ANEXO No 1

DESCRIPCION DE LOS MODULOS PRINCIPALES DEL SISTEMA

Código del Sistema Control de Pérdidas

Módulos Principales

a) Módulo Cálculo de Energía

Contiene declaraciones globales del Sistema, funciones para calcular valores en sucres, energía de abonados, luminarias y servicios ocasionales, se describen algunas de ellas.

- **Función calculo_demanda**

Retorna el valor de demanda (Watios) de acuerdo al consumo (kWh) del abonado, ésta función es utilizada para valorar el consumo de la demanda en sucres, el código de ésta función es el siguiente.

```
Function calculo_demanda(kw, deman As Double)
Dim dem As Double
dem = 0
'kw kilowatios de demanda
'deman valor por demanda del pliego tarifario
        If kw > 0 Then
          If kw < 10 Then
             dem = dem + kw * 0.9 * deman
          Else
             dem = dem + 10 * 0.9 * deman
          End If
        End If
        If (kw > 10) Then
           If kw < 30 Then
              dem = dem + ((kw - 10) * 0.8 * deman)
           Else
              dem = dem + 20 * 0.8 * deman
           End If
        End If
        If (kw > 30) Then
           If kw < 80 Then
              dem = dem + (kw * 0.7 * deman)
           Else
           dem = dem + 50 * 0.7 * deman
           End If
        End If
        If (kw > 80) Then
           If kw < 80 Then
              dem = dem + kw * 0.5 * deman
           Else
              dem = dem + ((kw - 80) * 0.5 * deman)
           End If
        End If
```

```
calculo_demanda = dem
End Function
```

- **Función calculo_sucres**

De acuerdo al Pliego Tarifario vigente, tarifa del Abonado, consumo, demanda y procedimientos de facturación utilizados por la empresa, este procedimiento realiza la valoración en sucres del consumo de cada usuario, el código se muestra a continuación.

```
Function calculo_sucres(tarifa As String, kilowatios As Long, Demanda As Long) As Long
Dim subsidio As String
Dim valor As Integer
Dim descuento As Recordset
Dim a(1 To 4, 1 To 15) As Double
Dim aux As Long

aux = 1
valor = 1
subs = 0  'valor subsidio compensatorio
subs_cr = 0 'valor subsidio cruzado
comer = 0 'valor por comercialización
Set db = CurrentDb
subsidio = "SELECT subsidio.tarifa, subsidio.[Valor Subsidio], subsidio.Consumo " _
& "FROM subsidio " _
& "WHERE (((subsidio.tarifa)='R'));"

While Not sucres.EOF
   a(1, aux) = sucres!Rango  'fila 1 valores de rango
   a(2, aux) = sucres![Valor energia]    'fila 2 valores de energia
   a(3, aux) = sucres![Valor Comer]      'fila 3 valores de comercialización o demanda
   aux = aux + 1
   sucres.MoveNext
Wend
Select Case tarifa
Case "CD"
      aux = kilowatios * a(2, 1) + calculo_demanda(Demanda, a(3, 1))
Case "OD"
      aux = kilowatios * a(2, 1) + calculo_demanda(Demanda, a(3, 1))
Case "ED"
      aux = kilowatios * a(2, 1) + calculo_demanda(Demanda, a(3, 1))
Case "DP"
      aux = kilowatios * a(2, 1) + calculo_demanda(Demanda, a(3, 1))
Case "ID"
      aux = kilowatios * a(2, 1) + calculo_demanda(Demanda, a(3, 1))
Case "BA"
      aux = kilowatios * a(2, 1) + calculo_demanda(Demanda, a(3, 1))
Case "AD"
```

```
      aux = kilowatios * a(2, 1) + calculo_demanda(Demanda, a(3, 1))
Case "BD"
      aux = kilowatios * a(2, 1) + calculo_demanda(Demanda, a(3, 1))
Case "RT"
      aux = kilowatios * a(2, 1) + a(3, 1)
Case "AP"
      aux = kilowatios * a(2, 1)
Case Else
aux = 0
 For valor = 1 To sucres.RecordCount
   If ((a(1, valor) <= kilowatios) And (valor = sucres.RecordCount)) Then
         aux = aux + ((kilowatios - a(1, valor - 1)) * a(2, valor)) 'caso en el que el rango es mayor
         que  el consumo se realiza la diferencia con el rango anterior y
         comer = a(3, valor)  'valor de comercialización
      If ((tarifa = "R") And (kilowatios <= 100)) Then
         Set descuento = db.OpenRecordset(subsidio)
         If descuento.RecordCount > 0 Then
            descuento.MoveLast
            descuento.MoveFirst
         End If
         While Not descuento.EOF
            If kilowatios = descuento!consumo Then
               'lum = aux '**energía neta para alumbr. publico
               subs = descuento![Valor Subsidio]
               descuento.MoveLast
            End If
            descuento.MoveNext
         Wend
         descuento.Close
      End If
      If ((tarifa = "R") And (kilowatios > 100)) Then
         subs_cr = kilowatios * 82
      End If
   Else  'termina último rango del pliego tarifario

       If a(1, valor) < kilowatios Then
          If valor = 1 Then
            aux = aux + a(2, 1)
          Else
          aux = aux + a(2, valor) * (a(1, valor) - a(1, valor - 1))

         End If              'a(1, valor) - a(1, valor - 1) da la diferencia entre rango y rango
       Else
         If (valor = 1) Then
            comer = a(3, valor)  'valor de comercialización
             aux = aux + a(2, valor)  '+ a(3, valor) 'kilowatios menor al primer rango del pliego;
          energía y comercialización
          Else
                    aux = aux + ((kilowatios - a(1, valor - 1)) * a(2, valor)) 'caso en el que el rango
```

```
            'el consumo para establecer los kilowatios restantes que serán multiplicados por el
            comer = a(3, valor)  'valor de comercialización
        End If
        valor = sucres.RecordCount

    End If
End If 'controla el último rango del pliego tarifario

Next     'final del bucle de aplicación del pliego tarifario
End Select
calculo_sucres = aux
End Function
```

