Anexo C

ÍNDEX

C.1 PLC Panasonic con AED d'HBM 3
C.1.1 Bloque de función "SistemaPesaje" 3
C.1.1 Bloque de función "Configuracion_FB" 16
C.2 PLC Panasonic con SWIFT de Utilcell 42
C.3. Arduino 56

C.1 PLC Panasonic con AED d'HBM

C.1.1 Bloque de función "SistemaPesaje"

```
CASE iEstado OF

(* Estado Esperando *)
0:
      if (bMuestreo AND NOT CommunicationError) then
            bPesoTreal := false;
            bPesoProducto := false;
            (* Después de enviar la Peticion de Cero hay que leer el
            peso para comprobar si el peso es 0 *)
            if (bPeticionCero AND NOT bPeticionLeerPesada) then
                  iEstado := 1;
            (* Leer el peso del AED *)
            elsif (bPeticionLeerPesada) then
                  iEstado := 2;
            (* Petición de iniciar pesada i partir de entrada digital
            *)
            elsif (bPesant) then
                  (* Esperamos el tiempo de la duración de pesada para
                  leer el peso *)
                  iEstado := 2;
                  bPeticioPesadaTemp := false;
            (* Leer peso en tiempo real *)
            else
                  iEstado := 3;
            end_if;

            bMuestreo := false;
      end_if;


(* Petición de Cero *)
1:
      if (bModo_Bus) then
```

```
        sDireccion := INT_TO_STRING(iDireccionBus);
        if  (iDireccionBus <> iDireccionAnterior) then
              iDireccionAnterior := iDireccionBus;
              (* Decimos que dispositivo recibirá los siguientes
              comandos *)
              sEnvioDatos := CONCAT('S', sDireccion, ';');

              (* Copiamos los caracteres de la variable
              "sEnvioDatos" a "awBufferTransmision" desde la
              posición 1 *)
              F10_BKMV(s1_Start := Adr_Of_VarOffs(Var :=
              sEnvioDatos, Offs := 2),
              s2_End := AdrLast_Of_Var(sEnvioDatos),
              d_Start => awBufferTransmision[1]);

              (* Transmitimos la trama por el iPuerto
              correspondiente *)
              F159_MTRN(s_Start := awBufferTransmision[0], n_Number
              := LEN(sEnvioDatos), d_Port := iPuertoTemp);

              (* Esperamos 10ms para volver a transmitir *)
              bTimer :=  true;
              TTransmitir(start := bTimer, SV := iOne, T =>
              bTransmitir);
              if (bTransmitir) then
                    sEnvioDatos := 'CDL;';
                    bTransmitir := false;
                    bTimer := false;
              end_if;
        else
              sEnvioDatos := 'CDL;';
        end_if;
else
        sEnvioDatos := 'CDL;';
end_if;

(* Copiamos los caracteres de la variable "sEnvioDatos" a
"awBufferTransmision" desde la posición 1 *)
F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
Offs := 2),
s2_End := AdrLast_Of_Var(sEnvioDatos),
```

```
        d_Start => awBufferTransmision[1]);

        (* Transmitimos la trama por el iPuerto correspondiente *)
        F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
        LEN(sEnvioDatos), d_Port := iPuertoTemp);

        iEstado := 4;
        iEstadoAnterior := 1;

(* Petición de leer pesada *)
2:
        if (bModo_Bus) then
                (* Cada iTiempoMuestreo hace una consulta a cada AED hasta
                consultarlos todos *)
                if (iNumeroAEDs >= iContador) then
                        iDireccionBus := iContador;
                        iContador := iContador + 1;
                else
                        iContador := 2;
                        (* Comenzamos en la dirección 1 *)
                        iDireccionBus := 1;
                end_if;

                sDireccion := INT_TO_STRING(iDireccionBus);
                if  (iDireccionBus <> iDireccionAnterior) then
                        iDireccionAnterior := iDireccionBus;
                        (* Decimos que dispositivo recibirá los siguientes
                        comandos *)
                        sEnvioDatos := CONCAT('S', sDireccion, ';');

                        (* Copiamos los caracteres de la variable
                        "sEnvioDatos" a "awBufferTransmision" desde la
                        posición 1 *)
                        F10_BKMV(s1_Start := Adr_Of_VarOffs(Var :=
                        sEnvioDatos, Offs := 2),
                        s2_End := AdrLast_Of_Var(sEnvioDatos),
                        d_Start => awBufferTransmision[1]);

                        (* Transmitimos la trama por el iPuerto
                        correspondiente *)
```

```
                F159_MTRN(s_Start := awBufferTransmision[0], n_Number
                := LEN(sEnvioDatos), d_Port := iPuertoTemp);

                (* Esperamos 10ms para volver a transmitir *)
                bTimer :=  true;
                TTransmitir2(start := bTimer, SV := iOne, T =>
                bTransmitir);
                if (bTransmitir) then
                      sEnvioDatos := 'MAV?;';
                      bTransmitir := false;
                      bTimer :=  false;
                end_if;
            else
                sEnvioDatos := 'MAV?;';
            end_if;
        else
            sEnvioDatos := 'MAV?;';
        end_if;

        (* Copiamos los caracteres de la variable "sEnvioDatos" a
        "awBufferTransmision" desde la posición 1 *)
        F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
        Offs := 2),
        s2_End := AdrLast_Of_Var(sEnvioDatos),
        d_Start => awBufferTransmision[1]);

        (* Transmitimos la trama por el iPuerto correspondiente *)

        F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
        LEN(sEnvioDatos), d_Port := iPuertoTemp);

        iEstado := 4;
        iEstadoAnterior := 2;


(* Leer el peso en tiempo real *)
3:
        if (bModo_Bus) then
            (* Cada iTiempoMuestreo hace una consulta i cada AED hasta
            consultarlos todos *)
            if (iNumeroAEDs >= iContador) then
```

```
            iDireccionBus := iContador;
            iContador := iContador + 1;
        else
            iContador := 2;
            (* Comenzamos a la dirección 1 *)
            iDireccionBus := 1;
        end_if;

        sDireccion := INT_TO_STRING(iDireccionBus);
        if  (iDireccionBus <> iDireccionAnterior) then
            iDireccionAnterior := iDireccionBus;
            (* Decimos que dispositivo recibirá los siguientes
            comandos *)
            sEnvioDatos := CONCAT('S', sDireccion, ';');

            (* Copiamos los caracteres de la variable
            "sEnvioDatos" a "awBufferTransmision" desde la
            posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var :=
            sEnvioDatos, Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto
            correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number
            := LEN(sEnvioDatos), d_Port := iPuertoTemp);

            (* Esperamos 10ms para volver a transmitir *)
            bTimer :=  true;
            TTransmitir3(start := bTimer, SV := iOne, T =>
            bTransmitir);
            if (bTransmitir) then
                sEnvioDatos := 'MSV?;';
                bTransmitir := false;
                bTimer := false;
            end_if;
        else
            sEnvioDatos := 'MSV?;';
        end_if;
    else
```

```
            sEnvioDatos := 'MSV?;';
        end_if;
        (* Copiamos los caracteres de la variable "sEnvioDatos" a
        "awBufferTransmision" desde la posición 1 *)
        F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
        Offs := 2),
        s2_End := AdrLast_Of_Var(sEnvioDatos),
        d_Start => awBufferTransmision[1]);

        (* Transmitimos la trama por el iPuerto correspondiente *)
        F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
        LEN(sEnvioDatos), d_Port := iPuertoTemp);

        iEstado := 4;
        iEstadoAnterior := 3;


(* Respuesta *)
4:
        bErrorRespuesta := false;

        (* iTimeout *)
        if (iIntentos < iNIntentos) then
            (* Sino recibimos nada antes del tiempo de iTimeout
            volveremos a enviar la trama el n° de intentos establecido
            *)
            if (bNoRespuesta) then
                iIntentos := iIntentos + 1;
                iEstado := 0;
                bNoRespuesta := false;
            end_if;
        else
            (* Si se han realizado el n° de intentos establecido se
            activa la salida "bErrorComunicacion" *)
            if (bNoRespuesta) then
                bErrorComunicacion:= true;
                iEstado := 0;
                bNoRespuesta := false;
            end_if;
        end_if;
```

```
        if (bReceptionDone AND NOT CommunicationError) then
              (* Si hemos recibido algo *)
              iIntentos := 0;
              bErrorComunicacion := false;
              bErrorRespuesta := false;

              awBufferRecepcion := ReceiveData(iPuertoTemp);
              iEstado := 20;
        end_if;


(* Guardamos los datos y calculamos el CSM *)
20:
        if (bRecepcion AND NOT CommunicationError) then
              (* Guardamos los datos recibidos a la variable
              "sTempRecepcion" *)

              (* Los dos primeros caracteres de un string son el número
              máximo de caracteres de la cadena y el número de caracteres
              que hay *)
              (* La posición 0 del buffer de recepción indica los
              caracteres recibidos *)
              F0_MV(awBufferRecepcion[0], Adr_Of_Var(sTempRecepcion));
              F0_MV(awBufferRecepcion[0],
              Adr_Of_VarOffs(sTempRecepcion,1));

              (* Copiamos los caracteres de la variable
              "awBufferRecepcion" a "sTempRecepcion" desde la posición 1
              *)
