DIRECTORY
  ImageDefs: FROM "imagedefs",
  Mopcodes: FROM "mopcodes",
  AltoDefs: FROM "altodefs",
  SegmentDefs: FROM "segmentdefs",
  ControlDefs: FROM "controldefs",

DEFINITIONS FROM AltoDefs, SegmentDefs, ControlDefs;

WartDefs: DEFINITIONS = BEGIN

  BootTableLimit: INTEGER = 17777B;
  BootIndex: TYPE = POINTER [0..BootTableLimit] TO BootScriptEntry;
  NullBootIndex: BootIndex = LAST[BootIndex];
  SegmentBootIndex: TYPE = POINTER [0..BootTableLimit] TO Segment BootScriptEntry;

  BootScriptHeader: TYPE = MACHINE DEPENDENT RECORD [
      tablebase: [0..177777B], -- origin of the script table
      av, gft, sd: POINTER,
      ffvm, lfvmp: [0..AltoDefs.MaxVMPage],
      loadState: SegmentBootIndex,
      fill1: [0..7],
      initLoadState: SegmentBootIndex,
      fill2: [0..7],
      bcd: SegmentBootIndex,
      fill3: [0..7],
      nub, user: GlobalFrameHandle]; -- who to start

  BootCommand: TYPE = {bc0, bc1, bc2, bc3, bc4, bc5, bc6, bc7};

  BootScriptEntry: TYPE = MACHINE DEPENDENT RECORD [body: SELECT option: * FROM
    Command => [
      command: BootCommand,
      fill: [0..1777B],
    SwapOutCode => [
      fill: BootIndex,
      frame: GlobalFrameHandle],
    Segment => [
      fill1: [0..777B],
      data: BOOLEAN,
      access: SegmentDefs.AccessOptions,
      base: CARDINAL,
      pages: [1..MaxVMPage+1],
      vmpage: [0..MaxVMPage],
      handle: FileSegmentHandle],
    CodeLink => [
      codeseg: SegmentBootIndex,
      frame: GlobalFrameHandle],
    Unlock => [
      seg: SegmentBootIndex],
    OpenFile => [
      fill: BootIndex],
    Stop => [
      fill: BootIndex],
    ENDCASE]

END.