

**PLC**

**Descripción**  
**Instrucciones**  
**Funcionamiento del programa**

**Editor / Compilador**

**Descripción**  
**Escribir un programa (nociones básicas)**

**Descripción PLC**

- 8 entradas digitales.
- 6 salidas digitales.
- Teclado display.
- Indicación luminosa para entradas y salidas (en el frente).
- Mensajes de funcionamiento por display.
- Promedio de 120 líneas de programa.
- Comunicación serie.
- Armado de redes Maestro/Esclavo
- Inspección y/o modificación de variables desde teclado.
- 12 temporizadores.
- 4 contadores.
- Alimentación de 18 a 36 Vcc.
- Soporta hasta 20 ms de interrupción de alimentación.
- Retención de programa y variables en E2PROM (no requiere mantenimiento de baterías).
- Protección de operación y de programa con contraseña.
- Supervisión de funcionamiento con WatchDog

\*NOTA: el modelo básico tiene solamente 2 entradas y 2 salidas.



- Común sal
- Sal 6
- Sal 5
- Sal 4
- Sal 3
- Ent 8
- Ent 7
- Ent 6
- Ent 5
- Ent 4
- Ent 3
- Alim + •
- Alim •
- Común sal •
- Sal 1 •
- Sal 2 •
- Ent 1 •
- Ent 2 •
- Común ent •
- 0 •
- Rx •
- Tx •

Aspecto del CP21 y descripción de los bornes traseros.

**Entradas digitales**

Entradas de tensión para niveles lógicos 1 (activado) o 0 (desactivado).  
Todas las entradas están referidas a un solo común y son bidireccionales.

<u>Nivel lógico</u>	<u>Rango de tensión</u>
0	-2 a +2v
1	-27 a -10 y +10 a +27v

En los rangos de -9 a -2v y +2v a +9v no puede garantizarse el estado lógico adoptado (indeterminado).

**Salidas digitales**

Salidas a relé distribuidas en 2 grupos con un solo común para cada uno: salidas S1 y S2 comparten un común y S3, S4, S5 y S6 comparten otro.

<u>Tensión máx.</u>	<u>Corriente máx.</u>	<u>Tipo carga</u>
250Vca	1A	resistiva
250Vca	0.2A (1)	inductiva
30Vcc	1A	resistiva
30Vcc	0.5A (2)	inductiva

- (1) Requiere supresor para corriente alterna (RC)
- (2) Requiere supresor para corriente continua (diodo rápido)

**Capacidad de memoria**

**Programa**

La mayoría de las instrucciones tienen una longitud de 2 bytes (instrucción + dato). Le siguen las de 3, luego las de 1 y algunas con más de 3 bytes. Por esta razón se estima que los 256 bytes destinados al programa serán ocupados con alrededor de unas 120 líneas de instrucciones.

**Usuario**

Todas las variables para el usuario tienen asignado un nombre para facilitar su uso. consta de las 8 entradas, 6 salidas, 46 internas, 16 internas con retención y las de solo lectura: contadores y temporizadores.

**Mapa de memoria usuario**

**Entradas**

e1	equ	0
e2	equ	1
e3	equ	2
e4	equ	3
e5	equ	4
e6	equ	5
e7	equ	6
e8	equ	7

**Salidas**

s1	equ	8
s2	equ	9
s3	equ	10
s4	equ	11
s5	equ	12
s6	equ	13

## MicroPLC modelo CP21

Bellplast S.R.L.

www.caipe.com

### Variables

V01	equ	14
v02	equ	15
v03	equ	16
v04	equ	17
v05	equ	18
v06	equ	19
v07	equ	20
v08	equ	21
v09	equ	22
v10	equ	23
v11	equ	24
v12	equ	25
v13	equ	26
v14	equ	27
v15	equ	28
v16	equ	29
v17	equ	30
v18	equ	31
v19	equ	32
v20	equ	33
v21	equ	34
v22	equ	35
v23	equ	36
v24	equ	37
v25	equ	38
v26	equ	39
v27	equ	40
v28	equ	41
v29	equ	42
v30	equ	43
v31	equ	44
v32	equ	45
v33	equ	46
v34	equ	47
v35	equ	48
v36	equ	49
v37	equ	50
v38	equ	51
v39	equ	52
v40	equ	53
v41	equ	54
v42	equ	55
v43	equ	56
v45	equ	58
v46	equ	59

### Variables con retención

re01	equ	60
re02	equ	61
re03	equ	62
re04	equ	63
re05	equ	64
re06	equ	65
re07	equ	66
re08	equ	67
re09	equ	68
re10	equ	69
re11	equ	70
re12	equ	71
re13	equ	72
re14	equ	73
re15	equ	74
re16	equ	75

### Salida de contadores

Cnt1	equ	76
Cnt2	equ	77
Cnt3	equ	78
Cnt4	equ	79

### Salida de temporizadores variables

Tpv1	equ	80
Tpv2	equ	81
Tpv3	equ	82
Tpv4	equ	83

### Rápidos

tpf1	equ	84
tpf2	equ	85

### Salida de temporizadores fijos

tp1	equ	86
tp2	equ	87
tp3	equ	88
tp4	equ	89

### Rápidos

tpf1	equ	90
tpf2	equ	91

## Teclado

### Descripción

Permite inspeccionar y modificar las variables con retención, los valores de cuenta de los temporizadores y contadores. Consta de 4 teclas de goma y su funcionalidad es fija (no puede cambiarse por programa).

### Inspección/modificación de variables

Presionando simultáneamente FUN y ◀ se entra en el modo de inspección/modificación de variables. Si el programa incluye contraseña, esta será necesaria para inspeccionar y modificar las variables. El funcionamiento no es interrumpido durante la inspección/modificación. Las 16 variables con retención, el preseteo de los contadores, temporizadores variables y modo de presentación del contador 1 son accesibles en este modo.

Presionando la tecla FUN por mas de 3 segundos se entra en el modo de inspección/modificación restringido donde solo algunas variables son accesibles y no se requiere contraseña. Las variables con retención 15 y 16, el preseteo del contador 1, temporizador 1 y temporizador rápido 1 son accesibles en este modo.

Con las teclas ▲ y ▼ se recorre el menú donde se observan los nombres de las variables. Al presionar FUN se ingresa a inspeccionar o modificar la variable selecta. Presionando nuevamente FUN se acepta el valor y vuelve al menú. Para salir del menú se deben presionar FUN + ▲.

Un timer de teclado abortará el modo de inspección/modificación de variables si pasan mas de 30 segundos sin presionar una tecla. Los cambios realizados tendrán efecto, pero no se guardarán en la memoria por lo que no estarán presentes cuando se reinicie el equipo.

### Abortar programa

La ejecución del programa puede abortarse presionando simultáneamente las teclas ▲ y ▼ cuando se enciende el equipo (ver watchdog).

### Versión de ROM

Durante el funcionamiento normal puede conocerse la versión de sistema operativo presionando la tecla ◀.

## Display

Mensajes de funcionamiento del PLC e inspección y modificación de variables durante la programación. Consta de 4 dígitos rojos de alto brillo. Su funcionalidad es fija y no puede cambiarse por programa.

Run	El PLC está corriendo el programa
No p	El PLC no tiene programa cargado o no es válido
Stop	El PLC está detenido (no ejecuta programa)
Prog	El PLC está recibiendo un programa
Save	El PLC está grabando el programa en la memoria no volátil
W.dog	Watch Dog
Send	El PLC está enviando su programa

*RUN*: puede no verse este mensaje, sino el estado de cuenta del contador 1, según como se lo halla programado desde el teclado display.

*Prog*: cuando entra en este modo la ejecución del programa se detiene y todas las variables (incluidas las salidas) retienen su estado. Para que el programa continúe o reinicie (modo run) es necesario enviarle una orden al PLC.

*Watch Dog*: el estado de watch dog(wdog) es un estado de alarma a causa de una condición no tolerable como la ejecución errática de un programa, corrupción de datos o fallas eléctricas. Para salir de esta condición es necesario apagar el equipo. Puede darse el caso de un programa dañado impidiendo que el PLC entre en modo run. Para restablecer el PLC debe abortarse la ejecución del programa presionando las teclas ▲ y ▼ cuando se enciende el equipo (pasa a modo no p).

*Send*: el PLC devuelve el programa sin interrumpir la ejecución del mismo.

### Contraseña

La contraseña tiene las siguientes virtudes:

- Operación restringida de teclado: requiere contraseña para acceder.
- Protección de programa: el programa no puede ser extraído del PLC si no coincide la contraseña.

La contraseña tiene 65535 combinaciones posibles, y puede representarse con los números del 0 al 9 y las letras A, B, C, D, E y F (0001 a FFFF). Esta codificación se conoce como hexadecimal. Cuando el valor es cero (0000) equivale a contraseña desactivada permitiendo el libre acceso al PLC.

La contraseña es puesta en el PLC solamente en el momento de enviarle el programa. No es posible activarla desde el teclado.

### Temporizadores

El temporizador tiene una sola entrada (activado/reset) y salida. Hay 4 variantes: TP es temporizador fijo por programa, TPF es temporizador fijo por programa rápido, TPV es temporizador variable y TPVF es temporizador variable rápido. El estado de tiempo no es accesible desde el programa. Otros detalles en las instrucciones para temporizador TP, TPF, TPV y TPVF.

La entrada del temporizador es el valor del acumulador en el momento de ejecutar la instrucción de temporizador. La salida es una variable que toma el valor 1 cuando se cumple el tiempo.

Solo y por única vez el acumulador tendrá el valor 1 después de ejecutar la instrucción temporizador en el momento en que halla llegado al tiempo.

