```
// DLBL IJSYSIN,'COBOL TRANSLATION',yy/ddd
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=volid,SHR
// OPTION SYM,ERRS,NODECK,CATAL
   PHASE COBSAMPL,*
   INCLUDE DFHECI
// EXEC FCOBOL,SIZE=...
// EXEC LNKEDT
/&
// JOB RESET
CLOSE SYSIPT,00C
/&
```

Batch and MPS batch

```
// JOB COBSAMPL
// DLBL IJSYSPH,'COBOL TRANSLATION',yy/ddd
// EXTENT SYSPCH,balance of extent information
ASSGN SYSPCH,DISK,VOL=volid,SHR
// EXEC DFHECP1$,SIZE=...
 CBL LIB,XOPTS(DLI)
       •
       •
SOURCE DECK
       •
       •
/*
CLOSE SYSPCH,PUNCH
// DLBL IJSYSIN,'COBOL TRANSLATION',yy/ddd
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=volid,SHR
// OPTION SYM,ERRS,NODECK,CATAL
   PHASE COBSAMPL,*
   INCLUDE DLZLICBL
   INCLUDE DLZBPJRA
// EXEC FCOBOL,SIZE=...
   ENTRY CBLCALLA
// EXEC LNKEDT
/&
// JOB RESET
CLOSE SYSIPT,00C
/&
```

Execution

Job Control Statements

Batch

```
//  JOB     UPDATE
//  UPSI    00000000
//  ASSGN   SYS011,182
//  TLBL    LOGOUT
//  DLBL    INVPRT1,'INVENTORY',99/365,VSAM
//  DLBL    INVPRTZ,'INVENTORY-OFLOW',99/365,VSAM
//  EXEC    DLZRRC00,SIZE=...
DLI,INVUPDT,INVMSTR
            •
            •
            DATA CARDS IF REQUIRED
            •
            •
/*
/&
```

MPS Batch

```
//  JOB     UPDATE
//  UPSI    00000000
//  EXEC    DLZMPI00,SIZE=...
DLI,INVUPDT,INVMSTR
            •
            •
            DATA CARDS IF REQUIRED
            •
            •
/*
/&
```

MPS Batch Using MPS Restart

```
//  JOB     UPDATE
//  MTC     REW,280
//  ASSGN   SYS100,280
//  EXEC    DLZMPI00,SIZE=...
DLR,INVUPDT,INVMSTR
            •
            •
            DATA CARDS IF REQUIRED
            •
            •
/*
/&
```

Restarting an MPS Restart Program

```
//  JOB     UPDATE
//  MTC     REW,280
//  ASSGN   SYS100,280
//  RSTRT   SYS100,0010
DLR,INVUPDT,INVMSTR
/*
/&
```

Note: Obtain actual values for ASSGN, TLBL, DLBL, EXTENT, and UPSI statements from data base administration.

Parameter Statement

```
{DLI},progname,psbname[,{buff}]
{DLR}                   {1   }
[,HDBFR=({bufno}[,dbdname1,dbdname2,...])][,...]
        {32  }
[,HSBFR=({indno},{ksdsbuf},[{esdsbuf}],dbdname3)][,...]
        {3   } {2      } {2      }
[,TRACE=modname][,ASLOG=YES]
[,LOG=({TAPE },{PAUSE  })]
      {DISK1} {NOPAUSE}
      {DISK2}
```

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.

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: High Level Programming Interface* for details of parameters and keywords.

IBM

DL/I DOS/VS Reference Summary: High Level Programming Interface

SX24-5120-2

Third Edition (January 1984)

This edition, SX24-5120-2, is a major revision of SX24-5120-1 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: High Level Programming Interface*, SH24-5009.

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.

Printed in U.S.A.

