Using the Power of HLASM's I/O Exits

SHARE 115 in Boston

August, 2010

John R. Ehrman
ehrman@us.ibm.com

IBM Silicon Valley Laboratory
555 Bailey Avenue
San Jose, CA 95141

Synopsis:

This tutorial introduces the powerful capabilities of the I/O exits supported by the High Level Assembler, and provides examples of their use.

The examples in this document are for purposes of illustration only, and no warranty of correctness or applicability is implied or expressed.

Using the Power
of
HLASM's I/O Exits

SHARE 115 in Boston

August, 2010

John R. Ehrman
ehrman@us.ibm.com

IBM Silicon Valley Laboratory
555 Bailey Avenue
San Jose, CA 95141

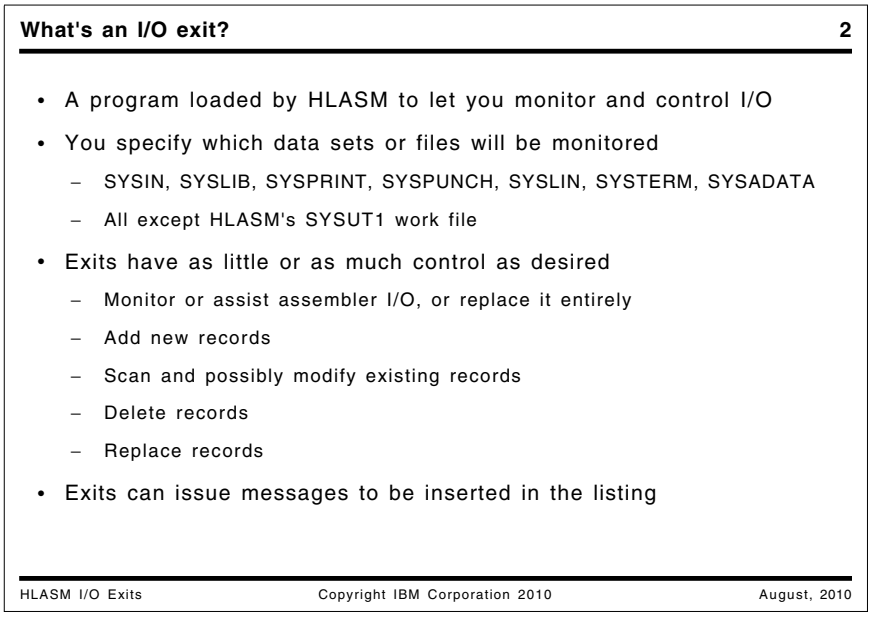
High Level Assembler Input-Output Exits

Overview **1**

- What is an I/O exit, and what does it do?
- How do I create one?
- How do I invoke it and control it?
- How does HLASM interact with exits?
- How does an exit control I/O?
  - Some typical exit actions
- Other considerations
- IBM-supplied exits
- Example of a simple LISTING exit

Overview

HLASM supports powerful and flexible exit interfaces for all of its input and output data sets (except its utility data set, with DDname SYSUT1). These exits allow you to take many different actions on records read and written by the assembler.



What is an I/O exit, and what does it do?

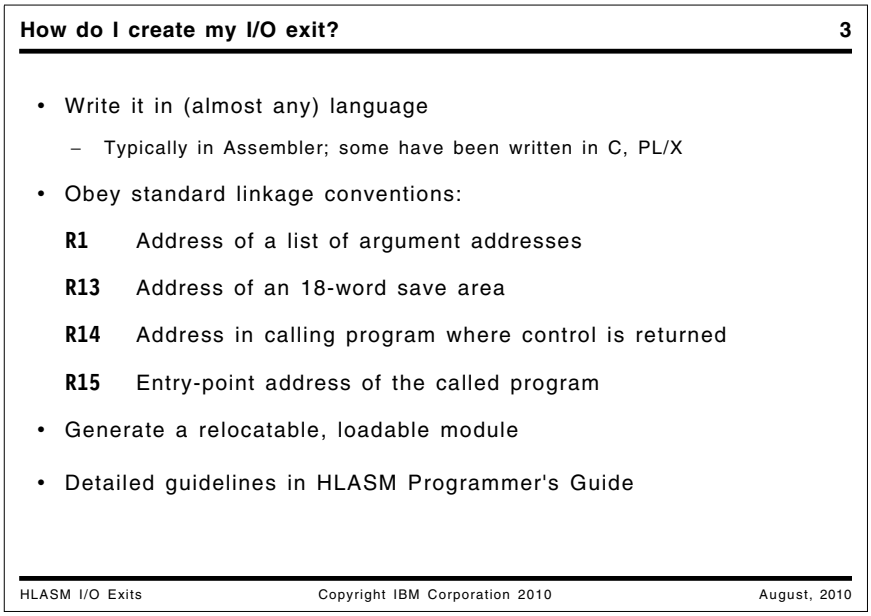
An I/O exit is a program that HLASM loads into memory during initialization. You specify it in the assembler options when you first invoke the assembler. You can request that your exit work with any of the assembler's "external" files (that is, any DDname other than SYSUT1, which HLASM may need to use as a utility file).

An exit routine can examine, modify, add, or delete records as they pass to and from the assembler; it can also share I/O activity with the assembler, or replace HLASM's I/O activity entirely by its own.

It can also create messages to be inserted in the program's listing.

All exits use the same interface, and a single exit routine can support more than one type of exit.

Full information about HLASM's interactions with I/O exits is provided in in Chapter 4 of the *High Level Assembler Programmer's Guide*, including a detailed example of an exit.

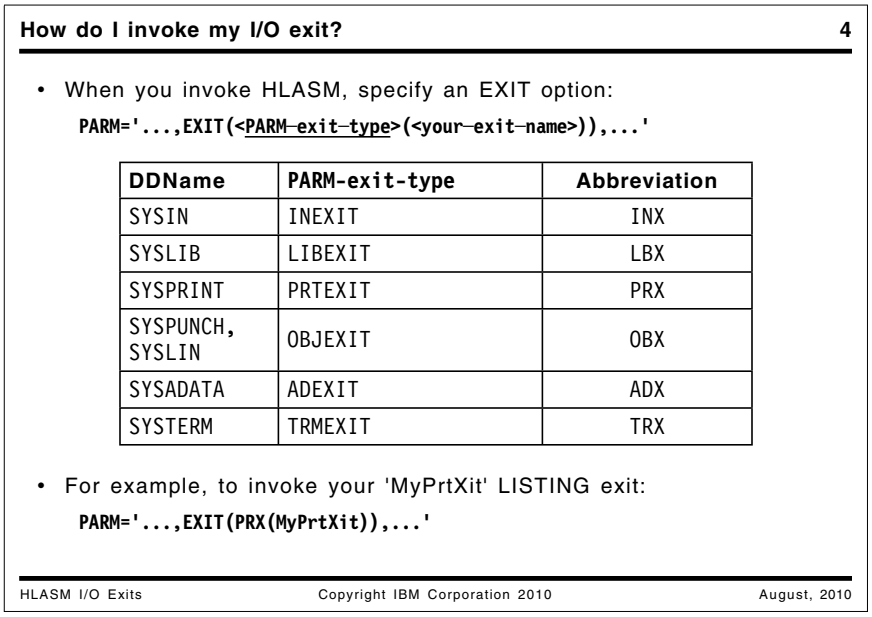


How do I create my I/O exit?

