```
-- DebugTable.Mesa;
-- Edited by:
--           Sandman on March 12, 1978  11:29 AM
--           Barbara on May 15, 1978  10:21 AM

DIRECTORY
  BinaryDefs: FROM "binarydefs" USING [CommandTab],
  CommandDefs: FROM "commanddefs" USING [
    CommandName, ErrorCode, FTPCode, ICode, IDCode, SignalCode],
  CommandTabDefs: FROM "commandtabdefs" USING [DSRptr],
  IODefs: FROM "iodefs" USING [WriteChar],
  MiscDefs: FROM "miscdefs" USING [DestroyFakeModule],
  SegmentDefs: FROM "segmentdefs" USING [
    FileSegmentAddress, FileSegmentHandle, SwapIn, Unlock],
  StringDefs: FROM "stringdefs" USING [SubString, SubStringDescriptor];

DebugTable: PROGRAM
IMPORTS BinaryDefs, IODefs, MiscDefs, SegmentDefs
EXPORTS CommandDefs =

BEGIN

-- DSRp and desc.base are set by LockStringTable

DSRp: CommandTabDefs.DSRptr;
desc: StringDefs.SubStringDescriptor;
ss: StringDefs.SubString = @desc;

 -- string table management

tableState: {in, out} ← out;

stringTableSeg: SegmentDefs.FileSegmentHandle;
offset: CARDINAL;

OpenStringTable: PROCEDURE RETURNS [CommandTabDefs.DSRptr] =
  BEGIN OPEN SegmentDefs;
  SELECT tableState FROM
    out =>  SwapIn[stringTableSeg];
    ENDCASE;
  tableState ← in;
  RETURN [LOOPHOLE[FileSegmentAddress[stringTableSeg]+offset]];
  END;

CloseStringTable: PROCEDURE =
  BEGIN
  IF tableState = in THEN
    BEGIN
    SegmentDefs.Unlock[stringTableSeg];
    tableState ← out;
    END;
  RETURN
  END;

LockStringTable: PROCEDURE =
  BEGIN
  DSRp ← OpenStringTable[];
  ss.base ← LOOPHOLE[DSRp + DSRp.relativebase, STRING];
  RETURN
  END;

WriteCommandString: PUBLIC PROCEDURE [n: CommandDefs.CommandName] =
  BEGIN
  LockStringTable[];
  ss.offset ← DSRp.CommandStrings[n].offset;
  ss.length ← DSRp.CommandStrings[n].length;
  WriteSubString[ss];
  CloseStringTable[];
  RETURN
  END;

GetCommandString: PUBLIC PROCEDURE [n: CommandDefs.CommandName, command: StringDefs.SubString] =
  BEGIN
  LockStringTable[];
  command.base ← LOOPHOLE[DSRp + DSRp.relativebase, STRING];
  command.offset ← DSRp.CommandStrings[n].offset;
```

```
  command.length ← DSRp.CommandStrings[n].length;
  CloseStringTable[];
  RETURN
  END;

WriteSignalString: PUBLIC PROCEDURE [n: CommandDefs.SignalCode] =
  BEGIN
  LockStringTable[];
  ss.offset ← DSRp.SignalMessages[n].offset;
  ss.length ← DSRp.SignalMessages[n].length;
  WriteSubString[ss];
  CloseStringTable[];
  RETURN
  END;

WriteErrorString: PUBLIC PROCEDURE [n: CommandDefs.ErrorCode] =
  BEGIN
  LockStringTable[];
  ss.offset ← DSRp.ErrorMessages[n].offset;
  ss.length ← DSRp.ErrorMessages[n].length;
  WriteSubString[ss];
  CloseStringTable[];
  RETURN
  END;

WriteIDString: PUBLIC PROCEDURE [n: CommandDefs.IDCode] =
  BEGIN
  LockStringTable[];
  ss.offset ← DSRp.IDStrings[n].offset;
  ss.length ← DSRp.IDStrings[n].length;
  WriteSubString[ss];
  CloseStringTable[];
  RETURN
  END;

WriteFTPString: PUBLIC PROCEDURE [n: CommandDefs.FTPCode] =
  BEGIN
  LockStringTable[];
  ss.offset ← DSRp.FTPMessages[n].offset;
  ss.length ← DSRp.FTPMessages[n].length;
  WriteSubString[ss];
  CloseStringTable[];
  RETURN
  END;

WriteIString: PUBLIC PROCEDURE [n: CommandDefs.ICode] =
  BEGIN
  LockStringTable[];
  ss.offset ← DSRp.InterpretMessages[n].offset;
  ss.length ← DSRp.InterpretMessages[n].length;
  WriteSubString[ss];
  CloseStringTable[];
  RETURN
  END;

Init: PROCEDURE =
  BEGIN
  [stringTableSeg, offset] ←
   MiscDefs.DestroyFakeModule[LOOPHOLE[BinaryDefs.CommandTab]];
  RETURN
  END;

WriteSubString: PROCEDURE [s: StringDefs.SubString] =
  BEGIN -- type substring
  i: CARDINAL;
  FOR i IN [0..s.length)
    DO IODefs.WriteChar[s.base[s.offset + i]] ENDLOOP;
  RETURN
  END;

Init[];

END...
```