```
-- BindErrors.mesa; edited by Johnsson on August 30, 1978  9:24 PM

DIRECTORY
  BcdControlDefs: FROM "bcdcontroldefs",
  BcdDefs: FROM "bcddefs",
  BcdErrorDefs: FROM "bcderrordefs",
  BcdFileDefs: FROM "bcdfiledefs",
  BcdTabDefs: FROM "bcdtabdefs",
  BcdTreeDefs: FROM "bcdtreedefs",
  IODefs: FROM "iodefs",
  SegmentDefs: FROM "segmentdefs",
  StreamDefs: FROM "streamdefs",
  StringDefs: FROM "stringdefs",
  TableDefs: FROM "tabledefs",
  TimeDefs: FROM "timedefs";

DEFINITIONS FROM BcdTabDefs, BcdDefs, BcdErrorDefs, IODefs;

BindErrors: PROGRAM [data: BcdControlDefs.BinderData]
  IMPORTS BcdTabDefs, IODefs, StreamDefs, TableDefs, TimeDefs
  EXPORTS BcdControlDefs, BcdErrorDefs =
  BEGIN

  SubStringDescriptor: TYPE = StringDefs.SubStringDescriptor;
  SubString: TYPE = StringDefs.SubString;

  GetModule: PUBLIC SIGNAL RETURNS [errorMti: MTIndex] = CODE;
  GetInterface: PUBLIC SIGNAL RETURNS [errorExpi: EXPIndex] = CODE;
  GetSti: PUBLIC SIGNAL RETURNS [errorSti: STIndex] = CODE;

  PrintTextLine: PUBLIC PROCEDURE [i: CARDINAL] =
    BEGIN  OPEN StreamDefs, IODefs;
    start, lineIndex: StreamIndex;
    stream: StreamHandle ← data.sourcestream;
    char: CHARACTER;
    n: [1..100];
    start ← lineIndex ← NormalizeIndex[[page:0, byte:i]];
    FOR n IN [1..100] UNTIL lineIndex = [0, 0]
      DO
      lineIndex ← ModifyIndex[lineIndex, -1];
      SetIndex[stream, lineIndex];
      IF stream.get[stream] = CR THEN EXIT;
      start ← lineIndex;
      ENDLOOP;
    SetIndex[stream, start];
    FOR n IN [1..100]
      DO
      char ← stream.get[stream
        !StreamError => EXIT];
      SELECT char FROM
        CR, ControlZ => EXIT;
        ENDCASE => WriteChar[char];
      ENDLOOP;
    WriteChar[CR];
    RETURN
    END;

  WriteEOL: PROCEDURE =
    BEGIN
    IF ~NewLine[] THEN WriteChar[CR];
    RETURN
    END;

  Space: PROCEDURE =
    BEGIN
    WriteChar[SP];
    RETURN
    END;

  ErrorLog: PROCEDURE [class: ErrorClass] =
    BEGIN
    IF data.textIndex # BcdControlDefs.NullSourceIndex THEN
      BEGIN
      WriteString[", at "L];
      WriteName[data.currentname];
      WriteChar['[];
```

```
      WriteNumber[data.textIndex,[base:8, columns:1, unsigned:TRUE, zerofill: FALSE]];
      WriteChar[']];
      WriteEOL[];
      PrintTextLine[data.textIndex];
      END;
    IF class = error THEN
      BEGIN
      data.errors ← TRUE;
      data.errorcount ← data.errorcount + 1;
      END
    ELSE
      BEGIN
      data.warnings ← TRUE;
      data.warningcount ← data.warningcount + 1;
      END;
    IF data.debug THEN
      BEGIN
      WriteString["  Pause to debug"L];
      [] ← ReadChar[];
      WriteEOL[];
      END;
    RETURN
    END;

  Prefix: PROCEDURE [class: ErrorClass] =
    BEGIN
    WriteEOL[];
    IF class = warning THEN WriteString["Warning: "L];
    END;

  WriteNameBase: PROCEDURE[name: NameRecord, s: NameString] =
    BEGIN
    i: CARDINAL;
    offset, length: CARDINAL;
    offset ← name;
    length ← s.size[name];
    IF offset+length > s.string.length THEN RETURN;
    FOR i IN [offset..offset+MIN[length,100]) DO
      WriteChar[s.string.text[i]];
      ENDLOOP;
    RETURN
    END;

  WriteName: PROCEDURE[name: NameRecord] =
    BEGIN
    WriteNameBase[name, LOOPHOLE[TableDefs.TableBounds[sstype].base]];
    RETURN
    END;

  WriteVersion: PROCEDURE[v: POINTER TO VersionStamp] =
    BEGIN
    octal: NumberFormat = [base:8, zerofill:FALSE, unsigned:TRUE, columns:1];
    s: STRING ← [20];
    IF v.time = [0,0] THEN
      BEGIN WriteString["(Null Version)"L]; RETURN END;
    TimeDefs.AppendDayTime[s, TimeDefs.UnpackDT[v.time]];
    WriteChar['(];
    WriteString[s];
    Space[];
    WriteNumber[v.net,octal]; WriteChar['#];
    WriteNumber[v.host,octal]; WriteChar['#];
    IF v.zapped THEN WriteString[" Zapped!"L];
    WriteChar[')];
    RETURN
    END;

  WriteHti: PROCEDURE[hti: HTIndex] =
    BEGIN
    ss: SubStringDescriptor;
    i: CARDINAL;
