High Level Assembler Exits

Practical examples of invoking High Level Assembler exits and why they are useful

by ColeSoft, a leader in all things assembler
Introduction

Today we will talk about modifying the behavior of assembler SYSIN, SYSLIB, and SYSPRINT handling.

This is accomplished by writing an assembler "exit" load module that is called by the assembler during its processing of all of its external files.

We will see specific example for modifying SYSIN, SYSLIB, and SYSPRINT.
Introduction

Here are a few examples of why we might want to do this.

- Implement "C" style comments and #ifdef
- Implement a bi-lingual C / Assembler input stream and macros
- Implement labels such as MY_LABEL:
- Remove ASA carriage controls and replace them with blank lines to enable prettier viewing of listings.
What we will cover

1. What exits do?
2. What are they useful for?
3. What exits are there?
4. How to build and invoke the exits
5. Two concrete examples
6. Where can I learn more about this?
1) What exits do

• Provide a way to modify the flow of source code, object data, symbolic data, and listings during the assembly process.

• Provide a way to alter the meaning of the data streams. For example, you could define new source code rules or alter the appearance of the listings.

• Exits are called by the assembler during processing using standard linkages and formal, documented APIs. We will focus on z/OS today.
2) What are they useful for?

- Exits can supply additional input
  - Inject new records
  - Can add new data sources
- Exits can modify inputs and outputs
  - Can alter the content of records
  - Can suppress records
- Exits can extract information from the assembly and save it elsewhere
  - Enforce coding standards
3) What exits are there?

- There are seven exits, in four categories
  - Source and Library
  - Listing and Term
  - Punch and Object
  - ADATA

We will talk about each of these…
Source Exit

- Provides new SYSIN records
- Changes SYSIN records
- Deletes SYSIN records
- Potential uses:
  - Enforcing coding standards.
  - Reading source more than 72 bytes wide.
  - Implement your own definition of a “blank” line.
  - Implement "C" /* */ comments and #ifdef
  - Extend the language, such as providing a new syntax like LABEL:
Library Exit

- Provides new SYSLIB macro/copy records
- Changes SYSLIB macro/copy records
- Deletes SYSLIB macro/copy records
- Potential uses:
  - Same benefits as for Source exit
  - Substituting different macro libraries
Listing & Term Exits

• Provide new SYSPRINT and SYSTEM records
• Changes SYSPRINT and SYSTEM records
• Deletes SYSPRINT and SYSTEM records
• Potential uses:
  – “Remove” ASA carriage control and supply blanks lines instead to make listings more readable on displays and text editors
  – Convert listings directly to HTML
  – Annotate listings
Punch & Object Exits

- Provides new SYSLIN and SYSPUNCH records
- Changes SYSLIN and SYSPUNCH records
- Deletes SYSLIN and SYSPUNCH records
- Potential uses:
  - Extract information from object records
  - Could write different data to SYSLIN and SYSPUNCH
  - Alter object records
ADATA Exits

- Provides new SYSADATA records
- Changes SYSADATA records
- Deletes SYSADATA records
- Potential uses:
  - Extract information from object records
  - Thin out ADATA information
  - Collect data for debuggers
4) How to build and invoke an exit

• You can have one load module for each exit or one for all of them.

• Assemble the exit and then Link-Edit it into a load module with AMODE31 and REUS.
  – //LKED.SYSLMOD DD DSN=MY.LOADLIB(MYEXIT),…

• Add the loadlib to the assembler’s JOBLIB or STEPLIB, or place the module in the LNKLIST.
  – //ASM.STEPLIB DD
    // DD DSN=MY.LOADLIB,DISP=SHR

• The exit’s load module name is specified by the assembler’s JCL PARM=‘xxxEXIT(MYEXIT)’
5) Two concrete examples

- Sample Source and Library exit to implement a new syntax
  - LABEL:
  - Shows how to alter records
- Sample Listing exit to “convert” ASA carriage control to blank lines
  - Makes listings easier to read on many text editors
  - Shows how to alter and inject records

The point of these examples is to illustrate how we can alter and inject records. These could be extended to implement many other strategies such as a bi-lingual C/ASM SYSIN stream.
5.1 Implementing a LABEL: syntax

