```
.*********************************************************************
.* Copyright 2009 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   - 06/05/09                                                 *
.*********************************************************************
.* 09/29/09 RPI 1086 UPDATE TO SUPPORT CBL AND EXEC CICS SOURCES
.* 10/10/09 RPI 1089 FIXES AND ENHANCEMENTS:
.*            1.  ALLOW TRE WITHOUT PATH BY ADDING SYSDAT(.) SYSPCH(.)
.*            2.  DISPLAY ALL ERROR MESSAGES ON TRS WITH ERR= PREFIX
.*                AND DIDSPLAY TOTAL ERRORS AT END.
.*            3.  FORCE ALL MODULE NAMES TO UPPERCASE FOR COMPARES.
.*            4.  ELIMINATE MAIN PARAMETER AND EXTRACT MAIN PROGRAM
.*                NAME FROM FIRST TRE TRACE LOAD.
.*            5.  DISPLAY TOTAL CBL, ECC, ASM, AND ECA SOURCE LINES
.*                LOADED FROM EACH INCLUDED PRN MODULE FILE AND ALSO
.*                SHOW TOTAL LISTED AT END OF REPORT.
.*            6.  DISPLAY TOTAL SKIPPED ASM LINES AND MESSAGES FROM
.*                TRE TRACE FILE.
.*            7.  ISSUE ERROR IF NO ASM LINES LOADED FROM INCLUDED
.*                PRN MODULE FILE.
.*            8.  DISPLAY TOTAL LST AND PRN FILES PROCESSED.
.*            9.  ISSUE ERROR AT END IF INCLUDED MODULE NOT FOUND
.*           10.  RESET LST AND PRN FILE AREAD TO FORCE REREAD ON
.*                MODULES LOADED MORE THAN ONCE.
.*           11.  ADD NEW LOADLIB(PATH) FOR ALL LST AND PRN FILES ELSE
.*                USE TRE TRACE PATH FOR LST AND LST INCLUDE PRN PATH.
.*           12.  ISSUE ERROR AND ABORT IF LST/PRN NOT FOUND
.*                VIA LOADLIB OR TRE LOAD PATH OR LST INCLUDE PATH
.*********************************************************************
.* OPTIONS SYSPARM(%1+%?+%9)
.*   1 - TRACE TRE FILE WITH OPTIONAL PATH MUST BE FIRST PARM
.*   2 - ALL/NOALL       - INCLUDE ALL MLC AND CBL SOURCE          OFF
.*   3 - ASM/NOASM       - INCLUDE MLC ASSEMBLER SOURCE LINES      ON
.*   4 - CBL/NOCBL       - INCLUDE COBOL SOURCE AND EXEC CICS      ON
.*   5 - DETAIL/NODETAIL - INCLUDE BOTH TRE AND MLC SOURCE ALINES  OFF
.*   6 - EXCLUDE(NAME1+NAME2+NAMEN) - EXCLUDE PRN NAMES
.*   7 - INCLUDE(NAME1+NAME2+NAMEN) - INCLUDE PRN NAMES
.*       EXCLUDE AND INCLUDE ARE MUTUALLY EXCLUSIVE SO ONLY USE ONE
.*   8 - LOADLIB(PATH) - USE LOADLIB PATH FOR ALL LST/PRN FILES
.*       ELSE USE LST PATH FROM TRE TRACE LOAD
.*       AND PRN PATHS FROM LST INCLUDES
.*   9 - MSG/NOMSG       - INCLUDE TRACE, WTO, DUMP, AND TRE MESSAGES
.*  10 - TIME/NOTIME     - INCLUDE TIME-STAMP IF FOUND             OFF
.*  OUTPUT %1.TRS  - ZPAR TRACE EXECUTION OF CBL/MLC/ECC/ECA SOURCE
.* STEPS:
.*  1.  READ %1.TRE AND SIMULATE LOAD OF MAIN.390 AT FIRST INSTR. ADDR.
.*  2.  FOR EACH LOAD SVC PERFORM THE FOLLOWING:
.*      A.  SKIP LOADING IF NOT ON INCLUDE LIST OR ON EXCLUDE LIST
.*      B.  READ LOAD_NAME.LST TO GET INCLUDED MODULE NAMES
.*      C.  READ EACH INCLUDE MODULE.PRN AND STORE SOURCE LINES
.*          BY RESOLVED PSW_ADDR KEY.  INCLUDE INSTRUCTION LABEL FROM
.*          PRIOR DS 0H OR EQU * IF FOUND.