    IF hti = HTNull THEN RETURN;
    SubStringForHash[@ss,hti];
    FOR i IN [ss.offset..ss.offset+ss.length) DO
      WriteChar[ss.base[i]];
      ENDLOOP;
    RETURN
```

```
    END;

  HtiForSti: PROCEDURE[sti: STIndex] RETURNS [HTIndex] =
    BEGIN
    IF sti = STNull THEN RETURN[HTNull];
    RETURN[(TableDefs.TableBounds[sttype].base+sti).hti]
    END;

  InterfaceName: PROCEDURE[expi: EXPIndex] RETURNS [NameRecord] =
    BEGIN
    IF expi = EXPNull THEN RETURN[NullName];
    RETURN[(TableDefs.TableBounds[exptype].base+expi).name]
    END;

  ModuleName: PROCEDURE[mti: MTIndex] RETURNS [NameRecord] =
    BEGIN
    IF mti = MTNull THEN RETURN[NullName];
    RETURN[(TableDefs.TableBounds[mttype].base+mti).name]
    END;

  Error: PUBLIC PROCEDURE [class: ErrorClass, s: STRING] =
    BEGIN
    Prefix[class];
    WriteString[s];
    ErrorLog[class];
    RETURN
    END;

  ErrorSti: PUBLIC PROCEDURE [class: ErrorClass, s: STRING, sti: STIndex] =
    BEGIN
    Prefix[class];
    WriteHti[HtiForSti[sti]]; Space[]; WriteString[s];
    ErrorLog[class];
    RETURN
    END;

  ErrorHti: PUBLIC PROCEDURE [class: ErrorClass, s: STRING, hti: HTIndex] =
    BEGIN
    sti: STIndex;
    Prefix[class];
    WriteHti[hti]; Space[]; WriteString[s];
    sti ← SIGNAL GetSti;
    IF sti # STNull THEN
      BEGIN
      WriteString[" (in "L];
      WriteHti[HtiForSti[sti]];
      WriteChar[')];
      END;
    ErrorLog[class];
    RETURN
    END;

  ErrorName: PUBLIC PROCEDURE [class: ErrorClass, s: STRING, name: NameRecord] =
    BEGIN
    Prefix[class];
    WriteName[name]; Space[]; WriteString[s];
    ErrorLog[class];
    RETURN
    END;

  ErrorItem: PUBLIC PROCEDURE [class: ErrorClass, s: STRING, i: CARDINAL] =
    BEGIN
    sti: STIndex;
    expi: EXPIndex;
    Prefix[class];
    WriteString["Item "L]; WriteDecimal[i];
    expi ← SIGNAL GetInterface;
    IF expi # EXPNull THEN
      BEGIN
      WriteString[" in interface "L];
      WriteName[InterfaceName[expi]];
      END;
    Space[]; WriteString[s];
    sti ← SIGNAL GetSti;
    IF sti # STNull THEN
      BEGIN
```

```
    WriteString[" (in "L];
    WriteHti[HtiForSti[sti]];
    WriteChar[')];
    END;
  ErrorLog[class];
  RETURN
  END;

ErrorModule: PUBLIC PROCEDURE [class: ErrorClass, s: STRING, mti: MTIndex] =
  BEGIN
  sti: STIndex;
  Prefix[class];
  WriteName[ModuleName[mti]]; Space[];
  WriteString[s];
  sti ← SIGNAL GetSti;
  IF sti # STNull THEN
    BEGIN
    WriteString[" (in "L];
    WriteHti[HtiForSti[sti]];
    WriteChar[')];
    END;
  ErrorLog[class];
  RETURN
  END;

ErrorInterface: PUBLIC PROCEDURE [class: ErrorClass, s: STRING, name: NameRecord, ep: CARDINAL] =
  BEGIN
  mti: MTIndex;
  Prefix[class];
  WriteString["Item "L]; WriteDecimal[ep];
  WriteString[" in interface "L];
  IF name = NullName THEN WriteString["(unknown)"L] ELSE WriteName[name];
  Space[]; WriteString[s];
  mti ← SIGNAL GetModule;
  IF mti # MTNull THEN
    BEGIN
    WriteString[" (in "L];
    WriteName[ModuleName[mti]];
    WriteChar[')];
    END;
  ErrorLog[class];
  RETURN
  END;

ErrorNameBase: PUBLIC PROCEDURE [
  class: ErrorClass, s: STRING, name: NameRecord, base: NameString] =
  BEGIN
  Prefix[class];
  WriteNameBase[name, base]; Space[];
  WriteString[s];
  ErrorLog[class];
  RETURN
  END;

ErrorFile: PUBLIC PROCEDURE [class: ErrorClass, s: STRING, fti: FTIndex] =
  BEGIN
  ftb: TableDefs.TableBase = TableDefs.TableBounds[fttype].base;
  Prefix[class];
  WriteName[(ftb+fti).name]; Space[];
  WriteString[s];
  ErrorLog[class];
  RETURN
  END;

Error2Files: PUBLIC PROCEDURE [class: ErrorClass, s: STRING, ft1, ft2: FTIndex] =
  BEGIN
  ftb: TableDefs.TableBase = TableDefs.TableBounds[fttype].base;
  Prefix[class];
  WriteName[(ftb+ft1).name]; WriteVersion[@(ftb+ft1).version];
  Space[]; WriteString[s]; Space[];
  WriteName[(ftb+ft2).name]; WriteVersion[@(ftb+ft2).version];
  ErrorLog[class];
  RETURN
  END;

END...
```