

GUÍA DE USO

Versión 0.75



"IDE4PLC Guía de Uso" por Marcelo F. Chichirí se distribuye bajo una Licencia Creative Commons Atribucion - Compartir Igual 4.0 Internacional. Para ver una copia de esta licencia, visita: <http://creativecommons.org/licenses/by-sa/4.0/>



Índice

1	Introducción	5
1.1	Introducción	5
1.2	Divisiones de esta guía	5
1.3	Alcance	5
1.4	Licencia de uso de esta Guía	6
2	Visión general del sistema.....	7
2.1	¿Qué es un PLC?	7
2.1.1	Programación de un PLC	7
2.1.2	Funciones de un PLC.....	7
2.1.3	Aplicaciones de los PLC	7
2.1.4	Entorno de Programación de PLC	7
2.2	¿Qué es IEC-61131?.....	8
2.3	¿Qué es el lenguaje Ladder?	9
2.4	¿Qué es IDE4PLC?.....	9
2.5	¿Qué es la CIAA?	10
2.6	¿Qué es la EDU-CIAA?	11
2.6.1	Bloques funcionales	11
2.6.2	Diagrama en bloques de la plataforma	11
2.7	Hardware de entrada y salida de la EDU-CIAA.....	12
2.7.1	Pulsadores TEC	13
2.7.2	Indicadores LED	13
3	El entorno IDE4PLC.....	15
3.1	Ejecución de IDE4PLC	15
3.2	Interfaz Gráfica de Usuario (GUI)	15
3.2.1	Ventana Board (Selección de tarjeta a usar).....	17
3.2.1.1	Placa EDU-CIAA-NXP	17
3.2.1.2	Placa CIAA-NXP	18
3.2.2	Editor de Unidad de Organización de Programa (POU)	20
3.2.2.1	Declaración de variables	21
3.2.2.1.1	Agregar nuevas variables	22
3.2.2.1.1.1	Variables del tipo VAR.....	23
3.2.2.1.1.2	Variables del tipo VAR_TEMP	23
3.2.2.1.2	Eliminar variables	24
3.2.2.2	Generar código C	24

3.2.2.3	Compilar Código C.....	24
3.2.2.4	Descargar Programa	24
3.2.2.5	Generar código C, Compilar y Descargar al dispositivo	25
3.2.2.6	Limpiar Compilación de Código C	25
3.2.2.7	Guardar POU.....	26
3.2.2.8	Eliminar POU	26
3.2.2.8.1	Haciendo un programa Ladder	27
3.2.2.8.1.1	Insertando un nuevo elemento Laddder	29
3.2.2.8.1.2	Configurando un elemento Ladder	30
3.2.2.8.1.3	Editando y eliminando un elemento Laddder.....	30
3.2.2.8.1.4	Abriendo y cerrando una rama paralela	31
3.2.2.8.1.5	Accesos directos	35
4	Elementos del Lenguaje Ladder	37
4.1	Elementos del lenguaje Ladder	37
4.1.1	Tipos de datos soportados por IDE4PLC	37
4.1.1.1	Tipo de dato TIME	37
4.1.1.2	Tipo de dato INT.....	37
4.1.1.3	Tipo de dato BOOL.....	37
4.1.2	Datos generales de los elementos Ladder	38
4.1.3	Resumen de los elementos del lenguaje Ladder.....	38
4.1.3.1	Resumen de Contactos y Bobinas.....	38
4.1.3.2	Llamado a funciones	39
4.1.3.2.1	Funciones de Comparación	39
4.1.3.2.2	Resumen de Funciones de Aritméticas	40
4.1.3.3	Llamado a Bloque de Funciones	42
4.1.3.3.1	Resumen de Funciones de Circuitos Biestables	42
4.1.3.3.2	Resumen de Funciones de detección de Flancos.....	42
4.1.3.3.3	Resumen de Funciones Contadores	43
4.1.3.3.4	Resumen de Funciones Temporizadores	44
4.1.4	Descripción de Elementos Ladder	45
4.1.4.1	Descripción de Bobina	45
4.1.4.2	Descripción de Bobina Negada	46
4.1.4.3	Descripción de Contacto Normal Abierto	47
4.1.4.4	Descripción de Contacto Normal Cerrado	47
4.1.4.5	Descripción de las Funciones de Comparación	48
4.1.4.6	Descripción de las Funciones Aritméticas.....	50
4.1.4.7	Descripción de las Funciones de Circuitos Biestables	53

4.1.4.8	Descripción de las Funciones de Detección de Flancos	54
4.1.4.9	Descripción de las Funciones Contadores	56
4.1.4.9.1	Contador ascendente (CTU)	56
4.1.4.9.2	Contador descendente (CTD).....	57
4.1.4.9.3	Contador descendente/descendente (CTUD).....	59
4.1.4.9.4	Resumen de las Funciones Contadores	60
4.1.4.10	Descripción de las Funciones Temporizadores	60
4.1.4.10.1	Temporizador con retardo a la conexión (TON)	61
4.1.4.10.2	Temporizador con retardo a la desconexión (TOF).....	61
4.1.4.10.3	Temporizador de impulsos (TP)	62
4.1.4.10.4	Resumen de Temporizadores	63
5	Programas Ladder ejemplos.....	65
5.1	Circuitos de ejemplo.....	65
5.1.1	Circuito con retención con contactos	65
5.1.2	Circuito con retención con biestables.....	65
5.1.3	LEDs intermitentes	65
5.1.4	Encendido escalonado por cuenta de señales	66
5.1.5	Secuenciador con permanencia	67
5.1.6	Secuenciador con permanencia y selección de velocidad	68
5.1.7	Secuenciador automático de tres leds sin permanencia	68
5.1.8	Secuenciador manual de tres leds sin permanencia.....	69
5.1.9	Secuenciador semiauto encendido/apagado de tres leds con permanencia	70
5.1.10	Secuenciador automático de encendido/apagado de tres leds con permanencia... 71	
5.1.11	Secuenciador auto de enc/apag de tres leds con permanencia e inicio/parada..... 72	
6	Historial de versiones	74
6.1	Versión 1.0.4	74

1 Introducción

1.1 Introducción

El objetivo de esta guía es que todo aquel que esté interesado utilizar una CIAA¹, y en especial una EDU-CIAA², como un PLC³, utilizando el programa IDE4PLC, pueda contar con un documento al cual referirse para dar sus primeros pasos.

Este texto no pretende enseñar a programar en lenguaje Ladder ni dar conocimientos profundos de lo que es un PLC, como funciona ni como se diseña o construye. La idea es dar un marco general para que el lector pueda comenzar a desarrollar sus proyectos en dicho lenguaje y seguir aprendiendo a partir de allí.

Esta guía usa como referencia la versión 1.0.2 de IDE4PLC. Si bien podrían existir diferencias con otras versiones, la mayoría de los comentarios, explicaciones y referencias aquí plasmadas son válidas para todas las versiones de IDE4PLC.