First, decide which of the files you are interested in: you can write separate exit routines for each, or a single routine that handles records for more than one file.

You can write your exit in any language that supports standard linkage conventions, but remember that HLASM runs in a “minimal” operating system environment (that is, not under Language Environment).

After your exit routine has been assembled or compiled, use the generated object code to create a relocatable, loadable module. It may have any residence mode (RMode) and addressing mode (AMode) other than 64.



How do I invoke my I/O exit?

You specify the type and name of your exit routine in the options passed to HLASM when it is first called. The types of exits are shown in the slide above.

HLASM scans and sets the values of all invocation parameters, and then loads the exit routine(s) you specified. If any exit routine can't be loaded, HLASM issues a message and terminates the assembly.

Exits are loaded before the options summary is printed, so the summary may show NOEXIT, even if you specify an EXIT option but the corresponding file option is NO.

For example, you might specify parameters like

```
PARM='LIST,OBJECT,EXIT(PRTEXIT(MYPRTXIT)),ADATA'
```

If your parameter string suppresses one of HLASM's files – for example, you specify the NOOBJECT option, as in

```
PARM='EXIT(OBJEXIT(MyObjXit)),NOOBJECT'
```

the OBJEXIT parameter is ignored and HLASM will not attempt to load the exit routine. Remember that the OBJ exit used for both DECK and PUNCH options, but HLASM uses different **exit-type** calls for each.

How do I control my I/O exit? 5

- In two ways:
  1. At the time the exit is first invoked (OPENed) by HLASM
  2. At any time during the assembly

1. At invocation time: you specify an exit-parameter string:

   EXIT(<PARM-exit-type>(<your-exit-name>(<your-exit-parm>)))

   - where **<your-exit-parm>** is 1-64 characters long
     (with paired parenthesess, apostrophes, and ampersands)
   - For example, the single character 'E' will be passed to your exit when it is OPENed:

     EXIT(PRX(MyPrtXit(E)))

HLASM I/O Exits August, 2010

How do I control my I/O exit? ... 6

2. Dynamically: with the assembler's **EXITCTL** statement:

   EXITCTL <EXITCTL-exit-type>,control-value[,control-value]

   - From 1 to 4 32-bit binary **control-values** can be specified;
     4 values are always passed to the exit

If PARM-exit-type is	Then EXITCTL-exit-type is
INEXIT	SOURCE
LIBEXIT	LIBRARY
PRTEXIT	LISTING
OBJEXIT	PUNCH
OBJEXIT	OBJECT
ADEXIT	ADATA
TRMEXIT	TERM

   - Example:

     EXITCTL LISTING,55,23,-1 (4th passed value is 0)

HLASM I/O Exits August, 2010

How can I control my I/O exit?

You can control the behavior of your exit in two ways:

1. You can specify an exit-parameter string of 1 to 64 characters to be passed to your exit routine when it is first called by HLASM.
2. If you want your exit to respond to different conditions during the assembly, you can specify from one to four 32-bit integer values on an EXITCTL statement; these values are provided to your exit routine the next time it is called by HLASM.

Invocation parameters

If you want your exit to prepare itself in different ways at different times, you can specify an exit-parameter string as part of HLASM's EXIT parameter: follow the name of your exit with your exit-parameter enclosed in parentheses, as in

```
PARM='LIST,OBJECT,EXIT(PRTEXIT(MYPRTXIT(ABCDE1234))),ADATA'
```

HLASM preserves your exit-parameter string until your exit is first invoked, when it passes this string and its length to you on the first (OPEN) call to the exit.

If you want to include parentheses, ampersands, or apostrophes in your exit-parameter, be sure they are properly paired so that HLASM can correctly analyze its invocation PARM string.

The EXITCTL statement

The EXITCTL statement lets you control the actions of your I/O exit as the assembly progresses. It is written in the form

```
EXITCTL  exit-type,value-1,value-2,value-3,value-4
```

The four operands of the statement are passed to your exit as 32-bit integer values.

1. All EXITCTL values are initially binary zero, so if any is not specified, zero will be passed.
2. The **value-n** operands following the **exit-type** operand can be specified in either of two ways:
   a. As a decimal value (optionally signed).
   b. As an expression of the form ***±n** where ***** represents the current value of the `value` and is added or subtracted from it.

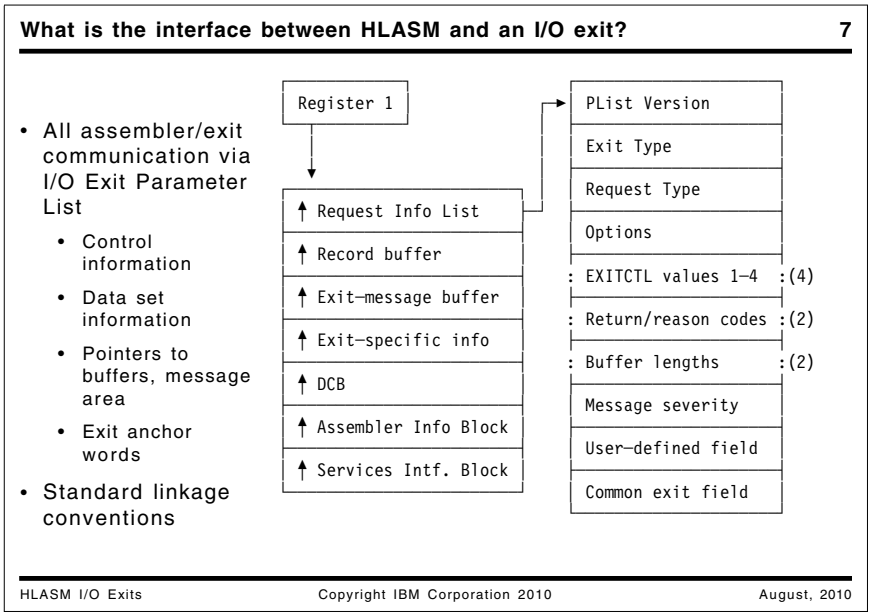
The EXITCTL values are set at the time records are being read or written by the assembler; this means that input exits are active during the earlier phases of the assembly, and output exits are active during the later phases of the assembly. Thus, you should not expect to create complex interactions among different exits that depend on a particular sequence of EXITCTL statements.

