SLAC Computation Group
Stanford, California

CGTM No. 160

XASM11/XLINK11

A PDP-11 CROSS-ASSEMBLER/LINKER

User's Manual

by

S. Steppel and H.E. Syrett

August 23, 1974

Version 2.0

TABLE OF CONTENTS

		Page
1.	Introduction	1
2.	XASM11: The Cross-Assembler	2
2.1	Control Card Options	2
2.2	Assembler Directives	3
2.2.1	Storage Directives	3
2.2.2	Object Module Directives	4
2.2.3	Input/Output Directives	5
2.2.4	Conditional Assembly	5
2.3	Operating Procedures	6
3.	XLINK11: The Cross-Linker	9
3.1	Linker Parameters	9
3.1.1	T/B - Top or Bottom Loading	10
3.1.2	TR - Transfer Address Handling	11
3.1.3	MAP/NOMAP - Optional Listing	12
3.1.4	ALT - Alternate Output Routine	12
3.2	Operating Procedures	13
Appendix A:	Sample Run	17
Appendix B:	In-Line Procedures	18

1. INTRODUCTION

The purpose of making available on the IBM System/360, an assembler for PDP-11 Assembler Language is threefold:

(1) to free the PDP-11 from the burden of assemblies so that it can run dedicated to a particular application;

(2) to speed up debugging of assembly errors;

(3) to allow storage and editing of PDP-11 source programs on the /360.

This last point is especially important when a powerful text editor, such as SLAC WYLBUR, can be used on the /360.

This paper provides a User's Manual for XASM11, a cross-assembler for the PDP-11, and XLINK11, the corresponding cross-linker, both written in /360 assembler language at the SLAC Computation Group. It is assumed that the user is familiar with the PDP-11 PAL-11R Assembler and the Link-11 Linker Programmer's Manuals. Documentation for maintaining or modifying XASM11 and XLINK11 is not included in this paper, but is provided elsewhere.

2. XASM11: THE CROSS-ASSEMBLER

XASM11 assembles the input language defined in the PAL-11R Assembler Programmer's Manual, except for certain assembler directives discussed below (Section 2.2). Output includes a symbol table and listing with code and flags as described in the PAL-11R manual. Two features are provided which are not available in PAL-11R. These are:

(1) a cross-reference listing, incorporated into the symbol table listing;

(2) user-variable page headings, which include the date and time of the assembly (see Section 2.2.3).

The present version of XASM11 (Version 2.0) does not include any macro-facility.

Assembler options have been revised to accept card input rather than the paper tape or teletype input available on the PDP-11. Since most /360 systems include a card punch but no paper tape punch, the binary object output of XASM11 was designed to be in 80-column card-image format. Thus, XASM11 object modules can only be used as input to the corresponding cross-linker, XLINK11, and are not acceptable as input to the standard PDP-11 linker. Printer control indicators have also been provided in XASM11 to allow line-spacing and page-ejection as desired by the programmer.

2.1 Control Card Options

The PAL-11R Assembler has various options which are specified at the invocation of the assembler. Those provided by the Cross Assembler are:

T Symbol Table. Suppress printout of the symbol table and named CSECT table.

L Assembly Listing. Suppress listing of the source program and the generated object code except for statements flagged in error, which are always printed.

B Object Module. Do not produce a binary object module in card-image format.

An assembler control card serves the purpose of specifying these options. It is identified An assembler control card serves the purpose of specifying these options. It is identified by 'XASM ' in the first five columns, followed by a list of options (in any order) separated by commas. The card may look as follows:

```
XASM B,L,T
```

If the card is omitted, the default options are to provide a full listing of the symbol and named CSECT tables, the source program and generated code, and to produce a binary object deck.

Note that each PDP-11 source program in the batch may be prefaced by a control card. If it is, the defaults are reset before the options specified are applied. If there is no control card, the options for the previous program remain in effect.

2.2 Assembler Directives

The assembler directives (or pseudo-ops) fall into several categories: storage, object module, I/O control, and conditional assembly. All but the conditional assembly directives have been implemented in Version 2.0 of XASM11, and three new I/O control directives have been introduced. For details of directive processing logic, see Section 5.1.7.

2.2.1 Storage Directives

All storage directives are identical to those described in the PAL-11R Assembler Programmer's Manual. They are:

.BYTE	E,E,...	Assigns a byte of storage for each expression value.
.WORD	E,E,...	Assigns a word of storage for each expression value.
.ASCII	/XX...X/	Assigns a byte of storage for each character.
.RAD50	/XXX/	Assigns one byte of storage for the expression value in Radix-50 form.
.EVEN		Aligns the location counter on the next full word boundary.

2.2.2 Object Module Directives

The object module directives enable the programmer to design his program in separate control sections either absolute or relocatable. A directive is provided to enable communication between these sections. These directives have been implemented, thus allowing large programs to be assembled in separate sections; the object modules from separate assemblies may be stored on the /360 and later linked together to form a complete load module on magnetic tape, using the PDP-11 Cross-Linker (XLINK11).