- **Función energia_cal**

La función calcula la energía consumida por abonados, luminarias y servicios ocasionales, de acuerdo a los procedimientos propuestos por la empresa y la Unidad de Control de Pérdidas, se muestra el código:

```
Function  energia_cal(No_transf, fi, ff As String)
Dim kvar, Demanda, aux, dias, Energia
Dim estad As Recordset
Dim rst, rst1 As Recordset
Dim strSQL, lumin, transf, est_transf As String
Dim dias_serv
Dim esferas As Recordset
Dim esf As String
Dim rebotados
Dim kwh1, ener_lum 'variables locales para calcular energía de luminarias, abonados y servicios
ocasionales
Dim anio, anio1, mes, mes1, dia, dia1 As Integer
Dim Abonado
   kvar = 0
   rebotados = 0
   Demanda = 0
   kwh1 = 0
   ener_lum = 0
   Abonado = 0
   anio = Year(fi)
   anio1 = Year(ff)
   mes = Month(fi)
   mes1 = Month(ff)
   dia = day(fi)
   dia1 = day(ff)
   'fi fecha inicial del estudio
   'ff fecha final del estudio
   dias = DateValue(ff) - DateValue(fi)
```

```
   '*** consulta consumo abonados
strSQL = "SELECT medidor.Transformador, medidor.Tarifa, medidor.factor,[Lectura
tomada].[Nro Medidor], [Lectura tomada].[Fecha inicio], [Lectura tomada].[Fecha final], [Lectura
tomada].[Kwh inicial], [Lectura tomada].[Kwh final], [Lectura tomada].[Kvarh inicial], [Lectura
tomada].[Kvarh final], [Lectura tomada].[Demanda inicial], [Lectura tomada].[Demanda final],
[Kwh final]-[Kwh inicial] AS kwh, [Kvarh final]-[Kvarh inicial] AS kvar, [Demanda final]-
[Demanda inicial] AS demanda " _
& "FROM medidor INNER JOIN [Lectura tomada] ON medidor.num_medidor = [Lectura
tomada].[Nro Medidor] " _
& "WHERE (((medidor.Transformador)='" & No_transf & "') AND (([Lectura tomada].[Fecha
inicio])=#" & mes & "/" & dia & "/" & anio & "#) AND (([Lectura tomada].[Fecha final])=#" &
mes1 & "/" & dia1 & "/" & anio1 & "#));"

Set rst = db.OpenRecordset(strSQL)
aux = rst.RecordCount
If aux >= 1 Then
      rst.MoveLast
      rst.MoveFirst
End If
   Abonado = rst.RecordCount
   nro_abonados = nro_abonados + Abonado
   While Not rst.EOF
   If rst!kwh < 0 Then
   '****consultar nro de esferas de cada medidor rebotado
   esf = "SELECT medidor.num_medidor, tipo_medidor.num_esferas " _
   & "FROM tipo_medidor INNER JOIN medidor ON tipo_medidor.tipo_medidor =
        medidor.tipo_medidor " _
   & "WHERE (((medidor.num_medidor)='" & rst![Nro Medidor] & "'));"
      Set esferas = db.OpenRecordset(esf)
      esferas.MoveLast
      esferas.MoveFirst
   Select Case esferas!num_esferas
   Case 3: rebotados = 999 - rst![Kwh inicial] + rst![Kwh final]
   Case 4: rebotados = 9999 - rst![Kwh inicial] + rst![Kwh final]
   Case 5: rebotados = 99999 - rst![Kwh inicial] + rst![Kwh final]
   Case 6: rebotados = 999999 - rst![Kwh inicial] + rst![Kwh final]
   Case 7: rebotados = 9999999 - rst![Kwh inicial] + rst![Kwh final]
   Case 8: rebotados = 99999999 - rst![Kwh inicial] + rst![Kwh final]
   End Select
   esferas.Close
   Debug.Print rst![Kwh inicial] & " = " & rst![Kwh final]
   MsgBox rst![Nro Medidor] & " = " & rebotados
   kwh1 = kwh1 + rebotados * rst!factor
   Else
      kwh1 = kwh1 + rst!kwh * rst!factor
   End If

   rst.MoveNext
   Wend
```

```
   rst.Close
   '**** CALCULO ENERGIA LUMINARIAS
