

ZSTRMAC.ZSM

```
*****
* Copyright 2008 Automated Software Tools Corporation *
* This source code is part of z390 assembler/emulator package *
* The z390 package is distributed under GNU general public license *
* Author - Don Higgins *
* Date - 08/13/08 *
*****
* 08/13/22 RPI 896 TRANSLATE Z390 ZSTRMAC EXTENSIONS TO STD HLASM
* 1. Z390 BOOTSTRAP VER - RT\TEST\ZSTRMAC1.MLC
* 1. STRUCTURED VERSION - LINKLIB\ZSTRMAC.ZSM
* 2. GEN HLASM COMP VER - LINKLIB\ZSTRMAC.MLC VIA ZSTRMAC1
* 09/17/08 RPI 911 CHANGE ASELECT TO ACASE AND APM TO ACALL AND
* SUPPORT LOWER CASE
*****
* ZSTRMAC READS SYSUT1 SOURCE FILE AND OUTPUTS SYSUT2 SOURCE FILE
* WITH TRANSLATION OF FOLLOWING Z390 ZSTRMAC EXTENSIONS TO STD HLASM:
* 1. AIF (EXP) > AIF (NOT(EXP)).AIF_N_B
* > .....
* 2. AELSEIF (EXP) > AGO .AIF_N_E
* > .AIF_B AIF (EXP).AIF_N_B+1
* > .....
* 3. AELSE > AGO .AIF_N_E
* > .AIF_N_B+1 ANOP
* > .....
* 4. AEND > .AIF_N_E ANOP
* 5. ACALL NAME > &ACALL_N SETA B
* > AGO .ACL_N
* > .ACL_N_B ANOP
* > .....
* 6. AENTRY NAME > .ACL_N ANOP
* > .....
* 7. AEXIT > AGO .ACL_N_E (EXIT NON AIF STRUCURE)
* > .....
* AEND > .ACL_N_E AGO (&ACALL_N).ACL_N_1,.ACL_N_2,
* > .ACL_N_B
* 8. AWHILE (EXP) > .AWH_N_T AIF (NOT(EXP)).AWH_N_E
* > .....
* AEND > AGO .AWH_N_T
* > .AWH_N_E ANOP
* > .....
* 9. AUNTIL (EXP) > AGO .AUN_N
* > .AUN_N_T AIF (EXP).AUN_N_E
* > .AUN_N ANOP
* > .....
* AEND > AGO .AUN_N_T
```

ZSTRMAC.ZSM

```

*          > .AUN_N_E ANOP
*          > .....
* 10. ACASE (EXP) > AGO .ACS_N_AGO
* 11. AWHEN V1,V2 > .ACS_N_B1 ANOP VN=(N,C'?',X'??', OR (V1,V2)
*          > .....
*     AWHEN V2    > AGO .ACS_N_E
*          > .ACS_N_B2 ANOP
*          > .....
*     AELSE      > AGO .ACS_N_E
*          > .ACS_N_X ANOP
*          > .....
*     AEND       > AGO .ACS_N_E
*          > .ACS_N_G AGO (EXP).ACS_N_B1,.ACS_N_X,.ACS_N_B2
*          > AGO .ACS_N_X
*          > .ACS_N_E ANOP
* 12. :label stmt > place label in label field without the :
*          and indent the stmt to start at the original :
*

```

\* NOTES:

- \* 1. THE ORIGINAL BOOTSTRAP VERSION IS IN RT\TEST\ZSTRMAC1.MLC  
 \* ALONG WITH THE FIRST TEST PROGRAM TESTZSM1.ZSM WHICH IS  
 \* TRANSLATED TO TESTZSM1.MLC USING ZSTRMAC1.MLC.
- \* 2. TO RUN TRANSLATOR USING HLASM:
  - \* A. REMOVE DDNAME= EXTENSIONS FROM AREAD AND PUNCH
  - \* B. PLACE INPUT SOURCE AFTER PROGRAM SOURCE IN SYSIN.
  - \* C. CHANGE EOF LOGIC TO CHECK FOR EOF RECORD SUCH AS "END"

\*\*\*\*\*

MACRO

ZSTRMAC

```

LCLA  &ERRORS          TOTAL ERROR MESSAGES
LCLA  &AEND_TOT,&AENTRY_TOT,&AEXIT_TOT,&AIF_TOT,&ACALL_TOT
LCLA  &ACASE_TOT,&AUNTIL_TOT,&AWHEN_TOT,&AWHILE_TOT
LCLC  &TEXT           LINE OF TEXT READ BY READ_TEXT
LCLB  &EOF            END OF FILE
LCLA  &LINE           TOTAL INPUT LINES
LCLB  &GEN_AIF_ERR    SYNTAX ERROR IN GEN_AIF
LCLB  &FIND_NAME_ERR  SYNTAX ERROR FINDING ACALL/AENTRY NAME
LCLB  &FIND_PARM_ERR  SYNTAX ERROR FINDING FIRST PARM
LCLB  &FIND_EXP_ERR   SYNTAX ERROR FINDING (..) FOR AIF/ACASE
LCLB  &GET_VALUE_ERR  ERROR PARSING DEC, '?', OR X'??'
LCLA  &LVL            CURRENT LEVEL OF STRUCTURE
LCLC  &LVL_TYPE(50)  TYPE AIF/ACASE/AENTRY
LCLA  &LVL_TCNT(50)  TYPE INSTANCE COUNTER
LCLB  &LVL_TEND(50)  TYPE END LABEL REQ FOR MULT BLKS
LCLA  &LVL_BCNT(50)  BLOCK COUNTER WITHIN TYPE INSTANCE

```

ZSTRMAC.ZSM

LCLC &LVL\_ACASE(50) ACASE COMPUTED AGO STATEMENT  
 LCLA &LVL\_ACASE\_FIRST(50) ACASE FIRST WHEN VALUE 0-255  
 LCLA &LVL\_ACASE\_LAST(50) ACASE LAST WHEN VALUE 0-255  
 LCLB &LVL\_AELSE(50) AELSE BLOCK DEFINED FOR ACASE  
 LCLA &IS\_OP START OF OPCODE  
 LCLA &IS\_OP\_END ENDOF OF OPCODE+1  
 LCLA &IS\_EXP START OF AIF EXP (...)  
 LCLA &ACALL\_INDEX INDEX TO ACALL/AENTRY VIA FIND\_NAME  
 LCLA &ACALL\_NAME\_TOT TOTAL PERFORMED ROUTINES  
 LCLC &ACALL\_NAME(100) NAMES OF PERFORMED ROUTINES  
 LCLA &ACALL\_CNT(100) EXIT COUNT FOR ROUTINES  
 LCLB &ACALL\_DEF(100) FLAG FOR DUP AND MISSING ERRORS

