LAMPIRAN

FormSORequisition.pas

```
unit FormSORequisition;

interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
  Vcl.Grids, AdvObj, BaseGrid, AdvGrid, DBAdvGrid, Vcl.Buttons,
  Vcl.StdCtrls, Vcl.Mask, AdvDropDown, AdvCustomGridDropDown,
  AdvGridDropDown, Data.DB, Data.Win.ADODB, TaskDialog,
  Vcl.ExtCtrls, Vcl.ComCtrls;
type
  TfrmSoRequisition = class(TForm)
    gbHeader: TGroupBox;
    btnSave: TSpeedButton;
    StaticText1: TStaticText;
    cbCustomer: TAdvGridDropDown;
    StaticText3: TStaticText;
    edtRequisition: TEdit;
    GroupBox1: TGroupBox;
    gbInfo: TGroupBox;
    dtpRequisition: TDateTimePicker;
    Edit2: TEdit;
    Edit4: TEdit;
    Edit3: TEdit;
    edtCustomerName: TEdit;
    Edit16: TEdit;
    txtStatus: TStaticText;
    Edit5: TEdit;
    edtAddress: TEdit;
    Edit6: TEdit;
    edtCity: TEdit;
    Edit9: TEdit;
    edtContactPerson: TEdit;
    Edit10: TEdit;
    edtPhone: TEdit;
    Edit14: TEdit;
    edtEmail: TEdit;
    gbInput: TGroupBox;
```

```
btnSaveDetail: TSpeedButton;
btnEditDetail: TSpeedButton;
btnDeleteDetail: TSpeedButton;
txtStatusDetail: TStaticText;
gbHasil: TGroupBox;
DBAdvGrid1: TDBAdvGrid;
edtTotal: TEdit;
qrySQL: TADOQuery;
GroupBox2: TGroupBox;
Edit11: TEdit;
edtStockCode: TEdit;
btnFind: TSpeedButton;
Edit12: TEdit;
edtDescription: TEdit;
Edit13: TEdit;
edtQty: TEdit;
edtUOM: TEdit;
qrySORTemp: TADOQuery;
dataTemp: TDataSource;
qrySORTempLine: TSmallintField;
qrySORTempStockCode: TWideStringField;
qrySORTempDescription: TWideStringField;
qrySORTempQty: TBCDField;
Edit15: TEdit;
edtRoll: TEdit;
qrySORTempRoll: TIntegerField;
edtFindSerial: TSpeedButton;
qrySQL2: TADOQuery;
Edit17: TEdit;
cbRibToStockCode: TAdvGridDropDown;
qrySORTempRibs: TWideStringField;
qrySORTempRibsToStockCode: TWideStringField;
edtType: TEdit;
rollstok: TEdit;
procedure FormShow(Sender: TObject);
procedure IsiComboBox;
procedure OpenQrySQL;
procedure Bersihkan;
procedure BersihkanInput;
procedure cbCustomerChange(Sender: TObject);
procedure DataCustomer;
procedure btnSaveDetailClick(Sender: TObject);
procedure SaveTemp;
```

```
    procedure btnFindClick(Sender: TObject);
    procedure edtQtyKeyPress(Sender: TObject; var Key: Char);
    procedure btnEditDetailClick(Sender: TObject);
    procedure btnDeleteDetailClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure ClearTemp;
    procedure btnSaveClick(Sender: TObject);
    procedure BuatNoRequisition;
    procedure UpdateNoRequisition;
    procedure SaveRequisition;
    procedure SaveMaster;
    procedure SaveDetail;
    procedure Tampilan;
    procedure DBAdvGrid1Click(Sender: TObject);
    procedure edtFindSerialClick(Sender: TObject);
    procedure GetRibToStock;
    procedure IsiComboRib;
    procedure edtRollKeyPress(Sender: TObject; var Key: Char);
    function CekInput : Boolean;
    function CekAll : Boolean;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmSoRequisition: TfrmSoRequisition;
  NextRequisition, Requisition, temp, status : String;
  NextSuffixRequisition, NoUrut, Line, i, simpan, totaldata : Integer;
  RollOnHand, RollAllocated, RollAvailable, tempRoll, orderRoll : Integer;
  QtyOnHand, QtyAllocated, QtyAvailable, tempQty, orderQty : Double;

implementation
uses  FormComponent, GlobalUnit, Numbering, FormStockCodes,
FormListSOR,
      FormSerialNumberChoose;

{$R *.dfm}
procedure TfrmSoRequisition.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  if (not (qrySORTemp.IsEmpty)) and (status <> 'simpan')then
```

```
  begin
    if MessageDlg('You want to Cancel Creating Requisition?',
       mtConfirmation, [mbYes,mbNo], 0, mbYes) = mrYes then
    begin
      ClearTemp;
      qrySORTemp.Refresh;
      qrySORTemp.Close;
    end
    else Action := caNone;
  end;
end;

procedure TfrmSoRequisition.ClearTemp;
begin
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
             '[SoRequisitionTemp]';
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
             '[InvSerialNumberTemp]';
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
end;

procedure TfrmSoRequisition.FormShow(Sender: TObject);
begin
  Bersihkan;
  if (txtStatus.Caption = 'OPEN') then
  begin
    ClearTemp;
    qrySORTemp.Close;
    qrySORTemp.Open;
    edtRequisition.Text            := 'NEW';
```

```
    cbCustomer.Text              := '';
    cbCustomer.Enabled           := True;
    cbCustomer.SetFocus;
    dtpRequisition.DateTime      := Now;
    edtFindSerial.Enabled        := False;
    GroupBox1.Enabled            := False;
  end
  else
  begin
    edtRequisition.Text          :=
frmListSOR.qrySORMs.FieldByName('Requisition').AsString;
    cbCustomer.Text              := frmListSOR.cbCustomer.Text;
    cbCustomer.Enabled           := False;
    dtpRequisition.DateTime      :=
frmListSOR.qrySORMs.FieldByName('RequisitionDate').AsDateTime;
    dtpRequisition.Enabled       := False;
    edtFindSerial.Enabled        := False;
    btnFind.Enabled              := False;
    edtRoll.Enabled              := False;
    gbInfo.Enabled               := True;
    gbInput.Enabled              := False;
    btnSaveDetail.Enabled        := False;
    DataCustomer;
    Tampilan;
  end;
  IsiComboBox;
end;

procedure TfrmSoRequisition.Bersihkan;
begin
  edtCustomerName.Text  := '';
  edtAddress.Text       := '';
  edtCity.Text          := '';
  edtContactPerson.Text := '';
  edtPhone.Text         := '';
  edtEmail.Text         := '';
  BersihkanInput;
  simpan    := 0;
  totaldata := 0;
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
            ' [InvSerialNumberTemp]';
  with qrySQL do
  begin
```

```
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
  btnSave.Enabled              := False;
  btnEditDetail.Enabled        := False;
  btnDeleteDetail.Enabled      := False;
end;

procedure TfrmSoRequisition.BersihkanInput;
begin
  edtStockCode.Text            := '';
  cbRibToStockCode.Text        := '';
  edtDescription.Text          := '';
  rollstok.Text                := '';
  edtRoll.Text                 := '';
  edtQty.Text                  := '';
  edtType.Text                 := '';
  edtUOM.Text                  := '';
  txtStatusDetail.Caption      := 'NEW';
  cbRibToStockCode.Enabled:= False;
  edtRoll.Enabled              := False;
end;

procedure TfrmSoRequisition.btnDeleteDetailClick(Sender: TObject);
var Jawab : Integer;
begin
  if (qrySORTemp.IsEmpty) then
  begin
    MessageDlg('Table is empty!', mtError, [mbOK], 0);
  end
  else
  begin
    Jawab :=  MessageDlgPos('Are you sure want to delete this order?',
              mtConfirmation, mbOKCancel, 0,Left + 300, Top + 230);
    if (Jawab = mrOK) then
    begin
      qrySORTemp.Delete;
    end;
    txtStatusDetail.Caption := 'NEW';
    BersihkanInput;
  end;
```

```
end;

procedure TfrmSoRequisition.btnEditDetailClick(Sender: TObject);
var i : Integer;
begin
  if not(qrySORTemp.IsEmpty) then
  begin
    txtStatusDetail.Caption   := 'EDIT';
    btnEditDetail.Enabled     := False;
    btnDeleteDetail.Enabled := False;
    btnSaveDetail.Enabled     := True;
    with qrySORTemp do
    begin
      NoUrut                        := FieldByName('Line').AsInteger;
      edtStockCode.Text             := FieldByName('StockCode').AsString;
      edtRoll.Text                  := FieldByName('Roll').AsString;
      edtQty.Text                   := FieldByName('Qty').AsString;
      if (txtStatus.Caption = 'OPEN') then
      begin
        if FieldByName('Ribs').AsString = 'Y' then
        begin
          IsiComboRib;
          i := 0;
          repeat
            cbRibToStockCode.ItemIndex := i;
            i := i+1;
          until ((cbRibToStockCode.Text =
                 FieldByName('RibsToStockCode').AsString)
                 or (i > cbRibToStockCode.Items.Count));
        end
        else
        begin
          cbRibToStockCode.Clear;
          cbRibToStockCode.Enabled := False;
        end;
      end
        else
        begin
          cbRibToStockCode.Enabled := False;
          cbRibToStockCode.Text := FieldByName('RibsToStockCode').AsString;
          edtFindSerial.Enabled   := True;
        end;
    end;
```

```
    temp := 'SELECT a.Satuan, a.Type, a.Description, b.RollAvailable' +
            ' FROM ' + frmComponent.lblDB.Caption +
            '[InvMaster] a, [InvWarehouse] b ' +
            'WHERE  a.StockCode = b.StockCode AND a.StockCode = ' +
            QuotedStr(edtStockCode.Text);
    OpenQrySQL;
    with qrySQL do
    begin
      edtType.Text     := FieldByName('Type').AsString;
      edtUOM.Text   := FieldByName('Satuan').AsString;
      edtDescription.Text := FieldByName('Description').AsString;
      rollstok.Text := IntToStr(qrySQL.FieldByName('RollAvailable').AsInteger);
    end;
    qrySQL.Close;
    temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
            ' [SoRequisitionTemp] WHERE Line = ' + IntToStr(NoUrut);
    with qrySQL do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      ExecSQL;
    end;
    qrySORTemp.Close;
    qrySORTemp.Open;
    if txtStatus.Caption = 'OPEN' then
      edtRoll.Enabled := True;
  end;
end;

procedure TfrmSoRequisition.btnFindClick(Sender: TObject);
begin
  frmStockCodes.lblAsal.Caption := 'SOR';
  frmStockCodes.ShowModal;
  GetRibToStock;
  edtQty.Text := '0';
  edtRoll.Text:= '0';
  edtRoll.Enabled:= True;
  edtRoll.SetFocus;
end;

procedure TfrmSoRequisition.GetRibToStock;
begin
```

```
  temp := 'SELECT ProductClass FROM ' + frmComponent.lblDB.Caption+
            '[InvMaster] WHERE StockCode = ' +
            QuotedStr(edtStockCode.Text);
  OpenQrySQL;
  if qrySQL.FieldByName('ProductClass').AsString = 'R'  then
  begin
    cbRibToStockCode.Enabled := True;
    IsiComboRib;
  end
  else
  begin
    cbRibToStockCode.Items.Clear;
    cbRibToStockCode.Enabled := False;
  end;
end;

procedure TfrmSoRequisition.IsiComboRib;
begin
   cbRibToStockCode.Items.Clear;
   temp := 'SELECT DISTINCT StockCode, Description FROM ' +
             frmComponent.lblDB.Caption +
             '[SoRequisitionTemp] WHERE Ribs = ' + QuotedStr('N');
   OpenQrySQL;
   while not(qrySQL.Eof) do
   begin
     with cbRibToStockCode.Items.Add do
     begin
      Text.Add(qrySQL.FieldByName('StockCode').AsString);
      Text.Add(qrySQL.FieldByName('Description').AsString);
     end;
     qrySQL.Next;
   end;
   cbRibToStockCode.Enabled := True;
end;

procedure TfrmSoRequisition.btnSaveClick(Sender: TObject);
begin
  if (txtStatus.Caption = 'OPEN') then
  begin
    BuatNoRequisition;
  end;
  if (CekAll) then
  begin
```

```
    SaveRequisition;
  end;
end;

procedure TfrmSoRequisition.BuatNoRequisition;
var Panjang : Integer;
    Prefix  : String;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[SetupCode] WHERE [Transaction] = ' +
            QuotedStr('SO Requisition');
  OpenQrySQL;
  with qrySQL do
  begin
    Prefix              := FieldByName('Prefix').AsString;
    Panjang             := FieldByName('Length').AsInteger;
    Requisition         := FieldByName('Next').AsString;
    NextSuffixRequisition := (FieldByName('Suffix').AsInteger + 1);
    NextRequisition  :=
UpdateCode(IntToStr(NextSuffixRequisition),Prefix,Panjang);
    edtRequisition.Text := Requisition;
  end;
end;

procedure TfrmSoRequisition.SaveRequisition;
begin
  if not (frmComponent.adoWingga.InTransaction) then
  begin
    frmComponent.adoWingga.BeginTrans;
    try
      SaveMaster;
      SaveDetail;
      frmComponent.adoWingga.CommitTrans;
      MessageDlg('Save data success !'+#13#10+'SOR number : ' +
      edtRequisition.Text, mtConfirmation, [mbOK], 0);
      if (txtStatus.Caption = 'OPEN') then
      begin
        UpdateNoRequisition;
      end;
      frmListSOR.cbCustomer.Text:= cbCustomer.Text;
      if (txtStatus.Caption = 'OPEN') then
      begin
        frmListSOR.cbStatus.ItemIndex := 0;
```

```
      end
      else
      begin
        frmListSOR.cbStatus.ItemIndex := 1;
        frmListSOR.btnIsiQty.Enabled := False;
      end;
      frmListSOR.RefreshMaster;
      frmListSOR.RefreshDetail;
      frmListSOR.edtFind.Text   := edtRequisition.Text;
      ClearTemp;
      status := 'simpan';
      PostMessage(Self.Handle,wm_close,0,0);
    Except
      frmComponent.adoWingga.RollbackTrans;
      MessageDlg('Save data failed !', mtError, [mbOK], 0);
    end; // end try
  end; // end if ado.InTrans
end;

procedure TfrmSoRequisition.SaveMaster;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[SoRequisitionMaster] WHERE Requisition = ' +
             QuotedStr(edtRequisition.Text);
  OpenQrySQL;
  with qrySQL do
  begin
    if (txtStatus.Caption = 'OPEN') then
    begin
      Insert;
      FieldByName('Requisition').AsString             := edtRequisition.Text;
      FieldByName('RequisitionDate').AsDateTime:= dtpRequisition.Date;
      FieldByName('Customer').AsString                := cbCustomer.Text;
      FieldByName('Status').AsString                  := 'OPEN';
      FieldByName('AddID').AsString             :=
frmComponent.lblUserName.Caption;
    end
    else if (txtStatus.Caption = 'EDIT') then
    begin
      Edit;
      FieldByName('Status').AsString           := 'COMPLETE';
      FieldByName('EditID').AsString           :=
frmComponent.lblUserName.Caption;
```

```
    end;
    Post;
  end;
  Close;
end;

procedure TfrmSoRequisition.SaveDetail;
begin
  if (txtStatus.Caption = 'EDIT') then
  begin
    with qrySQL do
    begin
      temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
                '[SoRequisitionDetail] WHERE Requisition = ' +
                QuotedStr(edtRequisition.Text);
      Close;
      SQL.Clear;
      SQL.Add(temp);
      ExecSQL;
    end;
  end;
  Line := 1;
  qrySORTemp.First;
  while not(qrySORTemp.Eof) do
  begin
    temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[SoRequisitionDetail] WHERE Requisition = ' +
              QuotedStr(edtRequisition.Text);
    OpenQrySQL;
    with qrySQL do
    begin
      Insert;
      FieldByName('Requisition').AsString:= edtRequisition.Text;
      FieldByName('Line').AsInteger         := Line;
      FieldByName('StockCode').AsString  :=
qrySORTemp.FieldByName('StockCode').AsString;
      FieldByName('RibsToStockCode').AsString:=
qrySORTemp.FieldByName('RibsToStockCode').AsString;
      FieldByName('Roll').AsInteger      :=
qrySORTemp.FieldByName('Roll').AsInteger;
      FieldByName('Qty').AsFloat         :=
qrySORTemp.FieldByName('Qty').AsFloat;
      Post;
```

```
end;
qrySQL.Close;
if (txtStatus.Caption = 'OPEN') then
begin
  orderRoll := qrySORTemp.FieldByName('Roll').AsInteger;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[InvWarehouse] WHERE StockCode = ' +
            QuotedStr(qrySORTemp.FieldByName('StockCode').AsString);
  with qrySQL2 do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
    RollOnHand   := FieldByName('RollOnHand').AsInteger;
    RollAllocated := FieldByName('RollAllocated').AsInteger;
    RollAvailable := FieldByName('RollAvailable').AsInteger;
    Edit;
    tempRoll := RollAllocated + OrderRoll;
    FieldByName('RollAllocated').AsFloat := tempRoll;
    FieldByName('RollAvailable').AsFloat := RollOnHand - tempRoll;
    Post;
  end;
  qrySQL2.Close;
end
else
begin
  orderQty := qrySORTemp.FieldByName('Qty').AsFloat;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[InvWarehouse] WHERE StockCode = ' +
            QuotedStr(qrySORTemp.FieldByName('StockCode').AsString);
  with qrySQL2 do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
    QtyOnHand   := FieldByName('QtyOnHand').AsFloat;
    QtyAllocated := FieldByName('QtyAllocated').AsFloat;
    QtyAvailable := FieldByName('QtyAvailable').AsFloat;
    Edit;
    tempQty := QtyAllocated + OrderQty;
    FieldByName('QtyAllocated').AsFloat := tempQty;
```

```
      FieldByName('QtyAvailable').AsFloat := QtyOnHand - tempQty;
      Post;
    end;
    qrySQL2.Close;
    qrySQL.Close;
  end;
  Line := Line + 1;
  qrySORTemp.Next;
end;
temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
          '[InvSerialNumberTemp]';
OpenQrySQL;
qrySQL.First;
while not(qrySQL.Eof) do
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[InvSerialNumberHistory] WHERE SerialNumber = ' +
            QuotedStr(qrySQL.FieldByName('SerialNumber').AsString);
  with qrySQL2 do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
    Insert;
    FieldByName('Date').AsDateTime              := frmComponent.GetDate();
    FieldByName('SerialNumber').AsString      :=
qrySQL.FieldByName('SerialNumber').AsString;
    FieldByName('StockCode').AsString          :=
qrySQL.FieldByName('StockCode').AsString;
    for i := 1 to 24 do
      FieldByName('Qty'+IntToStr(i)).AsFloat  :=
qrySQL.FieldByName('Qty'+IntToStr(i)).AsFloat;
    FieldByName('SalesOrderRequisition').AsString:= edtRequisition.Text;
    Post;
  end;
  qrySQL2.Close;
  temp := 'SELECT * FROM InvSerialNumber WHERE SerialNumber ='+
            QuotedStr(qrySQL.FieldByName('SerialNumber').AsString);
  with qrySQL2 do
  begin
    Close;
    SQL.Clear;
```

```
      SQL.Add(temp);
      Open;
      Edit;
      for i := 1 to 24 do
      begin
        if qrySQL.FieldByName('Qty' + IntToStr(i)).AsFloat <> 0 then
        begin
          FieldByName('Status' + IntToStr(i)).AsString := 'ALLOCATED';
        end;
      end;
      Post;
    end;
    qrySQL2.Close;
    qrySQL.Next;
  end;
end;

procedure TfrmSoRequisition.UpdateNoRequisition;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[SetupCode] WHERE [Transaction] = ' + QuotedStr('SO Requisition');
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
    Edit;
    FieldByName('Next').AsString    := NextRequisition;
    FieldByName('Suffix').AsInteger := NextSuffixRequisition;
    Post;
  end;
end;

procedure TfrmSoRequisition.btnSaveDetailClick(Sender: TObject);
begin
  if (CekInput) then
  begin
    SaveTemp;
    simpan := simpan + 1;
    cbCustomer.Enabled := False;
    BersihkanInput;
    btnEditDetail.Enabled     := True;
```

```
    btnDeleteDetail.Enabled := True;
    if (simpan = totaldata) and (txtStatus.Caption<> 'OPEN') then
    begin
      btnSave.Enabled := True;
      edtFindSerial.Enabled   := False;
    end
    else if (txtStatus.Caption = 'OPEN') then
    begin
      btnSave.Enabled := True;
    end
    else if (txtStatus.Caption <> 'OPEN') then
    begin
      edtFindSerial.Enabled   := False;
    end;
  end;
end;

procedure TfrmSoRequisition.SaveTemp;
begin
  with qrySORTemp do
  begin
    Close;
    Open;
    if (not(IsEmpty)) then
    begin
      if (txtStatusDetail.Caption = 'NEW') then
      begin
        Last;
        NoUrut := FieldByName('Line').AsInteger + 1;
      end
      else if (txtStatusDetail.Caption = 'EDIT') then
      begin
        NoUrut := NoUrut;
      end;
    end
    else if (IsEmpty) then
      NoUrut := 1;
    Insert;
    FieldByName('Line').AsInteger       := NoUrut;
    FieldByName('StockCode').AsString   := edtStockCode.Text;
    FieldByName('Description').AsString := edtDescription.Text;
    if cbRibToStockCode.Text <> '' then
    begin
```

```
      FieldByName('Ribs').AsString            := 'Y';
      FieldByName('RibsToStockCode').AsString := cbRibToStockCode.Text;
    end
    else
      FieldByName('RibsToStockCode').AsString := '-';
    FieldByName('Roll').AsInteger       := StrToInt(edtRoll.Text);
    FieldByName('Qty').AsFloat          := StrToFloat(edtQty.Text);
    Post;
    Refresh;
  end;
end;

procedure TfrmSoRequisition.cbCustomerChange(Sender: TObject);
begin
  GroupBox1.Enabled := True;
  DataCustomer;
end;

procedure TfrmSoRequisition.DataCustomer;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[ArCustomer] WHERE CustomerCode = ' +
             QuotedStr(cbCustomer.Text);
  OpenQrySQL;

  edtCustomerName.Text  := qrySQL.FieldByName('CustomerName').AsString;
  edtAddress.Text           := qrySQL.FieldByName('Address').AsString;
  edtCity.Text                 := qrySQL.FieldByName('City').AsString;
  edtContactPerson.Text :=   qrySQL.FieldByName('ContactPerson').AsString;
  edtPhone.Text         := qrySQL.FieldByName('Phone').AsString;
  edtEmail.Text         := qrySQL.FieldByName('Email').AsString;
  qrySQL.Close;
end;

procedure TfrmSoRequisition.DBAdvGrid1Click(Sender: TObject);
begin
  if not qrySORTemp.Eof then
  begin
    gbInput.Enabled         := True;
    btnEditDetail.Enabled   := True;
    btnDeleteDetail.Enabled := True;
  end;
end;
```

```
procedure TfrmSoRequisition.IsiComboBox;
begin
  cbCustomer.Items.Clear;
  temp := 'SELECT CustomerCode, CustomerName FROM ' +
             frmComponent.lblDB.Caption +
             '[ArCustomer] ORDER BY CustomerCode';
  OpenQrySQL;
  if (qrySQL.Eof) then
  begin
    Messagedlg('Customer is empty. Ask your administrator!', mtError,
[mbok],0);
    Close;
  end;
  while not (qrySQl.Eof) do
  begin
    with cbCustomer.Items.Add do
    begin
      Text.Add(qrySQL.FieldByName('CustomerCode').AsString);
      Text.Add(qrySQL.FieldByName('CustomerName').AsString);
    end;
    qrySQL.Next;
  end;
  Close;
end;

procedure TfrmSoRequisition.OpenQrySQL;
begin
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;
end;

function TfrmSoRequisition.CekInput : Boolean;
begin
  CekInput := True;
  if (StringReplace(edtStockCode.Text, ' ', '',
    [rfReplaceAll, rfIgnoreCase]) = '') then
  begin
    MessageDlg('Stock Code required!', mtError, [mbOK], 0);
```

```
    edtStockCode.Text := '';
    CekInput := False;
  end
  else if ((txtStatus.Caption <> 'OPEN') and (edtQty.Text = '')) or
        ((txtStatus.Caption <> 'OPEN') and (edtQty.Text = '0')) then
  begin
    MessageDlg('Quantity required!', mtError, [mbOK], 0);
    CekInput := False;
  end
  else if ((txtStatus.Caption = 'OPEN') and (edtRoll.Text = '')) or
        ((txtStatus.Caption = 'OPEN') and (edtRoll.Text = '0')) then
  begin
    MessageDlg('Roll required!', mtError, [mbOK], 0);
    edtRoll.SetFocus;
    CekInput := False;
  end
  else if (txtStatus.Caption = 'OPEN') and
        ((StrToInt(edtRoll.Text)) > (StrToInt(rollstok.Text))) then
  begin
    MessageDlg('Maaf, hanya ada '+rollstok.Text+' roll yang tersedia!', mtError,
[mbOK], 0);
    edtRoll.Text := '0';
    CekInput := False;
  end;
end;
function TfrmSoRequisition.CekAll : Boolean;
var
Jawab: Integer;
begin
  CekAll := True;
  if (StringReplace(cbCustomer.Text, ' ', '', [rfReplaceAll, rfIgnoreCase]) = '')
then
  begin
    MessageDlg('Customer required!', mtError, [mbOK], 0);
    cbCustomer.SetFocus;
    CekAll := False;
  end
  else if (qrySORTemp.IsEmpty) then
  begin
    MessageDlg('Data is empty!', mtError, [mbOK], 0);
    CekAll := False;
  end
  else
```

```
  begin
    Jawab := MessageDlg('Do you want to save this requisition?',
mtConfirmation,
      mbOKCancel, 0);
    if Jawab = mrCancel then
    begin
      CekAll := False;
    end;
  end;
end;

procedure TfrmSoRequisition.edtFindSerialClick(Sender: TObject);
begin
  for i := 1 to 24 do
  begin
    TEdit(frmSerialNumberChoose.FindComponent('edtQty' +
IntToStr(i))).Clear;
    TCheckBox(frmSerialNumberChoose.FindComponent('CheckBox' +
IntToStr(i))).Enabled := False;
  end;
  frmSerialNumberChoose.lblAsal.Caption          := 'INVOUT';
  frmSerialNumberChoose.edtStockCode.Text        := edtStockCode.Text;
  frmSerialNumberChoose.txtTotal.Caption         := edtQty.Text;
  frmSerialNumberChoose.txtRoll.Caption          := edtRoll.Text;
  frmSerialNumberChoose.ShowModal;
end;

procedure TfrmSoRequisition.edtQtyKeyPress(Sender: TObject; var Key: Char);
begin
  if not (Key in [#8, '0'..'9', '.']) then
  begin
    Key := #0;
  end
  else if (Key = '.') and (Pos(Key, edtQty.Text) > 0) then
  begin
    Key := #0;
  end;
end;

procedure TfrmSoRequisition.edtRollKeyPress(Sender: TObject; var Key:
Char);
var DecimalSeparator : char;
begin
```

```
  if not (Key in [#8, '0'..'9', '.']) then
  begin
    Key := #0;
  end
  else if (Key = '.') and (Pos(Key, edtRoll.Text) > 0) then
  begin
    Key := #0;
  end;
end;

procedure TfrmSoRequisition.Tampilan;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[SoRequisitionDetail] WHERE Requisition = ' +
              QuotedStr(edtRequisition.Text);
  OpenQrySQL;
  if not qrySQL.Eof then
  begin
    qrySQL.First;
    while not qrySQl.Eof do
    begin
      with qrySORTemp do
      begin
        Close;
        Open;
        Insert;
        FieldByName('Line').AsInteger       :=
qrySQL.FieldByName('Line').AsInteger;
        FieldByName('StockCode').AsString   :=
qrySQL.FieldByName('StockCode').AsString;
        FieldByName('RibsToStockCode').AsString:=
qrySQL.FieldByName('RibsToStockCode').AsString;
        FieldByName('Roll').AsInteger       :=
qrySQL.FieldByName('Roll').AsInteger;
        FieldByName('Qty').AsFloat          := qrySQL.FieldByName('Qty').AsFloat;
        Post;
        Close;
        Open;
      end;
      totaldata := totaldata + 1;
      qrySQL.Next;
    end;
  end;
```

```
end;
end.
