DL/I DOS/VS Reference Summary: CALL Programming Interface

SX24-5103-4

Fifth Edition (January 1984)

This edition, SX24-5103-4, is a major revision of SX24-5103-3 and applies to Version 1, Release 7 (Version 1.7) of IBM System/370 Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS), Program Number 5746-XX1. This reference summary will be updated from time to time; however, the base documentation is the authoritative source and will be the first to reflect changes. Information herein is extracted from *DL/I DOS/VS Application Programming: CALL and RQDLI Interfaces*, SH12-5411-6.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below. Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

This publication has been produced by IBM Corporation, Programming Publications, Dept G60, P.O. Box 6, Endicott, New York, U.S.A. 13760.

COBOL ENTRY AND RETURN

ENTRY 'DLITCBL' USING pcb-name-1[, . . . , pcb-name-n] .
GOBACK.

PL/I ENTRY AND RETURN

DLITPLI:PROCEDURE(pcb_ptr[, . . . ,pcbn_ptr])OPTIONS(MAIN);
RETURN;

ASSEMBLER ENTRY AND RETURN

```
PROGRAM  CSECT                   ENTRY POINT
         USING   *,BASE          ADDRESSABILITY
         SAVE    (14,12)         SAVE DL/I REGS
         LR      BASE,15         ENTRY POINT ADDR
         MVC     PCBS,0(1)       SAVE PCB LIST
*        REG  1 CONTAINS PTR TO PCB ADDR LIST
*        REG 13 CONTAINS PTR TO DL/I SAVE AREA
*        REG 14 CONTAINS DL/I RETURN ADDR
*        REG 15 CONTAINS PGM ENTRY POINT
         :
         RETURN  (14,12)         RELOAD DL/I REGS
*                                AND RETURN CTRL
         :
         END
```

COBOL PCB MASK

```
01 PCBNAME.
   02 DBD-NAME              PICTURE X(8).
   02 SEG-LEVEL             PICTURE XX.
   02 STATUS-CODE           PICTURE XX.
   02 PROC-OPTIONS          PICTURE XXXX.
   02 RESERVE-DL/I          PICTURE S9(5) COMP.
   02 SEG-NAME-FB           PICTURE X(8).
   02 LENGTH-FB-KEY         PICTURE S9(5) COMP.
   02 NUMB-SENS-SEGS        PICTURE S9(5) COMP.
   02 KEY-FB-AREA           PICTURE X(n).
```

PL/I PCB MASK

```
DECLARE PCB_POINTER POINTER;
DECLARE 1 PCBNAME           BASED (PCB_POINTER),
        2 DBD_NAME          CHAR(8),
        2 SEG_LEVEL         CHAR(2),
        2 STATUS_CODE       CHAR(2),
        2 PROC_OPTIONS      CHAR(4),
        2 RESERVED_DLI      FIXED BIN(31,0),
        2 SEG_NAME_FB       CHAR(8),
        2 LENGTH_FB_KEY     FIXED BIN(31,0),
        2 NUMB_SENS_SEGS    FIXED BIN(31,0),
        2 KEY_FB_AREA       CHAR(n);
```

ASSEMBLER PCB MASK

```
PCBNAME DSECT
DBNAME   DS      CL8      DATA BASE NAME
SGLEVEL  DS      CL2      SEGMENT LEVEL
STATUS   DS      CL2      STATUS CODE
PROCOPT  DS      CL4      PROC OPTIONS
         DS      F        RESERVED
SGNAME   DS      CL8      SEGMENT NAME
KEYLEN   DS      F        LENGTH OF KEY
NUMSGS   DS      F        NO. OF SENSITIVE SEGS
KEYAREA  DS      0CLn     KEY FEEDBACK AREA
```

COBOL CALL–GENERAL FORMAT

CALL 'CBLTDLI' USING [parm–count,] call–function, db–pcb–name,
i/o area[, ssa, ssa2, . . . , ssa15] .

COBOL CALL–ONLINE SCHEDULING

CALL 'CBLTDLI' USING [parm-count,] call-function
[, psbname [, uibparm]] .