Cable PLC - PC

### Cable y puerto de comunicación

El cable de programación consiste de 2 conectores y 3 conductores:



El orden es : 2 -- Tx, 3 -- Rx y 5 -- 0. El conector es del tipo D hembra, el otro puede ser terminales alineados o un borne triple enchufable. Si la PC solo dispone de conector D de 25 contactos será necesario adquirir un adaptador.

Conector tipo D hembra de 9 contactos visto de atrás.

Pin	Nombre	Descripción
1	CD	Carrier Detect
2	RXD	Receive Data
3	TXD	Transmit Data
4	DTR	Data Terminal Ready
5	GND	System Ground
6	DSR	Data Set Ready
7	RTS	Request to Send
8	CTS	Clear to Send
9	RI	Ring Indicator

## Instrucciones

Todas las instrucciones son lógicas y pueden ser representadas en un diagrama de contactos.

<u>Instrucción</u>	<u>Función</u>
AND <i>var</i>	<i>var</i> & ACC -> ACC
ANDN <i>var</i>	not( <i>var</i> ) & ACC -> ACC
ANDP	ACC1 & ACC -> ACC máx. 3 veces consecutivas
CLRIF1 <i>var 1 var 2</i>	CLRIF1 <i>var1 var2</i>
CR	control reset 0xff -> CS
CS	control set ACC -> CS solo 1 vez
CTV <i>Nº</i>	Contador variable de 1 al 4
END	retorno al principio - salida del programa
EQU	EQUivalencia (igualar)
GOSUB <i>label</i>	Salto a subrutina (llamada)
GOTO <i>label</i>	Salto incondicional
IF0 <i>label</i>	si ACC = 0 saltar a <i>label</i>
IF1 <i>label</i>	si ACC = 1 saltar a <i>label</i>
IFV1 <i>label</i>	si <i>var</i> = 1 saltar a <i>label</i>
JP0 <i>label</i>	si ACC = 0 gosub <i>label</i>
JP1 <i>label</i>	si ACC = 1 gosub <i>label</i>
JPV1 <i>label</i>	si <i>var</i> = 1 gosub <i>label</i>
LD <i>var</i>	<i>var</i> -> ACC
LDI <i>cte</i>	<i>cte</i> -> ACC
LDN <i>var</i>	NOT( <i>var</i> ) -> ACC

NOT <i>var</i>	NOT( <i>var</i> ) -> <i>var</i>
NOTA	NOT(ACC) -> ACC
OR <i>var</i>	<i>var</i> or ACC -> ACC
ORN <i>var</i>	NOT( <i>var</i> ) or ACC -> ACC
ORP	ACC1 & ACC -> ACC máx. 3 veces consecutivas
OSC <i>ton toff</i>	Oscilador, tiempo ON y tiempo OFF
OUT <i>var</i>	ACC -> <i>var</i>
RJP	Retorno desde una llamada
RST <i>rep var</i>	0x00 -> <i>var rep</i> variables consecutivas
RCT	Reset de todos los contadores si ACC = 1
RTP	Reset de todos los temporizadores si ACC = 1
SET <i>var</i>	0xff -> <i>var</i>
SETB <i>var</i>	0xff -> <i>var</i> ACC no cambia
SETIF1 <i>var 1 var 2</i>	set <i>var1</i> si <i>var2</i> = 1
TP <i>Nº temp tiempo</i>	Nombre temporizador y tiempo en segundos
TPF <i>Nº temp tiempo</i>	Nombre temporizador y tiempo en segundos
TPV <i>Nº temp</i>	Nombre temporizador
TPVF <i>Nº temp</i>	Nombre temporizador
XOR <i>var</i>	Operación lógica XOR con la variable <i>var</i>

### AND (Función lógica Y)

ACC AND *var* -> ACC

Modo de uso: AND *var*  
Bytes usados: 2

Realiza la operación lógica Y entre la variable *var* y el acumulador. El resultado queda en el acumulador. Ej.:

```
LD    v02
AND   v01
```

El resultado será 1 si el estado del acumulador (ACC) y el de la variable (*var*) es 1. De lo contrario siempre dará 0.

### ANDN (Y negado)

ACC AND NOT(*var*) -> ACC

Modo de uso: ANDN *var*  
Bytes usados: 2

Realiza la operación lógica Y entre la variable *var* negada y el acumulador. El resultado queda en el acumulador. Ej.:

```
LD    v01
ANDN  v02
```

El resultado será 0 si el estado del acumulador (ACC) y el de la variable (*var*) es 1. De lo contrario siempre dará 1.

### ANDP (Y previo)

ACC AND ACC1 -> ACC

Modo de uso: ANDP  
Bytes usados: 1

Realiza la operación lógica Y entre el valor anterior del acumulador y el acumulador actual. El resultado queda en el acumulador.

Cada vez que se hace una ANDP se produce una rotación inversa de acumuladores. Hay un acumulador principal y 3 secundarios. La rotación se puede representar de la siguiente forma: *resultado* -> ACC  
ACC1 <- ACC2 <- ACC3.

### CLRIF1(Reset si es 1)

si var 1 = 1 0 -> var 2

Modo de uso: CLRIF1 var 1 var 2  
Bytes usados: 3

Carga la variable *var1* con el valor 0 si el bit 0 de la variable *var2* está en 1. La variable *var1* puede ser cualquiera entre e0 y re16. La variable *var2* puede ser cualquiera entre e0 y tpf2. El estado del acumulador no es alterado.

### CR CS (Control Reset y Control Set)

Modo de uso: CS  
Bytes usados: 1  
Modo de uso: CR  
Bytes usados: 1

Set y reset del control maestro. El control maestro realiza una operación lógica Y en las instrucciones LD, LDN, OR Y ORN cuando estas son ejecutadas. CS copia el valor actual del acumulador al control maestro. CR carga al control maestro con un valor neutral.

LD E3  
OR E1  
CS  
LD E2  
OUT S1  
OUT S2  
CR

### CTV (Contador variable)

CTV N°\_temp

Modo de uso: CTV contador  
Bytes usados: 2

Contador variable interno hasta 9999 (luego vuelve a 0) con un corte programable por teclado. Hay 4 contadores, siendo el 1 el único que puede ser presentado por display o retener su valor al apagar el equipo.

LD	E1	;entrada_cuenta
LD	E2	;reset_cuenta
CTV	Ctv2	

En el ejemplo se ve el método básico para usar un contador. La entrada de contaje es el acumulador anterior (AC1) y la entrada de puesta a cero es el acumulador (ACC). Si ACC (entrada de puesta a cero) está en 1 el contador es reseteado (su valor de cuenta es puesto en cero). Si AC1 (entrada de contaje) pasa de 0 a 1 (entre la ejecución actual y la anterior) el valor de cuenta es incrementado en 1.

El uso de la instrucción es: CTV *Nº\_contador* (1 al 4). Para conocer el estado de su salida se invoca de forma similar: CTVNº. Ej.: LD CTV1 (cargar el acumulador con el valor de la salida del contador variable 1).

El valor de corte programable sirve para indicarle al contador en que valor de contaje la salida del mismo debe adoptar el valor 1. Cuando la cuenta llega a este valor y la salida pasa a valor 1, el acumulador queda por única vez con valor 1, sino siempre sale en 0.

El contador tiene entrada de contaje y reset y una salida. Los contadores son 4 y no se puede acceder a su estado de cuenta desde el programa.

Las entradas de contaje y reset son el valor del acumulador previo y el acumulador respectivamente al momento de ejecutar la instrucción contador.

El contador cuenta las transiciones de 0 a 1 de la entrada de contaje (de 1 a 0 es ignorado). Un valor 1 en la entrada de reset pondrá incondicionalmente el estado de cuenta y la salida en cero.

La salida es una variable que se pone en 1 cuando el contador llega al valor preestablecido (desde el teclado). Esta permanecerá con ese valor hasta que el contador sea restablecido a cero. Solo y por única vez cuando el contador llegue al valor preestablecido el acumulador adoptará el valor 1 después de ejecutarse la instrucción contador.

## **END** (Fin vuelta de programa)

Modo de uso:	END
Bytes usados:	1

Terminación del scan: vuelve a ejecutar desde la 1ª instrucción del programa. El contador de anidación de llamadas a subrutinas es restablecido a 0, por lo que no quedan pendientes retornos de subrutinas.

## **GOSUB** (ir a subrutina)

Modo de uso:	GOSUB <i>label</i>
Bytes usados:	2

Llamada incondicional a una subrutina. La subrutina comienza en la etiqueta indicada en la instrucción (*label*). Ej.:

*GOSUB proc\_3*

*proc\_3*      *AND ...*

No pueden anidarse mas de 5 llamadas, es decir, llamar de una subrutina a otra mas de 5 veces. Ver Saltos y subrutinas.

### **GOTO** (ir a)

Modo de uso: GOTO *label*  
Bytes usados: 2

Salto incondicional a ejecutar las instrucciones a partir de la línea indicada con la etiqueta *label*. Ej.:

*GOTO alarma*

*alarma: OUT ....*

Ver Saltos y subrutinas.

### **IFO** (Si 0) *si ACC = 0 saltar*

Modo de uso: IFO*label*  
Bytes usados: 2

Salto condicional: si el bit 0 del acumulador tiene valor falso (0), la próxima instrucción que se ejecutará es la apuntada por la etiqueta *label*. Ej.:

*Proc\_2: IFO proc\_2  
LD ...*

Si la condición es válida se ejecutarán las instrucciones a partir de la línea llamada *proc\_2*. Ver Saltos y subrutinas.

### **IF1** (Si 1) *si ACC = 1 saltar*

Modo de uso: IF1*label*  
Bytes usados: 2

Salto condicional: si el bit 0 del acumulador tiene valor verdadero (1), la próxima instrucción que se ejecutará es la apuntada por la etiqueta *label*. Ej.:

*IF1 proc\_2*

*Proc\_2: LD ...*

Si la condición es válida se ejecutarán las instrucciones a partir de la línea llamada *proc\_2*. Ver Saltos y subrutinas.