```

FormSalesOrder.pas

```
unit FormSalesOrder;

interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
  frxClass, Data.DB, Data.Win.ADODB, frxDBSet, Vcl.StdCtrls, Vcl.Grids,
  AdvObj, BaseGrid, AdvGrid, DBAdvGrid,Vcl.ComCtrls, Vcl.Mask,
  AdvDropDown, AdvCustomGridDropDown, AdvGridDropDown,
  Vcl.Buttons, Vcl.ExtCtrls, StrUtils;
type
  TfrmSalesOrder = class(TForm)
    gbHeader: TGroupBox;
    btnSave: TSpeedButton;
    btnPost: TSpeedButton;
    btnDO: TSpeedButton;
    StaticText1: TStaticText;
    dtpOrder: TDateTimePicker;
    StaticText2: TStaticText;
    gbHeader2: TGroupBox;
    edtSO: TEdit;
    StaticText3: TStaticText;
    GroupBox1: TGroupBox;
    Edit11: TEdit;
    edtStockCode: TEdit;
    Edit3: TEdit;
    edtDescription: TEdit;
    Edit16: TEdit;
    Edit40: TEdit;
    btnFindSerial: TSpeedButton;
    Edit5: TEdit;
    edtQty: TEdit;
    Edit14: TEdit;
    edtPrice: TEdit;
    Memo3: TMemo;
    Edit12: TEdit;
    Edit13: TEdit;
```

```
rbPersenDetail: TRadioButton;
rbAmountDetail: TRadioButton;
Edit7: TEdit;
edtDiskonDetail: TEdit;
btnSaveDetail: TSpeedButton;
btnEditDetail: TSpeedButton;
GroupBox2: TGroupBox;
Edit15: TEdit;
Edit19: TEdit;
edtDiskonHeader: TEdit;
Edit8: TEdit;
Edit9: TEdit;
chkTax: TCheckBox;
Edit17: TEdit;
edtWarna: TEdit;
edtViewWarna: TEdit;
GroupBox3: TGroupBox;
Edit27: TEdit;
Edit37: TEdit;
edtSupir: TEdit;
edtKenek: TEdit;
edtDO: TEdit;
gbDetail: TGroupBox;
Splitter1: TSplitter;
DBAdvGrid1: TDBAdvGrid;
DBAdvGrid2: TDBAdvGrid;
GroupBox5: TGroupBox;
Label3: TLabel;
edtTotal: TEdit;
qryDtl: TADOQuery;
qryDtlLine: TSmallintField;
qryDtlStockCode: TWideStringField;
qryDtlRibsToStockCode: TWideStringField;
qryDtlQty: TBCDField;
qryDtlPrice: TBCDField;
qryDtlDiskon: TBCDField;
qryDtlTotal: TFMTBCDField;
qryDtlDiskonPersen: TBCDField;
qryDtlDiskonAmount: TBCDField;
ds: TDataSource;
edtSOR: TEdit;
edtCustomer: TEdit;
qrySQL: TADOQuery;
```

```
qrySQL2: TADOQuery;
qrySQL3: TADOQuery;
qryDtlRoll: TIntegerField;
edtRibToStockCode: TEdit;
edtUOM: TEdit;
Edit2: TEdit;
edtRoll: TEdit;
dsDtlDelivery: TDataSource;
qryDtlDelivery: TADOQuery;
qryDtlDeliveryDeliveryLine: TSmallintField;
qryDtlDeliverySerialNumber: TWideStringField;
qryDtlDeliveryQty1: TFMTBCDField;
qryDtlDeliveryQty2: TFMTBCDField;
qryDtlDeliveryQty3: TFMTBCDField;
qryDtlDeliveryQty4: TFMTBCDField;
qryDtlDeliveryQty5: TFMTBCDField;
qryDtlDeliveryQty6: TFMTBCDField;
qryDtlDeliveryQty7: TFMTBCDField;
qryDtlDeliveryQty8: TFMTBCDField;
qryDtlDeliveryQty9: TFMTBCDField;
qryDtlDeliveryQty10: TFMTBCDField;
qryDtlDeliveryQty11: TFMTBCDField;
qryDtlDeliveryQty12: TFMTBCDField;
qryDtlDeliveryQty13: TFMTBCDField;
qryDtlDeliveryQty14: TFMTBCDField;
qryDtlDeliveryQty15: TFMTBCDField;
qryDtlDeliveryQty16: TFMTBCDField;
qryDtlDeliveryQty17: TFMTBCDField;
qryDtlDeliveryQty18: TFMTBCDField;
qryDtlDeliveryQty19: TFMTBCDField;
qryDtlDeliveryQty20: TFMTBCDField;
qryDtlDeliveryQty21: TFMTBCDField;
qryDtlDeliveryQty22: TFMTBCDField;
qryDtlDeliveryQty23: TFMTBCDField;
qryDtlDeliveryQty24: TFMTBCDField;
qryDtlDeliveryLine: TSmallintField;
qryDtlDeliveryDeliveryQty: TBCDField;
qryDtlDeliveryPrice: TBCDField;
qryDtlDeliveryTotal: TBCDField;
qryDtlDeliveryViewWarna: TWideStringField;
txtStatus: TStaticText;
qryRepTemp: TADOQuery;
qryRepTempSerialNumber: TWideStringField;
```

```
qryRepTempQty1: TFMTBCDField;
qryRepTempQty2: TFMTBCDField;
qryRepTempQty3: TFMTBCDField;
qryRepTempQty4: TFMTBCDField;
qryRepTempQty5: TFMTBCDField;
qryRepTempQty6: TFMTBCDField;
qryRepTempQty7: TFMTBCDField;
qryRepTempQty8: TFMTBCDField;
qryRepTempQty9: TFMTBCDField;
qryRepTempQty10: TFMTBCDField;
qryRepTempQty11: TFMTBCDField;
qryRepTempQty12: TFMTBCDField;
qryRepTempQty13: TFMTBCDField;
qryRepTempQty14: TFMTBCDField;
qryRepTempQty15: TFMTBCDField;
qryRepTempQty16: TFMTBCDField;
qryRepTempQty17: TFMTBCDField;
qryRepTempQty18: TFMTBCDField;
qryRepTempQty19: TFMTBCDField;
qryRepTempQty20: TFMTBCDField;
qryRepTempQty21: TFMTBCDField;
qryRepTempQty22: TFMTBCDField;
qryRepTempQty23: TFMTBCDField;
qryRepTempQty24: TFMTBCDField;
qryRepTempDeliveryQty: TBCDField;
qryRepTempStockCode: TWideStringField;
qryRepTempViewWarna: TWideStringField;
qryRepTempStockCodeRib: TWideStringField;
qryRepTempViewWarnaRib: TWideStringField;
qryRepTempDeliveryQtyRib: TBCDField;
qryRepTempQtyRib1: TFMTBCDField;
qryRepTempQtyRib2: TFMTBCDField;
qryRepTempQtyRib3: TFMTBCDField;
qryRepTempQtyRib4: TFMTBCDField;
qryRepTempQtyRib5: TFMTBCDField;
qryRepTempQtyRib6: TFMTBCDField;
qryRepTempQtyRib7: TFMTBCDField;
qryRepTempQtyRib8: TFMTBCDField;
qryRepTempQtyRib9: TFMTBCDField;
qryRepTempQtyRib10: TFMTBCDField;
qryRepTempQtyRib11: TFMTBCDField;
qryRepTempQtyRib12: TFMTBCDField;
qryRepTempQtyRib13: TFMTBCDField;
```

```
qryRepTempQtyRib14: TFMTBCDField;
qryRepTempQtyRib15: TFMTBCDField;
qryRepTempQtyRib16: TFMTBCDField;
qryRepTempQtyRib17: TFMTBCDField;
qryRepTempQtyRib18: TFMTBCDField;
qryRepTempQtyRib19: TFMTBCDField;
qryRepTempQtyRib20: TFMTBCDField;
qryRepTempQtyRib21: TFMTBCDField;
qryRepTempQtyRib22: TFMTBCDField;
qryRepTempQtyRib23: TFMTBCDField;
qryRepTempQtyRib24: TFMTBCDField;
qryRepTempPrice: TBCDField;
qryRepTempPriceRib: TBCDField;
qryRepTempDelivery: TWideStringField;
qryRep2: TADOQuery;
qryRep2SalesOrder: TWideStringField;
qryRep2Line: TSmallintField;
qryRep2StockCode: TWideStringField;
qryRep2RibsToStockCode: TWideStringField;
qryRep2SerialNumber: TWideStringField;
qryRep2OrderQty: TBCDField;
qryRep2DeliveryQty: TBCDField;
qryRep2ReturnQty: TBCDField;
qryRep2Price: TBCDField;
qryRep2DiskonPersen: TBCDField;
qryRep2DiskonAmount: TBCDField;
qryRep2Diskon: TBCDField;
qryRep2TotalOrder: TFMTBCDField;
qryRep2GrandTotal: TFMTBCDField;
qryRep2TimeStamp: TBytesField;
qryRep3: TADOQuery;
qryRep3SalesOrder: TWideStringField;
qryRep3OrderDate: TDateTimeField;
qryRep3CustomerCode: TWideStringField;
qryRep3OrderTotal: TBCDField;
qryRep3DiskonPersen: TBCDField;
qryRep3Diskon: TBCDField;
qryRep3TotalDiskon: TBCDField;
qryRep3Tax: TBCDField;
qryRep3Taxable: TStringField;
qryRep3GrandTotal: TFMTBCDField;
qryRep3TermsCode: TStringField;
qryRep3Status: TWideStringField;
```

```
    qryRep3Warna: TWideStringField;
    qryRep3Supir: TWideStringField;
    qryRep3Kenek: TWideStringField;
    qryRep3AddID: TWideStringField;
    qryRep3TimeStamp: TBytesField;
    frxDBDatasetRepTemp: TfrxDBDataset;
    frxDBDatasetRep2: TfrxDBDataset;
    frxDBDatasetRep3: TfrxDBDataset;
    frxRepDel: TfrxReport;
    dtpNow: TDateTimePicker;
    edtType: TEdit;
    cbTerm: TComboBox;
    Edit1: TEdit;
    qryDtlDeliveryStockCode: TWideStringField;
    qryDtlDeliveryRibsToStockCode: TWideStringField;
    qryRepTempLine: TIntegerField;
    edtWarna2: TEdit;
    procedure FormShow(Sender: TObject);
    procedure Tampilan;
    procedure Bersihkan;
    procedure BersihkanInput;
    procedure BersihkanTemp;
    procedure OpenQrySQL;
    procedure OpenQrySQL2;
    procedure OpenQrySQL3;
    procedure btnEditDetailClick(Sender: TObject);
    procedure CekDiskonEdit;
    procedure btnSaveDetailClick(Sender: TObject);
    procedure CreateHarga;
    procedure SaveTemp;
    procedure btnFindSerialClick(Sender: TObject);
    procedure Anjuran;
    procedure OpenQryTemp;
    procedure rbPersenDetailClick(Sender: TObject);
    procedure rbAmountDetailClick(Sender: TObject);
    procedure DBAdvGrid2ClickCell(Sender: TObject; ARow, ACol: Integer);
    procedure BuatNoDelivery;
    procedure UpdateNoDelivery;
    procedure BuatNoSO;
    procedure UpdateNoSO;
    procedure DBAdvGrid2DblClickCell(Sender: TObject; ARow, ACol:
Integer);
    procedure btnSaveClick(Sender: TObject);
```

```
    function CekValidAll : Boolean;
    procedure SaveAll;
    procedure GetDiskonHeader;
    procedure Pajak;
    procedure HapusDetail;
    procedure SaveMaster;
    procedure SaveDetail;
    procedure UpdateInventory;
    procedure UpdateStatusSerial;
    procedure GetCost;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnDOClick(Sender: TObject);
    procedure IsiReport;
    procedure PrintNoSo;
    procedure btnPostClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmSalesOrder: TfrmSalesOrder;

implementation
uses Numbering, FormComponent, FormSerialNumberChoose,
FormListSalesOrder;
var temp: String;
    Total, TotalQtySOR: Double;
    NoSO,NoArPayment,NoArInvoice,NoReceivable : String;
    NextSuffixNoSO,NextSuffixNoArPayment,NextSuffixNoArInvoice,
    NextSuffixNoReceivable : Integer;
    MetodeNoSO,MetodeNoArPayment,MetodeNoArInvoice,
    MetodeNoReceivable : String;
    NextNoSO,NextNoArPayment,NextNoArInvoice,NextNoReceivable :
    String;
    NoDelivery, NextNoDelivery, metodeNoDelivery : String;
    NextSuffixDelivery : Integer;
    TotalQty, TotalBalance, DiskonHeader, PersenHeader, AmountHeader,
    Tax : Double;
    MetodeDiskonStock, MetodeDiskonHeader: String;
    hargaStock, subtotalStock, DiskonStock, PersenStock, AmountStock:
    Double;
```

```
    StockCode, Description, Warehouse, UOM, Notes, Category, Tipe,
    temp3 : String;
    Qty, harga, subtotal, Persen, Amount, DiskonDetail : Double;
    NoUrut, Line, Warna, NoLine: Integer;
    CurrentCost, QtyOnHand, MtdQtyReceived, QtyAllocated,
    QtyAvailable : Double;
    DiskonPersenHeader, OrderQty, Price : Double;
    OrderRoll, RollOnHand, RollAllocated : Integer;
    SubTotalDetail, SubTotalHeader, SubTotalCost, HargaSatuan : Double;
    InvoiceTotal : Double;
    StatusPajak : String; TotalPajak : Double;
    TipePendapatan, KategoriPendapatan : String;
    QtySold : Double;
    SerialNumber, PriceCode : String;
    UOM2,AlternateUOM, MetodeKonversi, UOMTerkecil,
    UOMTerbesar : String;
    Pembanding : Double;
    HargaDiEdt : String;
    CostSaatJual, RepTotal, RepTotalRib, RepPrice, RepPriceRib : Double;
    CountRoll, CountRollRib, RepLine, i, j, k, l, m : Integer;
    x , y, a, b : Array[1..24] of Double;
    temp_left, temp_description : String;
    simpan, totaldata, TotalRollSOR : Integer;

{$R *.dfm}

procedure TfrmSalesOrder.btnDOClick(Sender: TObject);
begin
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
            '[SoDeliveryDetailInputTemp]';
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
  IsiReport;
  with qryRep2 do
  begin
    Close;
    Parameters[0].Value := edtSO.Text;
  end;
```

```
  with qryRep3 do
  begin
    Close;
    Parameters[0].Value := edtSO.Text;
  end;
  PrintNoSo;
end;

procedure TfrmSalesOrder.IsiReport;
begin
  CountRoll          := 0;
  CountRollRib       := 0;
  RepTotal           := 0;
  RepTotalRib        := 0;
  RepPrice           := 0;
  RepPriceRib        := 0;
  RepLine            := 0;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[SoDeliveryDetail] WHERE SalesOrder = ' +
            QuotedStr(edtSO.Text);
  OpenQrySQL2;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[SoDeliveryDetailInputTemp]';
  OpenQrySQL;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[InvMaster] WHERE StockCode = ' +
            QuotedStr(qrySQL2.FieldByName('StockCode').AsString);
  OpenQrySQL3;
  temp_description := qrySQL3.FieldByName('Description').AsString;
  while not(qrySQL2.Eof) do
  begin
    i  := 1;
    j  := 0;
    k := 1;
    l  := 0;
    m := 1;
    temp_left := LeftStr(qrySQL2.FieldByName('StockCode').AsString, 2);
    if (temp_left = 'RB') then
    begin
      for i := 1 to 24 do
      begin
        if qrySQL2.FieldByName('Qty' + IntToStr(i)).AsFloat <> 0 then
        begin
```

```
        CountRollRib := CountRollRib+1;
        l     := l + 1;
        y[l]  := qrySQL2.FieldByName('Qty' + IntToStr(i)).AsFloat;
      end;
      while m <= l do
      begin
        b[m] := y[m];
        m := m + 1;
      end;
    end;
  end
  else
  begin
    for i := 1 to 24 do
    begin
      if qrySQL2.FieldByName('Qty' + IntToStr(i)).AsFloat <> 0 then
      begin
        CountRoll := CountRoll+1;
        j     := j + 1;
        x[j]  := qrySQL2.FieldByName('Qty' + IntToStr(i)).AsFloat;
      end;
      while k <= j do
      begin
        a[k] := x[k];
        k := k + 1;
      end;
    end;
  end;
  RepLine := RepLine + 1;
  with qrySQL do
  begin
    if (qrySQL2.FieldByName('RibsToStockCode').AsString) = '-' then
    begin
      Insert;
      FieldByName('Line').AsInteger  := RepLine;
      FieldByName('Delivery').AsString :=
qrySQL2.FieldByName('Delivery').AsString;
      FieldByName('SerialNumber').AsString :=
qrySQL2.FieldByName('SerialNumber').AsString;
      FieldByName('DeliveryQty').AsFloat :=
qrySQL2.FieldByName('DeliveryQty').AsFloat; //OrderQty;
      FieldByName('ViewWarna').AsString :=
qrySQL2.FieldByName('ViewWarna').AsString;
```

```
      if (temp_left <> 'RB') then
      begin
        FieldByName('StockCode').AsString :=
qrySQL2.FieldByName('StockCode').AsString;
        FieldByName('Price').AsFloat  :=
qrySQL2.FieldByName('Price').AsFloat;
        RepTotal := RepTotal + qrySQL2.FieldByName('DeliveryQty').AsFloat;
        RepPrice := qrySQL2.FieldByName('Price').AsFloat;
        for k := 1 to j do
        begin
          if a[k] <> 0 then
          begin
            FieldByName('Qty'+IntToStr(k)).AsFloat := a[k];
          end;
        end;
      end
      else
      begin
        FieldByName('StockCodeRib').AsString :=
qrySQL2.FieldByName('StockCode').AsString;
        FieldByName('DeliveryQtyRib').AsFloat :=
qrySQL2.FieldByName('DeliveryQty').AsFloat; //OrderQty;
        FieldByName('ViewWarnaRib').AsString :=
qrySQL2.FieldByName('ViewWarna').AsString;
        FieldByName('PriceRib').AsFloat  :=
qrySQL2.FieldByName('Price').AsFloat;
        RepTotalRib := RepTotalRib +
qrySQL2.FieldByName('DeliveryQty').AsFloat;
        RepPriceRib := qrySQL2.FieldByName('Price').AsFloat;
        for m := 1 to l do
        begin
          if b[m] <> 0 then
          begin
            FieldByName('QtyRib'+IntToStr(m)).AsFloat := b[m];
          end;
        end;
      end;
      Post;
    end
    else if (qrySQL2.FieldByName('RibsToStockCode').AsString) <> '-' then
    begin
      i := 0;
      qrySQL.First;
```

```
      while not(qrySQL.Eof) do
      begin
        if ((qrySQL.FieldByName('StockCode').AsString) =
           (qrySQL2.FieldByName('RibsToStockCode').AsString)) AND (i = 0)
then
        begin
          Edit;
          FieldByName('StockCodeRib').AsString :=
qrySQL2.FieldByName('StockCode').AsString;
          FieldByName('DeliveryQtyRib').AsFloat :=
qrySQL2.FieldByName('DeliveryQty').AsFloat; //OrderQty;
          FieldByName('ViewWarnaRib').AsString :=
qrySQL2.FieldByName('ViewWarna').AsString;
          FieldByName('PriceRib').AsFloat  :=
qrySQL2.FieldByName('Price').AsFloat;
          RepTotalRib := RepTotalRib +
qrySQL2.FieldByName('DeliveryQty').AsFloat;
          RepPriceRib := qrySQL2.FieldByName('Price').AsFloat;
          Post;
          for m := 1 to l do
          begin
            if b[m] <> 0 then
            begin
              with qrySQL3 do
              begin
                SQL.Clear;
                SQL.Add('update '+frmComponent.lblDB.Caption+
                        'SoDeliveryDetailInputTemp set QtyRib'+IntToStr(m)+
                        ' = '+FloatToStr(b[m])+' where StockCodeRib='+
                       QuotedStr(qrySQL2.FieldByName('StockCode').AsString));
                ExecSQL;
              end;
            end;
          end;
          i := i + 1;
        end;
        qrySQL.Next;
      end;
    end;
  end;
  qrySQL2.Next;
 end;
end;
```

```
procedure TfrmSalesOrder.PrintNoSo;
var
  Memo: TfrxMemoView;
  Component: TfrxComponent;
begin
  with qryRepTemp do
  begin
    close;
    Open;
  end;
  Component := frxRepDel.FindObject('time');
  Memo      := Component as TfrxMemoView;
  Memo.Text := FormatDateTime('c',dtpNow.DateTime);
  Component := frxRepDel.FindObject('tanggal');
  Memo := Component as TfrxMemoView;
  Memo.Text := FormatDateTime('dd/MM/yy',dtpOrder.DateTime);
  Component := frxRepDel.FindObject('counter');
  Memo      := Component as TfrxMemoView;
  Memo.Text := FloatToStr(CountRoll);
  Component := frxRepDel.FindObject('description');
  Memo      := Component as TfrxMemoView;
  Memo.Text := temp_description;
  Component := frxRepDel.FindObject('counterRib');
  Memo      := Component as TfrxMemoView;
  Memo.Text := FloatToStr(CountRollRib);
  Component := frxRepDel.FindObject('banyakKg');