Esta guía se basa en el documento “Informe TF IACI UNQ Eric N. Pernia.pdf” de la Tesis de grado del Ingeniero Eric Pernia de la carrera de “Ingeniería en Automatización y Control Industrial” de la Universidad Nacional de Quilmes, el cual puede ser bajado del sitio “<https://ide4plc.wordpress.com/>”.

También se nutrió de información de la página del proyecto EDU-CIAA, la cual puede ser consultada en “<http://www.proyecto-ciaa.com.ar/>”.

1.2 Divisiones de esta guía

Esta guía se divide en las siguientes partes principales:

- Capítulo 1 (Introducción): Hace una breve introducción, alcance y licencia de uso de esta guía.
- Capítulo 2 (Visión general del sistema): Da una visión general del sistema, de la aplicación IDE4PLC y de la EDU-CIAA NXP.
- Capítulo 3 (El entorno IDE4PLC): Explica como usar la aplicación IDE4PLC.
- Capítulo 4 (Elementos del Lenguaje Ladder): explica cada uno de los elementos del lenguaje Ladder incluidos en IDE4PLC.
- Capítulo 5 (Programas Ladder ejemplos): Da algunos programas de ejemplo en lenguaje Ladder para IDE4PLC para ser ejecutados en EDU-CIAA NXP.
- Capítulo 6 (Historial de versiones): Indica la evolución de las distintas versiones de IDE4PLC.

1.3 Alcance

Esta guía cubre los aspectos de uso del entorno IDE4PLC versión v1.0.4 para la programación de PLC en lenguaje Ladder bajo normas IEC 61131-3, pero no incluye aspectos técnicos del desarrollo de la aplicación ni del hardware de la EDU-CIAA.

Aunque esta guía está basada en Windows, la aplicación se ejecuta de igual manera en Linux y MAC OS X.

Si bien se incluyen algunos ejemplos de programas básicos en lenguaje Ladder para demostrar la funcionalidad de los elementos que lo componen (como contactos, bobinas, bloques de función, etc.) y algunos ejemplos adicionales, este no es un manual para aprender programación en Ladder. Se asume que el lector tiene, al menos, conocimientos básicos de programación en este lenguaje.

Tampoco se incluye el procedimiento de instalación del entorno IDE4PLC para EDU-CIAA.

Se puede obtener muy buena documentación técnica respecto al diseño de la aplicación y del proceso de instalación del software IDE4PLC, así como la última versión el propio software, en las siguientes páginas web:

¹ CIAA : Computadora Industrial Abierta Argentina

² EDU-CIAA : Versión Educativa de la CIAA

³ PLC : Controlador Lógico Programable

<https://ide4plc.wordpress.com/>

<http://www.proyecto-ciaa.com.ar/devwiki/doku.php?id=desarrollo:software-plc>

<https://groups.google.com/forum/#!forum/ciaa-software-plc>

1.4 Licencia de uso de esta Guía

“IDE4PLC Guía de Uso” Se distribuye bajo una Licencia Creative Commons Atribucion - Compartir Igual 4.0 Internacional.



Reconocimiento (Attribution): En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.



Compartir Igual (Share alike): La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.



Reconocimiento – CompartirIgual (by-sa): Se permite el uso comercial de la obra y de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Para ver mayor detalle de esta licencia ver los siguientes sitios:

<http://creativecommons.org/licenses/by-sa/4.0/>

<http://es.creativecommons.org/blog/licencias/>

2 Visión general del sistema

2.1 ¿Qué es un PLC?

Un Controlador Lógico Programable (en adelante PLC, llamado así por sus siglas en inglés), o Autómata Programable, es un sistema electrónico programable diseñado para controlar diversos tipos de máquinas o procesos en tiempo real en un entorno industrial. Está compuesto por un microprocesador o microcontrolador, memorias, interfaces de entradas/salidas y circuitos adicionales.

Puede programarse utilizando distintos lenguajes, tanto textuales como gráficos.

2.1.1 Programación de un PLC

Para programar un PLC se debe comunicar al usuario con el Hardware PLC mediante un dispositivo que permita escribir y poner a punto el programa que se ejecutará. Este dispositivo se conoce como "consola de programación", que puede ser un dispositivo con pantalla diseñado específicamente para tal fin, o bien, una computadora de propósito general, corriendo un Software con las mismas habilidades (más común en la actualidad).

2.1.2 Funciones de un PLC

Un PLC es capaz de:

1. Adquirir datos del proceso por medio de sensores conectados electrónicamente a sus entradas digitales y analógicas.
2. Almacenar datos en memoria.
3. Tomar decisiones mediante un programa en memoria ingresado por el usuario que consiste de instrucciones, las cuales implementan funciones específicas tales como lógicas, secuenciales, temporización, contadores y matemáticas.
4. Actuar sobre el proceso mediante sus salidas digitales y analógicas conectadas eléctricamente a actuadores.
5. Comunicarse con otros sistemas externos.

2.1.3 Aplicaciones de los PLC

El PLC es el equipo más utilizado en la automatización industrial y domótica. Es por esto, que en el desarrollo de la práctica de su profesión un Ingeniero en Automatización y Control Industrial, utiliza estos equipos con mucha frecuencia. Existen seis áreas generales de aplicación de PLC:

1. Control secuencial.
2. Control de movimiento.
3. Control de procesos.
4. Monitoreo y supervisión de procesos.
5. Administración de datos.
6. Comunicaciones.

2.1.4 Entorno de Programación de PLC

El Entorno de Programación le permite al usuario realizar las siguientes tareas:

- Crear y editar un programa utilizando un lenguaje de programación de PLC.
- Compilar el programa.
- Descargar el programa a la memoria del Hardware PLC.
- Grabar el programa creado y todo su entorno de programación.

Para cumplir con estos requerimientos, se plantea el esquema general de la solución en la Figura 2-1, en donde se muestran las partes que constituyen el sistema y sus relaciones.