.*  3.  IF INDEX FOUND
.*         IF NEW COBOL MLC REF LINE
.*            WRITE COBOL MLC LINE WITH COBOL LINE ID
.*         ENDIF
.*         IF DETAIL
.*            WRITE DETAIL TRE LINE WITH INSTRUCTION OPERAND DATA
.*         ENDIF
.*         IF ASM
.*            WRITE GENERATED ASM SOURCE LINE WITH OPERAND LABELS
.*         END
.*      ELSE IF NOT INSTR AND (MSG)
.*         WRITE MSG LINE SUCH AS WTO, TRACE, OR ERROR
.*      ENDIF
.*  4.  REPEAT TO END OF TRE
.* NOTES:
.*   1.  ZPARTRA.BAT SETS NOCBL (OMITS CBL_KEY FROM ASM LINES)
.*   2.  ZPARTRC.BAT SETS NOASM (OMITS ASM LINE FOLLOWING CBL_KEY)
.*   3.  ZPARTRS.BAT DEFAULT CBL AND ASM FOR MIXED TRACING
.*   4.  IF INC/EXC OMITTED DEFAULT IS INCLUDE(MAIN/TRE)
.*   5.  INC OVERRIDES EXC
.*   6.  INC/EXC CAN HAVE ENDING * FOR WILDCARD NAMES
.*   7.  ALL OVERRIDES NOASM/NOCBL
.*   8.  DETAIL WILL FORCE COBOL INIT SOURCE ASM TRACE FOR CBL MODULES
.*********************************************************************
```

```
.*
.* INIT VAR
.*
         :&VERSION SETC 'V1.5.01b 10/28/09'
         :&MAX_RC  SETA 0      MAXIMUM RETURN CODE FOR ANY ERRORS
         :&TRE_PATH SETC ''    INPUT  TRE FILE PATH
         :&CDE_PATH SETC ''    INPUT  CDE LOAD PATH FROM TRE LOAD REC
         :&LOADLIB  SETC ''    PATH FOR ALL LST/PRN ELSE USE CDE PATHS
         :&MAIN_NAME SETC ''   NAME OF MAIN MODULE FROM TRE LOAD REC
         :&TRE_DSN SETC ''     INPUT  TRE FILE VIA AREAD ID=1
         :&LST_DSN SETC ''     INPUT  LST FILE VIA AREAD ID=2
         :&PRN_DSN SETC ''     PNPUT  PRN FILE VIA AREAD ID=3
         :&TRS_DSN SETC ''     OUTPUT TRS FILE VIA PUNCH ID=0 DEFAULT
         :&TRS_LINE SETC ''    OUTPUT LINE
         :&TOT_TRE SETA 0      INPUT  TRE RECORDS
         :&TOT_LST_REC SETA 0  INPUT  LST RECORDS FOR EACH 390 MODULE
         :&TOT_PRN_REC SETA 0  INPUT  PRN RECORDS FOR EACH OBJ MODULE
         :&TOT_LST_FIL SETA 0  INPUT  LST FILES FOR EACH 390 MODULE
         :&TOT_PRN_INC SETA 0  INPUT  PRN FILES INCLUDED
         :&TOT_PRN_EXC SETA 0  INPUT  PRN FILES EXCLUDED
         :&TOT_CBL_KEY SETA 0  INPUT  CBL LINE KEYS (C + PSW FIRST INS)
         :&TOT_ECC_KEY SETA 0  INPUT  ECC LINE KEYS (E + PSW FIRST INS)
         :&TOT_ASM_KEY SETA 0  INPUT  ASM LINE KEYS (A + PSW EACH  INS)
         :&TOT_ECA_KEY SETA 0  INPUT  ECA LINE KEYS (X + PSW FIRST INS)
         :&TOT_SKIP_ASM SETA 0  SKIPPED ASM DUE TO NOASM OR INC/EXC
         :&TOT_SKIP_MSG SETA 0  SKIPPED MSG DUE TO NOMSG OR INC/EXC
         :&TOT_TRS SETA 0      OUTPUT TRS RECORDS
         :&TOT_CBL SETA 0      OUTPUT CBL SOURCE LINES
         :&TOT_ECC SETA 0      OUTPUT EXEC CICS CBL SOURCE LINES
         :&TOT_ASM SETA 0      OUTPUT ASM SOURCE LINES
         :&TOT_ECA SETA 0      OUTPUT EXEC CICS ASN SOURCE LINES
         :&TOT_MSG SETA 0      OUTPUT MSG LINES (WTO,TRACE,ERRORS,ETC)
         :&TOT_ERR SETA 0      OUTPUT ERR LINES (MNOTE ERROR MSGS)
         :&LST_INC SETA 0      INCLUDED MODULE PRN FILES
         :&PRN_CBL SETA 0      LOADED CBL SOURCE LINES
         :&PRN_ECC SETA 0      LOADED EXEC CICS CBL SOURCE LINES
         :&PRN_ASM SETA 0      LOADED ASM SOURCE LINES
         :&PRN_ECA SETA 0      LOADED EXEC CICS ASN SOURCE LINES
         :&NP SETA 0
         GBLC &PARM(9)          PARMS FROM SYSPARM
         :&NINC SETA 0
         LCLC &INC_NAME(10)    INCLUDE MODULE NAME
         LCLA &INC_CNT(10)     INCLUDE MODULE COUNT (ERR IF NOT FOUND)
         :&NEXC SETA 0
         LCLC &EXC_NAME(10)    EXCLUDE MODULES BY PRN NAME
         :&TIME SETC ''        ASSUME NO TIME-STAMP
         :&OPT_ALL    SETB 0   ASSUME TRACE ALL TRE CODE VS INC/EXC
         :&OPT_ASM    SETB 1   ASSUME ASSEMBLER SOURCE LISTING INCLUDED
         :&OPT_CBL    SETB 1   ASSUME INCLUDE COBOL SOURCE
         :&OPT_DETAIL SETB 0   ASSUME NO 2ND LINE WITH OPERAND DETAILS
         :&OPT_INC    SETB 0   ASSUME INCLUDE MODULE, SET BY CHECK_NAME
         :&OPT_MSG    SETB 1   ASSUME INCLUDE ALL TRACE, WTO, DUMP MSGS
         :&OPT_TIME   SETB 1   ASSUME INCLUDE TIME-STAMP IF FOUND
         :&PGM_TYPE   SETC 'A' ASSUME ASM VS 'C' FOR ZCOBOL OR CICS E/X
         LCLC &INC_MOD(10)
         :&INC_TOT    SETA 0
         LCLC &EXC_MOD(10)
         :&EXC_TOT    SETA 0
         :&SKIP_INS_CNT   SETA 0
         :&SKIP_MSG_CNT   SETA 0
         :&SKIP_BRK_CNT   SETA 0
         :&CBL_KEY        SETC ''  KEY ASSIGNED AT NEXT ASM DURING LOAD
         :&CBL_LINE       SETC ''
         :&LAST_CBL_KEY   SETC ''  KEY FOR PREV ASM STMT IN TRACE
         :&LAST_CBL_LINE  SETC ''
         :&LAST_BRK       SETB 0
         :&EZ390_FOUND    SETB 0
.*
.* MAIN
.*
         ACALL INIT
         ACALL GET_TRE
         AIF ('&REC' EQ '')
             :&ERR_LVL SETA 16
             :&ERR_MSG SETC 'TRE TRACE FILE NOT FOUND - &TRE_DSN'
             ACALL PUT_ERR
             ACALL TERM
         AEND
```

```
AWHILE ('&REC' NE '')
      ACTR 4096
      :&MSG SETB 1              ASSUME REC IS MSG VS INSTR
