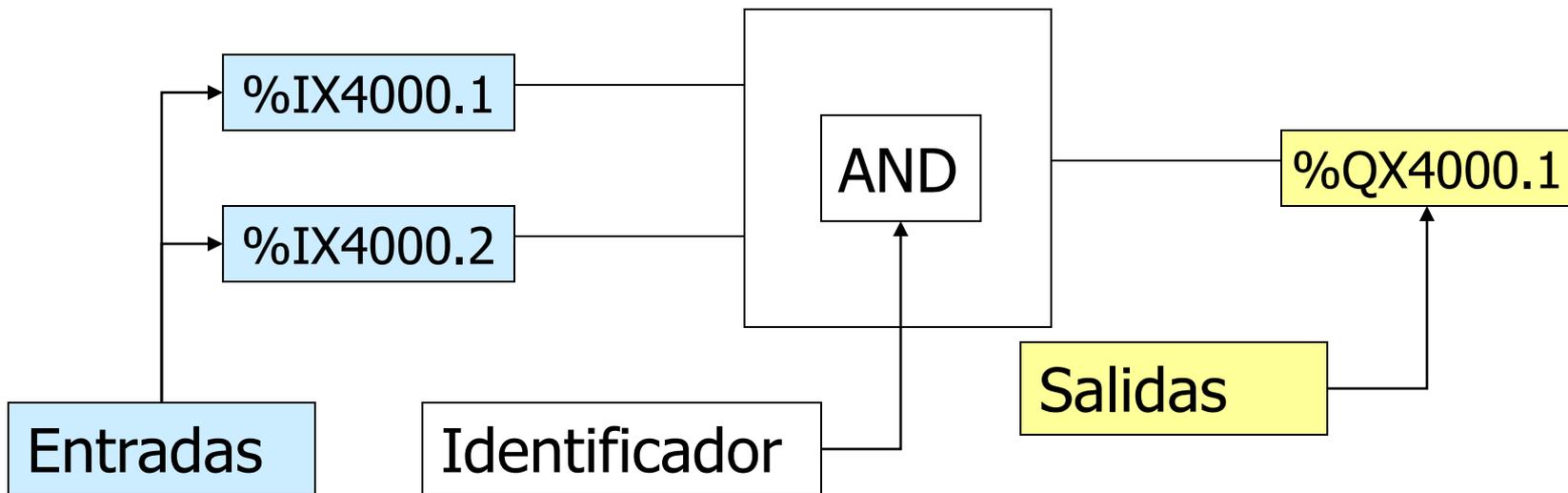


Lenguaje FBD (Function Block Diagram)

Controladores Lógicos Programables

El Bloque Funcional

Bloque funcional: rectángulo con entradas, salidas e identificador



Los tipos de datos dependen del bloque

Estructura de programa FBD

Programa FBD: consiste en conexiones entre bloques funcionales y datos, por líneas de conexión

El programa se ejecuta de arriba abajo, y de izquierda a derecha

Se permite conectar la salida de un bloque a la entrada de otro

Bloques Funcionales

- Se definen sobre la base de “templates” de bloques
- Template de bloque: programa que define el bloque
- Para utilizar un template de bloque, la tarea declara una instancia del template
- Bibliotecas propias de usuario (re-uso)

Funciones

- Aceptan múltiples parámetros de entrada
- Salida única

Función vs Bloque Funcional

- Diferencia entre bloque funcional y función:
 - Número de salidas:
 - Función permite sólo una salida
 - Bloque funcional permite más de una
 - Variables persistentes (conservan valor entre ejecuciones):
 - Función no
 - Bloque funcional sí

Grupos de instrucciones

Clasificación:

Funciones binarias

Entradas y salidas de tipo binario: AND, OR, XOR

Número de entradas variable; posibilidad de negación de entradas

Funciones de timers y contadores

Detalladas en la clase sobre lenguaje LD

Funciones de palabras/reales

Funciones de comparación: el resultado es un bit

Funciones aritméticas: el resultado es una palabra/real (+, -, *, /)

Funciones lógicas bit a bit: AND, OR, XOR, Shift, Rotate

Funciones de control de programa

JUMP; lectura o escritura inmediata de E/S

Funciones de control

PI, PID, etc. *

Funciones de comunicación

Comunicación por protocolos, por ejemplo MODBUS

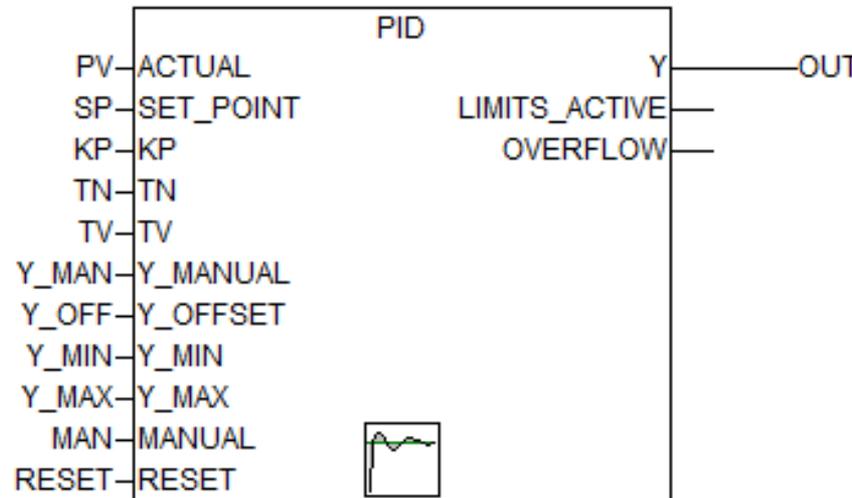
Funciones de conversión de formato

PACK y UNPACK

El 90% de los programas se resuelve con el 20% de las instrucciones

Grupos de instrucciones

- Funciones de control: PI, PID, etc.



$$OUT = Y = KP \left(e + \frac{1}{TN} \int e \cdot dt + TV \frac{\partial e}{\partial t} \right) + Y_OFFSET; e = SP - PV$$

$$PID(s) = KP \left(1 + \frac{1}{TN \cdot s} + TV \cdot s \right)$$

LD o FBD

Algunas diferencias entre un programa FBD y un programa LD:

La implementación de las funciones lógicas de bits

La concatenación de bloques funcionales:

Permitida en FBD

No permitida en LD

Ejemplo Bomba

Se desea escribir un programa que controle el encendido - apagado de una bomba.

La bomba será encendida si:

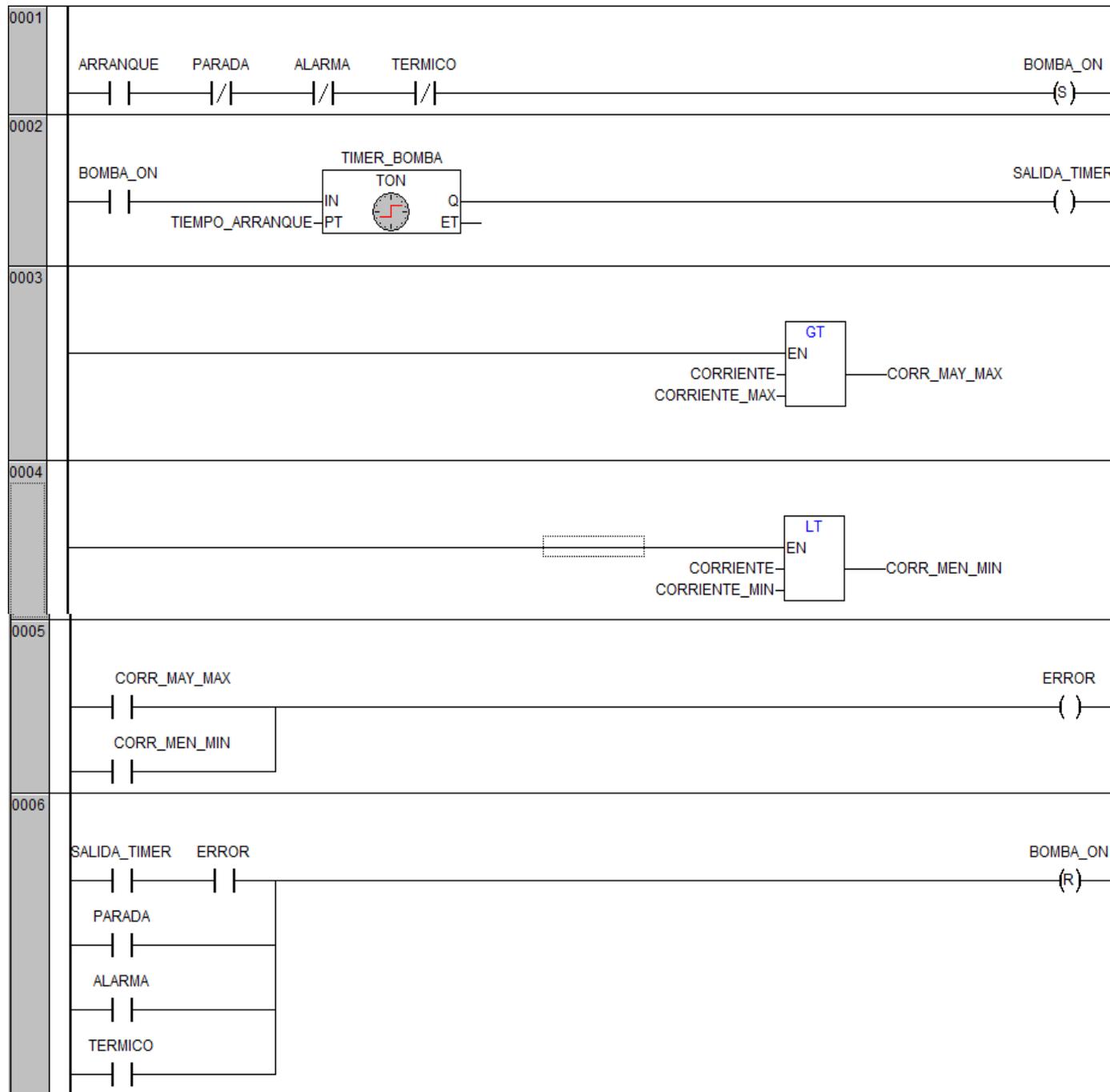
- 1) Se pulsa el botón de arranque.
- 2) La protección térmica está deshabilitada.
- 3) Está abierto el botón de alarma.
- 4) Está abierto el botón de parada.

Desde un tiempo T después del encendido, no puede haber ni sobre corriente ni baja corriente. Expresado de otra forma, desde un tiempo T después del arranque, la corriente I debe cumplir $I_{\text{MIN}} < I < I_{\text{MAX}}$, siendo I_{MIN} e I_{MAX} límites prefijados.

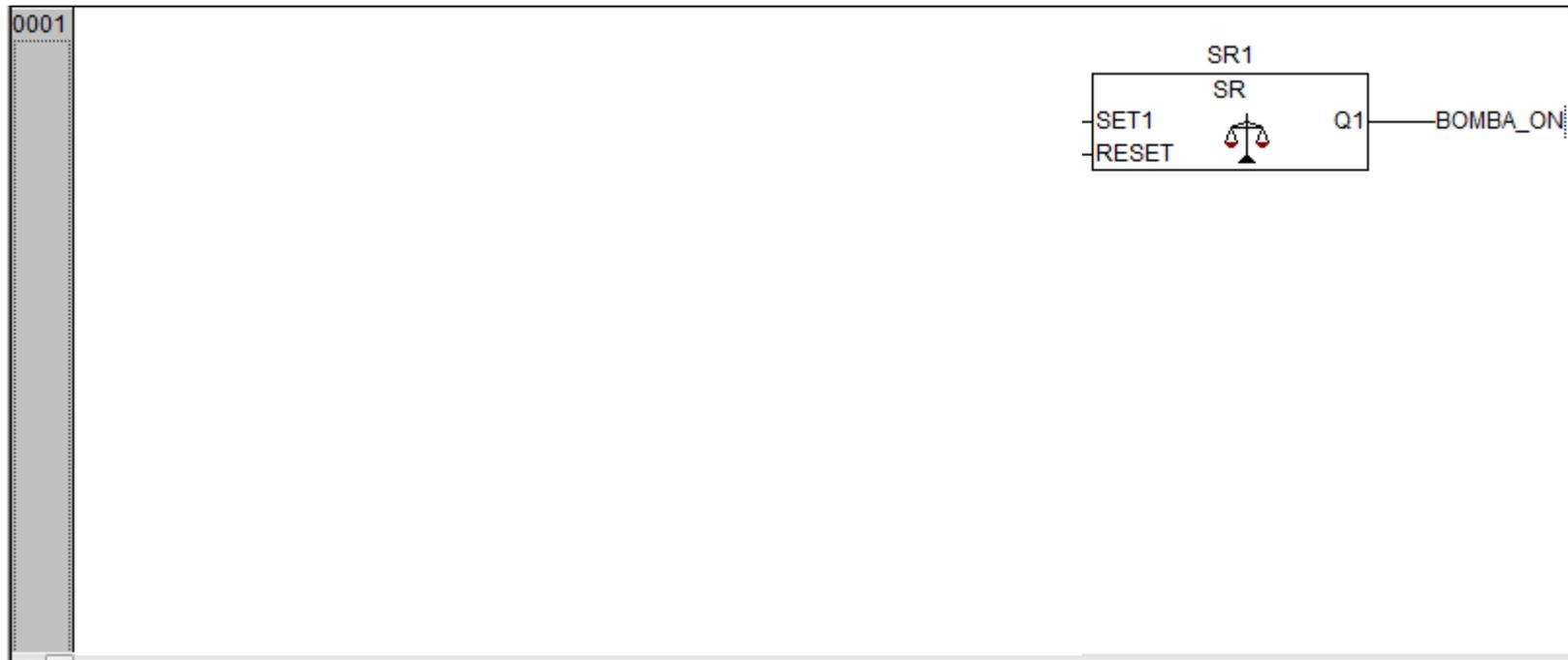
Ejemplo Bomba

El motor de la bomba se apagará si:

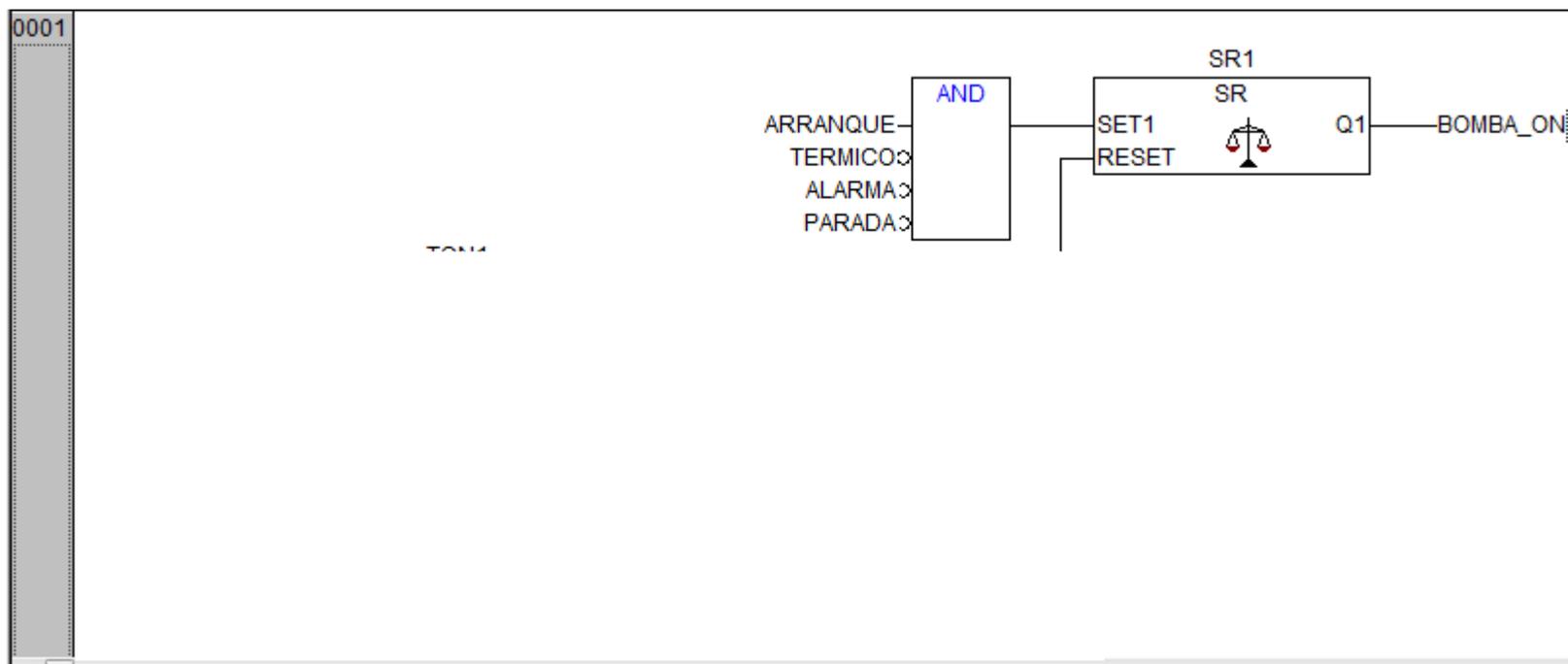
- 1) Se pulsa el botón de parada.
- 2) Se cierra la protección térmica.
- 3) Se pulsa el botón de alarma.
- 4) Los límites de corriente no son los correctos.



