TRANSLATIONS STRUCTURE CHECK PROCEDURES

2-WIRE NO. 1 ELECTRONIC SWITCHING SYSTEM

CONTENTS PAGE

1. GENERAL 1

A. Program Description 2

B. Program Limitations 2

2. PRELIMINARY PRECAUTION AND PROCEDURE 3

3. PROGRAM REQUEST PROCEDURE . . . 3

A. Procedure for APT-503, Issue 19 and Later Programs 3

B. Procedure for APT-503, Issue 18 and Earlier Programs 4

4. FINAL PROCEDURE 4

5. OUTPUT MESSAGES 5

6. ABBREVIATIONS 11

1. GENERAL

1.01 This section gives procedures for using the translations structure check auxiliary program (XLCK) part of MOD 5 auxiliary program package APT-03) (for the 2-Wire No. 1 Electronic Switching System (ESS). The program operates as a library program in program store (PS) module 05 of PS 0 (or module 15 of PS 1) and requires that the special library module be loaded as described in PA-1A500 or Section 231-147-301. The XLCK program is covered in PR-1A511.

1.02 This section is reissued to update for APT-03, Issue 21 (PR-1A511, Issue 22) and to add a caution. Revision arrows are used to emphasize the more significant changes. This reissue does not affect the Equipment Test List (ETL).

1.03 This section applies to the following generics:

(a) 1E7 (APT-03, Issue 21; PR-1A511, Issue 22)

(b) 1E6 (APT-03, Issue 20; PR-1A511, Issue 21)

(c) 1E5 (APT-03, Issue 19; PR-1A511, Issue 19)

(d) 1E4 (APT-03, Issue 18; PR-1A511, Issue 18)

(e) 1E3 (APT-03, Issue 17.1; PR-1A511, Issue 17).

1.04 Abbreviations used in this section are listed in Part 6.

1.05 The translations structure check program is used to detect certain irregularities in translation structure of an operating office, primarily in its use of memory. The types of errors or invalid conditions the program is designed to detect are as follows:

(1) Overlapping memory blocks (where a block is any table, subtranslator, common block, list or auxiliary block) invalidly shared by two or more translations.

(2) Invalid word numbers in auxiliary blocks (ie, auxiliary blocks having lengths greater than 2050 decimal words).

(3) Idle link lists that loop or point outside the translation area of program store.

(4) Lost memory (ie, memory that is not on any idle link list nor a part of any translation).

(5) Out-of-range addresses (ie, address pointing outside the translation area in program store).

(6) For APT-03, Issue 20 and later, an invalid head table length of zero is found in the head table lengths translator when an address is found in the master head table.

(7) Selected head table lengths are checked for agreement as follows (APT-03, Issue 21 and later):

(a) Auxiliary master head table length equals D(160).

(b) Trunk group number length equals trunk group number supplementary length (doubled if left half).

(c) Directory number length equals number group number to rate center length.

(d) Trunk network number to trunk group number length equals trunk network number to peripheral equipment number length.

1.06 By periodically examining semipermanent translation memory of an office, any propagation of errors resulting from faulty use of translation memory can be avoided. Also, the results obtained from this program can be an important factor in determining whether or not a repacking of translations (an expensive and lengthy process) is warranted.

A. Program Description

1.07 The use of translation memory is checked by associating one bit of a call store busy/idle table with each translation word. As each translation word is checked, a corresponding bit is set in the busy/idle table. However, before a bit is set to busy, the bit is checked to determine if it is already set, indicating an overlap. When a pass through all the translation words is finished, the busy/idle table is scanned for unmarked bits which determine the lost memory.

1.08 If insufficient call store is available, the translation program store area to be checked must be divided into check blocks, each having as many words as there are bits available in the call store busy/idle table. For each check block a pass through all the translations is performed. On this pass, the proper bits in the busy/idle table are marked and any overlaps detected are stored. If any overlaps are found, a second pass through all translations is needed. In this second pass, each translation block is compared against each overlapping block found in the first pass.

