```
@A+
//==== Business-Control ===============================================
//
//  Prozedur    BA1_Planung_Walzen (BSP)
//                  OHNE E_R_G
//  Info
//
//
//  25.06.2018  AH  Erstellung der Prozedur
//  16.07.2018  AH  Umbau auf Pool
//  22.10.2018  AH  Erweiterungen
//  16.04.2019  AH  Umsortieren in Planung ist temporär
//  08.11.2019  AH  "Teilungszeit" ausgebaut
//  08.11.2019  AH  neue Spalten für Termine (Proj.1994/239)
//  05.04.2022  AH  ERX
//
//  Subprozeduren
//    SUB Start();
//
//
//     BAG.P.Plan.StartInfo  # aPlanname; Format 2019/4/1
//
//=========================================================================
@I:Def_Global
@I:Def_Rights
@I:Def_BAG
```

```
define begin
  cDebug      : (gUsername='AHx')

  cModulName  : 'vonSC_BagPlanWalzen'

  cJIT        : '#AUTOJIT#'

  cTitle      : 'Walzenplanung'
  cMenuName   : 'BA1.Feinplanung'

  cMDI        : gMdiMath
  cDLPlan     : $dl.Plan
  cDLPool     : $dl.Pool
  cDLPoolName : 'dl.Pool'
  cDLPlanName : 'dl.Plan'
  cDLPoolFilter : $dl.PoolFilter

  c_Vorwalzen : 'VORWAL'
  cRuesten    : 15
//  cTeilen     : 5
  cKeinKal    : 'KEIN KALENDER'

  cClmRecId   : 1
  cClmFolge   : 2
  cClmStatus  : 3
```

```
  cClmTlg     : 4
  cClmDau     : 5
  cClmDau2    : 6
  cClmDauSum  : 7
  cClmStart   : 8
  cClmEnde    : 9
  cClmProg    : 10
  cClmGuete   : 11
  cClmFest    : 12
  cClmBAG     : 13
  cClmStich   : 14
  cClmVon     : 15
  cClmBis     : 16
  cClmInputD  : 17
  cClmOutputD : 18
  cClmOutputDTol : 19
  cClmInputB  : 20
  cClmBTol    : 21
  cClmRinge   : 22
  cClmGew     : 23
  cClmTerminW : 24
  cClmTerminZ : 25
  cClmCustom  : 26
end;

declare StartInner();
```

```
declare _702NachDl(aDL :int; aTxt : int; aIstPlan : logic);
declare _Vorherigeswalzen(aTxt : int; aSet : logic; aPos : int; var aVon : int) : int
declare _FolgendesWalzen(aTxt : int; aSet : logic; aPos : int; var aVon : int) : int
declare _Recalc(aDat  : date; aTim  : time; aMitDauer : logic)

//=========================================================================
// Start
//  Call SFX_Planung_Walzen:Start
//=========================================================================
sub Start();
begin
  if (cMDI<>0) then begin
    Lib_guiCom:ReOpenMDI(cMDI);
    RETURN;
  end;

  RecBufClear(998);
  Sel.Art.von.ArtNr   # '50';
  Sel.BAG.Res.Gruppe  # 3;
  Sel.BAG.Res.Nummer  # 3;
  Sel.von.Datum       # today;
  Sel.bis.Datum       # today;
  Sel.Bis.Datum->vmDayModify(31*6);
  if (cDebug) then begin
    Sel.BAG.Res.Gruppe  # 1;
    Sel.BAG.Res.Nummer  # 1;
```

```
//    gSelected # 1;
//    StartInner();
//    RETURN;
  end;

  cMDI # Lib_GuiCom:AddChildWindow(gMDI,'BA1.Planung.Walzen.Sel',here+':StartInner', true);
  Lib_GuiCom:RunChildWindow(cMDI);
end;



//=========================================================================
// StartInner
//
//=========================================================================
sub StartInner();
local begin
  Erx       : int;
  vSel      : int;
  vSelName  : alpha;
  vQ        : alpha(4000);
  vQ2       : alpha(4000);
  vDlPlan   : int;
  vDlPool   : int;
  vHdl      : int;
  vTxt      : int;
  vI,vJ,vK  : int;
```

```
  vTree     : int;
  vItem     : int;
  vSortkey  : alpha;
  vDat      : date;
  vTim      : time;
  vPlanName : alpha;
end;
begin

  vPlanName # Sel.Art.Bis.ArtNr;

  if (gSelected=0) then RETURN;
  gSelected # 0;
  if (cMDI<>0) then RETURN;


  // Dialog starten...
  cMDI # Lib_GuiCom:OpenMdi(gFrmMain, 'BA1.Planung.Walzen', _WinAddHidden);
  VarInstance(WindowBonus,cnvIA(cMDI->wpcustom));
  vDlPool # Winsearch(cMDI, cDlPoolName);
  vDlPlan # Winsearch(cMDI, cDlPlanName);
  vTxt # TextOpen(16);

  // ALTE PLANUNG GGF FÜLLEN----------------------------------------------
  if (vPlanname<>'') then begin
    if (lib_Strings:Strings_Count(vPlanname,'/')=0) then
```

```
    vPlanname # vPlanname + '/1';
  if (lib_Strings:Strings_Count(vPlanname,'/')=1) then begin
    vDat # today;
    vPlanname # aint(vDat->vpYear)+'/'+vPlanname;
  end;

  vQ # '';
  Lib_Sel:QAlpha( var vQ, 'BAG.P.Plan.StartInfo', '=^', vPlanname);

  // Selektion aufbauen...
  vSel # SelCreate(702, 8); // nach STARTTERMIN
//   vSel->SelAddSortFld( FldInfoByName( 'BAG.P.Reihenfolge', _fldSbrNumber ), FldInfoByName( 'BAG.P
  Erx # vSel->SelDefQuery('', vQ);
  if (Erx != 0) then Lib_Sel:QError(vSel);

  // speichern, starten und Name merken...
  vSelName # Lib_Sel:SaveRun(var vSel,0,n);

  // Positionen mit Planungslock versehen...
  FOR Erx # RecRead(702,vSel,_RecFirst)
  LOOP Erx # RecRead(702,vSel,_RecNext)
  WHILE (Erx<=_rLocked) do begin
    if (vDat=0.0.0) then begin
      vDat  # BAG.P.Plan.StartDat;
      vTim  # BAG.P.Plan.StartZeit;
    end;
```

```
      _702NachDl(vDlPlan, vTxt, true);
    END;
    SelClose(vSel);
    SelDelete(702, vSelName);
  end;  // Planung füllen


  // POOL FÜLLEN-----------------------------------------------
  vQ # '';
  if (cDebug) then begin
    Lib_Sel:QInt( var vQ, 'BAG.P.Nummer', '=', 1514);
    Lib_Sel:QAlpha( var vQ, 'BAG.P.Aktion', '>', '0');
    Lib_Sel:QAlpha( var vQ, 'BAG.P.Aktion', '!=', c_BAG_VSB);
    Lib_Sel:QInt( var vQ, 'BAG.P.Reihenfolge', '=', 0);
  end
  else begin
    if ( Sel.BAG.Res.Gruppe != 0 ) then
      Lib_Sel:QInt( var vQ, 'BAG.P.Ressource.Grp', '=', Sel.BAG.Res.Gruppe );
    if ( Sel.BAG.Res.Nummer != 0 ) then
      Lib_Sel:QInt( var vQ, 'BAG.P.Ressource', '=', Sel.BAG.Res.Nummer );
//    Lib_Sel:QDate( var vQ, 'BAG.P.Plan.StartDat', '>=', Sel.von.Datum);
    Lib_Sel:QDate( var vQ, 'BAG.P.Plan.StartDat', '<=', Sel.bis.Datum);
    Lib_Sel:QDate( var vQ, 'BAG.P.Fertig.Dat', '=', 0.0.0 );
//    vQ # vQ + ' AND !BAG.P.ExternYN';
    vQ2 # '';
```

```
  Lib_Sel:QAlpha( var vQ2, 'BAG.P.Aktion', '=', c_BAG_Walz);
  Lib_Sel:QAlpha( var vQ2, 'BAG.P.Aktion', '=', c_VorWalzen,'OR');
  vQ # vQ + ' AND ('+vQ2+')';

  if (Sel.Art.von.ArtNr<>'') then begin
    vQ2 # '';
    Lib_Sel:QAlpha( var vQ2, 'BAG.P.Zusatz', '=', '');
    Lib_Sel:QAlpha( var vQ2, 'BAG.P.Zusatz', '=*', '*'+Sel.Art.Von.ArtNr+'*', 'OR');
    vQ2 # '(' + vQ2 + ')';
    vQ # vQ + ' AND '+vQ2;
  end;
  Lib_Sel:QInt( var vQ, 'BAG.P.Reihenfolge', '=', 0);
end;

vQ2 # '';
Lib_Sel:QAlpha( var vQ2, 'BAG.Löschmarker', '=', '' );
Lib_Sel:QLogic( var vQ2, 'BAG.VorlageYN', false);
vQ # vQ + ' AND ( LinkCount(Kopf) > 0) ';

// Selektion aufbauen...
vSel # SelCreate(702, 6); // nach LEVEL

// Verknüpfen mit BAG Kopfdaten
vSel->SelAddLink('', 700, 702, 1, 'Kopf');

// nach Level sortieren...
