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

- 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

- A program loaded by HLASM to let you monitor and control I/O
- You specify which data sets or files will be monitored
  - SYSIN, SYSLIB, SYSPRINT, SYSPUNCH, SYSLIN, SYSTERM, SYSADATA
  - All except HLASM's SYSUT1 work file
- Exits have as little or as much control as desired
  - Monitor or assist assembler I/O, or replace it entirely
  - Add new records
  - Scan and possibly modify existing records
  - Delete records
  - Replace records
- Exits can issue messages to be inserted in the listing

- Write it in (almost any) language
  - Typically in Assembler; some have been written in C, PL/X
- Obey standard linkage conventions:

  R1 Address of a list of argument addresses

  R13 Address of an 18-word save area

  R14 Address in calling program where control is returned

  R15 Entry-point address of the called program
- Generate a relocatable, loadable module
- Detailed guidelines in HLASM Programmer's Guide

- When you invoke HLASM, specify an EXIT option:

  PARM='...,EXIT(<PARM-exit-type>(<your-exit-name>)),...'

DDName	PARM-exit-type	Abbreviation
SYSIN	INEXIT	INX
SYSLIB	LIBEXIT	LBX
SYSPRINT	PRTEXIT	PRX
SYSPUNCH, SYSLIN	OBJEXIT	OBX
SYSADATA	ADEXIT	ADX
SYSTERM	TRMEXIT	TRX

- For example, to invoke your 'MyPrtXit' LISTING exit:

  PARM='...,EXIT(PRX(MyPrtXit)),...'

- 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)))

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

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

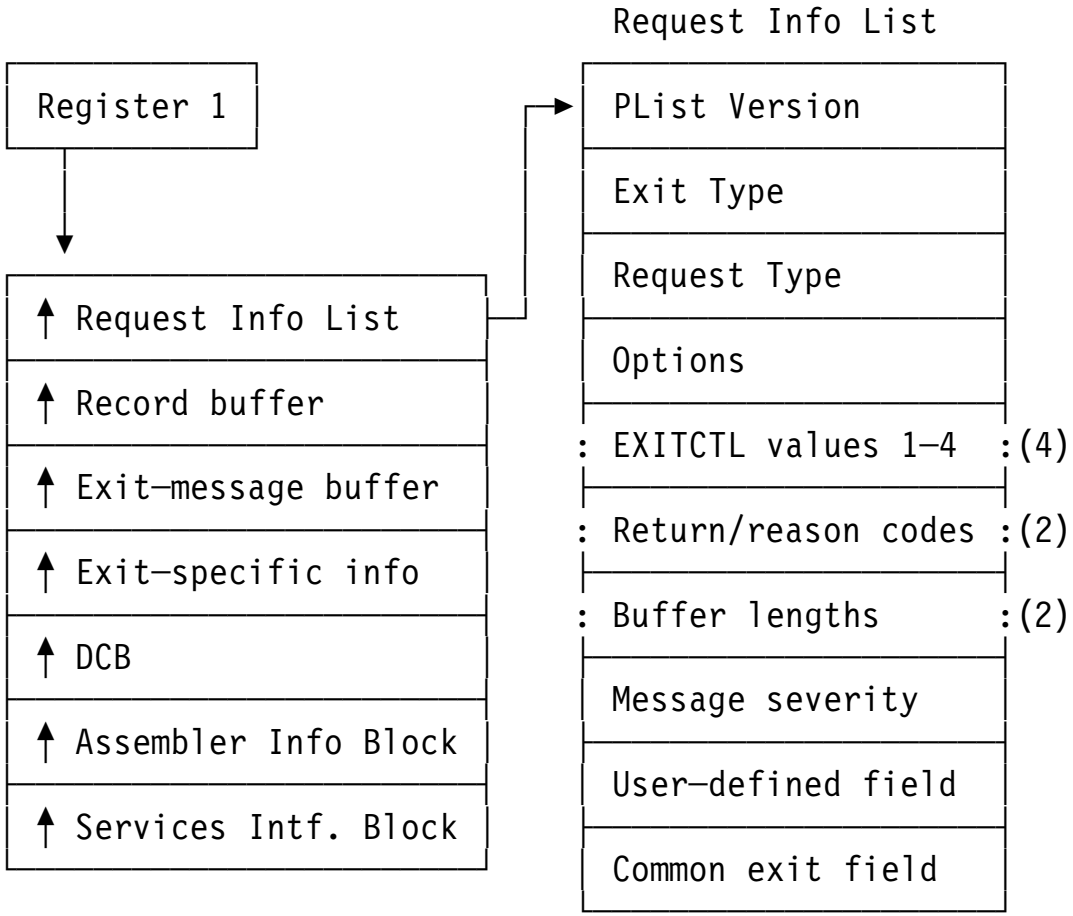
- From 1 to 4 32-bit binary **control-values** can be specified; 4 values are always passed to the exit

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)

- All assembler/exit communication via I/O Exit Parameter List
  - Control information
  - Data set information
  - Pointers to buffers, message area
  - Exit anchor words
- Standard linkage conventions



- HLASM makes three main types of call to each exit:

  OPEN First call to the exit

  PROCESS Process the supplied record, or provide one

  CLOSE Last call to the exit

- Other types are available for more complex types of exit processing:
  - **READ, WRITE** for exits that do their own I/O
  - **PROCESS MACRO, PROCESS COPY, FIND MACRO, FIND COPY MEMBER, END OF MEMBER** for Library exits
  - **REINIT** for exits needing reinitialization in batch assemblies

- Typical types of calls to exits:

OPEN	Indicate how I/O is done; allocate resources such as working storage
PROCESS	HLASM is doing I/O; change, discard, and add records
READ	Exit is doing I/O; change, discard, and add records
WRITE	Exit is doing I/O; change, discard, and add records
CLOSE	Free resources; no further calls to the exit

- Everything is controlled by

  1. HLASM's Request Type to the exit, ***and***
  2. the exit's Return and Reason Codes to HLASM

- 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

Some things to be careful about:

- Recommended practice: verify exit type on each call.
- Be VERY careful if your exit scans listing or TERM records and inserts messages: may create a loop!
- Passing character values in EXITCTL statements: use built-in conditional assembly function SIGNED or A2C
- Making your exit reenterable, or portable across operating systems: use the HLASM Services Interface (HSI)
  - Described in Appendix N of the HLASM *Programmer's Guide*
- You can't issue a message on a normal CLOSE return.
- Your exit MUST remember its own processing status

- Five sample exits provided with HLASM as optional source materials
  - Check the ASMASAMP library

Name	Type	Purpose
ASMAXPRT	Listing	Options page deleted or moved to end of listing, and summary page optionally deleted
ASMAXINV	Input	Accepts V-format SYSIN records
ASMAXADC	ADATA	Controls selection of output records
ASMAXADR	ADATA	Reformats selected records from R5 to R4 format
ASMAXADT	ADATA	Extracts and formats data about SYSLIB macro/COPY members

- 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.

- 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!