.TITLE	name	Assigns a name to the object module (truncated to the first six characters).
.GLOBL	sym1,sym2,...	Declares each symbol in the list as global.
.ASECT		Specifies the beginning or resumption of the absolute program section.
.CSECT		Specifies the beginning or resumption of unnamed relocatable control section.
.CSECT	sym	Specifies the beginning or resumption of the relocatable control section named "sym".
.LIMIT		Generates two words for the linker to fill in the upper and lower bounds of the load module's relocatable code.

2.2.3 Input/Output Directives

Input/Output directives signal the assembler that the program is finished, or should be continued from another tape.

.END tr	Physical end of program with transfer address "tr" (which may be omitted).
.EOT	Physical end of one in a series of input tapes.

The latter is necessary under PDP-11 DOS since serial input tapes can be handled by the system. Since input to the Cross Assembler is cards, there is no need for the .EOT directive; it is merely ignored by XASM11.

Three new directives have been added to control the output listing:

.SPACE k	Print k blank lines. Print one blank line if no operand.
.EJECT	Skip to the top of the next page.
.HEAD text	Same as .EJECT, but the text (everything on the card to the right of the blank following .HEAD up to column 72) is printed at the top of the new page and all subsequent pages until the next .HEAD directive. encountered.

.SPACE and .EJECT replace the line feed and form feed characters which can be input from a teletype, but not from cards.

2.2.4 Conditional Assembly

The conditional assembly directives provide an additional convenience for the programmer. They are not, however, crucial to the construction of a program, and are ignored by Version 2.0 of XASM11. They may be implemented in in future versions.

2.3 Operating Procedures

In this section, the job control statements needed to run the cross-assembler are described. A model catalogued procedure is presented along with instructions on how to use it.

The procedure designed for use at SLAC (where it is cataloged, and therefore need not be included in-line) is as follows:

```
1.  //CGXASM11    PROC
2.  //ASM    EXEC  PGM=XASM11,REGION=76K
3.  //*
4.  //*      STEP  ASM OF PROCEDURE CGXASM11
5.  //*
6.  //OBJECT  DD  DSN=&&OBJ,UNIT=SYSDA,
7.  //       DISP=(MOD,PASS),SPACE=(CYL,(1,1)),
8.  //       DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520)
9.  //STEPLIB DD DSN=WYL.CG.PUB.LOADMODS,DISP=SHR
10. //SYSPRINT DD SYSOUT=A
11. //SYSUDUMP DD SYSOUT=C
12. //WFILEDD  DD UNIT=SYSDA,SPACE=(CYL,(1,1))
13. //       PEND
```

The above procedure is available to users of SLAC WYLBUR as WYL.CG.PUB.LIB*CGXASM11 (on cat).

The function of each card is briefly described below.

Card 1 — Specifies the start of a procedure named CGXASM11.

Card 2 — Specifies the start of a step named ASM in which the program XASM11 is to be executed in a 76K region.

Cards 3-5 — Are comments.

Cards 6-8 — Define the object module output file whose DD name is OBJECT. It is given a temporary dataset- name of &&OBJ and allocated space on system direct access storage, one cylinder at a time. It will be passed on to later job steps but deleted at the end of the job. It is a card-image file, blocked to a half-track on a 2314 disk.

Card 9 — Specifies the program library to be used for this step. In particular, the program XASM11 is a member of the load module library WYL.CG.PUB.LOADMODS.

Card 10 — Defines the standard print file (output class A).

Card 11 — Defines the output file for a core dump in the event of an abnormal end, i.e., if the cross-assembler itself ABENDS. The dump will be printed in output class C.

Card 12 — Defines the work file which holds intermediate text records between passes of the cross-assembler. It is assigned space a cylinder at a time on system direct access storage.

Card 13 — Signifies the end of the in-line procedure CGXASM11.

To use this procedure, a job should have the following general form:

```
//  JOB  ...
//CGXASM11  PROC          }
      .                    }
  body of procedure         }  not needed at SLAC
      .                    }
//   PEND                  }
//stepname EXEC CGXASM11
//ASM.SYSIN   DD   *
      .
  PDP-11 source code
      .
//
```

To override any JCL parameters in the in-line procedure CGXASM11, cards should be inserted between the EXEC CGXASM11 and the SYSIN DD * cards, in alphabetical order by DDname. For example, to obtain an object deck on punched cards, the statement

```
//ASM.OBJECT    DD   SYSOUT=B,DCB=BLKSIZE=80
```

should be inserted. To suppress printing of the assembly listing or generation of the object module, it is recommended that the assembler control card options (see Section 2.1) be used rather than JCL.

If the DSNAME parameter on the OBJECT DD card is overridden so as to specify a permanent rather than a temporary data set, the JCL as written in the CGXASM11 procedure works both for the run which initially creates the data set and for subsequent runs. These later runs cause additional object modules to be concatenated to the current contents of the file.

The return code on normal completion of an XASM11 run is the number of errors flagged by the cross-assembler in the entire batch of source decks. Thus, a return code of 0 indicates a clean assembly.

The minimum region size in which Version 2.- of XASM11 will execute is 54k. This size can be easily reduced by decreasing the size of the symbol and cross-reference tables, or by reducing the size of IO buffers.

NOTE: In order to create a load module for the PDP-11, the cross-linker, XLINK11, must be used. An in-line procedure, CGX11, has been provided to perform an assembly and then to link the object modules together into a load module. The first step of this procedure is identical to the ASM step of CGXASM11. This procedure is cataloged at SLAC, and is also available to SLAC WYLBUR users as WYL.CG.PUB.LIB*CGX11 (on cat). For a listing of it, see Appendix B.

