CAPITULO 4

Desarrollo del proyecto

4.1 Módulos programados

En este capítulo y el siguiente vamos a detallar el código desarrollado así como el SW utilizado para la programación del SCADA. Dichos SW ha sido Control Builder 3.2 (dedicamos este capítulo para hablar de él) y Process Portal A (tratado en el posterior capítulo). Control Builder 3.2 es la aplicación para Windows 2000 Professional para la programación y la configuración de los controladores de ABB mientras Process Portal A es la herramienta, mediante la cual, se lleva a cabo de creación, configuración y mantenimiento del sistema de control.

4.1.1 Introducción a Control Builder

Como hemos dicho en el párrafo anterior Control Builder es el programa utilizado para desarrollar el código que necesitamos para nuestro proyecto así como para la configuración. CB nos permite programar en distintos lenguajes: FBD (Function Block Diagram), ST (Structured Text), LD (Ladder Diagram), SFC (Secuential Function Charts) e IL (Instruction List). La elección del tipo de lenguaje de programación depende del ingeniero en última instancia aunque algunos lenguajes se adaptan mejor que otros a nuestras necesidades. Así, por ejemplo, para hacer una secuencia podemos usar ST pero parece más razonable utilizar un lenguaje creado específicamente para esta misión como el SFC.

En este proyecto hemos realizado códigos en ST, FBD y SFC. CB, además, nos da la posibilidad de mezclar en una misma función distintos tipos de lenguajes como veremos en los ejemplos.

CB también se encarga de la configuración de los controladores de ABB. Podemos configurar el PLC, los módulos de entradas y salidas digitales y analógicos y demás módulos que deseemos insertar a nuestro controlador. También podemos configurar la comunicación. Por ejemplo, en Ethernet definimos direcciones IP, máscara de subred, etc. Todo esto lo iremos viendo, poco a poco, a medida que desarrollemos el capítulo.

A continuación mostramos la pantalla inicial:

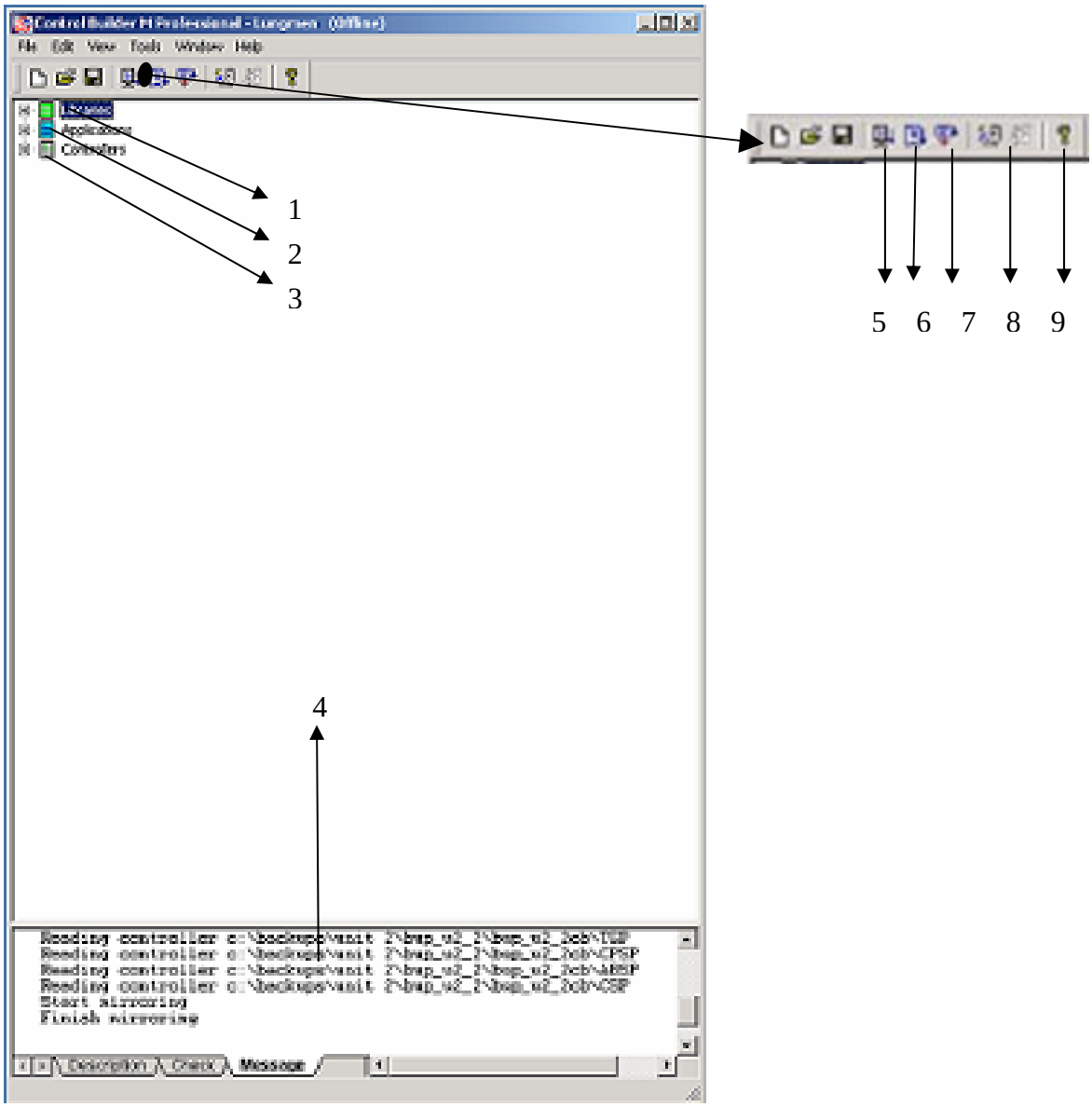

FIGURA 4.1

1. Libraries: Contiene las librerías que contienen los bloques básicos a partir de los cuáles generaremos todo la aplicación
2. Appliccations: Aplicaciones creadas
3. Controllers: Controladores configurados para ejecutar la aplicación
4. Console: Nos información general de los procesos que vamos realizando
5. Simulate: Activa el modo de simulación
6. On/Off: Se pone en modo on-line/ off-line

7. Download: Descarga el código en el PLC y se pone en modo on-line
8. Security: En caso de poner contraseña nos prohíbe realizar operaciones si no la conocemos
9. Help: ayuda

Si desplegamos Libraries podemos ver lo siguiente:

FIGURA 4.2

No son más que una serie de librerías, la mayoría añadidas por defecto cuando creamos un nuevo proyecto, como SystemLib, AlarmEvenLib e IconLib. La primera (SystemLib) contiene las funciones básicas del sistema como contadores, timers, etc; la segunda (AlarmEvenLib) posee toda función relacionada con la gestión de alarmas y, la tercera, (IconLib) lo relacionado con los iconos necesarios para hacer códigos. Todos ellos están especificados en el estándar IEC 61131-3.

Además, aquí también se añaden las librerías creadas por nosotros. Si nos fijamos la segunda carpeta se llama LungmenLib. Ésta contendrá las funciones básicas diseñadas por el programador como válvula, motor, analizador, etc… a partir de las cuales generaremos todas las válvulas, motores, analizadores, etc… que necesitemos para nuestro proyecto.

Como vemos la librería LungmenLib a su vez se divide en tres subcarpetas: Data Types, Function Block Types y Control Module Types.

Los “Data Types” son tipos de datos compuestos por agrupación de tipos básicos, lo cual presenta una agrupación de registros más útil para informar de un determinado elemento, por ejemplo, una válvula.

Los “Function Block Types” desarrollan una tarea cada vez que son llamados.

Los”Control Module Types” desarrollan una tarea de manera constante. Presentan características que le otorgan una funcionalidad diferente y más completa que la de un “Function Block Types”.

Si nos fijamos en la figura 4.2 vemos que el analizador tiene asociada una función llamada AlarmCondM. Esta función se encarga de gestionar todo el proceso de alarmas que es necesario para la interfaz hombre-máquina y nos da información que será utilizada en el PPA. La mayoría de las funciones básicas creada poseen la función AlarmCondM como iremos viendo.

4.1.2 Funciones creadas:

A continuación vamos a describir cada uno de los “Fuction Block Types” y “Control Module Types” utilizados, empezamos por los “Fuction Block Types” :

- **AlarmAnnuciator**:

Es la función encargada de generar las señales necesarias cuando se produce una alarma. Así en caso de que ésta ocurra se generará una indicación en el SCADA pero, además, activará una señal sonora y encenderá una bombilla en el C&RP. El protocolo a seguir por parte de la bombilla, en caso de alarma, es el siguiente:

Cuando se detecte una alarma se generará una señal audible en el C&RP, además se iluminará la bombilla correspondiente parpadeando de forma rápida y, al presionarse el botón de reconocimiento se silenciará la señal audible y la bombilla pasará a quedarse encendida sin parpadear.

Cuando la condición que ha hecho saltar la alarma vuelva a estado normal, la bombilla parpadeará de manera lenta y continuará así hasta que se pulse el botón de reset. Si la condición vuelve a normal antes que el botón de reconocimiento sea pulsado,

la señal audible desaparecerá y la bombilla cambia directamente de rápido a lento parpadeo.

Además de esta indicación, en la pantalla del Puesto de Operación (Ingeniería) aparecerá un texto de alarma que nos indicará cuando se ha producido la alarma, quien la ha producido y cual ha sido el motivo. También, al igual que en el caso anterior, sigue un protocolo:

Cuando ocurre una alarma aparece en rojo en el listado de alarmas, si reconocemos la alarma, el texto se vuelve amarillo. Si la alarma ha desaparecido estando reconocida (amarillo) desaparecerá el texto asociado. Si la alarma no está reconocida (rojo) y desaparece dicha alarma antes que la reconozcamos, la alarma desaparece al reconocerla (no se pone el amarillo).

Además, el operador dispondrá de páginas con resúmenes de alarmas, que le indicarán cuando han ocurrido (día y hora), tipo de alarma, etc.

Las variables externas utilizadas se muestran en la figura 4.3:

	Name	Data Type	Attributes	Direction	Initial Value	Description
1	Alarm_input	bool	retain	in		Alarm input
2	Alarm_Ack	bool	retain	in		Alarm Acknowledge button
3	Alarm_Reset	bool	retain	in		Alarm Reset button
4	Lamp_Test	bool	retain	in		Alarm test button
5	Alarm_output	bool	retain	out		Alarm output
6	Sound_output	bool	retain	out		Audible anunciaton
7	AlarmCondMi	dint	retain	in		ack from Scada

FIGURA 4.3

Las variables locales son:

	Name	Data Type	Attributes	Initial Valu	Description
1	ResetAck	bool	retain		Removed flag acknowledge
2	Acknowledge	bool	retain		Flag Acknowledge
3	quickly	bool	retain		Fast bright of alarm annunciator
4	slowly	bool	retain		Slow bright of alarm annunciator
5	brightness	bool	retain		Permanent bright of alarm annunciator
6	cycle	time	retain		Cycle time of alarm annunciator
7	t_quickly	time	retain	t#500ms	Cycle time for fast bright
8	t_slowly	time	retain	t#2s	Cycle time for slow bright
9	enable_trigger	bool	retain		It enables the trigger function block
10	blink_output	bool	retain		Output of blink function block
11	Alarm_Active	bool	retain		Alarm Active
12	AlarmCondBool	bool	retain		if AlarmCondMi==5-->AlarmCondBool=true
13	Gen_Ack	bool	retain		general ack

FIGURA 4.4

Las subfunciones utilizadas son:

	Name	Function Block Type	Task Connection	Description
1	R_trig1	R_Trig		
2	R_trig2	R_Trig		
3	R_trig3	R_Trig		
4	RS1	RS		
5	blink	Trigger		
6	RS2	SR		

FIGURA 4.5

Donde R_trig genera un pulso de duración un ciclo de reloj cuando la entrada conmuta de “0” a “1”; RS es un biestable RS; Trigger genera un tren de pulsos periódico (definido por el programador) cuando su entrada *Enable* está habilitada; y SR es un biestable SR.

A continuación mostramos el código desarrollado:

```
(* Pulses detection *)

R_trig1( Clk := Alarm_input );
R_trig2( Clk := Alarm_Ack );
R_trig3( Clk := Alarm_Reset );
```

```
(* Acknowledge *)

ResetAck := R_trig1.Q or (Alarm_Active and slowly and R_trig3.Q );
RS1( Set := (Alarm_Ack and Alarm_Active),
    R1 := ResetAck,
    Q1 => Acknowledge );
RS2( S1 := Alarm_input,
    Reset := ResetAck,
    Q1 => Alarm_Active);

(* Quickly *)

if Alarm_Active and not Acknowledge and Alarm_input then
  quickly:=true;
else
  quickly:=false;

end_if;

(* Permanent brightness  *)

if Alarm_Active and Acknowledge and Alarm_input then
  brightness:=true;
else
  brightness:=false;
end_if;
```

```
(* Slowly *)

if Alarm_Active and not Alarm_input then
  slowly:=true;
else
  slowly:=false;
end_if;

(* Outputs *)

if not quickly and not slowly and not brightness and not Lamp_Test then
  Alarm_output := false;
  enable_trigger := false;
  sound_output:=false;
elsif brightness then
  Alarm_output := true;
  enable_trigger := false;
  sound_output:=false;
elsif quickly then
  cycle := t_quickly;
  enable_trigger := true;
  sound_output:=true;
elsif slowly then
  cycle := t_slowly;
  enable_trigger := true;
  sound_output:=false;
end_if;
```

```
blink( Request := enable_trigger,
      Periodic := enable_trigger,
      Period := cycle,
      Out => blink_output );

if quickly or slowly then
  Alarm_output := blink_output ;
end_if;

if Lamp_Test then
  Alarm_output := true;
  sound_output := true;
end_if;
```

- **Contadores**:

El objetivo de esta función es garantizar que existe comunicación con el cliente OPC. Para ello envía cada segundo el valor de un contador que va de 0 a 10000. Cuando llega a 10000 vuelve a empezar en 0. El cliente OPC comprueba que dicho valor le está llegando, en el momento que dicho contador no llegue se entenderá que las comunicaciones han caído y se seguirán las acciones necesarias de seguridad.

