```
                                ZSTRMAC1.MLC
*****************************************************************
* 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                                             *
*****************************************************************
* 08/22/08 RPI 896 rt\test\ZSTRMAC1.MLC is the last non-structured
*          macro version of ZSTRMAC I ever expect to write.  It is
*          being used to bootstrap the conversion of the utility
*          linklib\ZSTRMAC.ZSM into linklib\ZSTRMAC.MLC using the
*          z390 ZSTRMAC extensions to elimiate all explicit AGO and
*          macro labels plus indent labeled statements starting with :
*****************************************************************
* 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_G
* 11. AWHEN V1       > .ASE_N_B1 ANOP   VN=(N,C'?', OR X'??')
*                    >   ......
*     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. THIS IS THE LAST NON-STRUCTURED Z390 MACRO CODE PROGRAM
*     I EVER PLAN TO WRITE!  <G> I SHOULD HAVE WRITTEN IT IN 1974
*     BEFORE I WROTE STRFORT TRANSLATOR FOR FORTRAN IN ASSEMBLER AND
*     FORTRAN, BUT THERE WERE NO AREAD AND PUNCH EXTENSIONS THEN AND
*     THERE WAS NO WAY TO INTEGRATE THE SUPPORT INTO THE MAINFRAME
*     MACRO ASSEMBLERS BEFORE Z390.  SEE ACM SIGPLAN FEB 1975.
*  2. ONCE THIS BOOTSTRAP VERSION IS WORKING, I'LL REWRITE ZSTRMAC.ZSM
*     USING STRUCTURED MACRO SUPPORT AND GENERATE FINAL EXECUTABLE
*     VERSION OF ZSTRMAC.MLC
*  3. TO RUN THE BOOTSTRAP VERSION USING HLASM, REMOVE THE DDNAME
*     EXTENDED PARMS ON AREAD AND PUNCH, PLACE SOURCE TO COVERT IN THE
*     SYSIN INPUT STREAM AFTER SOURCE PROGRAM FOR AREAD, AND CHANGE
*     LOGIC TO DETECT SPECIFIC END OF FILE RECORD SUCH AS "END".
*********************************************************************
         MACRO
         ZSTRMAC
         LCLA  &LINES         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
         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_PARM         START OF PARM
         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_TOT         TOTAL PERFORMED ROUTINES
         LCLC  &APM_NAME(100)   NAMES OF PERFORMED ROUTINES
         LCLA  &APM_CNT(100)    EXIT COUNT FOR ROUTINES
.*
.* READ LOGICAL RECORD INTO &REC WITH TRAILING COMMENTS IF ANY
.*
.READ_REC ANOP
&REC     AREAD DDNAME=SYSUT1
         ACTR  10000
         AIF   ('&REC' EQ '').EOF
&LINE    SETA  &LINE+1
         AIF   (K'&REC LT 72).PROC_REC
         AIF   ('&REC'(72,1) EQ ' ').PROC_REC
&REC     SETC  '&REC'(1,71)
.READ_CONT ANOP
&CONT    AREAD DDNAME=SYSUT1
         AIF   ('&CONT' EQ '').ERR1
&LINE    SETA  &LINE+1
         AIF   (K'&CONT LT 72).LAST_SHORT
         AIF   ('&CONT'(72,1) EQ ' ').LAST_LONG
&REC     SETC  '&REC'.'&CONT'(16,71-15)
         AGO   .READ_CONT
.LAST_SHORT ANOP
         AIF   (K'&CONT LT 16).ERR2
&REC     SETC  '&REC'.'&CONT'(16,K'&CONT-15)
         AGO   .PROC_REC
.LAST_LONG ANOP
&REC     SETC  '&REC'.'&CONT'(16,71-15)
.*
.* PROCESS REC BY SCANNING FOR A??? OPCODES AND GENERATING
.* COMMENT AND GENERATED CODE ELSE COPY REC
.*
.PROC_REC  ANOP
&IS_OP   SETA -1
&IS_OP_END SETA -1
&I       SETA ('&REC' INDEX ' ')
         AIF  (&I LE 0).COPY_REC
```

```
         AIF  ('&REC'(1,1) EQ '*').COPY_REC
         AIF  ('&REC'(1,2) EQ '.*').COPY_REC
.FIND_OP_START ANOP
&I       SETA &I+1
         AIF  (&I GT K'&REC).COPY_REC
         AIF  ('&REC'(&I,1) EQ ' ').FIND_OP_START
&IS_OP   SETA &I
.FIND_OP_END ANOP
&I       SETA &I+1
         AIF  (&I GT K'&REC).SET_OPCODE
         AIF  ('&REC'(&I,1) NE ' ').FIND_OP_END
.SET_OPCODE ANOP
&IS_OP_END SETA &I
&OPCODE  SETC (UPPER '&REC'(&IS_OP,&IS_OP_END-&IS_OP))
.*
.* CHECK OPCODE FOR A??? AND PROCESS ELSE COPY REC
.*
         AIF   ('&OPCODE' EQ 'AIF').AIF
         AIF   ('&OPCODE' EQ 'AELSE').AELSE
         AIF   ('&OPCODE' EQ 'AELSEIF').AELSEIF
         AIF   ('&OPCODE' EQ 'AEND').AEND
         AIF   ('&OPCODE' EQ 'APM').APM
         AIF   ('&OPCODE' EQ 'AENTRY').AENTRY
         AIF   ('&OPCODE' EQ 'AEXIT').AEXIT
         AIF   ('&OPCODE' EQ 'AWHILE').AWHILE
         AIF   ('&OPCODE' EQ 'AUNTIL').AUNTIL
         AIF   ('&OPCODE' EQ 'ASELECT').ASELECT
         AIF   ('&OPCODE' EQ 'AWHEN').AWHEN
         AGO   .COPY_REC
.*
.*  COPY UNKNOWN RECORDS WITHOUT CHANGE EXCEPT FOR
.*  MOVING LABEL FROM :LABEL TO LABEL FIELD
.*
.COPY_REC ANOP
          AIF  (&IS_OP LE 0).COPY_COLON_END
          AIF  (&IS_OP_END LE 0).COPY_COLON_END
          AIF  ('&REC'(&IS_OP,1) NE ':').COPY_COLON_END
&FIND_PARM SETA 3
          AGO  .FIND_PARM
.FIND_PARM_3 ANOP
         AIF   (&FIND_PARM_ERR).ERR18
&SPACES  SETA  &IS_OP-K'&OPCODE
         AIF   (&SPACES GT 0).SPACES_OK1
&SPACES  SETA  1
.SPACES_OK1 ANOP
```

```
&REC SETC '&REC'(&IS_OP+1,K'&OPCODE-1).(&SPACES)' '.'&REC'(&IS_PARM,*)
.COPY_COLON_END ANOP
&PCH_REC  SETC '&REC'
&PUNCH_REC SETA 1
         AGO   .PUNCH_REC
.PUNCH_REC_1 ANOP
         AGO   .READ_REC
.*
.* AELSE - GEN MACRO COMMENT AND GEN AGO TO AEND AND LABEL FOR ALT.