      AIF  (K'&REC GT 25)
           AIF  ('&REC'(2,9) EQ 'CDE LOAD=')
               :&TRS_LINE SETC 'MSG=&REC'
               ACALL PUT_MSG
               :&MSG SETB 0
               :&LOAD_ADDR SETC '&REC'(12,7) IGNORE AMODE
               :&LOAD_LEN  SETC '&REC'(24,8)
               :&LST_DSN   SETC '&REC'(38,*)
               :&LST_DSN   SETC '&LST_DSN'(1,K'&LST_DSN-4).'.LX
      ST'
               ACALL LOAD_LST
           AELSEIF ('&REC'(2,1) EQ '0' OR '&REC'(2,1) EQ '8')
             AIF ('&REC'(11,1) NE '*')  EXCLUDE DUMP LINES
                 :&MSG SETB 0           REC IS INSTRUCTION
                 ACALL PROCESS_ASM_TRE
             AEND
           AEND
      AEND
      AIF  (&MSG)
           AIF ('&REC' EQ ' ')
              :&SKIP_BRK_CNT SETA &SKIP_BRK_CNT+1
           AELSEIF (&OPT_MSG)
               :&TRS_LINE SETC 'MSG=&REC'
               ACALL PUT_MSG
           AELSE
              :&TOT_SKIP_MSG SETA &TOT_SKIP_MSG+1
              :&SKIP_MSG_CNT SETA &SKIP_MSG_CNT+1
           AEND
      AEND
      ACALL GET_TRE
AEND
:&I SETA 1
AWHILE (&I LE &NINC)
      AIF (&INC_CNT(&I) EQ 0)
          :&ERR_LVL SETA 8
          :&ERR_MSG SETC 'INCLUDE MODULE NOT FOUND - &INC_NAMEX
```

```
            (&I)'
                ACALL PUT_ERR
            AEND
            :&I SETA &I+1
      AEND
      :&TIME SETC ''
      :&TRS_LINE SETC 'TRS= INPUT   FILES   LST=&TOT_LST_FIL PRN INCX
            =&TOT_PRN_INC PRN EXC=&TOT_PRN_EXC'
      ACALL PUT_TRS_LINE
      :&TRS_LINE SETC 'TRS= INPUT   RECORDS TRE=&TOT_TRE LST=&TOT_LSX
            T_REC PRN=&TOT_PRN_REC'
      ACALL PUT_TRS_LINE
      :&TRS_LINE SETC 'TRS= LOADED  SOURCE  CBL=&TOT_CBL_KEY ECC=&TOX
            T_ECC_KEY ASM=&TOT_ASM_KEY ECA=&TOT_ECA_KEY'
      ACALL PUT_TRS_LINE
      :&TRS_LINE SETC 'TRS= SKIPPED RECORDS ASM=&TOT_SKIP_ASM MSG=&TX
            OT_SKIP_MSG'
      ACALL PUT_TRS_LINE
      :&TRS_LINE SETC 'TRS= OUTPUT  RECORDS CBL=&TOT_CBL ECC=&TOT_ECX
            C ASM=&TOT_ASM ECA=&TOT_ECA'
      ACALL PUT_TRS_LINE
      :&TOT_TRS SETA &TOT_TRS+1  ADD LAST LINE TO DISPLAY COUNT
      :&TRS_LINE SETC 'TRS= OUTPUT  RECORDS ERR=&TOT_ERR MSG=&TOT_MSX
            G TRS=&TOT_TRS'
      ACALL PUT_TRS_LINE
      ACALL TERM
.*
.* PROCESS ASM TRACE LINE
.*
      AENTRY PROCESS_ASM_TRE
      :&ASM_KEY SETC 'A'.'&REC'(3,7)
      LCLC &(&ASM_KEY)
      :&ASM_LINE SETC '&(&ASM_KEY)'
      AIF ('&ASM_LINE' NE '')
         AIF (&OPT_CBL)
           AIF (NOT &OPT_DETAIL AND NOT &OPT_ALL                    X
              AND '&ASM_LINE'(1,1) EQ 'C'                           X
              AND '&ASM_LINE'(2,1) EQ '?') SKIP CBL INIT ASM CODE
             :&SKIP_INS_CNT SETA &SKIP_INS_CNT+1
```

```
    :&TOT_SKIP_ASM SETA &TOT_SKIP_ASM+1
    AEXIT AENTRY
  AEND
AEND
:&CBL_KEY  SETC '&ASM_LINE'(1,8)
AIF (&OPT_CBL AND '&LAST_CBL_KEY' NE '&CBL_KEY')
    :&LAST_CBL_KEY SETC '&CBL_KEY'
    AIF ('&CBL_KEY'(2,1) NE '?')  IS THIS VALID CBL KEY
        AIF (&OPT_ASM)
            :&TRS_LINE SETC ' '
            ACALL PUT_TRS_LINE
        AEND
        LCLC &(&CBL_KEY)
        AIF ('&CBL_KEY'(1,1) EQ 'C')
            :&TRS_LINE SETC 'CBL=&(&CBL_KEY)'  CBL
            :&TOT_CBL SETA &TOT_CBL+1
        AELSEIF ('&CBL_KEY'(1,1) EQ 'E')
            :&TRS_LINE SETC 'ECC=&(&CBL_KEY)'  ECC
            :&TOT_ECC SETA &TOT_ECC+1
        AELSE
            :&TRS_LINE SETC 'ECA=&(&CBL_KEY)'  ECA
            :&TOT_ECA SETA &TOT_ECA+1
        AEND
        ACALL PUT_TRS_LINE     OUTPUT CBL LINE
        AIF (&OPT_ASM)
            :&TRS_LINE SETC ' '
            ACALL PUT_TRS_LINE
        AEND
    AEND
AEND
AIF (&OPT_DETAIL)
    :&TRS_LINE SETC '&REC'(1,24).' '.'&REC'(25,*)
    ACALL PUT_TRS_LINE     OUTPUT TRE LINE
    AIF (NOT &OPT_CBL)
        :&TRS_LINE SETC ' '.'&ASM_LINE'          NO CBLKEY
    AELSE
        :&TRS_LINE SETC ' '.'&ASM_LINE'(9,*)     SKP CBLKEY
    AEND
    ACALL PUT_TRS_LINE     OUTPUT ASM LINE
```

```
                :&TOT_ASM SETA &TOT_ASM+1
            AELSEIF (&OPT_ASM)
                AIF (NOT &OPT_CBL)  IS THERE CBL KEY TO SKIP
                    :&TRS_LINE SETC '&REC'(1,25).'&ASM_LINE'
                AELSE
                    :&TRS_LINE SETC '&REC'(1,25).'&ASM_LINE'(9,*)
                AEND
                ACALL PUT_TRS_LINE
                :&TOT_ASM SETA &TOT_ASM+1
            AELSE
                :&SKIP_INS_CNT SETA &SKIP_INS_CNT+1
                :&TOT_SKIP_ASM SETA &TOT_SKIP_ASM+1
            AEND
         AEND
         AEND
.*
.* LOAD 390 LST - READ LST AND FOR EACH INCLUDE READ PRN
.*               AND STORE SOURCE LINES USING CREATED NAME
.*               FROM RESOLVED PSW_ADDR = LOAD_ADDR + REL ADDR
.*               OF NEXT INSTRUCTION AFTER SOURCE LINE
.* NOTES:
.*   1.  IF MAIN_NAME NOT DEFINED, IT IS SET FROM FIRST TRE CDE LOAD
.*       AND IF NOT ALL AND NO INC/EXC SET DEFAULT INC(MAIN_NAME)
.*   3.  USE TRE FILE PATH, ELSE USE TRE FILE CDE LOAD MESSAGE PATH.
