```
-- InternalNubControl.Mesa
-- Edited by:
--          Sandman on May 3, 1978  3:50 PM
--          Barbara on July 20, 1978  4:49 PM

DIRECTORY
  AltoDefs: FROM "altodefs" USING [PageSize],
  AltoFileDefs: FROM "altofiledefs" USING [FA],
  ControlDefs: FROM "controldefs" USING [
    GFT, GlobalFrameHandle, NullGlobalFrame],
  DebugData: FROM "debugdata" USING [textwindow, userwindow],
  DebugMiscDefs: FROM "debugmiscdefs" USING [
    DebugAbort, DebugInit, DebugProceed, Quit],
  FSPDefs: FROM "fspdefs" USING [AddToNewZone],
  ImageDefs: FROM "imagedefs" USING [
    AddFileRequest, FileRequest, ImageVersion, StopMesa],
  InternalNubDefs: FROM "internalnubdefs" USING [
    DebuggerDebuggerInstall, Install],
  IODefs: FROM "iodefs" USING [
    ControlX, CR, DEL, LineOverflow, NewLine, NUL, NumberFormat, ReadChar,
    ReadID, ReadNumber, Rubout, SP, WriteChar, WriteNumber, WriteOctal,
    WriteString],
  LoaderDefs: FROM "loaderdefs" USING [Load, Loader, New, VersionMismatch],
  LoaderUtilityDefs: FROM "loaderutilitydefs" USING [FileNotFound],
  MiscDefs: FROM "miscdefs",
  Mopcodes: FROM "mopcodes" USING [zKFCB],
  ProcessDefs: FROM "processdefs" USING [Aborted],
  RectangleDefs: FROM "rectangledefs" USING [
    BMHandle, CreateRectangle, GetDefaultBitmap, ReallocateBitmap,
    RectanglesB, Rptr],
  SDDefs: FROM "sddefs" USING [sCallDebugger],
  SegmentDefs: FROM "segmentdefs" USING [
    DefaultVersion, FileSegmentHandle, InsufficientVM, LockFile, NewFile,
    Read, ReleaseFile, UnlockFile],
  StreamDefs: FROM "streamdefs" USING [
    Append, CreateByteStream, CreateDisplayStream, GetDefaultKey, GetFA,
    JumpToFA, KeyStreams, Read, StreamHandle, Write],
  StringDefs: FROM "stringdefs" USING [
    AppendChar, AppendString, EquivalentString, InvalidNumber],
  SystemDefs: FROM "systemdefs" USING [AllocatePages, FreePages, HeapZone],
  TimeDefs: FROM "timedefs" USING [AppendDayTime, UnpackDT],
  WindowDefs: FROM "windowdefs" USING [
    CreateDisplayWindow, GetCurrentDisplayWindow, RepaintDisplayWindows,
    SetCurrentDisplayWindow, WindowHandle, WindowsB];

InternalNubControl: PROGRAM
  IMPORTS DDptr: DebugData, DebugMiscDefs, FSPDefs, ImageDefs,
    InternalNubDefs, IODefs, LoaderDefs, LoaderUtilityDefs, ProcessDefs,
    RectangleDefs, SegmentDefs, StreamDefs, StringDefs, TimeDefs, WindowDefs,
    SystemDefs
  EXPORTS InternalNubDefs =

BEGIN

-- System Signals are converted to these to prevent NubCommand
-- from catching user generated signals
Delete: SIGNAL = CODE;                          -- nee Rubout
StringTooLong: ERROR = CODE;                    -- nee LineOverflow
BadFile: ERROR [badname: STRING] = CODE;        -- nee FileNameError
BadNumber: ERROR = CODE;                        -- nee InvalidNumber
BadVersion: SIGNAL [badname: STRING] = CODE;    -- nee VersionMismatch

CallDebugger: PROCEDURE =
  MACHINE CODE BEGIN Mopcodes.zKFCB, SDDefs.sCallDebugger END;

level: CARDINAL ← 0;
installed: BOOLEAN ← FALSE;
useCommandLine, skipImage: BOOLEAN ← TRUE;
defaultframe: GlobalFrameHandle ← ControlDefs.NullGlobalFrame;

GlobalFrameHandle: TYPE = ControlDefs.GlobalFrameHandle;