1.09 Overlaps are printed out as they are found, followed by any lost memory in the check block. This procedure is repeated for each check block until the entire program store range requested is completed.

1.10 The busy/idle table uses the recent change auxiliary area. Since the program does not perform recent change hunts (except for delayed service orders), it will only account for translation irregularities as of the previous card writing. Any permanent or temporary recent changes that have entered the system since the last card writing are invisible to the program.

1.11 Running the program soon after card writing assures the maximum size busy/idle table which results in the minimum number of times the program must go through all the translators.

1.12 The program also uses the 128-word call store block P6IM0, which is normally used by the card writing program. This block is used to store overlaps detected during the first pass.

1.13 The XLCK program will abort if it detects another program using either the recent change auxiliary area or P6IM0.

B. Program Limitations

1.14 The program is limited in the amount of locating information it can store for a given overlap. Translation blocks requiring more than three levels of indexing to completely identify them cannot be uniquely referred to. For example, if an auxiliary block in a speed calling list is involved in an overlap, the information XLCK will report on this auxiliary block will be the address, type, length, master head table index, head table index, and subtranslator index. It does not have room to store the index into the speed calling list pointing to this auxiliary block. The user will have to read the list and search for the particular auxiliary block.

1.15 There are six types of translation blocks that would require more than three levels of indexing. They are as follows:

(1) Auxiliary blocks in a speed calling list

(2) Hunt lists

(3) Outdial lists

(4) Digit interpreter tables

(5) Auxiliary blocks in digit interpreter tables

(6) Speed calling lists in a centrex speed calling block.

1.16 An exact match overlap means that all the locating information for two blocks of an over-

lap are identical. The program will report this by having identical information for both block 1 and block 2 of the overlap message. If a third translation block overlaps an exact match overlap, one of three series of overlap messages will be outputted, depending on the order in which the overlaps are detected. A more detailed explanation can be found in the listing with the description of the LIB11-3 output message or in Part 5 of this section.

2. PRELIMINARY PRECAUTION AND PROCEDURE

Caution: If any translation structural changes are made while running XLCK, rerun the program to assure that the changes did not cause system troubles elsewhere. Failure to do so may cause unpredictable consequences. For example, if a maintenance person zeroed translation data found out of range while running XLCK, and did not rerun the program after making the changes, AMA billing data could be lost.

2.01 PS module 05 or 15 must be loaded with the proper auxiliary program package; therefore, before running XLCK, perform the preliminary procedures in PA-1A500 or Section 231-147-301 for inserting an auxiliary test module, configuring auxiliary test programs into service, and using the LIB-EDIT-. message.

2.02 If the RC-UPDATE message has not already been typed in as part of the translation update (card writing) procedure, type in

RC-UPDATE-.

This consolidates the recent change area and insures the maximum size of the recent change auxiliary area available for use by XLCK.

3. PROGRAM REQUEST PROCEDURE

A. Procedure for APT-503, Issue 19 and Later Programs

3.01 For APT-503, Issue 19 and later programs, there are two different tests making up the program. These are Test 00 and Test 01. It is recommended that Test 00 be run before Test 01.

3.02 The whole program, including Tests 00 and 01, may be aborted at any time by typing in the message:

LIB-START-00,00,0,00.

Test 00

3.03 Test 00 is used to detect the following errors:

- Overlapping memory blocks
- Lost memory
- Idle link lists that loop
- Auxiliary blocks with lengths greater than 2050 decimal words
- Invalid program store addresses
- For APT-03, Issue 20 and later, an invalid head table length of zero
- For APT-03, Issue 21 and later, head table lengths that do not agree.

3.04 The following procedure is used to request Test 00 of the program.

Note: The address range may be for either the right or left half of program store translation data, or both. To request that both the right half and the corresponding left half of program store be checked, set aaaaaaa equal to the first right half address and bbbbbbb equal to the last left half address.