The EXITCTL values for each exit type are available:

SOURCE/LIBRARY
at the record following the EXITCTL statement

LISTING/ADATA/TRM
with the output record containing the EXITCTL statement

OBJECT/PUNCH
when the next record is passed to the exit. More than one EXITCTL statement may intervene, because object records contain machine language text from multiple source statements.



What is the interface between HLASM and an I/O exit?

The interface between HLASM and I/O exits establishes a “coroutine” interaction: both the assembler and the exit routine must cooperate, with neither being fully in control of the other. All interactions take place through the parameter list illustrated in Figure 1 on page 8.

Every time your exit routine is called, HLASM puts in General Register 1 the address of a “primary address list” of 6 or 7 addresses:

1. The address of the “Request Information List”. This list is illustrated in Figure 1 on page 8.
2. The address of a record buffer where HLASM places records to be analyzed by your exit, and into which your exit places new or modified records.
3. The address of a 255-byte buffer into which your exit can place messages to be inserted into the listing. Your exit puts the length of the message in the Request Information List.
4. The address of an exit-specific information block, into which HLASM places differing information depending on the type of the exit.
5. The address of HLASM's Data Control Block (DCB), in case your exit needs access to its contents.
6. The address of a 36-byte block of information describing HLASM itself: its version, release, modification level, PTF level, and its operating system environment.
7. This field contains the address of the HLASM Services Interface, which can be used to request that HLASM provide certain system services so that the exit need not invoke the host system environment itself.

   Note: This seventh address may or may not be present. HLASM sets the high-order bit of the last address in this primary address list, so your exit should

inspect that bit of the sixth address to determine whether or not the seventh address is present.

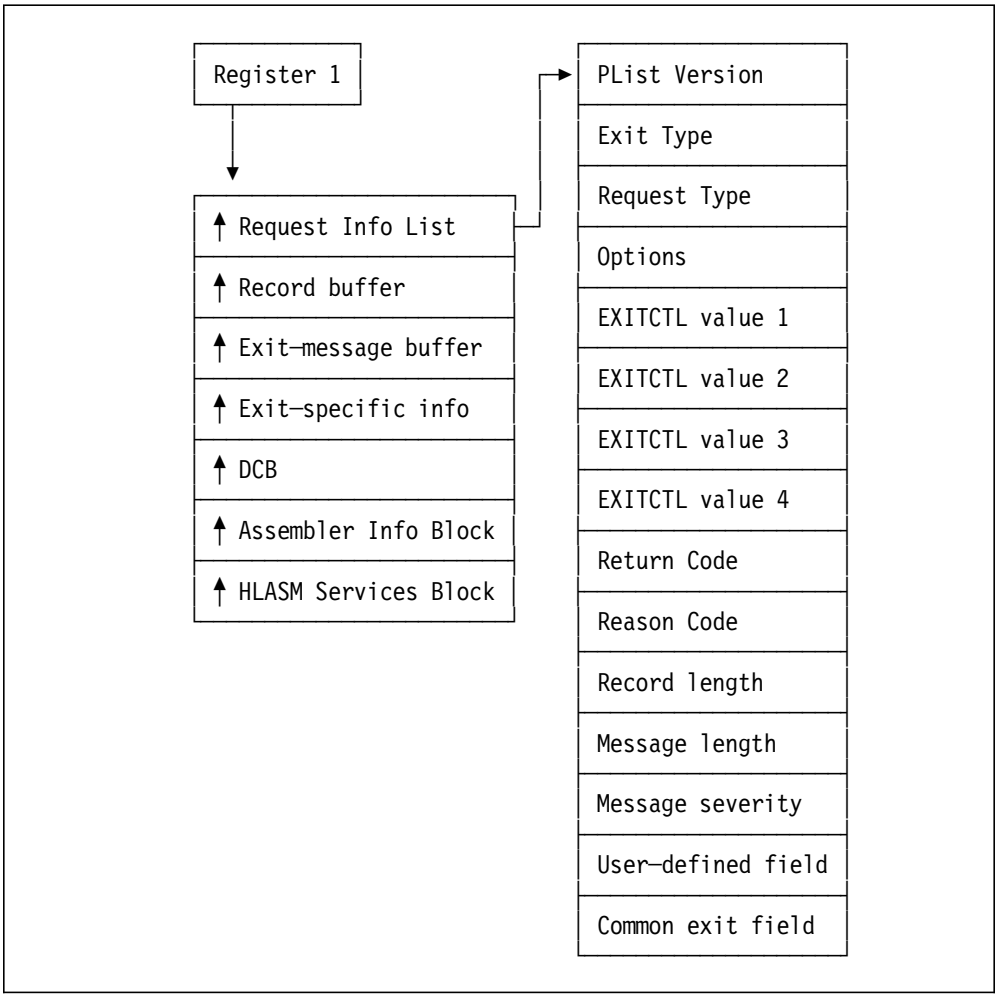


Figure 1. I/O Exit Parameter List

The I/O exit interface uses standard OS linkage conventions, and the parameter list follows standard OS parameter-passing conventions.

Mapping the Communication and Work Areas

The I/O exit parameter lists are mapped by DSECTs generated by the ASMAXITP macro, which is supplied with HLASM in the ASMAMAC library along with the macros used for installing and customizing the assembler.

The first address in the Primary Address List points to the Request Information List, an array of fullword integers. These values control the interactions between HLASM and your exit.

Word 1 This word indicates the version of the parameter list.

Word 2 This word indicates the exit type, and tells the exit routine which file it is expected to work with. The values passed in this word by HLASM, and their symbolic names, are shown in this table:

Exit Type	Value	Symbol
SOURCE	1	AXPTSRC
LIBRARY	2	AXPTLIB
LISTING	3	AXPTLST
PUNCH	4	AXPTPUN
OBJECT	5	AXPTOBJ
ADATA	6	AXPTAD
TERM	7	AXPTTRM

Word 3 This word indicates the request type, such as opening or closing the exit, processing a record, or performing a read or write.

Word 4 This OPTIONS word indicates optional additional information about the exit's activity, such as changes to data set names or the type of record to be processed, depending on the exit type:

- SOURCE/LIBRARY exits: Data set and member name information; if a LIBRARY exit is doing library I/O, it can use this field to pass information back to HLASM.
- LISTING: the type of listing record.
- Others: none, the field is set to zero.