PL/I CALL–GENERAL FORMAT

CALL PLITDLI(parm-count, call-function, db-pcb-name, i/o area
[, ssa, ssa2, . . . , ssa15]);

PL/I CALL–ONLINE SCHEDULING

CALL PLITDLI (parm-count, call-function
[, psbname [, uibparm]]);

ASSEMBLER PARAMETER LIST AND CALL–GENERAL FORMAT

```
PARMLIST  DC      A(PARMCT)
FUNC      DC      A(DLIFUNC)
PCB       DC      A(0)
IOAREA    DC      A(0)
SSA1      DC      A(0)
           :
SSA15     DC      A(0)
           :
          LA      R1,PARMLIST       Note: CALL becomes
          CALL    { ASMTDLI }             CALLDLI
                  { CBLTDLI }             for online.
```

ASSEMBLER CALL–ONLINE SCHEDULING

```
CALLDLI { ASMTDLI } ,([ parm-count, ] call-function
        { CBLTDLI }
     [, psbname [, uibparm ] ] )
```

CALL PARAMETERS

parm-count — names a fullword binary field containing the number of call parameters (excluding count itself). Up to 18 possible values.

function — names a 4-byte field containing the code for the function the call performs. Possible values are:

GUbb	GET UNIQUE
GHUb	GET HOLD UNIQUE
GNbb	GET NEXT
GHNb	GET HOLD NEXT
GNPb	GET NEXT WITHIN PARENT
GHNP	GET HOLD NEXT WITHIN PARENT
DLET	DELETE
REPL	REPLACE
ISRT	INSERT
CHKP	CHECKPOINT
PCBb	SCHEDULING CALL *
TERM	TERMINATION

* CICS/VS transactions only

db-pcb-name — names the PCB mask.

i/o-area — names input/output area for data base segments.

ssa — names a field containing an SSA.

ssa2 . . ssa15 — names fields containing additional SSAs.

psbname — name of PSB program uses.

uibparm — name of fullword location containing address of User Interface Block.

QUALIFIED SSA FORMAT ④ ⑤

(optional) (optional)

				BOOLEAN STATEMENT												
elements	Segment Name	Command Codes		Begin Qualif.	Qualification Statement # 1			Boolean Operator	Qualification Statement # 2			Boolean Operator	Qualification Statement # n			End Qualif.
contents	Name of Segment Type	*	Code Characters	'('	Field Name	R O	Compar. Value	'&' '*' or '+' '|'	Field Name	R O	Compar. Value	'&' '*' or '+' '|'	Field Name	R O	Compar. Value	')'
number of bytes	8	1	Var.	1	8	2	1 to 255	1	8	2	1 to 255	1	8	2	1 to 255	1

Segment Name: Must be 8 bytes long, with rightmost (trailing) blanks to fill out the field as required.

Command Codes: The command codes are optional. An asterisk (*) following the segment name indicates the presence of one or more command codes. A blank or left parenthesis is the delimiter.

Left Parenthesis, '(': Indicates the beginning of a segment qualification statement.

Qualification Statement: Is indicated by a left parenthesis following either the segment name or, if present, command codes.

Segment Field Name: Name of a segment field that appears in the description of that segment type in the DBD. The name is 8 characters long, with rightmost blanks as required to fill 8 bytes.

RO (Relational Operator): Set of two characters that express the manner in which the contents of the field referred to by the segment field name are to be tested against the comparative value.

Operator	**Meaning**
ƀ=	Equal to
=>	Equal to or greater than
=<	Equal to or less than
ƀ>	Greater than
ƀ<	Less than
¬=	Not equal to

Note: The operator characters above may be used in any order (for example, => or >=).

Comparative Value: Value against which the contents of the field referred to by the segment field name is to be tested. The length of this field must be equal to the length of the named field as defined in the DBD.

Boolean Operator or Right Parenthesis, ')': Following the comparative value is either a Boolean operator, relating this qualification statement to the next qualification statement, or a right parenthesis as the delimiter.

