peek at conout
INCLUDE N$MSG           ; MSG Sender
INCLUDE N$DIRECT        ; Volume directory management
INCLUDE N$ALLOC         ; Volume space management
INCLUDE N$DATE          ; Date & Time conversion
INCLUDE N$CLOCK         ; TOD, MSEC, DELAY
INCLUDE N$TIMER         ; TEB Maintanence
INCLUDE N$UTILIT        ; Set & Point to various areas
INCLUDE N$MISC          ; Internal subroutines
INCLUDE N$COMPAR        ; COMPARE/DISPATCH
INCLUDE N$ESC           ; CONSOLE ESCAPE
INCLUDE N$SEQIO         ; Non Disk Sequential I/O
INCLUDE N$OPEN          ; OPEN
INCLUDE N$CLOSE         ; CLOSE
INCLUDE N$SEQDIO        ; Seq Disk I/O
INCLUDE N$DIRIO         ; Direct I/O
INCLUDE N$IXIO          ; Indexed file I/O
INCLUDE N$LOADER        ; QUIT,FETCH,LOAD,PGMINIT,EXCMD
INCLUDE N$ERROR         ; ERRDISP,ERRQUIT
INCLUDE N$NEWSYS        ; NEWSYS
;
; Start of user area
;
INCLUDE N$IPL           ; Initial Program Loader
INCLUDE N$MUSIZE        ; Size all banks
INCLUDE VG              ; Interupts, clocks
INCLUDE VGDISK          ; Disk driver
INCLUDE VGBANK          ; Bank select
INCLUDE N$MUIM1         ; INT mode 1
INCLUDE VGINIT          ; IPL init routine
DEFINE RST10=RET
DEFINE RST18=RET
DEFINE RST20=RET
DEFINE RST28=DEBUG
DEFINE RST30=RET
DEFINE RST38=RET
DEFINE NMI=RET
DEFINE DISK0=VGDISK
DEFINE DISK1=VGDISK
DEFINE DISK2=VGDISK
DEFINE DISK3=VGDISK
IGNORE DISK4,DISK5,DISK6,DISK7
SET NUCLEUS+3=55        ; CLK1=55
SET NUCLEUS+0EH=07BH    ; Set switches ERRTEXT,RTC,HIST,MODE2 on
```

```
SET NUCLEUS+3EH=255     ; Multi-user switch
SET NUCLEUS+46H=250     ; 4 Mhz
SET CLKSW=085H          ; RTC avail, RST5=BP
SET LUB+0=0             ; Disk = DEV1
SET LUB+8=16            ; CONIN=SYSTEM.DEV17
SET LUB+9=16            ; CONOUT=SYSTEM.DEV17
SET UCB0+23=250,30,10   ; Disk STP, SET
SET UCB1+23=250,30,10
SET UCB2+23=250,30,10
SET UCB3+23=250,30,10
SET UCB16+2=79          ; Line length
SET UCB16+3=23          ; Page length
SET UCB16+4=6           ; Class = 6 (Hazeltine)
ORIGIN 0
END
```

A.3 Example Three: Multiple PABs

This example shows a simple program example that uses two PABs. This program is incomplete in that it only checks to see if the operator has requested a help message display. When the operator has not requested a help message the program exits. At this point is where the normal program logic would be coded, possibly using additional PABs or the same ones.

