```
@A+
//===== Business-Control ============================================
//
//  Prozedur    BA1_F_Data
//                  OHNE E_R_G
//  Info
//
//
//  21.06.2004  AI  Erstellung der Prozedur
//  30.12.2009  AI  Teilungszahl bei Stückzahlberechnung wieder aktiviert
//  08.01.2010  AI  Ausführungen auch in geplante Outputs in die zwei Felder übernehmen
//  04.11.2010  AI  Erweiterung für LFA-MultiLFS
//  09.12.2010  AI  Tafeln etc. runden Stückzahl ab
//  26.01.2011  AI  Schopf nur ab 1 kg
//  23.11.2011  AI  Änderungen des Einsatzes bei XzuY refreshed Output
//  09.01.2012  AI  Prj. 1161/386: RESTCOIL ist Differenz der Einsatz Stückzahl
//  01.02.2012  AI  ALLE XzuY-Reste werden gleich behandelt Prj. 1326/200
//  29.02.2012  AI  UpdateOutput: addiert die Fertigungen vom Spalten neu
//  16.03.2012  AI  InsertFahrt nimmt Kommission ggf. von IO auf
//  25.06.2012  AI  "Errechneplanmengen": alle 1zu1 AGs nehmen Einsatz Stk + Gewicht
//  25.09.2012  ST  "PassendesMatAusAuftrag" hinzugefügt
//  10.10.2012  AI  WirdEigenYN=true setzt bei Lohn NICHT die BAG.IO.Kommission
//  26.10.2012  AI  "Output" trägt zunächst leere Kommission, die per Bedingungen gefüllt wird
//  07.06.2013  AI  "BerechnePlanmenge" bei Tafeln von Artikel aus ArtikelstammGewicht
//  02.10.2013  AH  "InsertFahrt" reicht Auftragsnummer weiter durch (Projekt 1332/100)
//  07.10.2013  AH  Schopferzeugung bugfix
```

// 28.10.2013 AH Schopferzeugung bugfix (Warengruppe)

// 28.10.2013 AH Fix: Löschen von Einsatz, der einen Schopf hat, löscht auch die Ausbringung Schopf +

// 14.11.2013 AH Fix: "UpdateXzuY": bei Schopf AnfagnsID als Output-UrsprungsID übernehmen

// 06.03.2014 AH "UpdateSchopf" kann "gezwungen" werden Schopf anzulegen (bei BIS) und löscht kein

// 16.05.2014 AH "UpdateVSB" unterscheidet FAHR-VSBs

// 04.06.2014 AH "UpdateSchopf" bildet auch aus theoretischem Mat.

// 16.12.2014 AH "Insert" setzte ggf. Menge

// 09.02.2015 AH "FertNachPos"

// 10.03.2015 AH "Rename703Text"

// 10.04.2015 AH Auftrags-SL in Kommission aktiviert

// 24.04.2015 AH "BindeAnVorlage", "BelegeKommissionsDaten"

// 28.04.2015 AH "UpdateXzuY" versucht erkennt Zustand "ReinIstRaus" (Änderung der Planoutputmeng

// 28.07.2015 AH Arbeitsgang Schaelen

// 18.08.2015 AH Arbeitsgang QTeil überarbeitet

// 20.10.2015 AH "FertNachPos" Fix für Bufferversprung 703

// 04.01.2016 AH "AusKommission" warnt bei Rahmenvertrag

// 15.02.2016 AH Prj.1556/82, Walzen behält Stückzahl im Rest

// 14.03.2016 AH "AusKommission" refeshed Maske + landaten bei manueller Kommissionseingabe

// 17.03.2016 AH Neu: Feld "BAG.P.Status"

// 01.06.2016 ST Fix: "InsertFahrt" nimmt Kommission von korrekt auf Mat (War vorher Erx>200)

// 20.10.2016 AH VSB nach LFA-Umlager prüft, ob Kommission schon im Einsatz stand und meldet dann

// 03.01.2017 AH Outputs bekommen als Lagerorte immer den Produzenten

// 11.07.2017 AH AFX "BA1.F.BelegeKomDaten"

// 04.09.2017 AH Fix: Toleranzen werden auch immer übernommen, wenn Abmessung übernommen wir

// 17.01.2018 ST Arbeitsgang "Umlagern" hinzugefügt

// 12.03.2018 AH "AusKundenArtNr" mit Option "aNurBefuellte"

// 20.04.2018 AH 1zu1 nutzt auch für Fertigung "ErrechnePlanmengen"

// 18.05.2018 ST Fix: Errechne Planmengen verspringt Input nicht mehr

// 02.10.2018 AH Fix: Fahr-Fertigung belegt KdNr und Reserviert vor

// 11.12.2018 AH AFX "BAG.F.Replace.Pre"

// 14.01.2019 AH Fix: Planstückzahl bei Walzen beachtet Teilungen

// 25.02.2019 AH Fix: Rekursives Durchrechnen der Kinder bei "UpdateOutputKind", macht auch Autoteil

// 06.03.2019 AH "UpdateOutputKind" kann BA-Pos löschen mit Parameter

// 10.04.2019 AH AFX "BAG.Set.AufABemerkung"

// 26.04.2019 AH "UpdateVSB" löscht Aktion, wenn Vorgänderpos. erledigt ist

// 23.05.2019 AH "Update1zuY" rechnet/summiert Fertigungsgewicht immer selber

// 24.05.2019 AH Beim Einsatz von WE gegen VSB sollen keine Kgmm-Fehler kommen

// 13.06.2019 AH "Update..." setzt Lageranschrift laut Ext.Prod. der Vorgänger-Position

// 16.07.2019 AH "BessererLFA", der nur Restmengen anzeigt und ba BA-Ketten sich nicht addiert

// 30.09.2019 AH "BelegeKommissionsDaten" kann M_enge

// 14.10.2019 AH BAG.IO.OhneRestYN erzeugt keinen Schopf (Proj. 1808/109)

// 13.12.2019 AH Neu: CopyAufToVpg, CopyAdrToVpg, "BelegeKommissionsDaten" kann V_erpackung

// 17.01.2020 AH Fix: Update... gibt "aIgnoreKind" weiter

// 27.07.2021 AH ERX

// 11.10.2021 AH AG "Bereit"

// 10.11.2021 AH AFX "BAG.F.UpdateOutput.Post"

// 16.11.2021 ST Fix: "BerechnePlanmengen" auch bei Arbeitsgang c_BAG_Pack

// 24.05.2022 AH Edit: Tafeln übeträgt immer die Abmessung ins Output (auch bei NULL)

// 08.06.2022 AH Neu: WalzSpulen

// 2022-07-05 AH DEADLOCK

// 2022-09-01 AH AFX "BAG.F.Data.CopyAufToVpg"

// 2022-09-23 AH Spulbreite eingebaut

```
//  2022-12-20  AH  neue BA-MEH-Logik
//  2023-05-23  AH  "ErrechnePlanmengen" Fix für Sägen
//  2023-08-04  AH  QTeil überarbeitet
//
//  Subprozeduren
//    SUB Rename703Text(aNr : int; aPAlt : int; aFAlt : int; aPNeu : int; aFNeu : int; opt aCopy : logic););
//    SUB Erzeuge703ausAuf(aFert : int) : logic;
//    SUB CheckFertigung2Einsatz() : logic;
//    SUB BildeRADString() : alpha;
//    SUB Delete() : logic;
//    SUB UpdateSchopf(aDel : logic) : logic;
//    SUB UpdateVSB(aDel : logic) : logic;
//    SUB Update1zu1(aDatei : int; aDel : logic) : logic;
//    SUB Update1zuY(aDel : logic) : logic;
//    SUB UpdateXzuY(aDel : logic) : logic;
//    sub UpdateArtPrd(aDel : logic) : logic;
//    SUB UpdateOutput(aDatei : int; opt aDel : logic; opt aDel702 : logic) : logic;
//    SUB UpdateOutputKind(aDel : logic; opt aDel702 : logic; opt aIgnoreKgMM : logic) : logic;
//    SUB UpdateRestCoil() : logic;
//    SUB SetRidRad(aTyp : alpha);
//    SUB CopyAdrToVpg(aBaNr : int) : int
//    SUB CopyAufToVpg(aBaNr : int) : int
//    SUB SumInput();
//    SUB AusKommission(aAufNr  : int; vAufPos : int)
//    SUB AusKundenArtNr(opt aNurBefuellte : logic);
//    SUB ErrechnePlanmengen(aStk : logic; aGew : logic; aMenge : logic) : logic
```

```
//    SUB Splitten(aAnz : int; aStk : int; aGew : float; aMenge  : float; aAufNr : int; aAufPos : int;) : logic;
//    SUB Versand() : logic;
//    SUB Insert(aLock : int; aGrund : alpha) : int;
//    SUB Replace(aLock : int; aGrund : alpha) : int;
//    SUB insertFahrt() : int;
//    SUB PassendesMatAusAuftrag()  : int
//    SUB _PassendesMatAusAuftrag_Data(var aSortTreeHandle : int;)
//    SUB _PassendesMatAusAuftrag_Data_Pos(aSortItem : int; var aRecord : alpha[])
//    SUB _PassendesMatAusAuftrag_Data_Sort(aRowIndex : int) : alpha//
//    SUB FertNachPos(aBAG1 : int; aPos1 : int; aFert : int; aBAG2 : int; aPos2 : int;) : logic;
//    SUB BindeAnVorlage(aBAG : int; aPos : int; aFert : int; aVorlageBAG : int; aAufNr : int; aAufPos : int) :
//    SUB BelegeKommisisonsDaten(aString : alpha; aAufNr : int; aAufPos : int);
//
//============================================================================
@I:Def_Global
@I:Def_BAG
@I:Def_Aktionen

@define defBessererLFA

declare Insert(aLock : int; aGrund : alpha) : int;
declare Replace(aLock : int; aGrund : alpha) : int;
declare UpdateOutputKind(aDel : logic; opt aDel702 : logic; opt aIgnoreKgmm : logic) : logic;
declare UpdateOutput(aDatei : int; opt aDel : logic;opt aDel702 : logic; opt aIgnoreKgMM : logic; opt aLfaNi
declare ErrechnePlanmengen(aStk : logic; aGew : logic; aMenge : logic; opt aNur1Input  : logic) : logic
```

```
//=========================================================================
//  Rename703Text
//
//=========================================================================
sub Rename703Text(
  aNr       : int;
  aPAlt     : int;
  aFAlt     : int;
  aPNeu     : int;
  aFNeu     : int;
  opt aCopy : logic);
local begin
  vA, vB  : alpha;
end;
begin
  vA # '~703.'+CnvAI(aNr,_FmtNumLeadZero | _FmtNumNoGroup,0,8)+CnvAI(aPAlt,_FmtNumLeadZero |
  vB # '~703.'+CnvAI(aNr,_FmtNumLeadZero | _FmtNumNoGroup,0,8)+CnvAI(aPNeu,_FmtNumLeadZero
  TextDelete(vB, 0);
  if (aCopy) then
    TextCopy(vA, vB, 0)
  else
    TextRename(vA, vB, 0);
end;


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

```
//  Erzeuge703ausAuf
//
//=========================================================================
SUB Erzeuge703ausAuf(aFert : int) : logic;
local begin
  Erx   : int;
end
begin

  if (aFert=0) then begin
    Erx # RecLink(703,702,4,_recLast);  // letzte Fertigung holen
    if (Erx>_rLocked) then aFert # 1
    else aFert # BAG.F.Fertigung + 1;
  end;

  RecBufClear(703);
  BAG.F.Nummer              # BAG.P.Nummer;
  BAG.F.Position            # BAG.P.Position;
  BAG.F.Fertigung           # aFert;
  BAG.F.Warengruppe         # Auf.P.Warengruppe;
  BAG.F.Auftragsnummer      # Auf.P.Nummer;
  BAG.F.Auftragspos         # Auf.P.Position;
  BAG.F.Kommission          # aInt(BAG.F.Auftragsnummer)  + '/' +aint(BAG.F.Auftragspos);  // Kommission
  "BAG.F.ReservFürKunde"    # Auf.P.KundenNr;
  "BAG.F.KostenträgerYN"    # y;
  BAG.F.Dicke               # Auf.P.Dicke;
```

```
  BAG.F.Breite              # Auf.P.Breite;
  "BAG.F.Länge"             # "Auf.P.Länge";
  "BAG.F.Güte"              # "Auf.P.Güte";
  "BAG.F.Gütenstufe"        # "Auf.P.Gütenstufe";
  BAG.F.Streifenanzahl      # 1;
  BAG.F.MEH                 # Auf.P.MEH.Einsatz;  // 2022-12-19 AH BA1_P_Data:ErmittleMEH();
  BAG.F.Gewicht             # "Auf.P.Gewicht";
  "BAG.F.Stückzahl"         # "Auf.P.Stückzahl";
  BAG.F.Menge               # "Auf.P.Gewicht";

  // Anker...
  RunAFX('BAG.F.AusKommission','');

  Erx # Insert(0,'AUTO');
  if (Erx<>_rOK) then begin
    RETURN false;
  end;

  RETURN true;
end;


//=========================================================================
// CheckFertigung2Einsatz
//             Güte, Stufe, Dicke Fertigung gegen Einsatz prüfen und rot färben
//=========================================================================
```

```
sub CheckFertigung2Einsatz() : logic;
local begin
  vBuf701 : int;
end;
begin

  if($Lb.Guete.E->wpcaption <> "BAG.F.Güte") and ("BAG.F.Güte" <> '')  then
    $Lb.Guete.E -> wpColBkg # _winColLightRed;
  else
    $Lb.Guete.E -> wpColBkg # _WinColParent;

  if($Lb.GuetenStufe.E->wpcaption  <> "BAG.F.Gütenstufe") and ("BAG.F.Gütenstufe" <> '') then
    $Lb.GuetenStufe.E -> wpColBkg # _winColLightRed;
  else
    $Lb.GuetenStufe.E -> wpColBkg # _WinColParent;

  case BAG.P.Aktion of

    c_BAG_AbLaeng : begin
    end;


    c_BAG_Saegen : begin
    end;
```

```
    c_BAG_Walz: begin
    end;



    otherwise begin
      if ($lb.Dicke.E<>0) then begin
        if($Lb.Dicke.E->wpcaption <> ANum(BAG.F.Dicke, Set.Stellen.Dicke)) and (BAG.F.Dicke <> 0.0) ther
          $Lb.Dicke.E -> wpColBkg # _winColLightred;
        else
          $Lb.Dicke.E -> wpColBkg # _WinColParent;
      end;
    end;
  end; // case

  RETURN true;
end;



//=========================================================================
// BildeRADString
//             generiert einen String mit dem RAD von allen Einsaetzen
//=========================================================================
sub BildeRADString() : alpha;
local begin
  Erx             : int;
  vBuf701         : int;
```

```
  vBuf702           : int;
  vMaxFertigungsRID : float;
  vRAD.E            : alpha(120);
  vRAD              : float;
end;
begin

  vBuf701 # RekSave(701);
  vBuf702 # RekSave(702);

  vRAD.E # '';
  Erx # RecLink(702, 703, 2, _recFirst);
  if(Erx > _rLocked) then
    RecBufClear(702);

  Erx # RecLink(701, 702, 2, _recFirst);
  WHILE (Erx <= _rLocked) DO BEGIN
    if (BAG.IO.BruderID=0) then begin
      vMaxFertigungsRID # BA1_IO_Data:MaxFertigungsRID();
      if (vMaxFertigungsRID <> 0.0) then begin
        vRAD # Lib_berechnungen:RAD_aus_KgStkBDichteRIDTlg(BAG.IO.Plan.Out.GewN, BAG.IO.Plan.O
        if(vRAD <> 0.0) then begin
          if(vRAD.E <> '') then
            vRAD.E # StrCut(vRAD.E + '; ',1,120);

          vRAD.E # StrCut(vRAD.E + cnvAF(vRAD),1,120);
```

```
          if (strlen(vRAD.E) >80) then
            BREAK;


        end;
      end;
    end;
    Erx # RecLink(701, 702, 2, _recNext);
  END;
  RekRestore(vBuf701);
  RekRestore(vBuf702);
  RETURN vRAD.E;
end;



//=========================================================================
//  Delete
//
//=========================================================================
sub Delete(
  opt aOhne999 : logic;
  ) : logic;
local begin
  Erx : int;
end;
begin
```

```
  TRANSON;

  // Text löschen...
  Erx # TxtDelete('~703.'+CnvAI(BAG.F.Nummer,_FmtNumLeadZero | _FmtNumNoGroup,0,8)+CnvAI(BAG
  if (Erx=_rDeadLock) then begin
    TRANSBRK;
    RETURN false;
  end;

  // Ausführungen löschen...
  WHILE (RecLink(705,703,8,_recFirst)<=_rLocked) do begin
    Erx # RekDelete(705,0,'MAN');
    if (Erx=_rDeadLock) then begin
      TRANSBRK;
      RETURN false;
    end;
  END;

  if (RekDelete(703,0,'AUTO')<>_rOK) then begin
    TRANSBRK;
    RETURN false;
  end;

@ifdef LogFlow
debug('Delete ID:'+aint(BAG.IO.ID));
@endif
```

```
  // Fertigmaterial löschen
  if (UpdateOutput(703,true)=false) then begin
    TRANSBRK;
    RETURN false;
  end;

  TRANSOFF;

  RETURN true;
end;


//=========================================================================
//  UpdateSchopf
//
//=========================================================================
sub UpdateSchopf(
  aDel        : logic;
  opt aMuss   : logic;
  opt aDel702 : logic) : logic;
local begin
  Erx     : int;
  vBuf701 : int;
  vBuf702 : int;
  vBuf703 : int;
```

```
  vVorID  : int;
  vOK     : logic;
  vGew    : float;
  vStk    : int;
  vNew    : logic;
  vL      : float;
  vA      : alpha;
  vI      : int;
  v701MAT : int;
  vVonID  : int;
end;
begin

  if (BA1_P_Data:DarfSchopfHaben()=false) then
    RETURN true;

  aDel    # n;
  vBuf701 # RekSave(701);
  vBuf703 # RekSave(703);

@ifdef LogFlow
debug('Update Schopf....');
@endif

  v701Mat # RecBufCreate(701);
```

```
  begin
    // gesamten Einsatz addieren
    FOR Erx # RecLink(701,702,2,_RecFirst)
    LOOP Erx # RecLink(701,702,2,_RecNext)
    WHILE (Erx<=_rLocked) do begin
      if (BAG.IO.OhneRestYN) then CYCLE;

      if (BA1_IO_I_data:IstMatBeistellung()) then aMuss # true;

      RecLink(819,701,7,_recFirst);          // Warengruppe holen
      if (BAG.IO.BruderID=0) and
        ((BAG.IO.Materialtyp=c_IO_Theo) or    // 04.06.2014 AH
         (BAG.IO.Materialtyp=c_IO_Mat) or (BAG.IO.Materialtyp=c_IO_BAG)) and
        ("BAG.IO.LöschenYN"=n) then begin
        RecBufCopy(701,v701Mat);
  //todo(aint(bag.io.id)+'  '+anum(BAG.IO.Plan.In.GewN,0)+'  '+anum(BAG.IO.Plan.Out.GewN,0));
        vStk  # vStk + (BAG.IO.Plan.In.Stk-BAG.IO.Plan.Out.Stk);
        vGew  # vGew + (BAG.IO.Plan.In.GewN-BAG.IO.Plan.Out.GewN);  // war GewB-GewN
        vL # Lib_Berechnungen:L_aus_KgStkDBDichte2(BAG.IO.Plan.In.GewN-BAG.IO.Plan.Out.GewN, 1, E
//debug('KEY701 gew:'+anum(vGew,0)+'  L:'+anum(vL,0));
      end;
    END;
  end;

  RecBufCopy(vBuf701, 701);
//todox(aint(vStk)+'Stk '+anum(vGew,0)+'kg '+aint(bag.io.id));
```

```
  // 2.7.2013 CHECK erzeugt KEINEN Schopf
  if (BAG.P.Aktion=c_BAG_Check) then aDel # y;



  RecBufClear(703);
  BAG.F.Nummer          # BAG.P.Nummer;
  BAG.F.Position        # BAG.P.Position;
  BAG.F.Fertigung       # 999;
  BAG.F.Block           # '';
  BAG.F.AutomatischYN   # y;
//  BAG.F.MEH             # BA1_P_Data:ErmittleMEH();
  BAG.F.MEH # 'kg';     // 10.11.2016 Reste sind immer KG
//BAG.F.MEH # BAG.IO.MEH.In;
//debug(bag.f.meh);
  // 06.03.2014 AH
  // existieren bereits Verwiegungen??? -> Dann ENDE
  if (aMuss=false) and ((vGew<=1.0) or (aDel)) then begin
    aMuss # RecLinkInfo(707,703,10,_recCount)>0;
  end;

  // soll Schopf gebildet werden?
  if (aMuss) or ((vGew>=1.0) and (aDel=n)) then begin
    Erx # RecRead(703,1,_rectest)
    if (Erx<=_rLocked) then begin
```

```
      vNew # n;
      RecRead(703,1,0);
    end
    else begin
      vNew # y;
    end;

    BAG.F.Warengruppe     # v701Mat->BAG.IO.Warengruppe;

//  26.09.2013  AH ff:
//    if (BAG.P.Aktion<>c_BAG_Abcoil) and
//      (("BAG.P.Typ.1In-1OutYN") or ("BAG.P.Typ.xIn-yOutYN")) then begin
    "BAG.F.Stückzahl"     # vStk;//BAG.IO.Plan.In.Stk - BAG.IO.Plan.Out.Stk;
//    end;


    // Prj. 1556/82 AH 15.02.2016
    if (BAG.P.Aktion=c_BAG_Walz) then begin
      "BAG.F.Stückzahl"     # BAG.IO.Ist.In.Stk;
    end;

    // Prj. 1161/386
    if ("BAG.P.Typ.1in-yOutYN") then begin
  //    if (BAG.IO.OhneRestYN) and
        if (BAG.IO.Plan.In.Stk<>BAG.IO.Plan.Out.Stk) then begin
        "BAG.F.Stückzahl"     # BAG.IO.Ist.In.Stk - BAG.IO.Plan.Out.Stk;
```

```
      end;
    end;


//    if ("BAG.F.Stückzahl"<=0) and (aMuss=false) then
//      "BAG.F.Stückzahl"     # 1;

    BAG.F.Gewicht         # vGew;
    if (BAG.F.MEH='kg') then BAG.F.Menge # Rnd(BAG.F.Gewicht, Set.Stellen.Menge);
    if (BAG.F.MEH='qm') then begin
      vL # Lib_Berechnungen:L_aus_KgStkDBDichte2(BAG.F.Gewicht, 1, v701Mat->BAG.IO.Dicke, v701Ma
      BAG.F.Menge # Rnd((v701Mat->"BAG.IO.Breite" * vL / 1000000.0),Set.Stellen.Menge);
    end;
    if (BAG.F.MEH='m') then begin
      vL # Lib_Berechnungen:L_aus_KgStkDBDichte2(BAG.F.Gewicht, 1, v701Mat->BAG.IO.Dicke, v701Ma
      BAG.F.Menge # Rnd((vL / 1000.0), Set.Stellen.Menge);
    end;

//    BAG.F.Breite          # BAG.IO.Breite;
    BAG.F.Streifenanzahl  # 1;

    RecBufDestroy(v701Mat);

    if (vNew) then begin
      BAG.F.Anlage.Datum    # today;
      BAG.F.Anlage.Zeit     # now;
```

```
      BAG.F.Anlage.User     # gusername;
      Erx # Insert(0,'MAN');
    end
    else begin
      Erx # RecRead(703,1,_recLock | _recNoLoad);
      if (Erx=_rOK) then Erx # Replace(_recUnlock,'MAN');
    end;
    if (Erx<>_rOK) then begin
      RekRestore(vBuf701);
      RekRestore(vBuf703);
@ifdef LogFlow
debug('...update Schopf ERROR');
@endif
      RETURN false;
    end;



    // Ausbringung updaten
    if (UpdateOutput(703, n)=false) then begin
      RekRestore(vBuf701);
      RekRestore(vBuf703);
@ifdef LogFlow
debugx('...update Schopf ERROR');
@endif
      RETURN false;
    end;
```

```
  end
  else begin    // kein Schopf mehr !