The logical AND is expressed by the EBCDIC character '&' or '*'. The logical OR is expressed by the EBCDIC character '+' or '|'.

UNQUALIFIED SSA FORMAT

SEG NAME	ƀ
8	1

SSA USAGE

FUNCTION CODE	MUST SPECIFY SSA	MAY SPECIFY SSA	MUST NOT SPECIFY SSA
	Unqualified, qualified, and multiple SSAs are permitted except as noted.		
GU, GHU		X	
GN, GHN		X	
GNP, GHNP		X	
DLET			X[1]
REPL			X[1]
ISRT	X[2]		

[1] An unqualified SSA is allowed to select the segment from the path of segments just retrieved using a path command code call.

[2] At least one unqualified SSA must be specified. Multiple SSAs are allowed, but if they are a mixture of qualified and unqualified SSAs, the last SSA must be unqualified.

COBOL SSA DEFINITION EXAMPLE ⑥

```
01 SSA
   02 SEGMENT–NAME       PIC X(8)    VALUE ‘ROOTSEGM’.
   02 OPEN–QUALIFIC      PIC X       VALUE ‘(’.
   02 KEY–FIELD–NAM1     PIC X(8)    VALUE ‘ROOTKEYƀ’.
   02 RELATION–OP1       PIC X(2)    VALUE ‘=ƀ’.
   02 KEY–FIELD–VAL1     PIC X(4)    VALUE ‘0013’.
   02 BOOL–QUAL          PIC X       VALUE ‘+’.
   02 KEY–FIELD–NAM2     PIC X(8)    VALUE ‘ROOTKEYƀ’.
   02 RELATION–OP2       PIC X(2)    VALUE ‘=ƀ’.
   02 KEY–FIELD–VAL2     PIC X(4)    VALUE ‘0015’.
   02 CLOSE–QUALIFIC     PIC X       VALUE ‘)’.
```

PL/I SSA DECLARATION EXAMPLE

```
DECLARE 1 SSA
   2 SEGMENT_NAME        CHAR(8)     INIT(‘ROOTSEGM’),
   2 OPEN_QUALIF         CHAR(1)     INIT(‘(’),
   2 KEY_FIELD_NAM1      CHAR(8)     INIT(‘ROOTKEYƀ’),
   2 RELATION_OP1        CHAR(2)     INIT(‘=ƀ’),
   2 KEY_FIELD_VAL1      CHAR(4)     INIT(‘0013’),
   2 BOOL_QUALIF         CHAR(1)     INIT(‘+’),
   2 KEY_FIELD_NAM2      CHAR(8)     INIT(‘ROOTKEYƀ’),
   2 RELATION_OP2        CHAR(2)     INIT(‘=ƀ’),
   2 KEY_FIELD_VAL2      CHAR(4)     INIT(‘0015’),
   2 CLOSE_QUALIF        CHAR(1)     INIT(‘)’);
```

ASSEMBLER SSA DEFINITION EXAMPLE

```
SSA   DS 0F                   SEGMENT SEARCH ARGUMENT
      DC CL8‘ROOTSEGM’        SEGMENT NAME
      DC C‘(’
      DC CL8‘ROOTKEY’         KEY FIELD NAME
      DC CL2‘=’               RELATIONAL OPERATOR
      DC CL4‘0013’            KEY FIELD VALUE
      DC C‘+’                 BOOLEAN OPERATOR
      DC CL8‘ROOTKEYƀ’        KEY FIELD NAME
      DC CL2‘=ƀ’              RELATIONAL OPERATOR
      DC CL4‘0015’            KEY FIELD VALUE
      DC C‘)’
```

SSAs WITH COMMAND CODES ⑦

Unqualified SSA Format

(optional)

elements	Segment Name	Command Codes		Ending Delimiter
contents	Name of Segment Type	*	Code Characters	ƀ
number of bytes	8	1	Variable	1

Note: For the qualified SSA format with command codes, see Qualified SSA Format on panels 4 and 5.