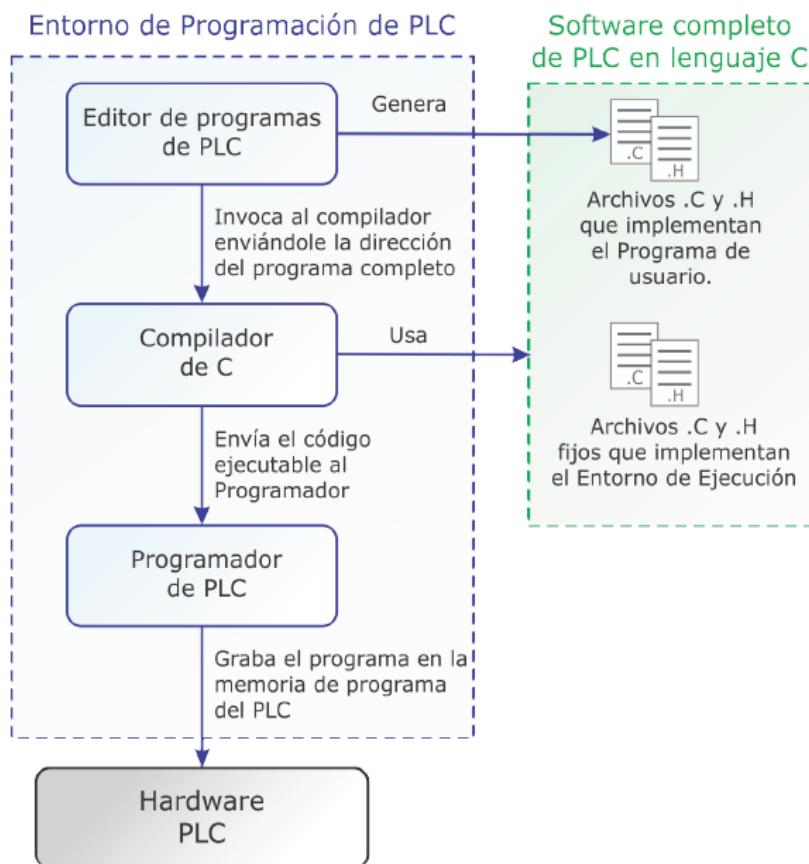


Figura 2-1: Modelo del Sistema

- **Editor de programas de PLC:** El usuario desarrolla su programa, en lenguaje Ladder, de acuerdo a la norma IEC 61131-3, utilizando el Editor de programas IDE4PLC. Este editor es responsable de la traducción del programa de usuario al lenguaje de programación C. Luego, combinando estos archivos generados, con archivos fijos preexistentes, completa el Software de PLC en lenguaje C.
- **Compilador de C:** Posteriormente, invoca al compilador de C enviándole la dirección de dicho Software. El compilador genera el código ejecutable, compilando el programa completo.
- **Programador de PLC:** El código ejecutable generado en el paso anterior es incorporado, mediante un componente al que se llama Programador de PLC, al hardware del PLC (EDU-CIAA o CIAA).

2.2 ¿Qué es IEC-61131?

Existen diversos fabricantes de PLC, tipos y modelos adecuados a las distintas necesidades de la industria. Esto, a través de los años y los avances tecnológicos, producidos desde el primer PLC comercial (alrededor de 1968), generó muchas incompatibilidades entre sistemas, con el consecuente problema para el usuario que debía adaptarse a las particularidades de cada empresa. Adicionalmente, además del lenguaje Ladder inicial, se crearon otros lenguajes que tampoco tenían una norma que los regulara. Para intentar resolver este problema las empresas integrantes de la IEC (*International Electrotechnical Commission*), una organización de normalización en los campos electrónicos, eléctrico y tecnologías asociadas, crearon la norma IEC 61131; logrando el primer paso en la estandarización de los Controladores Lógicos Programables y sus periféricos, incluyendo los lenguajes de programación que se deben utilizar.

La finalidad de esta norma es:

1. Definir e identificar las características principales que se refieren a la selección y aplicación de los PLC y sus periféricos.
2. Especificar los requisitos mínimos para las características funcionales, las condiciones de servicio, los aspectos constructivos, la seguridad general y los ensayos aplicables a los PLC y sus periféricos.

3. Definir los lenguajes de programación de uso mas corriente, las reglas sintácticas y semánticas, el juego de instrucciones fundamentales, los ensayos y los medios de ampliación y adaptación de los equipos.
4. Dar a los usuarios una información de carácter general y unas directrices de aplicación.
5. Definir las comunicaciones entre los PLC y otros sistemas.

Esta norma, está formada por las siguientes partes:

- IEC 61131-1. Información general.
- IEC 61131-2. Especificaciones y ensayos de los equipos.
- IEC 61131-3. Lenguajes de programación.
- IEC 61131-4. Guías de usuario.
- IEC 61131-5. Comunicaciones.
- IEC 61131-6. Reservada.
- IEC 61131-7. Fuzzy Control.
- IEC 61131-8. Guías de programación.

La versión actual de IDE4PLC para EDU-CIAA cubre la implementación de la programación en lenguaje Ladder descripto en la norma IEC 61131 parte 3 del 2003 (IEC 61131-3 : 2003).

2.3 ¿Qué es el lenguaje Ladder?

Es un lenguaje gráfico, derivado de los circuitos de relés. Mediante símbolos representa contactos, bobinas, etc. Los elementos básicos son las bobinas de los relés físicos y los contactos, normal abiertos y normal cerrados, del mismo relé. Adicionalmente a los relé básicos, y valiéndose de la flexibilidad de implementar la lógica de relé de manera programada, los distintos fabricantes le agregaron bloque de funciones de distintos tipos que no se podrían implementar de una manera estrictamente cableada.

Su principal ventaja, por la cual fue aceptado rápidamente por el personal técnico que hacia circuitos con relé, es su similitud con circuitos cableados.

Adicionalmente, otra ventaja es que los símbolos básicos están normalizados según el estándar IEC y son empleados por muchos de los fabricantes actuales de PLC.

Se debe recordar que mientras que en el diagrama eléctrico todas las acciones ocurren simultáneamente, en el programa se realizan en forma secuencial, siguiendo el orden en el que los "escalones" fueron escritos, y que a diferencia de los relés y contactos reales (cuyo número está determinado por la implementación física de estos elementos), en el PLC se puede considerar que existen infinitos contactos auxiliares para cada entrada, salida, relé auxiliar o interno, etc.

2.4 ¿Qué es IDE4PLC?

IDE4PLC es el software que permite ejecutar cada una las tareas descriptas anteriormente que termina descargando el programa del PLC, generado en Lenguaje Ladder, al hardware de la EDU-CIAAA.

Este software fue desarrollado por el Ingeniero Eric Pernia en su trabajo final de grado “Diseño de software y hardware de un controlador lógico programable (PLC) y su entorno de programación” teniendo como Director de Tesis al Dr. Carlos Lombardi, ambos de la carrera de Ingeniería en Automatización y Control Industrial del Departamento de Ciencia y Tecnología de la Universidad Nacional de Quilmes (<http://www.unq.edu.ar/>).

Toda la información relacionada con este trabajo puede consultarse y obtenerse de la pagina [“https://ide4plc.wordpress.com/”](https://ide4plc.wordpress.com/).

El objetivo principal de este entorno de programación es proporcionar a los estudiantes de carreras de automatización, electrónica y sistemas un entorno para el aprendizaje de la programación de PLCs; la decisión de adoptar un tipo de interfaz gráfica que rescata elementos de uso común a las utilizadas en la industria apunta a brindar a los estudiantes una herramienta que, siendo de bajo costo, les permita al mismo tiempo experimentar con interfaces de usuario similares a las que probablemente deberán utilizar en su carrera profesional.