3. XLINK11: THE CROSS-LINKER

XLINK11 takes as input object decks produced by XASM11 and creates a load module suitable for running on the PDP-11. The handling of the object modules is generally similar to that of LINK-11, described in the LINK-11 Linker Programmer's Manual. A complete load map is produced, if desired, in a format close to the one produced by LINK-11. A number of additional features have been incorporated into XLINK11, and these include:

(1) output of the load module as a core image in a single record, as an alternate to the standard PDP-11 DOS load module format;

(2) listing of the load map contains all information in decimal as well as octal, side by side;

(3) A complete set of comprehensible error messages, with indication of the level of severity, and a statement of the total number of errors;

(4) a set of linker parameters which differ from the switches used by LINK-11, and which contains some new options (see Section 3.1).

XLINK11 can take as input the object decks produced on a given run of XASM11 and combine them, if desired, with previously assembled object decks to produce the load module.

3.1 Linker Parameters

Corresponding to the nine control switches used by LINK-11 (see Section 3.2.1 of the LINK-11 Linker Programmer's Manual) are a number of PARM field options in the LINK step EXEC card. Of the nine switches, three (/B, /T, and /TR) have been kept as PARM options, and are almost unchanged in function. The rest (/U, /OD, /TA, /CC, /E, and /L) are not necessary for XLINK11, and they have been dropped. Two new parameters (/MAP and /ALT) have been added, dealing with features not previously available. All XLINK11 parameters have defaults and may be omitted if the defaults are satisfactory. No blanks are permitted in the PARM field character string.

3.1.1 T/B - Top or Bottom Loading

The /T and /B parameters are identical in format and function to /T and /B switches of LINK-11. They are used to control the placement or relocation of the object program. The ways in which these parameters may be used are:

/T load downward, starting at top of available core (48108 is the default value for top of available core. This assumes a 24K-word machine, and leaves room at the top for the DOS Absolute and Boot Loaders.)

/B load upward, starting at bottom of available core (default address is 0).

/T:n set top of core at the octal number n, and load downward from that point.

/B:n set bottom of core at the octal number n, and load upward from that point.

If there is no /T or /B parameter in the PARM field, the default IS TO ASSUME /T.

During the creation of a load module, only one /T or /B parameter can be utilized. If more than one appears in the PARM field, the first is used, all others are ignored, and a warning message is printed.

If a top or bottom of core value, n, is specified it must be an unsigned, octal number. If it is not, an error message is printed and the and the run is aborted. If the octal number is odd, a warning is printed and the run is allowed to continue.

3.1.2 TR - Transfer Address Handling

This parameter, corresponding to the /TR switch of LINK-11, controls the specification of the Transfer Address (called Entry Point by IBM). The ways in which it may be used in the PARM field are:

/TR	take the first transfer address specified in an object module of the input file as the transfer address for this load module;
/TR:n	take the octal number n as the transfer address. If n is not an unsigned octal number, an error message is printed and the run is aborted. If n is an odd octal number, a warning is printed and the run is allowed to continue with the number specified. Note that this corresponds closely to the way LINK-11 handles such errors (the PDP-11 will cease execution when it finds an odd transfer address after loading).
/TR:XXXXXX	take the global symbol 'XXXXXX' as the transfer address for this load module. If the name specified is longer than six characters, the first six are used and a warning is printed. If the name is not a defined global symbol, an error message is printed and the run is aborted.

If no /TR parameter appears in the PARM field, the default is to assume /TR, i.e., take the first transfer address (specified on a .END card) found in the input object file. If no transfer address was specified in any of the input object modules, and none was explicitly defined in the PARM field, a warning is printed and the default transfer address value 1 is used.

If a /TR parameter with an explicit transfer address specification ('/TR:n' or '/TR:XXXXXX') appears in the PARM field, a /TR parameter appearing after it will be ignored, and a warning will be printed to indicate multiply defined parameters. A /TR parameter with no explicit transfer address ('/TR') will not block reading of later /TR parameters; it is merely ignored, since it corresponds to the default condition.

Note: All /TR parameters, and transfer addresses specified in the input object modules, are handled identically whether the output load module is in regular or core-image format (see Section 3.1.4). In the core-image case, however, the transfer address is not included in the load module, and it must be specified by the PDP-11 operator when execution of the loaded program is desired.

3.1.3 MAP/NOMAP - Optional Listing

These two parameters, not present in LINK-11, control the printing of the load module map. The formats in the PARM field are:

/MAP print a complete load module map;

/NOMAP suppress printing of the load module map.

If neither parameter is present in the PARM field, the default /MAP is assumed. If more than one such parameter appears in the PARM field, the first is used, all others are ignored and a warning is printed.

3.1.4 ALT - Alternate Output Routine

This parameter controls an option that has no counterpart in LINK-11, the production of the load module in core-image format as an alternate to standard PDP-11 DOS load module format. It is used in the PARM field as follows:

/ALT (may be abbreviated to /A) causes the load module to be written out as a single record, containing a partial core image of the length required by this run of XLINK11.