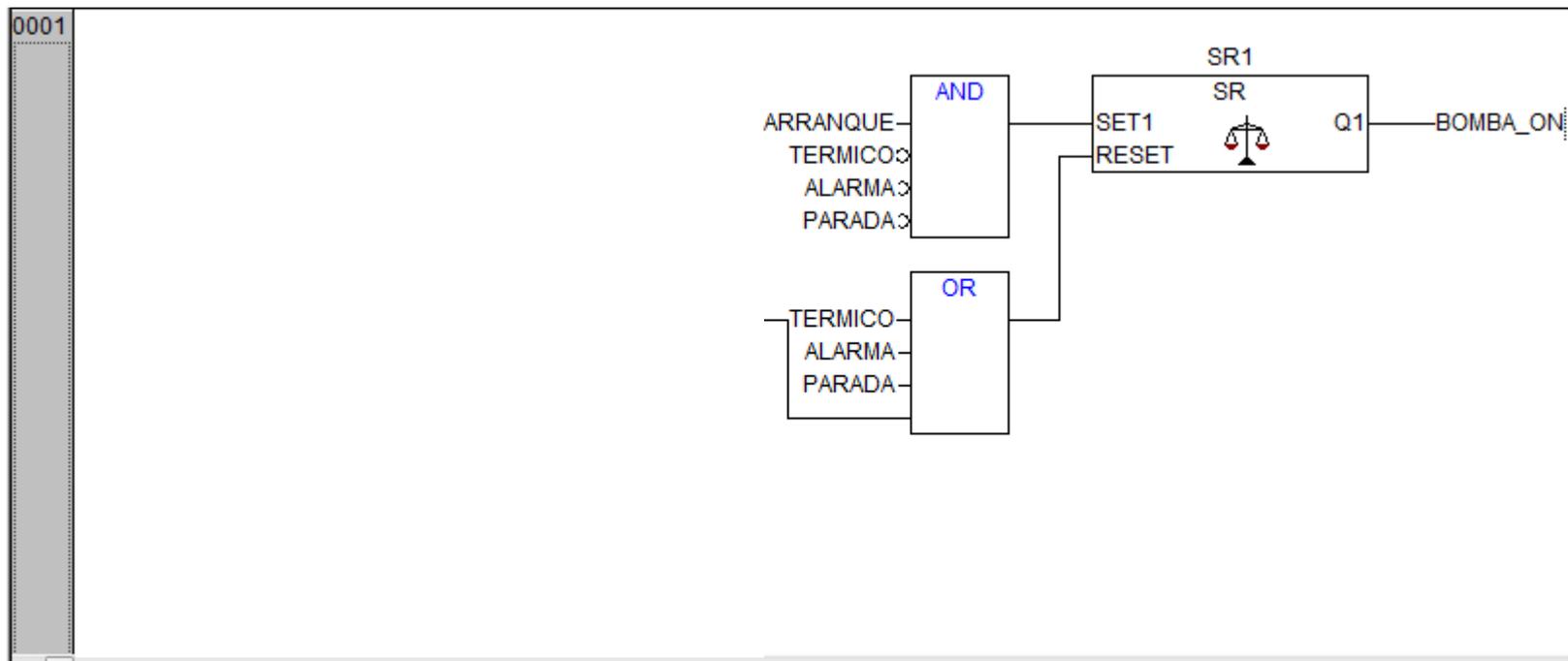
Ejemplo Bomba en FBD



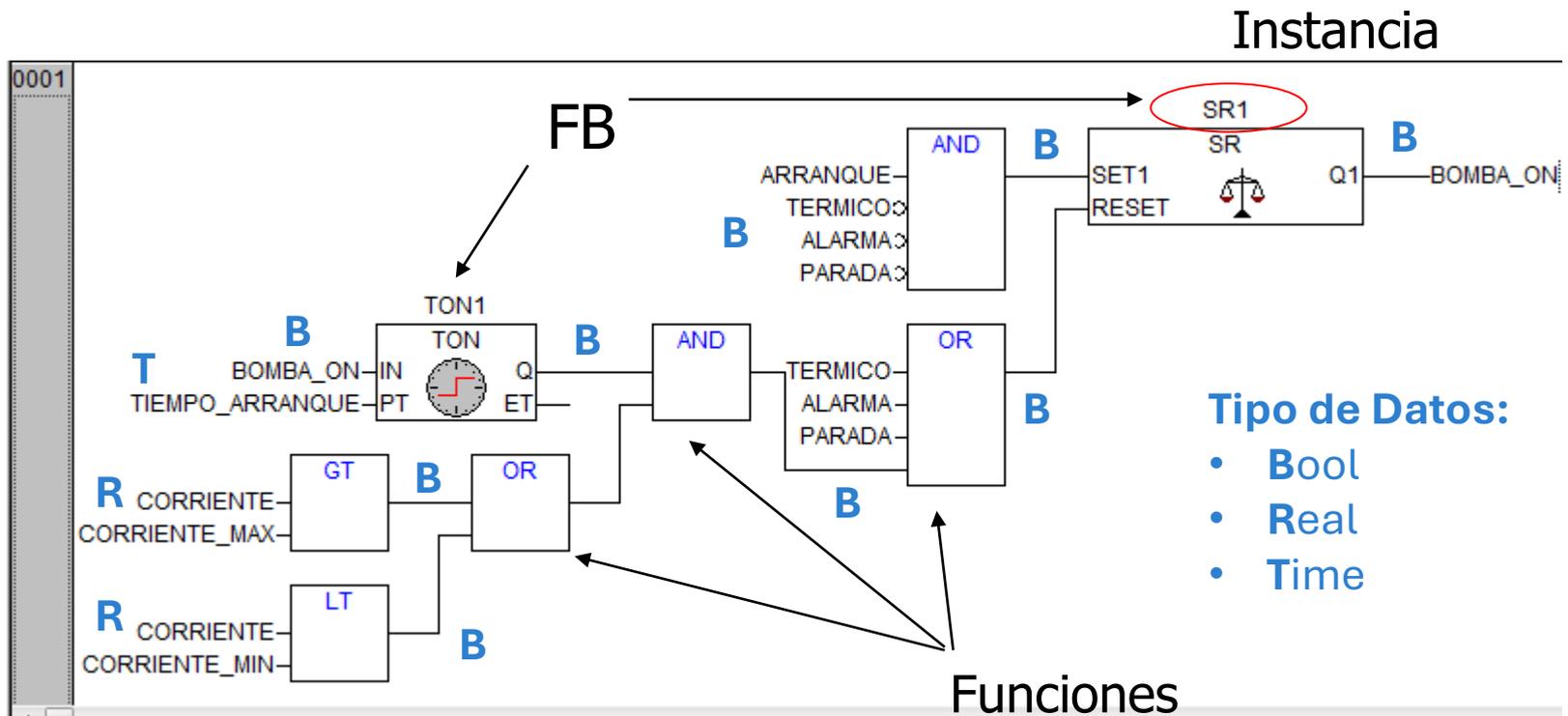
Ejemplo Bomba en FBD



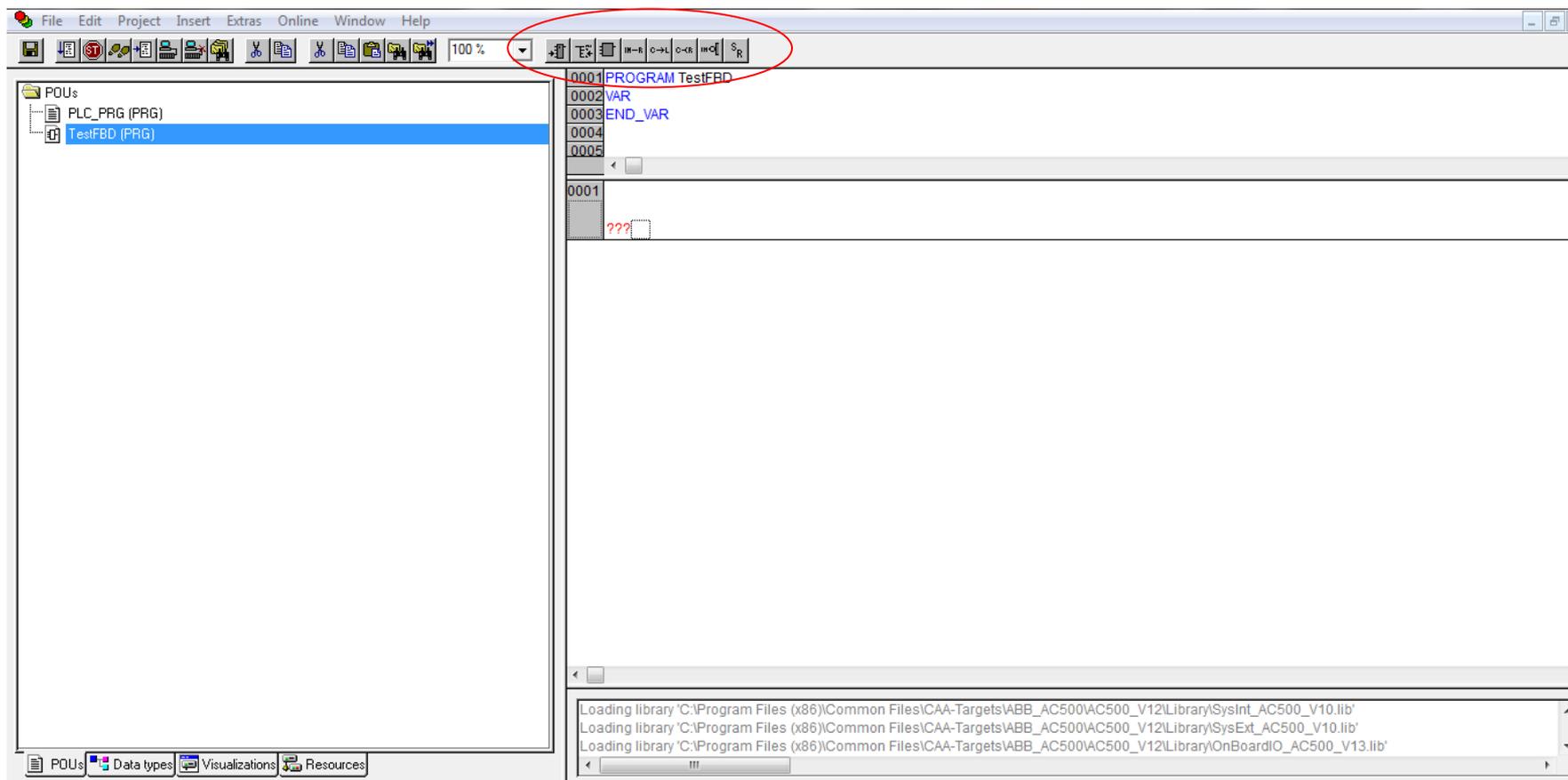
Ejemplo Bomba en FBD



Ejemplo Bomba en FBD



Ambiente en FBD



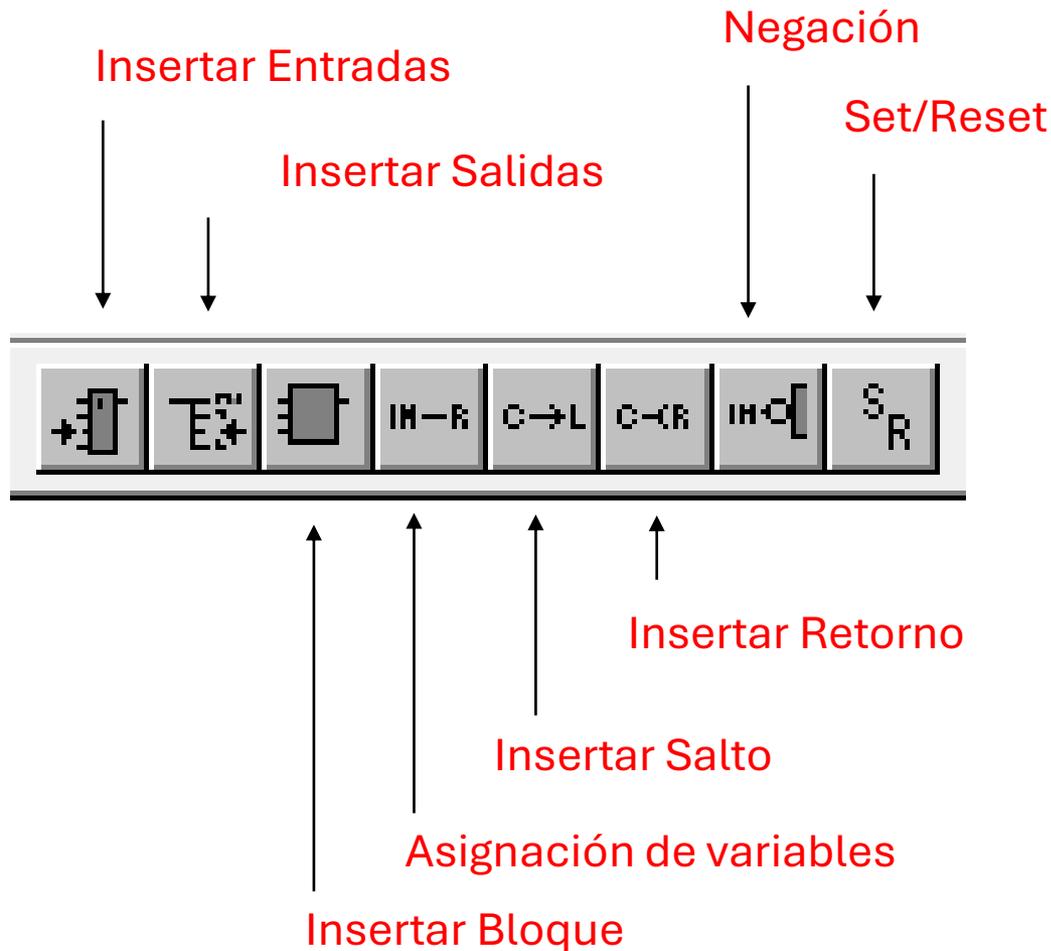
The screenshot displays a software interface for PLC programming. The main window is titled "File Edit Project Insert Extras Online Window Help". Below the menu bar is a toolbar with various icons for file operations, editing, and execution. A red circle highlights the toolbar area. The left pane shows a project tree with "POUs" containing "PLC_PRG (PRG)" and "TestFBD (PRG)". The main editor area shows a ladder logic program with the following code:

```
0001 PROGRAM TestFBD
0002 VAR
0003 END_VAR
0004
0005
0001
???
```