.*
         AENTRY LOAD_LST
         :&LST_REC AREAD ID=2,DSNAME='X'   RESET TO REREAD
         :&TOT_LST_FIL SETA &TOT_LST_FIL+1
         :&LOAD_LOC SETA X2A('&LOAD_ADDR')
         :&LOAD_END SETA &LOAD_LOC+X2A('&LOAD_LEN')
         :&PATH_NAME SETC '&LST_DSN'
         ACALL GET_PATH_NAME
         :&CDE_PATH SETC '&PATH'
         AIF ('&MAIN_NAME' EQ '')
             :&MAIN_NAME SETC '&NAME'
             AIF (NOT &OPT_ALL AND &NINC+&NEXC EQ 0)
                 :&NINC SETA 1
                 :&INC_NAME(1) SETC (UPPER '&NAME')
                 :&TRS_LINE SETC 'TRS= DEFAULT INCLUDE(&NAME)'
```

```
        ACALL PUT_TRS_LINE
    AEND
AEND
AIF ('&LOADLIB' NE '')
    :&LST_DSN SETC '&LOADLIB\&NAME..LST'
AELSE
    AIF ('&CDE_PATH' NE '')
        :&LST_DSN SETC '&CDE_PATH\&NAME..LST' TRY CDE PATH
    AELSE
        :&LST_DSN SETC '&NAME..LST'
    AEND
AEND
:&LST_REC AREAD ID=2,DSNAME='&LST_DSN'
AIF ('&LST_REC' EQ '')
    :&ERR_LVL SETA 8
    :&ERR_MSG SETC 'LST FILE NOT FOUND - &LST_DSN - EXCLUDED'
    ACALL PUT_ERR
    AEXIT AENTRY
AEND
:&TRS_LINE SETC 'TRS= SCANNING  &LST_DSN LOAD=&LOAD_ADDR'
ACALL PUT_MSG
:&LST_INC SETA 0
AWHILE ('&LST_REC' NE '')
      :&TOT_LST_REC SETA &TOT_LST_REC+1
      ACTR 4096
      AIF ('&LST_REC'(1,17) EQ 'LZ390I INCLUDE = ')
          :&TOT_INC SETA &TOT_INC+1
          :&PRN_DSN SETC '&LST_REC'(18,*)   LOAD ?.OBJ
          :&PRN_DSN SETC '&PRN_DSN'(1,K'&PRN_DSN-4).'.PRN'
          ACALL LOAD_PRN
          :&LOAD_LOC SETA &LOAD_LOC+&MOD_LEN
      AEND
      :&LST_REC AREAD ID=2,DSNAME='&LST_DSN'
AEND
AIF   (&LOAD_LOC NE &LOAD_END)
      :&DIFF SETA &LOAD_END-&LOAD_LOC
      :&ERR_LVL SETA 8
      :&ERR_MSG SETC 'LST VS PRN LENGTH ERROR - &DIFF - EXCLUDX
      ING'
```

```
               ACALL PUT_ERR
               AEXIT AENTRY
         AEND
         AIF   (&TOT_INC EQ 0)
               :&ERR_LVL SETA 16
               :&ERR_MSG SETC 'NO INCLUDES FOUND IN &LST_DSN - ABORT'
               ACALL PUT_ERR
               ACALL TERM
         AEND
         AEND
.*
.* LOAD PRN - LOAD CBL, ECC, ASM, AND ECCA SOURCE LINES IF INCLUDED
.*           AT PSW_ADDR = LOAD_ADDR + MOD_ADDR
.* NOTES:
.*   1.  USE OVERRIDE INCLUDE PATH, ELSE USE TRE PATH, ELSE USE
.*       LST INCLUDE PATH.