. \*  
 . \*  
 . \*

READ SYUT1 AND OUTPUT SYSUT2 WITH STRUCTURED MACRO CODE

```

ACALL READ_REC
AWHILE (NOT &EOF)
    ACALL PROC_REC
    ACALL READ_REC
AEND
:&ACALL_INDEX SETA 1
AWHILE (&ACALL_INDEX LE &ACALL_NAME_TOT)
    AIF (NOT &ACALL_DEF(&ACALL_INDEX))
        :&MSG SETC 'MISSING AENTRY FOR
&ACALL_NAME(&ACALL_INX
    DEX) '
        ACALL ERR_MSG
    AEND
    :&ACALL_INDEX SETA &ACALL_INDEX+1
AEND
  
```

```

MNOTE 'ZSTRMAC GENERATED LINES = &LINE'
MNOTE 'ZSTRMAC TOTAL ERRORS = &ERRORS'
MNOTE 'ZSTRMAC TOTAL AEND = &AEND_TOT'
MNOTE 'ZSTRMAC TOTAL AENTRY = &AENTRY_TOT'
MNOTE 'ZSTRMAC TOTAL AEXIT = &AEXIT_TOT'
MNOTE 'ZSTRMAC TOTAL AIF = &AIF_TOT'
MNOTE 'ZSTRMAC TOTAL ACALL = &ACALL_TOT'
MNOTE 'ZSTRMAC TOTAL ACASE = &ACASE_TOT'
MNOTE 'ZSTRMAC TOTAL AWHEN = &AWHEN_TOT'
MNOTE 'ZSTRMAC TOTAL AWHILE = &AWHILE_TOT'
MNOTE 'ZSTRMAC TOTAL AUNTIL = &AUNTIL_TOT'
  
```

. \*  
 . \*  
 . \*

READ LOGICAL RECORD INTO &REC WITH TRAILING COMMENTS IF ANY

```

AENTRY READ_REC
  
```

ZSTRMAC.ZSM

```

ACALL READ_TEXT
ACTR 10000
AIF (NOT &EOF)
    AIF (K'&TEXT GE 72)
        :&REC SETC '&TEXT'(1,71)
        AIF ('&TEXT'(72,1) NE ' ')
            ACALL READ_TEXT
            AWHILE (NOT &EOF
X
                AND K'&TEXT GE 72
X
                AND '&TEXT'(1,15) EQ (15)' '
X
                AND '&TEXT'(72,1) NE ' ')
                :&REC SETC '&REC'.'&TEXT'(16,71-15)
                ACALL READ_TEXT
            AEND
        AIF (NOT &EOF)
            AIF (K'&TEXT GE 16
X
                AND '&TEXT'(1,15) EQ (15)' ' )
                :&REC SETC '&REC'.'&TEXT'(16,*)
            AELSE
                :&MSG SETC 'INVALID CONTINUATION'
                ACALL ERR_MSG
            AEND
        AELSE
            :&MSG SETC 'END OF FILE ON CONTINUE'
            ACALL ERR_MSG
        AEND
    AEND
AELSE
    :&REC SETC '&TEXT'(1,*)
AEND
AEND
AEND

```

```

.*
.* READ LOGICAL LINE INTO &TEXT AND SET &EOF IF END OF FILE
.*

```

```

AENTRY READ_TEXT
:&TEXT AREAD DDNAME=SYSUT1
AIF ('&TEXT' EQ ' ')
    :&EOF SETB 1
AELSE
    :&LINE SETA &LINE+1

```

ZSTRMAC.ZSM

AEND

AEND

```
.*
.* PROCESS REC BY SCANNING FOR A??? OPCODES AND GENERATING
.* COMMENT AND GENERATED CODE ELSE COPY REC
.*
```

```
AENTRY PROC_REC
ACALL FIND_OPCODE
AIF  ('&OPCODE'(1,1) NE 'A')
    ACALL COPY_REC
AELSEIF ('&OPCODE' EQ 'AIF')
    ACALL PROC_AIF
AELSEIF ('&OPCODE' EQ 'AELSE')
    ACALL PROC_AELSE
AELSEIF ('&OPCODE' EQ 'AELSEIF')
    ACALL PROC_AELSEIF
AELSEIF ('&OPCODE' EQ 'AEND')
    ACALL PROC_AEND
AELSEIF ('&OPCODE' EQ 'ACALL')
    ACALL PROC_ACALL
AELSEIF ('&OPCODE' EQ 'AENTRY')
    ACALL PROC_AENTRY
AELSEIF ('&OPCODE' EQ 'AEXIT')
    ACALL PROC_AEXIT
AELSEIF ('&OPCODE' EQ 'AWHILE')
    ACALL PROC_AWHILE
AELSEIF ('&OPCODE' EQ 'AUNTIL')
    ACALL PROC_AUNTIL
AELSEIF ('&OPCODE' EQ 'ACASE')
    ACALL PROC_ACASE
AELSEIF ('&OPCODE' EQ 'AWHEN')
    ACALL PROC_AWHEN
AELSE
    ACALL COPY_REC
AEND
AEND
```

```
.*
.* FIND_OPCODE - SET &OPCODE, &IS_OP, AND &IS_OP_END
.*
```

```
AENTRY FIND_OPCODE
:&OPCODE SETC ' '
:&IS_OP  SETA 0
:&IS_OP_END SETA 0
:&I  SETA ('&REC' INDEX ' ')
AIF  (&I GT 0)
```

ZSTRMAC.ZSM

```

:&J SETA ('&REC'(&I,*) FIND 'A:')
AIF (&J EQ 0)
    AEXIT AENTRY NOT A???? SO DON'T RETURN OPCODE
AELSEIF ('&REC'(1,2) EQ '.*')
    AEXIT AENTRY NO OPCODE FOR COMMENTS WITH A? EITHER
AELSEIF ('&REC'(1,1) EQ '*')
    AEXIT AENTRY
AELSEIF ('&REC'(&I,&J-1) NE (&J-&I)' ')
    AEXIT AENTRY
AEND
:&I SETA &I+&J-1
AIF (&I LT K'&REC-1)
    :&IS_OP SETA &I
    :&J SETA ('&REC'(&I,*) INDEX ' ')
    AIF (&J EQ 0)
        :&I SETA K'&REC+1
    AELSE
        :&I SETA &I+&J-1
    AEND
    :&OPCODE SETC (UPPER '&REC'(&IS_OP,&I-&IS_OP))
    :&IS_OP_END SETA &I
AEND