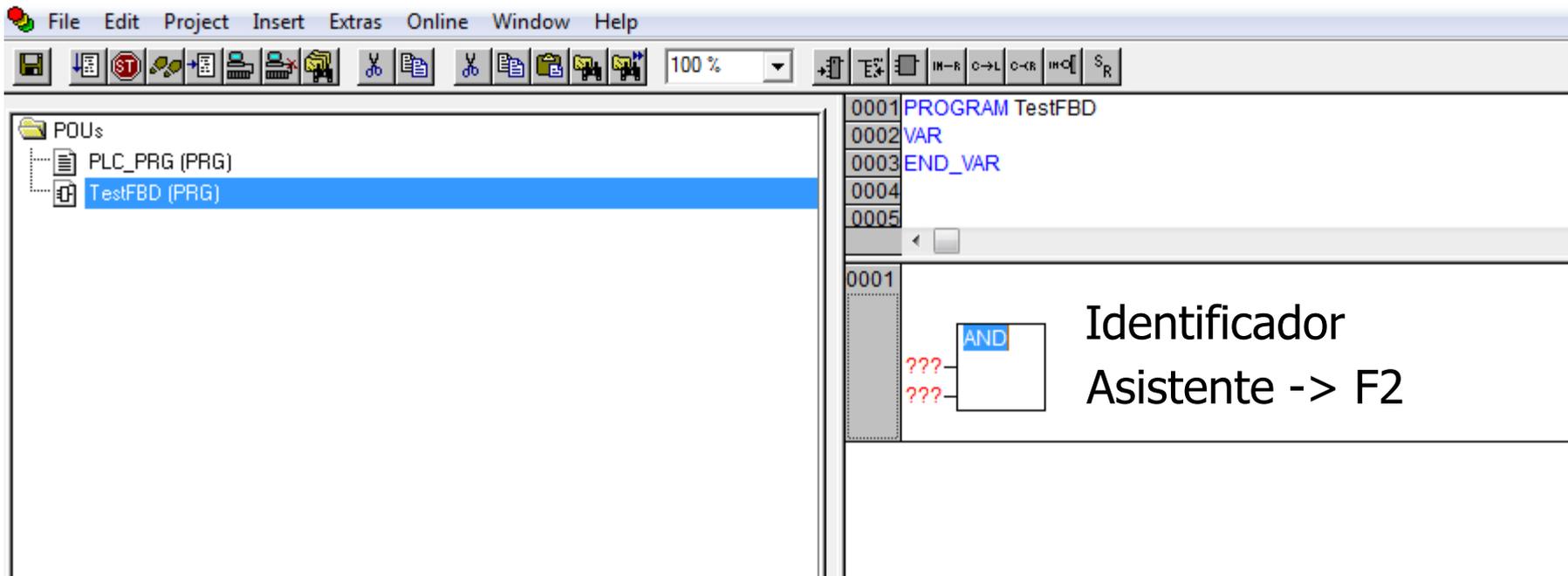
The status bar at the bottom indicates the current page is "POUs" and shows icons for "Data types", "Visualizations", and "Resources". The bottom right corner displays a message box with the following text:

```
Loading library 'C:\Program Files (x86)\Common Files\CAA-Targets\ABB_AC500\AC500_V12\Library\SysInt_AC500_V10.lib'
Loading library 'C:\Program Files (x86)\Common Files\CAA-Targets\ABB_AC500\AC500_V12\Library\SysExt_AC500_V10.lib'
Loading library 'C:\Program Files (x86)\Common Files\CAA-Targets\ABB_AC500\AC500_V12\Library\OnBoardIO_AC500_V13.lib'
```