(1) Type in

LIB-START-11,00,0,0e.

e = 3 to allow printout of lost memory blocks.

= 4 to inhibit printout of lost memory blocks smaller than specified in the next paragraph in variable ddddddd.

The system responds with ØK.

(2) Type one of the following:

(a) If e = 3 in the LIB-START message:

LIB-OCT-0aaaaaaa/
0bbbbbbb/
0ccccccc.

(b) If e = 4 in the LIB-START message:

LIB-OCT-0aaaaaaa/
0bbbbbbb/
0ccccccc/
0ddddddd.

aaaaaaa = Program store address where check is to begin. See Note.

bbbbbbb = Program store address where check is to end. See Note.

ccccccc = 0000000 if program is to determine the maximum size for a check block ***or***

= Size of check block (usually specified by user only if program finds too many overlaps).

ddddddd = Maximum size of lost memory blocks for which printouts will be inhibited (eg, if ddddddd = 5, no lost memory printouts will be printed for blocks equal to or smaller than 5.

The system responds with an ØK after each line of the message.

Test 01

3.05 Test 01 returns lost memory to the available program store link lists. Lost memory will not be returned to the available space link lists if any other structural errors exist.

Note: Unlinked MODS will be added to the available space link list if specified in the address range aaaaaaa to bbbbbbb. Unlinked MODS are those available for use but not on the link list of available space.

3.06 The following procedure is used to request Test 01 of the program.

(1) Type in

LIB-START-11, 01,0,0e.

e = 3 to allow printout of lost memory blocks.

= 4 to inhibit printout of lost memory blocks.

The system responds with ØK.

(2) Type one of the following:

(a) If e = 3 in the LIB-START message, type

LIB-OCT-0aaaaaaa/
0bbbbbbb/
0ccccccc.

(b) If e = 4 in the LIB-START message, type

LIB-OCT-0aaaaaaa/
0bbbbbbb/
0ccccccc/
0ddddddd.

aaaaaaa = Program store address where check is to begin (see Note, paragraph 3.04).

bbbbbbb = Program store address where check is to end (see Note, paragraph 3.04).

ccccccc = 0000000 if program is to determine the maximum size for a check block ***or***

= Size of check block (usually specified by user only if program finds too many overlaps).

ddddddd = Maximum size of lost memory blocks for which printouts will be inhibited (eg, if ddddddd = 5, no lost memory printouts will be printed for blocks equal to or smaller than 5).

The system responds with an ØK after each line of the message.

B. Procedure for APT-503, Issue 18 and Earlier Programs

3.07 For APT-503, Issue 18 and earlier programs, perform Test 00 (paragraphs 3.03 and 3.04). Do not perform Test 01.

4. FINAL PROCEDURE

4.01 After all tasks desired to be performed by XLCK have been finished, configure the MOD 5 containing XLCK out of service, remove the PS module and replace it with the original PS module 05

using procedures covered in PA-1A500 or Section 231-147-301.

5. OUTPUT MESSAGES

5.01 As a result of requesting XLCK with the input messages in Part 3, the following output messages will provide the necessary information about any invalid or irregular program store translation block. All data is octal unless noted otherwise. Blocks for LIB11 output messages are shown in Fig. 1.

LIB11-1

5.02 If the system prints out

LIB11-1 ááaa bbbbbbb ccccccc.

ááaa = END—The check of data from bbbbbbb to ccccccc as requested in paragraph 3.01 has been completed.

= OK—A check block specified by starting address bbbbbbb and ending address ccccccc has been checked and any invalid conditions will be reported.

= LIST—The idle link list of octal length specified by bbbbbbb either loops or ends in some abnormal way at the list specified by ccccccc.

= OVFL—Because of excessive overlaps in the check block specified by bbbbbbb and ccccccc in Part 3, the error buffer has overflowed. Try again with a smaller check block size specified by the LIB-ØCT message in Part 3.

bbbbbbb = Starting address.

ccccccc = End address.

