

@A+

//===== Business-Control =====

//

// Prozedur Con\_Cata

// OHNE E\_R\_G

// Info: Enthält Funktionen für die Controllingfunktionen

//

//

// 18.03.2010 ST Erstellung der Prozedur

// 15.06.2010 ST Umstellung Generierung der Vorgaben in Sollwerte

// 27.06.2011 TM ^Fortschrittsbalken bei Generierung von Vorgabe- und Istdaten

// 13.07.2011 ST Fehlerbehebung Recalc bzgl. Artikelnummer / Güte

// 03.08.2011 ST Auftragsart hinzugefügt

// 26.04.2012 ST Fehlerbehebung Recalc, Fortschrittsbalken hat nicht das Richtige angezeigt

// 12.08.2014 AH Neu: "Read"

// 17.08.2016 AH Erlöskorrektur beachten

// 31.10.2019 AH auch Auftrags, Bestell und Angebotserfassung

// 05.04.2022 AH ERX

//

// Subprozeduren

// SUB Read(aTyp : alpha; aJahr : int; aAdr : int; aVert : int; aAufArt : int; aWGr : int; aAGr : int; aArtNr : a

// SUB Recalc()

// SUB Generieren()

// SUB \_inMarkInt(aFile : int; aFld1 : alpha; aVal : int) : logic

// SUB \_inMarkAlpha(aFile : int; aFld1 : alpha; aVal : alpha) : logic

// SUB \_insertCon()

```
// SUB _genLoopStat(aJahr : int;aTyp : int; aKndAlle : logic; aKndMark: logic;
//
//          aVerAlle: logic; aVerMark : logic;aWgrAlle : logic; aWgrMark : logic;
//
//          aArgAlle : logic; aArgMark : logic; aArtAlle : logic;aArtMark : logic;
//
//          aGteAlle : logic;aGteMark : logic; )

// SUB _genLoopAll(aJahr : int;aTyp : int; aKndAlle : logic; aKndMark: logic;
//
//          aVerAlle: logic; aVerMark : logic;aWgrAlle : logic; aWgrMark : logic;
//
//          aArgAlle : logic; aArgMark : logic; aArtAlle : logic;aArtMark : logic;
//
//          aGteAlle : logic;aGteMark : logic; )

// SUB GenerierenVorgaben()

// SUB AbweichungIstSoll(aDat : date; aWas : alpha; aTyp : alpha; opt aAdr : int; opt aVer : int; opt aArt :
//
//=====
```

@I:Def\_Global

define begin

```
cKunden      : 100

cVetreter    : 110

cWarengruppen : 819

cKostenstellen : 846

cArtikelgruppen : 826

cArtikel      : 250
```

end;

declare Recalc(aAlle : logic);

LOCAL begin

```
gvJahr   : int;    // Jahr des Kennzeichens

vProgress : handle; // Handle für Fortschrittsbalken

end;
```

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

```
// Read
```

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

```
SUB Read(
```

```
    aTyp   : alpha;
```

```
    aJahr  : int;
```

```
    aAdr   : int;
```

```
    aVert  : int;
```

```
    aAufArt : int;
```

```
    aWGr   : int;
```

```
    aAGr   : int;
```

```
    aArtNr : alpha) : int
```

```
local begin
```

```
    Erx : int;
```

```
end;
```

```
begin
```

```
    RecBufClear(950);
```

```
    Con.Type      # aTyp;
```

```
    Con.Jahr      # aJahr;
```

```
    Con.Adressnummer # aAdr;
```

```
    Con.Vertreternr # aVert;
```

```

Con.Warengruppe # aWgr;

Con.Artikelgruppe # aAgr;

Con.Artikelnummer # aArtNr;

Con.Auftragsart # aAufArt;

Erx # RecRead(950,1,0);

if (Erx>_rLocked) then RecBufClear(950);

RETURN Erx;

end;

```

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

```
// _inMarkWord(...)
```

```
// Prüft die Markierten Datensätze, ob ein Wert darin vorkommt
```

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

```
sub _inMarkWord(aFile : int; aFld1 : alpha; aVal : word) : logic
```

```
local begin
```

```
    // Markierungsprüfung
```

```
    vFound      : logic;
```

```
    vMarkItem   : handle;
```

```
    vMarkMFile,
```

```
    vMarkMID    : int;
```

```
    vErx        : int;
```

```
end
```

```
begin
```

```
    vFound # false;
```

```

// Satz in Markierung enthalten?

FOR vMarkItem # gMarkList->CteRead(_CteFirst);

LOOP vMarkItem # gMarkList->CteRead(_CteNext,vMarkItem);

WHILE (vMarkItem <> 0) DO BEGIN

    Lib_Mark:TokenMark(vMarkItem,var vMarkMFile, var vMarkMID);

    if (vMarkMFile = aFile) then begin

        vErx # RecRead(aFile,0,_RecId,vMarkMID);

        if (FldWordByName(aFld1) = aVal) then begin

            vFound # true;

            BREAK;

        end;

    end;

END;

return vFound;

end;

//=====

// _inMarkInt(...)

//   Prüft die Markierten Datensätze, ob ein Wert darin vorkommt

//=====

sub _inMarkInt(aFile : int; aFld1 : alpha; aVal : int) : logic

local begin

    // Markierungsprüfung

    vFound      : logic;

```

```

vMarkItem      : handle;

vMarkMFile,

vMarkMID       : int;

vErx          : int;

end

begin

    vFound # false;

    // Satz in Markierung enthalten?

    FOR vMarkItem # gMarkList->CteRead(_CteFirst);

    LOOP vMarkItem # gMarkList->CteRead(_CteNext,vMarkItem);

    WHILE (vMarkItem <> 0) DO BEGIN

        Lib_Mark:TokenMark(vMarkItem,var vMarkMFile, var vMarkMID);

        if (vMarkMFile = aFile) then begin

            vErx # RecRead(aFile,0,_RecId,vMarkMID);

            if (FldIntByName(aFld1) = aVal) then begin

                vFound # true;

                BREAK;

            end;

        end;

    end;

END;

    return vFound;

end;

```

```

//=====

// _inMarkAlpha(...)

//   Prüft die Markierten Datensätze, ob ein Wert darin vorkommt

//=====

sub _inMarkAlpha(aFile : int; aFld1 : alpha; aVal : alpha) : logic

local begin

    // Markierungsprüfung

    vFound      : logic;

    vMarkItem    : handle;

    vMarkMFile,

    vMarkMID     : int;

    vErx        : int;

end

begin

    vFound # false;

    // Satz in Markierung enthalten?

    FOR vMarkItem # gMarkList->CteRead(_CteFirst);

    LOOP vMarkItem # gMarkList->CteRead(_CteNext,vMarkItem);

    WHILE (vMarkItem <> 0) DO BEGIN

        Lib_Mark:TokenMark(vMarkItem,var vMarkMFile, var vMarkMID);

        if (vMarkMFile = aFile) then begin

            vErx # RecRead(aFile,0,_RecId,vMarkMID);

            if (FldAlphaByName(aFld1) = aVal) then begin

                vFound # true;

                BREAK;

```

```

    end;

end;

END;


return vFound;

end;


//=====

// _insertCon(...)

//   Fügt einen Datensatz in die Controllingliste ein

//=====

sub _insertCon()

local begin

    Erx : int;

end;

begin

    // letzten angehängen Slash abschneiden

    Con.Bezeichnung # StrAdj(Con.Bezeichnung, _StrAll);

    Con.Bezeichnung # StrCut(Con.Bezeichnung,0,Strlen(Con.Bezeichnung)-1);


    // ST neu

    if (Con.Dateinr = 250) and (Con.Artikelnummer <> "") then begin

        // Artikel MEH Lesen

        Erx # RecLink(250,950,5,0); // Artikel holen

        if (Erx<=_rLocked) then

```



```

    Con.MEH # Art.MEH;

end

else begin

    // Material

    Con.MEH # 't';

end;


// Neuen Datensatz anlegen

if (RecRead(950,1,_RecTest) <> _rOK) then begin

//  debug(Con.Bezeichnung);

    Con.Refreshdatum # today;

    Con.Refreshzeit # now;

    RekInsert(950,_recUnlock,'AUTO');

end;


end;


//=====

// _genLoopStat(...)

//  Durchläuft die Statistik Datei und legt für passende Einträge

//  Kennzahlen an

//=====

sub _genLoopStat(

    aJahr    : int;

```

```
aTyp      : int;
aKndAlle  : logic;
aKndMark  : logic;
aVerAlle  : logic;
aVerMark  : logic;
aAArAlle  : logic;
aAArMark  : logic;
aWgrAlle  : logic;
aWgrMark  : logic;
aArgAlle  : logic;
aArgMark  : logic;
aArtAlle  : logic;
aArtMark  : logic;
aGteAlle  : logic;
aGteMark  : logic;
)
```

```
local begin
```

```
    Erx      : int;
```

```
    // Debuginfo
```

```
    vTmp : int;
```

```
    // Kennzahlenprüfung
```

```
    vOK          : int;
```

```
    vTitleLength : int;
```

```
    vTitleCnt     : int;
```

```
// Progressbar  
  
vMax          : int;  
  
// vProgress    : handle;  
  
vProgressTxt   : alpha;
```

```
// Einstellungen
```

```
vKndFile,  
  
vVerFile,  
  
vAArFile,  
  
vWgrFile,  
  
vArgFile,  
  
vArtFile,  
  
vGteFile      : int;
```

```
end;
```

```
begin
```

```
vKndFile      # 100;  
  
vVerFile      # 110;  
  
vAArFile      # 835;  
  
vWgrFile      # 819;  
  
vArgFile      # 826;  
  
vArtFile      # 250;  
  
vGteFile      # 832;
```

```
vMax # RecInfo(899,_RecCount);  
  
vProgressTxt # 'Prüfe ' + Aint(vMAx) + ' Statistikseinträge';  
  
vProgress # Lib_Progress:Init(vProgressTxt , vMax);
```

```
// Bezeichnungslänge errechnen
```

```
vTitleLength # 0;  
  
vTitleCnt # 1;  
  
if ((aKndAlle) or (aKndMark)) then inc(vTitleCnt);  
  
if ((aVerAlle) or (aVerMark)) then inc(vTitleCnt);  
  
if ((aAArAlle) or (aAArMark)) then inc(vTitleCnt);  
  
if ((aWgrAlle) or (aWgrMark)) then inc(vTitleCnt);  
  
if ((aArgAlle) or (aArgMark)) then inc(vTitleCnt);  
  
if ((aArtAlle) or (aArtMark)) then inc(vTitleCnt);  
  
if ((aGteAlle) or (aGteMark)) then inc(vTitleCnt);  
  
vTitleLength # 59 / (vTitleCnt);
```

```
// Alle Umsätze durchlaufen
```

```
FOR Erx # RecRead(899,1,_recFirst);  
  
LOOP Erx # RecRead(899,1,_recNext);  
  
WHILE (Erx <= _rLocked) DO BEGIN
```

```
    // Progress
```

```
    if (lvProgress->Lib_Progress:Step()) then begin  
  
        vProgress->Lib_Progress:Term();  
  
        RETURN;
```

end;