DoCommand: PROCEDURE =
  BEGIN OPEN ControlDefs;
  f: GlobalFrameHandle;
  p: PROCESS;
```

```
  IF skipImage THEN SkipImage[];
  WritePrompt[];
  f ← IF useCommandLine THEN LoadSystem[] ELSE Command[];
  IF f # NullGlobalFrame THEN
    BEGIN p ← FORK StartModule[LOOPHOLE[f]]; JOIN p; END;
  IF install THEN
    BEGIN WritePrompt[]; WriteCommand[install]; SIGNAL Install; END;
  IF internalInstall THEN
    BEGIN WritePrompt[]; WriteCommand[xstall]; SIGNAL InternalInstall; END;
  END;

StartModule: PROCEDURE [f: PROGRAM] =
  BEGIN
  BEGIN
  ENABLE ProcessDefs.Aborted, DebugMiscDefs.DebugAbort => CONTINUE;
  IF ~LOOPHOLE[f, GlobalFrameHandle].started THEN START f ELSE RESTART f;
  END;
  RETURN
  END;

WritePrompt: PROCEDURE =
  BEGIN
  WriteEOL[];
  THROUGH [1..level] DO IODefs.WriteChar['/]; ENDLOOP;
  RETURN
  END;

Commands: TYPE = {new, start, read, write, debug, install, xstall,
  proceed, quit, octal, bitmap};

WriteCommand: PROCEDURE [command: Commands] =
  BEGIN
  IODefs.WriteString[SELECT command FROM
    new => "New"L,
    start => "Start"L,
    read => "Read"L,
    write => "Write"L,
    debug => "Debug"L,
    install => "Install"L,
    xstall => "↑Xstall"L,
    proceed => "Proceed"L,
    quit => "Quit"L,
    octal => "Octal "L,
    bitmap => "Bitmap pages: "L,
    ENDCASE => "?"L];
  END;

Command: PROCEDURE RETURNS [GlobalFrameHandle] =
  BEGIN OPEN IODefs;
  c: CHARACTER;
  f: GlobalFrameHandle;
  SELECT c←ReadChar[] FROM
    'N,'n =>
      BEGIN
      WriteCommand[new];
      f ← LoadModule[];
      IF f # ControlDefs.NullGlobalFrame THEN defaultframe ← f;
      END;

    'S,'s =>
      BEGIN
      WriteCommand[start];
      f ← getgframe[];
      WriteEOL[];
      RETURN[f];
      END;

    'D,'d =>
      BEGIN WriteCommand[debug]; confirm[]; CallDebugger[]; END;

    'Q,'q =>
      BEGIN WriteCommand[quit]; confirm[]; SIGNAL DebugMiscDefs.Quit; END;

    'P,'p =>
      BEGIN
      WriteCommand[proceed];
```

```
      confirm[];
      SIGNAL DebugMiscDefs.DebugProceed;
      END;

    'O,'o =>
      BEGIN
      WriteCommand[octal];
      SELECT ReadChar[] FROM
      'R, 'r => BEGIN WriteCommand[read]; OctalRead[]; END;
      'W, 'w => BEGIN WriteCommand[write]; OctalWrite[]; END;
      ENDCASE => WriteChar['?];
      END;

    'I,'i =>
      BEGIN WriteCommand[install]; confirm[]; SIGNAL Install END;

    ControlX =>
      BEGIN
      WriteCommand[xstall];
      confirm[];
      SIGNAL InternalInstall;
      END;

    'B,'b =>
      BEGIN OPEN RectangleDefs;
      pages: CARDINAL;
      b: BMHandle = GetDefaultBitmap[];
      WriteCommand[bitmap];
      pages ← ReadNumber[BMPages, 10 !
        IODefs.Rubout => ERROR Delete;
        StringDefs.InvalidNumber => ERROR BadNumber;
        IODefs.LineOverflow => ERROR StringTooLong];
      pages ← MIN[pages, 90D];
      ReallocateBitmap[b, pages, BMWordsPerLine
        ! SegmentDefs.InsufficientVM => BEGIN pages ← pages - 2; RETRY END];
      WindowDefs.RepaintDisplayWindows[b];
      BMPages ← pages;
      END;

    CR,SP => NULL;

    '? =>
      BEGIN
      WriteString["Commands are: "L];