BLK
.*
.AELSE   ANOP
&PCH_REC SETC '.*'.'&REC'(3,*)
&PUNCH_REC SETA 6
         AGO   .PUNCH_REC
.PUNCH_REC_6 ANOP
         AIF   (&LVL LT 1).ERR7
         AIF   (&LVL_TYPE(&LVL) EQ 'AIF').AELSE_AIF
         AIF   (&LVL_TYPE(&LVL) EQ 'ASELECT').AELSE_ASELECT
         AGO   .ERR7
.AELSE_AIF ANOP
&LVL_TEND(&LVL) SETB 1  REQUEST AEND TO GEN END TARGET
&PCH_REC SETC (&IS_OP+1)' '.'AGO   .AIF_&LVL_TCNT(&LVL)_E'
&PUNCH_REC SETA 7
         AGO   .PUNCH_REC
.PUNCH_REC_7 ANOP
&PCH_REC SETC '.AIF_&LVL_TCNT(&LVL)_&LVL_BCNT(&LVL)'
&PUNCH_LAB SETA 1
         AGO   .PUNCH_LAB
.PUNCH_LAB_1 ANOP
&LVL_BCNT(&LVL) SETA 0  RESET TO INDICATE NO BLK LABEL PENDING
         AGO   .READ_REC
.AELSE_ASELECT ANOP
         AIF   (&LVL_BCNT(&LVL) EQ 0).AELSE_ASELECT_LAB
&PCH_REC SETC (&IS_OP+1)' '.'AGO   .ASE_&LVL_TCNT(&LVL)_E'
&PUNCH_REC SETA 29
         AGO   .PUNCH_REC
.PUNCH_REC_29 ANOP
.AELSE_ASELECT_LAB ANOP
&LVL_AELSE(&LVL) SETB 1  INDICATE AELSE BLOCK DEFINED FOR ASELECT
&PCH_REC SETC '.ASE_&LVL_TCNT(&LVL)_X'
&PUNCH_LAB SETA 14
         AGO   .PUNCH_LAB
.PUNCH_LAB_14 ANOP
         AGO   .READ_REC
```

```
.*
.* AELSEIF - GEN MACRO COMMENT AND GEN AIF TO END OF BLK,CUR BLK LAB
.*
.AELSEIF ANOP
         AIF   (&LVL LT 1).ERR8
         AIF   (&LVL_TYPE(&LVL) NE 'AIF').ERR8
&PCH_REC SETC '.*'.'&REC'(3,*)
&PUNCH_REC SETA 9
         AGO   .PUNCH_REC
.PUNCH_REC_9 ANOP
&LVL_TEND(&LVL) SETB 1 REQUEST AEND TO GEN END
&PCH_REC SETC (&IS_OP+1)' '.'AGO   .AIF_&LVL_TCNT(&LVL)_E'
&PUNCH_REC SETA 10
         AGO   .PUNCH_REC
.PUNCH_REC_10 ANOP
&PCH_REC SETC '.AIF_&LVL_TCNT(&LVL)_&LVL_BCNT(&LVL)'
&PUNCH_LAB SETA 2
         AGO   .PUNCH_LAB
.PUNCH_LAB_2 ANOP
&LVL_BCNT(&LVL) SETA &LVL_BCNT(&LVL)+1 NEW TARGET FOR END OF ELSE
&GEN_AIF_TRUE SETB 0                  GEN BRANCH IF FALSE
&GEN_AIF_TAG  SETC '&LVL_BCNT(&LVL)'
&GEN_AIF SETA 2
         AGO   .GEN_AIF
.GEN_AIF_2 ANOP
         AIF   (&GEN_AIF_ERR).ERR9
&PUNCH_REC SETA 12
         AGO   .PUNCH_REC
.PUNCH_REC_12 ANOP
         AGO   .READ_REC
.*
.* AEND - GEN TERMINATION FOR AENTRY,AIF,ASELECT,AUNTIL,AWHILE
.*
.AEND    ANOP
&PCH_REC SETC '.*'.'&REC'(3,*)
&PUNCH_REC SETA 4
         AGO   .PUNCH_REC
.PUNCH_REC_4 ANOP
         AIF   (&LVL LT 1).ERR5
         AIF   (&LVL_TYPE(&LVL) EQ 'AIF').AEND_AIF
         AIF   (&LVL_TYPE(&LVL) EQ 'AWHILE').AEND_AWHILE
         AIF   (&LVL_TYPE(&LVL) EQ 'ASELECT').AEND_ASELECT
         AIF   (&LVL_TYPE(&LVL) EQ 'AENTRY').AEND_AENTRY
         AIF   (&LVL_TYPE(&LVL) EQ 'AUNTIL').AEND_AUNTIL
         AGO   .ERR6
```

```
.AEND_AENTRY ANOP
&APM_INDEX SETA &LVL_BCNT(&LVL)
         AIF   (NOT &LVL_TEND(&LVL)).AEND_AENTRY_NO_END
&PCH_REC SETC  '.APM_&APM_INDEX._E'
&PUNCH_LAB SETA 17
         AGO   .PUNCH_LAB
.PUNCH_LAB_17 ANOP
.AEND_AENTRY_NO_END ANOP
         AIF   (&APM_CNT(&APM_INDEX) EQ 0).AEND_AENTRY_SKIP
&PCH_REC SETC  (&IS_OP+1)' '.'AGO
(&&APM_&APM_INDEX._&APM_NAME(&APM_IX
               NDEX)).APM_&APM_INDEX._1'
&I       SETA  1
.AEND_AENTRY_LOOP ANOP
&I       SETA  &I+1
         AIF   (&I GT &APM_CNT(&APM_INDEX)).AEND_AENTRY_AGO
&PCH_REC SETC  '&PCH_REC,.APM_&APM_INDEX._&I'
         AGO   .AEND_AENTRY_LOOP
.AEND_AENTRY_AGO ANOP
&PUNCH_REC SETA 23
         AGO   .PUNCH_REC