              F10_BKMV(s1_Start := Adr_Of_VarOffs(Var :=
              awBufferRecepcion, Offs := 1),
              s2_End := AdrLast_Of_Var(awBufferRecepcion),
              d_Start => Adr_Of_VarOffs(sTempRecepcion,2));
              sTempRecepcion := MID(IN := sTempRecepcion, L := 14, P :=
              2);

              (* Calculamos el CSM (EXOR)*)
              if (NOT(bPeticionCero OR bPeticionLeerPesada) AND
              bChecksum) then
                    wTotHex1 := awBufferRecepcion[2];
                    sHex1 := WORD_TO_STRING(wTotHex1);
```

```
        sHex1 := RIGHT(IN := sHex1, L := 2);
        wHex1 := STRING_TO_WORD(sHex1);

        wTotHex2 := awBufferRecepcion[3];
        sHex2 := WORD_TO_STRING(wTotHex2);
        sHex2 := RIGHT(IN := sHex2, L := 2);
        wHex2 := STRING_TO_WORD(sHex2);

        sStr1 := WORD_TO_STRING(wHex1);
        sStr2 := WORD_TO_STRING(wHex2);
        sPesoHex := CONCAT(sStr1,sStr2);
        dwPes := STRING_TO_DWORD(sPesoHex);
        diPes := DWORD_TO_DINT(dwPes);
        sPeso := DINT_TO_STRING(diPes);
        sPesoRecibido := sPeso;
        sTempRecepcion := sPesoRecibido;

        FP_BCC(s1_Control := 16#12,
         s2_Start := awBufferRecepcion[1],
         n_Bytes := 3,
         d => wCrcCalculado);

        wCrcOrigen := wTotHex2;

        IF (wCrcOrigen = wCrcCalculado) then
              bDatosOk := true;
        end_if;
end_if;

(* Comprobamos si el CSM está activado y es correcto*)
if (bDatosOk AND bChecksum) then
        bLlegirPes := true;
elsif (NOT bChecksum) then
        bLlegirPes := true;
else
        bLlegirPes := false;
end_if;

(* Analizaremos la respuesta *)
if (iEstadoAnterior = 1) then
        iEstado := 5;
```

```
            elsif (iEstadoAnterior = 2) then
                  iEstado := 6;
            elsif (iEstadoAnterior = 3) then
                  iEstado := 7;
            end_if;
            bRecepcion := false;

            (* Comprobamos si la comanda no ha sido válida
            (Retorna un ? ) *)
            if (awBufferRecepcion[1]=16#3F0A) then
                  bErrorRespuesta := true;
                  iEstado := 0;
            end_if;
      end_if;


(* Después de haber enviado una petición de Cero *)
5:
      iRespuesta := STRING_TO_INT(sTempRecepcion);
      if (iRespuesta = 0) then
            (* Esperamos 10ms para comprobar si el peso es 0*)
            bTimer :=  true;
            TCero(start := bTimer, SV := iDelay, T => bCero);
            bCero2 := true;
      else
            iEstado := 0;
      end_if;

      (* Vamos al iEstado "bPeticionLeerPesada" para leer el peso *)
      if (bCero and bCero2) then
            bPeticionLeerPesada := true;
            bMuestreo := true;
            bCero2 := false;
            bTimer := false;
            iEstado := 0;
      end_if;


(* Comprobamos cual es el peso del producto *)
6:
      if (bLlegirPes) then
```

```
        (* Si antes se ha hecho una petición de 0 con respuesta
        satisfactoria comprobamos si el peso es 0. *)
        if (bPeticionLeerPesada) then
              (* Se guarda el valor del peso *)
              DiPesoProducto := STRING_TO_DINT(sTempRecepcion);

              (* Si no hay valor la comanda MAV? retorna overflow
              *)
              if (DiPesoProducto = -1638400) then
                    DiPesoProducto := 0;
              end_if;

              if (DiPesoProducto=0 and bCero) then
                    (* Peso correcto *)
                    bCero := false;
                    (* Desactivamos la Petición de Cero porque se
                    ha hecho con éxito *)
                    bPeticionCero := false;
              elsif (DiPesoProducto<>0 and bCero) then
                    (* Se volverá a hacer la Petición de Cero *)
                    bMuestreo := true;
                    bCero := false;
              end_if;

              (* En el siguiente iEstado no tenemos que volver a
              leer el peso *)
              bPeticionLeerPesada:= false;
        else
              (* Se guarda el valor del peso *)
              DiPesoProducto := STRING_TO_DINT(sTempRecepcion);
              bPesoProducto := true;

              (* Si no hay valor la comanda MAV? retorna overflow
              *)
              if (DiPesoProducto = -1638400) then
                    DiPesoProducto := 0;
                    bPesoProducto := false;
              end_if;
        end_if;
else
        bErrorRespuesta := true;
```

```
        end_if;

        iEstado := 0;


(* Comprobamos cual es el peso en tiempo real *)
7:
        (* Comprobamos el CSM *)
        if (bLlegirPes) then
                DiPesoTreal := STRING_TO_DINT(sTempRecepcion);
                bPesoTreal := true;
        else
                bErrorRespuesta := true;
        end_if;
        iEstado := 0;

END_CASE;


if (iPuerto = 1) then
        CommunicationError := sys_blsComPort1CommunicationError;
        iPuertoTemp := SYS_COM1_PORT;
elsif (iPuerto = 2) then
        CommunicationError := sys_blsComPort2CommunicationError;
        iPuertoTemp := SYS_COM2_PORT;
end_if;

(* Cada iTiempoMuestreo comprobará el iEstado "Esperando" *)
TMuestreo(start := (iEstado = 0) and bActivacionPesada, SV :=
iTiempoMuestreo, T => bMuestreo);

bReceptionDone := IsReceptionDone(Port := iPuertoTemp);
(* iTimeout para la recibida de datos *)
TNoRespuesta(start := (iEstado = 4) AND NOT bReceptionDone, SV :=
iTimeout, T => bNoRespuesta);

(* Petición pesada *)
if (bPeticionPesada) then
        bPeticioPesadaTemp := true;
end_if;
```

```
TPesant(start := bPeticioPesadaTemp, SV := iDuracionPesada, T =>
bPesant);

(* Suma de comprobación de los parámetros legalmente relevantes. Si la
suma es diferente a la esperada se activa un bit de alarma *)
uiSumaParametrosActuales := INT_TO_UINT(iTiempoMuestreo + iTimeout +
iDuracionPesada);
if (uiSumaParametrosActuales <> uiSumaComprovacionParametros) then
      bParametrosModificados := TRUE;
else
      bParametrosModificados := FALSE;
end_if;


(* Bit que indica que se ha calibrado *)
if (bCalibracionRealizada) then
      (* Data que hay que volver a calibrar. Después de 1 año y medio
      *)
      iMesHacerCalibracion := sys_iClockCalendarMonth + 6;
      if (iMesHacerCalibracion > 12) then
            iMesHacerCalibracion := sys_iClockCalendarMonth - 12;
            iAñoHacerCalibracion := sys_iClockCalendarYear_2digits + 2;
      else
            iAñoHacerCalibracion := sys_iClockCalendarYear_2digits + 1;
      end_if;
      iDiaHacerCalibracion := sys_iClockCalendarDay;
      bCalibracionRealizada := FALSE;
end_if;

(* Si no se ha calibrado *)
if (iAñoHacerCalibracion = 0 AND iMesHacerCalibracion = 0 AND
iDiaHacerCalibracion = 0) then
      bCalibrar := FALSE;
(* Si se ha calibrado *)
else
      (* Si se cumple que estamos en el día que hay que calibrar o más
      tarde, se activa un bit de alarma *)
      if (sys_iClockCalendarYear_2digits > iAñoHacerCalibracion) then
            bCalibrar := TRUE;
      elsif (sys_iClockCalendarYear_2digits = iAñoHacerCalibracion)
      then
```

```
            if (sys_iClockCalendarMonth > iMesHacerCalibracion) then
                  bCalibrar := TRUE;
            elsif (sys_iClockCalendarMonth = iMesHacerCalibracion) then
                  if (sys_iClockCalendarDay >= iDiaHacerCalibracion)
                  then
                        bCalibrar := TRUE;
                  else
                        bCalibrar := FALSE;
                  end_if;
            else
                  bCalibrar := FALSE;
            end_if;
      else
            bCalibrar := FALSE;
      end_if;
end_if;
```

C.1.1 Bloque de función "Configuracion_FB"

```
(* Este bloque sirve para configurar una serie de parámetros *)

if (iPassword = 6114) then

CASE iEstado OF

0:

if (iNumeroInstruccion = 0) then
      iEstado := 0;

(* Select: Seleccionar dispositivo en modo bus  *)
elsif (iNumeroInstruccion = 1) then
      iEstado := 1;

(* COF: Escoger el formato de los datos de salida *)
elsif (iNumeroInstruccion = 2) then
      iEstado := 2;

(* IMD: Escoger la función modo trigger para las entradas digitales
IN1 i IN2 *)
elsif (iNumeroInstruccion = 3) then
      iEstado := 3;

(* TDD: Guardar a la EEPROM los parámetros de configuración cambiados
a la RAM *)
elsif (iNumeroInstruccion = 4) then
      iEstado := 4;

(* TRC: Configuración modo trigger *)
elsif (iNumeroInstruccion = 5) then
      iEstado := 5;

(* TAS: Selección de peso neto o bruto *)
elsif (iNumeroInstruccion = 6) then
      iEstado := 6;

(* PVS: Configurar valores extremos *)
```

```
elsif (iNumeroInstruccion = 7) then
      iEstado := 7;

(* LFT: Mode Legal for Trade *)
elsif (iNumeroInstruccion = 8) then
      iEstado := 8;

(* DPW: Definir contraseña *)
elsif (iNumeroInstruccion = 9) then
      iEstado := 9;

(* SPW: Introducir contraseña *)
elsif (iNumeroInstruccion = 10) then
      iEstado := 10;

(* FMD: Escoger modo del filtro *)
elsif (iNumeroInstruccion = 11) then
      iEstado := 11;


(* ASF: Escoger frecuencia de corte del filtro  *)
elsif (iNumeroInstruccion = 12) then
      iEstado := 12;

(* ICR: Escoger la frecuencia de muestreo *)
elsif (iNumeroInstruccion = 13) then
      iEstado := 13;

(* LDW: Poner cero en la curva característica *)
elsif (iNumeroInstruccion = 14) then
      iEstado := 14;

(* LWT: Escalaje de la curva característica *)
elsif (iNumeroInstruccion = 15) then
      iEstado := 15;

(* CWT: Calibración de pesaje *)
elsif (iNumeroInstruccion = 16) then
      iEstado := 16;

(* TCR: Contador del Legal For Trade *)
```

```
elsif (iNumeroInstruccion = 17) then
      iEstado := 17;

(* CRC: Cyclic Redundancy Check *)
elsif (iNumeroInstruccion = 18) then
      iEstado := 18;

(* CSM: Checksum *)
elsif (iNumeroInstruccion = 19) then
      iEstado := 19;

end_if;