El código es el siguiente:

```
Pulse2seg( Enable := Enable,
           PulseTime := PulseTime,
           PeriodTime := PeriodTime,
           Out => Pulse );

Trigger(Clk := Pulse);

if(trigger.Q) then

  if(signalCounter = 10000) then
    signalCounter := 0;
  else
    signalCounter := signalCounter + 1;
  end_if;

end_if;
```

Pulse2seg es una función que genera un tren de pulsos de periodo PeriodTime (2 segundos en nuestro caso) y duración del nivel de subida de PulseTime(1 segundo). Está siempre habilitada gracias a que el valor Enable está por defecto a "1" y la salida la guardamos en la variable Pulse.

- **Hotwellpulses**:

Esta función es la que tiene el control de la secuencia de apertura y cierre de las válvulas en el área CSS. Cuando se pasa a modo automático y se activa (START)

comienza la secuencia; la primera válvula se abre permaneciendo abierta 5 segundos; pasado ese tiempo pasa a la siguiente válvula cerrándose la anterior y así sucesivamente.

Este código ha sido realizado en un lenguaje llamado SFC (Secuential Function Charts) especialmente diseñado para este tipo de aplicaciones secuenciales.

Mostramos la secuencia:

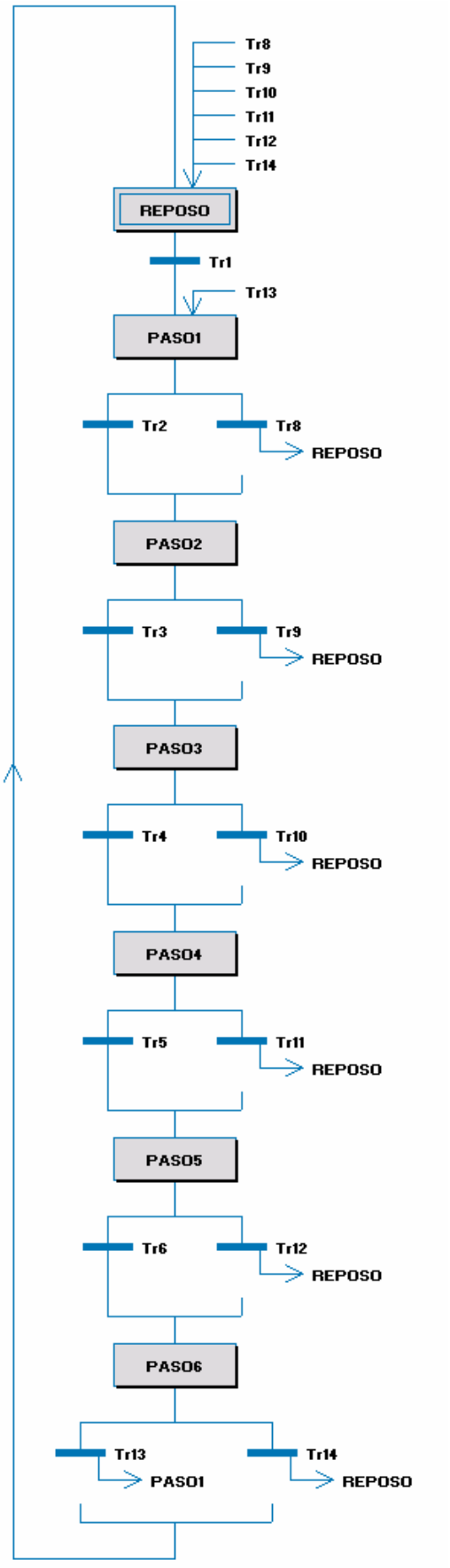
Tr8
Tr9
Tr10
Tr11
Tr12
Tr14
REPOSO
Tr1
Tr13
PASO1
Tr2
Tr8
REPOSO
PASO2
Tr3
Tr9
REPOSO
PASO3
Tr4
Tr10
REPOSO
PASO4
Tr5
Tr11
REPOSO
PASO5
Tr6
Tr12
REPOSO
PASO6
Tr13
PASO1
Tr14
REPOSO

FIGURA 4.6

Reposo: (inicializamos las válvulas poniéndolas en estado cerrado. Sólo se ejecuta una vez cuando la condición **Tr1** es verdadera)

Output1:= false;

Output2:= false;

Output3:= false;

Output4:= false;

Output5:= false;

Output6:= false;

Tr1: (Hasta que esté en modo automático y todas las válvulas cerradas no pasamos al siguiente paso)

(AUTO) and (not(valve1 or Valve2 or valve3 or Valve4 or valve5 or Valve6))

Paso1: (Abre la primera válvula y cierra la última)

Output1:= true;

Output6:= false;

Tr2: (Hasta que no pasa TIME_ELAPSED (5 segundos) y la válvula 1 esté abierta y la 6 cerrada no continúa)

(PASO1.T>= TIME_ELAPSED) and (not valve6 and valve1)

Tr8: (Si pasamos a modo manual nos vamos al paso **REPOSO**)

Not(AUTO)

Paso2: (Abre la segunda válvula y cierra la primera)

Output2:= true;

Output1:= false;

Tr3: (Hasta que no pasa TIME_ELAPSED y la válvula 2 esté abierta y la 1 cerrada no continúa)

(PASO2.T>= TIME_ELAPSED) and (not valve1 and valve2)

Tr9: (Si pasamos a modo manual nos vamos al paso **REPOSO**)

Not(AUTO)

Paso3: (Abre la tercera válvula y cierra la segunda)

Output3:= true;

Output2:= false;

Tr4: (Hasta que no pasa TIME_ELAPSED y la válvula 3 esté abierta y la 2 cerrada no continúa)

(PASO3.T>= TIME_ELAPSED) and (not valve2 and valve3)

Tr10: (Si pasamos a modo manual nos vamos al paso **REPOSO**)

Not(AUTO)

Paso4: (Abre la cuarta válvula y cierra la tercera)

Output4:= true;

Output3:= false;

Tr5: (Hasta que no pasa TIME_ELAPSED y la válvula 4 esté abierta y la 3 cerrada no continúa)

(PASO4.T>= TIME_ELAPSED) and (not valve3 and valve4)

Tr11: (Si pasamos a modo manual nos vamos al paso **REPOSO**)

Not(AUTO)

Paso5: (Abre la quinta válvula y cierra la cuarta)

Output5:= true;

Output4:= false;

Tr6: (Hasta que no pasa TIME_ELAPSED y la válvula 5 esté abierta y la 4 cerrada no continúa)

(PASO5.T>= TIME_ELAPSED) and (not valve4 and valve5)

Tr12: (Si pasamos a modo manual nos vamos al paso **REPOSO**)

Not(AUTO)

Paso6: (Abre la sexta válvula y cierra la quinta)

Output6:= true;

Output5:= false;

Tr13: (Hasta que no pasa TIME_ELAPSED y la válvula 6 esté abierta y la 5 cerrada no continúa. Volvemos al PASO1 de la secuencia)

(PASO6.T>= TIME_ELAPSED) and (not valve5 and valve6)

Tr14: (Si pasamos a modo manual nos vamos al paso **REPOSO**)

Not(AUTO)

Una vez visto todos los “Fuction Block Types” vamos a ver los”Control Module Types”:

- **Analyzer:**

  Contiene el código del analizador de medidas.

```
(* Measure_output *)
Measure_output:= Measure_input;

(* High concentration alarm *)
if (Measure_input -AlarmValue) >= 0 then
        MaxValueAlarm := true;
else
        MaxValueAlarm := false;
end_if;

(* Analizar Fault *)

AnalyzerFault_Output := not AnalyzerFault_Input;
```

La primera parte del código sirve para volcar la entrada (Measure_Input) a la salida (Measure_Output). A continuación, comprobamos si el nivel de alarma (AlarmValue) es inferior a la medida tomada o no para que la variable MaxValueAlarm tome el valor deseado (dicho parámetro está asociado a una AlarmCondM que se explicará posteriormente).

Finalmente, invertimos la señal AnalizarFault_input obteniendo AnalizarFault_output que activará un Alarmannunciator.

Además, tiene asociado un AlarmCondM. A continuación vemos los parámetros que hay que insertar para funcione correctamente:

	Name	Data Type	Initial Valu	Parameter	Directio	Description
1	Signal	bool		MaxValueAlarm	in_out	IN Internal monitored Signal
2	ExtTimeStamp	bool	false		in_out	IN EDIT False = Signal is monitc
3	SignalID	string[20]	''		in_out	IN EDIT Identifier of the signal wi
4	UseSigToInit	bool	false		in_out	IN EDIT True = Use Signal to ge
5	SrcName	string[30]		Name	in_out	IN EDIT Name of the source has
6	Message	string[60]		'CONCENTRATION TOO HIGH'	in_out	IN Description and/or information
7	Severity	dint	500		in_out	IN EDIT Severity of the alarm co
8	Class	dint	1		in_out	IN EDIT Class this alarm conditi
9	Inverted	bool	false		in_out	IN EDIT Tells if monitored Signal
10	AckRule	dint	1		in_out	IN EDIT Acknowledge rule. 1=No
11	FilterTime	time	0s		in_out	IN EDIT Positive pulses on Signa
12	EnDetection	bool	true		in_out	IN If true, the Signal is currently
13	AckCond	bool	false		in_out	IN Acknowledge alarm condition
14	DisCond	bool	false		in_out	IN Disable alarm condition on po
15	EnCond	bool	false		in_out	IN Enable alarm condition on pos
16	CondState	dint	Default		in_out	OUT Alarm condition state (0-6)
17	Error	bool	Default		in_out	OUT Indicates an error with True
18	Status	dint	Default		in_out	OUT Shows the status code of t

FIGURA 4.7

Signal: Es la señal que monitoriza la alarma. Si es igual a"1" la activa si es "0" indica que no hay alarma. En nuestro caso la señal es MaxValueAlarm

Srcname: Nos mostrará en el PPA el nombre del equipo afectado. Es este caso copiará lo que ponga en Name(parámetro definido en la función)

Message: Nos mostrará en el PPA una descripción del motivo de la alarma. En nuestro caso “CONCENTRATION TOO HIGH”.

- **Chiller**:

Esta función tan solo vuelca la entrada invertida (Trouble_input) a la salida (Trouble_output) y a otra salida(Trouble_output_H) pero, en este caso, sin invertir.

(Chiller Trouble *)*

Trouble_output := not Trouble_input;

Trouble_output_H := not Trouble_output;

La variable Trouble_output_H se utiliza para el cliente OPC (Foxboro RMU)

Tiene asociado un AlarmCondM. Vemos los parámetros que hay que insertar:

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Signal	bool		Trouble_output	in_out	IN Internal monitored Signal
2	ExtTimeStamp	bool	false		in_out	IN EDIT False = Signal is monitc
3	SignalID	string[20]	''		in_out	IN EDIT Identifier of the signal wi
4	UseSigToInit	bool	false		in_out	IN EDIT True = Use Signal to ge
5	SrcName	string[30]		Name	in_out	IN EDIT Name of the source has
6	Message	string[60]		'TROUBLE'	in_out	IN Description and/or informatior
7	Severity	dint	500		in_out	IN EDIT Severity of the alarm co
8	Class	dint	1		in_out	IN EDIT Class this alarm conditi
9	Inverted	bool	false		in_out	IN EDIT Tells if monitored Signal
10	AckRule	dint	1		in_out	IN EDIT Acknowledge rule. 1=Nc
11	FilterTime	time	0s		in_out	IN EDIT Positive pulses on Signa
12	EnDetection	bool	true		in_out	IN If true, the Signal is currently
13	AckCond	bool	false		in_out	IN Acknowledge alarm condition
14	DisCond	bool	false		in_out	IN Disable alarm condition on po
15	EnCond	bool	false		in_out	IN Enable alarm condition on po:
16	CondState	dint	Default		in_out	OUT Alarm condition state (0-6)
17	Error	bool	Default		in_out	OUT Indicates an error with True
18	Status	dint	Default		in_out	OUT Shows the status code of t

FIGURA 4.8

- **IsothermalBath:**

Únicamente copia la entrada invertida (Lowlevel_input) a la salida (Lowlevel_output).

(Low Level Isothermal Bath*)*

Lowlevel_output := not Lowlevel_input;

Vemos su AlarmCondM:

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Signal	bool		LowLevel_output	in_out	IN Internal monitored Signal
2	ExtTimeStamp	bool	false		in_out	IN EDIT False = Signal is monitc
3	SignalID	string[20]	''		in_out	IN EDIT Identifier of the signal wi
4	UseSigToInit	bool	false		in_out	IN EDIT True = Use Signal to ge
5	SrcName	string[30]		Name	in_out	IN EDIT Name of the source has
6	Message	string[60]		'LOW LEVEL'	in_out	IN Description and/or informatior
7	Severity	dint	500		in_out	IN EDIT Severity of the alarm co
8	Class	dint	1		in_out	IN EDIT Class this alarm conditi
9	Inverted	bool	false		in_out	IN EDIT Tells if monitored Signal
10	AckRule	dint	1		in_out	IN EDIT Acknowledge rule. 1=Nc
11	FilterTime	time	0s		in_out	IN EDIT Positive pulses on Signa
12	EnDetection	bool	true		in_out	IN If true, the Signal is currently
13	AckCond	bool	false		in_out	IN Acknowledge alarm condition
14	DisCond	bool	false		in_out	IN Disable alarm condition on po
15	EnCond	bool	false		in_out	IN Enable alarm condition on po:
16	CondState	dint	Default		in_out	OUT Alarm condition state (0-6)
17	Error	bool	Default		in_out	OUT Indicates an error with True
18	Status	dint	Default		in_out	OUT Shows the status code of t

FIGURA 4.9

- **Panel**:

Esta función se encarga de la gestión del C&RP. Así nos informa de la temperatura del C&RP (Temp_output = Temp_input) y si está la puerta abierta o no(Door_output = Door_input).