Words 5-8 These words contain the four EXITCTL values most recently specified by the source program.

Word 9 This word contains the Return Code from the exit.

Word 10 This word contains the Reason Code from the exit.

Word 11 This word contains the length in the record buffer of the record provided to the exit by HLASM, or provided by the exit to HLASM.

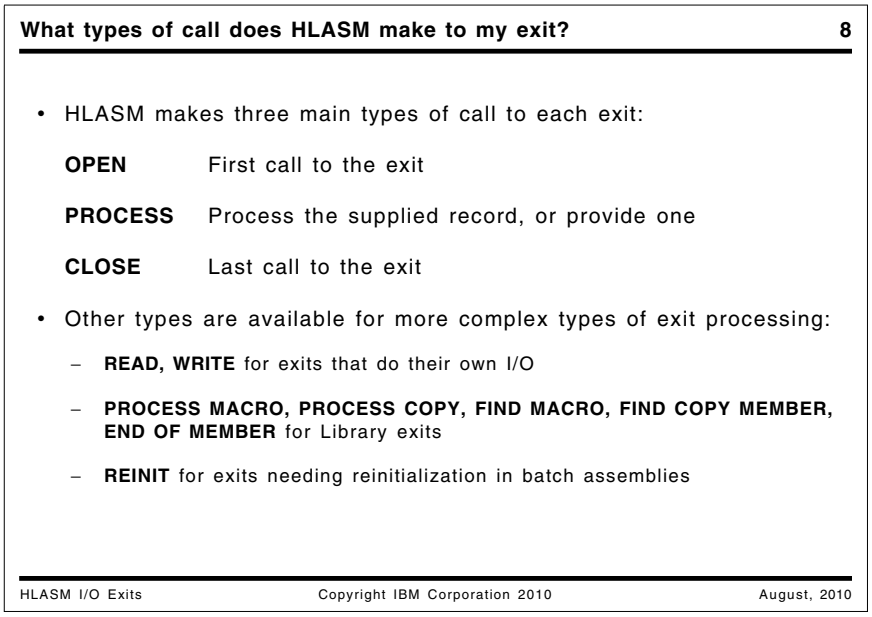
Word 12 If your exit provides a message to be inserted in the listing, this word contains the length of the message in the message buffer addressed by the third address in the primary address list. A nonzero length indicates the presence of a message.

Word 13 This word is the error severity to be associated with your message; it should be a multiple of 4 between 0 and 16.

Word 14 This word is initialized to zero by HLASM before the initial call to the exit, and is unmodified thereafter. It can be used for any purpose by the exit; a typical use might be to hold the address of additional working storage obtained by the exit.

Word 15 This word is is shared by all I/O exits, and provides a common anchor for data to be shared among exits. It is initialized to zero by the assembler before any exits are called, and is not later modified by HLASM.

The actions taken by the exit and by HLASM are determined by the values of the request type in Word 3 (when HLASM calls the exit) and the Return and Reason Code in Words 9 and 10 (when the exit returns to HLASM).



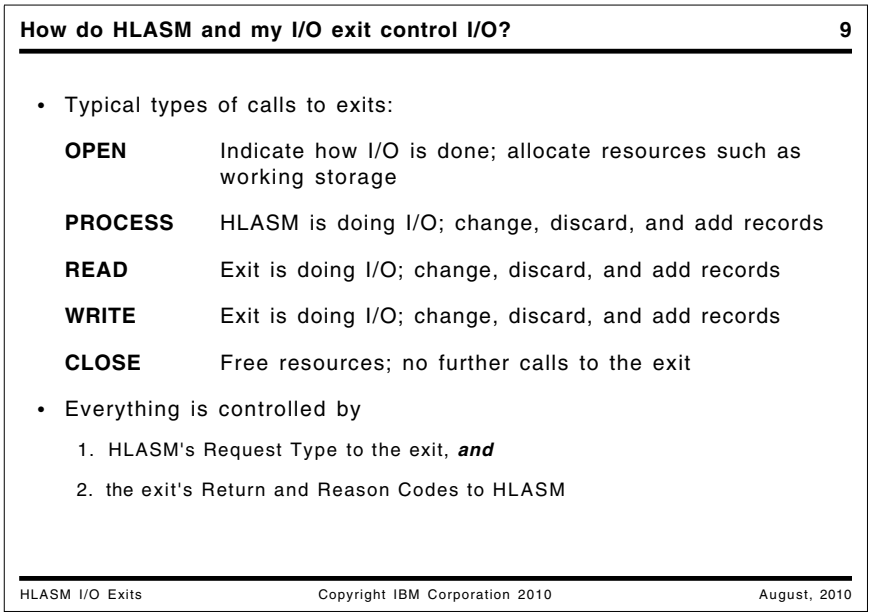
What types of call does HLASM make to my exit?

Each exit is called during assembler initialization with an OPEN request, to see if the exit wants to do I/O to its own data set; if not, HLASM opens the data set. That is, the exit indicates whether it or HLASM should to the I/O operations.

Any Return Code 16 causes HLASM to make no further calls to the exit.

This is the approximate processing sequence of I/O operations used by HLASM:

1. Pass 1:
   - SOURCE: (HLASM needs to scan for *PROCESS records).
   - LISTING: To produce the options summary.
   - TERM: In case of errors in the options scan.
   - LIBRARY: If macros need expanding, or COPY files must be read.
   - ADATA: Records are produced with the Job ID, and options details.
2. Pass 2:
   - PUNCH/OBJECT: The External Dictionary (ESD) records are generated first, followed by text (TXT), Relocation Dictionary (RLD), and END records.
   - LISTING/TERM: The rest of the listing is created, along with TERM messages for diagnostics.
   - ADATA: The remainder of the ADATA records are produced.

