```
-- ControlDefs.Mesa  Edited by Sandman on August 23, 1977  9:34 PM

DIRECTORY
  Mopcodes: FROM "mopcodes",
  AltoDefs: FROM "altodefs",
  SegmentDefs: FROM "segmentdefs";

DEFINITIONS FROM AltoDefs;

ControlDefs: DEFINITIONS =
  BEGIN

  -- control link definitions

  ControlLinkTag: TYPE = [frametag .. unboundtag];
    frametag: CARDINAL = 0;
    procdesctag: CARDINAL = 1;
    signaldesctag: CARDINAL = procdesctag;
    indirecttag: CARDINAL = 2;
    unboundtag: CARDINAL = 3;

  ExtendedControlLinkTag: TYPE = {frame, procDesc, indirect, uninitialized, representation};

  ControlLink: TYPE = MACHINE DEPENDENT RECORD [
    SELECT COMPUTED ExtendedControlLinkTag FROM
      frame => [
        frameLink: FrameHandle],
      procDesc => [
        procLink: UNSPECIFIED],
      indirect => [
        indirectLink: POINTER TO ControlLink],
      uninitialized => [
        info: UnboundDesc],
      representation => [
        data: [0..377777B],
        type: ControlLinkTag],
      ENDCASE];

  GetReturnLink: MACHINE CODE RETURNS [ControlLink] = INLINE [Mopcodes.zLLB, returnOffset];
  GetReturnFrame: MACHINE CODE RETURNS [FrameHandle] = INLINE [Mopcodes.zLLB, returnOffset];

  FrameLink: TYPE = MACHINE DEPENDENT RECORD [
    frame: FrameHandle];

  ProcDesc: TYPE = MACHINE DEPENDENT RECORD [
    gftindex: GFTIndex,
    epoffset: [0..eprange),
    tag: ControlLinkTag];

  SignalDesc: TYPE = ProcDesc;

  IndirectLink: TYPE = MACHINE DEPENDENT RECORD [
    link: POINTER TO ControlLink];

  UnboundDesc: TYPE = MACHINE DEPENDENT RECORD [
    gftindex: GFTIndex,
    descindex: [0..eprange),
    tag: ControlLinkTag];

  TrapLink: ControlLink = ControlLink [
    representation[data:0, type:frametag]];

  PortTag: TYPE = {clink, plink};

  PortHandle: TYPE = POINTER TO Port;

  Port: TYPE = MACHINE DEPENDENT RECORD [
    pendingFrame: ControlLink,
    destPort: SELECT COMPUTED PortTag FROM
      clink => [
        link: ControlLink],
      plink => [
        port: PortHandle],
      ENDCASE];
```

```
-- frame definitions

FrameClass: TYPE = {global, local, signal, catch};

FrameBase: TYPE = MACHINE DEPENDENT RECORD [
  accesslink: GlobalFrameHandle,
  pc: WordPC,
  returnlink: ControlLink,
  extensions: SELECT COMPUTED FrameClass FROM
    global => [
      codebase: POINTER,
      gftindex: ProcDesc,
      ownerlink: GlobalFrameHandle,
      bindentry, bindlink: GlobalFrameHandle,
      codesegment: SegmentDefs.FileSegmentHandle,
      symbolsegment: SegmentDefs.FileSegmentHandle],
    local => [
      unused: UNSPECIFIED],
    signal => [
      mark: BOOLEAN,
      unused: [0..77777B]],
    catch => [
      unused: UNSPECIFIED,
      staticlink: FrameHandle],
    ENDCASE];

FrameHandle: TYPE = POINTER TO FrameBase;
NULLFrame: GlobalFrameHandle = LOOPHOLE[0];
GlobalFrameHandle: TYPE = POINTER TO global FrameBase;

Alloc: MACHINE CODE [CARDINAL] RETURNS [POINTER] = INLINE[Mopcodes.zALLOC];
Free: MACHINE CODE [POINTER] = INLINE[Mopcodes.zFREE];


-- The following offsets are used by the compiler and MUST
-- reflect the field offsets in the definition of FrameBase

accessOffset: CARDINAL = 0;
pcOffset: CARDINAL = 1;
returnOffset: CARDINAL = 2;
codebaseOffset: CARDINAL = 3;
gftiOffset: CARDINAL = 4;
ownerOffset: CARDINAL = 5;
bindentryOffset: CARDINAL = 6;
bindlinkOffset: CARDINAL = 7;
codesegmentOffset: CARDINAL = 8;
symbolsegmentOffset: CARDINAL = 9;

-- efficiently addressable portion of frames

globalbase: CARDINAL = 10;
globalslots: CARDINAL = 8;