```

AEND  
AEND

```

.*
.* COPY UNKNOWN RECORDS WITH :LABEL MOVED TO LABEL FIELD
.*

```

```

AENTRY COPY_REC
AIF (K'&OPCODE GT 1

```

X

```

    AND &IS_OP_END LT K'&REC)
    AIF ('&REC'(&IS_OP,1) EQ ':')
        ACALL FIND_PARM
        AIF (NOT &FIND_PARM_ERR)
            :&SPACES SETA &IS_OP-K'&OPCODE
            AIF (&SPACES LE 0)
                :&SPACES SETA 1
            AEND
            :&REC SETC
'&REC'(&IS_OP+1,K'&OPCODE-1).(&SPACX
    ES)' '.'&REC'(&IS_PARM,*)
        AEND
    AEND
AEND
:&PCH_REC SETC '&REC'

```

ZSTRMAC.ZSM

ACALL PUNCH\_REC  
AEND

.\*  
.\* AELSE - GEN MACRO COMMENT AND GEN AGO TO AEND AND LABEL FOR ALT.  
BLK

.\*  
AENTRY PROC\_AELSE  
: &AELSE\_TOT SETA &AELSE\_TOT+1  
: &PCH\_REC SETC '.\*'.'&REC'(3,\*)  
ACALL PUNCH\_REC  
AIF (&LVL GE 1)  
AIF (&LVL\_TYPE(&LVL) EQ 'AIF')  
ACALL PROC\_AELSE\_AIF  
AELSEIF (&LVL\_TYPE(&LVL) EQ 'ACASE')  
ACALL PROC\_AELSE\_ACASE  
AELSE  
: &MSG SETC 'INVALID AELSE TYPE &LVL\_TYPE(&LVL)'  
ACALL ERR\_MSG  
AEND  
AELSE  
: &MSG SETC 'MISSING AIF OR ACASE'  
ACALL ERR\_MSG  
AEND  
AEND

.\*  
.\* AELSE\_AIF  
.\*  
AENTRY PROC\_AELSE\_AIF  
: &LVL\_TEND(&LVL) SETB 1 REQUEST AEND TO GEN END TARGET  
: &PCH\_REC SETC (&IS\_OP+1)' '.'AGO .AIF\_&LVL\_TCNT(&LVL)\_E'  
ACALL PUNCH\_REC  
: &PCH\_REC SETC '.AIF\_&LVL\_TCNT(&LVL)\_&LVL\_BCNT(&LVL)'  
ACALL PUNCH\_LAB  
: &LVL\_BCNT(&LVL) SETA 0 RESET TO INDICATE NO BLK LABEL REQ  
AEND

.\*  
.\* AELSE\_ACASE  
.\*  
AENTRY PROC\_AELSE\_ACASE  
AIF (&LVL\_BCNT(&LVL) GT 0)  
: &PCH\_REC SETC (&IS\_OP+1)' '.'AGO  
.ACS\_&LVL\_TCNT(&LVL)X  
\_E'  
ACALL PUNCH\_REC  
AEND

ZSTRMAC.ZSM

```

:&LVL_AELSE(&LVL) SETB 1 INDICATE AELSE BLOCK DEFINED
:&PCH_REC SETC '.ACS_&LVL_TCNT(&LVL)_X'
ACALL PUNCH_LAB
AEND

```

.  
\*  
\*  
.

AELSEIF - GEN MACRO COMMENT AND GEN AIF TO END OF BLK,CUR BLK LAB

```

AENTRY PROC_AELSEIF
:&AELSEIF_TOT SETA &AELSEIF_TOT+1
:&PCH_REC SETC '.*'.'&REC'(3,*)
ACALL PUNCH_REC
AIF (&LVL GE 1)
    AIF (&LVL_TYPE(&LVL) EQ 'AIF')
        :&LVL_TEND(&LVL) SETB 1 REQUEST AEND TO GEN END
        :&PCH_REC SETC (&IS_OP+1)' '.'AGO

```

```

.AIF_&LVL_TCNT(&X
    LVL)_E'
    ACALL PUNCH_REC
    :&PCH_REC SETC
'.AIF_&LVL_TCNT(&LVL)_&LVL_BCNT(&LVL)X

```

```

    ACALL PUNCH_LAB
    :&LVL_BCNT(&LVL) SETA &LVL_BCNT(&LVL)+1 NEW TARGET
    :&GEN_AIF_TRUE SETB 0 GEN BRANCH IF FALSE
    :&GEN_AIF_TAG SETC '&LVL_BCNT(&LVL)X
ACALL GEN_AIF
AIF (&GEN_AIF_ERR)
    :&MSG SETC 'AELSEIF AIF ERROR'
    ACALL ERR_MSG

```

```

    AELSE
        ACALL PUNCH_REC
    AEND

```

```

AELSE
    :&MSG SETC 'AELSEIF MISSING AIF ERROR'
    ACALL ERR_MSG

```

```

AEND

```

```

AELSE
    :&MSG SETC 'AELSEIF MISSING AIF ERROR'
    ACALL ERR_MSG

```

```

AEND
AEND

```

.  
\*  
\*  
.