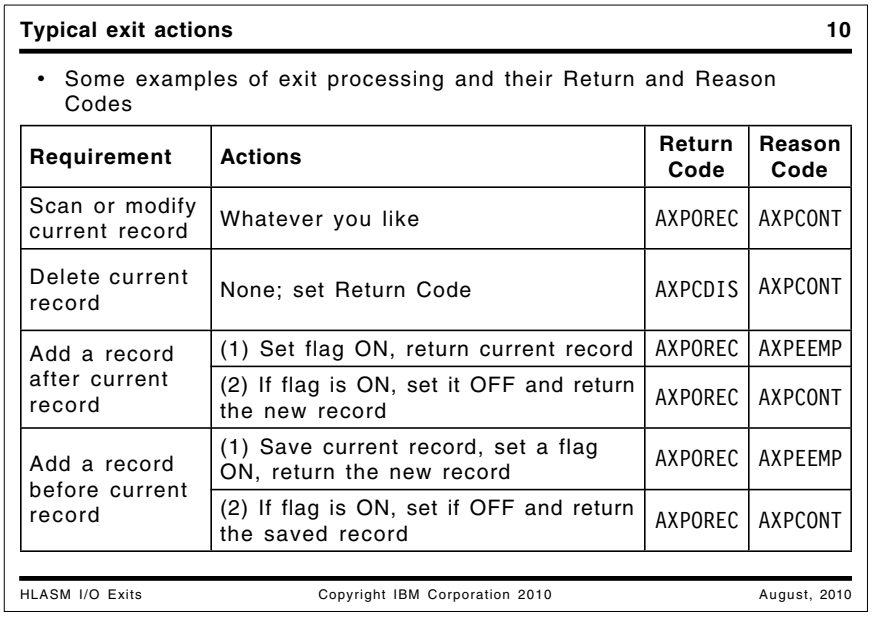


How do HLASM and my I/O exit control I/O?

The Return and Reason Codes set by your exit tell HLASM what to do with each record. Some examples of Return and Reason Codes are shown in these two tables, using the symbolic forms generated by the ASMAXITP macro.

Return Code	Value	Meaning
AXPAOPN	0	OPEN: The assembler opens the file
AXPCOPN	4	OPEN: The exit opens the file
AXPAOPL	8	OPEN: Both the assembler and the exit open the file
AXPOREC	0	PROCESS: Return the record in the buffer
AXPCDIS	4	PROCESS: Discard the record in the buffer
AXPCMNF	4	PROCESS: Member not found
AXPCDWN	16	ANY: The exit is no longer active, don't call again

Reason Code	Value	Meaning
AXPCONT	0	PROCESS: Continue normally
AXPEEMP	4	PROCESS: Return to the exit with an empty buffer



Typical exit actions 10

- Some examples of exit processing and their Return and Reason Codes

Requirement	Actions	Return Code	Reason Code
Scan or modify current record	Whatever you like	AXPOREC	AXPCONT
Delete current record	None; set Return Code	AXPCDIS	AXPCONT
Add a record after current record	(1) Set flag ON, return current record	AXPOREC	AXPEEMP
	(2) If flag is ON, set it OFF and return the new record	AXPOREC	AXPCONT
Add a record before current record	(1) Save current record, set a flag ON, return the new record	AXPOREC	AXPEEMP
	(2) If flag is ON, set if OFF and return the saved record	AXPOREC	AXPCONT

HLASM I/O Exits Copyright IBM Corporation 2010 August, 2010

Typical exit actions

The slide above uses the symbolic definitions generated by the ASMAXITP macro that maps the I/O exit communication areas.

Exits that let HLASM do the I/O operations are generally simplest. Some of the common operations are:

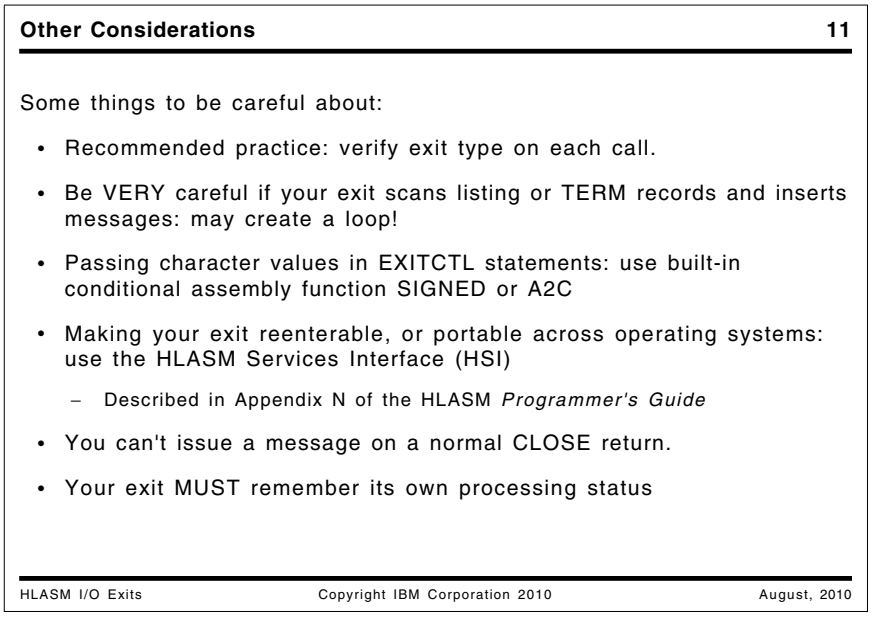
- Delete the current record:
  1. Set the Return Code to AXPCDIS to tell HLASM to discard the current record in the buffer.
  2. Set the Reason Code to AXPCONT to tell HLASM to continue
- Scan and/or modify the current record:
  1. Set the Return Code to AXPOREC to tell HLASM to use the current record in the buffer.
  2. Set the Reason Code to AXPCONT to tell HLASM to continue
- Add a record:
  - *After* the current record in the buffer:
    1. Set a flag ON in the exit's work area.
    2. Set the Return Code to AXPOREC to tell HLASM to use the current record in the buffer.
    3. Set the Reason Code to AXPEEMP to tell HLASM to return to the exit with an empty buffer
    4. On the next call, test the flag: if it's ON, put the added record into the buffer and its length in the Request List AXPBUFL field. Set the flag OFF.
    5. Set the Return Code to AXPOREC to tell HLASM to use the newly added record in the buffer.
    6. Set the Reason Code to AXPCONT to tell HLASM to continue.

- *Before* the current record in the buffer:

    1. Save the current record in the buffer in the exit's work area, and save its length. Then, set a flag in the work area ON.
    2. Move the newly added record to the buffer, and set its length in the AXPBUFL field.
    3. Set the Return Code to AXPOREC to tell HLASM to use the current record in the buffer.
    4. Set the Reason Code to AXPEEMP to tell HLASM to return to the exit with an empty buffer
    5. On the next call, test the flag: if ON, move the saved (previously current) record to the buffer, and store its length in the AXPBUFL field.
    6. Set the Return Code to AXPOREC to tell HLASM to use the newly added record in the buffer.
    7. Set the Reason Code to AXPCONT to tell HLASM to continue.

This may appear complicated, but it's actually simple in practice.

The important aspect of these operations is that the exit MUST keep track of its own state, so it knows which steps to take when HLASM makes the next PROCESS call. HLASM has no knowledge of what the exit plans to do; it follows the “instructions” given by the Return and Reason Codes.