      WriteCommand[new];
      WriteChar[',']; WriteChar[' ']; WriteCommand[start];
      WriteChar[',']; WriteChar[' ']; WriteCommand[octal]; WriteCommand[read];
      WriteChar[',']; WriteChar[' ']; WriteCommand[octal]; WriteCommand[write];
      WriteChar[',']; WriteChar[' ']; WriteCommand[proceed];
      WriteChar[',']; WriteChar[' ']; WriteCommand[quit];
      WriteChar[',']; WriteChar[' ']; WriteCommand[debug];
      WriteChar[',']; WriteChar[' ']; WriteCommand[install];
      WriteChar[',']; WriteChar[' ']; WriteCommand[bitmap];
      END;
    ENDCASE =>
      BEGIN WriteChar[c]; WriteChar['?]; END;
  RETURN[ControlDefs.NullGlobalFrame]
  END;

InstallDebugger: PROCEDURE [mode: {internal, external}] =
  BEGIN
  IF installed THEN WriteError[installed]
  ELSE
    BEGIN
    IF useCommandLine THEN CleanUpCommandLine[];
    installed ← TRUE;
    InitSwitches[];
    SELECT mode FROM
      internal => InternalNubDefs.DebuggerDebuggerInstall[];
      ENDCASE => InternalNubDefs.Install[];
    END;
  END;

LoadModule: PROCEDURE RETURNS [g: GlobalFrameHandle] =
  BEGIN
  name: STRING ← [40];
```

```
  ext: STRING ← [10];
  switches: STRING ← [10];
  i: CARDINAL ← 0;
  get: PROCEDURE RETURNS [c: CHARACTER] =
    BEGIN
    IF i = idstring.length THEN RETURN[IODefs.NUL];
    c ← idstring[i];
    i ← i + 1;
    RETURN[c]
    END;
  IODefs.WriteString[" Filename: "L];
  IODefs.ReadID[idstring ! IODefs.Rubout => ERROR Delete;
    IODefs.LineOverflow => ERROR StringTooLong];
  GetToken[get, name, ext, switches];
  IF ext.length = 0 THEN StringDefs.AppendString[ext, "bcd"L];
  StringDefs.AppendChar[name, '.];
  StringDefs.AppendString[name, ext];
  ProcessSwitches[switches];
  RETURN[LoadNew[name, framelinks]];
  END;

LoadNew: PROCEDURE [name: STRING, framelinks: BOOLEAN]
  RETURNS [g: GlobalFrameHandle] =
  BEGIN OPEN LoaderDefs;
  bcd: SegmentDefs.FileSegmentHandle;
  bcd ← LoaderDefs.Load[name
    ! BadFile, UNWIND => NULL; ANY => ERROR BadFile[name]];
  g ← LoaderDefs.New[bcd, framelinks, FALSE
    ! BadFile, BadVersion, UNWIND => NULL;
      LoaderDefs.VersionMismatch =>
        BEGIN SIGNAL BadVersion[name]; RESUME END;
      LoaderUtilityDefs.FileNotFound => ERROR BadFile[name];
      ANY => ERROR BadFile[name]];
  IODefs.WriteString[" -- "L];
  IODefs.WriteOctal[g];
  WriteEOL[];
  RETURN
  END;

getgframe: PROCEDURE RETURNS [f: GlobalFrameHandle] =
  BEGIN OPEN ControlDefs, IODefs;
  WriteString[" Global frame: "L];
  f ← ReadNumber[defaultframe, 8 !
    IODefs.Rubout => ERROR Delete;
    StringDefs.InvalidNumber => ERROR BadNumber;
    IODefs.LineOverflow => ERROR StringTooLong];
  IF ControlDefs.GFT[f.gfi].frame # f THEN
    BEGIN
    WriteString[" not a global frame: "L];
    SIGNAL DebugMiscDefs.DebugAbort
    END;
  defaultframe ← f;
  RETURN
  END;

idstring: STRING ← [40];

confirm: PROCEDURE =
  BEGIN OPEN IODefs;
  WriteString[" [confirm] "L];
  DO
    SELECT ReadChar[] FROM
      CR => EXIT;
      DEL => SIGNAL Delete;
      ENDCASE => WriteChar['?];
    ENDLOOP;
  WriteChar[CR];
  RETURN
  END;

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

