**DISASSEMBLER USER GUIDE**  
  
 DISASM is a multi-pass disassembler. Source assembler statements are  
 produced from a CSECT of a linkedited module. The module may contain  
 multiple CSECTs, however DISASM only disassembles one CSECT per  
 execution. To disassemble, specify a DISMOD DD card (and optional  
 concatenations) specifying load module libraries (no member names).  
 In addition to load modules, DISASM also accepts object decks; the  
 object libraries are specified on a DISMOD DD card as either one  
 sequential file, or one or more concatenated (card format) PDSs.  
 Note that in any one run, DISMOD points to either object decks or  
 load module libaries; the types must not be mixed.  
 The module to be processed is specified on a MODULE control card in  
 the DISIN DD; the CSECT, unless first in the module, on a CSECT card.  
  
 In addition to disassembly, DISASM also supports retrieving source  
 text from other formats. To rebuild source from SYSADATA output  
 (assembly with the ADATA option), include a DISADATA DD. To rebuild  
 source from a saved assembly listing, specify a DISLIST DD. These DDs  
 specify a sequential data set or PDS member.  
  
 Control statements inappropriate for the function are ignored.  
 When more than one of these three DDs is present, only one is acted  
 upon, starting with DISADATA, then DISLIST, then DISMOD.  
  
 For DISADATA input, use an ADATA control statement to specify how you  
 wish to treat MACRO and COPY code. By default, expanded lines are  
 converted to comments, retaining the source MACRO or COPY statements.  
 The MACRO and COPY operands, respectively, change the MACRO or COPY  
 to a comment, and produce the expanded statements instead.  
 For DISLIST, the LIST control card serves the same function.  
  
 All other statements apply to disassembly from load modules or object  
 decks. Control statements:  
  
 Comment statement (optional)  
  
 Columns 1 - 1 \* (asterisk)  
 Comments may be specified at any time. Comments  
 will be copied to the output listing.  
  
 ABEND statement (optional)  
  
 Columns 1 - 5 'ABEND'  
 Causes normal program termination to produce an  
 abnormal termination.  
  
  
 ADATA statement (optional)  
  
 Columns 1 - 5 Literal 'ADATA'.  
 By default, presence of DISADATA rebuilds the  
 source information from the file with the omission  
 of statements expanded by COPY and MACRO requests.  
 10 - 12/13 Literal MAC or literal COPY, in any order, to  
 20 - 22/23 specify that the COPY or MACRO invocation be made  
 a comment, and the expanded code be punched  
 instead.  
  
 ASM START statement (optional)  
  
 Columns 1 - 9 Literal 'ASM START'.  
 Delimits the start of input for the assembler.  
 Any statements specified are also copied to the  
 end of the disassembled punched output. This  
 allows DSECTs to be assembled so the labels  
 can be used for data references. The disassembler  
 places a PRINT OPSYN COM statement prior to the  
 user input, so that all statements should be  
 expanded.  
  
  
 ASM END statement (optional unless ASM START is used)  
  
 Columns 1 - 7 Literal 'ASM END'.  
 Delimits the end of the assembler input.

BASE statement (optional)  
  
 Columns 1 - 4 Literal 'BASE'.  
 10 - 12 Base register. Register 1 may be specified as 1 or  
 R1, register 10 may be specified as 10 or R10.  
 Registers 10 to 15 may also be specified as A to  
 F or RA to RF.  
 20 - 27 Starting displacement into the CSECT that the base  
 register is in effect. Displacements are given as  
 hex values 1 to 8 digits long, leading zeros are  
 not required (0A0 is equivalent to A0).  
 This address corresponds to the location where you  
 want a USING statement generated.  
 30 - 37 Optional. Ending displacement into the CSECT that  
 the base register is in effect. If omitted, the  
 ending displacement will be 4096 from the starting  
 value (in columns 20-27). Coding is the same as  
 for the starting displacement. This corresponds  
 to the location of the DROP statement.  
 40 - 47 Displacement the base register refers to. For  
 example if the base register is R12 and R12 is  
 used as the base in instructions starting at  
 +X'20', and R12 is set to point to +X'22C' into  
 the CSECT, the BASE statement would be:  
  
 BASE R12 20 22C  
  
 If a base register is defined, a PREFIX statement  
 is also required. Multiple BASE statements for the  
 same register and base address are permitted,  
 corresponding to multiple USING and DROPs for it.  
 CSECT statement (optional)  
  
 Columns 1 - 5 Literal 'CSECT'.  
 10 - 17 CSECT's name. If the CSECT is not specified, the  
 first CSECT in the module will be disassembled.  
  
  
 DATA statement (optional)  
  
 Columns 1 - 4 Literal 'DATA'.  
 10 - 17 Starting displacement into the CSECT where a data  
 area begins. DATA statements identify areas that  
 do not contain instructions. See the BASE  
 statement for displacement coding syntax.  
 20 - 27 Displacement where the data end.  
  