    Erx # RecRead(703,1,_rectest);
    if (Erx<=_rLocked) then begin
      RecRead(703,1,0);

      Erx # RecLink(701,703,4,_recFirst);   // Output loopen
      WHILE (Erx<=_rLockeD) do begin
        if (BAG.IO.NachBAG<>0) then begin
@ifdef LogFlow
debugx('...update Schopf ERROR bei ID '+aint(bag.io.id));
@endif
          Error(701038,'');
          RETURN false;
        end;
        Erx # RecLink(701,703,4,_recNext);
      END;

      Erx # Rekdelete(703,0,'AUTO');
      if (Erx<>_rOK) then begin
        RekRestore(vBuf701);
        RekRestore(vBuf703);
@ifdef LogFlow
```

```
debugx('...update Schopf ERROR');
@endif
        RETURN false;
      end;

      // Ausbringung updaten
      if (UpdateOutput(703,y)=false) then begin
        RekRestore(vBuf701);
        RekRestore(vBuf703);
@ifdef LogFlow
debugx('...update Schopf ERROR');
@endif
        RETURN false;
      end;
    end;

  end;

  RekRestore(vBuf701);
  RekRestore(vBuf703);
@ifdef LogFlow
debugx('...update Schopf!');
@endif
  RETURN true;
end;
```

```
//=========================================================================
//  UpdateVSB
//           geplante Menge dem Auftrag melden
//=========================================================================
sub UpdateVSB(
  aDel          : logic;
  opt aPerTodo  : logic) : logic;
local begin
  Erx     : int;
  v701    : int;
  v702    : int;
  vVorID  : int;
  vNachID : int;
  vNeu    : logic;
  vOK     : logic;
  vFahrVK : logic;
  vBuf404 : int;
  vKLim   : float;
  vDat1   : date;
  vDat2   : date;
  vVorgaengerDone : logic;
  vFahrEchtMenge  : float;
  vFahrEchtNetto  : float;
  vFahrEchtBrutto : float;
  vFahrEchtStk    : int;
```

```
  vStk      : int;
  vGewN     : float;
  vGewB     : float;
  vM        : float;
  vGewGes   : float;
end;
begin

  if (BAG.P.Auftragsnr=0) or (BAG.P.Auftragspos=0) then RETURN true;
  if (BAG.IO.Materialtyp<>c_IO_BAG) then RETURN true;

  vVorID # BAG.IO.ID;

  v702 # RekSave(702);
  if (BAG.P.nummer<>BAG.IO.VonBAG) or (BAG.P.Position<>BAG.IO.VonPosition) then begin // 09.09.20
    BAG.P.Nummer    # BAG.IO.VonBAG;
    BAG.P.Position  # BAG.IO.VonPosition;
    Erx # RecRead(702,1,0);
    if (Erx>_rLocked) then RecBufClear(702);
  end;

  if ((BAG.P.Aktion=c_BAG_Fahr) or (BAG.P.AKtion=c_BAG_Umlager) or (BAG.P.Aktion=c_BAG_Versand
    if (BAG.P.ZielVerkaufYN) then begin
      vFahrVK # y;

      // 15.07.2019 AH: Tatsächliche Mengen abziehen
```

```
@ifdef defBessererLFA
    v701 # RekSave(701);
    if (BAG.IO.VonID<>0) then begin   // 09.09.2020 AH
      BAG.IO.ID # BAG.IO.VonID;
      Erx # Recread(701,1,0);
      if (Erx<=_rLocked) and (BAG.IO.Materialtyp=c_IO_Mat) then begin
        vFahrEchtNetto  # vFahrEchtNetto +  BAG.IO.Plan.In.GewN;
        vFahrEchtBrutto # vFahrEchtBrutto +  BAG.IO.Plan.In.GewB;
        vFahrEchtStk    # vFahrEchtStk + BAG.IO.Plan.In.Stk;
        vFahrEchtMenge  # vFahrEchtMenge + BAG.IO.Plan.In.Menge;
      end;
    end
    else begin
      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) then begin
          vFahrEchtNetto  # vFahrEchtNetto +  BAG.IO.Plan.In.GewN;
          vFahrEchtBrutto # vFahrEchtBrutto +  BAG.IO.Plan.In.GewB;
          vFahrEchtStk    # vFahrEchtStk + BAG.IO.Plan.In.Stk;
          vFahrEchtMenge  # vFahrEchtMenge + BAG.IO.Plan.In.Menge;
        end;
      END;
    end;
    RekRestore(v701);
@endif
```

```
  end
  else begin
    // 20.10.2016 AH : Umlagern MIT Kommission meldet gar nichts, wenn Kommission schon am Anfang
    // Ursprung prüfen..
    if (BAG.IO.UrsprungsID<>0) then begin
      v701 # RecBufCreate(701);
      v701->BAG.IO.Nummer # BAG.IO.Nummer;
      v701->BAG.IO.ID     # BAG.IO.UrsprungsID;
      Erx # RecRead(v701,1,0);
      if (Erx<=_rLocked) then begin
        if (v701->BAG.IO.Auftragsnr=v702->BAG.P.Auftragsnr) and (v701->BAG.IO.Auftragspos=v702->BA
          RecBufDestroy(v701);
          RekRestore(v702); // 03.02.2020
          RETURN true;
        end;
      end;
      RecBufDestroy(v701);
    end;
  end;
end;

if ("BAG.P.Löschmarker"<>'') then vVorgaengerDone # y;

vDat1 # BAG.P.Plan.StartDat;
vDat2 # BAG.P.Plan.EndDat;
RekRestore(v702);
```

```
  Auf.P.Nummer      # BAG.P.Auftragsnr;
  Auf.P.Position    # BAG.P.Auftragspos;
  Erx # RecRead(401,1,0);       // Auftrag holen
  if (Erx<=_rLockeD) then begin
    Erx # RecLink(400,401,3,_RecFirst);   // Kopf holen
    if (Erx<=_rLocked) and (Auf.Vorgangstyp<>c_Auf) then Erx # _rNoRec;
  end;
  if (Erx>_rLocked) then begin
    RETURN true;
  end;

  // Termine in VSB übernehmen
  Erx # RecRead(702,1,_recLock);
  if (Erx=_rOK) then begin
//    BAG.P.Plan.StartDat # vDat1;
//    BAG.P.Plan.EndDat   # vDat2;
    if (Auf.P.TerminZusage<>0.0.0) then
      BAG.P.Fenster.MaxDat  # Auf.P.TerminZusage
    else
      BAG.P.Fenster.MaxDat  # Auf.P.Termin1Wunsch;
    Erx # BA1_P_Data:Replace(_recUnlock,'AUTO');
  end;
  if (Erx<>_rOK) then RETURN false;   // 2022-07-05 AH DEADLOCK

  vStk  # Max(BAG.IO.Plan.In.Stk - BAG.IO.Ist.In.Stk - vFahrEchtStk, 0);
```

```
vGewN # Max(BAG.IO.Plan.In.GewN - BAG.IO.Ist.In.GewN - vFahrEchtNetto, 0.0);
vGewB # max(BAG.IO.Plan.In.GewB - BAG.IO.Ist.In.GewB - vFahrEchtBrutto, 0.0);
vM    # Max(BAG.IO.Plan.In.Menge - BAG.IO.Ist.In.Menge - vFahrEchtMenge, 0.0);
vGewGes # BAG.IO.Plan.In.GewN - BAG.IO.Ist.In.GewN;

Auf.A.Aktionsnr   # BAG.IO.VonBAG;
Auf.A.Aktionspos  # BAG.IO.VonPosition;
Auf.A.Aktionspos2 # BAG.IO.ID;
Auf.A.Aktionstyp  # c_Akt_BA_Plan;

if (vFahrVK) then begin // 01.12.2015 AH : "BA FA" suchen, sonst "BA S"
  Auf.A.Aktionstyp  # c_Akt_BA_Plan_Fahr;
  Erx # RecRead(404,2,0);
  if (Erx>_rMultikey) then begin
    Auf.A.Aktionsnr   # BAG.IO.VonBAG;
    Auf.A.Aktionspos  # BAG.IO.VonPosition;
    Auf.A.Aktionspos2 # BAG.IO.ID;
    Auf.A.Aktionstyp  # c_Akt_BA_Plan;
  end;
end;

Erx # RecRead(404,2,0);
if (Erx=_rLocked) then begin
  RETURN false;
end;
if (Erx<=_rMultikey) then begin    // erstmal auf jeden Fall löschen!!!
```

```
    vBuf404 # RekSave(404);
    if (Auf_A_Data:Entfernen()=false) then begin
      RecBufDestroy(vBuf404);
      RETURN false;
    end;
  end;

  if (aDel) then begin          // Löschen?
    if (vBuf404<>0) then RecBufDestroy(vBuf404);
    RETURN true;
  end;

  RecBufClear(404);             // Aktion neu anlegen
  Auf.A.Aktionsnr   # BAG.IO.VonBAG;
  Auf.A.Aktionspos  # BAG.IO.VonPosition;
  Auf.A.Aktionspos2 # BAG.IO.ID;
  if (vBuf404<>0) then
    Auf.A.Aktion      # vBuf404->Auf.A.Aktion;
  Auf.A.Aktionstyp  # c_Akt_BA_Plan;
  Auf.A.Nummer        # BAG.P.Auftragsnr;
  Auf.A.Position      # BAG.P.Auftragspos;
  Auf.A.Aktionsdatum  # BAG.P.Fenster.MinDat;
//  if (BAG.P.Fenster.MinZei>0:0) and (Auf.A.Aktionsdatum<>0.0.0) then Auf.A.Aktionsdatum->vmDayModi
  Auf.A.TerminStart   # BAG.P.Plan.StartDat
  Auf.A.TerminEnde    # BAG.P.Plan.EndDat;
  "Auf.A.Stückzahl"   # vStk;
```

```
  Auf.A.Gewicht       # vGewB;
  Auf.A.NettoGewicht  # vGewN;
  Auf.A.Bemerkung     # c_AktBem_BA_Plan;


// 16.05.2014
  if (vFahrVK) then begin
    Auf.A.Aktionstyp  # c_Akt_BA_Plan_Fahr;
    Auf.A.Bemerkung     # c_AktBem_BA_Plan_Fahr;
@ifdef defBessererLFA
    if (vGewGes>0.0) and (vGewGes<>vGewN) then Auf.A.Bemerkung # Auf.A.Bemerkung + ' ('+anum(vGe
@endif
  end;


  if (Auf.P.MEH.Einsatz=BAG.IO.MEH.In) then begin
    Auf.A.Menge       # vM;
    Auf.A.MEH         # Auf.P.MEH.Einsatz;
  end
  else if (StrCnv(Auf.P.MEH.Einsatz,_StrUppeR)='KG') then begin
    Auf.A.Menge       # vGewN;
    Auf.A.MEH         # Auf.P.MEH.Einsatz;
  end
  else if (StrCnv(Auf.P.MEH.Einsatz,_StrUpper)='T') then begin
    Auf.A.Menge       # Rnd(vGewN / 1000.0,Set.Stellen.Menge);
    Auf.A.MEH         # Auf.P.MEH.Einsatz;
  end
  else if (StrCnv(Auf.P.MEH.Einsatz,_StrUpper)='STK') then begin
```

```
    Auf.A.Menge       # CnvFI(vStk);
    Auf.A.MEH         # Auf.P.MEH.Einsatz;
  end;



// 01.02.2017
  if (StrCnv(Auf.P.MEH.Preis,_StrUppeR)='KG') or (StrCnv(Auf.P.MEH.Preis,_StrUpper)='T') then begin
    if (VwA.Nummer<>Auf.P.Verwiegungsart) then begin
      Erx # RecLink(818,401,9,_recFirst);   // Verwiegungsart holen
      if (Erx<>_rok) then begin
        RecBufClear(818);
        VWa.NettoYN # Y;
      end;
    end;
    if (VwA.NettoYN) then
      Auf.A.Menge.Preis # vGewN
    else
      Auf.A.Menge.Preis # vGewb;


    Auf.A.MEH.Preis   # Auf.P.MEH.Preis;
    if (StrCnv(Auf.P.MEH.Preis,_StrUpper)='T') then begin
      Auf.A.Menge.Preis # Rnd(Auf.A.Menge.Preis / 1000.0,Set.Stellen.Menge);
    end;
  end
  else if (Auf.P.MEH.Preis=BAG.IO.MEH.In) then begin
    Auf.A.Menge.Preis # vM;
```

```
  Auf.A.MEH.Preis   # Auf.P.MEH.Preis;
end
else if (StrCnv(Auf.P.MEH.Preis,_StrUpper)='STK') then begin
  Auf.A.Menge.Preis # CnvFI(vStk);
  Auf.A.MEH.Preis   # Auf.P.MEH.Preis;
end;

if (vVorgaengerDone) then ''Auf.A.Löschmarker'' # '*';

RunAFX('BAG.Set.Auf.Aktion','');

vOK # Auf_A_Data:NeuAnlegen(y,n,aPerTodo)=_rOK;   // 09.09.2020
if (vBuf404<>0) then begin
  if (vOK) then begin
    Erx # RecRead(404,1,_recLock);
    if (erx=_rOK) then begin
      Auf.A.Anlage.Datum # vBuf404->Auf.A.Anlage.Datum;
      Auf.A.Anlage.Zeit  # vBuf404->Auf.A.Anlage.Zeit;
      Auf.A.Anlage.User  # vBuf404->Auf.A.Anlage.User;
      Erx # RekReplace(404,_recUnlock,'AUTO');
    end;
    if (erx<>_rOK) then RETURN false;     // 2022-07-05 AH DEADLOCK
  end;
  RecBufDestroy(vBuf404);
end;
```

```
  // Kreditlimitprüfung...
//    if ("Set.KLP.BA-Fertigung"<>'') then begin
//      if (Adr_K_Data:Kreditlimit_BA(Auf.Rechnungsempf, "Set.KLP.BA-Fertigung", var vKLim)=false) then b
//        RETURN false;
//      end;
//    end;

  RETURN vOK;

end;


//=========================================================================
//  Update1zu1
//      also z.B. Walzen, Fahren, QS
//=========================================================================
sub Update1zu1(
  aDatei  : int;
  aDel    : logic;
  aDel702 : logic;
  opt aIgnoreKgMM : logic) : logic;
local begin
  Erx          : int;
  vBuf701      : int;
  vBuf702      : int;
```

```
  vVorID        : int;
  vNachID       : int;
  vNeu          : logic;
  vReinIstRaus  : logic;
  vOK           : logic;
  vL            : float;
  vStk          : int;
  vGew          : float;
  vM            : float;
  vID           : int;
end;
begin
  // Atikeleinsatz erzeugt keine Ausbringung...
  if (BAG.IO.Materialtyp=c_IO_Beistell) then RETURN true;

//todo('update pos:'+aint(bag.p.position)+'  datei:'+aint(aDatei)+'  id:'+aint(bag.io.id));

  vBuf701 # RekSave(701);
  vVorID # BAG.IO.ID;

  RecBufClear(701);
  Erx # RecLink(701,703,4,_recFirst);   // Output loopen
  WHILE (Erx=_rok) do begin
    if (BAG.IO.VonID=vVorID) then BREAK;
    Erx # RecLink(701,703,4,_recNext);
  END;
```

```
if (Erx=_rLocked) then begin
  RekRestore(vBuf701);
  RETURN false;
end;

if (Erx<>_rOK) or (BAG.IO.VonID<>vVorID) then begin  // Nicht gefunden?
  if (aDel) then begin              // Löschen? dann so ok
    RekRestore(vBuf701);
    RETURN true;
  end;

  Erx # RecLink(701,700,3,_recLast);
  if (Erx>_rLocked) then vID # 0
  else vID # BAG.IO.ID;

  vNeu # y;                         // Neuanlage
  RecBufClear(701);
  RecBufCopy(vBuf701,701);
  BAG.IO.Nummer         # BAG.F.Nummer;
  BAG.IO.ID             # vID;
  BAG.IO.VonBAG         # BAG.F.Nummer;
  BAG.IO.VonPosition    # BAG.F.Position;
  BAG.IO.VonFertigung   # BAG.F.Fertigung;
  BAG.IO.VonID          # vVorID;
  BAG.IO.NachBAG        # 0;
  BAG.IO.NachPosition   # 0;
```

```
  BAG.IO.NachFertigung  # 0;
  BAG.IO.Materialtyp    # c_IO_BAG;
  BAG.IO.Plan.In.Stk    # 0;
  BAG.IO.Plan.In.GewN   # 0.0;
  BAG.IO.Plan.In.GewB   # 0.0;
  BAG.IO.Plan.In.Menge  # 0.0;
  BAG.IO.Materialnr     # 0;
  BAG.IO.MaterialRstNr  # 0;
  BAG.IO.Ist.In.Stk     # 0;
  BAG.IO.Ist.In.GewN    # 0.0;
  BAG.IO.Ist.In.GewB    # 0.0;
  BAG.IO.Ist.In.Menge   # 0.0;

end
else begin

  if (BAG.IO.Plan.In.Stk=BAG.IO.Plan.Out.Stk) and
    (BAG.IO.Plan.In.GewN=BAG.IO.Plan.Out.GewN) and
    (BAG.IO.Plan.In.GewB=BAG.IO.Plan.Out.GewB) and
    (BAG.IO.Plan.In.Menge=BAG.IO.Plan.Out.Meng) then vReinIstRaus # y;

  if (aDel) then begin                // existierenden löschen?

    // Fertigung wird weiterbearbeitet???
    if (BAG.IO.NachBAG<>0) then begin
      vOk # UpdateOutputKind(aDel, aDel702, aIgnorekgmm);   // 17.01.2020 AH "aIgnorekgMm" angehäng
```

```
      if (vOK=false) then begin
        RekRestore(vBuf701);
        RETURN false;
      end;
    end;

    Erx # BA1_IO_Data:Delete(0,'AUTO');
    if (Erx<>_rOK) then begin
      RekRestore(vBuf701);
      RETURN false;
    end;

    RekRestore(vBuf701);
    Erx # RecRead(701,1,_recLock);
    if (erx=_rOK) then begin
      BAG.IO.NachID     # 0;
      //BAG.IO.Materialnr # 0;
      Erx # BA1_IO_Data:Replace(_recUnlock,'AUTO');
    end;
    RETURN Erx=_rOK;
  end;

  RecRead(701,1,0);
  vNeu # n;
end;
```

```
// Daten von Einsatz und Fertigung zusammenführen
BAG.IO.UrsprungsID    # vBuf701->BAG.IO.UrsprungsID;
"BAG.IO.Güte"         # vBuf701->"BAG.IO.Güte";
"BAG.IO.Gütenstufe"   # vBuf701->"BAG.IO.Gütenstufe";
BAG.IO.Dicke          # vBuf701->BAG.IO.Dicke;
BAG.IO.Breite         # vBuf701->BAG.IO.Breite;
BAG.IO.Spulbreite     # vBuf701->BAG.IO.Spulbreite;
"BAG.IO.Länge"        # vBuf701->"BAG.IO.Länge";
BAG.IO.DickenTol      # vBuf701->BAG.IO.DickenTol;
BAG.IO.BreitenTol     # vBuf701->BAG.IO.BreitenTol;
"BAG.IO.LängenTol"    # vBuf701->"BAG.IO.LängenTol";
BAG.IO.Lageradresse   # vBuf701->BAG.IO.Lageradresse;
BAG.IO.Lageranschr    # vBuf701->BAG.IO.Lageranschr;

if ("BAG.F.Güte"<>'') then          "BAG.IO.Güte"       # "BAG.F.Güte";
if ("BAG.F.Gütenstufe"<>'') then    "BAG.IO.Gütenstufe" # "BAG.F.Gütenstufe";

if (BAG.F.Dicke<>0.0) then begin
  BAG.IO.Dicke        # BAG.F.Dicke;
  BAG.IO.Dickentol    # BAG.F.Dickentol;
end;
if (BAG.F.Breite<>0.0) then begin
  BAG.IO.Breite       # BAG.F.Breite;
  BAG.IO.BreitenTol   # BAG.F.BreitenTol;
end;
```

```
if ("BAG.F.Länge"<>0.0) then begin
  "BAG.IO.Länge"      # "BAG.F.Länge";
  "BAG.IO.Längentol"  # "BAG.F.LängenTol";
end;
if (BAG.F.Spulbreite<>0.0) then begin
  BAG.IO.Spulbreite   # BAG.F.Spulbreite;
end;

if (BAG.F.Warengruppe<>0) then      BAG.IO.Warengruppe  # BAG.F.Warengruppe;
if (BAG.F.Artikelnummer<>'') then   BAG.IO.Artikelnr    # BAG.F.Artikelnummer;
if (BAG.F.AusfOben<>'') or (BAG.F.AusfUnten<>'') then begin
  BAG.IO.AusfOben     # BAG.F.AusfOben;
  BAG.IO.AusfUnten    # BAG.F.AusfUnten;
end;

// 26.10.2012 AI: Kommission erstmal löschen
BAG.IO.Auftragsnr     # 0;
BAG.IO.Auftragspos    # 0;
BAG.IO.AuftragsFert   # 0;
if (BAG.F.Auftragsnummer<>0) then begin
  BAG.IO.Auftragsnr     # BAG.F.Auftragsnummer;
  BAG.IO.Auftragspos    # BAG.F.Auftragspos;
  BAG.IO.AuftragsFert   # BAG.F.Auftragsfertig;
end
else if (BAG.P.Auftragsnr<>0) then begin
  if (BAG.F.WirdEigenYN=false) then begin
```

```
    BAG.IO.Auftragsnr     # BAG.P.Auftragsnr;
    BAG.IO.Auftragspos    # BAG.P.Auftragspos;
    BAG.IO.AuftragsFert   # 0;
  end;
end
else begin
  BAG.IO.Auftragsnr     # vBuf701->BAG.IO.Auftragsnr;
  BAG.IO.Auftragspos    # vBuf701->BAG.IO.Auftragspos;
  BAG.IO.AuftragsFert   # vBuf701->BAG.IO.Auftragsfert;
end;
BAG.F.Bemerkung # '';
if (BAG.IO.Auftragsnr<>0) then
  BAG.IO.Bemerkung  # cnvai(BAG.IO.Auftragsnr)+'/'+cnvai(BAG.IO.Auftragspos);

if ((BAG.P.Aktion=c_BAG_Fahr) or (BAG.P.Aktion=c_BAG_Umlager) or (BAG.P.Aktion=c_BAG_Versand
  if (BAG.P.Zieladresse<>0) then begin
    BAG.IO.Lageradresse # BAG.P.Zieladresse;
    BAG.IO.Lageranschr  # BAG.P.Zielanschrift;
  end;
end
else if (BAG.P.ExterneLiefNr<>0) then begin   // 03.01.2017 AH: Lagerorte liegen immer beim Produzente
  Erx # RecLink(100,702,7,_RecFirst);   // Produzent holen
  if (Erx<=_rLocked) then begin
    BAG.IO.Lageradresse # Adr.Nummer;
    BAG.IO.Lageranschr  # 1;
    if (BAG.P.ExterneLiefAns<>0) then
```

```
        BAG.IO.Lageranschr # BAG.P.ExterneLiefAns;
    end;
  end;

//debug('HIER:'+aint(bag.io.plan.out.stk)+'Stk   '+aint(bag.io.teilungen)+'TLG');
//debug('HIERB:'+aint(bag.f.fertigung)+' '+aint(vBuf701->bag.io.plan.out.stk)+'Stk   '+aint(vBuf701->bag.io.t
  BAG.IO.Plan.In.Stk    # vBuf701->BAG.IO.Plan.Out.Stk * (vBuf701->BAG.IO.Teilungen+1);

  if (BAG.F.Fertigung=999) then
    BAG.IO.Plan.In.Stk  # "BAG.F.Stückzahl";

  BAG.IO.Plan.In.GewN   # vBuf701->BAG.IO.Plan.Out.GewN;
  BAG.IO.Plan.In.GewB   # vBuf701->BAG.IO.Plan.Out.GewB;
  BAG.IO.Plan.In.Menge  # vBuf701->BAG.IO.Plan.Out.Meng;

// 2022-12-19 AH  BAG.IO.MEH.IN         # 'kg';
//  if (BAG.P.Aktion=c_BAG_Fahr) or (BAG.P.Aktion = c_BAG_Umlager) or (BAG.P.Aktion=c_BAG_Versan
//    (BAG.P.Aktion=c_BAG_Check) then // 04.04.2022 AH, LZM QS nach Tafeln
//    BAG.IO.MEH.In       # vBuf701->BAG.IO.MEH.Out;
//  BAG.IO.MEH.Out        # vBuf701->BAG.IO.MEH.Out;
  BAG.IO.MEH.In         # vBuf701->BAG.IO.MEH.Out;
  BAG.IO.MEH.Out        # BAG.IO.MEH.Out;

  // Restfertigung?
  if (BAG.F.Fertigung=999) then begin
  // Prj.1556/82, Walzen behält Stückzahl im Rest
```

```
    if (BAG.P.Aktion=c_BAG_Walz) or (BAG.P.Aktion=c_BAG_WalzSpulen) then
      vStk # vBuf701->BAG.IO.Plan.In.Stk;
    else
      vStk # vBuf701->BAG.IO.Plan.In.Stk - vBuf701->BAG.IO.Plan.Out.Stk;
    vGew # vBuf701->BAG.IO.Plan.In.GewN - vBuf701->BAG.IO.Plan.Out.GewN;

//    BAG.IO.Plan.In.GewN  # BAG.F.Gewicht;
//    BAG.IO.Plan.In.GewB  # BAG.F.Gewicht;
    BAG.IO.Plan.In.GewN   # vGew;
    BAG.IO.Plan.In.GewB   # vGew;
    BAG.IO.Plan.In.Stk    # vStk;
    if (BAG.IO.MEH.In='Stk') then
      BAG.IO.Plan.In.Menge # cnvfi(BAG.IO.Plan.In.Stk)
    else if (BAG.IO.MEH.In='kg') then
      BAG.IO.Plan.In.Menge  # BAG.IO.Plan.In.GewN
    else if (BAG.IO.MEH.In='t') then
      BAG.IO.Plan.In.Menge  # Rnd(BAG.IO.Plan.In.GewN/1000.0,Set.Stellen.Menge)
    else if (BAG.IO.MEH.In='m') then
//      BAG.IO.Plan.In.Menge  # Rnd("BAG.IO.Länge" * cnvfi("BAG.F.Stückzahl") / 1000.0, Set.Stellen.Meng
      BAG.IO.Plan.In.Menge  # Rnd("BAG.IO.Länge" * cnvfi(BAG.IO.Plan.In.Stk) / 1000.0, Set.Stellen.Menge
    else if (BAG.IO.MEH.In='qm') then begin
      vL # "BAG.IO.Länge";
      if (vL=0.0) then begin
        RecLink(819,701,7,_recFirst);   // Warengruppe holen
        vL # Lib_Berechnungen:L_aus_KgStkDBDichte2(BAG.IO.Plan.In.GewN, BAG.IO.Plan.In.Stk, BAG.IO
      end;
```

```
      BAG.IO.Plan.In.Menge # (BAG.IO.Breite * vL / 1000000.0) * Cnvfi(BAG.IO.Plan.In.Stk);
    end;
  end;  // Restfertigung
// usw. TODO



  if (BAG.IO.NachBAG<>0) then begin     // NÄCHSTEN Schritt prüfen
    vBuf702 # RekSave(702);
    BAG.P.Nummer    # BAG.IO.NachBAG;
    BAG.P.Position  # BAG.IO.NachPosition;
    Erx # RecRead(702,1,0);
    if (Erx=_rOK) then vReinIstRaus # vReinIstRaus or BAG.P.Typ.VSBYN;
    RekRestore(vBuf702);
  end;

  // Eingang=Ausgang bei Weiterbearbeitung
  if (vReinIstRaus) or (BAG.IO.NachBAG=0) then begin
//todo('rein ist raus bei ID '+aint(bag.io.id));
    BAG.IO.Plan.Out.Stk    # BAG.IO.Plan.In.Stk;
    BAG.IO.Plan.Out.GewN   # BAG.IO.Plan.In.GewN;
    BAG.IO.Plan.Out.GewB   # BAG.IO.Plan.In.GewB;
    BAG.IO.Plan.Out.Meng   # BAG.IO.Plan.In.Menge;
  end;

  if (vNeu=n) then begin
    Erx # RecRead(701,1,_RecLock | _RecNoLoad);
```

```
  if (erx=_rOK) then begin
    //BAG.IO.Materialnr # 0;
    Erx # BA1_IO_Data:Replace(_recUnlock,'AUTO');
  end;
  if (Erx<>_rOK) then begin   // 2022-07-05 AH DEADLOCK
    RekRestore(vBuf701);
    RETURN false;
  end;
  vNachID # BAG.IO.ID;
end
else begin                    // NEUANLAGE
  REPEAT
    BAG.IO.ID       # BAG.IO.ID + 1;
    Erx # BA1_IO_Data:Insert(0,'AUTO');
    if (Erx=_rdeadLock) then begin
      RekRestore(vBuf701);
      RETURN false;
    end;
  UNTIL (Erx=_rOK);
  vNachID # BAG.IO.ID;
end;