AEND - GEN TERMINATION FOR AENTRY,AIF,ACASE,AUNTIL,AWHILE

```

AENTRY PROC_AEND

```

ZSTRMAC.ZSM

```

:&AEND_TOT SETA &AEND_TOT+1
:&PCH_REC SETC '.*'. '&REC'(3,*)
ACALL PUNCH_REC
AIF (&LVL GE 1)
    AIF (&LVL_TYPE(&LVL) EQ 'AIF')
        ACALL PROC_AEND_AIF
    AELSEIF (&LVL_TYPE(&LVL) EQ 'AWHILE')
        ACALL PROC_AEND_AWHILE
    AELSEIF (&LVL_TYPE(&LVL) EQ 'ACASE')
        ACALL PROC_AEND_ACASE
    AELSEIF (&LVL_TYPE(&LVL) EQ 'AENTRY')
        ACALL PROC_AEND_AENTRY
    AELSEIF (&LVL_TYPE(&LVL) EQ 'AUNTIL')
        ACALL PROC_AEND_AUNTIL
    AELSE
        :&MSG SETC 'AEND INVALID TYPE &LVL_TYPE(&LVL)'
        ACALL ERR_MSG
    AEND
AELSE
    :&MSG SETC 'AEND MISSING AIF OR OTHER STRUCTURE'
    ACALL ERR_MSG
AEND
AEND
.*
.* AEND_AENTRY
.*
AENTRY PROC_AEND_AENTRY
:&ACALL_INDEX SETA &LVL_BCNT(&LVL)
AIF (&ACALL_CNT(&ACALL_INDEX) GT 0)
    AIF (&LVL_TEND(&LVL))
        :&PCH_REC SETC '.ACL_&ACALL_INDEX._E'
        ACALL PUNCH_LAB
    AEND
    :&PCH_REC SETC (&IS_OP+1)' ' .'AGO
(&&ACALL_&ACALL_INDEX
    X._&ACALL_NAME(&ACALL_INDEX)).ACL_&ACALL_INDEX._1'
    :&I SETA 2
    AWHILE (&I LE &ACALL_CNT(&ACALL_INDEX))
        :&PCH_REC SETC '&PCH_REC, .ACL_&ACALL_INDEX._&I'
        :&I SETA &I+1
    AEND
    ACALL PUNCH_REC
AELSE
    :&MSG SETC 'AENTRY &ACALL_NAME(&ACALL_INDEX) NOT USED'
    ACALL ERR_MSG

```

ZSTRMAC.ZSM

```

AEND
:&PCH_REC SETC '.ACL_&ACALL_INDEX._SKIP'
ACALL PUNCH_LAB
:&LVL SETA &LVL-1      CURRENT LEVEL
AEND

```

```

.*
.* AEND_AIF
.*

```

```

AENTRY PROC_AEND_AIF
AIF (&LVL_BCNT(&LVL) GT 0)
:&PCH_REC SETC '.AIF_&LVL_TCNT(&LVL)_&LVL_BCNT(&LVL) '
ACALL PUNCH_LAB
AEND
AIF (&LVL_TEND(&LVL))
:&PCH_REC SETC '.AIF_&LVL_TCNT(&LVL)_E '
ACALL PUNCH_LAB
AEND
:&LVL SETA &LVL-1      CURRENT LEVEL
AEND

```

```

.*
.* AEND_AUNTIL
.*

```

```

AENTRY PROC_AEND_AUNTIL
:&PCH_REC SETC (&IS_OP+1)' '. 'AGO .AUN_&LVL_TCNT(&LVL)_T '
ACALL PUNCH_REC
:&PCH_REC SETC '.AUN_&LVL_TCNT(&LVL)_E '
ACALL PUNCH_LAB
:&LVL SETA &LVL-1      CURRENT LEVEL
AEND

```

```

.*
.* AEND_AWHILE
.*

```

```

AENTRY PROC_AEND_AWHILE
:&PCH_REC SETC (&IS_OP+1)' '. 'AGO .AWH_&LVL_TCNT(&LVL)_T '
ACALL PUNCH_REC
:&PCH_REC SETC '.AWH_&LVL_TCNT(&LVL)_E '
ACALL PUNCH_LAB
:&LVL SETA &LVL-1      CURRENT LEVEL
AEND

```

```

.*
.* AEND_ACASE
.*

```

```

AENTRY PROC_AEND_ACASE
AIF (&LVL_BCNT(&LVL) GT 0)
:&PCH_REC SETC (&IS_OP+1)' '. 'AGO

```

ZSTRMAC.ZSM

```

.ACS_&LVL_TCNT(&LVL)X
  _E'
  ACALL PUNCH_REC
  :&PCH_REC SETC '.ACS_&LVL_TCNT(&LVL)_G'
  ACALL PUNCH_LAB
  AIF (&LVL_AELSE(&LVL))
    :&ELSE_LAB SETC '.ACS_&LVL_TCNT(&LVL)_X'
  AELSE
    :&ELSE_LAB SETC '.ACS_&LVL_TCNT(&LVL)_E'
  AEND
  :&PCH_REC SETC '&LVL_ACASE(&LVL)'
  AIF (&LVL_ACASE_FIRST(&LVL) NE 1)
    :&OFFSET SETC '+1-&LVL_ACASE_FIRST(&LVL)'
    :&PCH_REC SETC
'&PCH_REC'(1,K'&PCH_REC-1).'&OFFSET)X
  '
  AEND
  :&VAL_BLK SETC 'ACASE_&LVL_TCNT(&LVL)_VAL_BLK'
  :&VALUE SETA &LVL_ACASE_FIRST(&LVL)
  :&COMMA SETC ','
  AWHILE (&VALUE LE &LVL_ACASE_LAST(&LVL))
    AIF (&(&VAL_BLK)(&VALUE+1) GT 0)
      :&PCH_REC SETC
'&PCH_REC&COMMA..ACS_&LVL_TX
  CNT(&LVL)_&(&VAL_BLK)(&VALUE+1)'
      :&COMMA SETC ','
    AELSE
      :&PCH_REC SETC '&PCH_REC&COMMA&ELSE_LAB'
      :&COMMA SETC ','
    AEND
      :&VALUE SETA &VALUE+1
  AEND
  ACALL PUNCH_REC
  AIF (&LVL_AELSE(&LVL))
    :&PCH_REC SETC (&IS_OP+1)' '.'AGO
.ACS_&LVL_TCNTX
  (&LVL)_X'
  ACALL PUNCH_REC
  AEND
  :&PCH_REC SETC '.ACS_&LVL_TCNT(&LVL)_E'
  ACALL PUNCH_LAB
  :&LVL SETA &LVL-1 CURRENT LEVEL
  AELSE
    :&MSG SETC 'NO WHEN FOUND FOR ACASE'
    ACALL ERR_MSG

```

ZSTRMAC.ZSM

AEND

AEND

.\*

.\* AENTRY - GEN AGO BRANCH AROUND PENTRY/PEND AND LABEL FOR ENTRY

.\*

AENTRY PROC\_AENTRY

:&AENTRY\_TOT SETA &AENTRY\_TOT+1

:&PCH\_REC SETC '.\*'. '&REC'(3,\*)

ACALL PUNCH\_REC

ACALL FIND\_NAME

AIF (&FIND\_NAME\_ERR)

:&MSG SETC 'AENTRY NAME NOT FOUND'

ACALL ERR\_MSG

AELSEIF (&ACALL\_DEF(&ACALL\_INDEX))

:&MSG SETC 'AENTRY DUPLICATE NAME FOUND - &NAME'

ACALL ERR\_MSG

AELSE

:&ACALL\_DEF(&ACALL\_INDEX) SETB 1 SET DEFINITION FLAG

:&LVL SETA &LVL+1

:&LVL\_TYPE(&LVL) SETC 'AENTRY'

:&LVL\_TEND(&LVL) SETB 0 RESET END LABEL

REQ.