1:
      if (NOT bLeer AND ((diP1 >= 0 AND diP1 <= 89) OR diP1 = 96 OR
      diP1 = 97 OR diP1 = 98 OR diP1 = 99)) then
            sValor := DINT_TO_STRING(diP1);
            sEnvioDatos := CONCAT('S', sValor, ';');
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iNumeroInstruccion := 0;
            (* No retorna respuesta *)
            bErrorRespuesta := false;
            bRespuestaOk := false;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
      end_if;
      iEstado := 0;

2:
```

```
        if (bLeer) then
              sEnvioDatos := 'COF?;';
              (* Copiamos los caracteres de la variable "sEnvioDatos" a
              "awBufferTransmision" desde la posición 1 *)
              F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
              Offs := 2),
              s2_End := AdrLast_Of_Var(sEnvioDatos),
              d_Start => awBufferTransmision[1]);

              (* Transmitimos la trama por el iPuerto correspondiente *)
              F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
              LEN(sEnvioDatos), d_Port := iPuertoTemp);

              iEstado := 40;
              iEstadoAnterior := 2;
        elsif (diP1 >= 0 AND diP1 <= 143) then
              sValor := DINT_TO_STRING(diP1);
              sEnvioDatos := CONCAT('COF', sValor, ';');
              (* Copiamos los caracteres de la variable "sEnvioDatos" a
              "awBufferTransmision" desde la posición 1 *)
              F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
              Offs := 2),
              s2_End := AdrLast_Of_Var(sEnvioDatos),
              d_Start => awBufferTransmision[1]);

              (* Transmitimos la trama por el iPuerto correspondiente *)
              F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
              LEN(sEnvioDatos), d_Port := iPuertoTemp);

              iEstado := 40;
              iEstadoAnterior := 2;
        else
              bRespuestaOk := false;
              bErrorRespuesta := true;
              iEstado := 0;
        end_if;

3:
        if (bLeer) then
              sEnvioDatos := 'IMD?;';
```

```
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 3;
      elsif (diP1 >= 0 AND diP1 <=2) then
            sValor :=DINT_TO_STRING(diP1);
            sEnvioDatos := CONCAT('IMD', sValor, ';');
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 3;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
      end_if;

(* Enviar un 1, para guardar a memoria EEPROM *)
4:
      if (NOT bLeer AND diP1 >= 0 AND diP1 <=2) then
            sValor := DINT_TO_STRING(diP1);
            sEnvioDatos := CONCAT('TDD', sValor, ';');
```

```
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 4;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
      end_if;

5:
      if (bLeer) then
            sEnvioDatos := 'TRC?;';
      else
            (* Función Trigger ON/OFF *)
            if (diP1 = 0 OR diP1 = 1) then
                  sValor1 := DINT_TO_STRING(diP1);
            else
                  bRespuestaOk := false;
                  bErrorRespuesta := true;
                  iEstado := 0;
                  RETURN;
            end_if;
            (* Método de trigger *)
            if (diP2 >= 0 AND diP2 <= 2) then
                  sValor2 := DINT_TO_STRING(diP2);
            else
                  bRespuestaOk := false;
                  bErrorRespuesta := true;
                  iEstado := 0;
                  RETURN;
            end_if;
```

```
        (* Nivel de trigger *)
        if (diP3 >= -1599999 AND diP3 <= 1599999) then
              sValor3 := DINT_TO_STRING(diP3);
        else
              bRespuestaOk := false;
              bErrorRespuesta := true;
              iEstado := 0;
              RETURN;
        end_if;
        (* P2 < 2: Numero de valores mesurados en el tiempo de
        estabilización
        P2 > 1: Tolerancia en dígitos *)
        if (diP4 >= 0 AND diP4 <= 99) then
              sValor4 := DINT_TO_STRING(diP4);
        else
              bRespuestaOk := false;
              bErrorRespuesta := true;
              iEstado := 0;
              RETURN;
        end_if;
        (* P2 < 2: Numero de valores mesurados en el tiempo de
        medida.
        P2 > 1: Numero de valores validos del resultado de trigger.
        *)
        if (diP5 >= 0 AND diP5 <= 99) then
              sValor5 := DINT_TO_STRING(diP5);
        else
              bRespuestaOk := false;
              bErrorRespuesta := true;
              iEstado := 0;
              RETURN;
        end_if;
        sEnvioDatos := CONCAT('TRC', sValor1, ',', sValor2, ',',
        sValor3, ',', sValor4, ',', sValor5, ';');
end_if;
(* Copiamos los caracteres de la variable "sEnvioDatos" a
"awBufferTransmision" desde la posición 1 *)
F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos, Offs :=
2),
s2_End := AdrLast_Of_Var(sEnvioDatos),
d_Start => awBufferTransmision[1]);
```

```
        (* Transmitimos la trama por el iPuerto correspondiente *)
        F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
        LEN(sEnvioDatos), d_Port := iPuertoTemp);

        iEstado := 40;
        iEstadoAnterior := 5;

6:
        if (bLeer) then
              sEnvioDatos := 'TAS?;';
              (* Copiamos los caracteres de la variable "sEnvioDatos" a
              "awBufferTransmision" desde la posición 1 *)
              F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
              Offs := 2),
              s2_End := AdrLast_Of_Var(sEnvioDatos),
              d_Start => awBufferTransmision[1]);



              (* Transmitimos la trama por el iPuerto correspondiente *)
              F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
              LEN(sEnvioDatos), d_Port := iPuertoTemp);

              iEstado := 40;
              iEstadoAnterior := 6;
        elsif (diP1 = 0 OR diP1 = 1) then
              sValor := DINT_TO_STRING(diP1);
              sEnvioDatos := CONCAT('TAS', sValor, ';');
              (* Copiamos los caracteres de la variable "sEnvioDatos" a
              "awBufferTransmision" desde la posición 1 *)
              F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
              Offs := 2),
              s2_End := AdrLast_Of_Var(sEnvioDatos),
              d_Start => awBufferTransmision[1]);

              (* Transmitimos la trama por el iPuerto correspondiente *)
              F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
              LEN(sEnvioDatos), d_Port := iPuertoTemp);

              iEstado := 40;
```

```
            iEstadoAnterior := 6;
        else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
        end_if;

7:
        if (bLeer) then
            sEnvioDatos := 'PVS?;';
        else
            if (diP1 = 0 OR diP1 = 1) then
                sValor1 := DINT_TO_STRING(diP1);
            else
                bRespuestaOk := false;
                bErrorRespuesta := true;
                iEstado := 0;
                RETURN;
            end_if;

            if (diP2 >= 0 AND diP2 <= 2) then
                sValor2 :=DINT_TO_STRING(diP2);
            else
                bRespuestaOk := false;
                bErrorRespuesta := true;
                iEstado := 0;
                RETURN;
            end_if;
            sEnvioDatos := CONCAT('PVS', sValor1,  ',', sValor2, ';');
        end_if;
        (* Copiamos los caracteres de la variable "sEnvioDatos" a
        "awBufferTransmision" desde la posición 1 *)
        F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos, Offs :=
        2),
        s2_End := AdrLast_Of_Var(sEnvioDatos),
        d_Start => awBufferTransmision[1]);

        (* Transmitimos la trama por el iPuerto correspondiente *)
        F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
        LEN(sEnvioDatos), d_Port := iPuertoTemp);
```

```
        iEstado := 40;
        iEstadoAnterior := 7;

8:
        if (bLeer) then
              sEnvioDatos := 'LFT?;';
              (* Copiamos los caracteres de la variable "sEnvioDatos" a
              "awBufferTransmision" desde la posición 1 *)
              F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
              Offs := 2),
              s2_End := AdrLast_Of_Var(sEnvioDatos),
              d_Start => awBufferTransmision[1]);

              (* Transmitimos la trama por el iPuerto correspondiente *)
              F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
              LEN(sEnvioDatos), d_Port := iPuertoTemp);

              iEstado := 40;
              iEstadoAnterior := 8;
        elsif (diP1 >= 0 AND diP1 <= 2) then
              sValor := DINT_TO_STRING(diP1);
              sEnvioDatos := CONCAT('LFT', sValor, ';');
              (* Copiamos los caracteres de la variable "sEnvioDatos" a
              "awBufferTransmision" desde la posición 1 *)
              F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
              Offs := 2),
              s2_End := AdrLast_Of_Var(sEnvioDatos),
              d_Start => awBufferTransmision[1]);

              (* Transmitimos la trama por el iPuerto correspondiente *)
              F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
              LEN(sEnvioDatos), d_Port := iPuertoTemp);

              iEstado := 40;
              iEstadoAnterior := 8;
        else
              bRespuestaOk := false;
              bErrorRespuesta := true;
              iEstado := 0;
        end_if;
```

```
9:
      if (NOT bLeer AND LEN(sDPW) <= 7) then
            sEnvioDatos := CONCAT('DPW"', sDPW, '"', ';');
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 9;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
      end_if;

10:
      if (NOT bLeer AND LEN(sSPW) <= 7) then
            sEnvioDatos := CONCAT('SPW"', sSPW, '"', ';');