Trigger	Function	PCB Option	Key Feedback Option
{EXECUTE} DLI {EXEC }	{GET NEXT} {GN }	[USING PCB(exp)]	[KEYFEEDBACK(ref) [FEEDBACKLEN(exp)]]
{EXECUTE} DLI {EXEC }	{GET NEXT IN PARENT} {GNP }		[KEYFEEDBACK(ref) [FEEDBACKLEN(exp)]]
{EXECUTE} DLI {EXEC }	{GET UNIQUE} {GU }	[USING PCB(exp)]	[KEYFEEDBACK(ref) [FEEDBACKLEN(exp)]]
{EXECUTE} DLI {EXEC }	{INSERT} {ISRT }	[USING PCB(exp)]	
{EXECUTE} DLI {EXEC }	{REPLACE} {REPL }	[USING PCB(exp)]	
{EXECUTE} DLI {EXEC }	{DELETE} {DLET }	[USING PCB(exp)]	
{EXECUTE} DLI {EXEC }	LOAD	[USING PCB(exp)]	
{EXECUTE} DLI {EXEC }	{CHECKPOINT} {CHKP }		
{EXECUTE} DLI {EXEC }	{SCHEDULE} {SCHD }		
{EXECUTE} DLI {EXEC }	{TERMINATE} {TERM }		

Notes:

1. Wherever the word "name" appears in the syntax of a command, the following rules apply:
   - You must code an identifier consisting of alphabetic and numeric characters only, with the first character being alphabetic; or a literal string constant enclosed in single quotes. Both are interpreted as being a literal string.
   - A variable can be specified for "name" only on a LOAD command. In this case, the identifier must be enclosed in a double set of parentheses to indicate that it represents a variable rather than a constant. For example: SEGMENT((VARNAME)).
2. Wherever the word "expression" (abbreviated as "exp") appears in the syntax of a command (except the CHECKPOINT command), it is to be replaced by an expression in the host language. In COBOL, "exp" is any valid integer variable or integer constant. In PL/I, "exp" is any valid expression that converts to the integer data type. If a variable is specified for "expression," it should be declared as halfword binary.

   For CHECKPOINT, "expression" should convert to an eight-byte character string.
3. Wherever the word "reference" (abbreviated as "ref") appears in the syntax of a command, it is to be replaced by a name previously defined in the host language of the application program.
4. The following examples illustrate the acceptable methods of specifying arguments for the various command options.

Parent Segment(s) (0-14)	Object Segment	Delimiter
[[{FIRST}][VARIABLE] SEGMENT(name) {LAST } [INTO(ref)[LOCKED][OFFSET(exp)][SEGLENGTH(exp)]] [WHERE(name op ref[{AND}name op ref]...) {OR } [FIELDLENGTH(exp[,exp]...)]]]...	[[{FIRST}][VARIABLE] SEGMENT(name)] {LAST } INTO(ref)[LOCKED][OFFSET(exp)][SEGLENGTH(exp)] [WHERE(name op ref[{AND}name op ref]...) {OR } [FIELDLENGTH(exp[,exp]...)]]...	{END-EXEC} {;}
[[{FIRST}][VARIABLE] SEGMENT(name) {LAST } [INTO(ref)[LOCKED][OFFSET(exp)][SEGLENGTH(exp)]] [WHERE(name op ref[{AND}name op ref]...) {OR } [FIELDLENGTH(exp[,exp]...)]]...	[[{FIRST}][VARIABLE] SEGMENT(name)] {LAST } INTO(ref)[LOCKED][OFFSET(exp)][SEGLENGTH(exp)] [WHERE(name op ref[{AND}name op ref]...) {OR } [FIELDLENGTH(exp[,exp]...)]]...	{END-EXEC} {;}
[[LAST][VARIABLE] SEGMENT(name) [INTO(ref)[LOCKED][OFFSET(exp)][SEGLENGTH(exp)]] [WHERE(name op ref[{AND}name op ref]...) {OR } [FIELDLENGTH(exp[,exp]...)]]...	[LAST][VARIABLE] SEGMENT(name) INTO(ref)[LOCKED][OFFSET(exp)][SEGLENGTH(exp)] [WHERE(name op ref[{AND}name op ref]...) {OR } [FIELDLENGTH(exp[,exp]...)]]...	{END-EXEC} {;}
[[{FIRST}][VARIABLE] SEGMENT(name) {LAST } [FROM(ref) [SEGLENGTH(exp)]] [WHERE(name op ref[{AND}name op ref]...) {OR } [FIELDLENGTH(exp[,exp]...)]]...	[{FIRST}][VARIABLE] SEGMENT(name) {LAST } FROM(ref)[OFFSET(exp)][SEGLENGTH(exp)]	{END-EXEC} {;}
[[VARIABLE] SEGMENT(name) FROM(ref)[OFFSET(exp)][SEGLENGTH(exp)]]...	[VARIABLE] SEGMENT(name) FROM(ref)[OFFSET(exp)][SEGLENGTH(exp)]	{END-EXEC} {;}
	[VARIABLE] SEGMENT(name) FROM(ref)[SEGLENGTH(exp)]	{END-EXEC} {;}
	[VARIABLE] SEGMENT(name) FROM(ref)[SEGLENGTH(exp)]	{END-EXEC} {;}
ID(exp)		{END-EXEC} {;}
PSB(name)		{END-EXEC} {;}
		{END-EXEC} {;}