.PUNCH_REC_23 ANOP
.AEND_AENTRY_SKIP ANOP
&PCH_REC SETC  '.APM_&APM_INDEX._SKIP'
&PUNCH_LAB SETA 11
         AGO   .PUNCH_LAB
.PUNCH_LAB_11 ANOP
&LVL     SETA  &LVL-1     CURRENT LEVEL
         AGO   .READ_REC
.AEND_AIF ANOP
         AIF   (&LVL_BCNT(&LVL) EQ 0).AEND_SKIP_BLAB
&PCH_REC SETC  '.AIF_&LVL_TCNT(&LVL)_&LVL_BCNT(&LVL)'
&PUNCH_LAB SETA 3
         AGO   .PUNCH_LAB
.PUNCH_LAB_3 ANOP
.AEND_SKIP_BLAB ANOP
         AIF  (NOT &LVL_TEND(&LVL)).AEND_AIF_NO_END
&PCH_REC SETC '.AIF_&LVL_TCNT(&LVL)_E'
&PUNCH_LAB SETA 4
         AGO   .PUNCH_LAB
.PUNCH_LAB_4 ANOP
.AEND_AIF_NO_END ANOP
&LVL     SETA  &LVL-1     CURRENT LEVEL
         AGO   .READ_REC
.AEND_AUNTIL ANOP
```

```
&PCH_REC SETC (&IS_OP+1)' '.'AGO   .AUN_&LVL_TCNT(&LVL)_T'
&PUNCH_REC SETA 14
         AGO   .PUNCH_REC
.PUNCH_REC_14 ANOP
&PCH_REC SETC '.AUN_&LVL_TCNT(&LVL)_E'
&PUNCH_LAB SETA 5
         AGO   .PUNCH_LAB
.PUNCH_LAB_5 ANOP
&LVL     SETA  &LVL-1     CURRENT LEVEL
         AGO   .READ_REC
.AEND_AWHILE ANOP
&PCH_REC SETC (&IS_OP+1)' '.'AGO   .AWH_&LVL_TCNT(&LVL)_T'
&PUNCH_REC SETA 17
         AGO   .PUNCH_REC
.PUNCH_REC_17 ANOP
&PCH_REC SETC '.AWH_&LVL_TCNT(&LVL)_E'
&PUNCH_LAB SETA 9
         AGO   .PUNCH_LAB
.PUNCH_LAB_9 ANOP
&LVL     SETA  &LVL-1     CURRENT LEVEL
         AGO   .READ_REC
.AEND_ASELECT ANOP
         AIF   (&LVL_BCNT(&LVL) EQ 0).ERR17 NO WHEN DEFINED
&PCH_REC SETC (&IS_OP+1)' '.'AGO   .ASE_&LVL_TCNT(&LVL)_E'
&PUNCH_REC SETA 32
         AGO   .PUNCH_REC
.PUNCH_REC_32 ANOP
.AEND_ASELECT_LAB ANOP
&PCH_REC SETC '.ASE_&LVL_TCNT(&LVL)_G'
&PUNCH_LAB SETA 15
         AGO   .PUNCH_LAB
.PUNCH_LAB_15 ANOP
&ELSE_LAB SETC '.ASE_&LVL_TCNT(&LVL)_E'
         AIF  (NOT &LVL_AELSE(&LVL)).AEND_ASELECT_GEN_AGO
&ELSE_LAB SETC '.ASE_&LVL_TCNT(&LVL)_X'
.AEND_ASELECT_GEN_AGO ANOP
&PCH_REC SETC '&LVL_ASELECT(&LVL)'
         AIF  (&LVL_ASELECT_FIRST(&LVL) EQ 1)).AEND_ASELECT_OFFSET_END
&OFFSET  SETC '+1-&LVL_ASELECT_FIRST(&LVL)'
&PCH_REC SETC '&LVL_ASELECT(&LVL)'(1,K'&PCH_REC-1).'&OFFSET)'
.AEND_ASELECT_OFFSET_END ANOP
&VAL_BLK SETC  'ASELECT_&LVL_TCNT(&LVL)_VAL_BLK'
&VALUE   SETA  &LVL_ASELECT_FIRST(&LVL)-1
&COMMA   SETC  ''
.ASELECT_GEN_LOOP ANOP
```

```
&VALUE   SETA  &VALUE+1
         AIF   (&VALUE GT &LVL_ASELECT_LAST(&LVL)).ASELECT_GEN_AGO_END
&INDEX   SETA  &VALUE+1
         AIF   (&(&VAL_BLK)(&INDEX) GT 0).ASELECT_HIT
&PCH_REC SETC  '&PCH_REC&COMMA&ELSE_LAB'
&COMMA   SETC  ','
         AGO   .ASELECT_GEN_LOOP
.ASELECT_HIT ANOP
&PCH_REC SETC
'&PCH_REC&COMMA..ASE_&LVL_TCNT(&LVL)_&(&VAL_BLK)(&INDEX)X
               '
&COMMA   SETC  ','
         AGO   .ASELECT_GEN_LOOP
.ASELECT_GEN_AGO_END ANOP
&PUNCH_REC SETA 33
         AGO   .PUNCH_REC
.PUNCH_REC_33 ANOP
         AIF   (NOT &LVL_AELSE(&LVL)).AEND_ASELECT_END
&PCH_REC SETC (&IS_OP+1)' '.'AGO   .ASE_&LVL_TCNT(&LVL)_X'
&PUNCH_REC SETA 31
         AGO   .PUNCH_REC
.PUNCH_REC_31 ANOP
.AEND_ASELECT_END ANOP
&PCH_REC SETC '.ASE_&LVL_TCNT(&LVL)_E'
&PUNCH_LAB SETA 16
         AGO   .PUNCH_LAB
.PUNCH_LAB_16 ANOP
&LVL     SETA  &LVL-1     CURRENT LEVEL
         AGO   .READ_REC
.*
.* AENTRY - GEN AGO BRANCH AROUND PENTRY/PEND AND LABEL FOR ENTRY

.*
.AENTRY  ANOP
&PCH_REC SETC '.*'.'&REC'(3,*)
&PUNCH_REC SETA 22