  
 FILL statement (optional)  
  
 Columns 1 - 6 Literal 'FILLER' or 'FILL' or 'ZDATA'.  
 10 - 17 Starting displacement into the CSECT where data  
 storage begins. FILL statements identify areas that  
 do not contain data, and the contents of which are  
 irrelevant to proper operation. These result from  
 DS and ORG statements in the original source.  
 20 - 27 Displacement where the data end.

LABEL statement (optional)

Columns 1 - 5 Literal 'LABEL'.  
 10 - 17 User specified label for a specific data location  
 or instruction. Should not have a name of the form  
 DATAxxxx or PPPPxxxx (prefix) + hex offset to  
 avoid conflicts with program generated labels.  
 20 - 27 Displacement of the named item in the CSECT.  
  
  
 LINE/LINES/MAXLINES statement (optional).  
  
 Columns 1 - 4 Literal 'LINE'  
 or  
 1 - 5 Literal 'LINES'  
 or  
 1 - 9 Literal 'MAXLINES'  
 10 - 12 Maximum number of lines to print per page on the  
 DISPRINT and DISDEBUG data sets. Line values are  
 2 to 3 digits, minimum is 10 lines. Specify the  
 value left justified, leading zeros may be omitted.  
 Default line count is 60 lines per page.  
  
  
 LIST statement (optional, alias of ADATA)  
  
 Columns 1 - 4 Literal 'LIST'.  
 By default, presence of DISLIST rebuilds the  
 source information from the file with the omission  
 of statements expanded by COPY and MACRO requests.  
 10 - 12/13 Literal MAC or literal COPY, in any order, to  
 20 - 22/23 specify that the COPY or MACRO invocation be made  
 a comment, and the expanded code be punched  
 instead.  
  
  
 MODULE statement (required)  
  
 Columns 1 - 6 Literal 'MODULE'.  
 10 - 17 Load module's name. This module must be in a  
 library specified by the DISMOD DD.  
 20 - 27 Optional CSECT name; may also be supplied on CSECT.

OPCODES statement (optional)  
  
 1 - 7 Literal 'OPCODES' (or OPCODE or INST)  
 10 - 70 free format name:  
 1-8 character load module name of an opcode table;  
 see DISOPnnn source for examples. Provided are  
 DISOP36S (all S/360 instructions including I/O)  
 DISOP37B (all S/370 instructions available under  
 Hercules, e.g., BAS, BASR, MVCIN)  
 DISOPAPP, DISOPAP2 - S/370 selected opcodes for  
 applications program (no decimal, no floating  
 point, no I/O, no privileged instruction).  
 These may be used to reduce false instruction  
 decoding in DATA areas. Use DISOPAPP when base  
 registers are double-word aligned, else DISOPAP2.

OPTIONS statement (optional)  
  
  
 1 - 7 Literal 'OPTIONS' (or OPTION or OPT)  
 10 - 70 free format options:  
 ABSREG¦AREG produces all register numbers in  
 absolute form (0-15), rather than mnemonic (R0-R15)  
 allows correct assignment of labels for assembled  
 control block mappings.  
 BCOPCODE¦BC produces all branches with BC and BCR,  
 rather than the extended mnemonics (e.g., B, BE, BOR)  
 FIXSECT changes all blank CSECT names to  
 PC#nnnnn where nnnnn is the ESDID to allow recovery  
 of unnamed sections (e.g., blank COMMON)  
 IXSWAP¦SWAPIX when an RX instruction has a zero  
 base register, swap with the index register. This  
 allows correct assignment of labels for assembled  
 control block mappings.  
 LOWER¦UPLOW Expand C rather than X for lower  
 case characters  
 PLS¦RAT produce registers in @nn PL/S style  
 ROUND Pads CSECT to double word length.  
 S360 ¦ S/360 use System 360 instruction set  
 (note that SSM is omitted; too many DC errors)  
 S370 ¦ S/370 use System 370 instruction set  
 <default is 370> (does not include SSM)  
 S390 ¦ S/390 use System 390 instruction set  
 (does not include SSM and Vector Facility codes)  
  
  
 PREFIX statement (optional)  
  
  
 1 - 6 literal 'PREFIX'.  
 10 - 14 1 to 4 character label prefix. Labels generated  
 due to references within the CSECT will be  
 PPPPxxxx where PPPP is the prefix specified here  
 and xxxx is the displacement into the CSECT or  
 a sequential number (0010, 0020, 0030, etc).  
 If the prefix is less than 4 characters long, the  
 numeric portion will be padded on the left with  
 zeros (labels will always be 8 characters long).  
 If the PREFIX statement is used, base registers  
 must also be defined. PRINT statement (optional)  
  
 1 - 5 Literal 'PRINT'  
 10 - 70 free format print selections. By default, DISASM  
 prints the module directory information, the  
 Relocation Dictionary data (RLD), and the CSECT  
 text in hex. When a PRINT statement is used, it  
 replaces all default choices. Select from these:  
 DIR ¦ DIRECTORY module directory information  
 RLD relocation dictionary  
 ASM assembled instruction listing  
 PUN create output deck on DISPUNCH  
 XRF ¦ XREF label cross-reference (sets ASM)  
 DEFAULT all of the above  
 HEX ¦ TEXT CSECT text in hexadecimal  
 SYM ¦ SYMT Symbol table data  
 DAT ¦ ADATA ADATA records  
 LBL ¦ LABEL supplied and generated LABEL data  
 ESD ¦ CESD module CESD information  
 BUG ¦ DEBUG debug on DISPRINT, not DISDEBUG  
 MOST all of the above  
 TRC ¦ TRACE ON error print the trace table.  
 ALL all of the above  
  
