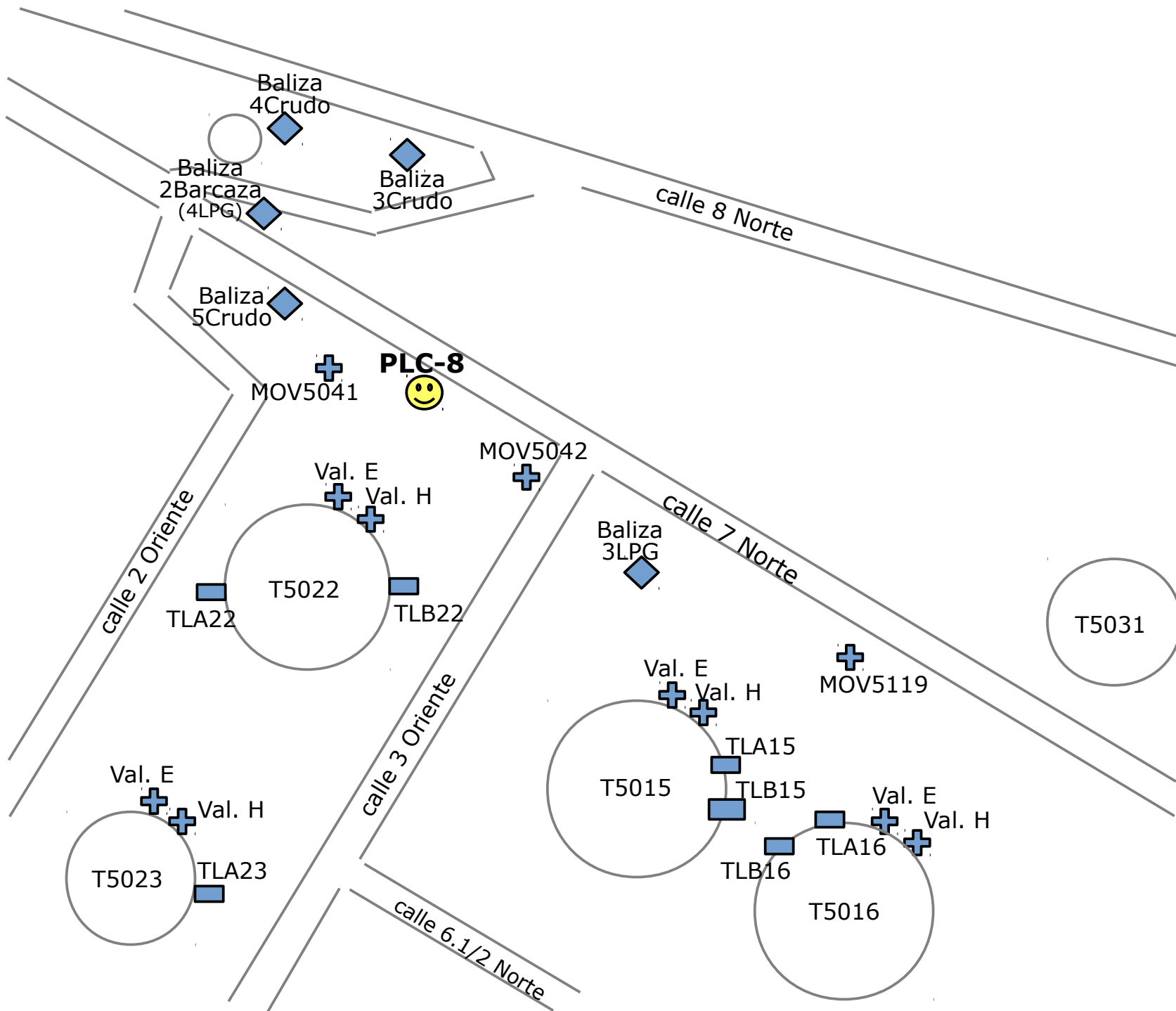


PLC-8

con M340



Mayo 2017



Los PLC **Schneider M340** son el reemplazo natural, de los ya discontinuados **Modicon 984**, que antes tenia. Los M340 son modulares, en un formato muy similar, a los antiguos.

Hardware:

Existen racks de distintos largos, para distintas cantidades de módulos. Además, un rack se puede extender conectándole mas racks en cascada.

Cada rack debe tener en su lado izquierdo un modulo fuente de alimentación, para alimentar los módulos que contiene (power supply BMX-CPS320).

El primer rack debe tener en su primera ranura un modulo procesador, y en las siguientes ranuras puede tener modulos de cualquier tipo.

Los racks que son extensiones de un rack primario, no necesitan modulo CPU, y pueden tener módulos de cualquier tipo en cualquier posición.

Existen muchos tipos distintos de módulos, por ejemplo:

Modulo de 16 entradas discretas, de 24 volt continuos (M340 BMX-DDI1602).

Modulo de 8 entradas analogas, de 4 a 20 mili Amperes (M340-BMX-AMO0410).

Modulo Procesador CPU modelo 2020 (M340-BMX-CPU342020).

Modulo de 8 salidas discretas, de rele (M340-BMX-DRA0805).

Modulo de comunicación serial aislado (M340-BMX-NOM0200)

Etc...

Aunque la misma CPU, ya tiene puertos de comunicación, estos NO deben usarse para comunicación remota (con DCS), solo son para comunicación local (en el mismo gabinete), por que los puertos de comunicación en la CPU no tienen aislacion galvanica. Por eso, se agrego un modulo de comunicación serial aislado, en cada plc, para comunicarlo con DCS.

Software:

Los Schneider M340 se programan con el software **Unity Pro** version 8.0, o superior.

Se pueden ordenar los módulos sobre los rack, como se quiera, pero la posición física elegida, debe corresponder con las direcciones lógicas, que se usen en el programa cargado en la CPU.

Cada variable en el programa puede ser designada por una "dirección lógica" O por un "tag". Las variables que tengan dirección lógica y además, también un tag, se puede hacer referencia a la misma variable, por su tag, o por su dirección lógica.

Las variables que solo tengan dirección lógica, pueden ser una dirección física (canal en algún modulo de entrada o salida), o dirección Modbus, que puede ser leída o escrita por algún puerto de comunicación con otro equipo.

Las variables que solo tengan un tag, y no tienen dirección lógica, no pueden ser asociadas a una entrada o salida física, ni leídas por comunicacion, por lo que solo son variables internas del mismo programa.

Todas las numeraciones empiezan desde cero, por ejemplo un modulo de 8 canales, tiene desde el canal 0, hasta el canal 7.

Nomenclatura de las direcciones lógicas:

%I.r.m.c = Entrada discreta en rack **r**, modulo **m**, canal **c**

ejemplo %I.0.4.2 es la tercera entrada discreta (canal 2) en modulo 4 del rack 0.

%Q.r.m.c = Salida discreta en rack **r**, modulo **m**, canal **c**

ejemplo %Q.1.3.7 es la salida discreta, canal 7, en modulo 3 del rack 1 (segundo rack).

%IW.r.m.c = Entrada análoga en rack **r**, modulo **m**, canal **c**

ejemplo %IW.0.5.1 es la segunda entrada análoga (canal 1) en modulo 5 del rack 0.

%QW.r.m.c = Salida análoga en rack **r**, modulo **m**, canal **c**

ejemplo %QW.0.7.1 es la segunda salida análoga (canal 1) en modulo 7 del rack 0.

%M.x = Bit Memoria numero x, en Modbus se les conoce como "Coil"

ejemplo %M.87 es la memoria discreta 87 (en Modbus es "coil" 0:0088).

%MW.x = Word Memoria (entero 16bit), en Modbus equivalen a los "Holding Register"

ejemplo %MW.2 es Word de memoria numero 2 (en Modbus es 4:0003).

Hay otros tipos de datos: tipo doble word (entero 32bit), tipo Date, tipo Real, tipo String, etc. Pero esos tipos no pueden tener una dirección Modbus o una dirección física, variables de esos tipos deben ser nombradas por un tag.

La programación de los Schneider M340 con el software **Unity Pro** puede hacerse con cualquiera de los lenguajes de programación estandarizados según las normas IEC. Los mas conocidos de estos son "Ladder", "Bloques de funciones", y "Texto Estructurado".