```

```
//  vSel->SelAddSortFld(1, 17, _KeyFldAttrUpperCase | _KeyFldAttrReverse );
  Erx # vSel->SelDefQuery('', vQ);
  Erx # vSel->SelDefQuery('Kopf', vQ2 );
  if (Erx != 0) then Lib_Sel:QError(vSel);

  // speichern, starten und Name merken...
  vSelName # Lib_Sel:SaveRun(var vSel,0,n);

  vTree # CteOpen(_CteTreeCI);    // Rambaum anlegen

  // Positionen mit Planungslock versehen...
  FOR Erx # RecRead(702,vSel,_RecFirst)
  LOOP Erx # RecRead(702,vSel,_RecNext)
  WHILE (Erx<=_rLocked) do begin
    vSortkey # '';
    FOR Erx # RecLink(701,702,2,_recFirst)    // Input loopen
    LOOP Erx # RecLink(701,702,2,_recNext)
    WHILE (Erx<=_rLocked) do begin
      if ((BAG.IO.Materialtyp=c_IO_Mat) or
        (BAG.IO.Materialtyp=c_IO_Theo) or
        (BAG.IO.Materialtyp=c_IO_BAG)) and (BAG.IO.VonFertigmeld=0) then begin
//        if (cDebug) then
//          vSortKey # cnvAI(BAG.P.Nummer,_FmtNumLeadZero|_fmtNumNoGroup,0,10)+cnvAi(BAG.P.Posi
//        else
          vSortKey # cnvai(BAG.P.Reihenfolge,_FmtNumLeadZero|_FmtNumNoGroup,0,10)+'|'+cnvAF(9999.
//          vSortKey # cnvAF(9999.0 - BAG.IO.Breite,_FmtNumLeadZero|_fmtNumNoGroup,0,2,10)+cnvAF(9
```

```
        BREAK;
      end;
    END;
    Sort_ItemAdd(vTree,vSortKey,702,RecInfo(702,_RecId));
  END;
//    RecRead(702,1,_recLock);
//    BAG.P.PLanLock.UsrID # gUserID;
//    BA1_P_Data:Replace(_recUnlock,'MAN');
  FOR   vItem # Sort_ItemFirst(vTree) // RAMBAUM
  loop  vItem # Sort_ItemNext(vTree,vItem)
  WHILE (vItem != 0) DO BEGIN
    RecRead(cnvIA(vItem->spCustom), 0, 0, vItem->spID); // Datensatz holen
    _702NachDl(vDlPool, vTxt, false);
  END;
  CteClear(vTree, true);
  CteClose(vTree);
  SelClose(vSel);
  SelDelete(702, vSelName);


  TextClose(vTxt);


  // Vorbelegungen
  vHdl # Winsearch(cMDI, 'lbNr');
  vHdl->wpCustom # vPlanname;
```

```
  vHdl # Winsearch(cMDI, 'edRuestzeit');
  vHdl->wpCaptionInt # cRuesten;
//  vHdl # Winsearch(cMDI, 'edTelungszeit');
//  vHdl->wpCaptionInt # cTeilen;
  vHdl # Winsearch(cMDI, 'lb.Ressource');
  Rso.Gruppe # Sel.BAG.Res.Gruppe;
  Rso.Nummer # Sel.BAG.Res.Nummer;
  Erx # RecRead(160,1,0);   // Ressource holen
  if (Erx>_rLocked) then ReCbufClear(160);
  vHdl->wpcaption # aint(Rso.Gruppe)+'/'+aint(Rso.Nummer)+' '+Rso.Stichwort;
  vHdl # Winsearch(cMDI, 'lb.Programm');
  vHdl->wpCaption # Sel.Art.von.ArtNr;
  vHdl # Winsearch(cMDI, 'lb.Datumstext');
  vHdl->wpCaption # cnvad(Sel.von.Datum)+' - '+cnvad(Sel.bis.Datum);

  if (vDat=0.0.0) then begin
    vDat  # today;
    vTim  # 6:0;
  end;

  vHdl # WinSearch(cMDI, 'edDatum');
  vHdl->wpCaptiondate # vDat;
  vHdl # WinSearch(cMDI, 'edZeit');
  vHdl->wpCaptionTime # vTim;

  vHdl # WinSearch(cMDI, 'lbText');
```

```
  vTxt # TextOpen(20);
  vHdl->wpCustom # aint(vTxt);

  // Kalender bauen
//  TextAddLine(vTxt,'KALENDERGRUPPE '+aint(Rso.Gruppe));
  BA1_Planung_Subs:KTextBuild(vTxt, Rso.Gruppe, vDat);
//  TextWrite(vTxt, 'd:\debug\debug2.txt',_TextExtern);
//  _Recalc(vDat, vTim, false);

  // Anzeigen
  vDLPool->wpCurrentInt # 0;
  if (vPlanname='') then
    gTitle # 'NEUE '+cTitle
  else
    gTitle # cTitle + ' '+vPlanname;

  cMDI->wpCaption # gTitle;
  cMDI->WinUpdate(_WinUpdOn);
//  gMdiWorkbench->Winfocusset(false);
  cMDI->Winfocusset(true);
end;


//=========================================================================
//=========================================================================
sub  _AbleDrucken(aAble : logic);
```

```
local begin
  vI    : int;
  vHdl2 : int;
end
begin

  vHdl2 # Winsearch(gMDI, 'btDruck');
  if (vHdl2<>0) then begin
    vHdl2->wpDisabled # !aAble;
    if (aAble) then vHdl2->wpStyleButton # _WinStyleButtonTBar
    else vHdl2->wpStyleButton # _WinStyleButtonNormal;
  end;
  vHdl2 # Winsearch(gMDI, 'btDruck1');
  if (vHdl2<>0) then begin
    vHdl2->wpDisabled # !aAble;
    if (aAble) then vHdl2->wpStyleButton # _WinStyleButtonTBar
    else vHdl2->wpStyleButton # _WinStyleButtonNormal;
  end;
  vHdl2 # Winsearch(gMDI, 'btDruckAlle');
  if (vHdl2<>0) then begin
    vHdl2->wpDisabled # !aAble;
    if (aAble) then vHdl2->wpStyleButton # _WinStyleButtonTBar
    else vHdl2->wpStyleButton # _WinStyleButtonNormal;
  end;
end;
```

```
//=========================================================================
//=========================================================================
sub _SetSum(
  aAnz    : int;
  aDau    : int;
  aGew    : int;
)
local begin
  vAnzHdl : int;
  vDauHdl : int;
  vGewHdl : int;
  vT      : int;
  vH      : int;
end;
begin
  vAnzHdl # WinSearch(cMDI, 'lb.Summe.Anz');
  vDauHdl # WinSearch(cMDI, 'lb.Summe.Dauer');
  vGewHdl # WinSearch(cMDI, 'lb.Summe.Gew');

  vAnzHdl->wpCaption  # aint(aAnz);
  vDauHdl->wpCUSTOM   # aint(aDau);
  vGewHdl->wpCaption  # cnvai(aGew);

  vT # aDau / (24 * 60);
  aDau # aDau - (vT * 24 *60);
```

```
  vH # aDau / 60;
  aDau # aDau - (vH * 60);
  vDauHdl # WinSearch(cMDI, 'lb.Summe.Tag');
  vDauHdl->wpCaption # aint(vT);
  vDauHdl # WinSearch(cMDI, 'lb.Summe.H');
  vDauHdl->wpCaption # aint(vH);
  vDauHdl # WinSearch(cMDI, 'lb.Summe.min');
  vDauHdl->wpCaption # aint(aDau);
end;



//=========================================================================
//=========================================================================
sub _AddSum(
  aAnz    : int;
  aDauer  : int;
  aGew    : int;
)
local begin
  vAnz    : int;
  vDau    : int;
  vGew    : int;
  vAnzHdl : int;
  vDauHdl : int;
  vGewHdl : int;
  vT      : int;
```

```
  vH      : int;
end;
begin
  vAnzHdl # WinSearch(cMDI, 'lb.Summe.Anz');
  vDauHdl # WinSearch(cMDI, 'lb.Summe.Dauer');
  vGewHdl # WinSearch(cMDI, 'lb.Summe.Gew');

  vAnz # cnvia(vAnzHdl->wpCaption) + aAnz;
  vDau # cnvia(vDauHdl->wpCUSTOM) + aDauer;
  vGew # cnvia(vGewHdl->wpCaption) + aGew;

  vAnzHdl->wpCaption  # aint(vAnz);
  vDauHdl->wpCUSTOM   # aint(vDau);
  vGewHdl->wpCaption  # cnvai(vGew);

  vT # vDau / (24 * 60);
  vDau # vDau - (vT * 24 *60);
  vH # vDau / 60;
  vDau # vDau - (vH * 60);
  vDauHdl # WinSearch(cMDI, 'lb.Summe.Tag');
  vDauHdl->wpCaption # aint(vT);
  vDauHdl # WinSearch(cMDI, 'lb.Summe.H');
  vDauHdl->wpCaption # aint(vH);
  vDauHdl # WinSearch(cMDI, 'lb.Summe.min');
  vDauHdl->wpCaption # aint(vDau);
end;
```

```
//=========================================================================
//=========================================================================
sub _Recalc(
  aDat      : date;
  aTim      : time;
  aMitDauer : logic;
)
local begin
  vI    : int;
  vA,vB : alpha;
  vDat1 : date;
  vTim1 : time;
  vDat2 : date;
  vTim2 : time;
  vDau  : int;
  vDau2 : int;
  vTlg  : int;
  vHdl  : int;
  vTxt  : int;
  vTlgM : int;
  vGew  : int;
  vSDau : int;
  vSGew : int;
  vSAnz : int;
```

```
end;
begin
//  vTlgM # $edTelungszeit->wpCaptionint;