Other Considerations

Some things to remember when writing your exits:

1. It's a good practice to always verify the type of call each time your exit is entered from HLASM
2. Be careful if you insert messages from a TERM exit, as you may get a re-entry to your TERM exit! Similarly, if your LISTING exit checks for diagnostic messages, be sure to ignore your own inserted messages that start with ASMA70xx.
3. Even though the EXITCTL instruction passes four fullword integer values to the designated exit routine, you may sometimes need to specify character or other forms for your exit. They must be converted first to numeric form. For example, if you want to pass the four characters WXYZ to your exit, you might try writing

   ```
   EXITCTL SOURCE,C'WXYZ',0,0,0   Doesn't work!
   ```

   but that won't work. Instead, use conditional assembly functions:

   ```
   &C    SETC  'C''WXYZ'''          Create a self-defining term
   &A    SETA  &C                   Assign its value to &A
   &C    SETC  SIGNED(&A)           Create a signed numeric value
         EXITCTL  SOURCE,&C,0,0,0   Use that as the operand
   ```

4. If your exit must work across multiple operating system environments such as z/OS, z/VM, z/VSE, and “zLinux”, you could write multi-path code with the attendant problems of ensuring that the correct macro and function libraries are available at assembly time.

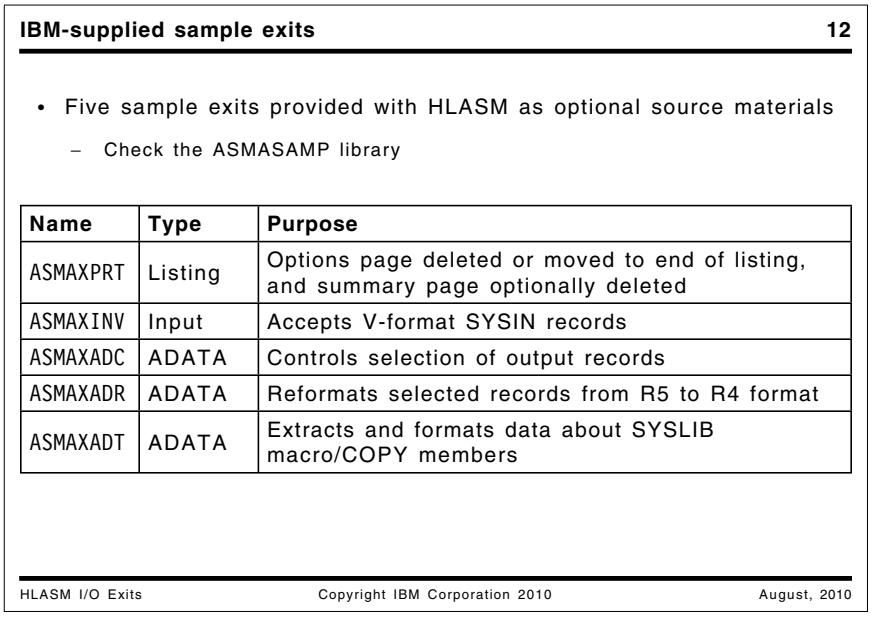
   The simpler approach is to use the HLASM Services Interface (HSI). It provides four services:

   a. Obtain storage
   b. Free storage

c. Write to “operator”
d. Get time and date

Because HLASM itself needs these capabilities, it already has the necessary system calls; they are provided to exits through the HSI. Details are provided in the HLASM *Programmer's Guide*, Appendix N.

5. No messages are recognized by HLASM on return from a CLOSE call, unless the Return Code is 16, indicating premature termination of the exit.
6. If your exit must do more than one task on each PROCESS call, it must retain status information to know what to do on each call from HLASM.



IBM-supplied sample exits

Five sample exits are provided with HLASM (except for the VSE Edition) in the ASMASAMP library.

- Print exit ASMAXPRT optionally moves the list of assembly options from the head to the end of the listing.
- Input exit ASMAXINV accepts variable-format (V-format) input records, and converts them to the fixed format required by the assembler.
- Three ADATA exits:
  - ASMAXADT extracts information from the SYSADATA file and produces fixed-format records for each macro or COPY file, indicating the library from which it was read.
  - ASMAXADC controls selection of output records, letting you choose the records to be written
  - ASMAXADR reformats selected records to the format produced by HLASM R4, in case you use tools that depend on the older format.

Each of these sample exits provides a useful function while also illustrating typical exit coding techniques.

A simple listing exit 13

- Each user's installation sets local standards for error “seriousness”
  - HLASM's diagnostic severity levels may be too low for some cases
  - Changing individual severity levels “globally” could adversely affect other users
- Solution: provide a tailorable listing exit that scans for chosen diagnostics
  - Issue a message with a user-specified increased severity level
- Use OPTIONS field of the Request List to check listing record type
  - Types 15 (AXPOPERR, options error) and 35 (AXPSOERR, source error)
  - If found, check the record for one of the chosen diagnostics
  - If found, insert the severity-increase message
- A listing of `MyPrtXit` is attached.

HLASM I/O Exits August, 2010

A simple listing exit ... 14

- A sample test program:

```
*Process SUPRWARN(435)
TRC      RSect ,
         Using *,15
         L     15,*+1  (Misaligned operand)
         ST    0,*     (Store into RSect)
         DS    F
         End
```

- Output generated by the exit, when invoked by `EX(PRX(MyPrtXit))`:

```
   Loc  Object Code    Addr1 Addr2  Stmt   Source Statement
                                       1 *Process SUPRWARN(435)
000000                 00000 0000C     2 TRC      RSect ,
                 R:F   00000           3          Using *,15
000000 58F0 F001             00001     4          L     15,*+1
** ASMA033I Storage alignment for *+1 unfavorable
** ASMA701W LISTING: ** Severity of previous message increased
000004 5000 F004             00004     5          ST    0,*
** ASMA036W Reentrant check failed
** ASMA702E LISTING: ** Severity of previous message increased
000008                                  6          DS    F
                                        7          End
```

- Severity increase is on the **exit's** message, not HLASM's!

HLASM I/O Exits August, 2010

Example of a simple listing exit

This sample LISTING exit scans listing records for the presence of diagnostic records starting with the characters `ASMA` at offset +4 from the start of the record (the offset skips the carriage control character and the three characters **). The full eight characters of the diagnostic are compared to entries in a table; if a match is found, the exit adds its own message with a new higher severity level.