Examples:

- SEGMENT(name)
  For GET, INSERT, REPLACE, DELETE, and LOAD:

  ```
  SEGMENT(ARTIST)
  ```

  or

  ```
  SEGMENT('ARTIST')
  ```

  where ARTIST is the name of a segment type defined to DL/I during data base generation. It is not defined in the application program.

  For LOAD only, this additional form is allowed:

  ```
  SEGMENT((SEGNAME))
  ```

  where SEGNAME is defined in the application program as either

  ```
  77 SEGNAME PIC X(8) VALUE 'ARTIST'.      (COBOL)
  ```

  or

  ```
  DECLARE SEGNAME CHAR(8) INIT('ARTIST'); (PL/I)
  ```

- INTO(reference) or FROM(reference)
  For GET commands:

  ```
  INTO(INAREA)
  ```

  where INAREA is defined in the application program as either

  ```
  77 INAREA PIC X(nn).       (COBOL)
  ```

  or

  ```
  DECLARE INAREA CHAR(nn);  (PL/I)
  ```

For INSERT, REPLACE, DELETE, and LOAD:

```
FROM(OUTAREA)
```

where OUTAREA is defined in the application program in the same way as INAREA.

nn is equal to the length of the segment to be processed.

- SEGLENGTH(exp), FIELDLENGTH(exp), OFFSET(exp), PCB(exp), or FEEDBACKLEN(exp)
  For data base function commands:

```
SEGLENGTH(4)
FIELDLENGTH(4)
OFFSET(4)
PCB(4)
FEEDBACKLEN(4)
```

or

```
SEGLENGTH(NUM)
FIELDLENGTH(NUM)
OFFSET(NUM)
PCB(NUM)
FEEDBACKLEN(NUM)
```

where NUM is defined in the application program as either

```
77 NUM COMP PIC S9999 VALUE IS +4.  (COBOL)
```

or

```
DECLARE NUM FIXED BIN(15) INIT(4);  (PL/I)
```

- WHERE(name op reference)[{AND|OR} name op reference] . . .) [FIELDLENGTH(expression[,expression] . . .)]

For GET and INSERT commands:

```
WHERE(TYPENO<C800 AND TYPENO>C700 AND DESCR=JOBS
or MONTHS>C100)
FIELDLENGTH (3,3,8,3)
```

where TYPENO, DESCR, and MONTHS are the names of fields defined to DL/I during data base generation. They are not defined in the application program.

C800, C700, JOBS, and C100 are defined in the application program as either

(COBOL)

```
77  C800    PIC S999 COMP  VALUE IS 800.
77  C700    PIC S999 COMP  VALUE IS 700.
77  C100    PIC S999 COMP  VALUE IS 100.
77  JOBS    PIC X(nn)      VALUE 'COMIC'.
```

or

(PL/I)

```
DECLARE C800   FIXED BIN(11) INIT (800);
DECLARE C700   FIXED BIN(11) INIT (700);
DECLARE C100   FIXED BIN(11) INIT (100);
DECLARE JOBS   CHAR(nn)      INIT ('COMIC');
```

nn is equal to the length of the field as defined to DL/I. In the above example, "(nn)" is "(8)".

FIELDLENGTH is a one-for-one integer description of the length of field of C800, C700, JOBS, and C100.

- PSB(name)
  For the SCHEDULE command:

```
PSB(PSB01)
```

or

```
PSB('PSB01')
```

where PSB01 is the name of a PSB defined to DL/I during PSB generation.

