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
*********************************************************************
* 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. APM NAME      > &APM_N SETA B
*                   >   AGO .APM_N
*                   > .APM_N_B ANOP
*                   >   ......
*  6. AENTRY NAME   > .APM_N ANOP
*                   >   ......
*  7. AEXIT         >   AGO .APM_N_E    (EXIT NON AIF STRUCURE)
*                   >   ......
*     AEND          >   .APM_N_E AGO (&APM_N).APM_N_1,.APM_N_2,
*                   >                                 .APM_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
*                   > .AUN_N_E ANOP
*                   >   ......
* 10. ASELECT (EXP) >   AGO .ASE_N_AGO
* 11. AWHEN V1,V2   > .ASE_N_B1 ANOP  VN=(N,C'?',X'??', OR (V1,V2)
*                   >   ......
*     AWHEN V2      >   AGO .ASE_N_E
*                   > .ASE_N_B2 ANOP
*                   >   ......
*     AELSE         >   AGO .ASE_N_E
*                   > .ASE_N_X ANOP
*                   >   ......
*     AEND          >   AGO .ASE_N_E
*                   > .ASE_N_G AGO (EXP).ASE_N_B1,.ASE_N_X,.ASE_N_B2
*                   >   AGO .ASE_N_X
*                   > .ASE_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,&APM_TOT
         LCLA  &ASELECT_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 APM/AENTRY NAME
         LCLB  &FIND_PARM_ERR SYNTAX ERROR FINDING FIRST PARM
         LCLB  &FIND_EXP_ERR  SYNTAX ERROR FINDING (..) FOR
AIF/ASELECT
         LCLB  &GET_VALUE_ERR ERROR PARSING DEC, '?', OR X'??'
         LCLA  &LVL           CURRENT LEVEL OF STRUCTURE
         LCLC  &LVL_TYPE(50) TYPE AIF/ASELECT/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
         LCLC  &LVL_ASELECT(50) ASELECT COMPUTED AGO STATEMENT
         LCLA  &LVL_ASELECT_FIRST(50) ASELECT FIRST WHEN VALUE 0-255
         LCLA  &LVL_ASELECT_LAST(50)  ASELECT LAST  WHEN VALUE 0-255
         LCLB  &LVL_AELSE(50)   AELSE BLOCK DEFINED FOR ASELECT
         LCLA  &IS_OP           START OF OPCODE
         LCLA  &IS_OP_END       ENDOF OF OPCODE+1
         LCLA  &IS_EXP          START OF AIF EXP (...)
         LCLA  &APM_INDEX       INDEX TO APM/AENTRY NAME VIA FIND_NAME
         LCLA  &APM_NAME_TOT    TOTAL PERFORMED ROUTINES
         LCLC  &APM_NAME(100)   NAMES OF PERFORMED ROUTINES
         LCLA  &APM_CNT(100)    EXIT COUNT FOR ROUTINES
         LCLB  &APM_DEF(100)    FLAG FOR DUP AND MISSING ERRORS
.*
.* READ SYUT1 AND OUTPUT SYSUT2 WITH STRUCTURED MACRO CODE
.*
         APM   READ_REC
         AWHILE (NOT &EOF)
               APM PROC_REC
               APM READ_REC
         AEND
         :&APM_INDEX SETA 1
         AWHILE (&APM_INDEX LE &APM_NAME_TOT)
               AIF (NOT &APM_DEF(&APM_INDEX))
                   :&MSG SETC 'MISSING AENTRY FOR
&APM_NAME(&APM_INDEX)X
               '
                   APM  ERR_MSG
               AEND
               :&APM_INDEX SETA &APM_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 APM       = &APM_TOT'
         MNOTE 'ZSTRMAC TOTAL ASELECT   = &ASELECT_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
         APM   READ_TEXT
         ACTR  10000
         AIF   (NOT &EOF)
               AIF (K'&TEXT GE 72)
                   :&REC SETC '&TEXT'(1,71)
                   AIF ('&TEXT'(72,1) NE ' ')
                       APM  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)
                            APM  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'
                                 APM  ERR_MSG
                            AEND
                       AELSE