El entorno de programación se implementó sobre Pharo-Smalltalk, un ambiente de desarrollo que permite que el software construido funcione en Windows, Linux y MAC OS X; de esta forma se logró también la

independencia del sistema operativo del equipo en el que se ejecuta dicho entorno. Pharo es una herramienta libre, lo que permitió minimizar los costos de desarrollo.

2.5 ¿Qué es la CIAA?

El Proyecto CIAA (**Computadora Industrial Abierta Argentina**) nació en el año 2013 como una iniciativa conjunta entre el sector académico y el industrial, representados por la ACSE (<http://www.sase.com.ar/asociacion-civil-sistemas-embebidos/>) y CADIEEL (<http://www.cadieel.org.ar/esp/index.php>), respectivamente.

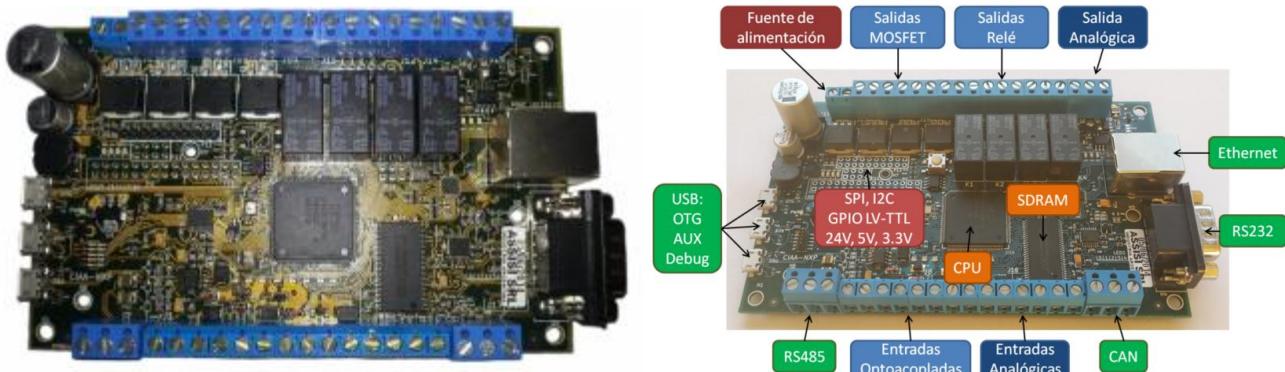


Figura 2-2: Placa CIAA NXP

Los objetivos del Proyecto CIAA son:

1. Impulsar el desarrollo tecnológico nacional.
2. Darle visibilidad positiva a la electrónica argentina.
3. Generar cambios estructurales en la forma en la que se desarrollan y utilizan los conocimientos.

Todo esto en el marco de un trabajo libre, colaborativo y articulado entre industria y academia.

Para lograr los objetivos, el primer paso fue articular el trabajo de decenas de Instituciones, Universidades, Empresas y Desarrolladores para diseñar la primera versión de la CIAA, la denominada "CIAA-NXP", por estar basada en un procesador de la empresa NXP Semiconductors.

La CIAA-NXP es la primera y única computadora del mundo que reúne dos cualidades:

1. Ser Industrial, ya que su diseño está preparado para las exigencias de confiabilidad, temperatura, vibraciones, ruido electromagnético, tensiones, cortocircuitos, etc., que demandan los productos y procesos industriales.
2. Ser Abierta, ya que toda la información sobre su diseño de hardware, firmware, software, etc. está libremente disponible en internet bajo la Licencia BSD, para que cualquiera la utilice como quiera.

Para avanzar aún más se desarrollaron versiones de la CIAA basadas en procesadores de otras marcas, como la CIAA-FSL, la CIAA-INTEL, la CIAA-PIC, etc.

En consecuencia, la CIAA además de ser la primera computadora industrial abierta, es también la primera computadora realmente libre del mundo, ya que su diseño no está atado a los procesadores de una determinada compañía, como ocurre con otras computadoras abiertas.

La CIAA es fabricada y comercializada por distintas empresas argentinas, a las que se pueden sumar muchas más, sin ninguna limitación, ya que su diseño es libre y abierto, al punto de que cualquier interesado puede fabricar su propia versión de la plataforma.

No hay que perder de vista que el Proyecto CIAA es mucho más que hardware, ya que también incluye un entorno IDE para su programación en lenguaje C, el soporte de Linux, un entorno de programación en lenguaje tipo PLC, el diseño de un gabinete y los primeros diseños de algunos de sus circuitos integrados.

Además, se diseñó una versión educativa de la plataforma, la EDU-CIAA, más simple y de menor costo, para lograr un impacto en la enseñanza primaria, secundaria y universitaria.

2.6 ¿Qué es la EDU-CIAA?

La EDU-CIAA es una versión de bajo costo de la versión CIAA. En especial EDU-CIAA NXP es una versión de bajo costo de la CIAA-NXP pensada para la enseñanza universitaria, terciaria y secundaria.



Figura 2-3: Placa EDU-CIAA NXP

2.6.1 Bloques funcionales

La EDU-CIAA está basada en la CIAA-NXP, por ser la primera versión de la CIAA que se encuentra disponible. Por lo tanto su microcontrolador es también el LPC4337 (dual core ARM Cortex-M4F y Cortex-M0).

Sin embargo, con el objetivo de abaratar costos y reducir su complejidad la EDU-CIAA incorpora sólo algunas de las funcionalidades de la CIAA.

A su vez, con el fin de permitir el desarrollo de algunas prácticas sencillas sin que sea necesario recurrir a hardware adicional, incluye además algunos recursos que no están presentes en la CIAA.

2.6.2 Diagrama en bloques de la plataforma

En la siguiente figura se observa un diagrama en bloques de la EDU-CIAA basada en LPC4337.

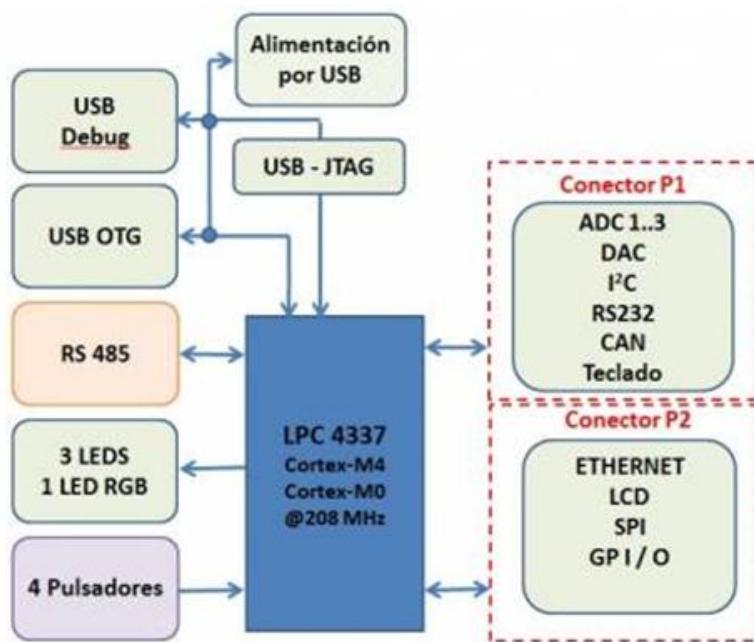


Figura 2-4: Diagrama en Bloques EDU-CIAA NXP

La EDU-CIAA cuenta con los siguientes módulos:

- 2 puertos micro-USB (uno para aplicaciones y debugging, otro para alimentación).
- 4 salidas digitales implementadas con leds RGB.
- 4 entradas digitales con pulsadores.
- 1 puerto de comunicaciones RS 485 con bornera.
- 2 conectores de expansión:
 - P1:
 - 3 entradas analógicas (ADC0...2),
 - 1 salida analógica (DAC0),
 - 1 conexión para un teclado de 3x4,
 - 12 pines genéricos de I/O.
 - P2:
 - 1 puerto Ethernet,
 - 1 puerto CAN,
 - 1 puerto SPI,
 - 1 puerto I2C,
 - 12 pines genéricos de I/O.

Para mayor detalle de la EDU-CIAA se puede consultar los siguientes sitios que contienen una excelente descripción de todo el proyecto CIAA, además de circuitos, especificaciones, software y todo tipo de documentación para el material necesario para entender y trabajar con cada una de las versiones de la CIAA.

<http://www.proyecto-ciaa.com.ar/>

<http://www.proyecto-ciaa.com.ar/devwiki/doku.php?id=desarrollo:edu-ciaa:edu-ciaa-nxp>

2.7 Hardware de entrada y salida de la EDU-CIAA

La EDU-CIAA tiene entradas salidas tanto analógicas como digitales de propósito general ruteados en los conectores P1 y P2, así como también cuatro entradas digitales con pulsadores y siete salidas con LEDs. En el siguiente link se pueden consultar todos los esquemáticos de la placa.

<https://github.com/ciaa/Hardware/blob/master/PCB/EDU-NXP/Doc/Schematic.pdf>

A continuación se muestran los circuitos de entrada de pulsadores y salida de LEDs incorporados en la EDU-CIAA-NXP ya que los mismos son muy útiles a fines educativos porque permiten implementar y probar rápidamente circuitos sin necesidad de elementos adicionales.

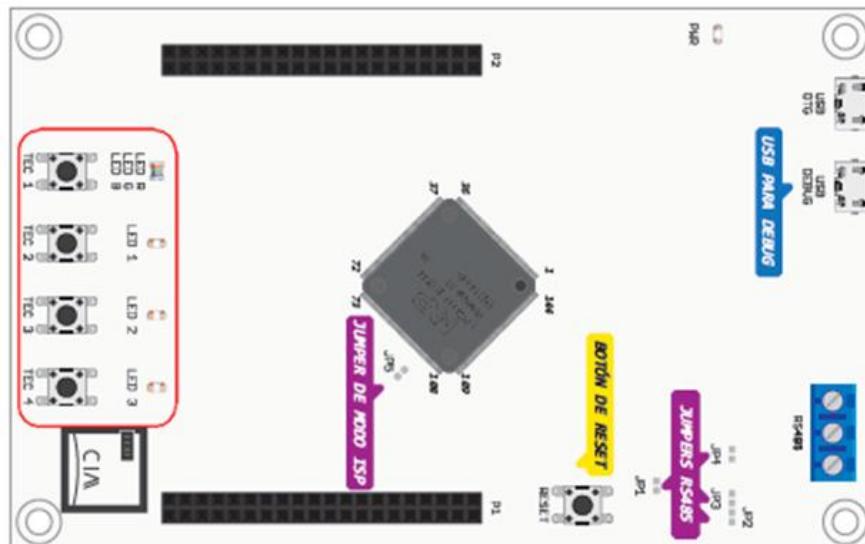


Figura 2-5: Pulsadores y LEDs en la EDU-CIAA

2.7.1 Pulsadores TEC

Como se indicó en 2.6.2, la EDU-CIAA tiene cuatro pulsadores incorporados, identificados en la placa como TEC1, TEC 2, TEC 3 y TEC 4 (correspondiente a las variables TEC1, TEC2, TEC3 y TEC4 de IDE4PLC)⁴. La Figura 2-6 muestra el circuito del TEC 1 que se repite para TEC 2, TEC 3 y TEC 4.

Como se ve en este circuito, la entrada del microcontrolador (en este caso PULS_0) está permanentemente energizada (entrada en "1") mientras el pulsador no está pulsado. Cuando el pulsador se pulsa, la entrada al microcontrolador se pone a masa (entrada en "0"). Este comportamiento es adecuado desde el punto de vista eléctrico pero generaría confusión en circuitos Ladder, ya que un contacto TEC estaría permanentemente activado sin pulsar el pulsador y al hacerlo el contacto del Ladder se desenergizaría.

Para evitar este comportamiento confuso IDE4PLC invierte la lógica de los contactos TEC para que tengan un comportamiento más natural, es decir que el contacto se energize cuando se pulse el pulsador y permanezca desenergizado cuando no esté pulsado.

Lo dicho anteriormente puede resumirse en la siguiente tabla.

Pulsador	Entrada
No Pulsado	Desactivada
Pulsado	Activada

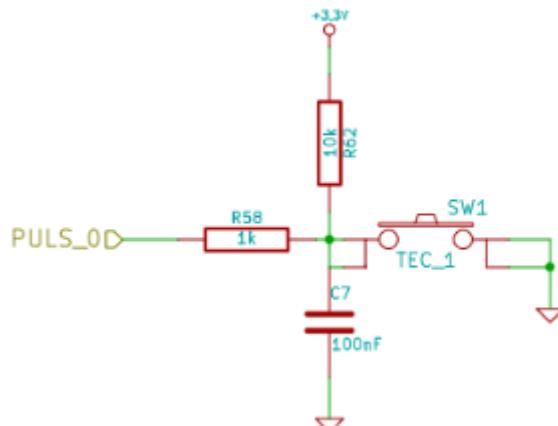


Figura 2-6: Circuito pulsador TEC

2.7.2 Indicadores LED

Como se indicó en 2.6.2, la EDU-CIAA tiene tres LEDs simples, identificados en la placa como LED 1, LED 2 y LED 3 (correspondiente a las variables LED1, LED2 y LED3 de IDE4PLC)⁵ y un LED RGB identificados en la placa como LED R, LED G y LED B (correspondiente a las variables LEDR, LEDG y LEDB de IDE4PLC)⁶. La siguiente figura muestra el circuito del LED 1 que se repite para LED 2 y LED 3, y del LED RGB.