// Fertigung wird weiterbearbeitet??? Dann weiter in die TIEFE
if (BAG.IO.NachBAG<>0) then begin
  vOk # UpdateOutputKind(aDel, aDel702, aIgnorekgmm);   // 17.01.2020 AH "aIgnorekgMm" angehängt
  if (vOK=false) then begin
```

```
      RekRestore(vBuf701);
      RETURN false;
    end;
  end;

  RekRestore(vBuf701);

  // NachID setzen
  Erx # RecRead(701,1,_recLock);
  if (Erx=_rOK) then begin
    BAG.IO.NachID     # vNachID;
    //BAG.IO.Materialnr # 0;
    Erx # BA1_IO_Data:Replace(_recUnlock,'AUTO');
  end;

  RETURN (Erx=_rOK);

end;


//=========================================================================
//  Update1zuY
//      also z.B. Spalten; "pro Einsatz kommen Y raus"
//=========================================================================
sub Update1zuY(
  aDel        : logic;
```

```
  opt aDel702 : logic;
  opt aIgnoreKgMM : logic) : logic;
local begin
  Erx          : int;
  vBuf701      : int;
  vBuf702      : int;
  vNeu         : logic;
  vVorID       : int;
  vOK          : logic;
  vGew         : float;
  vReinIstRaus  : logic;
  vL           : float;
end;
begin

  // Atikeleinsatz erzeugt keine Ausbringung...
  if (BAG.IO.Materialtyp=c_IO_Beistell) then RETURN true;

  vBuf701 # RekSave(701);

  vVorID # BAG.IO.ID;

  vNeu # y;
  RecBufClear(701);        // genau DIESEN Output finden
  BAG.IO.VonBAG       # BAG.F.Nummer;
  BAG.IO.VonPosition  # BAG.F.Position;
```

```
BAG.IO.VonFertigung # BAG.F.Fertigung;
BAG.IO.VonID        # vVorID;
Erx # RecRead(701,3,0);
if (BAG.IO.VonBAG=BAG.P.Nummer) and
   (BAG.IO.VonPosition=BAG.P.Position) and
   (BAG.IO.VonFertigung=BAG.F.Fertigung) and
   (BAG.IO.VonID=vVorID) and
   ((Erx<=_rLocked) or (Erx=_rMultikey)) then vNeu # n;
if (aDel) then begin
  vOk # y;
  if (vNeu=n) then begin

    // Fertigung wird weiterbearbeitet???
    if (BAG.IO.NachBAG<>0) then begin
      vOk # UpdateOutputKind(aDel, aDel702, aIgnorekgmm);   // 17.01.2020 AH "aIgnorekgMm" angehäng
      if (vOK=false) then begin
        RekRestore(vBuf701);
        RETURN false;
      end;
    end;

    Erx # BA1_IO_Data:Delete(0,'AUTO');
    vOk # (Erx=_rOK);
  end;
  RekRestore(vBuf701);
  RETURN vok;
```

```
end;


if (vNeu) then begin                  // Neuanlage??
  RecBufClear(701);
  RecBufCopy(vBuf701,701);
  BAG.IO.Nummer         # BAG.F.Nummer;
  BAG.IO.ID             # 0;
  BAG.IO.VonBAG         # BAG.F.Nummer;
  BAG.IO.VonPosition    # BAG.F.Position;
  BAG.IO.VonFertigung   # BAG.F.Fertigung;
  BAG.IO.VonID          # vVorID;
  BAG.IO.NachBAG        # 0;
  BAG.IO.NachPosition   # 0;
  BAG.IO.NachFertigung  # 0;
  BAG.IO.Materialtyp    # c_IO_BAG;
  //BAG.IO.Materialnr     # 0;
  BAG.IO.Plan.In.Stk    # 0;
  BAG.IO.Plan.In.GewN   # 0.0;
  BAG.IO.Plan.In.GewB   # 0.0;
  BAG.IO.Plan.In.Menge  # 0.0;
  BAG.IO.Ist.In.Stk     # 0;
  BAG.IO.Ist.In.GewN    # 0.0;
  BAG.IO.Ist.In.GewB    # 0.0;
  BAG.IO.Ist.In.Menge   # 0.0;
end
else begin
```

```
  RecRead(701,1,0);
  if (BAG.IO.Plan.In.Stk=BAG.IO.Plan.Out.Stk) and
    (BAG.IO.Plan.In.GewN=BAG.IO.Plan.Out.GewN) and
    (BAG.IO.Plan.In.GewB=BAG.IO.Plan.Out.GewB) and
    (BAG.IO.Plan.In.Menge=BAG.IO.Plan.Out.Meng) then vReinIstRaus # y;
end;



// Daten von Einsatz und Fertigung zusammenführen
BAG.IO.UrsprungsID    # vBuf701->BAG.IO.UrsprungsID;
"BAG.IO.Güte"         # vBuf701->"BAG.IO.Güte";
"BAG.IO.Gütenstufe"   # vBuf701->"BAG.IO.Gütenstufe";
BAG.IO.Dicke          # vBuf701->BAG.IO.Dicke;
BAG.IO.Breite         # vBuf701->BAG.IO.Breite;
BAG.IO.Spulbreite     # vBuf701->BAG.IO.Spulbreite;
"BAG.IO.Länge"        # vBuf701->"BAG.IO.Länge";
BAG.IO.DickenTol      # vBuf701->BAG.IO.DickenTol;
BAG.IO.BreitenTol     # vBuf701->BAG.IO.BreitenTol;
"BAG.IO.LängenTol"    # vBuf701->"BAG.IO.LängenTol";
BAG.IO.Lageradresse   # vBuf701->BAG.IO.Lageradresse;
BAG.IO.Lageranschr    # vBuf701->BAG.IO.Lageranschr;

if (BAG.P.ExterneLiefNr<>0) then begin   // 03.01.2017 AH: Lagerorte liegen immer beim Produzenten
  Erx # RecLink(100,702,7,_RecFirst);   // Produzent holen
  if (Erx<=_rLocked) then begin
    BAG.IO.Lageradresse # Adr.Nummer;
```

```
    BAG.IO.Lageranschr  # 1;
    if (BAG.P.ExterneLiefAns<>0) then
      BAG.IO.Lageranschr # BAG.P.ExterneLiefAns;
  end;
end;

BAG.IO.Plan.In.Stk    # vBuf701->BAG.IO.Plan.Out.Stk;
BAG.IO.Plan.In.GewN   # vBuf701->BAG.IO.Plan.Out.GewN;
BAG.IO.Plan.In.GewB   # vBuf701->BAG.IO.Plan.Out.GewB

BAG.IO.Plan.In.Menge  # vBuf701->BAG.IO.Plan.Out.Meng;
BAG.IO.MEH.In         # vBuf701->BAG.IO.MEH.Out;
BAG.IO.MEH.Out        # 'kg';

if ("BAG.F.Güte"<>'') then          "BAG.IO.Güte"       # "BAG.F.Güte";
if ("BAG.F.Gütenstufe"<>'') then    "BAG.IO.Gütenstufe" # "BAG.F.Gütenstufe";
if (BAG.F.Dicke<>0.0) then begin
  BAG.IO.Dicke        # BAG.F.Dicke;
  BAG.IO.Dickentol    # BAG.F.Dickentol;
end;
if (BAG.F.Breite<>0.0) then begin
  BAG.IO.Breite       # BAG.F.Breite;
  BAG.IO.BreitenTol   # BAG.F.BreitenTol;
end;
if ("BAG.F.Länge"<>0.0) then begin
  "BAG.IO.Länge"      # "BAG.F.Länge";
```

```
  "BAG.IO.Längentol"  # "BAG.F.LängenTol";
end;
if (BAG.F.Spulbreite<>0.0) then begin
  BAG.IO.Spulbreite   # BAG.F.Spulbreite;
end;
if (BAG.F.Warengruppe<>0) then      BAG.IO.Warengruppe  # BAG.F.Warengruppe;
if (BAG.F.Artikelnummer<>'') then   BAG.IO.Artikelnr    # BAG.F.Artikelnummer;

// 26.10.2012 AI: Kommission erstmal löschen
BAG.IO.Auftragsnr     # 0;
BAG.IO.Auftragspos    # 0;
BAG.IO.AuftragsFert   # 0;
if (BAG.F.Auftragsnummer<>0) then begin
  BAG.IO.Auftragsnr     # BAG.F.Auftragsnummer;
  BAG.IO.Auftragspos    # BAG.F.Auftragspos;
  BAG.IO.AuftragsFert   # BAG.F.Auftragsfertig;
end
else if (BAG.P.Auftragsnr<>0) then begin
  if (BAG.F.WirdEigenYN=false) then begin
    BAG.IO.Auftragsnr     # BAG.P.Auftragsnr;
    BAG.IO.Auftragspos    # BAG.P.Auftragspos;
    BAG.IO.AuftragsFert   # 0;
  end;
end
else begin
  BAG.IO.Auftragsnr     # vBuf701->BAG.IO.Auftragsnr;
```

```
    BAG.IO.Auftragspos    # vBuf701->BAG.IO.Auftragspos;
    BAG.IO.AuftragsFert   # vBuf701->BAG.IO.Auftragsfert;
  end;
  BAG.F.Bemerkung # '';
  if (BAG.IO.Auftragsnr<>0) then
    BAG.IO.Bemerkung  # cnvai(BAG.IO.Auftragsnr)+'/'+cnvai(BAG.IO.Auftragspos);

  if (BAG.F.Fertigung<>999) then begin
    if (BAG.P.Aktion=c_BAG_QTEIL) then begin
//      Erx # RekLink(819,703,5,_recFirst);   // Warengruppe holen
// 2023-08-04 AH      vGew # Lib_Berechnungen:kg_aus_StkDBLDichte2(Max(BAG.F.Streifenanzahl,1), BA
      BAG.IO.Plan.In.GewN # vBuf701->BAG.IO.Plan.In.GewN;
      BAG.IO.Plan.In.GewB # vBuf701->BAG.IO.Plan.In.GewB;
      vGew # BAG.IO.Plan.In.GewN;
      BAG.IO.Plan.In.Stk   # "BAG.F.Streifenanzahl" * (vBuf701->BAG.IO.Plan.Out.Stk * (vBuf701->BAG.IO.
    end
    else if (BAG.P.Aktion=c_BAG_Schael) then begin
      BAG.IO.Plan.In.Stk  # (vBuf701->BAG.IO.Plan.Out.Stk * (vBuf701->BAG.IO.Teilungen+1));
      vL # Lib_Berechnungen:L_aus_KgStkDichteRIDRAD(vBuf701->BAG.IO.Plan.In.GewN, BAG.IO.Plan.In
//debugx('theo Len:'+anum(vL,2));
      BAG.IO.Plan.In.GewN # Lib_Berechnungen:KG_aus_StkDBLWgrArt(BAG.IO.Plan.In.Stk, BAG.F.Dicke
      vGew # BAG.IO.Plan.In.GewN;
//debugx('= '+anum(bag.io.plan.in.gewn,2));
      BAG.IO.Plan.In.GewB # BAG.IO.Plan.In.GewB;
    end
    else if (BAG.P.Aktion=c_BAG_Pack) then begin  // von BSP!?
```

```
    if (vBuf701->BAG.IO.Breite * BAG.IO.Breite * CnvFI(BAG.F.Streifenanzahl)<>0.0) then begin
      BAG.IO.Plan.In.GewN  # RND(BAG.IO.Plan.In.GewN / vBuf701->BAG.IO.Breite * BAG.IO.Breite * Cn
      vGew # BAG.IO.Plan.In.GewN;
      BAG.IO.Plan.In.GewB  # RND(BAG.IO.Plan.In.GewB   / vBuf701->BAG.IO.Breite * BAG.IO.Breite * C
    end;
    BAG.IO.Plan.In.Stk   # "BAG.F.Stückzahl" * (vBuf701->BAG.IO.Plan.Out.Stk);
  end
  else begin
    if (vBuf701->BAG.IO.Breite * BAG.IO.Breite * CnvFI(BAG.F.Streifenanzahl)<>0.0) then begin
      BAG.IO.Plan.In.GewN  # RND(BAG.IO.Plan.In.GewN / vBuf701->BAG.IO.Breite * BAG.IO.Breite * Cn
      vGew # BAG.IO.Plan.In.GewN;
      BAG.IO.Plan.In.GewB  # RND(BAG.IO.Plan.In.GewB   / vBuf701->BAG.IO.Breite * BAG.IO.Breite * C
    end;
//debug('streifen:'+aint(bag.f.streifenanzahl)+'   OutStk:'+aint(vBuf701->BAG.IO.Plan.Out.Stk)+'   TLG:'+ai
    BAG.IO.Plan.In.Stk   # "BAG.F.Streifenanzahl" * (vBuf701->BAG.IO.Plan.Out.Stk * (vBuf701->BAG.IO.
  end;
  if (BAG.IO.MEH.In='Stk') then
    BAG.IO.Plan.In.Menge # cnvfi(BAG.IO.Plan.In.Stk)
  else if (BAG.IO.MEH.In='kg') then
    BAG.IO.Plan.In.Menge  # vGew
  else if (BAG.IO.MEH.In='t') then
    BAG.IO.Plan.In.Menge  # Rnd(vGew/1000.0,Set.Stellen.Menge)
  else if (BAG.IO.MEH.In='qm') then
    BAG.IO.Plan.In.Menge # (BAG.IO.Breite * "BAG.IO.Länge" / 1000000.0) * Cnvfi(BAG.IO.Plan.In.Stk);
end
else begin  // Restcoil?
```

```
    BAG.IO.Plan.In.Stk   # "BAG.F.Stückzahl";
    BAG.IO.Plan.In.GewN  # vBuf701->BAG.IO.Plan.In.GewN - vBuf701->BAG.IO.Plan.Out.GewN;
    BAG.IO.Plan.In.GewB  # vBuf701->BAG.IO.Plan.In.GewB - vBuf701->BAG.IO.Plan.Out.GewB;
    BAG.IO.Plan.In.Menge # vBuf701->BAG.IO.Plan.In.Menge - vBuf701->BAG.IO.Plan.Out.Meng;
  end;

// usw. TODO

  if (BAG.IO.NachBAG<>0) then begin     // NÄCHSTEN Schritt prüfen
    vBuf702 # RekSave(702);
    BAG.P.Nummer    # BAG.IO.NachBAG;
    BAG.P.Position  # BAG.IO.NachPosition;
    Erx # RecRead(702,1,0);
    if (Erx=_rOK) then vReinIstRaus # vReinIstRaus or BAG.P.Typ.VSBYN;
    RekRestore(vBuf702);
  end;

  // Eingang=Ausgang bei Weiterbearbeitung
  if (vReinIstRaus) or (BAG.IO.NachBAG=0) then begin
    BAG.IO.Plan.Out.Stk    # BAG.IO.Plan.In.Stk;
    BAG.IO.Plan.Out.GewN   # BAG.IO.Plan.In.GewN;
    BAG.IO.Plan.Out.GewB   # BAG.IO.Plan.In.GewB;
    BAG.IO.Plan.Out.Meng   # BAG.IO.Plan.In.Menge;
  end;

//debug('up3:'+cnvai(bag.io.id)+'  ME:'+cnvaf(bag.io.plan.Out.meng)+'   KG:'+cnvaf(bag.io.plan.Out.gewB))
```

```
if (vNeu=n) then begin
  Erx # RecRead(701,1,_RecLock | _recNoLoad);
  if (erx=_rOK) then begin
    //BAG.IO.Materialnr # 0;
    Erx # BA1_IO_Data:Replace(_recUnlock,'AUTO');
  end;
end
else begin                      // NEUANLAGE
  REPEAT
    BAG.IO.ID         # BAG.IO.ID + 1;
    Erx # BA1_IO_Data:Insert(0,'AUTO');
    if (Erx=_rdeadLock) then BREAK;
  UNTIL (Erx=_rOK);
end;
if (Erx<>_rOK) then begin
  RekRestore(vBuf701);
  RETURN false;
end;

// Fertigung wird weiterbearbeitet??? Dann weiter in die TIEFE
if (BAG.IO.NachBAG<>0) then begin
  vOk # UpdateOutputKind(aDel, aDel702, aIgnorekgmm);   // 17.01.2020 AH "aIgnorekgMm" angehängt
  if (vOK=false) then begin
    RekRestore(vBuf701);
    RETURN false;
```

```
    end;
  end;

  RekRestore(vBuf701);
  RETURN true;
end;


//=========================================================================
//  UpdateXzuY
//    also z.B. FESTE Anzahl fertigen: Coils->Tafeln
//=========================================================================
sub UpdateXzuY(
  aDel        : logic;
  opt aDel702 : logic;
  opt aIgnoreKgMM : logic) : logic;
local begin
  Erx           : int;
  vBuf701       : int;
  vBuf702       : int;
  vNeu          : logic;
  vOK           : logic;
  vReinIstRaus  : logic;
  vL            : float;
  vVorID        : int;
end;
```

```
begin

  // Atikeleinsatz erzeugt keine Ausbringung...
// 26.07.2011 AI  if (BAG.IO.Materialtyp=c_IO_Beistell) then RETURN true;
// 07.10.2013 AH  Projekt 1161/353 + 1455/40
  if (BAG.IO.Materialtyp=c_IO_Beistell) then RETURN true;

@ifdef LogFlow
debug('update XzuY...');
@endif

  vBuf701 # RekSave(701);
  if (BAG.F.Fertigung=999) then begin
    vVorID  # BAG.IO.ID;
    aDel # "BAG.IO.LöschenYN" or (aDel);
  end;

  vNeu # y;
  RecBufClear(701);         // genau DIESEN Output finden
  BAG.IO.VonBAG       # BAG.F.Nummer;
  BAG.IO.VonPosition  # BAG.F.Position;
  BAG.IO.VonFertigung # BAG.F.Fertigung;
  BAG.IO.VonID        # vVorID;
  Erx # RecRead(701,3,0);
//todo('suche:'+aint(bag.f.nummer)+'/'+aint(bag.f.position)+'/'+aint(BAG.F.Fertigung)+'  ID:'+aint(vVorID));
  if (BAG.IO.VonBAG=BAG.F.Nummer) and
```

```
    (BAG.IO.VonPosition=BAG.F.Position) and
    (BAG.IO.VonID=vVorID) and
    (BAG.IO.VonFertigung=BAG.F.Fertigung) and
    ((Erx<=_rLocked) or (Erx=_rMultikey)) then vNeu # n;

  if (aDel) then begin
    vOk # y;

    if (vNeu=n) then begin
      // Fertigung wird weiterbearbeitet???
      if (BAG.IO.NachBAG<>0) then begin
        vOk # UpdateOutputKind(aDel, aDel702, aIgnorekgmm);   // 17.01.2020 AH "aIgnorekgMm" angehäng
        if (vOK=false) then begin
          RekRestore(vBuf701);
@ifdef LogFlow
debug('...update XzuY');
@endif
          RETURN false;
        end;
      end;
      Erx # BA1_IO_Data:Delete(0,'AUTO');
      vOk # (Erx=_rOK);
    end;
    RekRestore(vBuf701);
@ifdef LogFlow
debug('...update XzuY');
```

```
@endif
if (vOK=false) then todo('X');
    RETURN vok;
  end;


  if (vNeu) then begin                  // Neuanlage??
    RecBufClear(701);
    RecBufCopy(vBuf701,701);
    BAG.IO.Nummer         # BAG.F.Nummer;
    BAG.IO.ID             # 0;
    BAG.IO.VonBAG         # BAG.F.Nummer;
    BAG.IO.VonPosition    # BAG.F.Position;
    BAG.IO.VonFertigung   # BAG.F.Fertigung;
    BAG.IO.VonID          # vVorID;
    BAG.IO.NachBAG        # 0;
    BAG.IO.NachPosition   # 0;
    BAG.IO.NachFertigung  # 0;
    BAG.IO.Materialtyp    # c_IO_BAG;
    BAG.IO.Materialnr     # 0;
    BAG.IO.Ist.In.Stk     # 0;
    BAG.IO.Ist.In.GewN    # 0.0;
    BAG.IO.Ist.In.GewB    # 0.0;
    BAG.IO.Ist.In.Menge   # 0.0;
//2022-12-19  AH    BAG.IO.MEH.IN         # BAG.F.MEH; //vBuf701->BAG.IO.MEH.Out;
//    BAG.IO.MEH.Out        # BAG.F.MEH;
```

```
  BAG.IO.BruderID       # 0;
  BAG.IO.VonFertigmeld  # 0;
  BAG.IO.Ist.Out.Stk    # 0;
  BAG.IO.Ist.Out.GewN   # 0.0;
  BAG.IO.Ist.Out.GewB   # 0.0;
  BAG.IO.Ist.Out.Menge  # 0.0;
end
else begin
  RecRead(701,1,0);
  // 28.04.2015:
  if (BAG.IO.Plan.In.Stk=BAG.IO.Plan.Out.Stk) and
    (BAG.IO.Plan.In.GewN=BAG.IO.Plan.Out.GewN) and
    (BAG.IO.Plan.In.GewB=BAG.IO.Plan.Out.GewB) and
    (BAG.IO.Plan.In.Menge=BAG.IO.Plan.Out.Meng) then vReinIstRaus # y;
end;



// Daten von Einsatz und Fertigung zusammenführen
"BAG.IO.Güte"         # vBuf701->"BAG.IO.Güte";
"BAG.IO.Gütenstufe"   # vBuf701->"BAG.IO.Gütenstufe";
BAG.IO.Dicke          # vBuf701->BAG.IO.Dicke;
BAG.IO.Breite         # vBuf701->BAG.IO.Breite;
BAG.IO.Spulbreite     # vBuf701->BAG.IO.Spulbreite;
"BAG.IO.Länge"        # vBuf701->"BAG.IO.Länge";
BAG.IO.DickenTol      # vBuf701->BAG.IO.DickenTol;
BAG.IO.BreitenTol     # vBuf701->BAG.IO.BreitenTol;
```

```
  "BAG.IO.LängenTol"    # vBuf701->"BAG.IO.LängenTol";
  BAG.IO.Lageradresse   # vBuf701->BAG.IO.Lageradresse;
  BAG.IO.Lageranschr    # vBuf701->BAG.IO.Lageranschr;

  if (BAG.P.ExterneLiefNr<>0) then begin   // 03.01.2017 AH: Lagerorte liegen immer beim Produzenten
    Erx # RecLink(100,702,7,_RecFirst);   // Produzent holen
    if (Erx<=_rLocked) then begin
      BAG.IO.Lageradresse # Adr.Nummer;
      BAG.IO.Lageranschr  # 1;
      if (BAG.P.ExterneLiefAns<>0) then
        BAG.IO.Lageranschr # BAG.P.ExterneLiefAns;
    end;
  end;

// 2022-12-19 AH  BAG.IO.MEH.IN         # BAG.F.MEH;
//  BAG.IO.MEH.Out        # BAG.F.MEH;

  if ("BAG.F.Güte"<>'') then          "BAG.IO.Güte"       # "BAG.F.Güte";
  if ("BAG.F.Gütenstufe"<>'') then    "BAG.IO.Gütenstufe" # "BAG.F.Gütenstufe";
  if (BAG.F.Dicke<>0.0) then begin
    BAG.IO.Dicke        # BAG.F.Dicke;
    BAG.IO.Dickentol    # BAG.F.Dickentol;
  end;
  if (BAG.F.Breite<>0.0) or ((BAG.P.Aktion=c_BAG_Tafel) and (BAG.F.AutomatischYN=false)) then begin
    BAG.IO.Breite       # BAG.F.Breite;
    BAG.IO.BreitenTol   # BAG.F.BreitenTol;
```

```
end;
if ("BAG.F.Länge"<>0.0) or ((BAG.P.Aktion=c_BAG_Tafel) and (BAG.F.AutomatischYN=false)) then begi
  "BAG.IO.Länge"      # "BAG.F.Länge";
  "BAG.IO.Längentol"  # "BAG.F.LängenTol";
end;
if (BAG.F.Spulbreite<>0.0) then begin
  BAG.IO.Spulbreite   # BAG.F.Spulbreite;
end;

if (BAG.F.Warengruppe<>0) then      BAG.IO.Warengruppe  # BAG.F.Warengruppe;
if (BAG.F.Artikelnummer<>'') then   BAG.IO.Artikelnr    # BAG.F.Artikelnummer;

// 26.10.2012 AI: Kommission erstmal löschen
BAG.IO.Auftragsnr     # 0;
BAG.IO.Auftragspos    # 0;
BAG.IO.AuftragsFert   # 0;
if (BAG.F.Auftragsnummer<>0) then begin
  BAG.IO.Auftragsnr     # BAG.F.Auftragsnummer;
  BAG.IO.Auftragspos    # BAG.F.Auftragspos;
  BAG.IO.AuftragsFert   # BAG.F.Auftragsfertig;
end
else if (BAG.P.Auftragsnr<>0) then begin
  if (BAG.F.WirdEigenYN=false) then begin
    BAG.IO.Auftragsnr     # BAG.P.Auftragsnr;
    BAG.IO.Auftragspos    # BAG.P.Auftragspos;
    BAG.IO.AuftragsFert   # 0;
```

```
  end;
end
else begin
  BAG.IO.Auftragsnr     # vBuf701->BAG.IO.Auftragsnr;
  BAG.IO.Auftragspos    # vBuf701->BAG.IO.Auftragspos;
  BAG.IO.AuftragsFert   # vBuf701->BAG.IO.Auftragsfert;
end;