COMMAND CODES

L Under an established parent, retrieve the last occurrence of this segment type.

F Start with the first occurrence of this segment under its parent in attempting to satisfy this level of the call.

D For retrieval calls with multiple SSAs, move all segments in the path (up to and including the segment that satisfies the particular SSA in which this command code is present) to the user's I/O area. This is called a path call.

For INSERT calls, this command code allows a single call to insert multiple segments in a hierarchical path.

N When a REPLACE call follows a path call, use this command code for segments in the path that do not need to be replaced.

Q This command code causes DL/I to lock the segment(s) returned by the call to prevent modification by another task.

U This command code prevents the position from being moved from a segment during a search of its hierarchical dependents.

V This command code is similar to 'U' except that 'V' is automatically set at all higher levels in the call. This means that DL/I, while attempting to satisfy a call, cannot move from the existing position at the level at which 'V' is specified unless the code is disregarded.

– Null command code. Allows use of SSA in command code format without specifying a command code. Null can be replaced during execution with appropriate codes.

USE OF COMMAND CODE BY FUNCTION

Command Code	GU GHU	GN GHN	GNP GHNP	DLET	REPL	ISRT	CHKP
D	A	A	A	D	D	A	D
F	D	A	A	D	D	A	D
L	A	A	A	D	D	A	D
N	D	D	D	D	A	D	D
Q	A	A	A	D	D	D	D
U	A	A	A	D	D	A	D
V	A	A	A	D	D	A	D

A = Applicable

D = Disregarded

JOB CONTROL STATEMENTS FOR BATCH/MPS PROGRAMS

8

To Compile and Link-Edit your program:

```
// JOB COMPILE/LINK
// OPTION CATAL
   PHASE phasename,*
[INCLUDE DLZBPJRA  (for COBOL)]
// EXEC compname
         YOUR PROGRAM
/*
[INCLUDE IBMBPJRA   ](for PL/I)
[        {CBLCALLA} ](for COBOL)
[ENTRY   {PLICALLB} ](for PL/I)
/*
// EXEC LNKEDT
/&
```

If the automatic library lookup (AUTOLINK) feature is suppressed, an INCLUDE statement must be used to link-edit the DL/I language interface module (DLZLI000).

phasename   is the name of your module to be stored in the core image library.

compname   is the name of your compiler.

FCOBOL	– for the DOS/VS COBOL Compiler or the Full or Subset ANS COBOL Compiler.
PLIOPT	– for the PL/I Optimizing Compiler.
ASSEMBLY	– for the assembler
RPG II	– see RPG II panels

To Run Your Batch Program

```
// JOB EXECUTE
// UPSI     )
// ASSGN    ( Obtain these values
// TLBL     > from the data base
// DLBL     ( administrator
// EXTENT   )
// EXEC DLZRRC00,SIZE=size
{DLI} ,progname,psbname[, {buff} ]
{DLR}                     {1   }
[,HDBFR=({bufno} [,dbdname1,dbdname2, . . . ] )] [, . . . ]
         {32   }
[,HSBFR=({indno},{ksdsbuf},[{esdbuf} ],dbdname3)]
         {3    } {2      }  {2     }
[,TRACE=modname] [,ASLOG=YES]
        {TAPE }
[,LOG=( {DISK1}, {PAUSE  })]
        {DISK2}  {NOPAUSE}
/&
```

DLI   The DLI function code is required for DL/I batch programs or MPS batch programs that do not use the MPS Restart facility.

DLR   The DLR function code is required for MPS batch programs using the MPS Restart facility.

progname   name of program to be executed.

psbname   name of PSB referenced by program.

buff   number of data base subpools required.

9

The parameters HDBFR, HSBFR, TRACE, ASLOG, and LOG, can be used with continuation statements if input is on SYSIPT. If input is on SYSLOG, continuation statements are not permitted. See DL/I DOS/VS Application Programming: CALL and RQDLI Interfaces for details of parameters and keywords.