  Memo      := Component as TfrxMemoView;
  Memo.Text := FloatToStrF(RepTotal, ffNumber,10,2);
  Component := frxRepDel.FindObject('banyakKgRib');
  Memo      := Component as TfrxMemoView;
  Memo.Text := FloatToStrF(RepTotalRib, ffNumber,10,2);
  if RepPrice <> 0 then
  begin
    Component := frxRepDel.FindObject('price');
    Memo      := Component as TfrxMemoView;
    Memo.Text := FloatToStrF(RepPrice, ffNumber,10,2);
  end
  else
  begin
    Component := frxRepDel.FindObject('price');
    Memo      := Component as TfrxMemoView;
    Memo.Text := '';
  end;
```

```
  if RepPriceRib <> 0 then
  begin
    Component := frxRepDel.FindObject('priceRib');
    Memo      := Component as TfrxMemoView;
    Memo.Text := FloatToStrF(RepPriceRib, ffNumber,10,2);
  end
  else
  begin
    Component := frxRepDel.FindObject('priceRib');
    Memo      := Component as TfrxMemoView;
    Memo.Text := '';
  end;
  with qrySQL do
  begin
    temp := 'SELECT * FROM ArCustomer WHERE CustomerCode = ' +
                QuotedStr(edtCustomer.Text);
    OpenQrySQL;
  end;
    Component := frxRepDel.FindObject('customername');
    Memo := Component as TfrxMemoView;
    Memo.Text :=  qrySQL.FieldByName('CustomerName').AsString;
    Component := frxRepDel.FindObject('alamat');
    Memo := Component as TfrxMemoView;
    Memo.Text := qrySQL.FieldByName('Address').AsString;
    Component := frxRepDel.FindObject('telepon');
    Memo := Component as TfrxMemoView;
    Memo.Text :=  qrySQL.FieldByName('phone').AsString;
    Component := frxRepDel.FindObject('alamat2');
    Memo := Component as TfrxMemoView;
    Memo.Text :=  qrySQL.FieldByName('City').AsString;
  frxRepDel.ShowReport;
end;

procedure TfrmSalesOrder.btnEditDetailClick(Sender: TObject);
begin
    if not qryDtl.IsEmpty then
    begin
      btnSaveDetail.Enabled := True;
      btnEditDetail.Enabled := False;
      Total               := Total - qryDtl.FieldByName('Total').AsFloat;
      edtTotal.Text    := FloatToStrF(Total, ffNumber, 10, 2);
      NoUrut              := qryDtl.FieldByName('Line').AsInteger;
      edtStockCode.Text := qryDtl.FieldByName('StockCode').AsString;
```

```
    edtRibToStockCode.Text:=
qryDtl.FieldByName('RibsToStockCode').AsString;
    edtRoll.Text          := qryDtl.FieldByName('Roll').AsString;
    edtQty.Text           := qryDtl.FieldByName('Qty').AsString;
    temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[InvMaster] WHERE StockCode= '+
              QuotedStr(edtStockCode.Text);
    OpenQrySQL;
    if edtPrice.Text <> '' then
    begin
      edtPrice.Text           := qryDtl.FieldByName('Price').AsString;
      edtViewWarna.Text := qryDtl.FieldByName('ViewWarna').AsString;
    end
    else
      edtPrice.Text           := qrySQL.FieldByName('SellingPrice').AsString;
    edtViewWarna.Text   := qrySQL.FieldByName('ViewWarna').AsString;
    edtUOM.Text             := qrySQL.FieldByName('Satuan').AsString;
    edtType.Text            := qrySQL.FieldByName('Type').AsString;
    edtDescription.Text    := qrySQL.FieldByName('Description').AsString;
    CekDiskonEdit;
    temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
              ' [SalDetailTemp] WHERE Line = ' + IntToStr(NoUrut);
    with qrySQL do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      ExecSQL;
    end;
    qryDtl.Close;
    qryDtl.Open;
    temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
              '[SoDeliveryDetailTemp] WHERE Line = ' + IntToStr(NoUrut);
    with qrySQL do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      ExecSQL;
    end;
    qryDtlDelivery.Close;
    qryDtlDelivery.Open;
  end
```

```
end;

procedure TfrmSalesOrder.btnFindSerialClick(Sender: TObject);
begin
  for i := 1 to 24 do
  begin
    TEdit(frmSerialNumberChoose.FindComponent('edtQty' +
IntToStr(i))).Clear;
    TCheckBox(frmSerialNumberChoose.FindComponent('CheckBox' +
IntToStr(i))).Enabled := false;
  end;
  frmSerialNumberChoose.lblAsal.Caption  := 'SAL3';
  frmSerialNumberChoose.txtTotal.Caption := edtQty.Text;
  frmSerialNumberChoose.txtRoll.Caption := edtRoll.Text;
  frmSerialNumberChoose.txtRollBaru.Visible := False;
  frmSerialNumberChoose.StaticText6.Visible:= False;
  frmSerialNumberChoose.StaticText9.Visible:= False;
  frmSerialNumberChoose.edtStockCode.Text := edtStockCode.Text;
  frmSerialNumberChoose.edtSOR.Text := edtSOR.Text;
  frmSerialNumberChoose.ShowModal;
end;

procedure TfrmSalesOrder.btnPostClick(Sender: TObject);
var
jawab : integer;
begin
  DiskonPersenHeader  := 0;
  OrderQty            := 0;
  Price               := 0;
  DiskonDetail        := 0;
  subtotal            := 0;
  SubTotalDetail      := 0;
  SubTotalHeader      := 0;
  HargaSatuan         := 0;
  Jawab := MessageDlg('Do you want to post this order?', mtConfirmation,
  mbOKCancel, 0);
  if Jawab = mrOk then
  begin
    temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[SalDetail] WHERE SalesOrder = ' + QuotedStr(edtSO.Text);
    OpenQrySQL;
    temp := 'SELECT DiskonPersen FROM ' +
            frmComponent.lblDB.Caption +
```

```
            '[SoMaster] WHERE SalesOrder = ' + QuotedStr(edtSO.Text);
  OpenQrySQL2;
  qrySQL.First;
  while not(qrySQL.EoF) do
  begin
    DiskonPersenHeader := qrySQL2.FieldByName('DiskonPersen').AsFloat;
    OrderQty    := qrySQL.FieldByName('DeliveryQty').AsFloat;
    OrderRoll   := qrySQL.FieldByName('Roll').AsInteger;
    Price       := qrySQL.FieldByName('Price').AsFloat;
    DiskonDetail:= qrySQL.FieldByName('Diskon').AsFloat;
    SubTotalDetail:= OrderQty * Price - DiskonDetail;
    SubTotalHeader:= SubTotalDetail * DiskonPersenHeader / 100;
    SubTotal      := SubTotalDetail - SubTotalHeader;
    HargaSatuan   := SubTotal / OrderQty;
    temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[InvWarehouse] WHERE StockCode = ' +
              QuotedStr(qrySQL.FieldByName('StockCode').AsString);
    with qrySQL3 do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
      QtyOnHand     := FieldByName('QtyOnHand').AsFloat;
      QtyAllocated  := FieldByName('QtyAllocated').AsFloat;
      QtyAvailable  := FieldByName('QtyAvailable').AsFloat;
      RollOnHand    := FieldByName('RollOnHand').AsInteger;
      RollAllocated := FieldByName('RollAllocated').AsInteger;
      Edit;
      FieldByName('QtyOnHand').AsFloat          := QtyOnHand - OrderQty;
      FieldByName('RollOnHand').AsInteger       := RollOnHand - OrderRoll;
      FieldByName('QtyAllocated').AsFloat       := QtyAllocated - OrderQty;
      FieldByName('RollAllocated').AsInteger    := RollAllocated - OrderRoll;
      FieldByName('QtyOnDelivery').AsFloat      :=
FieldByName('QtyOnDelivery').AsFloat + OrderQty;
      FieldByName('RollOnDelivery').AsInteger   :=
FieldByName('RollOnDelivery').AsInteger + OrderRoll;
      Post;
    end;
    qrySQL.Next;
  end;
  temp := 'UPDATE SoMaster SET Status = ' +
            QuotedStr('COMPLETE-DO') +
```

```
            'WHERE SalesOrder = ' + QuotedStr(edtSO.Text);
    qrySQL3.Close;
    qrySQL3.SQL.Clear;
    qrySQL3.SQL.Add(temp);
    qrySQL3.ExecSQL;
    temp := 'UPDATE SoDeliveryMaster SET StatusSO = ' +
            QuotedStr('COMPLETE-DO') +
            'WHERE SalesOrder = ' + QuotedStr(edtSO.Text);
    qrySQL3.Close;
    qrySQL3.SQL.Clear;
    qrySQL3.SQL.Add(temp);
    qrySQL3.ExecSQL;
    frmSalesOrderList.cbCustomer.Text := edtCustomer.Text;
    frmSalesOrderList.cbStatus.ItemIndex := 2;
    frmSalesOrderList.btnDelivery.Enabled := false;
    frmSalesOrderList.gbMasterSO.Visible:= True;
    frmSalesOrderList.gbMaster.Visible  := False;
    frmSalesOrderList.gbDetailSO.Visible:= True;
    frmSalesOrderList.gbDetail.Visible  := False;
    frmSalesOrderList.RefreshMaster;
    frmSalesOrderList.RefreshDetail;
    frmSalesOrderList.edtFind.Text    := edtSO.Text;
    MessageDlg('Save data success ! Data can not be changed ! ',
mtConfirmation, [mbOK], 0);
    btnPost.Enabled := False;
    btnSave.Enabled := False;
    GroupBox1.Enabled := False;
    GroupBox2.Enabled  := False;
    GroupBox3.Enabled  := False;
  end;
end;

procedure TfrmSalesOrder.btnSaveClick(Sender: TObject);
begin
  BuatNoSO;
  BuatNoDelivery;
  if CekValidAll then
  begin
    SaveAll;
    frmSalesOrderList.cbCustomer.Text    := edtCustomer.Text;
    frmSalesOrderList.cbStatus.ItemIndex := 1;
    frmSalesOrderList.btnAdd.Enabled     := False;
    frmSalesOrderList.btnDelivery.Enabled:= True;
```

```
    frmSalesOrderList.gbMasterSO.Visible := True;
    frmSalesOrderList.gbMaster.Visible   := False;
    frmSalesOrderList.gbDetailSO.Visible := True;
    frmSalesOrderList.gbDetail.Visible   := False;
    frmSalesOrderList.RefreshMaster;
    frmSalesOrderList.RefreshDetail;
    frmSalesOrderList.edtFind.Text       := edtSO.Text;
    btnSaveDetail.Enabled := True;
    btnDO.Enabled := True;
    btnPost.Enabled := True;
  end;
end;

procedure TfrmSalesOrder.GetDiskonHeader;
begin
  if (StringReplace(edtDiskonHeader.Text, ' ', '', [rfReplaceAll, rfIgnoreCase]) =
'')
  then
  begin
    DiskonHeader := 0;
    edtDiskonHeader.Text := '0';
  end;
  DiskonHeader := StrToFloat(edtDiskonHeader.Text);
  DiskonHeader := (Total * DiskonHeader / 100);
end;

procedure TfrmSalesOrder.Pajak;
begin
  if chkTax.Checked = True then
  begin
    Tax := (Total - DiskonHeader) * 0.1;
    StatusPajak := 'Y';
    TotalPajak := InvoiceTotal * 0.1;
  end
  else
  begin
    Tax := 0;
    StatusPajak := 'N';
    TotalPajak := 0;
  end;
end;
```

```
procedure TfrmSalesOrder.HapusDetail;
begin
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
            '[SalDetail] WHERE SalesOrder = ' + QuotedStr(NoSO);
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
            '[SoDeliveryDetail] WHERE SalesOrder = ' + QuotedStr(NoSO);
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
            '[InvSerialNumberHistory] WHERE SalesOrder = ' +
            QuotedStr(NoSO);
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
end;

procedure TfrmSalesOrder.SaveAll;
var Jawab : Integer;
begin
  GetDiskonHeader;
  Pajak;
  if not frmComponent.adoWingga.InTransaction then
  begin
    frmComponent.adoWingga.BeginTrans;
    try
      SaveMaster;
      HapusDetail;
```

```
      SaveDetail;
      UpdateNoSO;
      UpdateNoDelivery;
      frmComponent.adoWingga.CommitTrans;
      MessageDlg('Save data success !'+#13#10+'SO number : ' +
edtSO.Text+#13#10+'DO number : ' + edtDO.Text, mtConfirmation, [mbOK],
0);
      btnSave.Enabled    := True;
    Except
      on E: Exception do
      begin
      frmComponent.adoWingga.RollbackTrans;
      MessageDlg('Save data failed !', mtError, [mbOK], 0);
      ShowMessage(e.Message);
      end;
    end; // end try
  end; // end if ado.InTrans
end;

procedure TfrmSalesOrder.SaveMaster;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[SoMaster] WHERE SalesOrder = ' + QuotedStr(NoSO);
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
    if txtStatus.Caption = 'OPEN' then
    begin
      Insert;
      FieldByName('SalesOrder').AsString  := NoSO;
      FieldByName('Requisition').AsString := edtSOR.Text;
      FieldByName('OrderDate').AsDateTime := dtpOrder.DateTime;
      FieldByName('CustomerCode').AsString:= edtCustomer.Text;
      FieldByName('AddID').AsString       :=
frmComponent.lblUserName.Caption;
      FieldByName('Warna').AsString       := IntToStr(Warna);
    end
    else
    begin
      Edit;
```

```
    FieldByName('Warna').AsString       := edtWarna.Text;
  end;
  FieldByName('OrderTotal').AsFloat   := Total;
  FieldByName('DiskonPersen').AsFloat := StrToFloat(edtDiskonHeader.Text);
  FieldByName('Diskon').AsFloat       := DiskonHeader;
  FieldByName('Tax').AsFloat          := Tax;
  FieldByName('Taxable').AsString     := StatusPajak;
  FieldByName('TermsCode').AsString   := cbTerm.Text;
  FieldByName('Status').AsString      := 'COMPLETE-SO';
  FieldByName('Supir').AsString       := edtSupir.Text;
  FieldByName('Kenek').AsString       := edtKenek.Text;
  Post;
end;
temp := 'UPDATE SoRequisitionMaster SET SalesOrder = ' +
          QuotedStr(edtSO.Text) + 'WHERE Requisition = ' +
          QuotedStr(edtSOR.Text);
qrySQL2.Close;
qrySQL2.SQL.Clear;
qrySQL2.SQL.Add(temp);
qrySQL2.ExecSQL;
temp := 'SELECT SUM(DeliveryQty) AS TotalQty FROM ' +
          frmComponent.lblDB.Caption + '[SoDeliveryDetailTemp] ';
OpenQrySQL;
TotalQty := qrySQL.FieldByName('TotalQty').AsFloat;
temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
          '[SoDeliveryMaster] WHERE SalesOrder = ' +
          QuotedStr(edtSO.Text) + ' AND Delivery = ' +
          QuotedStr(edtDO.Text);
OpenQrySQL;
with qrySQL do
begin
  if txtStatus.Caption = 'OPEN' then
  begin
    Insert;
    FieldByName('Customer').AsString      := edtCustomer.Text;
    FieldByName('SalesOrder').AsString    := edtSO.Text;
    FieldByName('Delivery').AsString      := edtDO.Text;
    FieldByName('DeliveryDate').AsDateTime:= dtpOrder.DateTime;
    FieldByName('StatusSO').AsString      := 'COMPLETE-SO';
  end
  else
  begin
    Edit;
```

```
    end;
    FieldByName('DeliveryTotal').AsFloat  := TotalQty;
    FieldByName('AddID').AsString         :=
frmComponent.lblUserName.Caption;
    Post;
  end;
end;

procedure TfrmSalesOrder.SaveDetail;
var j : byte;
begin
  InvoiceTotal := 0;
    qryDtl.First;
    while not(qryDtl.EoF) do
    begin
      GetCost;
      temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
                '[SalDetail] WHERE SalesOrder = ' + QuotedStr(NoSO);
      OpenQrySQL;
      qrySQL.Insert;
      qrySQL.FieldByName('SalesOrder').AsString     := NoSO;
      qrySQL.FieldByName('Line').AsInteger          :=
qryDtl.FieldByName('Line').AsInteger; //Line + 1;
      qrySQL.FieldByName('StockCode').AsString      :=
qryDtl.FieldByName('StockCode').AsString;
      qrySQL.FieldByName('RibsToStockCode').AsString:=
qryDtl.FieldByName('RibsToStockCode').AsString;
      qrySQL.FieldByName('OrderQty').AsFloat        :=
qryDtl.FieldByName('Qty').AsFloat; //OrderQty;
      qrySQL.FieldByName('DeliveryQty').AsFloat     :=
qryDtl.FieldByName('Qty').AsFloat; //OrderQty;
      qrySQL.FieldByName('Price').AsFloat           :=
qryDtl.FieldByName('Price').AsFloat; //Price;
      qrySQL.FieldByName('DiskonPersen').AsCurrency :=
qryDtl.FieldByName('DiskonPersen').AsCurrency;
      qrySQL.FieldByName('DiskonAmount').AsCurrency :=
qryDtl.FieldByName('DiskonAmount').AsCurrency;
      qrySQL.FieldByName('Diskon').AsCurrency       :=
qryDtl.FieldByName('Diskon').AsCurrency; //DiskonDetail;
      qrySQL.FieldByName('Roll').AsInteger          :=
qryDtl.FieldByName('Roll').AsInteger;
      qrySQL.Post;
      UpdateInventory;
```

```
    InvoiceTotal := InvoiceTotal + (HargaSatuan * OrderQty);
    with qryDtlDelivery do
    begin
      Close;
      Parameters[0].Value := qryDtl.FieldByName('Line').AsString;
      Open;
    end;
    temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[SoDeliveryDetail] WHERE SalesOrder = ' +
              QuotedStr(edtSO.Text) +
              ' AND Line = ' + IntToStr(Line + 1)  + ' AND Delivery = ' +
              QuotedStr(edtDO.Text);
    OpenQrySQL2;
    temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[InvSerialNumberHistory] WHERE SerialNumber = ' +
         QuotedStr(qryDtlDelivery.FieldByName('SerialNumber').AsString);
    OpenQrySQL;
    while not qryDtlDelivery.Eof do
    begin
      with qrySQL2 do
      begin
        Insert;
        FieldByName('SalesOrder').AsString    := edtSO.Text;
        FieldByName('Line').AsInteger         :=
qryDtlDelivery.FieldByName('Line').AsInteger; //Line + 1;
        FieldByName('Delivery').AsString      := edtDO.Text;
        FieldByName('DeliveryLine').AsInteger :=
qryDtlDelivery.FieldByName('DeliveryLine').AsInteger; //Line +1;
        FieldByName('StockCode').AsString     :=
qryDtlDelivery.FieldByName('StockCode').AsString;
        FieldByName('SerialNumber').AsString  :=
qryDtlDelivery.FieldByName('SerialNumber').AsString;
        FieldByName('DeliveryQty').AsFloat    :=
qryDtlDelivery.FieldByName('DeliveryQty').AsFloat; //OrderQty;
        FieldByName('Price').AsFloat          :=
qryDtlDelivery.FieldByName('Price').AsFloat; //Price;
        FieldByName('Total').AsFloat          :=
qryDtlDelivery.FieldByName('Price').AsFloat *
qryDtlDelivery.FieldByName('DeliveryQty').AsFloat; //Price * OrderQty;
