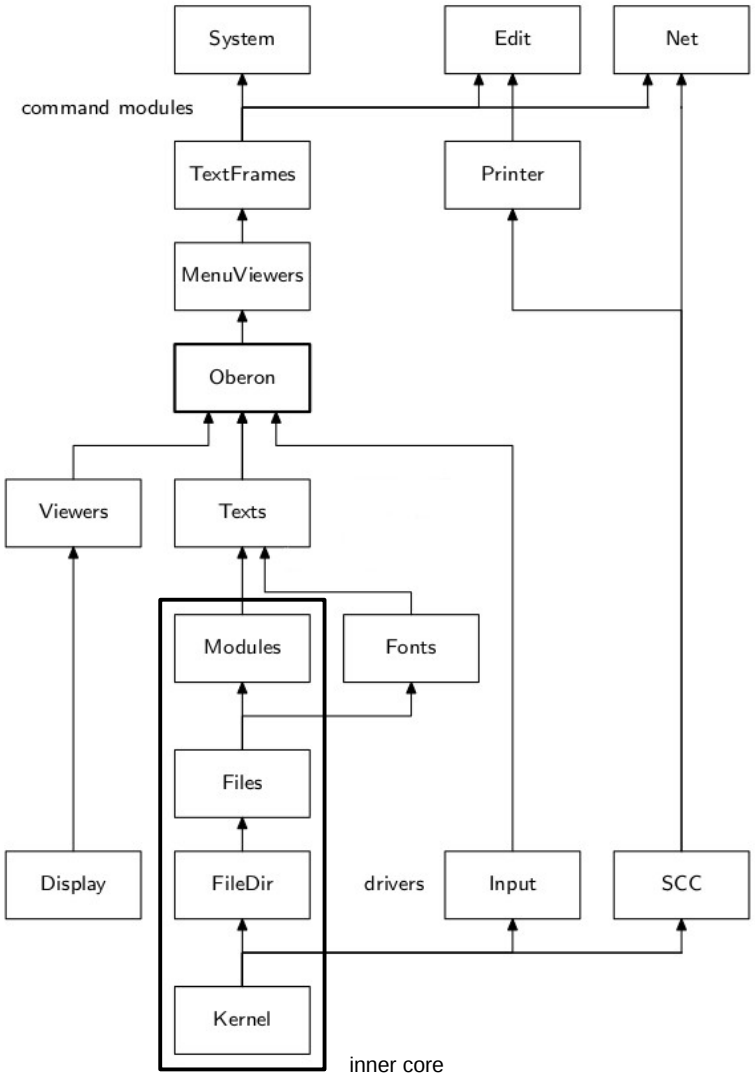
Oberon Core Structure and Module Interfaces



[inner core - modules in alphabetical order]

```
DEFINITION FileDir;
DEFINITION Files;
DEFINITION Kernel;
DEFINITION Modules;
```

DEFINITION FileDir; (*NW 12.1.86 / 23.8.90 / 15.8.2013*)

```
CONST
  FnLength = 32;
  SecTabSize = 64;
  ExTabSize = 12;
  SectorSize = 1024;
  IndexSize = SectorSize DIV 4; (*no. of entries in index sector*)
  HeaderSize = 352;
  DirRootAdr = 29;
  DirPgSize = 24;
  DirMark = 9B1EA38DH;
  HeaderMark = 9BA71D86H;

TYPE
  FileName = ARRAY FnLength OF CHAR;
  SectorTable = ARRAY SecTabSize OF INTEGER;
  ExtensionTable = ARRAY ExTabSize OF INTEGER;
  EntryHandler = PROCEDURE (name: FileName; sec: INTEGER;
    VAR continue: BOOLEAN);

  FileHeader = RECORD (*first page of each file on disk*)
    mark: INTEGER;
    name: FileName;
    aleng, bleng, date: INTEGER;
    ext: ExtensionTable;
    sec: SectorTable;
  END ;

  FileHd = POINTER TO FileHeader;
  IndexSector = ARRAY IndexSize OF INTEGER;
  DataSector = ARRAY SectorSize OF BYTE;

  DirEntry = RECORD name: FileName; adr, p: INTEGER; END ;
  DirPage = RECORD (*B-tree node*)
    mark: INTEGER;
    m: INTEGER; (*no. of elements on page*)
    p0: INTEGER;
    e: ARRAY DirPgSize OF DirEntry;
  END ;

PROCEDURE Search(name: FileName; VAR A: INTEGER);
PROCEDURE Insert(name: FileName; fad: INTEGER);
PROCEDURE Delete(name: FileName; VAR fad: INTEGER);
PROCEDURE Enumerate(prefix: ARRAY OF CHAR; proc: EntryHandler);
PROCEDURE Init;

END FileDir.