procbase: CARDINAL = globalbase + globalslots;
localbase: CARDINAL = 4;
localslots: CARDINAL = 8;
framelink: CARDINAL = localbase;
lprocslots, procslots: CARDINAL = 16;

-- code segments

WordPC: TYPE = RECORD [INTEGER];
BytePC: TYPE = RECORD [CARDINAL];

InstWord: TYPE = MACHINE DEPENDENT RECORD [
  oddbyte, evenbyte: BYTE];

fielddescriptor: TYPE = MACHINE DEPENDENT RECORD [
  posn, size: [0..17B]];

epmin: CARDINAL = 1;  -- lower bound (module dependent)
eprange: CARDINAL = 32;

CsegPrefix: TYPE = MACHINE DEPENDENT RECORD [
  swapinfo: WORD,
  ngfi: [1..4],
  linkbase: [globalbase..globalbase+16),
```

```
    nlinks: [0..1777B],
    EntryVector: ARRAY [0..epmin) OF EntryVectorItem];

  EntryVectorItem: TYPE = MACHINE DEPENDENT RECORD [
    initialpc: WordPC,
    defaults: BOOLEAN,
    nparams: [0..177B],
    framesize: [0..377B]];

  MainBodyIndex: CARDINAL = 0;


  -- Global Frame Table definitions

  GFTItem: TYPE = MACHINE DEPENDENT RECORD [
    frame: GlobalFrameHandle,
    epbase: CARDINAL];

  GFTIndex: TYPE = [0..777B];
  GFTNull, NullGFTIndex: GFTIndex = LAST[GFTIndex];
  NULLEpBase: CARDINAL = LAST[CARDINAL];

  MaxGFTLength: CARDINAL = (LAST[GFTIndex]+1)*SIZE[GFTItem]*SIZE[GFTItem];


  -- system frame allocation vector

  maxallocslot: CARDINAL = 19;
  NULLAllocLink: POINTER = LOOPHOLE[1];
  AllocationVectorSize: CARDINAL = (maxallocslot+3)/2 * 2;


  -- control registers

  GFTreg: CARDINAL = 1;          -- global frame table base
  SVreg, SDreg: CARDINAL = 2;    -- system transfer vector
  AVreg: CARDINAL = 3;           -- allocation vector base
  WDCreg: CARDINAL = 4;          -- wakeup disable counter
    ReadWDC: MACHINE CODE RETURNS [CARDINAL] = INLINE [Mopcodes.zRR, WDCreg];
    WriteWDC: MACHINE CODE [CARDINAL] = INLINE [Mopcodes.zWR, WDCreg];

  Lreg: CARDINAL = 375B;                 -- local frame
  Greg: CARDINAL = 376B;                 -- global frame
  Creg: CARDINAL = 377B;                 -- code base

  maxparmsinstack: CARDINAL = 5;         -- maximum parameter depth

  StateVector: TYPE = MACHINE DEPENDENT RECORD [
    stk: ARRAY[0..7] OF UNSPECIFIED,
    instbyte: BYTE,
    fill: [0..17B],
    stkptr: [0..17B],
    X, Y: UNSPECIFIED];

  -- indices in system transfer vector (including trap codes)

  SystemDispatchSize: CARDINAL = PageSize-AllocationVectorSize;

  sBRK: CARDINAL = 0;
  sAlternateBreak: CARDINAL = 1;
  sStackError: CARDINAL = 2;
  sAllocListEmpty: CARDINAL = 6;
  sControlFault: CARDINAL = 7;
  sCsegSwappedOut: CARDINAL = 10B;
  sAlloc: CARDINAL = 11B;
  sFree: CARDINAL = 12B;
  sUnbound: CARDINAL = 13B;

  sSignalList: CARDINAL = 20B;
  sSignal: CARDINAL = 21B;
  sErrorList: CARDINAL = 22B;
  sError: CARDINAL = 23B;
  sResetPC: CARDINAL = 24B;
  sResumeError: CARDINAL = 25B;
  sUnnamedError: CARDINAL = 26B;
  sUncaughtSignal: CARDINAL = 27B;
```

```
  sBLTE: CARDINAL = 30B;
  sDivSS: CARDINAL = 31B;
  sStringInit: CARDINAL = 32B;

  sLoad: CARDINAL = 33B;
  sNew: CARDINAL = 34B;
  sCopy: CARDINAL = 35B;
  sBind: CARDINAL = 36B;
  sUnNew: CARDINAL = 37B;

  sCoreSwap: CARDINAL = 40B;
  sProcessBreakpoint: CARDINAL = 41B;
  sInterrupt: CARDINAL = 42B;
  sGoingAway: CARDINAL = 43B;
  sAddFileRequest: CARDINAL = 44B;
  sThisSpaceAvailable: CARDINAL = 45B;

  sPortInit: CARDINAL = 46B;
  sInPortInit: CARDINAL = sPortInit;
  sOutPortInit: CARDINAL = sPortInit+1;

  sGFTLength: CARDINAL = 50B;
  sBYTBLTE: CARDINAL = 51B;
  sStart: CARDINAL = 52B;
  sRestart: CARDINAL = 53B;

  END.
```