.*   2.  ISSUE ERROR AND ABORT IF LST NOT FOUND
.*   3.  PROCESS
.*
         AENTRY LOAD_PRN
         :&PRN_REC AREAD ID=3,DSNAME='X'   RESET TO REREAD
         :&PGM_TYPE SETC 'A'  ASSUME ASSEMBLER VS ZCOBOL
         :&CBL_KEY_TYPE  SETC 'C' ASSUME C=CBL, VS EX CICS E=ECC
         :&MOD_LEN   SETA 0   TOTAL LEN OF ALL CSECTS IN MODULE
         :&PRN_CBL   SETA 0   CBL SOURCE LINES LOADED FROM PRN
         :&PRN_ECC   SETA 0   ECC SOURCE LINES LOADED FROM PRN
         :&PRN_ASM   SETA 0   ASM SOURCE LINES LOADED FROM PRN
         :&PRN_ECA   SETA 0   ECA SOURCE LINES LOADED FROM PRN
         :&LAB_HEX   SETC ''
         :&LAB_NAME  SETC ''
         :&CBL_LINE SETC ''
         :&CBL_KEY  SETC ''
         :&PATH_NAME SETC '&PRN_DSN'
         ACALL GET_PATH_NAME
         :&CDE_PATH SETC '&PATH'
         :&CDE_NAME SETC '&NAME'
         :&NAME SETC (UPPER '&NAME       '(1,8))
         ACALL CHECK_NAME
         AIF (&OPT_INC)
```

```
    :&INC_CNT(&I) SETA &INC_CNT(&I)+1 COUNT INCLUDE LOADS
    :&TOT_PRN_INC SETA &TOT_PRN_INC+1
AELSE
    :&TOT_PRN_EXC SETA &TOT_PRN_EXC+1
AEND
AIF ('&LOADLIB' NE '')
    :&PRN_DSN SETC '&LOADLIB\&CDE_NAME..PRN' TRY LOADLIB PATH
AELSE
    AIF ('&CDE_PATH' NE '')
        :&PRN_DSN SETC '&CDE_PATH\&CDE_NAME..PRN' TRY CDE PATH
    AELSE
        :&PRN_DSN SETC '&CDE_NAME..PRN'
    AEND
AEND
:&PRN_REC AREAD ID=3,DSNAME='&PRN_DSN'
AIF ('&PRN_REC' EQ '')
    :&ERR_LVL SETA 8
    :&ERR_MSG SETC 'PRN FILE NOT FOUND &PRN_DSN - EXCLUDING'
    ACALL PUT_ERR
    AEXIT AENTRY
AEND
:&PSW_HEX  SETC A2X(&LOAD_LOC)
:&PSW_ADDR SETC '000000&PSW_HEX'(K'&PSW_HEX,7)
AIF   (&OPT_INC)
      :&TRS_LINE SETC 'TRS= INCLUDING &PRN_DSN LOAD=&PSW_ADDR'
      ACALL PUT_MSG
AELSE
      :&TRS_LINE SETC 'TRS= EXCLUDING &PRN_DSN LOAD=&PSW_ADDR'
      ACALL PUT_MSG
AEND
AWHILE ('&PRN_REC' NE '')
      :&TOT_PRN_REC SETA &TOT_PRN_REC+1
      ACTR 4096
      AIF (K'&PRN_REC GT 54                                   X
           AND '&PRN_REC'(54,1) EQ '*')     COMMENT
           AIF (NOT &OPT_INC)
               AEXIT AENTRY EXCLUDING MODULE AFTER ESD LENGTHS
           AEND
           AIF ('&PRN_REC'(55,2) EQ 'ZC')     ZCOBOL CALL
```

```
         :&CBL_KEY_TYPE SETC 'C'  ASSUME COBOL
         AIF (K'&PRN_REC GT 83)
             AIF ('&PRN_REC'(73,10) EQ 'EXEC  CICS')
                 :&CBL_KEY_TYPE SETC 'E' EX CICS CBL
             AEND
         AEND
         AIF ('&CBL_KEY_TYPE' EQ 'C')
             :&TOT_CBL_KEY SETA &TOT_CBL_KEY+1
             :&PRN_CBL SETA &PRN_CBL+1
         AELSE
             :&TOT_ECC_KEY SETA &TOT_ECC_KEY+1
             :&PRN_ECC SETA &PRN_ECC+1
         AEND
         AIF ('&CBL_KEY' EQ '' AND '&CBL_LINE' NE '')
             :&CBL_LINE SETC '&CBL_LINE ; '.'&PRN_REC'(7X
3,*)
         AELSE
             :&CBL_LINE SETC '&NAME'.'&PRN_REC'(57,*)
         AEND
         :&CBL_KEY  SETC ''  WILL BE SET AT NEXT ASM
     AELSEIF ('&PRN_REC'(55,7) EQ ' ZCOBOL')
         :&PGM_TYPE SETC 'C'
     AEND
AELSEIF (K'&PRN_REC GT 65                               X
         AND '&PRN_REC'(54,11) EQ ' EXEC CICS,')
      AIF (NOT &OPT_INC)
         AEXIT AENTRY EXCLUDING MODULE AFTER ESD LENGTHS
      AEND
     :&CBL_KEY_TYPE SETC 'X'   EXEC CICS ASSEMBLER
     :&TOT_ECA_KEY SETA &TOT_ECA_KEY+1
     :&PRN_ECA SETA &PRN_ECA+1
     AIF ('&CBL_KEY' EQ '' AND '&CBL_LINE' NE '')
         :&CBL_LINE SETC '&CBL_LINE ; '.'&PRN_REC'(54,*)
     AELSE
         :&CBL_LINE SETC '&NAME'.'&PRN_REC'(54,*)
     AEND
     :&CBL_KEY  SETC ''  WILL BE SET AT NEXT ASM
AELSEIF (K'&PRN_REC GT 54                               X
     AND '&PRN_REC'(1,1)  EQ '0'                        X
```

```
      AND '&PRN_REC'(7,1)  EQ ' '                        X
      AND '&PRN_REC'(8,1)  NE ' '                        X
      AND '&PRN_REC'(20,1) EQ ' ')      ASM INSTR LINE
     AIF (NOT &OPT_INC)
         AEXIT AENTRY  EXCLUDING MODULE AFTER ESD LENGTHS
     AEND
     :&REL_HEX  SETC '&PRN_REC'(1,6)
     :&REL_LOC  SETA X2A(&REL_HEX)
     :&PSW_LOC  SETA &LOAD_LOC+&REL_LOC
     :&PSW_HEX  SETC A2X(&PSW_LOC)
     :&PSW_ADDR SETC '000000&PSW_HEX'(K'&PSW_HEX,7)
     :&ASM_KEY  SETC 'A&PSW_ADDR'
     AIF (&OPT_CBL)
         AIF ('&CBL_KEY' EQ '')
             AIF ('&CBL_LINE' NE '')
                 :&CBL_KEY SETC '&CBL_KEY_TYPE&PSW_ADDR'
                 :&(&CBL_KEY) SETC '&CBL_LINE'
             AELSE
                 :&CBL_KEY SETC '&PGM_TYPE???????' UNDEF
             AEND
         AELSEIF ('&PRN_REC'(53,1) NE '+') END OF ECA?