  vHdl # WinSearch(cMDI, 'lbText');
  vTxt # cnvia(vHdl->wpCustom);

  vDat1 # aDat;
  vTim1 # aTim;

  FOR vI # 1
  LOOP inc(vI)
  WHILE (vI<=WinLstDatLineInfo(cDLPlan, _WinLstDatInfoCount)) do begin
    WinLstCellGet(cDLPlan, vTlg , cClmTlg, vI);
    WinLstCellGet(cDLPlan, vDau , cClmDau, vI);
    WinLstCellGet(cDLPlan, vDau2, cClmDau2, vI);
    WinLstCellGet(cDLPlan, vGew , cClmGew, vI);
    if (aMitDauer) then begin
      vDau2 # 0;
      if (vI=1) then vDau2 # $edRuestzeit->wpCaptionint;
      vDau2 # vDau2 + (vTlg * vTlgM);
      WinLstCellSet(cDLPlan, vDau2, cClmDau2, vI);
      WinLstCellSet(cDLPlan, vDau2+vDau, cClmDauSum, vI);
    end;

    if (BA1_Planung_Subs:KTextFind(vTxt, var vDat1, var vTim1, vDau + vDau2, var vDat2, var vTim2)) the
```

```
      vA # cnvad(vDat1)+' '+cnvat(vTim1);
      vB # cnvad(vDat2)+' '+cnvat(vTim2);
    end
    else begin
      vA # cKeinKal;
      vB # cKeinKal;
      vDat2 # vDat1;
      vTim2 # vTim1;
      Lib_Berechnungen:TerminModify(var vDat2, var vTim2, cnvfi(vDau + vDau2));
    end;
    WinLstCellSet(cDLPlan, vA, cClmStart, vI);
    WinLstCellSet(cDLPlan, vB, cClmEnde, vI);

    vSGew # vSGEw + vGew;
    vSAnz # vSAnz + 1;
    vSDau # vSDAu + vDau+vDau2;

    // Ende vom Vorgänger ist Srart vom nächsten...
    vDat1 # vDat2;
    vTim1 # vTim2;
  END;

  _SetSum(vSAnz, vSDau, vSGew);
end;
```

```
//=========================================================================

//=========================================================================
sub _VorherigesWalzen(
  aTxt      : int;
  aSet      : logic;
  aPos      : int;    // Start 0
  var aVon  : int;    // start 0
  ) : int;
local begin
  Erx       : int;
  v702  : int;
  vVor  : int;
  vA    : alpha;
end;
begin

//if (BAG.P.Nummer=1514) and (BAG.P.Position>4) then debug('_sucheVG KEY702');

  if (BAG.P.Aktion<>C_BAG_Walz) then begin
//debug('ENDE');
    RETURN 0;
  end;

  if (aSet=false) then aVon # aVon + 1;

//debug('aVon : '+aint(aVon));
```

```
  v702 # RekSave(702);
  FOR Erx # RecLink(701,v702,2,_recFirst)    // Input loopen
  LOOP Erx # RecLink(701,v702,2,_recNext)
  WHILE (Erx<=_rLocked) do begin
    if (BAG.IO.Materialtyp<>c_IO_BAG) then CYCLE;
    Erx # RecLink(702, 701,2,_RecFirst);    // VonPos holen
    if (Erx<=_rLocked) and ((BAG.P.Aktion=C_BAG_Walz) or (BAG.P.Aktion='VORWAL')) then begin
      vVor # 1 + _VorherigesWalzen(aTxt, aSet, aPos-1, var aVon);
//debug('bekomme : '+aint(vVor));
    end;
    BREAK;
  END;
  RekRestore(v702);

  if (aSet) then begin
    vA # 'BA'+aint(BAG.P.nummer)+'/'+aint(BAG.P.Position)+'|'+aint(aPos)+' / '+aint(aVon);
//debug(vA);
    TextAddLine(aTxt,vA);
  end;

  RETURN vVor;
end;


//=========================================================================
//=========================================================================
```

```
sub _FolgendesWalzen(
  aTxt      : int;
  aSet      : logic;
  aPos      : int;    // Start 0
  var aVon  : int;    // start 0
  ) : int;
local begin
  Erx       : int;
  v702  : int;
  vNF   : int;
  vA    : alpha;
end;
begin

//if (BAG.P.Nummer=1514) and (BAG.P.Position>4) then debug('_sucheNF KEY702');

  if (BAG.P.Aktion<>C_BAG_Walz) then begin
//debug('ENDE');
    RETURN 0;
  end;

  v702 # RekSave(702);
  FOR Erx # RecLink(701,v702,3,_recFirst)    // Output loopen
  LOOP Erx # RecLink(701,v702,3,_recNext)
  WHILE (Erx<=_rLocked) do begin
    if (BAG.IO.Materialtyp<>c_IO_BAG) then CYCLE;
```

```
    if (BAG.IO.NachBAG=BAG.P.Nummer) then begin
      Erx # RecLink(702, 701,4,_RecFirst);    // NachPos holen
      if (Erx<=_rLocked) and ((BAG.P.Aktion=C_BAG_Walz) or (BAG.P.Aktion=c_Vorwalzen)) then begin
        if (aSet=false) then
          aVon # aVon + 1;
        vNF # 1 + _FolgendesWalzen(aTxt, aSet, aPos+1, var aVon);
  //debug('bekomme : '+aint(vVor));
      end;
    end;
    BREAK;
  END;
  RekRestore(v702);

    if (aSet) then begin
      vA # 'BA'+aint(BAG.P.nummer)+'/'+aint(BAG.P.Position)+'|'+aint(aPos)+' / '+aint(aVon);
//debug(vA);
      TextAddLine(aTxt,vA);
    end;

  RETURN vNF;
end;


//=========================================================================
//=========================================================================
sub _702NachDL(
```

```
  aDL       : int;
  aTxt      : int;
  aIstPlan  : logic;
);
local begin
  Erx       : int;
  vGuete    : alpha;
  vFest     : alpha;
  vGew      : int;
  vStich    : alpha;
  vStatus   : alpha;
  vVon      : alpha;
  vBis      : alpha;
  vInputD   : float;
  vInputB   : float;
  vOutputD  : float;
  vTlg      : int;
  vDauer    : int;
  vDauer2   : int;
  vX        : float;
  vZeile    : int;
  vI        : int;
  vA        : alpha;
  vGes      : int;
  vVor, vNach : int;
  vStart    : alpha;
```

```
  vEnde     : alpha;
  vRinge    : int;
  vCustom   : alpha;
  vBTol     : alpha;
  vOutputDTol : alpha;
  vTerm     : date;
  vTermText : alpha;
  vTerm2    : date;
  vTerm2Text : alpha;
  vTerm3    : date;
  vKW, vJahr  : word;

end;
begin

  // "Stiche" suchen
  vI # TextSearch(aTxt, 1, 1, _TextSearchCI, 'BA'+aint(BAG.P.Nummer)+'/'+aint(BAG.P.Position));
  if (vI>0) then begin
    vA # TextLineRead(aTxt, vI, 0);
    vStich # Str_Token(vA, '|',2);
  end
  else begin
    // Vorherige Walzen suchen:
    vVor  #  _VorherigesWalzen(aTxt, false, 0, var vGes);
    vNach # _FolgendesWalzen(aTxt, false,  0, var vGes);
    _VorherigesWalzen(aTxt, true, vVor+1, var vGes);
```

```
  _FolgendesWalzen(aTxt, true, vVor+1, var vGes);
  if (vGes>0) then begin
    vStich # aint(vVor+1)+' / '+aint(vGes);
  end;
end;
vStatus # Str_Token(BA1_Planung_Subs:GetStatus(aTxt),'|',2);



vTlg # -1;

FOR Erx # RecLink(701,702,2,_recFirst)    // Input loopen
LOOP Erx # RecLink(701,702,2,_recNext)
WHILE (Erx<=_rLocked) do begin
  if ((BAG.IO.Materialtyp=c_IO_Mat) or
    (BAG.IO.Materialtyp=c_IO_Theo) or
    (BAG.IO.Materialtyp=c_IO_BAG)) and (BAG.IO.VonFertigmeld=0) then begin
    vGuete  # "BAG.IO.Güte";
    vInputD # BAG.IO.Dicke;
    vInputB # BAG.IO.Breite;

    if (vTlg=-1) then
      vTlg # BAG.IO.Teilungen;
    if (vTlg<>BAG.IO.Teilungen) then
      vTlg # -2;

    RecBufClear(200);
```

```
    BA1_Planung_Subs:Get701Mat();
    vX # 0.0;
    if (Mat.Analysenummer<>0) then begin
      Lys.Analysenr # Mat.Analysenummer;
      Erx # RecReaD(231,1,0);
      if (Erx<=_rLocked) then
        vX # Lys.Zugfestigkeit;
    end;
    if (vFest<>'') then
      vFest # vFest + '/';

    vFest   # StrCut(vFest + anum(vX,3),1,80);
    vGew    # vGew + cnvif(BAG.IO.Plan.Out.GewN);
    vRinge  # vRinge + BAG.IO.Plan.Out.Stk;
  end;
END;



Erx # RecLink(703,702,4,_recFirsT);   // Fertigung holen
vOutputD  # BAG.F.Dicke;
vOutputDTol # BAG.F.DickenTol;
vBTol       # BAG.F.BreitenTol;   // !!!!