LIB11-2

5.03 If the system prints the following message, it shows that the given auxiliary block length is supposed to be greater than 32 words, but the length is either less than 32 words or greater than 2050 words, or the abbreviated dialing bits (ADB) are improperly set for the type of line.

LIB11-2 INV AUX

AUX BLK	HH-IX	HT-IX	ST-IX
aaaaaaa	bbbb	cccc	ddddd

aaaaaaa = Address of the auxiliary block.

bbbb = Master head table index.

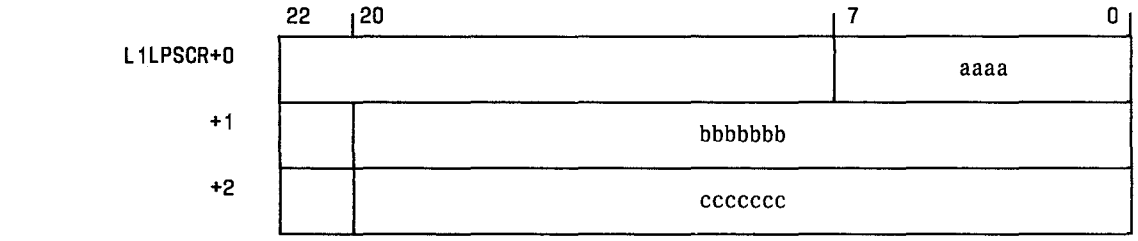
cccc = Head table index.

ddddd = Subtranslator index.

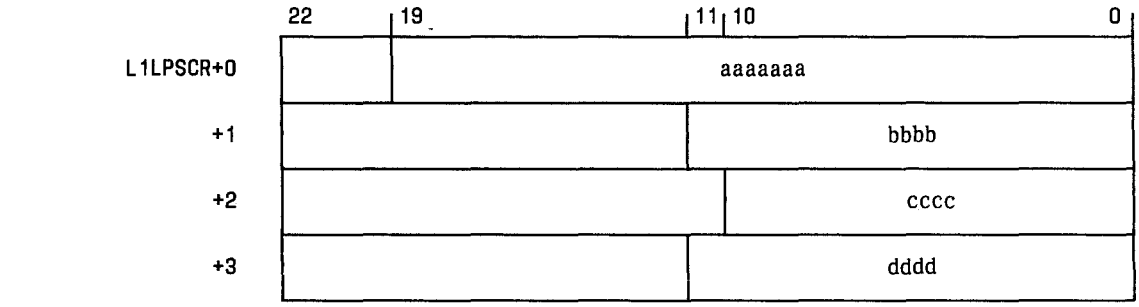
LIB11-3

5.04 If the system prints the following message, it indicates two translation program store blocks that overlap. Enough information is provided to verify that the blocks overlap.