This is a small test program for the exit, which scans for two diagnostics: `ASMA033I` (for unfavorable operand alignment) and `ASMA036W` (for an instruction that stores into an RSECT or fails a check specified by the RENT option).

```
*Process SUPRWARN(435)
TRC      RSect ,
         Using *,15
         L     15,*+1  (Misaligned operand)
         ST    0,*     (Store into RSect)
         DS    F
         End
```

When HLASM is invoked with the EX(PRX(MYPRTXIT)) option, the listing produced by this test program looks like this:

```
  Loc  Object Code    Addr1 Addr2  Stmt   Source Statement
                                      1 *Process SUPRWARN(435)
000000                00000 0000C     2 TRC      RSect ,
                 R:F  00000           3          Using *,15
000000 58F0 F001             00001     4          L     15,*+1
** ASMA033I Storage alignment for *+1 unfavorable
** ASMA701W LISTING: ** Severity of previous message increased
000004 5000 F004             00004     5          ST    0,*
** ASMA036W Reentrant check failed
** ASMA702E LISTING: ** Severity of previous message increased
000008                                  6          DS    F
                                        7          End
```

The inserted message has the increased severity; the severity of the diagnostic generated by HLASM is unchanged.

The exit routine keeps the “interesting” messages in a separate **MyPrtXtM** control section, so you can update just that section and relink it with the processing code, rather than reassembling the entire exit.