Permite programar por tareas, y estas dividir las en "secciones" para lograr un resultado mas ordenado. Cada sección puede tener módulos en distintos lenguajes. La libertad de

poder combinar distintos lenguajes, permite usar el lenguaje mas adecuado para cada modulo del programa

Se pueden crear funciones derivadas (sub-rutinas) en cual quiera de estos lenguajes. Y estas sub-rutinas pueden ser usadas recursiva mente en las "secciones" del programa, para no tener que repetir muchas veces el mismo código, y que todas las partes iguales funcionen igual.

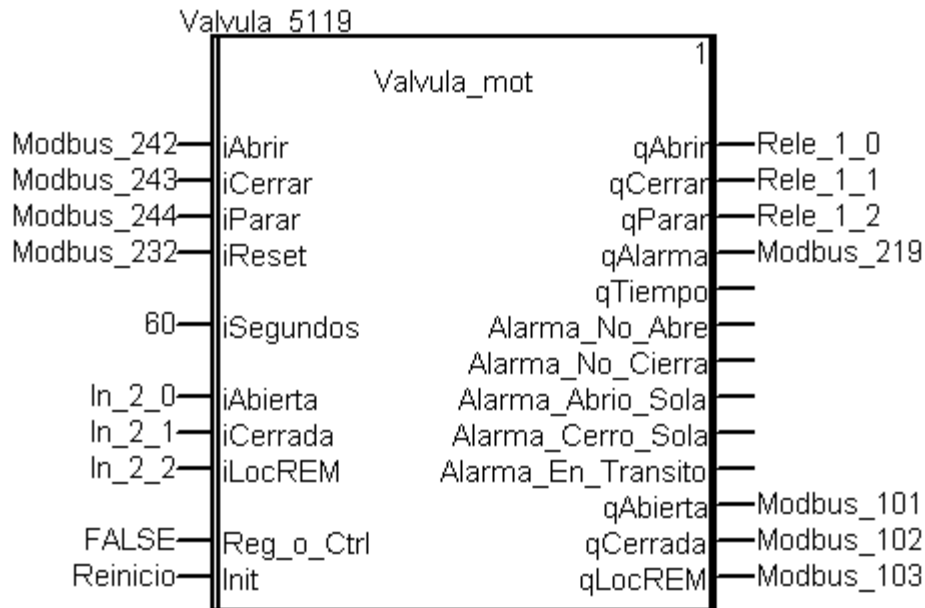
Para la programación de los plc remotos en Terminal Quintero, se definieron algunas sub-rutinas (funciones derivadas) con lógicas comunes para todos los equipos del mismo tipo. De manera de lograr que todas las válvulas compartan la misma lógica, y todas los motores compartan la misma lógica. Solo se dejan sin usan, las partes de la lógica que no apliquen, según cada caso en particular.

Lógica para Válvulas:

Todas las válvulas motorizadas usan esta misma lógica.

En la siguiente imagen, un ejemplo de aplicación de la función "Valvula_mot".

Al lado izquierdo del bloque, las señales conectadas a las entradas de la función. A la derecha del bloque, las señales conectadas a las salidas de la función. Si la función tiene una entrada desconectada, asume un valor por defecto, para esa variable. Si la función tiene una salida desconectada, es por que no se esta usando esa parte.



El bloque de lógica (función) "**Valvula_mot**" revive las las siguientes señales, (tiene las siguientes **entradas**):

iAbrir = Orden Abrir desde Sala de Control, un pulso que llega por comunicación.

iCerrar = Orden Cerrar desde Sala de Control, un pulso que llega por comunicación.

iParar = Orden Parar desde Sala de Control, un pulso que llega por comunicación.

iAbierta = Estado desde terreno, cuando válvula esta totalmente abierta.

iCerrada = Estado desde terreno, cuando válvula esta totalmente cerrada.

iLocREM = Estado desde terreno, conectado indica esta en Remoto.

iReset = Limpiar alarma desde Sala de Control, un pulso que llega por comunicación.

iSegundos = Segundos a esperar que abra o cierre (es un valor fijo). Debe ajustarse según el tiempo máximo que puede demorarse cada válvula en particular.

Reg_o_Ctrl = Deshabilita alarma por mucho tiempo en transición (ni abierta ni cerrada). Para válvulas reguladoras o de control dejar en True. Para válvulas de corte, dejar desconectado (o en False), para que este habilitada la alarma por mucho tiempo en transición.

Init = Deshabilita y limpia todas las alarmas. Esta conectado a un pulso que genera el mismo PLC cuando parte (solo una vez cuando parte), para que no se activen alarmas de las válvulas, al reiniciar el PLC.

El bloque de lógica (función) "**Valvula_mot**" entrega las las siguientes señales, (tiene las siguientes **salidas**):

qAbrir = Comando Abrir hacia terreno, un pulso cableado hasta el control de la válvula.

qCerrar = Comando Cerrar hacia terreno un pulso cableado hasta el control de la válvula.

qParar = Comando Parar hacia terreno un pulso cableado hasta el control de la válvula.

qAlarma = Resumen alarmas hacia Sala de Control, estado se enviar por comunicación.

qAbierta = indicación estado hacia Sala de Control, estado se enviar por comunicación.

qCerrada = indicación estado hacia Sala de Control, estado se enviar por comunicación.

iLocREM = indicación estado hacia Sala de Control, estado se enviar por comunicación.

qTiempo = Muestra el tiempo en segundos que esta usando, como tiempo máximo para completar su recorrido. Es lo mismo que la entrada "iSegundos", si esta entrada tiene un valor valido.

Alarma_ ... = Indica cual de los motivos, activa el resumen de alarma. No están conectadas. Al ver en linea (con el computador conectado al PLC), la lógica funcionando, permite ver cual de ellas esta activada y cual no.

Lógica para Abrir, dentro de la función "Valvula_mot"

(* Solo acepta como **orden Abrir**, el cambio de 0 a 1 de la entrada iAbrir.*)

Trig_Abrir (CLK := iAbrir, Q => bAbrir);

(*Timer Off Delay *)

Tm_pulso_abrir (IN := bAbrir,

PT := T#1S,

Q => PulsoAbrir);

(* Activa rele Abrir, si llego orden de abrir, y no esta la orden parar, hasta que este abierta, o alarme por que no abre.*)

IF (PulsoAbrir OR qAbrir) AND NOT iAbierta AND NOT iParar AND NOT Alarm1 THEN

qAbrir := TRUE;

ELSE

qAbrir := FALSE;

END_IF;

Lógica para Cerrar, dentro de la función "Valvula_mot"

(* Solo acepta como orden Cerrar, el cambio de 0 a 1 de la entrada iCerrar.*)

Trig_Cerrar (CLK := iCerrar, Q => bCerrar);

(*Timer Off Delay *)

Tm_pulso_cerrar (IN := bCerrar,

PT := T#1S,

Q => PulsoCerrar);

(* Activa rele Cerrar, si llega orden de cerrar, y no esta la orden parar, hasta que este cerrada, o alarma por que no cierra.*)

IF (PulsoCerrar OR qCerrar) AND NOT iCerrada AND NOT iParar AND NOT Alarm2 THEN

qCerrar := TRUE;

ELSE

qCerrar := FALSE;

END_IF;

Lógica de Parar, dentro de la función "Valvula_mot"

(* Activa rele Parar, siempre que este orden de Parar. Tambien cancela las ordenes de abrir o cerrar.*)

qParar := iParar;

Lógica de Alarmas, dentro de la función "Valvula_mot"

(* Validar el valor de segundos ingresado, como tiempo maximo que espera el cambio de estado, luego de enviar un comando hacia terreno. El tiempo debe estar entre 3 y 600 segundos. *)

IF iSegundos < 10 THEN mSeg_Llegar:=10000; mSeg_Iniciar:=2500; END_IF;

IF iSegundos > 600 THEN mSeg_Llegar:=600000; mSeg_Iniciar:=150000; END_IF;

IF (iSegundos >= 10) AND (iSegundos <= 600) THEN

mSeg_Llegar := (iSegundos * 1000);

mSeg_Iniciar := (iSegundos * 250);

END_IF;