BAG.F.Bemerkung # '';
if (BAG.IO.Auftragsnr<>0) then
  BAG.IO.Bemerkung  # cnvai(BAG.IO.Auftragsnr)+'/'+cnvai(BAG.IO.Auftragspos);

if (BAG.P.Aktion=c_BAG_Tafel) and (BAG.F.AutomatischYN) then
  "BAG.IO.Länge" # 0.0; // Restcoil ohne Länge

if (BAG.P.Aktion=c_BAG_AbCoil) and (BAG.F.AutomatischYN) then
  "BAG.IO.Länge" # 0.0; // Restcoil ohne Länge

if (BAG.P.Aktion=c_BAG_Spulen) then
  "BAG.IO.Länge" # 0.0; // Spulen ohne Länge

if (BAG.F.Fertigung<999) then begin
  BAG.IO.Plan.In.Stk   # "BAG.F.Stückzahl";
  BAG.IO.Plan.In.GewN  # BAG.F.Gewicht;
  BAG.IO.Plan.In.GewB  # BAG.F.Gewicht;
end
```

```
  else begin
    BAG.IO.Plan.In.Stk   # vBuf701->BAG.IO.Plan.In.Stk - vBuf701->BAG.IO.Plan.Out.Stk;
    BAG.IO.Plan.In.GewN  # vBuf701->BAG.IO.Plan.In.GewN - vBuf701->BAg.IO.Plan.Out.GewN;
    BAG.IO.Plan.In.GewB  # vBuf701->BAG.IO.Plan.In.GewB - vBuf701->BAg.IO.Plan.Out.GewB;
    // 06.03.2014 AH:
    if (BAG.IO.Plan.In.GewN + BAG.IO.Plan.In.GewB <> 0.0) then
      if (BAG.IO.Plan.in.Stk=0) then BAG.IO.Plan.in.Stk # 1;

    if (vBuf701->BAG.IO.OhneRestYN) then begin
      BAG.IO.Plan.In.Stk   # 0;
      BAG.IO.Plan.In.GewN  # 0.0;
      BAG.IO.Plan.In.GewB  # 0.0;
    end;


//debug('**** ausID:'+aint(vBuf701->bag.io.id)+' : '+aint(vBuf701->bag.io.plan.in.stk)+'-'+aint(vBuf701->bag.
//if (vBuf701->"BAG.IO.LöschenYN") then debug('******** GELÖSCH!!!!!!!!!!');
  end;

  if (BAG.IO.MEH.In='Stk') then
    BAG.IO.Plan.In.Menge # cnvfi(BAG.IO.Plan.In.Stk)
  else if (BAG.IO.MEH.In='kg') then
    BAG.IO.Plan.In.Menge  # BAG.IO.Plan.In.GewN
  else if (BAG.IO.MEH.In='t') then
    BAG.IO.Plan.In.Menge  # Rnd(BAG.IO.Plan.In.GewN/1000.0,Set.Stellen.Menge)
  else if (BAG.IO.MEH.In='m') then
```

```
    BAG.IO.Plan.In.Menge  # Rnd("BAG.IO.Länge" * cnvfi("BAG.F.Stückzahl") / 1000.0, Set.Stellen.Menge)
  else if (BAG.IO.MEH.In='qm') then begin
    vL # "BAG.IO.Länge";
    if (vL=0.0) then begin
      RecLink(819,701,7,_recFirst);   // Warengruppe holen
      vL # Lib_Berechnungen:L_aus_KgStkDBDichte2(BAG.IO.Plan.In.GewN, BAG.IO.Plan.In.Stk, BAG.IO.
    end;
    if (BAG.IO.Plan.In.Stk<>0) then
      BAG.IO.Plan.In.Menge # (BAG.IO.Breite * vL / 1000000.0) * Cnvfi(BAG.IO.Plan.In.Stk);
    else
      BAG.IO.Plan.In.Menge # (BAG.IO.Breite * vL / 1000000.0);
  end;
//todo(BAG.IO.MEH.In+'   '+BAG.IO.MEH.Out+'  '+anum(bag.io.plan.in.menge,2));
  if (BAG.P.Aktion=c_BAG_Spulen) then
    BAG.IO.Plan.In.Menge  # BAG.F.Menge;

// usw. TODO

  if (BAG.IO.NachBAG<>0) then begin     // NÄCHSTEN Schritt prüfen
    vBuf702 # RekSave(702);
    BAG.P.Nummer    # BAG.IO.NachBAG;
    BAG.P.Position  # BAG.IO.NachPosition;
    Erx # RecRead(702,1,0);
    if (Erx=_rOK) then vReinIstRaus # vReinIstRaus or BAG.P.Typ.VSBYN;
    RekRestore(vBuf702);
  end;
```

```
// Eingang=Ausgang bei Weiterbearbeitung
if (vReinIstRaus) or (BAG.IO.NachBAG=0) then begin
  BAG.IO.Plan.Out.Stk   # BAG.IO.Plan.In.Stk;
  BAG.IO.Plan.Out.GewN  # BAG.IO.Plan.In.GewN;
  BAG.IO.Plan.Out.GewB  # BAG.IO.Plan.In.GewB;
  if (BAG.IO.MEH.In=BAG.IO.MEH.Out) then
    BAG.IO.Plan.Out.Meng  # BAG.IO.Plan.In.Menge
  else if (BAG.F.MEH=BAG.IO.MEH.Out) then
    BAG.IO.Plan.Out.Meng  # BAG.F.Menge
  else if (BAG.IO.MEH.Out='Stk') then
    BAG.IO.Plan.In.Menge # cnvfi(BAG.IO.Plan.Out.Stk)
  else if (BAG.IO.MEH.Out='kg') then
    BAG.IO.Plan.In.Menge  # BAG.IO.Plan.Out.GewN
  else if (BAG.IO.MEH.Out='t') then
    BAG.IO.Plan.In.Menge  # Rnd(BAG.IO.Plan.Out.GewN/1000.0,Set.Stellen.Menge);
end;


if (vNeu=n) then begin
  BAG.IO.UrsprungsID    # BAG.IO.ID;
  // 14.11.2013 bei Schopf AnfagnsID übernehmen
  if (BAG.F.Fertigung=999) then
    BAG.IO.UrsprungsID    # vBuf701->BAG.IO.UrSprungsID;

  Erx # RecRead(701,1,_RecLock | _recNoLoad);
```

```
    if (erx=_rOK) then begin
      //BAG.IO.Materialnr # 0;
      Erx # BA1_IO_Data:Replace(_recUnlock,'AUTO');
    end;
  end
  else begin                        // NEUANLAGE
    REPEAT
      BAG.IO.ID             # BAG.IO.ID + 1;
      BAG.IO.UrsprungsID    # BAG.IO.ID;
      // 14.11.2013 bei Schopf AnfagnsID übernehmen
      if (BAG.F.Fertigung=999) then
        BAG.IO.UrsprungsID    # vBuf701->BAG.IO.UrSprungsID;
      Erx # BA1_IO_Data:Insert(0,'AUTO');
//debug('NEU KEY701 '+anum(BAG.IO.Plan.In.GewN,0));
      if (Erx=_rDeadLock) or (Erx=99) then BREAK;
    UNTIL (Erx=_rOK);
  end;


  if (Erx<>_rOK) then begin
    RekRestore(vBuf701);
@ifdef LogFlow
debug('...update XzuY');
@endif
    RETURN false;
  end;
```

```
  // Fertigung wird weiterbearbeitet???
  if (BAG.IO.NachBAG<>0) then begin
    vOk # UpdateOutputKind(aDel, aDel702, aIgnorekgmm);   // 17.01.2020 AH "aIgnorekgMm" angehängt
// AI
    if (vOK=false) then begin
      RekRestore(vBuf701);
@ifdef LogFlow
debug('...update XzuY');
@endif
      RETURN false;
    end;
  end;

  RekRestore(vBuf701);
@ifdef LogFlow
debug('...update XzuY!');
@endif
  RETURN true;
end;



//=========================================================================
//  UpdateArtPrd
//
//=========================================================================
```

```
sub UpdateArtPrd(
  aDel        : logic;
  opt aDel702 : logic) : logic;
local begin
  Erx          : int;
  vBuf701      : int;
  vBuf702      : int;
  vNeu         : logic;
  vOK          : logic;
  vReinIstRaus  : logic;
  vL           : float;
  vVorID       : int;
end;
begin

@ifdef LogFlow
debug('update ArdPrd...');
@endif

  vBuf701 # RekSave(701);

  vNeu # y;
  RecBufClear(701);         // genau DIESEN Output finden
  BAG.IO.VonBAG       # BAG.F.Nummer;
  BAG.IO.VonPosition  # BAG.F.Position;
  BAG.IO.VonFertigung # BAG.F.Fertigung;
```

```
  BAG.IO.VonID        # vVorID;
  Erx # RecRead(701,3,0);
  if (BAG.IO.VonBAG=BAG.F.Nummer) and
     (BAG.IO.VonPosition=BAG.F.Position) and
     (BAG.IO.VonID=vVorID) and
     (BAG.IO.VonFertigung=BAG.F.Fertigung) and
     ((Erx<=_rLocked) or (Erx=_rMultikey)) then vNeu # n;

  if (aDel) then begin
    vOk # y;

    if (vNeu=n) then begin
      // Fertigung wird weiterbearbeitet???
      if (BAG.IO.NachBAG<>0) then begin
        vOk # UpdateOutputKind(aDel, aDel702);
        if (vOK=false) then begin
          RekRestore(vBuf701);
@ifdef LogFlow
debug('...update ArtPrd');
@endif
          RETURN false;
        end;
      end;
      Erx # BA1_IO_Data:Delete(0,'AUTO');
      vOk # (Erx=_rOK);
    end;
```

```
    RekRestore(vBuf701);
@ifdef LogFlow
debug('...update XzuY');
@endif
if (vOK=false) then todo('X');
    RETURN vok;
  end;


  if (vNeu) then begin                  // Neuanlage??
    RecBufClear(701);
    RecBufCopy(vBuf701,701);
    BAG.IO.Nummer         # BAG.F.Nummer;
    BAG.IO.ID             # 0;
    BAG.IO.VonBAG         # BAG.F.Nummer;
    BAG.IO.VonPosition    # BAG.F.Position;
    BAG.IO.VonFertigung   # BAG.F.Fertigung;
    BAG.IO.VonID          # vVorID;
    BAG.IO.NachBAG        # 0;
    BAG.IO.NachPosition   # 0;
    BAG.IO.NachFertigung  # 0;
    BAG.IO.Materialtyp    # c_IO_BAG;
    BAG.IO.Materialnr     # 0;
    BAG.IO.Ist.In.Stk     # 0;
    BAG.IO.Ist.In.GewN    # 0.0;
    BAG.IO.Ist.In.GewB    # 0.0;
    BAG.IO.Ist.In.Menge   # 0.0;
```

```
  BAG.IO.MEH.IN         # BAG.F.MEH; //vBuf701->BAG.IO.MEH.Out;
  BAG.IO.MEH.Out        # BAG.F.MEH;


  BAG.IO.BruderID       # 0;
  BAG.IO.VonFertigmeld  # 0;
  BAG.IO.Ist.Out.Stk    # 0;
  BAG.IO.Ist.Out.GewN   # 0.0;
  BAG.IO.Ist.Out.GewB   # 0.0;
  BAG.IO.Ist.Out.Menge  # 0.0;
end
else begin
  RecRead(701,1,0);
end;



// Daten von Einsatz und Fertigung zusammenführen
"BAG.IO.Güte"         # vBuf701->"BAG.IO.Güte";
"BAG.IO.Gütenstufe"   # vBuf701->"BAG.IO.Gütenstufe";
BAG.IO.Dicke          # vBuf701->BAG.IO.Dicke;
BAG.IO.Breite         # vBuf701->BAG.IO.Breite;
BAG.IO.Spulbreite     # vBuf701->BAG.IO.Spulbreite;
"BAG.IO.Länge"        # vBuf701->"BAG.IO.Länge";
BAG.IO.DickenTol      # vBuf701->BAG.IO.DickenTol;
BAG.IO.BreitenTol     # vBuf701->BAG.IO.BreitenTol;
"BAG.IO.LängenTol"    # vBuf701->"BAG.IO.LängenTol";
```

```
BAG.IO.Lageradresse   # vBuf701->BAG.IO.Lageradresse;
BAG.IO.Lageranschr    # vBuf701->BAG.IO.Lageranschr;

if (BAG.P.ExterneLiefNr<>0) then begin   // 03.01.2017 AH: Lagerorte liegen immer beim Produzenten
  Erx # RecLink(100,702,7,_RecFirst);   // Produzent holen
  if (Erx<=_rLocked) then begin
    BAG.IO.Lageradresse # Adr.Nummer;
    BAG.IO.Lageranschr  # 1;
    if (BAG.P.ExterneLiefAns<>0) then
      BAG.IO.Lageranschr # BAG.P.ExterneLiefAns;
  end;
end;

BAG.IO.MEH.IN         # BAG.F.MEH;
BAG.IO.MEH.Out        # BAG.F.MEH;

if ("BAG.F.Güte"<>'') then          "BAG.IO.Güte"       # "BAG.F.Güte";
if ("BAG.F.Gütenstufe"<>'') then    "BAG.IO.Gütenstufe" # "BAG.F.Gütenstufe";
if (BAG.F.Dicke<>0.0) then begin
  BAG.IO.Dicke        # BAG.F.Dicke;
  BAG.IO.Dickentol    # BAG.F.Dickentol;
end;
if (BAG.F.Breite<>0.0) then begin
  BAG.IO.Breite       # BAG.F.Breite;
  BAG.IO.BreitenTol   # BAG.F.BreitenTol;
end;
```

```
if ("BAG.F.Länge"<>0.0) then begin
  "BAG.IO.Länge"      # "BAG.F.Länge";
  "BAG.IO.Längentol"  # "BAG.F.LängenTol";
end;
if (BAG.F.Spulbreite<>0.0) then begin
  BAG.IO.Spulbreite   # BAG.F.Spulbreite;
end;
if (BAG.F.Warengruppe<>0) then      BAG.IO.Warengruppe  # BAG.F.Warengruppe;
if (BAG.F.Artikelnummer<>'') then   BAG.IO.Artikelnr    # BAG.F.Artikelnummer;

// 26.10.2012 AI: Kommission erstmal löschen
BAG.IO.Auftragsnr     # 0;
BAG.IO.Auftragspos    # 0;
BAG.IO.AuftragsFert   # 0;
if (BAG.F.Auftragsnummer<>0) then begin
  BAG.IO.Auftragsnr     # BAG.F.Auftragsnummer;
  BAG.IO.Auftragspos    # BAG.F.Auftragspos;
  BAG.IO.AuftragsFert   # BAG.F.Auftragsfertig;
end
else if (BAG.P.Auftragsnr<>0) then begin
  if (BAG.F.WirdEigenYN=false) then begin
    BAG.IO.Auftragsnr     # BAG.P.Auftragsnr;
    BAG.IO.Auftragspos    # BAG.P.Auftragspos;
    BAG.IO.AuftragsFert   # 0;
  end;
end
```

```
else begin
  BAG.IO.Auftragsnr     # vBuf701->BAG.IO.Auftragsnr;
  BAG.IO.Auftragspos    # vBuf701->BAG.IO.Auftragspos;
  BAG.IO.AuftragsFert   # vBuf701->BAG.IO.Auftragsfert;
end;
BAG.F.Bemerkung # '';
if (BAG.IO.Auftragsnr<>0) then
  BAG.IO.Bemerkung  # cnvai(BAG.IO.Auftragsnr)+'/'+cnvai(BAG.IO.Auftragspos);

BAG.IO.Plan.In.Stk   # "BAG.F.Stückzahl";
BAG.IO.Plan.In.GewN  # BAG.F.Gewicht;
BAG.IO.Plan.In.GewB  # BAG.F.Gewicht;

if (BAG.IO.MEH.In='Stk') then
  BAG.IO.Plan.In.Menge # cnvfi(BAG.IO.Plan.In.Stk)
else if (BAG.IO.MEH.In='kg') then
  BAG.IO.Plan.In.Menge  # BAG.IO.Plan.In.GewN
else if (BAG.IO.MEH.In='t') then
  BAG.IO.Plan.In.Menge  # Rnd(BAG.IO.Plan.In.GewN/1000.0,Set.Stellen.Menge)
else if (BAG.IO.MEH.In='m') then
  BAG.IO.Plan.In.Menge  # Rnd("BAG.IO.Länge" * cnvfi("BAG.F.Stückzahl") / 1000.0, Set.Stellen.Menge)
else if (BAG.IO.MEH.In='qm') then begin
  vL # "BAG.IO.Länge";
  if (vL=0.0) then begin
    RecLink(819,701,7,_recFirst);   // Warengruppe holen
    vL # Lib_Berechnungen:L_aus_KgStkDBDichte2(BAG.IO.Plan.In.GewN, BAG.IO.Plan.In.Stk, BAG.IO.
```

```
    end;
    if (BAG.IO.Plan.In.Stk<>0) then
      BAG.IO.Plan.In.Menge # (BAG.IO.Breite * vL / 1000000.0) * Cnvfi(BAG.IO.Plan.In.Stk);
    else
      BAG.IO.Plan.In.Menge # (BAG.IO.Breite * vL / 1000000.0);
  end;
//todo(BAG.IO.MEH.In+'   '+BAG.IO.MEH.Out+'  '+anum(bag.io.plan.in.menge,2));
  if (BAG.P.Aktion=c_BAG_Spulen) then
    BAG.IO.PLan.In.Menge  # BAG.F.Menge;

// usw. TODO

  if (BAG.IO.NachBAG<>0) then begin     // NÄCHSTEN Schritt prüfen
    vBuf702 # RekSave(702);
    BAG.P.Nummer    # BAG.IO.NachBAG;
    BAG.P.Position  # BAG.IO.NachPosition;
    Erx # RecRead(702,1,0);
    if (Erx=_rOK) then vReinIstRaus # BAG.P.Typ.VSBYN;
    RekRestore(vBuf702);
  end;

  // Eingang=Ausgang bei Weiterbearbeitung
  if (vReinIstRaus) or (BAG.IO.NachBAG=0) then begin
    BAG.IO.Plan.Out.Stk   # BAG.IO.Plan.In.Stk;
    BAG.IO.Plan.Out.GewN  # BAG.IO.Plan.In.GewN;
    BAG.IO.Plan.Out.GewB  # BAG.IO.Plan.In.GewB;
```

```
  if (BAG.IO.MEH.In=BAG.IO.MEH.Out) then
    BAG.IO.Plan.Out.Meng  # BAG.IO.Plan.In.Menge
  else if (BAG.F.MEH=BAG.IO.MEH.Out) then
    BAG.IO.Plan.Out.Meng  # BAG.F.Menge
  else if (BAG.IO.MEH.Out='Stk') then
    BAG.IO.Plan.In.Menge # cnvfi(BAG.IO.Plan.Out.Stk)
  else if (BAG.IO.MEH.Out='kg') then
    BAG.IO.Plan.In.Menge  # BAG.IO.Plan.Out.GewN
  else if (BAG.IO.MEH.Out='t') then
    BAG.IO.Plan.In.Menge  # Rnd(BAG.IO.Plan.Out.GewN/1000.0,Set.Stellen.Menge);
end;



if (vNeu=n) then begin
  BAG.IO.UrsprungsID    # BAG.IO.ID;
  Erx # RecRead(701,1,_RecLock | _recNoLoad);
  if (Erx=_rOK) then begin
    //BAG.IO.Materialnr # 0;
    Erx # BA1_IO_Data:Replace(_recUnlock,'AUTO');
  end;
end
else begin                      // NEUANLAGE
  REPEAT
    BAG.IO.ID             # BAG.IO.ID + 1;
    BAG.IO.UrsprungsID    # BAG.IO.ID;
    Erx # BA1_IO_Data:Insert(0,'AUTO');
```

```
      if (Erx=_rDeadLock) then BREAK;
    UNTIL (Erx=_rOK);
  end;


  if (Erx<>_rOK) then begin
    RekRestore(vBuf701);
@ifdef LogFlow
debug('...update XzuY');
@endif
    RETURN false;
  end;



  // Fertigung wird weiterbearbeitet???
  if (BAG.IO.NachBAG<>0) then begin
    vOk # UpdateOutputKind(aDel, aDel702);
// AI
    if (vOK=false) then begin
      RekRestore(vBuf701);
@ifdef LogFlow
debug('...update XzuY');
@endif
      RETURN false;
    end;
  end;
```

```
  RekRestore(vBuf701);
@ifdef LogFlow
debug('...update XzuY!');
@endif
  RETURN true;
end;



//=========================================================================
//  UpdateOutput
//        Summiert BAG.F. auch neu
//=========================================================================
sub UpdateOutput(
  aDatei              : int;
  opt aDel            : logic;
  opt aDel702         : logic;
  opt aIgnoreKgMM     : logic;
  opt aLfaNichtUpdaten  : logic;    // 17.02.2020 AH: neuer Para
  opt aPerTodo        : logic;    // 09.09.2020 AH
  ) : logic;
local begin
  Erx       : int;
  vPPos     : Int;
  vLevel    : int;
  vBuf701   : int;
  vBuf702   : int;
```

```
  vBuf703   : int;
  vNeu      : logic;
  vOk       : logic;
end;
begin
//debugX('--------- OupdateOutput '+aint(aDAtei)+'   KEY702   KEY703');

  vBuf701 # RekSave(701);
  vBuf702 # RekSave(702);
  vBuf703 # RekSave(703);



@ifdef LogFlow
debug('update Output...');
@endif



  // RESTFERTIGUNG?
  // dann genau prüfen, welcher Einsatz einen Schopf bildet
  if (BAG.F.Fertigung=999) and (aDatei=703) and
    (("BAG.P.Typ.1In-YOutYN"=y) or ("BAG.P.Typ.xIn-yOutYN"=y)) then begin
    vOk # true;
    Erx # RecLink(701,702,2,_recFirst); // Input loopen
    WHILE (Erx=_rok) and (vOK) do begin
      if (BAG.IO.vonFertigmeld=0) then begin
        vOk # UpdateXzuY(aDel,n, aIgnoreKgmm);
```

```
      end;
      Erx # RecLink(701,702,2,_recNext);
    END;

    if (Erx=_rLocked) then vOk # false;

    RekRestore(vBuf701);
    RekRestore(vBuf702);
    RekRestore(vBuf703);

    if (vOK) then RunAFX('BAG.F.UpdateOutput.Post',aint(aDatei));
@ifdef LogFlow
debug('...update Output');
@endif
    RETURN vOk;
  end;

// AI
//return true;
//todo(aint(bag.p.position));
  // ARTIKELPRODUKTION ?=================================================
  if (BAG.P.Aktion=c_BAG_ArtPrd) then begin

    // Modifikation Einsatz?? -------------------------
    if (aDatei=701) then begin
      vOk # y;
```

```
  Erx # RecLink(703,702,4,_RecFirst);   // Fertigungen loopen
  WHILE (Erx=_rOK) and (vOK) do begin
    if (BAG.IO.nachFertigung=0) or (BAG.IO.nachFertigung=BAG.F.Fertigung) then begin
      if (BAG.IO.vonFertigmeld=0) and (BAG.F.Fertigung<>999) then begin
        BA2_P_Data:SumInput(701);
        Erx # RecRead(703,1,_recLock);
        if (erx<>_rOK) then vOK # false;
        if (vOK) then begin
          ErrechnePlanmengen(y,y,y);
          Erx # RekReplace(703,_recunlock,'AUTO');
          if (erx<>_rOK) then vOK # false;
          if (vOK) then vOk # UpdateXzuY(n,n, aIgnoreKgmm);         // Update
        end;
      end;
    end;
    Erx # RecLink(703,702,4,_RecNext);   // Fertigungen loopen
  END;


end
    // Modifikation Fertigung?? ------------------------
else if (aDatei=703) then begin
  vOk # UpdateXzuY(aDel,n, aIgnoreKgmm);             // Update
end
else begin
  vOk # y;
```

```
    end;


    if (vOK) then begin  // ggf. Lohnarbeitsgang in Auftrag schreiben
      vOk # BA1_P_Data:UpdateAufAktion(n);
    end;

    RekRestore(vBuf701);
    RekRestore(vBuf702);
    RekRestore(vBuf703);

    if (vOK) then RunAFX('BAG.F.UpdateOutput.Post',aint(aDatei));
@ifdef LogFlow
debug('...update Output');
@endif
    RETURN vOk;

  end;  // ArtProd


  // SÄGEN?? ================================================================
  if (BAG.P.Aktion=c_BAG_Saegen) then begin

    // Modifikation Einsatz?? -------------------------
    if (aDatei=701) then begin
      vOk # y;
```

```
  Erx # RecLink(703,702,4,_RecFirst);   // Fertigungen loopen
  WHILE (Erx=_rOK) and (vOK) do begin
    if (BAG.IO.nachFertigung=0) or (BAG.IO.nachFertigung=BAG.F.Fertigung) then begin
      if (BAG.IO.vonFertigmeld=0) and (BAG.F.Fertigung<>999) then begin
        BA2_P_Data:SumInput(701);
        Erx # RecRead(703,1,_recLock);
        if (Erx<>_rOK) then vOK # false;
        if (vOK) then begin
          ErrechnePlanmengen(y,y,y);
          Erx # RekReplace(703,_recunlock,'AUTO');
          if (Erx<>_rOK) then vOK # false;
          if (vOK) then vOk # UpdateXzuY(n,n, aIgnoreKgmm);         // Update
        end;
      end;
    end;
    Erx # RecLink(703,702,4,_RecNext);   // Fertigungen loopen
  END;


end
    // Modifikation Fertigung?? ------------------------
else if (aDatei=703) then begin
  vOk # UpdateXzuY(aDel,n, aIgnoreKgmm);             // Update
end
else begin
  vOk # y;
```

```
    end;



    // ggf. Schopf bilden
    if (vOK) and ((aDatei<>703) or (BAG.F.AutomatischYN=n)) then begin
      vOk # UpdateSchopf(aDel);
    end;

    if (vOK) then begin  // ggf. Lohnarbeitsgang in Auftrag schreiben
      vOk # BA1_P_Data:UpdateAufAktion(n);
    end;

    RekRestore(vBuf701);
    RekRestore(vBuf702);
    RekRestore(vBuf703);

    if (vOK) then RunAFX('BAG.F.UpdateOutput.Post',aint(aDatei));
@ifdef LogFlow
debug('...update Output');
@endif
    RETURN vOk;

  end;  // SÄGEN



  // 1zu1 Arbeitsgang ========================================
```

```
// 1zu1 Arbeitsgang ========================================

// 1zu1 Arbeitsgang ========================================

if ("BAG.P.Typ.1In-1OutYN"=y) then begin


  // Modifikation Einsatz?? -------------------------

  if (aDatei=701) and (BA1_IO_I_data:IstMatBeistellung()=false) then begin

    vOk # y;

    if (BAG.IO.vonFertigmeld=0) then begin

      if (BAG.IO.nachFertigung=0) then begin

        FOR Erx # RecLink(703,702,4,_RecFirst)  // alle Fertigungen loopen

        LOOP Erx # RecLink(703,702,4,_RecNext)

        WHILE (Erx=_rOK) and (vOK) do begin

          // 20.04.2018 AH: Fertigung neu rechnen...