The following slides will illustrate an implementation of our sample exit that converts

```
THIS_IS_A_LABEL:

  to

THIS_IS_A_LABEL      DS  0H
```
Implementing a LABEL: syntax

Many programmers are used to coding

\texttt{THIS\_IS\_A\_LABEL DS 0H}

or

\texttt{THIS\_IS\_A\_LABEL EQU *}

But these are (I think) ugly.
Implementing a LABEL: syntax

One solution is to have a macro, named LABEL, and code this:

```plaintext
MACRO
&NAME    LABEL &DUMMY,&ALIGN=H
AIF    ('&NAME' EQ '').NL
PUSH    PRINT,NOPRINT
AIF    (&SYSNEST GT 1).PR DO NOT SUPPRESS FOR INNERS
PRINT    NOGEN,NOPRINT
.PR    ANOP
&NAME    DS 0&ALIGN
.POP    PRINT,NOPRINT
.NL    ANOP
MEND

THIS_IS_A_LABEL    LABEL
```

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
rschreiber@colesoft.com  http://www.colesoft.com  Booth 415

8/12/2015
Implementing a LABEL: syntax

Or, perhaps, better yet...

**THIS_IS_A_LABEL:**

- To do this requires the use of the Source and probably Library exit too.
  - Source exit handles : from SYSIN
  - Library exit handles : from SYSLIB, macros and copy code
Implementing a LABEL: syntax

• Each exit is called for three functions
  – OPEN
    • Provides the ability to perform post-open processing, like opening a different dataset
    • We just return R15=0 to indicate to use normal SYSIN/SYSLIB datasets.
  – CLOSE
    • Provides an opportunity to clean up.
    • We just return R15=0 to indicate operation (none) succeeded.
Implementing a LABEL: syntax

- Uses Source and Library exits
  - Processes SYSIN, MACRO, COPY statements
    - Scan for xxxxxxxxxxxxxx: starting in column 1
    - Replace : with DS 0H or EQU * or whatever you want
  - Return with R15=0 to hand modified card image to assembler
Implementing a LABEL: syntax

TITLE 'PROCESS SOURCE AND LIBRARY EXITS'

CHECKTYP DS 0H

CLC AXPRTYP,=A(AXPROP) OPEN REQUEST?
BE EXIT00 YES, GO HANDLE IT

CLC AXPRTYP,=A(AXPRCLS) CLOSE REQUEST?
BE EXIT00 YES, GO HANDLE IT

CLC AXPRTYP,=A(AXPRREAD) READ REQUEST?
BE EXIT16 YES, DON'T CALL BACK

CLC AXPRTYP,=A(AXPRPRO) PROCESS SYSIN REQUEST?
BE PROCESS YES, GO HANDLE

CLC AXPRTYP,=A(AXPRPMAC) PROCESS MACRO REQUEST?
BE PROCESS YES, GO HANDLE

CLC AXPRTYP,=A(AXPRPCPY) PROCESS COPY REQUEST?
BE PROCESS YES, GO HANDLE

B EXIT16 UNKNOWN, DON'T CALL BACK

PROCESS DS 0H

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
rschreiber@colesoft.com http://www.colesoft.com Booth 415 8/12/2015
Implementing a LABEL: syntax

```
TM     STATUS,CONTSTMT     WAS THE PRIOR CONTINUED?
BO     FLUSH              YES, DO NOT PROCESS THIS
CLI    CONTINUE,C' '      IS THIS A CONTINUED LINE?
BNE    MARKIT             YES, DON'T MESS WITH CONTINUED

CLI    COMMENT,C'*'      IS THIS A COMMENT?
BE     EXIT00             DON'T MESS WITH COMMENTS
CLC    COMMENT(2),=C'.*'  DON'T MESS WITH COMMENTS
BE     EXIT00

*   CONVERT LINES WITH LABEL: TO LABEL DS 0H
*
*
TEST1   DS    0H
CLI    SRCLINE,C' '      START WITH LABEL?
BE     EXIT00             NO, LEAVE IT ALONE
```
Implementing a LABEL: syntax