// Nur Daten aus dem angegebenen Jahr prüfen

vTmp # DateYear(Sta.Re.Datum)+1900;

if (vTmp <> aJahr) then

    CYCLE;

// Vorbelegung für Prüfung

RecBufClear(950);

vOK # 0;

// Metadaten

Con.Type # ";

Con.Jahr # aJahr;

Con.Dateinr # aTyp;

if(RecLink(vKndFile,899,1,0) > \_rLocked) then RecBufClear(vKndFile); // Kunden

if(RecLink(vVerFile,899,4,0) > \_rLocked) then RecBufClear(vVerFile); // Vertreter

if(RecLink(vArtFile,899,3,0) > \_rLocked) then RecBufClear(vArtFile); // Artikel

AAr.Nummer # Sta.Auf.Auftragsart;

if (Recread(vAArFile,1,0) > \_rLocked) then RecBufClear(vAArFile);

Wgr.Nummer # Sta.Auf.Warengruppe;

if (Recread(vWgrFile,1,0) > \_rLocked) then RecBufClear(vWgrFile);

AGr.Nummer # Sta.Auf.Artikelgrp;

if (Recread(vArgFile,1,0) > \_rLocked) then RecBufClear(vArgFile);

```
"MQu.Güte1" # "Sta.Auf.Güte";
```

```
if (Recread(vGteFile,2,0) > _rNokey) AND ("MQu.Güte1" <> "Sta.Auf.Güte") then RecBufClear(vGteFile)
```

```
// -----
```

```
// KUNDEN
```

```
// -----
```

```
if ((aKndAlle) or (aKndMark)) AND (Sta.Auf.Kunden.Nr <> 0) then begin
```

```
    // Markierte Sätze prüfen
```

```
    if (aKndMark) AND (_inMarkInt(vKndFile, 'Adr.Kundennr',Sta.Auf.Kunden.Nr) = false) then
```

```
        CYCLE;
```

```
    // Daten übernehmen
```

```
    Con.Adressnummer # Adr.Nummer;
```

```
    Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin)+ StrCut(Sta.Auf.Kunden.SW,1,vTitleLength)
```

```
    _insertCon();
```

```
end;
```

```
// -----
```

```
// VERTRETER
```

```
// -----
```

```
if ((aVerAlle) or (aVerMark)) AND (Sta.Re.Vertreter.Nr <> 0) then begin
```

```
    // Markierte Sätze prüfen
```

```
if (aVerMark) AND (_inMarkInt(vVerFile, 'Ver.Nummer',Sta.Re.Vertreter.Nr) = false) then  
    CYCLE;
```

```
// Daten übernehmen
```

```
Con.Vertreternr # Ver.Nummer;
```

```
Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin)+ StrCut(Sta.Re.Vertreter.SW ,1,vTitleLength),
```

```
_insertCon();
```

```
end;
```

```
// -----
```

```
// AUFTRAGSART
```

```
// -----
```

```
if ((aAArAlle) or (aAArMark)) AND (Sta.Auf.Auftragsart <> 0) then begin
```

```
// Markierte Sätze prüfen
```

```
if (aAArMark) AND (_inMarkInt(vAArFile, 'AAr.Nummer',Sta.Auf.Auftragsart) = false) then
```

```
    CYCLE;
```

```
// Daten übernehmen
```

```
Con.Auftragsart # AAr.Nummer;
```

```
Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin)+ StrCut(AAr.Bezeichnung,1,vTitleLength)+ '/'
```

```
_insertCon();
```

```
end;
```

```
// -----
```

```
// WARENGRUPPEN
```

```
// -----
```

```
if ((aWgrAlle) or (aWgrMark)) AND (Sta.Auf.Warengruppe <> 0) then begin
```

```
    // Markierte Sätze prüfen
```

```
    if (aWgrMark) AND (_inMarkWord(vWgrFile, 'Wgr.Nummer',Sta.Auf.Warengruppe) = false) then
```

```
        CYCLE;
```

```
    // Daten übernehmen
```

```
    Con.Warengruppe # Wgr.Nummer;
```

```
    Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin) + StrCut(Wgr.Bezeichnung.L1,1,vTitleLength)
```

```
    _insertCon();
```

```
end;
```

```
// -----
```

```
// Artikelgruppen
```

```
// -----
```

```
if ((aArgAlle) or (aArgMark)) AND (Sta.Auf.Artikelgrp <> 0) then begin
```

```
    // Markierte Sätze prüfen
```

```
    if (aArgMark) AND (_inMarkWord(vArgFile, 'Arg.Nummer',Sta.Auf.Artikelgrp) = false) then
```

```
        CYCLE;
```

```
    // Daten übernehmen
```

```
    Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin) + StrCut(Agr.Bezeichnung.L1,1,vTitleLength)
```



```
Con.Artikelgruppe # Agr.Nummer;
```

```
_insertCon();
```

```
end;
```

```
// -----
```

```
// Artikelnummer
```

```
// -----
```

```
if ((aArtAlle) or (aArtMark)) AND (Sta.Auf.Artikel.Nr <> "") then begin
```

```
    // Markierte Sätze prüfen
```

```
    if (aArtMark) AND (_inMarkAlpha(vArtFile, 'Art.Nummer',Sta.Auf.Artikel.Nr) = false) then
```

```
        CYCLE;
```

```
    // Daten übernehmen
```

```
    Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin) + StrCut(Art.Nummer,1,vTitleLength) + '/';
```

```
    Con.Artikelnummer # Art.Nummer;
```

```
    _insertCon();
```

```
end;
```

```
// -----
```

```
// Güte
```

```
// -----
```

```
if ((aGteAlle) or (aGteMark)) AND ("Sta.Auf.Güte" <> "") then begin
```

```
    // Markierte Sätze prüfen
```

```
    if (aGteMark) AND (_inMarkAlpha(vGteFile, 'MQu.Güte1',"Sta.Auf.Güte") = false) then
```

CYCLE;

// Daten übernehmen

Con.Bezeichnung # StrAdj(Con.Bezeichnung,\_StrBegin) + StrCut("MQu.Güte1",1,vTitleLength) + '/';

Con.Artikelnummer # "MQu.Güte1"; // Artikelnummer auch für Güte

\_insertCon();

end;

// Nächster Umsatz

END;

vProgress->Lib\_Progress:Term();

end; // sub \_genLoopStat(

//=====

// \_genLoopAll(...)

// Durchläuft alle Dateien hierachisch und legt entsprechende

// Kennzahlen an

//=====

```
sub _genLoopAll(  
    aJahr    : int;  
    aTyp     : int;  
    aKndAlle : logic;  
    aKndMark : logic;  
    aVerAlle : logic;  
    aVerMark : logic;  
    aAArAlle : logic;  
    aAArMark : logic;  
    aWgrAlle : logic;  
    aWgrMark : logic;  
    aArgAlle : logic;  
    aArgMark : logic;  
    aArtAlle : logic;  
    aArtMark : logic;  
    aGteAlle : logic;  
    aGteMark : logic;  
)
```

```
local begin
```

```
    Erx    : int;  
  
    vOK      : int;  
  
    vMax      : int;  
  
    // vProgress    : handle;  
  
    vProgressTxt    : alpha;  
  
  
    vTitleLength    : int;
```

vTitleCnt : int;  
vArtikelJN : logic;  
vKennzahlOk : logic;  
vDbgLine : alpha(200);

vMarkItem : handle;  
vMarkMFile,  
vMarkMID : int;

vKundeFile : int;  
vKundeErx : int;  
vKundeFirst : logic;  
vKundeMarkItem : handle;  
vKundeMarkMFile,  
vKundeMarkMID : int;  
vKundeMarkCnt : int;

vVerFile : int;  
vVerErx : int;  
vVerFirst : logic;  
vVerMarkItem : handle;  
vVerMarkMFile,  
vVerMarkMID : int;  
vVerMarkCnt : int;

vAArFile : int;

vAArErx : int;  
vAArFirst : logic;  
vAArMarkItem : handle;  
vAArMarkMFile,  
vAArMarkMID : int;  
vAArMarkCnt : int;

vWgrFile : int;  
vWgrErx : int;  
vWgrFirst : logic;  
vWgrMarkItem : handle;  
vWgrMarkMFile,  
vWgrMarkMID : int;  
vWgrMarkCnt : int;

vArgFile : int;  
vArgErx : int;  
vArgFirst : logic;  
vArgMarkItem : handle;  
vArgMarkMFile,  
vArgMarkMID : int;  
vArgMarkCnt : int;

vArtFile : int;  
vArtErx : int;  
vArtFirst : logic;

vArtMarkItem : handle;

vArtMarkMFile,

vArtMarkMID : int;

vArtMarkCnt : int;

vGteFile : int;

vGteErx : int;

vGteFirst : logic;

vGteMarkItem : handle;

vGteMarkMFile,

vGteMarkMID : int;

vGteMarkCnt : int;

end;

begin

vArtikelJN # (aTyp = 250);

vKundeFile # 100;

vKundeErx # \_rNoRec;

vKundeFirst # true;

vVerFile # 110;

vVerErx # \_rNoRec;

vVerFirst # true;

vAArFile # 835;

vAArErx # \_rNoRec;

vAArFirst # true;

vWgrFile # 819;

vWgrErx # \_rNoRec;

vWgrFirst # true;

vArgFile # 826;

vArgErx # \_rNoRec;

vArgFirst # true;

vArtFile # 250;

vArtErx # \_rNoRec;

vArtFirst # true;

vGteFile # 832;

vGteErx # \_rNoRec;

vGteFirst # true;

vMax # 1;

// Alle Markierungen durchlaufen und passende Einträge Zählen

FOR vMarkItem # gMarkList->CteRead(\_CteFirst);