:&LVL\_TCNT(&LVL) SETA &AENTRY\_TOT

:&LVL\_BCNT(&LVL) SETA &ACALL\_INDEX SAVE FOR AEND

:&PCH\_REC SETC (&IS\_OP+1)' '.'AGO

.ACL &ACALL\_INDEX.\_SX

KIP'

ACALL PUNCH\_REC

:&PCH\_REC SETC

' .ACL &ACALL\_INDEX.\_&ACALL\_NAME(&ACALL\_INX  
DEX)'

ACALL PUNCH\_LAB

AEND

AEND

.\*

.\* AEXIT - EXIT TO FIRST MATCHING TYPE FOUND

.\*

AENTRY PROC\_AEXIT

:&AEXIT\_TOT SETA &AEXIT\_TOT+1

:&PCH\_REC SETC '.\*'. '&REC'(3,\*)

ACALL PUNCH\_REC

ACALL FIND\_PARM

AIF (&FIND\_PARM\_ERR)

ZSTRMAC.ZSM

```

    :&MSG SETC 'AEXIT TYPE PARM NOT FOUND'
    ACALL ERR_MSG
    AEXIT AENTRY
AEND
    :&EXIT_LVL SETA 0
    :&TEST_LVL SETA &LVL
    AWHILE    (&TEST_LVL GT 0)
        AIF    (&LVL_TYPE(&TEST_LVL) EQ '&PARM')
            :&EXIT_LVL SETA &TEST_LVL
            :&TEST_LVL SETA 0
        AELSE
            :&TEST_LVL SETA &TEST_LVL-1
    AEND
AEND
    AIF    (&EXIT_LVL GT 0)
        :&LVL_TEND(&EXIT_LVL) SETB 1    REQUEST END LABEL
        AIF    (&LVL_TYPE(&EXIT_LVL) EQ 'AENTRY')
            :&ACALL_INDEX SETA &LVL_BCNT(&EXIT_LVL)
            :&PCH_REC SETC (&IS_OP+1)' '.'AGO
    .ACL_&ACALL_INDX
        EX._E'
        ACALL PUNCH_REC
    AELSE
        :&PCH_REC SETC (&IS_OP+1)' '.'AGO
    .'.'&LVL_TYPE(&X
        EXIT_LVL)'(1,3).'&LVL_TCNT(&EXIT_LVL)_E'
        ACALL PUNCH_REC
    AEND
    AELSE
        :&MSG SETC 'AEXIT NOT WITHIN AENTRY, AWHILE, ACASE'
        ACALL ERR_MSG
    AEND
    AEND
.*
.* AIF - GEN MACRO COMMENT AND AIF TO GENERATED END LABEL AT NEXT
LEVEL
.*
    AENTRY PROC_AIF
    :&AIF_TOT SETA  &AIF_TOT+1      AIF COUNTER
    :&LVL      SETA  &LVL+1        CURRENT LEVEL
    :&LVL_TYPE(&LVL) SETC 'AIF' CURRENT LEVEL TYPE
    :&LVL_TCNT(&LVL) SETA &AIF_TOT PRIMARY TYPE COUNTER
    :&LVL_TEND(&LVL) SETB 0          RESET REQ FOR AELSEIF END
LABEL
    :&LVL_BCNT(&LVL) SETA 1          BLOCK COUNTER (ELSEIF, WHEN)

```

```

                                ZSTRMAC.ZSM
:&PCH_REC SETC '.*'.'&REC'(3,*)
ACALL PUNCH_REC
:&GEN_AIF_TRUE SETB 0                                GEN BRANCH IF FALSE
:&GEN_AIF_TAG SETC '&LVL_BCNT(&LVL) '
ACALL GEN_AIF
AIF (&GEN_AIF_ERR)
    :&MSG SETC 'AIF EXPRESSION SYNTAX ERROR'
    ACALL ERR_MSG
AELSE
    ACALL PUNCH_REC
AEND
AEND
.*
.* ACALL - GEN AGO TO PERFORMED ROUTINE
.*

AENTRY PROC_ACALL
:&ACALL_TOT SETA &ACALL_TOT+1
:&PCH_REC SETC '.*'.'&REC'(3,*)
ACALL PUNCH_REC
ACALL FIND_NAME
AIF (&FIND_NAME_ERR)
    :&MSG SETC 'ACALL NAME SYNTAX ERROR'
    ACALL ERR_MSG
AELSE
    :&ACALL_CNT(&ACALL_INDEX) SETA
&ACALL_CNT(&ACALL_INDEX)+X
    1
    :&PCH_REC SETC
'&&ACALL_&ACALL_INDEX._&ACALL_NAME(&ACALLX
    _INDEX) '
    :&SPACES SETA &IS_OP-K'&PCH_REC+1
    AIF (&SPACES LE 0)
        :&SPACES SETA 1
    AEND
    :&PCH_REC SETC '&PCH_REC'.(&SPACES)' '.'SETA
&ACALL_CX
    NT(&ACALL_INDEX) '
    ACALL PUNCH_REC
    :&PCH_REC SETC (&IS_OP+1)' '.'AGO
.&ACL_&ACALL_INDEX._&X
    ACALL_NAME(&ACALL_INDEX) '
    ACALL PUNCH_REC
    :&PCH_REC SETC
'&ACL_&ACALL_INDEX._&ACALL_CNT(&ACALL_INDX