  
 REPLACE STATEMENT (OPTIONAL)  
  
 Columns 1 - 9 Literal 'REPLACE' or 'REP'  
 10- One to eight digit offset, followed by blank,  
 followed by one or more comma separated fields,  
 consisting of 1-8 hex characters or a quoted string.  
 You may use either paired apostrophes or paired  
 quotes; embedded quotes of the same type are doubled.  
 This card changes module text in storage only, and  
 may be used to alter dates, etc. prior to sending  
 generated output to a compare program, alter BAL  
 to BAS, and L to ICM, and more.  
  
  
 SEQ LABEL STATEMENT (OPTIONAL)  
  
 Columns 1 - 9 Literal 'SEQ LABEL'.  
 Labels generated for labels within the CSECT will  
 be PPPPxxxx where PPPP is the prefix specified on  
 in the LABEL or PREFIX statement and xxxx is either  
 the displacement of the label into the CSECT or a  
 sequential number. By default xxxx will be the  
 displacement in hex. SEQ LABEL requests the labels  
 be generated with sequential numbers. This may be  
 desirable if the source is to be modified and the  
 labels may no longer be at the same displacement.  
  
  
 START STATEMENT (OPTIONAL)  
  
 Columns 1 - 5 Literal ‘START’

10- One to eight digit offset to be added to the displacement  
 field in generated listing. Does not affect anything else  
 OR

\* for CSECT offset in load module

SYSADATA statement (optional, alias of ADATA and LIST)  
  
 Columns 1 - 8 Literal 'SYSADATA'.  
 By default, presence of DISADATA rebuilds the  
 source information from the file with the omission  
 of statements expanded by COPY and MACRO requests.  
 10 - 12/13 Literal MAC or literal COPY, in any order, to  
 20 - 22/23 specify that the COPY or MACRO invocation be made  
 a comment, and the expanded code be punched  
 instead.  
  
  
 USING STATEMENT (OPTIONAL)  
  
 Columns 1 - 5 Literal 'USING'.  
 10 - 17 DSECT's name. Any DSECTs referenced on USING  
 statements must be defined via assembler input.  
 20 - 27 Label within the DSECT. In some cases the base  
 register may be set to point a label within the  
 DSECT rather than at displacement zero. If the  
 base points to displacement zero, leave this  
 parameter blank.  
 \*\*\* Due to an error in the code, a label definined on  
 \*\*\* an EQU may not be detected.  
 30 - 32 Base register. See BASE statement for base reg  
 syntax.  
 40 - 47 If the register specified as the base is used as  
 and the base in all cases where it appears in the  
 50 - 57 code, leave these parameters blank. If the base  
 is only used as a base for the specified DSECT at  
 in a range of instructions, specify the beginning  
 displacement in columns 40-47 and the ending  
 displacement in columns 50-57. The same base may  
 be specified for the same DSECT multiple times with  
 different ranges if necessary.  
  
  
 VERIFY STATEMENT (OPTIONAL)  
  
 Columns 1 - 9 Literal 'VERIFY' or 'VER'  
 10- One to eight digit offset, followed by blank,  
 followed by one or more comma separated fields,  
 consisting of 1-8 hex characters or a quoted string.  
 You may use either paired apostrophes or paired  
 quotes; embedded quotes of the same type are doubled.  
 This statement compares user supplied text with the  
 CSECT text loaded into storage, and terminates  
 disassembly on a mismatch. The primary function  
 is to avoid improper REPLACE operations. JCL REQUIREMENTS  
  
 //.... EXEC PGM=DISASM01,REGION=nnnnK REQ  
 //STEPLIB DD DSN=xxxx,DISP=SHR OPT  
 //SYSPRINT DD DSN=&&PRT,DISP=(NEW,PASS), OPT  
 // UNIT=SYSDA,  
 // SPACE=(TRK,(15,15)),  
 // DCB=(RECFM=FBM,LRECL=121,BLKSIZE=12100)  
 //SYSIN DD DSN=&&IN,DISP=(NEW,PASS), OPT  
 // UNIT=SYSDA,  
 // SPACE=(TRK,(15,15)),  
 // DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)  
 //SYSLIB DD DSN=xxxx,DISP=SHR OPT  
 //SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1)) OPT  
 //SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1)) XF ONLY OPT  
 //SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1)) XF ONLY OPT  
 //SYSPUNCH DD DUMMY OPT  
 //DISDEBUG DD SYSOUT=\* OPT  
 //DISPRINT DD SYSOUT=\* REQ  
 //DISPUNCH DD SYSOUT=class OPT  
 //DISMOD DD DISP=SHR,DSN=load.mod.pds REQ  
 // DD DISP=SHR,DSN=load.mod.pds2 ..... OPT  
 //DISIN DD \* REQ  
 control statements  
 //  
  
If DISASM01 is in a LINKLIST library then the STEPLIB is not required.  
If the assembler input (ASM START/ASM END) is used, SYSPRINT, SYSIN,  
SYSLIB, SYSUTn, and SYSPUNCH are required, otherwise they may be  
omitted.  
If the DISPUNCH DD is present, the source will be generated as an  
80-byte file suitable for input to an assembler.  
Any macros or copy statements specified as assembler input must be  
available in a library in the SYSLIB concatenation.  
  