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 10;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
```

```
            iEstado := 0;
        end_if;

11:
        if (bLeer) then
            sEnvioDatos := 'FMD?;';
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 11;
        elsif (diP1 >= 0 AND diP1 <= 5) then
            sValor := DINT_TO_STRING(diP1);
            sEnvioDatos := CONCAT('FMD', sValor, ';');
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 11;
        else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
        end_if;
```

```
12:
      if (bLeer) then
            sEnvioDatos := 'ASF?;';
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 12;
      elsif (diP1 >= 0 AND diP1 <= 10) then
            sValor := DINT_TO_STRING(diP1);
            sEnvioDatos := CONCAT('ASF', sValor, ';');
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 12;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
      end_if;

13:
      if (bLeer) then
            sEnvioDatos := 'ICR?;';
```

```
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 13;
      elsif (diP1 >= 0 AND diP1 <= 7) then
            sValor := DINT_TO_STRING(diP1);
            sEnvioDatos := CONCAT('ICR', sValor, ';');

            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 13;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
      end_if;

14:
      if (bLeer) then
            sEnvioDatos := 'LDW?;';
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
```

```
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 14;
      elsif (diP1 >= -1599999 AND diP1 <= 1599999) then
            sValor := DINT_TO_STRING(diP1);
            sEnvioDatos := CONCAT('LDW', sValor, ';');
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 14;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
      end_if;

15:
      if (bLeer) then
            sEnvioDatos := 'LWT?;';
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
```

```
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 15;
      elsif (diP1 >= -1599999 AND diP1 <= 1599999) then
            sValor := DINT_TO_STRING(diP1);
            sEnvioDatos := CONCAT('LWT', sValor, ';');
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 15;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
      end_if;

16:
      if (bLeer) then
            sEnvioDatos := 'CWT?;';
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
```

```
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 16;
      elsif (diP1 >= 200000 AND diP1 <= 1200000) then
            sValor := DINT_TO_STRING(diP1);
            sEnvioDatos := CONCAT('CWT', sValor, ';');
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 16;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
      end_if;

17:
      if (bLeer) then
            sEnvioDatos := 'TCR?;';
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);
```

```
            iEstado := 40;
            iEstadoAnterior := 17;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
      end_if;

18:
      if (bLeer) then
            sEnvioDatos := 'CRC?;';
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 18;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
      end_if;

19:
      if (bLeer) then
            sEnvioDatos := 'CSM?;';

            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);
```

```
            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 19;
      elsif (diP1 >= 0 AND diP1 <= 2) then
            sValor := DINT_TO_STRING(diP1);
            sEnvioDatos := CONCAT('CSM', sValor, ';');
            (* Copiamos los caracteres de la variable "sEnvioDatos" a
            "awBufferTransmision" desde la posición 1 *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos,
            Offs := 2),
            s2_End := AdrLast_Of_Var(sEnvioDatos),
            d_Start => awBufferTransmision[1]);

            (* Transmitimos la trama por el iPuerto correspondiente *)
            F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
            LEN(sEnvioDatos), d_Port := iPuertoTemp);

            iEstado := 40;
            iEstadoAnterior := 19;
      else
            bRespuestaOk := false;
            bErrorRespuesta := true;
            iEstado := 0;
      end_if;

(* Respuesta *)
40:
      bErrorRespuesta := false;
      bRespuestaOk := false;
      diRespuestaP1 := 0;
      diRespuestaP2 :=  0;
      diRespuestaP3 :=  0;
      diRespuestaP4 :=  0;
      diRespuestaP5 :=  0;

      (* timeout *)
      if (iIntentos < iNumeroIntentos) then
```

```
            (* Sino recibimos nada antes del tiempo de timeout
            volveremos a enviar la trama el n° de intentos establecido
            *)
            if (bNoRespuesta) then
                  iIntentos := iIntentos + 1;
                  iEstado := 0;
                  bNoRespuesta := false;
            end_if;
      else
            (* Si se han realizado el n° de intentos establecido se
            activa la salida "bErrorComunicacion" *)
            if (bNoRespuesta) then
                  bErrorComunicacion:= true;
                  iEstado := 0;
                  bNoRespuesta := false;
            end_if;
      end_if;

      if (bReceptionDone AND NOT CommunicationError) then
            (* Si hemos recibido algo *)
            iIntentos := 0;
            bErrorComunicacion := false;

            awBufferRecepcion := ReceiveData(iPuertoTemp);
            iEstado := 41;
      end_if;

(* Guardamos los datos *)
41:
      if (bRecepcion AND NOT CommunicationError) then
            (* Guardamos los datos recibidos a la variable
            "sTempRecepcion" *)

            (* Los dos primeros caracteres de un string son el número
            máximo de caracteres de la cadena y el numero de caracteres
            que hay *)
            (* La posición 0 del buffer de recepción indica los
            caracteres recibidos *)
            F0_MV(awBufferRecepcion[0], Adr_Of_Var(sTempRecepcion));
            F0_MV(awBufferRecepcion[0],
            Adr_Of_VarOffs(sTempRecepcion,1));
```

```
(* Copiamos los caracteres de la variable
"awBufferRecepcion" a "sTempRecepcion" desde la posición 1.
*)
F10_BKMV(s1_Start := Adr_Of_VarOffs(Var :=
awBufferRecepcion, Offs := 1),
s2_End := AdrLast_Of_Var(awBufferRecepcion),
d_Start => Adr_Of_VarOffs(sTempRecepcion,2));

(* Analizaremos la respuesta *)
if (iEstadoAnterior = 2 OR iEstadoAnterior = 3 OR
iEstadoAnterior = 6 OR iEstadoAnterior = 8 OR
iEstadoAnterior = 11 OR iEstadoAnterior = 12 OR
iEstadoAnterior = 13 OR iEstadoAnterior = 14 OR
iEstadoAnterior = 15 OR iEstadoAnterior = 16 OR
iEstadoAnterior = 19) then
      iEstado := 25;
elsif (iEstadoAnterior = 4) then
      iEstado := 26;
elsif (iEstadoAnterior = 5) then
      iEstado := 27;
elsif (iEstadoAnterior = 7) then
      iEstado := 28;
elsif (iEstadoAnterior = 9) then
      iEstado := 29;
elsif (iEstadoAnterior = 10) then
      iEstado := 30;
elsif (iEstadoAnterior = 17) then
      iEstado := 31;
elsif (iEstadoAnterior = 18) then
      iEstado := 32;
end_if;


(* Comprobamos si la comanda no ha sido válida
(Retorna un ? ) *)
if (awBufferRecepcion[1]=16#3F0A) then
      bErrorRespuesta := true;
      iEstado := 0;
end_if;
```

```
            bRecepcion :=  false;
      end_if;

(* COF, IMD, TAS, LFT, FMD, ASF, ICR, LDW, LWT, CWT, CSM *)
25:
      diRespuestaP1 := STRING_TO_DINT(sTempRecepcion);    (* Aquí hay
      la respuesta *)
      if (bLeer) then
            iNumeroInstruccion := 0;
            bRespuestaOk := true;
            iEstado := 0;
      else
            if (diRespuestaP1 = 0) then (* Si la respuesta es correcta
            *)
                  iNumeroInstruccion := 0;
                  bRespuestaOk := true;
                  iEstado := 0;
            end_if;
      end_if;

(* TDD *)
26:
      diRespuestaP1 := STRING_TO_DINT(sTempRecepcion);    (* Aquí hay
      la respuesta *)
      if (diRespuestaP1 = 0) then  (* Si la respuesta es correcta *)
                  iNumeroInstruccion := 0;
                  bRespuestaOk := true;
                  iEstado := 0;
      end_if;

(* TRC *)
27:
      bRespuestaOk := true;
      if (bLeer) then
            (* Guardamos el 1r número *)
            diRespuestaP1 := STRING_TO_DINT(MID(sTempRecepcion,1,1));

            (* Guardamos el 2n número *)
            iPosicion := FIND(sTempRecepcion,',');
            iPosicionNumero := LEN(sTempRecepcion) - iPosicion;
            sTempRecepcion := RIGHT(sTempRecepcion,iPosicionNumero);
```

```
        iPosicion2 := FIND(sTempRecepcion,',');
        iNumCaracteres := iPosicion2 - 1;
        diRespuestaP2 :=
        STRING_TO_DINT(MID(sTempRecepcion,iNumCaracteres,1));

        (* Guardamos el 3r número *)
        iPosicion := FIND(sTempRecepcion,',');
        iPosicionNumero := LEN(sTempRecepcion) - iPosicion;
        sTempRecepcion := RIGHT(sTempRecepcion,iPosicionNumero);
        iPosicion2 := FIND(sTempRecepcion,',');
        iNumCaracteres := iPosicion2 - 1;
        diRespuestaP3 :=
        STRING_TO_DINT(MID(sTempRecepcion,iNumCaracteres,1));

        (* Guardamos el 4t número *)
        iPosicion := FIND(sTempRecepcion,',');
        iPosicionNumero := LEN(sTempRecepcion) - iPosicion;
        sTempRecepcion := RIGHT(sTempRecepcion,iPosicionNumero);
        iPosicion2 := FIND(sTempRecepcion,',');
        iNumCaracteres := iPosicion2 - 1;
        diRespuestaP4 :=
        STRING_TO_DINT(MID(sTempRecepcion,iNumCaracteres,1));

        (* Guardamos el 5é número *)
        iPosicion := FIND(sTempRecepcion,',');
        iPosicionNumero := LEN(sTempRecepcion) - iPosicion;
        sTempRecepcion := RIGHT(sTempRecepcion,iPosicionNumero);
        iNumCaracteres := LEN(sTempRecepcion);
        diRespuestaP5 :=
        STRING_TO_DINT(MID(sTempRecepcion,iNumCaracteres,1));