If the /ALT parameter does not appear in the PARM field, a load module in standard PDP-11 DOS format is produced.

When the /ALT parameter is specified, a PDP-11 core-image area of 64K bytes is kept internally by XLINK11. During the run, the PDP-11 machine code is stored into this area at the appropriate locations, and the upper and lower limit addresses are determined. At the end of the run, a partial core-image is written out on 9-track tape, to be used for loading the PDP-11. This partial core-image begins at the lower limit address, and the number of bytes it contains is the smallest multiple of 256 bytes which includes the contiguous area from the

lower to the upper limit address. Thus the load module may "overhang" the highest location actually used by as much as 254 bytes. The /T parameter to the linker does not protect against this happening. (For example, the parameter '/T:40000' may produce a load module tape which would alter the contents of PDP-11 core memory up to octal address 40376.)

The load module limits and size are printed out at the end of the load module map. Since code in ASECT's affects the load module limits, and since the size must be a multiple of 256 bytes, the load module limits and size will generally differ from the program limits and size printed out in the load module map.

The transfer address is not included in the load module, but it is printed out in the listing. It is up to the PDP-11 operator to keep track of it for each tape he loads.

3.2 Operating Procedures

This section describes the job control statements needed to run the cross-linker. A model cataloged procedure is presented, with a detailed description and instructions on how to use it.

The procedure designed for use at SLAC (where it is cataloged, and need not be included in-line) is as follows:

```
 1.  //CGXLNK11    PROC
 2.  //LINK  EXEC  PGM=XLINK11,REGION=150K
 3.  //*
 4.  //*     STEP  LINK OF PROCEDURE CGXLNK11
 5.  //*
 6.  //LINKIN  DD  DDNAME=SYSIN
 7.  //LMOD    DD  DSN=&&LNK,UNIT=SYSDA,DISP=(NEW,DELETE),
 8.  //        SPACE=(TRK,(1,1))
 9.  //STEPLIB DD  DSN=WYL.CG.PUB.LOADMODS,DISP=SHR
10.  //SYSPRINT DD SYSOUT=A
11.  //SYSUDUMP DD SYSOUT=C
12.  //        PEND
```

The above procedure (in addition to being cataloged at SLAC) is available to users of SLAC WYLBUR as WYL.CG.PUB.LIB#CGXLNK11 (on cat). The function of each card is described below.

Card 1	Specifies the start of a procedure named CGXLNK11.
Card 2	Specifies the start of a step named LINK in which the program XLINK11 is to be executed in a 150K region.
Cards 3-5	Are comments.

Card 6	Specifies that the object module input file (LINKIN), will also be called SYSIN, and may be input as a card object deck.
Cards 7-8	Defines the load module output file (LMOD) to be a temporary file named &&LNK, which is allocated space on system direct access storage, one track at a time. The DCB parameter should not be specified since it depends on whether regular or alternate output is desired (see 3.1.4), and the appropriate values are assigned by XLINK11 internally.
Card 9	Specifies the program library to be used for this step. In particular, the program XLINK11 is a member of the load module library WYL.CG.PUB.LCADMODS.
Card 10	Defines the standard print file (output class A).
Card 11	Defines the output file (class C) for a core dump in the event of an abnormal end; that is, if the cross-linker itself detects an abnormal situation and ABENDS.
Card 12	Signifies the end of the in-line procedure CGXLNK11.

To use this procedure, a job should have the following general form:

```
//    JOB   ...
//CGXLNK11  PROC          }
      .                    }
  body of procedure         }  not needed at SLAC
      .                    }
//          PEND           }
//stepname  EXEC   CGXLNK11,PARM.LINK='...'
//LINK.SYSIN  DD  *
      .
  input object module deck
      .
//
```

Note: In addition to the procedures CGXASM11 and CGXLNK11, a third procedure named CGX11 is cataloged at SLAC, which is a two-step procedure that combines the cross-assembly and cross-linking functions. The first step corresponds to step ASM of procedure CGXASM11, and the second step corresponds to step LINK of procedure CGXLNK11. No step or DDnames have been changed, and overrides are handled in the usual manner. In CGX11, the output object module is automatically passed to the second step for linking. A listing of CGXASM11, CGXLNK11,and CGX11 is provided in Appendix B.

To oVerride any JCL parameters in the procedure CGXLNK11, cards should be inserted after the EXEC CGXLNK11 card in alphabetical order by DDname. For example, if one wishes to save the load module for transfer to the PDP-11, calling it WYL.CG.SZS.MODULE on WYL009, the following cards

```
//LINK.LMOD   DD   DSN=WYL.CG.SZS.MODULE,UNIT=DISK,
//            VOL=SER=WYL009,DISP=(NEW,KEEP)
```

should be inserted.

Using job control statements, object modules created by different assemblies can be linked together into a single load module. For example, if it is desired to link together the object module in the current runstream and the modules WYL.CG.SZS.MOD1 and WYL.CG.SZS.MOD2, created in previous runs, which are both cataloged data sets, the cards

```
//LINK.LINKIN   DD   DSN=WYL.CG.SZS.MOD1,DISP=SHR
//              DD   DSN=WYL.CG.SZS.MOD2,DISP=SHR
//LINK.SYSIN    DD   *
        .
  Input Object Module Deck
        .
//
```

may be used. As another example, if the procedure CGX11 is used and it is desired to link the object module produced by the cross-assembly in the first step with some previously produced object module, the card

```
//LINK.SYSIN    DD   DSN=WYL.CG.SZS.MOD1,DISP=SHR
```

may be used.