For SYSADATA source reconstruction:  
  
 //.... EXEC PGM=DISASM01,REGION=nnnnK REQ  
 //STEPLIB DD DSN=xxxx,DISP=SHR OPT  
 //DISDEBUG DD SYSOUT=\* OPT  
 //DISPRINT DD SYSOUT=\* REQ  
 //DISPUNCH DD SYSOUT=class OPT  
 //DISADATA DD DSN=xxxx,DISP=SHR (sequential or member) REQ  
 //DISIN DD \* OPT  
 control statements OPT  
 //  
  
For SYSPRINT source reconstruction:  
  
 //.... EXEC PGM=DISASM01,REGION=nnnnK REQ  
 //STEPLIB DD DSN=xxxx,DISP=SHR OPT  
 //DISDEBUG DD SYSOUT=\* OPT  
 //DISPRINT DD SYSOUT=\* REQ  
 //DISPUNCH DD SYSOUT=class OPT  
 //DISLIST DD DSN=xxxx,DISP=SHR (sequential or member) REQ  
 //DISIN DD \* OPT  
 control statements OPT  
 // ABEND CODES and MESSAGES  
  
 ABEND 001 User requested an ABEND via the ABEND statement.  
  
 ABEND 002 Unknown return code from BLDL.  
  
 ABEND 003 Unknown RLD with unknown type encountered.  
  
 ABEND 004 Internal error, RLD data remaining went negative in  
 DISASM05.  
  
 ABEND 005 Internal error, attempt to generate an instruction on  
 on an odd address boundary.  
  
 ABEND 009 Internal error, attempt to generate an instruction  
 failed on an invalid op-code.  
  
 DISASM0101I NO DISPUNCH DD PRESENT, NO SOURCE WILL BE GENERATED  
 Informational only. The DISPUNCH DD was not present  
 in the JCL, no source deck will be produced.  
  
 DISASM0102I ABEND REQUESTED, PROGRAM ABNORMALLY TERMINATING  
 For diagnostic purposes, an ABEND command was added  
 to cause an abend just prior to freeing the trace table  
 and terminating. This message is issued immediately  
 before the abend, and indicates the abend was due to  
 the user's request and not due to a problem.  
 DISASM will also have abended with a user 001.  
  
 DISASM0103I \*\*\*\*\*\*\* DISASSEMBLY COMPLETE \*\*\*\*\*\*\*  
 Indicates that the disassembly was successful.  
  
 DISASM0104E DISIN DD STATEMENT MISSING, PROCESSING WILL BE ABORTED  
 DISASM requires the DISIN DD statement. DISASM must  
 know at minimum which module it is to disassemble. The  
 MODULE and all other control statements are provided to  
 DISASM via DISIN.  
  
 DISASM0105E DISMOD DD STATEMENT MISSING, PROCESSING WILL BE ABORTED  
 DISASM loads the module to be disassembled from a  
 library specified by the DISMOD DD, this DD is therefore  
 required (unless DISADATA or DISLIST present).  
  
 DISASM0106E ERROR(S) IN CONTROL STATEMENTS, EXECUTION ABORTED  
 Error(s) were detected in the control statements by  
 module DISASM02. Messages are issued to identify the  
 statement(s) in error. Correct the control statements  
 and re-run.  
  
 DISASM0107E ERROR(S) IN LOADING OBJECT MODULE, EXECUTION ABORTED  
 Error(s) were encountered by DISASM03 in loading the  
 specified module. One of the following occurred:  
 1) the BLDL for the member failed  
 2) the POINT for the member (or real member) failed  
 3) the requested CSECT was not in the specified  
 module  
 4) RLD data overlapped a user defined data area  
 Further messages will have been issued to identify the  
 cause.  
  
 DISASM0108E ERROR(S) IN PRINTING TEXT. EXECUTION ABORTED  
 Should never occur. At present there are no errors  
 detected by the text print module, DISASM06.  
  
 DISASM0109E ERROR(S) IN ASSEMBLING DSECTS, EXECUTION ABORTED  
 Either the return code from the assembler was greater  
 than 4, or some other error in interpreting the DSECTs  
 occurred. Messages will have been issued to identify  
 the cause.  
  
 DISASM0110E ERROR(S) IN GENERATING LABEL TABLE, EXECUTION ABORTED  
 An error condition was detected by DISASM08. Messages  
 will have been issued to identify the cause.  
  