                            :&MSG SETC 'END OF FILE ON CONTINUE'
                            APM  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
         AEND
         AEND
.*
.* PROCESS REC BY SCANNING FOR A??? OPCODES AND GENERATING
.* COMMENT AND GENERATED CODE ELSE COPY REC
.*
         AENTRY PROC_REC
         APM   FIND_OPCODE
         AIF   ('&OPCODE'(1,1) NE 'A')
               APM COPY_REC
         AELSEIF   ('&OPCODE' EQ 'AIF')
               APM PROC_AIF
         AELSEIF   ('&OPCODE' EQ 'AELSE')
               APM PROC_AELSE
         AELSEIF   ('&OPCODE' EQ 'AELSEIF')
               APM PROC_AELSEIF
         AELSEIF   ('&OPCODE' EQ 'AEND')
               APM PROC_AEND
         AELSEIF   ('&OPCODE' EQ 'APM')
               APM PROC_APM
         AELSEIF   ('&OPCODE' EQ 'AENTRY')
               APM PROC_AENTRY
         AELSEIF   ('&OPCODE' EQ 'AEXIT')
               APM PROC_AEXIT
         AELSEIF   ('&OPCODE' EQ 'AWHILE')
               APM PROC_AWHILE
         AELSEIF   ('&OPCODE' EQ 'AUNTIL')
               APM PROC_AUNTIL
         AELSEIF   ('&OPCODE' EQ 'ASELECT')
               APM PROC_ASELECT
         AELSEIF   ('&OPCODE' EQ 'AWHEN')
              APM  PROC_AWHEN
         AELSE
              APM  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)
              :&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 ':')
                    APM  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'
          APM  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,*)
         APM   PUNCH_REC
         AIF   (&LVL GE 1)
               AIF   (&LVL_TYPE(&LVL) EQ 'AIF')
                     APM PROC_AELSE_AIF
               AELSEIF   (&LVL_TYPE(&LVL) EQ 'ASELECT')
                     APM PROC_AELSE_ASELECT
               AELSE
                     :&MSG SETC 'INVALID AELSE TYPE &LVL_TYPE(&LVL)'
                     APM ERR_MSG
               AEND
         AELSE
               :&MSG SETC 'MISSING AIF OR ASELECT'
               APM 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'
         APM   PUNCH_REC
         :&PCH_REC SETC '.AIF_&LVL_TCNT(&LVL)_&LVL_BCNT(&LVL)'
         APM   PUNCH_LAB
         :&LVL_BCNT(&LVL) SETA 0  RESET TO INDICATE NO BLK LABEL REQ
         AEND
.*
.* AELSE_ASELECT
.*
         AENTRY PROC_AELSE_ASELECT
         AIF   (&LVL_BCNT(&LVL) GT 0)
               :&PCH_REC SETC (&IS_OP+1)' '.'AGO
.ASE_&LVL_TCNT(&LVL)X
               _E'
               APM  PUNCH_REC
         AEND
         :&LVL_AELSE(&LVL) SETB 1  INDICATE AELSE BLOCK DEFINED
         :&PCH_REC SETC '.ASE_&LVL_TCNT(&LVL)_X'
         APM  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,*)
         APM  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'
                   APM  PUNCH_REC
                   :&PCH_REC SETC
'.AIF_&LVL_TCNT(&LVL)_&LVL_BCNT(&LVL)X
               '
                   APM  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)'
                   APM  GEN_AIF
                   AIF   (&GEN_AIF_ERR)
                         :&MSG SETC 'AELSEIF AIF ERROR'
                         APM  ERR_MSG
                   AELSE
                         APM  PUNCH_REC
                   AEND
               AELSE
                   :&MSG SETC 'AELSEIF MISSING AIF ERROR'
                   APM  ERR_MSG
               AEND
         AELSE
               :&MSG SETC 'AELSEIF MISSING AIF ERROR'
               APM  ERR_MSG
         AEND
         AEND
.*
.* AEND - GEN TERMINATION FOR AENTRY,AIF,ASELECT,AUNTIL,AWHILE
.*
         AENTRY PROC_AEND
         :&AEND_TOT SETA &AEND_TOT+1
         :&PCH_REC SETC '.*'.'&REC'(3,*)
         APM  PUNCH_REC
         AIF   (&LVL GE 1)