(Control Panel temperatura *)*

Temp_output : = Temp_input;

(Door Panel Open *)*

Door_output = Door_input;

Vemos su AlarmCondM:

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Signal	bool		Door_output	in_out	IN Internal monitored Signal
2	ExtTimeStamp	bool	false		in_out	IN EDIT False = Signal is monitored; True = SignalID is monitored wi
3	SignalID	string[20]	''		in_out	IN EDIT Identifier of the signal with external time stamp. Valid when E
4	UseSigToInit	bool	false		in_out	IN EDIT True = Use Signal to get an init value. Valid when ExtTimeSt
5	SrcName	string[30]		Name	in_out	IN EDIT Name of the source has to be unique together with the condi
6	Message	string[60]		'PANEL DOOR OPEN'	in_out	IN Description and/or information about the alarm condition
7	Severity	dint	500		in_out	IN EDIT Severity of the alarm condition. From 1 (low) to 1000 (high)
8	Class	dint	1		in_out	IN EDIT Class this alarm condition belongs to. From 1 to 9999
9	Inverted	bool	false		in_out	IN EDIT Tells if monitored Signal is interpreted inverted
10	AckRule	dint	1		in_out	IN EDIT Acknowledge rule. 1=Normal ack. 2=No ack., 3=Ack. reset
11	FilterTime	time	0s		in_out	IN EDIT Positive pulses on Signal shorter than this will not be noted (
12	EnDetection	bool	true		in_out	IN If true, the Signal is currently being checked
13	AckCond	bool	false		in_out	IN Acknowledge alarm condition on positive edge
14	DisCond	bool	false		in_out	IN Disable alarm condition on positive edge
15	EnCond	bool	false		in_out	IN Enable alarm condition on positive edge
16	CondState	dint	Default		in_out	OUT Alarm condition state (0-6)
17	Error	bool	Default		in_out	OUT Indicates an error with True
18	Status	dint	Default		in_out	OUT Shows the status code of the last execution

FIGURA 4.10

- **Panel Temperature**:

Complementa la información del C&RP. Nos informa sobre un sensor de temperatura asociado a la temperatura de dicho C&RP. La salida (TS_output) está negada respecto a la entrada (TS_input).

(TS activation *)*

TS_output : = not TS_input;

Vemos su AlarmCondM:

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Signal	bool		TS_Output	in_out	IN Internal monitored Signal
2	ExtTimeStamp	bool	false		in_out	IN EDIT False = Signal is monitc
3	SignalID	string[20]	''		in_out	IN EDIT Identifier of the signal wi
4	UseSigToInit	bool	false		in_out	IN EDIT True = Use Signal to ge
5	SrcName	string[30]		Name	in_out	IN EDIT Name of the source has
6	Message	string[60]		'TS ACTED'	in_out	IN Description and/or informatior
7	Severity	dint	500		in_out	IN EDIT Severity of the alarm co
8	Class	dint	1		in_out	IN EDIT Class this alarm conditi
9	Inverted	bool	false		in_out	IN EDIT Tells if monitored Signal
10	AckRule	dint	1		in_out	IN EDIT Acknowledge rule. 1=Nc
11	FilterTime	time	0s		in_out	IN EDIT Positive pulses on Signa
12	EnDetection	bool	true		in_out	IN If true, the Signal is currently
13	AckCond	bool	false		in_out	IN Acknowledge alarm condition
14	DisCond	bool	false		in_out	IN Disable alarm condition on po
15	EnCond	bool	false		in_out	IN Enable alarm condition on po:
16	CondState	dint	Default	Ack_scada	in_out	OUT Alarm condition state (0-6)
17	Error	bool	Default		in_out	OUT Indicates an error with True
18	Status	dint	Default		in_out	OUT Shows the status code of t

FIGURA 4.11

- **PLC Status**:

Nos da la información acerca del estado del PLC así como de las tarjetas asociadas a él. Si AllUnits es igual a “1” indica que todo es correcto y pondrá la variable PLC_Trouble a “1” y la variable PLC_Trouble_H a “0”, en caso contrario, se invertirán las señales.

```
(* PLC Trouble *)

if ( AllUnits = 1) then
        PLC_Trouble := true;
        PLC_Trouble_H := false;
else
```

PLC_Trouble := false;

PLC_Trouble_H := true;

end_if;

Donde la variable PLC_Trouble_H se utiliza para el cliente OPC (Foxboro RMU)

Vemos su AlarmCondM:

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Signal	bool		PLC_Trouble	in_out	IN Internal monitored Signal
2	ExtTimeStamp	bool	false		in_out	IN EDIT False = Signal is monitc
3	SignalID	string[20]	''		in_out	IN EDIT Identifier of the signal wi
4	UseSigToInit	bool	false		in_out	IN EDIT True = Use Signal to ge
5	SrcName	string[30]		Name	in_out	IN EDIT Name of the source has
6	Message	string[60]		'TROUBLE'	in_out	IN Description and/or informatior
7	Severity	dint	500		in_out	IN EDIT Severity of the alarm co
8	Class	dint	1		in_out	IN EDIT Class this alarm conditi
9	Inverted	bool	false		in_out	IN EDIT Tells if monitored Signal
10	AckRule	dint	1		in_out	IN EDIT Acknowledge rule. 1=Nc
11	FilterTime	time	0s		in_out	IN EDIT Positive pulses on Signa
12	EnDetection	bool	true		in_out	IN If true, the Signal is currently
13	AckCond	bool	false		in_out	IN Acknowledge alarm condition
14	DisCond	bool	false		in_out	IN Disable alarm condition on po
15	EnCond	bool	false		in_out	IN Enable alarm condition on po:
16	CondState	dint	Default		in_out	OUT Alarm condition state (0-6)
17	Error	bool	Default		in_out	OUT Indicates an error with True
18	Status	dint	Default		in_out	OUT Shows the status code of t

FIGURA 4.12

- **Power**:

Esta función contiene todo el control y supervisión de la tensión de alimentación del PLC y las tarjetas de entrada y salida. Primero mediante el parámetro Powerout sabremos si alguna de las fuentes de alimentación ha caído (Analyzer Panel 120V, Sample Conditioning 120V, Sample Conditioning 480V y Control and Recorder panel 120V). Posteriormente, volcamos la entrada de cada fuente a una salida negada

(SRC120out = not SRC120in , SRC480out = not SRC480in , AP120out = not AP120in y CRP120out = not CRP120in):

```
(* Power *)

Powerout := not (SRC120in and  SRC480in and AP120in and CRP120in ) ;

SRC120out := not SRC120in;
SRC480out := not SRC480in;
AP120out := not AP120in;
CRP120out := not CRP120in;

SRC120 := SRC120out;
SRC480 := SRC480out;
AP120 := AP120out;
CRP120 := CRP120out;

(* Alarm *)

Alarm1(Signal := AP120.
       SrcName := Name,
       Message := desc_AP120 );

Alarm2(Signal := CRP120.
       SrcName := Name,
       Message := desc_CRP120 );
```

Alarm3(Signal := SCR120.

SrcName := Name,

Message := desc_SCR120);

Alarm4(Signal := SCR480.

SrcName := Name,

Message := desc_SCR480);

La última parte del código gestiona las alarmas que se pasarán al PPA. El significado y la utilización es idéntica que en AlarmCondM, simplemente que aquí se pasa directamente en el propio código mediante “Function Blocks”.

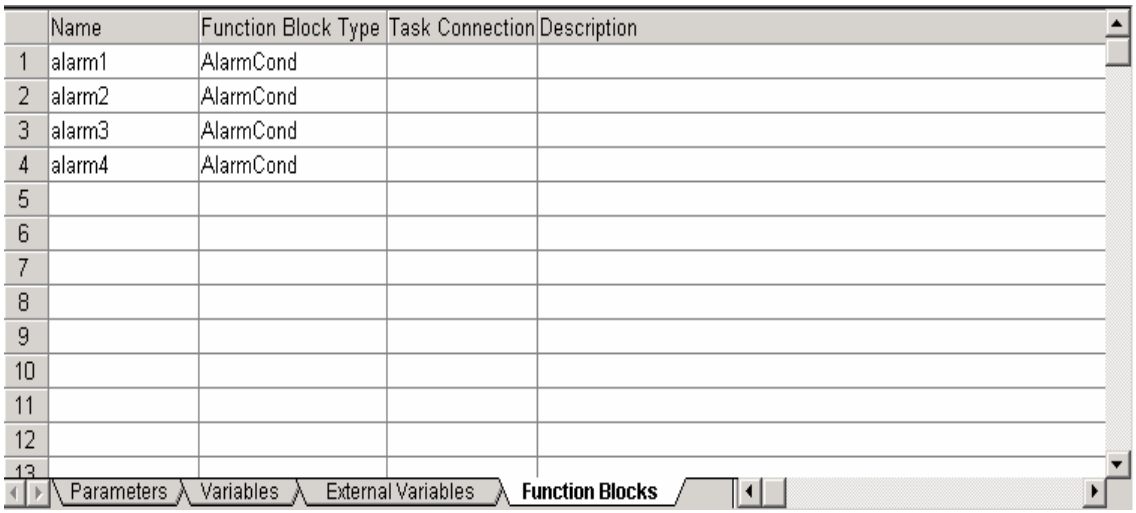

FIGURA 4.13

- **Pump**:

Se encarga de la gestión de las bombas. Como vemos pasamos del texto estructurado (ST) a la programación en bloques (FBD). Este código posee dos ventanas:

En la primera ventana,

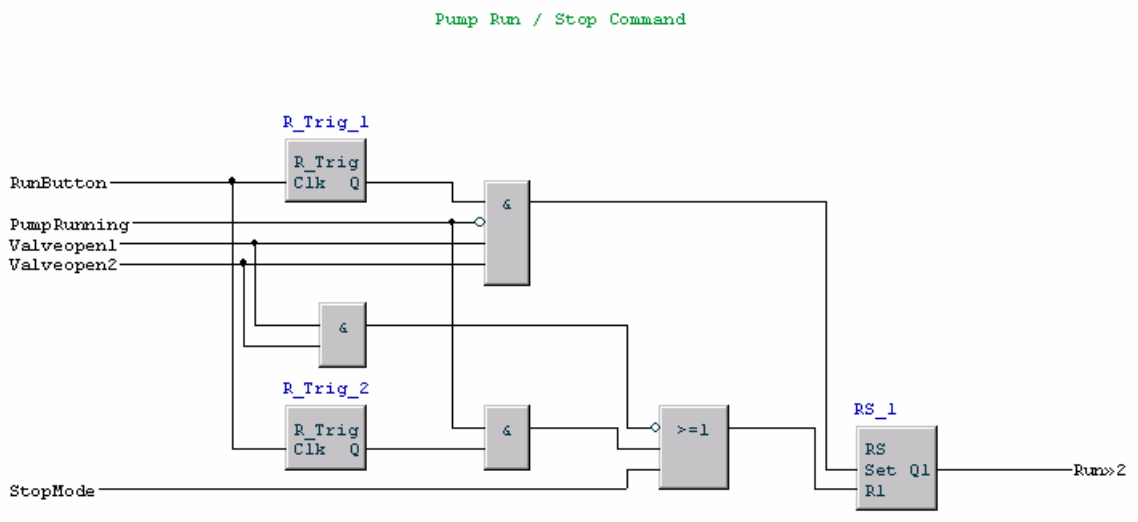

FIGURA 4.14

Si queremos encender los motores debemos seguir los siguientes pasos:

1. Dar la orden de encendido (Runbutton). Este parámetro significa tanto orden de encendido como apagado ,si está encendido y pulso apaga los motores y si está apagado y pulso los enciende,
2. una vez activada esta señal se genera un pulso,
3. si el motor esta parado y las válvulas de entrada (Valveopen1) y salida (Valveopen2) están abiertas se da orden de arrancar el motor (Run).

Para parar el motor se deben dar cualquiera de las siguientes causas:

- Que alguna de las válvulas esté cerrada
- Que este el sistema arrancado se dé la orden de apagado de motor

- Que el sistema esté parado en modo automático

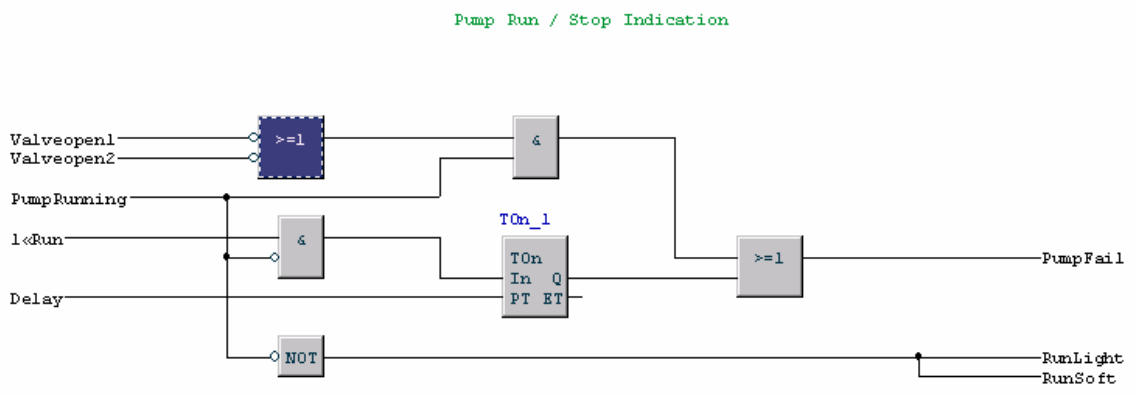

FIGURA 4.15

En la segunda pantalla, esperamos un tiempo una vez dada la orden de arranque (Run), si pasado ese tiempo no se ha arrancado el motor (PumpRunning) se da una alarma. Finalmente, tenemos dos parámetros que indican si está en funcionamiento en motor o no; uno es una señal visual en la pantalla del SCADA (Runlight) y otro mediante un parámetro que se utilizará en el PPA (RunSoft).

En este código usamos dos funciones (R_trig y TOn) que vamos a explicar más detalladamente. La primera, R_trig, genera un pulso a partir de un flanco de subida de la señal de entrada independientemente de la duración de ésta. La segunda, Ton, genera un "1" a la salida cuando ha pasado un tiempo (Delay) desde que se ha activado la entrada (In).