        FieldByName('ViewWarna').AsString     :=
qryDtlDelivery.FieldByName('ViewWarna').AsString;
        for j := 1 to 24 do
        begin
```

```
          FieldByName('Qty'+IntToStr(j)).AsFloat :=
qryDtlDelivery.FieldByName('Qty'+IntToStr(j)).AsFloat;
          if(qryDtlDelivery.FieldByName('Qty'+IntToStr(j)).AsFloat<>0) then
            FieldByName('DeliveryRoll').AsInteger :=
FieldByName('DeliveryRoll').AsInteger + 1;
        end;
        FieldByName('RibsToStockCode').AsString:=
qryDtl.FieldByName('RibsToStockCode').AsString;
        Post;
      end;
      with qrySQL do
      begin
        Insert;
        FieldByName('Date').AsDateTime            := frmComponent.GetDate();
        FieldByName('SerialNumber').AsString      :=
qryDtlDelivery.FieldByName('SerialNumber').AsString;
        FieldByName('StockCode').AsString         :=
qryDtlDelivery.FieldByName('StockCode').AsString;
        for j := 1 to 24 do
          FieldByName('Qty'+IntToStr(j)).AsFloat  :=
qryDtlDelivery.FieldByName('Qty'+IntToStr(j)).AsFloat;
        FieldByName('SalesOrder').AsString        := edtSO.Text;
        FieldByName('DeliveryOrder').AsString     := edtDO.Text;
        FieldByName('SalesOrderRequisition').AsString:= edtSOR.Text;
        Post;
      end;
      qryDtlDelivery.Next;
    end;
    UpdateStatusSerial;
    qryDtl.Next;
  end;
end;

procedure TfrmSalesOrder.GetCost;
begin
  temp := 'SELECT CustomerCode,DiskonPersen FROM ' +
            frmComponent.lblDB.Caption +
            '[SoMaster] WHERE SalesOrder = ' + QuotedStr(edtSO.Text);
  OpenQrySQL2;
  DiskonPersenHeader := qrySQL2.FieldByName('DiskonPersen').AsFloat;
  OrderQty := qryDtl.FieldByName('Qty').AsFloat;
  Price := qryDtl.FieldByName('Price').AsFloat;
  DiskonDetail := qryDtl.FieldByName('Diskon').AsFloat;
```

```
    SubTotalDetail := OrderQty * Price - DiskonDetail;
    SubTotalHeader := SubTotalDetail * DiskonPersenHeader / 100;
    SubTotal := SubTotalDetail - SubTotalHeader;
    HargaSatuan := SubTotal / OrderQty;
end;

procedure TfrmSalesOrder.UpdateInventory;
var tempQty : Double;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[InvMovements]';
  with qrySQL3 do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
    Insert;
    FieldByName('StockCode').AsString :=
qryDtl.FieldByName('StockCode').AsString;
    FieldByName('EntryDate').AsDateTime := frmComponent.GetDate();
    FieldByName('MovementType').AsString := 'S';
    FieldByName('TrnQty').AsFloat := -OrderQty;
    FieldByName('TrnValue').AsFloat := HargaSatuan * OrderQty;
    FieldByName('EnteredCost').AsFloat := 0;
    FieldByName('Reference').AsString := 'SAL';
    FieldByName('SalesOrder').AsString := edtSO.Text;
    Post;
  end;
end;

procedure TfrmSalesOrder.UpdateStatusSerial;
begin
  temp := 'SELECT * FROM InvSerialNumberTemp';
  OpenQrySQL;
  if not qrySQL.Eof then
  begin
    qrySQL.First;
    while not qrySQl.Eof do
    begin
      temp := 'SELECT * FROM InvSerialNumber WHERE '+
                'SerialNumber= '+
                QuotedStr(qrySQL.FieldByName('SerialNumber').AsString) +
```

```
                ' AND StockCode = ' +
                QuotedStr(qrySQL.FieldByName('StockCode').AsString);
      OpenQrySQL2;
      with qrySQL2 do
      begin
        Edit;
        for j := 1 to 24 do
        begin
          if qrySQL.FieldByName('Qty' + IntToStr(j)).AsFloat <> 0 then
          begin
            FieldByName('Status' + IntToStr(j)).AsString := 'ORDER';
          end;
        end;
        Post;
        qrySQL.Next;
      end;
    end;
  end;
end;

function TfrmSalesOrder.CekValidAll : Boolean;
var
  Jawab: Integer;
begin
  CekValidAll := True;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[SalDetailTemp] WHERE Price = 0 ';
  OpenQrySQL;
  if (StringReplace(edtSO.Text, ' ', '', [rfReplaceAll, rfIgnoreCase]) = '')
  then
  begin
    MessageDlg('Transaction number required!', mtError, [mbOK], 0);
    edtSO.SetFocus;
    CekValidAll := False;
  end
  else if (MetodeDiskonHeader = 'A') AND (StrToFloat(edtDiskonHeader.Text)
> 100)
  then
  begin
    MessageDlg('Discount cannot be more than 100%!', mtError, [mbOK], 0);
    edtDiskonHeader.SetFocus;
    CekValidAll := False;
  end
```

```
  else if (MetodeDiskonHeader = 'B') AND (StrToFloat(edtDiskonHeader.Text)
> Total)
  then
  begin
    MessageDlg('Discount cannot be more than total order!', mtError, [mbOK],
0);
    edtDiskonHeader.SetFocus;
    CekValidAll := False;
  end
  else if (StringReplace(edtDiskonHeader.Text, ' ', '', [rfReplaceAll,
rfIgnoreCase]) = '')
         OR (edtDiskonHeader.Text = '') then
  begin
    MessageDlg('Discount cannot be empty!', mtError, [mbOK], 0);
    if edtDiskonHeader.Enabled = True then
    begin
      edtDiskonHeader.SetFocus;
    end
    else
    begin
      edtDiskonHeader.Enabled := True;
      edtDiskonHeader.SetFocus;
    end;
    CekValidAll := False;
  end
  else if (edtPrice.Text <> '0')then
  begin
    Jawab := MessageDlg('Do you want to discard the changes?', mtConfirmation,
      mbOKCancel, 0);
    if Jawab = mrCancel then
    begin
      BersihkanInput;
      CekValidAll := False;
    end
    else
    begin
      Jawab := MessageDlg('Do you want to save this order?', mtConfirmation,
        mbOKCancel, 0);
      if Jawab = mrCancel then
      begin
        CekValidAll := False;
      end;
    end;
```

```
  end
  else
  begin
    Jawab := MessageDlg('Do you want to save this order?', mtConfirmation,
      mbOKCancel, 0);
    if Jawab = mrCancel then
    begin
      CekValidAll := False;
    end;
  end;
end;

procedure TfrmSalesOrder.btnSaveDetailClick(Sender: TObject);
begin
  HargaDiEdt := StringReplace(edtPrice.Text,'.00','',[rfReplaceAll,
rfIgnoreCase]);
  HargaDiEdt := StringReplace(HargaDiEdt,',','',[rfReplaceAll, rfIgnoreCase]);
  HargaDiEdt := StringReplace(HargaDiEdt,'.','',[rfReplaceAll, rfIgnoreCase]);
  if (edtPrice.Text <> '') AND (HargaDiEdt <> FloatToStr(harga)) then
  begin
    harga := StrToFloat(edtPrice.Text);
  end
  else begin
    harga := harga;
  end;
  CreateHarga;
  SaveTemp;
  simpan := simpan + 1;
  BersihkanInput;
  btnSaveDetail.Enabled := False;
  btnEditDetail.Enabled := True;
  if (simpan = totaldata) and (txtStatus.Caption = 'OPEN') then
  begin
    btnSave.Enabled:= True;
  end
  else if (txtStatus.Caption <> 'OPEN') then
  begin
    btnSave.Enabled:= True;
  end
end;

procedure TfrmSalesOrder.SaveTemp;
var NoDeliveryLine : Byte;
```

```
    QtyDelivery    : Double;
begin
    StockCode   := edtStockCode.Text;
    Qty         := StrToFloat(edtQty.Text);
    harga       := hargaStock;
    Persen      := PersenStock;
    Amount      := AmountStock;
    DiskonDetail:= DiskonStock;
    subtotal    := subtotalStock;
    with qryDtl do
    begin
      Close;
      Open;
      if (not(IsEmpty)) then
        NoUrut := NoUrut;
    end;
    with qryDtl do
    begin
      Insert;
      FieldByName('Line').AsInteger     := NoUrut;
      FieldByName('StockCode').AsString := StockCode;
      if edtRibToStockCode.Text <> '-' then
      begin
        FieldByName('RibsToStockCode').AsString := edtRibToStockCode.Text;
      end
      else
        FieldByName('RibsToStockCode').AsString := '-';
      FieldByName('Qty').AsFloat          := Qty;
      FieldByName('Roll').AsInteger       := StrToInt(edtRoll.Text);
      FieldByName('Price').AsFloat        := harga;
      FieldByName('DiskonPersen').AsFloat := Persen;
      FieldByName('DiskonAmount').AsFloat := Amount;
      FieldByName('Diskon').AsFloat       := DiskonDetail;
      Post;
      temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
                '[SoDeliveryDetailTemp]';
      with qrySQL do
      begin
        Close;
        SQL.Clear;
        SQL.Add(temp);
        Open;
        temp:= 'SELECT * FROM ' + frmComponent.lblDB.Caption +
```

```
                '[InvSerialNumberTemp] WHERE StockCode = ' +
                QuotedStr(edtStockCode.Text);
        OpenQrySQL2;
        NoDeliveryLine := 1;
        while not qrySQL2.eof do
        begin
          QtyDelivery := 0;
          for j := 1 to 24 do
            QtyDelivery := QtyDelivery + qrySQL2.FieldByName('Qty' +
IntToStr(j)).AsFloat;
          Insert;
          FieldByName('Line').AsInteger         := NoUrut;
          FieldByName('DeliveryLine').AsInteger := NoDeliveryLine;
          FieldByName('StockCode').AsString := StockCode;
          FieldByName('RibsToStockCode').AsString := edtRibToStockCode.Text;
          FieldByName('SerialNumber').AsString  :=
qrySql2.FieldByName('SerialNumber').AsString;
          FieldByName('DeliveryQty').AsFloat    := QtyDelivery;
          FieldByName('Price').AsFloat          := harga;
          FieldByName('Total').AsFloat          :=
harga*(qrySql2.FieldByName('TotalQty').AsFloat);
          for j := 1 to 24 do
            FieldByName('Qty' + IntToStr(j)).AsFloat:= qrySql2.FieldByName('Qty'
+ IntToStr(j)).AsFloat;
          FieldByName('ViewWarna').AsString     := edtViewWarna.Text;
          Post;
          NoDeliveryLine := NoDeliveryLine + 1;
          qrySQL2.Next;
        end;
      end;
      Total             := Total + ((Qty*harga)-DiskonDetail);
      edtTotal.Text := FloatToStrF(Total, ffNumber, 10, 2);
      Anjuran;
      Close;
      Open;
      OpenQryTemp;
    end;
end;

procedure TfrmSalesOrder.Anjuran;
var TambahPajak, SudahDiskon, TambahDiskon, Persen : Double;
begin
  TambahPajak := 0;
```

```
  TambahDiskon := 0;
  SudahDiskon := 0;
  Persen := 0;
  if ((edtDiskonHeader.Text <> '') AND (StrToFloat(edtDiskonHeader.Text) >
0))
  then
  begin
    Persen := StrToFloat(edtDiskonHeader.Text);
    TambahDiskon := (Total * Persen / 100);
  end
  else
  begin
    Persen := 0;
    TambahDiskon := 0;
  end;
  SudahDiskon := Total - TambahDiskon;
  if chkTax.Checked = True then
  begin
    TambahPajak := (SudahDiskon * 0.1);
  end
  else
  begin
    TambahPajak := 0;
  end;
end;

procedure TfrmSalesOrder.OpenQryTemp;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[SalDetailTemp]';
    with qryDtl do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
    end;
  temp := 'SELECT StockCode FROM [SalDetailTemp]';
  OpenQrySQL;
end;

procedure TfrmSalesOrder.rbAmountDetailClick(Sender: TObject);
begin
```

```
  MetodeDiskonStock := '2';
  edtDiskonDetail.Enabled := True;
end;

procedure TfrmSalesOrder.rbPersenDetailClick(Sender: TObject);
begin
  MetodeDiskonStock := '1';
  edtDiskonDetail.Enabled := True;
end;

procedure TfrmSalesOrder.CreateHarga;
begin
  hargaStock:= harga;
  subtotalStock := StrToFloat(edtQty.Text) * harga;
  DiskonStock := StrToFloat(edtDiskonDetail.Text);
  if MetodeDiskonStock = '1' then
  begin
    PersenStock := DiskonStock;
    AmountStock := 0;
    DiskonStock := (subtotalStock * DiskonStock / 100);
    subtotalStock := subtotalStock - DiskonStock;
  end
  else if MetodeDiskonStock = '2' then
  begin
    subtotalStock := subtotalStock - DiskonStock;
    AmountStock := DiskonStock;
    PersenStock := 0;
  end
  else subtotalStock := subtotalStock;
end;

procedure TfrmSalesOrder.DBAdvGrid2ClickCell(Sender: TObject; ARow,
  ACol: Integer);
begin
    with qryDtlDelivery do
    begin
      Close;
      Parameters[0].Value := qryDtl.FieldByName('Line').AsString;
      Open;
    end;
  btnEditDetail.Enabled   := True;
end;
```

```
procedure TfrmSalesOrder.DBAdvGrid2DblClickCell(Sender: TObject; ARow,
  ACol: Integer);
begin
  btnEditDetail.Click;
end;

procedure TfrmSalesOrder.CekDiskonEdit;
begin
    if (qryDtl.FieldByName('DiskonPersen').AsFloat > 0) AND
        (qryDtl.FieldByName('Diskon').AsFloat > 0) then
    begin
      rbPersenDetail.Checked := True;
      MetodeDiskonStock := '1';
      edtDiskonDetail.Text :=
FloatToStr(qryDtl.FieldByName('DiskonPersen').AsFloat);
    end
    else if (qryDtl.FieldByName('DiskonAmount').AsFloat > 0) AND
      (qryDtl.FieldByName('Diskon').AsFloat > 0) then
    begin
      rbAmountDetail.Checked := True;
      MetodeDiskonStock := '2';
      edtDiskonDetail.Text :=
FloatToStr(qryDtl.FieldByName('DiskonAmount').AsFloat);
    end
    else if (qryDtl.FieldByName('Diskon').AsFloat = 0) then
    begin
        edtDiskonDetail.Text := '0';
    end;
end;

procedure TfrmSalesOrder.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  Bersihkan;
end;

procedure TfrmSalesOrder.FormShow(Sender: TObject);
begin
  if txtStatus.Caption = 'OPEN' then
  begin
    btnSave.Enabled         := False;
    btnSaveDetail.Enabled   := False;
    btnEditDetail.Enabled   := True;
```

```
    btnDO.Enabled           := False;
    btnPost.Enabled         := False;
    Bersihkan;
    edtDO.Text := 'NEW';
    edtSO.Text := 'NEW';
  end
  else
  begin
    edtDiskonHeader.Text    := '0';
    btnSave.Enabled         := True;
    btnSaveDetail.Enabled   := False;
    btnEditDetail.Enabled   := True;
    btnDO.Enabled           := True;
    btnPost.Enabled         := True;
    BersihkanInput;
    BersihkanTemp;
  end;
  dtpNow.DateTime := Now;
  Tampilan;
end;

procedure TfrmSalesOrder.BuatNoDelivery;
var Panjang : Integer;
    Prefix : String;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[SetupCode] WHERE [Transaction] = ' +
             QuotedStr('Delivery Order');
  OpenQrySQL;
  with qrySQL do
  begin
      Prefix :=  FieldByName('Prefix').AsString;
      Panjang := FieldByName('Length').AsInteger;
      if edtDO.Text = 'NEW' then
      begin
        NoDelivery := (FieldByName('Next').AsString);
        NextSuffixDelivery := (FieldByName('Suffix').AsInteger + 1);
        NextNoDelivery :=
UpdateCode(IntToStr(NextSuffixDelivery),Prefix,Panjang);
      end
      else NoDelivery := edtDO.Text;
      edtDO.Text := NoDelivery;
  end;
```

```
end;

procedure TfrmSalesOrder.UpdateNoDelivery;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[SetupCode] WHERE [Transaction] = ' +
            QuotedStr('Delivery Order');
  OpenQrySQL;
  with qrySQL do
  begin
     Edit;
     FieldByName('Next').AsString := NextNoDelivery;
     FieldByName('Suffix').AsInteger := NextSuffixDelivery;
     Post;
  end;
end;

procedure TfrmSalesOrder.BuatNoSO;
var Panjang : Integer;
    Prefix : String;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[SetupCode] WHERE [Transaction] = ' +
            QuotedStr('Sales Order');
  OpenQrySQL;
  with qrySQL do
  begin
     Prefix :=  FieldByName('Prefix').AsString;
     Panjang := FieldByName('Length').AsInteger;
     if edtSO.Text = 'NEW' then
     begin
       NoSO := (FieldByName('Next').AsString);
       NextSuffixNoSO := (FieldByName('Suffix').AsInteger + 1);
       NextNoSO := UpdateCode(IntToStr(NextSuffixNoSO),Prefix,Panjang);
     end
     else NoSO := edtSO.Text;
     edtSO.Text := NoSO;
  end;
end;

procedure TfrmSalesOrder.UpdateNoSO;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
```

```
            '[SetupCode] WHERE [Transaction] = ' +
            QuotedStr('Sales Order');
  OpenQrySQL;
  with qrySQL do
  begin
     Edit;
     FieldByName('Next').AsString := NextNoSO;
     FieldByName('Suffix').AsInteger := NextSuffixNoSO;
     Post;
  end;
end;

procedure TfrmSalesOrder.Bersihkan;
begin
  { ---Header--- }
  edtSO.Text       := '';
  edtDO.Text       := '';
  edtDiskonHeader.Enabled := True;
  edtDiskonHeader.Text    := '0';
  chkTax.Checked    := False;
  edtWarna.Text     := '';
  edtSupir.Text     := '';
  edtKenek.Text     := '';
  dtpOrder.Date     := now;
  BersihkanInput;
  BersihkanTemp;
  { ---Inisialisasi--- }
  edtTotal.Text := '0';
  Warna         := 0;
  NoUrut        := 0;
  Total         := 0;
  TotalQty      := 0;
  DiskonDetail  := 0;
  DiskonHeader  := 0;
  Persen        := 0;
  PersenHeader  := 0;
  Amount        := 0;
  AmountHeader  := 0;
  MetodeDiskonStock  := '';
  MetodeDiskonHeader := '';
  HargaDiEdt    := '';
  simpan        := 0;
  totaldata     := 0;
```

```
  GroupBox1.Enabled:= True;
  GroupBox2.Enabled:= True;
  GroupBox3.Enabled:= True;
end;

procedure TfrmSalesOrder.BersihkanInput;
begin
  edtStockCode.Text     := '';
  edtDescription.Text   := '';
  edtViewWarna.Text     := '';
  edtRibToStockCode.Text:= '';
  edtQty.Text           := '';
  edtRoll.Text          := '';
  edtPrice.Text         := '';
  edtDiskonDetail.Text  := '';
  edtUOM.Text           := '';
  edtType.Text          := '';
