# INVERNADERO DE GESTION AUTOMATIZADA

## Anejo: Manual de programación

Fernando Prado López

E.U.P Sevilla

# MANUAL DE PROGRAMACIÓN

# 1. LISTA DE INSTRUCCIONES

## 1.1 EXPRESION

Instrucción: Expresión o asignación

Parámetros: Ninguno.

Devuelve: Nada.

Descripción: se entiende por expresión cualquier tipo de asignación del tipo P1 = P2 donde P1 es una variable y P2 puede ser una variable, una constante o una función/instrucción.

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| a = 2 | La variable "a" vale 2 |
| SalidaAlarma = a | La variable "SalidaAlarma" toma el valor de la variable "a" , o sea, 2 |
| a = b+c | La variable "a" toma el valor de la suma de las variables "b" y "c" |
| a = TemperaturaKTY() | La variable "a" toma el valor que devuelve la función TemperaturaKTY() |

## 1.2. FOR…TO

Instrucción: For P1 to P2 step P3 ... P4 next

Parámetros: P1: Valor inicial de la variable de control.

P2: Condición de fin de bucle.

P3: Incremento.

P4: Instrucciones a realizar dentro del bucle.

Devuelve: Nada.

Descripción: La instrucción for es un bucle que permite ejecutar un grupo de instrucciones un determinado número de veces. Siempre debe indicarse su fin

Uso: for InicializacionVariableDeControl to CondicionFinal step incremento

Instrucciones que se ejecutarán next

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| for i = 1 to 10 step 2<br>....acciones...<br><br>next | Empezando con la variable i=1, e incrementándola de 2 en 2 hasta que i=10 se ejecutarán las acciones. Así, i tomará los valores 1, 3, 5, 7 y 9 antes de salir del bucle, por lo que las acciones se ejecutarán cinco veces |

1.3. IF…THEN

Instrucción: If P1 then P2 endif

Parámetros: P1: Condición que se desea examinar.

P2: Acciones que se harán si se cumple la condición.

Devuelve: Nada.

Descripción: La instrucción if ... then se utiliza para que el programa ejecute una serie de instrucciones en caso de cumplirse una determinada condición. En caso de cumplirse P1 el programa ejecutará las instrucciones inmediatas a then (P2) mientras que si no se cumple saltará al fin de instrucción endif. Una instrucción if siempre tiene que acabar con un endif.

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| If Presion<500 then<br>Print(0,0,"Presión baja!")<br>endif | Si el valor de la variable Presion es inferior a 500, entonces se muestra el mensaje "Presión baja!" por el display. En caso de que Presion sea igual o mayor a |

| | |
|---|---|
| | 500 (es decir, no se cumple la condición), se continua el programa. |

1.4. IF…THEN….ELSE

Instrucción: If P1 then P2 else P3 endif

Parámetros: P1: Condición que se desea examinar.

P2: Acciones que se harán si se cumple la condición.

P3: Acciones que se harán si no se cumple la condición.

Devuelve: Nada.

Descripción: La instrucción if ... then se utiliza para que el programa ejecute una serie de instrucciones en caso de cumplirse una determinada condición. En caso de cumplirse P1 el programa ejecutará las instrucciones inmediatas a then (P2) mientras que si no se cumple saltará al siguiente paso, que puede ser un else (otra serie de instrucciones, P3) o bien el fin de instrucción endif. Una instrucción if siempre tiene que acabar con un endif.

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| If Presion<500 then<br>Print(0,0,"Presión baja!")<br>else<br>Print(0,0,"Presión correcta.")<br>endif | Si el valor de la variable Presion es inferior a 500, entonces se muestra el mensaje "Presión baja!" por el display. En caso de que Presion sea igual o mayor a 500, entonces se muestra el mensaje "Presión correcta.". |

1.5. WHILE

Instrucción: While P1 ... P2 wend

Parámetros: P1: Condición durante la que se ejecutará el bucle.

P2: Instrucciones que se ejecutarán mientras se cumpla P1

Devuelve: Nada.

Descripción: La instrucción while es un bucle que permite ejecutar una serie de instrucciones mientras se cumpla la condición P1. Siempre debe indicarse su final mediante un wend.

Uso: while condición

Instrucciones

wend

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| i = 0<br>while i<3<br>Instrucciones<br>i = i +1<br>wend | Inicializamos la variable "i" con valor cero. Mientras "i" sea menor que 3 se ejecutarán las instrucciones.<br>Instrucciones que se ejecutarán.<br>En cada vuelta "i" incrementará 1.<br>En el momento que "i" sea 3 o mayor, se saldrá del bucle. |

1.6. AND

Instrucción: AND

Parámetros: Ninguno.

Devuelve: Nada.

Descripción: La instrucción And se usa únicamente con IF y significa 'y'.

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| if EntFotocel=1 And Temp.T1>15 then<br>SalidaAlarma = 1<br>SalidaBomba = 0 | Si la variable "EntradaFotocelula" vale 1 y la variable "TemperaturaT1" es mayor que 15ºC, se ejecuta SalidaAlarma=1 y |

| | |
|---|---|
| endif | SalidaBomba=0. Únicamente se ejecutará si *las dos* condiciones se cumplen, en caso contrario saltará a endif. |

1.7. OR

Instrucción: OR

Parámetros: Ninguno.

Devuelve: Nada

Descripción: La instrucción OR se utiliza exclusivamente con la instrucción IF y simplemente significa O.

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| if EntFotocel =1 OR Temp.T1>15 then<br>SalidaAlarma = 1<br>Salida bomba=0<br>etc<br>endif | Si la variable "EntradaFotocelula" vale 1 o la temperatura T1 es mayor que 15ºC se ejecuta SalidaAlarma = 1, etc. mientras que si no se cumple ninguna de estas condiciones salta a endif y sale de la instrucción. |

1.8. RESET

Instrucción: Reset( )

Parámetros: Ninguno.

Devuelve: Nada.

Descripción: La función Reset( ) reinicia el Regulador, de forma que el programa vuelve a empezar desde cero. Es el equivalente a desconectar físicamente el Regulador y volver a conectarlo.

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| Reset( ) | Se resetea el Regulador. |

1.9. ENTRADAS Y SALIDAS

Una vez configuradas las terminales TIR en el Panel de E/S y asignado un nombre a cada punto de entrada/salida, el procedimiento para operar con los valores leídos será el siguiente:

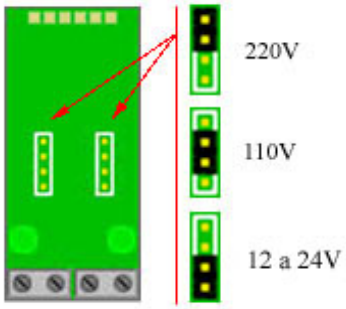
1.9.1 ENTRADAS LÓGICAS

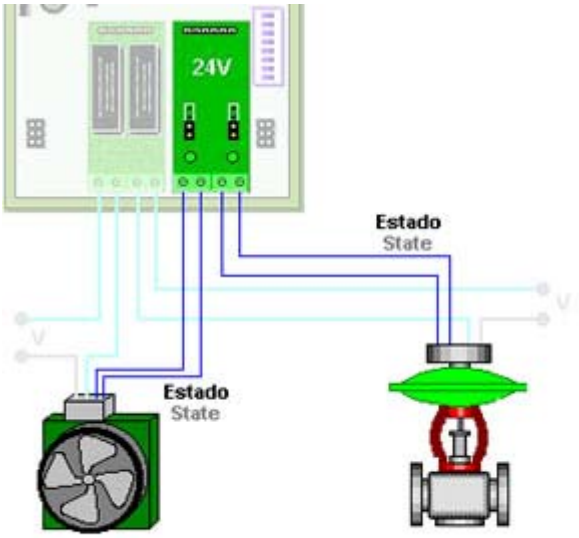
En la variable asignada nos devuelve un 1 ó un ON si la entrada está a nivel alto, y un 0 ó un OFF si está a nivel bajo.

Ejemplo: EntradaDetectorSala

if Entr.DetectSala = 1 then… EQUIVALE A…if Entr.DetectSala = ON then...

if Entr.DetectSala = 0 then… EQUIVALE A…if Entr.DetectSala = OFF then...



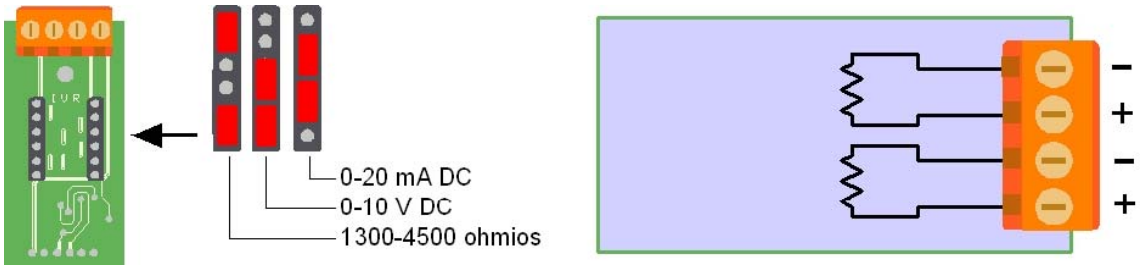



### 1.9.2 ENTRADAS ANALÓGICAS

En la variable asignada nos devuelve un valor de entre 47 y 2047 puntos (es decir, 2000 puntos de resolución), que pueden ser utilizados directamente o bien mediante las funciones de conversión:

| | | | |
|---|---|---|---|
| Temperatura KTY( ) | Valor mínimo | 180 puntos | -28.5ºC |
| | Valor máximo | 1910 puntos | +150ºC |
| | Valor 0ºC | 500 puntos | 0ºC |
| | Valor 100ºC | 1500 puntos | 100ºC |
| Tension ( ) | Valor mínimo | 500 puntos | 0 Voltios |
| | Valor máximo | 1500 puntos | 10 Voltios |
| Intensidad( ) | Valor mínimo | 500 puntos | 0 miliamperios |
| | Valor máximo | 1500 puntos | 20miliamperios |
| Humedad( ) | Valor mínimo | 500 puntos | 0% |
| | Valor máximo | 1500 puntos | 100% |
| Luminosidad ( ) | Valor mínimo | 500 puntos | 0% |
| | Valor máximo | 1500 puntos | 100% |

Ejemplo: LuzSala1=Luminosidad (SondaLuz) --> devuelve un valor entre 0% y 100%.




### 1.9.3 SALIDAS LÓGICAS

En la variable asignada le damos el valor correspondiente: un "1" o un ON si queremos el relé cerrado (conduciendo) y un "0" o un OFF si queremos el relé abierto (no conduciendo).

Ejemplos: EntradaDetectorSala

(1) if EntradaDetectorSala =1 then ReleSala=1

EQUIVALE A → if EntradaDetectorSala = 1 then ReleSala=ON

(2) if EntradaDetectorSala = 1 then ReleSala=0

EQUIVALE A → if EntradaDetectorSala = 1 then ReleSala=OFF

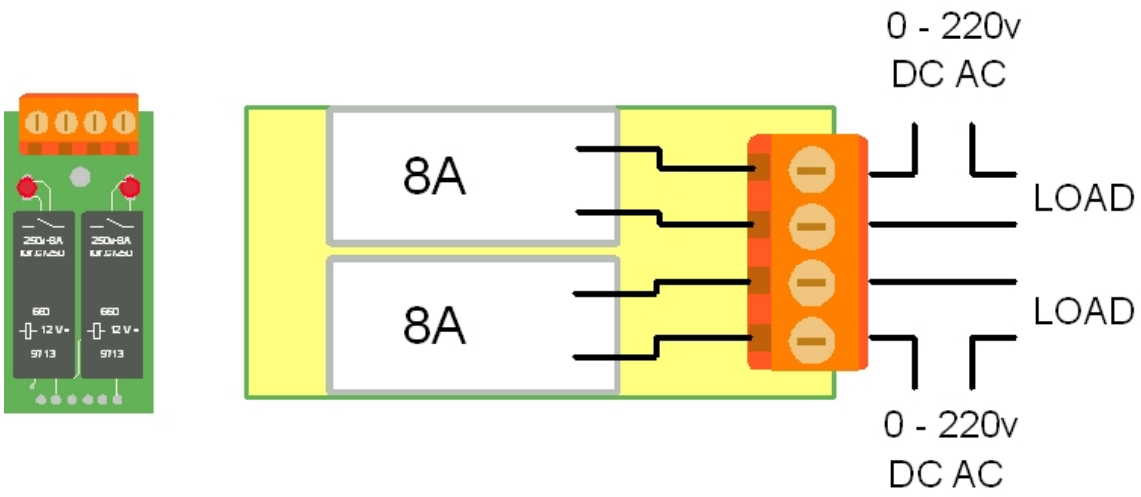



Fig: Salida lógica a relé

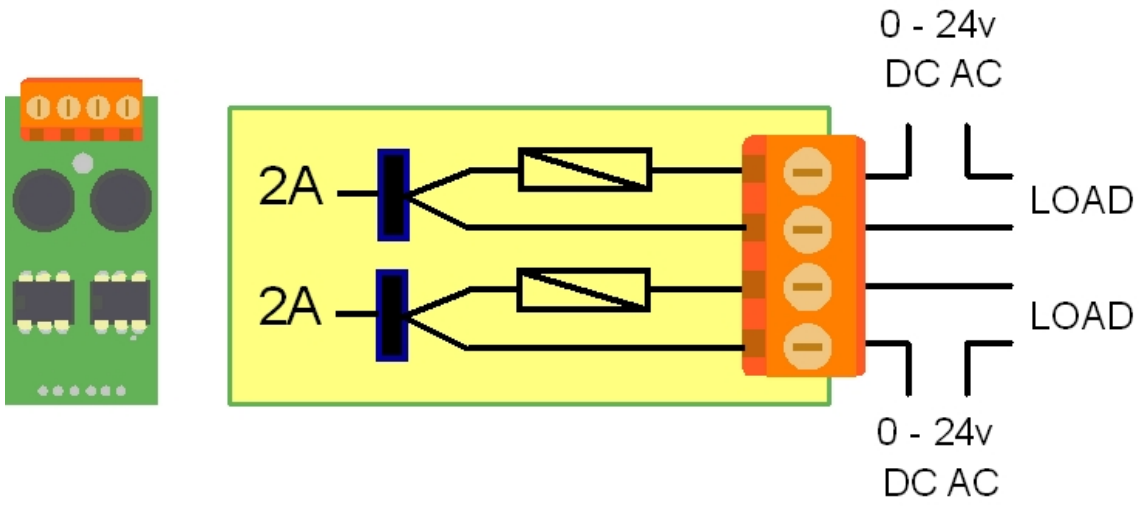
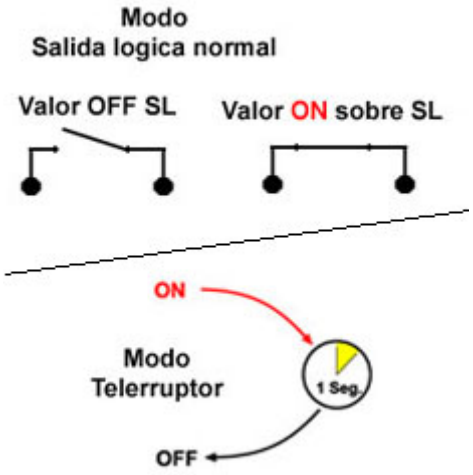