```

DEFINITION Files; (*NW 11.1.86 ... 25.12.95 / 15.8.2013*)

```
TYPE
  File = POINTER TO FileDesc;
  Rider = RECORD eof: BOOLEAN; res: INTEGER END ;

PROCEDURE Old(name: ARRAY OF CHAR): File;
PROCEDURE New(name: ARRAY OF CHAR): File;
PROCEDURE Register(f: File);
PROCEDURE Close(f: File);
PROCEDURE Purge(f: File);
PROCEDURE Delete(name: ARRAY OF CHAR; VAR res: INTEGER);
PROCEDURE Rename(old, new: ARRAY OF CHAR; VAR res: INTEGER);
PROCEDURE Length(f: File): INTEGER;
PROCEDURE Date(f: File): INTEGER;
PROCEDURE Set(VAR r: Rider; f: File; pos: INTEGER);
PROCEDURE Pos(VAR r: Rider): INTEGER;
PROCEDURE Base(VAR r: Rider): File;
PROCEDURE ReadByte(VAR r: Rider; VAR x: BYTE);
PROCEDURE ReadBytes(VAR r: Rider; VAR x: ARRAY OF BYTE; n: INTEGER);
PROCEDURE Read(VAR r: Rider; VAR ch: CHAR);
PROCEDURE ReadInt(VAR r: Rider; VAR x: INTEGER);
PROCEDURE ReadSet(VAR r: Rider; VAR x: SET);
PROCEDURE ReadReal(VAR r: Rider; VAR x: REAL);
PROCEDURE ReadString(VAR r: Rider; VAR x: ARRAY OF CHAR);
PROCEDURE ReadNum(VAR r: Rider; VAR x: INTEGER);
PROCEDURE WriteByte(VAR r: Rider; x: BYTE);
PROCEDURE WriteBytes(VAR r: Rider; VAR x: ARRAY OF BYTE; n: INTEGER);
PROCEDURE Write(VAR r: Rider; ch: CHAR);
PROCEDURE WriteInt(VAR r: Rider; x: INTEGER);
PROCEDURE WriteSet(VAR r: Rider; x: SET);
PROCEDURE WriteReal(VAR r: Rider; x: REAL);
PROCEDURE WriteString(VAR r: Rider; x: ARRAY OF CHAR);
PROCEDURE WriteNum(VAR r: Rider; x: INTEGER);

(*---------------------------System use---------------------------*)
PROCEDURE Init;
PROCEDURE RestoreList; (*after mark phase of garbage collection*)

END Files.
```

```
DEFINITION Kernel; (*NW/PR  11.4.86 / 27.12.95 / 4.2.2014*)

CONST
  SectorLength = 1024;

TYPE
  Sector = ARRAY SectorLength OF BYTE;

VAR
  allocated, NofSectors, heapOrg, heapLim, stackOrg, stackSize,
    MemLim: INTEGER;

(* ---------- New: heap allocation ----------*)
PROCEDURE New(VAR ptr: INTEGER; tag: INTEGER);

(* ---------- Garbage collector ----------*)
PROCEDURE Mark(pref: INTEGER);
PROCEDURE Scan;

(* ---------- Disk storage management ----------*)
PROCEDURE InitSecMap;
PROCEDURE MarkSector(sec: INTEGER);
PROCEDURE FreeSector(sec: INTEGER);
PROCEDURE AllocSector(hint: INTEGER; VAR sec: INTEGER);
PROCEDURE GetSector(src: INTEGER; VAR dest: Sector);
PROCEDURE PutSector(dest: INTEGER; VAR src: Sector);

(*-------- Miscellaneous procedures----------*)
PROCEDURE Time(): INTEGER;
PROCEDURE Clock(): INTEGER;
PROCEDURE SetClock(dt: INTEGER);
PROCEDURE Install(Padr, at: INTEGER);
PROCEDURE Init;

END Kernel.
