11. PROGRAM DEVELOPMENT AND TESTING

11.1 Introduction

The source data for a CREDIT program may be entered to the system under DOS by one of three media:

- Cassette (standard assignment)
- Punched cards
- Console typewriter

The sequence of processes necessary to develop a program is shown on foil 42, and all these processes take place under DOS.

The testing of programs, however, takes place under control of the TOSS Monitor, as has been mentioned before.

The processors required for the development of a credit application are described in this section.

11.2 CREDIT Translator

The CREDIT Translator is called into execution by the TRA command, and performs the following actions:

- Each module is processed separately by the Translator.
- This produces an Intermediate Object Code module, which must be made permanent by the KPF command, unless it contains the DDIV for the entire application.
- The instructions in these modules use a byte-oriented addressing system, and this code is printed on the output listings at the left hand side.
- Each module may contain references to:
  - Labels in the same module
  - Literals in the same module
  - Labels in other CREDIT modules
  - Assembler application modules
  - Assembler system routines

The first type of reference is satisfied by the CREDIT Translator.

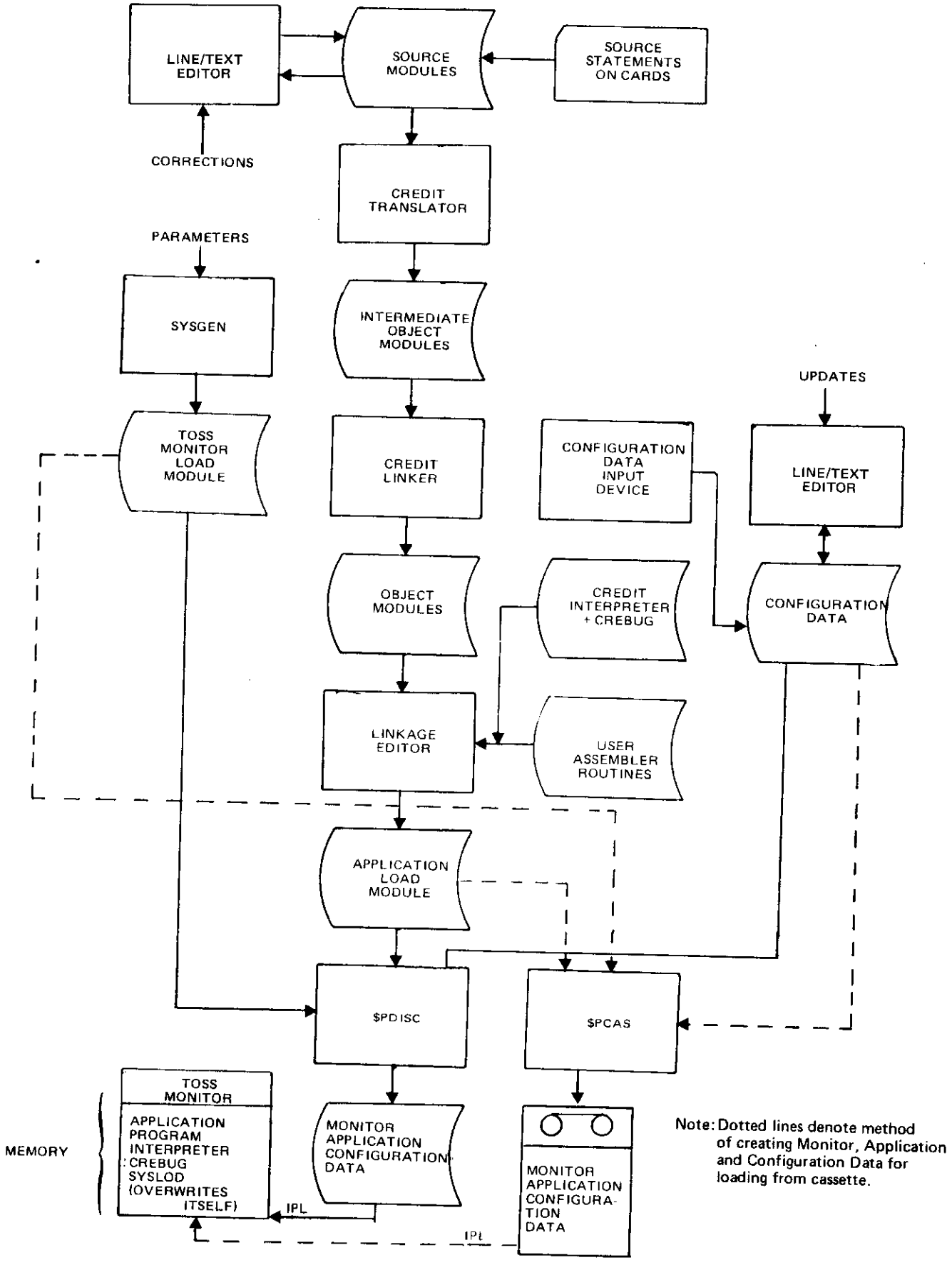
It is recommended that all temporary files be scratched before running the Translator.