To Run Your MPS Program

```
// JOB EXECUTE MPS
// UPSI – see DBA for value
// EXEC DLZMPI00,SIZE=size
DLI,progname,psbname
             o
             o
/&
```

To Run MPS Program Using MPS Restart

```
// JOB EXECUTE RESTART
// ASSGN – see DBA for value
// EXEC DLZMPI00,SIZE=size
DLR,progname,psbname
             o
             o
/&
```

To Restart an MPS Program

```
// JOB EXECUTE RESTART
// ASSGN – see DBA for value
// RSTRT device,checkptid
DLR,progname,psbname
/&
```

ONLINE RESPONSE CODES

Condition		Response Code
NORESP	– Normal Reponse	X'00'
INVREQ	– Invalid Request	X'08'
NOTOPEN*	– Not open	X'0C'

*** NOTOPEN can only occur on scheduling call.**

Scheduling Call

NORESP   indicates normal completion; field UIBPCBAL, or TCADLPCB if UIB not used, or UAPCBL in RPG II, contains the address of the PCB list.

INVREQ   indicates the field UIBDLTR, or TCADLTR if UIB not used, or UDLTR in RPG II, contains one of these error codes:

X'01'	PSB name of scheduling call is not in the PSB directory.
X'03'	No termination (TERM) call has followed the previous scheduling (PCB) call.
X'05'	The PSB could not be initialized by DL/I online initialization.
X'06'	PSB in the scheduling call is not defined or invalid in the ACT entry.
X'09'	An MPS batch program attempted to issue a PCB call for a read-only PSB or for a nonexclusive PSB if program isolation is active.
X'FF'	The DL/I interface has been terminated or DL/I initialization failed.

(continued)

(continued) ⑩

NOTOPEN indicates that one or more DBD entries associated with this PSB are stopped or a scheduling conflict with an MPS task has occured.

Field UIBDLTR, or TCADLTR if UIB is not used, or UDLTR in RPG II, contains one of these error codes:

X'01' One or more DBD entries associated with the PSB are stopped.

X'02' A scheduling conflict with a currently active MPS batch partition occurred.

Data Base Call

NORESP indicates normal completion.

INVREQ indicates the field UIBDLTR, or TCADLTR if UIB not used, or UDLTR in RPG II, contains one of these error codes:

X'08' A DL/I call was made but the task has not scheduled a PSB.

X'FF' The DL/I interface has been terminated or DL/I initialization failed.

Termination Call

NORESP indicates that the DL/I resources have been released.

INVREQ indicates the field UIBDLTR, or TCADLTR if UIB not used, or UDLTR in RPG II, contains one of these error codes:

X'07' "TERM" requested but task not scheduled.

X'FF' The DL/I interface has been terminated or DL/I initialization failed.

CICS/VS TRACE TABLE ENTRIES FOR DL/I DOS/VS ⑪

	ENTRY ID	TYPE OF REQUEST	RESERVED	TCA ID NO.	TRACE INFORMATION
BYTES	0	1-2	3-4	5-7	8-15

	RESOURCE NAME	R14 CONTENTS	TIME OF DAY
BYTES	16-23	24-27	28-31

Byte	
0	Trace code X'F8', indicating DL/I trace entry.
1-2	Code indicating type of request that was made.
3-4	Reserved
5-7	CICS/VS transaction id (packed decimal).
8-15	Contains data unique to each type of request. Details below.
16-23	Contains the resource name (blank for DL/I).
24-27	Contains the address of a DL/I control block or module (unique to each request type). Details below.
28-31	Time-of-day (in units of 32 microseconds).

Type of Request	8	9	10	11	12	13	14	15
'S' (type 1)	PSBNAME							L/R

PSBNAME = name of the PSB being scheduled

L/R = type of data base (' ' = local, '+' = remote, '*' = extended remote PSB)

	8	9	10	11	12	13	14	15
'S' (type 2)	Mnemonic Status Code						Exit Cond.	