There are two formats for the output load module produced by XLINK11. One is designed to closely parallel the PDP11 DOS Monitor Loader load module format, as described in PDP11 documentation. Each block of code is preceded by a standard 6-byte header, and a checksum character is appended to the end of the block. The transfer address is contained in the last block, containing only the transfer address and no code. The alternate output routine, described in Section 3.1.4, is in core-image format.

In the present version, the maximum size load module that can be produced is 64K bytes. There is no reason why this size cannot be increased if the need arises.* The current version of XLINK11 will run in a 150K region.

*To do this, the value of MAXCORE in the subroutine SYMTAB should be changed, and more space for a core image should be provided in the IMAGE subroutine.

APPENDIX A: Sample Run

This appendix contains a brief sample run of both the cross-assembler and cross-linker, using the cataloged procedure CGX11. There is no need of an assembler control card since we desire a full listing and a binary object module produced by the cross-assembler. For illustrative purposes, the following linker parameters were specified:

/TR:CURR	transfer address to be the location assigned to the global symbol CURR (a CSECT name, so transfer address is the start of that CSECT).
/B:20000	will load upward, starting at location 20000 octal.
/ALT	use alternate output routine, creating the load module in core-image format.

Since neither /MAP nor /NOMAP are specified, the default is to provide a full load module map.

The input runstream is:

```
//      JOB       ...
//      EXEC      CGX11,PARM.LINK='//B:20000,//TR:CURR,//ALT'
//ASM.SYSIN       DD   *
       .HEAD      PDP-11 CROSS-ASSEMBLER XASM11 -- TEST RUN
       .
       PDP-11 source deck
       .
//
```