```

ZSTRMAC.ZSM

EX)'  
ACALL PUNCH\_LAB

AEND  
AEND

.\*  
.\* ACASE - GEN AGO TO .ACS\_N\_AGO AND SAVE AGO EXPRESSION  
.\*

AENTRY PROC\_ACASE  
:&ACASE\_TOT SETA &ACASE\_TOT+1 ACASE COUNTER  
:&LVL SETA &LVL+1 CURRENT LEVEL  
:&LVL\_TYPE(&LVL) SETC 'ACASE' CURRENT LEVEL TYPE  
:&LVL\_TCNT(&LVL) SETA &ACASE\_TOT ACASE INSTANCE  
:&LVL\_BCNT(&LVL) SETA 0 RESET ACASE AWHEN BLOCKS  
:&LVL\_AELSE(&LVL) SETB 0 ASSUME NO AELSE BLOCK  
:&VAL\_BLK SETC 'ACASE\_&LVL\_TCNT(&LVL)\_VAL\_BLK'  
LCLA &(&VAL\_BLK)(256)  
:&LVL\_ACASE\_FIRST(&LVL) SETA 257  
:&LVL\_ACASE\_LAST(&LVL) SETA -1  
:&PCH\_REC SETC '.\*'.&REC'(3,\*)

ACALL PUNCH\_REC  
ACALL FIND\_EXP  
AIF (&FIND\_EXP\_ERR)  
:&MSG SETC 'ACASE EXPRESSION ERROR'  
ACALL ERR\_MSG

AELSE  
:&LVL\_ACASE(&LVL) SETC (&IS\_OP+1)' '.'AGO

'.&REC'(&ISX  
\_EXP,&IS\_EXP\_END-&IS\_EXP+1)  
:&I SETA 1  
AWHILE (&I LE 256)  
:&(&VAL\_BLK)(&I) SETA 0  
:&I SETA &I+1

AEND  
:&PCH\_REC SETC (&IS\_OP+1)' '.'AGO

.ACS\_&LVL\_TCNT(&LVL)X  
\_G'  
ACALL PUNCH\_REC

AEND  
AEND

.\*  
.\* AUNTIL - GEN AGO TO BLOCK, THEN LABEL TEST AIF TO EXIT  
.\*

AENTRY PROC\_AUNTIL  
:&AUNTIL\_TOT SETA &AUNTIL\_TOT+1 AUNTIL COUNTER  
:&LVL SETA &LVL+1 CURRENT LEVEL

ZSTRMAC.ZSM

```

:&LVL_TYPE(&LVL) SETC 'AUNTIL' CURRENT LEVEL TYPE
:&LVL_TCNT(&LVL) SETA &AUNTIL_TOT PRIMARY TYPE COUNTER
:&PCH_REC SETC '.*'.'&REC'(3,*)
ACALL PUNCH_REC
:&PCH_REC SETC (&IS_OP+1)' '.'AGO .AUN_&LVL_TCNT(&LVL)'
ACALL PUNCH_REC
:&PCH_REC SETC '.AUN_&LVL_TCNT(&LVL)_T'
ACALL PUNCH_REC
:&GEN_AIF_TRUE SETB 1 GEN BRANCH IF TRUE
:&GEN_AIF_TAG SETC 'E'
ACALL GEN_AIF
AIF (&GEN_AIF_ERR)
:&MSG SETC 'AUNTIL EXPRESSION ERROR'
ACALL ERR_MSG
AELSE
ACALL PUNCH_REC
AEND
:&PCH_REC SETC '.AUN_&LVL_TCNT(&LVL)'
ACALL PUNCH_REC
AEND

```

.\*

.\* AWHEN - GEN .ACS\_N\_I LABEL FOR INDEX AND UPDATE INDEX VAL\_BLK

.\*

```

AENTRY PROC_AWHEN
:&PCH_REC SETC '.*'.'&REC'(3,*)
ACALL PUNCH_REC
:&AWHEN_TOT SETA &AWHEN_TOT+1
AIF (&LVL GE 1)
:&VAL_BLK SETC 'ACASE_&LVL_TCNT(&LVL)_VAL_BLK'
AIF (&LVL_TYPE(&LVL) EQ 'ACASE')
AIF (&LVL_BCNT(&LVL) GT 0 OR &LVL_AELSE(&LVL))
:&PCH_REC SETC (&IS_OP+1)' '.'AGO

```

.ACS\_&LVLX

```

    _TCNT(&LVL)_E'
        ACALL PUNCH_REC
    AEND
    :&LVL_BCNT(&LVL) SETA &LVL_BCNT(&LVL)+1
    ACALL FIND_PARM
    AIF (&FIND_PARM_ERR)
        :&MSG SETC 'AWHEN VALUE ERROR'
    ACALL ERR_MSG
    AELSE
        ACALL PROC_AWHEN_VALUES
    AEND
    :&PCH_REC SETC

```

```

                                ZSTRMAC.ZSM
'.ACS_&LVL_TCNT(&LVL)_&LVL_BCNT(&LVLX
    )'
        ACALL PUNCH_LAB
    AELSE
        :&MSG SETC 'AWHEN MISSING ACASE'
        ACALL ERR_MSG
    AEND
AELSE
    :&MSG SETC 'AWHEN MISSING ACASE'
    ACALL ERR_MSG
AEND
AEND
.*
.* PROC_WHEN_VALUES V1,V2,(V3,V4) WHERE VN = DEC, C'?', OR X'??'
.*

AENTRY PROC_AWHEN_VALUES
:&VALUE_CNT SETA 0
AWHILE (&IS_PARM LE K'&REC)
    ACASE (C2A('&REC'(&IS_PARM,1)))
        AWHEN C'(' SET RANGE (V1,V2)
            :&IS_PARM SETA &IS_PARM+1
            ACALL GET_VALUE
            AIF (&GET_VALUE_ERR)
                :&MSG SETC 'INVALID RANGE VALUE'
                ACALL ERR_MSG
                AEXIT AENTRY          EXIT AFTER VALUE ERROR
        AEND
        :&VALUE1 SETA &VALUE
        AIF ('&REC'(&IS_PARM,1) NE ',')
            :&MSG SETC 'MISSING RANGE ,'
            ACALL ERR_MSG
            AEXIT AENTRY
        AEND
        :&IS_PARM SETA &IS_PARM+1
        ACALL GET_VALUE
        AIF (&GET_VALUE_ERR)
            :&MSG SETC 'INVALID RANGE VALUE'
            ACALL ERR_MSG
            AEXIT AENTRY          EXIT AFTER VALUE ERROR
        AEND
        :&VALUE2 SETA &VALUE
        AIF ('&REC'(&IS_PARM,1) NE ')')
            :&MSG SETC 'MISSING RANGE )'
            ACALL ERR_MSG
            AEXIT AENTRY