11.3 CREDIT Linker

The resulting object modules are then processed by the CREDIT linker. This is called into execution by the TLK command, and performs the following functions:

- Solves references to the second two types of reference described above.
- Links together the object modules to form word-oriented object modules.

Keyword	Page in Manual
TLK	M11 6.12.43
TRA	M11 6.12.44
KPF	M11 6.12.12


LINE/TEXT EDITOR
SOURCE MODULES
SOURCE STATEMENTS ON CARDS
CORRECTIONS
CREDIT TRANSLATOR
PARAMETERS
SYSGEN
INTERMEDIATE OBJECT MODULES
UPDATES
TOSS MONITOR LOAD MODULE
CREDIT LINKER
CONFIGURATION DATA INPUT DEVICE
LINE/TEXT EDITOR
OBJECT MODULES
CREDIT INTERPRETER + CREBUG
CONFIGURATION DATA
LINKAGE EDITOR
USER ASSEMBLER ROUTINES
APPLICATION LOAD MODULE
$PDISC
$PCAS
MEMORY
TOSS MONITOR
APPLICATION PROGRAM INTERPRETER :CREBUG SYSLOD (OVERWRITES ITSELF)
MONITOR APPLICATION CONFIGURATION DATA
MONITOR APPLICATION CONFIGURA-TION DATA
IPL
IPL
Note: Dotted lines denote method of creating Monitor, Application and Configuration Data for loading from cassette.

11.3.1 Segmentation

It is possible for an application to be split up into a number of 'segments'. There is always one segment, number 00, which contains the DDIV, Interpreter, and Assembler subroutines. This segment is always resident in memory during application running. The remaining segments may be disk resident, or memory resident, depending on a) the wishes of the programmer, and b) the size of memory of the machine in use. If no action is taken by the programmer on segmentation, then the result of the TRA and TLK processing described above will be an unsegmented program, i.e. segment 00 will contain the entire application. If segmenting is required, the segments are built up by use of the INC and NOD commands.

11.4 Linkage Editor

The output modules from the CREDIT linker are processed by the Linkage Editor. This is called into execution by the LKE command, and perfoms the following functions:

- Links together all the modules from the Linker to form one application load module.
- Solves the remaining references between modules and system routines.
- Includes Assembler application routines if required.
- Includes the CREDIT Interpreter.
- Includes the CREDIT configuration program.
- Includes the CREDIT Debugger, unless explicitly excluded.

11.5 CREDIT Interpreter

The load module created by the above processors can not be executed directly by the machine, but must be in a format suitable for execution under the control of the TOSS Monitor. The way in which the moving from DOS to TOSS is explained below. Once the load module is in memory, it must be interpreted by the CREDIT Interpreter, that is the functions in the CREDIT intermediate object code are called into execution by the Interpreter by means of calls to Assembler system routines.

11.6 CREDIT Configurator

After system configuration, which is covered later, the CREDIT configurator takes control: this sets up all the required workblocks, stacks, data set buffers, and task control areas that are required by the tasks to be executed.

Following this, control is handed to the Interpreter, and the program commences execution.