An example of a simple listing exit

```
  Loc  Object Code    Addr1 Addr2  Stmt   Source Statement
                                      1 *Process NoDXref,NoMXref,Using(NoMap)
                                      2 ********************************************************************
                                      3 * DISCLAIMER: The following program is provided for tutorial and     *
                                      4 * illustrative purposes. It is not supported by IBM, and no warranty *
                                      5 * of any kind is provided. IBM shall not be liable for any damages   *
                                      6 * from your use of this sample program.                             *
                                      7 ********************************************************************
                                      8 * This sample exit illustrates basic features of HLASM's I/O exits.  *
                                      9 * Check ASMAnnns messages, and issue a severity-increase message for *
                                     10 * message for selected diagnostics.                                  *
                                     11 ********************************************************************
                                     12 *        Usage:
                                     13 *        (1) Modify the entries in control section 'MyPrtXtM' to
                                     14 *            contain the 8-byte character string of each message
                                     15 *            whose severity is to be increased, followed by a halfword
                                     16 *            integer with the desired new severity to be assigned.
                                     17 *        (2) Create a loadable module from this source program.
                                     18 *        (3) Specify the HLASM option EXIT(PRX(MyPrtXit)) when
                                     19 *            assembling programs with messages to have its severity
                                     20 *            increased.
                                     21 *
                                     22 *        Method of operation:
                                     23 *
                                     24 *        (1) Open entry: validate parameters, terminate if errors
                                     25 *        (2) Process entry:
                                     26 *            (a) If not a type 15 or 35 (diagnostic) line, return
                                     27 *            (b) Else check for specified message numbers
                                     28 *                1. If not found, return
                                     29 *                2. If found, add severity-increase message
                                     30 *        (3) Close entry: return
                                     31 ********************************************************************
                                     32          Print NoGen
000000                00000 00184    33 MyPrtXit RSect
                                     34 MyPrtXit AMode Any
                                     35 MyPrtXit RMode Any
                                     36 *                              Register Equates
                      00000          37 R0       Equ   0
                      00001          38 R1       Equ   1
                      00002          39 R2       Equ   2
                      00009          40 R9       Equ   9                 Request Information List Pointer
                      0000A          41 R10      Equ   10                Buffer Pointer
                      0000B          42 R11      Equ   11                Error Buffer Pointer
                      0000C          43 R12      Equ   12                Program Base
                      0000F          44 R15      Equ   15
                                     45 *                              Displacements
                      00000          46 D0       Equ   0
                      00001          47 D1       Equ   1
                      00002          48 D2       Equ   2
                      00004          49 D4       Equ   4
                                     50 *                              Lengths
                      00001          51 L1       Equ   1
                      00008          52 L8       Equ   8
                      000FF          53 L255     Equ   255
                                     54 *                              Other Equates
                      00000          55 Null     Equ   0                 Null reason code
                                     58 ********************************************************************
                                     59 *        Entry point for all invocations from HLASM                *
                                     60 ********************************************************************
000000 47F0 F01C            0001C    61          Save  (14,12),,MyPrtXit-&SysDatC.-&SysTime. Save registers
000020 18CF                          68          LR    R12,R15           R12 will be program base
                 R:C  00000          69          Using MyPrtXit,R12
                                     70 *                              Validate entry type
```

```
                R:1  00000          71          Using AXPXITP,R1          R1 points to primary parm list
000022 989B 1000             00000  72          LM    R9,R11,AXPRIP       Addresses of first three items
                R:9  00000          73          Using AXPRIL,R9           R9  --> Request Information List
                R:A  00000          74          Using Buffer,R10          R10 --> Working Buffer
                R:B  00000          75          Using Err_Buff,R11        R11 --> Error Buffer
                                    76          Drop  R1
000026 5810 9004             00004  77          L     R1,AXPTYPE          Get exit type
00002A A71E 0003             00003  78          CHI   R1,AXPTLST          Check for listing exit
00002E A774 0048             000BE  79          JNE   Wrong_Exit_Type     Can't continue, wrong exit type
                                    80 *                            Have been called as LISTING exit
000032 5810 9008             00008  81          L     R1,AXPRTYP          Get request type
000036 A71E 0001             00001  82          CHI   R1,AXPROPN          Check for OPEN request
00003A A784 0039             000AC  83          JE    Open_Request        Branch to process OPEN entry
00003E A71E 0005             00005  84          CHI   R1,AXPRPRO          Check for PROCESS request
000042 A784 0008             00052  85          JE    Process_Request     Branch to process PROCESS entry
000046 A71E 0002             00002  86          CHI   R1,AXPRCLS          Check for CLOSE request
00004A A784 0031             000AC  87          JE    Close_Request       Branch to process CLOSE entry
00004E A7F4 003C             000C6  88          J     Bad_Type_Request    Branch if unknown request
                                    89 ************************************************************************
                                    90 *        PROCESS Request                                               *
                                    91 ************************************************************************
000052                              92 Process_Request  DC  OH
000052 5810 900C             0000C  93          L     R1,AXPOPTS          Check options value
000056 A71E 000F             0000F  94          CHI   R1,AXPOPERR         Is it an options error message?
00005A A784 0006             00066  95          JE    LookItUp            Yes, see if we're interested in it
00005E A71E 0023             00023  96          CHI   R1,AXPSOERR         Is it an error-message line?
000062 A774 0025             000AC  97          JNE   Return              If not, just return to HLASM
000066 5810 C178             00178  98 LookItUp L     R1,=A(MyPrtXtM)     Get address of messages section
                R:1  00188          99          Using MyPrtXtM,R1         Provide addressability
00006A 4800 1000             00188 100          LH    R0,NErrs            Number of items
00006E 4110 1002             0018A 101          LA    R1,ErrList          Prepare to scan interesting items
                                   102          Drop  R1
000072                             103 ChkErr   DC    0H
000072 D507 1000 A004 00000  00004 104          CLC   D0(L8,R1),Line+D4   Compare this item
000078 A784 0008             00088 105          JE    BumpIt              If matched, create error message
00007C A71A 000A             0000A 106          AHI   R1,LErrItem         Step to next item
000080 A706 FFF9             00072 107          JCT   R0,ChkErr           And continue matching
000084 A7F4 0014             000AC 108          J     Return              No match, return to HLASM
000088                             109 BumpIt   DC    0H
000088 4800 1008             00008 110          LH    R0,L8(0,R1)         Get new severity
00008C 5000 9030             00030 111          ST    R0,AXPSEVC          Set severity code for HLASM
000090 9240 B000       00000       112          MVI   Err_Msg,C' '        Blank the error message buffer
000094 D2FD B001 B000 00001 00000  113          MVC   Err_Msg+D1(Err_Len-L1),Err_Msg  Propagate blanks
00009A A718 0029             00029 114          LHI   R1,L'Bump_Msg       Message length
00009E 5010 902C             0002C 115          ST    R1,AXPERRL          Store for HLASM
0000A2 D228 B000 C0FC 00000 000FC  116          MVC   Err_Msg(L'Bump_Msg),Bump_Msg   Move error message
0000A8 A7F4 0002             000AC 117          J     Return              Return to HLASM
                                   118 ************************************************************************
                                   119 *        PROCESS Return                                                *
                                   120 ************************************************************************
0000AC                             121 Return   DC    0H
                                   122 ************************************************************************
                                   123 *        OPEN and CLOSE Requests                                       *
                                   124 ************************************************************************
0000AC                             125 Open_Request   DC  0H
0000AC                             126 Close_Request  DC  0H
0000AC D203 9020 C17C 00020 0017C  127          MVC   AXPRETC,=A(AXPCONT) Set return code for "normal" action
0000B2 D203 9024 C180 00024 00180  128          MVC   AXPREAC,=A(Null)    Set reason code
0000B8 98EC D00C             0000C 129          RETURN  (14,12)           Return to HLASM
0000BE                             134 Wrong_Exit_Type   DC  0H
0000BE 4110 C125             00125 135          LA    R1,Bad_Exit         Point to error message info
0000C2 A7F4 0006             000CE 136          J     Proc_Error          Branch to process error message
0000C6                             138 Bad_Type_Request  DC  0H
0000C6 4110 C149             00149 139          LA    R1,Bad_Type         Point to error message info
0000CA A7F4 0002             000CE 140          J     Proc_Error          Branch to process error message
```

```
                                    143 *        Handle error conditions and information messages

0000CE                              145 Proc_Error DC  OH                 Process error messages
0000CE 1722                         146          XR    R2,R2              Clear register for message length
0000D0 4320 1001            00001   147          IC    R2,D1(,R1)         Get message severity
0000D4 5020 9030            00030   148          ST    R2,AXPSEVC         Store severity code
0000D8 4320 1000            00000   149          IC    R2,D0(,R1)         Get message length
0000DC A72E 00FF            000FF   150          CHI   R2,Err_Len         Compare to buffer size
0000E0 A7D4 0004            000E8   151          JNH   Proc_Text_2        Branch if it will fit
0000E4 A728 00FF            000FF   152          LHI   R2,Err_Len         Truncate overly wordy messages
0000E8                              153 Proc_Text_2  DC  OH
0000E8 5020 902C            0002C   154          ST    R2,AXPERRL         Store true error message length
0000EC 0620                         155          BCTR  R2,0               Decrement length for executed move
0000EE 4420 C0F6            000F6   156          EX    R2,Move_Msg        Move message to buffer
0000F2 A7F4 FFDD            000AC   157          J     Return             Return to assembler
0000F6 D200 B000 1002 00000 00002   158 Move_Msg MVC   Err_Msg(*-*),D2(R1) Move message to error buffer
                                    159          Drop  R9                 Pointer to req list
                                    160          Drop  R12                Program base
                                    161 ********************************************************************
                                    162 *        Error Messages                                            *
                                    163 ********************************************************************
0000FC 5C5C40E285A58599             164 Bump_Msg       DC  C'** Severity of previous message increased'
000125 2214                         165 Bad_Exit       DC  AL1(L'Wrong_Exit_Msg,AXPCBAD) Length and severity
000127 C5A789A340838193             166 Wrong_Exit_Msg DC  C'Exit called for other than LISTING'
000149 2A14                         167 Bad_Type       DC  AL1(L'Bad_Type_Msg,AXPCBAD)   Length and severity
00014B C995A58193898440             168 Bad_Type_Msg   DC  C'Invalid action or operation type requested'
000178                              169          LTORG
000178 00000188                     170                =A(MyPrtXtM)
00017C 00000000                     171                =A(AXPCONT)
000180 00000000                     172                =A(Null)
000188                 00188 00016  173 MyPrtXtM RSect ,                  Separate section with message #s
                                    174 ********************************************************************
                                    175 *        Message prefixes whose severity is to be increased
                                    176 ********************************************************************
000188 0002                         177 NErrs    DC    Y(NMsgs)
00018A                              178 ErrList  DC    OH
00018A C1E2D4C1F0F3F6E6             179          DC    CL(L8)'ASMA036W',H'8'   Message and new severity
                       0000A        180 LErrItem Equ   *-ErrList               Length of one entry
000194 C1E2D4C1F0F3F3C9             181          DC    CL(L8)'ASMA033I',H'4'   Message and new severity
                                    182 ******** DC    CL(L8)'next msg',H'?'   Template for more messages
                       00002        183 NMsgs    Equ   (*-ErrList)/LErrItem    Number of messages to check
                                    184 ********************************************************************
                                    185 *        Dummy Sections                                            *
                                    186 ********************************************************************
000000                 00000 000FF  187 Buffer   DSect ,                  Mapping of input record buffer
000000                              188 Line     DS    CL(L255)' '         HLASM listing message record
                       000FF        189 Buff_Len Equ   *-Buffer            Buffer length
000000                 00000 000FF  190 Err_Buff DSect ,                  Mapping of error message buffer
000000                              191 Err_Msg  DS    CL(L255)' '         Allocated space for messages
                       000FF        192 Err_Len  Equ   *-Err_Buff          Buffer length
                                    193 *        Mapping of Assembler I/O Exit Work Areas
                                    194          ASMAXITP  PRINT=NOGEN
                                    196+         PRINT NOGEN
                                    487          End
```