         AGO   .PUNCH_REC
.PUNCH_REC_22 ANOP
         AIF  (&LVL NE 0).ERR19
&LVL     SETA &LVL+1
&LVL_TYPE(&LVL) SETC 'AENTRY'
&LVL_TEND(&LVL) SETB 0        RESET REQ FOR END LABEL FOR MULT BLKS
&FIND_NAME SETA 2
         AGO   .FIND_NAME
.FIND_NAME_2 ANOP
```

```
         AIF  (&FIND_NAME_ERR).ERR11
&LVL_BCNT(&LVL) SETA &APM_INDEX    SAVE FOR AEND
&PCH_REC SETC (&IS_OP+1)' '.'AGO   .APM_&APM_INDEX._SKIP'
&PUNCH_REC SETA 24
         AGO   .PUNCH_REC
.PUNCH_REC_24 ANOP
&PCH_REC SETC '.APM_&APM_INDEX._&APM_NAME(&APM_INDEX)'
&PUNCH_LAB SETA 12
         AGO   .PUNCH_LAB
.PUNCH_LAB_12 ANOP
         AGO   .READ_REC
.*
.* AEXIT - GEN AGO TO END AND REQUEST END LABEL

.*
.AEXIT   ANOP
&PCH_REC SETC '.*'.'&REC'(3,*)
&PUNCH_REC SETA 18
         AGO   .PUNCH_REC
.PUNCH_REC_18 ANOP
&EXIT_LVL SETA &LVL
.AEXIT_TEST ANOP
         AIF   (&EXIT_LVL LT 1).ERR10
&FIND_PARM SETA 4
         AGO   .FIND_PARM
.FIND_PARM_4 ANOP
         AIF   (&FIND_PARM_ERR).ERR20
         AIF   (&LVL_TYPE(&EXIT_LVL) EQ '&PARM').AEXIT_GEN
&EXIT_LVL SETA &EXIT_LVL-1
         AGO   .AEXIT_TEST
.AEXIT_GEN ANOP
         AIF   (&LVL_TYPE(&EXIT_LVL) EQ 'AENTRY').AEXIT_AENTRY
&PCH_REC SETC  (&IS_OP+1)' '.'AGO
.'.'&LVL_TYPE(&EXIT_LVL)'(1,3).'_&LX
               VL_TCNT(&EXIT_LVL)_E'
         AGO   .AEXIT_PCH
.AEXIT_AENTRY ANOP
&APM_INDEX SETA &LVL_BCNT(&EXIT_LVL) GET NAME INDEX SAVED BY AENTRY
&PCH_REC SETC  (&IS_OP+1)' '.'AGO   .APM_&APM_INDEX._E'
.AEXIT_PCH ANOP
&PUNCH_REC SETA 19
         AGO   .PUNCH_REC
.PUNCH_REC_19 ANOP
&LVL_TEND(&EXIT_LVL) SETB 1
         AGO   .READ_REC
```

```
.*
.* AIF - GEN MACRO COMMENT AND AIF TO GENERATED END LABEL AT NEXT
LEVEL
.*
.AIF     ANOP
&AIF_CNT SETA  &AIF_CNT+1     AIF COUNTER
&LVL     SETA  &LVL+1     CURRENT LEVEL
&LVL_TYPE(&LVL) SETC 'AIF' CURRENT LEVEL TYPE
&LVL_TCNT(&LVL) SETA &AIF_CNT PRIMARY TYPE COUNTER
&LVL_TEND(&LVL) SETB 0        RESET REQ FOR END LABEL FOR MULT BLKS
&LVL_BCNT(&LVL) SETA 1        BLOCK COUNTER (ELSE, ELSEIF, WHEN)
&PCH_REC SETC '.*'.'&REC'(3,*)
&PUNCH_REC SETA 2
         AGO   .PUNCH_REC
.PUNCH_REC_2 ANOP
&GEN_AIF_TRUE SETB 0                  GEN BRANCH IF FALSE
&GEN_AIF_TAG  SETC '&LVL_BCNT(&LVL)'
&GEN_AIF SETA  1
         AGO   .GEN_AIF             GEN AIF IN &PCH_REC
.GEN_AIF_1 ANOP
         AIF   (&GEN_AIF_ERR).ERR4
&PUNCH_REC SETA 3
         AGO  .PUNCH_REC           PUNCH GEN'D AIF
.PUNCH_REC_3 ANOP
         AGO  .READ_REC
.*
.* APM - GEN AGO TO PERFORMED ROUTINE
.*
.APM     ANOP
&PCH_REC SETC '.*'.'&REC'(3,*)
&PUNCH_REC SETA 20
         AGO   .PUNCH_REC
.PUNCH_REC_20 ANOP
&FIND_NAME SETA 1
         AGO  .FIND_NAME
.FIND_NAME_1 ANOP
         AIF  (&FIND_NAME_ERR).ERR11
&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 GE 1).SKIP_SPACES1
&SPACES       SETA 1
.SKIP_SPACES1 ANOP
&PCH_REC SETC '&PCH_REC'.(&SPACES)' '.'SETA    &APM_CNT(&APM_INDEX)'
&PUNCH_REC SETA 25
```

```
         AGO   .PUNCH_REC
.PUNCH_REC_25 ANOP
&PCH_REC SETC (&IS_OP+1)' '.'AGO
.APM_&APM_INDEX._&APM_NAME(&APM_INDEX
               X)'
&PUNCH_REC SETA 21
         AGO   .PUNCH_REC
.PUNCH_REC_21 ANOP
&PCH_REC SETC '.APM_&APM_INDEX._&APM_CNT(&APM_INDEX)'
&PUNCH_LAB SETA 10
         AGO   .PUNCH_LAB
.PUNCH_LAB_10 ANOP
         AGO   .READ_REC