Keyword	Page in manual
INC	M11 6.12.11
NOD	M11 6.12.20

11.7 CREDIT Debugger

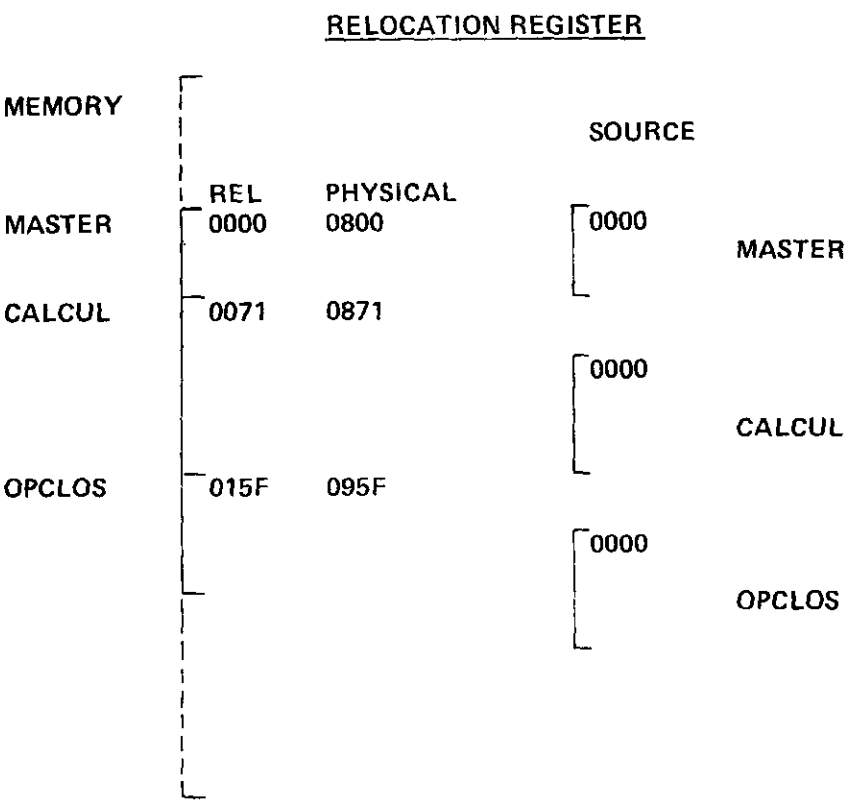
The CREDIT debugging program (CREBUG) is an interactive diagnostic task which runs under the control of the TOSS monitor. It is used to control execution of the application in the following way:-

- Traps may be set
- Variables may be examined and modified
- Trace may be turned off

etc.

CREBUG is specified as a special task at SYSGEN time; it runs at a priority leval higher than that of the application, to enable the application task to be interupted, it also has a special task identifier TB.
The programmer can use the Translator and linkage lists to set traps, verify the contents of data items change elements in the picture pool etc.

Keyword	Page in manual
Debugger	M04 4.1.1




```
REG.D          CONTAINS 0800

1Q             /0000    0071,D
REG.1          CONTAINS 0871

2Q             /0000    015F,D
REG.2          CONTAINS 095D
```

LOAD MAP

MODULE	LOC	ERROR
MASTER	0000	
CALCUL	0071	
OPCLOS	015F	
OPOPEN	0192	
READN	01E3	
SYCLOS	0254	
SYSOPN	0283	
BOOK	02E8	

MASTER	0000	APPLICATION PROGRAM
CALCUL	0071	
OPCLOS	015F	
OPOPEN	0192	
READN	01E3	
SYCLOS	0254	
SYSOPN	0283	
BOOK	02E8	

11.8 Line Editor and Text Editor

If errors do occur while testing, it is possible to correct some of them via the debugger. However, at some stage the source code has to be corrected or updated. This can be done easily by using one of the DOS processors, the Line Editor/Text Editor. By the use of various commands available, the source code can have lines amended, inserted and deleted.

Keyword	Page in manual
Line editor	M11 8.2.1
Text editor	M11 Z.1.1

11.9 CREDIT Translator Listings

During the processing of the CREDIT Translator, a listing is produced (unless specifically suppressed), containing:

- CREDIT Source statements
- Intermediate Object code
- Error messages

The heading of the listing contains the Release Number of the Translator in use, and the date, the heading Data Division or Procedure Division, as appropriate, and the name of the module from the IDENT statement. Reading from left to right across the page, the following appear:

LOC = Location Counter

This is a four digit hexadecimal counter, which is increased by one every time a byte of intermediate object code is produced, for the Procedure Division only. This counter is used when using the CREDIT Debugger, to display and/or amend the contents of memory. For the data division, the counter is the index value of items within workblocks, where the first digit is the workblock number and the second the number of the item in the workblock. Thus 32 = Workblock 3, Item 2 (Workblocks start at 1, Items 0). Note that Boolean data items, for which one word is reserved in each block, the second number is the bit within the word at the start of the block. These numbers are also used for DSET statements, where they are again an index value of the DSET within the task.