Comprobamos su AlarmCondM:

	Name	Data Type	Initial Valu	Parameter	Directio	Description
1	Signal	bool		PumpFail	in_out	IN Internal monitored Signal
2	ExtTimeStamp	bool	false		in_out	IN EDIT False = Signal is monitc
3	SignalID	string[20]	''		in_out	IN EDIT Identifier of the signal wi
4	UseSigToInit	bool	false		in_out	IN EDIT True = Use Signal to ge
5	SrcName	string[30]		Name	in_out	IN EDIT Name of the source has
6	Message	string[60]		'PUMP FAIL'	in_out	IN Description and/or informatior
7	Severity	dint	500		in_out	IN EDIT Severity of the alarm co
8	Class	dint	1		in_out	IN EDIT Class this alarm conditi
9	Inverted	bool	false		in_out	IN EDIT Tells if monitored Signal
10	AckRule	dint	1		in_out	IN EDIT Acknowledge rule. 1=No
11	FilterTime	time	0s		in_out	IN EDIT Positive pulses on Signa
12	EnDetection	bool	true		in_out	IN If true, the Signal is currently
13	AckCond	bool	false		in_out	IN Acknowledge alarm condition
14	DisCond	bool	false		in_out	IN Disable alarm condition on po
15	EnCond	bool	false		in_out	IN Enable alarm condition on po:
16	CondState	dint	Default		in_out	OUT Alarm condition state (0-6)
17	Error	bool	Default		in_out	OUT Indicates an error with True
18	Status	dint	Default		in_out	OUT Shows the status code of t

FIGURA 4.16

- **Simple_valve**:

Se utilizarán en las válvulas asociadas a los motores que poseen una lógica diferente a las válvulas que realizan la secuencia en el área CSS. Únicamente copian el valor de entrada (valvein) a la salida (valveout).

(Valve *)*

valveout := valvein;

- **Start:**

Esta función se encarga de la dinámica de arranque y parada de los motores y válvulas en el área CSS. Veamos el protocolo a seguir:

Cuando el sistema está encendido (apagado), la luz de encendido (apagado) se iluminará en el C&RP. Cuando el sistema esté apagado en modo automático, se abortará

la secuencia de muestreo y cerrará automáticamente todas las líneas. También se apagarán los motores.

Cuando el sistema esté encendido en modo automático, el sistema arrancará la secuencia de muestreo.

Si el sistema está en modo manual solo será posible accionar a mano las válvulas y motores desde el C&RP.

Cuando el modo automático este activo, las válvulas se abrirán y cerrarán secuencialmente por el PLC.

Cuando el selector pasa de automático a manual y el sistema está trabajando, la secuencia se abortará y la línea que esté siendo muestreada en ese momento continuará abierta hasta que el operario desee cerrarla.

Cuando el selector pase de manual a automático, el sistema de control cerrará todas las líneas y, posteriormente, comenzará la secuencia abriendo la primera línea a muestrear de la secuencia.

A continuación mostramos el código y los parámetros utilizados para realizar dicha función:

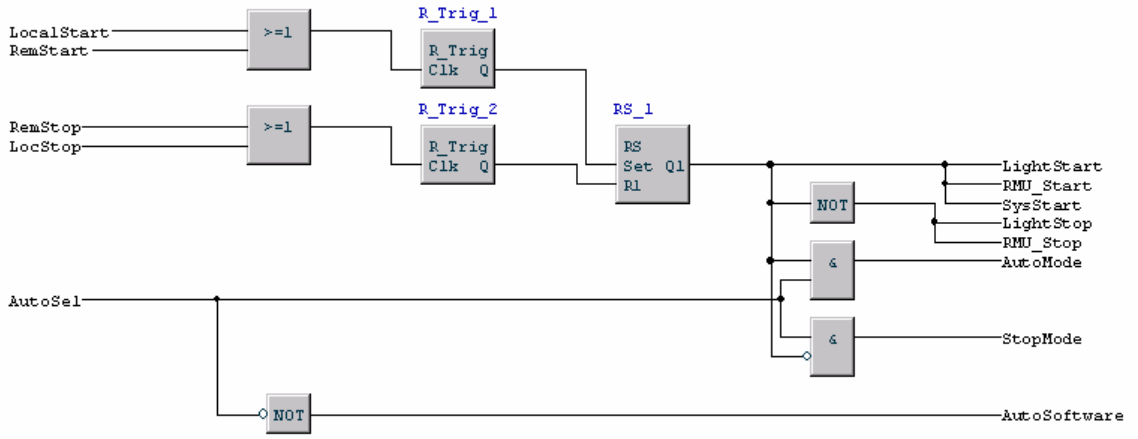

FIGURA 4.17

	Name	Data Type	Initial Value	Description
1	Name	string		
2	Description	string		
3	LocalStart	bool		Local System Start command
4	RemStart	bool		Remote System Start command
5	LocStop	bool		Local System Stop command
6	RemStop	bool		Remote System Stop command
7	LightStart	bool		System Start Light Indication
8	LightStop	bool		System Stop Light Indication
9	RMU_Start	bool		System Start wired to RMU Foxboro
10	RMU_Stop	bool		System Stop wired to RMU Foxboro
11	SysStart	bool		Sofware System Start signal
12	AutoSel	bool		Automatic / Manual Selector
13	AutoMode	bool		System started in Automatic Mode
14	StopMode	bool		System stoped in Automatic Mode
15	AutoSoftware	bool		Automatic software output

FIGURA 4.18

- **TemperatureSwicth**:

Su único cometido el activar una señal visual en la pantalla del SCADA en caso de que se exceda un límite de temperatura, generando a la vez una alarma. (TS_output= TS_input)

(TS activation *)*

TS_output := TS_input;

Veamos el AlarmCondM asociado al sensor:

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Signal	bool		TS_Output	in_out	IN Internal monitored Signal
2	ExtTimeStamp	bool	false		in_out	IN EDIT False = Signal is monitored; True = SignalID is monitored with external time stamp
3	SignalID	string[20]	''		in_out	IN EDIT Identifier of the signal with external time stamp. Valid when ExtTimeStamp = True
4	UseSigToInit	bool	false		in_out	IN EDIT True = Use Signal to get an init value. Valid when ExtTimeStamp = True
5	SrcName	string[30]		Name	in_out	IN EDIT Name of the source has to be unique together with the condition name
6	Message	string[60]		'TS ACTED'	in_out	IN Description and/or information about the alarm condition
7	Severity	dint	500		in_out	IN EDIT Severity of the alarm condition. From 1 (low) to 1000 (high)
8	Class	dint	1		in_out	IN EDIT Class this alarm condition belongs to. From 1 to 9999
9	Inverted	bool	false		in_out	IN EDIT Tells if monitored Signal is interpreted inverted
10	AckRule	dint	1		in_out	IN EDIT Acknowledge rule. 1=Normal ack. 2=No ack., 3=Ack. reset condition state
11	FilterTime	time	0s		in_out	IN EDIT Positive pulses on Signal shorter than this will not be noted (0-3600s).Valid when ExtTimeStamp = Fals
12	EnDetection	bool	true		in_out	IN If true, the Signal is currently being checked
13	AckCond	bool	false		in_out	IN Acknowledge alarm condition on positive edge
14	DisCond	bool	false		in_out	IN Disable alarm condition on positive edge
15	EnCond	bool	false		in_out	IN Enable alarm condition on positive edge
16	CondState	dint	Default	Ack_scada	in_out	OUT Alarm condition state (0-6)
17	Error	bool	Default		in_out	OUT Indicates an error with True
18	Status	dint	Default		in_out	OUT Shows the status code of the last execution

FIGURA 4.19

- **Valve**:

Estás válvulas son las que se utilizarán en el área CSS, deben poseer toda la gestión relacionada con los modos manual y automático. Observemos los bloques para luego explicarlos:

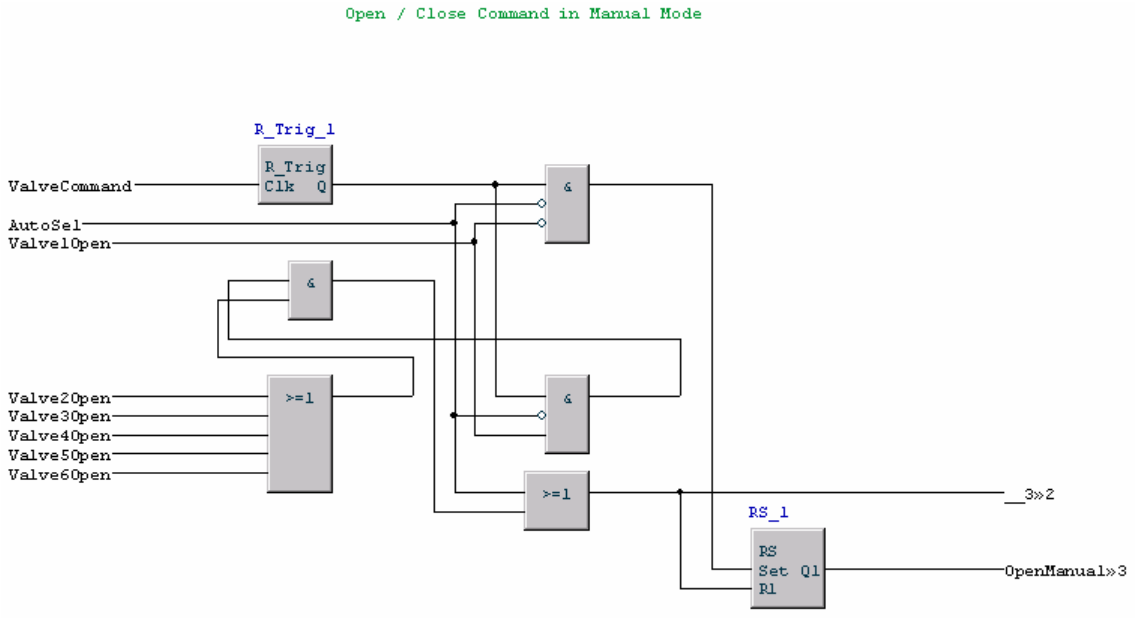

FIGURA 4.20

En esta figura vemos los comandos de apertura y cierre en modo manual. Para abrir la válvula debe estar el sistema en modo manual, la válvula cerrada y darse la

orden de apertura. Para cerrar la válvula es más complejo, puede deberse a dos condiciones: que el sistema esté en automático o que estando en modo manual se dé la orden de cerrar y esté abierta la válvula (hay que tener en cuenta que no se pueden tener cerradas todas las válvulas ya que aumentaría la presión en las líneas al seguir el flujo de líquido y no tener por donde continuar y, además, dañaría las bombas que existen después al no tener líquido que bombear)

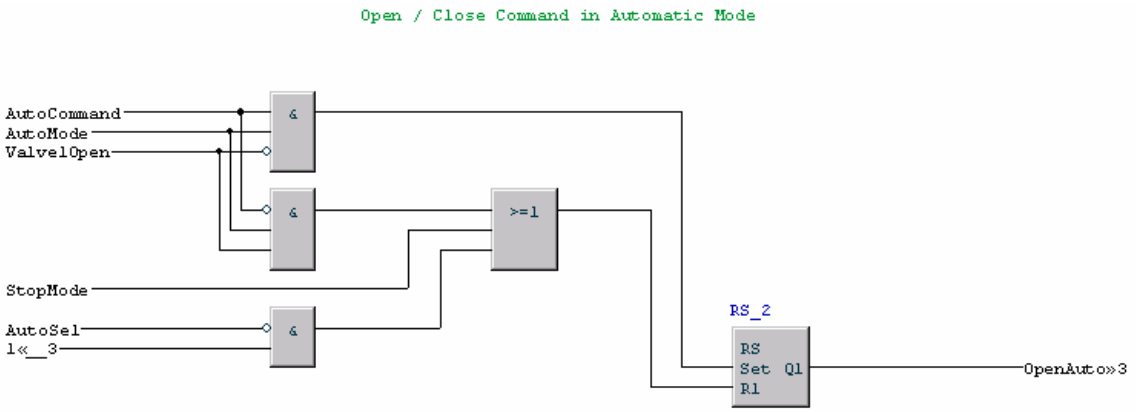

FIGURA 4.21

En esta figura vemos los comandos de apertura y cierre en modo automático. Si está en modo automático, la válvula está cerrada y se da el comando de apertura, se abre la válvula. Para cerrar la válvula es más complejo, puede deberse a tres condiciones: que el sistema esté apagado en modo automático; que estando en modo automático se dé la orden de cerrar y esté abierta la válvula o que estando en modo manual se haya dado la orden de cierre (este último caso se debe porque si estoy en modo automático y hay una válvula abierta, la orden de apertura automática se queda activada aún cuando pasa a manual y debo desactivarla, pero ya no puedo desde modo automático porque estoy en modo manual)

La última figura nos muestra el comando final de apertura o cierre así como las indicaciones que provoca.

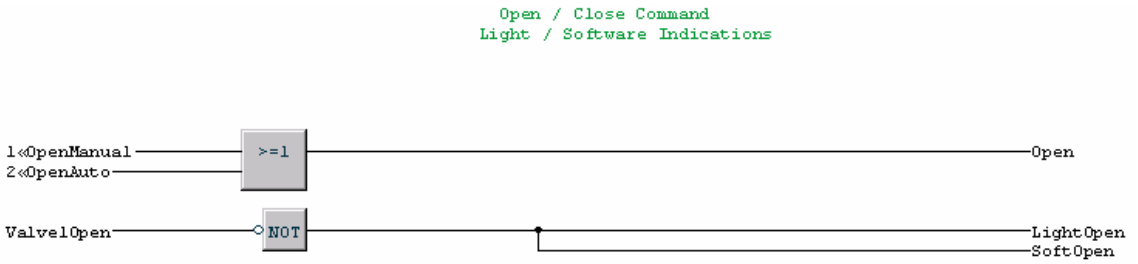

FIGURA 4.22

4.1.3 Aplicaciones

Ya hemos visto todos los bloques funcionales incluidos en nuestra librería LungmenLib. Estas funciones son genéricas, es decir, ahora hay que definirlas para cada una las válvulas, motores, sensores de temperatura, etc. Tenemos que indicar cada una de las salidas y entradas asociadas que, posteriormente, irán a las bornas de los módulos de entrada y salida. Por lo que abandonamos la carpeta Libraries y nos adentramos en la carpeta Applications.

Para realizar esta tarea tenemos que irnos a *Applications/<TSP_Application>/Control Modules* (El texto situado entre < > define cada una de las áreas del proyecto. En nuestro caso TSP_Application, CPSP_Application, ABSP_Application y CSP_Application). En *Control Modules* vamos añadiendo los bloques que deseemos asociándoles un nombre específico.