Seg_Llegar := DINT_TO_TIME (mSeg_Llegar); (* tiempo para llegar al otro estado *)

Seg_Iniciar := DINT_TO_TIME (mSeg_Iniciar); (* para dejar el estado actual *)

qTiempo := Seg_Llegar;

(* _____ *)

(* **Alarm1** = No llega hasta abierta, al enviar el comando abrir. Esta alarma se limpia con nueva orden de abrir o con orden de Reset *)

(*Timer On Delay *)

Tm_Abr (IN := qAbrir,

PT := Seg_Llegar,

Q => Tm_abr_out);

IF (Tm_abr_out OR Alarm1) AND NOT PulsoAbrir AND NOT iReset AND NOT Init THEN

Alarm1 := TRUE;

ELSE

Alarm1 := FALSE;

END_IF;

(* _____ *)

(* **Alarm2** = no llega hasta cerrada, al enviar el comando cerrar. Esta alarma se limpia con nueva orden de cerrar o con orden de Reset *)

(*Timer On Delay *)

```

Tm_Cerr (IN := qCerrar,
    PT := Seg_Llegar,
    Q => Tm_cerr_out);
IF (Tm_cerr_out OR Alarm2) AND NOT PulsoCerrar AND
NOT iReset AND NOT Init THEN
    Alarm2 := TRUE;
ELSE
    Alarm2 := FALSE;
END_IF;
(* _____ *)
(* Alarm3 = continua cerrada, al enviar el comando abrir. Esta alarma se limpia con nueva orden
de abrir o con orden de Reset *)
(*Timer On Delay *)
Tm_nAbr (IN := (qAbrir AND iCerrada),
    PT := Seg_Iniciar,
    Q => Tm_nAbr_out);
IF (Tm_nAbr_out OR Alarm3) AND NOT PulsoAbrir AND
NOT iReset AND NOT Init THEN
    Alarm3 := TRUE;
ELSE
    Alarm3 := FALSE;
END_IF;
(* _____ *)
(* Alarm4 = continua abierta, al enviar el comando cerrar. Esta alarma se limpia con nueva
orden de cerrar o con orden de Reset *)
(*Timer On Delay *)
Tm_nCerr (IN := (qCerrar AND iAbierta),
    PT := Seg_Iniciar,
    Q => Tm_nCerr_out);
IF (Tm_nCerr_out OR Alarm4) AND NOT PulsoCerrar AND
NOT iReset AND NOT Init THEN
    Alarm4 := TRUE;
ELSE
    Alarm4 := FALSE;
END_IF;
(* _____ *)

```

(* **Alarm5** = Se detecto abierta, sin haber enviado el comando abrir. Esta alarma se limpia con nueva orden de cerrar o con orden de Reset. Si la valvula es reguladora o de control, esta alarma no activa el resumen de alarmas *)

Trig_Abrir (CLK := iAbierta, Q => bAbrir);

IF ((bAbrir AND NOT qAbrir) OR Alarm5) AND NOT qCerrar AND NOT iReset AND NOT Init THEN

Alarm5 := True;

ELSE

Alarm5 := False;

END_IF;

(* _____ *)

(* **Alarm6** = valvula se cerro, sin haber enviado el comando cerrar. Esta alarma se limpia con nueva orden de abrir o con orden de Reset. Si la valvula es reguladora o de control, esta alarma no activa el resumen de alarmas *)

Trig_Cerro (CLK := iCerrada, Q => bCerro);

IF ((bCerro AND NOT qCerrar) OR Alarm6) AND NOT qAbrir AND NOT iReset AND NOT Init THEN

Alarm6 := True;

ELSE

Alarm6 := False;

END_IF;

(* _____ *)

(* **Alarm7** = mucho tiempo en transito, no abierta ni cerrada. Esta alarma esta deshabilitada si es valvula reguladora o de control.*)

IF NOT iAbierta AND NOT iCerrada THEN

enTransito := True;

ELSE

enTransito := False;

END_IF;

IF qAbrir OR qCerrar THEN

Seg_trans := Seg_Llegar + t#3s;

ELSE

Seg_trans := t#3s;

END_IF;

(*Timer On Delay *)

Tm_Trans (IN := (enTransito AND NOT Reg_o_Ctrl),

PT := Seg_trans,

```

    Q => Alarm7 );

(* _____ *)
(* Alarm8 = Indica abierta y cerrada al mismo tiempo. *)
(*Timer On Delay *)
Tm_error_z (IN := (iAbierta AND iCerrada),
    PT := t#3s,
    Q => Alarm8 );

(* _____ *)
(* qALARMA = resumen alarmas se activa con cualquier alarma *)
IF Alarm1 OR Alarm2 OR
Alarm3 OR Alarm4 OR
(Alarm5 AND NOT Reg_o_Ctrl) OR
(Alarm6 AND NOT Reg_o_Ctrl) OR
Alarm7 OR Alarm8 THEN
    qAlarma := TRUE;
ELSE
    qAlarma := FALSE;
END_IF;
(* _____ *)

```

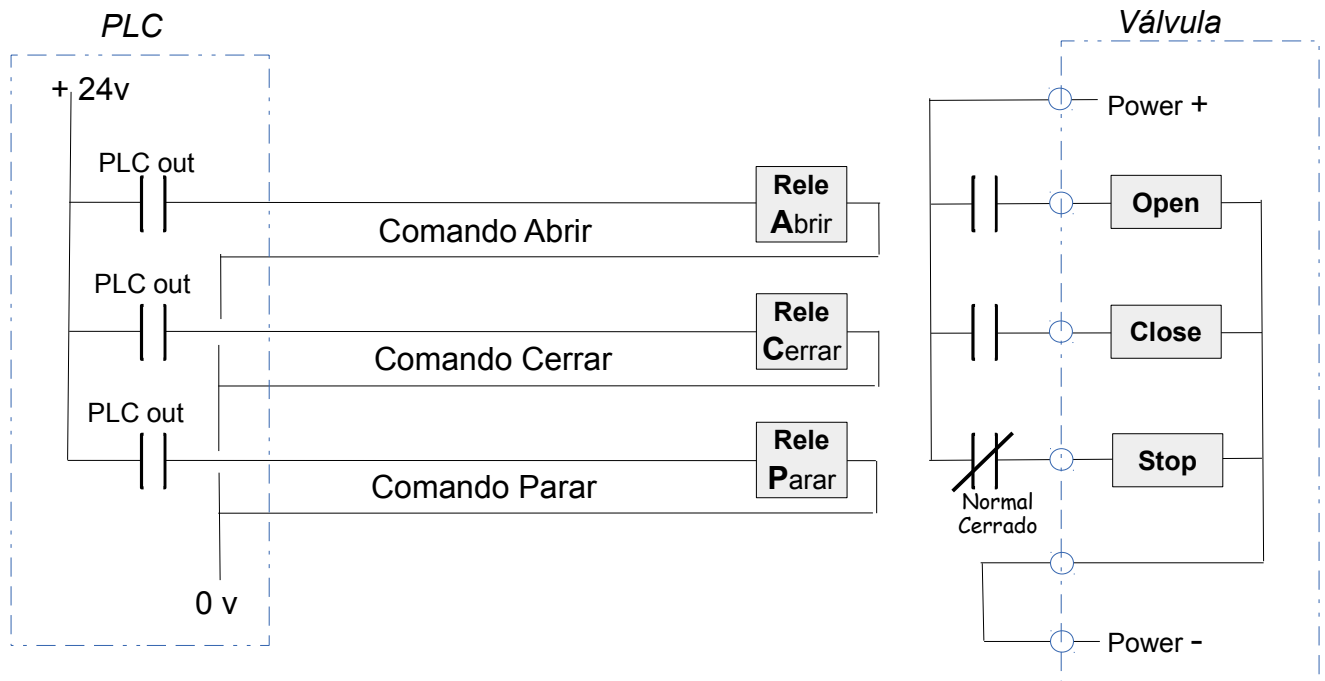
Lógica de indicación de estados, dentro de la función "Valvula_mot"

```

(* Indicacion de estados *)
qAbierta := iAbierta;
qCerrada := iCerrada;
qLocREM := iLocREM;
Alarma_No_Abre := Alarm1 OR Alarm3;
Alarma_No_Cierra := Alarm2 OR Alarm4;
Alarma_Abrio_Sola := Alarm5;
Alarma_Cerro_Sola := Alarm6;
Alarma_En_Transito := Alarm7 OR Alarm8;