               AIF   (&LVL_TYPE(&LVL) EQ 'AIF')
                     APM PROC_AEND_AIF
               AELSEIF   (&LVL_TYPE(&LVL) EQ 'AWHILE')
                     APM PROC_AEND_AWHILE
               AELSEIF   (&LVL_TYPE(&LVL) EQ 'ASELECT')
                     APM PROC_AEND_ASELECT
               AELSEIF   (&LVL_TYPE(&LVL) EQ 'AENTRY')
                     APM PROC_AEND_AENTRY
               AELSEIF   (&LVL_TYPE(&LVL) EQ 'AUNTIL')
                     APM PROC_AEND_AUNTIL
               AELSE
                    :&MSG SETC 'AEND INVALID TYPE &LVL_TYPE(&LVL)'
                    APM  ERR_MSG
               AEND
         AELSE
               :&MSG SETC 'AEND MISSING AIF OR OTHER STRUCTURE'
               APM  ERR_MSG
         AEND
         AEND
.*
.* AEND_AENTRY
.*
         AENTRY PROC_AEND_AENTRY
         :&APM_INDEX SETA &LVL_BCNT(&LVL)
         AIF   (&APM_CNT(&APM_INDEX) GT 0)
               AIF  (&LVL_TEND(&LVL))
                    :&PCH_REC SETC '.APM_&APM_INDEX._E'
                    APM PUNCH_LAB
               AEND
               :&PCH_REC SETC (&IS_OP+1)' '.'AGO
(&&APM_&APM_INDEX._&X
               APM_NAME(&APM_INDEX)).APM_&APM_INDEX._1'
               :&I SETA 2
               AWHILE (&I LE &APM_CNT(&APM_INDEX))
                  :&PCH_REC SETC '&PCH_REC,.APM_&APM_INDEX._&I'
                  :&I SETA &I+1
               AEND
               APM PUNCH_REC
         AELSE
               :&MSG SETC 'AENTRY &APM_NAME(&APM_INDEX) NOT USED'
               APM ERR_MSG
         AEND
         :&PCH_REC SETC '.APM_&APM_INDEX._SKIP'
         APM 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)'
               APM PUNCH_LAB
         AEND
         AIF   (&LVL_TEND(&LVL))
               :&PCH_REC SETC '.AIF_&LVL_TCNT(&LVL)_E'
               APM 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'
         APM  PUNCH_REC
         :&PCH_REC SETC '.AUN_&LVL_TCNT(&LVL)_E'
         APM  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'
         APM  PUNCH_REC
         :&PCH_REC SETC '.AWH_&LVL_TCNT(&LVL)_E'
         APM  PUNCH_LAB
         :&LVL     SETA  &LVL-1     CURRENT LEVEL
         AEND
.*
.* AEND_ASELECT
.*
         AENTRY PROC_AEND_ASELECT
         AIF   (&LVL_BCNT(&LVL) GT 0)
               :&PCH_REC SETC (&IS_OP+1)' '.'AGO
.ASE_&LVL_TCNT(&LVL)X
```

```
               _E'
               APM  PUNCH_REC
               :&PCH_REC SETC '.ASE_&LVL_TCNT(&LVL)_G'
               APM  PUNCH_LAB
               AIF  (&LVL_AELSE(&LVL))
                    :&ELSE_LAB SETC '.ASE_&LVL_TCNT(&LVL)_X'
               AELSE
                    :&ELSE_LAB SETC '.ASE_&LVL_TCNT(&LVL)_E'
               AEND
               :&PCH_REC SETC '&LVL_ASELECT(&LVL)'
               AIF  (&LVL_ASELECT_FIRST(&LVL) NE 1))
                    :&OFFSET  SETC '+1-&LVL_ASELECT_FIRST(&LVL)'
                    :&PCH_REC SETC
'&PCH_REC'(1,K'&PCH_REC-1).'&OFFSET)X
               '
               AEND
               :&VAL_BLK SETC  'ASELECT_&LVL_TCNT(&LVL)_VAL_BLK'
               :&VALUE SETA  &LVL_ASELECT_FIRST(&LVL)
               :&COMMA SETC  ''
               AWHILE (&VALUE LE &LVL_ASELECT_LAST(&LVL))
                      AIF   (&(&VAL_BLK)(&VALUE+1) GT 0)
                            :&PCH_REC SETC
'&PCH_REC&COMMA..ASE_&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
               APM  PUNCH_REC
               AIF   (&LVL_AELSE(&LVL))
                     :&PCH_REC SETC (&IS_OP+1)' '.'AGO
.ASE_&LVL_TCNTX
               (&LVL)_X'
                     APM  PUNCH_REC
               AEND
               :&PCH_REC SETC '.ASE_&LVL_TCNT(&LVL)_E'
               APM  PUNCH_LAB