### **IFV1** (Si var = 1 saltar) *si var = 1 saltar*

Modo de uso: IFV1*var label*

Bytes usados: 3

Salto condicional: si el bit 0 de la variable *var* tiene valor verdadero (1), la próxima instrucción que se ejecutará es la apuntada por la etiqueta *label*. Ej.:

*IFV1 E1 etq02*

*etq02: LD ...*

Si la condición es válida se ejecutarán las instrucciones a partir de la línea llamada *etq2*. Ver Saltos y subrutinas.

### **JP0 (Saltar si 0)**

*si ACC = 0 saltar a subrutina*

Modo de uso: *JP0 label*

Bytes usados: 2

Llamada a una subrutina si el bit 0 del acumulador es falso (valor = 0). La subrutina comienza en la etiqueta indicada en la instrucción (*label*). Ej.:

*JP0 proc\_3*

*proc\_3 AND ...*

No pueden anidarse mas de 5 llamadas, es decir, llamar de una subrutina a otra mas de 5 veces. Ver Saltos y subrutinas

### **JP1 (Saltar si 1)**

*si ACC = 1 saltar a subrutina*

Modo de uso: *JP1 label*

Bytes usados: 2

Llamada a una subrutina si el bit 0 del acumulador es verdadero (valor = 1). La subrutina comienza en la etiqueta indicada en la instrucción (*label*). Ej.:

*JP1 proc\_3*

*proc\_3 AND ...*

No pueden anidarse mas de 5 llamadas, es decir, llamar de una subrutina a otra mas de 5 veces. Ver Saltos y subrutinas

### **JPV1 (Si var = 1 saltar)**

*si var = 1 saltar a subrutina*

Modo de uso: *JPV1var label*

Bytes usados: 3

Llamada a una subrutina si el bit 0 de la variable *var* tiene valor verdadero (1), la próxima instrucción

que se ejecutará es la apuntada por la etiqueta *label*. Ej.:

*JPV1 E1 etq02*

*etq02: LD ...*

Si la condición es válida se ejecutarán las instrucciones a partir de la línea llamada *etq2*. Ver Saltos y subrutinas.

### **LD** (Cargar acumulador ACC)

*var -> ACC*

Modo de uso: LD *var*  
Bytes usados: 2

Carga el acumulador con el contenido de la variable *var* señalada en la instrucción. Ej.: si la variable V02 tiene el valor 1, al ejecutar LD V02 el acumulador también tendrá el valor 1.

Cada vez que se hace un LD se produce una rotación de acumuladores. Hay un acumulador principal y 3 secundarios. La rotación se puede representar de la siguiente forma: *valor -> ACC -> AC1 -> AC2 -> AC3*.

### **LDI** (Cargar ACC con constante)

*cte -> ACC*

Modo de uso: LDI *cte*  
Bytes usados: 2

Carga el acumulador con el valor instantáneo *cte*. Ej. : si la instrucción a ejecutar es LDI 1, el acumulador también tendrá el valor 1.

Cada vez que se hace un LDI se produce una rotación de acumuladores. Hay un acumulador principal y 3 secundarios. La rotación se puede representar de la siguiente forma: *valor -> ACC -> AC1 -> AC2 -> AC3*.

### **LDN** (Cargar negado ACC)

*NOT(var) -> ACC*

Modo de uso: LDN *var*  
Bytes usados: 2

Carga el acumulador con el contenido negado de la variable señalada en la instrucción. Ej. : si la variable V02 tiene el valor 1, al ejecutar LD V02 el acumulador también tendrá el valor 0.

Cada vez que se hace un LD se produce una rotación de acumuladores. Hay un acumulador principal y 3 secundarios. La rotación se puede representar de la siguiente forma: *valor -> ACC -> AC1 -> AC2 -> AC3*.

### **NOT** (Negar var)

*NOT(var) -> var, también usado para indicar un valor invertido*

Modo de uso: not *var*  
Bytes usados:2

Niega el valor de la variable: invierte el estado de los bits (los 1s en 0s y viceversa).

NOT e1

### NOTA (Niega acumulador)

NOT (ACC) -> ACC

Modo de uso: NOTA  
Bytes usados: 2

Niega el estado del acumulador, cambiando el valor de todos los bits de 1s a 0s y viceversa.

### OR (Función lógica O)

ACC or var -> ACC

Modo de uso: OR var  
Bytes usados: 2

Realiza la operación lógica O entre la variable *var* y el acumulador. El resultado queda en el acumulador. Ej. :

LD E2  
OR v01

### ORN (O negado)

ACC or var -> ACC

Modo de uso: ORN var  
Bytes usados: 2

Realiza la operación lógica O negada entre la variable *var* y el acumulador. El resultado queda en el acumulador. Ej. :

LD E3  
ORN V02

### ORP (O previo)

ACC or ACC1 -> ACC

Modo de uso: ORP  
Bytes usados: 1

Realiza la operación lógica O entre el valor anterior del acumulador y el acumulador actual. El resultado queda en el acumulador.

Cada vez que se hace una ORP se produce una rotación inversa de acumuladores. Hay un acumulador principal y 3 secundarios. La rotación se puede representar de la siguiente forma: *resultado -> ACC AC1 <- AC2 <- AC3*.

### OSC (Oscilador)

OSC tiempo\_on (1) tiempo\_off (0) -> ACC

Modo de uso: OSC tiempo\_on tiempo\_off  
Bytes usados: 3

Carga el acumulador con 1 y 0 (después de ejecutar la instrucción) alternativamente durante los tiempos activado y desactivado respectivamente. Necesita que el acumulador esté en 1 al ejecutar la instrucción para activarlo. Los tiempos máximos son de 25 segundos cada uno.

LD E1 ;activa oscilador (si está en 1)  
OSC 1 1 ;1 segundo activado y 1 desactivado  
OUT S2 ;sale por salida 2

En el ejemplo el acumulador tendrá el valor 1 durante 1 segundo y 0 durante 1 segundo. Este estado es copiado a la variable S2 (que coincide con la salida 2). Cuando se ejecute la instrucción estando el acumulador con valor 0, el oscilador será reseteado y por lo tanto su salida (el acumulador) será 0.

**OUT (Copia acumulador)**

ACC -> var

Modo de uso: OUT var  
Bytes usados: 2

Copia el contenido del acumulador a una variable var. Ej. : OUT S2 hará que el acumulador y la variable S2 (que coincide con la salida 2) tengan el mismo valor.

**RJP (Retorno de subrutina)**

Modo de uso: RJP  
Bytes usados: 1

Retorno de subrutina: retorna a ejecutar las instrucciones que siguen después de la instrucción que llamó a la subrutina. Ver Saltos y subrutinas.

**RST (Volver a 0)**

si ACC = 1 0 -> cte variables a partir de var

Modo de uso: RST cte var  
Bytes usados: 3

La instrucción RST (RST var) responde a la siguiente tabla y el valor del acumulador es al momento de ejecutar la instrucción:

Acc	var		var	Acc	
antes			desp.		
0	0		0	0	la variable y el acumulador no cambian
0	1	RST	1	0	la variable y el acumulador no cambian
1	0		0	1	la variable ya estaba reseteada
1	1		0	1	la variable es reseteada

RST se repetirá tantas veces como se indique en cte: si es 0 no hace nada. Ej.: RST 2 S2 pondrá a

cero 2 variables consecutivas a partir de S2 (o sea S2 y S3) si el acumulador tenía el valor 1 al momento de ejecutar la instrucción.

**RCT (Reset contadores)**

si ACC = 1 0 -> todos contadores

Modo de uso: RCT  
Bytes usados: 1

Pone a cero el estado de cuenta de todos los contadores si el bit 0 del acumulador está en 1 cuando se ejecuta la instrucción.

**RTP (Reset temporizadores)**

si ACC = 1 0 -> todos temporizadores

Modo de uso: RTP  
Bytes usados: 1

Pone a cero el estado de cuenta de todos los temporizadores si el bit 0 del acumulador está en 1 cuando se ejecuta la instrucción.

**SET (Asentar)**

si ACC = 1 1 -> var

Modo de uso: SET var  
Bytes usados: 2

La instrucción SET (ej.: SET var) responde a la tabla siguiente y el valor del acumulador es al momento de ejecutar la instrucción:

Acc	var antes		var desp.	Acc	
0	0		0	0	la variable y el acumulador no cambian
0	1	SET	1	0	la variable y el acumulador no cambian
1	0		1	1	la variable es seteada (el ACC sale en 1)
1	1		1	0	la variable ya estaba seteada (el ACC sale en 0)

Ej.: activar salida 1 si el acumulador está en 1

**SETB (Asentar siempre)**

si ACC = 1 1 -> var

Modo de uso: SETB var  
Bytes usados: 2

Pone el valor de la variable var en 1 si el acumulador está en 1 (bit 0) en el momento de ejecutar la instrucción. El acumulador no es alterado. Ej. : SET S1, la salida 1 pasará a tener el valor 1, si ya estaba en 1 el acumulador no cambia.

**SETIF1 (Asentar si es 1)**

si var 1 = 1 1 -> var 2

Modo de uso: SETIF1 *var1 var2*  
Bytes usados: 3

Similar a la instrucción SET. Pone todos los bits de la variable *var1* en 1 si el bit 0 de la variable *var 2* está en 1. La variable *var 1* puede ser cualquiera entre e0 y re16. La variable *var 2* puede ser cualquiera entre e0 y tpf2. El estado del acumulador es alterado alterado de la misma forma que en la instrucción SET.