```

Conexión Típica de Válvulas con PLC

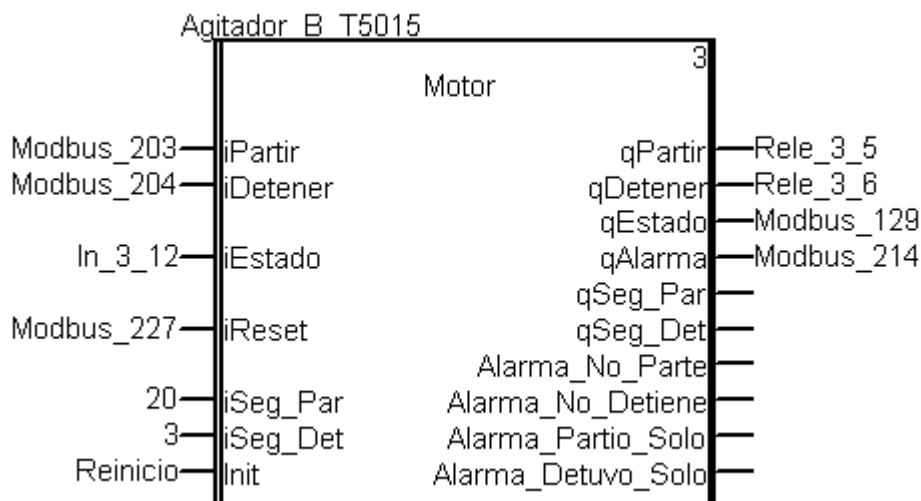


Lógica para Motores:

Todos los motores usan esta misma lógica, sean de bombas o de agitadores.

En la siguiente imagen, un ejemplo de aplicación de la función "Motor".

Al lado izquierdo del bloque, las señales conectadas a las entradas de la función. A la derecha del bloque, las señales conectadas a las salidas de la función. Si la función tiene una entrada desconectada, asume un valor por defecto, para esa variable. Si la función tiene una salida desconectada, es por que no se esta usando esa parte.



El bloque de lógica (función) "**Motor**" revive las las siguientes señales, (tiene las siguientes **entradas**):

iPartir = Orden Partir desde Sala de Control, un pulso que llega por comunicación.

iDetener = Orden Detener desde Sala de Control, un pulso que llega por comunicación.

iEstado = Estado desde terreno, cableado desde circuito eléctrico del motor.

iReset = Limpiar alarma desde Sala de Control, un pulso que llega por comunicación.

iSeg_Run = Segundos a esperar que parta (es un valor fijo). Debe ajustarse según el tiempo máximo que puede demorar en partir, cada motor en particular.

iSeg_Stop = Segundos a esperar que se detiene (es un valor fijo). Debe ajustarse según el tiempo máximo que puede demorar en detenerse, cada motor en particular.

Init = Deshabilita y limpia todas las alarmas. Esta conectado a un pulso que genera el mismo PLC cuando parte (solo una vez cuando parte), para que no se activen alarmas de los motores, al reiniciar el PLC.

El bloque de lógica (función) "**Motor**" entrega las las siguientes señales, (tiene las siguientes **salidas**):

qPartir = Comando Partir hacia terreno, cableado hasta el circuito eléctrico del motor.

qDetener = Comando Detener hacia terreno, cableado hasta el cto eléctrico del motor.

qAlarma = Resumen alarma hacia Sala de Control, estado se enviar por comunicación.

qEstado = Indicación estado hacia Sala de Control, estado se enviar por comunicación.

qSeg_Par = Muestra el tiempo en segundos que esta usando, como tiempo máximo para revivir la confirmación de que partió. Es lo mismo que la entrada "iSeg_Run", si esta entrada tiene un valor valido.

qSeg_Det = Muestra el tiempo en segundos que esta usando, como tiempo máximo para revivir la confirmación de que se detuvo. Es lo mismo que la entrada "iSeg_Stop", si esta entrada tiene un valor valido.

Alarma_ ... = Indica cual de los motivos, activa el resumen de alarma. No están conectadas. Al ver en linea (con el computador conectado al PLC), la lógica funcionando, permite ver cual de ellas esta activada y cual no.

Lógica de Partir, dentro de la función "Motor"

(* Solo acepta como **orden Partir**, el cambio de 0 a 1 de la entrada iPartir.*)

Trig_Partir (CLK := iPartir, Q => bPartir);

Tm_pulso_part (IN := bPartir, PT := T#1S, Q => PulsoPartir);

(* Activa rele Partir si llega orden de Partir, hasta que motor este andando, o alarma por que no parte.*)

IF (PulsoPartir OR qPartir) AND NOT iEstado AND NOT Alarm1 THEN

qPartir := TRUE;

ELSE

qPartir := FALSE;

END_IF;

Lógica de Parar, dentro de la función "Motor"

(* Solo acepta como orden Detener, el cambio de 0 a 1 de la entrada iDetener.*)

Trig_Detener (CLK := iDetener, Q => bDetener);

(* Activa rele Detener, durante 2 segundos, si a llegado orden de Detener *)

Tm_pulso_det (IN := bDetener, PT := T#2S, Q => PulsoDetener);

IF (PulsoDetener OR espDetener) AND iEstado AND NOT Alarm2 THEN

espDetener := TRUE;

ELSE

espDetener := FALSE;

END_IF;

IF PulsoDetener OR espDetener THEN

qDetener := TRUE;

ELSE

qDetener := FALSE;

END_IF;

Lógica de Alarmas, dentro de la función "Motor"

(* Validar el valor de segundos ingresado, como tiempo maximo que espera el cambio de estado, luego de enviar un comando hacia terreno. El tiempo para partir debe ser entre 3 y 120 segundos. El tiempo para detener debe ser entre 3 y 120 segundos. *)

IF iSeg_Par < 3 THEN Seg_Run := 3000; END_IF;

IF iSeg_Par > 120 THEN Seg_Run := 120000; END_IF;

IF (iSeg_Par >= 3) AND (iSeg_Par <= 120) THEN Seg_Run := (iSeg_Par * 1000);
END_IF;

Seg_R := DINT_TO_TIME (Seg_Run);

qSeg_Par := Seg_R;

IF iSeg_Det < 3 THEN Seg_Stop := 3000; END_IF;

IF iSeg_Det > 120 THEN Seg_Stop := 120000; END_IF;

IF (iSeg_Det >= 3) AND (iSeg_Det <= 120) THEN Seg_Stop := (iSeg_Det * 1000);
END_IF;

Seg_S := DINT_TO_TIME (Seg_Stop);

qSeg_Det := Seg_S;


```

(*)
(* Alarm1 = motor no parte, al enviar el comando. Esta alarma se limpia con nueva orden de
partir o con orden de Reset *)
Tm_Part (IN := qPartir, PT := Seg_R, Q => Tm_part_out);
IF (Tm_part_out OR Alarm1) AND NOT PulsoPartir AND
NOT iReset AND NOT Init THEN
    Alarm1 := TRUE;
ELSE
    Alarm1 := FALSE;
END_IF;
(*)
(* Alarm2 = motor no se detiene, al enviar el comando. Esta alarma se limpia con nueva orden
de detener o con orden de Reset*)
Tm_Det (IN := qDetener, PT := Seg_S, Q => Tm_det_out);
IF (Tm_det_out OR Alarm2) AND NOT PulsoDetener AND
NOT iReset AND NOT Init THEN
    Alarm2 := TRUE;
ELSE
    Alarm2 := FALSE;
END_IF;
(*)
(* Alarm3 = alarma motor partio, sin haber enviado el comando. Esta alarma se limpia con orden
de detener o con orden de Reset*)
rTrigEstado (CLK := iEstado, Q => bPartio);
IF ((bPartio AND NOT qPartir) OR Alarm3) AND NOT PulsoDetener AND
NOT iReset AND NOT Init THEN
    Alarm3 := TRUE;
ELSE
    Alarm3 := FALSE;
END_IF;
(*)
(* Alarm4 = alarma motor se detuvo, sin haber enviado el comando. Esta alarma se limpia con
orden de partir o con orden de Reset*)
fTrigEstado (CLK := (NOT iEstado), Q => bDetuvo);
IF ((bDetuvo AND NOT qDetener) OR Alarm4) AND NOT PulsoPartir AND
NOT iReset AND NOT Init THEN