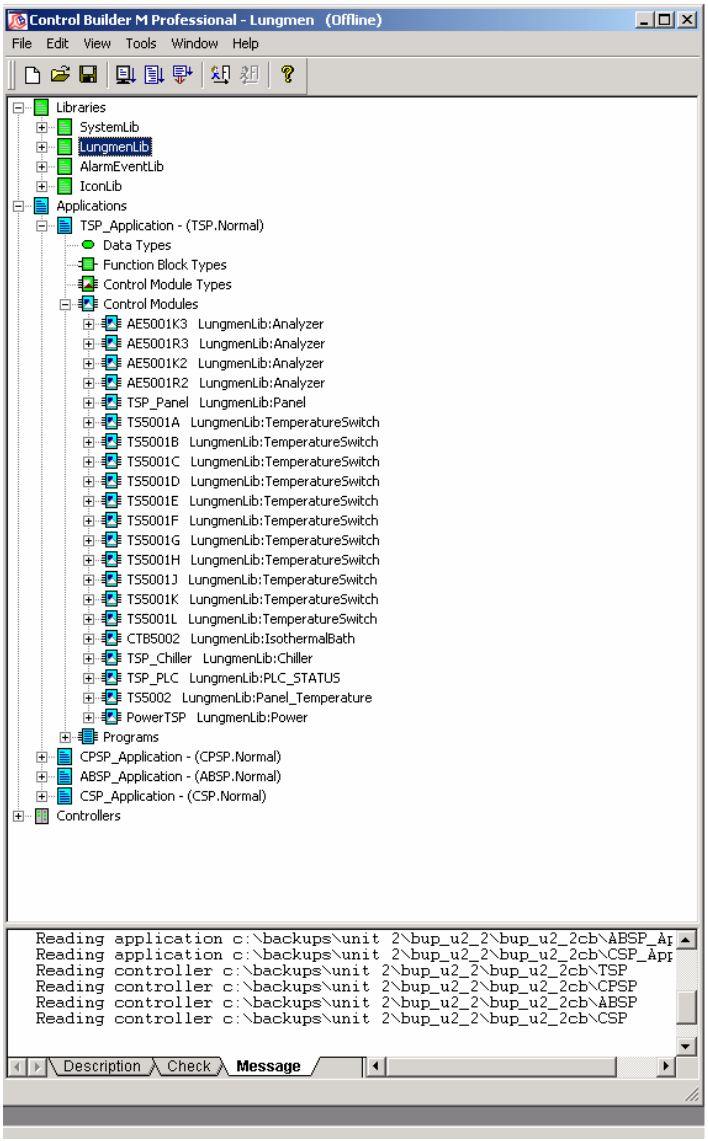

FIGURA 4.23

Para no cansar al lector, mostraremos de cada uno de los bloques funcionales solo un ejemplo.

- **Analyzer 2GG2-AE-5001K3:**

Observamos en la figura 4.24 que son una serie de columnas de las cuales solo la cuarta (*parámetros*) la podemos rellenar.

Describamos las seis columnas: la primera columna indica el nombre con que hemos definido las variables en los bloques funcionales descritos en los párrafos anteriores; la segunda nos dice el formato de las variables(cadena de caracteres, binarias, reales, etc.) ; la tercera nos informa si hemos dado valores iniciales a las variables; la cuarta y más importante es donde escribimos el parámetro de la base de datos asociado con la variable genérica del bloque funcional ; la quinta nos define si las variables son de entrada, salida o bidireccional y la sexta es una breve descripción de la variable.

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Name	string		'2G62-AE-5001K3'	in_out	
2	Description	string		'UNIT2 TSP FEEDWATER SPECIFIC COND. ANALYZER'	in_out	
3	Measure_input	real		A2G62AE5001K3	in_out	Analyzer measure
4	AlarmValue	real		A2G62OS5001K3	in_out	Measure value which enables the alarm
5	Measure_output	real		P2G62AE5001K3A	in_out	Analyzer measure sent to the SCADA
6	MaxValueAlarm	bool		O2G62AE5001K3B	in_out	High concentration alarm
7	AnalyzerFault_In	bool		I2G62AE5001K3	in_out	Analyzer Fault Input
8	AnalyzerFault_O	bool		O2G62AE5001K3C	in_out	Analyzer Fault Output
9	Unit	string		'µS/cm'	in_out	Analyzer Unit
10	MaxRange	real		A2G62OS5001K3MX	in_out	Normal Range Max. Value
11	MinRange	real		A2G62OS5001K3MN	in_out	Normal Range Min. Value
12	mnHW	real		MN0	in_out	
13	mxHW	real		MX19	in_out	

FIGURA 4.24

Las cadenas de caracteres o “strings” nos permiten describir el bloque o alguna variable y no tiene que estar asociada a la base de datos.

- **Panel 2G62-SPL-5002:**

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Name	string		'2G62-SPL-5002'	in_out	
2	Description	string		'UNIT2 TSP CONTROL AND RECORDER PANEL TEMPERATURE'	in_out	
3	Temp_input	real		A2G62TT5002	in_out	Control Panel Temperature input
4	Temp_output	real		P2G62TT5002A	in_out	Control Panel Temperature output
5	Door_input	bool		I2G62FB5002	in_out	Door Panel open input
6	Door_output	bool		O2G62FB5002A	in_out	Door Panel open output

FIGURA 4.25

- **TemperatureSwitch 2G62-TS-5001A:**

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Name	string		'2G62-TS-5001A'	in_out	
2	Description	string		'UNIT2 TSP MSR 1 LOW PRESSURE TURBINE'	in_out	
3	TS_input	bool		I2G62TS5001A	in_out	TS activation Input
4	TS_Output	bool		O2G62TS5001AB	in_out	TS activation Output
5	Ack_scada	dint		P2G62TS5001A	in_out	

FIGURA 4.26

- **Isothermal Bath 2G62-CTB-5002:**

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Name	string		'2G62-CTB-5002'	in_out	
2	Description	string		'UNIT2 TSP ISOTHERMAL BATH'	in_out	
3	LowLevel_input	bool		I2G62CTB5002	in_out	Low Level Isothermal Bath Input
4	LowLevel_output	bool		O2G62CTB5002B	in_out	Low Level Isothermal Bath Output

FIGURA 4.27

- **Chiller 2G62-CHU-5003:**

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Name	string		'2G62-CHU-5003'	in_out	
2	Description	string		'UNIT2 TSP CHILLER TROUBLE'	in_out	
3	Trouble_input	bool		I2G62CHU5003	in_out	Chiller Trouble Input
4	Trouble_output	bool		O2G62CHU5003B	in_out	Chiller Trouble Output
5	Trouble_output_H	bool		O2G62CHU5003C	in_out	Hardware Foxboro RMU Chiller Trouble Output

FIGURA 4.28

- **PLC Status:**

	Name	Data Type	Initial Valu	Parameter	Directio	Description
1	Name	string		'PLC_STATUS'	in_out	
2	Description	string		'PLC_STATUS'	in_out	
3	trouble	bool		O2G62TSPTRBB	in_out	
4	estado	HwStatus		AllUnitStatus	in_out	
5	trouble_h	bool		O2G62TSPTRBC	in_out	
6	Card1	HwStatus		AI810_1	in_out	Card 1 Hardware Status
7	Card2	HwStatus		DI810_1	in_out	Card 2 Hardware Status
8	Card3	HwStatus		DI810_2	in_out	Card 3 Hardware Status
9	Card4	HwStatus		DO810_1	in_out	Card 4 Hardware Status
10	Card5	HwStatus		DO810_2	in_out	Card 5 Hardware Status
11	Card6	HwStatus		DO820_1	in_out	Card 6 Hardware Status
12	Card7	HwStatus		CardSpare	in_out	Card 7 Hardware Status
13	PLCStatus	HwStatus		PLCStatus	in_out	

FIGURA 4.29

- **Panel Temperature 2G62-TS-5002:**

	Name	Data Type	Initial Valu	Parameter	Directio	Description
1	Name	string		'2G62-TS-5002'	in_out	
2	Description	string		'UNIT2 TSP CONTROL PANEL TEMPERATURE'	in_out	
3	TS_input	bool		I2G62TS5002	in_out	TS activation Input
4	TS_Output	bool		O2G62TS5002B	in_out	TS activation Output
5	Ack_scada	dint		P2G62TS5002	in_out	

FIGURA 4.30

- **Power:**

	Name	Data Type	Initial Valu	Parameter	Directio	Description
1	Name	string		'TSP_POWER_AVAILABLE'	in_out	
2	Description	string		'TSP POWER AVAILABLE'	in_out	
3	SCR120in	bool		I2G62RELE5001A	in_out	Sample Conditioning Rack 120V fault
4	SCR480in	bool		boolsparein	in_out	Sample Conditioning Rack 480V fault
5	AP120in	bool		I2G62RELE5001B	in_out	Analyzer Panel 120V fault
6	CRP120in	bool		I2G62RELE5002	in_out	Control and Recorder panel 120V fault
7	Powerout	bool		PowerAvaTSP	in_out	Power Avalilable
8	Powerout_h	bool		O2G62TSPPWRB	in_out	Power Available
9	SCR120out	bool		O2G62RELE5001A	in_out	Sample Conditioning Rack 120V fault
10	SCR480out	bool		boolspareout	in_out	Sample Conditioning Rack 480V fault
11	AP120out	bool		O2G62RELE5001B	in_out	Analyzer Panel 120V fault
12	CRP120out	bool		O2G62RELE5002	in_out	Control and Recorder panel 120V fault

FIGURA 4.31

- **Start:**

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Name	string		'UNIT2 CSP'	in_out	
2	Description	string		'SYSTEM OPERATION'	in_out	
3	LocalStart	bool		I2G62SSBSS5011A	in_out	Local System Start command
4	RemStart	bool		I1FXRMUSTART	in_out	Remote System Start command
5	LocStop	bool		I2G62SSBSS5011B	in_out	Local System Stop command
6	RemStop	bool		I1FXRMUSTOP	in_out	Remote System Stop command
7	LightStart	bool		O2G62SYSSTRA	in_out	System Start Light Indication
8	LightStop	bool		O2G62SYSSTPA	in_out	System Stop Light Indication
9	RMU_Start	bool		O2G62SYSSTRB	in_out	System Start wired to RMU Foxboro
10	RMU_Stop	bool		O2G62SYSSTPB	in_out	System Stop wired to RMU Foxboro
11	SysStart	bool		O2G62SYSSTRC	in_out	Sofware System Start signal
12	AutoSel	bool		I2G62MAS5011A	in_out	Automatic / Manual Selector
13	AutoMode	bool		AutomaticMode	in_out	System started in Automatic Mode
14	ManMode	bool		ManualMode	in_out	System started in Manual Mode
15	AutoSoftware	bool		O2G62MAS5011A	in_out	Automatic software output

FIGURA 4.32

- **Valve 2G62-SBV-5295:**

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Name	string		'2G62-SBV-5295'	in_out	
2	Description	string		'UNIT2 CSP COND HOTWELL SHELL A OUTLET WEST END FB '	in_out	
3	SystemStarted	bool		O2G62SYSSTRC	in_out	System Started
4	AutoMode	bool		AutomaticMode	in_out	System started in Automatic Mode
5	ManMode	bool		ManualMode	in_out	System started in Manual Mode
6	ValveCommand	bool		I2G62HOTB5011A	in_out	Open / Close valve command
7	AutoCommand	bool		PLS1	in_out	Open / Close Auto valve command
8	Valve1Open	bool		I2G62SBV5295	in_out	Valve to be controlled
9	Valve2Open	bool		I2G62SBV5297	in_out	Valve 2 open
10	Valve3Open	bool		I2G62SBV5299	in_out	Valve 3 open
11	Valve4Open	bool		I2G62SBV5301	in_out	Valve 4 open
12	Valve5Open	bool		I2G62SBV5303	in_out	Valve 5 open
13	Valve6Open	bool		I2G62SBV5305	in_out	Valve 6 open
14	LightOpen	bool		O2G62SBV5295A	in_out	Valve open Light indication
15	SoftOpen	bool		O2G62SBV5295B	in_out	Valve open Software indication
16	Open	bool		O2G62SBV5295	in_out	Valve Open command

FIGURA 4.33

- **Pump 2G62-P-5010A:**

	Name	Data Type	Initial Valu	Parameter	Direction	Description
1	Name	string		'2G62-P-5010A'	in_out	
2	Description	string		'UNIT2 CSP PUMP'	in_out	
3	RunButton	bool		I2G62PSS5011A	in_out	Stop / Run button command
4	PumpRunning	bool		I2G62P5010A	in_out	Pump running
5	Valveopen1	bool		I2G62V5306	in_out	Valve 1 open
6	Valveopen2	bool		I2G62V5307	in_out	Valve 2 open
7	SysStart	bool		O2G62SYSSTRC	in_out	System Started
8	Run	bool		O2G62P5010AA	in_out	Pump run command
9	PumpFail	bool		O2G62P5011AFLB	in_out	Pump Fail. Software signal
10	RunLight	bool		O2G62PSL5011AA	in_out	Light run indication
11	RunSoft	bool		O2G62PSL5011AB	in_out	Run indication. Software signal

FIGURA 4.34

- **Simple_ valve 2G62-SBV-5312:**

	Name	Data Type	Initial Valu	Parameter	Directio	Description
1	Name	string		'2G62-SBV-5312'	in_out	
2	Description	string		'UNIT2 CSP 2G62-P-5010B INPUT FB VLV OPEN'	in_out	
3	valvein	bool		I2G62V5312	in_out	valve in
4	valveout	bool		P2G62V5312	in_out	valve out

FIGURA 4.35

Finalmente, solo nos queda explicar de la carpeta *Applications*, la subcarpeta *Programs*. En *Programs* tenemos programas que se utilizan de manera diferente a los "Control Module". Funcionan con "Function Block Types" y se suelen utilizar para tareas simples que se ejecutan solo cuando son llamadas. Tenemos tres subcarpetas como vemos en la siguiente figura 4.36:

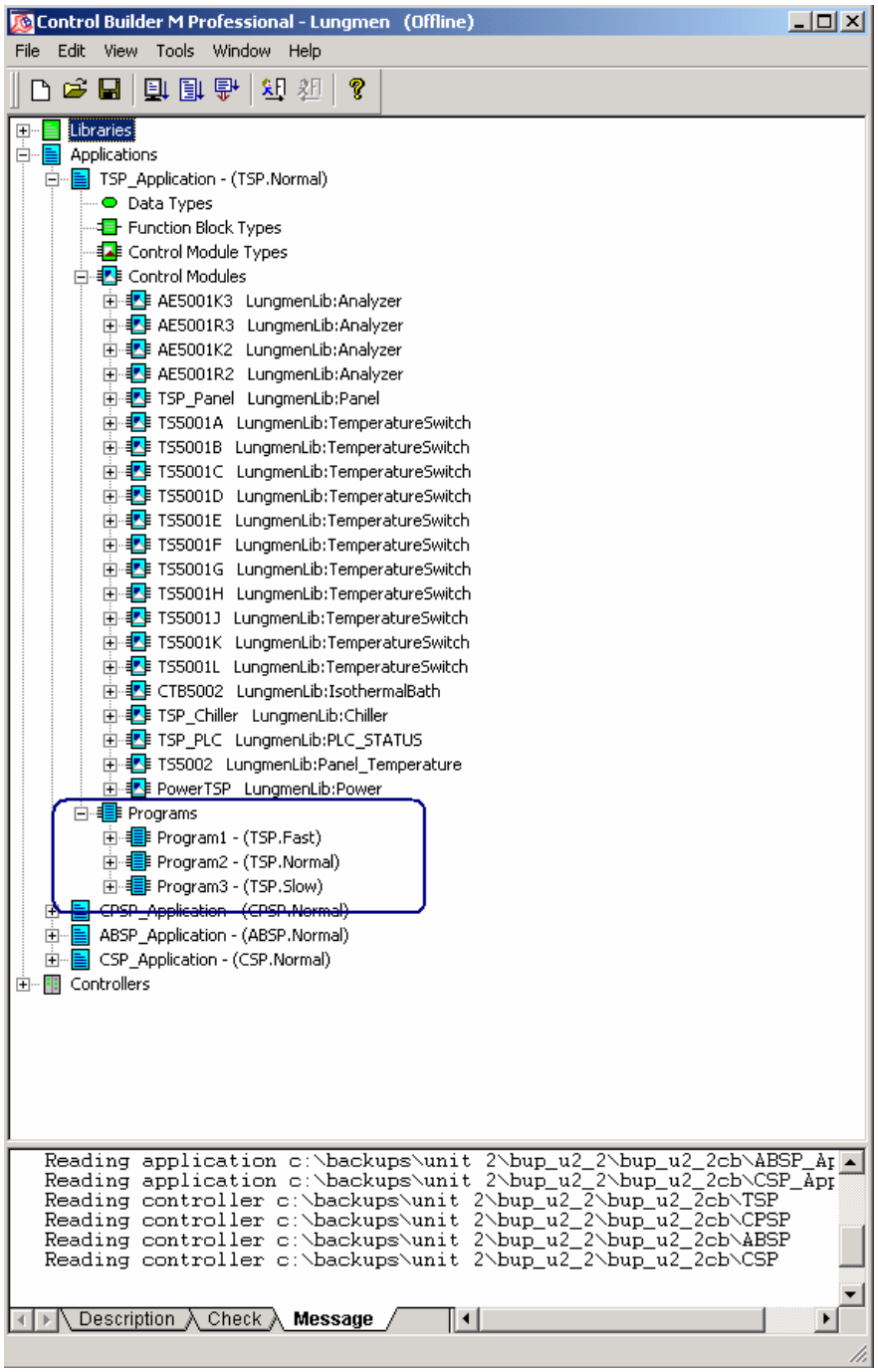

FIGURA 4.36

La diferencia entre ellas es la velocidad de ejecución (fast, normal y slow). Dichas velocidades pueden ser definidas por el programador. Nosotros hemos definido fast =50 ms , normal = 250 ms y slow = 1000 ms.

En la ejecución normal hemos introducido la gestión de los AlarmAnnuciator, vemos el código:

```
(* Alarm Annunciator alarms *)

(* 2G62-TS-5001A *)

AlarmAnnunciator1( Alarm_input := I2G62TS5001A ,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5001AA,
        Sound_output => SOUND1 );

(* 2G62-TS-5001B *)

AlarmAnnunciator2( Alarm_input := I2G62TS5001B,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5001BA,
        Sound_output => SOUND2 );

(* 2G62-TS-5001C *)

AlarmAnnunciator3( Alarm_input := I2G62TS5001C ,
        Alarm_Ack := I2G62BUT5002B,
```

```
        Alarm_Reset := I2G62BUT5002A,
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5001CA,
        Sound_output => SOUND3 );

(* 2G62-TS-5001D *)

AlarmAnnunciator4( Alarm_input := I2G62TS5001D,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5001DA,
        Sound_output => SOUND4 );

(* 2G62-TS-5001E *)

AlarmAnnunciator5( Alarm_input := I2G62TS5001E,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5001EA,
        Sound_output => SOUND5 );

(* 2G62-TS-5001F *)

AlarmAnnunciator6( Alarm_input := I2G62TS5001F,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
```

```
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5001FA,
        Sound_output => SOUND6 );

(* 2G62-TS-5001G *)

AlarmAnnunciator7( Alarm_input := I2G62TS5001G,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5001GA,
        Sound_output => SOUND7 );

(* 2G62-TS-5001H *)

AlarmAnnunciator8( Alarm_input := I2G62TS5001H,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5001HA,
        Sound_output => SOUND8 );


(* 2G62-TS-5001J *)

AlarmAnnunciator9( Alarm_input := I2G62TS5001J,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
```

```
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5001JA,
        Sound_output => SOUND9 );


(* 2G62-TS-5001K *)

AlarmAnnunciator10( Alarm_input := I2G62TS5001K,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5001KA,
        Sound_output => SOUND10 );

(* 2G62-TS-5001L *)

AlarmAnnunciator11( Alarm_input := I2G62TS5001L,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5001LA,
        Sound_output => SOUND11 );

(* 2G62-TS-5002 *)

AlarmAnnunciator12( Alarm_input := O2G62TS5002B,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
```

```
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TS5002A,
        Sound_output => SOUND12 );

(* 2G62-CTB-5002 *)

AlarmAnnunciator13( Alarm_input := O2G62CTB5002B,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62CTB5002A,
        Sound_output => SOUND13 );

(* PLC TROUBLE *)

AlarmAnnunciator14( Alarm_input := O2G62TSPTRBB,
        Alarm_Ack := I2G62BUT5002B,
        Alarm_Reset := I2G62BUT5002A,
        Lamp_Test := I2G62BUT5002C,
        Alarm_output => O2G62TSPTRBA,
        Sound_output => SOUND14 );



(* TSP POWER AVAILABLE *)

AlarmAnnunciator22( Alarm_input :=PowerAvaTSP ,
        Alarm_Ack := I2G62BUT5002B,
```

```
            Alarm_Reset := I2G62BUT5002A,
            Lamp_Test := I2G62BUT5002C,
            Alarm_output => P2G62TSPPWRA,
            Sound_output => SOUND22 );

O2G62TSPPWRA:= not P2G62TSPPWRA;

(* 2G62-CHU-5003 *)

AlarmAnnunciator15( Alarm_input := O2G62CHU5003B,
            Alarm_Ack := I2G62BUT5002B,
            Alarm_Reset := I2G62BUT5002A,
            Lamp_Test := I2G62BUT5002C,
            Alarm_output => O2G62CHU5003A,
            Sound_output => SOUND15 );


(* Analyzers Fail *)

AnalyzerFail := (not I2G62AE5001K3) or (not I2G62AE5001R3) or (not I2G62AE5001K2)
          or (not I2G62AE5001R2);
AlarmAnnunciator16( Alarm_input := AnalyzerFail,
            Alarm_Ack := I2G62BUT5002B,
            Alarm_Reset := I2G62BUT5002A,
            Lamp_Test := I2G62BUT5002C,
            Alarm_output => O2G62AE5001FLA,
            Sound_output => SOUND16 );
```

```
(* Analyzer AE5001K3 High Concentration *)

AlarmAnnunciator18( Alarm_input := O2G62AE5001K3B,
                    Alarm_Ack := I2G62BUT5002B,
                    Alarm_Reset := I2G62BUT5002A,
                    Lamp_Test := I2G62BUT5002C,
                    Alarm_output => O2G62AE5001K3A,
                    Sound_output => SOUND18 );


(* Analyzer AE5001R3 High Concentration *)

AlarmAnnunciator19( Alarm_input := O2G62AE5001R3B,
                    Alarm_Ack := I2G62BUT5002B,
                    Alarm_Reset := I2G62BUT5002A,
                    Lamp_Test := I2G62BUT5002C,
                    Alarm_output => O2G62AE5001R3A,
                    Sound_output => SOUND19 );


(* Analyzer AE5001K2 High Concentration *)

AlarmAnnunciator20( Alarm_input := O2G62AE5001K2B,
                    Alarm_Ack := I2G62BUT5002B,
                    Alarm_Reset := I2G62BUT5002A,
                    Lamp_Test := I2G62BUT5002C,
                    Alarm_output => O2G62AE5001K2A,
```

Sound_output => SOUND20);

(Analyzer AE5001R2 High Concentration *)*

AlarmAnnunciator21(Alarm_input := O2G62AE5001R2B,

Alarm_Ack := I2G62BUT5002B,

Alarm_Reset := I2G62BUT5002A,

Lamp_Test := I2G62BUT5002C,

Alarm_output => O2G62AE5001R2A,

Sound_output => SOUND21);

(Audible Annunciation *)*

O2G62AA5002:= SOUND1 or SOUND2 or SOUND3 or SOUND4 or SOUND5 or SOUND6
or SOUND7 or SOUND8 or SOUND9 or SOUND10 or SOUND11 or SOUND12
or SOUND13 or SOUND14 or SOUND15 or SOUND16 or SOUND17 or SOUND18
or SOUND19 or SOUND20 or SOUND21 or SOUND22;

En la ejecución lenta se encuentran una serie de programas que están por defecto y sirven para el diagnóstico y definición parámetros internos. Se muestran en la figura 4.37 y en la ejecución rápida no tenemos ninguna aplicación insertada.

```
(* System support *)

PowerFailureInfos( NoOfPFSinceRst => NoOfPFSinceRst,
                   DurOfPFSinceRst => DurOfPFSinceRst,
                   OccOfRst => OccOfRst,
                   NoOfPFAtLastOcc => NoOfPFAtLastOcc,
                   DurOfPFAtLastOcc => DurOfPFAtLastOcc,
                   OccOfLastPF => OccOfLastPF );

SystemDiagnostics( QuotaExc => QuotaExc,
                   DownLoadQuotaExc => DownLoadQuotaExc);

SetTime( );


SetTimeZone( );

(* *);
```

FIGURA 4.37

4.1.4 Controladores

Con este último punto ya hemos descrito la segunda carpeta principal Applications, ahora tan solo nos queda explicar la tercera carpeta Controllers que contiene información de carácter HW como número de módulos que forman cada área, configuraciones de puertos, direcciones IP, etc.

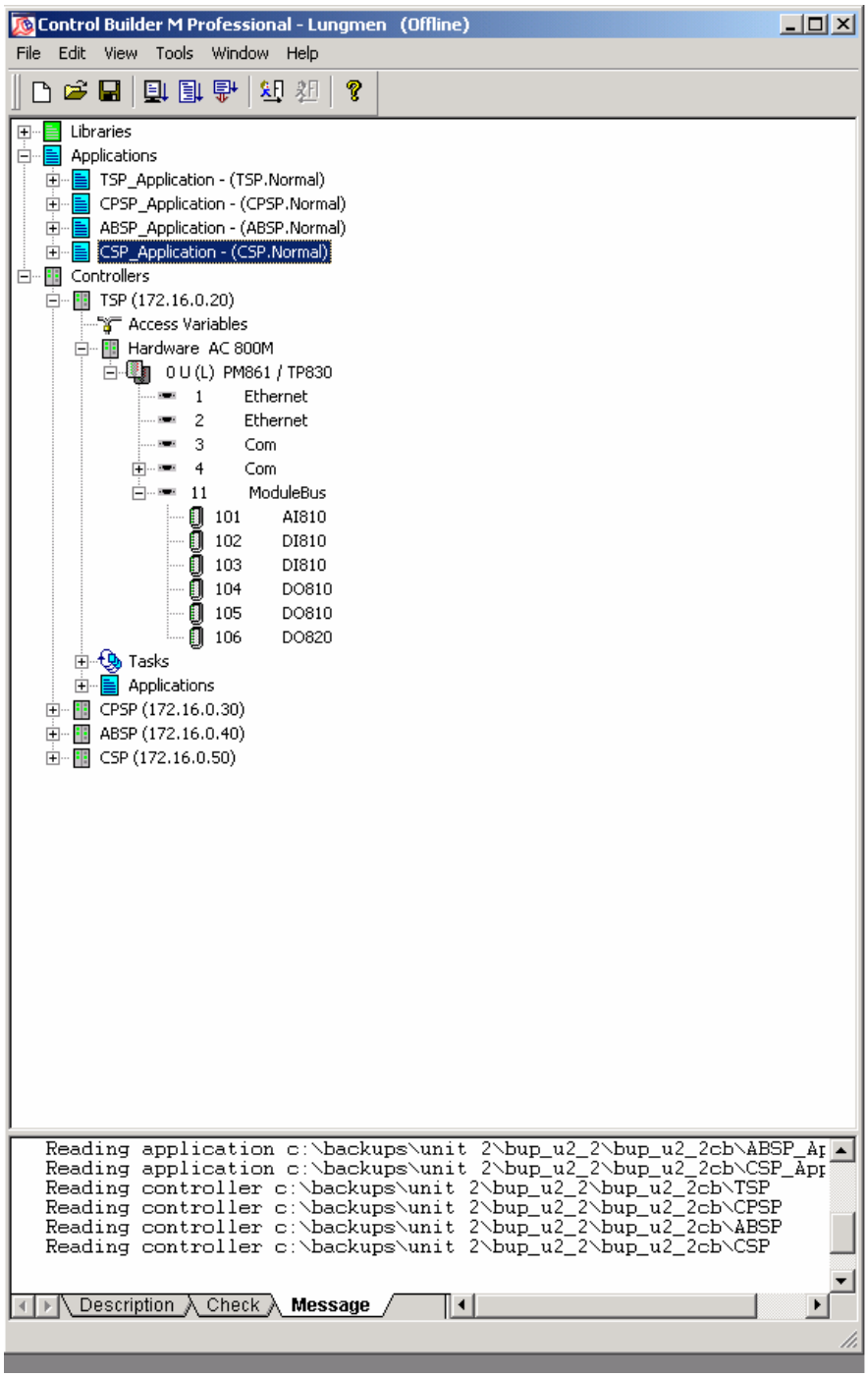

FIGURA 4.38

Como vemos cada área posee su propia carpeta (con su propia dirección IP), de entre las subcarpetas que posee cada área la más importante es HW AC 800M. Esta

carpeta tiene la arquitectura que posee cada área así como todos los parámetros que necesitemos configurar.