### TP (Temporizador fijo)

Modo de uso: TP *Nº\_temp tiempo*  
Bytes usados: 4

Temporizador interno hasta 999.9 segundos con un corte programable. Hay 4 temporizadores de este tipo.

LD	e3	
TP	tp1	60.5
SET	s1	

En el ejemplo se ve el método básico para usar un temporizador. La entrada de habilitación o reset es el acumulador (ACC), cuyo valor se toma de la entrada E3. Si ACC está en 1 el temporizador está habilitado, caso contrario es puesto a cero (reset).

El tiempo está fijado en un poco mas de 1 minuto (60.5 segundos), y cuando se cumpla este tiempo (siempre y cuando esté habilitado) se accionará la salida 1.

El uso de la instrucción es: TP *Nº\_temp tiempo*. Para conocer el estado de su salida se invoca de forma similar: Ej.: LD TP1 (cargar el acumulador con el valor de la salida del temporizador 1).

El valor de corte (tiempo) sirve para indicarle al temporizador en que valor de tiempo la salida del mismo debe adoptar el valor 1. Cuando la cuenta de tiempo llega a este valor y la salida pasa a valor 1, el temporizador se detiene y el acumulador queda por única vez con valor 1 (después de ejecutar la instrucción), sino siempre sale en 0.

### TPF (Temporizador fijo rápido)

Modo de uso: TPF *Nº\_temp tiempo*  
Bytes usados: 4

Temporizador interno hasta 99.99 segundos con un corte programable. Hay 2 temporizadores de este tipo.

LD	e3	
TPF	tpf1	60.50
SET	s1	

En el ejemplo se ve el método básico para usar un temporizador. La entrada de habilitación o reset es el acumulador (ACC), cuyo valor se toma de la entrada E3. Si ACC está en 1 el temporizador está habilitado, caso contrario es puesto a cero (reset).

El tiempo está fijado en un poco mas de 1 minuto (60.5 segundos), y cuando se cumpla este tiempo (siempre y cuando esté habilitado) se accionará la salida 1.

El uso de la instrucción es: TPF *Nº\_temp tiempo*. Para conocer el estado de su salida se invoca de forma similar: Ej.: LD TP1 (cargar el acumulador con el valor de la salida del temporizador 1).

El valor de corte (tiempo) sirve para indicarle al temporizador en que valor de tiempo la salida del

mismo debe adoptar el valor 1. Cuando la cuenta de tiempo llega a este valor y la salida pasa a valor 1, el temporizador se detiene y el acumulador queda por única vez con valor 1 (después de ejecutar la instrucción), sino siempre sale en 0.

### TPV (Temporizador variable)

Modo de uso: TPV  $N^{\circ}$ \_temp  
Bytes usados: 3

Temporizador interno hasta 999.9 segundos con un corte programable desde teclado-display. Hay 4 temporizadores de este tipo.

LD e3  
TPV tpv1  
SET s1

En el ejemplo se ve el método básico para usar un temporizador. La entrada de habilitación o reset es el acumulador (ACC), cuyo valor se toma de la entrada E3. Si ACC está en 1 el temporizador está habilitado, caso contrario es puesto a cero (reset).

El tiempo está fijado desde el teclado-display, y cuando se cumpla este tiempo (siempre y cuando esté habilitado) se accionará la salida 1.

El uso de la instrucción es: TPV  $N^{\circ}$ \_temp. Para conocer el estado de su salida se invoca de forma similar: Ej.: LD TPV1 (cargar el acumulador con el valor de la salida del temporizador 1).

El valor de corte (tiempo fijado desde el teclado-display) sirve para indicarle al temporizador en que valor de tiempo la salida del mismo debe adoptar el valor 1. Cuando la cuenta de tiempo llega a este valor y la salida pasa a valor 1, el temporizador se detiene y el acumulador queda por única vez con valor 1 (después de ejecutar la instrucción), sino siempre sale en 0.

### TPVF (Temporizador variable rápido)

Modo de uso: TPVF  $N^{\circ}$ \_temp  
Bytes usados: 3

Temporizador interno hasta 99.99 segundos con un corte programable desde teclado-display. Hay 2 temporizadores de este tipo.

LD e3  
TPVF tpvf1  
SET s1

En el ejemplo se ve el método básico para usar un temporizador. La entrada de habilitación o reset es el acumulador (ACC), cuyo valor se toma de la entrada E3. Si ACC está en 1 el temporizador está habilitado, caso contrario es puesto a cero (reset).

El tiempo está fijado desde el teclado-display, y cuando se cumpla este tiempo (siempre y cuando esté habilitado) se accionará la salida 1.

El uso de la instrucción es: TPVF  $N^{\circ}$ \_temp. Para conocer el estado de su salida se invoca de forma similar: Ej.: LD TPVF1 (cargar el acumulador con el valor de la salida del temporizador 1).

El valor de corte (tiempo fijado desde el teclado-display) sirve para indicarle al temporizador en que valor de tiempo la salida del mismo debe adoptar el valor 1. Cuando la cuenta de tiempo llega a este valor y la salida pasa a valor 1, el temporizador se detiene y el acumulador queda por única vez con valor 1 (después de ejecutar la instrucción), sino siempre sale en 0.

**XOR (O exclusiva)**

ACC XOR var -> ACC

Modo de uso: XOR var  
Bytes usados: 2

Realiza la operación lógica XOR entre la variable *var* y el acumulador. El resultado queda en el acumulador. La variable *var* puede ser cualquiera entre E0 y TPF2.

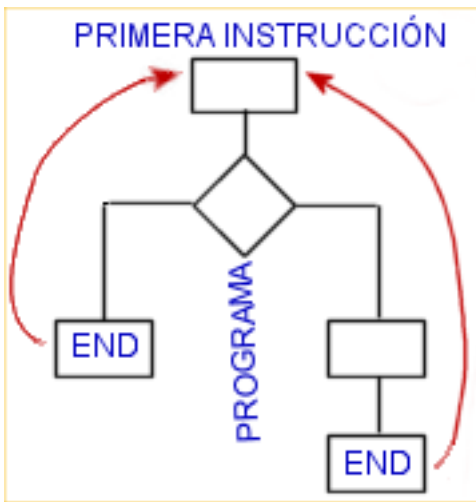
El resultado de la operación XOR será 1 si ambos valores (el del acumulador y el de la variable) son distintos y 0 si son iguales.

LD e2  
XOR v01

En este ejemplo invertirá o no la lectura de la entrada E2 según el valor de la variable V01.

**Funcionamiento del programa**

El PLC ejecutará siempre el programa como si fuese un círculo cerrado, es decir, que cuando llegue al final (en una instrucción END), inmediatamente comenzará a ejecutar desde el principio. A esto se lo llama vuelta de programa. A causa de esto muchas instrucciones son ejecutadas siempre.



No es posible hacer saltos hacia atrás. Por seguridad no está habilitado. Ver ejemplo.

Entre 'vuelta' y 'vuelta' es actualizado el estado de entradas y salidas, display y comunicación. Por lo tanto si se ejecutara un OUT S1 no se manifestará inmediatamente en la salida sino hasta que concluya la vuelta de programa.

LD e1  
LDI 1  
OUT s1

Internamente el estado de la salida 1 cambia como se ve en el ejemplo, pero la salida física del PLC adoptará el último estado definido por el programa cuando termine la vuelta de programa.

## NOT e1

Aquí el estado de la entrada 1 es invertido, y las instrucciones de lectura que lean esta entrada leerán el valor invertido. Es que internamente su valor real puede ser alterado. Al iniciar una nueva vuelta de programa el valor interno de la entrada 1 (o cualquier otra) es refrescado con el valor real.

La ejecución del programa es supervisada por el WatchDog

Ver ejemplos de programas.

## Ejemplo programa

Como se ve, en el Paso\_2 no hay END (lo añade el compilador, siempre y cuando sea el último). Tampoco las líneas son terminadas (no llegan a la derecha). El compilador asume que si están terminadas por lo que no es necesario dibujarlas hasta el final.

La traducción del diagrama a instrucciones queda como sigue:

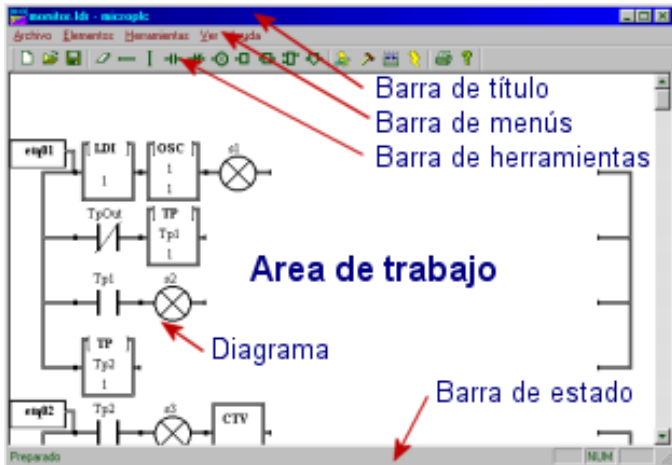
```
ld e1
if1 Paso_2
out s1
end
```

```
Paso_2:
ldi 0
out s1
end
```

El END fue añadido al final del Paso\_2 (por ser el último).

### Descripción Editor

Editor-compilador gráfico (ladder). De arriba hacia abajo tiene los siguientes elementos:



- Barra de título
- Barra de menú
- Barra de herramientas
- Area de trabajo (edición de diagrama)
- Barra de estado.
- Cortar, copiar y pegar (bloques)

Para dibujar un diagrama use el mouse y la paleta de herramientas.

### Barra Menú

Menús desplegables

Archivo Elementos Herramientas Ver Ayuda

Usando la tecla Alt y luego los cursores o el mouse pueden recorrerse todos los menús.