        iNumeroInstruccion := 0;
        bRespuestaOk := true;
        iEstado := 0;
else
        diRespuestaP1 := STRING_TO_DINT(sTempRecepcion);    (* Aquí
        hay la respuesta *)
        if (diRespuestaP1 = 0) then (* Si la respuesta es correcta
        *)
                iNumeroInstruccion := 0;
                bRespuestaOk := true;
```

```
                iEstado := 0;
            end_if;
        end_if;


(* PVS *)
28:
        if (bLeer) then
            (* Guardamos el 1r número *)
            diRespuestaP1 := STRING_TO_DINT(MID(sTempRecepcion,1,1));

            (* Guardamos el 2n número *)
            diRespuestaP2 := STRING_TO_DINT(MID(sTempRecepcion,1,3));

            iNumeroInstruccion := 0;
            bRespuestaOk := true;
            iEstado := 0;
        else
            diRespuestaP1 := STRING_TO_DINT(sTempRecepcion);    (* Aquí
            hay la respuesta *)
            if (diRespuestaP1 = 0) then  (* Si la respuesta es correcta
            *)
                        iNumeroInstruccion := 0;
                        bRespuestaOk := true;
                        iEstado := 0;
            end_if;
        end_if;

(* DPW *)
29:
        diRespuestaP1 := STRING_TO_DINT(sTempRecepcion);    (* Aquí hay
        la respuesta *)
        if (diRespuestaP1 = 0) then  (* Si la respuesta es correcta *)
            iNumeroInstruccion := 0;
            bRespuestaOk := true;
            iEstado := 0;
        end_if;

(* SPW *)
30:
```

```
        diRespuestaP1 := STRING_TO_DINT(sTempRecepcion);    (* Aquí hay
        la respuesta *)
        if (diRespuestaP1 = 0) then  (* Si la respuesta es correcta *)
              iNumeroInstruccion := 0;
              bRespuestaOk := true;
              iEstado := 0;
        end_if;

(* TCR*)
31:
        diRespuestaP1 := STRING_TO_DINT(sTempRecepcion);    (* Aquí hay
        la respuesta *)
        if (bLeer) then
              iNumeroInstruccion := 0;
              bRespuestaOk := true;
              iEstado := 0;
        end_if;

(* CRC  *)
32:
        diRespuestaP1 := STRING_TO_DINT(sTempRecepcion);    (* Aquí hay
        la respuesta *)
        if (bLeer) then
              iNumeroInstruccion := 0;
              bRespuestaOk := true;
              iEstado := 0;
        end_if;

END_CASE;

bReceptionDone := IsReceptionDone(Port := iPuertoTemp);
(* Timeout para la recibida de datos *)
TNoRespuesta(start := iEstado = 40 AND NOT bReceptionDone, SV :=
iTimeout, T => bNoRespuesta);

if (iPuerto = 1) then
        CommunicationError := sys_blsComPort1CommunicationError;
        iPuertoTemp := SYS_COM1_PORT;
elsif (iPuerto = 2) then
        CommunicationError := sys_blsComPort2CommunicationError;
        iPuertoTemp := SYS_COM2_PORT;
```

```
end_if;

end_if;
```

C.2 PLC Panasonic con SWIFT de Utilcell

```
CASE iEstado OF

(* iEstado Esperando *)
0:
      if (bMuestreo AND NOT bCommunicationError) then

            if (iNumeroBasculas = 0 AND bModoBus AND NOT bPeticionCom)
            then
                  (* MODO BUS *)
                  (* Miramos que básculas están conectadas (Con un
                  while no funciona del todo bien...)   *)
                  (*
                  while (i<16) do
                        if (abBasculasActivadas[i]) then
                              iProva2 := iProva2 + 1;

                              bMarxa := true;
                              aiOrdreLectura[iNumeroBasculas]:= i;
                              iNumeroBasculas:= iNumeroBasculas + 1;
                        end_if;
                        i := i + 1;
                  end_while;
                  *)

                  if (abBasculasActivadas[0] OR abBasculasActivadas[1]
                  OR abBasculasActivadas[2] OR abBasculasActivadas[3]
                  OR abBasculasActivadas[4] OR abBasculasActivadas[5]
                  OR abBasculasActivadas[6] OR abBasculasActivadas[7]
                  OR abBasculasActivadas[8] OR abBasculasActivadas[9]
                  OR abBasculasActivadas[10] OR abBasculasActivadas[11]
                  OR abBasculasActivadas[12] OR abBasculasActivadas[13]
                  OR abBasculasActivadas[14] OR abBasculasActivadas[15]
                  ) then
                        bMarxa := TRUE;
                  else
                        bMarxa := FALSE;
                  end_if;

                  iNumeroBasculas:= 0;
                  if (abBasculasActivadas[0]) then
                        aiOrdreLectura[iNumeroBasculas]:= 1;
                        iNumeroBasculas:= iNumeroBasculas + 1;
                  else
                        aiOrdreLectura[iNumeroBasculas]:= 0;
                  end_if;

                  if (abBasculasActivadas[1]) then
                        aiOrdreLectura[iNumeroBasculas]:= 2;
                        iNumeroBasculas:= iNumeroBasculas + 1;
                  else
                        aiOrdreLectura[iNumeroBasculas]:= 0;
                  end_if;

                  if (abBasculasActivadas[2]) then
                        aiOrdreLectura[iNumeroBasculas]:= 3;
                        iNumeroBasculas:= iNumeroBasculas + 1;
                  else
```

```
      aiOrdreLectura[iNumeroBasculas]:= 0;
end_if;

if (abBasculasActivadas[3]) then
    aiOrdreLectura[iNumeroBasculas]:= 4;
    iNumeroBasculas:= iNumeroBasculas + 1;
else
    aiOrdreLectura[iNumeroBasculas]:= 0;
end_if;

if (abBasculasActivadas[4]) then
    aiOrdreLectura[iNumeroBasculas]:= 5;
    iNumeroBasculas:= iNumeroBasculas + 1;
else
    aiOrdreLectura[iNumeroBasculas]:= 0;
end_if;

if (abBasculasActivadas[5]) then
    aiOrdreLectura[iNumeroBasculas]:= 6;
    iNumeroBasculas:= iNumeroBasculas + 1;
else
    aiOrdreLectura[iNumeroBasculas]:= 0;
end_if;

if (abBasculasActivadas[6]) then
    aiOrdreLectura[iNumeroBasculas]:= 7;
    iNumeroBasculas:= iNumeroBasculas + 1;
else
    aiOrdreLectura[iNumeroBasculas]:= 0;
end_if;

if (abBasculasActivadas[7]) then
    aiOrdreLectura[iNumeroBasculas]:= 8;
    iNumeroBasculas:= iNumeroBasculas + 1;
else
    aiOrdreLectura[iNumeroBasculas]:= 0;
end_if;

if (abBasculasActivadas[8]) then
    aiOrdreLectura[iNumeroBasculas]:= 9;
    iNumeroBasculas:= iNumeroBasculas + 1;
else
    aiOrdreLectura[iNumeroBasculas]:= 0;
end_if;

if (abBasculasActivadas[9]) then
    aiOrdreLectura[iNumeroBasculas]:= 10;
    iNumeroBasculas:= iNumeroBasculas + 1;
else
    aiOrdreLectura[iNumeroBasculas]:= 0;
end_if;

if (abBasculasActivadas[10]) then
    aiOrdreLectura[iNumeroBasculas]:= 11;
    iNumeroBasculas:= iNumeroBasculas + 1;
else
    aiOrdreLectura[iNumeroBasculas]:= 0;
end_if;

if (abBasculasActivadas[11]) then
    aiOrdreLectura[iNumeroBasculas]:= 12;
```

```
            iNumeroBasculas:= iNumeroBasculas + 1;
        else
            aiOrdreLectura[iNumeroBasculas]:= 0;
        end_if;

        if (abBasculasActivadas[12]) then
            aiOrdreLectura[iNumeroBasculas]:= 13;
            iNumeroBasculas:= iNumeroBasculas + 1;
        else
            aiOrdreLectura[iNumeroBasculas]:= 0;
        end_if;

        if (abBasculasActivadas[13]) then
            aiOrdreLectura[iNumeroBasculas]:= 14;
            iNumeroBasculas:= iNumeroBasculas + 1;
        else
            aiOrdreLectura[iNumeroBasculas]:= 0;
        end_if;

        if (abBasculasActivadas[14]) then
            aiOrdreLectura[iNumeroBasculas]:= 15;
            iNumeroBasculas:= iNumeroBasculas + 1;
        else
            aiOrdreLectura[iNumeroBasculas]:= 0;
        end_if;

        if (abBasculasActivadas[15]) then
            aiOrdreLectura[iNumeroBasculas]:= 16;
            iNumeroBasculas:= iNumeroBasculas + 1;
        else
            aiOrdreLectura[iNumeroBasculas]:= 0;
        end_if;

end_if;

bPeticionCom := false;
bPesoTreal := false;
bPesoProducto := false;
bPesoBruto := false;
(* Después de enviar la Petición de Cero hay que entrar a
bPeticionLeerpesada para comprobar si el peso es 0 *)
if (bPeticionCero AND NOT bPeticionLeerPesada) then
      iEstado := 1;
(* Leer el peso de la báscula *)
elsif (bPeticionLeerPesada) then
      iEstado := 2;
(* Petición de iniciar pesada a partir de entrada digital
*)
elsif (bPesant) then
(* Esperamos el tiempo de la duración de pesada para leer
el peso *)
      iEstado := 2;
      bPeticioPesadaTemp := false;
(* Petición de iniciar pesada a partir de comunicaciones *)
elsif (bPeticionPesadaComunicaciones) then
      iEstado := 3;
(* Leer peso en tiempo real *)
else
      iEstado := 4;
end_if;
```

```
            bMuestreo := false;
      end_if;


(* Petición de Cero *)
1:
      if (bModoBus) then
            (* Miramos si el número tiene un solo digito *)
            if (iDireccionBus >= 0 AND iDireccionBus < 10) then
                  bDigito := true;
            end_if;

            (* Guardamos la dirección obtenida gracias a la variable de
            entrada "iDireccionBus" *)
            sDireccion := INT_TO_STRING(iDireccionBus);
            if (MID(IN := sDireccion, L := 1, P := 1)=' ') then
                  sDireccion := MID(IN := sDireccion, L := 1, P := 2);
            end_if;

            (* Preparamos la trama a enviar, donde se indica que
            dispositivo recibirá la trama *)
            if (bDigito) then
                  sEnvioDatos := CONCAT('#0', sDireccion, 'Z');
                  bDigito := false;
            else
                  sEnvioDatos := CONCAT('#', sDireccion, 'Z');
            end_if;
      else
            sEnvioDatos := 'Z';
      end_if;

      (* Copiamos los caracteres de la variable "sEnvioDatos" a
      "awBufferTransmision" desde la posición 1 *)
      F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos, Offs :=
      2),
      s2_End := AdrLast_Of_Var(sEnvioDatos),
      d_Start => awBufferTransmision[1]);