Mnemonic Status Code = mnemonic representing exit conditions as shown below:

Mnemonic	Exit Cond.	Meaning
SUCESS	00 00	Successfully scheduled
NOƀPSB	08 01	PSB not in directory (PDIR)
RESCHD	08 03	Task already scheduled
PSBERR	08 05	PSB initialization error
PSBATH	08 06	PSB not in program ACT entry
RETERM	08 07	Task already terminated
NOSCHD	08 08	Task not scheduled
ILEGAL	08 09	Illegal MPS scheduling call
DLIDWN	08 FF	DL/I not active
NOTOPN	0C 01	Data base not opened or stopped
MPSCFL	0C 02	Scheduling conflict with MPS task
UNDEFN	xx xx	Undefined return code

⑫

Type of Request	8	9	10	11	12	13	14	15
'D' (type 1)	Data Base Name							

Data Base Name = name of the DMB being accessed

	8	9	10	11	12	13	14	15
'D' (type 2)	Call Function Code				DL/I Status		Exit Cond.	

Call Function Code = see "function" under Call Parameters
DL/I Status = see status code panel
Exit Conditions = see 'S' (type 2)

	8	9	10	11	12	13	14	15
'T'	Why Term Used						Exit Cond.	

Why Term Used = descriptive code as shown below:

Code	Description
ABEND	Abnormal end while being scheduled
CANCEL	Task canceled by operator
DEADLK	Abnormal end by DL/I – PI scheduling deadlock
GETVIS	Abnormal end by DL/I – no space for PI
RETERM	TERM call issued when task not scheduled
SYNCPT	CICS/VS sync point call
SYSTEM	Normal end while being scheduled
USER	User issued a DL/I'TERM' or 'Tb̸b̸b̸' call

Exit Conditions = see 'S' (type 2)

R14 Contents Field:

Type of Request	
'S' (type 1)	Address of DL/I System Contents Directory (SCD)
'S' (type 2)	Address of CICS:VS TCA for task
'D' (type 1)	Address of DL/I call parameter list
'D' (type 2)	Address of PCB
'T'	Address of DL/I Program Request Handler

RPG II

⑬

Note: all RPG II information is contained in this and the adjacent panels, except for the online response codes.

RPG II Entry and Return

```
Position  6  18       27 28    32    43    48      54 55 56
          H                                            B
          •
          C  *ENTRY PLIST
          C           PARM     PCB01
          •
          C           PARM     PCB05
          •
          C           SETON                LR
```

RPG II PCB MASK

```
Position  6 7  14    19 20      43      46 47   49   51 52 53   58
          I PCBij      DS
          I                                 1         8       DBDNij
          I                                 9       1 0       SEGLij
          I                               1 1       1 2       STCDij
          I                               1 3       1 6       PROCij
          I                         B     1 7       2 0       RESRij
          I                               1 7       2 0       RESBij
          I                               2 1       2 8       SEGNij
          I                         B     2 9       3 2    0  KEYLij
          I                               2 9       3 2       KEYBij
          I                         B     3 3       3 6    0  SSGNij
          I                               3 3       3 6       SSGBij
          I                               3 7     n n n       KEYAij
```

RPG II FORMATS

RQDLI (positions 7, 8=b̸b̸,Ln, or SR):

```
Position  6 7 8  18        27 28   32 33            42      56 57
          Cb̸ b̸  func-name RQDLI file-name(opt.)          ind.
```

ELEM (FROM|INTO) (positions 7, 8=b̸b̸,Ln, or SR. Positions 18-27=FROM or INTO):

```
Position  6 7 8  18  27 28  32    43      48 49    52
          Cb̸ b̸  FROM ELEM     var-name     opt.
```

ELEM (PCB) (positions 7, 8=b̸b̸,Ln, or SR):

```
Position  6 7 8  18  27 28  32    43     48 49 52
          Cb̸ b̸   PCB  ELEM    var-name  opt.
```

ELEM (CHKPID) (positions 7, 8=b̸b̸,Ln or SR):

```
Position  6 7 8  18     27  28  32  3342  43    48 49  52
          Cb̸ b̸  CHKPID  ELEM literal var-name   opt.
```