### Archivo

- Nuevo: puede invocarse directamente con las teclas Ctrl + N.
- Abrir: puede invocarse directamente con las teclas Ctrl + A.
- Guardar: puede invocarse directamente con las teclas Ctrl + G.
- Guardar como: igual que guardar, solo que pedirá que se especifique un nombre.
- Imprimir: imprime el diagrama en la impresora selecta. Puede invocarse directamente con las teclas Ctrl + I.
- Presentación preliminar: visualización del aspecto que tendría si fuese impreso.
- Configuración impresora: permite setear los modos de la impresora.
- 1 al 4: últimos archivos con los que se ha trabajado. Permite cargarlos inmediatamente.
- Salir: terminar la sesión del Editor-compilador.

## Elementos

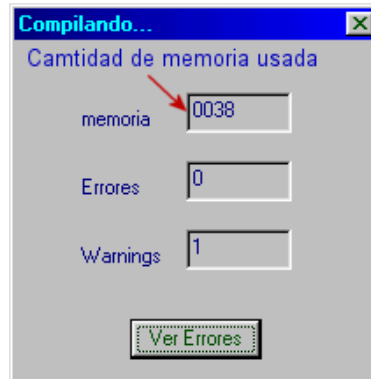
Los mismos elementos del diagrama que aparecen en la barra de herramientas.

## Herramientas

Compilar  
Enviar  
Extraer programa  
Monitor  
Comandos al PLC  
Puerto serie

## **Compilar**

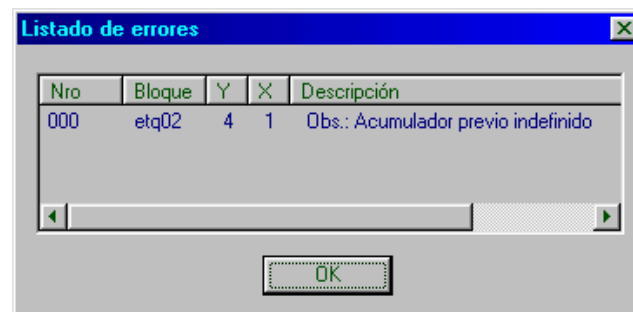
Convierte el diagrama al código binario para enviárselo al PLC



Siempre aparecerá el reporte de memoria usada, errores y advertencias. La extensión máxima del programa es de 256 bytes (memoria disponible).

Errores son aquellos que impiden la compilación. No será posible enviarle (lo que halla resultado) al PLC.

Advertencias (warnings) son aquellos errores que pueden no ocasionar inconvenientes o ser una simple observación. En el caso de haber errores o advertencias siempre se presentará un reporte de donde y tipo de error.

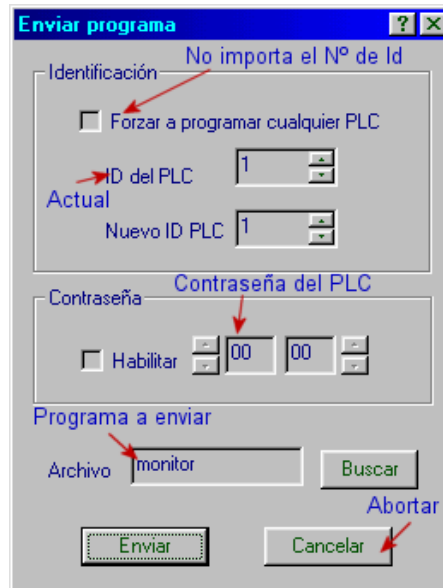


Haciendo click sobre algún ítem en la columna Nro. se resaltará inmediatamente sobre el diagrama el elemento cuestionado.

Si no hubo errores graves el paso siguiente es el envío del programa.

### Enviar Programa

Ventana previa al envío del programa.



Hay 2 modos de enviarle el programa al PLC:

- Por una red distinguiéndolo por su ID
- Forzándolo (tomará el programa sin tener en cuenta el ID)

Puede asignarse un nuevo N° de ID a un PLC que está en red cambiando el valor en el casillero correspondiente.

La contraseña permite proteger el programa: no puede ser extraído del PLC o accederse a los parámetros (desde el teclado) si esta no se conoce.

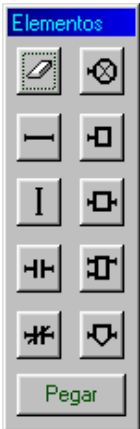
Puede enviar otro programa cambiando el nombre del programa a enviar o buscándolo con el botón buscar.

Si el PLC entró en WatchDog deberá abortarse el programa antes de poder enviarle otro (ver teclado).

### Paleta de herramientas

Con un click derecho sobre el área de edición aparece la paleta. Haciendo click sobre cada elemento de esta se elige para luego colocarlo en el diagrama (con otro click). Inmediatamente aparecerá la ventana de edición de parámetros, donde se asignará el valor correcto al elemento (N° de entrada, o salida, o tiempo o lo que corresponda al tipo de elemento selecto).

El botón pegar funciona en conjunto con los bloques (cortar y copiar). Permite recuperar el bloque, o sea colocarlo en el diagrama a partir de donde se esté posicionado el puntero del mouse.



Conjunto de elementos para construir el diagrama. Están todos repetidos en la barra de herramientas.

## Comandos al PLC

Controla al PLC vía comunicación serie



El botón "Proceder" realiza la acción selecta. Detener el programa tiene la opción de apagar las salidas cuando este es detenido.

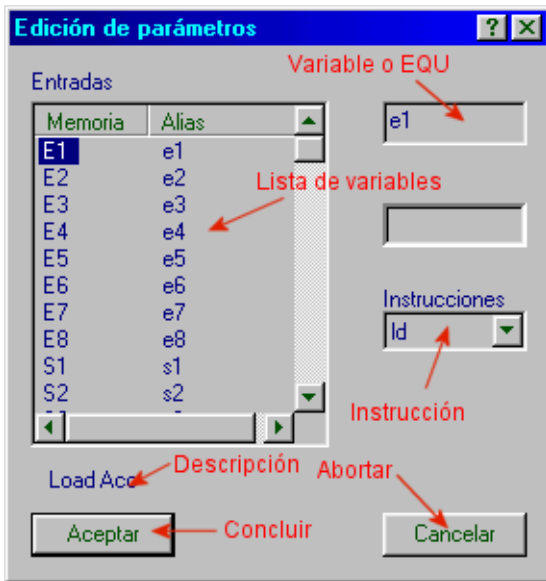
Ejecutar programa tiene 3 opciones:

1. Reset Total: equivale a apagarlo y volver a encenderlo
2. Reset variables: vuelve a 0 el valor de las variables.
3. Mantiene variables y el punto de ejecución del programa para poder continuarlo.

Reset Contadores fuerza a 0 el estado de cuenta y la salida de todos los contadores.

## Edición de parámetros

[Ver instrucciones](#) [Descripción editor](#)



Ventana típica para insertar o modificar un elemento del diagrama.

**Variable o EQU (Alias):** nombre de la variable nombre descriptivo asignado. Aquí la variable E1 (entrada) tiene asignado el nombre e1, pero puede usarse otro mas adecuado al caso. No es igual en todas las instrucciones.

- Elija la variable de la lista haciendo un click sobre ella.
- Haga un click dentro del casillero de Variable o EQU.
- Escriba el nombre que desee asignarle.
- Haga click sobre el botón Aceptar.

**Lista de variables:** todas las variables (*var*) disponibles para el usuario.

**Instrucción:** lista desplegable con un menú de instrucciones.

**Cancelar:** desechar todas las operaciones realizadas.

**Aceptar:** concretar todas las operaciones realizadas.

**Descripción:** descripción breve de la instrucción usada.



## CAJA 1

Este elemento agrupa las siguientes instrucciones:

END  
GOTO  
RJP

## CAJA 2



Este elemento agrupa las siguientes instrucciones:

CLRIF1  
GOSUB  
NOT  
NOTA  
OSC  
RCT  
RST  
RTP  
SET  
SETB  
SETIF1  
TP  
TPF  
TPV  
TPVF  
XOR

## **CAJA CONTADOR**



Este elemento agrupa a los 4 contadores representados por el siguiente cuadro.

Ver contador

## **SALTO CONDICIONAL**



Este elemento agrupa a las siguientes instrucciones.

IF0  
IF1  
IFV1  
JP0  
JP1  
JPV1

## **LINEA HORIZONTAL**



Dibuja una línea horizontal con cada click en la posición del cursor. Se usa para unir los elementos del diagrama.

Haciendo click derecho puede seleccionarse secuencialmente línea vertical, horizontal y borrado (aparte de abrir la paleta de elementos)

...con 1 click



...con mas click en lugares distintos.



## LINEA VERTICAL



Dibuja una línea vertical con cada click en la posición del cursor. Se usa para unir los elementos del diagrama.

Haciendo click derecho puede seleccionarse secuencialmente línea vertical, horizontal y borrado (aparte de abrir la paleta de elementos)

...con 1 click



...con mas click en lugares distintos.

## BORRADOR



Seleciones esta herramienta para borrar algún elemento del diagrama.

No permite recuperar el elemento borrado

Asegúrese que no esté selecto en la barra de herramientas antes de hacer click en algún elemento para editarlo.

## Barra de Herramientas

Barra de herramientas. Aquí puede verse el elemento seleccionado.



### Herramientas de archivo




Crear un diagrama nuevo. Si ha realizado cambios en el diagrama antes de elegir esta opción se le preguntará si quiere *Descartar los cambios*.





Abrir un diagrama existente (cargarlo desde el disco). Si ha realizado cambios en el diagrama antes


de elegir esta opción se le preguntará si quiere *Descartar los cambios*.