          Erx # RecRead(703,1,_RecLock);

          if (vOK) then begin

            ErrechnePlanmengen(y,y,y);

            Erx # BA1_F_Data:Replace(_recUnlock,'AUTO');

            if (erx<>_rOK) then vOK # false;

            if (vOK) then vOk # Update1zu1(701,aDel,aDel702, aIgnoreKgmm);            // Update

          end;

        END;

      end

      else begin

        Erx # RecLink(703,701,10,_recFirst);    // Nach-Fergigung holen

        if (Erx<=_rLocked) then begin

          // 20.04.2018 AH: Fertigung neu rechnen...
```

```
        Erx # RecRead(703,1,_RecLock);
        if (erx<>_rOK) then vOK # false;
        if (vOK) then begin
          ErrechnePlanmengen(y,y,y);
          Erx # BA1_F_Data:Replace(_recUnlock,'AUTO');
          if (erx<>_rOK) then vOK # false;
          if (vOK) then vOk # Update1zu1(701,aDel, aDel702, aIgnoreKgmm);            // Update
        end;
      end;
    end;
  end;


end
    // Modifikation Fertigung?? -------------------------
else if (aDatei=703) then begin


  vOk # true;
  Erx # RecLink(701,702,2,_recFirst); // Input loopen
  WHILE (Erx=_rok) and (vOK) do begin
    if (BAG.IO.vonFertigmeld=0) then begin
      vOk # Update1zu1(aDatei,aDel, aDel702, aIgnoreKgmm);  // Update
    end;
    Erx # RecLink(701,702,2,_recNext);
  END;
  if (Erx=_rLocked) then vOk # false;
end
```

```
    else begin
      vOk # y;
    end;

    // ggf. Schopf bilden
    if (vOK) and (aDel=n) and
      ((aDatei<>703) or (BAG.F.AutomatischYN=n)) and
      ((BAG.P.aktion<>c_BAG_Fahr) and (BAG.P.Aktion<>c_BAG_Umlager) and (BAG.P.Aktion<>c_BAG_
      vOk # UpdateSchopf(aDel);

    if (vOK) then   // ggf. Lohnarbeitsgang in Auftrag schreiben
      vOk # BA1_P_Data:UpdateAufAktion(n);
//debugx(BAG.P.Aktion);
    if (vOK) and (aLfaNichtUpdaten=false) then begin
      // passenden LFS erzeugen/updaten
      if (BAG.P.Aktion=c_BAG_Fahr) OR (Bag.P.Aktion=c_BAG_Umlager) then begin
        if (aDatei=701) then begin
          Erx # RecLink(703,701,10,_recFirst);    // nachFertigung holen
          vOK # Lfs_LFA_Data:ErzeugeLFSausLFA();
        end
        else begin
          if (aDatei=702) then begin
            Erx # RecLink(703,702,4,_RecFirst);   // Fertigungen loopen
            WHILE (Erx=_rOK) and (vOK) do begin
              vOK # Lfs_LFA_Data:ErzeugeLFSausLFA();
              Erx # RecLink(703,702,4,_RecNext);  // Fertigungen loopen
```

```
          END;
        end;
      end;
    end;

    // passenden LFS erzeugen/updaten
    if (BAG.P.Aktion=c_BAG_Versand) then begin
      vOK # VsP_Data:ErzeugePoolZumVersand();
    end;
  end;

  RekRestore(vBuf701);
  RekRestore(vBuf702);
  RekRestore(vBuf703);

  if (vOK) then RunAFX('BAG.F.UpdateOutput.Post',aint(aDatei));
@ifdef LogFlow
debug('...update Output');
@endif
  RETURN vOk;
 end;


 // 1zuY Arbeitsgang ========================================
 // 1zuY Arbeitsgang ========================================
 // 1zuY Arbeitsgang ========================================
```

```
if ("BAG.P.Typ.1In-yOutYN"=y) then begin


  // Modifikation Einsatz?? -------------------------
  if (aDatei=701) and (BA1_IO_I_data:IstMatBeistellung()=false) then begin
    vOk # true;
    Erx # RecLink(703,702,4,_RecFirst);   // Fertigungen loopen
    WHILE (Erx=_rOK) and (vOK) do begin
      if (BAG.IO.vonFertigmeld=0) then
        vOk # Update1zuY(aDel,n, aIgnoreKgmm);             // Update
      Erx # RecLink(703,702,4,_RecNext);
    END;
    RecBufCopy(vBuf701,701);
    if (Erx=_rLocked) then vOk # n;
  end
      // Modifikation Fertigung?? -------------------------
  else if (aDatei=703) then begin


    vOk # true;
    Erx # RecLink(701,702,2,_recFirst);   // Input loopen
    WHILE (Erx=_rok) and (vOK) do begin
      if (BAG.IO.vonFertigmeld=0) then begin
        vOk # Update1zuY(aDel,n, aIgnoreKgmm);             // Update
      end;
      Erx # RecLink(701,702,2,_recNext);
    END;
    if (Erx=_rLocked) then vOk # false;
```

```
end
else begin
  vOk # y;
end;

// ggf. Schopf bilden
if (vOK) and ((aDatei<>703) or (BAG.F.AutomatischYN=n)) then begin
  vOk # UpdateSchopf(aDel);
end;

if (vOK) then       // ggf. Lohnarbeitsgang in Auftrag schreiben
  vOk # BA1_P_Data:UpdateAufAktion(n);
end;

// Fertigungen von z.B: Spalten neu summieren
// hier statt in BA1_F_Main:RecSave
if (vOK) then begin

  FOR Erx # RecLink(703,702,4,_RecFirst)  // Fertigungen loopen
  LOOP Erx # RecLink(703,702,4,_recNext)
  WHILE (Erx<=_rLockeD) and (vOK) do begin

    Erx # RecRead(703,1,_recLock);
    if (Erx<>_rOK) then begin
      vOK # false;
      BREAK;
```

```
        end;
        "BAG.F.Stückzahl" # 0;
// 23.05.2019 AH 1zuX kann immer Gewichte neu rechnen:
// 23.05.2019 if (BAG.P.Aktion=c_BAG_Spalt) or (BAG.P.Aktion=c_BAG_Schael) or (BAG.P.Aktion=c_BAG
        "BAG.F.Gewicht" # 0.0;

        FOR Erx # RecLink(701,703,4,_recFirst)  // Output loopen
        LOOP Erx # RecLink(701,703,4,_recNext)
        WHILE (Erx<=_rLocked) do begin

          if (BAG.IO.Materialtyp<>c_IO_BAG) then CYCLE;

          "BAG.F.Stückzahl" # "BAG.F.Stückzahl" + BAG.IO.Plan.In.Stk;

          if (BAG.P.Aktion=c_BAG_Spalt) then
            "BAG.F.Gewicht" # "BAG.F.Gewicht" + BAG.IO.Plan.In.Menge
          else if (BAG.P.Aktion=c_BAG_Qteil) then
            "BAG.F.Gewicht" # "BAG.F.Gewicht" + BAG.IO.Plan.In.Menge
          else if (BAG.P.Aktion=c_BAG_Schael) then
            "BAG.F.Gewicht" # "BAG.F.Gewicht" + BAG.IO.Plan.In.GewN
          else
// 23.05.2019 AH 1zuX kann immer Gewichte neu rechnen:
            "BAG.F.Gewicht" # "BAG.F.Gewicht" + BAG.IO.Plan.In.Menge;
        END;

//debugx(aint("BAG.F.Stückzahl"));
```

```
        Erx # BA1_F_Data:Replace(_recUnlock,'AUTO');
        if (Erx<>_rOK) then vOK # false;
      END;  // Fertigungen
//    end;

    RekRestore(vBuf701);
    RekRestore(vBuf702);
    RekRestore(vBuf703);
    if (vOK) then RunAFX('BAG.F.UpdateOutput.Post',aint(aDatei));
@ifdef LogFlow
debug('...update Output');
@endif
    RETURN vOk;

  end;


  // XzuY Arbeitsgang ========================================
  // XzuY Arbeitsgang ========================================
  // XzuY Arbeitsgang ========================================
  if ("BAG.P.Typ.xIn-yOutYN"=y) then begin

    // Modifikation Einsatz?? -------------------------
    if (aDatei=701) and (BA1_IO_I_data:IstMatBeistellung()=false) then begin
      vOk # y;
//      if (gUsername='AH') then
```

```
// neu 23.11.2011
      Erx # RecLink(703,702,4,_RecFirst);   // Fertigungen loopen
      WHILE (Erx=_rOK) and (vOK) do begin
        if (BAG.F.Fertigung<>999) and
          ((BAG.IO.nachFertigung=0) or (BAG.IO.nachFertigung=BAG.F.Fertigung)) then begin
          if (BAG.P.Aktion=c_BAG_WalzSpulen) then begin
            Erx # RecRead(703,1,_RecLock);
            if (Erx<>_rOK) then vOK # false;
            if (vOK) then begin
              ErrechnePlanmengen(y,y,y);
              Erx # BA1_F_Data:Replace(_recUnlock,'AUTO');
            end;
          end;
          if (vOK) then vOk # UpdateXzuY(n,n, aIgnoreKgmm);          // Update
        end;
        Erx # RecLink(703,702,4,_RecNext);   // Fertigungen loopen
      END;
    end
        // Modifikation Fertigung?? -------------------------
    else if (aDatei=703) then begin
//      if (BAG.IO.vonFertigmeld=0) then begin ??? COPY&PASTE Fehler ??
        vOk # UpdateXzuY(aDel,n, aIgnoreKgmm);             // Update
//      end;
    end
    else begin
      vOk # y;
```

```
    end;

    // ggf. Schopf bilden
    if (vOK) and ((aDatei<>703) or (BAG.F.AutomatischYN=n)) then begin
      vOk # UpdateSchopf(aDel);
    end;

    if (vOK) then begin  // ggf. Lohnarbeitsgang in Auftrag schreiben
      vOk # BA1_P_Data:UpdateAufAktion(n);
    end;

    RekRestore(vBuf701);
    RekRestore(vBuf702);
    RekRestore(vBuf703);

    if (vOK) then RunAFX('BAG.F.UpdateOutput.Post',aint(aDatei));
@ifdef LogFlow
debug('...update Output');
@endif
    RETURN vOk;

  end;


  // VSB  Arbeitsgang ========================================
  // VSB  Arbeitsgang ========================================
```

```
// VSB  Arbeitsgang ========================================
if (BAG.P.Typ.VSBYN=y) then begin

  // Modifikation Position? -------------------------
  if (aDatei=702) then begin
    vOk # true;
    Erx # RecLink(701,702,2,_recFirst);   // Input loopen
    WHILE (Erx=_rok) and (vOK) do begin
      if (BAG.IO.vonFertigmeld=0) then begin
        vOk # UpdateVSB(aDel, aPerTodo);              // Update
      end;
      Erx # RecLink(701,702,2,_recNext);
    END;
    if (Erx=_rLocked) then vOk # false;
  end
  else begin
    vOk # UpdateVSB(aDel, aPerTodo);               // Update
  end;

  if (vOK) then                         // ggf. Lohnarbeitsgang in Auftrag schreiben
    vOk # BA1_P_Data:UpdateAufAktion(n);

end;

if (vOK) then RunAFX('BAG.F.UpdateOutput.Post',aint(aDatei));
```

```
  RekRestore(vBuf701);
  RekRestore(vBuf702);
  RekRestore(vBuf703);
@ifdef LogFlow
debug('...update Output');
@endif
  RETURN vOK;

end;


//=========================================================================
//  UpdateOutputKind
//              zu einem Output
//=========================================================================
sub UpdateOutputKind(
  aDel        : logic;
  opt aDel702 : logic;
  opt aIgnoreKgMM : logic;
) : logic;
local begin
  Erx       : int;
  vPPos     : Int;
  vLevel    : int;
  vBuf701   : int;
  vBuf702   : int;
```

```
  vBuf703   : int;
  vNeu      : logic;
  vOk       : logic;
  vKGMM_Kaputt  : logic;
end;
begin
  if (BAG.IO.NachBAG=0) then RETURN true;

  vBuf701 # RekSave(701);
  vBuf702 # RekSave(702);
  vBuf703 # RekSave(703);

  BAG.P.Nummer    # BAG.IO.NachBAG;    // Nachfolger Arbeitsgang holen
  BAG.P.Position  # BAG.IO.NachPosition;
  Erx # RecRead(702,1,0);
  if (Erx<>_rOK) then begin
    RekRestore(vBuf701);
    RekRestore(vBuf702);
    RekRestore(vBuf703);
    RETURN false;
  end;

@ifdef LogFlow
debug('update Kinder...');
@endif
```

```
  // 07.11.2013:
  if (aDel) then begin
    vOK # BA1_IO_I_Data:DeleteInput(aDel702);
  end
  else begin
    // 25.02.2019 AH: Autoteilung rekursiv...
    if (BA1_IO_data:Autoteilung(var vKGMM_Kaputt)=false) then begin
      RekRestore(vBuf701);
      RekRestore(vBuf702);
      RekRestore(vBuf703);
      RETURN false;
    end;

    // 24.05.2019 AH: Prj.1811/72: Beim Einsatz von WE gegen VSB sollen keine Kgmm-Fehler kommen!
    if (aIgnoreKgMM) then vKGMM_Kaputt # false;

    if (vKGMM_Kaputt) then begin
      ERROR(703006,aint(BAG.P.Position));
//EVTL NICHT ABBRECHEN !!!!

      RekRestore(vBuf701);
      RekRestore(vBuf702);
      RekRestore(vBuf703);
      RETURN false;
    end;
```

```
    // neu rechnen mit verändertem Input (701)
    vOk # UpdateOutput(701,aDel,n,aIgnorekgmm);     // 06.12.2021 AH mit aIgnoreKgMM
  end;


@ifdef LogFlow
debug('...update Kinder');
@endif

  RekRestore(vBuf701);
  RekRestore(vBuf702);
  RekRestore(vBuf703);

  RETURN vOK;


/****

  BAG.P.Nummer    # BAG.IO.NachBAG;    // Nachfolger Arbeitsgang holen
  BAG.P.Position  # BAG.IO.NachPosition;
  Erx # RecRead(702,1,0);
  if (Erx<>_rOK) then begin
    RekRestore(vBuf701);
    RekRestore(vBuf702);
    RekRestore(vBuf703);
    RETURN false;
```

```
end;

// 1zu1 Arbeitsgang ========================================
// 1zu1 Arbeitsgang ========================================
// 1zu1 Arbeitsgang ========================================
if ("BAG.P.Typ.1In-1OutYN"=y) then begin
  vOk # n;
  Erx # RecLink(703,701,10,_RecFirst);  // zugehörige Fertigung holen
  if (Erx<=_rLocked) then
    vok # Update1zu1(701,aDel);         // Update

  if (vOK) then begin                   // ggf. Lohnarbeitsgang in Auftrag schreiben
    BA1_P_Data:UpdateAufAktion(aDel);
  end;

  RekRestore(vBuf701);
  RekRestore(vBuf702);
  RekRestore(vBuf703);
  RETURN vOk;
end;


// 1zuY Arbeitsgang ========================================
// 1zuY Arbeitsgang ========================================
// 1zuY Arbeitsgang ========================================
if ("BAG.P.Typ.1In-yOutYN"=y) then begin
```

```
  vOk # true;
  Erx # RecLink(703,702,4,_RecFirst);   // Fertigungen loopen
  WHILE (Erx=_rOK) and (vOK) do begin
    vOk # Update1zuY(aDel);             // Update
    Erx # RecLink(703,702,4,_RecNext);
  END;

  if (vOK) then begin                   // ggf. Lohnarbeitsgang in Auftrag schreiben
    BA1_P_Data:UpdateAufAktion(aDel);
  end;
  RecBufCopy(vBuf701,701);
  if (Erx=_rLocked) then vOk # n
  else vOk # UpdateSchopf(aDel);

  RekRestore(vBuf701);
  RekRestore(vBuf702);
  RekRestore(vBuf703);
  RETURN vOk;

end;


// XzuY Arbeitsgang ========================================
// XzuY Arbeitsgang ========================================
// XzuY Arbeitsgang ========================================
if ("BAG.P.Typ.xIn-yOutYN"=y) then begin
```

```
    vOk # true;
    if (vOK) then begin                    // ggf. Lohnarbeitsgang in Auftrag schreiben
      BA1_P_Data:UpdateAufAktion(aDel);
    end;

    RekRestore(vBuf701);
    RekRestore(vBuf702);
    RekRestore(vBuf703);
    RETURN vOk;

  end;


  // VSB  Arbeitsgang ========================================
  // VSB  Arbeitsgang ========================================
  // VSB  Arbeitsgang ========================================
  vOk # true;
  Erx # RecLink(701,702,2,_recFirst);   // Input loopen
  WHILE (Erx=_rok) and (vOK) do begin
    vOk # UpdateVSB(aDel);              // Update
    Erx # RecLink(701,702,2,_recNext);
  END;

  if (Erx=_rLocked) then vOk # false;
//  vOk # UpdateVSB(aDel);            // Update
```

```
  RekRestore(vBuf701);
  RekRestore(vBuf702);
  RekRestore(vBuf703);
  RETURN vOK;
***/

end;



//=============================================================================
//  UpdateRestCoil
//
//=============================================================================
sub UpdateRestCoil() : logic;
local begin
  Erx   : int;
  vNew  : logic;

  vBB   : float;
  vBL   : float;
  vBGew : float;
  vBM   : float;
  vL    : float;
  vGew  : float;
  vM    : float;
```

```
  vInID  : int;
end;
begin

  case (BAG.P.Aktion) of

    c_BAG_TAFEL,c_BAG_ABCOIL : begin   // Tafelrest

      // Einsatz loopen
      Erx # RecLink(701,702,2,_RecFirst);
      WHILE (Erx<=_rLockeD) do begin
        if (BAG.IO.BruderID=0) then begin
          if (vInID=0) then vInID # BAG.IO.ID
          else vInID # 999;
        end;
        Erx # RecLink(701,702,2,_RecNext);
      END;

      if (vInId=0) or (vInID=999) then RETURN false;

      BAG.F.Block           # '^';
      if (BAG.P.Aktion=c_BAG_ABCOIL) then
        BA1_F_Abcoil_Main:RecalcRest(var vBB,var vBL,var vBGew,var vBM,var vL,var vGew,var vM, false)
      else
        BA1_F_Tafel_Main:RecalcRest(var vBB,var vBL,var vBGew,var vBM,var vL,var vGew,var vM, false);
```

```
TRANSON;


BAG.IO.Nummer # BAG.P.Nummer;
BAG.IO.ID     # vInID;
Erx # RecRead(701,1,_recLock);
if (Erx=_rOK) then begin
  BAG.IO.Plan.Out.GewN  # vGew;
  BAG.IO.Plan.Out.GewB  # vGew;
  BAG.IO.Plan.Out.Meng  # vM;
  //BAG.IO.Materialnr     # 0;
  Erx # BA1_IO_Data:replace(_recUnlock,'AUTO');
end;
if (erx<>_rOK) then begin   // 2022-07-05 AH DEADLOCK
  TRANSBRK;
  RETURN false;
end;

// Ausbringung updaten
if (UpdateOutput(701,n)=false) then begin
  TRANSBRK;
  RETURN false;
end;

TRANSOFF;
RETURN true;
```

```
    end;  // Tafel/Abcoil


  end;


end;



//=========================================================================
//  SumInput
//
//=========================================================================
sub SumInput(aMEH : alpha);
local begin
  Erx         : int;
  vStk        : int;
  vGewN,vGewB : float;
  vM          : float;
  vD,vB,vL    : float;
  vID         : int;
  vTheoL      : float;
end;
begin

  // Einsatzmaterial addieren
  Erx # RecLink(701,702,2,_recFirst);     // Input loopen
  WHILE (Erx<=_rLocked) do begin
```

```
if (BAG.IO.vonFertigmeld=0) and (BAG.IO.Materialtyp<>c_IO_ARt) and (BAG.IO.Materialtyp<>c_IO_Be
  vID # BAG.IO.ID;
  vStk  # vStk  + BAG.IO.Plan.Out.Stk;// * (BAG.IO.Teilungen + 1);
  vGewN # vGewN + BAG.IO.PLan.Out.GewN;
  vGewB # vGewB + BAG.IO.PLan.Out.GewB;

  if (BAG.IO.MEH.Out=aMEH) then begin
    vM    # vM   + BAG.IO.Plan.Out.Meng;
  end
  else begin
    vM # vM + Lib_Einheiten:WandleMEH(701, BAG.IO.Plan.Out.Stk, BAG.IO.Plan.Out.GewN, BAG.IO.P
  end;

  if (vD=0.0) then vD # BAG.IO.Dicke;
  if (vB=0.0) then vB # BAG.IO.Breite;
  if (vL=0.0) then vL # "BAG.IO.Länge";
  if (BAG.IO.Dicke<vD) and (BAG.IO.Dicke<>0.0) then     vD # BAG.IO.Dicke;
  if (BAG.IO.Breite<vB) and (BAG.IO.Breite<>0.0) then   vB # BAG.IO.Breite;
  if ("BAG.IO.Länge"<vL) and ("BAG.IO.Länge"<>0.0) then vL # "BAG.IO.Länge";

  if (BAG.P.Aktion=c_BAG_SCHAEL) then begin
    vTheoL # vTheoL + Lib_Berechnungen:L_aus_KgStkDBWgrArt(BAG.IO.Plan.Out.GewN, 1, BAG.IO.D
  end;

end;
```

```
    Erx # RecLink(701,702,2,_recNext);
  END;

  if (BAG.P.Aktion=c_BAG_SCHAEL) then vL # vTheoL;

  // gültigen Einsatz holen
  BAG.IO.Nummer # BAG.P.Nummer;
  BAG.IO.ID     # vID;
  Erx # RecRead(701,1,0);
  if (Erx>_rLocked) then RecBufClear(701);

  BAG.IO.Plan.Out.Stk   # vStk;
  BAG.IO.Plan.Out.GewN  # vGewN;
  BAG.IO.Plan.Out.GewB  # vGewB;
  BAG.IO.Plan.Out.Meng  # vM;
  BAG.IO.Dicke          # vD;
  BAG.IO.Breite         # vB;
  "BAG.IO.Länge"        # vL;
end;


//=========================================================================
//  07.01.2010 MS
//  SetRidRad
//    setzt RID RAD
//=========================================================================
```

```
sub SetRidRad(aTyp : alpha);
local begin
  Erx     : int;
  vBuf703 : int;
end;
begin

  Erx # RecLinkInfo(703, 702, 4, _recCount); // wie viele Fertigungen gibt es zu der Position schon?

  case aTyp of

    'Init' : begin
      if(Erx >= 1) then begin // existiert schon 1 Fertigung? wenn JA aus der ERSTEN Fertigung uebernehme
        vBuf703 # RekSave(703);
        Erx # RecLink(vBuf703, 702, 4, _recFirst);

        BAG.F.RID    # vBuf703->BAG.F.RID;
        BAG.F.RIDmax # vBuf703->BAG.F.RIDmax;
        BAG.F.RAD    # vBuf703->BAG.F.RAD;
        BAG.F.RADmax # vBuf703->BAG.F.RADmax;
        RecBufDestroy(vBuf703);
      end;
    end;

    'AusKommission' : begin
      if ((Erx = 0) and (Mode = c_ModeNew)) or
```

```
      (BAG.F.Fertigung = 1) then begin
     //or(Erx = 1) and (Mode = c_ModeEdit) then begin // erste Fertigung? wenn JA aus dem Auftrag ueberr
       BAG.F.RID         # Auf.P.RID;
       BAG.F.RIDmax      # Auf.P.RIDmax;
     end;
     BAG.F.RAD         # Auf.P.RAD;
     BAG.F.RADmax      # Auf.P.RADmax;
   end;


  end;
end;



//=========================================================================
//  CopyAdrToVpg
//        RES: VpgNummer
//=========================================================================
sub CopyAdrToVpg(aBaNr : int) : int
local begin
  Erx   : int;
  vI    : int;
  vX    : int;
  vOK   : logic;
end;
begin
```

```
vI # BAG.Nummer;
BAG.Nummer # aBANr;
Erx # RecLink(704,700,2,_recLast);   // letzte Verpackung holen
BAG.Nummer # vI;
if (Erx>_rLocked) then vX # 1
else vX # BAG.VPG.Verpackung + 1;

RecBufClear(704);
BAG.VPG.Nummer        # aBaNr;
BAG.Vpg.AbbindungL    # Adr.V.AbbindungL;
BAG.Vpg.AbbindungQ    # Adr.V.AbbindungQ;
BAG.Vpg.Zwischenlage  # Adr.V.Zwischenlage;
BAG.Vpg.Unterlage     # Adr.V.Unterlage;
BAG.Vpg.Umverpackung  # Adr.V.Umverpackung;
BAG.Vpg.Wicklung      # Adr.V.Wicklung;
BAG.Vpg.StehendYN     # Adr.V.StehendYN;
BAG.Vpg.LiegendYN     # Adr.V.LiegendYN;
BAG.Vpg.Nettoabzug    # Adr.V.Nettoabzug;
"BAG.Vpg.Stapelhöhe"  # "Adr.V.Stapelhöhe";
BAG.Vpg.StapelHAbzug  # Adr.V.StapelhAbzug;
BAG.Vpg.RingkgVon     # Adr.V.RingKgVon;
BAG.Vpg.RingkgBis     # Adr.V.RingKgBis;
BAG.Vpg.KgmmVon       # Adr.V.KgmmVon;
BAG.Vpg.KgmmBis       # Adr.V.KgmmBis;
"BAG.Vpg.StückProVE"  # "Adr.V.StückProVE";
BAG.Vpg.VEkgMax       # Adr.V.VEkgMax;
```