 DISASM0111E ERROR(S) IN GENERATING SOURCE, EXECUTION ABORTED  
 An error condition was detected by DISASM09. Messages  
 will have been issued to identify the cause.  
  
 DISASM0112E ERROR(S) PROCESSING DISADATA INPUT  
 An error condition was detected by DISASMDT. Verify  
 that the DISADATA DD points to valid SYSADATA output.  
  
 DISASM0113E ERROR(S) PROCESSING DISLIST INPUT  
 An error condition was detected by DISASMLS. Verify  
 that the DISADATA DD points to valid SYSPRINT output.  
  
 DISASM0201W NO OPTIONS CARDS PROCESSED  
 This is normal when the DISIN DD is missing. DISADATA or  
 DISLIST processing will use defaults.  
  
 DISASM0202W SEQUENTIALLY NUMBERED LABELS HAVE ALREADY BEEN REQUESTED  
 The SEQ LABEL statement is specified more than once in  
 the control statements.  
  
 DISASM0203E INVALID CONTROL STATEMENT  
 Columns 1-9 of a control statement contain an unknown  
 statement type. See the list of control statements  
 for valid control statements and their syntax.  
  
 DISASM0204E EXTRANEOUS DATA IN REGISTER PARAMETER  
 DISASM checks for several blanks following a register  
 specification on BASE and USING statements. This check  
 helps to make sure you specified parameters in the  
 correct columns. If the blanks are not present, this  
 message is issued and the program abort flag set.  
  
 DISASM0205E INVALID REGISTER REFERENCE  
 The register specified on a BASE or USING statement is  
 invalid. Valid values are 0-15, A-F, R0-R15, and RA-RF.  
  
 DISASM0206E END DISPLACEMENT IS REQUIRED WHEN BEGIN DISPLACEMENT IS GIVEN  
 On USING statements the beginning and ending displacement  
 values are optional. If the beginning displacement is  
 given, the ending displacement is also required.  
  
 DISASM0207E BEGIN DISPLACEMENT IS LARGER THAN END DISPLACEMENT  
 The beginning and ending displacement values specify  
 a range of instruction displacements. The end value  
 must be larger than the begin value.  
  
 DISASM0208E BAD HEX DIGIT IN DISPLACEMENT  
 An invalid character was found in a hex value. Valid  
 digits are 0-9 and A-F.  
  
 DISASM0209E TOO MANY DIGITS IN DISPLACEMENT  
 Displacement values must be 8 characters or less.  
  
 DISASM0210E END DISPLACEMENT NOT ALLOWED UNLESS START DISPLACEMENT  
 IS SPECIFIED  
 On USING statements the beginning and ending displacement  
 values are optional. If the ending displacement is  
 given, the beginning displacement is also required.  
  
 DISASM0211E LABEL PREFIX CANNOT BE BLANK  
 The prefix value on a PREFIX or LABEL statement has a  
 blank in the first byte.  
  
 DISASM0212E LABEL PREFIXES MUST BE 4 CHARACTERS OR LESS  
 The value on a PREFIX or LABEL statement is more than  
 4 characters long. Label prefixes must be 4 characters  
 or less.  
  
 DISASM0213E BASE REGISTER IS BLANK  
 The base register on a BASE or USING statement is blank.  
  
 DISASM0214E BASE REGISTER NAME EXCEEDS 3 CHARACTERS IN LENGTH  
 The base register on a BASE or USING statement is more  
 than 3 characters long. Valid values are 0-15, A-F,  
 R0-R15, and RA-RF.  
  
 DISASM0215E 'DATA' IS RESERVED FOR DATA AREA PREFIXES, CHOOSE  
 ANOTHER PREFIX  
  
 DISASM0216E INVALID DIGIT IN LINE/LINES/MAXLINES STATEMENT  
 The line count limit specified contains a non-numeric  
 digit.  
  
 DISASM0217E LINE COUNT VALUE ON A LINE/LINES/MAXLINES STATEMENT IS  
 TOO LONG OR CONTAINS EXTRANEOUS DATA  
 The line count value is 1 to 3 digits. Either the value  
 is more than 3 digits or there are other data in the  
 next 6 spaces.  
  
 DISASM0218E LINE COUNT VALUE ON A LINE/LINES/MAXLINES STATEMENT IS  
 BELOW MINIMUM ALLOWED  
 A minimum allowable line count value is specified in the  
 global options (member DISASMGB) when DISASM is genned.  
 The value you are requesting is below this minimum.  
  
 DISASM0219E LABEL PREFIX HAS ALREADY BEEN DEFINED, CHOOSE ONE OR THE  
 OTHER  
 DISASM only allows 1 label prefix value. The PREFIX or  
 LABEL statement has been found more than once. Choose  
 one of the prefixes and delete the second statement.  
  
 DISASM0220E LABEL PREFIX MUST BE DEFINED WHEN BASE REGISTER(S) ARE  
 DEFINED  
 When a BASE is defined, labels will be generated for  
 any points referenced within the CSECT. Labels require  
 a 1 to 4 character prefix specified by either the LABEL  
 or PREFIX statement.  
  