 Guardar (salvar) el diagrama en disco. Si ha hecho modificaciones en el diagrama, las grabará. Si es un diagrama nuevo (que no tiene nombre) primero le pedirá que le asigne uno. Es necesario para poder grabarlo.


## Herramientas de creación del diagrama


 Borrar (eliminar un elemento del diagrama). **Observar el elemento selecto de la barra antes de hacer click sobre el diagrama para evitar borrado accidental.**


 Línea horizontal. Se usa para unir los elementos del diagrama.

 Línea vertical. Se usa para unir los elementos del diagrama.


 Inserta la instrucción LD en el diagrama.


 Inserta la instrucción LDN en el diagrama.

 Inserta la instrucción OUT en el diagrama.


 Agrupación de instrucciones.


 Agrupación de instrucciones.


 Contadores.

 Instrucciones de salto.

## Herramientas de comunicación y compilación


 Abre la ventana del monitor de variables del PLC

 Abre la ventana de comandos del PLC

 Compila el diagrama para enviarlo al PLC.

 Envía programa compilado al PLC.

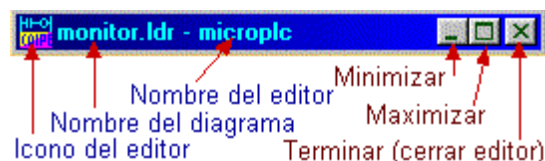
## Impresión y ayuda

 Imprime el diagrama. La impresión queda a cuenta de la impresora y su respectivo software.

 Invoca esta ayuda.

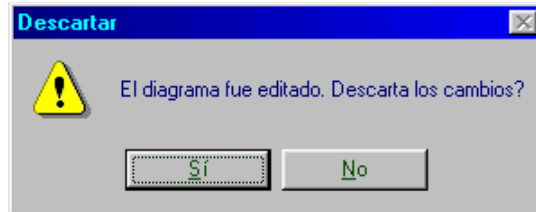
## Barra Título

Nombre del diagrama + nombre del editor



## Nuevo Diagrama

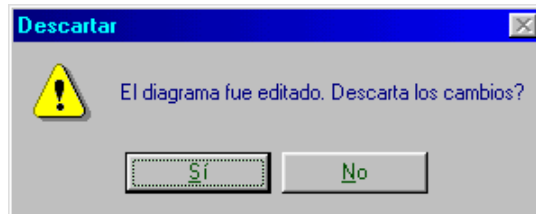
Crear un diagrama nuevo. Si ha realizado cambios en el diagrama antes de elegir esta opción se le preguntará si quiere "Descartar los cambios".



Si se elige "Si" el diagrama actual se perderá. Si se elige "No" se cancela la acción.

## Abrir Diagrama

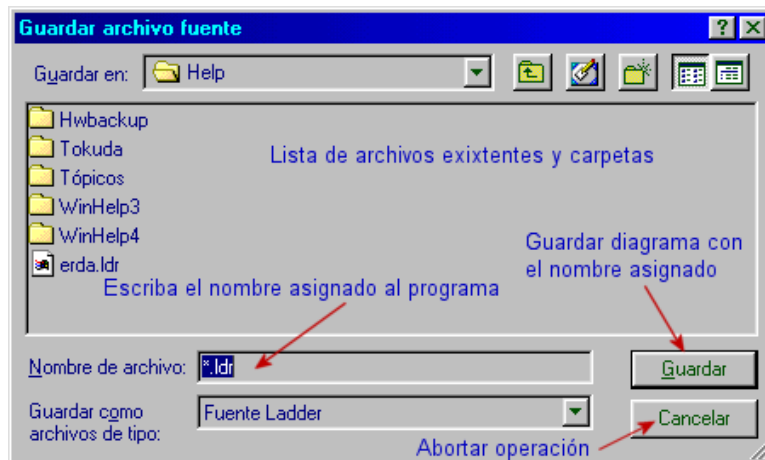
Abrir un diagrama existente (cargarlo desde el disco). Si ha realizado cambios en el diagrama antes de elegir esta opción se le preguntará si quiere *Descartar los cambios*.



Si se elige "Si" el diagrama actual se perderá. Si se elige "No" se cancela la acción.

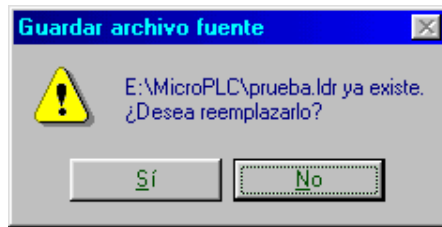
## Salvar Diagrama

Guardar (salvar) el diagrama en disco. Si ha hecho modificaciones en el diagrama, las grabará. Si es un diagrama nuevo (que no tiene nombre) primero le pedirá que le asigne uno. Es necesario para poder grabarlo.



Cuando se halla guardado esta ventana se cerrará sola. Si no se especifica un nombre la ventana persistirá.

Si ya existe un archivo con ese nombre el editor le preguntará si quiere reemplazarlo.



### Monitor de variables

El monitor tiene 2 modos de inspeccionar el valor de las variables (selectas) del PLC en tiempo real.

- 1) Desde la ventana del monitor
- 2) Sobre el diagrama. El color **verde** representa un 0 (apagado) y **rojo** representa 1 (encendido)

Ventana monitor



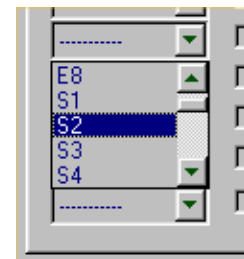
Haciendo click sobre el botón se activa el monitor directo sobre el diagrama. La barra de herramientas será sustituida por 2 botones (Recibir y Salir) y el estado de la comunicación.

El estado de la comunicación con el PLC se aprecia en la indicación .



Ambos puntos deben alternarse entre verde y rojo (transmisión y recepción), de lo contrario denotará que la comunicación no está funcionando.

En la ventana Monitor pueden verse simultáneamente 8 variables que pueden libremente.



Como puede verse, hay 4 grupos de estas 8 variables llamados Mapas. todas las variables selectas (32 en total) pueden guardarse como una configuración del monitor para poder usarse posteriormente.



Con estos botones se puede salvar o recuperar la configuración. Obviamente cada configuración tiene un nombre que se le asigna al momento de guardarla.



Número de identificación del PLC



Permite asociar las variables con un diagrama para poder ver las variables con el nombre asignado o EQU. La asociación es automática: solo basta escribir el nombre del diagrama o buscarlo con el botón.

### Identificación del PLC (Id PLC)

Permite distinguir al PLC en una red de PLCs



Los PLCs pueden montarse en una red propia del tipo maestro-esclavo. Para individualizar cada uno en la red se le asigna un número único. En teoría podrían haber hasta 255 PLCs en una red.

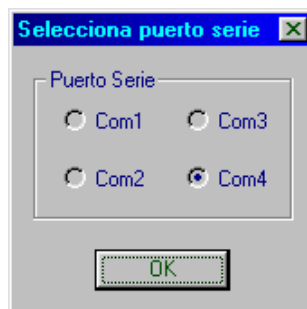
Todos los PLCs son esclavos: están a la espera de recibir una transmisión. Cuando un PLC recibe una transmisión verifica que le corresponda, es decir que tenga su mismo Id. Si no le corresponde descartará la transmisión recibida.

Con este modelo a través de la red se puede:

- Monitorear las variables del PLC.
- Actualizarle el programa.
- Extraerle el programa.
- Detenerlo o ponerlo en marcha.

El maestro o cabecera de red suele ser una PC con el software apropiado.

### Puerto Serie



La mayoría de las PCs tienen solo 2 puertos serie (Com1 y Com2). Elija el puerto que esté disponible para colocar el cable de programación.

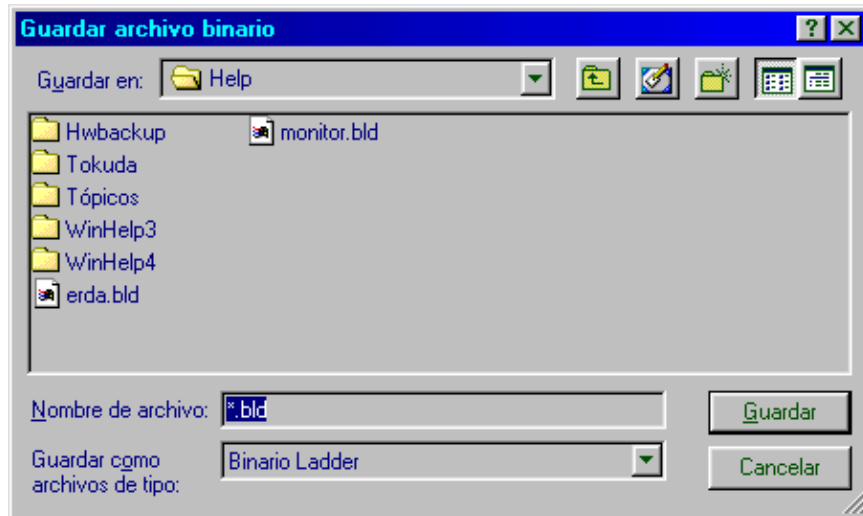
## Extraer programa

Recupera programa binario del PLC

Copia el programa binario de la memoria del PLC a un archivo para poder cargárselo a otro. No es posible convertirlo a diagrama.

### Pasos

1) Se le requerirá que le asigne un nombre:



Si el nombre asignado ya lo tiene otro archivo le preguntará si quiere sobrescribirlo.

2) Debe indicar de cual PLC obtendrá el programa por medio del número de identificación. Si tuviese contraseña deberá habilitarla y asignar el valor de esta entre los 2 casilleros.