```
BAG.Vpg.RechtwinkMax  # Adr.V.RechtwinkMax;
BAG.Vpg.EbenheitMax   # Adr.V.EbenheitMax;
"BAG.Vpg.SäbeligMax"  # "Adr.V.SäbeligkeitMax";
"BAG.Vpg.SäbelProM"   # "Adr.V.SäbelProM";
BAG.Vpg.Etikettentyp  # Adr.V.Etikettentyp;
BAG.Vpg.Verwiegart    # Adr.V.Verwiegungsart;
BAG.Vpg.VpgText1      # Adr.V.VpgText1;
BAG.Vpg.VpgText2      # Adr.V.VpgText2;
BAG.Vpg.VpgText3      # Adr.V.VpgText3;
BAG.Vpg.VpgText4      # Adr.V.VpgText4;
BAG.Vpg.VpgText5      # Adr.V.VpgText5;
BAG.Vpg.VpgText6      # Adr.V.VpgText6;

if (BAG.Vpg.AbbindungL+BAG.Vpg.AbbindungQ+"BAG.Vpg.StückProVE"<>0) then vOK # y;
if (BAG.Vpg.Wicklung+BAG.Vpg.Umverpackung+BAG.Vpg.Zwischenlage+BAG.Vpg.Unterlage+BAG.Vpg
  BAG.Vpg.VpgText3+BAG.Vpg.VpgText4+BAG.Vpg.VpgText5+BAG.Vpg.VpgText6<>'') then vOK # y;
if (BAG.Vpg.StehendYN or BAG.Vpg.LiegendYN) then vOK # y;
if (BAG.Vpg.Nettoabzug+"BAG.Vpg.Stapelhöhe"+BAG.Vpg.StapelHAbzug+
  BAG.Vpg.RingkgVon+BAG.Vpg.RingkgBis+BAG.Vpg.KgmmVon+BAG.Vpg.KgmmBis+
  BAG.Vpg.VEkgMax+BAG.Vpg.RechtwinkMax+BAG.Vpg.EbenheitMax+"BAG.Vpg.SäbeligMax"<>0.0) th
if (BAG.Vpg.Etikettentyp<>0) or (BAG.Vpg.Verwiegart<>0) then vOK # y;

if (vOK=false) then RETURN 0;

REPEAT
  BAG.VPG.Verpackung    # vX;
```

```
    vX # vX + 1;
    Erx # RekInsert(704,0,'AUTO');
    if (Erx=_rDeadLock) then RETURN 0;
  UNTIL (Erx=_rOK);

  RETURN BAG.Vpg.Verpackung;
end;


//=========================================================================
//  CopyAufToVpg
//        RES: VpgNummer
//=========================================================================
sub CopyAufToVpg(aBANr  : int) : int
local begin
  Erx   : int;
  vI    : int;
  vX    : int;
  vOK   : logic;
  vA    : alpha;
end;
begin

  vI # BAG.Nummer;
  BAG.Nummer # aBANr;
  Erx # RecLink(704,700,2,_recLast);   // letzte Verpackung holen
```

```
BAG.Nummer # vI;
if (Erx>_rLocked) then vX # 1
else vX # BAG.VPG.Verpackung + 1;

RecBufClear(704);
BAG.VPG.Nummer        # aBaNr;
BAG.Vpg.AbbindungL    # Auf.P.AbbindungL;
BAG.Vpg.AbbindungQ    # Auf.P.AbbindungQ;
BAG.Vpg.Zwischenlage  # Auf.P.Zwischenlage;
BAG.Vpg.Unterlage     # Auf.P.Unterlage;
BAG.Vpg.Umverpackung  # Auf.P.Umverpackung;
BAG.Vpg.Wicklung      # Auf.P.Wicklung;
BAG.Vpg.StehendYN     # Auf.P.StehendYN;
BAG.Vpg.LiegendYN     # Auf.P.LiegendYN;
BAG.Vpg.Nettoabzug    # Auf.P.Nettoabzug;
"BAG.Vpg.Stapelhöhe"  # "Auf.P.Stapelhöhe";
BAG.Vpg.StapelHAbzug  # Auf.P.StapelhAbzug;
BAG.Vpg.RingkgVon     # Auf.P.RingKgVon;
BAG.Vpg.RingkgBis     # Auf.P.RingKgBis;
BAG.Vpg.KgmmVon       # Auf.P.KgmmVon;
BAG.Vpg.KgmmBis       # Auf.P.KgmmBis;
"BAG.Vpg.StückProVE"  # "Auf.P.StückProVE";
BAG.Vpg.VEkgMax       # Auf.P.VEkgMax;
BAG.Vpg.RechtwinkMax  # Auf.P.RechtwinkMax;
BAG.Vpg.EbenheitMax   # Auf.P.EbenheitMax;
"BAG.Vpg.SäbeligMax"  # "Auf.P.SäbeligkeitMax";
```

```
"BAG.Vpg.SäbelProM"   # "Auf.P.SäbelProM";
BAG.Vpg.Etikettentyp  # Auf.P.Etikettentyp;
BAG.Vpg.Verwiegart    # Auf.P.Verwiegungsart;
BAG.Vpg.VpgText1      # Auf.P.VpgText1;
BAG.Vpg.VpgText2      # Auf.P.VpgText2;
BAG.Vpg.VpgText3      # Auf.P.VpgText3;
BAG.Vpg.VpgText4      # Auf.P.VpgText4;
BAG.Vpg.VpgText5      # Auf.P.VpgText5;
BAG.Vpg.VpgText6      # Auf.P.VpgText6;
BAG.Vpg.Skizzennr     # Auf.P.Skizzennummer;

if (BAG.Vpg.AbbindungL+BAG.Vpg.AbbindungQ+"BAG.Vpg.StückProVE"<>0) then vOK # y;
if (BAG.Vpg.Wicklung+BAG.Vpg.Umverpackung+BAG.Vpg.Zwischenlage+BAG.Vpg.Unterlage+BAG.Vpg
  BAG.Vpg.VpgText3+BAG.Vpg.VpgText4+BAG.Vpg.VpgText5+BAG.Vpg.VpgText6<>'') then vOK # y;
if (BAG.Vpg.StehendYN or BAG.Vpg.LiegendYN) then vOK # y;
if (BAG.Vpg.Nettoabzug+"BAG.Vpg.Stapelhöhe"+BAG.Vpg.StapelHAbzug+
  BAG.Vpg.RingkgVon+BAG.Vpg.RingkgBis+BAG.Vpg.KgmmVon+BAG.Vpg.KgmmBis+
  BAG.Vpg.VEkgMax+BAG.Vpg.RechtwinkMax+BAG.Vpg.EbenheitMax+"BAG.Vpg.SäbeligMax"<>0.0) th
if (BAG.Vpg.Etikettentyp<>0) or (BAG.Vpg.Verwiegart<>0) then vOK # y;

// 2022-09-01 AH
if (vOK) then vA # 'Y' else vA # 'N';
if (RunAFX('BAG.F.Data.CopyAufToVpg',vA)<>0) then begin
  if (AfxRes=_rNorec) then vOK # false;
end;
```

```
  if (vOK=false) then RETURN 0;

  REPEAT
    BAG.VPG.Verpackung    # vX;
    vX # vX + 1;
    Erx # RekInsert(704,0,'AUTO');
    if (Erx=_rDeadLock) then RETURN 0;
  UNTIL (Erx=_rOK);

  RETURN BAG.Vpg.Verpackung;
end;



//=========================================================================
//  AusKommission
//
//=========================================================================
sub AusKommission(
  aAufNr  : int;
  aAufPos : int;
  aRecId  : int;
);
local begin
  Erx   : int;
  vX    : int;
  vOK   : logic;
```

```
  vA    : alpha;
  vMan  : logic;
end;
begin

  if (StrCut(BAG.F.Kommission,1,1)='#') then RETURN;

  if (aAufNr=0) and (aRecId=0) then begin
    vMan # true;
    vA # StrCut(BAG.F.Kommission,1,1);
    vX # StrFind(BAG.F.Kommission,'/',0);
    if (vA>='0') and (vA<='9') and (vX<>0) then begin
      vA # Str_Token(BAG.F.Kommission,'/',1);
      aAufNr # CnvIa(va);
      vA # Str_Token(BAG.F.Kommission,'/',2);
      aAufPos # CnvIa(va);
    end;
  end;

  Erx # _rOK;
  If (Auf.P.Nummer<>aAufNr) or (Auf.P.Position<>aAufPos) or (aRecID<>0) then begin
    if (aRecId=0) then begin
      Auf.P.Nummer    # aAufNr;
      Auf.P.Position  # aAufPos;
      Erx # RecRead(401,1,0);
    end
```

```
  else begin
    Erx # RecRead(401,0,_RecId,aRecID);
  end;
end;

if (Erx<=_rLocked) and (Auf.P.Nummer<>Auf.Nummer) then begin
  Erx # RecLink(400,401,3,_RecFirst);   // Kopf holen
  if (Erx<=_rLocked) and (Auf.Vorgangstyp<>c_Auf) then Erx # _rNoRec;
end;
if (Erx>_rLocked) then begin
  BAG.F.Kommission        # '';
  BAG.F.Auftragsnummer    # 0;
  BAG.F.Auftragspos       # 0;
  "BAG.F.ReservFürKunde"  # 0;
  BAG.F.ReservierenYN     # y;
  RETURN;
end;

// 04.01.2016 AH:
if (Auf.LiefervertragYN) and (aRecID=0) then begin
  Msg(703009,'',0,0,0);
end;

BAG.F.Kommission        # AInt(Auf.P.Nummer) + '/' + AInt(Auf.P.Position);
BAG.F.Auftragsnummer    # Auf.P.Nummer;
BAG.F.Auftragspos       # Auf.P.Position;
```

```
  "BAG.F.ReservFürKunde"  # Auf.P.Kundennr;
  BAG.F.KundenArtNr       # Auf.P.KundenArtNr
  BAG.F.ReservierenYN     # y;
  "BAG.F.KostenträgerYN"  # y;
  BAG.F.PlanSchrottYN     # n;
  BAG.F.Warengruppe       # Auf.P.Warengruppe;
  BAG.F.Artikelnummer     # Auf.P.Artikelnr;
  if (BAG.F.Artikelnummer='') then begin
    BAG.F.Artikelnummer   # Auf.P.Strukturnr;
  end
  else begin  // 2023-02-03 AH
    Erx # RecLink(250,703,13,_recFirst);    // Artikel holen
    BAG.F.MEH             # ARt.MEH;
  end;

//  if (BAG.F.AutomatischYN=false) then begin
  if (BAG.F.Fertigung<999) then begin
    // Ausführung kopieren....
    Erx # RecLink(705,703,8,_recFirst);    // bisherige Ausführungen löschen
    WHILE (Erx<_rLocked) do begin
      Erx # RekDelete(705,0,'AUTO');
      if (Erx=_rDeadLock) then RETURN;
    END;

    BAG.F.AusfOben      # Auf.P.AusfOben;
    BAG.F.AusfUnten     # Auf.P.AusfUnten;
```

```
Erx # RecLink(402,401,11,_recFirst);    // Aufrrags-AF loopen...
WHILE (Erx<=_rLocked) do begin
  RecBufClear(705);
  BAG.AF.Nummer       # BAG.F.Nummer;
  BAG.AF.Position     # BAG.F.Position;
  BAG.AF.Fertigung    # BAG.F.Fertigung;
  BAG.AF.Seite        # Auf.AF.Seite;
  BAG.AF.lfdNr        # Auf.AF.lfdNr;
  BAG.AF.ObfNr        # Auf.AF.ObfNr;
  BAG.AF.BEzeichnung  # Auf.AF.Bezeichnung;
  BAG.AF.Zusatz       # Auf.AF.Zusatz;
  BAG.AF.Bemerkung    # Auf.AF.Bemerkung;
  "BAG.AF.Kürzel"     # "Auf.AF.Kürzel";
  Erx # RekInsert(705,0,'AUTO');
  if (Erx=_rDeadLock) then RETURN;
  Erx # RecLink(402,401,11,_recNext);
END;


// Feldübernahme abhängig vom Arbeitsgang
case (BAG.P.Aktion) of

  c_BAG_Spalt : begin
    BAG.F.Dicke       # Auf.P.Dicke;
    BAG.F.Dickentol   # Auf.P.Dickentol;
    BAG.F.Breite      # Auf.P.Breite;
```

```
  BAG.F.Breitentol  # Auf.P.Breitentol;
  "BAG.F.Gütenstufe"  # "Auf.P.Gütenstufe";
  "BAG.F.Güte"        # "Auf.P.Güte";
end;

c_BAG_Tafel : begin
  BAG.F.Dicke       # Auf.P.Dicke;
  BAG.F.Dickentol   # Auf.P.Dickentol;
  BAG.F.Breite      # Auf.P.Breite;
  BAG.F.Breitentol  # Auf.P.Breitentol;
  "BAG.F.Länge"     # "Auf.P.Länge";
  "BAG.F.Längentol" # "Auf.P.Längentol";
  "BAG.F.Gütenstufe"  # "Auf.P.Gütenstufe";
  "BAG.F.Güte"        # "Auf.P.Güte";
end;

c_BAG_AbCoil : begin
  BAG.F.Dicke       # Auf.P.Dicke;
  BAG.F.Dickentol   # Auf.P.Dickentol;
  BAG.F.Breite      # Auf.P.Breite;
  BAG.F.Breitentol  # Auf.P.Breitentol;
  "BAG.F.Länge"     # "Auf.P.Länge";
  "BAG.F.Längentol" # "Auf.P.Längentol";
  "BAG.F.Gütenstufe"  # "Auf.P.Gütenstufe";
  "BAG.F.Güte"        # "Auf.P.Güte";
end
```

```
c_BAG_Divers : begin
  BAG.F.Dicke       # Auf.P.Dicke;
  BAG.F.Dickentol   # Auf.P.Dickentol;
  BAG.F.Breite      # Auf.P.Breite;
  BAG.F.Breitentol  # Auf.P.Breitentol;
  "BAG.F.Länge"     # "Auf.P.Länge";
  "BAG.F.Längentol" # "Auf.P.Längentol";
  "BAG.F.Gütenstufe"  # "Auf.P.Gütenstufe";
  "BAG.F.Güte"        # "Auf.P.Güte";
end;

c_BAG_Walz, c_BAG_WalzSpulen : begin
  BAG.F.Dicke         # Auf.P.Dicke;
  BAG.F.Dickentol     # Auf.P.Dickentol;
  BAG.F.Breite        # Auf.P.Breite;       // 02.06.2022 AH, Proj. 2423/2
  BAG.F.Breitentol    # Auf.P.Breitentol;
  "BAG.F.Gütenstufe"  # "Auf.P.Gütenstufe";
  "BAG.F.Güte"        # "Auf.P.Güte";
end;

otherwise begin
  BAG.F.Dicke       # Auf.P.Dicke;
  BAG.F.Dickentol   # Auf.P.Dickentol;
  BAG.F.Breite      # Auf.P.Breite;
  BAG.F.Breitentol  # Auf.P.Breitentol;
```

```
      "BAG.F.Länge"     # "Auf.P.Länge";
      "BAG.F.Längentol" # "Auf.P.Längentol";
      "BAG.F.Gütenstufe"  # "Auf.P.Gütenstufe";
      "BAG.F.Güte"        # "Auf.P.Güte";
    end;

  end;

  SetRidRad('AusKommission'); // Rid / Rad setzen

  // Verpackung kopieren
  BAG.F.Verpackung # CopyAufToVpg(BAG.Nummer);

end;    // autoamtischYN?

// Etikettierung übernehmen
"BAG.F.Etk.Güte"  # "Auf.P.Etk.Güte";
BAG.F.Etk.Dicke   # Auf.P.Etk.Dicke;
BAG.F.Etk.Breite  # Auf.P.Etk.Breite;
"BAG.F.Etk.Länge" # "Auf.P.Etk.Länge";
BAG.F.Etk.Feld.1  # Auf.P.Etk.Feld.1;
BAG.F.Etk.Feld.2  # Auf.P.Etk.Feld.2;
BAG.F.Etk.Feld.3  # Auf.P.Etk.Feld.3;
BAG.F.Etk.Feld.4  # Auf.P.Etk.Feld.4;
BAG.F.Etk.Feld.5  # Auf.P.Etk.Feld.5;
```

```
  // bei manueller Eingabe der Kommisison alles mal refreshen...  14.03.2016 AH
  if (vMan) then begin
    ErrechnePlanmengen(y,y,y);
    gMDI->winUpdate(_WinUpdFld2Obj);
  end;

  // Anker...
  RunAFX('BAG.F.AusKommission','');

end;


//=========================================================================
//  AusKundenArtNr
//
//=========================================================================
sub AusKundenArtNr(opt aNurBefuellte : logic);
local begin
  Erx   : int;
  vX    : int;
  vOK   : logic;
end;
begin

  BAG.F.KundenArtNr # Adr.V.KundenArtNr
```

```
if (aNurBefuellte=false) or (Adr.V.Strukturnr<>'') then
  BAG.F.Artikelnummer # Adr.V.Strukturnr;
if (aNurBefuellte=false) or (Adr.V.Warengruppe<>0) then
  BAG.F.Warengruppe   # Adr.V.Warengruppe;
if (aNurBefuellte=false) or ("Adr.V.Güte"<>'') then
  "BAG.F.Güte"        # "Adr.V.Güte";
if (aNurBefuellte=false) or ("Adr.V.Gütenstufe"<>'') then
  "BAG.F.Gütenstufe"  # "Adr.V.Gütenstufe";

BAG.F.AusfOben      # Adr.V.AusfOben;
BAG.F.AusfUnten     # Adr.V.AusfUnten;
Erx # RecLink(106,105,1,_RecFirst);   // Ausführungen loopen
WHILE (Erx<=_rLocked) do begin

  RecBufClear(705);
  BAG.AF.Nummer       # BAG.F.Nummer;
  BAG.AF.Position     # BAG.F.Position;
  BAG.AF.Fertigung    # BAG.F.Fertigung;
  BAG.AF.Seite        # Adr.V.AF.Seite;
  BAG.AF.lfdNr        # Adr.V.AF.lfdNr;
  BAG.AF.ObfNr        # Adr.V.AF.ObfNr;
  BAG.AF.BEzeichnung  # Adr.V.AF.Bezeichnung;
  BAG.AF.Zusatz       # Adr.V.AF.Zusatz;
  BAG.AF.Bemerkung    # Adr.V.AF.Bemerkung;
  "BAG.AF.Kürzel"     # "Adr.V.AF.Kürzel";
```

```
  Erx # RekInsert(705,0,'AUTO');
  if (Erx=_rDeadLock) then RETURN;
  Erx # RecLink(106,105,1,_Recnext);
END;

if (aNurBefuellte=false) or (Adr.V.Dicke<>0.0) then
  BAG.F.Dicke         # Adr.V.Dicke;
if (aNurBefuellte=false) or (Adr.V.DickenTol<>'') then
  BAG.F.DickenTol     # Adr.V.DickenTol;
if (aNurBefuellte=false) or (Adr.V.Breite<>0.0) then
  BAG.F.Breite        # Adr.V.Breite;
if (aNurBefuellte=false) or (Adr.V.BreitenTol<>'') then
  BAG.F.BreitenTol    # Adr.V.BreitenTol;

if (BAG.P.Aktion=c_BAG_Tafel) or
  (BAG.P.Aktion=c_BAG_ABcoil) or
  (BAG.P.Aktion=c_BAG_Divers) then begin
  if (aNurBefuellte=false) or ("Adr.V.Länge"<>0.0) then
    "BAG.F.Länge"       # "Adr.V.Länge";
  if (aNurBefuellte=false) or ("Adr.V.LängenTol"<>'') then
    "BAG.F.LängenTol"   # "Adr.V.LängenTol";
end;
if ((RecLinkInfo(703, 702, 4, _recCount)=0) and (Mode = c_ModeNew)) or
  (BAG.F.Fertigung = 1) then begin
  BAG.F.RID           # Adr.V.RID;
  BAG.F.RIDmax        # Adr.V.RIDmax;
```

```
  end;
  BAG.F.RAD          # Adr.V.RAD;
  BAG.F.RADmax       # Adr.V.RADmax;

  // Verpackung kopieren
  BAG.F.Verpackung # CopyAdrToVpg(BAG.Nummer);

end;


//=========================================================================
//  ErrechnePlanmengen
//
//=========================================================================
sub ErrechnePlanmengen(
  aStk    : logic;
  aGew    : logic;
  aMenge  : logic;
  opt aNur1Input  : logic) : logic;
local begin
  Erx     : int;
  vX      : float;
  vGew    : float;
  vStk    : int;
  vBuf701 : int;
  vTeile  : int;
```

```
  vD      : float;
  vInMEH  : alpha;
  vInM    : float;
end;
begin
  // 2022-12-19 AH    F.MEH kommt aus EINSATZ
  if ("BAG.P.Typ.xIn-yOutYN"=false) then begin
    BAG.F.MEH # BAG.IO.MEH.Out;
  end;

  if (BAG.F.Fertigung=999) then RETURN true;

  // bei Fahren einmalig die Einsätze summieren...
  if (BAG.P.Aktion=c_BAG_Fahr09) OR (Bag.P.Aktion = c_BAG_Umlager) or (BAG.P.Aktion=c_BAG_Bere
    vBuf701 # RekSave(701);
    Erx # RecLink(701,703,3,_recFirst);   // Fertigung->Input loopen
    WHILE (Erx<=_rLockeD) do begin
      if (BAG.IO.BruderID=0) then begin
        vGew # vGew + BAG.IO.Plan.Out.GewN;
        vStk # vStk + BAG.IO.PLan.Out.Stk;
        if (vInMEH='') then vInmeh # BAG.IO.MEH.Out;
        if (vInMEH=BAG.IO.MEH.Out) then vinM # vinM + BAG.IO.Plan.Out.Meng
        else vinmeh # 'divers';
      end;
      Erx # RecLink(701,703,3,_recNext);
    END;
```

```
  //RecBufDestroy(vBuf701);
  RekRestore(vBuf701);
end;


if (aNur1Input=false) then begin
  // bei 1zu1 IMMER summieren : 26.06.2012 AI
  if (BAG.P.Aktion<>c_BAG_Fahr09) and (BAG.P.Aktion<>c_BAG_Umlager) and (BAG.P.Aktion<>c_BAG
    ("BAG.P.Typ.1In-1OutYN") then begin
    vBuf701 # RekSave(701);
    Erx # RecLink(701,702,2,_recFirst);   // Input loopen
    WHILE (Erx<=_rLockeD) do begin
      if (BAG.IO.BruderID=0) and (BAG.IO.Materialtyp<>c_IO_Beistell) then begin
        vGew # vGew + BAG.IO.Plan.Out.GewN;
        vStk # vStk + BAG.IO.PLan.Out.Stk;
        vTeile # vTEile + (BAG.IO.Plan.Out.Stk * (BAG.IO.Teilungen+1));
      end;
      Erx # RecLink(701,702,2,_recNext);
    END;
    RekRestore(vBuf701);
  end;
end
else begin
  vGew # BAG.IO.Plan.Out.GewN;
  vStk # BAG.IO.PLan.Out.Stk;
end;
```

```
//debugx('InStk:'+aint(vStk));

  // Stückzahl berechnen?
  if (aStk) then begin

    case BAG.P.Aktion of

      c_BAG_Saegen  : begin
        "BAG.F.Stückzahl" # (BAG.F.Streifenanzahl * BAG.IO.Plan.Out.Stk);
      end;



      c_BAG_Fahr09,
      c_BAG_Umlager,
      c_BAG_Umlager : begin
        "BAG.F.Stückzahl" # vStk;
      end;



      c_BAG_Spulen,
      c_BAG_abLaeng : begin
        if (BAG.F.Gewicht<>0.0) then begin
          // per Formel...
          "BAG.F.Stückzahl" # Lib_Berechnungen:Stk_aus_KgDBLWgrArt(BAG.F.Gewicht, BAG.IO.Dicke, BA
          // sonst per 3-Satz...
          if ("BAG.F.Stückzahl"=0) and ("BAG.IO.Plan.Out.Stk"<>0) and ("BAG.IO.Länge"<>0.0) then begin
```

```
      vX # BAG.IO.Plan.Out.GewN / cnvfi(BAG.IO.Plan.Out.Stk) / "BAG.IO.Länge";
      if (vX<>0.0) then begin
        "BAG.F.Stückzahl" # cnvif(BAG.F.Gewicht / vX / 1000.0);
        if (BAG.F.Gewicht > cnvfi("BAG.F.Stückzahl") * vX * 1000.0) then
          "BAG.F.Stückzahl" # "BAG.F.Stückzahl" + 1;
      end;
    end;
  end
  else if (BAG.F.Menge<>0.0) and ("BAG.F.Länge"<>0.0) then begin
    "BAG.F.Stückzahl" # cnvif(BAG.F.Menge*1000.0 / "BAG.F.Länge");
    if (BAG.F.Menge > cnvfi("BAG.F.Stückzahl") * "BAG.F.Länge" / 1000.0) then
      "BAG.F.Stückzahl" # "BAG.F.Stückzahl" + 1;
  end;
end;



c_BAG_Tafel : begin
  "BAG.F.Stückzahl" # 0;

  // oder besser über Artikel?
  if (BAG.F.Artikelnummer<>'') then begin
    Erx # RecLink(250,703,13,_recFirst);    // ARtikel holen
    if (Erx<=_rLocked) and ("Art.GewichtProStk"<>0.0) then begin
      vX # "Art.GewichtProStk";
      if (vX<>0.0) then begin
        vX # Trn(BAG.F.Gewicht / vX,0);
```

```
        if (vX<10000000.0) and (vX>=0.0) then
        "BAG.F.Stückzahl" # cnvif(vX);
      end;
    end;
  end;

  if ("BAG.F.Stückzahl"=0) and (BAG.F.Breite*"BAG.F.Länge"<>0.0) then begin
    if (BAG.F.Menge<>0.0) then begin
      "BAG.F.Stückzahl" # cnvif((BAG.F.Menge*1000000.0) div (BAG.F.Breite*"BAG.F.Länge"));
      if (BAG.F.Menge*1000000.0 % (BAG.F.Breite*"BAG.F.Länge")>0.0) then
        "BAG.F.Stückzahl" # "BAG.F.Stückzahl" + 1;
    end
    else if (BAG.F.Gewicht<>0.0) then begin
      RecLink(819,703,5,_recFirst);   // Warengruppe holen
      vX # Lib_Berechnungen:kg_aus_StkDBLDichte2(1000, BAG.IO.Dicke, BAG.F.Breite, "BAG.F.Läng
      if (vX<>0.0) then begin
        "BAG.F.Stückzahl" # cnvif(Trn(BAG.F.Gewicht / (vX / 1000.0) ,0));
      end;
    end;
  end;
end;


c_BAG_AbCoil : begin
  "BAG.F.Stückzahl" # 0;
  if (BAG.F.Breite*"BAG.F.Länge"<>0.0) then begin
```