```

```
DEFINITION Modules;  (*Link and load on RISC; NW 20.10.2013*)

TYPE
  Module = POINTER TO ModDesc;
  Command = PROCEDURE;
  ModuleName = ARRAY 32 OF CHAR;
  ModDesc = RECORD
    name: ModuleName;
    next: Module;
    key, num, size, refcnt: INTEGER;
    data, code, imp, cmd, ent, ptr: INTEGER  (*addresses*)
  END ;

VAR
  root: Module;
  MTOrg, AllocPtr, res: INTEGER;
  importing, imported: ModuleName;

PROCEDURE Load(name: ARRAY OF CHAR; VAR newmod: Module);
PROCEDURE ThisCommand(mod: Module; name: ARRAY OF CHAR): Command;
PROCEDURE Free(name: ARRAY OF CHAR);
PROCEDURE Init;

END Modules.

[end of inner core]
```

[outer core - modules in alphabetical order; SCC, Net, Printer not included]

```
DEFINITION Display;
DEFINITION Edit;
DEFINITION Fonts;
DEFINITION Input;
DEFINITION MenuViewers;
DEFINITION Oberon;
DEFINITION System;
DEFINITION TextFrames;
DEFINITION Texts;
DEFINITION Viewers;
```

```
DEFINITION Display;  (*NW 5.11.2013*)

CONST
  black = 0; white = 1; (*colors*)
  replace = 0; paint = 1; invert = 2; (*operation modes*)

TYPE
  Frame = POINTER TO FrameDesc;
  FrameMsg = RECORD END;
  Handler = PROCEDURE (F: Frame; VAR M: FrameMsg);
  FrameDesc = RECORD
    next, dsc: Frame;
    X, Y, W, H: INTEGER;
    handle: Handler
  END;

VAR
  Base, Width, Height: INTEGER;
  arrow, star, hook, updown, block, cross, grey: INTEGER;

PROCEDURE Dot (col, x, x, mode: INTEGER);
PROCEDURE ReplConst (col, x, y, w, h, mode: INTEGER);
PROCEDURE CopyPattern (col, patadr, x, y, mode: INTEGER);
PROCEDURE CopyBlock (sx, sy, w, h, dx, dy, mode: INTEGER);
PROCEDURE ReplPattern (col, patadr, x, y, w, h, mode: INTEGER);

END Display.
```

```
DEFINITION Edit; (*JG 2.11.90 / NW 18.1.92 /30.12.95 / 10.10.10 / NW
10.1.2013*)

PROCEDURE Open;
PROCEDURE Store;
PROCEDURE CopyLooks;
PROCEDURE ChangeFont;
PROCEDURE ChangeColor;
PROCEDURE ChangeOffset;
PROCEDURE Search;
PROCEDURE Locate;
PROCEDURE Recall;

END Edit.
```

```
DEFINITION Fonts; (*JG 18.11.90; PDR 8.6.12; NW 25.3.2013*)

TYPE
  Font = POINTER TO FontDesc;
  FontDesc = RECORD
    name: ARRAY 32 OF CHAR;
    height, minX, maxX, minY, maxY INTEGER;
    next: Font
  END;

VAR
  Default, root: Font;

PROCEDURE GetPat(fnt: Font; ch: CHAR;
   VAR dx, x, y, w, h, patadr: INTEGER);
PROCEDURE This (name: ARRAY OF CHAR): Font;
PROCEDURE Free(name: ARRAY OF CHAR);

END Fonts.