.*
.* ASELECT - GEN AGO TO .ASELECT_N_AGO AND SAVE AGO EXPRESSION
.*
.ASELECT     ANOP
&ASELECT_CNT SETA  &ASELECT_CNT+1     ASELECT COUNTER
&LVL     SETA  &LVL+1     CURRENT LEVEL
&LVL_TYPE(&LVL) SETC 'ASELECT' CURRENT LEVEL TYPE
&LVL_TCNT(&LVL) SETA &ASELECT_CNT ASELECT INSTANCE
&LVL_BCNT(&LVL)  SETA 0 RESET  AWHEN BLK COUNTER
&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,*)
&PUNCH_REC SETA 26
         AGO   .PUNCH_REC
.PUNCH_REC_26 ANOP
&FIND_EXP SETA 1
         AGO  .FIND_EXP
.FIND_EXP_1 ANOP
         AIF  (&FIND_EXP_ERR).ERR12
&LVL_ASELECT(&LVL) SETC (&IS_OP+1)' '.'AGO
'.'&REC'(&IS_EXP,&IS_EXP_EX
               ND-&IS_EXP+1)
&INDEX   SETA 0
.ASELECT_INIT_INDEX ANOP
&INDEX   SETA &INDEX+1
         AIF  (&INDEX GT 256).ASELECT_INIT_END
&(&VAL_BLK)(&INDEX) SETA 0
         AGO  .ASELECT_INIT_INDEX
.ASELECT_INIT_END ANOP
```

```
&PCH_REC SETC (&IS_OP+1)' '.'AGO   .ASE_&LVL_TCNT(&LVL)_G'
&PUNCH_REC SETA 27
         AGO  .PUNCH_REC           PUNCH GEN'D AIF
.PUNCH_REC_27 ANOP
         AGO  .READ_REC
.*
.* AUNTIL - GEN AGO TO BLOCK, THEN LABEL TEST AIF TO EXIT
.*
.AUNTIL  ANOP
&AUNTIL_CNT SETA  &AUNTIL_CNT+1   AUNTIL COUNTER
&LVL     SETA  &LVL+1     CURRENT LEVEL
&LVL_TYPE(&LVL) SETC 'AUNTIL' CURRENT LEVEL TYPE
&LVL_TCNT(&LVL) SETA &AUNTIL_CNT PRIMARY TYPE COUNTER
&PCH_REC SETC '.*'.'&REC'(3,*)
&PUNCH_REC SETA 13
         AGO   .PUNCH_REC
.PUNCH_REC_13 ANOP
&PCH_REC SETC (&IS_OP+1)' '.'AGO   .AUN_&LVL_TCNT(&LVL)'
&PUNCH_REC SETA 5
         AGO  .PUNCH_REC           PUNCH GEN'D AGO TO BLOCK
.PUNCH_REC_5  ANOP
&PCH_REC SETC '.AUN_&LVL_TCNT(&LVL)_T'
&PUNCH_LAB SETA 7
         AGO   .PUNCH_LAB
.PUNCH_LAB_7 ANOP
&GEN_AIF_TRUE SETB 1               GEN BRANCH IF TRUE
&GEN_AIF_TAG  SETC 'E'
&GEN_AIF SETA  3
         AGO   .GEN_AIF            GEN AIF IN &PCH_REC
.GEN_AIF_3 ANOP
         AIF   (&GEN_AIF_ERR).ERR4
&PUNCH_REC SETA 11
         AGO  .PUNCH_REC           PUNCH GEN'D AIF
.PUNCH_REC_11 ANOP
&PCH_REC SETC '.AUN_&LVL_TCNT(&LVL)'
&PUNCH_LAB SETA 6
         AGO   .PUNCH_LAB
.PUNCH_LAB_6 ANOP
         AGO  .READ_REC
.*
.* AWHEN - GEN .ASELECT_N_I LABEL FOR INDEX AND UPDATE INDEX VAL_BLK
.*
.AWHEN   ANOP
         AIF   (&LVL LT 1).ERR7
         AIF   (&LVL_TYPE(&LVL) NE 'ASELECT').ERR13
```

```
&PCH_REC SETC '.*'.'&REC'(3,*)
&PUNCH_REC SETA 28
         AGO   .PUNCH_REC
.PUNCH_REC_28 ANOP
         AIF   (&LVL_BCNT(&LVL) EQ 0).AWHEN_LAB
&PCH_REC SETC (&IS_OP+1)' '.'AGO   .ASE_&LVL_TCNT(&LVL)_E'
&PUNCH_REC SETA 30
         AGO   .PUNCH_REC
.PUNCH_REC_30 ANOP
.AWHEN_LAB ANOP
&LVL_BCNT(&LVL)  SETA &LVL_BCNT(&LVL)+1
&FIND_PARM SETA 2
         AGO   .FIND_PARM
.FIND_PARM_2 ANOP
         AIF   (&FIND_PARM_ERR).ERR14
         AIF   ('&PARM'(1,1) GE '0').AWHEN_DEC
         AIF   ('&PARM'(1,1) EQ 'C').AWHEN_CHAR
         AIF   ('&PARM'(1,1) EQ 'X').AWHEN_HEX
         AGO   .ERR14
.AWHEN_DEC ANOP
&VALUE   SETA  &PARM
         AGO   .AWHEN_CHK_INDEX
.AWHEN_CHAR ANOP
         AIF   (K'&PARM GT 4 OR &IS_PARM+3 GT K'&REC).ERR14
&VALUE   SETA  C2A('&REC'(&IS_PARM+2,1))
         AGO   .AWHEN_CHK_INDEX
.AWHEN_HEX ANOP
         AIF   (K'&PARM GT 5 OR &IS_PARM+3 GT K'&REC).ERR14
&VALUE   SETA  (X2A('&REC'(&ISPAMR+2,K'&PARM-3)))