LOOP vMarkItem # gMarkList->CteRead(\_CteNext,vMarkItem);

```
WHILE (vMarkItem <> 0) DO BEGIN
```

```
  Lib_Mark:TokenMark(vMarkItem,var vMarkMFile, var vMarkMID);
```

```
  case (vMarkMFile) of
```

```
    vKundeFile  : begin inc(vKundeMarkCnt); end;
```

```
    vVerFile    : begin inc(vVerMarkCnt); end;
```

```
    vAArFile    : begin inc(vAArMarkCnt); end;
```

```
    vWgrFile    : begin inc(vWgrMarkCnt); end;
```

```
    vArgFile    : begin inc(vArgMarkCnt); end;
```

```
    vArtFile    : begin inc(vArtMarkCnt); end;
```

```
    vGteFile    : begin inc(vGteMarkCnt); end;
```

```
  end;
```

```
END;
```

```
if (aKndAlle) then
```

```
  vMax # vMax * RecInfo(vKundeFile,_RecCount);
```

```
if (aKndMark) then
```

```
  vMax # vMax * vKundeMarkCnt;
```

```
if (aVerAlle) then
```

```
  vMax # vMax * RecInfo(vVerFile,_RecCount);
```

```
if (aVerMark) then
```

```
  vMax # vMax * vVerMarkCnt;
```

```
if (aAArAlle) then
```

```
  vMax # vMax * RecInfo(vAArFile,_RecCount);
```



if (aAArMark) then

vMax # vMax \* vAArMarkCnt;

if (aWgrAlle) then

vMax # vMax \* RecInfo(vWgrFile,\_RecCount);

if (aWgrMark) then

vMax # vMax \* vWgrMarkCnt;

if (aArgAlle) then

vMax # vMax \* RecInfo(vArgFile,\_RecCount);

if (aArgMark) then

vMax # vMax \* vArgMarkCnt;

if (aArtAlle) then

vMax # vMax \* RecInfo(vArtFile,\_RecCount);

if (aArtMark) then

vMax # vMax \* vArtMarkCnt;

if (aGteAlle) then

vMax # vMax \* RecInfo(vGteFile,\_RecCount);

if (aGteMark) then

vMax # vMax \* vGteMarkCnt;

vProgressTxt # 'Generiere '+ Aint(vMAx) + ' Kennzahlen';

vProgress # Lib\_Progress:Init(vProgressTxt , vMax);

```
// Bezeichnungslänge errechnen
```

```
vTitleLength # 0;
```

```
vTitleCnt # 1;
```

```
if ((aKndAlle) or (aKndMark)) then inc(vTitleCnt);
```

```
if ((aVerAlle) or (aVerMark)) then inc(vTitleCnt);
```

```
if ((aAArAlle) or (aAArMark)) then inc(vTitleCnt);
```

```
if ((aWgrAlle) or (aWgrMark)) then inc(vTitleCnt);
```

```
if ((aArgAlle) or (aArgMark)) then inc(vTitleCnt);
```

```
if ((aArtAlle) or (aArtMark)) then inc(vTitleCnt);
```

```
if ((aGteAlle) or (aGteMark)) then inc(vTitleCnt);
```

```
vTitleLength # 59 / (vTitleCnt);
```

```
REPEAT
```

```
  // Progress
```

```
  if (lvProgress->Lib_Progress:Step()) then begin
```

```
    vProgress->Lib_Progress:Term();
```

```
    RETURN;
```

```
  end;
```

```
RecBufClear(950);
```

```
vOK # 0;
```

```
/* ----- */
```

```

/* -- Kundenselektion  FIRST                                -- */

/* ----- */

if ((aKndAlle) or (aKndMark)) AND (vKundeFirst) then begin

    vKundeFirst # false;

    // RecRead für ALLE Kunden

    if (aKndAlle) then begin

        vKundeErx # RecRead( vKundeFile, 1, _RecFirst);

        if (vKundeErx <> _rOK) then

            vKundeErx # _rNoRec;

    end; // RecRead für ALLE Kunden


    // Alle Markierten durchgehen

    if (aKndMark) then begin

        // Jeden Kunden der Markierung finden

        REPEAT

            // Next oder First

            if (vKundeMarkItem = 0) then

                vKundeMarkItem # gMarkList->CteRead(_CteFirst);

            else

                vKundeMarkItem # gMarkList->CteRead(_CteNext,vKundeMarkItem);


        // Datensatz Lesen

        if (vKundeMarkItem <> 0) then begin

            Lib_Mark:TokenMark(vKundeMarkItem,var vKundeMarkMFile, var vKundeMarkMID);

            if (vKundeMarkMFile = vKundeFile) then

```

```

    vKundeErx # RecRead(vKundeFile,0,_RecId,vKundeMarkMID);

end;

UNTIL (vKundeMarkMFile = vKundeFile) OR (vKundeMarkItem = 0) ;

if (vKundeMarkItem = 0) then begin
    vKundeErx # _rNoRec;
end;

end; // Alle Markierten durchgehen

end;

```

```

/* ----- */
/* -- Vertreter FIRST -- */
/* ----- */

if ((aVerAlle) or (aVerMark)) AND (vVerFirst) then begin

```

```

    vVerFirst # false;

    // RecRead für ALLE Vertreter

    if (aVerAlle) then begin
        vVerErx # RecRead(vVerFile, 1, _RecFirst);
        if (vVerErx <> _rOK) then
            vVerErx # _rNoRec;
    end; // RecRead für ALLE Vertreter

```

```

// Alle Markierten durchgehen

if (aVerMark) then begin

```

```
// Jeden Vertreter der Markierung finden
```

```
REPEAT
```

```
  // Next oder First
```

```
  if (vVerMarkItem = 0) then
```

```
    vVerMarkItem # gMarkList->CteRead(_CteFirst);
```

```
  else
```

```
    vVerMarkItem # gMarkList->CteRead(_CteNext,vVerMarkItem);
```

```
  // Datensatz Lesen
```

```
  if (vVerMarkItem <> 0) then begin
```

```
    Lib_Mark:TokenMark(vVerMarkItem,var vVerMarkMFile, var vVerMarkMID);
```

```
    if (vVerMarkMFile = vVerFile) then
```

```
      vVerErx # RecRead(vVerFile,0,_RecId,vVerMarkMID);
```

```
    end;
```

```
  UNTIL (vVerMarkMFile = vVerFile) OR (vVerMarkItem = 0) ;
```

```
  if (vVerMarkItem = 0) then begin
```

```
    vVerErx # _rNoRec;
```

```
  end;
```

```
end; // Alle Markierten durchgehen
```

```
end;
```

```
/* ----- */
```

```
/* -- AUFTRAGSART FIRST -- */
```

```
/* ----- */
```

```

if ((aAArAlle) or (aAArMark)) AND (vAArFirst) then begin

    vAArFirst # false;

    // RecRead für ALLE Auftragsarten

    if (aAArAlle) then begin

        vAArErx # RecRead(vAArFile, 1, _RecFirst);

        if (vAArErx <> _rOK) then

            vAArErx # _rNoRec;

    end; // RecRead für ALLE Auftragsarten


    // Alle Markierten durchgehen

    if (aAArMark) then begin

        // Jede Auftragsart der Markierung finden

        REPEAT

            // Next oder First

            if (vAArMarkItem = 0) then

                vAArMarkItem # gMarkList->CteRead(_CteFirst);

            else

                vAArMarkItem # gMarkList->CteRead(_CteNext,vAArMarkItem);


            // Datensatz Lesen

            if (vAArMarkItem <> 0) then begin

                Lib_Mark:TokenMark(vAArMarkItem,var vAArMarkMFile, var vAArMarkMID);

                if (vAArMarkMFile = vAArFile) then

                    vAArErx # RecRead(vAArFile,0,_RecId,vAArMarkMID);

            end;

        UNTIL (vAArMarkMFile = vAArFile) OR (vAArMarkItem = 0) ;

```

```
if (vAArMarkItem = 0) then begin
```

```
    vAArErx # _rNoRec;
```

```
end;
```

```
end; // Alle Markierten durchgehen
```

```
end;
```

```
/* ----- */
```

```
/* -- Warengruppen  FIRST                      -- */
```

```
/* ----- */
```

```
if ((aWgrAlle) or (aWgrMark)) AND (vWgrFirst) then begin
```

```
    vWgrFirst # false;
```

```
    // RecRead für ALLE Warengruppen
```

```
    if (aWgrAlle) then begin
```

```
        vWgrErx # RecRead( vWgrFile, 1, _RecFirst);
```

```
        if (vWgrErx <> _rOK) then
```

```
            vWgrErx # _rNoRec;
```

```
    end; // RecRead für ALLE Warengruppen
```

```
    // Alle Markierten durchgehen
```

```
    if (aWgrMark) then begin
```

```
        // Jede Wgr der Markierung finden
```

```
        REPEAT
```

```
            // Next oder First
```

```
if (vWgrMarkItem = 0) then
```

```
    vWgrMarkItem # gMarkList->CteRead(_CteFirst);
```

```
else
```

```
    vWgrMarkItem # gMarkList->CteRead(_CteNext,vWgrMarkItem);
```

```
// Datensatz Lesen
```

```
if (vWgrMarkItem <> 0) then begin
```

```
    Lib_Mark:TokenMark(vWgrMarkItem,var vWgrMarkMFile, var vWgrMarkMID);
```

```
    if (vWgrMarkMFile = vWgrFile) then
```

```
        vWgrErx # RecRead(vWgrFile,0,_RecId,vWgrMarkMID);
```

```
end;
```

```
UNTIL (vWgrMarkMFile = vWgrFile) OR (vWgrMarkItem = 0) ;
```

```
if (vWgrMarkItem = 0) then begin
```

```
    vWgrErx # _rNoRec;
```

```
end;
```

```
end; // Alle Markierten durchgehen
```

```
end;
```

```
/* ----- */
```

```
/* -- Artikelgruppen  FIRST                      -- */
```

```
/* ----- */
```

```
if ((aArgAlle) or (aArgMark)) AND (vArgFirst) then begin
```

```
    vArgFirst # false;
```



```
// RecRead für ALLE Artikelgruppen
```

```
if (aArgAlle) then begin
```

```
  vArgErx # RecRead( vArgFile, 1, _RecFirst);
```

```
  if (vArgErx <> _rOK) then
```

```
    vArgErx # _rNoRec;
```

```
end; // RecRead für ALLE Artikelgruppen
```

```
// Alle Markierten durchgehen
```

```
if (aArgMark) then begin
```

```
  // Jede Wgr der Markierung finden
```

```
  REPEAT
```

```
    // Next oder First
```

```
    if (vArgMarkItem = 0) then
```

```
      vArgMarkItem # gMarkList->CteRead(_CteFirst);
```

```
    else
```

```
      vArgMarkItem # gMarkList->CteRead(_CteNext,vArgMarkItem);
```

```
  // Datensatz Lesen
```

```
  if (vArgMarkItem <> 0) then begin
```

```
    Lib_Mark:TokenMark(vArgMarkItem,var vArgMarkMFile, var vArgMarkMID);
```

```
    if (vArgMarkMFile = vArgFile) then
```

```
      vArgErx # RecRead(vArgFile,0,_RecId,vArgMarkMID);
```

```
  end;
```

```
UNTIL (vArgMarkMFile = vArgFile) OR (vArgMarkItem = 0) ;
```

```
if (vArgMarkItem = 0) then begin
```

```

    vArgErx # _rNoRec;

end;

end; // Alle Markierten durchgehen

end;

/* ----- */
/* -- Artikel  FIRST          -- */
/* ----- */

if ((aArtAlle) or (aArtMark)) AND (vArtFirst) then begin

    vArtFirst # false;

    // RecRead für ALLE Artikel

    if (aArtAlle) then begin

        vArtErx # RecRead( vArtFile, 1, _RecFirst);

        if (vArtErx <> _rOK) then

            vArtErx # _rNoRec;

    end; // RecRead für ALLE Artikel


    // Alle Markierten durchgehen

    if (aArtMark) then begin

        // Jede Wgr der Markierung finden

        REPEAT

            // Next oder First

            if (vArtMarkItem = 0) then

                vArtMarkItem # gMarkList->CteRead(_CteFirst);