```
comcmRequest: short ImageDefs.FileRequest ← [
  body: short[fill:, name: "Com.Cm."],
  file: NIL,
  access: SegmentDefs.Read,
  link: ];

Done: SIGNAL = CODE;
InternalInstall: SIGNAL = CODE;
Install: SIGNAL = CODE;

install, internalInstall, start, command, framelinks: BOOLEAN;

ProcessSwitches: PROCEDURE [s: STRING] =
  BEGIN
  i: CARDINAL;
  inverse: BOOLEAN ← FALSE;
  FOR i IN [0..s.length) DO
    SELECT s[i] FROM
      'c, 'C => BEGIN inverse ← FALSE; command ← TRUE END;
      'i, 'I => BEGIN inverse ← FALSE; install ← TRUE END;
      'x, 'X => BEGIN inverse ← FALSE; internalInstall ← TRUE END;
      's, 'S => IF inverse THEN inverse ← start ← FALSE ELSE start ← TRUE;
      'l, 'L => BEGIN inverse ← FALSE; framelinks ← FALSE END;
      '- => inverse ← TRUE;
      ENDCASE => inverse ← FALSE;
    ENDLOOP;
  END;

InitSwitches: PROCEDURE =
  BEGIN
  command ← internalInstall ← install ← FALSE;
  framelinks ← start ← TRUE;
  END;

GetToken: PROCEDURE [
  get: PROCEDURE RETURNS [CHARACTER], token, ext, switches: STRING] =
  BEGIN OPEN IODefs;
  s: STRING;
  c: CHARACTER;
  token.length ← ext.length ← switches.length ← 0;
  s ← token;
  WHILE (c ← get[]) # NUL DO
    SELECT c FROM
      SP, CR =>
        IF token.length # 0 OR ext.length # 0 OR switches.length # 0 THEN RETURN;
      '. => s ← ext;
      '/ => s ← switches;
      ENDCASE => StringDefs.AppendChar[s, c];
    ENDLOOP;
  RETURN
  END;

fa: AltoFileDefs.FA;

CleanUpCommandLine: PROCEDURE =
  BEGIN
  SegmentDefs.UnlockFile[comcmRequest.file];
  SegmentDefs.ReleaseFile[comcmRequest.file];
  comcmRequest.file ← NIL;
  useCommandLine ← FALSE;
  RETURN
  END;

LoadSystem: PROCEDURE RETURNS [user: GlobalFrameHandle] =
  BEGIN
  BEGIN
    ENABLE UNWIND => CleanUpCommandLine[];
    InitSwitches[];
    user ← LoadUser[@fa ! Done => GOTO done];
    IF ~start THEN user ← ControlDefs.NullGlobalFrame;
  EXITS
    done =>
      BEGIN user ← ControlDefs.NullGlobalFrame; CleanUpCommandLine[] END;
  END;
  RETURN
  END;
```

```
LoadUser: PROCEDURE [fa: POINTER TO AltoFileDefs.FA]
  RETURNS [user: GlobalFrameHandle] =
  BEGIN OPEN IODefs, StreamDefs;
  com: StreamHandle;
  name: STRING ← [40];
  ext: STRING ← [10];
  switches: STRING ← [10];
  get: PROCEDURE RETURNS [c: CHARACTER] =
    BEGIN
    IF com.endof[com] THEN RETURN[NUL];
    RETURN[com.get[com]];
    END;
  com ← CreateByteStream[comcmRequest.file, Read];
  BEGIN
    StreamDefs.JumpToFA[com, fa ! ANY => GO TO finished];
    GetToken[get, name, ext, switches];
    IF name.length = 0 THEN GO TO finished;
    IF ext.length = 0 THEN ext ← "bcd"L;
    StreamDefs.GetFA[com, fa];
    com.destroy[com];
    ProcessSwitches[switches];
    IF command THEN
      BEGIN
      ProcessSwitches[name];
      user ← ControlDefs.NullGlobalFrame;
      END
    ELSE
      BEGIN
      WriteString[name];
      StringDefs.AppendChar[name, '.];
      StringDefs.AppendString[name, ext];
      user ← LoadNew[name, framelinks];
      END;
  EXITS
    finished => BEGIN com.destroy[com]; SIGNAL Done; END;
  END;
  RETURN
  END;