vDauer    # cnvif(BAG.P.Plan.Dauer);

vDauer2 # BA1_Planung_Subs:GetSonderDauer();
```

```
  vDauer  # vDauer - vDauer2;



  if (aIstPlan) then begin
    _AddSum(1, vDauer, vGew);
  end;
  vStart # cnvad(BAG.P.Plan.StartDat)+' '+cnvat(BAG.P.Plan.StartZeit);
  vEnde  # cnvad(BAG.P.Plan.EndDat)+' '+cnvat(BAG.P.Plan.EndZeit);

  if (BAG.P.Fenster.MinDat>0.0.0) then
    vVon # cnvad(BAG.P.Fenster.MinDat)+' '+cnvat(BAG.P.Fenster.MinZei);
// 20.12.2018 : nicht Fenster, sondern Termin laut JIT
//  if (BAG.P.Fenster.MaxDat>0.0.0) then
//    vBis # cnvad(BAG.P.Fenster.MaxDat)+' '+cnvat(BAG.P.Fenster.MaxZei);
  vBis # vStart;



  RecLink(700,702,1,_recFirst);   // Kopf holen, 20.08.2020 AH
  BA1_Planung_Subs:FindeKommissionsTermine(var vTerm, var vTerm2, var vTerm3);
  if (vTerm<>0.0.0) then begin
    Lib_Berechnungen:KW_Aus_Datum(vTerm, var vKW, var vJahr);
    vTermText # aint(vKW)+'/'+aint(vJahr);
  end;
  if (vTerm2<>0.0.0) then begin
    Lib_Berechnungen:KW_Aus_Datum(vTerm2, var vKW, var vJahr);
    vTerm2Text # aint(vKW)+'/'+aint(vJahr);
```

```
end;



aDL->WinLstDatLineAdd(RecInfo(702,_recId)); // NEUE ZEILE
vZeile # _WinLstDatLineLast;

aDL->WinLstCellSet(vStatus,          cClmStatus ,vZeile);
aDL->WinLstCellSet(vTlg,             cClmTlg    ,vZeile);
aDL->WinLstCellSet(vDauer,           cClmDau    ,vZeile);
aDL->WinLstCellSet(vDauer2,          cClmDau2   ,vZeile);
aDL->WinLstCellSet(vDauer2+vDauer,   cClmDauSum ,vZeile);
aDL->WinLstCellSet(vStart,           cClmStart  ,vZeile);
aDL->WinLstCellSet(vEnde,            cClmEnde   ,vZeile);
aDL->WinLstCellSet(BAG.P.Zusatz,     cClmProg   ,vZeile);
aDL->WinLstCellSet(vGuete,           cClmGuete  ,vZeile);
aDL->WinLstCellSet(vFest,            cClmFest   ,vZeile);
aDL->WinLstCellSet(aint(BAG.P.Nummer)+'/'+aint(BAG.P.Position), cClmBag, vZeile);
aDL->WinLstCellSet(vStich,           cClmStich    ,vZeile);
aDL->WinLstCellSet(vVon,             cClmVon      ,vZeile);
aDL->WinLstCellSet(vBis,             cClmBis      ,vZeile);
aDL->WinLstCellSet(vInputD,          cClmInputD   ,vZeile);
aDL->WinLstCellSet(vOutputD,         cClmOutputD  ,vZeile);
aDL->WinLstCellSet(vOutputDTol,      cClmOutputDTol,vZeile);
aDL->WinLstCellSet(vInputB,          cClmInputB   ,vZeile);
aDL->WinLstCellSet(vBTol,            cClmBTol     ,vZeile);
```

```
  aDL->WinLstCellSet(vRinge,          cClmRinge    ,vZeile);
  aDL->WinLstCellSet(vGew,            cClmGew      ,vZeile);

  aDL->WinLstCellSet(vTerm2Text,      cClmTerminW  ,vZeile);
  aDL->WinLstCellSet(vTermText,       cClmTerminZ  ,vZeile);


  vCustom # cnvaf(10000.0-vInputB,_FmtNumLeadZero|_FmtNumNoGroup,0,3,10);
  vCustom # vCustom + cnvaf(10000.0-vInputD,_FmtNumLeadZero|_FmtNumNoGroup,0,3,10);
  aDL->WinLstCellSet(vCustom,         cClmCustom   ,vZeile);
end;


//=========================================================================
//=========================================================================
sub _FindePlanname(aName : alpha) : alpha
local begin
  Erx       : int;
  vA    : alpha;
  vDat  : date;
  vKW   : word;
  vJahr : word;
  vI    : int;
end;
begin
//debugx(aName);
```

```
  // neuen Namen suchen...
  if (aName='') then begin
    // 1. Eintrag bestimmt den Namen
    WinLstCellGet(cDLPlan, vA,    cClmStart, 1);
    if (vA='') then RETURN '';
    vDat  # cnvda(Str_Token(vA,' ',1));
    if (vDat=0.0.0) then RETURN '';
    Lib_Berechnungen:KW_aus_Datum(vDat, var vKW, var vJahr);

    Erx # _rOK;
    FOR vI # 1
    LOOP inc(vI);
    WHILE (Erx<=_rMultikey) do begin
      if (vI=100) then RETURN '';
      aName # aint(vJahr)+'/'+aint(vKW)+'/'+aint(vI);
      RecBufClear(702);
      BAG.P.Plan.StartInfo # aName;
      Erx # RecRead(702,10,0);    // BA-Position mit diesem Namen suchen
//debugx('Suche '+aName+' = Erx');
    END;
  end;

//debugx('neuer Name:'+aName);
  RETURN aName;
end;
```

```
//=====================================================================

//=====================================================================

sub SaveAll(aPlanname : alpha) : logic

local begin

  Erx     : int;

  vI      : int;

  vID     : int;

  vTlg    : int;

  vFolge  : int;

  vDau    : int;

  vDau2   : int;

  vA,vB   : alpha;

  vDat    : date;

  vTim    : time;

  vTxt    : int;

  vRTF    : int;

  vCT1    : caltime;

  vCT2    : caltime;

end;

begin


  if ($btRefresh2->wpVisible) then begin

    Msg(99,'Bitte zuerst einen REFRESH durchführen!',0,0,0);

    RETURN false;

  end;
```

```
if (aPlanname='') and (WinLstDatLineInfo(cDLPlan, _WinLstDatInfoCount)<=0) then begin
  Msg(99,'Keine Einträge vorhanden!',0,0,0);
  RETURN false;
end;

vTxt # TextOpen(16);

// ABHÄNGIGKEITEN PRÜFEN ----------------------------------------------------
if (BA1_Planung_Subs:CheckAbhaenigkeiten(cMDI, cDLPlanName, vTxt, cClmStart, cClmEnde, cClmBA
  TextClose(vTxt);
  RETURN false;
end;


// KONFLIKTE FINDEN ---------------------------------------------------------
if (BA1_Planung_Subs:CheckKonflikte(cMDI, cDLPlanName, vTxt, cClmStart, cClmEnde, cClmBAG)=fals
  TextClose(vTxt);
  RETURN false;
end;

aPlanname # _FindePlanname(aPlanname);

TextClose(vTxt);
```

```
  // VERBUCHEN ----------------------------------------------------------------
  TRANSON;

  // Einträge mit RF löschen, wenn in Pool oder Filter ----------
  FOR vI # 1
  LOOP inc(vI)
  WHILE (vI<=WinLstDatLineInfo(cDLPool, _WinLstDatInfoCount)) do begin
    WinLstCellGet(cDLPool, vID ,  cClmRecId, vI);
    WinLstCellGet(cDLPool, vFolge,cClmFolge, vI);
    if (vFolge=0) then CYCLE;

    Erx # RecRead(702, 0,_recId,vID);
    If (Erx<>_rOK) then begin
      TRANSBRK;
      WinLstCellGet(cDLPool, vA,    cClmBAG, vI);
      Msg(99,'BA '+vA+' kann nicht verändert werden!',0,0,0);
      RETURN false;
    end;

    // ÄNDERN....