```

else

vArtMarkItem # gMarkList->CteRead(\_CteNext,vArtMarkItem);

// Datensatz Lesen

if (vArtMarkItem <> 0) then begin

Lib\_Mark:TokenMark(vArtMarkItem,var vArtMarkMFile, var vArtMarkMID);

if (vArtMarkMFile = vArtFile) then

vArtErx # RecRead(vArtFile,0,\_RecId,vArtMarkMID);

end;

UNTIL (vArtMarkMFile = vArtFile) OR (vArtMarkItem = 0) ;

if (vArtMarkItem = 0) then begin

vArtErx # \_rNoRec;

end;

end; // Alle Markierten durchgehen

end;

/\* ----- \*/

/\* -- Gueten FIRST -- \*/

/\* ----- \*/

if ((aGteAlle) or (aGteMark)) AND (vGteFirst) then begin

vGteFirst # false;

// RecRead für ALLE Güten

if (aGteAlle) then begin

```

vGteErx # RecRead( vGteFile, 1, _RecFirst);

if (vGteErx <> _rOK) then

    vGteErx # _rNoRec;

end; // RecRead für ALLE Güten


// Alle Markierten durchgehen

if (aGteMark) then begin

    // Jede Güte der Markierung finden

    REPEAT

        // Next oder First

        if (vGteMarkItem = 0) then

            vGteMarkItem # gMarkList->CteRead(_CteFirst);

        else

            vGteMarkItem # gMarkList->CteRead(_CteNext,vGteMarkItem);

        // Datensatz Lesen

        if (vGteMarkItem <> 0) then begin

            Lib_Mark:TokenMark(vGteMarkItem,var vGteMarkMFile, var vGteMarkMID);

            if (vGteMarkMFile = vGteFile) then

                vGteErx # RecRead(vGteFile,0,_RecId,vGteMarkMID);

            end;

        UNTIL (vGteMarkMFile = vGteFile) OR (vGteMarkItem = 0) ;

        if (vGteMarkItem = 0) then begin

            vGteErx # _rNoRec;

        end;

    end;

```

```
end; // Alle Markierten durchgehen
```

```
end;
```

```
/* ----- */
```

```
/* -- Daten vorbelegen -- */
```

```
/* ----- */
```

```
if (vKundeErx <> _rNoRec) then begin
```

```
    inc(vOK);
```

```
    Con.Adressnummer # Adr.Nummer;
```

```
    Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin)+ StrCut(Adr.Stichwort,1,vTitleLength) + '/';
```

```
end;
```

```
if (vVerErx <> _rNoRec) then begin
```

```
    inc(vOK);
```

```
    Con.Vertreternr # Ver.Nummer;
```

```
    Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin)+ StrCut(Ver.Stichwort ,1,vTitleLength)+ '/';
```

```
end;
```

```
if (vAArErx <> _rNoRec) then begin
```

```
    inc(vOK);
```

```
    Con.Auftragsart # AAr.Nummer;
```

```
    Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin)+ StrCut(AAr.Bezeichnung ,1,vTitleLength)+ ' /
```

```
end;
```

```
if (vWgrErx <> _rNoRec) then begin
```

```
  inc(vOK);
```

```
  Con.Warengruppe # Wgr.Nummer;
```

```
  Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin) + StrCut(Wgr.Bezeichnung.L1,1,vTitleLength)
```

```
end;
```

```
if (vArgErx <> _rNoRec) then begin
```

```
  inc(vOK);
```

```
  Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin) + StrCut(Agr.Bezeichnung.L1,1,vTitleLength)
```

```
  Con.Artikelgruppe # Agr.Nummer;
```

```
end;
```

```
if (vArtErx <> _rNoRec) then begin
```

```
  inc(vOK);
```

```
  Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin) + StrCut(Art.Nummer,1,vTitleLength) + '/';
```

```
  Con.Artikelnummer # Art.Nummer;
```

```
end;
```

```
if (vGteErx <> _rNoRec) then begin
```

```
  inc(vOK);
```

```
  Con.Bezeichnung # StrAdj(Con.Bezeichnung,_StrBegin) + StrCut("MQu.Güte1",1,vTitleLength) + '/';
```

```
  Con.Artikelnummer # "MQu.Güte1"; // Artikelnummer auch für Güte
```

```
end;
```

```
/* ===== */
```

```

/* ===== */
/* ===== */
/* == ANLEGEN == */
/* ===== */
/* ===== */
/* ===== */

```

```

if (vOK > 0) then begin

```

```

    // Metadaten

```

```

    Con.Type # ";

```

```

    Con.Jahr # aJahr;

```

```

    Con.Dateinr # aTyp;

```

```

if (Con.Dateinr = 250) and (Con.Artikelnummer <> "") then begin

```

```

    // Artikel MEH Lesen

```

```

    Erx # RecLink(250,950,5,0); // Artikel holen

```

```

    if (Erx<=_rLocked) then

```

```

        Con.MEH # Art.MEH;

```

```

end

```

```

else begin

```

```

    // Material

```

```

    Con.MEH # 't';

```

```

end;

```

```

// letzten anhängen Slash abschneiden

```

```

Con.Bezeichnung # StrCut(con.Bezeichnung,0,Strlen(Con.Bezeichnung)-1);

```

```

Con.Refreshdatum # today;

```

```

Con.Refreshzeit # now;

RekInsert(950,_recUnlock,'AUTO');

//debug(Con.Bezeichnung);

end;

/* ----- */
/* -- Güte      NEXT      -- */
/* ----- */

if (aGteAlle) OR (aGteMark) then begin
  if (aGteAlle) then begin
    vGteErx # RecRead( vGteFile, 1, _RecNext);
    if (vGteErx <> _rOK) then
      vGteErx # _rNoRec;
  end;

  // Nächste Markierten durchgehen
  if (aGteMark) then begin
    REPEAT
      // Next oder First
      if (vGteMarkItem <> 0) then
        vGteMarkItem # gMarkList->CteRead(_CteNext,vGteMarkItem);

      // Datensatz Lesen
      if (vGteMarkItem <> 0) then begin
        Lib_Mark:TokenMark(vGteMarkItem,var vGteMarkMFile, var vGteMarkMID);
        if (vGteMarkMFile = vGteFile) then

```



```

    RecRead(vGteFile,0,_RecId,vGteMarkMID);

end;

UNTIL (vGteMarkMFile = vGteFile) OR (vGteMarkItem = 0) ;

if (vGteMarkItem = 0) then
    vGteErx # _rNoRec;
else
    vGteErx # _rOK;

end;

// Alle Innerliegenden Schleifen neustarten lassen
if (vGteErx = _rOK) then begin
//    vWgrFirst  # true;

    CYCLE;

end;

end;

/* ----- */
/* -- Artikel      NEXT                -- */
/* ----- */

if (aArtAlle) OR (aArtMark) then begin
if (aArtAlle) then begin
    vArtErx # RecRead( vArtFile, 1, _RecNext);

    if (vArtErx <> _rOK) then

        vArtErx # _rNoRec;

```

end;

// Nächste Markierten durchgehen

if (aArtMark) then begin

REPEAT

    // Next oder First

    if (vArtMarkItem <> 0) then

        vArtMarkItem # gMarkList->CteRead(\_CteNext,vArtMarkItem);

    // Datensatz Lesen

    if (vArtMarkItem <> 0) then begin

        Lib\_Mark:TokenMark(vArtMarkItem,var vArtMarkMFile, var vArtMarkMID);

        if (vArtMarkMFile = vArtFile) then

            RecRead(vArtFile,0,\_RecId,vArtMarkMID);

    end;

UNTIL (vArtMarkMFile = vArtFile) OR (vArtMarkItem = 0) ;

if (vArtMarkItem = 0) then

    vArtErx # \_rNoRec;

else

    vArtErx # \_rOK;

end;

// Alle Innerliegenden Schleifen neustarten lassen

if (vArtErx = \_rOK) then begin

```
//      vWgrFirst  # true;
```

```
    CYCLE;
```

```
end;
```

```
end;
```

```
/* ----- */
```

```
/* -- Artikelgruppe    NEXT                -- */
```

```
/* ----- */
```

```
if (aArgalle) OR (aArgMark) then begin
```

```
  if (aArgAlle) then begin
```

```
    vArgErx # RecRead( vArgFile, 1, _RecNext);
```

```
    if (vArgErx <> _rOK) then
```

```
      vArgErx # _rNoRec;
```

```
  end;
```

```
// Nächste Markierten durchgehen
```

```
if (aArgMark) then begin
```

```
  REPEAT
```

```
    // Next oder First
```

```
    if (vArgMarkItem <> 0) then
```

```
      vArgMarkItem # gMarkList->CteRead(_CteNext,vArgMarkItem);
```

```
    // Datensatz Lesen
```

```
    if (vArgMarkItem <> 0) then begin
```

```
      Lib_Mark:TokenMark(vArgMarkItem,var vArgMarkMFile, var vArgMarkMID);
```

```
if (vArgMarkMFile = vArgFile) then
```

```
    RecRead(vArgFile,0,_RecId,vArgMarkMID);
```

```
end;
```

```
UNTIL (vArgMarkMFile = vArgFile) OR (vArgMarkItem = 0) ;
```

```
if (vArgMarkItem = 0) then
```

```
    vArgErx # _rNoRec;
```

```
else
```

```
    vArgErx # _rOK;
```

```
end;
```

```
// Alle Innerliegenden Schleifen neustarten lassen
```

```
if (vArgErx = _rOK) then begin
```

```
    vArtFirst # true;
```

```
    CYCLE;
```

```
end;
```

```
end;
```

```
/* ----- */
```

```
/* -- Warengruppe    NEXT                      -- */
```

```
/* ----- */
```

```
if (aWgralle) OR (aWgrMark) then begin
```

```

if (aWgrAlle) then begin
    vWgrErx # RecRead( vWgrFile, 1, _RecNext);
    if (vWgrErx <> _rOK) then
        vWgrErx # _rNoRec;
end;

// Nächste Markierten durchgehen
if (aWgrMark) then begin
    REPEAT
        // Next oder First
        if (vWgrMarkItem <> 0) then
            vWgrMarkItem # gMarkList->CteRead(_CteNext,vWgrMarkItem);

        // Datensatz Lesen
        if (vWgrMarkItem <> 0) then begin
            Lib_Mark:TokenMark(vWgrMarkItem,var vWgrMarkMFile, var vWgrMarkMID);
            if (vWgrMarkMFile = vWgrFile) then
                RecRead(vWgrFile,0,_RecId,vWgrMarkMID);
        end;
    UNTIL (vWgrMarkMFile = vWgrFile) OR (vWgrMarkItem = 0) ;

    if (vWgrMarkItem = 0) then
        vWgrErx # _rNoRec;
    else
        vWgrErx # _rOK;