Herramientas FBD



Ambiente en FBD

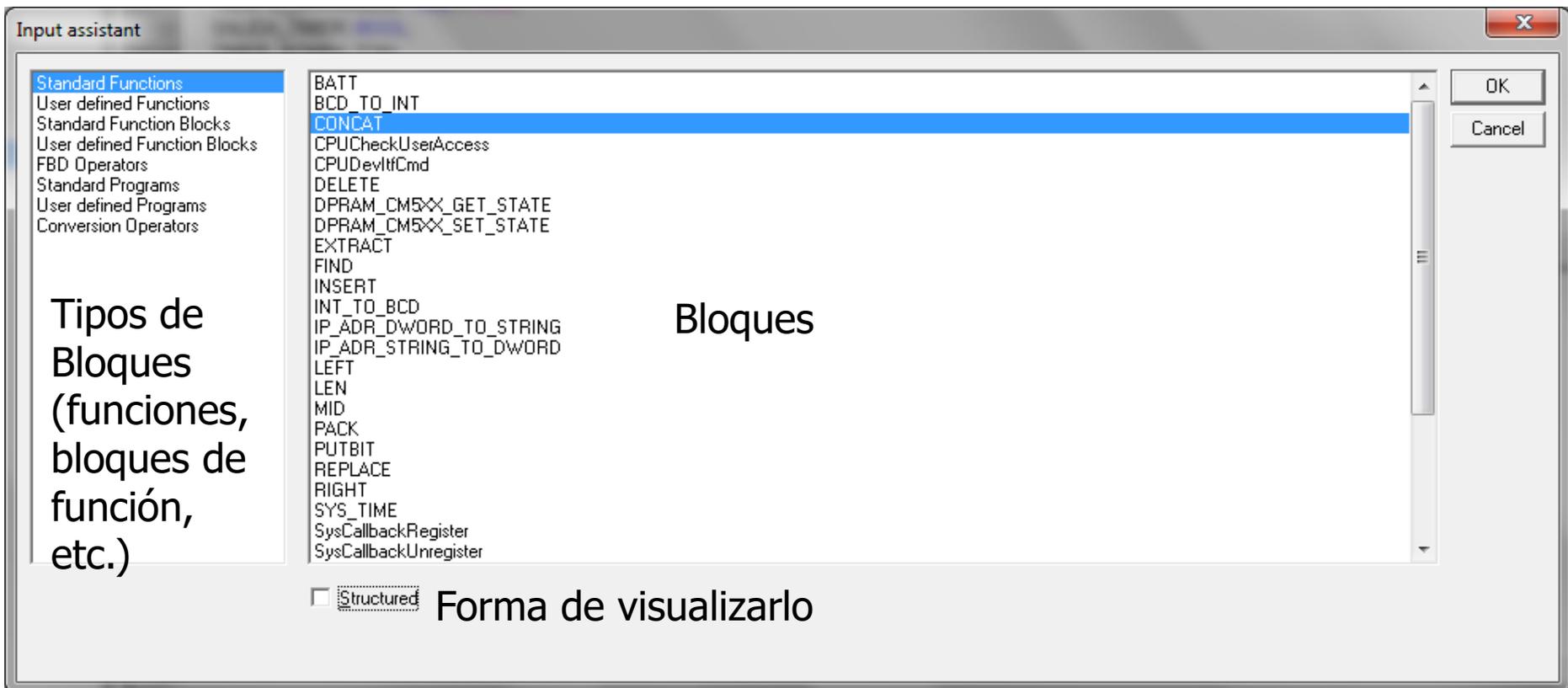


The screenshot displays a software interface for PLC programming. The top menu bar includes 'File', 'Edit', 'Project', 'Insert', 'Extras', 'Online', 'Window', and 'Help'. Below the menu is a toolbar with various icons for file operations and editing. The main workspace is divided into two panes. The left pane shows a project tree with a folder 'POUs' containing 'PLC_PRG (PRG)' and 'TestFBD (PRG)'. The right pane shows the ladder logic editor for 'PROGRAM TestFBD'. The editor displays the following code:

```
0001 PROGRAM TestFBD
0002 VAR
0003 END_VAR
0004
0005
```

Below the code, a ladder logic diagram is shown for step 0001. It features an 'AND' gate with two inputs, both labeled with red '???' characters. To the right of the diagram, the text 'Identificador Asistente -> F2' is displayed.

Bibliotecas de Bloques



Library Manager

- Menú “Window” -> “Library Manager”

Bibliotecas

```

BusDiag.lib 30.9.13 17:16:56
SysInt_AC500_V10.lib 8.10.13 10:08:36
SysExt_AC500_V10.lib 30.9.13 17:16:56
OnBoardIO_AC500_V13.lib 30.9.13 17:16:56
standard.lib 4.10.05 11:14:46
Iecsf.lib 13.4.06 15:51:28
Util.lib 18.5.10 15:14:28
SysLibTime.lib 30.9.13 17:16:56
SysTaskInfo.lib 30.9.13 17:16:56
SysLibMem.lib 30.9.13 17:16:56
SysLibInitLibrary.lib 30.9.13 17:17:30
SYSLIBCALLBACK.LIB 30.9.13 17:16:56
Ethernet_AC500_V10.lib 30.9.13 17:17:30
                
```

Declaración

```

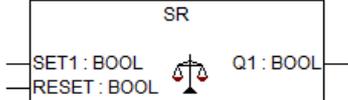
FUNCTION_BLOCK SR
(*
  Bistable function, set dominant
  Q1 = SET1 OR (NOT RESET AND Q1)
*)
VAR_INPUT
  SET1: BOOL;
  RESET: BOOL;
END_VAR
VAR_OUTPUT
  Q1: BOOL;
END_VAR
                
```

Bloques

- POUs
 - Bistable Function Blocks
 - RS (FB)
 - SEMA (FB)
 - SR (FB)
 - Counter
 - CTD (FB)
 - CTU (FB)
 - CTUD (FB)
 - String Functions
 - CONCAT (FUN)

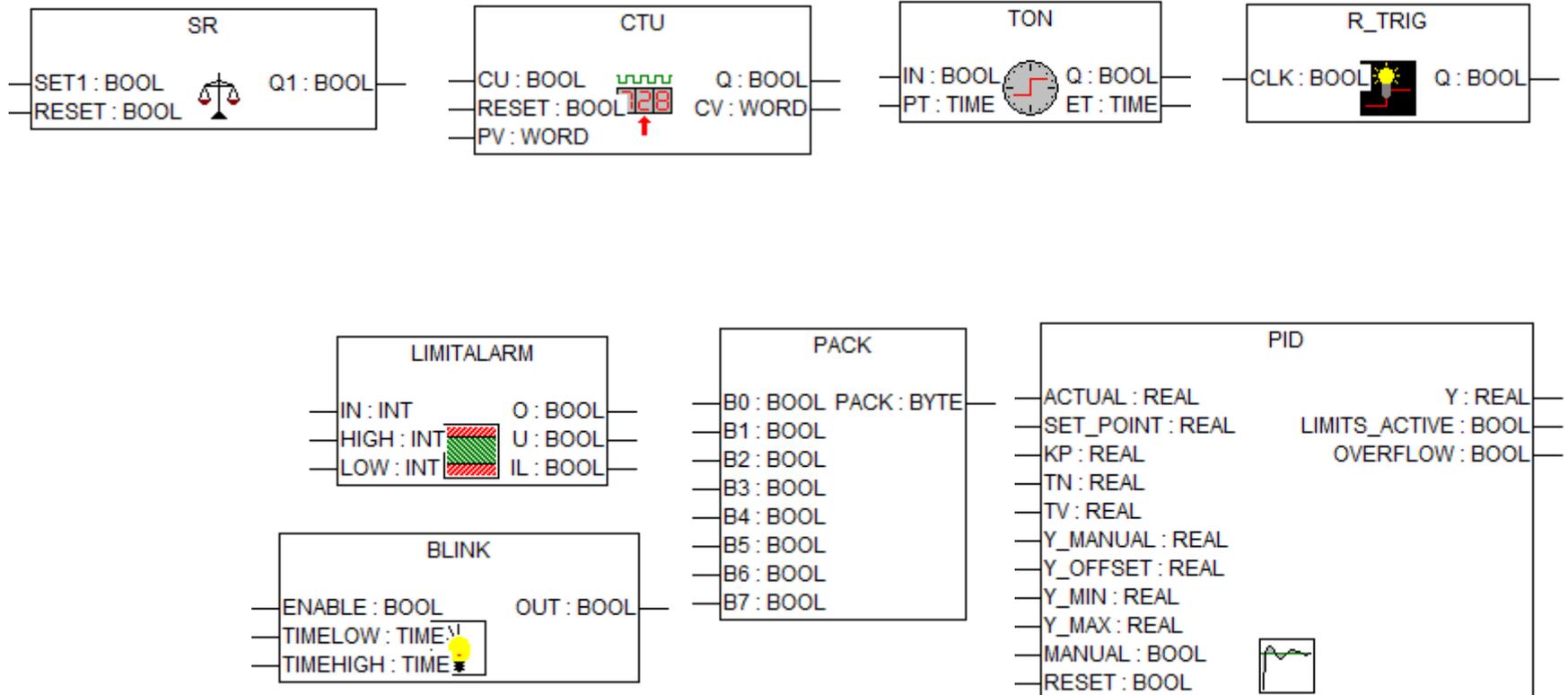
POUs | Data types | Visualizations | Global Variables

Representación



Library Manager

- Standard.lib:



Ayuda para Bloques – F1

POUs

- PLC_PRG (PRG)
- TestFBD (PRG)

```

0001 PROGRAM TestFBD
0002 VAR
0003 END_VAR
0004
0005
        
```

```

0001
        
```

BLINK
???

???
ENABLE
OUT

???
TIMELOW


???
TIMEHIGH

Identificador

F1 -> Ayuda



BLINK

Provided by [util.lib](#).