The next several pages contain the complete computer listing produced by this run.

```
 ISV40 JOB ORIGIN FROM GROUP=LOCAL    , DSP=IJP, DEVICE=SYA      , 9F9
//SZSCG719 JOB SZS$CG,CLASS=E                                              1.
//*MAIN HOLD=OUTPUT
// EXEC  CGX11,PARM.LINK='/TR:CURR,/B:20000,/ALT'                          2.
//ASM.SYSIN  DD  *                                                        3.
/*EOF


 LOCATE' 5672WYL.CG.PUB.LOADMODS
 AL5672OE001/WYL0060C04
 LOCATE' 5672WYL.CG.PUB.LOADMODS
 AL5672OE001/WYL0060C04


AMDS01 JOB 5672 (SZSCG719) IN SETUP ON MAIN=SYA
AMDS02          STEPLIB  USING D  WYL006 ON 233
 SZSCG719 IEF403I SZSCG719 STARTED     TIME=16.43.43
 SZSCG719 IEF234E D 90D,ASP90D
*SZSCG719*79 IECASPO 916 IS SZSCG719            ASP      ASPI0001
*SZSCG719*80 IECASPO 90D IS SZSCG719 A          ASP      SYSPRINT
 SZSCG719 IEC202E K 916,015672,NL,SZSCG719,ASM
 SZSCG719 IEF234E D 90D,ASP90D
*SZSCG719*83 IECASPO 90D IS SZSCG719 A          LINK     SYSPRINT
 SZSCG719 IEF404I SZSCG719 ENDED       TIME=16.44.02
//SZSCG719 JOB SZS$CG,CLASS=E                                              1.
// EXEC  CGX11,PARM.LINK='/TR:CURR,/B:20000,/ALT'                          2.
XXCGX11         PROC                                                    00001000
XXASM    EXEC   PGM=XASM11,REGION=76K                                    00002000
***                                                                      00003000
***      STEP   ASM   OF PROCEDURE CGX11                                 00004000
***      REFER ANY PROBLEMS ENCOUNTERED IN USING THIS PDP-11             00005000
***      CROSS-ASSEMBLER TO S. STEPPEL, SLAC COMPUTATION GROUP.          00006000
***                                                                      00007000
XXOBJECT   DD  DSN=&&OBJ,UNIT=SYSDA,DISP=(MOD,PASS),                    00008000
XX       SPACE=(CYL,(1,1)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520)         00009000
XXSTEPLIB  DD  DSN=WYL.CG.PUB.LOADMODS,DISP=SHR                          00010000
XXSYSPRINT DD  SYSOUT=A                                                  00011000
XXSYSUDUMP DD  SYSOUT=C                                                  00012000
XXWFILEDD  DD  UNIT=SYSDA,SPACE=(CYL,(1,1))                              00013000
//ASM.SYSIN DD UNIT=(CTC,,DEFER),DSNAME=&&ASPI0001,                    *  3.
// DISP=(OLD,DELETE),VOL=SER=015672,DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
IEF236I ALLOC. FOR SZSCG719 ASM
IEF237I 751   ALLOCATED TO OBJECT
IEF237I 233   ALLOCATED TO STEPLIB
IEF237I 90D   ALLOCATED TO SYSPRINT
IEF237I 915   ALLOCATED TO SYSUDUMP
IEF237I 751   ALLOCATED TO WFILEDD
IEF237I 916   ALLOCATED TO SYSIN
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF285I   SYS74228.T164311.RV001.SZSCG719.OBJ          PASSED
IEF285I   VOL SER NOS= WORK02.
IEF285I   WYL.CG.PUB.LOADMODS                          KEPT
IEF285I   VOL SER NOS= WYL006.
IEF285I   SYS74228.T164311.RV001.SZSCG719.ASPCA001     DELETED
IEF285I   VOL SER NOS= ASP90D.
IEF285I   SYS74228.T164311.RV001.SZSCG719.R0001150     DELETED
IEF285I   VOL SER NOS= WORK02.
IEF285I   SYS74228.T164311.RV001.SZSCG719.ASPI0001     DELETED
IEF285I   VOL SER NOS= 015672.
```

```
IEF373I STEP /ASM      / START 74228.1643
IEF374I STEP /ASM      / STOP  74228.1643 CPU   0MIN 00.25SEC STOR VIRT 128K
SMF001I   STEP  ASM          STEP NUMBER=      1                                                    RETURN=          0 DEC
SMF002I   DATE= 08/16/74     STRT=  16:43:43.89   STOP=  16:43:53.21   E.T.=        0:09.32    CPU=          0:00.25
SMF003I   CPU ID=  168-A     SYSTEM=    VS2 01.6   CORE REC=      76K   CORE USED=     128K    CHG FACTOR=     0.67
SMF004I   CORE=  VIRTUAL     PAGE INS=         0   PGE OUTS=         0
SMF005I   I/O COUNTS         3330/751=         1   2314/233=         8   CTC./90D=        128   CTC./915=         0
SMF005I   I/O COUNTS         3330/751=         5   CTC./916=        63
SMF006I   I/O TOTALS         OTHER=          191   7 TRK=            0   9 TRK=             0   DASD=            14
SMF007I   STEP CHARGES       OTHER=         0.64   7 TRK=         0.00   9 TRK=          0.00   DASD=          0.37
SMF008I   STEP CHARGES                                                   CPU=            0.17   TOTAL=         1.17
XXLINK   EXEC  PGM=XLINK11,REGION=150K                                                            00014000
***                                                                                               00015000
***      STEP  LINK  OF PROCEDURE CGX11                                                           00016000
***      REFER ANY PROBLEMS ENCOUNTERED IN USING THIS PDP-11                                      00017000
***      CROSS-LINKER TO S. STEPPEL, SLAC COMPUTATION GROUP.                                      00018000
***                                                                                               00019000
XXLINKIN   DD  DSN=*.ASM.OBJECT,DISP=(OLD,PASS)                                                   00020000
XX         DD  DDNAME=SYSIN                                                                       00021000
XXLMOD     DD  DSN=&&LNK,UNIT=SYSDA,DISP=(NEW,DELETE),                                            00022000
XX       SPACE=(TRK,(1,1))                                                                        00023000
XXSTEPLIB  DD  DSN=WYL.CG.PUB.LOADMODS,DISP=SHR                                                   00024000
XXSYSPRINT DD  SYSOUT=A                                                                           00025000
XXSYSUDUMP DD  SYSOUT=C                                                                           00026000
//
IEF236I ALLOC. FOR SZSCG719 LINK
IEF237I 751   ALLOCATED TO LINKIN
IEF237I 751   ALLOCATED TO LMOD
IEF237I 233   ALLOCATED TO STEPLIB
IEF237I 90D   ALLOCATED TO SYSPRINT
IEF237I 915   ALLOCATED TO SYSUDUMP
IEF142I - STEP WAS EXECUTED - COND CODE 0004
IEF285I   SYS74228.T164311.RV001.SZSCG719.OBJ          PASSED
IEF285I   VOL SER NOS= WORK02.
IEF285I   SYS74228.T164311.RV001.SZSCG719.LNK          DELETED
IEF285I   VOL SER NOS= WORK02.
IEF285I   WYL.CG.PUB.LOADMODS                          KEPT
IEF285I   VOL SER NOS= WYL006.
IEF285I   SYS74228.T164311.RV001.SZSCG719.ASPCA003     DELETED
IEF285I   VOL SER NOS= ASP90D.
IEF373I STEP /LINK     / START 74228.1643
IEF374I STEP /LINK     / STOP  74228.1644 CPU   0MIN 00.12SEC STOR VIRT 192K
SMF001I   STEP  LINK         STEP NUMBER=      2                                                    RETURN=          4 DEC
SMF002I   DATE= 08/16/74     STRT=  16:43:53.39   STOP=  16:44:01.68   E.T.=        0:08.29    CPU=          0:00.12
SMF003I   CPU ID=  168-A     SYSTEM=    VS2 01.6   CORE REC=     150K   CORE USED=     192K    CHG FACTOR=     1.00
SMF004I   CORE=  VIRTUAL     PAGE INS=         0   PGE OUTS=         0
SMF005I   I/O COUNTS         3330/751=         3   OTHR/000=         0   3330/751=          7   2314/233=          7
SMF005I   I/O COUNTS         CTC./90D=        31   CTC./915=         0
SMF006I   I/O TOTALS         OTHER=           31   7 TRK=            0   9 TRK=             0   DASD=            17
SMF007I   STEP CHARGES       OTHER=         0.15   7 TRK=         0.00   9 TRK=          0.00   DASD=          0.68
SMF008I   STEP CHARGES                                                   CPU=            0.12   TOTAL=         0.92
IEF285I   SYS74228.T164311.RV001.SZSCG719.OBJ          DELETED
IEF285I   VOL SER NOS= WORK02.
IEF375I  JOB /SZSCG719/ START 74228.1643
IEF376I  JOB /SZSCG719/ STOP  74228.1644 CPU   0MIN 00.37SEC
AMCS09 JOB 5672 (SZSCG719) IN BREAKDOWN
```