```

end;

// Alle Innerliegenden Schleifen neustarten lassen

if (vWgrErx = \_rOK) then begin

    vArgFirst # true;

    vArtFirst # true;

    vGteFirst # true;

    CYCLE;

end;

end;

/\* ----- \*/

/\* -- Auftragsart   NEXT                               -- \*/

/\* ----- \*/

if ((aAArAlle) OR (aAArMark)) then begin

    if (aAArAlle) then begin

        vAArErx # RecRead( vAArFile, 1, \_RecNext);

        if (vAArErx <> \_rOK) then

            vAArErx # \_rNoRec;

    end;

// Nächste Markierten durchgehen

if (aAArMark) then begin

    REPEAT

        // Next oder First

        if (vAArMarkItem <> 0) then

```
vAArMarkItem # gMarkList->CteRead(_CteNext,vAArMarkItem);
```

```
// Datensatz Lesen
```

```
if (vAArMarkItem <> 0) then begin
```

```
    Lib_Mark:TokenMark(vAArMarkItem,var vAArMarkMFile, var vAArMarkMID);
```

```
    if (vAArMarkMFile = vAArFile) then
```

```
        RecRead(vAArFile,0,_RecId,vAArMarkMID);
```

```
end;
```

```
UNTIL (vAArMarkMFile = vAArFile) OR (vAArMarkItem = 0) ;
```

```
if (vAArMarkItem = 0) then
```

```
    vAArErx # _rNoRec;
```

```
else
```

```
    vAArErx # _rOK;
```

```
end;
```

```
// Alle Innerliegenden Schleifen neustarten lassen
```

```
if (vAArErx = _rOK) then begin
```

```
    vWgrFirst # true;
```

```
    vArgFirst # true;
```

```
    vArtFirst # true;
```

```
    vGteFirst # true;
```

```
    CYCLE;
```

```
end;
```

```
end;
```

```
/* ----- */
```

```
/* -- Vertreter      NEXT      -- */
```

```
/* ----- */
```

```
if ((aVerAlle) OR (aVerMark)) then begin
```

```
  if (aVerAlle) then begin
```

```
    vVerErx # RecRead( vVerFile, 1, _RecNext);
```

```
    if (vVerErx <> _rOK) then
```

```
      vVerErx # _rNoRec;
```

```
  end;
```

```
// Nächste Markierten durchgehen
```

```
if (aVerMark) then begin
```

```
  REPEAT
```

```
    // Next oder First
```

```
    if (vVerMarkItem <> 0) then
```

```
      vVerMarkItem # gMarkList->CteRead(_CteNext,vVerMarkItem);
```

```
// Datensatz Lesen
```

```
if (vVerMarkItem <> 0) then begin
```

```
  Lib_Mark:TokenMark(vVerMarkItem,var vVerMarkMFile, var vVerMarkMID);
```

```
  if (vVerMarkMFile = vVerFile) then
```

```
    RecRead(vVerFile,0,_RecId,vVerMarkMID);
```

```
  end;
```

```
UNTIL (vVerMarkMFile = vVerFile) OR (vVerMarkItem = 0) ;
```

```
if (vVerMarkItem = 0) then
```



```

    vVerErx # _rNoRec;

else

    vVerErx # _rOK;

end;

// Alle Innerliegenden Schleifen neustarten lassen

if (vVerErx = _rOK) then begin

    vAArFirst # true;

    vWgrFirst # true;

    vArgFirst # true;

    vArtFirst # true;

    vGteFirst # true;

    CYCLE;

end;

end;

/* ----- */

/* -- Kundenselektion  NEXT                -- */

/* ----- */

if ((aKndAlle) OR (aKndMark)) then begin

    if (aKndAlle) then begin

        vKundeErx # RecRead( vKundeFile, 1, _RecNext);

        if (vKundeErx <> _rOK) then

            vKundeErx # _rNoRec;

        end;

    end;

end;

```

```
// Nächste Markierten durchgehen
```

```
if (aKndMark) then begin
```

```
  REPEAT
```

```
    // Next oder First
```

```
    if (vKundeMarkItem <> 0) then
```

```
      vKundeMarkItem # gMarkList->CteRead(_CteNext,vKundeMarkItem);
```

```
    // Datensatz Lesen
```

```
    if (vKundeMarkItem <> 0) then begin
```

```
      Lib_Mark:TokenMark(vKundeMarkItem,var vKundeMarkMFile, var vKundeMarkMID);
```

```
      if (vKundeMarkMFile = vKundeFile) then
```

```
        RecRead(vKundeFile,0,_RecId,vKundeMarkMID);
```

```
    end;
```

```
  UNTIL (vKundeMarkMFile = vKundeFile) OR (vKundeMarkItem = 0) ;
```

```
  if (vKundeMarkItem = 0) then
```

```
    vKundeErx # _rNoRec;
```

```
  else
```

```
    vKundeErx # _rOK;
```

```
end;
```

```
// Alle Innerliegenden Schleifen neustarten lassen
```

```
if (vKundeErx = _rOK) then begin
```

```
  vAArFirst  # true;
```

```
  vVerFirst  # true;
```

```
vWgrFirst # true;
```

```
vArgFirst # true;
```

```
vArtFirst # true;
```

```
vGteFirst # true;
```

```
CYCLE;
```

```
end;
```

```
end;
```

```
UNTIL (
```

```
(vKundeErx = _rNoRec) AND
```

```
(vVerErx = _rNoRec) AND
```

```
(vAArErx = _rNoRec) AND
```

```
(vWgrErx = _rNoRec) AND
```

```
(vArgErx = _rNoRec) AND
```

```
(vArtErx = _rNoRec) AND
```

```
(vGteErx = _rNoRec)
```

```
);
```

```
vProgress->Lib_Progress:Term();
```

```
end; // sub _genLoopAll(
```

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

```
// Generieren
```

```
// Startet die Generierung von Kennzahlen
```

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

```
sub Generieren()
```

```
local begin
```

```
    aJahr    : int;
```

```
    aTyp     : int;
```

```
    aNurMitUms : logic;
```

```
    aKndAlle  : logic;
```

```
    aKndMark  : logic;
```

```
    aVerAlle  : logic;
```

```
    aVerMark  : logic;
```

```
    aAArAlle  : logic;
```

```
    aAArMark  : logic;
```

```
    aWgrAlle  : logic;
```

```
    aWgrMark  : logic;
```

```
    aKstAlle  : logic;
```

```
    aKstMark  : logic;
```

```
    aArgAlle  : logic;
```

```
    aArgMark  : logic;
```

```
    aArtAlle  : logic;
```

```
    aArtMark  : logic;
```

```
    aGteAlle  : logic;
```

```
    aGteMark  : logic;
```

```
    aConTyp   : alpha;
```

```
end
```

begin

// Abfrage

```
if (Dlg_Standard:Con_Generieren(var aJahr,  
    var aTyp,  
    var aNurMitUms,  
    var aKndAlle,  
    var aKndMark,  
    var aVerAlle,  
    var aVerMark,  
    var aAArAlle,  
    var aAArMark,  
    var aWgrAlle,  
    var aWgrMark,  
    var aKstAlle,  
    var aKstMark,  
    var aArgAlle,  
    var aArgMark,  
    var aArtAlle,  
    var aArtMark,  
    var aGteAlle,  
    var aGteMark, var aConTyp)) then begin
```

// Jahr aus Dialog übernehmen

gvJahr # aJahr;

```
if (aNurMitUms) then begin
```

```
  _genLoopStat(aJahr,
```

```
    aTyp,
```

```
    aKndAlle,
```

```
    aKndMark,
```

```
    aVerAlle,
```

```
    aVerMark,
```

```
    aAArAlle,
```

```
    aAArMark,
```

```
    aWgrAlle,
```

```
    aWgrMark,
```

```
    aArgAlle,
```

```
    aArgMark,
```

```
    aArtAlle,
```

```
    aArtMark,
```

```
    aGteAlle,
```

```
    aGteMark);
```

```
end
```

```
else begin
```

```
  // Loop durch markierte Daten
```

```
  _genLoopAll(aJahr,
```

```
    aTyp,
```

```
    aKndAlle,
```

```
    aKndMark,
```

```
    aVerAlle,
```

```
aVerMark,  
aAArAlle,  
aAArMark,  
aWgrAlle,  
aWgrMark,  
aArgAlle,  
aArgMark,  
aArtAlle,  
aArtMark,  
aGteAlle,  
aGteMark);
```

```
end;
```

```
// Recalc
```

```
Recalc(true);
```

```
// Refresh
```

```
gZLList->WinUpdate(_WinUpdOn, _WinLstRecFromRecid | _WinLstRecDoSelect);
```

```
end;
```

```
end;
```

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

```
// GenerierenVorgaben
```

```
// Generiert Vorgaben für markierte Controllingdatensätze
```