Fig: Salida lógica a transistor

1.9.4 SALIDAS LÓGICAS TEMPORIZADAS

Esta función es especialmente útil en el uso de telerruptores: su funcionamiento es idéntico al de las Salidas Lógicas, exceptuando que al enviarle un "1" (ON, nivel alto) la salida únicamente estará a nivel alto durante 500ms. Hasta que la orden no varíe nuevamente de un nivel bajo a un alto, la salida lógica temporizada permanecerá a nivel

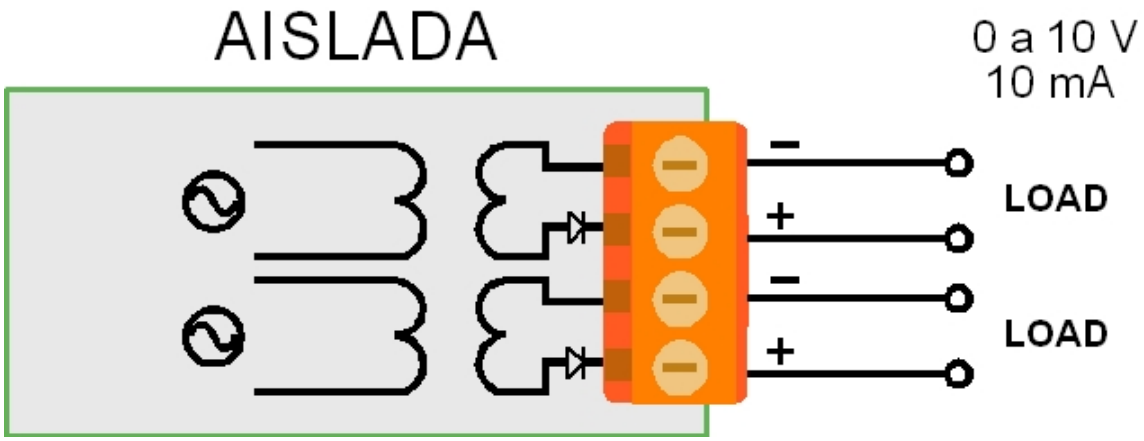
bajo. Para este tipo de salida se utiliza exactamente el mismo módulo que para las salidas lógicas normales, ya que la temporización se realiza mediante el software interno de las TIR: es suficiente con seleccionarla al determinar los módulos de la TIR que los monte.

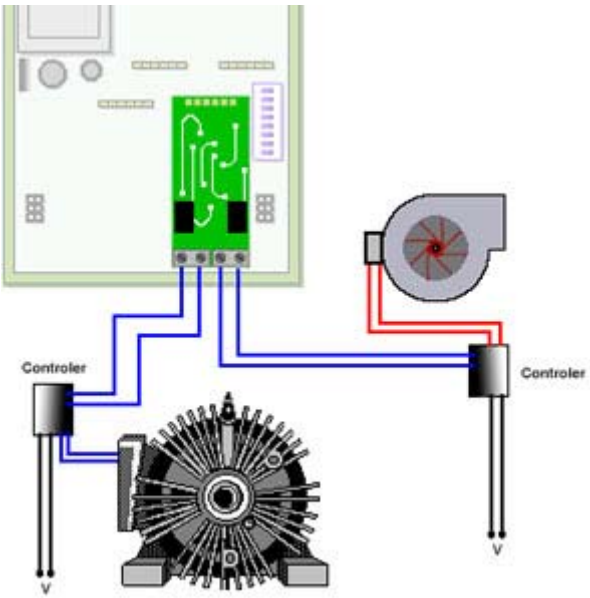



1.9.5 SALIDAS ANALÓGICAS

En la variable asignada le damos el valor deseado (entre 0 y 255 puntos), de manera que en la salida real obtendremos un nivel de tensión (entre 0 y 10 voltios).

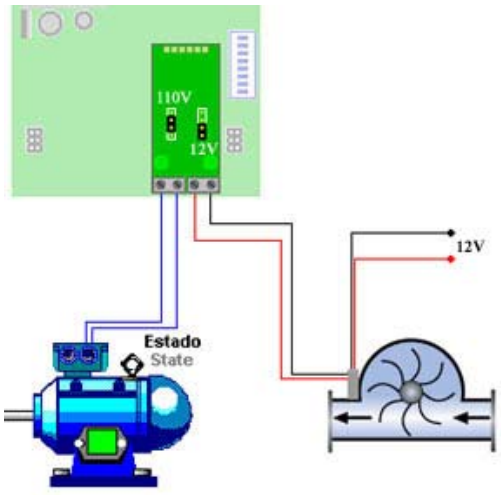
Ejemplo: if EntradaLuzExterior=100 then LuzInterior=100






1.9.6 CONTADORES

Utilizando el mismo módulo que las entradas lógicas (es suficiente con seleccionar el tipo Contador al determinar los módulos de la TIR modular) es posible contar una serie de pulsos aplicados al módulo, ya sean de 12V, 110V o 220V. El propio módulo incorpora un integrador para eliminar falsos pulsos, por lo que el conteo queda limitado a un máximo de 5 pulsos por segundo. El contador detecta flancos ascendentes.




Contador alimentación externa

1.10. PAUSE

Instrucción: Pause(P1)

Parámetros: P1: Tiempo de pausa, en milisegundos (mínimo 500).

Devuelve: Nada.

Descripción: La función Pause( ) realiza una pausa de tantos milisegundos como se le indiquen en el parámetro 1. El tiempo mínimo es de 500ms (medio segundo).

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| Pause(2000) | Pausa de 2000ms = 2 segundos. |

1.11. LLAMADA A SUBRUTINA

Instrucción: Llamada a Subrutina.

Parámetros: Ninguno.

Devuelve: Nada.

Descripción: Para proceder a ejecutar la SUBRUTINA declarada en SubNombre( ), simplemente se escribe su nombre seguido de un paréntesis vacio.

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| Sub CalculoTemp()<br>. . .<br>EndSub | La SUBRUTINA tiene el Nombre CalculoTemp() |
| if Temperatura>5 then<br>CalculoTemp()<br>endif | Si Temperatura>5 salta a SUBRUTINA CalculoTemp() |

1.12. SUB

Función: Sub P1( )

Parámetros: P1: Nombre de la subrutina que se desea crear.

Devuelve: Nada.

Descripción: La función Sub ... ( ) es la declaración del inicio de una SUBRUTINA, de nombre P1. El final de subrutina se indica mediante la función EndSub( ), mientras que la llamada a subrutina se efectuará simplemente escribiendo su nombre seguido del paréntesis vacio. Es necesario que las subrutinas se encuentren declaradas al principio del programa, antes de cualquier llamada a éstas: en caso contrario el compilador alertará de que la función llamada no existe.

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| Sub Cuadrado()<br>a = b*b<br>EndSub<br>b = 7<br>Cuadrado()<br>Print(0,0,a) | Se declara la subrutina Cuadrado().<br>Se indica que a tome el valor *b por b*.<br>Finaliza la declaración de subrutina.<br><br>b toma el valor 7.<br>Se llama a la subrutina Cuadrado().<br>Se muestra el resultado en la pantalla del display. |

1.13. ENDSUB

Función: EndSub

Parámetros: Ninguno.

Devuelve: Nada.

Descripción: La instrucción EndSub se utiliza para indicar el final de declaración de SUBRUTINA. El inicio de la declaración se indica mediante la función Sub…()

Ejemplos

| INSTRUCCION | ACCION |
|---|---|
| Sub CalculoTemp()<br>Temperatura=Temperatura+2<br>If Temperatura>80 Then<br>print(1,1,Fallo Sensor)<br>endif<br>EndSub | Declaración de la SUBRUTINA con el nombre Calculo Temp<br>SUBRUTINA Calculo Temp<br>Fin SUBRUTINA |

## 1.14. LISTA DE FUNCIONES

### 1.14.1 TEMPORIZADORES O TIMERS

Descripción: Un temporizador (o *timer*) es un elemento utilizado para medir una cantidad de tiempo prefijada en segundos (máximo 999.999). Se configuran en el menú *Configurar-->Temporizaciones* (o bien pulsando la tecla F8), asignándoles un nombre y un valor de conteo (en segundos).

Las funciones que trabajan con los temporizadores son:

1. ArrancaTimer( ): Inicia el conteo del temporizador

Instrucción: ArrancaTimer(P1)

Parámetros: P1 (nombre del temporizador que se desea iniciar)

Devuelve: Nada

2. TimerCumplido( ): Devuelve SI o NO en función de si se ha llegado o no al valor de conteo.

Instrucción: TimerCumplido(P1)

Parámetros: P1 (nombre del temporizador del que se desea conocer su estado)

Devuelve: SI - NO

3. ParaTimer( ): Detiene el conteo y deja el contador a 0

Instrucción: ParaTimer(P1)

Parámetros: P1 (nombre del temporizador que se desea detener)

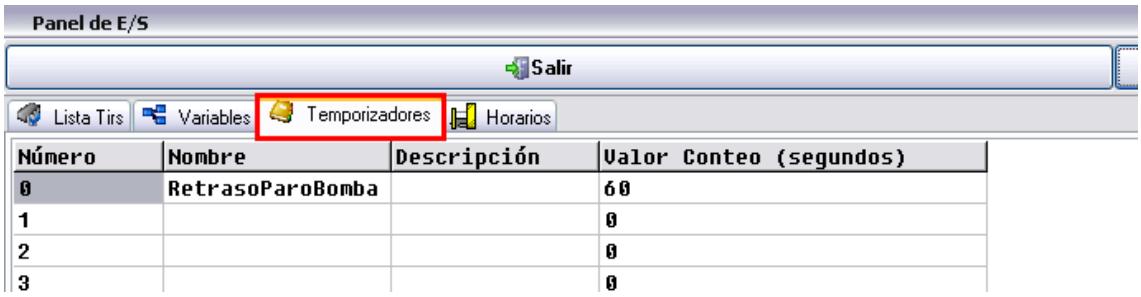
Devuelve: Nada

4. ValorContadorTimer( ): Devuelve el valor de conteo (del instante en que se ejecuta) en unidades de 1 segundo.

Instrucción: ValorContadorTimer(P1)

Parámetros: P1 (nombre del temporizador del que se desea conocer su valor de conteo actual)

Devuelve: El número por el que va contando el temporizador




Ejemplos

| INSTRUCCIÓN | ACCION |
|---|---|
| ArrancaTimer(Timer_1)<br><br>if TimerCumplido(Timer_1) = SI then<br>ParaTimer(Timer_1)<br>BombaPiscina = OFF<br>else<br>BombaPiscina = ON<br>endif | El temporizador Timer1 empieza a contar.<br>Se comprueba si ha terminado de contar,<br>Resetea el timer<br>se realiza la acción deseada<br>Si no ha terminado de contar<br>se realiza la acción deseada |
| Conteo=ValorContadorTimer(Timer1) | Se guarda en la variable *Conteo* el valor por el que va contando el temporizador Timer1 en este instante (en unidades de un segundo). |

1.14.2. RELOJ

Las funciones que trabajan leyendo del reloj-calendario interno del Regulador son:

1. GetAnio( ): Devuelve el número de años transcurridos desde 1800.

Descripción: La función GetAnio( ) devuelve el número de años que han transcurrido desde 1800.

Por ejemplo, si al ejecutar Anio = GetAnio( ) la variable *Anio* toma el valor 193 es que el año actual en la memoria del Regulador es 1800+193 = 1993.

Instrucción: GetAnio( )

Parámetros: Ninguno.

Devuelve: El año guardado en la fecha del regulador (años transcurridos desde 1800)

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Year = GetAnio() | Leemos el año actual y los guardamos en la variable Year.<br>Si estamos en el 2007, guardará el valor 207, ya que 2007-1800=207. |

2. GetHora( ):Devuelve la hora actual del reloj-calendario interno del Regulador (0-23).

Descripción: La función GetHora( ) devuelve la hora actual guardada en el reloj-calendario interno del Regulador (número del 0 al 23).

Función: GetHora( )

Parámetros: Ninguno.

Devuelve: La hora actual guardada en la fecha del regulador (número del 0 al 23).

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Horas = GetHora( ) | Leemos la hora actual y la guardamos en la variable *Horas*. |

3. GetMinutos( ): Devuelve los minutos actuales del reloj-calendario interno del Regulador (0-59).

Descripción: La función GetMinutos( ) devuelve el número de minutos de la hora guardada en el reloj-calendario interno del Regulador (0-59)

Función: GetMinutos( )

Parámetros: Ninguno.

Devuelve: Los minutos guardados en la hora del Regulador (0-59).

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Minutos = GetMinutos( ) | Leemos los minutos de la hora actual y los guardamos en la variable Minutos. |

4. GetSegundos( ): Devuelve los segundos actuales del reloj-calendario interno del Regulador (0-59).

Descripción: La función GetSegundos( ) devuelve el número de segundos de la hora guardada en el reloj-calendario interno del Regulador (0-59).

Función: GetSegundos( )

Parámetros: Ninguno.

Devuelve: Los segundos guardados en la hora del Regulador (0-59).

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Segundos = GetSegundos( ) | Leemos los segundos de la hora actual y los guardamos en la variable Segundos |

5. GetMes( ): Devuelve el mes actual del reloj-calendario interno del Regulador (1-12).

Descripción: La función GetMes( ) devuelve el número de mes actual según la fecha guardada en el reloj-calendario interno del Regulador (1-12).

Función: GetMes( )

Parámetros: Ninguno.

Devuelve: El mes guardado en la fecha del regulador (1-12).

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Mes = GetMes( ) | Leemos el mes actual y los guardamos en la variable Mes. |

6. GetDia( ): Devuelve el día actual del reloj-calendario interno del Regulador (1-31)
Descripción: La función GetDia( ) devuelve el día actual según la fecha guardada en el reloj-calendario interno del Regulador (1-31).
Función: GetDia( )
Parámetros: Ninguno.
Devuelve: El día guardado en la fecha del regulador (1-31)
Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Dia = GetDia( ) | Leemos el día actual y los guardamos en la variable Día. |

7. GetDiaSemana( ): Devuelve el dia de la semana actual del reloj-calendario interno del Regulador (0=domingo, 1=lunes, 2=martes, etc).
Descripción: La función GetDiaSemana( ) devuelve el número correspondiente al día de la semana de la fecha del reloj-calendario interno del Regulador, de forma que Domingo = 0, Lunes = 1, Martes = 2, Miércoles = 3, Jueves = 4, Viernes = 5 y Sábado = 6.
Función: GetDiaSemana( )
Parámetros: Ninguno.
Devuelve: El día de la semana guardado en la fecha del regulador (0-6).
Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| DiaDeSemana = GetDiaSemana( ) | Leemos el día actual de la semana y lo guardamos en la variable *DiaDeSemana*. Si devuelve un 4 es que estamos a Jueves. |

PUESTA EN HORA DEL REGULADOR DESDE EL PC: El Regulador tiene en memoria un reloj-calendario interno del que podemos leer y editar los datos según sea necesario (por ejemplo para configurar un Horario). Para ponerlo de forma automática

en hora (transferirle fecha y hora del PC con el que se esté editando), el proceso a seguir es:

1.- Compilar un programa cualquiera (Tecla F9).

2.- Abrir la ventana de Comunicaciones con el Regulador (tecla F10).

3.- Seleccionar tipo de bus (RS o RC7) adecuado.