- ID(expression)
  For the CHECKPOINT command:

```
ID(CHKPID)
```

or

```
ID('CHKP1000')
```

where CHKPID is defined in the application program as either

```
77 CHKPID PIC X(8) VALUE 'CHKP1000'.     (COBOL)
```

or

```
DECLARE CHKPID CHAR(8) INIT('CHKP1000'); (PL/I)
```

- KEYFEEDBACK(reference)
  Using the KEYFEEDBACK keyword:

```
KEYFEEDBACK(SKILLKEY)
```

where SKILLKEY is defined in the application program as either

```
01   SKILLKEY PIC X(nn).        (COBOL)
```

or

```
DCL  SKILLKEY CHAR(nn);         (PL/I)
```

nn is less than or equal to the length of the concatenated key as defined to DL/I.

Description of Functions:

CHECKPOINT (CHKP) — Commit changes to DL/I data bases currently pending for this task.

DELETE (DLET) — Delete segment and all of its physically dependent segments. Segment must be the last segment successfully retrieved using the same PCB.

GET NEXT (GN) — Retrieve next object segment (in hierarchical order) and, optionally, its parent(s).

GET NEXT IN PARENT (GNP) — Retrieve next object segment (in hierarchical order) and, optionally, its parent(s). Only segments under parentage previously established by successful GET UNIQUE or GET NEXT can be retrieved.

GET UNIQUE (GU) — Retrieve specified object segment and, optionally, its parent(s).

INSERT (ISRT) — Insert object segment and, optionally, its parent(s).

LOAD — (Batch only) Add a segment to a newly created data base.

REPLACE (REPL) — Replace object segment and, optionally, its parent(s). Segment(s) must be the last segment(s) successfully retrieved using the same PCB.

SCHEDULE (SCHD) — (Online only) Schedule a PSB (set of PCBs) for use by the task. Required before a DL/I data base command can be issued online.

TERMINATE (TERM) — (Online only) Release a PSB and commit changes to DL/I data bases currently pending for this task. Required before another PSB can be scheduled for the task.

Description of Options:

FEEDBACKLEN(expression) — Length of the area referenced by KEYFEEDBACK. Required with KEYFEEDBACK for COBOL.

FIELDLENGTH(expression) — Length of field used in WHERE option as defined to DL/I. Required with WHERE for COBOL.

FIRST — Start with first occurrence of segment type, under its parent, to satisfy this level of the command.

FROM(reference) — Area in program that contains segment to be used to update the data base.

ID(expression) — 8 character value naming a DL/I checkpoint.

INTO(reference) — Area in program large enough to contain segment being retrieved.

KEYFEEDBACK(reference) — Name of the area where the key feedback information is to be placed.

LAST — Select last occurrence of object segment from among segments having same immediate parent.

LOCKED — No other application can access this segment until this application terminates or takes a checkpoint.

OFFSET(expression) — Length of the logical child segment in a concatenated segment. The destination parent in the concatenated segment must be a variable length segment when OFFSET is specified.

PSB(name) — Name of the PSB to be scheduled as defined to DL/I.

SEGLENGTH(expression) — Length of the preceding SEGMENT as defined to DL/I. Required with SEGMENT for COBOL.

SEGMENT(name) — Name of the segment as defined to DL/I.

USING PCB(expression) — Index of the PCB to be used in this command. Default is (1), the first PCB.

VARIABLE — The following SEGMENT is a variable length DL/I segment.

WHERE(name op reference) — Name of the field in a segment and the range of values it can have to satisfy the command.

WHERE option operator values:

>	(greater than)
<	(less than)
=	(equal to)
¬=	(not equal to)
<=	(less than or equal to)
>=	(greater than or equal to)
=<	(equal to or less than)
¬>	(not greater than)
=>	(equal to or greater than)
¬<	(not less than)