lumin = "SELECT transf_luminaria.NoTransformador, transf_luminaria.Idluminaria, luminarias.Potencia, luminarias.[Potencia Balasto], transf_luminaria.Apagadas, transf_luminaria.Encendidas, transf_luminaria.Fecha, ([Potencia]+[Potencia Balasto])*[Encendidas] AS watios " _
& "FROM luminarias INNER JOIN transf_luminaria ON luminarias.IdLuminaria = transf_luminaria.Idluminaria " _
& "WHERE (((transf_luminaria.NoTransformador)='" & No_transf & "') AND ((transf_luminaria.Fecha) Between #" & mes & "/" & dia & "/" & anio & "# And #" & mes1 & "/" & dia1 & "/" & anio1 & "#));"

  Set rst1 = db.OpenRecordset(lumin)
  aux = rst1.RecordCount
  If aux >= 1 Then
      rst1.MoveLast
      rst1.MoveFirst
   End If
   dias = DateValue(ff) - DateValue(fi)
While Not rst1.EOF
      If rst1!Idluminaria = "sem" Then
         ener_lum = ener_lum + rst1!Watios * 15 * dias * 0.92  ' horas de encendido
      Else
         ener_lum = ener_lum + rst1!Watios * 12 * dias * 0.92
      End If
   rst1.MoveNext
   Wend

   '*** actualizar consumo de energia por abonados, luminarias, servicios ocasionales
est_transf = "SELECT estad_transf.Transformador, estad_transf.[Fecha inicial], estad_transf.[Fecha final], estad_transf.kwh, estad_transf.watios, estad_transf.ocasional, estad_transf.abonados " _
   & "FROM estad_transf " _
   & "WHERE (((estad_transf.Transformador)='" & No_transf & "') AND ((estad_transf.[Fecha inicial])=#" & mes & "/" & dia & "/" & anio & "#));"

   Set estad = db.OpenRecordset(est_transf)
   If estad.RecordCount >= 1 Then
      estad.MoveLast
      estad.MoveFirst
     ' Debug.Print estad!Transformador & " " & estad![Fecha inicial]
      estad.Edit 'modificando los valores de la tabla estad_transf
   Else
      estad.AddNew    '**** insertando a la tabla estad_transf
   End If

   estad!Transformador = No_transf
   estad![Fecha Inicial] = DateValue(fi)  ' #10/23/96#
   estad![Fecha Final] = DateValue(ff)   '#10/30/96#
```

```
    ener_lum = ener_lum / 1000    'energia consumida por las luminarias
    'lum = calculo_sucres("AP", Watios)
    Energia = ener_lum + kwh1
    estad!kwh = kwh1
    estad!Watios = ener_lum
    estad!ocasional = 0
    estad!abonados = Abonado
    estad.Update
    rst1.Close
    estad.Close
Watios = Watios + ener_lum
kwh = kwh + kwh1
energia_cal = ener_lum + kwh1
End Function
```

- **Función alumb**

Calcula el valor por alumbrado público (sucres) de acuerdo a los procedimientos de facturación establecidos en la empresa y la tarifa de cada abonado, se muestra el código.

```
Function alumb(tarif As String, valor As Long) As Long

Select Case tarif
Case "R"
      alumb = valor * 0.05
Case "S"
      alumb = valor * 0.05
Case "C"
      alumb = valor * 0.16
Case "EO"
      alumb = valor * 0.16
Case "CD"
      alumb = valor * 0.14
Case "OD"
      alumb = valor * 0.14
Case "IA"
      alumb = valor * 0.16
Case "ID"
      alumb = valor * 0.14
Case "BA"
      alumb = valor * 0.14
Case "AS"
      alumb = valor * 0.16
Case "BP"
      alumb = valor * 0.16
Case "AD"
      alumb = valor * 0.14
Case "BD"
```

```
      alumb = valor * 0.14
Case "ED"
      alumb = valor * 0.14
Case "DP"
      alumb = valor * 0.14
Case Else
   alumb = 0
End Select
End Function
```

b) Módulo calculo impuestos

Constan las funciones que calculan los rubros por seguro contra incendios, bomberos y electrificación rural, se muestra el código de cada una de ellas.