A. LIB11-1



B. LIB11-2 INV AUX



C. LIB11-3 OVERLAP

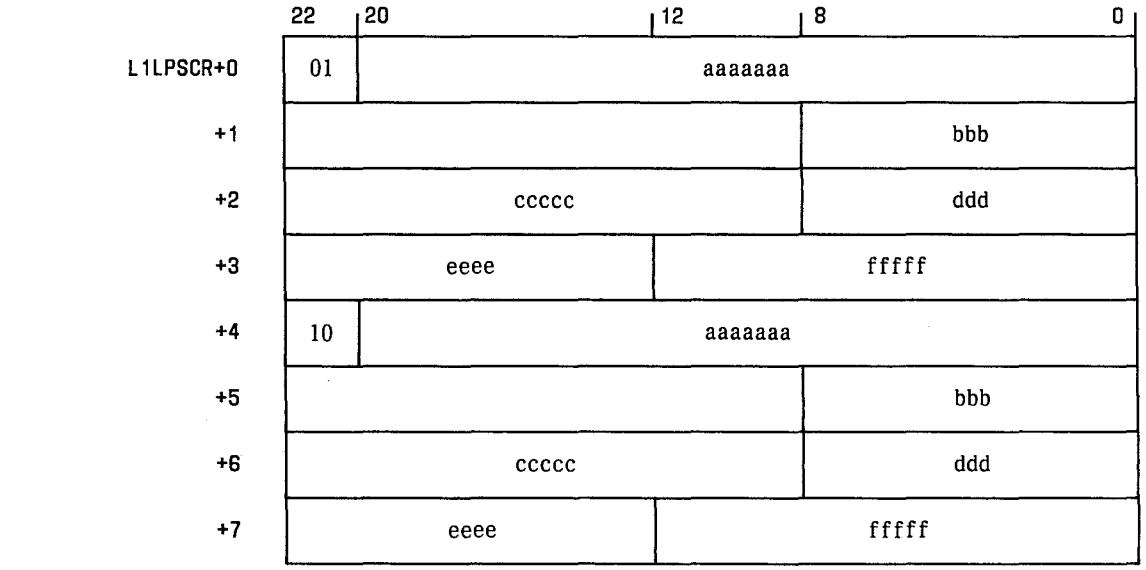
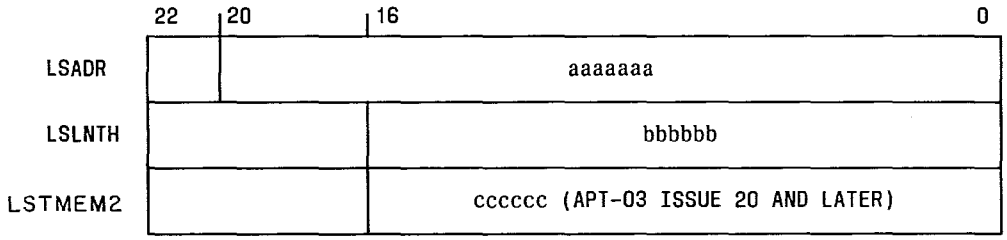
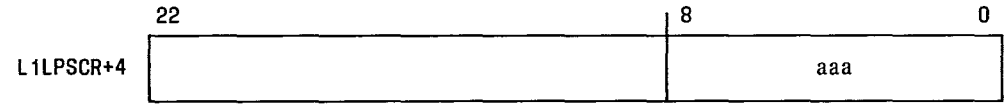


Fig. 1—Blocks for L1B11 Output Messages (Sheet 1 of 2)

D. LIB11-4 LOST MEMORY



E. LIB11-6 SIZE OF BUSY/IDLE TABLE



F. LIB11-7 INV HT LNGTH (APT-03 ISSUE 20 AND LATER)

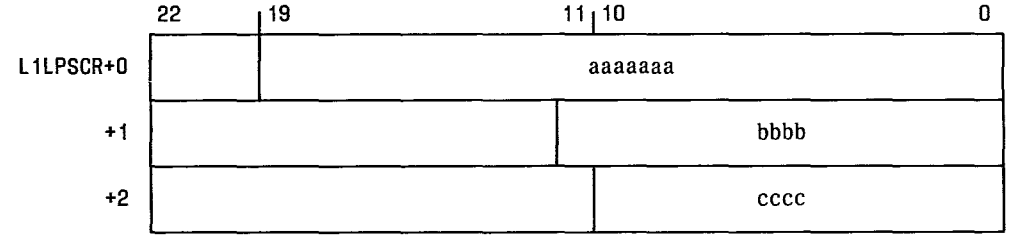


Fig. 1—Blocks for LIB11 Output Messages (Sheet 2 of 2)

```
LIB11-3                    OVERLAP
  BLK-1                    aaaaaaa
      TYPE                   bbb
      LNGH                   ccccc
      HHTL                   ddd
        HTI                  eeee
        STI                  fffff

  BLK-2                    aaaaaaa
      TYPE                   bbb
      LNGH                   ccccc
      HHTL                   ddd
        HTI                  eeee
        STI                  fffff
```

Note: For later issues of XLCK, the LIB11-3 message prints horizontally as in Fig. 2.