```
>EDIT EX3CODE ASSEMBLE
NEW FILE
EDIT
*INPUT
            TITLE     'Example 3 - multiple PABs'
EX3CODE:    REL
            ENTRY     EX3CODE
            EXTRN     HELP,HELPMSG,PROG
EX3CODE:
            PUSH      BC              ; Save drive code
            PUSH      DE              ; Save sector
            PUSH      HL              ; Save parameter pointer
            LD        B,9             ; Length
            LD        DE,HELP         ; Point to HELP lit
.TSTHELP:
            LD        A,(DE)          ; Get byte
            CP        (HL)            ; Compare with token
            JR        NZ,.NOHELP      ; Branch if not equal
            INC       DE              ; Else bump pointers
            INC       HL              ;
            DJNZ      .TSTHELP        ; Loop
                                      ; Is HELP request - display
            POP       HL              ; Restore regs
            POP       DE              ;
            POP       BC              ;
            LD        DE,HELPMSG      ; Point to help message
.PAGE:
            LD        B,9             ; Point to CONOUT
            SC        59              ; Get lines/page
            LD        B,C             ; Move to B reg
.LINE:
            LD        A,(DE)          ; Get character
            OR        A               ; Test if end
            RET       Z               ; Return to OASIS if is
            SC        2               ; Else display
            DJNZ      .LINE           ; Loop
            SC        49              ; Wait at bottom of page
            JR        .PAGE           ; Display next page
.NOHELP:
            POP       HL              ; Restore regs
            POP       DE              ;
            POP       BC              ;
            JP        PROG
            END       EX3CODE
*FILE
"EX3CODE.ASSEMBLE:A" filed

>EDIT EX3HELP ASSEMBLE:A
NEW FILE
EDIT
*INPUT
            TITLE     'Example 3 - Help Message Data'
EX3HELP:    REL
            ENTRY     HELP,HELPMSG
HELP:       DC        'HELP    ',13
HELPMSG:    DC        'Function: To illustrate an example',13
            DC        '          of a multi-PAB program',13
            DC        '          linkage',13
            DC        13
            DC        'Syntax: EX3CODE [(options[])]',13
            DC        13
            DC        'Where options are:',13
            DC        ' PRINTERn  output to printer # n',13
            DC        ' TYPE      output to the console',13
            DC        ' NOTYPE    suppress output',13
            DC        0
            END
*FILE
```

```
"EX3HELP.ASSEMBLE:A" filed

>EDIT EX3PROG ASSEMBLE A
NEW FILE
EDIT
*INPUT
          TITLE     'Example 3 - Program'
          ENTRY     PROG
EX3CODE:  REL
PROG:
          XOR       A           ; Clear return code
          SC        0           ; Exit
          END
*FILE
"EX3PROG.ASSEMBLE:A" filed

>MACRO EX3CODE
Pass one
Pass two
No assembly errors

>MACRO EX3HELP
Pass one
Pass two
No assembly errors

>MACRO EX3PROG
Pass one
Pass two
No assembly errors

>LINK
#INC EX3CODE,EX3PROG,EX3HELP
#NAME EX3:S
#END

LINK version 5.4B                                   04/30/80 12:08 Page 1

Memory map for EX3.COMMAND:A

PAB-name     Low  High Length Type      Entry     Addr

EX3CODE      0000 002B 002C   REL       EX3CODE   0000
                                        PROG      002B

EX3HELP      002F 011B 00F0   REL       HELP      002C
                                        HELPMSG   0038

Entry Address: 0000

Total Length:  011F (287 decimal)
```

APPENDIX B

LINK ERRORS & MESSAGES

**** File "xxxxxxxx.OBJECT" not found**

This message is displayed following an INCLUDE command of a file that cannot be found on any of the attached disk drives.

**** Including**

This message is displayed when the Linkage Editor is performing automatic includes following an END command.

**** Invalid character in expression**

This message is displayed when an invalid expression is detected. Expression may only contain valid symbols (one to eight characters in length, must start with a letter and contain only letters, digits, dollar signs, and periods) numeric constants (must start with a digit and contain only digits, the letters A through F, and may be terminated with the letter H) and the arithmetic operators: + - * /.

**** Invalid command**

**** Not Implemented**

**** Relocation error**

Indicates that an expression containing relocatable symbols is in error. Usually the error is one of the following: a difference between two relocatable symbols of different PABs; the sum of two relocatable symbols; the product of two relocatable symbols; the quotient of two relocatable symbols; the product or quotient of a relocatable symbol and an absolute symbol.

**** Too many segments**

Up to 128 segments or object modules may be included in one linkage.

**** Undefined symbol**

An expression using symbols or the DEFINE, IGNORE, or REPLACE command reference an undefined symbol (a symbol not specified by an entry or external definition record).