- **Función seg.**

En ésta función se realiza la valoración del impuesto por seguro contra incendios, el código se muestra a continuación.

```
Function seg(tarifa As String, kw As Long) As Long
   Select Case tarifa
   Case "R"
      If ((kw >= 0) And (kw <= 100)) Then
         seg = 118
      End If
      If ((kw >= 101) And (kw <= 500)) Then
         seg = 235
      End If
      If ((kw >= 501) And (kw <= 9999)) Then
         seg = 353
      End If
   Case "S"
      If ((kw >= 0) And (kw <= 100)) Then
         seg = 118
      End If
      If ((kw >= 101) And (kw <= 500)) Then
         seg = 235
      End If
      If ((kw >= 501) And (kw <= 9999)) Then
         seg = 353
      End If
   Case "C"
      If ((kw >= 0) And (kw <= 100)) Then
         seg = 294
      End If
```

```
      If ((kw >= 101) And (kw <= 9999)) Then
         seg = 412
      End If
   Case Else
      seg = 0
End Select
End Function
```

- **Función bombe.**

El valor por el servicio de bomberos es valorado en ésta función. Se muestra su código.

```
Function bombe(tarifa As String, kw As Long) As Long
   Select Case tarifa
   Case "R"
         bombe = 500
   Case "S"
         bombe = 500
   Case "C"
         bombe = 1500
   Case "IA"
         If ((kw >= 0) And (kw <= 9999)) Then
            bombe = 3000
         End If
   Case "ID"
         bombe = 6000
   Case Else
         bombe = 0
   End Select
End Function
```

- **Función elect_rural.**

Los valores por electrificación rural son calculados en ésta función, el código es el siguiente.

```
Function elect_rural(tarif As String, valor As Long) As Long
   Select Case tarif
   Case "IA"
         elect_rural = valor * 0.1
   Case "ID"
         elect_rural = valor * 0.1
   Case "C"
         elect_rural = valor * 0.1
   Case Else
         elect_rural = 0
```

```
   End Select
End Function
```

c) Módulo Refacturar

Contiene la función que valora los fraudes, se aplica el pliego tarifario para el mes respectivo de acuerdo al consumo del abonado y se calculan los valores para los impuestos vigentes (basura, bomberos, seguro contra incendios, electrificación rural, subsidios, etc.), la función se muestra.

```
Function refactura(prom1 As Long, Tiempo As Integer, mes_pliego As Integer, anio_pliego As Integer, tf As String) As Long
Dim dbs As Database
Dim criterio As String
Dim valor As Long
Dim total As Long

valor = 1
total = 0

alumb_aux = 0
bomb_aux = 0
seg_aux = 0
bas_aux = 0
electrural_aux = 0

criterio = "SELECT tarifa, Rango, [Valor energia], [Valor Comer] " _
& "FROM [valor tarifa] " _
& "WHERE ((tarifa='" & tf & "') AND (Month([Fecha Inicial])=" & mes_pliego & ") AND (Year([Fecha Inicial])= " & anio_pliego & "));"

Set dbs = CurrentDb
Set sucres = dbs.OpenRecordset(criterio)
If sucres.RecordCount > 0 Then
      sucres.MoveLast
      sucres.MoveFirst
      valor = calculo_sucres(tf, prom1, 0) 'consumo neto
      total = valor * Tiempo
      '***** ALUMBRADO
      alumb_aux = alumb(tf, valor)
      alumb_aux = alumb_aux * Tiempo
      'BOMBEROS
       bomb_aux = bombe(tf, prom1) * Tiempo '* contador
      'SEGURO
       seg_aux = seg(tf, prom1) * Tiempo '* contador
```

```
      'BASURA 10%
       bas_aux = valor * 0.1
       bas_aux = CLng(bas_aux)
       bas_aux = bas_aux * Tiempo
       'ELECTRIFICACION RURAL
       electrural_aux = elect_rural(tf, valor)
       electrural_aux = electrural_aux * Tiempo

      'total = total + comer * Tiempo + alumb_aux + bomb_aux + electrural_aux + seg_aux +
bas_aux + subs_cr * Tiempo
            'REDONDEO
              ' valor = total / 1000
              ' valor = valor * 1000
              ' If valor < total Then
              '     aux = aux + (total - valor)
              '     Me![alumbrado] = aux
              ' Else
              '     aux = aux - (valor - total)
              '     Me![alumbrado] = aux
              ' End If
            'REDONDEO
      refactura = total 'valor neto del consumo
Else
    MsgBox "No existe Pliego tarifario Ingresado"
    refactura = 0
End If
sucres.Close
dbs.Close
End Function
```