      (* Transmitimos la trama por el iPuerto correspondiente *)
      F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
      LEN(sEnvioDatos), d_Port := iPuertoTemp);

      (* Después de enviar el comando de Cero, vamos al iEstado=0 para
      leer el peso *)
      bPeticionLeerPesada := true;
      bMuestreo := true;
      bCero := true;
      iEstado := 0;


(* Petición de leer pesada *)
2:
      if (bModoBus) then
            (* Miramos si hay alguna báscula activada *)
            if (bMarxa) then
                  (* Si antes se ha enviado una petición de cero hay
                  que coger la dirección de la variable de entrada
                  "iDireccionBus" *)
                  if (bPeticionCero OR bPeticionPesadaComunicaciones)
                  then
                        iDireccionBusTemp := iDireccionBus;
```

```
            else
                  (* Cada iTs hace una consulta a cada báscula
                  que este activa hasta consultarlas todas *)
                  (* Guardamos la dirección obtenida gracias a la
                  variable de entrada "abBasculasActivadas" *)
                  iDireccionBusPesada :=
                  aiOrdreLectura[iPunterLectura];
                  iDireccionBusTemp := iDireccionBusPesada;
            end_if;

            (* Miramos si el número tiene un solo digito *)
            if (iDireccionBusTemp >= 0 AND iDireccionBusTemp <
            10) then
                  bDigito := true;
            end_if;
            sDireccion := INT_TO_STRING(iDireccionBusTemp);
            if (MID(IN := sDireccion, L := 1, P := 1)=' ') then
                  sDireccion := MID(IN := sDireccion, L := 1, P
                  := 2);
            end_if;

            (* Preparamos la trama a enviar, donde se indica que
            dispositivo recibirá la trama *)
            if (bDigito) then
                  sEnvioDatos := CONCAT('#0', sDireccion, 'CWR');
                  bDigito := false;
            else
                  sEnvioDatos := CONCAT('#', sDireccion, 'CWR');
            end_if;

            if (NOT bPeticionCero AND NOT
            bPeticionPesadaComunicaciones) then
                  (* Miramos si se ha enviado la comanda a todas
                  las basculas activas *)
                  iPunterLectura:= iPunterLectura + 1;
                  if (iPunterLectura >= iNumeroBasculas) then
                        iPunterLectura := 0;
                        iNumeroBasculas := 0;
                        i := 0;
                  end_if;
            end_if;
      (* Si no hay ninguna báscula activada *)
      else
            bErrorComunicacion := true;
            iEstado := 0;
            return;
      end_if;
(* Si no estamos en modo bus *)
else
      sEnvioDatos := 'CWR';
end_if;

(* Copiamos los caracteres de la variable "sEnvioDatos" a
"awBufferTransmision" desde la posición 1 *)
F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos, Offs :=
2),
s2_End := AdrLast_Of_Var(sEnvioDatos),
d_Start => awBufferTransmision[1]);

(* Transmitimos la trama por el iPuerto correspondiente *)
```

```
        F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
        LEN(sEnvioDatos), d_Port := iPuertoTemp);

        iEstado := 5;
        iEstadoAnterior := 2;


(* Petición de iniciar pesada a partir de comunicaciones *)
3:
        if (bModoBus) then
                (* Miramos si hay alguna báscula activada *)
                if (bMarxa) then
                        iDireccionBusPesada := iDireccionBus;
                        (* Miramos si el número tiene un solo digito *)
                        if (iDireccionBusPesada >= 0 AND iDireccionBusPesada
                        < 10) then
                                bDigito := true;
                        end_if;

                        (* Guardamos la dirección obtenida gracias a la
                        variable de entrada "iDireccionBus" *)
                        sDireccion := INT_TO_STRING(iDireccionBusPesada);
                        if (MID(IN := sDireccion, L := 1, P := 1)=' ') then
                                sDireccion := MID(IN := sDireccion, L := 1, P
                                := 2);
                        end_if;

                        (* Preparamos la trama a enviar, donde se indica que
                        dispositivo recibirá la trama *)
                        if (bDigito) then
                                sEnvioDatos := CONCAT('#0', sDireccion, 'CWI');
                                bDigito := false;
                        else
                                sEnvioDatos := CONCAT('#', sDireccion, 'CWI');
                        end_if;

                (* Si no hay ninguna báscula activada *)
                else
                        bErrorComunicacion := true;
                        iEstado := 0;
                        return;
                end_if;
        (* Si no estamos en modo bus *)
        else
                sEnvioDatos := 'CWI';
        end_if;

        (* Copiamos los caracteres de la variable "sEnvioDatos" a
        "awBufferTransmision" desde la posición 1 *)
        F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos, Offs :=
        2),
        s2_End := AdrLast_Of_Var(sEnvioDatos),
        d_Start => awBufferTransmision[1]);

        (* Transmitimos la trama por el iPuerto correspondiente *)
        F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
        LEN(sEnvioDatos), d_Port := iPuertoTemp);

        iEstado := 5;
        iEstadoAnterior := 3;
```

```
(* Leer el peso en tiempo real *)
4:
      if (bModoBus) then
            (* Miramos si hay alguna báscula activada *)
            if (bMarxa) then
                  (* Cada iTs hace una consulta a cada bascula que este
                  activa hasta consultarlas todas *)
                  iDireccionBusPesada :=
                  aiOrdreLectura[iPunterLectura];

                  (* Miramos si el número tiene un solo digito *)
                  if (iDireccionBusPesada >= 0 AND iDireccionBusPesada
                  < 10) then
                        bDigito := true;
                  end_if;

                  (* Guardamos la dirección obtenida gracias a la
                  variable de entrada "abBasculasActivadas" *)
                  sDireccion := INT_TO_STRING(iDireccionBusPesada);
                  if (MID(IN := sDireccion, L := 1, P := 1)=' ') then
                        sDireccion := MID(IN := sDireccion, L := 1, P
                        := 2);
                  end_if;

                  (* Preparamos la trama a enviar, donde se indica que
                  dispositivo recibirá la trama *)
                  if (bDigito) then
                        sEnvioDatos := CONCAT('#0', sDireccion, 'P');
                        bDigito := false;
                  else
                        sEnvioDatos := CONCAT('#', sDireccion, 'P');
                  end_if;

                  (* Miramos si se ha enviado la comanda a todas las
                  basculas activas *)
                  iPunterLectura:= iPunterLectura + 1;
                  if (iPunterLectura >= iNumeroBasculas) then
                        iPunterLectura := 0;
                        iNumeroBasculas:= 0;
                        i := 0;
                  end_if;

            (* Si no hay ninguna báscula activada *)
            else
                  bErrorComunicacion := true;
                  iEstado := 0;
                  return;
            end_if;
      (* Si no estamos en modo bus *)
      else
            sEnvioDatos := 'P';
      end_if;

      (* Copiamos los caracteres de la variable "sEnvioDatos" a
      "awBufferTransmision" desde la posición 1 *)
      F10_BKMV(s1_Start := Adr_Of_VarOffs(Var := sEnvioDatos, Offs :=
      2),
      s2_End := AdrLast_Of_Var(sEnvioDatos),
      d_Start => awBufferTransmision[1]);
```

```
      (* Transmitimos la trama por el iPuerto correspondiente *)
      F159_MTRN(s_Start := awBufferTransmision[0], n_Number :=
      LEN(sEnvioDatos), d_Port := iPuertoTemp);

      iEstado := 5;
      iEstadoAnterior := 4;


(* Respuesta *)
5:
      (* Sino recibimos nada antes del tiempo de iTimeout se activa la
      salida "bErrorComunicacion" *)
      if (bNoRespuesta) then
            bErrorComunicacion:= true;
            iEstado := 0;
            bNoRespuesta := false;
      end_if;

      (* Recibir respuesta *)
      if (DF(bReceptionDone) AND NOT bCommunicationError) then
            (*
            iLecturasPorMinuto := iLecturasPorMinuto + 1;
            *)

            (* Al activarse la variable "bReceptionDone" se guardan los
            datos recibidos en el buffer de recepción especificado. *)

            iIntentos := 0;
            bErrorComunicacion := false;
            bErrorRespuesta := false;

            (* Guardamos los datos recibidos a la variable
            "sTempRecepcion" *)
            (* Los dos primeros caracteres de un string son el número
            máximo de caracteres de la cadena y el número de caracteres
            que hay *)
            (* La posición 0 del buffer de recepción indica los
            caracteres recibidos *)
            F0_MV(awBufferRecepcion[0], Adr_Of_Var(sTempRecepcion));
            F0_MV(awBufferRecepcion[0],
Adr_Of_VarOffs(sTempRecepcion,1));

            (* Copiamos los caracteres de la variable
            "awBufferRecepcion" a "sTempRecepcion" desde la posición 1
            *)
            F10_BKMV(s1_Start := Adr_Of_VarOffs(Var :=
            awBufferRecepcion, Offs := 1),
            s2_End := AdrLast_Of_Var(awBufferRecepcion),
            d_Start => Adr_Of_VarOffs(sTempRecepcion,2));


            bReceptionDone := false;

            (* Analizaremos la respuesta *)
            if (iEstadoAnterior = 2) then
                  iEstado := 6;