LA   R2,SRCLINE
LHI  R4,1             INCREMENT
LA   R5, CONTINUE-1  LIMIT
TESTLOOP BXH  R2, R4, EXIT00  LOOK FOR A BLANK
CLI  0(R2), C' '     NORMAL LABEL?
BE   EXIT00          YES, LEAVE IT ALONE
CLI  0(R2), C':'     END OF A SPECIAL LABEL?
BE   HAVELBL         YES

*        WE HAVE FOUND A CARD LIKE "THIS_IS_A_LABEL: "
*
HAVELBL DS    0H
MVI   0(R2), C' '        REMOVE THE COLON
*
*        SEE IF WE CAN PUT IN 'DS 0H' AT THE PRETTIEST POINT
*
CLC   SRCLINE+8(10), =C' '  '
BNE   FLOATING
MVC   SRCLINE+8(10), =C' DS 0H ' YES, GREAT!
B    EXIT00
Implementing a LABEL: syntax

* WE NEED TO STUFF IN DS 0H -- THIS MIGHT OVERLAY
* THE COMMENT IF ITS TOO CLOSE TO THE COLON
*
FLOATING DS 0H
MVC 0(7,R2),=C' DS 0H '
B EXIT00

MARKIT OI STATUS,CONTSTMT THIS IS CONTINUED
B EXIT00 DO NOT ALTER IT

FLUSH CLI CONTINUE,C' ' STILL FURTHER?
BNE EXIT00 DO NOT ALTER IT
NI STATUS,255-CONTSTMT NO LONGER CONTINUED
B EXIT00 DO NOT ALTER IT
5.2 Implementing FBA carriage control conversion

• Provides a way to directly convert carriage controls for SYSPRINT to blank lines
  – Avoids needing a program to post-process your SYSPRINT.
• Useful when you specify //SYSPRINT DD PATH= to place listing in an HFS directory and they you can use OMVS cat to display it.
ASA carriage control

• Functions of the exit
  – OPEN
    • Just return R15=0 to indicate to use normal SYSPRINT dataset.
  – CLOSE
    • Just return R15=0 to indicate operation (none) succeeded.
ASA carriage control

• Functions of the exit
  – PROCESS SYSPRINT
    • ‘1’ – form feed.
      – Ignored, you might want to insert Ctrl-L Form Feed?
    • ‘ ’ – single space.
      – Ignored, its already how we want it.
    • ‘+’ – overprint (not used by the assembler.)
      – Ignored.
    • ‘0’ – double space.
      – we insert a blank line.
    • ‘-’ – triple space
      – we insert two blank lines.
ASA carriage control

TITLE 'PROCESS LISTING EXIT'
CHECKLST DS 0H

CLC AXPRTYP,=A(AXPROP N) OPEN REQUEST?
BE EXIT00 YES, GO HANDLE IT

CLC AXPRTYP,=A(AXPRCLS) CLOSE REQUEST?
BE EXIT00 YES, GO HANDLE IT

CLC AXPRTYP,=A(AXPRPRO) PROCESS REQUEST?
BE LSTPROC YES, GO HANDLE

B EXIT16 UNKNOWN, DON'T CALL AGAIN

LSTPROC DS 0H
ASA carriage control

* CHECK THE STACK TO SEE IF OLD LINES NEED SENDING?
   CLI   LINE1,0          ANY LEFT TO SEND?
   BE    NOMORE

L   R9,AXPBUFL          SET LENGTH
LA   R2,line1          SOURCE DATA
LHI  R3,133            LENGTH=133
ICM  R3,B'1000',=C' '  PAD BYTE

MVCL R8,R2             COPY RECORD INTO BUFFER

* POP THE STACK

MVC   LINE1(133),LINE2 "POP" THE LINE STACK
MVC   LINE2(133),LINE3 ""
MVC   LINE3(133),LINE4 ""
MVI    LINE4,0          ""

CLI   LINE1,0          DID WE JUST SEND THE LAST ONE?
BE    EXIT00          YES, RETURN LAST RECORD
B     EXIT04          NO, RETURN NON-LAST RECORD
ASA carriage control

NOMORE DS 0H
MVI LINE1,0
MVI LINE2,0
MVI LINE3,0
MVI LINE4,0

RESET ALL THE LINES
""
"
"