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

```
sub GenerierenVorgaben()
```

```
local begin
```

```
    aJahr      : int;
```

```
    aFaktMenge : float;
```

```
    aFaktUmsatz : float;
```

```
    aFaktDB    : float;
```

```
    vVal       : float;
```

```
    vltem      : handle;
```

```
    vMFile,vMID : int;
```

```
    i          : int;
```

```
end
```

```
begin
```

```
    // Abfrage
```

```
    if (Dlg_Standard:Con_GenerierenVorgaben(var aJahr,
```

```
        var aFaktMenge,
```

```
        var aFaktUmsatz,
```

```
        var aFaktDB)) then begin
```

```
        if (aJahr <> 0) then begin
```

```
            vProgress # Lib_Progress:Init( 'Vorgaben werden generiert...');
```

```
            // Markierungen durchlaufen
```

```
            vltem # gMarkList->CteRead(_CteFirst);
```



```

WHILE (vItem > 0) do begin

  Lib_Mark:TokenMark(vItem,var vMFile, var vMID);

  if (vMFile = 950) then begin

    vProgress->Lib_Progress:Step();

    RecRead(950,0,_RecId,vMID);


    // Markierten Datensatz mit neuen Werten eintragen

    Con.Bemerkung # 'Jahr ' + Aint(Con.Jahr) + ' +/-: ';

    if (aFaktMenge <> 0.0) then

      Con.Bemerkung # Con.Bemerkung + ' Menge ' +Anum(aFaktMenge,2) + '%';

    if (aFaktUmsatz <> 0.0) then

      Con.Bemerkung # Con.Bemerkung + ' Umsatz ' +Anum(aFaktUmsatz,2) + '%';

    if (aFaktDB <> 0.0) then

      Con.Bemerkung # Con.Bemerkung + ' DB ' +Anum(aFaktDB,2) + '%';


    Con.Jahr # aJahr;


    // Felder nullen, bzw. Berechnen

    FOR i # 1

    LOOP inc(i)

    WHILE (i <= 13)

    DO BEGIN


      // -----

      // MENGEN');

      vVal # FldFloat(950,3,i); // IST

```

vVal # vVal + (vVal\*0.01\*aFaktMenge);

FldDef(950,2,i, vVal); // SOLLWERT

FldDef(950,3,i, 0.0); // IST nullen

FldDef(950,4,i, 0.0); // SIM nullen

// -----');

// UMSATZ');

vVal # FldFloat(950,6,i); // IST

vVal # vVal + (vVal\*0.01\*aFaktUmsatz);

FldDef(950,5,i, vVal); // SOLLWERT

FldDef(950,6,i, 0.0); // IST nullen

FldDef(950,7,i, 0.0); // SIM nullen

// -----');

// DB');

vVal # FldFloat(950,12,i); // IST

vVal # vVal + (vVal\*0.01\*aFaktDB);

FldDef(950,11,i, vVal); // SOLLWERT

FldDef(950,12,i, 0.0); // IST nullen

FldDef(950,13,i, 0.0); // SIM nullen

// -----

// Proz

FldDef(950,8,i, 0.0); // SOLL nullen

FldDef(950,9,i, 0.0); // IST nullen

```
FldDef(950,10,i, 0.0); // SIM nullen
```

```
END;
```

```
Con.Refreshdatum # today;
```

```
Con.Refreshzeit # now;
```

```
// Datensatz anlegen
```

```
RekInsert(950,0,'AUTO');
```

```
end;
```

```
vltem # gMarkList->CteRead(_CteNext, vltem);
```

```
END;
```

```
end;
```

```
vProgress->Lib_Progress:Term();
```

```
// Recalc
```

```
Recalc(true);
```

```
// Refresh
```

```
WinUpdate(gMDI);
```

```
end;
```

```
end;
```

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

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

```
sub _AddWert(
```

```
    aDat : date;
```

```
    aM   : float;
```

```
    aUms : float;
```

```
    aDB  : float;
```

```
)
```

```
local begin
```

```
    vl : int;
```

```
end;
```

```
begin
```

```
    if (aDat=0.0.0) then RETURN;
```

```
    vl # aDat->vpMonth;
```

```
    // Menge
```

```
    FldDef(950,3,vl, FldFloat(950,3,vl) + aM);
```

```
    // Umsatz
```

```
    FldDef(950,6,vl, FldFloat(950,6,vl) + aUms);
```

```
    // DB
```

```
    FldDef(950,12,vl, FldFloat(950,12,vl) + aDB);
```

```
end;
```

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

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

```
sub _Summieren()
```

```
local begin
```

```

vM   : float;

vUms : float;

vDB   : float;

vI    : int;

vX    : float;

end;

begin

vM   # 0.0;

vUms # 0.0;

vDB  # 0.0;

FOR vI # 1 loop inc(vI) while (vI<=12) do begin

vM   # vM + FldFloat(950,3,vI);

vUms # vUms + FldFloat(950,6,vI);

vDB  # vDB + FldFloat(950,12,vI);

// Prozente

vX # Lib_Berechnungen:Prozent( FldFloat(950,12,vI), FldFloat(950,6,vI));

FldDef(950,9,vI, vX);

END;


Con.Ist.Menge.Sum   # vM;

Con.Ist.Umsatz.Sum  # vUms;

Con.Ist.DB.Sum      # vDB;

Con.Ist.Proz.Sum    # Lib_Berechnungen:Prozent(vDB, vUms);

end;

```

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

```
// _RecalcEingangAuf
```

```
//
```

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

```
sub _RecalcEingangAuf(
```

```
    aKdNr    : int;
```

```
    aVorgang : alpha)
```

```
local begin
```

```
    Erx      : int;
```

```
    vDat     : date;
```

```
    vDat2    : date;
```

```
    vQ400    : alpha(1000);
```

```
    vQ401    : alpha(1000);
```

```
    vQ250    : alpha(1000);
```

```
    vSel     : int;
```

```
    vSelName : alpha;
```

```
    vM       : float;
```

```
    vUms     : float;
```

```
    vKurs    : float;
```

```
end;
```

```
begin
```

```
    vDat # DateMake(1,1, Con.Jahr-1900);
```

```
    vDat2 # DateMake(31,12, Con.Jahr-1900);
```

```
// BESTAND 401 -----
```

```
vQ401 # ";
```

```
Lib_Sel:QInt( var vQ401, 'Auf.P.Nummer', '<', 1000000000 );
```

```
if (aKdNr<>0) then
```

```
    Lib_Sel:QInt( var vQ401, 'Auf.P.Kundennr', '=', aKdNr);
```

```
if (Con.Auftragsart<>0) then
```

```
    Lib_Sel:QInt( var vQ401, 'Auf.P.Auftragsart', '=', Con.Auftragsart);
```

```
if (Con.Warengruppe<>0) then
```

```
    Lib_Sel:QInt( var vQ401, 'Auf.P.Warengruppe', '=', Con.Warengruppe);
```

```
// Artikel oder Material ?
```

```
if (Con.Dateinr = 200) then begin
```

```
    if (Con.Artikelnummer<>") then begin
```

```
        Lib_Sel:QAlpha( var vQ401, "Auf.P.Güte", '=*', Con.Artikelnummer);
```

```
    end;
```

```
end
```

```
else begin
```

```
    if (Con.Artikelnummer<>") then begin
```

```
        Lib_Sel:QAlpha( var vQ401, 'Auf.P.Artikelnr', '=', Con.Artikelnummer);
```

```
    end;
```

```
    if (Con.Artikelgruppe<>0) then begin
```

```
        Lib_Sel:QInt( var vQ250, 'Art.Artikelgruppe', '=', Con.Artikelgruppe);
```

```
    end;
```

```
end;
```

```
if (vQ401 != "") then vQ401 # vQ401 + ' AND ';
```

```
vQ401 # vQ401 + 'LinkCount(Kopf) > 0 ';
```

```
if (vQ250 != "") then begin
```

```
    vQ401 # vQ401 + ' AND ';
```

```
    vQ401 # vQ401 + 'LinkCount(Art) > 0 ';
```

```
end;
```

```
// Selektionsquery für 400
```

```
vQ400 # "";
```

```
vQ400 # '(Auf.Vorgangstyp="" + aVorgang + "'');'
```

```
vQ400 # vQ400 + ' AND "Auf.LiefervertragYN"=false';
```

```
if (Con.Vertreternr != 0) then
```

```
    Lib_Sel:QInt( var vQ400, 'Auf.Vertreter', '=', Con.Vertreternr);
```

```
Lib_Sel:QVonBisD( var vQ400, 'Auf.Datum', vDat, vDat2);
```

```
// Selektion starten...
```

```
vSel # SelCreate( 401, 1 );
```

```
vSel->SelAddLink("", 400, 401, 3, 'Kopf');
```

```
if (vQ250<>") then
```

```
    vSel->SelAddLink("", 250, 401, 2, 'Art');
```

```
Erx # vSel->SelDefQuery("", vQ401 );
```

```
if (Erx <> 0) then begin
```

```
    Lib_Sel:QError(vSel);
```

```
    RETURN;
```



end;

Erx # vSel->SelDefQuery('Kopf', vQ400 );

if (Erx <> 0) then begin

Lib\_Sel:QError(vSel);

RETURN;

end;

if (vQ250<>'') then begin

Erx # vSel->SelDefQuery('Art', vQ250 );

if (Erx <> 0) then begin

Lib\_Sel:QError(vSel);

RETURN;

end;

end;

vSelName # Lib\_Sel:SaveRun( var vSel, 0);

FOR Erx # RecRead(401,vSel,\_recFirst)

LOOP Erx # RecRead(401,vSel,\_recNext)

WHILE (Erx<=\_rLocked) do begin

//debug('KEY401');

Erx # RecLink(400,401,3,\_recFirst); // Kopf holen

vKurs # 1.0;

if ("Auf.Währung"<>'') then begin

if ("Auf.WährungFixYN") then begin

vKurs # "Auf.Währungskurs";

end

else begin

RecLink(814,400,8,\_recfirst); // Währung holen

vKurs # "Wae.VK.Kurs";

end;

end;

// Menge

if (Auf.P.MEH.Einsatz=Con.MEH) then

vM # Auf.P.Menge

else if (Auf.P.MEH.Wunsch=Con.MEH) then

vM # Auf.P.Menge.Wunsch

else if (Con.MEH='t') then

vM # (Auf.P.Gewicht / 1000.0);

// Umsatz

vUms # Auf.P.Gesamtpreis;

vUms # Rnd(vUms / vKurs,2);

\_AddWert(Auf.Datum, vM, vUms, 0.0);

END;

SelDelete(401, vSelName);

// ABLAGE 411 -----

vQ401 # ";

vQ400 # '';

vQ250 # '';

if (aKdNr<>0) then

Lib\_Sel:QInt( var vQ401, 'Auf~P.Kundennr', '=', aKdNr);

if (Con.Auftragsart<>0) then

Lib\_Sel:QInt( var vQ401, 'Auf~P.Auftragsart', '=', Con.Auftragsart);

if (Con.Warengruppe<>0) then

Lib\_Sel:QInt( var vQ401, 'Auf~P.Warengruppe', '=', Con.Warengruppe);