 DISASM0221E LABEL PREFIX NOT VALID UNLESS BASE REGISTER(S) ARE DEFINED  
 Labels for the CSECT cannot be generated unless a BASE  
 is defined.  
  
 DISASM0222E THIS AREA OVERLAPS DATA AT xxxx TO xxxx  
 An area being defined as a DATA area overlaps another  
 area that has already been defined at xxxx to xxxx.  
  
 DISASM0223E BEGINNING DISPLACEMENT IS LARGER THAN ENDING DISPLACEMENT  
 The beginning displacement is larger than the ending  
 displacement on a BASE statement.  
  
 DISASM0224E MALFORMED STATEMENT; REQUIRES LABEL NAME/OFFSET/TYPE  
 A user specified LABEL statement lacks recognizable  
 name or displacement fields in the expected columns.  
  
 DISASM0225E OVERLAPPING OR DUPLICATE OFFSET.  
 The displacement on a LABEL statement conflicts with a  
 prior specification.  
  
 DISASM0226W ASSEMBLER INPUT AFTER END CARD IGNORED.  
 The text after ASM START normally terminates with an  
 ASM END control card. DISASM found a user supplied END  
 card, and additional cards that will be ignored.  
  
 DISASM0233W UNSUPPORTED OR UNDEFINED VALUE.  
 The request is not recognized, or requests a function  
 not currently supported; operand is too short or long.  
  
 DISASM0234W OPCODE MODULE NOT FOUND  
 The specified name is not in STEPLIB/LINKLIB.  
  
 DISASM0236E SYNTAX ERROR IN STATEMENT - reason  
 The VERIFY or REPLACE card is in error.  
 "reason" qualifies the error:  
 MALFORMED - no valid text found in columns 10-72  
 BAD OFFSET - offset field is invalid  
 PARSE ERROR - fields not separated or terminated  
 BAD HEX TEXT - field too long or invalid  
 QUOTE ERROR - unpaired quotes, or text after quote.  
  
 DISASM0301E SPECIFIED CSECT NOT FOUND  
 The CSECT specified on the CSECT statement is not a  
 part of the specified MODULE or the module contains no  
 CSECTs.  
  
 DISASM0302E UNKNOWN RETURN CODE FROM POINT MACRO  
 DISASM received a return code from the POINT macro that  
 it could not interpret.  
  
 DISASM0303E DCB EODAD ROUTINE DRIVEN, END OF CONTROL RECORDS NOT  
 DETECTED  
 The EODAD routine of the DISMOD DCB should never be  
 driven. The last record of the requested CSECT should  
 be detected. This is an internal logic error provided  
 the input is a valid load module.  
  
 DISASM0304E MODULE DOES NOT EXIST IN DISMOD LIBRARY  
 The member specified on the MODULE statement was not  
 found in the DISMOD library.  
  
 DISASM0305E PERMANENT I/O ERROR  
 BLDL received return code 08 with reason code 01.  
  
 DISASM0306E INSUFFICIENT VIRTUAL STORAGE  
 BLDL received return code 08 with reason code 04.  
  
 DISASM0307E DEB NOT IN KEY 0-7  
 BLDL received return code 08 with reason code 08.  
  
 DISASM0308E DEVICE DOES NOT SUPPORT BLOCK IDENTIFIER  
 POINT received return code 04 with reason code 00.  
  
 DISASM0309E INCORRECT PARAMETER  
 POINT received return code 08 with reason code 01.  
  
 DISASM0310E INCORRECT DEB OR DEBCHK ERROR  
 POINT received return code 08 with reason code 02.  
  
 DISASM0311E ENVIRONMENTAL ERROR  
 POINT received return code 08 with reason code 03.  
  
 DISASM0312E UNSUCCESSFUL CALL TO ESTAE  
 POINT received return code 08 with reason code 0B.  
  
 DISASM0313E UNSUCCESSFUL GETMAIN  
 POINT received return code 08 with reason code 0C.  
  
 DISASM0314E INPUT/OUTPUT ERROR  
 POINT received return code 0C with reason code 00.  
  
 DISASM0318I CSECT SIZE ROUNDED UP BY n PADDING BYTES  
 The ROUND options was specified and resulted in padding.  
  
 DISASM0318I SYM RECORD HAS UNKNOWN FORMAT  
 The module included SYM data from TEST option, but the  
 length or type was incorrect.  
  
 DISASM0320W MAIN MEMBER nnnn NOT FOUND; WIDOWED ALIAS?  
 The member specified is an alias, but the corresponding  
 main member name was not found or had an I/O error.  
 Processing continues with the alias name.  
  
 DISASM0501E UNKNOWN RLD DATA TYPE  
 RLD data specify an item that is of an unknown type.  
 DISASM will have abended with code user 003.  
  
 DISASM0502E RLD DATA REMAINING WENT NEGATIVE  
 Internal logic error in module DISASM05.  
 DISASM will have abended with code user 004.  
  