CLI SRCLINE,C'0'
BE ASA1
CARRIAGE SKIP?
YES, ONE BLANK LINE

CLI SRCLINE,C'-' CARRIAGE SKIP?
BE ASA2
YES, TWO BLANK LINES

B EXIT00
RETURN WITH LAST RECORD

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
rschreiber@colesoft.com  http://www.colesoft.com  Booth 415 8/12/2015
ASA carriage control

ASA1
DS 0H
MVI LINE1,C' '             SKIP 1 LINE
MVC LINE1+1(132),=132C' '  BLANKS
MVI LINE2,C' '             SINGLE SPACE
MVC LINE2+1(132),1(R8)     SAVE DATA
B LSTPROC

ASA2
DS 0H
MVI LINE1,C' '             SKIP 1 LINE
MVC LINE1+1(132),=132C' '  BLANKS
MVI LINE2,C' '             SKIP 1 LINE
MVC LINE2+1(132),=132C' '  BLANKS
MVI LINE3,C' '             SINGLE SPACE
MVC LINE3+1(132),1(R8)     SAVE DATA
B LSTPROC
AXPXITP upon entry – via R1

TCB#5 RB#1 ---------------------------------------------------------------XDC-CDF ISPF INTERFACE ------
XDC ===> FORMAT R1?
_  00000000_0001AA94 8f (A.S.ROBTEST) --- AXPXITP+0, @R1+0, @R5+0, PRIVATE+18A94
_    +0     +AXPXITP DSECT
_  +0 0001AAB0  +AXPRIP DS A  Pointer to Request Information (see be
_    +0o@R5
_  +0o@R1  0001AAB0  1AAB0  *....*
_    +4 00041C8C  +AXPBUPF DS A
_    +8 0001AAEC  +AXPERRP DS A
_    +8  +AXPERRBUFL EQU 255 Length of Error Buffer
_  +C 0001ABEC  +AXPSIP DS A
_    +10 00019D48  +AXPDCBP DS A
_    +14 00019E98  +AXPAIP DS A
_    +18 80019EBC  +AXPHSIP DS A
_    +18 **
_  +18 **
_  +18  +AXPBASEL EQU *-AXPRIP Length of base

Pointer to Buffer
Pointer to Error Buffer
Pointer to Exit Information(see below)
Pointer to DCB (MVS/CMS only)
-> Assembler Info Block (see below)
Pointer to Services Interface
block (see below) - bit 0 set on
to indicate end-of-list

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
rschreiber@colesoft.com http://www.colesoft.com Booth 415 8/12/2015
AXPRIL control block for SYSPRINT call

TCB#5 RB#1  

XDC  FORMAT : AXPRIL

00000000_0001AAB0 8f (A.S. ROBTEST)  AXPRIL L+0, @R10+0, @R3+0, @R1+1C, @R11+1C, @R5+1C,

AXPRIL TP+1C, PRI VATE+18A80

+0 00000003 +AXPRIL DSECT Request Information List
+4 +AXPLVER DS F EXIT list version number

+4 +AXPVER3 EQU 3 Exit Parameter List Version 3

+4 +AXPTSRC EQU 1 SOURCE (SYSI N)

+4 +AXPTLIB EQU 2 LIBRARY (SYSLIB)

+4 +AXPTLST EQU 3 LISTING (SYSPRINT)

+4 +AXPTPUN EQU 4 PUNCH (SYSPUNCH)

+4 +AXPTOBJ EQU 5 OBJECT (SYSLIN)

+4 +AXPTAD EQU 6 ADATA (SYSDATA)

+4 +AXPTTRM EQU 7 TERM (SYSTEM)

+8 00000005 +AXPRTP Y DS F Request Type (see values below)

+8 +AXPRPRRO EQU 5 PROCESS - exit receives control to inspect

+8 +AXPRPRO EQU 5 and/or modify record provided by

+8 +AXPRPRO EQU 5 the assembler (Not LIBRARY exit)

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
rschreiber@colesoft.com  http://www.colesoft.com  Booth 415  8/12/2015
Sample test program