ELEM (SSA) (positions 7, 8=b̸b̸,Ln or SR):

```
Position  6 7 8  18  27 28  32    43     48 49 52
          Cb̸ b̸   SSA  ELEM    var-name  opt.
```

ELEM (SSALIST) (positions 7, 8=b̸b̸,Ln, or SR):

```
Position  6 7 8  18     27  28  32  33          42
          Cb̸ b̸  SSALIST ELEM name-of-SSALIST
```

ELIST (positions 7, 8=b̸b̸,Ln, or SR):

```
Position  6 7 8  18                  27  28  32
          Cb̸ b   name-of-SSALIST  ELIST
```

⑭

USSA (positions 7, 8=ƀƀ,Ln or SR):

```
Position  6 7 8  18          27 28   32    56                 57
          C ƀ ƀ  segment-name  USSA        command code (opt.)
```

QSSA (positions 7, 8=ƀƀ,Ln or SR):

```
Position  6 7 8  18      27 28 32 33 42 43  48 49  51  54 55 56 57
          C ƀ ƀ  segment-  QSSA  seg-   comp-  length   rel.   comm
                 name            field  value  of seg.  op.    code
                                 name          field           (opt.)
```

ONLINE SCHEDULING CALL:

```
Position  6  18      27 28   32 33    42  43 48     56 57
          C  PCB        RQDLI                       ind.
          C  PSBNAME    ELEM  psb-nam
          C  SET        ELEM              BUIB
```

ONLINE TERMINATION CALL:

```
Position  6  18  27 28  32
          C  TERM RQDLI
```

DB-File DEFINITION (position 15=I, U, or O):

```
Position  6 7       14 15 16  19     24           27    40   46
          F file-name   I  D   F     max. seg. length        DB
```

CONTINUATION (K line):

```
Position  6  24                  27      53 54  59 60    65
          F  PCB key length (opt.)        K    PCB      PCBij
```

/INSERT STATEMENT (position 6=H, F, E, L, I, C, or O):

```
Position  6 7       14 15       16 17      24     50      74
          H /INSERT  sub-lib.    •  book name      comment
                     name or
                     ƀ
```

RPG II SSA DEFINITION EXAMPLE (qualified SSA):

```
Position  6 7       14 19 20     46 47    50 51    53      58
          I QULSSA     DS
          I                           1         8     SGNAME
          I                           9         9     SQUAL
          I                          1 0       1 7    KEYNAM
          I                          1 8       1 9    SOPR
          I                          2 0       2 5    SKEYVA
          I                          2 6       2 6    SENDCH
          •
Position  6  18      27 28  32 33      42 43     48
          C            MOVE  'ROOT     ' SGNAME
          C            MOVE  '('         SQUAL
          C            MOVE  'KEY      ' KEYNAM
          C            MOVE  ' = '       SOPR
          C            MOVE  'VVVVVV'    SKEYVA
          C            MOVE  ')'         SENDCH
          •
          C  SSA       ELEM              QULSSA
```

⑮

RPG II TRANSLATOR JOB CONTROL

JCL for Translator output on SYSPCH:

```
// JOB T
// EXEC  RPGIXLTR      (Translator)
     :                 (Source to be translated)
/*
/&
```

JCL for Translator output on SYS002:

```
// JOB TRPG
// UPSI 01
// EXEC RPGIXLTR
     :
/*
// EXEC RPGII
/&
```

JCL for Translator output on SYS003:

```
// JOB TAR
// UPSI 10
// EXEC RPGIXLTR
     :
/*
// EXEC RPGAUTO
/&
```