    PtD_Main:Memorize(702);
    RecRead(702,1,_RecLock);
//    BAG.P.Plan.StartInfo  # '';
    if (BAG.P.Plan.StartInfo<>cJIT) then BAG.P.Plan.StartInfo # ''; // 19.12.2018
    BAG.P.Reihenfolge     # 0;
    Erx # RekReplace(702);
```

```
    if (Erx=_rOK) then
      PtD_Main:Forget(702)
    else
      PtD_Main:Memorize(702);
  END;

  FOR vI # 1
  LOOP inc(vI)
  WHILE (vI<=WinLstDatLineInfo(cDLPoolFilter, _WinLstDatInfoCount)) do begin
    WinLstCellGet(cDLPoolFilter, vID ,  cClmRecId, vI);
    WinLstCellGet(cDLPoolFilter, vFolge,cClmFolge, vI);
    if (vFolge=0) then CYCLE;

    Erx # RecRead(702, 0,_recId,vID);
    If (Erx<>_rOK) then begin
      TRANSBRK;
      WinLstCellGet(cDLPoolFilter, vA,    cClmBAG, vI);
      Msg(99,'BA '+vA+' kann nicht verändert werden!',0,0,0);
      RETURN false;
    end;

    // ÄNDERN....
    PtD_Main:Memorize(702);
    RecRead(702,1,_RecLock);
//    BAG.P.Plan.StartInfo  # '';
    if (BAG.P.Plan.StartInfo<>cJIT) then BAG.P.Plan.StartInfo # ''; // 19.12.2018
```

```
  BAG.P.Reihenfolge     # 0;
  BAG.P.Plan.ManuellYN  # n;
  Erx # RekReplace(702);
  if (Erx=_rOK) then
    PtD_Main:Forget(702)
  else
    PtD_Main:Memorize(702);
END;



// PLANUNG --------
FOR vI # 1
LOOP inc(vI)
WHILE (vI<=WinLstDatLineInfo(cDLPlan, _WinLstDatInfoCount)) do begin
  WinLstCellGet(cDLPlan, vID ,  cClmRecId, vI);
  WinLstCellGet(cDLPlan, vTlg , cClmTlg, vI);
  WinLstCellGet(cDLPlan, vDau , cClmDau, vI);
  WinLstCellGet(cDLPlan, vDau2, cClmDau2, vI);
  WinLstCellGet(cDLPlan, vA,    cClmStart, vI);
  WinLstCellGet(cDLPlan, vB,    cClmEnde, vI);

  Erx # RecRead(702, 0,_recId,vID);
  If (Erx<>_rOK) then begin
    TRANSBRK;
    WinLstCellGet(cDLPlan, vA,    cClmBAG, vI);
```

```
    Msg(99,'BA '+vA+' kann nicht verändert werden!',0,0,0);
    RETURN false;
  end;

  If (vA=cKeinKal) or (vB=cKeinKal) then begin
    TRANSBRK;
    Msg(99,'Bitte Ressourcen-Kalender erst richtig ausfüllen!',0,0,0);
    RETURN false;
  end;

  // ÄNDERN....
  PtD_Main:Memorize(702);
  RecRead(702,1,_RecLock);
  vDat # cnvda(Str_Token(vA,' ',1));
  vTim # cnvta(Str_Token(vA,' ',2));
  BAG.P.Plan.StartDat   # vDat;
//Lib_Debug:Protokoll('!BSP_Log_Komisch', 'Set BA-Termin '+aint(BAG.P.Nummer)+'/'+aint(BAG.P.Positio
  BAG.P.Plan.StartZeit  # vTim;
  BAG.P.Plan.Dauer      # cnvfi(vDau + vDau2);
  vDat # cnvda(Str_Token(vB,' ',1));
  vTim # cnvta(Str_Token(vB,' ',2));
  BAG.P.Plan.EndDat     # vDat;
  BAG.P.Plan.EndZeit    # vTim;
  BAG.P.Plan.StartInfo  # aPlanname;
  BAG.P.Reihenfolge     # vI;
  BAG.P.Plan.ManuellYN  # y;
```

```
    if (BAG.P.Plan.StartInfo=cJIT) then BAG.P.Plan.StartInfo # ''; // 19.12.2018
    Erx # RekReplace(702);
    if (Erx=_rOK) then
      PtD_Main:Forget(702)
    else
      PtD_Main:Memorize(702);

    BA1_Planung_Subs:SetSonderDauer('', vDau2);

//debugx('save KEY702 mit Erx');

    FOR Erx # RecLink(701,702,2,_recFirst)    // Input loopen
    LOOP Erx # RecLink(701,702,2,_recNext)
    WHILE (Erx<=_rLocked) do begin

      if (BAG.IO.Teilungen=vTLG) then CYCLE;    // 29.10.2019

      if ((BAG.IO.Materialtyp=c_IO_Mat) or
        (BAG.IO.Materialtyp=c_IO_Theo) or
        (BAG.IO.Materialtyp=c_IO_BAG)) and (BAG.IO.VonFertigmeld=0) then begin
        Erx # RecRead(701,1,_recLock);
        BAG.IO.Teilungen      # vTlg;
        BAG.IO.AutoTeilungYN  # false;

        Erx # BA1_IO_Data:Replace(_recUnlock,'MAN');
        if (Erx<>_rOK) then begin
```

```
      TRANSBRK;
      WinLstCellGet(cDLPlan, vA,    cClmBAG, vI);
      Msg(99,'BA '+vA+' : Einsatz kann nicht verändert werden!',0,0,0);
      RETURN false;
    end;
    // Output aktualisieren
    if (BA1_F_Data:UpdateOutput(701,n)=false) then begin
      TRANSBRK;

      vA # gTitle;
      gTitle # 'BA '+aint(BAG.P.Nummer)+'/'+aint(BAG.P.Position);
      Error(701010,'');
      ErrorOutput;
      gTitle # vA;

      ErrorOutput;
      RETURN false;
    end;
    // alle Fertigungen neu errechnen
    if ("BAG.P.Typ.1In-1OutYN") or
       ("BAG.P.Typ.1In-yOutYN") then
      BA1_P_Data:ErrechnePlanmengen();
  end;
 END;  // Input

END;
```

```
  TRANSOFF;

  Msg(99,'Erfolgreich als Planung '+aPlanname+' gespeichert!',0,0,0);

  $bt.Save->wpcustom # '';    // Änderung vermerken
  _AbleDrucken(true);

  RETURN true;
end;


//=========================================================================
//=========================================================================
sub _SavenMussSein()
begin
  $bt.Save->wpcustom # 'change';    // Änderung vermerken
  _AbleDrucken(false);
end;


//=========================================================================
//=========================================================================
sub _RefreshMussSein(aDL : int)
begin
  if (aDL=cDlPool) then RETURN;
```

```
  _SavenMussSein();
  $btRefresh2->wpVisible # true;
end;


//=========================================================================
//=========================================================================
sub RefreshTermine()
local begin
  vHdl  : int;
  vDat  : date;
  vTim  : time;
  vTxt  : int;
end;
begin

  WinLayer(_WinLayerStart, gFrmMain, 20000, 'Berechne...', _WinLayerDarken);

  vHdl # WinSearch(cMDI, 'edDatum');
  vDat # vHdl->wpCaptiondate;
  vHdl->wpColBkg # _WinColWindow;

  vHdl # WinSearch(cMDI, 'edZeit');
  vTim # vHdl->wpCaptionTime;
```

```
  // Zeiten erstellen...