OC OPERANDS

These are the Operation Code and the Operands generated from the CREDIT source, in Intermediate Object Code, they are printed at the left hand side of the translator listing.

Where an operand is shown as LL, this is a reference to the literal pool, which is filled in by the CREDIT Linker later. This code is present for Literals (=X'6142')
Format Lists (FRMT)
Keytables (KTAB)

Where an operand is shown as RR, this is a reference to a subroutine within the same module, which is also filled in by the Linker.

Where an operand is shown as XX, this is a reference to an external routine, also filled in by the Linker.

STMT = Statement

This is a four digit decimal line number, and is the line number used when editing with the Line Editor on the source module.

LABEL OPCOD OPERANDS COMMENT

These are self-explanatory.

C = Continuation

This shows that the statement is continued on a new line, as it was too long for one source line.

ERROR MESSAGES

If the Translator detects an error in the source code, it prints an explanatory message under the statement in error, together with an asterisk to indicate the part of the statement that is incorrect.

At the end of the listing, the messages PROGRAM LENGTH and ERROR are printed. The program length is the hexadecimal number of bytes contained in the module, and the error count is a decimal count of the number of errors detected in the module.

In addition, two tables are listed at the end:

- Data item name table, showing all the data item names used in the module, with a U printed by them if they are not referenced within this module.
- Procedure label table, showing the labels (names) of all the PROC statements referred to in the module.

```
• CREDIT TRANSLATOR  REL 4.1 790523 •  DATA DIVISION              • IDENT EX7    -- • DATE 790920 • PAGE 0001 •
```

```
      IX                    LINE  LABEL     OPCODE    OPERANDS                    COMMENT
                            0000            IDENT     EX7
                            0001            DDIV
                            0002            TERM      A1
                            0003            CWB       CB1
      10                    0004  DSKB      DSET      FC=20,DEV=KB
      11                    0005  DSDY      DSET      FC=40,DEV=DY,BUFL=100
                            0006            START     GOA1
      1                     0007  CB1       BLK
      10                    0008  INLEN     BIN
      11                    0009  IX1       BIN
      12                    0010  AMNT      BCD       11D
      13                    0011  ACCNT     STRG      9
      14                    0012  IBUF      STRG      11
```

```
• CREDIT TRANSLATOR  REL 4.1 790523 •  PROCEDURE DIVISION          • IDENT EX7    -- • DATE 790920 • PAGE 0002

LOC  OC OPERANDS              LINE  LABEL    OPCODE  OPERANDS                    COMMENT

                              0013          PDIV
                              0014          ENTRY   GOA1
                              0015  KTB1    KTAB    X'23'
0000 01 10 LL                 0016  GOA1    MOVE    INLEN.=W'10'
0003 30 XX DD 14 KK 10 11     0017          NKI     DSKB.IBUF.KTB1.INLEN.IX1
000A 5A 0C                    0018          BERR    GOA1
000C 36 11 01 02              0019          IB      IX1.NUMIN
0010 5F 12                    0020          B       GOA1
0012 19 10 LL 16              0021  NUMIN   CBNE    INLEN.=W'10'.GOA1
0016 00 13 14                 0022          MOVE    ACCNT.IBUF
0019 01 10 LL                 0023  RDAM    MOVE    INLEN.=W'11'
001C 30 XX DD 14 KK 10 11     0024          NKI     DSKB.IBUF.KTB1.INLEN.IX1
0023 5A 0C                    0025          BERR    RDAM
0025 36 11 01 02              0026          IB      IX1.AMTIN
0029 5F 12                    0027          B       RDAM
002B 19 10 LL 16              0028  AMTIN   CBNE    INLEN.=W'11'.RDAM
002F 00 12 14                 0029          MOVE    AMNT.IBUF
0032 31 XX 92 FF              0030          EDWRT   DSDY.OUTF
0036 5F 38                    0031          B       GOA1
                              0032  ******
                              0033  OUTF    FRMT
0000 C1 LL                    0034          FCOPY   =X'2031'
0002 C3 06 ..                 0035          FTEXT   'ACCOUNTNR. '
0009 C0 13                    0036          FCOPY   ACCNT
0011 E8                       0037          FEOR
0012 C1 LL                    0038          FCOPY   =X'2030'
0014 C3 0B ..                 0039          FTEXT   'AMOUNT     '
0021 PP 12                    0040          FMEL    '99999999V99',AMNT
                              0041          FMEND
                              0042          END
```

```
* CREDIT TRANSLATOR  REL 4.3 790523 *  PROCEDURE DIVISION        * IDENT EX7    -- * DATE 790920 * PAGE 0003 *

LOC  OC OPERANDS            LINE  LABEL    OPCODE  OPERANDS                  COMMENT                    r


PROGRAM LENGTH = 0038  ERROR = 0000
```

```
* CREDIT TRANSLATOR  REL 4.1 790523 *  DATA ITEM NAME TABLE       * IDENT EX7   -- * DATE 790920 * PAGE 0004 *
```