SkipImage: PROCEDURE =
  BEGIN OPEN IODefs, StreamDefs;
  com: StreamHandle;
  name: STRING ← [40];
  ext: STRING ← [10];
  switches: STRING ← [10];
  get: PROCEDURE RETURNS [c: CHARACTER] =
    BEGIN
    IF com.endof[com] THEN RETURN[NUL];
    RETURN[com.get[com]];
    END;
  skipImage ← FALSE;
  IF comcmRequest.file = NIL THEN BEGIN useCommandLine ← FALSE; RETURN END;
  SegmentDefs.LockFile[comcmRequest.file];
  InitSwitches[];
  com ← CreateByteStream[comcmRequest.file, Read];
  StreamDefs.GetFA[com, @fa];
  GetToken[get, name, ext, switches];
  IF StringDefs.EquivalentString[ext, "image"L] THEN
  StreamDefs.GetFA[com, @fa];
  GetToken[get, name, ext, ext];
  IF name.length # 0 THEN StreamDefs.JumpToFA[com, @fa]
  ELSE useCommandLine ← FALSE;
  com.destroy[com];
  ProcessSwitches[switches];
  END;

WriteHerald: PROCEDURE [herald: STRING] =
  BEGIN
  h: STRING ← "Alto/Mesa Debugger 4.1 of "L;
  time: STRING ← [18];
  StringDefs.AppendString[herald, h];
  TimeDefs.AppendDayTime[herald,
    TimeDefs.UnpackDT[ImageDefs.ImageVersion[].time]];
  herald.length ← herald.length - 3;
  IODefs.WriteString[herald];
```

```
  TimeDefs.AppendDayTime[time,TimeDefs.UnpackDT[[0,0]]];
  time.length ← time.length - 3;
  WriteEOL[]; IODefs.WriteString[time];
  END;

ErrorName: TYPE = {XXX, file, number, toolong, exit, diffver, installed};

WriteError: PROCEDURE [error: ErrorName] =
  BEGIN
  IODefs.WriteString[SELECT error FROM
    file => "!File: "L,
    number => "!Number"L,
    toolong => "!String too long"L,
    exit => "Exiting top level Debugger!"L,
    diffver => " referenced in different versions"L,
    installed => " Already Installed!"L,
    ENDCASE => " XXX"L]
  END;

InternalDebugCommand: PUBLIC PROCEDURE =
  BEGIN
  level ← level + 1;
  DO
    BEGIN OPEN IODefs;
    DoCommand[ !
      UNWIND => level ← level - 1;
      Rubout, Delete => BEGIN WriteError[XXX]; CONTINUE END;
      ProcessDefs.Aborted => CONTINUE;
      DebugMiscDefs.DebugAbort => CONTINUE;
      BadFile =>
        BEGIN
        WriteEOL[];
        WriteError[file];
        WriteString[badname];
        CONTINUE
        END;
      BadNumber =>
        BEGIN WriteEOL[]; WriteError[number]; CONTINUE END;
      LineOverflow, StringTooLong =>
        BEGIN WriteEOL[]; WriteError[toolong]; CONTINUE END;
      BadVersion --[badname: STRING] -->
        BEGIN
        WriteEOL[];
        WriteError[file];
        WriteString[badname];
        WriteError[diffver];
        RESUME
        END;
      DebugMiscDefs.Quit =>
        BEGIN
        IF level # 1 THEN GOTO abort;
        WriteError[exit];
        confirm[];
        ImageDefs.StopMesa[]
        END;
      DebugMiscDefs.DebugProceed => EXIT;
      Install => GOTO install;
      InternalInstall => GOTO xstall];
    EXITS
      abort =>
        BEGIN
        level ← level - 1;
        SIGNAL DebugMiscDefs.DebugAbort;
        END;
      dontquit, continue => NULL;
      install => InstallDebugger[external];
      xstall => InstallDebugger[internal];
    END;
    ENDLOOP;
  level ← level - 1;
  RETURN
  END;

rcount, waddress, raddress: CARDINAL ← 0;