 DISASM0503E RLD POINTER LARGER THAN NUMBER OF ESD ITEMS  
 Some RLD items have corresponding ESD items. When  
 the pointer value is non-zero, it is the relative number  
 of the ESD item it is associated with. In this case the  
 pointer value is larger than the number of ESD items.  
  
 DISASM0504E RLD DATA AT xxxx TO xxxx OVERLAPS A USER DEFINED DATA  
 AREA AT xxxx TO xxxx  
 The DATA area will be adjusted to resolve the conflict.  
  
 DISASM0505W THIS RLD ITEM REFERENCES AN ADCON PREVIOUSLY ENCOUNTERED  
 Possible input error; execution continues without this RLD  
  
 DISASM0506I RLD ITEM AT xxxx RESOLVING TO A DIFFERENT CSECT  
 Processing detected an ADCON pointing to an external  
 location; unlike a VCON, it may have a displacement.  
 E.g., V(XYZ) vs. EXTRN XYZ / DC A(XYZ+64)  
  
 DISASM0507W AT LEAST ONE DATA DEFINITION WAS MODIFIED TO RESOLVE AN  
 RLD CONFLICT.  
 May be issued with or in addition to DISASM0504E. A DATA  
 and RLD overlap were resolved by adjusting the DATA.  
  
 DISASM0701I ASSEMBLER RETURN CODE WAS xxxx  
 If the return code is greater than 4, DISASM will abort  
 processing. If you need to view the assembler output,  
 allocate the DISDEBUG DD, or SYSTERM DD for diagnostics.  
  
 DISASM0702I NO OPTIONAL ASSEMBLER INPUT, NO DSECTS AVAILABLE  
 The optional ASM START/END packet was omitted, therefore  
 USING statements for DSECT items will be ignored.  
  
 DISASM0703E ERROR ASSEMBLING DSECTS, CHECK ASSEMBLER OUTPUT IN DISDE BUG  
 The return code from the assembler was greater than 4.  
  
 DISASM0801E DSECT xxxxxxxx IS NOT PRESENT, BUT IS REFERENCED ON A  
 USING STATEMENT.  
 Either 1) the source for the DSECT was not provided  
 2) the PRINT of the assembler was turned off  
 3) the name is misspelled  
 If you are sure the name is correct and the source is  
 present, allocate DISDEBUG and check the assembler output.  
  
 DISASM0802E INSTRUCTIONS OVERLAP DATA DEFINED AT xxxx TO xxxx  
 Instruction lengths are determined by the opcodes. If  
 a portion of an instruction overlaps into a DATA area,  
 this is considered an error condition. Either increase  
 the size of the DATA area to include the entire instruc-  
 tion, or decrease the size of the DATA area so it does  
 not overlap the end of the instruction.  
  
 DISASM0803W aaaaaaaa DISPLACEMENT IN A bbbbbbbb BLOCK IS CHANGED  
 FROM xxxx TO yyyy TO REFERENCE AN INSTRUCTION BOUNDARY  
 aaaaaaaa is either BEGINNING or ENDING  
 bbbbbbbb is either BASE or USING  
 The beginning and ending displacements on BASE and USING  
 statements must reference instruction boundaries. This  
 is because the USING or DROP statements generated cannot  
 occur in the middle of an instruction. If a displacement  
 is found that is not on instruction boundary, it is forced  
 to the beginning of the preceding instruction.  
  
 DISASM0804E INVALID OPCODE DURING REFERENCE TABLE GENERATION  
 Internal logic error in DISASM08.  
  
 DISASM0805E LABEL llllllll IS NOT IN DSECT dddddddd AS REQUESTED ON  
 A USING STATEMENT  
 If you are sure the label is in the DSECT, allocate  
 DISDEBUG to verify the assembler output. But note that  
 labels defined on an EQU statement cannot always be  
 resolved.  
  
 DISASM0806E LABEL NOT FOUND IN DSECT DURING REFERENCE TABLE GENERATION  
 Internal logic error in DISASM08.  
  
 DISASM0808E OVERLAPPING DATA AREAS NOT DETECTED BY DISASM02  
 Interal logic error.  
  
 DISASM0809E ATTEMPT TO LOCATE AN INSTRUCTION ON AN ODD DISPLACEMENT  
 BOUNDARY  
 Internal error in DISASM08, DISASM will abend with code  
 user 005.  
  
 DISASM0901E DATA AREA OVERLAPS AN INSTRUCTION, SHOULD HAVE BEEN  
 DETECTED BY DISASM08.  
 Internal logic error. See explanation of DISASM0802E.  
  
 DISASM0902E INVALID OPCODE DETECTED  
 Internal error.  
  
 DISASM0903E DC WITH LENGTH = ZERO DETECTED  
 Internal error.  
  
 DISASM0904E ATTEMPT TO GENERATE AN INSTRUCTION ON AN ODD ADDRESS  
 BOUNDARY  
 Internal error.  
  
 DISASMDT01E \*\*\*\* DATASET OPEN UNSUCCESSFUL \*\*\*\*  
 The OPEN request for DISADATA or DISPUNCH failed. Check  
 the system or job log for additional messages.  
  