NAME	REF	TYPE	NAME	REF	TYPE	NAME	REF	TYPE	NAME	REF	TYPE
ACCNT	13	STR	AMNT	12	BCD	IBUF	14	STR	INLEN	10	BIN
IX1	11	BIN									

```
* CREDIT TRANSLATOR  REL 4.1 790523 *  PROCEDURE LABELS          * IDENT EX7    * DATE 790920 * PAGE 0005 *
```

NAME	REF	TYPE	NAME	REF	TYPE	NAME	REF	TYPE	NAME	REF	TYPE
AMTIN	002B	ADR	GOA1	0000	ADR	KTB1	0000	KEY	NUMIN	0012	ADR
OUTF	0001	FOR	RDAM	0019	ADR	T:EDWR	0002	EXT	T:NKI	0004	EXT

```
PROG ELAPSED TIME:  00H-00M-32S-760MS-
```

11.10 CREDIT Linker listings

The following listings are produced by the CREDIT Linker:

- Load map
- Long branch table
- Call table
- Perform table
- Literal pool
- Format pool
- Keytable pool
- Segment map
- Address cross reference list
- Literal cross reference list
- Picture/format cross reference list
- Linker statistics

11.10.1 Load map

This is used for setting the relocation registers when debugging programs.

```
* CREDIT CODE LINKER REL 4.1 790523 * LOAD MAP  SEGMENT  00          * DATE 790920 * PAGE    1 *

LOC       MODULE    ERROR         COMMENT

0000      EX7                     TRA 4.1  79-09-20 F1 01111
```

```
• CREDIT CODE LINKER REL 4.1 790523 • CALL TABLE  SEGMENT  00                - • DATE 790920 • PAGE    2 •
```

LOC	DATA	IX	SYMBOL	DEFINED
003A	****	01	T:NKI	
003C	****	02	T:EOWR	

11.10.2 Call table

This table contains all referances to external routines (CALL instructions) which were not satisfied by the TLK command. Each time a reference is encountered in the intermediate code, the linkage editor (LKE command), replaces it by an 'index value' which points to the called address in the call table. During execution of the application program, the interpreter refers to the call table for actual destination addresses.

- LOC is the displacement of each entry in the table within segment zero.
- DATA is the call Address relative to the start of segment zero.
- IX is the index value (01 - FF).
- SYMBOL is the name of the external routine.
- DEFINED is not used in this table.11.10.3 Long branch table

```
• CREDIT CODE LINKER REL 4.1 790523 • LE TABLE  SEGMENT  00                    • DATE 790702 •

LOC    DATA       IX  SYMBOL  DEFINED

0DA4   00 0037    01  COMED2  MEL23T
0DA8   00 004E    02          MED23T
0DAC   00 0068    03          MED23T
0DB0   00 0051    04          MED23T
0DB4   00 0AB5    05          MED23T
0DB8   00 0CCC    06          MED23T
0DBC   00 02DA    07          MED23T
0DC0   00 00F4    08          MED23T
0DC4   00 000F    09          MED23T
0DC8   00 06B0    0A          MED23T
0DCC   00 015E    0B          MED23T
0DD0   00 0106    0C          MED23T
0DD4   00 0008    0D          MED23T
0DD8   00 06F1    0E          MED23T
0DDC   00 0195    0F          MED23T
0DE0   00 016F    10          MED23T
0DE4   00 0846    11          MED23T
0DE8   00 01D2    12          MED23T
0DEC   00 0083    13          MED23T
0DF0   00 0266    14          MED23T
0DF4   00 020D    15          MED23T
0DF8   00 02F8    16          MED23T
0DFC   00 03E5    17          MED23T
0E00   00 03E9    18          MED23T
0E04   00 0316    19          MED23T
```

11.10.3 Long branch table

In order to reduce the ammount of memory required for a long branch instruction, the linker (TLK) generates a table of destination addresses. Each time a long branch instruction is encountered in the intermediate code, the linker places the destination address (i.e. segment nimber and the address to be branched to) in the long branch table.