.AWHEN_CHK_INDEX ANOP
         AIF   (&VALUE LT 0 OR &VALUE GT 255).ERR16      OUT OF RANGE
         AIF   (&VALUE GE &LVL_ASELECT_FIRST(&LVL)).AWHEN_SKIP_FIRST
&LVL_ASELECT_FIRST(&LVL) SETA &VALUE
.AWHEN_SKIP_FIRST ANOP
         AIF   (&VALUE LE &LVL_ASELECT_LAST(&LVL)).AWHEN_SKIP_LAST
&LVL_ASELECT_LAST(&LVL) SETA &VALUE
.AWHEN_SKIP_LAST ANOP
&VAL_BLK   SETC  'ASELECT_&LVL_TCNT(&LVL)_VAL_BLK'
&INDEX   SETA  &VALUE+1
         AIF   (&(&VAL_BLK)(&INDEX) GT 0).ERR15 DUP
&(&VAL_BLK)(&INDEX) SETA &LVL_BCNT(&LVL) SET WHEN BLK # FOR VALUE
&PCH_REC SETC '.ASE_&LVL_TCNT(&LVL)_&LVL_BCNT(&LVL)'
&PUNCH_LAB SETA 13
         AGO   .PUNCH_LAB
.PUNCH_LAB_13 ANOP
```

```
         AGO   .READ_REC
.*
.* AWHILE - GEN LABELD AIF TO END
.*
.AWHILE  ANOP
&AWHILE_CNT SETA  &AWHILE_CNT+1   AWHILE COUNTER
&LVL     SETA  &LVL+1     CURRENT LEVEL
&LVL_TYPE(&LVL) SETC 'AWHILE' CURRENT LEVEL TYPE
&LVL_TCNT(&LVL) SETA &AWHILE_CNT PRIMARY TYPE COUNTER
&PCH_REC SETC '.*'.'&REC'(3,*)
&PUNCH_REC SETA 15
         AGO   .PUNCH_REC
.PUNCH_REC_15 ANOP
&PCH_REC SETC '.AWH_&LVL_TCNT(&LVL)_T'
&PUNCH_LAB SETA 8
         AGO   .PUNCH_LAB
.PUNCH_LAB_8 ANOP
&GEN_AIF_TRUE SETB 0                GEN BRANCH IF FALSE
&GEN_AIF_TAG  SETC 'E'
&GEN_AIF SETA  4
         AGO   .GEN_AIF             GEN AIF IN &PCH_REC
.GEN_AIF_4 ANOP
         AIF   (&GEN_AIF_ERR).ERR4
&PUNCH_REC SETA 16
         AGO  .PUNCH_REC           PUNCH GEN'D AIF
.PUNCH_REC_16 ANOP
         AGO  .READ_REC
.*
.* FIND_NAME OPERAND AND SET APM_INDEX TO EXISTING OR NEW ENTRY
.*
.FIND_NAME ANOP
&FIND_NAME_ERR SETB 0
&FIND_PARM SETA 1
         AGO  .FIND_PARM
.FIND_PARM_1 ANOP
         AIF  (&FIND_PARM_ERR).FIND_NAME_ERR
&NAME    SETC (UPPER '&PARM')
&APM_INDEX SETA 0
.FIND_NAME_INDEX ANOP
&APM_INDEX SETA &APM_INDEX+1
         AIF  (&APM_INDEX LE &APM_TOT).FIND_NAME_COMP
&APM_TOT SETA &APM_INDEX
&APM_NAME(&APM_INDEX) SETC '&NAME'
         AGO  .FIND_NAME_HIT
.FIND_NAME_COMP ANOP
```

```
         AIF  ('&APM_NAME(&APM_INDEX)' NE '&NAME').FIND_NAME_INDEX
.FIND_NAME_HIT ANOP
         AGO   (&FIND_NAME).FIND_NAME_1,.FIND_NAME_2
         AGO   .ERR3
.FIND_NAME_ERR ANOP
&FIND_NAME_ERR SETB 1
         AGO   .FIND_NAME_HIT
.*
.* FIND_PARM OPERAND TERMINATED WITH SPACE
.*
.FIND_PARM ANOP
&FIND_PARM_ERR SETB 0
&I       SETA &IS_OP_END-1
.FIND_PARM_START ANOP
&I       SETA &I+1
         AIF  (&I GT K'&REC).FIND_PARM_ERR
         AIF  ('&REC'(&I,1) EQ ' ').FIND_PARM_START
&IS_PARM SETA &I
.FIND_PARM_END ANOP
&I       SETA &I+1
         AIF  (&I GT K'&REC).FIND_PARM_SET
         AIF  ('&REC'(&I,1) NE ' ').FIND_PARM_END
.FIND_PARM_SET ANOP
&PARM    SETC  '&REC'(&IS_PARM,&I-&IS_PARM)
.FIND_PARM_AGO AGO
(&FIND_PARM).FIND_PARM_1,.FIND_PARM_2,.FIND_PARM_3,X
               .FIND_PARM_4
         AGO  .ERR3
.FIND_PARM_ERR ANOP
&FIND_PARM_ERR SETB 1
         AGO  .FIND_PARM_AGO
.*
.*
.* PUNCH LABEL WITH ANOP ALIGNED WITH AOP IF POSSIBLE
.*
.PUNCH_LAB ANOP
&SPACES  SETA  &IS_OP+1-K'&PCH_REC
         AIF   (&SPACES GT 0).SPACES_OK
&SPACES  SETA  1
.SPACES_OK ANOP
&PCH_REC SETC '&PCH_REC'.(&SPACES)' '.'ANOP'
&PUNCH_REC SETA 8
         AGO   .PUNCH_REC
.PUNCH_REC_8 ANOP
         AGO
```

```
(&PUNCH_LAB).PUNCH_LAB_1,.PUNCH_LAB_2,.PUNCH_LAB_3,.PUNCX