```

```

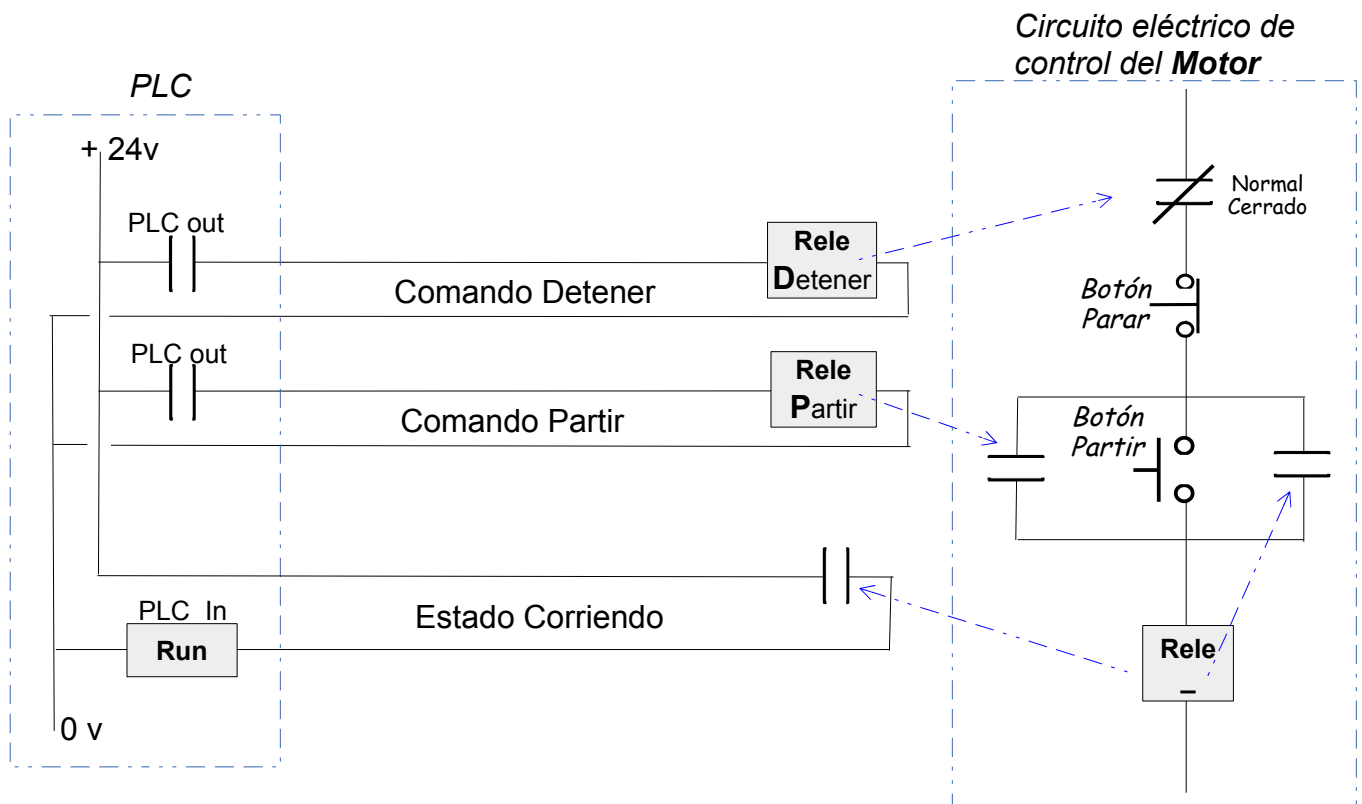
    Alarm4 := TRUE;
ELSE
    Alarm4 := FALSE;
END_IF;

(* _____ *)
(* qAlarma = resumen alarmas se activa con cualquier alarma *)
IF Alarm1 OR Alarm2 OR Alarm3 OR Alarm4 THEN
    qAlarma := TRUE;
ELSE
    qAlarma := FALSE;
END_IF;
Alarma_No_Parte := Alarm1;
Alarma_No_Detiene := Alarm2;
Alarma_Partio_Solo := Alarm3;
Alarma_Detuvo_Solo := Alarm4;

(* _____ *)
(* copia iEstado en qEstado *)
IF iEstado THEN
    qEstado := TRUE;
ELSE
    qEstado := FALSE;
END_IF;

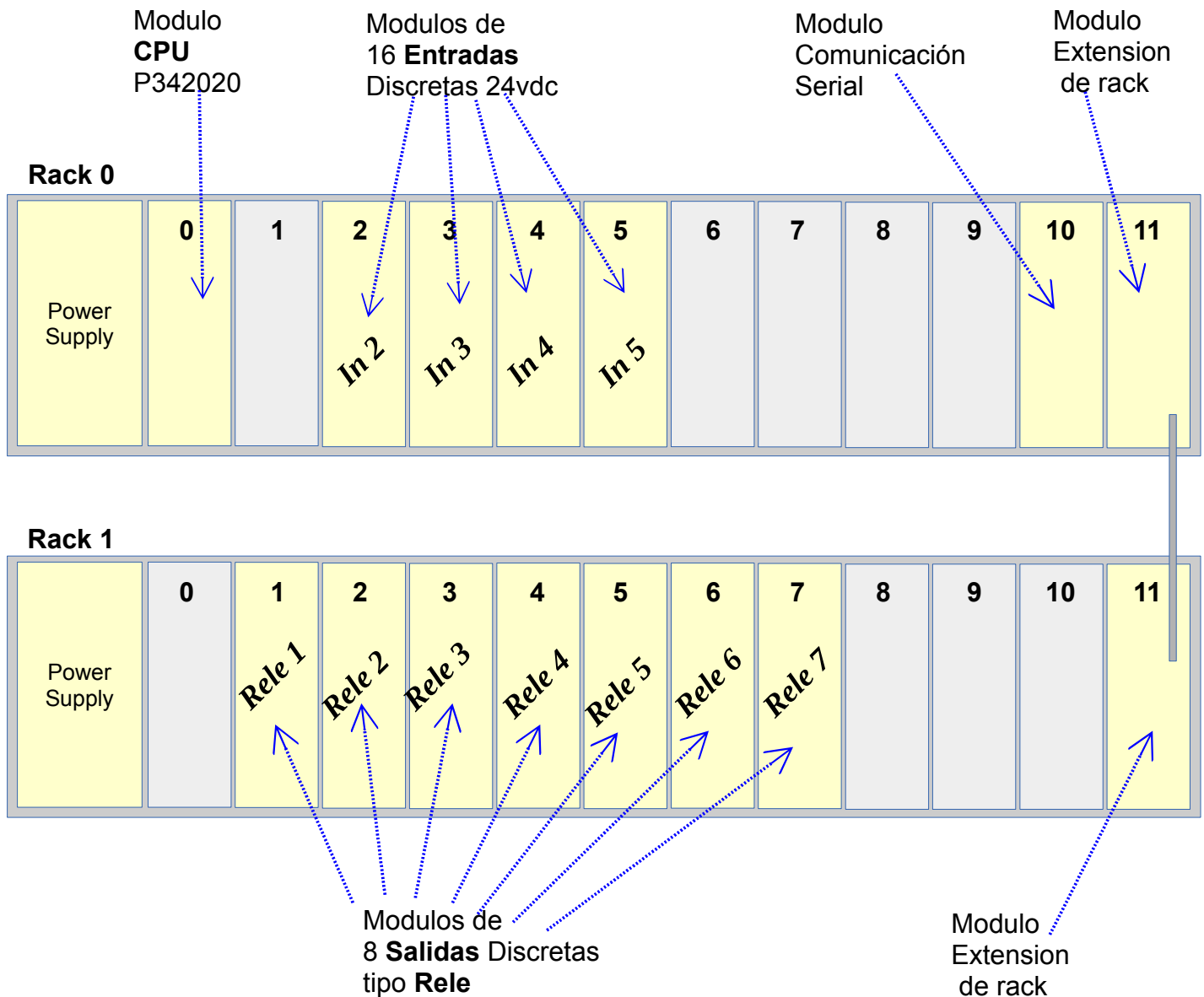
```