The three byte destination address in the long branch instruction is replaced by a one byte 'index value' which points to the destination address in the long branch table. During execution of the application program the interpreter refers to the long branch table for actual destination addresses.

- LOC is the displacement of each entry in the segment.
- DATA is the destination address and segment number.
- IX is the index of the entry in the table, and starts at the first number after the last number for the same type of table in segment zero; this applies to all tables.
- SYMBOL is the first instruction in the module containing the destination.
- DEFINED is the module containing the destination.

```
--------------------------------------------------------------------
* CREDIT CODE LINKER REL 4.1 790523 * PERFORM TABLE   STATEMENT  00
--------------------------------------------------------------------

LOC    DATA       IX  SYMBOL  DEFINED

0E8A   00 0284    01          MEDIST
0E8E   00 07E4    02          MEDIST
```

11.10.4 Perform table

This table contains the address of each CREDIT subroutine which is called (PERF or PERFI instructions) within this segment. It has the same layout as the long branch table. Each time a perform to a CREDIT subroutine is encountered, in the intermediate object code the subroutine name is replaced by an 'index value' which points to the subroutine address in the perform table.

- LOC is the displacement of each entry in the segment.
- DATA is the destination address.
- IX is the index of the entry in the table.
- SYMBOL is the name of the subroutine.
- DEFINED is the name of the module containing the subroutine.

```
---------------------------------------------------------------------------
• CREDIT CODE LINKER REL 4.1 790823 • LITERAL POOL  SEGMENT  00
---------------------------------------------------------------------------

IX    TYPE    LOC   DATA

10    BIN     003E  000A
11    BIN     0040  000B
12    STR     0042  2030
13    STR     0044  2031
```

11.10.5 Literal pool

The literal pool contains all the literals used in this segment. Each time a literal is encountered in the intermediate code is replaced by an 'index value' which points to the literal in the literal pool.

- IX is the index value of the entry (01-FF or 4100-41FF).
- TYPE is BIN, BCD or STR.
- LOC is the displacement of the literal within the segment.
- DATA is the hexadecimal representation of the literal.

```
* CREDIT CODE LINKER REL 4.1 790523 * PICTURE POOL  SEGMENT  00

IX    TYPE    LOC   DATA
10    PIC     0046  39393939393939563939
```

11.10.6 Picture pool

The picture pool contains all picture strings used in this segment. Each time a referance to a picture string is encountered in the intermediate code, it is replaced by an 'index value' which points to the picture string in the pool.

- IX is the index value of the entry (01-FF or 5100-51FF).
- TYPE indicates that the entry is a picture string (PIC).
- LOC is the displacement within the segment.
- DATA is the hexadecimal representation of the picture string.

```
* CREDIT CODE LINKER REL 4.1 790523 * KEYTABLE POOL   SEGMENT  00

 Y    TYPE    LOC   DATA

10    KEY     0051  0123
```

11.10.7 Keytable pool

The keytable pool contains all keytables used in [illegible]
in segment zero. Each time a referance to a keytable [illegible]
intermediate code, it is replaced by an 'index value' [illegible]
keytable in the pool.

- IX is the index value of the entry [illegible]
- TYPE indicates the entry is a [illegible]
- LOC is the displacement of the [illegible]
- DATA is the hexadecimal represent[illegible]

```
* CREDIT CODE LINKER REL 4.1 790523 * FORMAT POOL   SEGMENT  00          * DATE 790920 * PAGE    4 *

IX     TYPE     LOC   DATA

10     FMT      0053  C113C30B4143434F554E544E522E20CD13EAC112C30B414D4F554E5420202020201012
```

11.10.8 Format pool

The format pool contains all format lists used in the segment. Each time a referance to a format list is encountered in the intermediate code, it is replaced by an 'index value' which points to the format list in the pool.