H_LAB_4,.PUNCH_LAB_5,.PUNCH_LAB_6,.PUNCH_LAB_7,.PUNCH_LAX

B_8,.PUNCH_LAB_9,.PUNCH_LAB_10,.PUNCH_LAB_11,.PUNCH_LAB_X

12,.PUNCH_LAB_13,.PUNCH_LAB_14,.PUNCH_LAB_15,.PUNCH_LAB_X
               16,.PUNCH_LAB_17
         AGO   .ERR3
.
.* PUNCH &PCH_REC WITH CONTINUATION FORMATTING AND RETURN TO CALLER
.* BASED ON &PUNCH_REC
.*
.PUNCH_REC ANOP
         AIF   (K'&PCH_REC GE 72).PUNCH_FIRST_CONT
&TEXT    SETC  (DOUBLE '&PCH_REC')
         PUNCH '&TEXT',DDNAME=SYSUT2
         AGO   .PUNCH_REC_AGO
.PUNCH_FIRST_CONT ANOP
&TEXT    SETC  (DOUBLE '&PCH_REC'(1,71))
         PUNCH '&TEXT.X',DDNAME=SYSUT2
&I       SETA  72
.PUNCH_NEXT_CONT ANOP
         AIF   (K'&PCH_REC-&I LE 55).PUNCH_LAST_CONT
&TEXT    SETC  (DOUBLE '&PCH_REC'(&I,56))
         PUNCH '               &TEXT.X',DDNAME=SYSUT2
&I       SETA  &I+56
         AGO   .PUNCH_NEXT_CONT
.PUNCH_LAST_CONT ANOP
&TEXT    SETC  (DOUBLE '&PCH_REC'(&I,*))
         PUNCH '               &TEXT',DDNAME=SYSUT2
.PUNCH_REC_AGO AGO
(&PUNCH_REC).PUNCH_REC_1,.PUNCH_REC_2,.PUNCH_REC_3,.X

PUNCH_REC_4,.PUNCH_REC_5,.PUNCH_REC_6,.PUNCH_REC_7,.PUNCX

H_REC_8,.PUNCH_REC_9,.PUNCH_REC_10,.PUNCH_REC_11,.PUNCH_X

REC_12,.PUNCH_REC_13,.PUNCH_REC_14,.PUNCH_REC_15,.PUNCH_X

REC_16,.PUNCH_REC_17,.PUNCH_REC_18,.PUNCH_REC_19,.PUNCH_X

REC_20,.PUNCH_REC_21,.PUNCH_REC_22,.PUNCH_REC_23,.PUNCH_X

REC_24,.PUNCH_REC_25,.PUNCH_REC_26,.PUNCH_REC_27,.PUNCH_X
```

```
REC_28,.PUNCH_REC_29,.PUNCH_REC_30,.PUNCH_REC_31,.PUNCH_X
               REC_32,.PUNCH_REC_33
         AGO   .ERR3
.*
.* 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
.GEN_AIF ANOP
&AIF_GEN_ERR SETB 0
&FIND_EXP SETA 2
         AGO   .FIND_EXP
.FIND_EXP_2 ANOP
         AIF   (&FIND_EXP_ERR).GEN_AIF_ERR
&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_AIF_TAX
               G'
         AIF   (&GEN_AIF_TRUE).GEN_AIF_TRUE
.GEN_AIF_FALSE ANOP
&PCH_REC SETC  '&OP.(NOT&EXP)&LAB'
         AGO   .CHK_AIF_COM
.GEN_AIF_TRUE ANOP
&PCH_REC SETC  '&OP&EXP&LAB'
.CHK_AIF_COM ANOP
         AIF   (K'&REC EQ &IS_EXP_END).GEN_AIF_AGO
&PCH_REC SETC  '&PCH_REC '.'&REC'(&IS_EXP_END+1,*)
.GEN_AIF_AGO AGO
(&GEN_AIF),.GEN_AIF_1,.GEN_AIF_2,.GEN_AIF_3,.GEN_AIF_4
         AGO   .ERR3
.GEN_AIF_ERR ANOP
&GEN_AIF_ERR SETB 1
         AGO   .GEN_AIF_AGO
.*
.* FIND EXP - FIND EXPRESSION (..) AND SET IS_EXP AND IS_EXP_END
.*            SET FIND_EXP_ERR IF NOT FOUND
.*
.FIND_EXP ANOP
&IS_EXP  SETA  ('&REC' INDEX '(')
         AIF   (&IS_EXP LE 0).FIND_EXP_ERR
&IS_EXP_END SETA &IS_EXP
.FIND_LAST ANOP
```

```
         AIF   (&IS_EXP_END GE K'&REC).FIND_LAST_END
&I       SETA  ('&REC'(&IS_EXP_END+1,*) INDEX ')')
         AIF   (&I LE 0).FIND_LAST_END
&IS_EXP_END SETA &IS_EXP_END+&I
         AGO   .FIND_LAST
.FIND_LAST_END ANOP
         AIF   (&IS_EXP_END EQ &IS_EXP).FIND_EXP_ERR
.FIND_EXP_AGO AGO (&FIND_EXP).FIND_EXP_1,.FIND_EXP_2
         AGO   .ERR3
.FIND_EXP_ERR ANOP
&FIND_EXP_ERR SETB 1
         AGO  .FIND_EXP_AGO
.EOF     ANOP
         MNOTE 'ZSTRMAC CONVERTED &LINE LINES WITHOUT ERRORS'
         AGO   .EXIT
.ERR1    MNOTE 8,'ZSTRMAC ERROR 1 EOF ON CONTINUATION AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 1 EOF ON CONTINUATION AT LINE &LINE'
         AGO   .EXIT