OctalRead: PROCEDURE =
```

```
  BEGIN OPEN IODefs;
  j:  CARDINAL;
  n: INTEGER;
  i: INTEGER ← -1;
  WriteString[" @: "L];
  waddress ← raddress ← ReadNumber[raddress, 8 !
    IODefs.Rubout => ERROR Delete;
    StringDefs.InvalidNumber => ERROR BadNumber;
    IODefs.LineOverflow => ERROR StringTooLong];
  WriteString[" n(10): "L]; rcount ← ReadNumber[rcount, 10 !
    IODefs.Rubout => ERROR Delete;
    StringDefs.InvalidNumber => ERROR BadNumber;
    IODefs.LineOverflow => ERROR StringTooLong];
  FOR j IN [0..rcount)
    DO
    IF (i ← i+1) MOD 8 = 0 THEN
      BEGIN
      WriteEOL[];
      WriteOctal[raddress + j];
      WriteChar['/];
      END;
    WriteChar[' ];
    WriteNumber[n ← MEMORY[raddress + j], NumberFormat[8,FALSE,TRUE,6]];
    WriteChar[IF n ~IN[0..7] THEN 'B ELSE ' ];
    ENDLOOP;
  raddress ← raddress + rcount;
  END;

OctalWrite: PROCEDURE =
  BEGIN OPEN IODefs;
  WriteString[" @: "L];
  waddress ← ReadNumber[waddress, 8 !
    IODefs.Rubout => ERROR Delete;
    StringDefs.InvalidNumber => ERROR BadNumber;
    IODefs.LineOverflow => ERROR StringTooLong];
  WriteChar[SP]; WriteChar['←]; WriteChar[SP];
  MEMORY[waddress] ← ReadNumber[MEMORY[waddress], 8 !
    IODefs.Rubout => ERROR Delete;
    StringDefs.InvalidNumber => ERROR BadNumber;
    IODefs.LineOverflow => ERROR StringTooLong];
  waddress ← waddress + 1;
  RETURN
  END;

InitWindows: PROCEDURE =
  BEGIN OPEN SegmentDefs, StreamDefs, WindowDefs, RectangleDefs;
  default: WindowHandle = GetCurrentDisplayWindow[];
  usertypescript: STRING = "Mesa.Typescript"L;
  rect: Rptr;

  rect ← CreateRectangle[GetDefaultBitmap[],40,512,0,256];
  DDptr.textwindow ← CreateDisplayWindow[file, rect,
    CreateDisplayStream[rect], GetDefaultKey[], NIL];
  ReleaseFile[NewFile[usertypescript,Read+Write+Append,DefaultVersion]];
  rect ← CreateRectangle[GetDefaultBitmap[],20,512,20,256];
  DDptr.userwindow ← CreateDisplayWindow[file, rect,
    CreateDisplayStream[rect], GetDefaultKey[], usertypescript];
  SetCurrentDisplayWindow[default];
  RETURN
  END;

herald: STRING ← [44];

BMPages: CARDINAL ← 56;
BMWordsPerLine: CARDINAL = 32;

-- Main body

START StreamDefs.KeyStreams; -- Start keyboard process
START RectangleDefs.RectanglesB[BMPages, BMWordsPerLine];
START WindowDefs.WindowsB["Debug.TypeScript."];
ImageDefs.AddFileRequest[@comcmRequest];

STOP;

FSPDefs.AddToNewZone[z: SystemDefs.HeapZone[], length: AltoDefs.PageSize*4,
```

```
  base: SystemDefs.AllocatePages[4], deallocate: SystemDefs.FreePages];
RESTART RectangleDefs.RectanglesB;
RESTART WindowDefs.WindowsB;
START LoaderDefs.Loader;
START DDptr;

WriteHerald[herald];
START DebugMiscDefs.DebugInit[herald];

InitWindows[];
DO InternalDebugCommand[ ! DebugMiscDefs.DebugAbort => CONTINUE] ENDLOOP;

END...
```