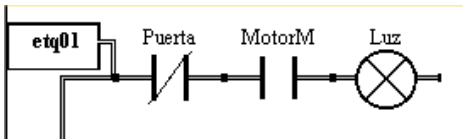
Si no tiene la contraseña no podrá obtener el programa.

**Programando**

Ejemplo de programación 1 (básico)

Supongamos que en una máquina cuando la puerta está abierta y el motor está en marcha debe encenderse la luz del tablero que indique que la puerta debe cerrarse.

Tenemos un sensor NA para la puerta y otro de movimiento para el motor.



```
LDN      Puerta      ;leemos (invertido -NOT) el
                        ;estado del sensor de puerta
AND      MotorM      ;"Y si está el motor en marcha".
                        ;Si la puerta está abierta (Puerta =
                        ;0) y el motor está en marcha
OUT      Luz         ;enciende la luz.
```

Al leer en forma invertida resulta como si el sensor de puerta cerrada fuese NC. Cuando esta está cerrada el acumulador ACC tendrá valor 0 (puerta cerrada = 1, NOT(Puerta) = 0).

Al hacer AND con 0 (0 AND cualquier\_cosa = 0) la salida Luz permanecerá desactivada, y por lo tanto la luz del tablero apagada. Ahora si la puerta está abierta (LDN Puerta = 1) y el motor parado (MotorM = 0) sucede lo mismo que el caso anterior (0 AND cualquier\_cosa = 0).

Solo cuando la puerta está abierta (LDN Puerta = 1, si está abierta) y el motor está en marcha (MotorM = 1) la salida que enciende la luz se activará.

Ejemplo de programación 2

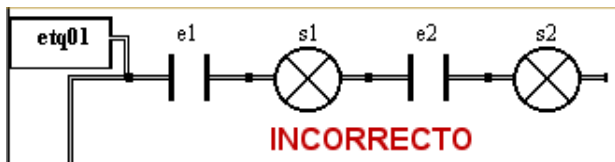
Supongamos que es necesario reflejar el estado de 2 entradas en 2 salidas. Entonces leemos la entrada (LD *entrada*) y la sacamos por la salida (OUT *salida*).

Ejemplo:

```
LD      E1           ;lee entrada
OUT     S1           ;y saca por salida
LD      E2           ;lee entrada
OUT     S2           ;y saca por salida
```

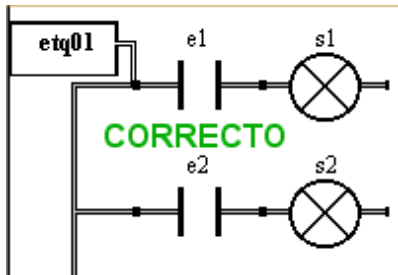
Resultado:

```
Ld  e1
Out s1
And e2
Out s2
```



Resultado:

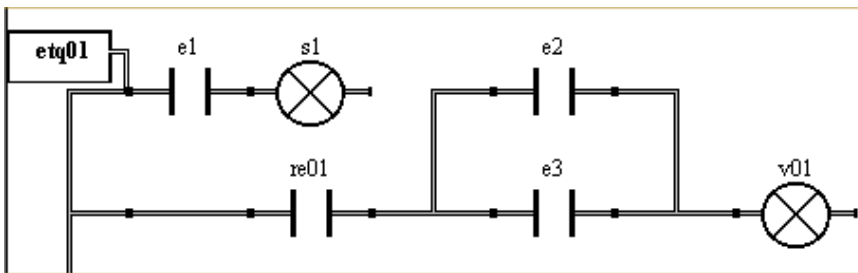
Ld e1  
Out s1  
Ld e2  
Out s2



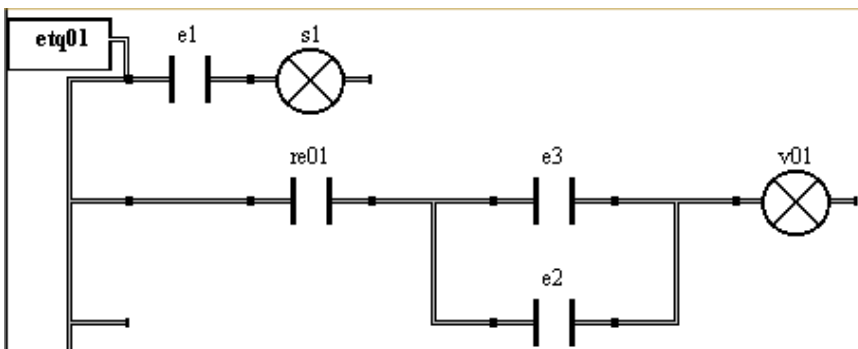
Este ejemplo nos advierte de la necesidad de escribir cada paso del programa en forma ordenada y sin que interfiera con otros. Para esto es necesario:

- Tener un objetivo claro.
- Haber hecho un análisis de todos los pasos necesarios para alcanzar el objetivo.

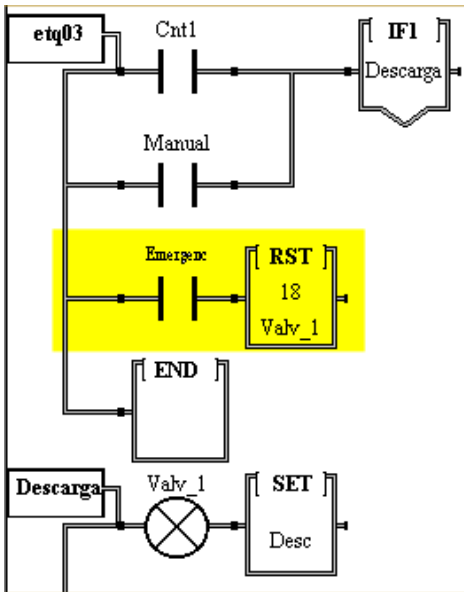
En el siguiente caso se ven 2 ramas independientes. El compilador lo interpretará bien, pero puede confundirse el que lee:



Esta es la forma correcta para el caso anterior para evitar confusiones, especialmente si se es novato:



Ejemplo de programación 3



El ejemplo tiene un salto (IF1 Descarga) que está antes que la parada de emergencia (resaltado en amarillo). Esto significa que en determinadas circunstancias (cuando en cada vuelta de programa se ejecute el salto) la parada de emergencia no tendrá efecto.

A causa de este ejemplo se deduce que es necesario observar el orden y las prioridades de los pasos del programa. Porque de la misma forma pueden saltarse pasos que pueden interpretarse como una falla del equipo cuando fue un error lógico.

#### Ejemplo de programación 4

Para retener un evento, como cuando un pulsador es presionado, usamos las instrucciones SET y RST.

```
LD    e2
SET   s1
```

La entrada E2 es leída y el evento (activación) es retenido en la salida S1. Esta "captura" del evento es realizada por la instrucción SET. Aunque la entrada sea desactivada el estado de la salida permanece.

```
LD    e3
RST   2    s2
```

En este caso el efecto es opuesto: cuando la entrada es activada se desactivan las salidas S2 y S3. La instrucción RST cumple la función opuesta a SET.

### Comunicación

El PLC posee un puerto de comunicación serie RS232 a 2400BPS. Por este canal se monitorea, programa y controla.

Es accesible por 3 bornes en la parte trasera. La razón de los bornes es que está pensado para una instalación fija (como una red).

Configuración del puerto

El puerto serie debe estar seteado en 2400Bps, 8 bits de palabra, sin paridad, 1 bit de stop (2400,8,n,1).

Protocolo de interrogación

Es posible saber el estado de las variables interrogándolo vía comunicación serie. Un paquete de 7 bytes es el total de información para la interrogación y la respuesta.

Id	CMD	Add	01	Add	01	CHK
----	-----	-----	----	-----	----	-----

Id : número de identificación del PLC  
CMD: comando 8  
Add : dirección a leer (ver memoria)  
1 : marca  
CHK : función XOR desde el 1º al 6º byte del paquete

Ejemplo

Leer el estado de las entradas E1 y E2, y la salida S6 y la variable V01 correspondientes al PLC 5. El paquete se compone de la siguiente forma:

05	08	00	01	13	01	CHK
----	----	----	----	----	----	-----

Como la interrogación es para 2 valores de 2 bytes (contiguos), poniendo la dirección del primero se leen ambos.

El cálculo del CHK se realiza de la siguiente manera:

$$\text{CHK} = 0$$
$$\text{CHK} = \text{Id XOR CMD XOR Add XOR 1 XOR Add XOR 1}$$

Para los valores propuestos  $\text{CHK} = 0$ . El orden para transmitir los datos es empezando por el Id y terminando por el CHK.

Si todo está en orden el PLC contestará de la siguiente forma:

05	08	E1	E2	S1	V01	CHK
----	----	----	----	----	-----	-----

En este caso el CHK es calculado de la misma forma por el PLC para corroborar la integridad del paquete enviado. Si realizamos la operación con el paquete recibido

$$\text{MiCHK} = 0$$
$$\text{MiCHK} = 5 \text{ XOR } 8 \text{ XOR } \text{E1} \text{ XOR } \text{E2} \text{ XOR } \text{S1} \text{ XOR } \text{V01}$$

y comprobamos que  $\text{MiCHK} = \text{CHK}$  podemos tener la certeza de que la integridad de los datos es buena.