  rbPersenDetail.Checked := False;
  rbAmountDetail.Checked := False;
  MetodeDiskonStock := '';
end;

procedure TfrmSalesOrder.BersihkanTemp;
begin
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
              ' [SalDetailTemp]';
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
    qryDtl.Close;
    qryDtl.Open;
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
             ' [SoDeliveryDetailTemp]';
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
```

```
  end;
  qryDtlDelivery.Close;
  qryDtlDelivery.Open;
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
            ' [InvSerialNumberTemp]';
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
end;

procedure TfrmSalesOrder.OpenQrySQL;
begin
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;
end;

procedure TfrmSalesOrder.OpenQrySQL2;
begin
  with qrySQL2 do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;
end;

procedure TfrmSalesOrder.OpenQrySQL3;
begin
  with qrySQL3 do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
```

```
    Open;
  end;
end;

procedure TfrmSalesOrder.Tampilan;
begin
  if txtStatus.Caption = 'OPEN' then
  begin
    temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[SoRequisitionDetail] WHERE Requisition = ' +
              QuotedStr(edtSOR.Text);
    OpenQrySQL;
    if not qrySQL.Eof then
    begin
      qrySQL.First;
      while not qrySQl.Eof do
      begin
        with qryDtl do
        begin
          Close;
          Open;
          Insert;
          FieldByName('Line').AsInteger          :=
qrySQL.FieldByName('Line').AsInteger;
          FieldByName('StockCode').AsString      :=
qrySQL.FieldByName('StockCode').AsString;
          FieldByName('RibsToStockCode').AsString :=
qrySQL.FieldByName('RibsToStockCode').AsString;
          FieldByName('Roll').AsInteger          :=
qrySQL.FieldByName('Roll').AsInteger;
          FieldByName('Qty').AsFloat             :=
qrySQL.FieldByName('Qty').AsFloat;
          Post;
          Close;
          Open;
        end;
        totaldata := totaldata + 1;
        Warna        := Warna + 1;
        edtWarna.Text := IntToStr(Warna)+'W';
        qrySQL.Next;
      end;
    end;
  end
```

```
  else
  begin
    temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[SalDetail] WHERE SalesOrder = ' + QuotedStr(edtSO.Text);
    OpenQrySQL;
    if not qrySQL.Eof then
    begin
      qrySQL.First;
      while not qrySQl.Eof do
      begin
        with qryDtl do
        begin
          Close;
          Open;
          Insert;
          FieldByName('Line').AsInteger          :=
qrySQL.FieldByName('Line').AsInteger; //Line + 1;
          FieldByName('StockCode').AsString      :=
qrySQL.FieldByName('StockCode').AsString;
          FieldByName('RibsToStockCode').AsString :=
qrySQL.FieldByName('RibsToStockCode').AsString;
          FieldByName('Roll').AsInteger          :=
qrySQL.FieldByName('Roll').AsInteger;
          FieldByName('Qty').AsFloat             :=
qrySQL.FieldByName('DeliveryQty').AsFloat; //OrderQty;
          FieldByName('Price').AsFloat           :=
qrySQL.FieldByName('Price').AsFloat; //Price;
          FieldByName('DiskonPersen').AsCurrency  :=
qrySQL.FieldByName('DiskonPersen').AsCurrency;
          FieldByName('DiskonAmount').AsCurrency  :=
qrySQL.FieldByName('DiskonAmount').AsCurrency;
          FieldByName('Diskon').AsCurrency       :=
qrySQL.FieldByName('Diskon').AsCurrency; //DiskonDetail;
          Post;
          Close;
          Open;
          Total        := Total + qrySQL.FieldByName('TotalOrder').AsFloat;
          edtTotal.Text := FloatToStrF(Total, ffNumber, 10, 2);
        end;
        qrySQL.Next;
      end;
    end;
  end;
```

```
  qrySQL.Close;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[InvSerialNumberHistory] WHERE SalesOrderRequisition = ' +
            QuotedStr(edtSOR.Text) + ' AND SalesOrder = ' + QuotedStr('');
  OpenQrySQL;
  qrySQL.First;
  while not qrySQL.Eof do
  begin
    TotalQtySOR := 0;
    TotalRollSOR:= 0;
    temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[InvSerialNumberTemp] WHERE SerialNumber = ' +
              QuotedStr(qrySQL.FieldByName('SerialNumber').AsString);
    with qrySQL2 do
    begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
      Insert;
      FieldByName('SerialNumber').AsString  :=
qrySQL.FieldByName('SerialNumber').AsString;
      FieldByName('StockCode').AsString     :=
qrySQL.FieldByName('StockCode').AsString;
      for i := 1 to 24 do
      begin
        FieldByName('Qty'+IntToStr(i)).AsFloat:=
qrySQL.FieldByName('Qty'+IntToStr(i)).AsFloat;
        TotalQtySOR := TotalQtySOR +
qrySQL.FieldByName('Qty'+IntToStr(i)).AsFloat;
        if FieldByName('Qty'+IntToStr(i)).AsFloat <> 0 then
          TotalRollSOR:= TotalRollSOR + 1;
      end;
      FieldByName('TotalQty').AsFloat     := TotalQtySOR;
      FieldByName('TotalRoll').AsInteger  := TotalRollSOR;
      Post;
    end;
    qrySQL2.Close;
    qrySQL.Next;
  end;
  qrySQL.Close;
  if txtStatus.Caption <> 'OPEN' then
  begin
```

```
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[SoDeliveryDetail] WHERE SalesOrder = ' +
            QuotedStr(edtSO.Text);
  OpenQrySQL;
  qrySQL.First;
  while not qrySQL.Eof do
  begin
    temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[SoDeliveryDetailTemp] ';
    OpenQrySQL2;
    with qrySQL2 do
    begin
      Insert;
      FieldByName('Line').AsInteger        :=
qrySQL.FieldByName('Line').AsInteger; //Line + 1;
      FieldByName('DeliveryLine').AsInteger :=
qrySQL.FieldByName('DeliveryLine').AsInteger; //Line +1;
      FieldByName('StockCode').AsString     :=
qrySQL.FieldByName('StockCode').AsString;
      FieldByName('SerialNumber').AsString  :=
qrySQL.FieldByName('SerialNumber').AsString;
      FieldByName('DeliveryQty').AsFloat    :=
qrySQL.FieldByName('DeliveryQty').AsFloat; //OrderQty;
      FieldByName('Price').AsFloat          :=
qrySQL.FieldByName('Price').AsFloat; //Price;
      FieldByName('Total').AsFloat          :=
qrySQL.FieldByName('Price').AsFloat *
qrySQL2.FieldByName('DeliveryQty').AsFloat; //Price * OrderQty;
      for j := 1 to 24 do
        FieldByName('Qty'+IntToStr(j)).AsFloat :=
qrySQL.FieldByName('Qty'+IntToStr(j)).AsFloat;
      FieldByName('ViewWarna').AsString     :=
qrySQL.FieldByName('ViewWarna').AsString;
      Post;
    end;
    qrySQL.Next;
  end;
 end;
end;
end.
```

FormReturnReceipts.pas

```
unit FormReturnReceipts;

interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics,Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
  Data.DB, Data.Win.ADODB, Vcl.StdCtrls,Vcl.Mask, AdvDropDown,
  AdvCustomGridDropDown, AdvGridDropDown, Vcl.Buttons;
type
  TfrmReturReceipts = class(TForm)
    GroupBox1: TGroupBox;
    btnPost: TSpeedButton;
    GroupBox2: TGroupBox;
    edtStckCode: TEdit;
    edtCurrentQty: TEdit;
    edtRecCost: TEdit;
    Edit7: TEdit;
    Edit8: TEdit;
    Edit12: TEdit;
    Edit14: TEdit;
    Edit18: TEdit;
    Edit3: TEdit;
    edtSerialNumber: TEdit;
    edtDesc: TEdit;
    GroupBox3: TGroupBox;
    edtInfoCurrCost: TEdit;
    edtInfoAllocated: TEdit;
    edtInfoOnHand: TEdit;
    edtInfoAvailable: TEdit;
    edtInfoStckCode: TEdit;
    Edit4: TEdit;
    Edit2: TEdit;
    Edit16: TEdit;
    Edit13: TEdit;
    Edit11: TEdit;
    qry: TADOQuery;
    qry2: TADOQuery;
    qry3: TADOQuery;
    qrySQL: TADOQuery;
    edtReceiptQty: TEdit;
    Edit9: TEdit;
```

```
    edtLot: TEdit;
    edtNoRoll: TEdit;
    edtNewQty: TEdit;
    edtSO: TEdit;
    edtDO: TEdit;
    edtRecCostView: TEdit;
    Edit1: TEdit;
    StaticText8: TStaticText;
    edtSOAsli: TEdit;
    uom1: TEdit;
    uom2: TEdit;
    uom3: TEdit;
    uom4: TEdit;
    uom5: TEdit;
    uom6: TEdit;
    Edit5: TEdit;
    procedure FormShow(Sender: TObject);
    procedure Bersihkan;
    procedure OpenQry;
    procedure InfoStock;
    procedure edtReceiptQtyChange(Sender: TObject);
    procedure edtReceiptQtyExit(Sender: TObject);
    procedure edtReceiptQtyKeyPress(Sender: TObject; var Key: Char);
    procedure btnPostClick(Sender: TObject);
    procedure BersihkanTempSerial;
    procedure Update;
    procedure SaveMasterSerial;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmReturReceipts: TfrmReturReceipts;

implementation
uses GlobalUnit, FormComponent, FormTemp, StrUtils, Numbering,
FormSalesReturNew;
var temp :string;
    temp_on_hand, temp_unit_cost, temp_total, tmp_qty : Double;
    QtyA, QtyB, QtyAvailable, QtyOnDelivery : Double;
    x, RollOnHand, RollAvailable, RollOnDelivery : Integer;
```

```
{$R *.dfm}

procedure TfrmReturReceipts.btnPostClick(Sender: TObject);
begin
  if tmp_qty <> 0 then
  begin
    Update;
    BersihkanTempSerial;
    PostMessage(Self.Handle,wm_close,0,0);
    frmSalesReturNew.RefreshMaster;
    frmSalesReturNew.RefreshDetail;
    frmSalesReturNew.edtFind2.Text := edtSO.Text;
  end
    else
    begin
      ShowMessage('There is an empty data!');
    end;
end;

procedure TfrmReturReceipts.edtReceiptQtyChange(Sender: TObject);
begin
  if edtReceiptQty.Text='' then
  begin
    tmp_qty:=0;
  end
    else
    begin
      tmp_qty:=StrToFloat(edtReceiptQty.Text);
    end;
  temp_total := temp_on_hand + tmp_qty;
  if temp_total > 0 then
  begin
    btnPost.Enabled:=true;
  end
    else btnPost.Enabled:=false;
end;

procedure TfrmReturReceipts.edtReceiptQtyExit(Sender: TObject);
begin
  if(edtReceiptQty.Text='')then
  begin
    edtReceiptQty.Text:='';
  end
```

```
  else
  begin
    if ((StrToFloat(edtCurrentQty.Text)) >= (StrToFloat(edtReceiptQty.Text)))
AND
       (edtReceiptQty.Text <> '0') then
    begin
edtReceiptQty.Text:=FloatToStrF(StrToFloat(edtReceiptQty.Text),ffFixed, 8,2);
      btnPost.Enabled := True;
      QtyA := StrToFloat(edtCurrentQty.Text);
      QtyB := StrToFloat(edtReceiptQty.Text);
      edtNewQty.Text := (FloatToStrF((QtyA - QtyB),ffFixed, 8,2));
    end
    else if ((StrToFloat(edtCurrentQty.Text)) < (StrToFloat(edtReceiptQty.Text)))
then
    begin
      MessageDlg('Batas maksimal '+(edtCurrentQty.Text)+' Kg !', mtError,
[mbOK], 0);
      btnPost.Enabled := False;
      edtReceiptQty.Text:= '';
      edtNewQty.Text    := '';
    end
    else if (edtReceiptQty.Text = '0') then
    begin
      MessageDlg('Field ini tidak boleh bernilai 0!', mtError, [mbOK], 0);
      btnPost.Enabled := False;
      edtReceiptQty.Text:= '';
      edtNewQty.Text    := '';
    end;
  end;
end;

procedure TfrmReturReceipts.edtReceiptQtyKeyPress(Sender: TObject;
  var Key: Char);
var DecimalSeparator : char;
begin
  DecimalSeparator:='.';
  if not (Key in [#8, '0'..'9', DecimalSeparator]) then begin
    Key := #0;
  end;
end;

procedure TfrmReturReceipts.FormShow(Sender: TObject);
begin
```

```
  Bersihkan;
  InfoStock;
  edtLot.Text := 'RET-'+edtSerialNumber.Text+'/R'+edtNoRoll.Text;
end;

procedure TfrmReturReceipts.Bersihkan;
begin
  edtReceiptQty.Text:= '';
  edtNewQty.Text    := '';
  QtyA    := 0;
  QtyB    := 0;
end;

procedure TfrmReturReceipts.OpenQry;
begin
  with qry do
   begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
   end;
end;

procedure TfrmReturReceipts.InfoStock;
begin
  temp:=  'SELECT a.*, b.* FROM '+
             frmComponent.lblDB.Caption+' [InvMaster] a, '+
             frmComponent.lblDB.Caption+' [InvWarehouse] b '+
             'WHERE a.StockCode = b.StockCode AND a.StockCode = ' +
             QuotedStr(edtStckCode.Text);
  OpenQry;
  edtInfoStckCode.Text:=qry.FieldByName('StockCode').AsString;
  edtDesc.Text        :=qry.FieldByName('Description').AsString;
  edtInfoOnHand.Text  :=
FloatToStrF(qry.FieldByName('QtyOnHand').AsFloat,ffFixed, 8,2);
  edtInfoAvailable.Text     :=
FloatToStrF(qry.FieldByName('QtyAvailable').AsFloat,ffFixed, 8,2);
  edtInfoAllocated.Text     :=
FloatToStrF(qry.FieldByName('QtyAllocated').AsFloat,ffFixed, 8,2);
  temp_on_hand        :=qry.FieldByName('QtyOnHand').AsFloat;
  temp_unit_cost      :=qry.FieldByName('UnitCost').AsFloat;
  edtInfoCurrCost.Text:='Rp  '+FloatToStrF(temp_unit_cost,ffFixed, 8,2);
```

```
  uom1.Text := qry.FieldByName('Satuan').AsString;
  uom2.Text := qry.FieldByName('Satuan').AsString;
  uom3.Text := qry.FieldByName('Satuan').AsString;
  uom4.Text := qry.FieldByName('Satuan').AsString;
  uom5.Text := qry.FieldByName('Satuan').AsString;
  uom6.Text := qry.FieldByName('Satuan').AsString;
  RollOnHand:= qry.FieldByName('RollOnHand').AsInteger;
  RollAvailable:= qry.FieldByName('RollAvailable').AsInteger;
  QtyAvailable:= qry.FieldByName('QtyAvailable').AsFloat;
  RollOnDelivery:= qry.FieldByName('RollOnDelivery').AsInteger;
  QtyOnDelivery:= qry.FieldByName('QtyOnDelivery').AsFloat;
end;

procedure TfrmReturReceipts.BersihkanTempSerial;
begin
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
             'InvSerialNumberTemp';
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
end;

procedure TfrmReturReceipts.Update;
var query : string;
    cal_rec_cost, unit_cost,entered_cost, Qty, Cost : Double;
begin
  Qty := 0;
  Cost:= 0;
  Qty := StrToFloat(edtReceiptQty.Text);
  Cost:= StrToFloat(edtRecCost.Text);
query := 'SELECT * FROM '+frmComponent.lblDB.Caption+
             '[InvWarehouse] where StockCode='+
             QuotedStr(edtInfoStckCode.Text);
  with qry2 do
  begin
    Close;
    SQL.Clear;
    SQL.Add(query);
    Open;
```

```
    if Qty <> 0 then
    begin
      cal_rec_cost:= Qty*Cost;
      entered_cost:= cal_rec_cost/Qty;
    end
    else
    begin
      cal_rec_cost:= 0;
      entered_cost:= 0;
    end;
unit_cost:=((temp_total*temp_unit_cost)+(cal_rec_cost))/(temp_total+Qty);
    if not (frmComponent.adoWingga.InTransaction) then
    begin
      frmComponent.adoWingga.BeginTrans;
      try
        Edit;
        FieldByName('QtyOnHand').AsFloat:=temp_total;
        FieldByName('QtyAvailable').AsFloat:=QtyAvailable+tmp_qty;
        FieldByName('RollOnHand').AsInteger:=RollOnHand + 1;
        FieldByName('RollAvailable').AsInteger:=RollAvailable+ 1;
        FieldByName('UnitCost').AsFloat:=unit_cost;
        Post;
        with qry3 do
        begin
          Close;
          SQL.Clear;
          SQL.Add('SELECT TOP 1 * FROM'+
                    frmComponent.lblDB.Caption+' [InvMovements]');
          Open;
          Insert;
          FieldByName('StockCode').AsString:=edtInfoStckCode.Text;
          FieldByName('EntryDate').AsDateTime:=frmComponent.GetDate();
          FieldByName('MovementType').AsString := 'I';
          FieldByName('TrnQty').AsFloat :=StrToFloat(edtReceiptQty.Text);
          FieldByName('TrnValue').AsFloat := (Qty)*(unit_cost);
          FieldByName('EnteredCost').AsFloat := entered_cost;
          FieldByName('UnitCost').AsFloat := unit_cost;
          FieldByName('Reference').AsString := 'RET';
          FieldByName('SalesOrder').AsString := edtSOAsli.Text;
          FieldByName('SoReturn').AsString   := edtSO.Text;
          FieldByName('CostValue').AsFloat := (Qty * entered_cost);
          Post;
        end;
```

```
      temp := 'UPDATE SoReturnMaster SET TotalRetur = TotalRetur - 1' +
                 'WHERE SoReturn = ' + QuotedStr(edtSO.Text);
      qry2.Close;
      qry2.SQL.Clear;
      qry2.SQL.Add(temp);
      qry2.ExecSQL;
      SaveMasterSerial;
      frmComponent.adoWingga.CommitTrans;
      ShowMessage('Data Updated');
      Except
        frmComponent.adoWingga.RollbackTrans;
        MessageDlg('Save data failed !', mtError, [mbOK], 0);
    end;
  end;
 end;
end;

procedure TfrmReturReceipts.SaveMasterSerial;
var x :Integer;
begin
 temp := 'SELECT * FROM InvSerialNumberTemp WHERE ' +
            ' StockCode = ' + QuotedStr(edtStckCode.Text);
 OpenQry;
 if not qry.Eof then
 begin
   temp := 'SELECT * FROM InvSerialNumber';
   with qry2 do
   begin
     Close;
     SQL.Clear;
     SQL.Add(temp);
     Open;
     qry.First;
     while not qry.Eof do
     begin
       Insert;
       FieldByName('StockCode').AsString :=
qry.FieldByName('StockCode').AsString;
       FieldByName('SerialNumber').AsString := edtLot.Text;
       for x := 1 to 24 do
       begin
         if qry.FieldByName('Qty'+IntToStr(x)).AsFloat <> 0 then
```

```
          FieldByName('Qty' +IntToStr(x)).AsFloat :=
StrToFloat(edtReceiptQty.Text);
        end;
        Post;
        qry.Next;
      end;
    end;
  end;
end;
end.