Cada uno de los LEDs se enciende cuando la salida del microcontrolador se activa y el LED permanece apagado cuando la salida correspondiente se desactiva.

Lo dicho anteriormente puede resumirse en la siguiente tabla.

Salida	Estado LED
Activada	Encendido
Desactivada	Apagado

⁴ En versiones anteriores se designaban como TEC_1, TEC_2, etc.

⁵ En versiones anteriores se designaban como LED_1, LED_2 y LED_3.

⁶ En versiones anteriores se designaban como LED_R, LED_G y LED_B.

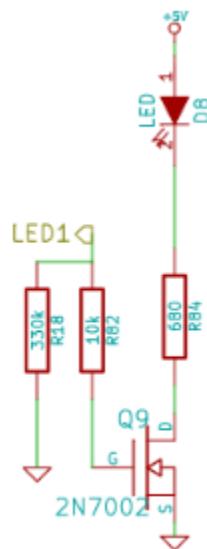


Figura 2-7: Circuito LED 1, LED 2 y LED 3

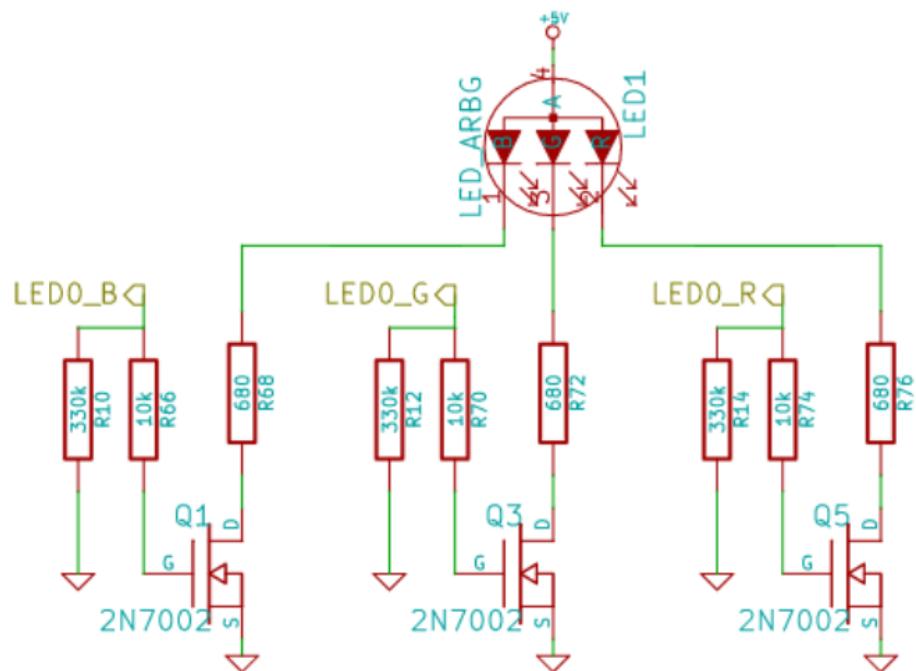


Figura 2-8: Circuito LED RGB

3 El entorno IDE4PLC

3.1 Ejecución de IDE4PLC



El ícono indicado ejecutará IDE4PLC para programar EDU-CIAA en lenguaje Ladder de acuerdo a norma IEC 61131-3.

Si, al ejecutar IDE4PLC, el programa Pharo encuentra solo un archivo .image usará este archivo para presentar el entorno gráfico. Si Usted nunca hizo una copia del archivo Pharo4.0.image Pharo se ejecutará directamente usando ese archivo. Si en el directorio IDE4PLC existe mas de un archivo .image Pharo mostrará todos los archivos .image para que Usted seleccione el archivo que contiene la imagen deseada.

Una imagen contiene las ventanas, programa Ladder, variables, y todos aquellos objetos exactamente en el mismo estado en el que se encontraban cuando fueron grabados.

3.2 Interfaz Gráfica de Usuario (GUI)

El diseño elegido para la interfaz grafica presenta elementos similares a los que se encuentran en herramientas ampliamente utilizadas en la industria.

De esta forma se presenta a los profesionales del área un entorno que les resulta familiar, facilitando la aceptación de los controladores que se desarrolle de acuerdo con el diseño propuesto.

Al iniciar la aplicación el usuario se encuentra con una pantalla similar al "escritorio" de cualquier sistema operativo de Computadora actual que posee iconos para acceder a las opciones del entorno.

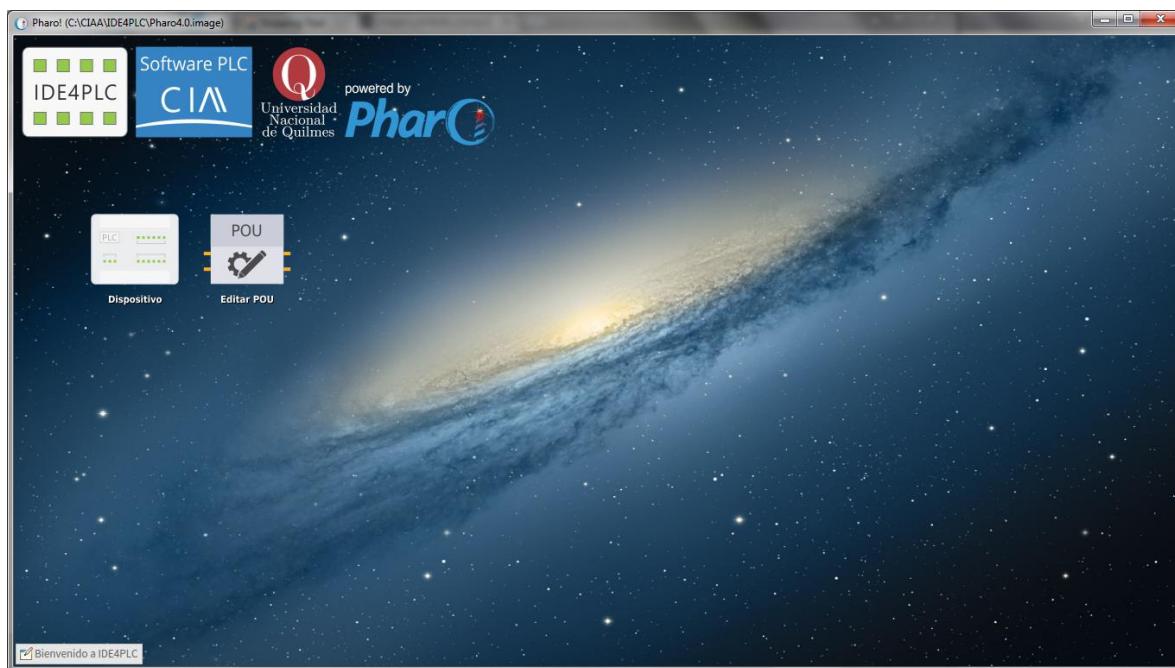


Figura 3-1: Interfaz gráfica

Al ejecutar IDE4PLC se abre la ventana principal con una ventana que brinda información acerca de la licencia de uso de IDE4PLC y permite ejecutar algunos comandos para navegar por los objetos componentes de la aplicación, como muestra la Figura 3-2.