// Artikel oder Material ?

if (Con.Dateinr = 200) then begin

if (Con.Artikelnummer<>") then begin

Lib\_Sel:QAlpha( var vQ401, '"Auf~P.Güte"', '=\*', Con.Artikelnummer);

end;

end

else begin

if (Con.Artikelnummer<>") then begin

Lib\_Sel:QAlpha( var vQ401, 'Auf~P.Artikelnr', '=', Con.Artikelnummer);

end;

if (Con.Artikelgruppe<>0) then begin

Lib\_Sel:QInt( var vQ250, 'Art.Artikelgruppe', '=', Con.Artikelgruppe);

end;

end;

if (vQ401 != '') then vQ401 # vQ401 + ' AND ';

vQ401 # vQ401 + 'LinkCount(Kopf) > 0 ';

```
if (vQ250 != "") then begin
```

```
    vQ401 # vQ401 + ' AND ';
```

```
    vQ401 # vQ401 + 'LinkCount(Art) > 0 ';
```

```
end;
```

```
// Selektionsquery für 410
```

```
vQ400 # ";
```

```
vQ400 # '("Auf~Vorgangstyp"="" + aVorgang + "');
```

```
vQ400 # vQ400 + ' AND "Auf~LiefervertragYN"=false';
```

```
if (Con.Vertreternr != 0) then
```

```
    Lib_Sel:QInt( var vQ400, 'Auf~Vertreter', '=', Con.Vertreternr);
```

```
Lib_Sel:QVonBisD( var vQ400, 'Auf~Datum', vDat, vDat2);
```

```
// Selektion starten...
```

```
vSel # SelCreate( 411, 1 );
```

```
vSel->SelAddLink(", 410, 411, 3, 'Kopf');
```

```
if (vQ250<>") then
```

```
    vSel->SelAddLink(", 250, 411, 2, 'Art');
```

```
Erx # vSel->SelDefQuery(", vQ401 );
```

```
if (Erx <> 0) then begin
```

```
    Lib_Sel:QError(vSel);
```

```
    RETURN;
```

```
end;
```

```

Erx # vSel->SelDefQuery('Kopf', vQ400 );

if (Erx <> 0) then begin

    Lib_Sel:QError(vSel);

    RETURN;

end;

if (vQ250<>'') then begin

    Erx # vSel->SelDefQuery('Art', vQ250 );

    if (Erx <> 0) then begin

        Lib_Sel:QError(vSel);

        RETURN;

    end;

end;

vSelName # Lib_Sel:SaveRun( var vSel, 0);


FOR Erx # RecRead(411,vSel,_recFirst)
LOOP Erx # RecRead(411,vSel,_recNext)
WHILE (Erx<=_rLocked) do begin
//debug('KEY401');

    Erx # RecLink(410,411,3,_recFirst); // Kopf holen

    RecBufCopy(410,400);

    RecbufCopy(411,401);

    vKurs # 1.0;

    if ("Auf.Währung"<>1) then begin

        if ("Auf.WährungFixYN") then begin

            vKurs # "Auf.Währungskurs";

```

```

end

else begin

    RecLink(814,400,8,_recfirst);  // Währung holen

    vKurs # "Wae.VK.Kurs";

end;

end;


// Menge

if (Auf.P.MEH.Einsatz=Con.MEH) then

    vM # Auf.P.Menge

else if (Auf.P.MEH.Wunsch=Con.MEH) then

    vM # Auf.P.Menge.Wunsch

else if (Con.MEH='t') then

    vM # (Auf.P.Gewicht / 1000.0);


// Umsatz

vUms # Auf.P.Gesamtpreis;

vUms # Rnd(vUms / vKurs,2);


_AddWert(Auf.Datum, vM, vUms, 0.0);

END;


SelDelete(411, vSelName);


_Summieren();

```

```

end;

//=====

// _RecalcEingangBest
//

//=====

sub _RecalcEingangBest(
    aLfNr    : int;
    aVorgang : alpha)
local begin
    Erx      : int;
    vDat      : date;
    vDat2     : date;
    vQ500     : alpha(1000);
    vQ501     : alpha(1000);
    vQ250     : alpha(1000);
    vSel      : int;
    vSelName  : alpha;
    vM        : float;
    vUms      : float;
    vKurs     : float;
end;

begin
    vDat # DateMake(1,1, Con.Jahr-1900);
    vDat2 # DateMake(31,12, Con.Jahr-1900);

```

// BESTAND 501 -----

vQ501 # ";

Lib\_Sel:QInt( var vQ501, 'Ein.P.Nummer', '<', 1000000000 );

if (aLfNr<>0) then

Lib\_Sel:QInt( var vQ501, 'Ein.P.Lieferantennr', '=', aLfNr);

if (Con.Auftragsart<>0) then

Lib\_Sel:QInt( var vQ501, 'Ein.P.Auftragsart', '=', Con.Auftragsart);

if (Con.Warengruppe<>0) then

Lib\_Sel:QInt( var vQ501, 'Ein.P.Warengruppe', '=', Con.Warengruppe);

// Artikel oder Material ?

if (Con.Dateinr = 200) then begin

if (Con.Artikelnummer<>") then begin

Lib\_Sel:QAlpha( var vQ501, '"Ein.P.Güte"', '=\*', Con.Artikelnummer);

end;

end

else begin

if (Con.Artikelnummer<>") then begin

Lib\_Sel:QAlpha( var vQ501, 'Ein.P.Artikelnr', '=', Con.Artikelnummer);

end;

if (Con.Artikelgruppe<>0) then begin

Lib\_Sel:QInt( var vQ250, 'Art.Artikelgruppe', '=', Con.Artikelgruppe);

end;



end;

if (vQ501 != "") then vQ501 # vQ501 + ' AND ';

vQ501 # vQ501 + 'LinkCount(Kopf) > 0 ';

if (vQ250 != "") then begin

    vQ501 # vQ501 + ' AND ';

    vQ501 # vQ501 + 'LinkCount(Art) > 0 ';

end;

// Selektionsquery für 500

vQ500 # ";

vQ500 # '(Ein.Vorgangstyp="' + aVorgang + '")';

vQ500 # vQ500 + ' AND "Ein.LiefervertragYN"=false';

// if (Con.Vertreternr != 0) then

//   Lib\_Sel:QInt( var vQ500, 'Ein.Vertreter', '=', Con.Vertreternr);

Lib\_Sel:QVonBisD( var vQ500, 'Ein.Datum', vDat, vDat2);

// Selektion starten...

vSel # SelCreate( 501, 1 );

vSel->SelAddLink("", 500, 501, 3, 'Kopf');

if (vQ250<>") then

    vSel->SelAddLink("", 250, 501, 2, 'Art');

Erx # vSel->SelDefQuery("", vQ501 );

if (Erx <> 0) then begin

```

Lib_Sel:QError(vSel);

RETURN;

end;

Erx # vSel->SelDefQuery('Kopf', vQ500 );

if (Erx <> 0) then begin

    Lib_Sel:QError(vSel);

    RETURN;

end;


if (vQ250<>") then begin

    Erx # vSel->SelDefQuery('Art', vQ250 );

    if (Erx <> 0) then begin

        Lib_Sel:QError(vSel);

        RETURN;

    end;

end;

vSelName # Lib_Sel:SaveRun( var vSel, 0);


FOR Erx # RecRead(501,vSel,_recFirst)
LOOP Erx # RecRead(501,vSel,_recNext)
WHILE (Erx<=_rLocked) do begin

    Erx # RecLink(500,501,3,_recFirst); // Kopf holen

    vKurs # 1.0;

    if ("Ein.Währung"<>1) then begin

        if ("Ein.WährungFixYN") then begin

            vKurs # "Ein.Währungskurs";

```

```
end  
  
else begin  
  
    RecLink(814,500,8,_recfirst);  // Währung holen  
  
    vKurs # "Wae.EK.Kurs";  
  
end;  
  
end;
```

```
// Menge
```

```
if (Ein.P.MEH=Con.MEH) then  
  
    vM # Ein.P.Menge  
  
else if (Ein.P.MEH.Wunsch=Con.MEH) then  
  
    vM # Ein.P.Menge.Wunsch  
  
else if (Con.MEH='t') then  
  
    vM # (Ein.P.Gewicht / 1000.0);
```

```
// Umsatz
```

```
vUms # Ein.P.Gesamtpreis;  
  
vUms # Rnd(vUms / vKurs,2);
```

```
_AddWert(Ein.Datum, vM, vUms, 0.0);  
  
END;
```

```
SelDelete(501, vSelName);
```

```
// ABLAGE 511 -----
```

vQ501 # ";

vQ500 # ";

vQ250 # ";

vQ501 # ";

if (aLfNr<>0) then

Lib\_Sel:QInt( var vQ501, 'Ein~P.Lieferantennr', '=', aLfNr);

if (Con.Auftragsart<>0) then

Lib\_Sel:QInt( var vQ501, 'Ein~P.Auftragsart', '=', Con.Auftragsart);

if (Con.Warengruppe<>0) then

Lib\_Sel:QInt( var vQ501, 'Ein~P.Warengruppe', '=', Con.Warengruppe);

// Artikel oder Material ?

if (Con.Dateinr = 200) then begin

if (Con.Artikelnummer<>") then begin

Lib\_Sel:QAlpha( var vQ501, ""Ein~P.Güte"", '=\*', Con.Artikelnummer);

end;

end

else begin

if (Con.Artikelnummer<>") then begin

Lib\_Sel:QAlpha( var vQ501, 'Ein~P.Artikelnr', '=', Con.Artikelnummer);

end;

if (Con.Artikelgruppe<>0) then begin

Lib\_Sel:QInt( var vQ250, 'Art.Artikelgruppe', '=', Con.Artikelgruppe);

end;

end;

if (vQ501 != "") then vQ501 # vQ501 + ' AND ';

vQ501 # vQ501 + 'LinkCount(Kopf) > 0 ';

if (vQ250 != "") then begin

vQ501 # vQ501 + ' AND ';

vQ501 # vQ501 + 'LinkCount(Art) > 0 ';

end;

// Selektionsquery für 500

vQ500 # ";

vQ500 # '(Ein.Vorgangstyp="" + aVorgang + " ';

vQ500 # vQ500 + ' AND "Ein~LiefervertragYN"=false';

// if (Con.Vertreternr != 0) then

// Lib\_Sel:QInt( var vQ500, 'Ein.Vertreter', '=', Con.Vertreternr);