```

FormInventory.pas

```
unit FormInventory;

interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes,Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
  Vcl.StdCtrls, Vcl.Grids, AdvObj,BaseGrid, AdvGrid, DBAdvGrid,
  Data.DB, Data.Win.ADODB, Vcl.Buttons, frxClass, StrUtils;
type
  TfrmInventory = class(TForm)
    dsInv: TDataSource;
    qryInv: TADOQuery;
    qryDtl: TADOQuery;
    GroupBox2: TGroupBox;
    DBAdvGrid1: TDBAdvGrid;
    edtFind: TEdit;
    Label1: TLabel;
    GroupBox1: TGroupBox;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    edtQtyonhand: TEdit;
    edtQtyavailable: TEdit;
    edtQtyallocated: TEdit;
    edtQtyondelivery: TEdit;
    edtQtysold: TEdit;
    Edit12: TEdit;
```

```
    edtRollonhand: TEdit;
    Edit14: TEdit;
    edtRollavailable: TEdit;
    Edit16: TEdit;
    edtRollallocated: TEdit;
    Edit18: TEdit;
    edtRollondelivery: TEdit;
    Edit20: TEdit;
    edtRollsold: TEdit;
    Edit22: TEdit;
    Edit23: TEdit;
    Edit24: TEdit;
    Edit25: TEdit;
    Edit26: TEdit;
    Edit27: TEdit;
    Edit28: TEdit;
    Edit29: TEdit;
    Edit30: TEdit;
    Edit31: TEdit;
    edtStockCode: TEdit;
    procedure FormShow(Sender: TObject);
    procedure OpenQryInv;
    procedure OpenQryDtl;
    procedure DBAdvGrid1ClickCell(Sender: TObject; ARow, ACol: Integer);
    procedure edtFindChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmInventory: TfrmInventory;

implementation
uses  FormComponent;
{$R *.dfm}
var
temp, temp_find1, temp_find2, temp_find3, temp_stck : string;

procedure TfrmInventory.DBAdvGrid1ClickCell(Sender: TObject;
ARow,ACol: Integer);
begin
```

```
temp_stck:=qryInv.FieldByName('StockCode').AsString;
temp:= 'SELECT QtyOnHand, RollOnHand, QtyAvailable, '+
        'RollAvailable, QtyAllocated, RollAllocated, QtyOnDelivery, '+
        'RollOnDelivery, MtdQtySold, RollSold, StockCode FROM ' +
        frmComponent.lblDB.Caption +
        '[InvWarehouse] where StockCode Like ' + QuotedStr(temp_stck);
OpenQryDtl;
if (frmComponent.lblUserName.Caption = '1') then
begin
  edtQtyonhand.Visible  := False;
  edtRollonhand.Visible := False;
  edtQtyallocated.Visible   := False;
  edtRollallocated.Visible  := False;
  edtQtyondelivery.Visible  := False;
  edtRollondelivery.Visible := False;
  edtQtysold.Visible        := False;
  edtRollsold.Visible       := False;
  Edit2.Visible:= False;
  Edit4.Visible := False;
  Edit5.Visible := False;
  Edit6.Visible := False;
  Edit12.Visible:= False;
  Edit16.Visible:= False;
  Edit18.Visible:= False;
  Edit20.Visible:= False;
  Edit22.Visible:= False;
  Edit24.Visible:= False;
  Edit25.Visible:= False;
  Edit26.Visible:= False;
  Edit27.Visible:= False;
  Edit29.Visible:= False;
  Edit30.Visible:= False;
  Edit31.Visible:= False;
end;
GroupBox1.Visible:= True;
edtStockCode.Text     := qryDtl.FieldByName('StockCode').AsString;
edtQtyonhand.Text     := qryDtl.FieldByName('QtyOnHand').AsString;
edtRollonhand.Text    := qryDtl.FieldByName('RollOnHand').AsString;
edtQtyavailable.Text  := qryDtl.FieldByName('QtyAvailable').AsString;
edtRollavailable.Text := qryDtl.FieldByName('RollAvailable').AsString;
edtQtyallocated.Text  := qryDtl.FieldByName('QtyAllocated').AsString;
edtRollallocated.Text := qryDtl.FieldByName('RollAllocated').AsString;
edtQtyondelivery.Text := qryDtl.FieldByName('QtyOnDelivery').AsString;
```

```
  edtRollondelivery.Text:= qryDtl.FieldByName('RollOnDelivery').AsString;
  edtQtysold.Text       := qryDtl.FieldByName('MtdQtySold').AsString;
  edtRollsold.Text      := qryDtl.FieldByName('RollSold').AsString;
end;

procedure TfrmInventory.edtFindChange(Sender: TObject);
begin
  temp_find1 := UpperCase('%'+edtFind.Text+'%');
  temp_find2 := UpperCase(edtFind.Text+'%');
  temp_find3 := UpperCase('%'+edtFind.Text);
  temp:='SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[InvMaster] a, [InvWarehouse] b where a.StockCode =' +
            ' b.StockCode '+
            'AND ((a.StockCode LIKE '+ QuotedStr(temp_find1) + ')' +
            'OR (a.StockCode LIKE '+ QuotedStr(temp_find2) + ')' +
            'OR (a.StockCode LIKE '+ QuotedStr(temp_find3) + ')' +
            'OR (a.Description LIKE '+ QuotedStr(temp_find1) + ')' +
            'OR (a.Description LIKE '+ QuotedStr(temp_find2) + ')' +
            'OR (a.Description LIKE '+ QuotedStr(temp_find3) + '))';
  OpenQryInv;
end;

procedure TfrmInventory.FormShow(Sender: TObject);
begin
  edtFind.Text := '';
  GroupBox1.Visible:= False;
  temp:= 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[InvMaster]';
  OpenQryInv;
  qryDtl.Close;
end;

procedure TfrmInventory.OpenQryInv;
begin
  with qryInv do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;
end;
```

```
procedure TfrmInventory.OpenQryDtl;
begin
  with qryDtl do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
  end;
end;
end.
```

FormARInvoice.pas

```
unit FormARInvoice;

interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
  frxClass, frxDBSet, Data.DB, Data.Win.ADODB, Vcl.Menus,
  Vcl.StdCtrls, Vcl.Mask, AdvDropDown, AdvCustomGridDropDown,
  AdvGridDropDown, Vcl.Buttons, Vcl.Grids, AdvObj, BaseGrid, AdvGrid,
  DBAdvGrid;
type
  TfrmARInvoice = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    btnPostInvoice: TSpeedButton;
    Label2: TLabel;
    edtInvoice: TEdit;
    GroupBox2: TGroupBox;
    gridDelivery: TDBAdvGrid;
    GroupBox3: TGroupBox;
    Label3: TLabel;
    edtTotal: TEdit;
    ds: TDataSource;
    qrySQL: TADOQuery;
    qryTemp: TADOQuery;
    qryDelivery: TADOQuery;
    qrySQL2: TADOQuery;
    qryDeliverySalesOrder: TWideStringField;
```

```
    qryDeliveryDelivery: TWideStringField;
    qryDeliveryDeliveryDate: TDateTimeField;
    qryDeliveryTotal: TFMTBCDField;
    cbCustomer: TAdvGridDropDown;
    qryInvoice2: TADOQuery;
    StaticText2: TStaticText;
    StaticText1: TStaticText;
    qryDeliveryCustomerCode: TWideStringField;
    qryDeliveryTermsCode: TStringField;
    qryInvoiceBaru: TADOQuery;
    frxDBDatasetInvoiceBaru: TfrxDBDataset;
    frxReportInvoiceBaru: TfrxReport;
    qryInvoiceBaruRow: TLargeintField;
    qryInvoiceBaruArInvoice: TWideStringField;
    qryInvoiceBaruDelivery: TWideStringField;
    qryInvoiceBaruSalesOrder: TWideStringField;
    qryInvoiceBaruAmount: TBCDField;
    cbArTerms: TComboBox;
    qryInvoiceBaruDeliveryDate: TDateTimeField;
    qryDeliveryInvoice: TWideStringField;
    function CekValid : Boolean;
    procedure IsiCombo;
    procedure TambahTemp;
    procedure KurangiTemp;
    procedure BersihkanTemp;
    procedure Bersihkan;
    procedure OpenQrySQL;
    procedure RefreshQryInvoiceBaru;
    procedure BuatNoArInvoice;
    procedure UpdateNoArInvoice;
    procedure CekMetodeNoArInvoice;
    procedure InvoiceFlag;
    procedure UpdateInventory;
    procedure SaveInvoice;
    procedure Tax;
    procedure PrintInvoice;
    procedure btnPostInvoiceClick(Sender: TObject);
    procedure cbCustomerSelect(Sender: TObject);
    procedure edtInvoiceExit(Sender: TObject);
    procedure gridDeliveryCheckBoxClick(Sender: TObject; ACol, ARow:
Integer;
      State: Boolean);
    procedure FormShow(Sender: TObject);
```

```
    procedure Refresh;
    procedure cbArTermsChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmARInvoice: TfrmARInvoice;
  temp, SO, StatusPajak  : String;
  metodeNoARInvoice, NoARInvoice, NextNoARInvoice : String;
  NextSuffix : Integer;
  InvoiceTotal, TotalPajak : Double;

implementation
uses FormComponent, GlobalUnit, Numbering;

{$R *.dfm}

procedure TfrmARInvoice.InvoiceFlag;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[SoDeliveryMaster] WHERE Delivery = ' +
            QuotedStr(qryTemp.FieldByName('Delivery').AsString);
  with qrySQL2 do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
    Edit;
    FieldByName('Invoice').AsString := edtInvoice.Text;
    Post;
  end;
end;

procedure TfrmARInvoice.UpdateInventory;
begin
  temp := 'SELECT * FROM ' +frmComponent.lblDB.Caption +
            ' ArInvoiceDetail a,  ' +
            frmComponent.lblDB.Caption +'SoDeliveryDetail b ' +
```

```
            'WHERE a.Delivery = b.Delivery AND a.ArInvoice = ' +
QuotedStr(edtInvoice.Text);
  with qryInvoice2 do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
    First;
    while not (Eof) do
    begin
      if FieldByName('StockCode').AsString <> '' then
      begin
        temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
                '[InvWarehouse] WHERE StockCode = ' +
                QuotedStr(FieldByName('StockCode').AsString);
        OpenQrySQL;
        qrySQL.Edit;
        qrySQL.FieldByName('QtyOnDelivery').AsFloat :=
qrySQl.FieldByName('QtyOnDelivery').AsFloat -
FieldByName('DeliveryQty').AsFloat;
        qrySQL.FieldByName('RollOnDelivery').AsInteger :=
qrySQl.FieldByName('RollOnDelivery').AsInteger -
FieldByName('DeliveryRoll').AsInteger;
        qrySQL.FieldByName('MtdQtySold').AsFloat :=
qrySQL.FieldByName('MtdQtySold').AsFloat +
FieldByName('DeliveryQty').AsFloat;
        qrySQL.FieldByName('RollSold').AsInteger :=
qrySQl.FieldByName('RollSold').AsInteger +
FieldByName('DeliveryRoll').AsInteger;
        qrySQL.Post;
      end;
      Next;
    end;
  end;
end;

procedure TfrmARInvoice.btnPostInvoiceClick(Sender: TObject);
begin
  if CekValid then
  begin
    if not frmComponent.adoWingga.InTransaction then
    begin
```

```
    frmComponent.adoWingga.BeginTrans;
    try
      BuatNoArInvoice;
      SaveInvoice;
      temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
                '[ArInvoiceTemp]';
      OpenQrySQL;
      if not qrySQL.Eof then
      begin
        temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
                  '[ArInvoiceDetail] WHERE ArInvoice = ' +
                 QuotedStr(edtInvoice.Text);
        OpenQrySQL;
        temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
                 '[ArInvoiceTemp] ';
        with qryTemp do
        begin
          Close;
          SQL.Clear;
          SQL.Add(temp);
          Open;
        end;
        qryTemp.First;
        while not qryTemp.Eof do
        begin
          qrySQL.Insert;
          qrySQL.FieldByName('ArInvoice').AsString := edtInvoice.Text;
          qrySQL.FieldByName('SalesOrder').AsString :=
qryTemp.FieldByName('SalesOrder').AsString;
          qrySQL.FieldByName('Delivery').AsString :=
qryTemp.FieldByName('Delivery').AsString;
          qrySQL.FieldByName('Amount').AsFloat :=
qryTemp.FieldByName('Total').AsFloat;
          qrySQL.Post;
          InvoiceFlag;
          qryTemp.Next;
        end;
        frmComponent.adoWingga.CommitTrans;
        MessageDlg('Invoice posted !'+#13#10+'Invoice number : ' +
edtInvoice.Text, mtConfirmation, [mbOK], 0);
        PrintInvoice;
        UpdateInventory;
        UpdateNoArInvoice;
```

```
        qryDelivery.Close;
        BersihkanTemp;
        Bersihkan;
        CekMetodeNoArInvoice;
        RefreshQryInvoiceBaru;
      end;
      Except
      on E: Exception do
      begin
      frmComponent.adoWingga.RollbackTrans;
      ShowMessage(e.Message);
      end;
    end;
  end; // end try
 end; // end if ado.InTrans
end;

procedure TfrmARInvoice.cbArTermsChange(Sender: TObject);
begin
 Refresh;
end;

procedure TfrmARInvoice.cbCustomerSelect(Sender: TObject);
begin
 if (StringReplace(edtInvoice.Text,' ',[rfReplaceAll,rfIgnoreCase]) <> '') then
 begin
  BersihkanTemp;
  InvoiceTotal := 0;
  edtTotal.Text := FloatToStrF(InvoiceTotal, ffNumber, 10, 2);
  Refresh;
 end
 else
 begin
  edtInvoice.SetFocus;
 end;
end;

procedure TfrmARInvoice.Refresh;
begin
 with qryDelivery do
  begin
   Close;
   Parameters[0].Value := cbCustomer.Text;
```

```
      Parameters[1].Value := cbArTerms.Text;
      Open;
      if not Eof then
      begin
        gridDelivery.Enabled := True;
      end;
    end;
end;

function TfrmARInvoice.CekValid : Boolean;
begin
  CekValid := True;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
              '[ArInvoiceTemp]';
  OpenQrySQL;
  if cbCustomer.Text = '' then
  begin
    MessageDlg('Customer required!',mtError,[mbOK],0);
    cbCustomer.SetFocus;
    CekValid := False;
  end
  else if (qrySQL.Eof) then
  begin
    MessageDlg('Data empty!',mtError,[mbOK],0);
    CekValid := False;
  end
end;

procedure TfrmARInvoice.edtInvoiceExit(Sender: TObject);
begin
    if (StringReplace(edtInvoice.Text,' ','',[rfReplaceAll,rfIgnoreCase]) = '') then
    begin
      qryDelivery.Close;
      BersihkanTemp;
      edtInvoice.SetFocus;
    end;
end;

procedure TfrmARInvoice.FormShow(Sender: TObject);
begin
  Bersihkan;
  IsiCombo;
  CekMetodeNoArInvoice;
```

```
end;

procedure TfrmARInvoice.gridDeliveryCheckBoxClick(Sender: TObject; ACol,
  ARow: Integer; State: Boolean);
begin
  if State then
  begin
    TambahTemp;
    InvoiceTotal := InvoiceTotal + qryDelivery.FieldByName('Total').AsFloat;
  end
  else
  begin
    KurangiTemp;
    InvoiceTotal := InvoiceTotal - qryDelivery.FieldByName('Total').AsFloat;
  end;
  edtTotal.Text := FloatToStrF(InvoiceTotal, ffNumber, 10, 2);
end;

procedure TfrmARInvoice.OpenQrySQL;
begin
  with qrySQL do
   begin
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
   end;
end;

procedure TfrmARInvoice.RefreshQryInvoiceBaru;
begin
  with qryInvoiceBaru do
   begin
      Close;
      Parameters[0].Value := edtInvoice.Text;
      Parameters[1].Value := cbCustomer.Text;
      Open;
   end;
end;

procedure TfrmARInvoice.Tax;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
```

```
            '[SoMaster] WHERE SalesOrder = ' +
            QuotedStr(qryDelivery.FieldByName('SalesOrder').AsString);
  with qrySQL2 do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    Open;
    StatusPajak := FieldByName('Taxable').AsString;
    if StatusPajak = 'Y' then
    begin
      StatusPajak := 'Y';
      TotalPajak := InvoiceTotal * 0.1;
    end
    else
    begin
      StatusPajak := 'N';
      TotalPajak := 0;
    end;
  end;
end;

procedure TfrmARInvoice.PrintInvoice;
var
  Memo: TfrxMemoView;
  Component: TfrxComponent;
begin
  RefreshQryInvoiceBaru;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[ArCustomer] WHERE CustomerCode = ' +
            QuotedStr(cbCustomer.Text);
  OpenQrySQL;
  Component := frxReportInvoiceBaru.FindObject('Invoice');
  Memo := Component as TfrxMemoView;
  Memo.Text := edtInvoice.Text;
  Component := frxReportInvoiceBaru.FindObject('Date');
  Memo := Component as TfrxMemoView;
  Memo.Text := FormatDateTime('dd MMMM yyy',frmComponent.GetDate());
  Component := frxReportInvoiceBaru.FindObject('CustomerName');
  Memo := Component as TfrxMemoView;
  Memo.Text := UpperCase(qrySQL.FieldByName('CustomerName').AsString);
  Component := frxReportInvoiceBaru.FindObject('Alamat');
  Memo := Component as TfrxMemoView;
```

```
  Memo.Text :=  qrySQL.FieldByName('Address').AsString;
  Component := frxReportInvoiceBaru.FindObject('Alamat2');
  Memo := Component as TfrxMemoView;
  Memo.Text :=  qrySQL.FieldByName('City').AsString;
  Component := frxReportInvoiceBaru.FindObject('Total');
  Memo := Component as TfrxMemoView;
  Memo.Text := edtTotal.Text;
  Component := frxReportInvoiceBaru.FindObject('Pajak');
  Memo := Component as TfrxMemoView;
  Memo.Text := FloatToStrF(TotalPajak, ffNumber, 10, 2);
  Component := frxReportInvoiceBaru.FindObject('GrandTotal');
  Memo := Component as TfrxMemoView;
  Memo.Text := FloatToStrF(InvoiceTotal + TotalPajak, ffNumber, 10,2);
  frxReportInvoiceBaru.ShowReport;
end;

procedure TfrmARInvoice.SaveInvoice;
begin
  Tax;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[ArInvoice] WHERE Customer ='+QuotedStr(cbCustomer.Text)+
             ' AND ArInvoice = ' + QuotedStr(edtInvoice.Text);
  OpenQrySQL;
  with qrySQL do
  begin
    Insert;
    FieldByName('Customer').AsString := cbCustomer.Text;
    FieldByName('ArInvoice').AsString := edtInvoice.Text;
    FieldByName('InvoiceDate').AsDateTime := frmComponent.GetDate();
    FieldByName('OriginalAmount').AsFloat := InvoiceTotal;
    FieldByName('Taxable').AsString := StatusPajak;
    FieldByName('TaxAmount').AsFloat := TotalPajak;
    FieldByName('Payment').AsFloat := 0;
    FieldByName('TermsCode').AsString :=
qryDelivery.FieldByName('TermsCode').AsString;
    temp := 'SELECT DATEADD(day, '+
               (qryDelivery.FieldByName('TermsCode').AsString) +
                ', FORMAT(SYSDATETIME(),' +
               QuotedStr('yyyy-MM-dd HH:mm:ss.fff') + ')) AS DueDate';
    with qrySQL2 do
    begin
      Close;
      SQL.Clear;
```

```
      SQL.Add(temp);
      Open;
    end;
    FieldByName('DueDate').AsDateTime :=
(qrySQL2.FieldByName('DueDate').AsDateTime);
    FieldByName('AddID').AsString := frmComponent.lblUserName.Caption;
    Post;
  end;
end;

procedure TfrmARInvoice.Bersihkan;
begin
  edtInvoice.Clear;
  cbArTerms.ItemIndex:=-1;
  qryDelivery.Close;
  edtTotal.Text := '0';
  InvoiceTotal := 0;
end;

procedure TfrmARInvoice.BersihkanTemp;
begin
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
             '[ArInvoiceTemp] ';
   with qrySQL do
   begin
     Close;
     SQL.Clear;
     SQL.Add(temp);
     ExecSQL;
   end;
end;

procedure TfrmARInvoice.UpdateNoArInvoice;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[SetupCode] WHERE [Transaction] = ' +
             QuotedStr('AR Invoice');
  OpenQrySQL;
  with qrySQL do
  begin
     Edit;
     FieldByName('Next').AsString := NextNoArInvoice;
     FieldByName('Suffix').AsInteger := NextSuffix;
```

```
    Post;
  end;
end;

procedure TfrmARInvoice.BuatNoArInvoice;
var Panjang : Integer;
   Prefix : String;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[SetupCode] WHERE [Transaction] = ' +
            QuotedStr('AR Invoice');
  OpenQrySQL;
  with qrySQL do
  begin
    Prefix :=  FieldByName('Prefix').AsString;
    Panjang := FieldByName('Length').AsInteger;
    NoArInvoice := (FieldByName('Next').AsString);
    NextSuffix := (FieldByName('Suffix').AsInteger + 1);
    NextNoArInvoice := UpdateCode(IntToStr(NextSuffix),Prefix,Panjang);
    edtInvoice.Text := NoArInvoice;
    edtInvoice.Enabled := False;
    edtInvoice.Color := clBtnFace;
  end;
end;

procedure TfrmARInvoice.CekMetodeNoArInvoice;
var Panjang : Integer;
   Prefix : String;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[SetupCode] WHERE [Transaction] = ' +
            QuotedStr('AR Invoice');
  OpenQrySQL;
  with qrySQL do
  begin
    edtInvoice.Enabled := False;
    edtInvoice.Color := clBtnFace;
    edtInvoice.Text := 'NEW';
  end;
end;

procedure TfrmARInvoice.TambahTemp;
begin
```

```
  if not qryDelivery.Eof then
  begin
    with qrySQL do
    begin
      temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
                '[ArInvoiceTemp]';
      OpenQrySQL;
      Insert;
      FieldByName('SalesOrder').AsString :=
qryDelivery.FieldByName('SalesOrder').AsString;
      FieldByName('Delivery').AsString :=
qryDelivery.FieldByName('Delivery').AsString;
      FieldByName('Total').AsFloat :=
qryDelivery.FieldByName('Total').AsFloat;
      Post;
    end;
  end;
end;

procedure TfrmARInvoice.KurangiTemp;
begin
  if not qryDelivery.Eof then
  begin
    with qrySQL do
    begin
      temp:=  'DELETE FROM ' + frmComponent.lblDB.Caption +
                '[ArInvoiceTemp] WHERE SalesOrder = ' +
                QuotedStr(qryDelivery.FieldByName('SalesOrder').AsString) +
                ' AND Delivery = ' +
                QuotedStr(qryDelivery.FieldByName('Delivery').AsString);
      Close;
      SQL.Clear;
      SQL.Add(temp);
      ExecSQL;
    end;
  end;
end;

procedure TfrmARInvoice.IsiCombo;
begin
  cbCustomer.Items.Clear;
  temp := 'SELECT CustomerCode, CustomerName FROM ' +
             frmComponent.lblDB.Caption +
```

```
            '[ArCustomer] ORDER BY CustomerCode ';
  OpenQrySQL;
  if qrySQL.Eof then
  begin
    Messagedlg('Customer is empty. Ask your administrator!', mtError,
[mbok],0);
    PostMessage(Self.Handle,wm_close,0,0);  //untuk membatalkan menampilkan
form
  end;
  while not qrySQL.Eof do
  begin
    with cbCustomer.Items.Add do
    begin
      Text.Add(qrySQL.FieldByName('CustomerCode').AsString);
      Text.Add(qrySQL.FieldByName('CustomerName').AsString);
    end;
    qrySQL.Next;
  end;
end;
end.