For each block (BLK-1 and BLK-2):

aaaaaaa = Address of the program store block.

bbb = Data code for type of block. (See explanation of data code in the paragraph immediately following this list of variables.)

ccccc = Length of block.

ddd = Master head table index.

eeee = Head table index.

fffff = Subtranslator index.

The data code (bbb) will indicate the type of block being marked. Its purpose is to aid in locating the problem when an overlap is found. Octal 100 is added (or equivalently OR'd) to the code to indicate memory associated with the master head table annex. Octal 200 is added to the code to indicate memory associated with delayed recent change service orders. The code has the following meaning:

1—Master head table

2—Head table

3—Subtranslator or common block or IDDD primary digit interpreter table

4—Auxiliary block

5—Speed calling list head table

6—Speed calling list

7—Subauxiliary block

10—Hunt list head table

11—Hunt list

12—Outdial list head table

13—Outdial list

14—Digit interpreter table (centrex, tandem, IDDD, DN-CTXN, T3-6D) or primary table (Toll 3/6D)

15—Auxiliary block in table

16—Link list

17—Fixed test words

20—Preferential hunt list head table

21—Preferential hunt list

Note: Block 1 and block 2 printouts may be identical if more than 3 levels of indexing are used to obtain the location of the overlapped area. For example, if a particular speed calling list contains two addresses pointing to the same auxiliary block, the information XLCK stores on this auxiliary block (address, length, master head table index, head table index, subtranslator index, and type) will be identical for the two auxiliary blocks.

The following types of translations are of this class:

(a) Auxiliary blocks in the same speed calling list.

(b) Hunt lists in the same hunt list head table.

(c) Outdial lists in the same outdial list head table.

(d) Digit interpreter tables.

(e) Auxiliary blocks in digit interpreter tables of the same Centrex group.

(f) Speed calling lists in the same centrex speed calling block. If the lists overlap, any aux block in the list will also come out as overlaps.

The situation of three or more overlapping identical memory blocks is also identifiable. In general, for N blocks there will be N−1 identical messages with block 1 and block 2 being identical in each message. If a third block overlaps two identical translation blocks, then one of three series of messages will be outputted, depending on the order in which the blocks are encountered in the first pass. Let I refer to the block of information pertaining to the identical overlaps, and let N refer to the nonidentical translation block overlapping the identical overlaps. Then the three series of messages can be defined as follows:

Series 1:	LIB11-3	LIB11-3	LIB11-3
	I	I	I
	I	N	N
Series 2:	LIB11-3	LIB-3	
	I	I	
	N	I	
Series 3:	LIB11-3	LIB-3	
	N	N	
	I	I	

Note 1: For left-half subtranslators, the subtranslator index must be divided by 2 to get the true index.

Note 2: To get the overlapped block of memory if the overlap is contained in the LEN translator and the office is 2:1 line concentration ratio (LCR), treat the subtranslators in pairs with the head table index of the printout associated with the first entry of a pair. The subtranslator index is added to the address in the head table. (The address must be adjusted for a mate frame.)

Note 3: If the overlap has an index in the auxiliary master head table which is equal to are greater than 20, the −1 word of the head table which contains the length of the head table will be treated separately from the head table itself.

LIB11-4

5.05 If the system prints out:

LIB11-4 LOST MEMORY
aaaaaaa 0(bbbbbb) D (cccccc)

aaaaaaa = Address of lost memory.

bbbbbb = Length of this block of memory in octal.

cccccc = Length of this block of memory in decimal (APT-03, Issue 20 and later).

This PS memory is currently not being used by any translation or on any link list and should be available for addition to the proper link list. Caution should be used to verify that this block has not been set aside for future use.