- IX is the index value of the entry (00-FF or 7100-71FF).
- TYPE is FMT for a Format list or FTB for a Format table.
- LOC is the displacement withtin the segment.
- DATA is the hexadecimal representation of the list or table.

```
--------------------------------------------------------------------------------
* CREDIT CODE LINKER REL 4.3 790523 * SEGMENT MAP
--------------------------------------------------------------------------------

   S E G M E N T                          N U M B E R   O F
NUMBER   TYPE   LENGTH    USAGE          MODULES    ERRORS

00         C        362                      1         0
```

11.10.9 Segment map

This map gives a listing of the number of segments, the number of modules in each segment, and the number of bytes per segment.

```
-------------------------------------------------------------------
* CREDIT CODE LINKER REL 4.1 790523 * CROSS REFERENCE LISTING
-------------------------------------------------------------------

SYMBOL   TYPE  VALUE    SEG-DEFINED.   REFERENCES

GOAL       S   00 0000  00-EX7
T:EDWR     C                           00-EX7     (1)
T:NKI      C                           00-EX7     (2)
```

11.10.10 Linker statistics per segment

The format of the linker statistics listing per segment, the contents of the listing are self-explanatory.

11.10.11 Address cross reference listing

This listing provides cross reference between statement/subroutine identifiers in the PDIV, and the modules/segments in which they are referenced.

.

11.11 Linkage Editor listings

The first page of this listing shows all the tables and routines used by the interpreter that are linked into the application, and their start addresses.

The second page of this listing shows all the symbols (in this case start points) used within the routines listed on Page 1.

Example: T:NKI has an address within the range of T:IO, and is a start point within the Input/Output driver module.

```
0008  S:GTAB
0014  T:AA1O    TRA 4.2
0052  T:DA1O    TRA 4.1
006E  T:ATAB    TRA 4.1
0072  U:BTAB    TRA 4.2
00?4  S:BTAB    TRA 4.1
00r6  C:CB1O    TRA 4.1
0098  D:CB1O    TRA 4.2
00AE  P:MTAB
00E2  P:PIL
0184  I:NTP     T.REL=4.
042A  I:EVR     T.REL=4.
0730  I:ADS     T.REL=4.
0896  I:CMP     T.REL=4.
0918  I:CFA     T.REL=4.
0994  I:MOV     T.REL=4.
0B9A  I:MUL     T.REL=4.
0D46  I:DIV     T.REL=4.
0F0E  I:NTF     T.REL=4.
0FA6  I:EDT     T.REL=4.
138E  I:EDTE    T.REL=4.
1450  I:EDS     T.REL=4.
1460  I:LTR     T.REL=4.
1500  I:EVS     T.REL=4.
1   2 I:TRA     T.REL=4.
1848  I:EAR     T.REL=4.
1B8E  I:MEA     T.REL=4.
1C10  I:ADA     T.REL=4.
1E10  TRNCHK    T.REL=4.
1E5A  TRNTAB    T.REL=4.
1E?4  TRNBUG    T.REL=4.
3DF6  T:IO      T.REL=4.
3670  T:LSEG    T.REL=4.
```