  vHdl # WinSearch(cMDI, 'lbText');
  vTxt # cnvia(vHdl->wpCustom);
  BA1_Planung_Subs:KTextBuild(vTxt, Rso.Gruppe, vDat);

  // neu einplanen...
  cDLPlan->wpAutoUpdate # false;
  _Recalc(vDat, vTim, true);
  cDLPlan->Winupdate(_WinUpdOn, _WinLstFromTop| _WinLstPosSelected);

  cDLPlan->winFocusset(true);

  $btRefresh2->wpVisible # false;

  WinLayer(_WinLayerEnd);

end;


//=========================================================================
//=========================================================================
sub _IsFiltered(aStatus : alpha) : logic
begin

  if (StrFind(aStatus,'bereit',1)>0) then
    RETURN ($cb.Filter.Bereit->wpCheckState=_WinStateChkChecked);
```

```
  if (StrFind(aStatus,'warte',1)>0) then
    RETURN ($cb.Filter.Theo->wpCheckState=_WinStateChkChecked);
  if (StrFind(aStatus,'fertig',1)>0) then
    RETURN ($cb.Filter.Erledigt->wpCheckState=_WinStateChkChecked);

  RETURN ($cb.Filter.Zum->wpCheckState=_WinStateChkChecked);
end;



//=========================================================================
//=========================================================================
sub RefreshFilter()
local begin
  vI    : int;
  vA    : alpha;
  vOK   : logic;
end;
begin

  WinLayer(_WinLayerStart, gFrmMain, 20000, 'Filterung...', _WinLayerDarken);
  cDLPool->wpAutoUpdate # false;

  // Pool in Ablage schieben...
  FOR vI # 1
  LOOP inc(vI)
  WHILE (vI<=WinLstDatLineInfo(cDLPool, _WinLstDatInfoCount)) do begin
```

```
    WinLstCellGet(cDLPool, vA , cClmStatus, vI);
    if (_IsFiltered(vA)=false) then CYCLE;
    // in ABLAGE schieben...
    Lib_DataList:Move(cDLPool, vI, cDlPoolFilter, 1);
//cDLPlan->WinLstDatLineRemove( vI);
    dec(vI);
  END;


  // Ablage in Pool schieben...
  FOR vI # 1
  LOOP inc(vI)
  WHILE (vI<=WinLstDatLineInfo(cDLPoolFilter, _WinLstDatInfoCount)) do begin
    WinLstCellGet(cDLPoolFilter, vA , cClmStatus, vI);
    if (_IsFiltered(vA)) then CYCLE;
    // in POOL schieben...
    Lib_DataList:Move(cDLPoolFilter, vI, cDlPool, 1);
    dec(vI);
  END;

  cDLPool->WinUpdate( _winUpdOn, _winLstPosTop);
  WinLayer(_WinLayerEnd);

end;
```

```
//=========================================================================
// EvtInit
//         Initialisieren der Applikation
//=========================================================================
sub EvtInit (
  aEvt      : event;
): logic
begin
  gTitle        # Translate( cTitle );
  gMenuName     # cMenuName;
  gMenuEvtProc  # here+':EvtMenuCommand';
  Mode      # c_modeEdList;
//  App_Main:EvtInit( aEvt );

  Lib_GuiCom:RecallList(cDLPlan, cTitle);     // Usersettings holen
  Lib_GuiCom:RecallList(cDLPool, cTitle);     // Usersettings holen
  Lib_GuiCom:RecallList(cDLPlan, cTitle);     // Usersettings holen DOPPELT weil sonst clmSTATUS nicht
  Lib_GuiCom:RecallList(cDLPool, cTitle);     // Usersettings holen
  App_Main:EvtInit( aEvt );
end;


//=========================================================================
//=========================================================================
sub StartEdit()
begin
```

```
  Lib_DataList:StartListEdit(cDLPlan, c_ModeEdListEdit, 0, _winLstEditClearChanged );
end;



//=========================================================================
//=========================================================================
sub _Resort(
  aDL   : int;
  aClm  : int)
local begin
  Erx       : int;
  vI    : int;
  vB,vD : float;
  vA    : alpha;
  vID   : int;
  vTree : int;
  vItem : int;
  vTxt  : int;
end;
begin
  // 16.04.2019 AH: Planung temporär umsortieren - NICHT permanent
  if (aDL<>cDLPool) then begin
    Winupdate(aDL,_winupdoff);
    vTree # CteOpen(_CteTree);
    FOR vI # 1
    LOOP inc(vI)
```

```
WHILE (vI<=WinLstDatLineInfo(aDL, _WinLstDatInfoCount)) do begin
  WinLstCellGet(aDL, vID, cClmRecID, vI);
  WinLstCellGet(aDL, vB, cClmInputB, vI);
  WinLstCellGet(aDL, vD, cClmInputD, vI);

  //Erx # RecRead(702, 0, _recID, vID); // BA-Pos holen
  vA # cnvaf(10000.0 - vB,_FmtNumLeadZero|_FmtNumNoGroup,0,3,10);
  vA # vA + cnvaf(10000.0 - vD,_FmtNumLeadZero|_FmtNumNoGroup,0,3,10);

  vTree->CteInsertItem(vA + '|'+aint(vID), vID, vA);
END;

aDL->WinLstDatLineRemove(_WinLstDatLineAll);
vTxt # TextOpen(20);
FOR  vItem # vTree->CteRead(_cteFirst );
LOOP vItem # vTree->CteRead(_cteNext, vItem );
WHILE ( vItem != 0 ) DO BEGIN
  vID # vItem->spId;
  Erx # RecRead(702, 0, _recID, vID); // BA-Pos holen
  _702NachDl(aDL, vTxt, false);
END;
TextClose(vTxt);
CteClear(vTree, true);
CteClose(vTree);
winupdate(aDL,_WinupdOn);
RETURN;
```

```
  end;



  Winupdate(aDL,_winupdoff);
  aClm->wpClmSortFlags # _WinClmSortFlagsAutoActive|_WinClmSortFlagsAutoSelected;
  winupdate(aDL,_WinupdSort);
  winupdate(aDL,_WinupdOn);
//  Winupdate(aDL, _Winupdon|_winupdSort, _WinLstFromFirst);
//  aClm->wpClmSortFlags # 0;
  RETURN;
end;



//=========================================================================
//  RecDel
//          Satz soll gelöscht werden
//=========================================================================
sub RecDel()
local begin
  vID     : int;
  vDau    : int;
  vGew    : int;
  vHdl    : int;
  vItem   : int;
  vNr     : int;
end;
```

```
begin

  if (Msg(99,'Sollen die markierten Einträge wieder in den Pool gesetzt werden?',_WinIcoQuestion,_WinDia

  REPEAT
    vHdl # cDLPlan->wpSelData;
    if (vHdl<>0) then begin
      vHdl # vHdl->wpData(_WinSelDataCteTree);
      if (vHdl<>0) then begin
        vItem # vHdl->CteRead(_CteFirst);
//      LOOP vItem # vHdl->CteRead(_CteNext, vItem);
//      WHILE (vItem<>0) do begin
//debugx(aint(vItem->spid)+' '+vItem->spName+' : '+vItem->spCustom);
        if (vItem>0) then begin
          vNr # vItem->spID;
          WinLstCellGet(cDLPlan, vID  , cClmRecId, vNr);
          WinLstCellGet(cDLPlan, vDau , cClmDau, vNr);
          WinLstCellGet(cDLPlan, vGew , cClmGew, vNr);
          _AddSum( -1, -vDau, -vGew);
//          cDLPlan->WinLstDatLineRemove( vNr );

          Lib_DataList:Move(cDLPlan, vNr, cDLPool, 1);

          CYCLE;
        end;
      end;
```

```
    end;
    BREAK;
  UNTIL (1=1);

  cDLPlan->WinMsdInsert(cDLPlan->wpCurrentInt);
  cDLPlan->WinUpdate( _winUpdOn, _winLstPosTop );

  _RefreshMussSein(cDLPlan);
end;



//=========================================================================
//=========================================================================
sub EvtChanged(
  aEvt                : event;        // Ereignis
) : logic;
begin
  _RefreshMussSein(cDLPlan);
  RETURN(true);
end;



//=========================================================================
//  EvtMenuCommand
//            Fokus vom Objekt wegbewegen
//=========================================================================
```

```
sub EvtMenuCommand (
  aEvt            : event;
  aMenuItem       : int;
) : logic
local begin
  Erx       : int;
  vHdl  : int;
  vID   : int;
end;
begin

  if (aMenuItem->wpName='Mnu.ZumBA') then begin
    vHdl # WinfocusGet();
    if (vHdl=cDLPlan) or (vHdl=cDLPool) then begin
      if (vHdl->wpCurrentInt>0) then begin
        WinLstCellGet(vHdl, vID, cClmRecID, vHdl->wpCurrentInt);
        Erx # RecRead(702, 0, _recID, vID); // BA-Pos holen
        if (Erx<=_rLocked) then begin
          Erx # RecLink(700,702,1,_recFirst); // BA holen
          gMDI # Lib_GuiCom:AddChildWindow(gMDI,'BA1.Combo.Verwaltung','',y);
          VarInstance(WindowBonus,cnvIA(gMDI->wpcustom));
          w_Command  # 'REPOS';
          w_Cmd_Para # AInt(vID);
          Lib_GuiCom:RunChildWindow(gMDI);
          RETURN true;
        end;
```

```
      end;
    end;
  end;


  if (aMenuItem->wpName='Mnu.DL.Refresh') then begin
    if (aEvt:Obj=cDLPlan) then
      RefreshTermine()
    else //if (aEvt:Obj=cDLPool) then
      RefreshFilter()
    RETURN true;
  end;

  if (aMenuItem->wpName='Mnu.DL.Delete') then begin
    vHdl # WinfocusGet();
    if (vHdl=cDLPlan) then
      RecDel();
    RETURN true;
  end;

  RETURN Lib_Datalist:EvtMenuCommand( aEvt, aMenuItem );
end;


//=========================================================================
//  EvtLstDataInit
```

```
//
//=========================================================================
sub EvtLstDataInit (
  aEvt            : event;
  aId             : int;
) : logic
local begin
  vA    : alpha;
  vHdl  : int;
end;
begin
  Lib_DataList:EvtLstDatainit(aEvt, aID);

  aEvt:Obj->WinLstCellSet(aId, cClmFolge, aId); // lfd. Zeilennummer

  // Status...