             :&CBL_KEY SETC '&PGM_TYPE???????' SET UNDEF
         AEND
     AEND
     :&SOURCE   SETC '&PRN_REC'(54,*)
     AIF ('&REL_HEX' EQ '&LAB_HEX')
         AIF ('&SOURCE'(1,1) EQ ' ')
             :&OP_IX SETA 2
             AWHILE (&OP_IX LE K'&SOURCE)
                AIF ('&SOURCE'(&OP_IX,1) NE ' ')
                    :&SOURCE SETC '&LAB_NAME '.'&SOURCE'(X
&OP_IX,*)
                    AEXIT AWHILE
                AEND
                :&OP_IX SETA &OP_IX+1
             AEND
        AEND
     AEND
     :&ASM_LINE SETC '&NAME &REL_HEX '.'&SOURCE'
```

```
          AIF (&OPT_CBL)
              :&(&ASM_KEY) SETC '&CBL_KEY&ASM_LINE'
          AELSE
              :&(&ASM_KEY) SETC '&ASM_LINE'
          AEND
          :&TOT_ASM_KEY SETA &TOT_ASM_KEY+1
          :&PRN_ASM SETA &PRN_ASM+1
      AELSEIF (K'&PRN_REC GT 54                               X
           AND '&PRN_REC'(1,1)  EQ '0'                        X
           AND '&PRN_REC'(7,18) EQ (18)' '                    X
           AND '&PRN_REC'(54,1) NE ' ')
           AIF  ('&PRN_REC'(1,6) EQ '&PRN_REC'(19,6)          X
                OR '&PRN_REC'(19,6) EQ (6)' ') REL EQU,DS/CST
                AIF (NOT &OPT_INC)
                    AEXIT AENTRY EXC MODULE AFTER ESD LENGTHS
               AEND
               :&SOURCE   SETC '&PRN_REC'(54,*)
               :&SPACE_IX SETA ('&SOURCE' FIND ' ')
               AIF (&SPACE_IX GT 1)
                   :&LAB_NAME SETC '&SOURCE'(1,&SPACE_IX-1)
                   AIF (&SPACE_IX LE 8)
                     :&LAB_NAME SETC '&LAB_NAME        '(1,8)
                   AEND
                   :&LAB_HEX  SETC '&PRN_REC'(1,6)
               AEND
           AEND
      AELSEIF (K'&PRN_REC GT 44                               X
           AND '&PRN_REC'(2,4)  EQ 'ESD='                     X
           AND '&PRN_REC'(37,8) EQ 'TYPE=CST') ADD ESD CST LNG
           :&CST_LEN  SETA X2A('&PRN_REC'(28,8))
           :&MOD_LEN  SETA &MOD_LEN+&CST_LEN
      AEND
      :&PRN_REC AREAD ID=3,DSNAME='&PRN_DSN'
AEND
:&TRS_LINE SETC 'TRS= LOADED SOURCE FROM PRN CBL=&PRN_CBL ECC=X
      &PRN_ECC ASM=&PRN_ASM ECA=&PRN_ECA'
ACALL PUT_MSG
AEND
```

```
.*
```

```
.* CHECK IF MODULE NAME TO BE INCLUDED
.*
         AENTRY CHECK_NAME
         :&OPT_INC SETB 1
         AIF   (&OPT_ALL)
               AEXIT AENTRY
         AEND
         :&I SETA 1
         AWHILE (&I LE &NINC)
               AIF ('&INC_NAME(&I)' EQ '&NAME'(1,K'&INC_NAME(&I)))
                   AEXIT AENTRY
               AEND
               :&I SETA &I+1
         AEND
         :&OPT_INC SETB 0
         AIF (&NINC GT 0)
             AEXIT AENTRY
         AEND
         :&I SETA 1
         AWHILE (&I LE &NEXC)
               AIF ('&EXC_NAME(&I)' EQ '&NAME'(1,K'&EXC_NAME(&I)))
                   AEXIT AENTRY
               AEND
               :&I SETA &I+1
         AEND
         :&OPT_INC SETB 1
         AEND
.*
.* INIT
.*
         AENTRY INIT
         ACALL INIT_PARMS
         :&TRS_LINE SETC 'TRS= ZPARTRS Z390 PROGRAM ANALYSIS REPORT TRAX
               CE SOURCE &VERSION'
         ACALL PUT_TRS_LINE
         :&TRS_LINE SETC 'TRS= SYSPARM=&PARMS'
         ACALL PUT_TRS_LINE
         :&I SETA 1
         AWHILE (&I LE &NP)
```

```
            :&TRS_LINE SETC 'TRS= PARM(&I)=&PARM(&I)'
            ACALL PUT_TRS_LINE
            :&I SETA &I+1
        AEND
        :&TRS_LINE SETC 'TRS= CURRENT DATE=&SYSDATE TIME=&SYSTIME'
        ACALL PUT_TRS_LINE
        AEND
.*
.*
.* INIT PARMS
.*
        AENTRY INIT_PARMS
        ACALL GET_PARMS
        AIF  ('&PARM(1)' EQ '')
             MNOTE 16,'ZPARTRS SYSPARM FILE PARM MISSING - ABORTING'
             MEXIT
        AEND
        :&PATH_NAME SETC '&PARM(1)'
        ACALL GET_PATH_NAME
        :&TRE_PATH SETC '&PATH'
        :&TRE_NAME SETC '&NAME'
        AIF ('&TRE_PATH' NE '')
            :&TRE_DSN SETC '&TRE_PATH\&TRE_NAME..TRE'
            :&TRS_DSN SETC '&TRE_PATH\&TRE_NAME..TRS'
        AELSE
            :&TRE_DSN SETC '&TRE_NAME..TRE'
            :&TRS_DSN SETC '&TRE_NAME..TRS'
        AEND
        :&I SETA 2
        AWHILE (&I LE &NP)
            AIF (UPPER '&PARM(&I)' EQ 'ALL')              ALL CBL+ASM
                :&OPT_ALL SETB 1
                :&OPT_INC SETB 1
                :&OPT_CBL SETB 1
                :&OPT_ASM SETB 1
            AELSEIF (UPPER '&PARM(&I)' EQ 'NOALL')              NOALL
                :&OPT_ALL SETB 0
                :&OPT_INC SETB 0
                :&OPT_CBL SETB 1
```

```
    :&OPT_ASM SETB 1
AELSEIF (UPPER '&PARM(&I)' EQ 'ASM')                ASM
    :&OPT_ASM  SETB 1
AELSEIF (UPPER '&PARM(&I)' EQ 'NOASM')              NOASM