            elsif (iEstadoAnterior = 3) then
                  iEstado := 7;
            elsif (iEstadoAnterior = 4) then
                  iEstado := 8;
            end_if;
```

```
        end_if;


(* Comprobamos cual es el peso del producto *)
6:
        (* MODO BUS *)
        if (bModoBus) then
                (* Comprobamos si la respuesta tiene la longitud correcta
                *)
                if (LEN(sTempRecepcion)=14) then
                        (* Guardamos el primer carácter de la respuesta *)
                        sModoBus := MID(IN := sTempRecepcion, L := 1, P :=
                        1);

                        (* Guardamos la dirección de la respuesta *)
                        sDireccionBusTemp := MID(IN := sTempRecepcion, L :=
                        2, P := 2);

                        (* Comprobamos si este trozo de respuesta es correcto
                        *)
                        if (sModoBus <> '<' OR iDireccionBusPesada <>
                        STRING_TO_INT(sDireccionBusTemp)) then
                                bErrorRespuesta := true;
                                iEstado := 0;
                                return;
                        end_if;

                        (* Quitamos los 3 primeros caracteres de la
                        respuesta, para así poder analizar la respuesta como
                        si no hubiera modo bus *)
                        sTempRecepcion := RIGHT(IN := sTempRecepcion, L :=
                        (LEN(sTempRecepcion)-3));

                (* Si la respuesta no tiene longitud 14 la respuesta es
                incorrecta *)
                else
                        bErrorRespuesta := true;
                        iEstado := 0;
                        return;
                end_if;
        end_if;

        (* SIN MODO BUS *)
        (* Comprobamos si la respuesta tiene la longitud correcta *)
        if (LEN(sTempRecepcion)=11) then
                (* Guardamos los tres primeros caracteres de la respuesta
                *)
                sNombreInstruccion := MID(IN := sTempRecepcion, L := 3, P
                := 1);

                (* Guardamos el cuarto carácter de la respuesta *)
                sEstadoPeso := MID(IN := sTempRecepcion, L := 1, P := 4);

                (* Si antes se ha hecho una petición de 0, comprobamos si
                el peso es 0. *)
                if (bPeticionLeerPesada) then
                        (* Comprobamos si la respuesta es correcta *)
                        if ((sNombreInstruccion <> 'CWR') OR (sEstadoPeso =
                        '3')) then
                                DiPesoProducto := 1; (* Peso diferente de 0 *)
```

```
        else
              (* Estado del peso: Vacio *)
              if (sEstadoPeso = '2') then
                    (* Cogemos los caracteres desde la
                    posición 5, donde se encuentra el peso *)
                    sTempRecepcion := MID(IN :=
                    sTempRecepcion, L :=7, P := 5);

                    (* Se guarda el valor del peso *)
                    DiPesoProducto :=
                    STRING_TO_DINT(sTempRecepcion);
              else
                    (* Peso diferente de 0 *)
                    DiPesoProducto := 1;
              end_if;
        end_if;

        (* Comprobamos si el peso es 0. *)
        if (DiPesoProducto=0 and bCero) then
              (* Peso correcto *)
              bCero := false;
              (* Desactivamos la Petición de Cero porque se
              ha hecho con éxito *)
              bPeticionCero := false;
        elsif (DiPesoProducto<>0 and bCero) then
              (* Se volverá a hacer la Petición de Cero *)
              bMuestreo := true;
              bCero := false;
        end_if;

        (* En el siguiente iEstado no tenemos que volver a
        leer el peso *)
        bPeticionLeerPesada:= false;

  (* Si queremos guardar el valor de una nueva pesada *)
  else
        (* Comprobamos si la respuesta es correcta *)
        if ((sNombreInstruccion <> 'CWR') OR (sEstadoPeso =
        '3')) then
              bErrorRespuesta := true;
        else
              (* Estado del peso: Nuevo *)
              if (sEstadoPeso = '1') then

                    (* Cogemos los caracteres desde la
                    posición 5, donde se encuentra el peso *)
                    if (LEN(sTempRecepcion) = 11) then
                          sTempRecepcion := MID(IN :=
                          sTempRecepcion, L :=7, P := 5);
                    end_if;

                    (* Se guarda el valor del peso *)
                    DiPesoProducto :=
                    STRING_TO_DINT(sTempRecepcion);
                    bPesoProducto := true;
              else
                    (* Si el estado del peso es uno de los
                    dos siguientes no hacemos nada:
                       0 -> Vacio
                       2 -> Leído
                    *)
```

```
                                if (NOT (sEstadoPeso = '0' OR sEstadoPeso
                                = '2')) then
                                      bErrorRespuesta := true;
                                end_if;
                          end_if;
                    end_if;
              end_if;
        (* Si la respuesta no tiene longitud 11 la respuesta es
        incorrecta *)
        else
              bErrorRespuesta := true;
        end_if;

        iEstado := 0;


(* Comprobamos si el comando para el arranque del ciclo de pesaje ha
sido correcto *)
7:
        (* MODO BUS *)
        if (bModoBus) then
              (* Comprobamos si la respuesta tiene la longitud correcta
              *)
              if (LEN(sTempRecepcion)=7) then
                    (* Guardamos el primer carácter de la respuesta *)
                    sModoBus := MID(IN := sTempRecepcion, L := 1, P :=
                    1);

                    (* Guardamos la dirección de la respuesta *)
                    sDireccionBusTemp := MID(IN := sTempRecepcion, L :=
                    2, P := 2);

                    (* Comprobamos si este trozo de respuesta es correcto
                    *)
                    if (sModoBus <> '<' OR iDireccionBusPesada <>
                    STRING_TO_INT(sDireccionBusTemp)) then
                          bErrorRespuesta := true;
                          iEstado := 0;
                          return;
                    end_if;

                    (* Quitamos los 3 primeros caracteres de la
                    respuesta, para así poder analizar la respuesta como
                    si no hubiera modo bus *)
                    sTempRecepcion := RIGHT(IN := sTempRecepcion, L :=
                    (LEN(sTempRecepcion)-3));

              (* Si la respuesta no tiene longitud 7 la respuesta es
              incorrecta *)
              else
                    bErrorRespuesta := true;
                    iEstado := 0;
                    return;
              end_if;
        end_if;

        (* SIN MODO BUS *)
        (* Comprobamos si la respuesta tiene la longitud correcta *)
        if (LEN(sTempRecepcion)=4) then
              (* Guardamos los tres primeros caracteres de la respuesta
              *)
```

```
            sNombreInstruccion := MID(IN := sTempRecepcion, L := 3,
            P := 1);

            (* Guardamos el cuarto carácter de la respuesta *)
            sEstadoPeso := MID(IN := sTempRecepcion, L := 1, P := 4);

      (* Si la respuesta no tiene longitud 7 la respuesta es
      incorrecta *)
      else
            bErrorRespuesta := true;
            iEstado := 0;
            return;
      end_if;

      (* Comprobamos si la respuesta es correcta *)
      if ((sNombreInstruccion <> 'CWI') OR (sEstadoPeso = 'N')) then
            bErrorRespuesta := true;
      else
            if (sEstadoPeso = 'A') then
                  bIniciarPesada := true;
                  bPeticionCom := true;
            else
                  bErrorRespuesta := true;
            end_if;
      end_if;

      iEstado := 0;


(* Comprobamos cual es el peso en tiempo real *)
8:
      (* MODO BUS *)
      if (bModoBus) then
            (* Comprobamos si la respuesta tiene la longitud correcta
            *)
            if (LEN(sTempRecepcion)=15) then
                  (* Guardamos el primer carácter de la respuesta *)
                  sModoBus := MID(IN := sTempRecepcion, L := 1, P :=
                  1);

                  (* Guardamos la dirección de la respuesta *)
                  sDireccionBusTemp := MID(IN := sTempRecepcion, L :=
                  2, P := 2);

                  (* Comprobamos si este trozo de respuesta es correcto
                  *)
                  if (sModoBus <> '<' OR iDireccionBusPesada <>
                  STRING_TO_INT(sDireccionBusTemp)) then
                        bErrorRespuesta := true;
                        iEstado := 0;
                        return;
                  end_if;

                  (* Quitamos los 3 primeros caracteres de la
                  respuesta, para así poder analizar la respuesta como
                  si no hubiera modo bus *)
                  sTempRecepcion := RIGHT(IN := sTempRecepcion, L :=
                  (LEN(sTempRecepcion)-3));

            (* Si la respuesta no tiene longitud 14 la respuesta es
            incorrecta *)
```

```
        else
                bErrorRespuesta := true;
                iEstado := 0;
                return;
        end_if;
end_if;

(* Eliminamos el carácter $02 que aparece *)
sTempRecepcion := RIGHT(IN := sTempRecepcion, L :=
(LEN(sTempRecepcion)-1));

(* SIN MODO BUS *)
(* Comprobamos si la comanda de respuesta ha sido válida *)
if (LEN(sTempRecepcion)=11) then
        (* Se guarda el valor del peso *)