               :&LVL     SETA  &LVL-1     CURRENT LEVEL
         AELSE
               :&MSG SETC 'NO WHEN FOUND FOR ASELECT'
               APM  ERR_MSG
         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,*)
         APM   PUNCH_REC
         APM   FIND_NAME
         AIF   (&FIND_NAME_ERR)
               :&MSG SETC 'AENTRY NAME NOT FOUND'
               APM  ERR_MSG
         AELSEIF (&APM_DEF(&APM_INDEX))
               :&MSG SETC 'AENTRY DUPLICATE NAME FOUND - &NAME'
               APM  ERR_MSG
         AELSE
               :&APM_DEF(&APM_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 &APM_INDEX    SAVE FOR AEND
               :&PCH_REC SETC (&IS_OP+1)' '.'AGO
.APM_&APM_INDEX._SKIX
               P'
               APM  PUNCH_REC
               :&PCH_REC SETC '.APM_&APM_INDEX._&APM_NAME(&APM_INDEX)'
               APM  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,*)
         APM  PUNCH_REC
         APM  FIND_PARM
         AIF  (&FIND_PARM_ERR)
              :&MSG SETC 'AEXIT TYPE PARM NOT FOUND'
              APM 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')
                    :&APM_INDEX SETA &LVL_BCNT(&EXIT_LVL)
                    :&PCH_REC SETC (&IS_OP+1)' '.'AGO
.APM_&APM_INDEXX
               ._E'
                    APM  PUNCH_REC
               AELSE
                    :&PCH_REC SETC (&IS_OP+1)' '.'AGO
.'.'&LVL_TYPE(&X
               EXIT_LVL)'(1,3).'_&LVL_TCNT(&EXIT_LVL)_E'
                    APM  PUNCH_REC
               AEND
         AELSE
               :&MSG SETC 'AEXIT NOT WITHIN AENTRY, AWHILE, ASELECT'
               APM  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)
         :&PCH_REC SETC '.*'.'&REC'(3,*)
         APM  PUNCH_REC
         :&GEN_AIF_TRUE SETB 0                 GEN BRANCH IF FALSE
         :&GEN_AIF_TAG  SETC '&LVL_BCNT(&LVL)'
         APM  GEN_AIF
         AIF   (&GEN_AIF_ERR)
               :&MSG SETC 'AIF EXPRESSION SYNTAX ERROR'
               APM  ERR_MSG
         AELSE
               APM  PUNCH_REC
         AEND
         AEND
.*
.* APM - GEN AGO TO PERFORMED ROUTINE
.*
         AENTRY PROC_APM
         :&APM_TOT SETA &APM_TOT+1
         :&PCH_REC SETC '.*'.'&REC'(3,*)
         APM   PUNCH_REC
         APM   FIND_NAME
         AIF   (&FIND_NAME_ERR)
               :&MSG SETC 'APM NAME SYNTAX ERROR'
               APM  ERR_MSG
         AELSE
               :&APM_CNT(&APM_INDEX) SETA &APM_CNT(&APM_INDEX)+1
               :&PCH_REC SETC
'&&APM_&APM_INDEX._&APM_NAME(&APM_INDEX)'
               :&SPACES SETA &IS_OP-K'&PCH_REC+1
               AIF (&SPACES LE 0)
                   :&SPACES SETA 1
               AEND
               :&PCH_REC SETC '&PCH_REC'.(&SPACES)' '.'SETA
&APM_CNTX
               (&APM_INDEX)'
               APM  PUNCH_REC
               :&PCH_REC SETC (&IS_OP+1)' '.'AGO
.APM_&APM_INDEX._&APX
               M_NAME(&APM_INDEX)'
               APM  PUNCH_REC
               :&PCH_REC SETC '.APM_&APM_INDEX._&APM_CNT(&APM_INDEX)'
               APM  PUNCH_LAB
         AEND
         AEND
.*
.* ASELECT - GEN AGO TO .ASELECT_N_AGO AND SAVE AGO EXPRESSION
.*
         AENTRY PROC_ASELECT
         :&ASELECT_TOT SETA  &ASELECT_TOT+1     ASELECT COUNTER
         :&LVL     SETA  &LVL+1     CURRENT LEVEL
         :&LVL_TYPE(&LVL) SETC 'ASELECT' CURRENT LEVEL TYPE
         :&LVL_TCNT(&LVL) SETA &ASELECT_TOT ASELECT INSTANCE
         :&LVL_BCNT(&LVL)  SETA 0 RESET ASELECT AWHEN BLOCKS
         :&LVL_AELSE(&LVL) SETB 0 ASSUME NO AELSE BLOCK
         :&VAL_BLK   SETC  'ASELECT_&LVL_TCNT(&LVL)_VAL_BLK'
         LCLA &(&VAL_BLK)(256)
         :&LVL_ASELECT_FIRST(&LVL) SETA 257
         :&LVL_ASELECT_LAST(&LVL)  SETA -1