Para ejecutar un comando es suficiente posicionarse a la derecha del punto de la sentencia y pulsar simultáneamente las teclas Alt d, al hacer esto el resultado del comando se presentará a la derecha del punto.

Algunos de los comandos que permite ejecutar se muestran en la siguiente tabla

Comando	Comentario
"Versión del software" IDE4PLC version.	Permite ver la versión de IDE4PLC. Por ejemplo. 'IDE4PLC Versión 1.0.4'
"Versión de CIAA-Firmware compatible con la versión actual de IDE4PLC" IDE4PLC ciaaFirmwareVersion.	Permite ver la versión de CIAA-Firmware que es compatible con la versión actual de IDE4PLC. Por ejemplo. 'Firmware Upa 1.0.0 LTS (release date 2015.12.23).'
"Modo desarrollador." IDE4PLC showDevelopTools.	Permite activar el menú contextual (que aparece cuando se pulsa el botón derecho de mouse en la pantalla principal del IDE4PLC) para desarrolladores.
"Modo usuario." IDE4PLC hideDevelopTools.	Permite desactivar el menú contextual (que aparece cuando se pulsa el botón derecho de mouse en la pantalla principal del IDE4PLC) para desarrolladores.
"Grabar la imagen de Pharo en el estado actual." IDE4PLC save.	Permite grabar la imagen de IDE4PLC en el estado actual. También se puede grabar desde el editor POU con el icono "Guardar POU".
"Inicializar el soft IDE4PLC." IDE4PLC initializeIDE4PLC.	Permite inicializar IDE4PLC, la POU y las variables. También reorganiza los iconos en caso de que se hayan movido.



No se recomienda cerrar la ventana de Bienvenida (con x) y grabar la aplicación ya que si alguna vez necesita de la misma no podrá volver a abrirla. Se recomienda minimizarla (con _) y continuar trabajando con la aplicación.

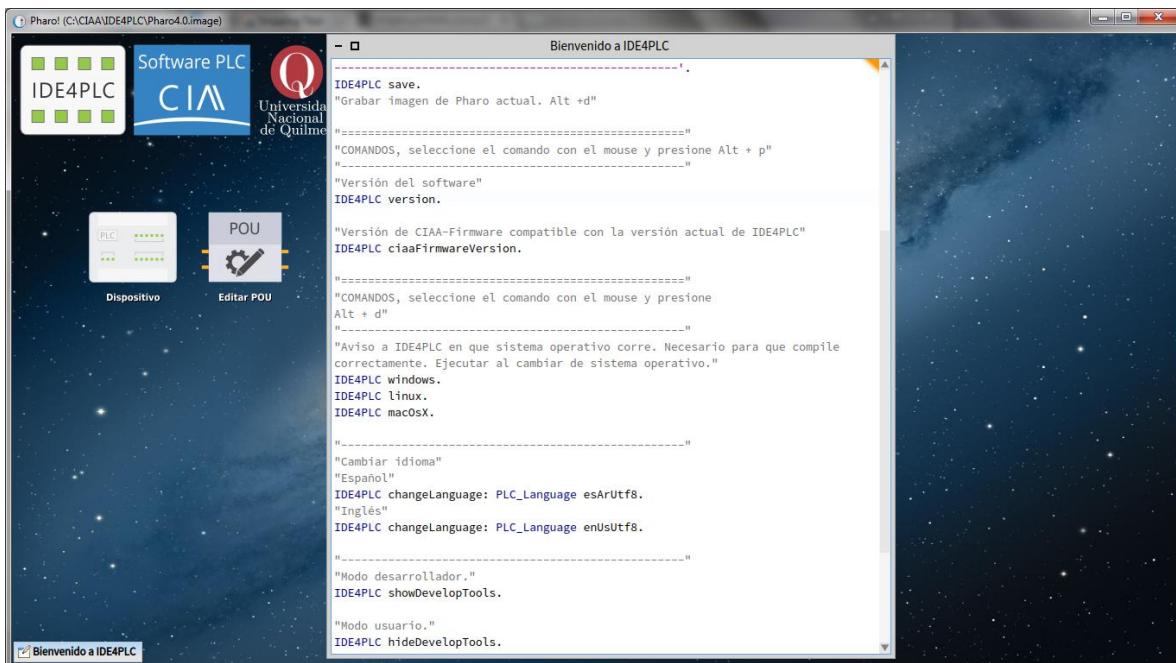


Figura 3-2: Ventana de bienvenida de IDE4PLC

Los iconos principales corresponden a "Dispositivo" y a "Editar POU".



El icono "Dispositivo" abre una ventana donde se puede seleccionar entre una placa EDU-CIAA-NXP y una placa CIAA-NXP, así como ver la descripción de la placa con sus entradas, salidas, puertos de comunicaciones y conectores de expansión.

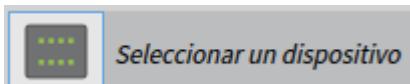


El icono “Editar POU” abre una ventana de “Editor de Unidad de Organización de Programa (POU)” donde se presentan todas las herramientas necesarias para crear un programa Ladder⁷.

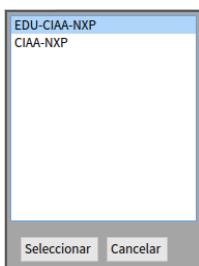
3.2.1 Ventana Board (Selección de tarjeta a usar)

Esta ventana permite seleccionar la placa a usar (Entre EDU-CIAA-NXP y CIAAA-NXP).

Pulsando sobre el icono “Seleccionar un dispositivo” se muestra una ventana en donde se puede seleccionar entre EDU-CIAA-NXP y CIAAA-NXP.



Icono de selección de dispositivo



Ventana de selección de placa. Seleccione la placa en esta pantalla y pulse el botón “Seleccionar”, o pulse “Cancelar” para cancelar la selección y mantener la misma placa.

Al seleccionar una nueva placa IDE4PLC cambiará la lista de variables pre-definidas y todos los programas se compilarán para la placa seleccionada.



Para seleccionar una placa distinta antes debe cerrar la ventana “Editor de Unidad de Organización de Programa (POU)”.

