```
-- BootUtilities.Mesa  Edited by Sandman on April 19, 1978  5:42 PM

DIRECTORY
  BcdDefs: FROM "bcddefs",
  BootCacheDefs: FROM "BootCacheDefs",
  BootmesaDefs: FROM "BootmesaDefs",
  ControlDefs: FROM "controldefs",
  LoaderBcdUtilDefs: FROM "loaderbcdutildefs",
  StringDefs: FROM "stringdefs";

DEFINITIONS FROM BcdDefs;

BootUtilities: PROGRAM
  IMPORTS BootCacheDefs, StringDefs
  EXPORTS BootmesaDefs, LoaderBcdUtilDefs = PUBLIC
  BEGIN

  SubStringDescriptor: TYPE = StringDefs.SubStringDescriptor;
  currentCti: CTIndex;

  bcd: LoaderBcdUtilDefs.BcdBase;

  mtb: CARDINAL;
  ssb: BcdDefs.NameString;

  ModuleName: PROCEDURE [frame: ControlDefs.GlobalFrameHandle, name: STRING] =
    BEGIN
    bname: SubStringDescriptor;
    mth: MTHandle;
    mti: MTIndex;
    cth: CTHandle;
    ctb: CARDINAL = LOOPHOLE[bcd+bcd.ctOffset];
    fw: BootmesaDefs.FirstFrameWord = BootCacheDefs.READ[frame];
    gfi: GFTIndex = fw.gfi;
    FindInstance: PROCEDURE [nth: NTHandle, nti: NTIndex] RETURNS [BOOLEAN] =
      BEGIN
      WITH n:nth.item SELECT FROM
        module => IF n.mti # mti THEN RETURN[FALSE];
        ENDCASE => RETURN[FALSE];
      bname.offset ← nth.name;
      bname.length ← ssb.size[nth.name];
      StringDefs.AppendSubString[name, @bname];
      StringDefs.AppendChar[name, ':];
      RETURN[TRUE];
      END;
    FindModule: PROCEDURE [mth: MTHandle, mti: MTIndex] RETURNS [BOOLEAN] =
      BEGIN
      RETURN[mth.gfi = gfi];
      END;
    bname.base ← @ssb.string;
    [mth, mti] ← EnumerateModuleTable[bcd, FindModule];
    IF mth.namedinstance THEN [] ← EnumerateNameTable[bcd, FindInstance];
    IF mth.config # FIRST[CTIndex] THEN
      BEGIN
      cth ← ctb+mth.config;
      bname.offset ← cth.name;
      bname.length ← ssb.size[cth.name];
      StringDefs.AppendSubString[name, @bname];
      StringDefs.AppendChar[name, '>];
      END;
    bname.offset ← mth.name;
    bname.length ← ssb.size[mth.name];
    StringDefs.AppendSubString[name, @bname];
    RETURN
    END;

  Frame: PROCEDURE [name: STRING] RETURNS [ControlDefs.GlobalFrameHandle] =
    BEGIN
    ss: SubStringDescriptor;
    bname: SubStringDescriptor;
    mti: MTIndex;
    nti: NTIndex;
    CheckName: PROCEDURE [nth: NTHandle, nti: NTIndex] RETURNS [BOOLEAN] =
      BEGIN
      WITH n:nth.item SELECT FROM
        module => mti ← n.mti;
```

```
      ENDCASE => RETURN[FALSE];
    bname.offset ← nth.name;
    bname.length ← ssb.size[nth.name];
    RETURN[StringDefs.EqualSubStrings[@ss, @bname]];
    END;
  CheckModule: PROCEDURE [mth: MTHandle, mti: MTIndex] RETURNS [BOOLEAN] =
    BEGIN
    IF mth.config # currentCti THEN RETURN[FALSE];
    bname.offset ← mth.name;
    bname.length ← ssb.size[mth.name];
    RETURN[StringDefs.EqualSubStrings[@ss, @bname]];
    END;
  ss ← [base: name, offset: 0, length: name.length];