IBM®

SX24-5120-02

STATUS CODES

STATUS CODE	COMMANDS: GET UNIQUE	GET NEXT	GET NEXT IN PARENT	DELETE	REPLACE	LOAD	INSERT	CHECKPOINT	SCHEDULE	TERMINATE	COMMAND COMPLETED	ERROR IN CMD 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 COMMAND
AC	X	X	X			X	X					X		HIERARCHICAL ERROR IN SEGMENT SELECTION
AD						X			X	X		X		INVALID FUNCTION PARAMETER
AH						X	X					X		COMMAND REQUIRES SEGMENT SELECTION, 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 QUALIFIED SEGMENT SELECTION FORMAT
AK	X	X	X			X	X					X		INVALID FIELD NAME IN COMMAND
AM	X	X	X	X	X	X	X					X		COMMAND 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 COMMAND
DX				X								X		VIOLATED DELETE RULE
GA		★	★								★			CROSSED HIERARCHICAL BOUNDARY INTO HIGHER LEVEL (RETURNED ONLY ON COMMANDS WITHOUT SEGMENT SELECTION)
GB		★												END OF DATA SET, LAST SEGMENT REACHED
GE	★	★	★				★							SEGMENT OR PARENT SEGMENT NOT FOUND
GK		★	★								★			DIFFERENT SEGMENT TYPE AT SAME LEVEL RETURNED (RETURNED ON UNQUALIFIED COMMANDS ONLY)
GP			X									X		A GNP COMMAND AND NO PARENT ESTABLISHED, OR REQUESTED SEGMENT LEVEL NOT LOWER THAN PARENT LEVEL
II							★							SEGMENT TO INSERT ALREADY EXISTS IN DATA BASE OR IS NON-UNIQUE
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 NON-UNIQUE
LC						X								KEY FIELD OF SEGMENTS OUT OF SEQUENCE
LD						X								NO PARENT FOR THIS SEGMENT HAS BEEN LOADED
LE						X								SEQUENCE OF SIBLING SEGMENT NOT THE SAME AS DBD SEQUENCE
NA					X							X		DATA IN SEARCH OR SUBSEQUENCE FIELD HAS BEEN CHANGED
NE				★	★		★				★		★	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

★ Indicates status code returned in DIB.

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

STATUS CODES (continued)

STATUS CODE	COMMANDS: GET UNIQUE	GET NEXT	GET NEXT IN PARENT	DELETE	REPLACE	LOAD	INSERT	CHECKPOINT	SCHEDULE	TERMINATE	COMMAND COMPLETED	ERROR IN CMD or CONVERSION	I/O or SYSTEM ERROR	DESCRIPTION
NO				X	X	X	X						X	I/O ERROR
						X								LOADING DUPLICATE SECONDARY INDEX POINTER SEGMENT
RX					X							X		VIOLATED REPLACE RULE
TA									X				X	PSB NOT IN DIRECTORY
TC									X			X		TASK ALREADY SCHEDULED
TE									X				X	PSB INITIALIZATION ERROR
TF									X				X	PSB NOT AUTHORIZED
TG										★				TERM ATTEMPTED WHEN PSB NOT SCHEDULED
TH	X	X	X	X	X	X	X	X				X		DATA BASE COMMAND ATTEMPTED WHEN PSB NOT SCHEDULED
TI							X					X		INVALID PATH INSERT
TJ	X	X	X	X	X	X	X	X	X	X			X	DL/I NOT ACTIVE
TK									X				X	DATA BASE NOT ACTIVE
TL									X				X	SCHEDULING CONFLICT WITH MPS TASK
TN	X	X	X	X	X	X	X	X	X	X		X	X	INVALID SYSTEM DIB ADDRESS
TO					X							X		PATH REPLACE ERROR
TP	X	X	X	X	X	X	X	X				X		INVALID PCB INDEX
V1					X	X	X					X		INVALID LENGTH FOR VARIABLE LENGTH SEGMENT
V2	X	X	X	X	X	X	X					X		SEGMENT LENGTH MISSING OR INVALID
V3	X	X	X				X					X		FIELD LENGTH MISSING OR INVALID
V4	X	X	X	X	X	X	X					X		INVALID LENGTH FOR VARIABLE LENGTH SEGMENT
V5	X	X	X		X		X					X		INVALID OFFSET
V8	X	X	X									X		KEY FEEDBACK LENGTH MISSING OR INVALID
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
bb	★	★	★	★	★	★	★	★	★	★	★			COMMAND COMPLETED SUCCESSFULLY

★ Indicates status code returned in DIB.