4.- Seleccionar puerto COM adecuado.

5.- Pulsar sobre el botón *Configurar Regulador*.

6.- Seleccionar el código adecuado.

7.- Pulsar sobre el botón *Poner en Hora*.

Estas funciones que se han visto hasta ahora eran las funciones que trabajan leyendo desde el reloj-calendario interno del regulador. Vamos a ver ahora las funciones que trabajan escribiendo en el reloj-calendario interno del Regulador son:

1. SetAnio( ): Establece el año guardado como actual en la fecha interna del Regulador.

Descripción: La función SetAnio(P1) establece como actual el año que le pasemos como parámetro P1

Instrucción: SetAnio(P1)

Parámetros: P1: año que se desea establecer como actual en la fecha del regulador

Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| SetAnio(2007) | En la fecha guardada por el regulador, el año actual pasará a ser 2007. |

2. SetHora( ):Establece la hora guardada como actual en la fecha interna del Regulador.

Descripción: La función SetHora(P1) establece como actual la hora que le pasemos como parámetro P1.

Instrucción: SetHora(P1)

Parámetros: P1: hora que se desea establecer como actual en la fecha del regulador

Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| SetHora(14) | En la fecha guardada por el regulador, la hora actual pasará a ser las dos de la tarde, dejando el valor de minutos tal como estaba. |

3. SetMinutos( ):Establece los minutos guardados como actuales en la fecha interna del Regulador.

Descripción: La función SetMinutos(P1) establece como actual el número de minutos que le pasemos como parámetro P1

Instrucción: SetMinutos(P1)

Parámetros: P1: minutos que se desean establecer como actual en la fecha del regulador

Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| SetMinutos(16) | En la fecha guardada por el regulador, los minutos de la hora actual pasarán a ser 16 (sin tocar para nada el número de hora). |

4. SetSegundos( ):Establece los segundos guardados como actuales en la fecha interna del Regulador.

Descripción: La función SetSegundos(P1) establece como actual el número de segundos que le pasemos como parámetro P1

Instrucción: SetSegundos(P1)

Parámetros: P1: segundos que se desean establecer como actuales en la fecha del regulador

Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| SetSegundos(16) | En la fecha guardada por el regulador, los segundos de la hora actual pasarán a ser 16 (sin tocar para nada el número de hora ni minutos). |

5. SetMes( ):Establece el mes guardado como actual en la fecha interna del Regulador.
Descripción: La función SetMes(P1) establece como actual el mes que le pasemos como parámetro P1
Instrucción: SetMes(P1)
Parámetros: P1: mes que se desea establecer como actual en la fecha del regulador
Devuelve: Nada
Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| SetMes(10) | En la fecha guardada por el regulador, el mes actual pasará a ser Octubre. |

6. SetDia( ):Establece el día del mes guardado como actual en la fecha interna del Regulador.
Descripción: La función SetDia(P1) establece como actual el día del mes que le pasemos como parámetro P1.
Instrucción: SetDia(P1)
Parámetros: P1: día del mes que se desea establecer como actual en la fecha del regulador.
Devuelve: Nada.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| SetDia(14) | En la fecha guardada por el regulador, el día del mes actual pasará a ser 14. |

7. SetDiaSemana( ): Establece el día de la semana guardado como actual en la fecha interna del Regulador. Hay que tener en cuenta que Domingo=0, Lunes=1, Martes=2, Miércoles=3, Jueves=4, Viernes=5 y Sábado=6.

Descripción: La función SetDiaSemana(P1) establece como actual el dia de la semana que le pasemos como parámetro P1, teniendo en cuenta que Domingo=0, Lunes=1, Martes=2, Miércoles=3, Jueves=4, Viernes=5 y Sábado=6.

Instrucción: SetDiaSemana(P1)

Parámetros: P1: día de la semana que se desea establecer como actual en la fecha del regulador.

Devuelve: Nada.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| SetDiaSemana(4) | En la fecha guardada por el regulador, el día de la semana actual pasará a ser Jueves. |

1.14.3. HORARIOS

Cada Horario (adecuadamente configurado y guardado en la tabla de horarios -Tecla F8-) tiene 5 parámetros:

Hora de inicio, Minutos de inicio, Hora de fin, Minutos de fin y Días de la semana activos. Un horario se considera *activo* (dentro de horario) cuando:

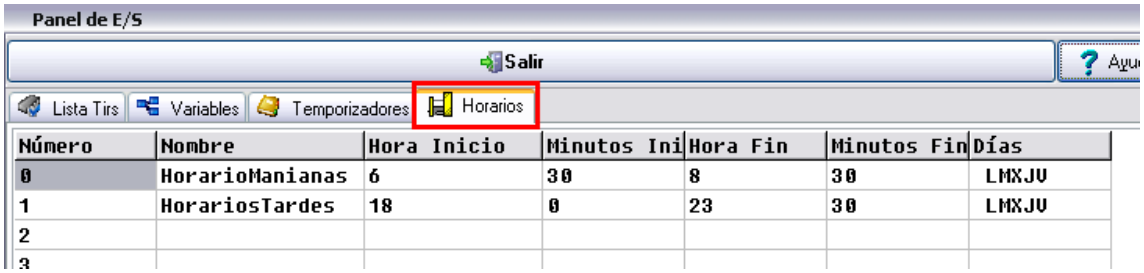
(HoraInicio:MinutoInicio) <= (HoraActual del Regulador)

AND

(HoraFin:MinutoFin) >= (HoraActual del Regulador)

Y además el día actual del Regulador está en la lista de días del horario.

NOTA: Cuando se arranca por primera vez el Regulador, la configuración de los horarios es la que se haya especificado en el Panel de Entrada/Salida (Tecla F8). Al utilizar las funciones de horario serán los valores determinados por éstas los que se guardarán en la Tabla de Horarios del Regulador (Panel de Entradas/Salidas -Tecla F8-, pestaña *Horarios, como se ve en la siguiente figura*)




| Número | Nombre | Hora Inicio | Minutos Ini | Hora Fin | Minutos Fin | Días |
|---|---|---|---|---|---|---|
| 0 | HorarioManianas | 6 | 30 | 8 | 30 | LMXJV |
| 1 | HorariosTardes | 18 | 0 | 23 | 30 | LMXJV |
| 2 | | | | | | |
| 3 | | | | | | |

Vamos a ver a continuación las funciones que trabajan con los Horarios:

1. EstaDentroDeHorario( ): Devuelve el estado de un Horario (activo/inactivo)

Descripción: La función EstaDentroDeHorario( ) indica si el Horario P1 especificado en la Tabla de Horarios (en el Panel de Entrada/Salida -Tecla F8-) está activo, devolviendo un SI o un NO.

Instrucción: EstaDentroDeHorario(P1)

Parámetros: P1: Horario del que se desea conocer su estado (si está activo o no).

Devuelve: Si – No

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| i = EstaDentroDeHorario(HorarioBomba)<br>if i = si then<br>SalidaBomba = on<br>Else<br>SalidaBomba = off<br>Endif | En la variable "i" tendremos Si o NO<br>Si estamos en el horario (SI), entonces:<br>--> Arrancamos la bomba<br>Si NO estamos en el horario, entonces:<br>--> Paramos la bomba |

2. Días( ): Establece los días de la semana en los que estará activo un Horario

Descripción: La función Días( ) establece los días de la semana en los que está activo un Horario (determinado por el parámetro P1). Para ello se utiliza el parámetro P2, que corresponde a un número determinado por el código de cada día de la semana:

Lunes = 1

Martes = 2

Miércoles = 4

Jueves = 8

Viernes = 16

Sábado = 32

Domingo = 64

Para activar más de un día de la semana en un mismo horario, se le pasará el parámetro P2 correspondiente a la suma de todos los códigos de todos los días que se quieran activar. Por ejemplo:

Si se quiere activar el HorarioCaldera únicamente los miércoles (código 4), se ejecutará: Días(HorarioCaldera,4).

Si se quiere activar el HorarioCaldera los lunes (código 1) , jueves (código 8) y domingos (código 64) se ejecutará:

Días(HorarioCaldera,73) ya que 1+8+64 = 73.

Instrucción: Días(P1,P2)

Parámetros: P1: Horario del que se desean establecer nuevos días activos.

P2: Días de la semana en los que está activo un determinado Horario.

Devuelve: Nada.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| HoraIni(HorarioCaldera,13)<br>MinIni(HorarioCaldera,15)<br>HoraFin(HorarioCaldera,14)<br>MinFin(HorarioCaldera,30)<br>Dias(HorarioCaldera,3) | El horario *HorarioCaldera* estará activo desde las 13:15 hasta las 14:30 todos los lunes y martes. |

3. HoraIni( ): Establece la hora del momento de inicio de un Horario (0-23).
Descripción: La función HoraIni( ) establece un nuevo valor para la hora de inicio de un determinado Horario
Instrucción: HoraIni(P1,P2)
Parámetros: P1: Horario del que se desea establecer una nueva hora de inicio.
P2: Nueva hora de inicio
Devuelve: Nada

4. HoraFin( ):Establece la hora del momento de fin de un Horario (0-23).
Descripción: La función HoraFin( ) establece un nuevo valor para la hora de fin de un determinado Horario
Instrucción: HoraFin(P1,P2)
Parámetros: P1: Horario del que se desea establecer una nueva hora de fin
P2: Nueva hora de fin
Devuelve: Nada

5. MinIni( ): Establece los minutos del momento de inicio de un Horario (0-59).
Descripción: La función MinIni( ) establece un nuevo valor de minutos para el momento de inicio de un determinado horario
Instrucción: MinIni(P1,P2)
Parámetros: P1: Horario del que se desea establecer un nuevo minuto de inicio.
P2: Nuevo minuto de inicio.
Devuelve: Nada.

6. MinFin( ):Establece los minutos del momento de fin de un Horario (0-59).
Descripción: La función MinFin( ) establece un nuevo valor de minutos para el momento de fin de un determinado Horario.
Instrucción: MinFin(P1,P2)
Parámetros: P1: Horario del que se desea establecer un nuevo minuto de fin.
P2: Nuevo minuto de fin.
Devuelve: Nada

7. InputDias( ):Devuelve el código de los días activos de un Horario

Descripción: La función InputDias( ) devuelve la suma de los códigos de los días de la semana en que se activará un determinado horario P1. Estos códigos están asignados, para cada día de la semana, del siguiente modo:

Lunes = 1

Martes = 2

Miércoles = 4

Jueves = 8

Viernes = 16

Sábado = 32

Domingo = 64

Para activar más de un día de la semana en un mismo horario, el código correspondiente será la suma de los códigos de todos los días que se quieran activar.

Instrucción: InputDias(P1)

Parámetros: P1: Horario del que se desea conocer los días de la semana en que se activará.

Devuelve: Suma de los códigos de los días de las semana activos.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| i = InputDias(H1) | Carga en la variable i el código correspondiente a los días activos del horario H1. Por ejemplo: si devuelve 73, el horario H1 estará activado los lunes (código 1), jueves (código 8) y domingos (código 64), ya que 1+8+64 = 73. |

8. InputHoraFin( ): Devuelve la hora establecida como fin de un Horario.

Descripción: La función InputHoraFin( ) devuelve la hora establecida como momento de fin del horario P1.

Instrucción: InputHoraFin(P1)

Parámetros: P1: Horario del que se desea conocer la hora de fin.

Devuelve: Hora configurada como fin del horario P1.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| i = InputHoraFin(H1) | Carga en la variable i la hora establecida como momento de fin del horario H |

9. InputHoraIni( ): Devuelve la hora establecida como inicio de un Horario( )

Descripción: La función InputHoraIni( ) devuelve la hora establecida como momento de inicio del horario P1.

Instrucción: InputHoraIni(P1)

Parámetros: P1: Horario del que se desea conocer la hora de inicio

Devuelve: Hora configurada como inicio del horario P1

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| i = InputHoraIni(H1) | Carga en la variable i la hora establecida como momento de inicio del horario H1. |

10. InputMinIni( ):Devuelve los minutos del momento establecido como inicio de un Horario

Descripción: La función InputMinIni( ) devuelve el valor actual -en minutos- asignado al momento de inicio de un determinado Horario

Instrucción: InputMinIni(P1)

Parámetros: P1: Horario del que se desea conocer el valor en minutos de su momento de inicio.

Devuelve: Minutos (0-59)

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| i = InputMiniIni(H1) | Guarda en la variable i los minutos del momento en que el horario H1 se activa. |

11. InputMinFin( ): Devuelve los minutos del momento establecido como fin de un Horario.

Descripción: La función InputMinFin( ) devuelve los minutos establecidos como momento de fin del horario P1.

Instrucción: InputMinFin(P1)

Parámetros: P1: Horario del que se desea conocer los minutos del momento de fin.

Devuelve: Minutos configurada como momento de fin del horario P1.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| i = InputMinFin(H1) | Carga en la variable i los minutos establecidos como momento de fin del horario H1. |

1.14.4. CLOCK

En este caso sólo tenemos una función que es la función GetMilisegundos( ):

1. GetMilisegundos( )

Descripción: La función GetMilisegundos( ) devuelve el valor del contador interno de milisegundos del Regulador. Dicho contador se pone a 0 cada vez que arrancamos el Regulador. Se puede utilizar como alternativa a un temporizador para saber cuántos

milisegundos han pasado entre 2 eventos.

Hay que tener presente que el contador empieza otra vez de cero cuando su conteo llega a 2 elevado a 32, es decir, cuando llega al conteo de 4.294.967.296 milisegundos.

Función: GetMilisegundos( )

Parámetros: Ninguno.

Devuelve: El número de milisegundos transcurridos desde que se ha encendido el Regulador.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Variable = GetMilisegundos( ) | En la variable Variable se guarda el número de milisegundos transcurridos desde que se ha encendido el Regulador. |

1.14.5. TIRs

Vemos las funciones relacionadas con los terminales TIR

1. DesHabilitaTir( )

Descripción: La función DesHabilitaTir( ) desactiva la terminal TIR especificada en el parámetro P1. Para volver a utilizar una TIR deshabilitada hay que activarla de nuevo mediante la función HabilitaTir( )

Instrucción: DesHabilitaTir(P1)

Parámetros: P1: número de TIR que se desea deshabilitar.

Devuelve: Nada.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| DesHabilitaTir(3) | Deshabilita la TIR número 3. |

2. EstadoTir( )

Descripción: La función EstadoTir( ) devuelve el estado de una terminal TIR:

OK: TIR habilitada y funcionando correctamente

NOCOMUNICA: Bus de comunicación cortado, o TIR apagada/averiada

DESHABILITADA: El programa no ha habilitado la TIR o ha ejecutado la función DesHabilitarTir( )

ENRESET: La TIR ha sido reseteada (deshabilitada) porque ha ocurrido un corte de suministro eléctrico.

Instrucción: EstadoTir(P1)

Parámetros: P1: Código de la TIR de la que se desea conocer el estado.

Devuelve: La palabra correspondiente al estado de la TIR: OK, NOCOMUNICA, DESHABILITADA o ENRESET.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Variable = EstadoTir(3) | En la variable "Variable" tenemos el estado de la TIR Nº 3. |

3. HabilitaTir( )

Descripción: La función HabilitaTir( ) activa la terminal TIR especificada en el parámetro P1. Al dar tensión al Regulador todas las TIR están deshabilitadas por defecto, por lo que al inicio del programa se deberán habilitar las que vayan a usarse.