Note: Ignore a block of lost memory if the address is a left-half address (bit 20 of address is set) and the length = 1. Such a block is of no value on the idle link list. Printouts can be inhibited for block sizes equal to or smaller than the ddddddd variable in the LIB-OCT input message (paragraph 3.01).

LIB11-5

5.06 If the system prints out:

LIB11-5 RC AUX ACTIVITY

This indicates that the pointers to the beginning and/or end of the recent change auxiliary area

LIB11-3 OVERLAP

BLK-1 aaaaaaa TYPE bbb LNGH ccccc HHTI ddd HTI eeee STI fffff

BLK-2 aaaaaaa TYPE bbb LNGH ccccc HHTI ddd HTI eeee STI fffff

Fig. 2—Format for LIB11-3 OVERLAP Output Message (XLCK Later Issues)

have changed. (The program uses the recent change auxiliary area for the busy/idle table). If either of the pointers point within the original recent change auxiliary area, the program will terminate.

LIB11-6

5.07 If the system prints out:

LIB11-6 SIZE OF BUSY/IDLE TABLE aaa

aaa = Length of busy/idle table in decimal.

The LIB11-6 message will be printed if the busy/idle table is 300 words or less as a warning that slower excution should be expected. A small busy/idle table means a long processing and printing time. The user may prefer to abort the program by typing LIB-START-00,00,0,00.

If using APT-503, Issue 18 and earlier programs and if aaa = 1, perform the following steps:

(a) Abort the program by typing:

LIB-START-00,00,0,00.

(b) Type:

RC-UPDATE-.

(c) Restart XLCK using the procedure in paragraph 3.01.

LIB11-7

5.08 For APT-03, Issue 20 and later, if the system prints out:

LIB11-7 INV HT LNGTH

HT ADDR	HH-IX	HT LNGTH
aaaaaaa	bbbb	cccc

aaaaaaa = Address of the invalid head table length.

bbbb = Head head table index.

cccc = Invalid head table length—always zero.

The LIB11-7 message will be printed if there is a head table address found in the master head table but no length is given in the head table lengths table. The head table and subsequent tables will not be marked (ie, they will appear as lost memory). Correct the length in the table, then rerun XLCK.

LIB11-8

5.09 ▶For APT-03, Issue 21 and later, if the system prints out:

LIB11-8 HT LENGTHS DO NOT AGREE

HH INDEX	HT LENGTH
aaaaaaaa	bbbbbbbb
cccccccc	dddddddd

aaaaaaaa/cccccccc = The head head table indexes.

bbbbbbbb/dddddddd = The respective lengths from the head table lengths table.

The LIB11-8 message will be printed if the length of the two translators do not agree [see paragraph 1.05(7)]. Check the office directory to determine which length is incorrect, then correct the length in the head table lengths table.◀

5.10 If the system prints out:

L1B MESSAGE - OCTAL

aaaaaaaa bbbbbbbb cccccccc dddddddd eeeeeeee

aaaaaaaa = Invalid address.

bbbbbbbb = Type of block.

cccccccc = Head head table index.

dddddddd = Head table index.

eeeeeeee = Subtranslator index.

This message indicates that an assumed program store address is not a valid PS translation address. Determine the location of the erroneous address, correct and rerun XLCK.

5.11 If the system prints out:

LIB11 RELO ABT ADDR XLCK aaaaa

the program has aborted. Find the value aaaaa in the abort catalog at the end of PR-1A511 to determine why. The program will abort if:

(a) Recent change update, consolidation, or count of modules is in progress.

(b) Card writing is in progress.

(c) Any recent change puts data in the recent change auxiliary area of call store.

(d) Too many overlaps have been found,

(e) The program store range specified in the input message is such that the end address is less than the beginning address.

6. ABBREVIATIONS

ADB	Abbreviated Dialing Bits
ESS	Electronic Switching System
LCR	Line Concentration Ratio
LEN	Line Equipment Number
PS	Program Store
XLCK	Translation Structure Check Program