         :&PCH_REC SETC '.*'.'&REC'(3,*)
         APM   PUNCH_REC
         APM   FIND_EXP
         AIF   (&FIND_EXP_ERR)
               :&MSG SETC 'ASELECT EXPRESSION ERROR'
               APM  ERR_MSG
         AELSE
               :&LVL_ASELECT(&LVL) SETC (&IS_OP+1)' '.'AGO
'.'&REC'(&X
               IS_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
.ASE_&LVL_TCNT(&LVL)X
               _G'
               APM  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
         :&LVL_TYPE(&LVL) SETC 'AUNTIL' CURRENT LEVEL TYPE
         :&LVL_TCNT(&LVL) SETA &AUNTIL_TOT PRIMARY TYPE COUNTER
         :&PCH_REC SETC '.*'.'&REC'(3,*)
         APM  PUNCH_REC
         :&PCH_REC SETC (&IS_OP+1)' '.'AGO   .AUN_&LVL_TCNT(&LVL)'
         APM  PUNCH_REC
         :&PCH_REC SETC '.AUN_&LVL_TCNT(&LVL)_T'
         APM  PUNCH_LAB
         :&GEN_AIF_TRUE SETB 1               GEN BRANCH IF TRUE
         :&GEN_AIF_TAG  SETC 'E'
         APM  GEN_AIF
         AIF   (&GEN_AIF_ERR)
               :&MSG SETC 'AUNTIL EXPRESSION ERROR'
               APM  ERR_MSG
         AELSE
               APM  PUNCH_REC
         AEND
         :&PCH_REC SETC '.AUN_&LVL_TCNT(&LVL)'
         APM  PUNCH_LAB
         AEND
.*
.* AWHEN - GEN .ASELECT_N_I LABEL FOR INDEX AND UPDATE INDEX VAL_BLK
.*
         AENTRY PROC_AWHEN
         :&PCH_REC SETC '.*'.'&REC'(3,*)
         APM  PUNCH_REC
         :&AWHEN_TOT SETA &AWHEN_TOT+1
         :&VAL_BLK   SETC  'ASELECT_&LVL_TCNT(&LVL)_VAL_BLK'
         AIF   (&LVL GE 1)
               AIF  (&LVL_TYPE(&LVL) EQ 'ASELECT')
                    AIF   (&LVL_BCNT(&LVL) GT 0 OR &LVL_AELSE(&LVL))
                          :&PCH_REC SETC (&IS_OP+1)' '.'AGO
.ASE_&LVLX
               _TCNT(&LVL)_E'
                          APM  PUNCH_REC
                    AEND
                    :&LVL_BCNT(&LVL) SETA &LVL_BCNT(&LVL)+1
                    APM  FIND_PARM
                    AIF   (&FIND_PARM_ERR)
                         :&MSG SETC 'AWHEN VALUE ERROR'
                         APM ERR_MSG
                    AELSE
                         APM PROC_AWHEN_VALUES
                    AEND
                    :&PCH_REC SETC
'.ASE_&LVL_TCNT(&LVL)_&LVL_BCNT(&LVLX
               )'
                    APM  PUNCH_LAB
               AELSE
                    :&MSG SETC 'AWHEN MISSING ASELECT'
                    APM ERR_MSG
               AEND
         AELSE
               :&MSG SETC 'AWHEN MISSING ASELECT'
```

```
             APM 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)
               ASELECT (C2A('&REC'(&IS_PARM,1)))
                    AWHEN C'('  SET RANGE (V1,V2)
                        :&IS_PARM SETA &IS_PARM+1
                        APM GET_VALUE
                        AIF  (&GET_VALUE_ERR)
                             :&MSG SETC 'INVALID RANGE VALUE'
                             APM  ERR_MSG
                             AEXIT AENTRY     EXIT AFTER VALUE ERROR
                        AEND
                        :&VALUE1 SETA &VALUE
                        AIF ('&REC'(&IS_PARM,1) NE ',')
                             :&MSG SETC 'MISSING RANGE ,'
                             APM ERR_MSG
                             AEXIT AENTRY
                        AEND
                        :&IS_PARM SETA &IS_PARM+1
                        APM GET_VALUE
                        AIF  (&GET_VALUE_ERR)
                             :&MSG SETC 'INVALID RANGE VALUE'
                             APM  ERR_MSG
                             AEXIT AENTRY  EXIT AFTER VALUE ERROR
                        AEND
                        :&VALUE2 SETA &VALUE