```

```
DEFINITION Input;  (*NW 5.10.86 / 15.11.90 Ceres-2; PDR 21.4.12 / NW
15.5.2013 Ceres-4*)

  PROCEDURE Available(): INTEGER;
  PROCEDURE Read(VAR ch: CHAR);
  PROCEDURE Mouse(VAR keys: SET; VAR x, y: INTEGER);
  PROCEDURE SetMouseLimits(w, h: INTEGER);
  PROCEDURE Init;

END Input.
```

```
DEFINITION MenuViewers;  (*JG 26.8.90 / 16.9.93 / NW 10.3.2013*)
  IMPORT Viewers, Display;

CONST extend = 0; reduce = 1; (*message ids*)

TYPE
  Viewer = POINTER TO ViewerDesc;
  ViewerDesc = RECORD (Viewers.ViewerDesc)
    menuH: INTEGER
  END;
  ModifyMsg = RECORD (Display.FrameMsg)
    id: INTEGER;
    dY, Y, H: INTEGER
  END;

PROCEDURE Handle (V: Display.Frame; VAR M: Display.FrameMsg);
PROCEDURE New (Menu, Main: Display.Frame; menuH, X, Y: INTEGER): Viewer;

END MenuViewers.
```

```
DEFINITION Oberon; (*JG 6.9.90 ... 13.8.94 / NW 14.4.2013 / 22.12.2013*)
  IMPORT Viewers, Fonts, Texts;

CONST (*message ids*)
  consume = 0; track = 1; defocus = 0; neutralize = 1; mark = 2;

TYPE
  Painter = PROCEDURE (x, y: INTEGER);
  Marker = RECORD Fade, Draw: Painter END;
  Cursor = RECORD
    marker: Marker; on: BOOLEAN; X, Y: INTEGER
  END;

  InputMsg = RECORD (Display.FrameMsg)
    id: INTEGER;
    keys: SET;
    X, Y: INTEGER;
    ch: CHAR;
    fnt: Fonts.Font;
    col, voff: INTEGER
  END;

  SelectionMsg = RECORD (Display.FrameMsg)
    time: INTEGER;
    text: Texts.Text;
    beg, end: INTEGER
  END;

  ControlMsg = RECORD (Display.FrameMsg)
    id, X, Y: INTEGER
  END;

  CopyMsg = RECORD (Display.FrameMsg)
    F: Display.Frame
  END;

  Task = POINTER TO TaskDesc;

  Handler = PROCEDURE;

  TaskDesc = RECORD period: INTEGER END;

  VAR
    User: ARRAY 8 OF CHAR; Password: INTEGER;
    Arrow, Star: Marker;
    Mouse, Pointer: Cursor;
    FocusViewer: Viewers.Viewer;
    Log: Texts.Text;
    Par: RECORD
      vwr: Viewers.Viewer;
      frame: Display.Frame;
      text: Texts.Text;
      pos: INTEGER
    END;
    CurFnt: Fonts.Font;
    CurCol, CurOff, NofTasks: INTEGER;

  (*user identification*)
  PROCEDURE SetUser (VAR user, password: ARRAY OF CHAR);

  (*time*)
  PROCEDURE Clock (): INTEGER;
  PROCEDURE SetClock (d: INTEGER);
  PROCEDURE Time (): INTEGER;

  (*cursor handling*)
  PROCEDURE DrawMouse (m: Marker; X, Y: INTEGER);
  PROCEDURE DrawMouseArrow (X, Y: INTEGER);
  PROCEDURE FadeMouse();
  PROCEDURE DrawPointer (X, Y: INTEGER);

  (*display management*)
  PROCEDURE RemoveMarks (X, Y, W, H: INTEGER);
  PROCEDURE OpenDisplay (UW, SW, H: INTEGER);
  PROCEDURE DisplayWidth (X: INTEGER): INTEGER;
  PROCEDURE DisplayHeight (X: INTEGER): INTEGER;
  PROCEDURE OpenTrack (X, W: INTEGER);
  PROCEDURE UserTrack (X: INTEGER): INTEGER;
  PROCEDURE SystemTrack (X: INTEGER): INTEGER;
  PROCEDURE AllocateUserViewer (DX: INTEGER; VAR X, Y: INTEGER);