Instrucción: HabilitaTir(P1)

Parámetros: P1: número de TIR que se desea habilitar.

Devuelve: Nada.

Un ejemplo de aplicación típico es cuando se corta el suministro eléctrico de la instalación y se paran todas las máquinas: puede evitarse una sobrecarga -cuando vuelva de nuevo el suministro eléctrico- si se van habilitando secuencialmente las TIRs. Véase la ayuda de la función InicioPrograma( )

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| HabilitaTir(3) | Habilita la TIR número 3. |

1.14.6. CONVERSIONES

1. Humedad( )

Descripción: La función Humedad( ) convierte la lectura de humedad de la entrada analógica correspondiente (adecuadamente configurada en el Panel de E/S) en un valor entre 0 y 100%.

Instrucción: Humedad(P1)

Parámetros: P1: Sonda de humedad que se desea leer

Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Variable = Humedad(HumeSala) | En este ejemplo tenemos una sonda de Humedad conectada en la TIR 1 punto 2 (con el nombre de HumeSala). Después de llamar a la función Humedad, obtenemos la humedad en % en la variable "Variable" |

2. Luminosidad( )

Descripción : La función Luminosidad( ) convierte la lectura de luminosidad de la entrada analógica correspondiente (adecuadamente configurada en el Panel de E/S) en un valor entre 0 y 100%.

Instrucción: Luminosidad(P1)

Parámetros: P1: Sonda de luminosidad que se desea leer

Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Variable = Luminosidad(LuzInterior ) | En este ejemplo tenemos una sonda de luminosidad conectada |

| | |
|---|---|
| | en la TIR 1 punto 2 (con el nombre de LuzInterior). Después de llamar a la función Luminosidad(), obtenemos la luminosidad en % en la variable "Variable" |

3. Tension( )

Descripción: La función Tensión( ) se utiliza para convertir el valor en puntos que nos devuelve una Entrada analógica (adecuadamente configurada en el Panel E/S) a una tensión entre 0 y 10V.

Instrucción: Tensión(P1)

Parámetros: P1: Sonda de tensión que se desea leer.

Devuelve: Float.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Variable = Tension(nivel1) | En este ejemplo tenemos una sonda de Nivel conectada en la TIR 1 punto 2 (con el nombre de Nivel1). Después de llamar a la función Tension, obtenemos la tensión entre 0 y 10 voltios en la variable "Variable" |

4. Intensidad( )

Descripción: La función Intensidad( ) se utiliza para convertir el valor en puntos que nos devuelve una Entrada analógica (adecuadamente configurada en el Panel E/S) a una Intensidad entre 0 y 20mA.

Instrucción: Intensidad(P1)

Parámetros: P1: Sonda de intensidad que se desea leer. Devuelve: Float.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Variable = Intensidad(nivel1) | En este ejemplo tenemos un lector de intensidad bomba1. Al llamar a la función intensidad() obtenemos un valor de intensidad entre 0 y 20 miliamperios. |

5. TemperaturaKTY( )

Descripción: La función TemperaturaKTY( ) devuelve la temperatura (en grados centígrados, ºC) correspondiente a la lectura térmica de la sonda asignada a una entrada analógica con nombre P1.

Instrucción: TemperaturaKTY(P1)

Parámetros: P1: Sonda de luminosidad que se desea leer. Devuelve: Float. Temperatura leída por la sonda de temperatura KTY.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Variable=TemperaturaKTY(Sonda interior) | En este ejemplo tenemos una sonda de temperatura conectada en la TIR 1 punto 2 (con el nombre de Sonda interior). Después de llamar a la función TemperaturaKTY(), obtenemos la temperatura en ºC en la variable "Variable" |

1.14.7. DISPLAY

Vemos a continuación las diferentes funciones que podemos encontrar para el display:

1. Beep( )

Descripción: La función Beep( ) controla el zumbador que tiene incorporado el Display de 2 líneas.

El parámetro P1 puede tener 2 valores: ON y OFF, que respectivamente activan y detienen el zumbido. Entre ambos se define la duración del zumbido mediante un Pause( )

Instrucción: Beep(P1)

Parámetros: P1: ON u OFF

Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Beep(ON)<br>Pause(1000)<br>Beep(OFF) | Pone en marcha el zumbador del display.)<br>Emite el zumbido durante un segundo (mil milisegundos).<br>Detiene el zumbador del display |

2. Cls( )

Descripción: La función Cls( ) borra completamente la pantalla del display de 2 líneas

Instrucción: Cls( )

Parámetros: Ninguno

Devuelve: Nada


3. Input( )

Descripción: La función Input( ) devuelve la última tecla que se ha pulsado en el display de 2 líneas.

Los posibles valores son TECLA0, TECLA1, TECLA2, TECLA3, TECLA4, TECLA5, TECLA6, TECLA7, TECLA8, TECLA9, TECLAARRIBA, TECLAABAJO, TECLAPUNTO, TECLAINTRO. Si no se ha pulsado ninguna tecla devuelve el valor NINGUNATECLA

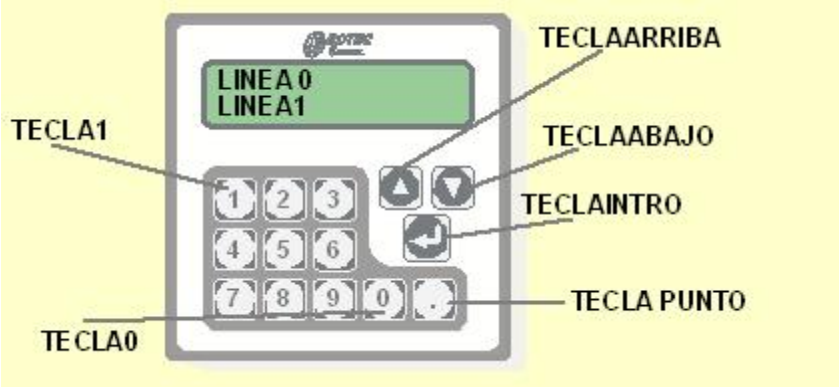
Instrucción: Input( )

Parámetros: Ninguno

Devuelve: Char. Última tecla que se ha pulsado en el display

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Entrada = input( ) | La variable "Entrada" se carga con el nombre de la última tecla pulsada. |




NOTA: Se recomienda llamar *una sola vez* a esta función, a fin de obtener una mayor velocidad al introducir datos por el display.

4. Print( )

Descripción: La función Print( ) muestra textos y valores de variables enel display de 2 líneas y 16 columnas.

Instrucción: Print(P1,P2,P3,P4,P5,P6)

Parámetros: P1: Línea del display en la que se desea escribir. Línea, puede tomar los valor 0 o 1 dependiendo de si se desea escribir en la superior o inferior.

P2: Columna del display en la que se desea escribir. Columna, puede tomar los valores de 0 a 15 (de izquierda a derecha) dependiendo de la columna en que se desea escribir.

P3: Texto o variable (valor) que se mostrará en pantalla. Texto que se desea mostrar (deberá ir entre comillas dobles ") o variable de la que se desea mostrar su valor actual.

P4: Texto o variable (valor) que se mostrará en pantalla. Texto que se desea mostrar (deberá ir entre comillas dobles ") o variable de la que se desea mostrar su valor actual.

P5: Texto o variable (valor) que se mostrará en pantalla. Texto que se desea mostrar (deberá ir entre comillas dobles ") o variable de la que se desea mostrar su valor actual.

P6: Texto o variable (valor) que se mostrará en pantalla. Texto que se desea mostrar (deberá ir entre comillas dobles ") o variable de la que se desea mostrar su valor actual.

Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Print(0,0,"HOLA") |  <br>Imprime en la línea 0, columna 0 el texto *HOLA* |
| Print(0,0,"HOLA")<br>Print(1,0,"VALOR = ",contador) |  <br>Imprime en la línea 0 - columna 0 el texto *HOLA* y en la fila 1 - columna 0 el texto *Valor*= , seguido del valor de la variable "contador". |

Nótese que esta función únicamente sobreescribe las celdas del display que se le manda escribir, dejando el resto tal como estaban. Es decir, al ejecutar Print(1,5,"@") el programa reemplazará el carácter que haya en la quinta columna de la fila inferior del display por una arroba, pero dejará todos los otros caracteres como estaban.

NOTA: El display tiene una frecuencia de refresco de 800ms aproximadamente, si se quiere visualizar un valor que cambie con una frecuencia inferior a ese tiempo, es posible que no se representen en pantalla todos los valores. El visualizador está diseñado para mostrar valores estáticos o de variación lenta.

5. TemperaturaDisplay( )

Descripción: La función TemperaturaDisplay( ) nos indica la temperatura en grados centigrados (ºC) medida por el sensor alojado en el display de 2 líneas.

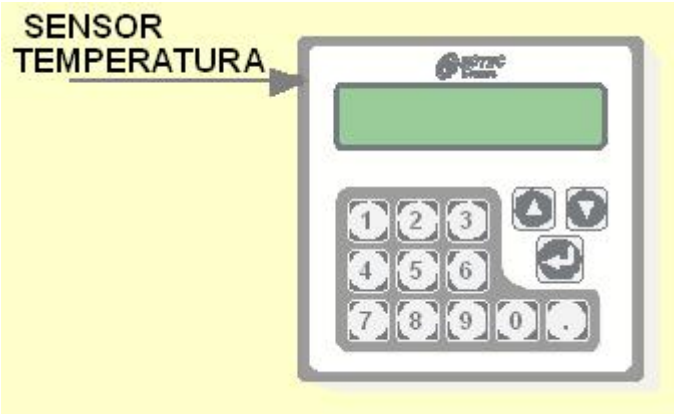
Función: TemperaturaDisplay( )

Parámetros: Ninguno.

Devuelve: Float. La temperatura detectada por el sensor integrado en el display.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Variable = TemperaturaDisplay( ) | En *Variable* se carga la temperatura del display. |



1.14.8. EDICION

1. InicioEdicion( )

Descripción: La función InicioEdicion( ) permite configurar la edición del teclado del display de 2 líneas para la introducción de datos por el usuario.

Esta función se utiliza conjuntamente con las funciones EnEdicion( ) y ValorEditado( ).

La edición termina cuando el usuario pulsa la tecla Intro.

La cantidad máxima de dígitos admitida por esta función es de 6.

Únicamente puede estar activa una función InicioEdicion() al mismo tiempo. En caso de ejecutarse una segunda vez, se perderían los datos con los que estuviera trabajando la primera.

Instrucción: InicioEdicion(P1,P2,P3)

Parámetros: P1: Línea en la que se colocará el cursor para escribir desde el display. Este parámetro (Línea) puede tomar los valores 0 o 1, correspondientes a la línea superior o inferior.

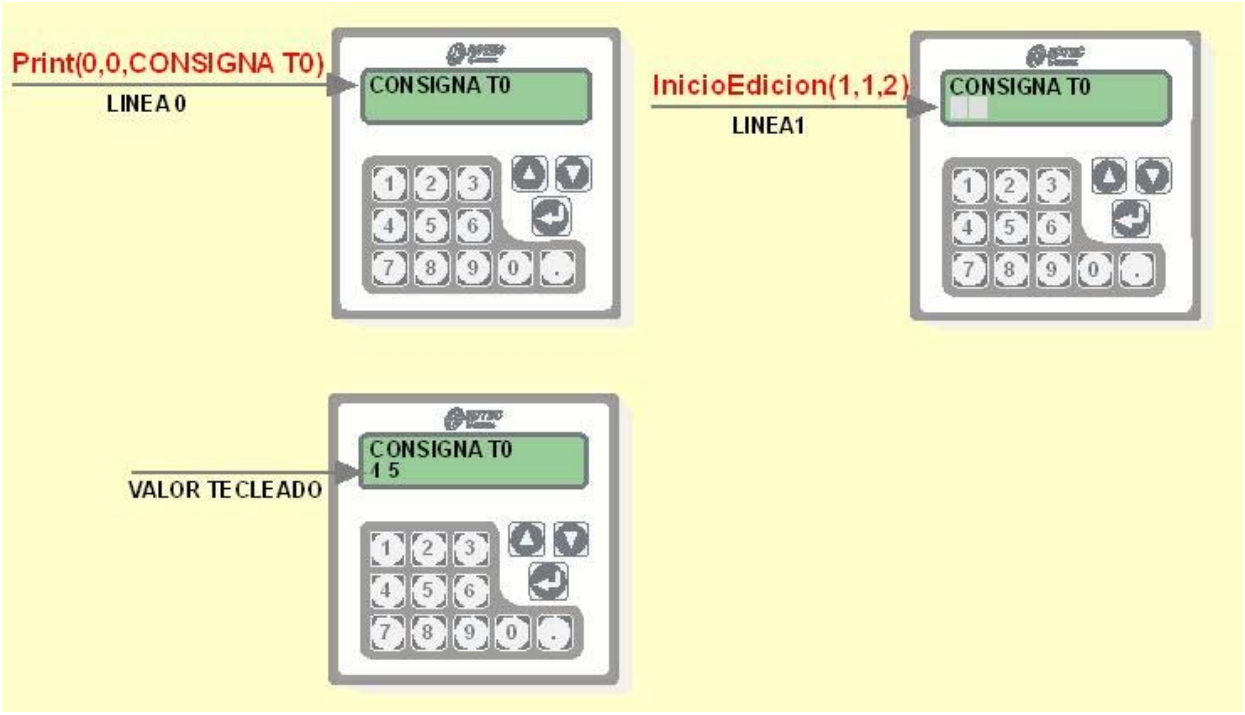
P2: Columna en la que se colocará el cursor para escribir desde el display. Este parámetro (Columna) puede tomar los valores de 0 a 15, correspondientes al número de columna en el que se quiere escribir.

P3: Número de dígitos que se permitirá escribir al usuario. Este parámetro (Tamaño), es el número máximo de dígitos que se permite introducir al usuario (que no podrá ser mayor de 6 dígitos).

Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Print (0,0,"Temperatura?") | Imprime el mensaje "Temperatura?" en la línea 0 columna 0. |
| InicioEdicion (1,0,2) | Permite que el usuario introduzca con el teclado un valor de hasta 2 dígitos en fila 1 columna 0. Se consigue que el usuario establezca un valor entre 0 y 99. |




2. EnEdicion( )

Descripción: La función EnEdicion( ) indica si el usuario ha terminado de editar los dígitos y pulsado INTRO en el display de 2 líneas. El inicio de la edición se ejecuta con la función InicioEdicion( ). EnEdición( ) devuelve SI si se sigue editando y NO cuando ya ha terminado.