```

```

                                ZSTRMAC.ZSM
                                AEND
                                :&IS_PARM SETA &IS_PARM+1
                                :&VALUE SETA &VALUE1
                                AWHILE (&VALUE LE &VALUE2)
                                    ACALL SET_VAL_BLK
                                    :&(&VAL_BLK)(&VALUE+1) SETA
&LVL_BCNT(&LVL)
                                    :&VALUE SETA &VALUE+1
                                AEND
                                AWHEN C' '
                                    AEXIT AWHILE
                                AWHEN C', '
                                    :&IS_PARM SETA &IS_PARM+1
                                AELSE
                                    ACALL GET_VALUE
                                    AIF (&GET_VALUE_ERR)
                                        :&MSG SETC 'INVALID VALUE'
                                        ACALL ERR_MSG
                                        AEXIT AENTRY
                                    AEND
                                    ACALL SET_VAL_BLK
                                AEND
                                AEND
                                AIF (&VALUE_CNT EQ 0)
                                    :&MSG SETC 'NO AWHEN VALUES FOUND'
                                    ACALL ERR_MSG
                                AEND
                                AEND
.*
.* SET_VAL_BLK  AWHEN BLOCK NUMBER FOR VALUE
.*
                                AENTRY SET_VAL_BLK
                                AIF (&VALUE LT &LVL_ACASE_FIRST(&LVL))
                                    :&LVL_ACASE_FIRST(&LVL) SETA &VALUE
                                AEND
                                AIF (&VALUE GT &LVL_ACASE_LAST(&LVL))
                                    :&LVL_ACASE_LAST(&LVL) SETA &VALUE
                                AEND
                                :&INDEX SETA &VALUE+1
                                AIF (&(&VAL_BLK)(&INDEX) NE 0)
                                    :&MSG SETC 'DUPLICATE AWHEN VALUE &VALUE'
                                    ACALL ERR_MSG
                                AEND
                                :&(&VAL_BLK)(&INDEX) SETA &LVL_BCNT(&LVL) SET BLK # FOR VAL
                                AEND

```

ZSTRMAC.ZSM

.\*  
.\*  
.\*

```

AENTRY GET_VALUE
:&GET_VALUE_ERR SETB 0
:&VALUE_SET      SETB 0
AIF  ('&REC'(&IS_PARM,1) GE '0')
      :&VALUE      SETA  0
      :&VALUE_EOF SETB 0
      AWHILE (&IS_PARM LE K'&REC)
          AIF ('&REC'(&IS_PARM,1) GE '0'
X
              AND '&REC'(&IS_PARM,1) LE '9')
                  :&VALUE_SET SETB 1
                  :&DIGIT SETA '&REC'(&IS_PARM,1)
                  :&VALUE SETA &VALUE*10+&DIGIT
                  :&IS_PARM SETA &IS_PARM+1
              AELSE
                  AEXIT AWHILE
              AEND
          AEND
      AELSEIF (UPPER '&REC'(&IS_PARM,1) EQ 'C') RPI 911
          AIF (&IS_PARM+3 LE K'&REC)
              AIF ('&REC'(&IS_PARM+1,1) EQ ''')
X
                  AND '&REC'(&IS_PARM+3,1) EQ ''')
                      :&VALUE SETA  C2A('&REC'(&IS_PARM+2,1))
                      :&IS_PARM SETA &IS_PARM+4  SKIP C'?'
                      :&VALUE_SET SETB 1
                  AELSE
                      :&GET_VALUE_ERR SETB 1
                  AEND
              AELSE
                  :&GET_VALUE_ERR SETB 1
              AEND
          AELSEIF (UPPER '&REC'(&IS_PARM,1) EQ 'X') RPI 911
              AIF (&IS_PARM+4 LE K'&REC)
                  AIF ('&REC'(&IS_PARM+1,1) EQ ''')
X
                      AND '&REC'(&IS_PARM+4,1) EQ ''')
                          :&VALUE SETA  X2A('&REC'(&IS_PARM+2,2))
                          :&IS_PARM SETA &IS_PARM+5  SKIP X'??'
                          :&VALUE_SET SETB 1
                      AELSE
                          :&GET_VALUE_ERR SETB 1

```

ZSTRMAC.ZSM

```

        AEND
    AELSE
        :&GET_VALUE_ERR SETB 1
    AEND
AELSE
    :&GET_VALUE_ERR SETB 1
AEND
AIF  (&VALUE_SET)
    :&VALUE_CNT SETA &VALUE_CNT+1
    AIF  (&VALUE LT 0 OR &VALUE GT 255)  OUT OF RANGE
        :&GET_VALUE_ERR SETB 1
    AEND
AELSE
    :&GET_VALUE_ERR SETB 1
AEND
AEND
.*
.* AWHILE - GEN LABELD AIF TO END
.*
    AENTRY PROC_AWHILE
        :&AWHILE_TOT SETA &AWHILE_TOT+1  AWHILE COUNTER
        :&LVL      SETA &LVL+1          CURRENT LEVEL
        :&LVL_TYPE(&LVL) SETC 'AWHILE' CURRENT LEVEL TYPE
        :&LVL_TCNT(&LVL) SETA &AWHILE_TOT PRIMARY TYPE COUNTER
        :&PCH_REC SETC '.*'.'&REC'(3,*)
    ACALL PUNCH_REC
        :&PCH_REC SETC '.AWH_&LVL_TCNT(&LVL)_T'
    ACALL PUNCH_LAB
        :&GEN_AIF_TRUE SETB 0              GEN BRANCH IF FALSE
        :&GEN_AIF_TAG  SETC 'E'
    ACALL GEN_AIF
    AIF  (&GEN_AIF_ERR)
        :&MSG SETC 'AWHILE EXPRESSION ERROR'
        ACALL ERR_MSG
    AELSE
        ACALL PUNCH_REC
    AEND
AEND
.*
.* FIND_NAME OPERAND AND SET ACALL_INDEX TO EXISTING OR NEW ENTRY
.* SET FIND_NAME_ERR IF PARM ERROR
.*
    AENTRY FIND_NAME
        :&FIND_NAME_ERR SETB 0
    ACALL FIND_PARM