STATUS CODES

	DATA BASE CALLS											
STATUS CODES	GU GHU	GN GHN	GNP GHNP	DLET	REPL	ISRT (LOAD)	ISRT (ADD)	CHKP	CALL COMPLETED	ERROR IN CALL or CONVERSION	I/O or SYSTEM ERROR	DESCRIPTION
AB	X	X	X	X	X	X	X			X		SEGMENT I/O AREA REQUIRED, NONE SPECIFIED IN CALL
AC	X	X	X			X	X			X		HIERARCHICAL ERROR IN SSAs
AD										X		INVALID FUNCTION PARAMETER
AH						X	X			X		CALL REQUIRES SSAs, NONE PROVIDED
AI	X	X	X	X	X	X	X				X	DATA MANAGEMENT OPEN ERROR
AJ	X	X	X	X	X	X	X			X		INVALID SSA QUALIFICATION FORMAT OR COMMAND CODE
AK	X	X	X			X	X			X		INVALID FIELD NAME IN CALL
AM	X	X	X	X	X	X	X			X		CALL FUNCTION NOT COMPATIBLE WITH PROCESSING OPTION OR SEGMENT OR PATH SENSITIVITY
AO	X	X	X	X	X	X	X				X	I/O ERROR
DA					X					X		SEGMENT KEY FIELD HAS BEEN CHANGED
DJ				X	X					X		NO PRECEDING SUCCESSFUL GET HOLD CALL
DX				X						X		VIOLATED DELETE RULE
GA		★	★						X			CROSSED HIERARCHICAL BOUNDARY INTO HIGHER LEVEL (RETURNED ONLY ON CALLS WITH NO SSA SPECIFIED)
GB		★										END OF DATA SET, LAST SEGMENT REACHED
GE	★	★	★				★					SEGMENT OR PARENT SEGMENT NOT FOUND
GK		★	★						X			DIFFERENT SEGMENT TYPE AT SAME LEVEL RETURNED (RETURNED ON UNQUALIFIED CALLS ONLY)
GP			X							X		A GNP CALL AND NO PARENT ESTABLISHED, OR REQUESTED SEGMENT LEVEL NOT LOWER THAN PARENT LEVEL
II							★					SEGMENT TO INSERT ALREADY EXISTS IN DATA BASE OR IS NONUNIQUE
IX							X			X		VIOLATED INSERT RULE
KA	X	X	X	X	X	X	X			X		NUMERIC TRUNCATION ERROR DURING CONVERSION
KB	X	X	X	X	X	X	X			X		CHARACTER TRUNCATION ERROR DURING CONVERSION
KC	X	X	X	X	X	X	X			X		INVALID PACKED/ZONED DECIMAL CHARACTER DURING CONVERSION
KD	X	X	X	X	X	X	X			X		TYPE CONFLICT DURING CONVERSION
KE					X					X		REPLACE VIOLATION
LB						★						SEGMENT TO INSERT ALREADY EXISTS IN DATA BASE OR IS NONUNIQUE
LC						★						KEY FIELD OF SEGMENTS OUT OF SEQUENCE
LD						★						NO PARENT FOR THIS SEGMENT HAS BEEN LOADED
LE						★						SEQUENCE OF SIBLING SEGMENT NOT THE SAME AS DBD SEQUENCE
NA					X					X		DATA IN SEARCH OR SUBSEQUENCE FIELD HAS BEEN CHANGED
NE				X	X		X		X		X	INDEX MAINTENANCE CANNOT FIND SEGMENT
NI				X	X	X	X				X	INDEX MAINTENANCE UNABLE TO OPEN INDEX DATA BASE
					X	X	X			X		DUPLICATE KEY FOUND FOR INDEX DATA BASE
NO				X	X	X	X				X	I/O ERROR
						X						INSERTION OF DUPLICATE SECONDARY INDEX POINTER SEGMENT
RX					X					X		VIOLATED REPLACE RULE
V1					X	X	X			X		INVALID LENGTH FOR VARIABLE LENGTH SEGMENT
XD								X				ERROR DURING DATA BASE BUFFER WRITE OUT
XH								X				DATA BASE LOGGING NOT ACTIVE
XR								X				ERROR DURING CHECKPOINT PROCESSING FOR MPS RESTART
b̸b̸	★	★	★	★	★	★	★	★	★			CALL COMPLETED SUCCESSFULLY

★ Indicates status code that could be expected as normal situation.

X Indicates status code that could be expected as an error situation.

SX24-5103-4

SX24-5103-04