COMPUTATION GROUP, STANFORD LINEAR ACCELERATOR CENTER

SYMBOL TABLE AND CROSS-REFERENCE LISTING

SYMBOL	VALUE	FLAGS	DEFN	REFERENCES
.	000140	002	****	27
ABS	000050		31	38 39 40 41 42 43
EXT	= ******	G	****	56 57 58 59 60 61
RELC	000024R	002	43	44 45 46 47 48 49
RELD	000014R		24	50 51 52 53 54 55
R3	=%000003		10	18 19 20 21 22 23 24 40 41 46 47 52 53 58 59

LIST OF NAMED CSECTS

CURR	000140	002

```
                                .TITLE TEST4
                            ;
                            ;       THIS TEST DECK IS DESIGNED TO EXERCISE THE OPERAND
                            ;       ROUTINE IN THE CROSS-ASSEMBLER, TO SEE THAT THE PROPER
                            ;       ADDRESSING MODES AND LINKER RLD COMMANDS ARE GENERATED
                            ;       FOR EVERY VALID COMBINATION OF ARGUMENT TYPES...
                            ;
                                .GLOBL EXT
            000003      R3  =  %3
                            ;
                            ;  CURRENT SECTION IS THE UNNAMED RELOCATABLE CSECT
                            ;
                            ;                               ADDR   FLG
                            ;                               MODE   CODE
                            ;                               ----   ----
                            ;
000000 005003               CLR    R3          ; 03    0
000002 005013               CLR    @R3         ; 13    0
000004 005013               CLR    (R3)        ; 13    0
000006 005023               CLR    (R3)+       ; 23    0
000010 005033               CLR    @(R3)+      ; 33    0
000012 005043               CLR    -(R3)       ; 43    0
000014 005053  RELO:        CLR    @-(R3)      ; 53    0
                            ;
                                .ASECT
            000050      . = . + 50
                            ;
                            ;  CURRENT SECTION IS NOW THE ABSOLUTE SECTION
                            ;
000050 005067  ABS:         CLR    76          ; 67    0    ASSEMBLER FIXUP
       000022
000054 005077               CLR    @76         ; 77    0    ASSEMBLER FIXUP
       000016
                            ;
                                .CSECT CURR
                            ;
                            ;  CURRENT SECTION IS NOW THE RELOCATABLE CSECT NAMED "CURR"
                            ;
000000 005027               CLR    #ABS        ; 27    0
       000050
000004 005037               CLR    @#ABS       ; 37    0
       000050
000010 005063               CLR    ABS(R3)     ; 63    0
       000050
000014 005073               CLR    @ABS(R3)    ; 73    0
       000050
000020 005067'              CLR    ABS         ; 67    3
       000050
000024 005077'  RELC:       CLR    @ABS        ; 77    3
       000050
000030 005027'              CLR    #RELC       ; 27    1
       000024
000034 005037'              CLR    @#RELC      ; 37    1
       000024
000040 005063'              CLR    RELC(R3)    ; 63    1
```

```
                000024
47      000044  005073'         CLR     @RELC(R3) ; 73    1                                   50.
                000024
48      000050  005067          CLR     RELC      ; 67    0   ASSEMBLER FIXUP                 51.
                177750
49      000054  005077          CLR     @RELC     ; 77    0   ASSEMBLER FIXUP                 52.
                177744
50      000060  005027'         CLR     #RELC     ; 27    2                                   53.
                000014
51      000064  005037'         CLR     @#RELC    ; 37    2                                   54.
                000014
52      000070  005063'         CLR     RELC(R3)  ; 63    2                                   55.
                000014
53      000074  005073'         CLR     @RELC(R3) ; 73    2                                   56.
                000014
54      000100  005067'         CLR     RELC      ; 67    4                                   57.
                000014
55      000104  005077'         CLR     @RELC     ; 77    4                                   58.
                000014
56      000110  005027'         CLR     #EXT      ; 27    2                                   59.
                000000
57      000114  005037'         CLR     @#EXT     ; 37    2                                   60.
                000000
58      000120  005063'         CLR     EXT(R3)   ; 63    2                                   61.
                000000
59      000124  005073'         CLR     @EXT(R3)  ; 73    2                                   62.
                000000
60      000130  005067'         CLR     EXT       ; 67    4                                   63.
                000000
61      000134  005077'         CLR     @EXT      ; 77    4                                   64.
                000000
62                      .END                                                                    65.
```