Instrucción: EnEdicion( )

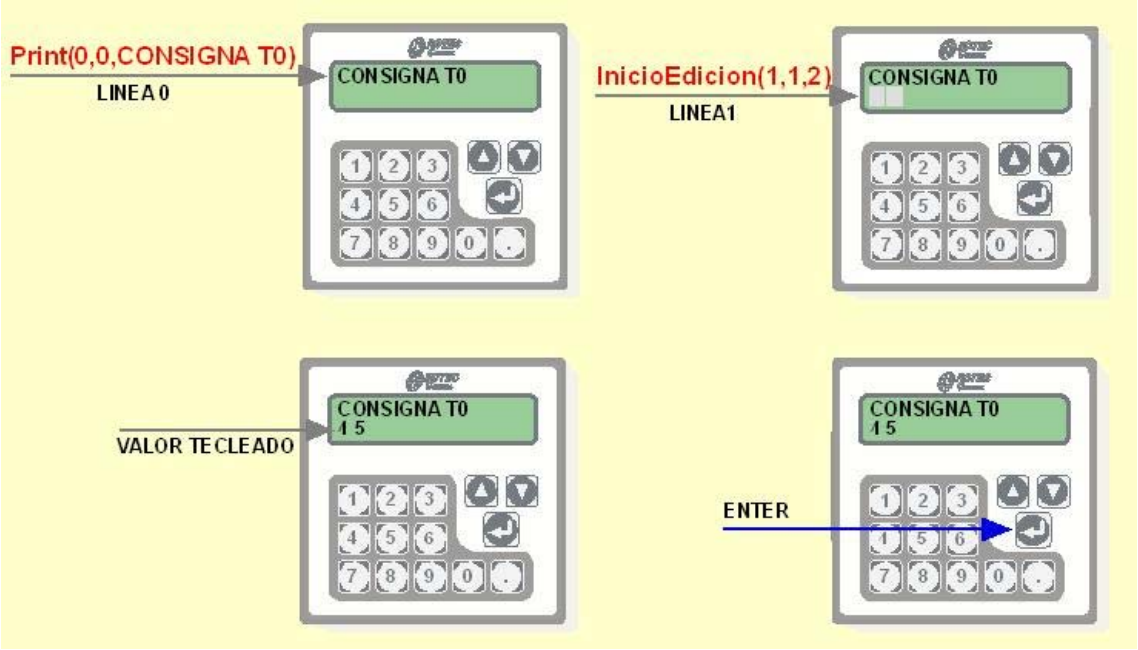
Parámetros: Ninguno

Devuelve: Char. SI o NO, dependiendo de si se ha terminado o no de introducir datos por el display.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Print (0,0,"Consigna de T0") | Imprime el texto Consigna de T0 |
| InicioEdicion(1,0,2) | Se permite que el usuario teclee un valor de hasta 2 dígitos en línea 1 columna 0. Se consigue que el usuario establezca un valor entre 0 y 99. |

| | |
|---|---|
| Variable = EnEdicion( ) | En la variable "Variable" tendremos un SI si sigue la edición (es decir, si aun no se ha pulsado la tecla INTRO después de introducir un valor) o un NO si aun no ha finalizado la adquisición del dato. |




3. ValorEditado( )

Descripción: La función ValorEditado( ) devuelve el valor tecleado por el usuario mediante el display de 2 líneas. El inicio de la edición se ejecuta con la función InicioEdicion( ), mientras que con la función EnEdicion( ) sabemos si el usuario ha terminado la edición (pulsando la tecla "intro" ).

Instrucción: ValorEditado( )

Parámetros: Ninguno

Devuelve: Float. El dato introducido por display mediante la función InicioEdicion( ).

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Print (0,0,"Consigna de T0") | Imprime el texto Consigna de T0 |
| InicioEdicion(1,0,2) | Se permite que el usuario teclee un valor de |

| | |
|---|---|
| | hasta 2 dígitos en línea 1 columna 0. Se consigue que el usuario establezca un valor entre 0 y 99. |
| Editando = EnEdicion( ) | En la variable "Editando" tenemos un SI si sigue la edición (es decir si aun no se ha pulsado la tecla intro después de introducir un valor), o sea no ha finalizado la adquisición del dato. |
| if (Editando=NO) then<br>resultado = ValorEditado( )<br>endif | Si en la variable "Editando" tenemos un NO, se recoge el valor editado en la variable "resultado". |

1.14.9. EEPROM

1. Poke( )

Descripción: La función Poke se utiliza para grabar en la EEPROM interna del Regulador. La EEPROM es un tipo especial de memoria que tiene la característica de que no pierde su valor cuando se va la alimentación eléctrica. La función escribe el contenido P2 (puede ser tanto un literal como una variable, de la que grabará su valor) en la posición P1. Téngase presente que la EEPROM tiene una vida limitada de aproximadamente 1.000.000 de escrituras.

ATENCIÓN: La función Poke( ) no tiene en cuenta si ya existe algún dato en la posición a grabar, por lo que la sobreescribirá irremediablemente!!

Como opción, la función devuelve, en una variable previamente creada por el usuario (a la que se recomienda nombrar como siguiente), la posición de la EEPROM en donde grabar el próximo dato. Es decir, una vez realizada la escritura devuelve la posición siguiente en la que se puede grabar sin temor a sobreescribir el dato que acaba de grabar. Este cálculo lo realiza de acuerdo con el espacio ocupado por el dato escrito:

| Espacio ocupado en EEPROM por cada tipo de variable | |
|---|---|
| Char | 1 Byte |

| | |
|---|---|
| Byte | 1 Byte |
| Integer | 2 Bytes |
| Long | 4 Bytes |
| Float | 4 Bytes |
| Horario | 5 Bytes |

Función: Poke(P1, P2)

Parámetros: P1: Posición en EEPROM, literal o variable, en la que se quiere grabar. Es decir, P1, especifica en qué lugar de la EEPROM se quiere grabar, es un valor entre 0 y 799 y puede ser una Variable o un Literal (Para grabar en la EEPROM se tiene que tener presente la longitud -en bytes- que ocupa el dato guardado en la posición anterior). P2: Dato que se desea grabar, literal o variable, o sea, especifica el literal o la variable que se escribe en la EEPROM. En caso de ser una variable, se guardará el contenido de ésta.

Devuelve: Opcionalmente, un byte (posición siguiente de la EEPROM)

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Poke(4,var_1) | Escribimos el contenido de la variable "var_1" a partir del 4º byte de la EEPROM. |
| Siguiente = Poke(4,var_1) | Opcional: en la variable "Siguiente" se guarda la siguiente posición libre de la EEPROM |

2. Peek( )

Descripción: La función Peek se utiliza para leer de la EEPROM interna del Regulador. La EEPROM es un tipo especial de memoria que tiene la característica de que no pierde su valor cuando se va la alimentación eléctrica.

Como opción, la función devuelve, en una variable previamente creada por el usuario (a la que se recomienda nombrar como siguiente), la posición de la EEPROM de donde

leer el próximo dato. Este cálculo lo realiza de acuerdo con el espacio ocupado por el dato escrito:

| Espacio ocupado en EEPROM por cada tipo de variable | |
|---|---|
| Char | 1 Byte |
| Byte | 1 Byte |
| Integer | 2 Bytes |
| Long | 4 Bytes |
| Float | 4 Bytes |
| Horario | 5 Bytes |

Instrucción: Peek(P1,P2)

Parámetros: P1: Lugar de la EEPROM del que se desea leer. Es decir, P1 especifica en qué lugar de la EEPROM se quiere leer. Es un valor entre 0 y 799 que puede ser un literal o el contenido de una variable. Para leer la posición siguiente en la EEPROM se tiene que tener presente la longitud que ocupa la variable o literal escrita en la posición anterior

P2: Variable en la que guardar el dato leído de la EEPROM, o sea, especifica la variable en la cual se deposita el dato leído de la EEPROM

Devuelve: Opcionalmente, la siguiente posición de memoria después de la del dato leído.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Peek(4,Var_1) | Lee a partir del 4º byte de la EEPROM y guarda el dato en la variable Var_1. |
| Siguiente = Peek(4,Var_1) | Opcional: se guarda en la variable "Siguiente" la posición en la que leer el próximo dato. |

## 1.14.10. MATEMATICAS

Vemos a continuación las funciones matemáticas de las que disponemos:

1. ValorAbsoluto Abs( )

Descripción: Devuelve el valor absoluto del parámetro P1.

Instrucción: Abs(P1)

Parámetros: P1: variable de la que se desea calcular el valor absoluto.

Devuelve: Valor absoluto del parámetro P1.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Var_1 = Abs(Var_2) | Si Var_2 es positiva, Var_1 valdrá Var_2.<br>Si Var_2 es negativa, Var_1 valdrá -(Var_2). |

2. Suma +

Descripción: Realiza la suma de dos operandos P1 y P2.

Función: +

Parámetros: P1: Valor que se desea sumar.

P2: Valor que se desea sumar

Devuelve: La suma de P1 y P2

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Suma = 12 + 13 | Se guardará en la variable *Suma* el valor 25, ya que 12+13=25. |

3. Resta -

Descripción: Realiza la resta de dos operandos P1 y P2

Función: -

Parámetros: P1: Valor del que se desea restar. P2: Valor que se desea restar.

Devuelve: La resta de P1 y P2

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Resta = 15 - 13 | Se guardará en la variable *Resta* el valor 2, ya que 15-13=2. |

4. Multiplicación *

Descripción: Realiza la multiplicación de dos operandos P1 y P2.

Función: *

Parámetros: P1: Valor que se desea multiplicar. P2: Valor que se desea multiplicar

Devuelve: El producto de P1 y P2

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Producto = 3 * 4 | Se guardará en la variable *Producto* el valor 12, ya que 3*4=12. |

5. División

Descripción: Realiza la división entre dos operandos P1 y P2

Instrucción: /

Parámetros: P1: Valor que se desea dividir. P2: Valor que se desea dividir.

Devuelve: El cociente de dividir P1 y P2

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Cociente = 12/4 | Se guardará en la variable *Cociente* el valor 3, ya que 12/4=3. |

6. Distinto de <>

Descripción: La instrucción <> (Distinto de) se usa únicamente con IF y devuelve un SI (un 1) en el caso de que dos parámetros sean distintos entre sí.

Instrucción: <> (Distinto de)

Parámetros: Ninguno

Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| if Var_1 <> Var_2 then<br>IniciarSistema()<br>endif | Si la variable Var_1 tiene en este momento un valor distinto a Var_2, se ejecutará la subrutina IniciarSistema(). |

7. Resto %

Descripción: Realiza la división entre dos operandos (P1 y P2) y devuelve el resto

Función: %

Parámetros: P1: Valor que se desea dividir. P2: Valor que se desea dividir.

Devuelve: El resto de dividir P1 y P2.

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Resto = 13%2 | Se guardará en la variable *Resto* el valor 1, ya que el resto de dividir 13 entre 2 es 1. |

NOTA: Esta función es útil, por ejemplo, para determinar si un número es par o impar: se divide entre 2 de manera que si el resto es cero el número es par, mientras que si el resto es uno es impar.

1.14.11. TELEFONO

Vemos a continuación la función EnviaSms( ), la única dentro de la categoría Teléfono:

1. EnviaSMS

Descripción: La función EnviaSms( ) permite enviar un mensaje de telefonía de tipo SMS (red GSM).

Instrucción: EnviaSms(P1,P2,P3)

Parámetros: P1: Número de teléfono al que enviar el mensaje SMS. Debe ir sin comillas ni espacios (Ejemplo:666123456)

P2: Texto del mensaje que se desea enviar. Debe ir entre comillas dobles ("Texto")

P3: Variable de la que se quiera enviar el valor. Debe escribirse su nombre sin comillas y se enviará su valor actual.

Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| EnviaSms (666123456,"Alarma: bomba caldera!") | Se envía el mensaje "Alarma: bomba caldera!" al télefono 666123456 |
| EnviaSms (666123456,"Estado de bomba = ", EstadoBomba) | Envía el mensaje "Estado de bomba =" seguido del valor de la variable *EstadoBomba* al teléfono 666123456. Por lo tanto el mensaje recibido sería, dependiendo de si la variable está a 1 o a 0:<br>Estado de bomba recirculación = 1<br>Estado de bomba recirculación = 0 |

Nota: es importante limitar el número de mensajes que se envían, procurando que el programa no entre en un bucle infinito de envíos: además del gasto que esto supondría, también se impedirían las comunicaciones con el PC a través del bus RS. Véase ejemplo SMS

1.14.12. HISTORICOS

Vemos a continuación la función Historico ( ), la única dentro de la categoría Historicos:

1. Historico( )

Descripción: Guarda un determinado dato en un fichero de la tarjeta de memoria MMC (máximo 256MB):

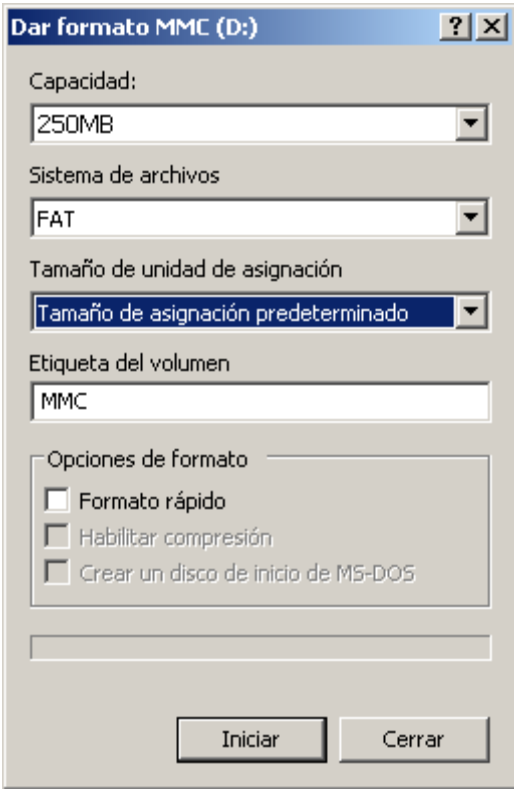

El tiempo mínimo entre capturas es de 1 segundo (1000ms). Los históricos se guardan en archivos tipo .CSV con nombre FX.csv, donde la F será común a todos los ficheros de históricos y la X representa el número de histórico. Si en un mismo fichero se graban varias capturas, automáticamente se guardarán cada una en una linea distinta. Cada una de estas líneas se guarda con formato:

AA/MM/DD hh:mm:ss VALOR

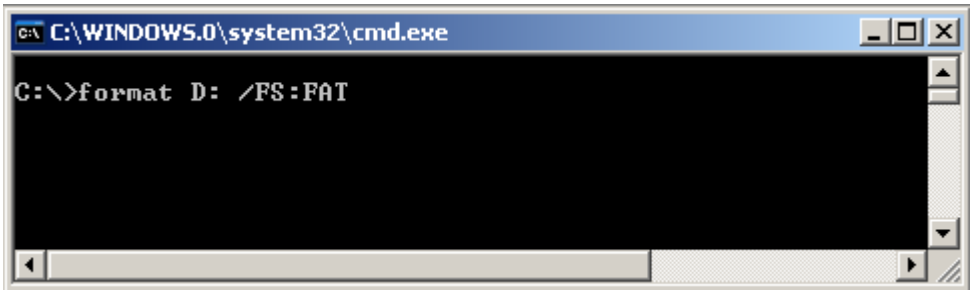
En donde AA son los dos dígitos finales del año, MM el número de mes, DD el número de día, hh la hora de la captura, mm los minutos de la captura, ss los segundos de la captura y VALOR el dato que se ha indicado grabar.

El LED verde es el de la MMC: al arrancar el Regulador, parpadeará repetidamente durante un segundo. Entonces empieza a buscar la tarjeta MMC: se encenderá y apagará durante unos cinco segundos hasta encontrarla. Una vez detectada, el LED quedará encendido fijo y listo para usarse. Para extraer la tarjeta es extremadamente importante que se pulse previamente el botón de extracción (junto al LED verde): en caso contrario, los datos guardados se perderán.