.ERR2    MNOTE 8,'ZSTRMAC ERROR 2 CONTINUATION TOO SHORT AT LINE
&LINE'
         PUNCH  '*ZSTRMAC ERROR 2 CONTINUATION TOO SHORT AT LINE
&LINE'
         AGO   .EXIT
.ERR3    MNOTE 8,'ZSTRMAC ERROR 3 INVALID AGO INDEX AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 3 INVALID AGO INDEX AT LINE &LINE'
         AGO   .EXIT
.ERR4    MNOTE 8,'ZSTRMAC ERROR 4 AIF SYNTAX ERROR AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 4 AIF SYNTAX ERROR AT LINE &LINE'
         AGO   .EXIT
.ERR5    MNOTE 8,'ZSTRMAC ERROR 5 AEND MISSING AIF ETC. AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 5 AEND MISSING AIF ETC. AT LINE &LINE'
         AGO   .EXIT
.ERR6    MNOTE 8,'ZSTRMAC ERROR 6 AEND UNDEFINED TYPE AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 6 AEND UNDEFINED TYPE AT LINE &LINE'
         AGO   .EXIT
.ERR7    MNOTE 8,'ZSTRMAC ERROR 7 AELSE MISSING AIF AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 7 AELSE MISSING AIF AT LINE &LINE'
         AGO   .EXIT
.ERR8    MNOTE 8,'ZSTRMAC ERROR 8 AELSEIF MISSING AIF AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 8 AELSEIF MISSING AIF AT LINE &LINE'
         AGO   .EXIT
.ERR9    MNOTE 8,'ZSTRMAC ERROR 7 ELSEIF SYNTAX ERROR AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 7 ELSEIF SYNTAX ERROR AT LINE &LINE'
         AGO   .EXIT
.ERR10   MNOTE 8,'ZSTRMAC ERROR 10 AEXIT MISSING PREV OP AT LINE
```

```
&LINE'
         PUNCH  '*ZSTRMAC ERROR 10 AEXIT MISSING PREV OP AT LINE
&LINE'
         AGO   .EXIT
.ERR11   MNOTE 8,'ZSTRMAC ERROR 11 APM NAME NOT FOUND AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 11 APM NAME NOT FOUND AT LINE &LINE'
         AGO   .EXIT
.ERR12   MNOTE 8,'ZSTRMAC ERROR 12 ASELECT EXP ERROR AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 12 ASEKECT EXP ERROR AT LINE &LINE'
         AGO   .EXIT
.ERR13   MNOTE 8,'ZSTRMAC ERROR 13 AWHEN W/O ASELECT AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 13 AWHEN W/O ASELECT AT LINE &LINE'
         AGO   .EXIT
.ERR14   MNOTE 8,'ZSTRMAC ERROR 14 AWHEN VALUE ERROR AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 14 AWHEN VALUE ERROR AT LINE &LINE'
         AGO   .EXIT
.ERR15   MNOTE 8,'ZSTRMAC ERROR 15 AWHEN DUP VALUE AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 15 AWHEN DUP VALUE AT LINE &LINE'
         AGO   .EXIT
.ERR16   MNOTE 8,'ZSTRMAC ERROR 16 AWHEN RANGE ERROR AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 16 AWHEN RANGE ERROR AT LINE &LINE'
         AGO   .EXIT
.ERR17   MNOTE 8,'ZSTRMAC ERROR 17 ASELECT NO AWHEN AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 17 ASELECT NO AWHEN AT LINE &LINE'
         AGO   .EXIT
.ERR18   MNOTE 8,'ZSTRMAC ERROR 18 COPY COLON ERROR AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 18 COPY COLON ERROR AT LINE &LINE'
         AGO   .EXIT
.ERR19   MNOTE 8,'ZSTRMAC ERROR 19 AENTRY LVL ERROR AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 19 AENTRY LVL ERROR AT LINE &LINE'
         AGO   .EXIT
.ERR20   MNOTE 8,'ZSTRMAC ERROR 20 AEXIT TYPE ERROR AT LINE &LINE'
         PUNCH  '*ZSTRMAC ERROR 20 AEXIT TYPE ERROR AT LINE &LINE'
         AGO   .EXIT
.EXIT    MEND
         ZSTRMAC
         END
```