 DISASMLS01E \*\*\*\* DATASET OPEN UNSUCCESSFUL \*\*\*\*  
 The OPEN request for DISLIST or DISPUNCH failed. Check  
 the system or job log for additional messages.  
  
 DISASMLS02E \*\*\*\* NO PROCESSABLE SOURCE RECORDS FOUND \*\*\*\*  
 The input file is not in SYSPRINT format, or it may  
 be from an older, unsupported assembler. Program description

This disassembler attempts to re-create assembler source from object  
code. Label references are generated if the base registers for the  
CSECT or DSECTs are defined.  
  
CSECT labels will be generated for data referenced within the CSECT  
if BASE statements define the base register(s). References to  
instructions will cause labels to be generated that occur at the  
displacement of the first byte of the instruction. If the reference  
is not on the first byte of the instruction, the label will be referred  
to with a "+displacement". For example if a program is self-modifying  
and sets the length in a MVC instruction with STC that references the  
MVC instruction, the generated source might look like:  
  
 .  
 STC R1,MAIN002C+1  
 .  
 .  
 MAIN002C MVC DATAOUT(1),0(R2)  
 .  
  
Labels will also be generated for any ENTRY points defined in the ESD  
data for the CSECT.  
  
DSECT labels require the DSECT's source to be assembled, and the base  
defined via a USING statement. The program invokes the XF assembler  
(IFOX00) if the &MVSXA flag in DISOPT is off, other it invokes the  
high-level assembler (ASMA90). The assembler output listing  
is scanned to determine the DSECT names, label names, and the displace-  
ment to the label in the DSECT. In order for the disassembler to  
detect DSECTs and the labels, the assembler print must be on. This is  
forced by DISASM02 by generating a PRINT OPSYN COM statement immediately  
after the ASM START. Any DSECT referred to by a USING statement must be  
defined by having the assembler assemble it.  
  
Assembler source statements are delimited by ASM START/ASM END. Any  
statements between the ASM START and ASM END statements will be copied  
to the assembler input data set and appended onto the end of the source  
written to DISPUNCH. Any macros not in the assembler input stream must  
be available in a library in the SYSLIB concatenation. If the return  
code from the assembler is greater than 4, the disassembler discontinues  
any further processing. If there is no assembler input, the SYSPRINT,  
SYSIN, SYSUTn, SYSPUNCH, and SYSLIB DDs may be omitted.  
  
If the DISPUNCH DD is present, the generated code will also be  
written to this data set as 80-byte records suitable as input to an  
assembler.  
  
Object code is considered to be an instruction if:  
 1) it is not in a defined data area  
 2) it is on an even address boundary  
 3) it is a valid opcode  
 4) it does not overlap into a defined data area or RLD item.  
 5) it is followed by another valid opcode or is an unconditional  
 branch or SVC.  
 6) a string of blanks will not be interpreted as a series of STH  
 instructions.  
  
The default opcodes include the S/370 instructions set except for  
I/O and SSM. The OPTIONS statement may be used to request S/360 or  
S/390 instructions. User customized tables may instead be loaded with  
an OPCODE statement.  
  
OPCODE tables in the distribution include:  
DISOPAPP - subset of 370 instructions for typical application code.  
DISOPAP2 - as above, with aligned displacements only (e.g., ST 1,3(,2)  
 would be treated as data because it's not word aligned.  
DISOP36S - all 360 instructions, including I/O  
DISOP37B - all 370 instructions, including I/O, BAS, and BASR  
  
Extended mnemonics are used if possible for branch instructions.  
  
SVCs are interpreted if possible (SVC 0A is identified as GETMAIN).  
  
The DISDEBUG DD statement is optional. If present, many internal  
fields and data chains will be printed to aid in debugging problems  
with the disassembler. The assembler output is copied to DISDEBUG  
for diagnosing errors with the assembler input.  
  
As an alternative to disassembling a load module, the program will  
rebuild source from SYSADATA output (supplied via a DISADATA DD), and  
from SYSPRINT output pointed to by a DISLIST DD. Only DISPUNCH and  
DISPRINT DDs are meaningful for these two options, but other DDs may  
remain present and will be ignored.  
  
Note that DISADATA and DISLIST will also be ignored when they are  
defined as DD DUMMY or DSN=NULLFILE.  
  
DISASM will also process an object module as input. Only one CSECT  
will be processed, and it may be specified on a sequential data set  
or in one or more concatenated partitioned data sets, either using the  
DD name DISMOD. The following are valid forms:  
//DISMOD DD DISP=SHR,DSN=HERC01.ASMOBJ sequential  
//DISMOD DD DISP=SHR,DSN=HERC01.OBJLIB(modname) sequential  
//DISMOD DD DISP=SHR,DSN=HERC01.OBJLIB PDS  
// DD DISP=SHR,DSN=HERC01.OBJIB2 up to 16 PDSs  
The PDS form requires a MODULE specification on DISIN.  
  
 Please report problems to Gerhard Postpischil at  
 gerhard@valley.net