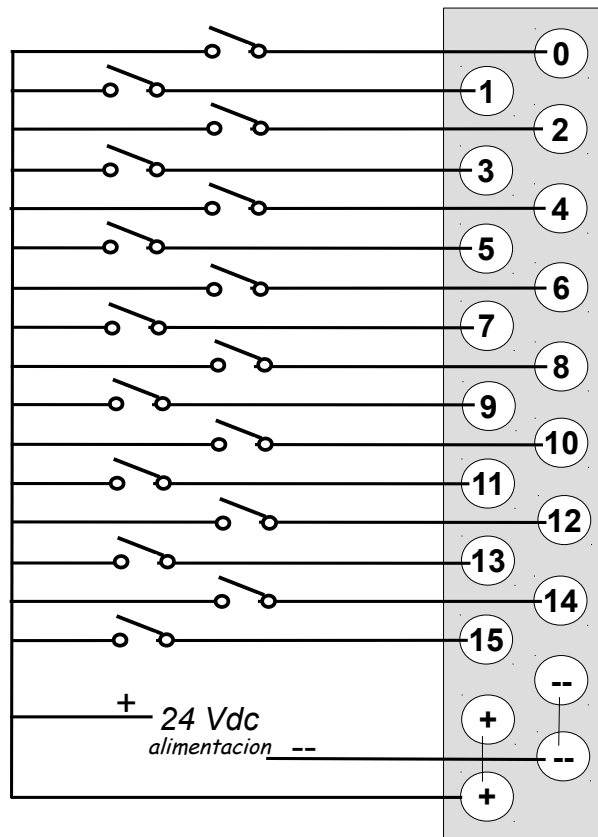
Conexión Típica de Motor con PLC



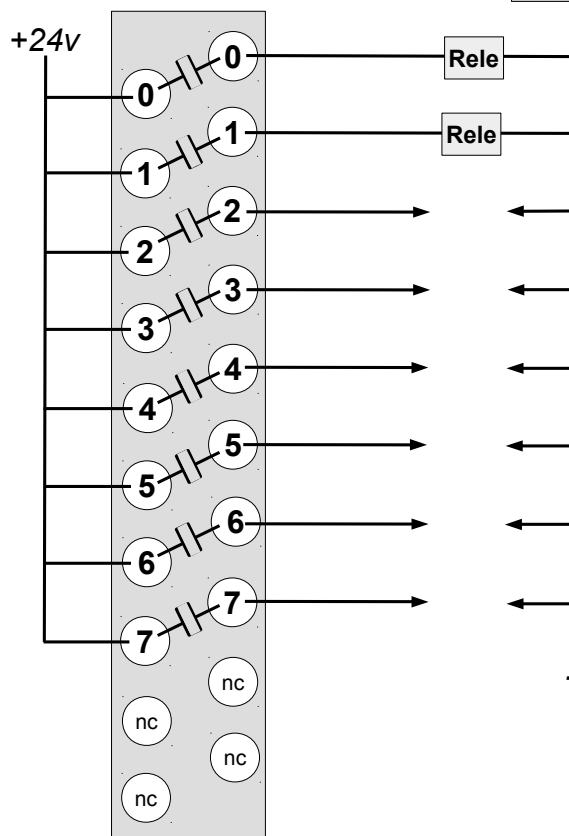
Conexionado entradas y salidas:

Distribución de módulos en el PLC-8





Cada
modulo de 16 entradas
discretas, de 24 volt
continuos.
M340 BMX-**DDI**1602



Cada
modulo de 8
salidas discretas,
de reles
M340-BMX-**DRA**0805

Rack 0, Modulo 2

Entradas Discretas desde %I.0.2.0 hasta %I.0.2.15

0	MOV-5119 Abierta
1	MOV-5119 Cerrada
2	MOV-5119 Loc/Rem
3	Val E de T5022 Abierta
4	Val E de T5022 Cerrada
5	Val H de T5022 Abierta
6	Val H de T5022 Cerrada
7	Val E de T5015 Abierta
8	Val E de T5015 Cerrada
9	Val H de T5015 Abierta
10	Val H de T5015 Cerrada
11	Val E de T5016 Abierta
12	Val E de T5016 Cerrada
13	Val H de T5016 Abierta
14	Val H de T5016 Cerrada
15	Val E de T5022 Loc/Rem

Rack 0, Modulo 3

Entradas Discretas desde %I.0.3.0 hasta %I.0.3.15

0	Val H de T5022 Loc/Rem
1	Val E de T5015 Loc/Rem (no conectado)
2	Val H de T5015 Loc/Rem (no conectado)
3	Val E de T5016 Loc/Rem (no conectado)
4	Val H de T5016 Loc/Rem (no conectado)
5	LAH-T5022 (Alarma)
6	LAH-T5016 (Alarma)
7	LAH-T5015 (Alarma)
8	LAH-T5007 (Alarma)
9	Agit. A de T5022 Run
10	Agit. B de T5022 run

11	Agit. A de T5015 Run
12	Agit. B de T5015 Run
13	Agit. A de T5016 Run
14	Agit. B de T5016 Run
15	Alarma Energia PLC (alarma 220v, cargador baterías, fuentes 24v)

Rack 0, Modulo 4

Entradas Discretas desde %I.0.4.0 hasta %I.0.4.15

0	Val. 5023-1 Abierta
1	Val. 5023-1 Cerrada
2	Val. 5023-1 Loc/Rem
3	Val. 5023-2 Abierta
4	Val. 5023-2 Cerrada
5	Val. 5023-2 Loc/Rem
6	LSHH-5023-1
7	Estado Agit. A de T5023
8	ZL-5603C (Estado Baliza 3Crudo)
9	EC-5603C (Alarma Baliza 3 crudo)
10	ZL-5604C (Estado Baliza 4crudo)
11	EC-5604C (Alarma Baliza 4Crudo)
12	ZL-5605C (Estado Baliza 5Crudo)
13	EC-5605C (Alarma Baliza 5Crudo)
14	ZL-5603LPG (Estado Baliza 3LPG)
15	EC-5603LPG (Alarma Baliza 3LPG)

Rack 0, Modulo 5

Entradas Discretas desde %I.0.5.0 hasta %I.0.5.15

0	ZL-5604LPG (Estado Baliza 2Barcaza)
1	EC-5604LPG (Alarma Baliza 2Barcaza)
2	Val. 5041 Loc/Rem
3	Val. 5041 Abierta

4	Val. 5041 Cerrada
5	Val. 5042 Loc/Rem
6	Val. 5042 Abierta
7	Val. 5042 Cerrada
8	libre
9	libre
10	libre
11	libre
12	libre
13	libre
14	libre
15	libre

Rack 1, Modulo 1

Salidas Discretas desde %Q.1.1.0 hasta %Q.1.1.7

0	Abrir MOV-5119
1	Cerrar MOV-5119
2	Parar MOV-5119
3	Abrir val E de T5022
4	Cerrar val E de T5022
5	Abrir val H de T5022
6	Cerrar val H de T5022
7	Abrir val E de T5015

Rack 1, Modulo 2

Salidas Discretas desde %Q.1.2.0 hasta %Q.1.2.7

0	Cerrar val E de T5015
1	Abrir val H de T5015
2	Cerrar val H de T5015
3	Abrir val E de T5016
4	Cerrar val E de T5016

5	Abrir val H de T5016
6	Cerrar val H de T5016
7	Partir Agit. A de T5022

Rack 1, Modulo 3

Salidas Discretas desde %Q.1.3.0 hasta %Q.1.3.7

0	Parar Agit. A de T5022
1	Partir Agit. B de T5022
2	Parar Agit. B de T5022
3	Partir Agit. A de T5015
4	Parar Agit. A de T5015
5	Partir Agit. B de T5015
6	Parar Agit. B de T5015
7	Partir Agit. A de T5016