  PROCEDURE AllocateSystemViewer (DX: INTEGER; VAR X, Y: INTEGER);
  PROCEDURE MarkedViewer (): Viewers.Viewer;
  PROCEDURE PassFocus (V: Viewers.Viewer);
  PROCEDURE OpenLog(T: Texts.Text);

  (*command interpretation*)
  PROCEDURE SetPar(F: Display.Frame; T: Texts.Text; pos: INTEGER);
  PROCEDURE Call (name: ARRAY OF CHAR; VAR res: INTEGER);
  PROCEDURE GetSelection (VAR text: Texts.Text;
    VAR beg, end, time: INTEGER);

  (*tasks*)
  PROCEDURE NewTask (h: Handler; period: INTEGER);
  PROCEDURE Install (T: Task);
  PROCEDURE Remove (T: Task);
  PROCEDURE Collect (count: INTEGER);

  (*looks*)
  PROCEDURE SetFont (fnt: Fonts.Font);
  PROCEDURE SetColor (col: INTEGER);
  PROCEDURE SetOffset (voff: INTEGER);

  (*main loop*)
  PROCEDURE Loop;

  (*system use*)
  PROCEDURE Reset;

END Oberon.
```

```
DEFINITION System; (*JG 3.10.90 / NW 12.10.93 / NW 18.5.2013*)

(*Toolbox for system control*)
PROCEDURE SetUser;
PROCEDURE SetFont;
PROCEDURE SetColor;
PROCEDURE SetOffset;
PROCEDURE Date;
PROCEDURE Collect;

(*Toolbox for standard display*)
PROCEDURE Open;  (*open viewer in system track*)
PROCEDURE Clear;  (*used to clear Log*)
PROCEDURE Close;
PROCEDURE CloseTrack;
PROCEDURE Recall;
PROCEDURE Copy;
PROCEDURE Grow;

(*Toolbox for module and font management*)
PROCEDURE Free;
PROCEDURE FreeFonts;

(*Toolbox of file system*)
PROCEDURE Directory;
PROCEDURE CopyFiles;
PROCEDURE RenameFiles;
PROCEDURE DeleteFiles;)

(*Toolbox for system inspection*)
PROCEDURE Watch;
PROCEDURE ShowModules;
PROCEDURE ShowCommands;
PROCEDURE ShowFonts;

(*display configuration*)
PROCEDURE ExtendDisplay;

END System;
```

```
DEFINITION TextFrames; (*JG 8.10.90 / NW 10.5.2013*)

CONST
  replace = 0; insert = 1; delete = 2; unmark = 3; (*message id*)

TYPE
  Location = RECORD
    org, pos: INTEGER;
    dx, x, y: INTEGER
  END;

  Frame = POINTER TO FrameDesc;
  FrameDesc = RECORD (Display.FrameDesc)
    text: Texts.Text;
    org: INTEGER;
    col: INTEGER;
    lsp: INTEGER;
    left, right, top, bot: INTEGER;
    markH: INTEGER;
    time: INTEGER;
    hasCar, hasSel: BOOLEAN;
    carloc: Location;
    selbeg, selend: Location
  END;

  UpdateMsg = RECORD (Display.FrameMsg)
    id: INTEGER;
    text: Texts.Text;
    beg, end: INTEGER
  END;

  VAR
    TBuf: Texts.Buffer;
    menuH, barW, left, right, top, bot, lsp: INTEGER; (*standard sizes*)

  (*------------------display support------------------------*)
  PROCEDURE Mark (F: Frame; on: BOOLEAN);
  PROCEDURE Restore (F: Frame);
  PROCEDURE Suspend (F: Frame);
  PROCEDURE Extend (F: Frame; newY: INTEGER);
  PROCEDURE Reduce (F: Frame; newY: INTEGER);
  PROCEDURE Show (F: Frame; pos: INTEGER);
  PROCEDURE Pos (F: Frame; X, Y: INTEGER): INTEGER;
  PROCEDURE SetCaret (F: Frame; pos: INTEGER);