    :&OPT_ASM  SETB 0
AELSEIF (UPPER '&PARM(&I)' EQ 'CBL')                CBL
    :&OPT_CBL  SETB 1
AELSEIF (UPPER '&PARM(&I)' EQ 'NOCBL')              NOCBL
    :&OPT_CBL  SETB 0
AELSEIF (UPPER '&PARM(&I)' EQ 'DETAIL')             DETAIL
    :&OPT_DETAIL SETB 1
    :&OPT_ASM    SETB 1
AELSEIF (UPPER '&PARM(&I)' EQ 'NODETAIL')         NODETAIL
    :&OPT_DETAIL SETB 0
AELSEIF (UPPER '&PARM(&I)'(1,8) EQ 'EXCLUDE(')    EXCLUDE
    :&PARMS SETC '&PARM(&I)'(9,*)
    :&NEXC SETA 1
    AWHILE ('&PARMS' NE '')
        :&J SETA ('&PARMS' FIND '+)')
        AIF (&J GT 0)
            AIF ('&PARMS'(&J-1,1) EQ '*')
                :&EXC_NAME(&NEXC) SETC (UPPER '&PARMS'(1,&X
  J-2))
            AELSE
                :&NAME SETC (UPPER '&PARMS'(1,&J-1).(7)' 'X
  )
                :&EXC_NAME(&NEXC) SETC '&NAME'(1,8)
            AEND
            :&PARMS SETC '&PARMS'(&J+1,*)
            AIF ('&PARMS' NE '')
                :&NEXC SETA &NEXC+1
            AEND
        AELSE
            :&ERR_LVL SETA 16
            :&ERR_MSG SETC 'EXCLUDE MISSING ) - ABORT'
            ACALL PUT_ERR
            ACALL TERM
        AEND
    AEND
```

```
AELSEIF (UPPER '&PARM(&I)'(1,8) EQ 'INCLUDE(')    INCLUDE
    :&PARMS SETC '&PARM(&I)'(9,*)
    :&NINC SETA 1
    AWHILE ('&PARMS' NE '')
        :&J SETA ('&PARMS' FIND '+)')
        AIF (&J GT 0)
            AIF ('&PARMS'(&J-1,1) EQ '*')
                :&PATH_NAME SETC '&PARMS'(1,&J-2)
                ACALL GET_PATH_NAME
                :&INC_NAME(&NINC) SETC (UPPER '&NAME')
            AELSE
                :&PATH_NAME SETC '&PARMS'(1,&J-1)
                ACALL GET_PATH_NAME
                :&NAME SETC (UPPER '&NAME       ')
                :&INC_NAME(&NINC) SETC '&NAME'(1,8)
            AEND
            :&PARMS SETC '&PARMS'(&J+1,*)
            AIF ('&PARMS' NE '')
                :&NINC SETA &NINC+1
            AEND
        AELSE
            :&ERR_LVL SETA 16
            :&ERR_MSG SETC 'INCLUDE MISSING ) - ABORT'
            ACALL PUT_ERR
            ACALL TERM
        AEND
    AEND
AELSEIF (UPPER '&PARM(&I)'(1,8) EQ 'LOADLIB(')    LOADLIB
    :&LOADLIB SETC '&PARM(&I)'(9,K'&PARM(&I)-9)
    AIF ('&LOADLIB'(K'&LOADLIB,1) EQ '\')
        :&LOADLIB SETC '&LOADLIB'(1,K'&LOADLIB-1)
    AEND
AELSEIF (UPPER '&PARM(&I)' EQ 'MSG')               MSG
    :&OPT_MSG  SETB 1
AELSEIF (UPPER '&PARM(&I)' EQ 'NOMSG')             NOMSG
    :&OPT_MSG  SETB 0
AELSEIF (UPPER '&PARM(&I)' EQ 'TIME')              TIME
    :&OPT_TIME SETB 1
AELSEIF (UPPER '&PARM(&I)' EQ 'NOTIME')            NOTIME
```

```
                 :&OPT_TIME SETB 0
             AELSE
                 :&ERR_LVL SETA 16
                 :&ERR_MSG SETC 'UNKNOWN OPTION &PARM(&I) - ABORT'
                 ACALL PUT_ERR
                 ACALL TERM
             AEND
             :&I SETA &I+1
         AEND
         :&PARMS SETC '&SYSPARM'
         AWHILE (K'&PARMS GT 1 AND '&PARMS'(K'&PARMS,1) EQ '+')
               :&PARMS SETC '&PARMS'(1,K'&PARMS-1)
         AEND
         AIF   (&NINC+&NEXC GT 0)
               AIF (&OPT_ALL)
                   :&ERR_LVL SETA 4
                   :&ERR_MSG SETC 'ALL OVERRIDES INCLUDE/EXCLUDE'
                   ACALL PUT_ERR
                   :&NINC SETA 0
                   :&NEXC SETA 0
               AELSEIF (&NINC GT 0 AND &NEXC GT 0)
                   :&ERR_LVL SETA 4
                   :&ERR_MSG SETC 'INCLUDE OVERIDES EXCLUDE'
                   ACALL PUT_ERR
                   :&NEXC SETA 0
               AEND
         AEND
         AEND
.*
.* GET PARMS(N) FROM SYSPARM(PARM1+PARMN)
.*
         AENTRY GET_PARMS
         AIF (K'&SYSPARM EQ 0)
             AEXIT AENTRY
         AEND
         :&PARMS SETC '&SYSPARM'
         :&NP SETA 1
         AWHILE ('&PARMS' NE '')
            :&J SETA ('&PARMS' FIND '+(')
            AIF (&J GT 0)
                :&CHAR SETC '&PARMS'(&J,1)
                AIF ('&CHAR' EQ '+')
                    :&PARM(&NP) SETC '&PARMS'(1,&J-1)
                    AIF ('&PARM(&NP)' EQ '')
                        :&NP SETA &NP-1
                        AEXIT AWHILE
                    AEND
                    :&PARMS    SETC '&PARMS'(&J+1,*)
                    :&NP       SETA &NP+1
                AELSE
                    :&K SETA ('&PARMS' FIND ')')
                    AIF (&K GT 0)
                        :&PARM(&NP) SETC '&PARMS'(1,&K)
                        AIF ('&PARM(&NP)' EQ '')
                            :&NP SETA &NP-1
                            AEXIT AWHILE
                        AEND
                        AIF (K'&PARMS GT &K)