Nota importante: La tarjeta MMC debe ser formateada con el sistema de archivos FAT, y NO como FAT32. Para ello, se puede hacer uso de la utilidad de formateo de Windows especificando sistema FAT




O bien desde la línea de comandos con la instrucción FORMAT *Unidad*: /FS:FAT, donde *Unidad* representa la letra de unidad que corresponda a la tarjeta MMC:




Instrucción: Histórico(P1,P2)

Parámetros: P1: Número de histórico que se desea guardar.

P2: Dato que se desea guardar (valor de variable).

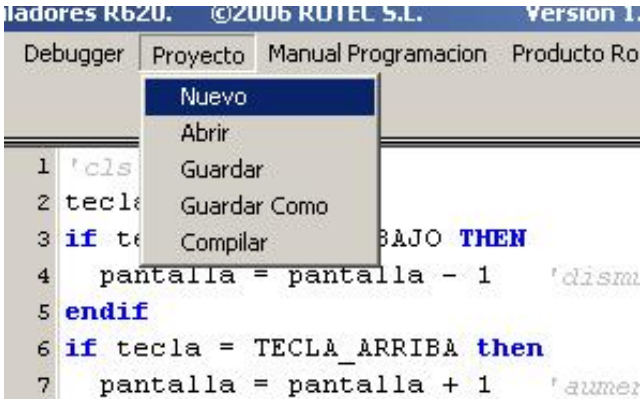
Devuelve: Nada

Ejemplo:

| INSTRUCCIÓN | ACCION |
|---|---|
| Histórico(1,TemperaturaAmbiente)<br>Pause(1000) | Cada segundo se guarda en el fichero F1.csv de la memoria MMC el valor de la variable TemperaturaAmbiente |

## 2. CREAR UN NUEVO PROYECTO

Para crear un nuevo proyecto, hacer clic en el menú *Proyecto --> Nuevo*:



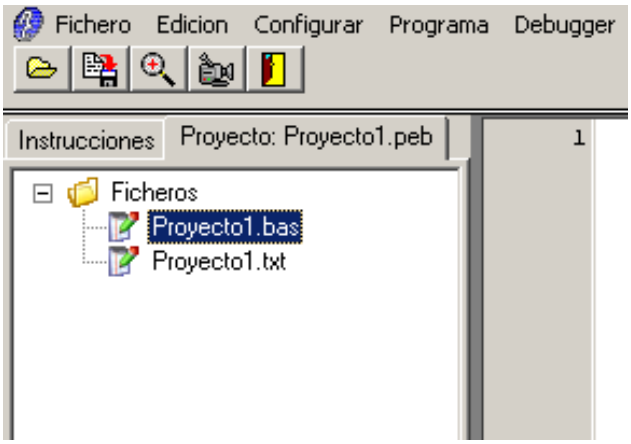

Asignarle un nombre (por ejemplo, *Proyecto1*) y guardarlo. Esto generará tres archivos:

1.- Proyecto1.bas : Archivo donde se guarda el código fuente del proyecto.

2.- Proyecto1.peb : Archivo de sistema (no editable).

3.- Proyecto1.txt : Aquí puede guardar todos los comentarios y notas que desee acerca de este proyecto.

Puede acceder y editar estos archivos en cualquier momento desde la pestaña *Proyecto* en la columna izquierda del compilador:

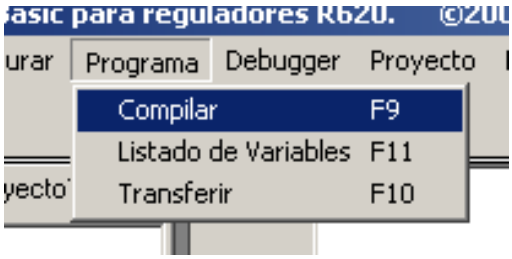



Además, desde el mismo menú *Proyecto* puede Abrir un nuevo proyecto, Guardar el que esté editando o Guardarlo con otro nombre. Se recomienda ir guardando el proyecto cada cierto tiempo.

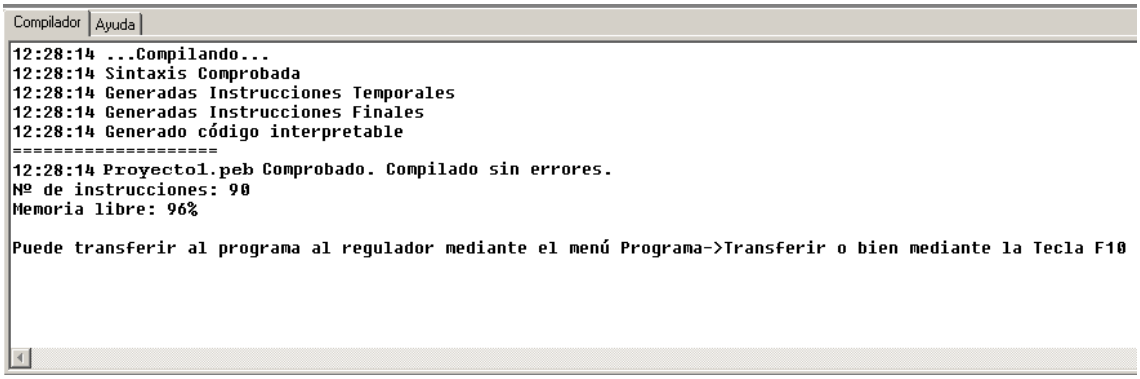
Puede ver un breve video explicativo de cómo crear nuevos proyectos pulsando AQUÍ (Se necesita una conexión a internet).

## 3. COMPILAR EL PROGRAMA

Una vez finalizada la edición del programa es necesario compilar el código fuente (traducirlo a código máquina) para que el regulador pueda interpretarlo. Para ello, guardar el proyecto (menú *Proyecto --> Guardar*) y seleccionar *Compilar* en el menú *Programa* (o bien pulse la tecla F9):




Espere hasta que el compilador acabe el proceso y preste atención a la ventana inferior por si surgieran errores:




En caso de existir errores, hay que tener en cuenta que únicamente aparecerá el primero de ellos: en la ventana del compilador se mostrará información sobre éste, así como el número de línea, tipo de error, etc. Una vez solucionado, aparecerá el siguiente y así hasta el último. Hasta que el programa no esté completamente libre de errores, no se permitirá transferirlo al regulador.

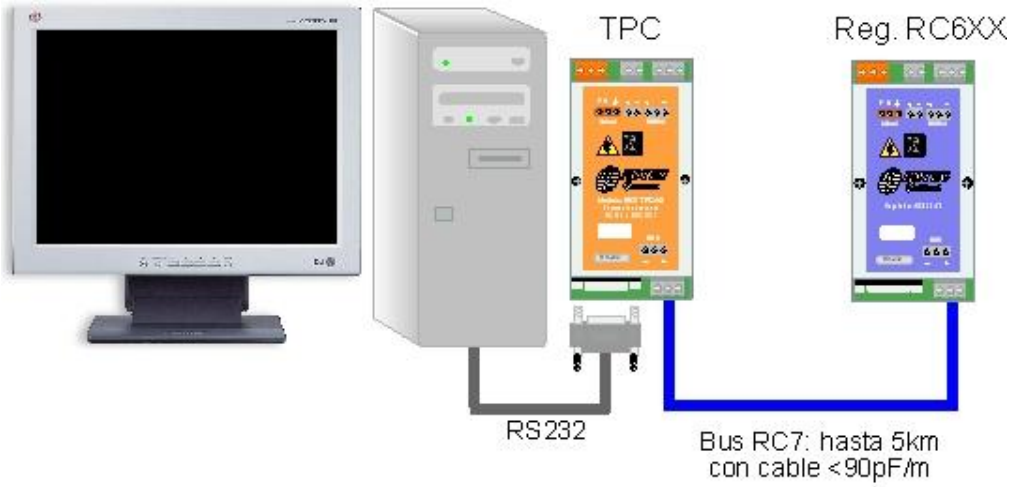
Puede ver un breve video explicativo de cómo compilar programas pulsando AQUI (se necesita una conexión a internet).

Si finalmente aparece el mensaje *Compilado sin errores*, el siguiente paso es transferir el programa al regulador.

## 4. TRANSFERIR EL PROGRAMA AL REGULADOR

Una vez compilado el programa con éxito (ver Compilar el programa) el siguiente paso es transferirlo al regulador. Para ello, existen dos posibilidades:

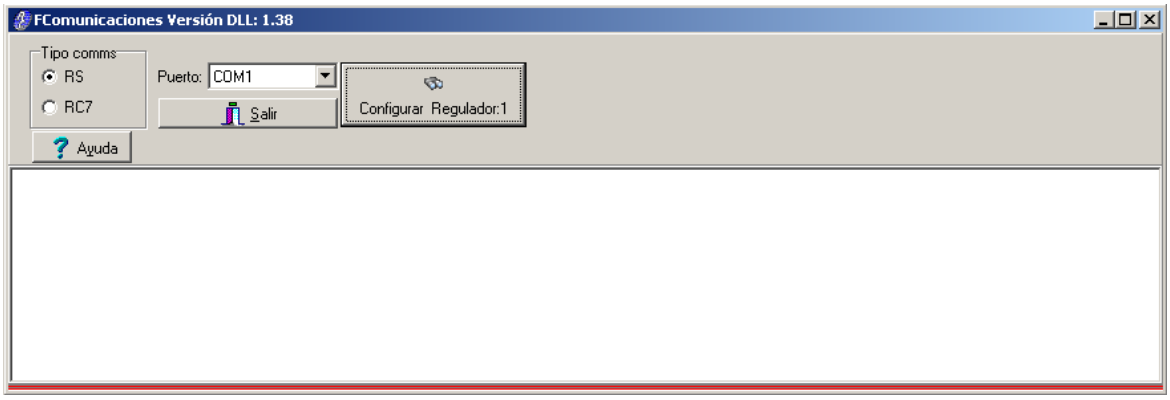
1.- Programarlo mediante el bus RC7 a través de una TPC:



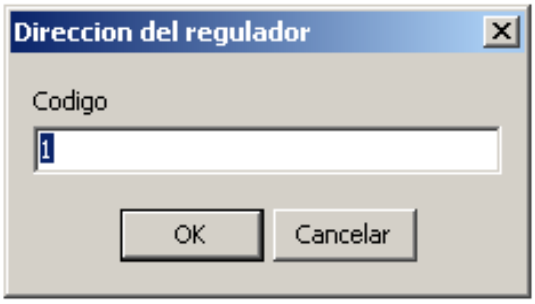

2.- Programarlo mediante bus RS directamente desde el PC (longitud máxima del cable, 2-3 metros):


Desde el EasyBasic una vez compilado el programa, seleccionar *Transferir* del menú *Programa*, o bien pulsar la tecla F10:

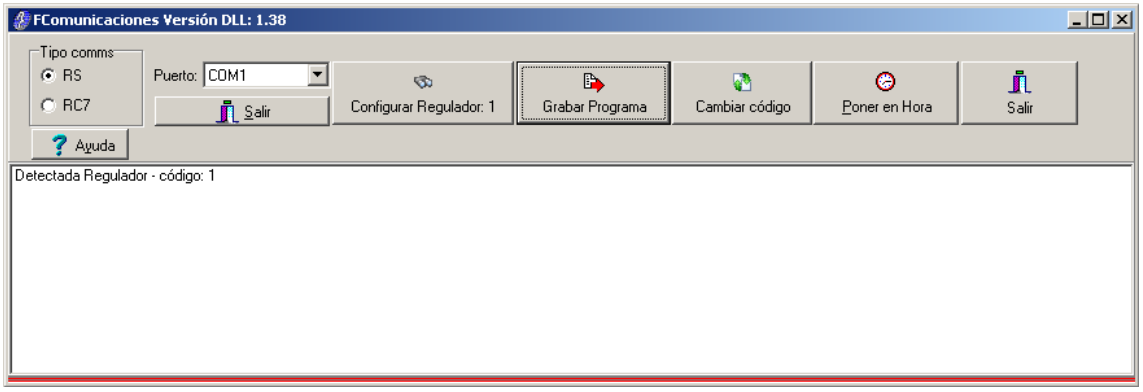
Se abrirá la ventana de comunicaciones, en la que hay que seleccionar el tipo de comunicación con el regulador (RS232 o RC7), el número de puerto COM que se utiliza (habitualmente el 1 o el 2) y pulsar sobre *Configurar Regulador*:




Acto seguido aparecerá la ventana de *Dirección del Regulador*. Dado que en una misma instalación pueden funcionar varios reguladores a la vez, es necesario que cada uno de ellos esté identificado por un código numérico exclusivo, desde 1 hasta 256. Por defecto, en todos los reguladores que salen de fábrica este código es el 1. Se recomienda marcar con tinta indeleble el código de regulador sobre éste.




Al asignarlo, deberá aparecer en la ventana de comunicaciones el mensaje *Detectado regulador - Código:_* y las opciones Grabar Programa, Cambiar Código, Poner en Hora y Salir.




Grabar Programa: Transfiere el programa al regulador con el código asignado previamente.

Cambiar Código: Permite cambiar el código de regulador (de fábrica todos los reguladores tienen código 1).

Poner en hora: Permite poner en hora el regulador (Ver Puesta en Hora).

Salir: Sale de la ventana de comunicaciones.

## 5. PUESTA EN HORA DEL REGULADOR

El Regulador tiene en memoria un reloj-calendario interno del que podemos leer y editar los datos según sea necesario (por ejemplo para configurar un Horario). Hay que tener en cuenta que de fábrica el reloj no viene puesto en hora. Para ponerlo de forma automática en hora (transferirle fecha y hora del PC con el que se esté editando), el proceso a seguir es:

1.- Compilar un programa cualquiera (Tecla F9).

2.- Abrir la ventana de *Comunicaciones con el Regulador* (tecla F10).

3.- Seleccionar tipo de bus adecuado (RS o RC7) .

4.- Seleccionar puerto COM adecuado.

5.- Pulsar sobre el botón *Configurar Regulador* y asegurarse de que se detecta.

6.- Seleccionar el código adecuado.

7.- Pulsar sobre el botón *Poner en Hora*.

Para comprobar que el Regulador se ha puesto correctamente en hora, el procedimiento es el que sigue:

1.- Abrir la ventana del Debugger (Tecla F12).

2.- Pulsar sobre el botón *Animar*.

3.- Abrir la pestaña *Regulador*.

4.- Comprobar que la fecha y hora coinciden con las del PC.

Además, el lenguaje EasyBasic cuenta con un conjunto de instrucciones especialmente diseñadas para tratar con la fecha y hora del Regulador directamente desde el programa.

## 6. ENTRADAS Y SALIDAS

Ver puntos 9 en instrucciones (Manual Rotec Easy Basic 1)

## 7. CONVERSIÓN DE UNIDADES ANALÓGICAS

El producto Rotec Control ofrece diferentes tipos de sondas de lectura del medio físico (temperatura, humedad...), para lo que es suficiente con configurar una terminal TIR con entrada analógica y asignarle un nombre en el programa. Las entradas analógicas de las terminales TIR pueden configurarse como de resistencia, de voltaje o de intensidad mediante unos pequeños puentes situados sobre su propia placa de circuito impreso.