                        AIF ('&REC'(&IS_PARM,1) NE ')')
                            :&MSG SETC 'MISSING RANGE )'
                            APM ERR_MSG
                            AEXIT AENTRY
                        AEND
                        :&IS_PARM SETA &IS_PARM+1
                        :&VALUE SETA &VALUE1
                        AWHILE (&VALUE LE &VALUE2)
                            APM 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
                        APM  GET_VALUE
                        AIF  (&GET_VALUE_ERR)
                             :&MSG SETC 'INVALID VALUE'
                             APM  ERR_MSG
                             AEXIT AENTRY
                        AEND
                        APM SET_VAL_BLK
               AEND
         AEND
         AIF   (&VALUE_CNT EQ 0)
               :&MSG SETC 'NO AWHEN VALUES FOUND'
               APM ERR_MSG
         AEND
         AEND
.*
.* SET_VAL_BLK  AWHEN BLOCK NUMBER FOR VALUE
.*
         AENTRY SET_VAL_BLK
         AIF   (&VALUE LT &LVL_ASELECT_FIRST(&LVL))
               :&LVL_ASELECT_FIRST(&LVL) SETA &VALUE
         AEND
         AIF   (&VALUE GT &LVL_ASELECT_LAST(&LVL))
               :&LVL_ASELECT_LAST(&LVL) SETA &VALUE
         AEND
         :&INDEX   SETA  &VALUE+1
         AIF   (&(&VAL_BLK)(&INDEX) NE 0)
               :&MSG SETC 'DUPLICATE AWHEN VALUE &VALUE'
               APM ERR_MSG
         AEND
         :&(&VAL_BLK)(&INDEX) SETA &LVL_BCNT(&LVL) SET BLK # FOR VAL
         AEND
.*
.* GET_VALUE - DEC, C'?', OR X'??'
.*
         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 ('&REC'(&IS_PARM,1) EQ 'C')
               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 ('&REC'(&IS_PARM,1) EQ 'X')
               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
                      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,*)
        APM  PUNCH_REC
        :&PCH_REC SETC '.AWH_&LVL_TCNT(&LVL)_T'
        APM  PUNCH_LAB
        :&GEN_AIF_TRUE SETB 0                GEN BRANCH IF FALSE
        :&GEN_AIF_TAG  SETC 'E'
        APM  GEN_AIF
        AIF   (&GEN_AIF_ERR)
              :&MSG SETC 'AWHILE EXPRESSION ERROR'
              APM ERR_MSG
        AELSE
              APM  PUNCH_REC
        AEND
        AEND
.*
.* FIND_NAME OPERAND AND SET APM_INDEX TO EXISTING OR NEW ENTRY
.* SET FIND_NAME_ERR IF PARM ERROR
.*
        AENTRY FIND_NAME
        :&FIND_NAME_ERR SETB 0
        APM   FIND_PARM
        AIF   (&FIND_PARM_ERR)
              :&FIND_NAME_ERR SETB 1
        AELSE
              :&NAME SETC (UPPER '&PARM')
              :&APM_INDEX SETA 1
              AWHILE (&APM_INDEX LE &APM_NAME_TOT)
                   AIF  ('&APM_NAME(&APM_INDEX)' EQ '&NAME')
                        AEXIT AENTRY  EXIT WITH APM_INDEX SET
                   AEND
                   :&APM_INDEX SETA &APM_INDEX+1
              AEND
              AIF  (&APM_INDEX GT &APM_NAME_TOT)
                   :&APM_NAME_TOT SETA &APM_INDEX
                   :&APM_NAME(&APM_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)
              :&SPACES  SETA  1
        AEND
        :&PCH_REC SETC '&PCH_REC'.(&SPACES)' '.'ANOP'
        APM  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
         APM   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
'.'.'&LVL_TYPE(&LVL)'(1,3).'_&LVL_TCNT(&LVL)_&GEN_C
               AIF_TAG'
         AIF   (NOT &GEN_AIF_TRUE)
               :&PCH_REC SETC  '&OP.(NOT&EXP)&LAB'
         AELSE
               :&PCH_REC SETC  '&OP&EXP&LAB'
         AEND
.CHK_AIF_COM ANOP
         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
         MEND
         ZSTRMAC
         END
```