  PROCEDURE TrackCaret (F: Frame; X, Y: INTEGER; VAR keysum: SET);
  PROCEDURE RemoveCaret (F: Frame);
  PROCEDURE SetSelection (F: Frame; beg, end: INTEGER);
  PROCEDURE TrackSelection (F: Frame; X, Y: INTEGER; VAR keysum: SET);
  PROCEDURE RemoveSelection (F: Frame);
  PROCEDURE TrackLine (F: Frame; X, Y: INTEGER; VAR org: INTEGER;
    VAR keysum: SET);
  PROCEDURE TrackWord (F: Frame; X, Y: INTEGER; VAR pos: INTEGER;
    VAR keysum: SET);
  PROCEDURE Replace (F: Frame; beg, end: INTEGER);
  PROCEDURE Insert (F: Frame; beg, end: INTEGER);
  PROCEDURE Delete (F: Frame; beg, end: INTEGER);
  PROCEDURE Recall (VAR B: Texts.Buffer);
  (*------------------message handling------------------------*)
  PROCEDURE NotifyDisplay (T: Texts.Text; op: INTEGER; beg, end: INTEGER);
  PROCEDURE Call (F: Frame; pos: INTEGER; new: BOOLEAN);
  PROCEDURE Write (F: Frame; ch: CHAR; fnt: Fonts.Font;
    col, voff: INTEGER);
  PROCEDURE Defocus (F: Frame);
  PROCEDURE Neutralize (F: Frame);
  PROCEDURE Modify (F: Frame; id, dY, Y, H: INTEGER);
  PROCEDURE Open (F: Frame; H: Display.Handler; T: Texts.Text;
    org: INTEGER;
  PROCEDURE Copy (F: Frame; VAR F1: Frame);
  PROCEDURE GetSelection (F: Frame; VAR text: Texts.Text;
    VAR beg, end, time: INTEGER);
  PROCEDURE Update (F: Frame; VAR M: UpdateMsg);
  PROCEDURE Edit (F: Frame; X, Y: INTEGER; Keys: SET);
  PROCEDURE Handle (frame: Display.Frame; VAR M: Display.FrameMsg);

  (*creation*)
  PROCEDURE Text (name: ARRAY OF CHAR): Texts.Text;
  PROCEDURE NewMenu (name, commands: ARRAY OF CHAR): Frame;
  PROCEDURE NewText (text: Texts.Text; pos: INTEGER): Frame;

END TextFrames.
```

```
DEFINITION Texts; (*JG 21.11.90 / NW 11.7.90 / 24.12.95 / 22.11.10 /
18.11.2014*)

  CONST (*scanner symbol classes*)
    Inval = 0;          (*invalid symbol*)
    Name = 1;           (*name s (length len)*)
    String = 2;         (*literal string s (length len)*)
    Int = 3;            (*integer i (decimal or hexadecimal)*)
    Real = 4;           (*real number x*)
    Char = 6;           (*special character c*)

    replace = 0; insert = 1; delete = 2; unmark = 3;  (*op-codes*)

  TYPE
    Text = POINTER TO TextDesc;
    Notifier = PROCEDURE (T: Text; op: INTEGER; beg, end: INTEGER);
    TextDesc = RECORD
      len: INTEGER;
      changed: BOOLEAN;
      notify: Notifier;
    END;

    Reader = RECORD
      eot: BOOLEAN;
      fnt: Fonts.Font;
      col, voff: INTEGER;
    END;

    Scanner = RECORD (Reader)
      nextCh: CHAR;
      line, class: INTEGER;
      i: INTEGER;
      x: REAL;
      c: CHAR;
      len: INTEGER;
      s: ARRAY 32 OF CHAR
    END;

    Buffer = POINTER TO BufDesc;
    BufDesc = RECORD
      len: INTEGER
    END;

    Writer = RECORD
      buf: Buffer;
      fnt: Fonts.Font;
      col, voff: INTEGER;
    END;