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

DIB

The DIB variables are defined for each host language as shown below.

```
For COBOL:                    For PL/I:

DIBVER    PICTURE X(2)        DIBVER    CHAR(2)
DIBSTAT   PICTURE X(2)        DIBSTAT   CHAR(2)
DIBSEGM   PICTURE X(8)        DIBSEGM   CHAR(8)
DIBFLAG   PICTURE X(1)        DIBFLAG   CHAR(1)
DIBSEGLV PICTURE X(2)         DIBSEGLV CHAR(2)
DIBKFBL   PIC S9(4) COMP      DIBKFBL FIXED BIN(15)
```

- DIBVER is the version of the translator used to translate the application program. (11)
- DIBSTAT is the DL/I status code.
- DIBSEGM is the name of the lowest level segment retrieved. DIBSEGM should be ignored following a CHECKPOINT, SCHEDULE, or TERMINATE command.
- DIBFLAG is a flag indicating that an online task had to wait for a resource owned by an MPS batch task (DIBFLAG = X'FF').
- DIBSEGLV is the hierarchical level of the lowest level segment retrieved. DIBSEGLV should be ignored following a CHECKPOINT, SCHEDULE, or TERMINATE command.
- DIBKFBL is the actual length of the concatenated key in the PCB when KEYFEEDBACK is specified on the DL/I command.

Entry to Batch and MPS Batch Programs

COBOL: The following statement must be the first in the procedure division of the main procedure:

```
ENTRY 'DLITCBL'.
```

PL/I: There is no special requirement.

Terminating the Program

Batch and MPS Batch

COBOL:

```
GOBACK.
```

PL/I:

```
RETURN;
```

Note: STOP RUN can not be used in a COBOL program as an exit statement, since control would not be returned to DL/I to allow it to release its resources and close the data bases and log. However, it can be coded after GOBACK. This prevents the compiler from giving a warning message and automatically generating a STOP RUN.

Online

	Command Level	Macro Level
COBOL	EXEC CICS RETURN END-EXEC.	DFHPC TYPE=RETURN
PL/I	EXEC CICS RETURN;	DFHPC TYPE=RETURN

Compilation and Link-editing

For PL/I:

Online

```
// JOB PLISAMPL
// DLBL IJSYSPH,'PL/I TRANSLATION',yy/ddd
// EXTENT SYSPCH,balance of extent information
ASSGN SYSPCH,DISK,VOL=volid,SHR
// EXEC DFHEPP1$,SIZE=...
*PROCESS INCLUDE,XOPTS(CICS,DLI);
        •
        •
SOURCE DECK
        •
        •
/*
CLOSE SYSPCH,PUNCH
// DLBL IJSYSIN,'PL/I TRANSLATION',yy/ddd
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=volid,SHR
// OPTION CATAL
   PHASE PLISAMPL,*
   INCLUDE DFHPL1I
// EXEC PLIOPT,SIZE=...
// EXEC LNKEDT
/&
// JOB RESET
CLOSE SYSIPT,00C
/&
```

Batch and MPS batch

```
// JOB PLISAMPL
// DLBL IJSYSPH,'PL/I TRANSLATION',yy/ddd
// EXTENT SYSPCH,balance of extent information
ASSGN SYSPCH,DISK,VOL=volid,SHR
// EXEC DFHEPP1$,SIZE=...
*PROCESS INCLUDE,XOPTS(DLI);
        •
        •
SOURCE DECK
        •
        •
/*
CLOSE SYSPCH,PUNCH
// DLBL IJSYSIN,'PL/I TRANSLATION',yy/ddd
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=volid,SHR
// OPTION CATAL
   PHASE PLISAMPL,*
// EXEC PLIOPT,SIZE=...
   INCLUDE DLZLIPLI
   INCLUDE IBMBPJRA
   ENTRY DLZLIPLI
// EXEC LNKEDT
/&
// JOB RESET
CLOSE SYSIPT,00C
/&
```

For COBOL:

Online

```
// JOB COBSAMPL
// DLBL IJSYSPH,'COBOL TRANSLATION',yy/ddd
// EXTENT SYSPCH,balance of extent information
ASSGN SYSPCH,DISK,VOL=volid,SHR
// EXEC DFHECP1$,SIZE=...
 CBL LIB,XOPTS(CICS,DLI)
        •
        •
SOURCE DECK
        •
        •
/*
CLOSE SYSPCH,PUNCH
```

(12)