```
        if (BAG.F.Menge<>0.0) then begin
          "BAG.F.Stückzahl" # cnvif((BAG.F.Menge*1000000.0) div (BAG.F.Breite*"BAG.F.Länge"));
        end
        else if (BAG.F.Gewicht<>0.0) then begin
          RecLink(819,703,5,_recFirst);   // Warengruppe holen
          vX # Lib_Berechnungen:kg_aus_StkDBLDichte2(1000, BAG.IO.Dicke, BAG.F.Breite, "BAG.F.Läng
          if (vX<>0.0) then begin
            "BAG.F.Stückzahl" # cnvif(Trn(BAG.F.Gewicht / (vX / 1000.0) ,0));
          end;
        end;
      end;
    end;



    c_BAG_SpaltSpulen, c_BAG_WalzSpulen,
    c_BAG_QTeil,  // 2023-08-04 AH
    c_BAG_Spalt : begin
      // 30.12.2009 AI: * Teilung
// 16.05.2022 AH        "BAG.F.Stückzahl" # BAG.F.Streifenanzahl * BAG.IO.Plan.Out.Stk * (BAG.IO.Teilung
      if (BAG.P.Aktion=c_BAG_Spalt) or (BAG.P.Aktion=c_BAG_QTeil) then begin    // 2022-09-12 AH    Pr
        "BAG.F.Stückzahl" # BAG.F.Streifenanzahl * BAG.IO.Plan.Out.Stk * (BAG.IO.Teilungen+1);
      end;
//debugx(aint("BAG.F.Stückzahl"));
    end;
```

```
      c_BAG_Schael, c_BAG_OBF, c_BAG_Kant, c_BAG_Check,c_BAG_Pack, c_BAG_Gluehen : begin
        "BAG.F.Stückzahl" # vStk;
      end;



      c_BAG_Walz : begin
        // 14.01.2019 AH:
//        "BAG.F.Stückzahl" # BAG.IO.Plan.Out.Stk * (BAG.IO.Teilungen+1);
        // 23.04.2020 AH:
        "BAG.F.Stückzahl" # vTeile;
       end;


    end;
  end;



  // Gewicht berechnen?
  if (aGew) then begin
    case BAG.P.Aktion of


      c_BAG_Saegen : begin
        // Dreisatz
        vX # 0.0;
        if (BAG.IO.Plan.Out.Stk<>0) and ("BAG.IO.Länge"<>0.0) then
          vX # BAG.IO.Plan.Out.GewN / cnvfi(BAG.IO.Plan.Out.Stk) / "BAG.IO.Länge";
        // oder besser über ARtikel?
```

```
    if (BAG.F.Artikelnummer<>'') then begin
      Erx # RecLink(250,703,13,_recFirst);    // ARtikel holen
      if (Erx<=_rLocked) and ("Art.GewichtProm"<>0.0) then begin
        vX # "Art.GewichtProm" / 10000.0;
      end;
    end;
//      RecLink(819,703,5,_recFirst);   // Warengruppe holen
    BAG.F.Gewicht # Rnd(vX * cnvfi(BAG.IO.Plan.Out.Stk * BAG.F.Streifenanzahl) * "BAG.F.Länge", Set
  end;



  c_BAG_Fahr09, c_BAG_Bereit,
  c_BAG_Umlager : begin
    BAG.F.Gewicht # Rnd(vGew, Set.Stellen.Gewicht);
  end;



  c_BAG_Spulen,
  c_BAG_abLaeng : begin
    // oer Formel...
    BAG.F.Gewicht # Lib_Berechnungen:KG_aus_StkDBLWgrArt("BAG.F.Stückzahl", BAG.IO.Dicke, BA
    // sonst 3-Satz...
    if (BAG.F.Gewicht=0.0) and ("BAG.IO.Plan.Out.Stk"<>0) and ("BAG.IO.Länge"<>0.0) then begin
      vX # BAG.IO.Plan.Out.GewN / cnvfi(BAG.IO.Plan.Out.Stk) / "BAG.IO.Länge";
      BAG.F.Gewicht # Rnd(vX * cnvfi("BAG.F.Stückzahl") * "BAG.F.Länge", Set.Stellen.Gewicht);
    end;
```

```
end;



c_BAG_Tafel : begin
  BAG.F.Gewicht # 0.0;
  // oder besser über Artikel?
  if (BAG.F.Artikelnummer<>'') then begin
    Erx # RecLink(250,703,13,_recFirst);    // ARtikel holen
    if (Erx<=_rLocked) and ("Art.GewichtProStk"<>0.0) then begin
      vX # "Art.GewichtProStk";
      "BAG.F.Gewicht" # Rnd(vX * cnvfi("BAG.F.Stückzahl") , Set.Stellen.Gewicht);
    end;
  end;

  if (BAG.F.Gewicht=0.0) then begin
    if (BAG.F.RAD=0.0) then begin   // 21.06.2022 AH war RAD
      RecLink(819,703,5,_recFirst);   // Warengruppe holen
      // 24.09.2014 am StÜCK ausrechnen
      BAG.F.Gewicht # Lib_Berechnungen:kg_aus_StkDBLDichte2("BAG.F.Stückzahl", BAG.IO.Dicke, B
    end
    else begin
    // 05.01.2022 AH: Ronde???
      vD # BAG.IO.Dicke;
      if (BAG.F.Dicke<>0.0) then vD # BAG.IO.Dicke;
      RecLink(819,703,5,_recFirst);   // Warengruppe holen
      BAG.F.Gewicht # Lib_Berechnungen:Kg_aus_StkDAdDichte2("BAG.F.Stückzahl", vD, BAG.F.RAD
```

```
    end;
  end;
end;



c_BAG_Abcoil : begin
  RecLink(819,703,5,_recFirst);   // Warengruppe holen
  BAG.F.Gewicht # Lib_Berechnungen:kg_aus_StkDBLDichte2("BAG.F.Stückzahl", BAG.IO.Dicke, BAG
end;



c_BAG_SpaltSpulen,
c_BAG_Spalt, c_BAG_QTeil : begin
  if (BAG.IO.Breite<>0.0) then
    BAG.F.Gewicht # RND(BAG.IO.Plan.Out.GewN / BAG.IO.Breite * (BAG.F.Breite * cnvfi(BAG.F.Strei
  else
    BAG.F.Gewicht # 0.0;
end;



c_BAG_WalzSpulen : begin
  BAG.F.Gewicht # RND(vGew, Set.Stellen.Gewicht);
end;



c_BAG_Schael, c_BAG_OBF, c_BAG_Kant, c_BAG_Check, c_BAG_Walz, c_BAG_Pack, c_BAG_Glu
```

```
      BAG.F.Gewicht # Rnd(vGew, Set.Stellen.Gewicht);
    end;


  end;


end;  // Gewicht



// Menge berechnen?
if (aMenge) then begin
  // 2022-12-19 AH
  if (BAG.P.Aktion=c_BAG_Saegen) and (BAG.F.MEH='m') then begin
    BAG.F.Menge # Rnd("BAG.F.Länge" / 10.0 * Cnvfi("BAG.F.Streifenanzahl") * cnvfi(BAG.IO.Plan.Out.S
  end;
  else if ((BAG.P.Aktion=c_BAG_Spulen) or (BAG.P.Aktion=c_BAG_abLaeng)) and (BAG.F.MEH='m') the
    BAG.F.Menge # Rnd("BAG.F.Länge" * Cnvfi("BAG.F.Stückzahl") / 1000.0,Set.Stellen.Menge);
  end;
end
else if ((BAG.P.Aktion=c_BAG_Tafel) or (BAG.P.Aktion=c_BAG_Abcoil)) and (BAG.F.MEH='qm') then be
  BAG.F.Menge # Rnd(BAG.F.Breite * Cnvfi("BAG.F.Stückzahl") * "BAG.F.Länge" / 1000000.0,Set.Steller
end;
else if ("BAG.P.Typ.1In-1OutYN") then begin
//debugx('KEY703 '+bag.f.meh+' <>'+vInMEH);
    if (StrCnv(BAG.F.MEH,_StrUpper)='KG') then  BAG.F.Menge # Rnd(vGew,Set.Stellen.Menge)
    else if (StrCnv(BAG.F.MEH,_StrUpper)='T') then   BAG.F.Menge # Rnd(vGew / 1000.0,Set.Stellen.Mer
    else if (StrCnv(BAG.F.MEH,_StrUpper)='STK') then BAG.F.Menge # CnvFI(vStk)
```

```
    else if (vINMEH=BAG.F.MEH) then BAG.F.Menge # vInM
    else BAG.F.Menge # Lib_Einheiten:WandleMEH(703, vStk , vGew, vInM, vInMEH, BAG.F.MEH);
  end;

  RETURN true;
end;


//=========================================================================
//  Splitten
//
//=========================================================================
sub Splitten(
  aBA   : int;
  aPos  : int;
  aFert : int;
) : logic;
local begin
  Erx     : int;
  vNr     : int;
  vBAG    : int;
  vTerm   : date;
  vVSBPos : int;

  vFPos   : int;
  vFFert  : int;
```

```
  vBuf703 : int;
  vBuf    : alpha(4000);
end;

begin

  // Position holen
  Erx # RecLink(702,703,2,_recFirst);
  if (Erx>_rLocked) then RETURN false;

  // mehrere Outputs?? -> FEHLER
  if (RecLinkInfo(701,703,4,_RecCount)>1) then RETURN false;

  // neuen Satz merken
  vBuf703 # RekSave(703);

  // Original holen
  BAG.F.Nummer    # aBA;
  BAG.F.Position  # aPos;
  BAG.F.Fertigung # aFert;
  Erx # RecRead(703,1,0);
  if (Erx>_rLocked) then begin
    RekRestore(vBuf703);
    RETURN false;
  end;
```

```
// nur unberührte Fertigungen lassen sich splitten
if (BAG.F.Kommission<>'') or (BAG.F.Fertig.Stk<>0) or
  (BAG.F.Fertig.Gew<>0.0) or (BAG.F.Fertig.Menge<>0.0) then begin
  RekRestore(vBuf703);
  RETURN false;
end;

// zuviel abgezogen? -> FEHLER
if (BAG.F.Gewicht - vBuf703->BAG.F.Gewicht<0.0) or
  (BAG.F.Menge - vBuf703->BAG.F.Menge<0.0) then begin
  RekRestore(vBuf703);
  RETURN false;
end;
// bei SPALTEN kann man die Breite bei Anzahl=1 manipulieren!
if (BAG.P.Aktion<>c_BAG_Spalt) and
  ("BAG.F.Stückzahl" - vBuf703->"BAG.F.Stückzahl"<0) then begin
  RekRestore(vBuf703);
  RETURN false;
end;
if (BAG.P.Aktion=c_BAG_Spalt) and (BAG.F.StreifenAnzahl>1) and
  ("BAG.F.Stückzahl" - vBuf703->"BAG.F.Stückzahl"<0) then begin
  RekRestore(vBuf703);
  RETURN false;
end;
```

```
// Auftrag holen
Auf.P.Nummer    # vBuf703->BAG.F.Auftragsnummer;
Auf.P.Position  # vBuf703->BAG.F.Auftragspos;
if (RecRead(401,1,0)>_rLocked) then begin
  RekRestore(vBuf703);
  RETURN false;
end;

// alles ok...
vBAG    # BAG.IO.Nummer;
vTerm   # BAG.P.Plan.StartDat;
if (BAG.P.Plan.EndDat<>0.0.0) then vTerm # BAG.P.Plan.EndDat;

TRANSON;

// bisherige Fertigung anpassen
Erx # RecRead(703,1,_recLock);
if (erx=_rOK) then begin
  if (BAG.P.Aktion=c_BAG_Spalt) and (BAG.F.StreifenAnzahl=1) then begin
    BAG.F.Breite          # BAG.F.Breite  - ( vBuf703->BAG.F.Breite * cnvfi(vBuf703->BAG.F.Streifenanzahl)
  end
  else begin
    BAG.F.StreifenAnzahl  # BAG.F.Streifenanzahl  - vBuf703->BAG.F.Streifenanzahl;
    "BAG.F.Stückzahl"     # "BAG.F.Stückzahl"     - vBuf703->"BAG.F.Stückzahl";
    BAG.F.Gewicht         # RND(BAG.F.Gewicht         - vBuf703->BAG.F.Gewicht, Set.Stellen.Gewicht);
    BAG.F.Menge           # RND(BAG.F.Menge           - vBuf703->BAG.F.Menge,Set.Stellen.Menge);
```

```
  end;
  Erx # Replace(_recUnlock,'AUTO');
end;
if (Erx<>_rOK) then begin
  RekRestore(vBuf703);
  TRANSBRK;
  RETURN false;
end;
// und durchrechnen...
if (UpdateOutput(703,n)=false) then begin
  RekRestore(vBuf703);
  TRANSBRK;
  RETURN false;
end;

// neue Fertigung anlegen
RekRestore(vBuf703);
BAG.F.Anlage.Datum      # Today;
BAG.F.Anlage.Zeit       # Now;
BAG.F.Anlage.User       # gUserName;
REPEAT
  Erx # Insert(0,'MAN');
  if (Erx<>_rOK) then BAG.F.Fertigung # BAG.F.Fertigung + 1;
UNTIL (Erx=_rOK);
vFPos   # BAG.F.Position;
vFFert  # BAG.F.Fertigung;
```

```
// und durchrechnen...
if (UpdateOutput(703,n)=false) then begin
  TRANSBRK;
  RETURN false;
end;



// neue VSB-Position generieren
RecBufClear(702);
BAG.P.Nummer        # vBAG;
BAG.P.Position      # BAG.P.Position + 1;
BAG.P.Aktion        # c_BAG_VSB;
BAG.P.Aktion2       # c_BAG_VSB;
BAG.P.Level         # 1;
BAG.P.Typ.VSBYN     # y;
BAG.P.ExternYN      # n;
BAG.P.Kommission    # AInt(BAG.F.Auftragsnummer)+'/'+AInt(BAG.F.Auftragspos);
BAG.P.Bezeichnung   # BAG.P.Aktion+' '+BAG.P.Kommission;
BAG.P.Auftragsnr    # BAG.F.Auftragsnummer;
BAG.P.Auftragspos   # BAG.F.Auftragspos;
BAG.P.Plan.StartDat # vTerm;
BAG.P.Plan.EndDat   # vTerm;
BAG.P.Anlage.Datum  # today;
BAG.P.Anlage.Zeit   # now;
BAG.P.Anlage.User   # gUsername;
```

```
REPEAT
  Erx # BA1_P_Data:Insert(0,'AUTO');
  if (Erx<>_rOK) then BAG.P.Position # BAG.P.Position + 1;
UNTIL (Erx=_rOK);
vVSBPos # BAG.P.Position;

BAG.F.Nummer    # vBAG;
BAG.F.Position  # vFPos;
BAG.F.Fertigung # vFFert;
RecRead(703,1,0);
// alle Outputs dieser Fertigung loopen
Erx # RecLink(701,703,4,_recFirst);
WHILE (Erx<=_rLocked) do begin
  if (BAG.IO.NachBAG=0) and (BAG.IO.NachPosition=0) then begin
    Erx # RecRead(701,1,_recLock);
    if (erx=_rOK) then begin
      BAG.IO.nachBAG      # vBAG;
      BAG.IO.nachPosition # vVSBPos;
      Erx # BA1_IO_Data:Replace(_recUnlock,'AUTO');
    end;
    if (Erx<>_rOK) then begin
      TRANSBRK;
      RETURN false;
    end;
  end;
```

```
    Erx # RecLink(701,703,4,_recNext);
  END;
  // und durchrechnen...
  if (UpdateOutput(702)=false) then begin
    TRANSBRK;
    RETURN false;
  end;

//  BA1_P_Data:RecalcThisLevel(var vBuf);
  Ba1_P_Data:UpdateSort();

  TRANSOFF;

  RETURN true
end;



//=========================================================================
//  Versand
//
//=========================================================================
sub Versand() : logic;
local begin
  Erx     : int;
  vBuf701 : handle;
```

```
  vBuf702 : handle;
  vBuf703 : handle;
  vBAG    : int;
  vVSDPos : int;
end;
begin

  if (BAG.F.Auftragsnummer=0) then RETURN false;
  Erx # RecLink(401,703,9,_recFirst);   // Auftragspos holen
  if (Erx>_rLocked) then RETURN false;
  Erx # RecLink(400,401,3,_recfirst);   // Auftragskopf holen
  if (Erx>_rLocked) then RETURN false;



  TRANSON;

  vBuf701 # RekSave(701);
  vBuf702 # RekSave(702);
  vBuf703 # RekSave(703);

  // BA-Position anlegen.....................
  RecBufClear(702);
  BAG.P.Nummer      # BAG.Nummer;
  BAG.P.Kosten.Wae  # 1;
  BAG.P.Kosten.PEH  # 1000;
  BAG.P.Kosten.MEH  # 'kg';
```

```
ArG.Aktion2 # c_BAG_Versand;
RecRead(828,1,0);
BAG.P.Aktion            # ArG.Aktion;
BAG.P.Aktion2           # ArG.Aktion2;
"BAG.P.Typ.1In-1OutYN"  # "ArG.Typ.1In-1OutYN";
"BAG.P.Typ.1In-yOutYN"  # "ArG.Typ.1In-yOutYN";
"BAG.P.Typ.xIn-yOutYN"  # "ArG.Typ.xIn-yOutYN";
"BAG.P.Typ.VSBYN"       # "ArG.Typ.VSBYN";
BAG.P.Bezeichnung       # ArG.Bezeichnung;

BAG.P.Zieladresse   # Auf.Lieferadresse;
BAG.P.Zielanschrift # Auf.Lieferanschrift;
Erx # RecLink(101,702,13,_recFirst);  // Zielanschrift holen
BAG.P.Zielstichwort # Adr.A.Stichwort;

if (BAG.P.Status='') and ("BAG.P.Löschmarker"='') then
  BA1_Data:SetStatus(c_BagStatus_Offen);
if ("BAG.P.Löschmarker"<>'') then
  BA1_Data:SetStatus(c_BagStatus_Fertig);

BAG.P.Position # 1;
REPEAT
  Erx # BA1_P_Data:Insert(0,'MAN');
  if (Erx<>_rOK) then BAG.P.Position # BAG.P.Position + 1;
UNTIl (Erx=_rOK);
```

```
// BA-Fertigung anlegen.....................
if ("BAG.P.Typ.1In-1OutYN") then begin
  RecBufClear(703);
  BAG.F.Nummer            # BAG.P.Nummer;
  BAG.F.Position          # BAG.P.Position;
  BAG.F.Fertigung         # 1;
  BAG.F.AutomatischYN     # y;
  "BAG.F.KostenträgerYN"  # y;
  BAG.F.MEH               # 'kg';
  Erx # Insert(0,'AUTO');
end;
RekRestore(vBuf703);


vBAG    # BAG.P.Nummer;
vVSDPos # BAG.P.Position;

// offene Outputs umbiegen.............................
Erx # Reclink(701,703,4,_RecFirst);   // Output loopen
WHILE (Erx<=_rLocked) do begin
  if (BAG.IO.NachBAG=0) and (BAG.IO.NachPosition=0) then begin
    Erx # RecRead(701,1,_recLock);
    if (erx=_rOK) then begin
      BAG.IO.nachBAG      # vBAG;
```

```
      BAG.IO.nachPosition # vVSDPos;
      Erx # BA1_IO_Data:Replace(_recUnlock,'AUTO');
    end;
    if (Erx<>_rOK) then begin
      TRANSBRK;
      RETURN false;
    end;
    if (UpdateOutput(701)<>y) then begin
      TRANSBRK;
      RETURN false;
    end;
  end;

  Erx # RecLink(701,703,4,_recNext);
END;

if (UpdateOutput(702)<>y) then begin
  TRANSBRK;
  RETURN false;
end;

RekRestore(vBuf701);
RekRestore(vBuf702);

TRANSOFF;
```

```
  Erx # RecLink(702,700,1,_recLast);

  BA1_P_Data:UpdateSort();

end;


//=========================================================================
//  Insert
//
//=========================================================================
SUB Insert(aLock : int; aGrund : alpha) : int;
local begin
  vA    : alpha;
  Erx   : int;
end;
begin

  if (BAG.F.Kommission='') then begin
    BAG.F.Auftragsnummer  # 0;
    BAG.F.Auftragspos     # 0;
    BAG.F.AuftragsFertig  # 0;
  end
  else begin
    vA # Str_Token(BAG.F.Kommission,'/',1);
    BAG.F.Auftragsnummer # CnvIa(vA);
```

```
    vA # Str_Token(BAG.F.Kommission,'/',2);
    BAG.F.AuftragsPos # CnviA(vA);
    vA # Str_Token(BAG.F.Kommission,'/',3);
    BAG.F.AuftragsFertig  # cnvia(vA);
  end;
//  BAG.F.Kommission          # aint(BAG.F.Auftragsnummer)  + '/' +ain(BAG.F.Auftragspos, _FmtNumNoGr

  BAG.F.Dickentol   # Lib_Berechnungen:Toleranzkorrektur("BAG.F.Dickentol",Set.Stellen.Dicke);
  vA # Str_Token(BAG.F.Dickentol,'/',1);
  BAG.F.Dickentol.Bis # BAG.F.Dicke + cnvfa(vA);
  vA # Str_Token(BAG.F.Dickentol,'/',2);
  BAG.F.Dickentol.Von # BAG.F.Dicke + cnvfa(vA);

  BAG.F.Breitentol  # Lib_Berechnungen:Toleranzkorrektur("BAG.F.Breitentol",Set.Stellen.Breite);
  vA # Str_Token(BAG.F.Breitentol,'/',1);
  BAG.F.Breitentol.Bis # BAG.F.Breite + cnvfa(vA);
  vA # Str_Token(BAG.F.Breitentol,'/',2);
  BAG.F.Breitentol.Von # BAG.F.Breite + cnvfa(vA);

  "BAG.F.Längentol" # Lib_Berechnungen:Toleranzkorrektur("BAG.F.Längentol","Set.Stellen.Länge");
  vA # Str_Token("BAG.F.Längentol",'/',1);
  "BAG.F.LÄngentol.Bis" # "BAG.F.Länge" + cnvfa(vA);
  vA # Str_Token("BAG.F.Längentol",'/',2);
  "BAG.F.Längentol.Von" # "BAG.F.Länge" + cnvfa(vA);

  // 16.12.2104:
```

```
  if (BAG.F.Menge=0.0) then begin
    if (BAG.F.MEH='kg') then BAG.F.Menge # BAG.F.Gewicht
    else if (BAG.F.MEH='t') then BAG.F.Menge # Rnd(BAG.F.Gewicht / 1000.0, Set.Stellen.Menge)
    else if (BAG.F.MEH='Stk') then BAG.F.Menge # cnvfi("BAG.F.Stückzahl");
  end;
  Erx # RekInsert(703,aLock,aGrund);

  Erg # Erx; // TODOERX
  RETURN Erx;
end;



//=========================================================================
//  Replace
//
//=========================================================================
SUB Replace(aLock : int; aGrund : alpha) : int;
local begin
  vA    : alpha;
  Erx   : int;
end;
begin

@ifdef LogFlow
debug('replace ID:'+aint(bag.io.id)+'  InSTK:'+aint(bag.io.plan.in.stk)+' outSTK:'+aint(bag.io.plan.out.stk));
@endif
```

```
if (RunAFX('BAG.F.Replace.Pre','')<0) then RETURN (AfxRes);


if (BAG.F.Kommission='') then begin
  BAG.F.Auftragsnummer  # 0;
  BAG.F.Auftragspos     # 0;
  BAG.F.AuftragsFertig  # 0;
end
else begin
  vA # Str_Token(BAG.F.Kommission,'/',1);
  BAG.F.Auftragsnummer # CnvIa(vA);
  vA # Str_Token(BAG.F.Kommission,'/',2);
  BAG.F.AuftragsPos # CnviA(vA);
  vA # Str_Token(BAG.F.Kommission,'/',3);
  BAG.F.AuftragsFertig  # cnvia(vA);
end;


BAG.F.Dickentol   # Lib_Berechnungen:Toleranzkorrektur("BAG.F.Dickentol",Set.Stellen.Dicke);
vA # Str_Token(BAG.F.Dickentol,'/',1);
BAG.F.Dickentol.Bis # BAG.F.Dicke + cnvfa(vA);
vA # Str_Token(BAG.F.Dickentol,'/',2);
BAG.F.Dickentol.Von # BAG.F.Dicke + cnvfa(vA);


BAG.F.Breitentol  # Lib_Berechnungen:Toleranzkorrektur("BAG.F.Breitentol",Set.Stellen.Breite);
vA # Str_Token(BAG.F.Breitentol,'/',1);
BAG.F.Breitentol.Bis # BAG.F.Breite + cnvfa(vA);
```

```
  vA # Str_Token(BAG.F.Breitentol,'/',2);
  BAG.F.Breitentol.Von # BAG.F.Breite + cnvfa(vA);

  "BAG.F.Längentol" # Lib_Berechnungen:Toleranzkorrektur("BAG.F.Längentol","Set.Stellen.Länge");
  vA # Str_Token("BAG.F.Längentol",'/',1);
  "BAG.F.LÄngentol.Bis" # "BAG.F.Länge" + cnvfa(vA);
  vA # Str_Token("BAG.F.Längentol",'/',2);
  "BAG.F.Längentol.Von" # "BAG.F.Länge" + cnvfa(vA);

  Erx # RekReplace(703,aLock,aGrund);

  Erg # Erx; // TODOERX
  RETURN Erx
end;


//=========================================================================
//  InsertFahrt
//
//=========================================================================
sub InsertFahrt() : int;
local begin
  Erx     : int;
  vBuf200 : int;
  vBuf703 : int;
end;
```

```
begin

@ifdef LogFlow
debug('insert ID:'+aint(bag.io.id)+'  InSTK:'+aint(bag.io.plan.in.stk)+' outSTK:'+aint(bag.io.plan.out.stk));
@endif

  if (BAG.IO.VonFertigung<>0) then begin
    vBuf703 # RecBufCreate(703);
    Erx # RecLink(vBuf703, 701,3,_recFirsT);  // Aus Fertigung holen
  end;