The function block BLINK generates a pulsating signal. The input consists of ENABLE of the type BOOL, as well as TIMELOW and TIMEHIGH of the type TIME. The output OUT is of the type BOOL.

If ENABLE is set to TRUE, BLINK begins to set the output for the time period TIMEHIGH to TRUE and afterwards to set it for the time period TIMELOW to FALSE.

When ENABLE is reset to FALSE, output OUT will not be changed, i.e. no further pulse will be generated. If you explicitly also want to get OUT FALSE when ENABLE is reset to FALSE, you might use "OUT AND ENABLE" (i.e. adding an AND box with parameter ENABLE) at the output.

Example in CFC:

Blinker

blink

TRUE

ENABLE

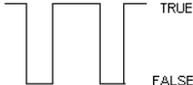
#2s

TIMELOW

#3s

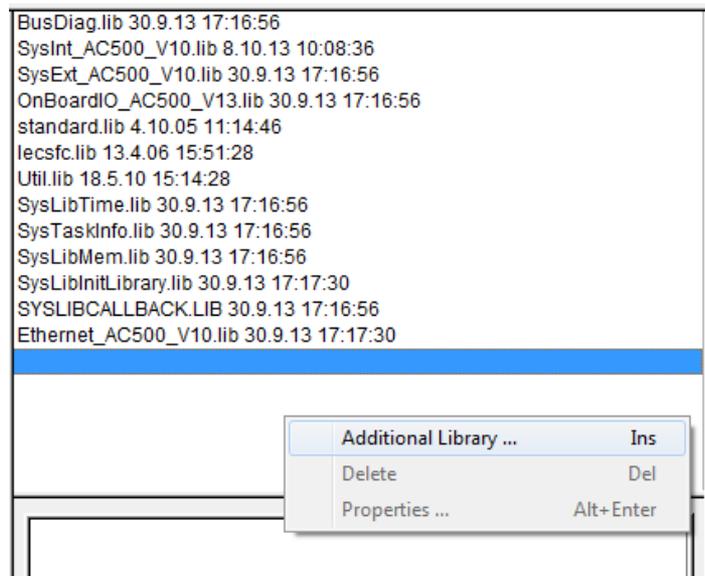
TIMEHIGH

OUT

Library Manager

- Agregar bibliotecas:



Bloques de Usuario

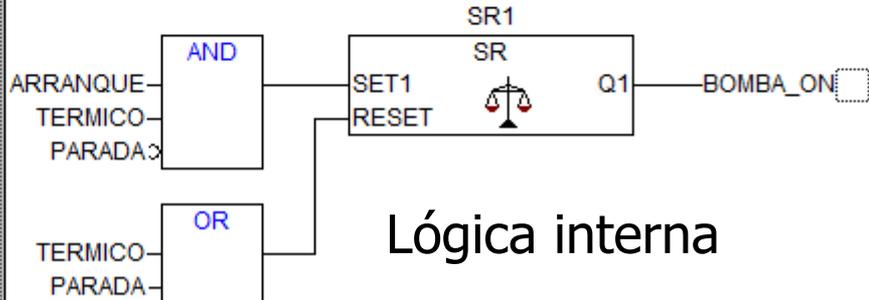
POUs

- Mis Bloques
 - Bomba (FB)
 - PLC_PRG (PRG)
 - ProgFBD (PRG)

Nuevo
"Function Block"

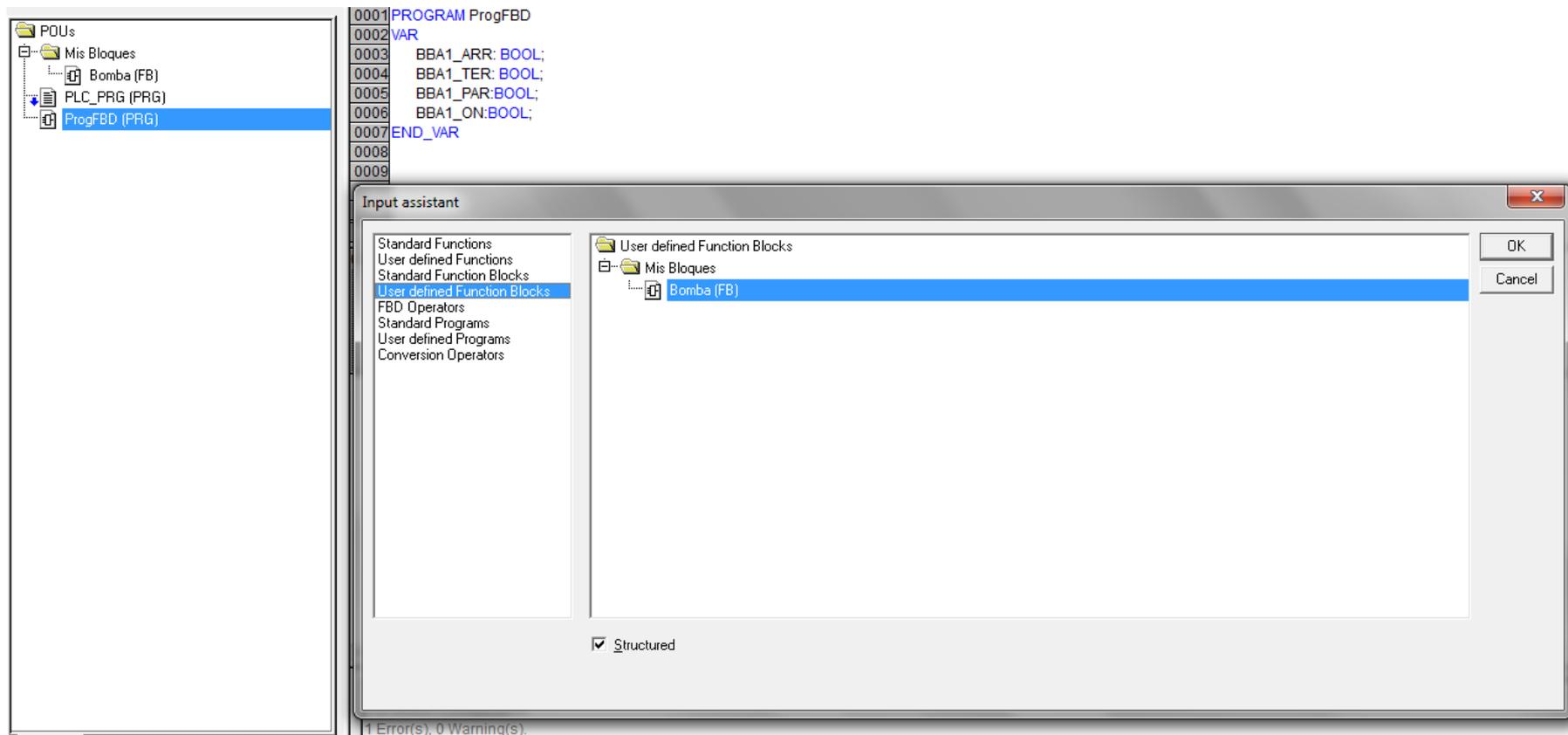
0001	FUNCTION_BLOCK Bomba	
0002	VAR_INPUT	
0003	ARRANQUE: BOOL;	
0004	TERMICO: BOOL;	
0005	PARADA: BOOL;	
0006	END_VAR	
0007	VAR_OUTPUT	
0008	BOMBA_ON:BOOL;	
0009	END_VAR	
0010	VAR	
0011	SR1: SR;	
0012	END_VAR	
0013		

0001	
------	--



Lógica interna

Bloques de Usuario



The screenshot displays a software interface for PLC programming. On the left, a project tree shows the following structure:

- POUs
 - Mis Bloques
 - Bomba (FB)
 - PLC_PRG (PRG)
 - ProgFBD (PRG) (selected)