Rack 1, Modulo 4

Salidas Discretas desde %Q.1.4.0 hasta %Q.1.4.7

0	Parar Agit. A de T5016
1	Partir Agit. B de T5016
2	Parar Agit. B de T5016
3	Reset Teleme T5015
4	Reset Teleme T5201
5	Alumbrado T5015
6	Abrir Val. 5023-1
7	Cerrar Val. 5023-1

Rack 1, Modulo 5

Salidas Discretas desde %Q.1.5.0 hasta %Q.1.5.7

0	Parar Val. 5023-1
---	-------------------

- | | |
|---|-------------------------|
| 1 | Abrir Val. 5023-2 |
| 2 | Cerrar Val. 5023-2 |
| 3 | Parar Val. 5023-2 |
| 4 | Partir Agit. A de T5023 |
| 5 | Parar Agit. A de T5023 |
| 6 | Alumbrado T5023 |
| 7 | Baliza 3Crudo |

Rack 1, Modulo 6

Salidas Discretas desde %Q.1.6.0 hasta %Q.1.6.7

- | | |
|---|---------------------|
| 0 | Baliza 4Crudo |
| 1 | Baliza 5Crudo |
| 2 | Baliza 3LPG |
| 3 | Bza 2Barcaza (4LPG) |
| 4 | libre |
| 5 | libre |
| 6 | libre |
| 7 | libre |

Rack 1, Modulo 7

Salidas Discretas desde %Q.1.7.0 hasta %Q.1.7.7

- | | |
|---|------------------|
| 0 | Abrir Val. 5041 |
| 1 | Cerrar Val. 5041 |
| 2 | Parar Val. 5041 |
| 3 | Abrir Val. 5042 |
| 4 | Cerrar Val. 5042 |
| 5 | Parar Val. 5042 |
| 6 | libre |
| 7 | libre |

Lógica de válvula **H** de **T5015**

Modbus 251	iAbrir	qAbrir	Rele 2 / 1
Modbus 252	iCerrar	qCerrar	Rele 2 / 2
<i>no conectado</i>	iParar	qParar	<i>no conectado</i>
Modbus 236	iReset alarma		
60	iSegundos	qAlama	Modbus 223
In 2 / 9	iAbierta	qAbierta	Modbus 110
In 2 / 10	iCerrada	qCerrada	Modbus 111
True	iLocRem	qLocRem	Modbus 119

Lógica de válvula **E** de **T5015**

Modbus 245	iAbrir	qAbrir	Rele 1 / 7
Modbus 246	iCerrar	qCerrar	Rele 2 / 0
<i>no conectado</i>	iParar	qParar	<i>no conectado</i>
Modbus 233	iReset alarma		
60	iSegundos	qAlama	Modbus 220
In 2 / 7	iAbierta	qAbierta	Modbus 108
In 2 / 8	iCerrada	qCerrada	Modbus 109
True	iLocRem	qLocRem	Modbus 118

Lógica de agitador **A** de **T5015**

Modbus 201	iPartir	qPartir	Rele 3 / 3
Modbus 202	iParar	qParar	Rele 3 / 4
Modbus 226	iReset alarma		
10	iSegundos	qAlama	Modbus 213
In 3 / 11	iEstado	qEstado	Modbus 128

Lógica de agitador **B** de **T5015**

Modbus 203	iPartir	qPartir	Rele 3 / 5
Modbus 204	iParar	qParar	Rele 3 / 6
Modbus 227	iReset alarma		
10	iSegundos	qAlama	Modbus 214
In 3 / 12	iEstado	qEstado	Modbus 129

Lógica de válvula **H** de **T5016**

Modbus 253	iAbrir	qAbrir	Rele 2 / 5
Modbus 254	iCerrar	qCerrar	Rele 2 / 6
<i>no conectado</i>	iParar	qParar	<i>no conectado</i>
Modbus 237	iReset alarma		
60	iSegundos	qAlama	Modbus 224
In 2 / 13	iAbierta	qAbierta	Modbus 114
In 2 / 14	iCerrada	qCerrada	Modbus 115
True	iLocRem	qLocRem	Modbus 121

Lógica de válvula **E** de **T5016**

Modbus 247	iAbrir	qAbrir	Rele 2 / 3
Modbus 248	iCerrar	qCerrar	Rele 2 / 4
<i>no conectado</i>	iParar	qParar	<i>no conectado</i>
Modbus 234	iReset alarma		
60	iSegundos	qAlama	Modbus 221
In 2 / 11	iAbierta	qAbierta	Modbus 112
In 2 / 12	iCerrada	qCerrada	Modbus 113
True	iLocRem	qLocRem	Modbus 120

Lógica de agitador **A** de **T5016**

Modbus 205	iPartir	qPartir	Rele 3 / 7
Modbus 206	iParar	qParar	Rele 4 / 0
Modbus 228	iReset alarma		
10	iSegundos	qAlama	Modbus 215
In 3 / 13	iEstado	qEstado	Modbus 130

Lógica de agitador **B** de **T5016**

Modbus 207	iPartir	qPartir	Rele 4 / 1
Modbus 208	iParar	qParar	Rele 4 / 2
Modbus 229	iReset alarma		
10	iSegundos	qAlama	Modbus 216
In 3 / 14	iEstado	qEstado	Modbus 131

Lógica de válvula **H** de **T5022**

Modbus 255	iAbrir	qAbrir	Rele 1 / 5
Modbus 256	iCerrar	qCerrar	Rele 1 / 6
<i>no conectado</i>	iParar	qParar	<i>no conectado</i>
Modbus 238	iReset alarma		
60	iSegundos	qAlama	Modbus 225
In 2 / 5	iAbierta	qAbierta	Modbus 106
In 2 / 6	iCerrada	qCerrada	Modbus 107
In 3 / 0	iLocRem	qLocRem	Modbus 117

Lógica de válvula **E** de **T5022**

Modbus 249	iAbrir	qAbrir	Rele 1 / 3
Modbus 250	iCerrar	qCerrar	Rele 1 / 4
<i>no conectado</i>	iParar	qParar	<i>no conectado</i>
Modbus 235	iReset alarma		
60	iSegundos	qAlama	Modbus 222
In 2 / 3	iAbierta	qAbierta	Modbus 104
In 2 / 4	iCerrada	qCerrada	Modbus 105
In 2 / 15	iLocRem	qLocRem	Modbus 116

Lógica de agitador **A** de **T5022**

Modbus 209	iPartir	qPartir	Rele 2 / 7
Modbus 210	iParar	qParar	Rele 3 / 0
Modbus 230	iReset alarma		
10	iSegundos	qAlama	Modbus 217
In 3 / 9	iEstado	qEstado	Modbus 126

Lógica de agitador **B** de **T5022**

Modbus 211	iPartir	qPartir	Rele 3 / 1
Modbus 212	iParar	qParar	Rele 3 / 2
Modbus 231	iReset alarma		
10	iSegundos	qAlama	Modbus 218
In 3 / 10	iEstado	qEstado	Modbus 127

Lógica de válvula **1H** de **T5023**

Modbus 261	iAbrir	qAbrir	Rele 4 / 6
Modbus 262	iCerrar	qCerrar	Rele 4 / 7
Modbus 263	iParar	qParar	Rele 5 / 0
Modbus 259	iReset alarma		
60	iSegundos	qAlama	Modbus 227
In 4 / 0	iAbierta	qAbierta	Modbus 133
In 4 / 1	iCerrada	qCerrada	Modbus 134
In 4 / 2	iLocRem	qLocRem	Modbus 135

Lógica de válvula **2E** de **T5023**

Modbus 264	iAbrir	qAbrir	Rele 5 / 1
Modbus 265	iCerrar	qCerrar	Rele 5 / 2
Modbus 266	iParar	qParar	Rele 5 / 3
Modbus 260	iReset alarma		
60	iSegundos	qAlama	Modbus 258
In 4 / 3	iAbierta	qAbierta	Modbus 136
In 4 / 4	iCerrada	qCerrada	Modbus 137
In 4 / 5	iLocRem	qLocRem	Modbus 138