El programa y equipo de control que exploran las terminales realizan la lectura del valor analógico en unas unidades propias denominadas PUNTOS. Este valor en puntos puede convertirse a las unidades necesarias mediante el ajuste de las siguientes escalas o tablas:

| Valor en PUNTOS | Resistencia (Ω). | Intensidad (mA) | Voltaje (V) |
|---|---|---|---|
| 500 | 1644 | 0 | 0 |
| 1500 | 3281 | 20 | 10 |

Además, en el lenguaje EasyBasic existen una serie de funciones diseñadas para facilitar la lectura de estos datos y representarlos de la forma más comprensible:
La función TemperaturaKTY( ) realiza la lectura del valor (en puntos) detectado por una sonda de temperatura (ya sea interior, exterior, de contacto y/o de inmersión) y lo devuelve convertido en grados centígrados.

| | | |
|---|---|---|
| 180 Puntos | = | -28.5ºC |
| 500 Puntos | = | 0,00ºC |
| 1500 Puntos | = | 100,00ºC |
| 1910 Puntos | = | 150,00ºC |

La función Humedad( ) realiza la lectura del valor (en puntos) detectado por una sonda de humedad y lo devuelve convertido en un porcentaje entre 0 y 100%.

| | | |
|---|---|---|
| 500 Puntos | = | 0,00% |
| 1500 Puntos | = | 100,00% |

La función Luminosidad( ) realiza la lectura del valor (en puntos) detectado por una sonda de luminosidad y lo devuelve convertido en un porcentaje entre 0 y 100%.

| | | |
|---|---|---|
| 500 Puntos | = | 0,00% |
| 1500 Puntos | = | 100,00% |

La función Tension( ) realiza la lectura del valor (en puntos) detectado por una sonda y lo devuelve convertido en un valor entre 0V y 10V.

| | | |
|---|---|---|
| 500 Puntos | = | 0,00V |
| 1500 Puntos | = | 10,00V |

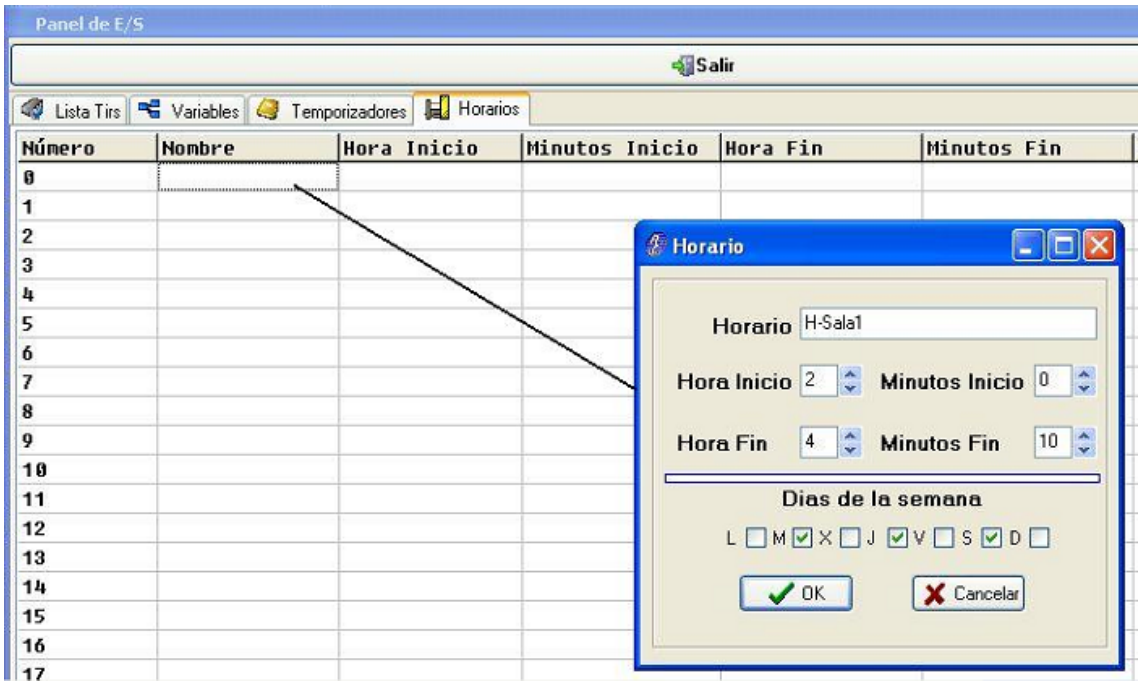
La función Intensidad( ) realiza la lectura del valor (en puntos) detectado por una sonda y lo devuelve convertido en un valor entre 0mA y 20mA.

| | | |
|---|---|---|
| 500 Puntos | = | 0mA |
| 1500 Puntos | = | 20,00mA |

## 8. TABLA DE HORARIOS

Cada Horario (adecuadamente configurado y guardado en la tabla de horarios -Tecla F8-) tiene 5 parámetros: Hora de inicio, Minutos de inicio, Hora de fin, Minutos de fin y Días de la semana activos. Un horario se considera *activo* (dentro de horario) cuando:

(HoraInicio:MinutoInicio) <= (HoraActual del Regulador)
AND
(HoraFin:MinutoFin) >= (HoraActual del Regulador)
y además el día actual del Regulador está en la lista de días activos del horario.
Para programar un determinado horario:
1.- Abrir la Tabla de Horarios del Panel E/S (Tecla F8 --> *Pestaña Horarios*, o bien *Menú Configurar --> Horarios*).
2.- Doble click sobre la casilla de Nombre del Horario número 0 (si es el primero que configuramos en este programa).
3.- En la ventana que se abrirá, determinar un nombre (no puede estar repetido), hora-minutos de inicio, hora-minutos de fin y días activos de la semana.
4.- Pulsar OK.
En la imagen inferior se ha configurado el horario con nombre *H-Sala1*, que empezará a las 2:00 y finalizará a las 4:10 todos los martes jueves y sábados.

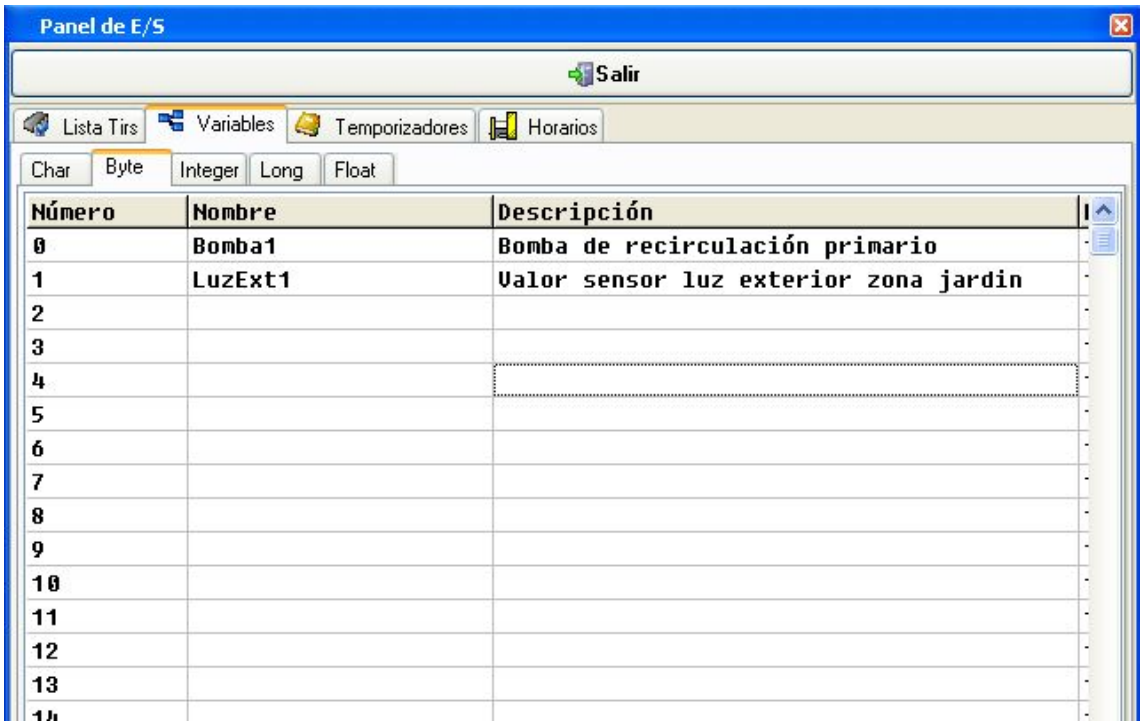



## 9. TABLA DE VARIABLES

Para poder utilizar variables, antes tienen que estar declaradas en la tabla del panel E/S (Tecla F8). Una vez abierto el panel, se selecciona la pestaña del tipo de variable que se va a usar, teniendo en cuenta que:

| | |
|---|---|
| CHAR | Contiene valores entre -128 y +127. |
| BYTE | Contiene valores entre 0 y +255. |
| INTEGER | Contiene valores entre -32767 y +32768. |
| LONG | Contiene valores entre -2147483647 y +2147483648. |
| FLOAT | Contiene valores con decimales. |

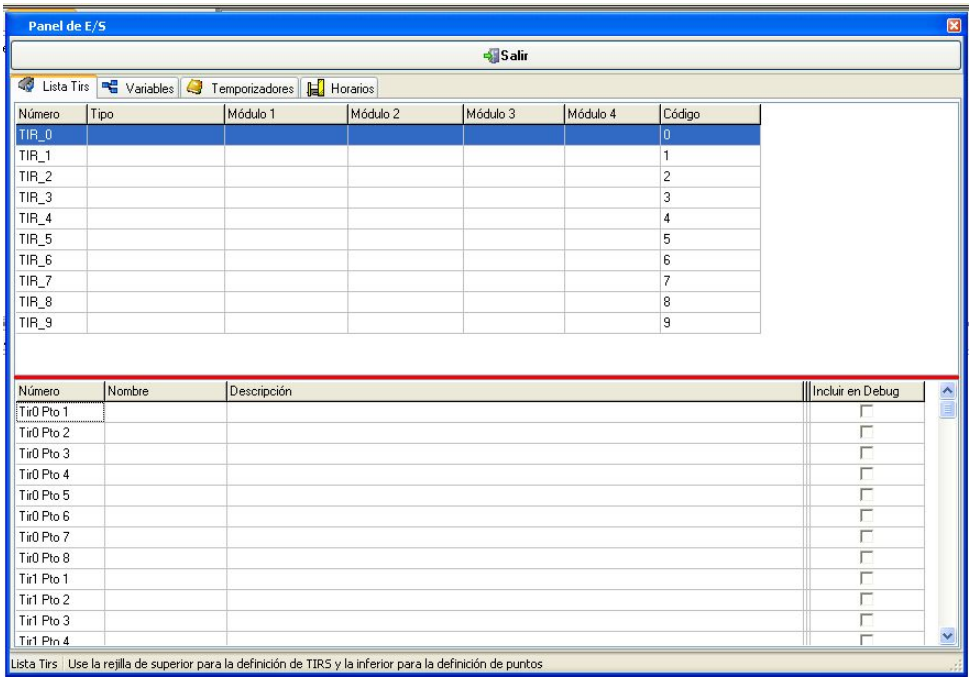
Se van declarando las variables asignándoles el nombre que se usará en el programa (no se aceptan espacios) y se les da una breve descripción aclaratoria:



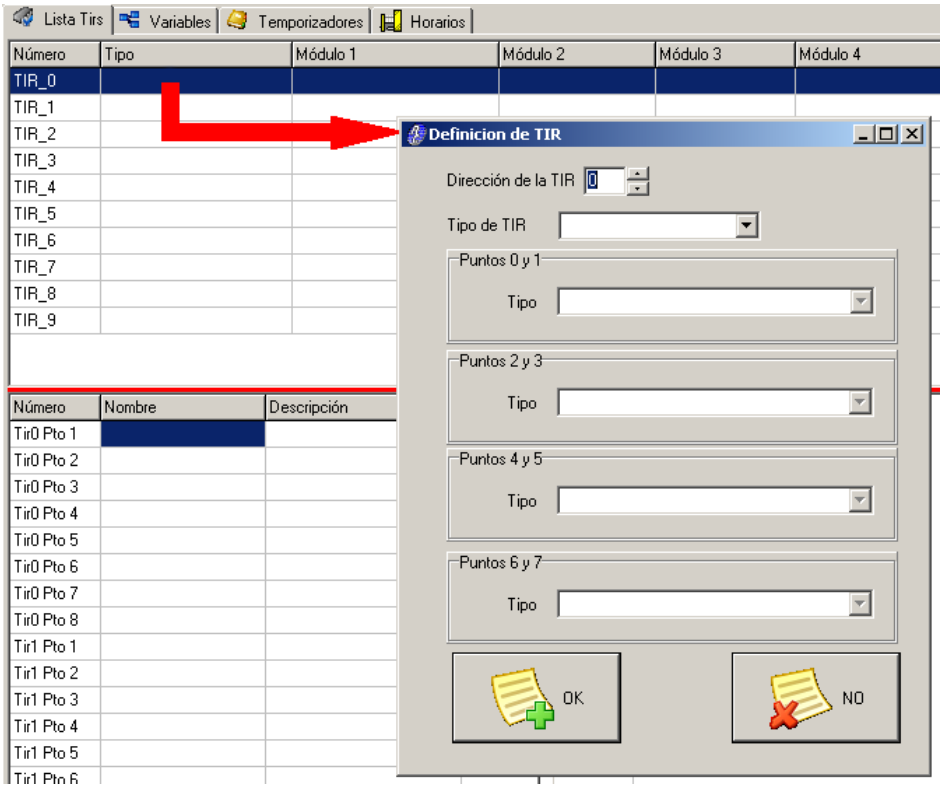

NOTA: se recomienda utilizar siempre el tipo de variable que más se ajuste al uso que se le va a dar, con el fin de ahorra memoria.

## 10. TABLA DE TIRS

Para acceder a la tabla de TIRs del Panel de E/S, pulsar sobre el menú *Configurar* --> *Tirs* o bien pulsar la tecla F8:




Al hacer doble clic en la celda Tipo de una TIR, aparece la ventana de *Definición de TIR*:



En donde:

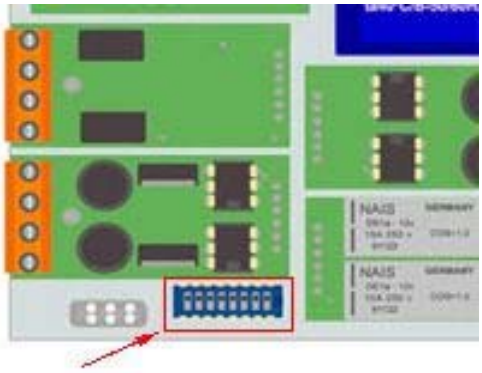
* Cada TIR tiene una dirección única y exclusiva (no puede repetirse), entre 0 y 255 (Ver Configurar TIRs).

* En tipo de TIR, si es modular, se deberá especificar el tipo de cada uno de los módulos que contenga.

Una vez definidas las TIRs, se debe asignar un nombre a cada punto de E/S. Para ello, en la mitad inferior del Panel E/S hacer doble clic en los nombres que se quieran asignar, así como añadir una breve descripción o incluirlos en el depurador. En caso de asignar un nombre que ya esté en uso por otro punto, el propio compilador añadirá un guión bajo al final del nombre de variable, haciéndolas diferentes.