  (* -------------------- Filing -------------------------*)
  PROCEDURE Load (VAR R: Files.Rider; T: Text);
  PROCEDURE Open (T: Text; name: ARRAY OF CHAR);
  PROCEDURE Store (VAR W: Files.Rider; T: Text);
  PROCEDURE Close (T: Text; name: ARRAY OF CHAR);

  (* -------------------- Editing ----------------------- *)
  PROCEDURE OpenBuf (B: Buffer);
  PROCEDURE Save (T: Text; beg, end: INTEGER; B: Buffer);
  PROCEDURE Copy (SB, DB: Buffer);
  PROCEDURE Insert (T: Text; pos: INTEGER; B: Buffer);
  PROCEDURE Append (T: Text; B: Buffer);
  PROCEDURE Delete (T: Text; beg, end: INTEGER; B: Buffer);
  PROCEDURE ChangeLooks (T: Text; beg, end: INTEGER; sel: SET;
    fnt: Fonts.Font; col, voff: INTEGER);
  PROCEDURE Attributes (T: Text; pos: INTEGER; VAR fnt: Fonts.Font;
    VAR col, voff: INTEGER);

  (* ------------------ Access: Readers ------------------------ *)
  PROCEDURE OpenReader (VAR R: Reader; T: Text; pos: INTEGER);
  PROCEDURE Read (VAR R: Reader; VAR ch: CHAR);
  PROCEDURE Pos (VAR R: Reader): INTEGER;

  (* ------------------ Access: Scanners (NW) ------------------------ *)
  PROCEDURE OpenScanner (VAR S: Scanner; T: Text; pos: INTEGER);
  PROCEDURE Scan (VAR S: Scanner);

  (* --------------- Access: Writers (NW) ----------------- *)
  PROCEDURE OpenWriter (VAR W: Writer);
  PROCEDURE SetFont (VAR W: Writer; fnt: Fonts.Font);
  PROCEDURE SetColor (VAR W: Writer; col: INTEGER);
  PROCEDURE SetOffset (VAR W: Writer; voff: INTEGER);
  PROCEDURE Write (VAR W: Writer; ch: CHAR);
  PROCEDURE WriteLn (VAR W: Writer);
  PROCEDURE WriteString (VAR W: Writer; s: ARRAY OF CHAR);
  PROCEDURE WriteInt (VAR W: Writer; x, n: INTEGER);
  PROCEDURE WriteHex (VAR W: Writer; x: INTEGER);
  PROCEDURE WriteReal (VAR W: Writer; x: REAL; n: INTEGER);
  PROCEDURE WriteRealFix (VAR W: Writer; x: REAL; n, k: INTEGER);
  PROCEDURE WriteClock (VAR W: Writer; d: INTEGER);

END Texts.
```

```
DEFINITION Viewers; (*JG 14.9.90 / NW 15.9.2013*)
  IMPORT Display;

CONST restore = 0; modify = 1; suspend = 2; (*message ids*)

TYPE
  Viewer = POINTER TO ViewerDesc;

  ViewerDesc = RECORD (Display.FrameDesc)
    state: INTEGER
  END;

  ViewerMsg = RECORD (Display.FrameMsg)
    id: INTEGER;
    X, Y, W, H: INTEGER;
    state: INTEGER
  END;

VAR curW, minH: INTEGER;

PROCEDURE Open (V: Viewer; X, Y: INTEGER);
PROCEDURE Change (V: Viewer; Y: INTEGER);
PROCEDURE Close (V: Viewer);
PROCEDURE Recall (VAR V: Viewer);
PROCEDURE This (X, Y: INTEGER): Viewer;
PROCEDURE Next (V: Viewer): Viewer;
PROCEDURE Locate (X, H: INTEGER; VAR fil, bot, alt, max: Viewer);
PROCEDURE InitTrack (W, H: INTEGER; Filler: Viewer);
PROCEDURE OpenTrack (X, W: INTEGER; Filler: Viewer);
PROCEDURE CloseTrack (X: INTEGER);
PROCEDURE Broadcast (VAR M: Display.FrameMsg);

END Viewers.
```

[end of outer core]