```

ZSTRMAC.ZSM

```

AIF  (&FIND_PARM_ERR)
      :&FIND_NAME_ERR SETB 1
AELSE
      :&NAME SETC (UPPER '&PARM')
      :&ACALL_INDEX SETA 1
      AWHILE (&ACALL_INDEX LE &ACALL_NAME_TOT)
          AIF  ('&ACALL_NAME(&ACALL_INDEX)' EQ '&NAME')
              AEXIT AENTRY  EXIT WITH ACALL_INDEX SET
          AEND
          :&ACALL_INDEX SETA &ACALL_INDEX+1
      AEND
      AIF  (&ACALL_INDEX GT &ACALL_NAME_TOT)
          :&ACALL_NAME_TOT SETA &ACALL_INDEX
          :&ACALL_NAME(&ACALL_INDEX) SETC '&NAME'
      AEND
AEND
AEND

```

```

.*
.* FIND_PARM OPERAND TERMINATED WITH SPACE
.* SET FIND_PARM_ERR IF ERROR
.*

```

```

AENTRY FIND_PARM
      :&PARM SETC ''
      :&FIND_PARM_ERR SETB 0
      :&IS_PARM SETA &IS_OP_END
      AWHILE (&IS_PARM LE K'&REC)
          AIF  ('&REC'(&IS_PARM,1) NE ' ')
              :&I SETA ('&REC'(&IS_PARM,*) INDEX ' ')
              AIF (&I GT 0 AND &IS_PARM+&I LE K'&REC)
                  :&PARM SETC '&REC'(&IS_PARM,&I-1)
              AELSE
                  :&PARM SETC '&REC'(&IS_PARM,*)
              AEND
          AEXIT AENTRY EXIT WITH PARM SET
      AEND
      :&IS_PARM SETA &IS_PARM+1
AEND
      :&FIND_PARM_ERR SETB 1
AEND

```

```

.*
.* PUNCH LABEL WITH ANOP ALIGNED WITH AOP IF POSSIBLE
.*

```

```

AENTRY PUNCH_LAB
      :&SPACES SETA &IS_OP+1-K'&PCH_REC
      AIF  (&SPACES LE 0)

```

ZSTRMAC.ZSM

:&SPACES SETA 1

AEND

:&PCH\_REC SETC '&PCH\_REC'.(&SPACES)' '.'ANOP'

ACALL PUNCH\_REC

AEND

.\*

.\* PUNCH &PCH\_REC WITH CONTINUATION FORMATTING AND RETURN TO CALLER

.\* BASED ON &PUNCH\_REC

.\*

AENTRY PUNCH\_REC

AIF (K'&PCH\_REC GE 72)

:&TEXT SETC (DOUBLE '&PCH\_REC'(1,71))

PUNCH '&TEXT.X',DDNAME=SYSUT2

:&I SETA 72

AWHILE (K'&PCH\_REC-&I GT 55)

:&TEXT SETC (DOUBLE '&PCH\_REC'(&I,56))

PUNCH ' &TEXT.X',DDNAME=SYSUT2

:&I SETA &I+56

AEND

AIF (&I LE K'&PCH\_REC)

:&TEXT SETC (DOUBLE '&PCH\_REC'(&I,\*))

PUNCH ' &TEXT',DDNAME=SYSUT2

AEND

AELSE

:&TEXT SETC (DOUBLE '&PCH\_REC')

PUNCH '&TEXT',DDNAME=SYSUT2

AEND

AEND

.\*

.\* GEN\_AIF - GENERATE AIF BRANCH

.\*

1. SET GEN\_AIF\_ERR TRUE/FALSE

.\*

2. BRANCH TRUE OR FALSE BASED ON GEN\_AIF\_TRUE

.\*

3. LABEL .&LVL\_TYPE(&LVL)\_&LVL\_TCNT(&LVL)\_&GEN\_AIF\_TAG

.\*

4. EXIT VIA COMPUTED AGO USING &GEN\_AIF

.\*

AENTRY GEN\_AIF

:&GEN\_AIF\_ERR SETB 0

ACALL FIND\_EXP

AIF (&FIND\_EXP\_ERR)

:&GEN\_AIF\_ERR SETB 1

AEXIT AENTRY

AEND

:&OP SETC (&IS\_OP+1)' '.'AIF'.(&IS\_EXP-&IS\_OP-3)' '

:&EXP SETC '&REC'(&IS\_EXP,&IS\_EXP\_END-&IS\_EXP+1)

:&LAB SETC

ZSTRMAC.ZSM

```
'.' '&LVL_TYPE(&LVL)'(1,3)['_&LVL_TCNT(&LVL)_&GEN_X
      AIF_TAG'
AIF   (NOT &GEN_AIF_TRUE)
      :&PCH_REC SETC  '&OP.(NOT&EXP)&LAB'
AELSE
      :&PCH_REC SETC  '&OP&EXP&LAB'
AEND
AIF   (&IS_EXP_END LT K'&REC)
      :&PCH_REC SETC  '&PCH_REC '.'&REC'(&IS_EXP_END+1,*)
```

COMS

```
AEND
AEND
```

```
.*
.* FIND EXP - FIND EXPRESSION (..) AND SET IS_EXP AND IS_EXP_END
.* SET FIND_EXP_ERR IF NOT FOUND
.*
```

```
AENTRY FIND_EXP
:&FIND_EXP_ERR SETB 0
:&IS_EXP SETA ('&REC' INDEX '(')
AIF   (&IS_EXP LE 0)
      :&FIND_EXP_ERR SETB 1
      AEXIT AENTRY
AEND
:&IS_EXP_END SETA &IS_EXP
:&I SETA ('&REC'(&IS_EXP_END+1,*) INDEX ')')
AWHILE (&I GT 0)
      :&IS_EXP_END SETA &IS_EXP_END+&I
      AIF (&IS_EXP_END LT K'&REC)
          :&I SETA ('&REC'(&IS_EXP_END+1,*) INDEX ')')
      AELSE
          :&I SETA 0
      AEND
AEND
AIF   (&IS_EXP_END EQ &IS_EXP)
      :&FIND_EXP_ERR SETB 1
AEND
AEND
```

```
.*
.* ERR_MSG ISSUE ERROR MESSAGE AND COUNT ERRORS
.*
```

```
AENTRY ERR_MSG
:&ERRORS SETA &ERRORS+1
MNOTE 8,'ZSTRMAC ERROR &MSG AT LINE &LINE'
PUNCH ' MNOTE 8''ZSTRMAC ERROR &MSG',DDNAME=SYSUT2
AEND
```

ZSTRMAC.ZSM

MEND  
ZSTRMAC  
END