Como muestra estudiemos *Controllers/TSP(172.16.0.20)/Harwdware AC 800M* ya que las demás áreas son exactamente iguales variando los módulos de entrada/salida que tienen y sus direcciones IP. Si pulsamos dos veces sobre el icono aparece la siguiente pantalla que posee dos partes importantes:

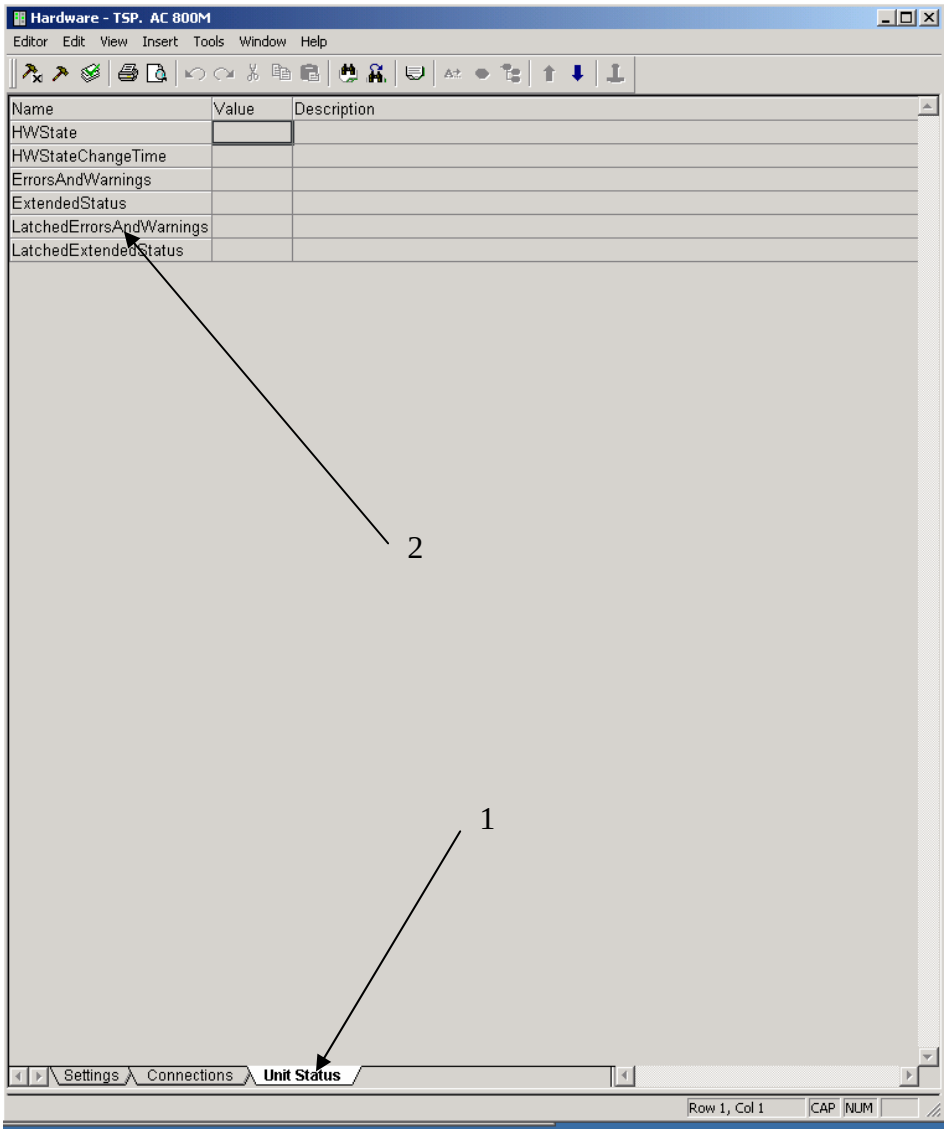

FIGURA 4.39

1.-Pestañas: Para acceder a las distintas páginas que forman esta pantalla (Setting, Connections, Unit Status)

2.-Parámetros: Son los parámetros a configurar o visualizar

En la pestaña Settings podemos configurar parámetros como el ciclo de tiempo de actualización de HwStatus(HWStatus contiene información sobre el conjunto del HW del área), habilitación de alarmas y eventos,…

Parameter	Value	Type	Unit	Min	Max
Copy unconnected channels	None	enum			
Latched hardware state	Enabled	enum			
System alarms on hardware unit	Enabled	enum			
Simple events on hardware units	Enabled	enum			
HwStatus update cycle time	1000	dint	ms	500	30000

FIGURA 4.40

En la pestaña Connections asignamos una variable de la Base de Datos para guardar la información sobre el conjunto del HW del área.

Channel	Name	Type	Variable	I/O Description
IW0	AllUnitStatus	dint	TSP_Application.AllUnitStatus.HwState	

FIGURA 4.41

En la pestaña Unit Status podemos visualizar los parámetros más importantes relacionados con el estado y las alarmas cuando estemos en On-line, lo cual nos dará una información vital a la hora de depurar el código o detectar fallos HW.

Name	Value	Description
HWState		
HWStateChangeTime		
ErrorsAndWarnings		
ExtendedStatus		
LatchedErrorsAndWarnings		
LatchedExtendedStatus		

FIGURA 4.42

Si continuamos desplegando carpetas la siguiente es *Controllers\TSP(172.16.0.20)\Harwdware AC 800M\PM861/TP860*. La pantalla principal posee tres pestañas: Setting, Connections y Unit Status.

En la pestaña Settings podemos configurar numerosos parámetros del PLC.

Parameter	Value	Type	Unit	Min	Max
Enable SattBus on TCP/IP	false	bool			
Routing method	rnrp	enum			
Default gateway		string			
System AE class	9950	dint		1	9999
System AE low severity	920	dint		1	1000
System AE medium severity	930	dint		1	1000
System AE high severity	940	dint		1	1000
System AE critical severity	950	dint		1	1000
System AE fatal severity	960	dint		1	1000
System events to controller log	Medium Severit	enum			
Event queue size	300	dint		10	3000
Max number of event queues	0	dint		0	5
Event subscription queue size	300	dint		200	3000
Max number of subscriptions	1	dint		0	5
Event subscription time out	360	dint	min	0	1440
Low level event buffer size	2	dint		2	4000
Ext undeclared event class	2	dint		1	9999
Ext undeclared event severity	200	dint		1	1000
RNRP Default network ID	172.16.0.0	string			
RNRP Number of own areas	2	dint		1	4
RNRP Number of remote areas	4	dint		0	8
RNRP Max Lost Messages	3	dint		1	10
RNRP Send Period	1	dint	s	1	60
RNRP Max no of hops	3	dint		1	3
Tool Routing	Disabled	enum			
Tool Routing max no of connecti	20	dint		5	50
CS CNCP ClockMaster Order Nu	0	dint		0	10
CS Protocol Type	CNCP	enum			
CS Time Set Enabled	True	enum			
CS Synch Interval	20	dint	s	1	240
CS SNTP ServerAddr 1		string			
CS SNTP ServerAddr 2		string			

FIGURA 4.43

En la pestaña Connections asignamos una variable de la Base de Datos para guardar la información sobre el HW del PLC.

Channel	Name	Type	Variable	I/O Description
IW0.0	UnitStatus	dint	TSP_Application.PLCStatus.HwState	

FIGURA 4.44

En la pestaña Unit Status podemos visualizar los parámetros más importantes relacionados con el estado y las alarmas cuando estemos en On-line.

Name	Value	Description
HWState		
HWStateChangeTime		
ErrorsAndWarnings		
ExtendedStatus		
LatchedErrorsAndWarnings		
LatchedExtendedStatus		

FIGURA 4.45

Si continuamos desplegando la carpeta (ver figura 4.38) nos aparecen las comunicaciones. De todas ellas nos interesan Ethernet y ModuleBus. Con el primer protocolo comunicamos el Nivel de Operación con el Nivel de Control; con el segundo comunicamos el Nivel de Control con el Nivel de Campo.

Vemos en detalle la configuración Ethernet. La pantalla principal posee tres pestañas: Setting, Connections, Unit Status.

En la pestaña Settings podemos configurar parámetros como la dirección IP, máscara de subred, etc.

Parameter	Value	Type	Unit	Min	Max
IP address	172.16.0.20	string			
IP subnet mask	255.255.252.0	string			
Network Area	0	dint		0	35
Path Number	0	dint		0	3
Node Number	20	dint		0	500
Network Area Local	false	bool			
Send Period	1	dint	s	1	60
Max Lost Messages	3	dint		1	10
Proxy router	0.0.0.0	string			
Target address	0.0.0.0	string			

FIGURA 4.46

En la pestaña Connections asignamos una variable de la Base de Datos para guardar la información sobre el HW del PLC.

Channel	Name	Type	Variable	I/O Description
IW0.1.0	UnitStatus	dint		

FIGURA 4.47

En la pestaña Unit Status podemos visualizar los parámetros más importantes relacionados con el estado y las alarmas cuando estemos en On-line.

Name	Value	Description
HWState		
HWStateChangeTime		
ErrorsAndWarnings		
ExtendedStatus		
LatchedErrorsAndWarnings		
LatchedExtendedStatus		

FIGURA 4.48

Finalmente, solo nos queda describir los módulos que utilizan ModuleBus. En el área TSS que hemos tomado como ejemplo tenemos: 1 AI810 (Entradas analógicas), 2 DI810 (Entradas digitales), 2 DO810 (Salidas digitales) y 1 DO820 (Salidas digitales).

Cada uno posee su propia configuración que vamos a detallar a continuación, tomaremos un ejemplo de cada tipo.

❖ Configuración ModuleBus de AI810:

La pantalla principal posee cinco pestañas: Settings, Connections, Scaling, Status y Unit Status.

La pestaña Settings como hemos visto ya en los párrafos anteriores define una serie de parámetros. En este caso si el canal está activado, filtros, rango de señal, etc.

Parameter	Value	Type	Unit	Min	Max
Activate channel 1	true	bool			
Activate channel 2	true	bool			
Activate channel 3	true	bool			
Activate channel 4	true	bool			
Activate channel 5	true	bool			
Activate channel 6	true	bool			
Activate channel 7	true	bool			
Activate channel 8	true	bool			
Filter time channel 1	0	dint	ms	0	65535
Filter time channel 2	0	dint	ms	0	65535
Filter time channel 3	0	dint	ms	0	65535
Filter time channel 4	0	dint	ms	0	65535
Filter time channel 5	0	dint	ms	0	65535
Filter time channel 6	0	dint	ms	0	65535
Filter time channel 7	0	dint	ms	0	65535
Filter time channel 8	0	dint	ms	0	65535
Linearization code channel 1	No Linearizatior	enum			
Linearization code channel 2	No Linearizatior	enum			
Linearization code channel 3	No Linearizatior	enum			
Linearization code channel 4	No Linearizatior	enum			
Linearization code channel 5	No Linearizatior	enum			
Linearization code channel 6	No Linearizatior	enum			
Linearization code channel 7	No Linearizatior	enum			
Linearization code channel 8	No Linearizatior	enum			
Signal range channel 1	4-20mA	enum			
Signal range channel 2	4-20mA	enum			
Signal range channel 3	4-20mA	enum			
Signal range channel 4	4-20mA	enum			
Signal range channel 5	4-20mA	enum			
Signal range channel 6	4-20mA	enum			
Signal range channel 7	4-20mA	enum			
Signal range channel 8	4-20mA	enum			
Deadband channel 1	Update every tir	enum	%		
Deadband channel 2	Update every tir	enum	%		
Deadband channel 3	Update every tir	enum	%		
Deadband channel 4	Update every tir	enum	%		
Deadband channel 5	Update every tir	enum	%		
Deadband channel 6	Update every tir	enum	%		
Deadband channel 7	Update every tir	enum	%		
Deadband channel 8	Update every tir	enum	%		

FIGURA 4.49

En la Pestaña Connections asignamos variables de la Base de Datos para guardar la información sobre las entradas analógicas del área.

Channel	Name	Type	Variable	I/O Description
IW0.11.101.1	Input 1	real	TSP_Application.A2G62AE5001K3	UNIT2 TSP FEEDWATER SPECIFIC COND. ANALYZER
IW0.11.101.2	Input 2	real	TSP_Application.A2G62AE5001R3	UNIT2 TSP CONDENSATE SPECIFIC COND. ANALYZER
IW0.11.101.3	Input 3	real	TSP_Application.A2G62AE5001K2	UNIT2 TSP FEEDWATER DISSOLVED O2 ANALYZER
IW0.11.101.4	Input 4	real	TSP_Application.A2G62AE5001R2	UNIT2 TSP CONDENSATE DISSOLVED O2 ANALYZER
IW0.11.101.5	Input 5	real	TSP_Application.A2G62TT5002	UNIT2 TSP CONTROL PANEL TEMPERATURE
IW0.11.101.6	Input 6	real	TSP_Application.SPARE_AI1	SPARE-AI1
IW0.11.101.7	Input 7	real	TSP_Application.SPARE_AI2	SPARE-AI2
IW0.11.101.8	Input 8	real	TSP_Application.SPARE_AI3	SPARE-AI3
IW0.11.101.9	UnitStatus	dint	TSP_Application.AI810_1.HwState	

FIGURA 4.50

La pestaña Scaling nos permite definir el rango de la entrada analógica, en *Fraction* podemos decidir el número de decimales

Channel	Variable	Min	Max	Unit	Fraction	Reversed
IW0.11.101.1	TSP_Application.A2G62AE5001K3	0.0	1.999		3	false
IW0.11.101.2	TSP_Application.A2G62AE5001R3	0.0	1.999		3	false
IW0.11.101.3	TSP_Application.A2G62AE5001K2	10.0	30.0		1	false
IW0.11.101.4	TSP_Application.A2G62AE5001R2	10.0	30.0		1	false
IW0.11.101.5	TSP_Application.A2G62TT5002	0.0	100.0		1	false
IW0.11.101.6	TSP_Application.SPARE_AI1	0.0	100.0		1	false
IW0.11.101.7	TSP_Application.SPARE_AI2	0.0	100.0		1	false
IW0.11.101.8	TSP_Application.SPARE_AI3	0.0	100.0		1	false

FIGURA 4.51

Status nos permite simular las variables cuando estemos en on-line. Tan solo tenemos que marcar el check-box de la columna *Forced* y escribir el la columna *Variable Value* es valor deseado.