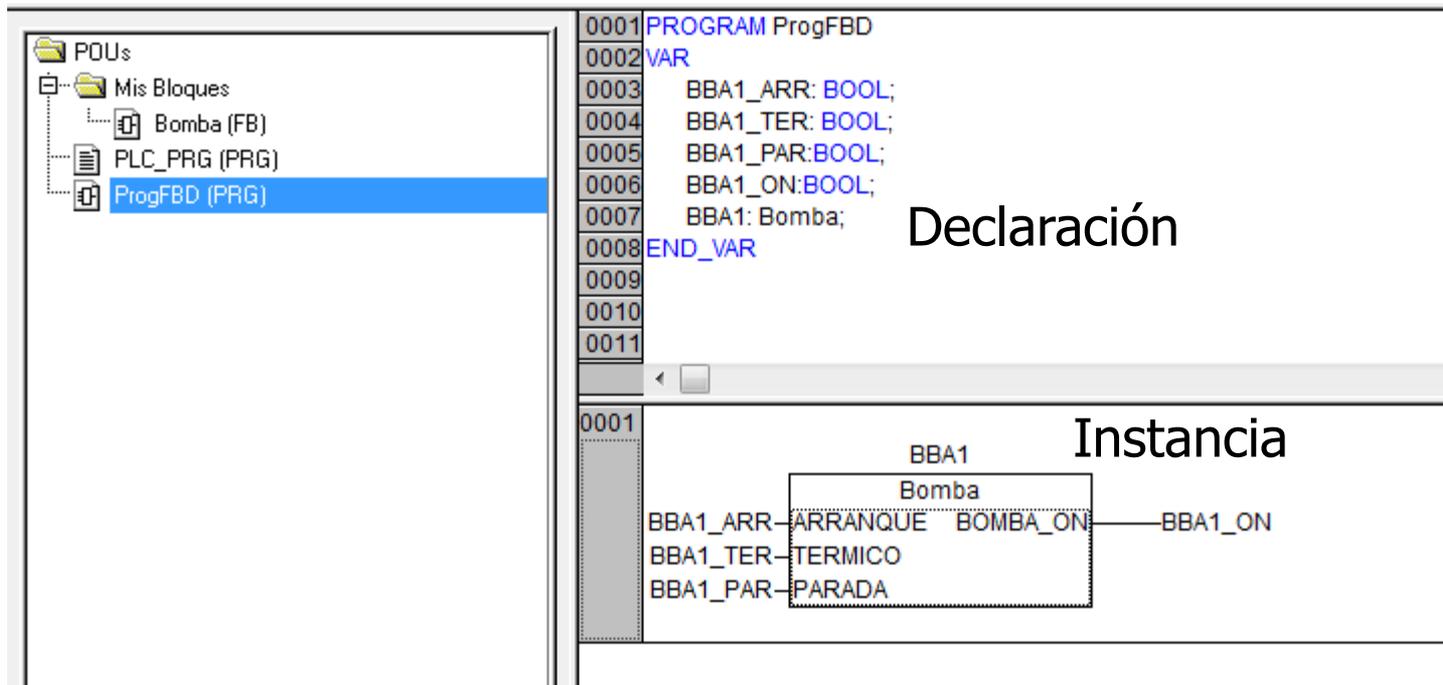
The main editor area shows the following code:

```
0001 PROGRAM ProgFBD
0002 VAR
0003   BBA1_ARR: BOOL;
0004   BBA1_TER: BOOL;
0005   BBA1_PAR: BOOL;
0006   BBA1_ON: BOOL;
0007 END_VAR
0008
0009
```

An 'Input assistant' dialog box is open, showing a list of function blocks. The 'User defined Function Blocks' section is expanded, and 'Bomba (FB)' is selected. The 'Structured' checkbox is checked. The dialog has 'OK' and 'Cancel' buttons.

At the bottom of the interface, a status bar indicates: 1 Error(s), 0 Warning(s).

Bloques de Usuario



The screenshot shows a software interface for PLC programming. On the left is a project tree with the following structure:

- POUs
 - Mis Bloques
 - Bomba (FB)
 - PLC_PRG (PRG)
 - ProgFBD (PRG)** (highlighted)

The main editor area is split into two sections:

Declaración (Declaration):

```

0001 PROGRAM ProgFBD
0002 VAR
0003     BBA1_ARR: BOOL;
0004     BBA1_TER: BOOL;
0005     BBA1_PAR: BOOL;
0006     BBA1_ON: BOOL;
0007     BBA1: Bomba;
0008 END_VAR
0009
0010
0011
  
```

Instancia (Instance):

0001

BBA1

Bomba

BBA1_ARR—ARRANQUE BOMBA_ON—BBA1_ON

BBA1_TER—TERMICO

BBA1_PAR—PARADA

Bloques de Usuario

POUs

- Mis Bloques
 - Bomba (FB)
 - PLC_PRG (PRG)
 - ProgFBD (PRG)

0001	PROGRAM ProgFBD
0002	VAR
0003	BBA1_ARR: BOOL;
0004	BBA1_TER: BOOL;
0005	BBA1_PAR: BOOL;
0006	BBA1_ON: BOOL;
0007	BBA1: Bomba;
0008	BBA2: Bomba;
0009	BBA2_ARR: BOOL;
0010	BBA2_TER: BOOL;
0011	BBA2_PAR: BOOL;
0012	BBA2_ON: BOOL;
0013	END_VAR

0001

BBA1

Bomba

BBA1_ARR	ARRANQUE	BOMBA_ON	BBA1_ON
BBA1_TER	TERMICO		
BBA1_PAR	PARADA		

0002

BBA2

Bomba

BBA2_ARR	ARRANQUE	BOMBA_ON	BBA2_ON <input style="width: 20px; height: 15px; border: 1px dashed gray;" type="checkbox"/>
BBA2_TER	TERMICO		
BBA2_PAR	PARADA		

¿Agregar un entrada más a cada bomba?

Bloques de Usuario

POUs

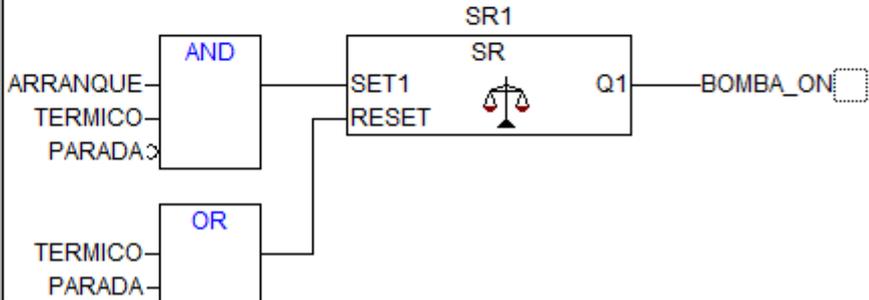
- Mis Bloques
 - Bomba (FB)
 - PLC_PRG (PRG)
 - ProgFBD (PRG)

```

0001 FUNCTION_BLOCK Bomba
0002 VAR_INPUT
0003     ARRANQUE: BOOL;
0004     TERMICO:  BOOL;
0005     PARADA:   BOOL;
0006 END_VAR
0007 VAR_OUTPUT
0008     BOMBA_ON:BOOL;
0009 END_VAR
0010 VAR
0011     SR1: SR;
0012 END_VAR
0013

```


**Agregar un
entrada más a
cada bomba**



Uso de bloques de funciones

- Re-uso de soluciones probadas
- Fácil modificación general
- Cajas negras con comportamiento conocido
- Simplifica el programa general

**Elemento clave de la norma IEC 61131 para
una alta calidad de software**