Lib\_Sel:QVonBisD( var vQ500, 'Ein~Datum', vDat, vDat2);

// Selektion starten...

vSel # SelCreate(511, 1);

vSel->SelAddLink(", 510, 511, 3, 'Kopf');

if (vQ250<>") then

vSel->SelAddLink(", 250, 511, 2, 'Art');

Erx # vSel->SelDefQuery(", vQ501 );

```
if (Erx <> 0) then begin
```

```
    Lib_Sel:QError(vSel);
```

```
    RETURN;
```

```
end;
```

```
Erx # vSel->SelDefQuery('Kopf', vQ500 );
```

```
if (Erx <> 0) then begin
```

```
    Lib_Sel:QError(vSel);
```

```
    RETURN;
```

```
end;
```

```
if (vQ250<>'') then begin
```

```
    Erx # vSel->SelDefQuery('Art', vQ250 );
```

```
    if (Erx <> 0) then begin
```

```
        Lib_Sel:QError(vSel);
```

```
        RETURN;
```

```
    end;
```

```
end;
```

```
vSelName # Lib_Sel:SaveRun( var vSel, 0);
```

```
FOR Erx # RecRead(511,vSel,_recFirst)
```

```
LOOP Erx # RecRead(511,vSel,_recNext)
```

```
WHILE (Erx<=_rLocked) do begin
```

```
    Erx # RecLink(500,501,3,_recFirst); // Kopf holen
```

```
    RecBufcopy(511,501);
```

```
    RecbufCopy(510,500);
```

```

vKurs # 1.0;

if ("Ein.Währung"<>1) then begin
  if ("Ein.WährungFixYN") then begin
    vKurs # "Ein.Währungskurs";
  end
  else begin
    RecLink(814,500,8,_recfirst);  // Währung holen
    vKurs # "Wae.EK.Kurs";
  end;
end;

// Menge

if (Ein.P.MEH=Con.MEH) then
  vM # Ein.P.Menge
else if (Ein.P.MEH.Wunsch=Con.MEH) then
  vM # Ein.P.Menge.Wunsch
else if (Con.MEH='t') then
  vM # (Ein.P.Gewicht / 1000.0);

// Umsatz

vUms # Ein.P.Gesamtpreis;
vUms # Rnd(vUms / vKurs,2);

_AddWert(Ein.Datum, vM, vUms, 0.0);

END;

```

```
SelDelete(511, vSelName);
```

```
_Summieren();
```

```
end;
```

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

```
// _RecalcVK
```

```
//
```

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

```
sub _RecalcVK(
```

```
    aKdNr : int;
```

```
    aVon  : date;
```

```
    aBis  : date)
```

```
local begin
```

```
    vI   : int;
```

```
    vM   : float;
```

```
    vUms : float;
```

```
    vDB  : float;
```

```
    vX   : float;
```

```
    vErxCon, vErxStat : int;
```

```
    vMax  : int;
```

```
    vMaxB : int;
```

```
end;
```



begin

```
RecBufClear(899);

Sta.Re.Datum # aVon;

FOR vErxStat # RecRead(899,5,0);      // Statistik loopen

LOOP vErxStat # RecRead(899,5,_recNext);

WHILE (vErxStat<=_rNoKey) and (Sta.Re.Datum>=aVon) and (Sta.Re.Datum<=aBis) do begin

    if (Sta.Re.StornoRechNr<>0) then CYCLE;

    if (aKdNr<>0) and (aKdNr<>Sta.Auf.Kunden.Nr) then CYCLE;

    if (Con.Vertreternr<>0) and (Con.Vertreternr<>Sta.Re.Vertreter.Nr) then CYCLE;

    if (Con.Auftragsart<>0) and (Con.Auftragsart<>Sta.Auf.Auftragsart) then CYCLE;

    if (Con.Warengruppe<>0) and (Con.Warengruppe<>Sta.Auf.Warengruppe) then CYCLE;

    if (Con.Artikelgruppe<>0) and (Con.Artikelgruppe<>Sta.Auf.Artikelgrp) then CYCLE;

    // Artikel oder Güte ?

    if (Con.Dateinr = 200) then begin

//      if (Con.Artikelnummer="") and (mqu.güte='Stw24') then vok # n;

        if (Con.Artikelnummer<>") then begin

            // Güte lesen

            "MQu.Güte1" # "Con.Artikelnummer";

            if (Recread(832,2,0) > _rNokey) AND ("MQu.Güte1" <> "Con.Artikelnummer") then

                RecBufClear(832);

            // 17.01.2020 AH: dank VBS

//      if ((Con.Artikelnummer="") and ("MQu.ErsetzenDurch" <>") and ("MQu.ErsetzenDurch" <> "Sta.Auf.Gü
```

```

//      if ("MQu.ErsetzenDurch"="") and (Con.Artikelnummer<>") and (Con.Artikelnummer<>"Sta.Auf.Güte"
      if ("MQu.ErsetzenDurch"="") and (Con.Artikelnummer<>") and (Con.Artikelnummer<>"Sta.Auf.Güte"))

      if (MQu.ErsetzenDurch<>") and ("MQu.ErsetzenDurch"<>"Sta.Auf.Güte") then CYCLE;

end;

end

else

      if (Con.Artikelnummer<>") and (Con.Artikelnummer<>Sta.Auf.Artikel.Nr) then CYCLE;


// Menge

if (Sta.MEH.Einsatz=Con.MEH) then

      vM # Sta.Menge.Einsatz

else if (Sta.MEH.VK=Con.MEH) then

      vM # Sta.Menge.VK

else if (Sta.MEH.VK='kg') and (Con.MEH='t') then

      vM # (Sta.Menge.VK / 1000.0)

else if (Sta.MEH.VK='t') and (Con.MEH='kg') then

      vM # (Sta.Menge.VK * 1000.0);


// Umsatz

vUms # Sta.Betrag.VK + Sta.Aufpreis.VK + Sta.Korrektur.VK;


// DB

vDB # Sta.Betrag.VK + Sta.Aufpreis.VK + Sta.Korrektur.VK - Sta.Betrag.EK;

```

```
_AddWert(Sta.Re.Datum, vM, vUms, vDB);
```

```
END; // STAT Loop
```

```
_Summieren();
```

```
end;
```

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

```
//
```

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

```
sub Recalc(aAlle : logic;)
```

```
local begin
```

```
  vDat : date;
```

```
  vDat2 : date;
```

```
  vI : int;
```

```
  vErxCon, vErxStat : int;
```

```
  vMax : int;
```

```
  vMaxB : int;
```

```
end;
```

```
begin
```

```
  APPOFF();
```

```
  RecbufClear(100);
```

```
  vMax # RecInfo(950,_RecCount);
```

```
vMaxB # RecInfo(899,_RecCount);
```

```
if (aAlle=false) then vMax # 1;
```

```
vProgress # Lib_Progress:Init( 'Berechnung von ' + Aint(vMax)+' Positionen',vMax);
```

```
// Controlling loopen -----
```

```
if (aAlle) then
```

```
    vErxCn # RecRead(950,1,_RecFirst)
```

```
else
```

```
    vErxCn # RecRead(950,1,0);
```

```
WHILE (vErxCn<=_rLocked) do begin
```

```
    if (vProgress->Lib_Progress:Step() = false) then begin
```

```
        vProgress->Lib_Progress:Term();
```

```
        APPON();
```

```
        RETURN;
```

```
    end;
```

```
RecRead(950,1,_recLock);
```

```
FOR vl # 1 loop inc(vl) while (vl<=13) do begin
```

```
    FldDef(950,3,vl,0.0);
```

```
    FldDef(950,6,vl,0.0);
```

```
    FldDef(950,9,vl,0.0);
```

```
    FldDef(950,12,vl,0.0);
```

```
END;
```

```
if (Con.Adressnummer<>0) then begin
    if (RecLink(100,950,1,0) > _rLocked) then    // Adresse holen
        RecbufClear(100);
end;
```

```
vDat # DateMake(1,1, Con.Jahr-1900);
vDat2 # DateMake(31,12, Con.Jahr-1900);
```

```
case (Con.Typ) of
    " : _RecalcVK(Adr.KundenNr, vDat, vDat2);
    'A' : _RecalcEingangAuf(Adr.KundenNr, c_AUF);
    'B' : _RecalcEingangBest(Adr.LieferantenNr, c_Bestellung);
    'G' : _RecalcEingangAuf(Adr.KundenNr, c_ANG);
end;
```

```
// DATUM SETZEN
```

```
Con.Refreshdatum # today;
Con.Refreshzeit # now;
RekReplace(950,_RecUnlock,'AUTO');
```

```
if (aAlle=false) then BREAK;
vErxCn # RecRead(950,1,_RecNext);
END; // EO Loop Controlling DS
```

```
APPON();
```

```
vProgress->Lib_Progress:Term();
```

```
Msg(999998,"",0,0,0);
```

```
end;
```

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

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

```
sub AbweichungIstSoll(
```

```
    aDat    : date;
```

```
    aWas    : alpha;
```

```
    aTyp    : alpha;
```

```
    opt aAdr : int;
```

```
    opt aVer : int;
```

```
    opt aArt : alpha;
```

```
    opt aWgr : int;
```

```
    opt aAgr : int;
```

```
    opt aAufA : int) : float;
```

```
local begin
```

```
    v950 : int;
```

```
    vP   : float;
```

```
    vI   : int;
```

```
end;
```

```
begin
```

```
    v950 # RekSave(950);
```

```
    if (Con.Typ<>aTyp) or (Con.Jahr<>aDat->vpyear) or
```

(Con.Adressnummer<>aAdr) or (Con.Vertreternr<>aVer) or (Con.Artikelnummer<>aArt) or  
(Con.Warengruppe<>aWgr) or (Con.Artikelgruppe<>aAgr) or (Con.Auftragsart<>aAufA) then begin

```
v950->Con.Typ      # aTyp;
v950->Con.Jahr     # aDat->vpyear;
v950->Con.Adressnummer # aAdr;
v950->Con.Vertreternr # aVer;
v950->Con.Artikelnummer # aArt;
v950->Con.Warengruppe # aWgr;
v950->Con.Artikelgruppe # aAgr;
v950->Con.Auftragsart # aAufA;
vI # RecRead(v950, 1, 0);
if (vI>_rLocked) then RecBufClear(v950);
end;

vI # aDat->vpMonth;

// Ist / soll + 100
if (aWas='DB') then vP # Lib_berechnungen:Prozent(FldFloat(v950, 12, vI), FldFloat(v950, 11, vI));

RecBufDestroy(v950);

RETURN vP;
end;
```

//=====