Lógica de agitador **A** de **T5023**

Modbus 269	iPartir	qPartir	Rele 5 / 4
Modbus 270	iParar	qParar	Rele 5 / 5
Modbus 268	iReset alarma		
10	iSegundos	qAlama	Modbus 267
In 4 / 7	iEstado	qEstado	Modbus 140

Lógica de válvula **MOV-5119**

Modbus 242	iAbrir	qAbrir	Rele 1 / 0
Modbus 243	iCerrar	qCerrar	Rele 1 / 1
Modbus 244	iParar	qParar	----- NOT ----- Rele 1 / 2
Modbus 232	iReset alarma		
60	iSegundos	qAlama	Modbus 219
In 2 / 0	iAbierta	qAbierta	Modbus 101
In 2 / 1	iCerrada	qCerrada	Modbus 102
In 2 / 2	iLocRem	qLocRem	Modbus 103

Lógica de válvula MOV-5041

Modbus 281	iAbrir	qAbrir	Rele 7 / 0
Modbus 282	iCerrar	qCerrar	Rele 7 / 1
Modbus 283	iParar	qParar	----- NOT ----- Rele 7 / 2
Modbus 279	iReset alarma		
60	iSegundos	qAlama	Modbus 277
In 5 / 3	iAbierta	qAbierta	Modbus 151
In 5 / 4	iCerrada	qCerrada	Modbus 152
In 5 / 2	iLocRem	qLocRem	Modbus 153

Lógica de válvula MOV-5042

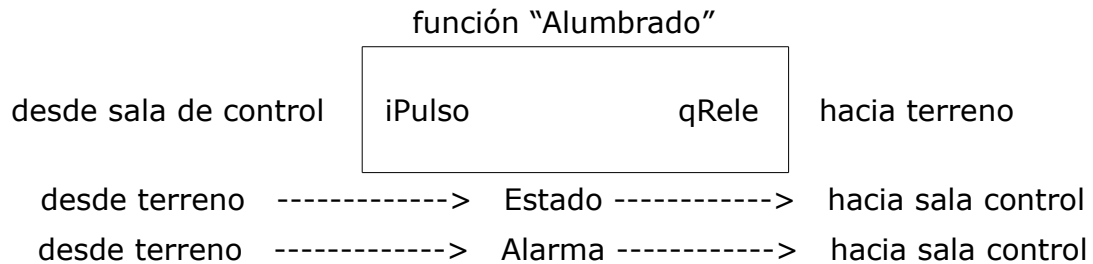
Modbus 284	iAbrir	qAbrir	Rele 7 / 3
Modbus 285	iCerrar	qCerrar	Rele 7 / 4
Modbus 286	iParar	qParar	----- NOT ----- Rele 7 / 5
Modbus 280	iReset alarma		
60	iSegundos	qAlama	Modbus 278
In 5 / 6	iAbierta	qAbierta	Modbus 154
In 5 / 7	iCerrada	qCerrada	Modbus 155
In 5 / 5	iLocRem	qLocRem	Modbus 156

Lógica Alumbrado y Balizas:

Todos los interruptores que encienden alumbrado o balizas, usan esta misma lógica.

En la siguiente imagen, un ejemplo de aplicación de la función "Alumbrado".

Al lado izquierdo del bloque, las señales conectadas a las entradas de la función. A la derecha del bloque, las señales conectadas a las salidas de la función.



Lógica dentro de la función "Alumbrado"

```
(* Cada vez que llega un pulso, invierte el estado del rele que enciende el alumbrado.*)  
Triger (CLK := iPulso, Q => bPulso);  
(* Detecta el cambio de 0 a 1 de la entrada iPulso *)  
IF bPulso THEN  
    (* solo cuando iPulso cambia de 0 a 1 *)  
    qRele := NOT iEstado;  
END_IF;
```

Desde sala de control, por comunicación, llega un pulso, cuando se detecta la llegada de este pulso se invierte el estado del relé, que enciende el alumbrado.

Entradas de estados se envían directo hacia sala de control, sin usarse en la logica.

Lógica de **Baliza 3 crudo**

Modbus 272	iPulso	qRele	Rele 5 / 7
In 4 / 8	----- Estado ----->		Modbus 40 y Modbus 141
In 4 / 9	----- Alarma ----->		Modbus 142

Lógica de **Baliza 4 crudo**

Modbus 273	iPulso	qRele	Rele 6 / 0
In 4 / 10	----- Estado ----->		Modbus 41 y Modbus 143
In 4 / 11	----- Alarma ----->		Modbus 144

Lógica de **Baliza 5 crudo**

Modbus 274	iPulso	qRele	Rele 6 / 1
In 4 / 12	----- Estado ----->		Modbus 42 y Modbus 145
In 4 / 13	----- Alarma ----->		Modbus 146

Lógica de **Baliza 3 LPG**

Modbus 275	iPulso	qRele	Rele 6 / 2
In 4 / 14	----- Estado ----->		Modbus 43 y Modbus 147
In 4 / 15	----- Alarma ----->		Modbus 148

Lógica de **Baliza 2 Barcaza** (antes 4 LPG)

Modbus 276	<div> iPulso qRele </div>	Rele 6 / 3
In 5 / 0	----- Estado ----->	Modbus 44 y Modbus 149
In 5 / 1	----- Alarma ----->	Modbus 150

Lógica de **Alumbrado T5015, 16, 22**

Modbus 239	<div> iPulso qRele </div>	Rele 4 / 5
Rele 4 / 5	----- Estado ----->	Modbus 30

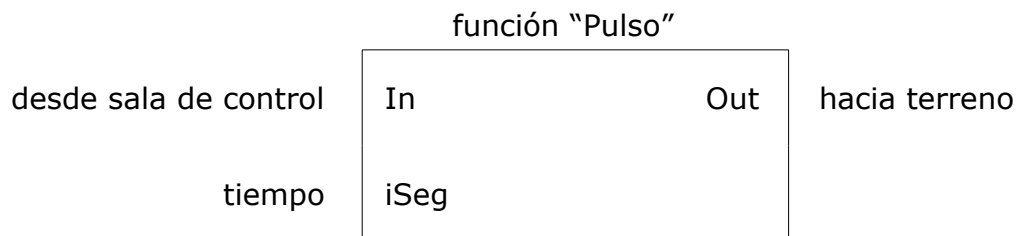
Lógica de **Alumbrado T5023**

Modbus 271	<div> iPulso qRele </div>	Rele 5 / 6
Rele 5 / 6	----- Estado ----->	Modbus 39

Lógica de Reset Telemédica:

Cuando llega la orden desde sala de control, se enciende el rele, solo por los segundos especificados.

Al lado izquierdo del bloque, las señales conectadas a las entradas de la función. A la derecha del bloque, las señales conectadas a las salidas de la función.



Lógica dentro de la función "Alumbrado"

```
(* Cada vez que llega un pulso, activa la salida por los segundos especificados *)
(* validar el valor en segundos ingresado *)
IF iSeg < 1 THEN mSeg := 1000; END_IF;
IF iSeg > 300 THEN mSeg := 300000; END_IF;
IF (iSeg >= 1) AND (iSeg <= 3000) THEN
    mSeg := (iSeg * 1000);
END_IF;
Segundos := DINT_TO_TIME (mSeg);

(* Si detecta el cambio de 0 a 1 de la entrada iPulso *)
Triger (CLK := In, Q => bIN);

(* Temporizador con retardo a la desconexion entrega un pulso de los segundos solicitados *)
Tiempo (IN := bIN, PT := Segundos, Q => Out);
qSeg := Segundos;
```

Desde sala de control, por comunicación, llega un pulso, cuando se detecta la llegada de este pulso se activa el relé, solo por los segundos especificados.

Reset Telemedida 5015

Modbus 240

2

In

Out

Rele 4 / 3

iSeg

Reset Telemedida 5201

Modbus 241

2

In

Out

Rele 4 / 4

iSeg

Otras Alarmas:

"LAH-T5022" In 3 / 5 -----> Modbus 122

"LAH-T5016" In 3 / 6 -----> Modbus 123

"LAH-T5015" In 3 / 7 -----> Modbus 124

"LAH-T5007" In 3 / 8 -----> Modbus 125

"LSHH-5023-1" In 4 / 6 -----> Modbus 139

Falla alimentación a PLC In 3 / 15 -----> Modbus 132