  vHdl # Winsearch(aEvt:obj, 'clmStatus');
  WinLstCellGet(aEvt:Obj, vA , cClmStatus, aId);
  if (StrFind(vA,'bereit',1)>0) then
    vHdl->wpClmColBkg # _WinColLightGray
  else if (StrFind(vA,'warte',1)>0) then
    vHdl->wpClmColBkg # _WinColLightYellow
  else if (StrFind(vA,'fertig',1)>0) then
    vHdl->wpClmColBkg # RGB(200,255,200)
  else
    vHdl->wpClmColBkg # RGB(200,200,255);
```

```
  // Termin...
  vHdl # Winsearch(aEvt:obj, 'clmTerminende');
  WinLstCellGet(aEvt:Obj, vA , cClmStart, aId);
  if (vA=cKeinKal) then
    vHdl->wpClmColBkg # _WinColLightRed;
  WinLstCellGet(aEvt:Obj, vA , cClmEnde, aId);
  if (vA=cKeinKal) then
    vHdl->wpClmColBkg # _WinColLightRed;
end;



//=========================================================================
//  EvtLstEditCommit
//
//=========================================================================
sub EvtLstEditCommit (
  aEvt            : event;
  aColumn         : int;
  aKey            : int;
  aFocusObject    : int;
) : logic
local begin
  Erx       : int;
  vA    : alpha(1000);
  vHdl  : int;
```

```
  vI,vJ : int;
  vDau  : int;
  vMax  : int;
end;
begin

  // 18.12.2018 AH: Sofort Editieren vom Programm
  if (aColumn->wpname='clmProgramm') then begin
    WinLstCellGet(aEvt:Obj, vI, cClmRecId, _WinLstDatLineCurrent);
    vHdl # Wininfo(aEvt:obj,_WinLstEditObject);
    vA # vHdl->wpcaption;
    Erx # RecRead(702, 0,_recId,vI);
    if (Erx<=_rLocked) then begin
      PtD_Main:Memorize(702);
      RecRead(702,1,_RecLock);
      BAG.P.Zusatz # StrCut(vA,1,32);
      Erx # RekReplace(702);
      if (Erx=_rOK) then
        PtD_Main:Forget(702)
      else
        PtD_Main:Memorize(702);
    end;
    Lib_Datalist:EvtLstEditCommit(aEvt, aColumn, aKey, aFocusObject);
    RETURN true;
  end;
```

```
  if (aColumn->wpname='clmFolge') then begin
    WinLstCellGet(aEvt:Obj, vI, cClmFolge, _WinLstDatLineCurrent);
    vHdl # Wininfo(aEvt:obj,_WinLstEditObject);
    vI # vHdl->wpcaptionInt;
    vMax # aEvt:Obj->WinLstDatLineInfo(_WinLstDatInfoCount);
    if (vI<0) then vI # 1
    else if (vI>vMax) then vI # vMax;
    vHdl->wpCaptionInt # vI;
    vJ # aEvt:Obj->wpCurrentInt;

    if (vI<>vJ) then begin
      aEvt:Obj->wpCurrentInt # 0;
      Lib_DataList:Move(aEvt:Obj, vJ, aEvt:obj, vI);
//      WinFocusSet($edRuestzeit, true);
//Winupdate(aEvt:obj, _WinUpdOn, _WinLstRecDoSelect);
//      _move(aEvt:obj, vJ, vI);
//      aEvt:Obj->wpautoupdate # false;
      aEvt:Obj->wpMultiselect # false;
      aEvt:Obj->wpCurrentInt # vI;
//aEvt:Obj->WinMsdInsert(vI);
      aEvt:Obj->wpMultiselect # true;
      aEvt:Obj->WinMsdInsert(vI);

//debugx('sel:'+aint(vI));
      //winFocusset(aEvT:Obj, true);
//      Winupdate(aEvt:obj, _WinUpdOn, _WinLstPosSelected);// _WinLstRecDoSelect);
```

```
//      aEvt:Obj->wpautoupdate # true;
//  aEvt:obj->wpColFocusBkg    # "Set.Col.RList.Cursor";
      _RefreshMussSein(aEvt:obj);
      RETURN true;
    end;
  end;

  if (aColumn->wpname='clmDauer') then begin
    WinLstCellGet(aEvt:Obj, vDau, cClmDau, _WinLstDatLineCurrent);
    vHdl # Wininfo(aEvt:obj,_WinLstEditObject);
    vDau # (vHdl->wpcaptionInt) - vDau; // Delta
    _AddSum(0, vDau, 0);
  end;

  Lib_Datalist:EvtLstEditCommit(aEvt, aColumn, aKey, aFocusObject);

  _RefreshMussSein(aEvt:obj);

  RETURN true;
end;



//=========================================================================
// EvtClose
//             Schliessen eines Fensters
//=========================================================================
```

```
sub EvtClose (
  aEvt            : event;
) : logic
local begin
  vHdl        : int;
  vI          : int;
  vAnz        : int;
  v703        : int;
end;
begin

  if ($bt.Save->wpcustom<>'') then begin
    if (Msg(99,'Alle Änderungen verwerfen?',_WinIcoQuestion,_WinDialogOkCancel,2) <> _Winidok) then R
  end;

  // Aufräumen...
  vHdl # WinSearch(cMDI, 'lbText');
  TextClose(cnvia(vHdl->wpCustom))

  Lib_GuiCom:RememberList(cDLPlan, cTitle);
  Lib_GuiCom:RememberList(cDLPool, cTitle);
  Lib_GuiCom:RememberWindow(aEvt:obj);

  RETURN true;
end;
```

```
//=============================================================================
//  EvtClicked
//
//=============================================================================
sub EvtClicked (
  aEvt            : event
) : logic
local begin
  vColumn         : int;
  vColType        : int;
end;
begin

  case ( aEvt:obj->wpName ) of
    'btDruck1' : begin
      BA1_Planung_Subs:Druck1($dl.Plan);
    end;
    'btDruckAlle' : begin
      BA1_Planung_Subs:DruckAll($dl.Plan);
    end;

    'btRefresh', 'btRefresh2' :
      RefreshTermine();

    'bt.RefreshFilter' :
```

```
      RefreshFilter();


    'bt.Save' : begin
      if (SaveAll($lbNr->wpCustom)) then begin
        gMDI->winclose();
        gSelected # 1;
        StartInner();
      end;
    end;
  end;
end;



//=========================================================================
//=========================================================================
sub EvtMouseItem(
  aEvt                : event;        // Ereignis
  aButton             : int;          // Maustaste
  aHitTest            : int;          // Hittest-Code
  aItem               : handle;       // Spalte oder Gantt-Intervall
  aID                 : bigint;       // RecID bei RecList / Zelle bei GanttGraph / Druckobjekt bei PrtJobPreview
) : logic;
local begin
  Erx       : int;
  vID     : int;
  vA      : alpha;
```

```
end;
begin
/**
  // Doppelklick?