// EXEC PGM=ASM90, PARM=('INEXIT(MYEXIT),LIBEXIT(MYEXIT)',
// 'PRTEXIT(MYEXIT)')
//SYSIN DD *
TEST CSECT
  USING *,15
  SPACE 1
  DS 20X
  SPACE 2
  LA 4,THIS_IS_A_LABEL
  SPACE 3
THIS_IS_A_LABEL:
  DC CL20'SOME TEXT'
END TEST
Sample listing without the exit

1

Active Usings: None

0

Loc  Object Code  Addr1     Addr2     Stmt  Source Statement  HLA
000000000  000000000  000002C     1  TEST  CSECT

R:F  00000000

000000000

4  DS  20X

-0000014  0000  0000  00000000  6  LA  4,THIS_IS_A_LABEL

** ASMA044E Undefined symbol - THIS_IS_A_LABEL
** ASMA435I Record 6 in ROB.ROBEXITT.JOB08902.D0000101.? on volume:

-8 THIS_IS_A_LABEL:

** ASMA142E Operation code not complete on first record
** ASMA435I Record 8 in ROB.ROBEXITT.JOB08902.D0000101.? on volume:

00000018  E2D6D4C540E3C5E7          9  DC  CL20'SOME TEXT'
00000000

10  END  TEST

1

Ordinary Symbol and Literal Cross Reference

-Symbol  Length  Value    Id  R  Type  Asm  Program  Defn  References

0TEST     1  00000000  00000001  J  1  10

THIS_IS_A_LABEL

***UNDEFINED***  00000000  U  6

0Statements Flagged

0  6(P1,6), 8(P1,8)
### Sample listing with the exit

<table>
<thead>
<tr>
<th>Loc HLA</th>
<th>Object Code</th>
<th>Addr1</th>
<th>Addr2</th>
<th>Stmt</th>
<th>Source Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>00000000</td>
<td>0000002C</td>
<td>1</td>
<td>TEST</td>
<td>CSECT</td>
</tr>
<tr>
<td></td>
<td>R:F 00000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>USING *,15</td>
</tr>
<tr>
<td>00000014</td>
<td>4140 F018</td>
<td></td>
<td>0000018</td>
<td>6</td>
<td>LA 4,TIS_IS_A_LABEL</td>
</tr>
<tr>
<td>0000018</td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>0000018 E2D6D4C540E3C5E7</td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>DC CL20'SOME TEXT'</td>
</tr>
<tr>
<td>00000000</td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>END TEST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Length</th>
<th>Value</th>
<th>Id</th>
<th>R Type</th>
<th>Asm</th>
<th>Program</th>
<th>Defn</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLA</td>
<td>HLA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST</td>
<td>1</td>
<td>00000000</td>
<td>00000001</td>
<td>J</td>
<td></td>
<td></td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>THIS_IS_A_LABEL</td>
<td>2</td>
<td>0000018</td>
<td>00000001</td>
<td>H H</td>
<td></td>
<td></td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>

---

**Ordinary Symbol and Literal Cross Reference**

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

rshreiber@colesoft.com  http://www.colesoft.com  Booth 415  8/12/2015
Auditing information

The assembler produces the following report to assist auditing the use of exits:

<table>
<thead>
<tr>
<th>Exit Type</th>
<th>Name</th>
<th>Calls</th>
<th>---Records---</th>
<th>Diagnostic Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Added</td>
<td>Deleted</td>
</tr>
<tr>
<td>LIBRARY</td>
<td>MYEXIT</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LISTING</td>
<td>MYEXIT</td>
<td>164</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>SOURCE</td>
<td>MYEXIT</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
7) Where can I learn more about this?

- IBM High Level Assembler for z/OS & z/VM & z/VSE
  - SG26-4641
  - Chapter 4 Providing user exits
Summary

• We have learned that Assembler exits can be a powerful tool to enhance the assembler language.
• We can implement "C" style comments and bi_lingual C/ASM
• We can use these to enforce coding standards.
• We can use these to produce better listings, especially for non-z/OS data streams.
Full Source Code

You can download the full source code for ASMEXITS from


You will be asked to agree to the usual disclaimers, etc.
Questions?
Trademarks and Copyrights

- IBM®, z/OS®, z/VM®, z/VSE®, and RACF® are registered trademarks of the IBM Corporation.
- Windows® is a registered trademark of Microsoft Corporation.
- Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.