                           :&PARMS    SETC '&PARMS'(&K+2,*)
                           :&NP       SETA &NP+1
                        AELSE
                           :&PARMS    SETC ''
                        AEND
                    AELSE
                        :&ERR_LVL SETA 16
                        :&ERR_MSG SETC 'SYSPARM MISSING ) - ABORT'
                        ACALL PUT_ERR
                        ACALL TERM
                    AEND
                AEND
            AELSE
                :&PARM(&NP) SETC '&PARMS'
                :&PARMS    SETC ''
            AEND
         AEND
         AEND
.*
.* GET TRE REC AND SAVE TRE NAME IF FIRST RECORD
```

```
.*
         AENTRY GET_TRE
         :&REC AREAD ID=1,DSNAME='&TRE_DSN'
         AIF   ('&REC' NE '')
               :&TOT_TRE SETA &TOT_TRE+1
               AIF  (K'&REC GT 25)
                    AIF ('&REC'(5,1) EQ '-')  RPI 1064
                        :&TIME SETC '&REC'(1,30)
                        :&REC SETC '&REC'(31,*) REMOVE TIMESTAMP
                        AIF ('&REC' EQ '')
                            :&REC SETC ' '   ALLOW BLANK TIMESTAMP
                        AEND
                    AEND
                    AIF ('&REC'(3,1) EQ ':')
                        :&TIME SETC '&REC'(1,9)
                        :&REC SETC '&REC'(10,*) REMOVE TIME FROM START
                    AEND
                    AIF ('&REC'(11,5) EQ 'EZ390')
                        :&EZ390_FOUND SETB 1
                    AELSEIF (NOT &EZ390_FOUND)
                        :&ERR_LVL SETA 16
                        :&ERR_MSG SETC 'EZ390 START NOT FOUND - ABORT'
                        ACALL PUT_ERR
                        ACALL TERM
                    AEND
               AEND
         AEND
         AEND
.*
.* GET PATH AND NAME FROM PATH_NAME
.*
         AENTRY GET_PATH_NAME
         :&PATH SETC ''
         :&NAME SETC '&PATH_NAME'
         :&SLASH_IX SETA K'&PATH_NAME-1
         AWHILE (&SLASH_IX GT 0)
             :&CHAR SETC '&PATH_NAME'(&SLASH_IX,1)
             AIF ('&CHAR' EQ '\' OR '&CHAR' EQ '/')
                 :&PATH SETC '&PATH_NAME'(1,&SLASH_IX-1)
                 :&NAME SETC '&PATH_NAME'(&SLASH_IX+1,*)
                 AEXIT AWHILE
             AEND
             :&SLASH_IX SETA &SLASH_IX-1
         AEND
         :&IPER SETA ('&NAME' FIND '. ')  FIND PERIOD OR SPACE
         AIF (&IPER GT 0)
             :&NAME SETC '&NAME'(1,&IPER-1)
         AEND
         AEND
.*
.* PUT ERR MESSAG ON TRS OUTPUT AND ISSUE MNOTE TO ERR LOG
.*
         AENTRY PUT_ERR
         :&TOT_ERR SETA &TOT_ERR+1
         :&TRS_LINE SETC 'ERR= ERROR LVL=&ERR_LVL &ERR_MSG'
         MNOTE &ERR_LVL,'&TRS_LINE'
         AIF (&MAX_RC LT &ERR_LVL)
             :&MAX_RC SETA &ERR_LVL
         AEND
         AIF ('&TRS_DSN' NE '')
             ACALL PUT_TRS_LINE
         AEND
         AEND
.*
.* PUT MSG VIA PUT_TRS_LINE IF OPT_MSG
.*
         AENTRY PUT_MSG
         :&TOT_MSG SETA &TOT_MSG+1
         AIF (&OPT_MSG)
             ACALL PUT_TRS_LINE
         AEND
         AEND
.*
.* TERMINATE AFTER MNOTE WITH TOTAL ERRORS AND MAX_RC
.*
         AENTRY TERM
         :&TRS_LINE SETC 'TRS= TOTAL MNOTE ERRORS=&TOT_ERR  MAX RETURN X
               CODE=&MAX_RC'
         ACALL PUT_TRS_LINE
         MNOTE &MAX_RC,'&TRS_LINE'
         MEXIT
         AEND
.*
.* PUT TRS OUTPUT FILE RECORD FROM TRS_LINE
.*
         AENTRY PUT_TRS_LINE
         AIF   (&OPT_ASM)
               :&SKIP_TOT SETA &SKIP_INS_CNT+&SKIP_MSG_CNT+&SKIP_BRK_CNX
               T
               AIF (&SKIP_TOT GT 0)
                     AIF (&SKIP_BRK_CNT EQ &SKIP_TOT)
                          AIF (NOT &LAST_BRK)
                             :&LAST_BRK SETB 1
                             :&TOT_TRS SETA &TOT_TRS+1
                             PUNCH ' ',DSNAME='&TRS_DSN'
                          AEND
                     AELSE
                          :&TOT_TRS SETA &TOT_TRS+1
                          PUNCH '.... SKIP INS =&SKIP_INS_CNT MSG =&SKIX
               P_MSG_CNT ....',DSNAME='&TRS_DSN'
                     AEND
                     :&SKIP_INS_CNT SETA 0
                     :&SKIP_MSG_CNT SETA 0
                     :&SKIP_BRK_CNT SETA 0
               AEND
         AEND
         AIF   ('&TRS_LINE' EQ ' ')
               AIF (&LAST_BRK)
                    AEXIT AENTRY
               AEND
         AEND
         :&LAST_BRK SETB 0
         AIF   (&OPT_TIME)
               :&TRS_LINE SETC (DOUBLE '&TIME&TRS_LINE')
         AELSE
               :&TRS_LINE SETC (DOUBLE '&TRS_LINE')
         AEND
```

```
PUNCH '&TRS_LINE',DSNAME='&TRS_DSN'
:&TOT_TRS SETA &TOT_TRS+1
AEND
END
```