//  if ((aButton & _WinMouseDouble)>0) then begin
//  end;
  if ( aId > 0 ) and ( aItem > 0 ) and ( aButton = _winMouseLeft | _winMouseDouble ) then begin
//    if ( aItem->wpCustom != '_SKIP' ) then begin
//    end;

    WinLstCellGet(aEvt:Obj, vID, cClmRecID, aID);
    Erx # RecRead(702, 0, _recID, vID); // BA-Pos holen

    Erx # RecLink(700,702,1,_recFirst); // BA holen
    gMDI # Lib_GuiCom:AddChildWindow(gMDI,'BA1.Combo.Verwaltung','',y);
//debugx('set repos');
    VarInstance(WindowBonus,cnvIA(gMDI->wpcustom));
    w_Command  # 'REPOS';
    w_Cmd_Para # AInt(vID);

    Lib_GuiCom:RunChildWindow(gMDI);
    RETURN true;
  end;
**/
  // JumpTo
  if (aButton = _winMousemiddle ) and ( aHitTest = _winHitLstView ) then begin
```

```
  if (aID=0) or (aItem=0) then RETURN true;
  if (aItem->wpname='clmBAG') then begin
    WinLstCellGet(aEvt:Obj, vA, cClmBAG, aID);
    BAG.Nummer # cnvia(Str_Token(vA,'/',1));
    BAG.P.Nummer    # BAG.Nummer;
    BAG.P.Position # cnvia(Str_Token(vA,'/',2));
    Erx # RecRead(700,1,0);
    if (Erx>_rMultikey) then RETURN true;
    Erx # RecRead(702,1,0);
    if (Erx>_rMultikey) then RETURN true;
    gMDI # Lib_GuiCom:AddChildWindow(gMDI,'BA1.Combo.Verwaltung','',y);
    Lib_GuiCom:RunChildWindow(gMDI);
  end
  RETURN true;
end;

if (aHitTest=_winHitLstHeader) and (aEvt:Obj<>0) and (aEvt:Obj<>cDLPool) and (aItem<>0) and (aItem->
  if (Msg(99,'Reihenfolge wirklich umsortieren?',_WinIcoQuestion, _WinDialogYesNo,2)=_WinIdYes) then
    _Resort(aEvt:Obj, Winsearch(aEvt:Obj, 'clmCustom'));
    RETURN false;
  end;
end;

if (aHitTest=_winHitLstHeader) and (aEvt:Obj<>0) and (aEvt:Obj=cDLPool) and (aItem<>0) and (aItem->w
  if (Msg(99,'Reihenfolge wirklich umsortieren?',_WinIcoQuestion, _WinDialogYesNo,2)=_WinIdYes) then
    _Resort(aEvt:Obj, Winsearch(aEvt:Obj, 'clmCustom'));
```

```
      RETURN true;
    end;
  end;

  RETURN Lib_Datalist:EvtMouseItem(aEvt, aButton, aHitTest, aItem, aID);
end;



//=========================================================================
//=========================================================================
sub EvtKeyItem(
  aEvt                : event;        // Ereignis
  aKey                : int;          // Taste
  aID                 : bigint;       // RecID bei RecList, Node-Deskriptor bei TreeView, Focus-Objekt bei Frame
) : logic;
begin

  // DELETE nur in PLanung
  if ( aKey = _WinKeyDelete) then begin
    if (aEvt:Obj=cDLPlan) then
      RecDel();
  end;

  // EDIT nur in Planung...
  if (aKey = _WinKeyTab) or ( aKey = _winKeyReturn ) then begin
    if (aEvt:Obj=cDLPlan) then
```

```
    RETURN Lib_DataList:EvtKeyItem(aEvt, aKey, aID);
  end;



  RETURN true;
end;



//=========================================================================
//=========================================================================
sub EvtDragInit(
  aEvt                : event;    // Ereignis
  aDataObject         : handle;   // Drag-Datenobjekt
  aEffect             : int;      // Rückgabe der erlaubten Effekte (_WinDropEffectNone = Cancel)
  aMouseBtn           : int;      // Verwendete Maustasten (optional)
  aDataPlace          : handle;
) : logic;
begin
 BA1_Planung_Subs:EvtDragInit(aEvt, aDataObject, var aEffect, aMouseBtn, aDataPlace, cModulName);
 RETURN true;
end;



//=========================================================================
//  EvtDrop
//=========================================================================
```

```
sub EvtDrop(
  aEvt              : event;    // Ereignis
  aDataObject       : handle;   // Drag-Datenobjekt
  aDataPlace        : handle;   // DropPlace-Objekt
  aEffect           : int;      // Eingabe: vom Benutzer gewählter Effekt, Ausgabe: durchgeführter Effekt
  aMouseBtn         : int;      // Verwendete Maustasten
) : logic;
local begin
  vData       : int;
  vItem       : int;
  vLine       : int;
  vPlace      : int;
  vA          : alpha;
  vVon, vNach : int;
  vMin        : int;
  vI          : int;
  vID         : int;
  vRunter     : logic;
  vPre,vPost  : int;
  vAnz        : int;
  vDL1, vDL2  : int;
  vDau, vDau2 : int;
  vGew        : int;
end;
begin
```

```
  //vHdl # cDL;
//  vDL1 # aEvt:Obj;

  if (aDataObject->wpFormatEnum(_WinDropDataUser)) then begin
    if (aDataObject->wpname=cModulName) then begin
      vDL1 # cnvia(aDataObject->wpcustom);
      vDL2 # aEvt:Obj;
      aEffect # _WinDropEffectCopy | _WinDropEffectMove;
      vData # aDataObject->wpData(_WinDropDataUser);
      vData # vData->wpData;
      if (vData=0) then RETURN false;

      vLine   # aDataPlace->wpArgInt(0);
      vPlace  # aDataPlace->wpDropPlace;

      // Einfügeposition.
      case vPlace of
        _WinDropPlaceAppend   : begin
        inc(vLine);//vA # 'NACH';//  inc(vLine);
        end;
      end;

// if (vPlace=_WinDropPlaceThis)   => Maus AUF einem Eintrag
      vDL1->winupdate(_winupdoff);
      if (vDL1<>vDL2) then
        vDL2->winupdate(_winupdoff);
```

```
      vMin # 32000;
      FOR vItem # vData->CteRead(_CteFirst)
      LOOP vItem # vData->CteRead(_CteNext, vItem)
      WHILE (vItem<>0) do begin

        vVon  # vItem->spid;
        vNach # vLine;
//debugx(aint(vVon)+' nach '+aint(vNach));
        if (vDL1=vDL2) then begin
          if (vVon=vNach) then CYCLE;
          vRunter # vVon<vNach;
          if (vRunter) then begin
  //          if (vPlace=_WinDropPlaceThis) then
            vVon  # vVon - vPre;
            vNach # vNach - 1;              // wegen REMOVE
          end
          else begin
            vNach # vNach + vPost;
          end;
          if (vNach=0) then CYCLE;
        end
        else begin  // Pool <-> Plan oder Plan <-> Pool
          vVon # vVon - vPre;
        end;
```

```
      WinLstCellGet(vDL1, vDau , cClmDau, vVon);
      WinLstCellGet(vDL1, vDau2, cClmDau2, vVon);
      WinLstCellGet(vDL1, vGew , cClmGew, vVon);

//debugx(aint(vVon)+' nach '+aint(vNach));
      Lib_DataList:Move(vDL1, vVon, vDL2, vNach);

      if (vDL1=vDL2) then begin
        if (vRunter) then
          inc(vPre)
        else
          inc(vPost);
      end
      else begin    // Pool <-> Plan
        inc(vPre);
        if (vDL1=cDlPool) then
          _AddSum(1, vDau, vGew);
        else
          _AddSum(-1, -vDau, -vGew);
      end;
//debugx('  '+aint(vVon)+' nach '+aint(vNach));
    END;

    _RefreshMussSein(vDL1);
    vDL1->WinUpdate(_WinUpdOn, _WinLstFromTop);
    vDL1->WinUpdate(_WinUpdSort);
```

```
      if (vDL1<>vDL2) then begin
        _RefreshMussSein(vDL2);
        vDL2->WinUpdate(_WinUpdOn, _WinLstFromTop);
        vDL2->WinUpdate(_WinUpdSort);
      end;
    end;
  end;

  RETURN(true);
end;



//=========================================================================
//=========================================================================
//=========================================================================
//=========================================================================
```