```

FormArPayment.pas

```
unit FormArPayment;

interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
  frxClass, frxDBSet, Data.DB, Vcl.Menus, Data.Win.ADODB, Vcl.Grids,
  AdvObj, BaseGrid, AdvGrid, DBAdvGrid, Vcl.StdCtrls, Vcl.ComCtrls,
  Vcl.Mask, AdvDropDown, AdvCustomGridDropDown,
  AdvGridDropDown,Vcl.Buttons;
type
  TfrmArPayment = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    btnPost: TSpeedButton;
    cbCustomer: TAdvGridDropDown;
    StaticText1: TStaticText;
    edtPayment: TEdit;
    GroupBox3: TGroupBox;
```

```
Edit1: TEdit;
edtPaymentAmount: TEdit;
Edit2: TEdit;
dtpTgl: TDateTimePicker;
Edit3: TEdit;
cbAccount: TAdvGridDropDown;
Edit4: TEdit;
edtBalance: TEdit;
Edit7: TEdit;
edtTotal: TEdit;
Edit6: TEdit;
edtDeposito: TEdit;
Edit5: TEdit;
edtInvoiceTotal: TEdit;
GroupBox2: TGroupBox;
grdInvoice: TDBAdvGrid;
qryInvoice: TADOQuery;
ds: TDataSource;
qrySQL: TADOQuery;
qryMs: TADOQuery;
qryDtl: TADOQuery;
qryPayment: TADOQuery;
qryPaymentInvoice: TWideStringField;
qryPaymentAmount: TBCDField;
qryPaymentRow: TLargeintField;
frxDBDatasetPayment: TfrxDBDataset;
frxReportPayment: TfrxReport;
qryInvoiceArInvoice: TWideStringField;
qryInvoiceInvoiceDate: TDateTimeField;
qryInvoiceOriginalAmount: TBCDField;
qryInvoiceTaxAmount: TBCDField;
qryInvoiceBalance: TFMTBCDField;
qryInvoicePayment: TBCDField;
edtGiro: TEdit;
qryInvoiceAmount: TFMTBCDField;
qryPaymentArPayment: TWideStringField;
function CekValid : Boolean;
procedure IsiCombo;
procedure Bersihkan;
procedure BersihkanInput;
procedure BersihkanTemp;
procedure OpenQrySQL;
procedure RefreshQryPayment;
```

```
    procedure TambahTemp;
    procedure KurangiTemp;
    procedure BuatNoPayment;
    procedure CekMetodeNoPayment;
    procedure UpdateNoPayment;
    procedure UpdateTransaksi;
    procedure UpdateJumlahBayar;
    procedure UpdateBalance;
    procedure InsertHistoryBayar;
    procedure InsertMasterPayment;
    procedure FormShow(Sender: TObject);
    procedure btnPostClick(Sender: TObject);
    procedure grdInvoiceCheckBoxClick(Sender: TObject; ACol, ARow: Integer;
      State: Boolean);
    procedure edtPaymentAmountKeyPress(Sender: TObject; var Key: Char);
    procedure cbAccountChange(Sender: TObject);
    procedure cbCustomerChange(Sender: TObject);
    procedure PrintPayment;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmArPayment: TfrmArPayment;
  temp : String;
  TotalPayment, ArBalance, TotalDeposit, TotalSeluruh,
  TotalTagihan :  Double;
  NoPayment, NextNoPayment, metodeNoPayment : String;
  NextSuffix : Integer;
  JumlahBayar : Currency;
  SisaDeposito : Currency;

implementation
uses FormComponent, GlobalUnit, Numbering;

{$R *.dfm}

function TfrmArPayment.CekValid : Boolean;
var
  InputBayar, Sisa, Deposito, TotalTagihan : Double;
begin
```

```
  InputBayar := StrToFloat(edtPaymentAmount.Text);
  Deposito := TotalDeposit;
  TotalTagihan := TotalPayment;
  CekValid := True;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[TblBank] WHERE Receivable = ' + QuotedStr(cbAccount.Text);
  OpenQrySQL;
  if cbCustomer.Text = '' then
  begin
    Messagedlg('Customer required!', mtError, [mbok],0);
    cbCustomer.SetFocus;
    CekValid := False;
  end
  else if (StringReplace(edtPayment.Text, ' ', '', [rfReplaceAll,rfIgnoreCase]) = '')
then
  begin
    Messagedlg('Payment number required!', mtError, [mbok],0);
    edtPayment.SetFocus;
    CekValid := False;
  end
  else if cbAccount.Text = '' then
  begin
    Messagedlg('Account required!', mtError, [mbok],0);
    cbAccount.SetFocus;
    CekValid := False;
  end
  else if (StringReplace(edtInvoiceTotal.Text, ' ', '', [rfReplaceAll,rfIgnoreCase])
= '')
        OR (TotalPayment = 0) then
  begin
    Messagedlg('Invoice total required!', mtError, [mbok],0);
    CekValid := False;
  end
  else if ((StringReplace(edtPaymentAmount.Text, ' ', '',
[rfReplaceAll,rfIgnoreCase]) = '')
        OR (StrToFloat(edtPaymentAmount.Text) = 0)) AND (TotalDeposit = 0)
then
  begin
    Messagedlg('Payment amount required!', mtError, [mbok],0);
    edtPaymentAmount.SetFocus;
    CekValid := False;
  end
  else if (StrToFloat(edtPaymentAmount.Text) > TotalPayment) then
```

```
  begin
    Messagedlg('Payment amount greater than invoice total!', mtError, [mbok],0);
    edtPaymentAmount.SetFocus;
    CekValid := False;
  end;
end;

procedure TfrmArPayment.IsiCombo;
begin
  cbCustomer.Items.Clear;
  temp := 'SELECT CustomerCode, CustomerName FROM ' +
              frmComponent.lblDB.Caption +
              '[ArCustomer] ORDER BY CustomerCode ';
  OpenQrySQL;
  if qrySQL.Eof then
  begin
    Messagedlg('Customer is empty. Ask your administrator!', mtError,
[mbok],0);
    PostMessage(Self.Handle,wm_close,0,0);
  end;
  while not qrySQL.Eof do
  begin
    with cbCustomer.Items.Add do
    begin
      Text.Add(qrySQL.FieldByName('CustomerCode').AsString);
      Text.Add(qrySQL.FieldByName('CustomerName').AsString);
    end;
    qrySQL.Next;
  end;
  cbAccount.Items.Clear;
  temp := 'SELECT Receivable , Description FROM ' +
              frmComponent.lblDB.Caption + '[TblBank] ';
  OpenQrySQL;
  if qrySQL.Eof then
  begin
    Messagedlg('Account Receivable is empty. Ask your administrator!', mtError,
[mbok],0);
    PostMessage(Self.Handle,wm_close,0,0);  //untuk membatalkan menampilkan
form
  end;
  while not qrySQL.Eof do
  begin
    with cbAccount.Items.Add do
```

```
    begin
      Text.Add(qrySQL.FieldByName('Receivable').AsString);
      Text.Add(qrySQL.FieldByName('Description').AsString);
    end;
    qrySQL.Next;
  end;
end;

procedure tfrmArPayment.BersihkanTemp;
begin
  temp := 'DELETE FROM ' + frmComponent.lblDB.Caption +
             '[ArPaymentTemp]';
  with qrySQL do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
end;

procedure tfrmArPayment.BersihkanInput;
begin
  edtBalance.Text := '0';
  cbAccount.ItemIndex := -1;
  edtInvoiceTotal.Text := '0';
  edtDeposito.Text := '0';
  edtTotal.Text := '0';
  edtPaymentAmount.Text := '0';
  TotalPayment := 0;
  TotalDeposit := 0;
  TotalTagihan := 0;
end;

procedure tfrmArPayment.Bersihkan;
begin
  cbCustomer.ItemIndex := -1;
  dtpTgl.DateTime := Now;
  BersihkanInput;
end;

procedure TfrmArPayment.OpenQrySQL;
begin
```

```
  with qrySQL do
   begin
     Close;
     SQL.Clear;
     SQL.Add(temp);
     Open;
   end;
end;

procedure TfrmArPayment.RefreshQryPayment;
begin
  with qryPayment do
   begin
     Close;
     Parameters[0].Value := edtPayment.Text;
     Open;
   end;
end;

procedure TfrmArPayment.TambahTemp;
begin
  if not (qryInvoice.Eof) then
  begin
    with qrySQL do
    begin
      temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
                '[ArPaymentTemp]';
      OpenQrySQL;
      Insert;
      FieldByName('Invoice').AsString :=
qryInvoice.FieldByName('ArInvoice').AsString;
      FieldByName('Amount').AsFloat :=
qryInvoice.FieldByName('Balance').AsFloat;
      Post;
    end;
  end;
end;

procedure TfrmArPayment.KurangiTemp;
begin
  if not qryInvoice.Eof then
  begin
    with qrySQL do
```

```
    begin
      temp:=  'DELETE FROM ' + frmComponent.lblDB.Caption +
                '[ArPaymentTemp] WHERE Invoice = ' +
                QuotedStr(qryInvoice.FieldByName('ArInvoice').AsString);
      Close;
      SQL.Clear;
      SQL.Add(temp);
      ExecSQL;
    end;
  end;
end;

procedure TfrmArPayment.BuatNoPayment;
var Panjang : Integer;
    Prefix : String;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[SetupCode] WHERE [Transaction] = ' +
             QuotedStr('AR Payment');
  OpenQrySQL;
  with qrySQL do
  begin
      Prefix :=  FieldByName('Prefix').AsString;
      Panjang := FieldByName('Length').AsInteger;
      NoPayment := (FieldByName('Next').AsString);
      NextSuffix := (FieldByName('Suffix').AsInteger + 1);
      NextNoPayment := UpdateCode(IntToStr(NextSuffix),Prefix,Panjang);
      edtPayment.Text := NoPayment;
      edtPayment.Enabled := False;
  end;
end;

procedure TfrmArPayment.cbAccountChange(Sender: TObject);
begin
    edtBalance.Visible := True;
    edtGiro.Visible := False;
    Edit4.Text := 'Balance';
    temp := 'SELECT Receivable, Description, Balance FROM ' +
               frmComponent.lblDB.Caption + '[TblBank] WHERE ' +
               'Receivable = ' + QuotedStr(cbAccount.Text);
    OpenQrySQL;
    ArBalance := qrySQL.FieldByName('Balance').AsFloat;
    edtBalance.Text := FloatToStrF(ArBalance, ffNumber, 10, 2);
```

```
end;

procedure TfrmArPayment.cbCustomerChange(Sender: TObject);
begin
  with qryInvoice do
  begin
    Close;
    Parameters[0].Value := cbCustomer.Text;
    Open;
  end;
  BersihkanInput;
  BersihkanTemp;
end;

procedure TfrmArPayment.CekMetodeNoPayment;
var Panjang : Integer;
    Prefix : String;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[SetupCode] WHERE [Transaction] = ' +
             QuotedStr('AR Payment');
  OpenQrySQL;
  with qrySQL do
  begin
      edtPayment.Text := 'NEW';
      edtPayment.Enabled := False;
  end;
end;

procedure TfrmArPayment.UpdateNoPayment;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[SetupCode] WHERE [Transaction] = ' +
             QuotedStr('AR Payment');
  OpenQrySQL;
  with qrySQL do
  begin
      Edit;
      FieldByName('Next').AsString := NextNoPayment;
      FieldByName('Suffix').AsInteger := NextSuffix;
      Post;
  end;
end;
```

```
procedure TfrmArPayment.UpdateTransaksi;
var
   NoUrut,i,j : integer;
   state: boolean;
   bookmark:string;
begin
  if not frmComponent.adoWingga.InTransaction then
  begin
   frmComponent.adoWingga.BeginTrans;
   try
     JumlahBayar := StrToCurr(edtPaymentAmount.Text) + Abs(TotalDeposit);
     temp :=  'SELECT * FROM ' + frmComponent.lblDB.Caption +
                '[ArPaymentTemp]';
     OpenQrySQL;
       qrySQL.First;
       while not (qrySQL.Eof) do
       begin
        if ((qrySQL.FieldByName('Amount').AsFloat) < 0) then
        begin
          TotalPayment := qrySQL.FieldByName('Amount').AsFloat;
        end
        else
        begin
          if (JumlahBayar >= qrySQL.FieldByName('Amount').AsFloat) then
          begin
            TotalPayment := qrySQL.FieldByName('Amount').AsFloat
          end
          else
          begin
            TotalPayment := JumlahBayar;
          end;
          JumlahBayar := JumlahBayar - TotalPayment;
        end;
        UpdateJumlahBayar;
        InsertHistoryBayar;
        qrySQL.Next;
       end;
       InsertMasterPayment;
       UpdateBalance;
       frmComponent.adoWingga.CommitTrans;
       MessageDlg('Save data success !'+#13#10+'Payment number : ' +
edtPayment.Text, mtConfirmation, [mbOK], 0);
       PrintPayment;
```

```
      UpdateNoPayment;
      CekMetodeNoPayment;
      qryInvoice.Close;
      Bersihkan;
      BersihkanTemp;
      IsiCombo;
      cbCustomer.SetFocus;
    Except ON E: Exception do begin
      frmComponent.adoWingga.RollbackTrans;
      MessageDlg(E.Message, mtError, [mbOK], 0);
    end;
    end; // end try
  end;
end;

procedure TfrmArPayment.PrintPayment;
var
  Memo: TfrxMemoView;
  Component: TfrxComponent;
begin
  RefreshQryPayment;
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
            '[ArCustomer] WHERE CustomerCode = ' +
            QuotedStr(cbCustomer.Text);
  OpenQrySQL;
  Component := frxReportPayment.FindObject('Payment');
  Memo := Component as TfrxMemoView;
  Memo.Text := edtPayment.Text;
  Component := frxReportPayment.FindObject('PaymentDate');
  Memo := Component as TfrxMemoView;
  Memo.Text := FormatDateTime('dd MMMM yyy',frmComponent.GetDate());
  Component := frxReportPayment.FindObject('CustomerName');
  Memo := Component as TfrxMemoView;
  Memo.Text := UpperCase(qrySQL.FieldByName('CustomerName').AsString);
  Component := frxReportPayment.FindObject('Alamat');
  Memo := Component as TfrxMemoView;
  Memo.Text :=  qrySQL.FieldByName('Address').AsString;
  Component := frxReportPayment.FindObject('Alamat2');
  Memo := Component as TfrxMemoView;
  Memo.Text :=  qrySQL.FieldByName('City').AsString;
  qrySQL.Close;
  Component := frxReportPayment.FindObject('Total');
  Memo := Component as TfrxMemoView;
```

```
  Memo.Text := edtInvoiceTotal.Text;
  Component := frxReportPayment.FindObject('TotalPayment');
  Memo := Component as TfrxMemoView;
  Memo.Text :=
FloatToStrF(StrToFloat(edtPaymentAmount.Text),ffNumber,10,2);
  temp := 'SELECT Receivable , Description FROM ' +
             frmComponent.lblDB.Caption + '[TblBank] ';
  OpenQrySQL;
  Component := frxReportPayment.FindObject('Account');
  Memo := Component as TfrxMemoView;
  Memo.Text :=  qrySQL.FieldByName('Description').AsString;
  frxReportPayment.ShowReport;
end;

procedure TfrmArPayment.UpdateJumlahBayar;
var SelisihDeposit : Double;
begin
  if ((qrySQL.FieldByName('Amount').AsFloat > 0) AND (TotalPayment > 0))
then
  begin
    temp := 'UPDATE ' + frmComponent.lblDB.Caption +
               '[ArInvoice] SET Payment = Payment + ' +
              QuotedStr(FloatToStr(TotalPayment)) +
               ' WHERE ArInvoice = ' +
               QuotedStr(qrySQL.FieldByName('Invoice').AsString);
  end
  else if ((qrySQL.FieldByName('Amount').AsFloat < 0) AND (TotalPayment <
0)) then
  begin
    SelisihDeposit := TotalDeposit - TotalPayment;
    if SelisihDeposit = 0 then
    begin
      temp := 'UPDATE ' + frmComponent.lblDB.Caption +
                '[ArInvoice] SET Payment = Payment + ' +
                QuotedStr(FloatToStr(TotalPayment)) +
                ' WHERE ArInvoice = ' +
                QuotedStr(qrySQL.FieldByName('Invoice').AsString);
    end
    else if SelisihDeposit > TotalPayment then
    begin
      temp := 'UPDATE ' + frmComponent.lblDB.Caption +
                '[ArInvoice] SET Payment = Payment + ' +
                QuotedStr(FloatToStr(SelisihDeposit-TotalPayment)) +
```

```
                ' WHERE ArInvoice = ' +
               QuotedStr(qrySQL.FieldByName('Invoice').AsString);
    end;
  end;
  with qryMs do
  begin
    Close;
    SQL.Clear;
    SQL.Add(temp);
    ExecSQL;
  end;
end;

procedure TfrmArPayment.InsertHistoryBayar;
begin
  with qryMs do
   begin
      temp :=  'SELECT * FROM '  + frmComponent.lblDB.Caption +
                '[ArPaymentDetail] WHERE Invoice = ' +
                QuotedStr(qrySQL.FieldByName('Invoice').AsString);
      Close;
      SQL.Clear;
      SQL.Add(temp);
      Open;
      Insert;
      FieldByName('Invoice').AsString :=
qrySQL.FieldByName('Invoice').AsString;
      FieldByName('ArPayment').AsString := NoPayment;
      FieldByName('Amount').AsCurrency := TotalPayment;
      Post;
   end;
end;

procedure TfrmArPayment.InsertMasterPayment;
begin
  with qryMs do
   begin
      temp :=  'SELECT * FROM '  + frmComponent.lblDB.Caption +
                '[ArPayment] WHERE ArPayment = ' +
                QuotedStr(edtPayment.Text);
      Close;
      SQL.Clear;
      SQL.Add(temp);
```

```
      Open;
      Insert;
      FieldByName('ArPayment').AsString := NoPayment;
      FieldByName('PaymentDate').AsDateTime := dtpTgl.DateTime;
      FieldByName('Amount').AsFloat := (StrToFloat(edtPaymentAmount.Text));
      FieldByName('Customer').AsString := cbCustomer.Text;
      FieldByName('Bank').AsString := cbAccount.Text;
      FieldByName('Notes').AsString := 'Payment of : ' + NoPayment;
      Post;
   end;
end;

procedure TfrmArPayment.UpdateBalance;
begin
  temp := 'SELECT * FROM ' + frmComponent.lblDB.Caption +
             '[TblBank] WHERE Receivable = ' + QuotedStr(cbAccount.Text);
  OpenQrySQL;
  with qrySQL do
  begin
    Edit;
    FieldByName('Balance').AsFloat := FieldByName('Balance').AsFloat +
(StrToFloat(edtPaymentAmount.Text));
    Post;
  end;
end;

procedure TfrmArPayment.FormShow(Sender: TObject);
begin
  qryInvoice.Close;
  Bersihkan;
  BersihkanTemp;
  IsiCombo;
  CekMetodeNoPayment;
  TotalPayment := 0;
  TotalDeposit := 0;
  TotalTagihan := 0;
  SisaDeposito := 0;
end;

procedure TfrmArPayment.grdInvoiceCheckBoxClick(Sender: TObject; ACol,
  ARow: Integer; State: Boolean);
var
  Status : String;
```

```
begin
  if State then
  begin
    if qryInvoice.FieldByName('Balance').AsFloat < 0 then
    begin
      if (TotalDeposit = 0) then
      begin
        TotalDeposit := TotalDeposit +
qryInvoice.FieldByName('Balance').AsFloat;
        Status := 'Y';
      end
      else if (abs((qryInvoice.FieldByName('Balance').AsFloat) + TotalDeposit)
<= TotalTagihan) then
      begin
        TotalDeposit := TotalDeposit +
qryInvoice.FieldByName('Balance').AsFloat;
        Status := 'Y';
      end
      else
      begin
        MessageDlg('Total deposito masih bisa digunakan untuk pembayaran!',
mtError, [mbOK],0);
        if grdInvoice.Columns[1].CheckFalse = 'N' then
        begin
          grdInvoice.Columns[1].CheckFalse := 'Y';
          grdInvoice.Columns[1].CheckTrue := 'N';
        end
        else if grdInvoice.Columns[1].CheckFalse = 'Y' then
        begin
          grdInvoice.Columns[1].CheckFalse := 'N';
          grdInvoice.Columns[1].CheckTrue := 'Y';
        end;
        Status := 'N';
      end;
    end
    else
    begin
      TotalTagihan := TotalTagihan +
qryInvoice.FieldByName('Balance').AsFloat;
      Status := 'Y';
    end;
    if Status = 'Y' then
    begin
```

```
      TambahTemp;
      TotalPayment := TotalPayment +
qryInvoice.FieldByName('Balance').AsFloat;
    end;
  end
  else
  begin
    if qryInvoice.FieldByName('Balance').AsFloat < 0 then
    begin
      TotalDeposit := TotalDeposit - qryInvoice.FieldByName('Balance').AsFloat;
    end
    else
    begin
      TotalTagihan := TotalTagihan -
qryInvoice.FieldByName('Balance').AsFloat;
    end;
    KurangiTemp;
    TotalPayment := TotalPayment -
qryInvoice.FieldByName('Balance').AsFloat;
  end;
  edtTotal.Text := FloatToStrF(TotalTagihan, ffNumber, 10, 2);
  edtDeposito.Text := FloatToStrF(abs(TotalDeposit), ffNumber, 10, 2);
  edtInvoiceTotal.Text := FloatToStrF(abs(TotalPayment), ffNumber, 10, 2);
  TotalSeluruh := TotalPayment;
end;

procedure TfrmArPayment.btnPostClick(Sender: TObject);
begin
  BuatNoPayment;
  if CekValid then
  begin
    UpdateTransaksi;
  end;
end;

procedure TfrmArPayment.edtPaymentAmountKeyPress(Sender: TObject;
  var Key: Char);
begin
  if not (Key in [#8, '0'..'9', '.']) then
  begin
    Key := #0;
  end
  else if (Key = '.') and (Pos(Key, edtPaymentAmount.Text) > 0) then
```

```
  begin
    Key := #0;
  end;
end;
end.