Programa ejemplo

MON.BAS incluido en el paquete de instalación.

```
'-----  
' Bellplast S.R.L. - http://www.caipe.com  
'-----  
'  
' Ejemplo de comunicación con micro PLC CAIPE  
'CP21. Este ejemplo no tiene soporte ni  
'garantía: solo es para fines didácticos  
'  
'Escrito por GAK - 24/04/01  
'MS-DOS QBasic, Quick Basic, QuickBasic Extended  
'y Visual Basic para MS-DOS  
'-----  
'  
'&H3F8 - COM1:          'puertos serie  
'&H2F8 - COM2:  
'&H3E8 - COM3:  
'&H2E8 - COM4:  
  
DIM Tx%(7), Rx%(7)      'matrices de transmisión  
                        'y recepción  
Bse% = &H2E8           'uso COM4:  
Baudios% = 2400         'velocidad  
CLS                     'limpio pantalla  
'-----  
' Inicialización del puerto serie  
' uso directo de la UART  
'-----  
'El puerto serie debe estar seteado en 2400Bps, 8  
'bits de palabra, sin paridad, 1 bit de stop  
'(2400,8,n,1).  
'  
BaudRate! = FIX(115200 / Baudios%)  
'  
OUT Bse% + 3, &H80      'habilito DLEN  
OUT Bse% + 0, CINT(BaudRate!)  
OUT Bse% + 1, &H0  
OUT Bse% + 3, &H3      '8n1  
Aux% = INP(Bse% + 4)  
OUT Bse% + 4, Aux% AND &HE  
'-----  
' Armado de paquete de interrogación  
'-----  
'Un paquete de 7 bytes es el total de información  
'para la interrogación y la respuesta.  
'Id CMD Add 1 Add 1 CHK  
'  
Tx%(1) = 1             'Id PLC = 1  
Tx%(2) = 8             'comando al PLC  
Tx%(3) = 0             'dirección a leer  
Tx%(4) = 1             'cte.  
Tx%(5) = 13            'dirección a leer
```

```

Tx%(6) = 1                                'cte.
Tx%(7) = 0                                'Checksum
'-----
'   Cálculo del CHK
'-----
'CHK : función XOR desde el 1° al 6° byte del
'paquete
'
FOR i% = 1 TO 6
    Tx%(7) = Tx%(i%) XOR Tx%(7)
NEXT
'-----
'   Transmisión del paquete
'-----
FOR i% = 1 TO 7                            '7 bytes
WaitForFreeTx:
    xtx% = (INP(Bse% + 5) AND &H20) XOR &H20
    IF xtx% = 0 THEN                        'si la UART
        OUT Bse%, Tx%(i%)                  'está vacía
    ELSE                                     'transmito
        GOTO WaitForFreeTx                 'sino espero
    END IF
NEXT
'-----
'   Recepción de respuesta
'-----
i% = 1
CHK% = 0
DO
    IF INKEY$ <> "" THEN EXIT DO           'aborta al
                                           'presionar una tecla
    Aux% = INP(Bse% + 5) AND 1              'espero byte
    IF Aux% = 1 THEN                        'si ya está
        Aux% = INP(Bse%)                   'lo tomo
        Rx%(i%) = Aux%
        CHK% = CHK% XOR Aux%               'calculo CHK
        i% = i% + 1
    END IF
    IF i% = 8 THEN EXIT DO
LOOP
'
IF CHK% <> 0 THEN PRINT "Error de checksum"
'-----
'   Presenta respuesta
'-----
PRINT
PRINT "Dirección   valor"
PRINT Tx%(3), Rx%(3)
PRINT Tx%(3) + 1, Rx%(4)
PRINT Tx%(5), Rx%(5)
PRINT Tx%(5) + 1, Rx%(6)

```

### WatchDog (perro guardián)

Supervisor interno de ejecución de programa. Si por alguna razón no llegara a completarse la vuelta de

programa, o este estuviese corrompido o no es la versión correcta de ROM y compilador el WatchDog interrumpirá el funcionamiento, apagará las salidas e indicará un mensaje por display.

Para salir de esta condición es necesario abortar (ver teclado)

### Guardar como

Guardar como: igual que guardar, solo que pedirá que se especifique un nombre.

### Acerca de

Muestra la versión y Copyright del Editor-compilador ladder CAIPE

### Preparar impresora

Acceso a los ajustes de la impresora instalada. Los ajustes o seteos corresponden a su propio software.

### Vista preliminar

Permite ver el aspecto que tendría el diagrama si fuese impreso en la impresora instalada.

### Historial

Carga el diagrama con ese nombre siguiendo las pautas de cargar un diagrama. Es uno de los últimos 4 diagramas con los que se ha trabajado. De esta forma puede recuperarse rápidamente.

### Salir

Termina la sección del Editor-compilador (sale del programa).

### Cortar Copiar Pegar

Permite definir un bloque del diagrama para copiarlo, eliminarlo o moverlo. Se activa presionando la tecla Shift y haciendo click (Shift + click).

Definir (o marcar) un bloque

- Hacer click sobre el botón Copiar.
- Llevar el puntero del mouse hasta el lugar donde se comenzará a marcar el bloque.
- Presionar la tecla Shift + click y sin soltar (ambas) arrastrar el mouse hasta que dentro del bloque queden los elementos que se desean seleccionar.
- Soltar Shift + click y el bloque quedará marcado.

### Cortar

Haciendo click en el botón Cortar el bloque (o sea, los elementos selectos) será quitado del diagrama y retenido en la memoria temporaria del editor.

### Copiar

Haciendo click en el botón Copiar el bloque (o sea, los elementos selectos) será copiado del diagrama y retenido en la memoria temporaria del editor.

### Pegar

Permite recuperar el bloque, o sea colocarlo en el diagrama. Esta función está en la paleta de herramientas.

### Cancelar

Aborta la operación.

### Monitor Gráfico - recibir/detener



Permite iniciar o detener el monitoreo de las variables directamente sobre el diagrama, observar el estado de la comunicación y salir de esta modalidad.

El estado de las variables es leído entre vuelta y vuelta de programa, por lo que se representará el último valor asignado antes de terminar la vuelta de programa.

## Glosario

### ACC

Acumulador. La mayoría de las operaciones son hechas con el acumulador. En este siempre queda el resultado.

### ACC1

1º acumulador secundario de 3.

Cada vez que se ejecuta una instrucción con rotación se produce un movimiento de acumuladores secundarios:

ACC > ACC1 > ACC2 > ACC3

Las instrucciones ANDP y ORP producen una rotación inversa.

### click

Operación realizada con el mouse en donde se mueve este hasta que el puntero visto en la pantalla se posicione en el elemento deseado y luego se presiona el botón izquierdo donde se podrá apreciar el característico sonido "click".

### click derecho

Operación realizada con el mouse en donde se mueve este hasta que el puntero visto en la pantalla se posicione en el elemento deseado y luego se presiona el botón derecho donde se podrá apreciar el característico sonido "click".

### cte

Constante: un valor numérico.

### cursores

4 teclas (con flechas) utilizadas para realizar 4 movimientos distintos.

**EQU**

Significa equivalencia (como el signo igual).Ej.: arg1 EQU arg2. Hace que el argumento 1(arg1) sea lo mismo que el argumento 2 (arg2). Dicho de otra manera significa que un objeto tiene 2 nombres, y cualquiera que se use es siempre el mismo objeto.

Por medio de esta instrucción podemos facilitar la escritura y comprensión del programa. Ej.:

```
e1      equ      parada
s1      equ      motor
.
.
CLRIF1  motor    parada
.
```

De esta forma es mucho mas comprensible que CLRIF1 s1 e1

**equ**

Significa equivalencia (como el signo igual).Ej.: arg1 EQU arg2. Hace que el argumento 1(arg1) sea lo mismo que el argumento 2 (arg2). Dicho de otra manera significa que un objeto tiene 2 nombres, y cualquiera que se use es siempre el mismo objeto.

Por medio de esta instrucción podemos facilitar la escritura y comprensión del programa. Ej.:

```
e1      equ      parada
s1      equ      motor
.
.
CLRIF1  motor    parada
.
```

De esta forma es mucho mas comprensible que CLRIF1 s1 e1

**label**

Etiqueta: nombre asignado a una línea. Se usa para las instrucciones de salto o llamada. Cada salto o llamada se hace a un label que corresponde a una sola línea. Ej: Label Instrucción

**memoria temporaria**

Lugar para guardar transitoriamente datos o una porción del diagrama como si fuera un portapapeles. No es el mismo portapapeles de Windows.

**memoria usada**

Cantidad de bytes usados. El total de memoria disponible para el programa es de 256 bytes.

**Nº temp**

Nombre distintivo de cada temporizador:

Fijos: de TP1 a TP4  
Fijos rápidos: TPF1 y TPF2  
Variables: de TPV1 a TPV4  
Variables rápidos: TPVF1 y TPVF2

**rep**

Repetición (veces).

**ROM**

Read Only Memori (Memoria de Solo Lectura). Es donde está grabado en forma inalterable el sistema operativo del PLC, encargado de manejar el teclado, display, comunicación y ejecutar el programa.

**tiempo**

Valor expresado en segundos. Puede tener decimales o centesimales, según la posición del punto decimal.

**tiempo off**

Tiempo que permanece desactivado.

**tiempo on**

Tiempo que permanece activado.

**toff**

Tiempo que permanece desactivado.

**ton**

Tiempo que permanece activado.

**var**

Una variable cualquiera del PLC: pueden ser entradas, salidas, variables de usuario, variables con retención, contadores o temporizadores.

**Bellplast S.R.L.**  
[www.caipe.com](http://www.caipe.com)

***CAIPE Automatización***

**Bellplast S.R.L.**

Av. Hipólito Yrigoyen 2166  
(1870) Avellaneda - Buenos Aires  
Argentina



Tel. 5411-4218-1840 (Rot) Fax  
5411-4218-1844

[www.caipe.com](http://www.caipe.com)

**En Uruguay**

**Mones Rose 6163 BIS –  
Carrasco  
Montevideo**

Tel 00598-2-6004992

[www.internet.com.uy/amp/caipe](http://www.internet.com.uy/amp/caipe)