NO ERRORS IN ABOVE ASSEMBLY

ERROR MESSAGE EXPLANATION

ERROR FLAG	MEANING
A	ADDRESSING ERROR
B	BOUNDING ERROR
D	DOUBLY DEFINED SYMBOL REFERENCED
I	ILLEGAL CHARACTER DETECTED
M	MULTIPLE DEFINITION OF LABEL
N	' ' MISSING IN DECIMAL NUMBER
P	LABEL DEFN DIFFERS BETWEEN PASSES
Q	QUESTIONABLE SYNTAX
R	INVALID REFERENCE TO REGISTER
T	TRUNCATION ERROR
U	UNDEFINED SYMBOL REFERENCED

			OCTAL		DECIMAL	
MODULE	SECTION	ENTRY	ADDRESS	SIZE	ADDRESS	SIZE
TEST4	ABS		000000	000000	00000	00000
	UNN		020000	000016	08192	00014
	CURR		020016	000140	08206	00096

****

****

	OCTAL		DECIMAL	
PROGRAM LIMITS:	020000	020154	08192	08300
PROGRAM SIZE:	000156		00110	

** WARNING -- UNDEFINED EXTERNAL REFERENCE -- EXT AT LOCATION 020130 OCTAL

** WARNING -- UNDEFINED EXTERNAL REFERENCE -- EXT AT LOCATION 020134 OCTAL

** WARNING -- UNDEFINED EXTERNAL REFERENCE -- EXT AT LOCATION 020140 OCTAL

** WARNING -- UNDEFINED EXTERNAL REFERENCE -- EXT AT LOCATION 020144 OCTAL

** WARNING -- UNDEFINED EXTERNAL REFERENCE -- EXT AT LOCATION 020150 OCTAL

** WARNING -- UNDEFINED EXTERNAL REFERENCE -- EXT AT LOCATION 020154 OCTAL

TRANSFER ADDRESS: 020016 08206

CORE-IMAGE LOAD MODULE:

	OCTAL		DECIMAL	
MODULE LIMITS:	000050	020450	00040	08488
MODULE SIZE:	020400		08448	

***** END OF LINKER RUN --- NUMBER OF WARNINGS = 6 *****

APPENDIX B: In-Line Procedures

The following three procedures, designed for use at SLAC (where they are cataloged and need not be placed in the runstream), may be put in-line and modified directly.

1. CGX11 - performs one run of the cross-assembler and passes the object module produced to the second step, where the cross- linker creates a load module. It is essentially a combination of the CGASM11 and CGXLNK11 procedures. For SLAC WYLBUR users, it is available as WYL.CG.PUB.LIB#CGX11 on CAT.

```
//CGX11         PROC
//ASM     EXEC  PGM=XASM11,REGION=76K
//*
//*       STEP  ASM  OF PROCEDURE CGX11
//*
//OBJECT    DD  DSN=&&OBJ,UNIT=SYSDA,DISP=(MOD,PASS),
//        SPACE=(CYL,(1,1)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520)
//STEPLIB   DD  DSN=WYL.CG.PUB.LOADMODS,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSUDUMP DD   SYSOUT=C
//WFILEDD   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//LINK    EXEC  PGM=XLINK11,REGION=150K
//*
//*       STEP  LINK  OF PROCEDURE CGX11
//*
//LINKIN    DD  DSN=*.ASM.OBJECT,DISP=(OLD,PASS)
//          DD  DDNAME=SYSIN
//LMOD      DD  DSN=&&LNK,UNIT=SYSDA,DISP=(NEW,DELETE),
//        SPACE=(TRK,(1,1))
//STEPLIB   DD  DSN=WYL.CG.PUB.LOADMODS,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSUDUMP DD   SYSOUT=C
//         PEND
```

2. CGXASM11 - performs one run of the cross-assembler. It is available as WYL.CG.PUB.LIB#CGXASM11 on CAT.

```
//CGXASM11      PROC
//ASM     EXEC  PGM=XASM11,REGION=76K
//*
//*       STEP  ASM  OF PROCEDURE CGXASM11
//*
//OBJECT    DD  DSN=&&OBJ,UNIT=SYSDA,DISP=(MOD,PASS),
//        SPACE=(CYL,(1,1)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520)
//STEPLIB   DD  DSN=WYL.CG.PUB.LOADMODS,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSUDUMP DD   SYSOUT=C
//WFILEDD   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//         PEND
```

3. CGXLNK11 - performs one run of the cross-linker to create a load module for the PDP-11. It is available as WYL.CG.PUB.LIB#CGXLNK11 on CAT.

```
//CGXLNK11     PROC
//LINK    EXEC  PGM=XLINK11,REGION=150K
//*
//*       STEP  LINK  OF PROCEDURE CGXLNK11
//*       REFER ANY PROBLEMS ENCOUNTERED IN USING THIS PDP-11
//*       CROSS-LINKER TO S. STEPPEL, SLAC COMPUTATION GROUP.
//*
//LINKIN    DD  DDNAME=SYSIN
//LMOD      DD  DSN=&&LNK,UNIT=SYSDA,DISP=(NEW,DELETE),
//         SPACE=(TRK,(1,1))
//STEPLIB   DD  DSN=WYL.CG.PUB.LOADMODS,DISP=SHR
//SYSPRINT  DD  SYSOUT=A
//SYSUDUMP  DD  SYSOUT=C
//         PEND
```