        sTempRecepcion2 := MID(IN := sTempRecepcion, L := 8, P :=
        1);

        DiPesoTreal := STRING_TO_DINT(sTempRecepcion2);
        bPesoTreal := true;

        (* Se guarda las unidades de mesura *)
        sUnidadesMesura := '';
        sUnidadesMesura := MID(IN := sTempRecepcion, L := 1, P :=
        9);
        if (sUnidadesMesura='G') then
                sUnidadesMesura:='g';
        elsif (sUnidadesMesura='K') then
                sUnidadesMesura:='kg';
        elsif (sUnidadesMesura='T') then
                sUnidadesMesura:='t';
        elsif (sUnidadesMesura='L') then
                sUnidadesMesura:='lb';
        elsif (sUnidadesMesura=' ') then
                sUnidadesMesura:=' ';
        else
                bErrorRespuesta := true;
                iEstado := 0;
                return;
        end_if;

        (* Se guarda si tenemos peso neto o peso bruto *)
        sPesoNetoBruto := '';
        sPesoNetoBruto := MID(IN := sTempRecepcion, L := 1, P :=
        10);
        (* Peso neto *)
        if (sPesoNetoBruto='N') then
                bPesoBruto := false;
        (* Peso bruto *)
        elsif (sPesoNetoBruto='G') then
                bPesoBruto := true;
        else
                bErrorRespuesta := true;
                iEstado := 0;
                return;
        end_if;

        (* Se guarda el estado del peso *)
        sEstado := '';
        sEstado := MID(IN := sTempRecepcion, L := 1, P := 11);
```

```
            (* Se comprueba si el estado es uno de los estados posibles
            *)
            if (NOT(sEstado=' ' OR sEstado='M' OR sEstado='O' OR
            sEstado='I')) then
                  bErrorRespuesta := true;
            end_if;

      (* Respuesta no válida *)
      else
            bErrorRespuesta:= true;
      end_if;

      iEstado := 0;

END_CASE;

(*
if (bReinicioContador) then
      iLecturasPorMinuto := 0;
end_if;
*)

if (iPuerto = 1) then
      bCommunicationError := sys_blsComPort1CommunicationError;
      iPuertoTemp := SYS_COM1_PORT;
elsif (iPuerto = 2) then
      bCommunicationError := sys_blsComPort2CommunicationError;
      iPuertoTemp := SYS_COM2_PORT;
end_if;

(* Cada iTs comprobará el iEstado "Esperando" *)
TMuestreo(start := (iEstado = 0) and bActivacionPesada, SV := iTs, T
=> bMuestreo);

bReceptionDone := IsReceptionDone(Port := iPuertoTemp);
(* iTimeout para la recibida de datos *)
TNoRespuesta(start := (iEstado = 5) , SV := iTimeout, T =>
bNoRespuesta);

(* Petición pesada *)
if (bPeticionPesada OR bIniciarPesada) then
      bPeticioPesadaTemp := true;
      bIniciarPesada := false;
end_if;
TPesant(start := bPeticioPesadaTemp, SV := iDuracionPesada, T =>
bPesant);
```

C.3. Arduino

```
#include <SoftwareSerial.h>          // Libreria para utilizar cualquier par de pines digitales
                                     // como pines RX i TX de un puerto serie "virtual"
#include <stdlib.h>        // Para utilizar la función strtol
#include "Timer.h"         // Para utilizar el modulo Timer
#define RxPin 10
#define TxPin 11
SoftwareSerial mySerial(RxPin, TxPin);   // Creamos un puerto serie virtual
Timer t;

char entrada[40] = "%01#RP0000032767**\r";
char SegonaTrama[10] = "%01**&\r";
char TramaInici[5];
bool bFinalLectura, bEsperaIniciTrama, bIniciLectura, bSegonaTrama, bPrimerCaracter,
bPrimerCaracter2, bPrimerCaracter3, bPrimerCaracter4, bErrorRespuesta, bPrimer, bSegon,
bTercer;
long int number, suma;
uint8_t i,j;
char four_caracters[5], four_caracters2[5] ;
char * pEnd;

void setup() {
  // put your setup code here, to run once:
  pinMode(RxPin, INPUT);
  pinMode(TxPin, OUTPUT);
  mySerial.begin(9600);
  Serial.begin(9600);
```

```
  t.every(5000,enviar);
}

void enviar() {
  for (int count = 0; entrada[count] != '\0'; count++){
    mySerial.write(entrada[count]);
  }
}

void enviar_siguiente_trama() {
  for (int count = 0; SegonaTrama[count] != '\0'; count++){
    mySerial.write(SegonaTrama[count]);
    Serial.write('\n');
    Serial.write(SegonaTrama[count]);
  }
}

void loop() {
  // put your main code here, to run repeatedly:
  t.update();
  if (bEsperaIniciTrama)
    delay(50);
  // Si leemos algo enviado desde el PLC y no hemos llegado al final del programa
  while (mySerial.available()>0 and not bFinalLectura){
    // Guardamos cada caracter recibido
    char c = mySerial.read();
```

```
// Buscamos el inicio del programa. Cuando lo encontramos empezamos a sumar los
siguientes caracteres hasta encontrar el fin de programa

if (not bIniciLectura){

  if (four_caracters[0]=='4' and four_caracters[1]=='B' and four_caracters2[0]=='0' and
four_caracters2[1]=='C')

    bIniciLectura = true;

  else if (four_caracters[2]=='4' and four_caracters[3]=='B' and four_caracters2[2]=='0' and
four_caracters2[3]=='C')

    bIniciLectura = true;

  else if (four_caracters2[0]=='4' and four_caracters2[1]=='B' and four_caracters[0]=='0' and
four_caracters[1]=='C')

    bIniciLectura = true;

  else if (four_caracters2[2]=='4' and four_caracters2[3]=='B' and four_caracters[2]=='0' and
four_caracters[3]=='C')

    bIniciLectura = true;


  if (bIniciLectura){

    Serial.println("Trobat caracter inici! ");

   // Borra el contenido del array "four_caracters" desde la posición 0 hasta el final sizeof

   memset(four_caracters,0,sizeof(four_caracters));
  memset(four_caracters2,0,sizeof(four_caracters2));

    j = 0;

    i = 0;

    bSegonaTrama = false;

  }

}


if (bIniciLectura) {

    // Miramos si hemos encontrado el final del programa
```

```
      if (four_caracters[0]=='0' and four_caracters[1]=='3' and four_caracters2[0]=='0' and
four_caracters2[1]=='A')

        bFinalLectura = true;

      else if (four_caracters[2]=='0' and four_caracters[3]=='3' and four_caracters2[2]=='0' and
four_caracters2[3]=='A')

        bFinalLectura = true;

      else if (four_caracters2[0]=='0' and four_caracters2[1]=='3' and four_caracters[0]=='0' and
four_caracters[1]=='A')

        bFinalLectura = true;

      else if (four_caracters2[2]=='0' and four_caracters2[3]=='3' and four_caracters[2]=='0' and
four_caracters[3]=='A')

        bFinalLectura = true;


      if (bFinalLectura){

        Serial.println("Trobat caracter final");

        memset(four_caracters,0,sizeof(four_caracters));

        memset(four_caracters2,0,sizeof(four_caracters2));

      }

    }


    // Cuando tenemos 4 caracteres miramos si hay que calcular la suma

    if (j == 4){

      j = 0;

      if (bSegonaTrama) {

        if (bIniciLectura) {

          //Serial.println("Trobat caracter inici");

          // Convertimos los carácteres recividos en Hexadecimal

          number = strtol(four_caracters,&pEnd,16);

          suma = suma + number;
```

```
        Serial.print("Numero: ");
        Serial.println(number);
        Serial.println(suma);
      }
      memset(four_caracters,0,sizeof(four_caracters));
      bSegonaTrama = false;
    }
    else {
      if (bIniciLectura) {
        //Serial.println("Trobat caracter inici");
        // Convertimos los carácteres recividos en Hexadecimal
        number = strtol(four_caracters2,&pEnd,16);
        suma = suma + number;
        Serial.print("Numero: ");
        Serial.println(number);
      }
      memset(four_caracters2,0,sizeof(four_caracters2));
      bSegonaTrama = true;
    }
  }
  // Miramos si se ha llegado al final de la trama.
  if (c == '&') {
    bEsperaIniciTrama = true;
    // Comprobamos en que trama estamos y borramos los 2 últimos caracteres añadidos, que son el checksum, y no hay que contarlo para hacer la suma de comprobación
    if (bSegonaTrama) {
      // Si acabamos de hacer la suma
```

```
if (four_caracters2[0]=='0' and four_caracters2[1]=='0' and four_caracters2[2]=='0' and four_caracters2[3]=='0'){
  four_caracters[2]='0';
  four_caracters[3]='0';
  bSegonaTrama = false;
  j=2;
}
// En caso contrario, hay el checksum en los dos primeros caracteres del array y lo borramos
else {
  four_caracters2[0]='0';
  four_caracters2[1]='0';
  j=0;
}
}
else {
if (four_caracters[0]=='0' and four_caracters[1]=='0' and four_caracters[2]=='0' and four_caracters[3]=='0'){
  four_caracters2[2]='0';
  four_caracters2[3]='0';
  bSegonaTrama = true;
  j=2;
}
else {
  four_caracters[0]='0';
  four_caracters[1]='0';
  j=0;
}
}
```

```
    // Se debe enviar una trama para que envie las otras tramas de respuesta
    enviar_siguiente_trama();
  }


  else {
   if (not bEsperaIniciTrama){
     if (bSegonaTrama) {
       Serial.write(c);
       four_caracters2[j++] = c;
     }
     else {
       Serial.write(c);
       four_caracters[j++] = c;
     }
   }
   else {
     // Guardamos el inicio de la trama para comprobar que es correcto. Cuando es así se
puede volver a sumar los caracteres
     TramaInici[i++] = c;
     for (int t = 0; t < sizeof(TramaInici);  ++t) {
       if (TramaInici[t]=='%'){
         bPrimer = true;
       }
       if (TramaInici[t]=='0'){
         bSegon = true;
       }
       if (TramaInici[t]=='1'){
```

```
          bTercer = true;
        }
      }

      if (bPrimer and bSegon and bTercer) {
        bEsperaIniciTrama = false;
        memset(TramaInici,0,sizeof(TramaInici));
        i=0;
        bPrimer = false;
        bSegon = false;
        bTercer = false;
      }
      else if (strlen(TramaInici) > 4) {
        memset(TramaInici,0,sizeof(TramaInici));
        i=0;
        bErrorRespuesta = true;
        Serial.print("Trama no valida");
        bFinalLectura = true;
      }
    }
  }
}
if (bFinalLectura){
    Serial.println("suma: ");
    Serial.print(suma);
  }
bFinalLectura = false;
```

```
  bIniciLectura = false;

  suma = 0;

  memset(four_caracters,0,sizeof(four_caracters));

  j = 0;

  i = 0;

  bSegonaTrama = false;

  bSegonaTrama = false;

}
```