## 10.1. CONFIGURAR TIRS

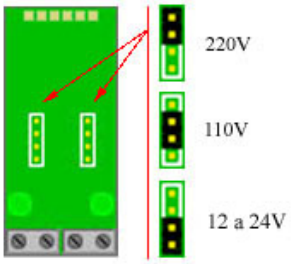
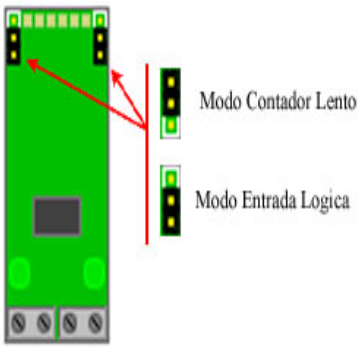
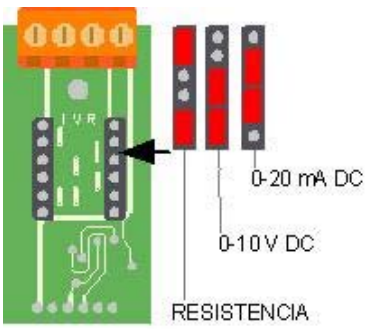
Cada una de las terminales TIR sólo necesita tener un código único asignado, distinto al del resto de terminales de la instalación. Este código se establece mediante el switch DIL de ocho vías situado sobre la propia placa de circuito impreso, en código binario, de forma que los posibles códigos van desde 0 hasta 255:




En las TIR modulares los módulos (y sus puntos de E/S) se numeran en sentido horario:

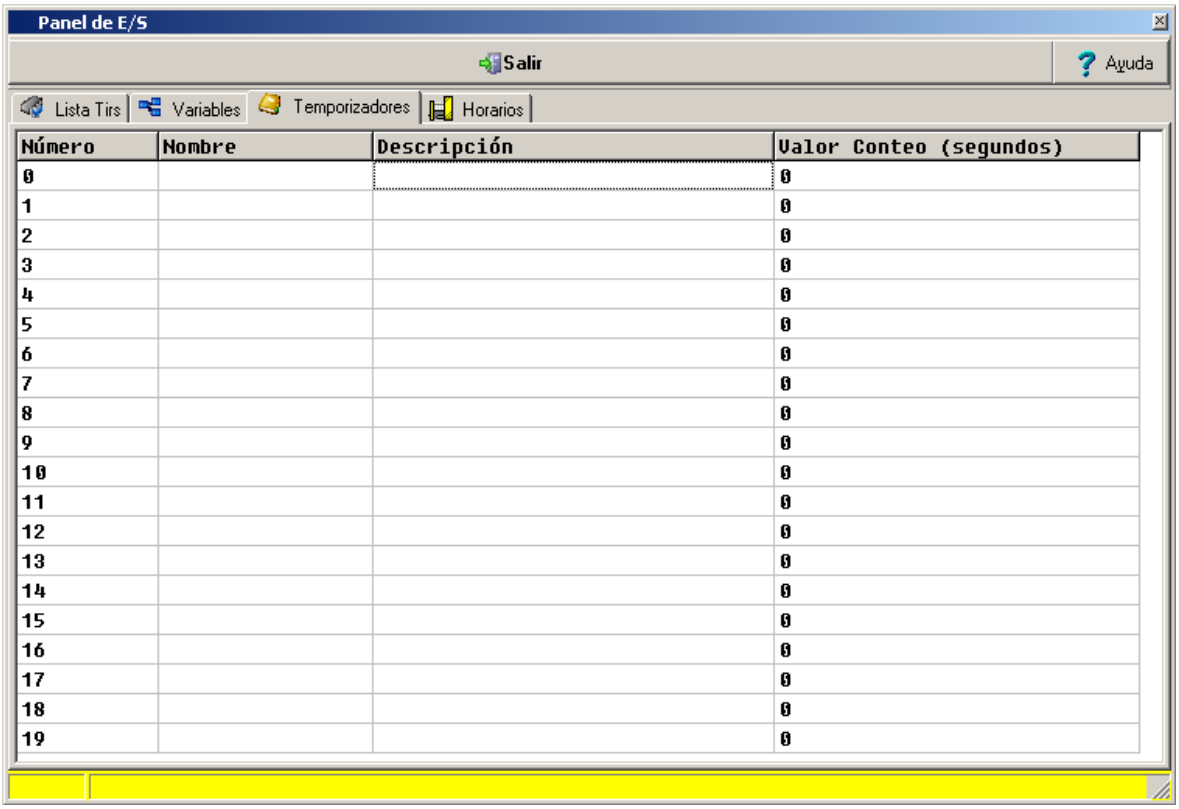

*Orden de los módulos en una TIR modular.*

A su vez, los módulos de entrada/salida lógica y analógica se configuran según la posición de los puentes que se encuentran sobre ellos:

| ENTRADA LÓGICA | ENTRADA LÓGICA-CONTADORES | ENTRADA ANALÓGICA |
|---|---|---|
|   |   |   |

## 11. TABLA DE TEMPORIZADORES

Para acceder a la tabla de temporizadores del Panel de E/S, pulsar sobre el menú *Configurar --> Temporizadores* o bien pulsar la tecla F8 y la pestaña correspondiente:




| Número | Nombre | Descripción | Valor Conteo (segundos) |
|---|---|---|---|
| 0 | | | 0 |
| 1 | | | 0 |
| 2 | | | 0 |
| 3 | | | 0 |
| 4 | | | 0 |
| 5 | | | 0 |
| 6 | | | 0 |
| 7 | | | 0 |
| 8 | | | 0 |
| 9 | | | 0 |
| 10 | | | 0 |
| 11 | | | 0 |
| 12 | | | 0 |
| 13 | | | 0 |
| 14 | | | 0 |
| 15 | | | 0 |
| 16 | | | 0 |
| 17 | | | 0 |
| 18 | | | 0 |
| 19 | | | 0 |

Para crear un nuevo temporizador, hacer doble clic sobre el nombre (no puede estar repetido), así como en la descripción. En la última columna introducir el tiempo que tardará en cumplirse el temporizador, en segundos.

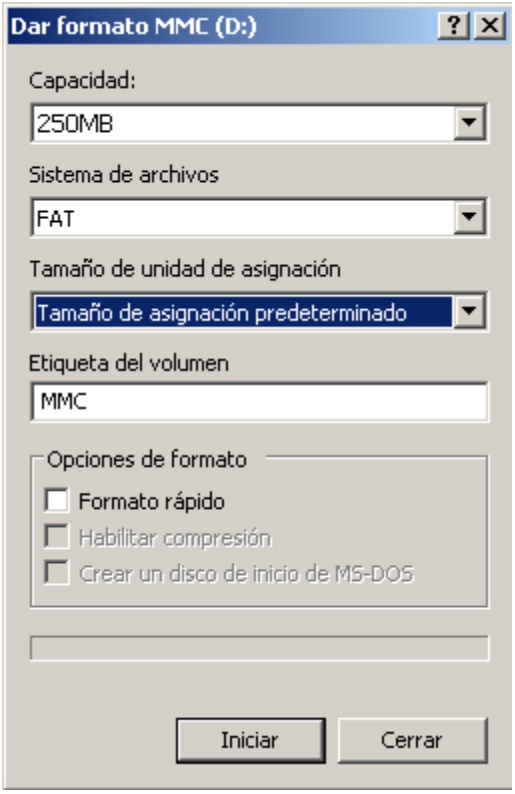
## 12. HISTÓRICOS

El regulador RC-640 de Rotec Control permite almacenar datos (lecturas, estados, etc) en una tarjeta de memoria MMC. Esto es especialmente útil en aquellas instalaciones que no están bajo una supervisión constante, ya que permite guardar un historial del estado del sistema para posteriormente descargarlo a un PC a través de un lector de tarjetas común y analizarlo posteriormente.

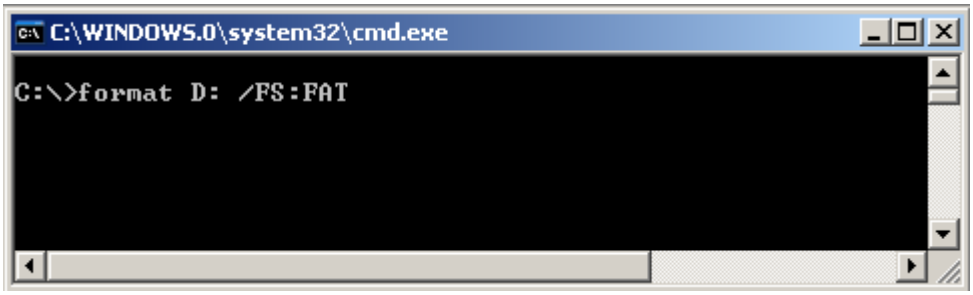
La captura y almacenaje de datos se hace mediante la instrucción Historico() del lenguaje EasyBasic, y hay que tener en cuenta que el tiempo mínimo entre capturas es

de 1 segundo (1000ms).

Nota importante: La tarjeta MMC debe ser formateada con el sistema de archivos FAT, y NO como FAT32. Para ello, se puede hacer uso de la utilidad de formateo de Windows especificando sistema FAT




O bien desde la línea de comandos con la instrucción FORMAT *Unidad*: /FS:FAT, donde *Unidad* representa la letra de unidad que corresponda a la tarjeta MMC:

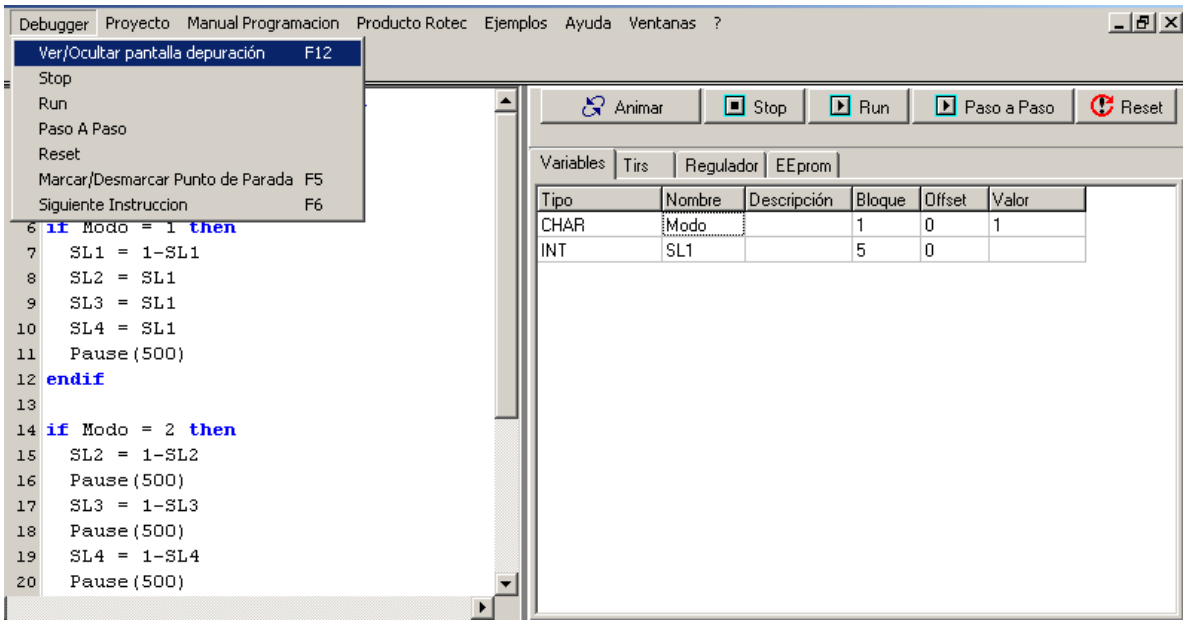


## 13. DEBUGGER

La aplicación EasyBasic de Rotec Control ofrece una herramienta de depuración (*debug*) que permite identificar y corregir errores de programación. En tiempo real es capaz de:

* Mostrar el valor actual, tipo y descripción de las variables que se utilizan en el programa.
* Mostrar el estado y configuración de las terminales TIR que se están utilizando.
* Mostrar el código, versión, fecha y estado del regulador.
* Mostrar la versión y punto de ejecución del intérprete Basic.
* Mostrar los datos de la memoria EEPROM.
* Iniciar, parar o resetear el regulador en cualquier momento.
* Establecer puntos de parada del programa (tecla F5).
* Cambiar los valores de las variables (clic derecho sobre el valor y *Modificar valor*).

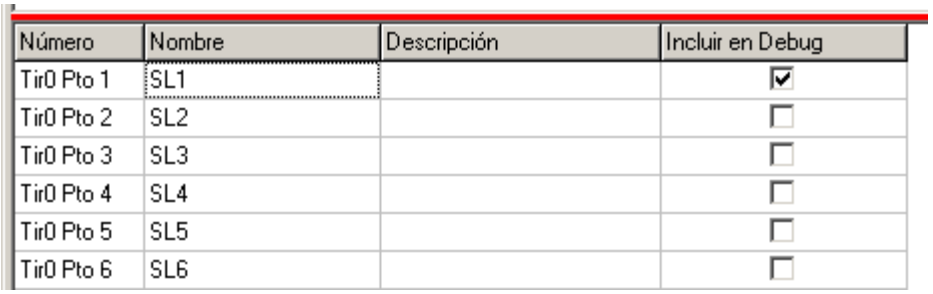
Para acceder a la ventana del depurador, hacer clic en el menú *Debugger* --> *Ver/Ocultar pantalla depurador* o bien pulsar la tecla F12:




Para activar el depurador es necesario que el regulador esté conectado y funcionando:

* Si se trata de depurar un programa recién creado, compilar el programa (tecla F9), transferirlo (tecla F10, transferir programa) y pulsar sobre el botón *Animar*.
* Si se desea depurar un programa sin transferirlo al regulador, basta con compilarlo, pulsar F10, configurar el regulador y salir de la ventana de comunicaciones.

Para añadir variables a la lista de visualización del debugger, acceder al Panel de E/S (tecla F8) y seleccionar la correspondiente casilla *Incluir en Debug*:



| Número | Nombre | Descripción | Incluir en Debug |
|---|---|---|---|
| Tir0 Pto 1 | SL1 | | ☑ |
| Tir0 Pto 2 | SL2 | | ☐ |
| Tir0 Pto 3 | SL3 | | ☐ |
| Tir0 Pto 4 | SL4 | | ☐ |
| Tir0 Pto 5 | SL5 | | ☐ |
| Tir0 Pto 6 | SL6 | | ☐ |

## 14. NORMAS DE EDICIÓN

1: Únicamente se puede poner una instrucción por línea.
2: Las instrucciones AND y OR solamente se pueden usar con IF.
3: En todas las instrucciones o funciones que incluyan paréntesis () es obligatorio ponerlos, incluso si están vacios.
4: Al encender el regulador, todas las TIR (Terminales) están deshabilitadas por defecto - *Ver HabilitaTir()* -.
5: El texto siempre va entre comillas dobles: *"Texto"*.
6: Los comentarios en el código fuente tienen que llevar una comilla simple delante: *'Comentario*.
7: Ninguna instrucción ni función permite espacios al escribirlas: EstaDentroDeHorario() .
8: En un IF puede usarse directamente el estado de una salida lógica: if Salida_Bomba_1=ON then ...
9: Es indistinto el uso de mayúsculas y minúsculas en los nombres de las instrucciones/funciones.
10: Cada vez que el programa arranca todas las variables toman valor cero.
11: La EEPROM tiene una vida limitada de aproximadamente 1.000.000 de escrituras.

12: Es aconsejable elegir bien el tipo de variable (char, integer, float...), especialmente para ahorrar memoria.