  bname.base ← @ssb.string;
  nti ← EnumerateNameTable[bcd, CheckName].nti;
  IF nti = NTNull THEN mti ← EnumerateModuleTable[bcd, CheckModule].mti;
  RETURN[IF mti = MTNull
    THEN ControlDefs.NullGlobalFrame
    ELSE BootCacheDefs.READ[@ControlDefs.GFT[(mtb+mti).gfi].frame]]
  END;

SetConfig: PROCEDURE [name: STRING] =
  BEGIN
  ss: SubStringDescriptor;
  bname: SubStringDescriptor;
  cti: CTIndex;
  CheckConfig: PROCEDURE [cth: CTHandle, cti: CTIndex] RETURNS [BOOLEAN] =
    BEGIN
    bname.offset ← cth.name;
    bname.length ← ssb.size[cth.name];
    RETURN[StringDefs.EqualSubStrings[@ss, @bname]];
    END;
  ss ← [base: name, offset: 0, length: name.length];
  bname.base ← @ssb.string;
  cti ← EnumerateConfigTable[bcd, CheckConfig].cti;
  IF cti = CTNull THEN RETURN;
  currentCti ← cti;
  RETURN
  END;

ResetConfig: PROCEDURE =
  BEGIN
  currentCti ← FIRST[CTIndex];
  RETURN
  END;

EnumerateConfigTable: PUBLIC PROCEDURE [bcd: LoaderBcdUtilDefs.BcdBase,
 proc: PROCEDURE [CTHandle, CTIndex] RETURNS [BOOLEAN]]
  RETURNS [cth: CTHandle, cti: CTIndex] =
  BEGIN
  ctb: CARDINAL = LOOPHOLE[bcd+bcd.ctOffset];
  FOR cti ← FIRST[CTIndex], cti + SIZE[CTRecord] UNTIL cti = bcd.ctLimit DO
    cth ← ctb+cti;
    IF proc[cth, cti] THEN RETURN [cth, cti];
    ENDLOOP;
  RETURN[NIL, CTNull];
  END;

EnumerateModuleTable: PUBLIC PROCEDURE [bcd: LoaderBcdUtilDefs.BcdBase,
 proc: PROCEDURE [MTHandle, MTIndex] RETURNS [BOOLEAN]]
  RETURNS [mth: MTHandle, mti: MTIndex] =
  BEGIN
  mtb: CARDINAL = LOOPHOLE[bcd+bcd.mtOffset];
  FOR mti ← FIRST[MTIndex], mti + SIZE[MTRecord] + (mtb+mti).frame.length
  UNTIL mti = bcd.mtLimit DO
    mth ← mtb+mti;
    IF proc[mth, mti] THEN RETURN [mth, mti];
    ENDLOOP;
  RETURN[NIL, MTNull];
  END;

EnumerateNameTable: PUBLIC PROCEDURE [bcd: LoaderBcdUtilDefs.BcdBase,
 proc: PUBLIC PROCEDURE [NTHandle, NTIndex] RETURNS [BOOLEAN]]
  RETURNS [nth: NTHandle, nti: NTIndex] =
  BEGIN
  ntb: CARDINAL = LOOPHOLE[bcd+bcd.ntOffset];
```

```
    FOR nti ← FIRST[NTIndex], nti + SIZE[NTRecord] UNTIL nti = bcd.ntLimit DO
      nth ← ntb+nti;
      IF proc[nth, nti] THEN RETURN [nth, nti];
      ENDLOOP;
    RETURN[NIL, NTNull];
    END;

  EnumerateSegTable: PUBLIC PROCEDURE [bcd: LoaderBcdUtilDefs.BcdBase,
   proc: PUBLIC PROCEDURE [SGHandle, SGIndex] RETURNS [BOOLEAN]]
    RETURNS [sgh: SGHandle, sgi: SGIndex] =
    BEGIN
    sgb: CARDINAL = LOOPHOLE[bcd+bcd.sgOffset];
    FOR sgi ← FIRST[SGIndex], sgi + SIZE[SGRecord] UNTIL sgi = bcd.sgLimit DO
      sgh ← sgb+sgi;
      IF proc[sgh, sgi] THEN RETURN [sgh, sgi];
      ENDLOOP;
    RETURN[NIL, SGNull];
    END;

  InitUtilities: PROCEDURE [b: LoaderBcdUtilDefs.BcdBase] =
    BEGIN
    bcd ← b;
    mtb ← LOOPHOLE[b+b.mtOffset];
    ssb ← LOOPHOLE[b+b.ssOffset];
    currentCti ← FIRST[CTIndex];
    RETURN
    END;

  END.
```