Channel	Channel Value	Forced	Variable Value	Variable
IW0.11.101.1		☐		TSP_Application.A2G62AE5001K3
IW0.11.101.2		☐		TSP_Application.A2G62AE5001R3
IW0.11.101.3		☐		TSP_Application.A2G62AE5001K2
IW0.11.101.4		☐		TSP_Application.A2G62AE5001R2
IW0.11.101.5		☐		TSP_Application.A2G62TT5002
IW0.11.101.6		☐		TSP_Application.SPARE_AI1
IW0.11.101.7		☐		TSP_Application.SPARE_AI2
IW0.11.101.8		☐		TSP_Application.SPARE_AI3

FIGURA 4.52

En Unit Status visualizamos los parámetros más importantes relacionados con el estado y las alarmas cuando estemos en On-line.

Name	Value	Description
HWState		
HWStateChangeTime		
ErrorsAndWarnings		
ExtendedStatus		
LatchedErrorsAndWarnings		
LatchedExtendedStatus		

FIGURA 4.53

❖ Configuración ModuleBus de DI810:

En este caso la pantalla principal posee cuatro pestañas: Settings, Connections, Status y Unit Status.

Settings: Podemos definir si los canales están activos o no, filtros, etc.

Parameter	Value	Type	Unit	Min	Max
Filter time	2	enum	ms		
Sensor power supervision	true	bool			
Activate channel 1	true	bool			
Activate channel 2	true	bool			
Activate channel 3	true	bool			
Activate channel 4	true	bool			
Activate channel 5	true	bool			
Activate channel 6	true	bool			
Activate channel 7	true	bool			
Activate channel 8	true	bool			
Activate channel 9	true	bool			
Activate channel 10	true	bool			
Activate channel 11	true	bool			
Activate channel 12	true	bool			
Activate channel 13	true	bool			
Activate channel 14	true	bool			
Activate channel 15	true	bool			
Activate channel 16	true	bool			

FIGURA 4.54

En la Pestaña Connections asignamos variables de la Base de Datos para guardar la información sobre las entradas digitales del área.

Channel	Name	Type	Variable	I/O Description
IX0.11.102.1	Input 1	bool	TSP_Application.I2G62TS5001A	UNIT2 TSP MSR 1 LOW PRESSURE TURBINE TEMPERATURE HIGH
IX0.11.102.2	Input 2	bool	TSP_Application.I2G62TS5001B	UNIT2 TSP MSR 2 LOW PRESSURE TURBINE TEMPERATURE HIGH
IX0.11.102.3	Input 3	bool	TSP_Application.I2G62TS5001C	UNIT2 TSP MSR 3 LOW PRESSURE TURBINE TEMPERATURE HIGH
IX0.11.102.4	Input 4	bool	TSP_Application.I2G62TS5001D	UNIT2 TSP MSR A 1ST STAGE REHEATER DT A1,A2,B1,B2 TEMPERATURE HIGH
IX0.11.102.5	Input 5	bool	TSP_Application.I2G62TS5001E	UNIT2 TSP MSR A 2ND STAGE REHEATER DT A1,A2 TEMPERATURE HIGH
IX0.11.102.6	Input 6	bool	TSP_Application.I2G62TS5001F	UNIT2 TSP MSR A 2ND STAGE REHEATER DT B1,B2 TEMPERATURE HIGH
IX0.11.102.7	Input 7	bool	TSP_Application.I2G62TS5001G	UNIT2 TSP MSR A SEPARATOR DT TEMPERATURE HIGH
IX0.11.102.8	Input 8	bool	TSP_Application.I2G62TS5001H	UNIT2 TSP MSR B SEPARATOR DT TEMPERATURE HIGH
IX0.11.102.9	Input 9	bool	TSP_Application.I2G62TS5001J	UNIT2 TSP GLAND STEAM SUPPLY TEMPERATURE HIGH
IX0.11.102.10	Input 10	bool	TSP_Application.I2G62TS5001K	UNIT2 TSP FEEDWATER TEMPERATURE HIGH
IX0.11.102.11	Input 11	bool	TSP_Application.I2G62TS5001L	UNIT2 TSP MAIN STEAM SYSTEM TEMPERATURE HIGH
IX0.11.102.12	Input 12	bool	TSP_Application.I2G62CHU5003	UNIT2 TSP CHILLER TROUBLE
IX0.11.102.13	Input 13	bool	TSP_Application.I2G62RELE5001A	UNIT2 TSP SAMPLE CONDITIONING RACK 120V POWER AVAILABLE
IX0.11.102.14	Input 14	bool	TSP_Application.I2G62BUT5002B	UNIT2 TSP ALARM ACKNOWLEDGE
IX0.11.102.15	Input 15	bool	TSP_Application.I2G62BUT5002A	UNIT2 TSP ALARM RESET
IX0.11.102.16	Input 16	bool	TSP_Application.I2G62AE5001K3	UNIT2 TSP FEEDWATER SPECIFIC COND. ANALYZER FAULT
IW0.11.102.17	All Inputs	dword		
IW0.11.102.18	Channel status	dword		
IW0.11.102.19	UnitStatus	dint	TSP_Application.DI810_1.HwState	

FIGURA 4.55

Status:

Channel	Channel Value	Forced	Variable Value	Variable
IX0.11.102.1		☐		TSP_Application.I2G62TS5001A
IX0.11.102.2		☐		TSP_Application.I2G62TS5001B
IX0.11.102.3		☐		TSP_Application.I2G62TS5001C
IX0.11.102.4		☐		TSP_Application.I2G62TS5001D
IX0.11.102.5		☐		TSP_Application.I2G62TS5001E
IX0.11.102.6		☐		TSP_Application.I2G62TS5001F
IX0.11.102.7		☐		TSP_Application.I2G62TS5001G
IX0.11.102.8		☐		TSP_Application.I2G62TS5001H
IX0.11.102.9		☐		TSP_Application.I2G62TS5001J
IX0.11.102.10		☐		TSP_Application.I2G62TS5001K
IX0.11.102.11		☐		TSP_Application.I2G62TS5001L
IX0.11.102.12		☐		TSP_Application.I2G62CHU5003
IX0.11.102.13		☐		TSP_Application.I2G62RELE5001A
IX0.11.102.14		☐		TSP_Application.I2G62BUT5002B
IX0.11.102.15		☐		TSP_Application.I2G62BUT5002A
IX0.11.102.16		☐		TSP_Application.I2G62AE5001K3
IW0.11.102.17		☐		
IW0.11.102.18		☐		

FIGURA 4.56

Unit Status:

Name	Value	Description
HWState		
HWStateChangeTime		
ErrorsAndWarnings		
ExtendedStatus		
LatchedErrorsAndWarnings		
LatchedExtendedStatus		

FIGURA 4.57

❖ Configuración ModuleBus de DO810:

En este caso la pantalla principal posee cuatro pestañas: Settings, Connections, Status y Unit Status.

Settings: Podemos definir si los canales están activos o no, supervisión de alimentación exterior, etc.

Parameter	Value	Type	Unit	Min	Max
External power supervision	true	bool			
Activate channel 1	true	bool			
Activate channel 2	true	bool			
Activate channel 3	true	bool			
Activate channel 4	true	bool			
Activate channel 5	true	bool			
Activate channel 6	true	bool			
Activate channel 7	true	bool			
Activate channel 8	true	bool			
Activate channel 9	true	bool			
Activate channel 10	true	bool			
Activate channel 11	true	bool			
Activate channel 12	true	bool			
Activate channel 13	true	bool			
Activate channel 14	true	bool			
Activate channel 15	true	bool			
Activate channel 16	true	bool			
OSP control channel 1	Set OSP value	enum			
OSP control channel 2	Set OSP value	enum			
OSP control channel 3	Set OSP value	enum			
OSP control channel 4	Set OSP value	enum			
OSP control channel 5	Set OSP value	enum			
OSP control channel 6	Set OSP value	enum			
OSP control channel 7	Set OSP value	enum			
OSP control channel 8	Set OSP value	enum			
OSP control channel 9	Set OSP value	enum			
OSP control channel 10	Set OSP value	enum			
OSP control channel 11	Set OSP value	enum			
OSP control channel 12	Set OSP value	enum			
OSP control channel 13	Set OSP value	enum			
OSP control channel 14	Set OSP value	enum			
OSP control channel 15	Set OSP value	enum			
OSP control channel 16	Set OSP value	enum			
OSP value channel 1	false	bool			
OSP value channel 2	false	bool			
OSP value channel 3	false	bool			
OSP value channel 4	false	bool			
OSP value channel 5	false	bool			
OSP value channel 6	false	bool			
OSP value channel 7	false	bool			
OSP value channel 8	false	bool			
OSP value channel 9	false	bool			
OSP value channel 10	false	bool			
OSP value channel 11	false	bool			
OSP value channel 12	false	bool			

OSP value channel 13	false	bool			
OSP value channel 14	false	bool			
OSP value channel 15	false	bool			
OSP value channel 16	false	bool			

FIGURA 4.58

En la Pestaña Connections asignamos variables de la Base de Datos para guardar la información sobre las salidas digitales del área.

Channel	Name	Type	Variable	I/O Description
QX0.11.104.1	Output 1	bool	TSP_Application.O2G62TS5001AA	UNIT2 TSP MSR 1 LOW PRESSURE TURBINE TEMPERATURE HIGH ALARM
QX0.11.104.2	Output 2	bool	TSP_Application.O2G62TS5001BA	UNIT2 TSP MSR 2 LOW PRESSURE TURBINE TEMPERATURE HIGH ALARM
QX0.11.104.3	Output 3	bool	TSP_Application.O2G62TS5001CA	UNIT2 TSP MSR 3 LOW PRESSURE TURBINE TEMPERATURE HIGH ALARM
QX0.11.104.4	Output 4	bool	TSP_Application.O2G62TS5001DA	UNIT2 TSP MSR A 1ST STAGE REHEATER DT A1,A2,B1,B2 TEMPERATURE HIGH ALARM
QX0.11.104.5	Output 5	bool	TSP_Application.O2G62TS5001EA	UNIT2 TSP MSR A 2ND STAGE REHEATER DT A1,A2 TEMPERATURE HIGH ALARM
QX0.11.104.6	Output 6	bool	TSP_Application.O2G62TS5001FA	UNIT2 TSP MSR A 2ND STAGE REHEATER DT B1,B2 TEMPERATURE HIGH ALARM
QX0.11.104.7	Output 7	bool	TSP_Application.O2G62TS5001GA	UNIT2 TSP MSR A SEPARATOR DT TEMPERATURE HIGH ALARM
QX0.11.104.8	Output 8	bool	TSP_Application.O2G62TS5001HA	UNIT2 TSP MSR B SEPARATOR DT TEMPERATURE HIGH ALARM
QX0.11.104.9	Output 9	bool	TSP_Application.O2G62TS5001JA	UNIT2 TSP GLAND STEAM SUPPLY TEMPERATURE ALARM HIGH
QX0.11.104.10	Output 10	bool	TSP_Application.O2G62TS5001KA	UNIT2 TSP FEEDWATER TEMPERATURE HIGH ALARM
QX0.11.104.11	Output 11	bool	TSP_Application.O2G62TS5001LA	UNIT2 TSP MAIN STEAM SYSTEM TEMPERATURE HIGH ALARM
QX0.11.104.12	Output 12	bool	TSP_Application.O2G62TS5002A	UNIT2 TSP CONTROL PANEL HIGH TEMP ALARM
QX0.11.104.13	Output 13	bool	TSP_Application.O2G62TSPTRBA	UNIT2 TSP PLC SYSTEM TROUBLE ALARM
QX0.11.104.14	Output 14	bool	TSP_Application.O2G62TSPPWRA	UNIT2 TSP POWER AVAILABLE ALARM
QX0.11.104.15	Output 15	bool	TSP_Application.O2G62CHU5003A	UNIT2 TSP CHILLER TROUBLE ALARM
QX0.11.104.16	Output 16	bool	TSP_Application.O2G62AE5001K3A	UNIT2 TSP FW SPEC COND. HIGH
QW0.11.104.17	All Outputs	dword		
IW0.11.104.18	Channel status	dword		
IW0.11.104.19	UnitStatus	dint	TSP_Application.DO810_1.HwState	

FIGURA 4.59

Status:

Channel	Channel Value	Forced	Variable Value	Variable
QX0.11.104.1		☐		TSP_Application.O2G62TS5001AA
QX0.11.104.2		☐		TSP_Application.O2G62TS5001BA
QX0.11.104.3		☐		TSP_Application.O2G62TS5001CA
QX0.11.104.4		☐		TSP_Application.O2G62TS5001DA
QX0.11.104.5		☐		TSP_Application.O2G62TS5001EA
QX0.11.104.6		☐		TSP_Application.O2G62TS5001FA
QX0.11.104.7		☐		TSP_Application.O2G62TS5001GA
QX0.11.104.8		☐		TSP_Application.O2G62TS5001HA
QX0.11.104.9		☐		TSP_Application.O2G62TS5001JA
QX0.11.104.10		☐		TSP_Application.O2G62TS5001KA
QX0.11.104.11		☐		TSP_Application.O2G62TS5001LA
QX0.11.104.12		☐		TSP_Application.O2G62TS5002A
QX0.11.104.13		☐		TSP_Application.O2G62TSPTRBA
QX0.11.104.14		☐		TSP_Application.O2G62TSPPWRA
QX0.11.104.15		☐		TSP_Application.O2G62CHU5003A
QX0.11.104.16		☐		TSP_Application.O2G62AE5001K3A
QW0.11.104.17		☐		
IW0.11.104.18		☐		

FIGURA 4.60

Unit Status:

Name	Value	Description
HWState		
HWStateChangeTime		
ErrorsAndWarnings		
ExtendedStatus		
LatchedErrorsAndWarnings		
LatchedExtendedStatus		

FIGURA 4.61

❖ Configuración ModuleBus de DO820:

La configuración es idéntica que en el módulo DO810, por lo que no vamos a aburrir al lector con más figuras semejantes al módulo anterior. Las únicas diferencias son que está pensado para aplicaciones con relé normalmente abierto y son 8 canales en vez de 16.