```

Form InvMov.pas

```
unit FormInvMov;

interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
  Vcl.ComCtrls, Vcl.Buttons, Vcl.StdCtrls, Data.DB, Data.Win.ADODB,
  Vcl.Grids, AdvObj, BaseGrid, AdvGrid, DBAdvGrid, frxClass, frxDBSet,
  Vcl.Mask, AdvDropDown, AdvCustomGridDropDown,
  AdvGridDropDown;
type
  TfrmInvMov = class(TForm)
    GroupBox6: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    btnStart: TSpeedButton;
    dsInv: TDataSource;
    qryInv: TADOQuery;
    Label3: TLabel;
    InvMovrep: TfrxReport;
    InvMovDataset: TfrxDBDataset;
    qryInvStockCode: TStringField;
    qryInvEntryDate: TDateTimeField;
    qryInvMovementType: TStringField;
    qryInvTrnQty: TFloatField;
    qryInvTrnValue: TFloatField;
    qryInvEnteredCost: TFloatField;
    qryInvUnitCost: TFloatField;
    qryInvReference: TStringField;
    qryInvSalesOrder: TStringField;
    qryInvSoCreditNote: TStringField;
    CalcInvMov: TfrxDBDataset;
    btnPrintInv: TSpeedButton;
    qrySal2: TADOQuery;
```

```
StringField1: TStringField;
StringField2: TStringField;
DateTimeField1: TDateTimeField;
StringField3: TStringField;
FloatField1: TFloatField;
FloatField2: TFloatField;
FloatField3: TFloatField;
StringField4: TStringField;
FloatField4: TFloatField;
StringField5: TStringField;
StringField6: TStringField;
StringField7: TStringField;
StringField8: TStringField;
StringField9: TStringField;
StringField10: TStringField;
StringField11: TStringField;
StringField12: TStringField;
StringField13: TStringField;
qrySal2CustomerCode: TStringField;
qrySal2OrderTotal: TFloatField;
qrySal2CostValue: TFloatField;
qryInvBefore: TFloatField;
qryInvAfter: TFloatField;
qryInvDescription: TStringField;
qrySal: TADOQuery;
StringField15: TStringField;
StringField16: TStringField;
DateTimeField2: TDateTimeField;
StringField17: TStringField;
FloatField7: TFloatField;
FloatField8: TFloatField;
FloatField9: TFloatField;
StringField18: TStringField;
FloatField10: TFloatField;
StringField19: TStringField;
StringField20: TStringField;
StringField21: TStringField;
StringField22: TStringField;
StringField23: TStringField;
StringField24: TStringField;
StringField25: TStringField;
StringField26: TStringField;
StringField27: TStringField;
```

```
    FloatField11: TFloatField;
    FloatField12: TFloatField;
    StringField28: TStringField;
    GroupBox2: TGroupBox;
    Label5: TLabel;
    Label4: TLabel;
    Label6: TLabel;
    dtpAwal: TDateTimePicker;
    dtpAkhir: TDateTimePicker;
    edtFind: TEdit;
    GroupBox8: TGroupBox;
    DBAdvGrid1: TDBAdvGrid;
    procedure Startup;
    procedure OpenQryInv;
    procedure FormShow(Sender: TObject);
    procedure btnStartClick(Sender: TObject);
    procedure btnPrintInvClick(Sender: TObject);
    procedure dtpAwalChange(Sender: TObject);
    procedure dtpAkhirChange(Sender: TObject);
    procedure RefreshMaster;
    procedure edtFindChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmInvMov: TfrmInvMov;

implementation
uses FormComponent, FormInventory, FormTemp, FormWhControl;
var qry, temp_find1, temp_find2, temp_find3, temp_find4 :string;
{$R *.dfm}
procedure TfrmInvMov.OpenQryInv;
begin
  with qryInv do
  begin
    Close;
    SQL.Clear;
    SQL.Add(qry);
    Open;
  end;
```

```
end;

procedure TfrmInvMov.btnPrintInvClick(Sender: TObject);
begin
  if not (qryInv.IsEmpty) then
  begin
    InvMovRep.ShowReport();
  end
    else
    begin
      ShowMessage('Empty result!');
    end;
end;

procedure TfrmInvMov.btnStartClick(Sender: TObject);
begin
  RefreshMaster;
end;

procedure TfrmInvMov.RefreshMaster;
begin
  if (StringReplace(edtFind.Text,' ','',[rfReplaceAll,rfIgnoreCase]) = '') then
  begin
    edtFind.Text := '';
  end;
  temp_find1 := UpperCase('%'+edtFind.Text+'%');
  temp_find2 := UpperCase(edtFind.Text+'%');
  temp_find3 := UpperCase('%'+edtFind.Text);
  temp_find4 := UpperCase(edtFind.Text);
  qry:='Select a.*, c.Description, b.QtyOnHand-(Select SUM(TrnQty) '+
          'from'+ frmComponent.lblDB.Caption+ '[InvMovements] where'+
          'StockCode=a.StockCode and ((TimeStamp>a.TimeStamp OR '+
          'TimeStamp=a.TimeStamp))) as [Before], b.QtyOnHand-(Select '+
          'SUM(TrnQty)  from'+frmComponent.lblDB.Caption+
          '[InvMovements] where StockCode=a.StockCode and '+
          '((TimeStamp>a.TimeStamp OR TimeStamp=a.TimeStamp))) +
          'a.TrnQty as [After] from '+
          frmComponent.lblDB.Caption+'[InvMovements] a join '+
          frmComponent.lblDB.Caption+'[InvWarehouse] b on '+
          'a.StockCode=b.StockCode join '+frmComponent.lblDB.Caption+
          '[InvMaster] c on b.StockCode=c.StockCode ' +
          ' AND (a.EntryDate >= CONVERT (DATETIME, ' +
          QuotedStr(FormatDateTime('yyyy-mm-dd',dtpAwal.Date) +
```

```
        ' 00:00:00') + ', 102)) AND (a.EntryDate <= CONVERT' +
        '(DATETIME, ' +
        QuotedStr(FormatDateTime('yyyy-mm- dd',dtpAkhir.Date) +
        ' 23:59:59') + ', 102))' +
        'AND ((a.StockCode LIKE '+ QuotedStr(temp_find1) + ')' +
        'OR (a.StockCode LIKE '+ QuotedStr(temp_find2) + ')' +
        'OR (a.StockCode LIKE '+ QuotedStr(temp_find3) + ')' +
        'OR (a.StockCode LIKE '+ QuotedStr(temp_find4) + ')' +
        'OR (a.Reference LIKE '+ QuotedStr(temp_find1) + ')' +
        'OR (a.Reference LIKE '+ QuotedStr(temp_find2) + ')' +
        'OR (a.Reference LIKE '+ QuotedStr(temp_find3) + ')' +
        'OR (a.Reference LIKE '+ QuotedStr(temp_find4) + ')'+
        'OR (a.SalesOrder LIKE '+ QuotedStr(temp_find1) + ')' +
        'OR (a.SalesOrder LIKE '+ QuotedStr(temp_find2) + ')' +
        'OR (a.SalesOrder LIKE '+ QuotedStr(temp_find3) + ')' +
        'OR (a.SalesOrder LIKE '+ QuotedStr(temp_find4) + ')' +
        'OR (a.SoReturn LIKE '+ QuotedStr(temp_find1) + ')' +
        'OR (a.SoReturn LIKE '+ QuotedStr(temp_find2) + ')' +
        'OR (a.SoReturn LIKE '+ QuotedStr(temp_find3) + ')' +
        'OR (a.SoReturn LIKE '+ QuotedStr(temp_find4) + '))'+
        'order by a.StockCode asc';
  OpenQryInv;
end;

procedure TfrmInvMov.dtpAkhirChange(Sender: TObject);
begin
  if dtpAwal.DateTime > dtpAkhir.DateTime then
    dtpAwal.DateTime := dtpAkhir.DateTime;
  RefreshMaster;
end;

procedure TfrmInvMov.dtpAwalChange(Sender: TObject);
begin
  if dtpAwal.DateTime > dtpAkhir.DateTime then
    dtpAwal.DateTime := dtpAkhir.DateTime;
  RefreshMaster;
end;

procedure TfrmInvMov.edtFindChange(Sender: TObject);
begin
  RefreshMaster;
end;
```

```
procedure TfrmInvMov.FormShow(Sender: TObject);
begin
  Startup;
end;

procedure TfrmInvMov.Startup;
begin
  qryInv.Close;
  dtpAwal.DateTime     := Date - StrToInt(FormatDateTime('dd',Date)) + 1;
  dtpAkhir.DateTime    := Now;
  edtFind.Text         := '';
end;
end.
```

Form SalesMov.pas

```
unit FormSalesMov;

interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
  Vcl.ComCtrls, Vcl.Buttons, Vcl.StdCtrls, Data.DB, Data.Win.ADODB,
  Vcl.Grids, AdvObj, BaseGrid, AdvGrid, DBAdvGrid, frxClass, frxDBSet,
  Vcl.Mask, AdvDropDown, AdvCustomGridDropDown,
  AdvGridDropDown;
type
  TfrmSalesMov = class(TForm)
    GroupBox6: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    btnStart: TSpeedButton;
    dsInv: TDataSource;
    qryInv: TADOQuery;
    Label3: TLabel;
    qryInvStockCode: TStringField;
    qryInvWarehouse: TStringField;
    qryInvEntryDate: TDateTimeField;
    qryInvMovementType: TStringField;
    qryInvTrnQty: TFloatField;
    qryInvTrnValue: TFloatField;
    qryInvEnteredCost: TFloatField;
```

```
qryInvTrnType: TStringField;
qryInvUnitCost: TFloatField;
qryInvReference: TStringField;
qryInvReference1: TStringField;
qryInvReference2: TStringField;
qryInvSalesOrder: TStringField;
qryInvDeliveryOrder: TStringField;
qryInvPurchaseOrder: TStringField;
qryInvReceipt: TStringField;
qryInvPoCreditNote: TStringField;
qryInvSoCreditNote: TStringField;
btnPrintSalesMov: TSpeedButton;
dsInvSales: TDataSource;
qrySal2: TADOQuery;
StringField1: TStringField;
StringField2: TStringField;
DateTimeField1: TDateTimeField;
StringField3: TStringField;
FloatField1: TFloatField;
FloatField2: TFloatField;
FloatField3: TFloatField;
StringField4: TStringField;
FloatField4: TFloatField;
StringField5: TStringField;
StringField6: TStringField;
StringField7: TStringField;
StringField8: TStringField;
StringField9: TStringField;
StringField10: TStringField;
StringField11: TStringField;
StringField12: TStringField;
StringField13: TStringField;
InvMovSalesrep: TfrxReport;
InvMovSalesDataset: TfrxDBDataset;
CalcInvMovSales: TfrxDBDataset;
qrySal2CustomerCode: TStringField;
qrySal2OrderTotal: TFloatField;
CalcInvMovSales2: TfrxDBDataset;
qrySal2CostValue: TFloatField;
qryInvBefore: TFloatField;
qryInvAfter: TFloatField;
qryInvDescription: TStringField;
qrySal: TADOQuery;
```

```
    GroupBox12: TGroupBox;
    DBAdvGrid2: TDBAdvGrid;
    GroupBox2: TGroupBox;
    Label5: TLabel;
    Label4: TLabel;
    Label6: TLabel;
    dtpAwal: TDateTimePicker;
    dtpAkhir: TDateTimePicker;
    edtFind: TEdit;
    qrySalOrderDate: TDateTimeField;
    qrySalCustomerCode: TWideStringField;
    qrySalSalesOrder: TWideStringField;
    qrySalRequisition: TWideStringField;
    qrySalStockCode: TWideStringField;
    qrySalRoll: TIntegerField;
    qrySalQty: TBCDField;
    qrySalRoll_1: TIntegerField;
    qrySalDeliveryQty: TBCDField;
    qrySalLine: TSmallintField;
    qrySalLine_1: TSmallintField;
    qrySalStockCode_1: TWideStringField;
    procedure Startup;
    procedure OpenQrySal;
    procedure FormShow(Sender: TObject);
    procedure btnStartClick(Sender: TObject);
    procedure btnPrintSalesMovClick(Sender: TObject);
    procedure dtpAwalChange(Sender: TObject);
    procedure dtpAkhirChange(Sender: TObject);
    procedure RefreshMaster;
    procedure edtFindChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmSalesMov: TfrmSalesMov;

implementation
uses FormComponent, FormInventory, FormTemp, FormWhControl;
var qry, temp_find1, temp_find2, temp_find3, temp_find4 :string;
{$R *.dfm}
```

```
procedure TfrmSalesMov.OpenQrySal;
begin
  with qrySal do
  begin
    Close;
    SQL.Clear;
    SQL.Add(qry);
    Open;
  end;
end;

procedure TfrmSalesMov.btnPrintSalesMovClick(Sender: TObject);
begin
  if not (qrySal.IsEmpty) then
  begin
    InvMovSalesrep.ShowReport();
  end
    else
    begin
      ShowMessage('Empty result!');
    end;
end;

procedure TfrmSalesMov.btnStartClick(Sender: TObject);
begin
  RefreshMaster;
end;

procedure TfrmSalesMov.RefreshMaster;
begin
  if (StringReplace(edtFind.Text,' ','',[rfReplaceAll,rfIgnoreCase]) = '') then
  begin
    edtFind.Text := '';
  end;
  temp_find1 := UpperCase('%'+edtFind.Text+'%');
  temp_find2 := UpperCase(edtFind.Text+'%');
  temp_find3 := UpperCase('%'+edtFind.Text);
  temp_find4 := UpperCase(edtFind.Text);
  qry := 'SELECT a.OrderDate, a.CustomerCode, a.SalesOrder, '+
           'a.Requisition, b.StockCode, b.Roll, b.Qty, c.Roll, c.DeliveryQty, '+
           'b.Line, c.Line, c.StockCode FROM SoMaster a, '+
           'SoRequisitionDetail b, SalDetail c where a.Requisition = '+
           'b.Requisition and a.SalesOrder = c.SalesOrder AND '+
```

```
        'b.StockCode = c.StockCode AND (a.OrderDate >= CONVERT' +
        '(DATETIME, ' +
        QuotedStr(FormatDateTime('yyyy-mm-dd',dtpAwal.Date) +
        ' 00:00:00') + ', 102)) AND (a.OrderDate <= CONVERT' +
        '(DATETIME, ' +
        QuotedStr(FormatDateTime('yyyy-mm-dd',dtpAkhir.Date) +
        ' 23:59:59') + ', 102))' +
        'AND ((b.StockCode LIKE '+ QuotedStr(temp_find1) + ')' +
        'OR (b.StockCode LIKE '+ QuotedStr(temp_find2) + ')' +
        'OR (b.StockCode LIKE '+ QuotedStr(temp_find3) + ')' +
        'OR (b.StockCode LIKE '+ QuotedStr(temp_find4) + ')' +
        'OR (a.Requisition LIKE '+ QuotedStr(temp_find1) + ')' +
        'OR (a.Requisition LIKE '+ QuotedStr(temp_find2) + ')' +
        'OR (a.Requisition LIKE '+ QuotedStr(temp_find3) + ')' +
        'OR (a.Requisition LIKE '+ QuotedStr(temp_find4) + ')'+
        'OR (a.CustomerCode LIKE '+ QuotedStr(temp_find1) + ')' +
        'OR (a.CustomerCode LIKE '+ QuotedStr(temp_find2) + ')' +
        'OR (a.CustomerCode LIKE '+ QuotedStr(temp_find3) + ')' +
        'OR (a.CustomerCode LIKE '+ QuotedStr(temp_find4) + ')' +
        'OR (a.SalesOrder LIKE '+ QuotedStr(temp_find1) + ')' +
        'OR (a.SalesOrder LIKE '+ QuotedStr(temp_find2) + ')' +
        'OR (a.SalesOrder LIKE '+ QuotedStr(temp_find3) + ')' +
        'OR (a.SalesOrder LIKE '+ QuotedStr(temp_find4) + '))';
  OpenQrySal;
end;

procedure TfrmSalesMov.dtpAkhirChange(Sender: TObject);
begin
  if dtpAwal.DateTime > dtpAkhir.DateTime then
    dtpAwal.DateTime := dtpAkhir.DateTime;
  RefreshMaster;
end;

procedure TfrmSalesMov.dtpAwalChange(Sender: TObject);
begin
  if dtpAwal.DateTime > dtpAkhir.DateTime then
    dtpAwal.DateTime := dtpAkhir.DateTime;
  RefreshMaster;
end;

procedure TfrmSalesMov.edtFindChange(Sender: TObject);
begin
  RefreshMaster;
```

```
end;

procedure TfrmSalesMov.FormShow(Sender: TObject);
begin
  Startup;
end;

procedure TfrmSalesMov.Startup;
begin
  qrySal.Close;
  dtpAwal.DateTime     := Date - StrToInt(FormatDateTime('dd',Date)) + 1;
  dtpAkhir.DateTime    := Now;
  edtFind.Text         := '';
end;
end.
```

FormTransMov.pas

```
unit FormTransMov;

interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
  System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
  Data.DB, Data.Win.ADODB, Vcl.Grids, AdvObj, BaseGrid, AdvGrid,
  DBAdvGrid, Vcl.Mask, AdvDropDown, AdvCustomGridDropDown,
  AdvGridDropDown, Vcl.ComCtrls, Vcl.StdCtrls,Vcl.Buttons, frxClass,
  frxDBSet, RTTI;
type
  TfrmTransMov = class(TForm)
    GroupBox6: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    btnStart: TSpeedButton;
    Label3: TLabel;
    btnPrint: TSpeedButton;
    GroupBox7: TGroupBox;
    DBAdvGrid1: TDBAdvGrid;
    dsTransMov: TDataSource;
    qryTransMov: TADOQuery;
    Edit14: TEdit;
    qryTransMovPaymentDate: TDateField;
```

```
    qryTransMovBank: TStringField;
    qryTransMovAmount: TFloatField;
    qryTransMovNotes: TStringField;
    qryTransMovCustomer: TStringField;
    qryTransMovArPayment: TStringField;
    TransMovRep: TfrxReport;
    TransMovRepDa: TfrxDBDataset;
    GroupBox1: TGroupBox;
    Label5: TLabel;
    dtpAwal: TDateTimePicker;
    Label4: TLabel;
    dtpAkhir: TDateTimePicker;
    Label6: TLabel;
    edtFind: TEdit;
    procedure Startup;
    procedure OpenQryTransMov;
    procedure FormShow(Sender: TObject);
    procedure btnStartClick(Sender: TObject);
    procedure btnPrintClick(Sender: TObject);
    procedure dtpAwalChange(Sender: TObject);
    procedure dtpAkhirChange(Sender: TObject);
    procedure RefreshMaster;
    procedure edtFindChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmTransMov: TfrmTransMov;

implementation
uses FormComponent;
var qry, temp_find1, temp_find2, temp_find3, temp_find4 :string;
{$R *.dfm}

procedure TfrmTransMov.btnPrintClick(Sender: TObject);
begin
  TransMovRep.ShowReport();
end;
```

```
procedure TfrmTransMov.btnStartClick(Sender: TObject);
begin
  RefreshMaster;
end;

procedure TfrmTransMov.dtpAwalChange(Sender: TObject);
begin
  if dtpAwal.DateTime > dtpAkhir.DateTime then
    dtpAwal.DateTime := dtpAkhir.DateTime;
  RefreshMaster;
end;

procedure TfrmTransMov.dtpAkhirChange(Sender: TObject);
begin
  if dtpAwal.DateTime > dtpAkhir.DateTime then
    dtpAwal.DateTime := dtpAkhir.DateTime;
  RefreshMaster;
end;

procedure TfrmTransMov.RefreshMaster;
begin
  if (StringReplace(edtFind.Text,' ',[rfReplaceAll,rfIgnoreCase]) = '') then
  begin
    edtFind.Text := '';
  end;
  temp_find1 := UpperCase('%'+edtFind.Text+'%');
  temp_find2 := UpperCase(edtFind.Text+'%');
  temp_find3 := UpperCase('%'+edtFind.Text);
  temp_find4 := UpperCase(edtFind.Text);
  qry:= 'SELECT * FROM ArPayment WHERE (PaymentDate >=' +
        'CONVERT (DATETIME, ' +
        QuotedStr(FormatDateTime('yyyy-mm-dd',dtpAwal.Date) +
        ' 00:00:00') + ', 102)) AND (PaymentDate <= CONVERT' +
        '(DATETIME, ' +
        QuotedStr(FormatDateTime('yyyy-mm-dd',dtpAkhir.Date) +
        ' 23:59:59') + ', 102))' +
        'AND ((ArPayment LIKE '+ QuotedStr(temp_find1) + ')' +
        'OR (ArPayment LIKE '+ QuotedStr(temp_find2) + ')' +
        'OR (ArPayment LIKE '+ QuotedStr(temp_find3) + ')' +
        'OR (ArPayment LIKE '+ QuotedStr(temp_find4) + ')' +
        'OR (Customer LIKE '+ QuotedStr(temp_find1) + ')' +
        'OR (Customer LIKE '+ QuotedStr(temp_find2) + ')' +
        'OR (Customer LIKE '+ QuotedStr(temp_find3) + ')' +
```

```
        'OR (Customer LIKE '+ QuotedStr(temp_find4) + ')' +
        'OR (Bank LIKE '+ QuotedStr(temp_find1) + ')' +
        'OR (Bank LIKE '+ QuotedStr(temp_find2) + ')' +
        'OR (Bank LIKE '+ QuotedStr(temp_find3) + ')' +
        'OR (Bank LIKE '+ QuotedStr(temp_find4) + '))';
  Edit14.Text:=qry;
  OpenQryTransMov;
end;

procedure TfrmTransMov.edtFindChange(Sender: TObject);
begin
  RefreshMaster;
end;
procedure TfrmTransMov.FormShow(Sender: TObject);
begin
  Startup;
end;

procedure TfrmtransMov.Startup;
begin
  dtpAwal.DateTime     := Date - StrToInt(FormatDateTime('dd',Date)) + 1;
  dtpAkhir.DateTime    := Now;
  edtFind.Text         := '';
end;

procedure TfrmtransMov.OpenQryTransMov;
begin
  with qryTransMov do
  begin
    Close;
    SQL.Clear;
    SQL.Add(qry);
    Open;
  end;
end;
end.
```