  RecBufClear(703);
  BAG.F.Nummer            # BAG.P.Nummer;
  BAG.F.Position          # BAG.P.Position;
  BAG.F.AutomatischYN     # y;
  "BAG.F.KostenträgerYN"  # y;
  BAG.F.MEH               # 'kg';
  BAG.F.Streifenanzahl    # 1;
  BAG.F.Fertigung         # 1;
  BAG.F.Gewicht           # BAG.IO.Plan.Out.GewN;
  "BAG.F.Stückzahl"       # BAG.IO.Plan.Out.Stk;
  BAG.F.Menge             # BAG.F.Gewicht;

  if (vBuf703<>0) then begin
    BAG.F.Kommission      # vBuf703->BAG.F.Kommission;
    BAG.F.Auftragsnummer  # vBuf703->BAG.F.Auftragsnummer;
```

```
    BAG.F.Auftragspos     # vBuf703->BAG.F.Auftragspos;
    BAG.F.AuftragsFertig  # vBuf703->BAG.F.AuftragsFertig;
    RecBufDestroy(vbuf703);
  end;
  // 02.10.2013 AH
  if (BAG.F.Auftragsnummer=0) then begin
//  else if (BAG.IO.Auftragsnr<>0) then begin
    BAG.F.Auftragsnummer  # BAG.IO.Auftragsnr;
    BAG.F.Auftragspos     # BAG.IO.Auftragspos;
    BAG.F.AuftragsFertig  # BAG.IO.Auftragsfert;
    BAG.F.Kommission      # aint(BAG.IO.Auftragsnr)+'/'+aint(BAG.IO.Auftragspos);
  end;
//  else if (BAG.IO.Materialnr<>0) then begin
  if (BAG.F.Auftragsnummer=0) and (BAG.IO.Materialnr<>0) then begin
    vBuf200 # RecBufCreate(200);
    Erx # Mat_Data:Read(BAG.IO.Materialnr, 0, vBuf200);
    if (Erx>=200) then begin
      BAG.F.Kommission      # vBuf200->Mat.Kommission;
      BAG.F.Auftragsnummer  # vBuf200->Mat.Auftragsnr;
      BAG.F.Auftragspos     # vBuf200->Mat.Auftragspos;
      BAG.F.AuftragsFertig  # 0;
    end;
    RecBufDestroy(vbuf200);
  end;

  // 02.10.2018 AH : Fix weil LFA aus Auftrag auch so arbeitet
```

```
  if (BAG.F.Auftragsnummer<>0) then begin
    Erx # Auf_Data:Read(BAG.F.Auftragsnummer, BAG.F.Auftragspos, n);
    if (Erx>=400) then begin
      "BAG.F.ReservFürKunde"  # Auf.P.Kundennr;
      BAG.F.ReservierenYN     # y;
    end;
  end;

  REPEAT
    Erx # RecRead(703,1,_recTest);
    if (Erx<=_rLocked) then inc(BAG.F.Fertigung);
  UNTIL (Erx>_rLocked);

  RETURN Insert(0,'AUTO');
end;



//=========================================================================
//  BA1_F_Data:PassendesMatAusAuftrag
//      Zeigt Passendes Material zum Einsatzcoil an
//=========================================================================
sub PassendesMatAusAuftrag()  : int
local begin
  Erx     : int;
  i       : int;
  vFeld   : alpha[100];
```

```
  vTyp    : int[100];
  vQInfo  : Alpha(1000);

  vGesamtbreite : float;
  vSumStr    : alpha;
end
begin


  // hier AFX Aufruf
  if (RunAFX('BA1.F.Info.Aufträge','')<>0) then
    RETURN 0;

  // Ersten Input zur Position lesen
  Erx # RecLink(701,702,2,_RecFirst);
  if (Erx <> _rOK) then
    RETURN 0;

  i # 1;
  vFeld[i] # 'Nummer';         vTyp[i]  # _TypeInt;    inc(i);
  vFeld[i] # 'Position';       vTyp[i]  # _TypeInt;    inc(i);
  vFeld[i] # 'Kunde';          vTyp[i]  # _TypeAlpha;  inc(i);
  vFeld[i] # 'Bestellt am';    vTyp[i]  # _TypeDate;   inc(i);
  vFeld[i] # 'Güte';           vTyp[i]  # _TypeAlpha;  inc(i);
  vFeld[i] # 'Oberfläche';     vTyp[i]  # _TypeAlpha;  inc(i);
  vFeld[i] # 'Dicke';          vTyp[i]  # _TypeFloat;  inc(i);
```

```
vFeld[i] # 'Breite';         vTyp[i]  # _TypeFloat;  inc(i);
vFeld[i] # 'Länge';          vTyp[i]  # _TypeFloat;  inc(i);
vFeld[i] # 'Stk';            vTyp[i]  # _TypeInt;    inc(i);
vFeld[i] # 'Bestellt kg';    vTyp[i]  # _TypeFloat;  vSumStr # vSumStr + Aint(i) + ','; inc(i);
vFeld[i] # 'Rest kg';        vTyp[i]  # _TypeFloat;  vSumStr # vSumStr + Aint(i) + ','; inc(i);
vFeld[i] # 'Termin';         vTyp[i]  # _TypeAlpha;  inc(i);

vQInfo  # '';
if ("BAG.IO.Güte" <> '') then
  vQInfo  # vQInfo + "BAG.IO.Güte" + '  ';

if ("BAG.IO.AusfOben" <> '') then
  vQInfo  # vQInfo + "BAG.IO.AusfOben" + '  ';

vQInfo  # vQInfo +  ANum("BAG.IO.Dicke",Set.Stellen.Dicke)          + ' x '     +
                    ANum("BAG.IO.Breite",Set.Stellen.Breite)        + ' mm   '  +
                    ANum("BAG.IO.Plan.In.Menge",Set.Stellen.Gewicht)  + ' kg';

// Alle Fertigungen der Position summieren
FOR   Erx # RecLink(703,702,4,_RecFirst);
LOOP  Erx # RecLink(703,702,4,_RecNext);
WHILE Erx = _rOK DO
  vGesamtbreite # vGesamtbreite + (BAG.F.Breite * CnvFI(BAG.F.Streifenanzahl));


if (BAG.IO.Breite <= vGesamtbreite) then
```

```
    return 0;

  vQInfo # vQInfo + '   Restbreite: ' + ANum(BAG.IO.Breite - vGesamtbreite,Set.Stellen.Breite) + ' mm';
  vQInfo # StrAdj(vQInfo, _StrBegin);

  Lib_QuickInfo:Show('passende Auftragspositionen für: '+ vQInfo, var vFeld,var vTyp, here+':PassendesM
end;



//=========================================================================
// PassendesMatAusAuftrag_Data
//      Ermittelt die darzustellenden Datensätze
//=========================================================================
sub PassendesMatAusAuftrag_Data(var aSortTreeHandle : int;)
local begin
  Erx         : int;
  vPrg        : int;
  vQ          : alpha(4096);
  vSel        : int;
  vSelName    : alpha;
  vSelCnt     : int;
  vCurrent    : int;
  vSortKey    : alpha;

  vGesamtbreite : float;
  vRestBreite   : float;
```

```
end;
begin

  vPrg # Lib_Progress:Init('Datenermittlung');

  // Alle Fertigungen der Position summieren
  FOR   Erx # RecLink(703,702,4,_RecFirst);
  LOOP  Erx # RecLink(703,702,4,_RecNext);
  WHILE Erx = _rOK DO BEGIN
    vGesamtbreite # vGesamtbreite + (BAG.F.Breite * CnvFI(BAG.F.Streifenanzahl));
  END;

  vRestBreite # BAG.IO.Breite - vGesamtbreite;

  //
  vQ # '';
  Lib_Sel:QAlpha(   var vQ, '"Auf.P.Löschmarker"', '=', '' );
  Lib_Sel:QInt(     var vQ, '"Auf.P.Nummer"',      '<',  1000000000 );
  Lib_Sel:QAlpha(   var vQ, '"Auf.P.Güte"',        '=*', "BAG.IO.Güte" );
  Lib_Sel:QVonBisF( var vQ, '"Auf.P.Dicke"',       "BAG.IO.Dicke", "BAG.IO.Dicke");
  Lib_Sel:QVonBisF( var vQ, '"Auf.P.Breite"',      0.0,            vRestBreite);
  Lib_Sel:QFloat(   var vQ, '"Auf.P.Prd.Rest.Gew"', '>',           0.0);

  vSel # SelCreate( 401, 1 );
  Erx # vSel->SelDefQuery('', vQ);
  if (Erx <> 0) then
```

```
  Lib_Sel:QError(vSel);


vSelName # Lib_Sel:SaveRun( var vSel, 0);
vSelCnt  # vSel->SelInfo(_SelCount);


FOR   Erx # RecRead(401,vSel,_RecFirst)
LOOP  Erx # RecRead(401,vSel,_RecNext)
WHILE Erx <= _rLocked DO BEGIN
  inc(vCurrent);
  vPrg->Lib_Progress:SetLabel('Sortierung ' + Aint(vCurrent) + '/' + Aint(vSelCnt))
  if (vPrg->Lib_Progress:Step() = false) then begin
    break;
  end;

  // Sortierungsschlüssel definieren
  vSortKey # "Auf.P.Güte" +
             cnvAF(Auf.P.Breite,_FmtNumLeadZero|_fmtNumNoGroup,0,2,10)+
             Auf.P.KundenSw;

  Sort_ItemAdd(aSortTreeHandle,vSortKey,401,RecInfo(401,_RecId));
END;

// Beenden
vSel->SelClose();
SelDelete(401, vSelName);
```

```
  vPrg->Lib_Progress:Term();
end;



//=========================================================================
// PassendesMatAusAuftrag_Data_Pos
//      Weist dem Zeilenarray die Entsprechenden Daten zu
//=========================================================================
sub PassendesMatAusAuftrag_Data_Pos(aSortItem : int; var aRecord : alpha[])
local begin
  i : int;
end;
begin
  RecRead(cnvIA(aSortItem->spCustom), 0, 0, aSortItem->spID); // Datensatz holen
  RecLink(400,401,3,0); // Auftragskopf lesen

  i # 1;
  if (cnvIA(aSortItem->spCustom) = 401) then begin
    aRecord[i] # Aint(Auf.P.Nummer);                        inc(i);
    aRecord[i] # Aint(Auf.P.Position);                      inc(i);
    aRecord[i] # Auf.P.KundenSW;                            inc(i);
    aRecord[i] # CnvAd(Auf.Datum);                          inc(i);
    aRecord[i] # "Auf.P.Güte";                              inc(i);
    aRecord[i] # "Auf.P.AusfOben";                          inc(i);
    aRecord[i] # ANum(Auf.P.Dicke,Set.Stellen.Dicke);       inc(i);
    aRecord[i] # ANum(Auf.P.Breite, Set.Stellen.Breite);    inc(i);
```

```
    aRecord[i] # ANum("Auf.P.Länge", "Set.Stellen.Länge");      inc(i);
    aRecord[i] # AInt("Auf.P.Stückzahl");                      inc(i);
    aRecord[i] # ANum(Auf.P.Gewicht,Set.Stellen.Gewicht);      inc(i);
    aRecord[i] # ANum(Auf.P.Prd.Rest.Gew,Set.Stellen.Gewicht); inc(i);
    aRecord[i] # CnvAd(Auf.P.TerminZusage);                    inc(i);
  end;

end;



//=========================================================================
// PassendesMatAusAuftrag_Data_Sort
//      Weist dem Zeilenarray die Entsprechenden Daten zu
//=========================================================================
sub PassendesMatAusAuftrag_Data_Sort(aRowIndex : int) : alpha
begin
  case (aRowIndex) of
    1 : begin RETURN Lib_Strings:IntForSort(  Auf.P.Nummer);         end;
    2 : begin RETURN Lib_Strings:IntForSort(  Auf.P.Nummer)  + '/' +
                  Lib_Strings:IntForSort(  Auf.P.Position);       end;
    3 : begin RETURN                         Auf.P.KundenSW;      end;
    4 : begin RETURN Lib_Strings:DateForSort( Auf.Datum);         end;
    5 : begin RETURN                         "Auf.P.Güte";        end;
    6 : begin RETURN                         "Auf.P.AusfOben";    end;
    7 : begin RETURN Lib_Strings:NumForSort(  Auf.P.Dicke);       end;
    8 : begin RETURN Lib_Strings:NumForSort(  Auf.P.Breite);      end;
```

```
    9 : begin RETURN Lib_Strings:NumForSort( "Auf.P.Länge");          end;
   10 : begin RETURN Lib_Strings:IntForSort( "Auf.P.Stückzahl");      end;
   11 : begin RETURN Lib_Strings:NumForSort(  Auf.P.Gewicht);         end;
   12 : begin RETURN Lib_Strings:NumForSort(  Auf.P.Prd.Rest.Gew);    end;
   13 : begin RETURN Lib_Strings:DateForSort( Auf.P.TerminZusage);    end;
  end;
end;



//=========================================================================
// FertNachPos
//
//=========================================================================
sub FertNachPos(
  aBAG1 : int;
  aPos1 : int;
  aFert : int;
  aBAG2 : int;
  aPos2 : int;
) : logic;
local begin
  Erx             : int;
  v701            : int;
  v702            : int;
  v703,v703b      : int;
  vKGMM1          : float;
```

```
  vKGMM2          : float;
  vKGMM_Kaputt    : logic;
end;
begin

  APPOFF();

  v701 # RekSave(701);
  v702 # RekSave(702);
  v703 # RekSave(703);

  BAG.F.Nummer    # aBAG1;
  BAG.F.Position  # aPos1;
  BAG.F.Fertigung # aFert;
  Erx # RecRead(703,1,0);
  if (Erx>_rLocked) then begin
    RekRestore(v701);
    RekRestore(v702);
    RekRestore(v703);
    RETURN false;
  end;

  TRANSON;

  v703b # RekSave(703);
  FOR Erx # RecLink(701, 703, 4,_recFirst)    // Output zu Fertigung loopen...
```

```
LOOP Erx # RecLink(701, 703, 4,_recNext)
WHILE (Erx<=_rLocked) do begin
  if (BAG.IO.Materialtyp=c_IO_BAG) and (BAG.IO.BruderID=0) and (BAG.IO.NachPosition=0) then begin
    if (BA1_IO_Data:OutputNachPos(BAG.IO.Nummer, BAG.IO.ID, aBAG2, aPos2)=false) then begin
      APPON();
      TRANSBRK;
      RecBufDestroy(v703b);
      RekRestore(v701);
      RekRestore(v702);
      RekRestore(v703);
      RETURN false;
    end;
    RecBufCopy(v703b,703);
  end;
END;
RecBufDestroy(v703b);

TRANSOFF;

RekRestore(v701);
RekRestore(v702);
RekRestore(v703);

APPON();

RETURN true;
```

```
end;



//=========================================================================
// FertNachQTiel
//
//=========================================================================
sub FertNachQTeil(
  aBAG1 : int;
  aPos1 : int;
  aFert : int;
  aTlg  : int;
) : logic;
local begin
  Erx             : int;
  v701            : int;
  v702            : int;
  v703,v703b      : int;
  vKGMM1          : float;
  vKGMM2          : float;
  vKGMM_Kaputt    : logic;
  vBAG2           : int;
  vPos2           : int;
end;
begin
```

```
APPOFF();


v701 # RekSave(701);
v702 # RekSave(702);
v703 # RekSave(703);


BAG.F.Nummer    # aBAG1;
BAG.F.Position  # aPos1;
BAG.F.Fertigung # aFert;
Erx # RecRead(703,1,0);
if (Erx>_rLocked) then begin
  APPON();
  RekRestore(v701);
  RekRestore(v702);
  RekRestore(v703);
  RETURN false;
end;


TRANSON;


// neue Pos anlegen:
FOR BAG.P.Position # BAG.P.Position;
LOOP inc(BAG.P.Position)
WHILE (RecRead(702,1,_recTest)<=_rOK) do begin
END;
vBAG2 # BAG.P.Nummer;
```

```
vPos2 # BAG.P.Position;


RecBufClear(702);
BAG.P.Nummer    # vBAG2;
BAG.P.Position  # vPos2;


BAG.P.Aktion2          # c_bag_QTeil;
BAG.P.ExternYN         # n;
BAG.P.Kosten.Wae       # 1;
BAG.P.Kosten.PEH       # 1000;
BAG.P.Kosten.MEH       # 'kg';
Erx # RecLink(828,702,8,_recFirst);   // Arbeitsgang holen
if (Erx>_rLocked) then begin
  APPON();
  RekRestore(v701);
  RekRestore(v702);
  RekRestore(v703);
  RETURN false;
end;
BAG.P.Aktion           # ArG.Aktion;
BAG.P.Aktion2          # ArG.Aktion2;
"BAG.P.Typ.1In-1OutYN"  # "ArG.Typ.1In-1OutYN";
"BAG.P.Typ.1In-yOutYN"  # "ArG.Typ.1In-yOutYN";
"BAG.P.Typ.xIn-yOutYN"  # "ArG.Typ.xIn-yOutYN";
"BAG.P.Typ.VSBYN"       # "ArG.Typ.VSBYN";
BAG.P.Bezeichnung       # ArG.Bezeichnung
```

```
BA1_Data:SetStatus(c_BagStatus_Offen);
if (BA1_P_Data:Insert(0,'MAN')<>_rOK) then begin
  APPON();
  RekRestore(v701);
  RekRestore(v702);
  RekRestore(v703);
  RETURN false;
end;
vBAG2 # BAG.P.Nummer;
vPos2 # BAG.P.Position;



v703b # RekSave(703);

// Fertigung anlegen
RecBufClear(703);
BAG.F.Nummer            # vBAG2;
BAG.F.Position          # vPos2;
BAG.F.Fertigung         # 1;
BAG.F.AutomatischYN     # n;
BAG.F.MEH               # 'kg';
BAG.F.Warengruppe       # 0;
"BAG.F.KostenträgerYN"  # BAG.F.Kommission<>'';
BAG.F.Streifenanzahl    # 1;
if (BAG.F.Artikelnummer<>'') then begin
  Erx # RekLink(250,703,13,_recfirst);   // Artikel holen
```

```
  if (Erx<=_rLocked) then begin
    BAG.F.Warengruppe # Art.Warengruppe;
    BAG.F.MEH         # Art.MEH;
  end;
end;
BAG.F.Anlage.Datum  # Today;
BAG.F.Anlage.Zeit   # Now;
BAG.F.Anlage.User   # gUserName;
Erx # Insert(0,'AUTO');
if (erx<>_rOK) then begin
  APPON();
  TRANSBRK;
  RecBufDestroy(v703b);
  RekRestore(v701);
  RekRestore(v702);
  RekRestore(v703);
  RETURN false;
end;

RecBufCopy(v703b,703);

FOR Erx # RecLink(701, 703, 4,_recFirst)    // Output zu Fertigung loopen...
LOOP Erx # RecLink(701, 703, 4,_recNext)
WHILE (Erx<=_rLocked) do begin
  if (BAG.IO.Materialtyp=c_IO_BAG) and (BAG.IO.BruderID=0) and (BAG.IO.NachPosition=0) then begin
    if (BA1_IO_Data:OutputNachPos(BAG.IO.Nummer, BAG.IO.ID, vBAG2, vPos2, aTlg)=false) then begin
```

```
        APPON();
        TRANSBRK;
        RecBufDestroy(v703b);
        RekRestore(v701);
        RekRestore(v702);
        RekRestore(v703);
        RETURN false;
      end;
      RecBufCopy(v703b,703);
    end;
  END;
  RecBufDestroy(v703b);



  TRANSOFF;


  RekRestore(v701);
  RekRestore(v702);
  RekRestore(v703);


  APPON();


  RETURN true;
end;
```

```
//=========================================================================
//  BindeAnVorlage +ERR
//
//=========================================================================
Sub BindeAnVorlage(
  aBAG        : int;
  aPos        : int;
  aFert       : int;
  aVorlageBAG : int;
  aAufNr      : int;
  aAufPos     : int;
) : logic;
local begin
  Erx           : int;
  vAnschlussID  : int;
  vFirst        : logic;
  vNachPos      : int;
  v701, v703    : int;
  vOK           : logic;
  vI            : int;
end;
begin

  if (BAG.Nummer<>aBAG) then begin
    BAG.Nummer # aBAG;
    Erx # RecRead(700,1,0);
```

```
  if (Erx>_rLocked) then RETURN false;
end;



TRANSON;

// Vorlage kopieren...
vAnschlussID # BA1_P_Data:ImportBA(aBAG, aVorlageBAG, aAufnr, aAufPos, true);
if (vAnschlussID=0) then begin
  TRANSBRK;
  Error(999999,ThisLine);
  RETURN false;
end;

BAG.IO.Nummer # aBAG
BAG.IO.ID     # vAnschlussID;
Erx # RecRead(701,1,0);
vNachPos # BAG.IO.NachPosition;

BAG.P.Nummer    # BAG.Nummer;
BAG.P.Position  # vNachPos;
Erx # RecRead(702,1,0);

BAG.F.Nummer    # aBAG;
BAG.F.Position  # aPos;
BAG.F.Fertigung # aFert;
```

```
  Erx # RecRead(703,1,0);
  if (Erx>_rLocked) then begin
    TRANSBRK;
    Error(999999,ThisLine);
    RETURN false;
  end;

  v703 # RekSave(703);
  vFirst # y;
  // Outputs loopen...
  FOR Erx # RecLink(701,703,4,_recFirst)
  LOOP Erx # RecLink(701,703,4,_recNext)
  WHILE (Erx<=_rLocked) do begin

    if (vFirst) then begin
      vFirst # false;

      vI # BAG.IO.ID;
      v701 # RekSave(701);
      vOK # BA1_IO_I_Data:TheorieWirdID(vAnschlussID, BAG.IO.ID);
      if (vOK) then begin
        vAnschlussID # vI;  // bisheriger Theoretischer Input wurde ja gerade ersetzt druch BAG.IO.ID (vI)
        RecBufCopy(v701, 701);
        // Output aktualisieren
//debugx('updateoutput KEY701 KEY702');
        if (BA1_F_Data:UpdateOutput(701,n)=false) then begin
```

```
      RekRestore(v701);
      RekRestore(v703);
      TRANSBRK;
      Error(999999,ThisLine);
      RETURN false;
    end;
  end;
  RekRestore(v701);
end
else begin
  Erx # RecRead(701,1,_recLock);
  if (erx=_rOK) then begin
    BAG.IO.NachBAG      # BAG.P.Nummer;
    BAG.IO.NachPosition # vNachPos;
    Erx # BA1_IO_Data:Replace(_recUnlock,'AUTO');
  end;
  if (Erx<>_rOK) then begin
    RekRestore(v703);
    TRANSBRK;
    Error(999999,ThisLine);
    RETURN false;
  end;
  // Output aktualisieren
  if (BA1_F_Data:UpdateOutput(701,n)=false) then begin
    RekRestore(v703);
    TRANSBRK;
```

```
      Error(999999,ThisLine);
      RETURN false;
    end;

    vOK # BA1_IO_I_Data:KlonenVon(vAnschlussID, true);
  end;
  if (vOK=false) then begin
    RekRestore(v703);
    TRANSBRK;
    Error(999999,ThisLine);
    RETURN false;
  end;
  RecBufCopy(v703, 703);
END;

RekRestore(v703);

TRANSOFF;

// alle Fertigungen neu errechnen
//debug('Berechne neu KEY702');
BA1_P_Data:ErrechnePlanmengen();


RETURN true;
```

```
end;


//=========================================================================
//  BelegeKommissionsDaten
//=========================================================================
sub BelegeKommisisonsDaten(
  aString : alpha;
  aAufNr  : int;
  aAufPos : int);
local begin
  Erx     : int;
  vAlles  : logic;
end;
begin

  if (RunAFX('BAG.F.BelegeKomDaten',aString+'|'+aint(aAufNr)+'|'+aint(aAufPos))<>0) then
    RETURN;

  if (StrCut(aString,1,1)<>'#') then RETURN;
  aString # StrCnv(aString, _Strupper);

  Erx # Auf_Data:Read(aAufNr, aAufPos, false);
  if (Erx<400) then begin
    BAG.F.Kommission # '';
    RETURN;
```

```
end;

vAlles # (StrCut(aString,2,1)='#');

BAG.F.Auftragsnummer  # aAufNr;
BAG.F.Auftragspos     # aAufPos;
BAG.F.Kommission      # AInt(aAufNr) + '/' + AInt(aAufPos);
BAG.F.KundenArtNr     # Auf.P.KundenArtNr;

// Verpackung 13.12.2019
if (StrFind(aString,'V',0)>0) or (vAlles) then begin
  BAG.F.Verpackung # CopyAufToVpg(BAG.F.Nummer);
end;
// Menge
if (StrFind(aString,'M',0)>0) or (vAlles) then begin
  BAG.F.Gewicht         # Auf.P.Gewicht;
end;
if (StrFind(aString,'A',0)>0) or (vAlles) then begin
  BAG.F.RAD         # Auf.P.RAD;
  BAG.F.RADmax      # Auf.P.RADmax;
end;
if (StrFind(aString,'B',0)>0) or (vAlles) then begin
  BAG.F.Breite      # Auf.P.Breite;
  BAG.F.BreitenTol  # Auf.P.BreitenTol;
end;
if (StrFind(aString,'D',0)>0) or (vAlles) then begin
```

```
    BAG.F.Dicke       # Auf.P.Dicke;
    BAG.F.DickenTol   # Auf.P.DickenTol;
  end;
  if (StrFind(aString,'G',0)>0) or (vAlles) then begin
    "BAG.F.Güte"        # "Auf.P.Güte";
    "BAG.F.Gütenstufe"  # "Auf.P.Gütenstufe";
  end;
  if (StrFind(aString,'I',0)>0) or (vAlles) then begin
    BAG.F.RID         # Auf.P.RID;
    BAG.F.RIDmax      # Auf.P.RIDmax;
  end;
  if (StrFind(aString,'L',0)>0) or (vAlles) then begin
    "BAG.F.Länge"     # "Auf.P.Länge";
    "BAG.F.LängenTol" # "Auf.P.LängenTol";
  end;
  // 24.07.2017 AH:
  if (StrFind(aString,'S',0)>0) or (vAlles) then begin
    "BAG.F.Stückzahl" # "Auf.P.Stückzahl";
  end;
  if (StrFind(aString,'W',0)>0) or (vAlles) then begin
    BAG.F.Warengruppe # Auf.P.Warengruppe;
  end;

end;
```

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