••• SYMBOL TABLE •••

```
C:CB1D 007A R   D:CB1D 009A R   G0A1   00E2 R   I:ADD  0730 R   I:BUF  1684 R
I:CHK  1880 R   I:CMP  0896 R   I:CPA  0918 R   I:CPY  155E R   I:DIV  0016 R
I:DLT  1482 R   I:EBR  1430 R   I:EBRB 1434 R   I:EBRI 1404 R   I:ECB  1678 R
I:ECPY 1180 R   I:ECTR 1196 R   I:ECW  1102 R   I:ECWC 110C R   I:ED1  1054 R
I:ED5  145C R   I:EDSK 101E R   I:EDT  1050 R   I:EDW  1056 R   I:EFIL 11E6 R
I:EFLA 10D2 R   I:EGTB 1450 R   I:EGTD 144E R   I:EINH 138E R   I:EKIH 138E R
  ENL  1182 R   I:EOR  11C8 R   I:EPIC 1202 R   I:ER10 1888 R   I:ER11 188C R
I:ER12 1890 R   I:ER13 1894 R   I:ER14 1898 R   I:ER15 189C R   I:ER16 18A0 R
I:ER17 18A4 R   I:ER18 18A8 R   I:ER19 18AC R   I:ER1A 18B0 R   I:ER1B 18B4 R
I:ER1C 18B8 R   I:ERR  18BC R   I:ERR0 1848 R   I:ERR1 184C R   I:ERR2 1850 R
I:ERR3 1854 R   I:ERR4 1858 R   I:ERR5 185C R   I:ERR6 1860 R   I:ERR7 1864 R
I:ERR8 1868 R   I:ERR9 186C R   I:ERRA 1870 R   I:ERRB 1874 R   I:ERRC 1878 R
 :ERRD 187C R   I:ERRE 1880 R   I:ERRF 1884 R   I:ESKI 1446 R   I:ESL  11A0 R
I:ETAB 1168 R   I:ETXT 1180 R   I:EVA0 04EC R   I:EVA1 044A R   I:EVA2 0442 R
 :EVA3 044E R   I:EVA5 0458 R   I:EVA6 0452 R   I:EVA7 045C R   I:EVB  06F6 R
  :EVI  06E4 R   I:EVIN 13C8 R   I:EVNO 1448 R   I:EVS1 1650 R   I:EVS2 1668 R
  :EVT  0FE8 R   I:EVT0 06A2 R   I:EXIT 1136 R   I:FFSN 1110 R   I:FML  068C R
 :HEX  1894 R   I:HEXB 188E R   I:HEXW 1894 R   I:INS  1500 R   I:MCM  14A4 R
 :MOV  0994 R   I:MUL  0B9A R   I:MVC  0982 R   I:NTFA 0F6E R   I:NTP  01F4 R
I:NTPA 01D4 R   I:NTPR 01FE R   I:NTR  16F6 R   I:PRT  181C R   I:RTO  0260 R
I:RT1  025e R   I:RT2  0248 R   I:SH1L 0614 R   I:SH1R 0F50 R   I:SUB  0736 R
 :TRA  17A4 R   I:TRA0 179E R   I:TR6  17EC R   I:TRC  01D2 R   I:ACP  155E R
 :BMS  00E2 R   P:END  0184 R   P:MTAB 00AE R   S:B1AB 0006 R   S:G1AB 0008 R
 :WA10 004A R   T:A1AB 0070 R   T:BAT  0120 R   T:CAT  011A R   T:CSEG 3674 R
 :DA10 0052 R   T:DSC0 3144 R   T:DSC1 3156 R   T:DSC2 3166 R   T:LD10 32FA R
  LDWE 326A R   T:EDWF 32BE R   T:EDW1 32D6 R   T:EDWR 32F0 R   T:FDS0 3392 R
I:FDSP 339E R   T:FMT  01A0 R   T:G0SP 339E R   T:GTCW 3564 R   T:I01  3214 R
I:I02  3218 R   T:I03  321C R   T:I04  3220 R   T:IORE 323A R   T:KEY  017C R
I:KI   30F6 R   T:KIAA 366C R   T:KIAC 366E R   T:KIPR 331A R   T:LIT  0174 R
T:LOFS 36B4 R   T:LOPS 36BE R   T:LSEG 3670 R   T:MWAI 31CA R   T:NKI  30FA R
T:PAT  0120 R   T:PIC  0178 R   T:READ 3128 R   T:STCW 3468 R   T:WAIT 31B4 R
T:WRIT 3104 R   T:XSTA 31A4 R   TB:ABT 1F6E R   TB:ENT 1FCA R   TB:RDC 20B2 R
TC:ABT 1CEA R   TC:CHK 1BEC R   TC:ERR 1DD2 R   TC:RDC 1CF6 R   TT:HLT 1E66 R
TT:LOP 0040 A   TT:MSG 1E70 R   TT:PRC 1E6A R   TT:SNO 1C6E R   TT:TID 1E6C R
TT:TON 1D5A R   TT:TRP 1D5C R   TT:TRS 0042 A   TT:VER 1DE2 R   TT:VMM 0040 A
TT:VON 1DE0 R   U:BTAB 0074 R
```

11.12 SYSLOD (Configuration data)

Two items are required in addition to the application load module for the execution of the program, these are:-

- The TOSS Monitor.
- Configuration data.

At system start, the Monitor is read into memory, followed by the application, and then, before the application is started, the system configuration program SYSLOD is executed. This performs the configuration of the system for the specific environment in which the application is to run.

Keyword	Page in manual
SYSLOD	MO4 3.4.1