3.2.1.1 Placa EDU-CIAA-NXP

Una vez seleccionada la placa EDU-CIAA-NXP se muestra la descripción de la placa EDU-CIAA. En la misma se puede ver:

En la placa:

- 2 puertos micro USB : uno para aplicaciones y debugging identificado como “DEBUG” y otro para alimentación identificado como “USB OTG”.
- 6 salidas digitales:
 - 3 mediante un led RGB, identificada en la placa como RGB. En el programa Ladder se puede hacer referencia a estas salidas como LEDR (led rojo), LEDG (led verde) y LEDB (led azul).
 - 3 mediante leds normales, identificadas en la placa como LED 1, LED 2 y LED 3. En el Ladder se puede hacer referencia a estas salidas como LED1, LED2 y LED3.
- 4 entradas digitales con pulsadores: identificadas como TEC 1, TEC 2, TEC 3 y TEC 4 en la placa. En el programa Ladder se puede hacer referencia a estas entradas como TEC1, TEC2, TEC3 y TEC4.
- Un puerto de comunicaciones RS-485 con bornera: identificado en la placa como RS485.

En el Conector de expansión CONO:

- 3 Entradas analógicas (AD0 a AD2)
- 1 salida analógica (DAC0)
- 1 conexión para un teclado 3x4
- 12 pines genéricos de I/O

⁷ Ladder: diagrama escalera para programación de PLC. Es uno de los lenguajes de programación de la norma IEC 61131-3.

En el Conector de expansión CON1:

- 1 Puerto Ethernet
- 1 puerto CAN
- 1 puerto SPI
- 1 puerto I2C
- 12 pinos genéricos de I/O



Figura 3-3: ventana Board EDU-CIAA-NXP

3.2.1.2 Placa CIAA-NXP

Una vez seleccionada la placa CIAA-NXP se muestra la descripción de la placa CIAA. En la misma se puede ver:

CPU:

- Microcontrolador NXP LPC 4337 JDB 144 (Dual-core Cortex-M4 + Cortex-M0 @ 204 MHz).
- Memoria SDRAM IS42S16400F. 64Mbit @ 143MHz.

Debugger:

- USB-to-JTAG FT2232H. Soportado por OpenOCD.
- Memoria EEPROM (utilizada por el FT2232H) AT93C46DN.
- Conector Cortex-Debug para Debugger externo.
- Buffer TXB0108 para desconectar el FT2232H del bus JTAG en caso que se use Debugger externo.

Memorias:

- Memorias internas del LPC4337.
- SDRAM 128 Mbit (IS42S16800F-7TL o compatible).
- Flash SPI, 32 Mbit, Quad I/O FAST_READ: 80 MHz clock rate or 40 MB/s effective data rate.
- EEPROM I2C 1 Mbit, 400 kHz. Almacenamiento de propósito general, datos de calibración del usuario, etc. 24AA1025.
- EEPROM I2C 2 kbit, 400 kHz. Unique Node Adress EUI-48 para implementación de MAC-Address. Almacenamiento de propósito general. 24AA025E48.

Entradas digitales:

- 8 entradas digitales optoaisladas.
- Rango de tensión de entrada asegurado 5VDC a 30VDC
- LED testigo de cada entradas.
- Corriente de entrada máxima asegurada de 10 mA, por regulador de corriente constante.
- Protección contra polaridad invertida o tensiones negativas.
- Protección contra transitorios.

Entradas analógicas:

- 4 Entradas analógicas configurables por Corriente/Tensión.
- Lazo de corriente 4-20 mA (con rango extendido 0-22 mA), impedancia de carga 237 Ohm.
- Control de tensión 0-10V, impedancia de entrada 45 KOhm.
- Protección para transitorios.

- Protección para filtrado de alta frecuencias.
- Protección por diodos de enclavamiento.
- Amplificador-Buffer, estable y Rail-to-Rail.
- Optimizado para aprovechar el rango del ADC (10 Bits).

Salidas Open-Drain

- 4 salidas Open-Drain.
- Tensión de Ruptura >24V, Corriente max. > 0.5 A.
- LED testigo en cada salida.
- Protección contra sobre-corriente de fuente.
- Protección contra sobre-corriente en transistor.

Salidas a Relé:

- 4 Salidas en Relay DPDT.
- Seleccionables NA/NC por 4 borneras, mediante switch en PCB.
- Capacidad de corriente para carga resistiva: 0,50A @ 125 VAC, 2A @ 30 VDC.
- LED testigo en cada salida.
- Protección contra sobre-corriente de fuente.

Salidas Analógicas:

- 1 salida analógica configurable como Corriente/tensión.
- Salida de tensión 0-10V (extendido a 10,5V) con impedancia de carga mínima de 3KOhm.
- Transmisor Activo para Lazo de corriente 4-20 mA (con rango extendido 0-22 mA), impedancia max 1 KOhm.
- Protección contra cortocircuito para salida de tensión.
- Protección por diodo de enclavamiento.
- Optimización para aprovechar el rango del DAC (110 Bits, 400 kSamples/s).

LV-GPIO:

- 14 GPIOs.
- I2C.
- SPI.
- 4 Alaog.
- Aux USB.

Interfaces de comunicaciones:

- Ethernet.
- USB On-The-Go.
- RS232.
- RS485.
- CAN.

Fuente de alimentación:

- Rango de tensión de entrada típico: 12VDC a 30 VDC.
- Fuente de Switching para reducción a 5V-3 A.
- Fuente lineal de 3.3V-1 A a partir de los 5V.
- Fuente filtrada de 3.3V para conversor ADC.
- Protección de polaridad en la entrada.
- Protección de sobre-corriente.
- Protección para transitorios.
- Protección por sobre-temperatura.
- LED testigo de encendido.
- Alimentación de gran parte de sus funcionalidades USB.

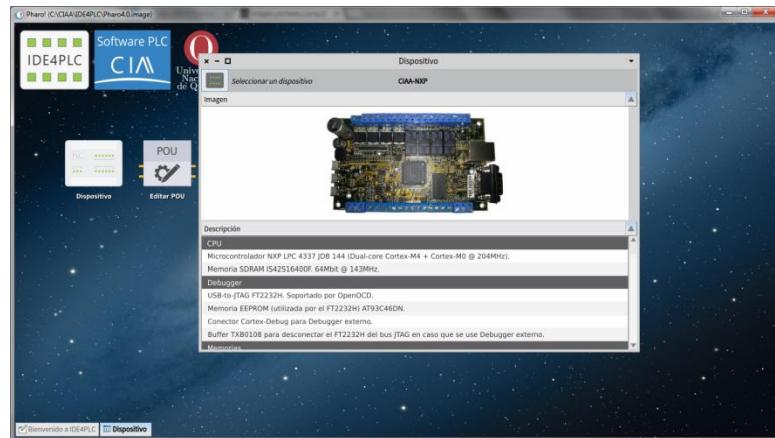


Figura 3-4: ventana Board EDU-CIAA-NXP

3.2.2 Editor de Unidad de Organización de Programa (POU)

La unidad de Organización de Programa es el lugar donde se desarrolla gráficamente el programa en lenguaje Ladder.

El programa IDE4PLC para EDU-CIAA solo permite definir una POU.

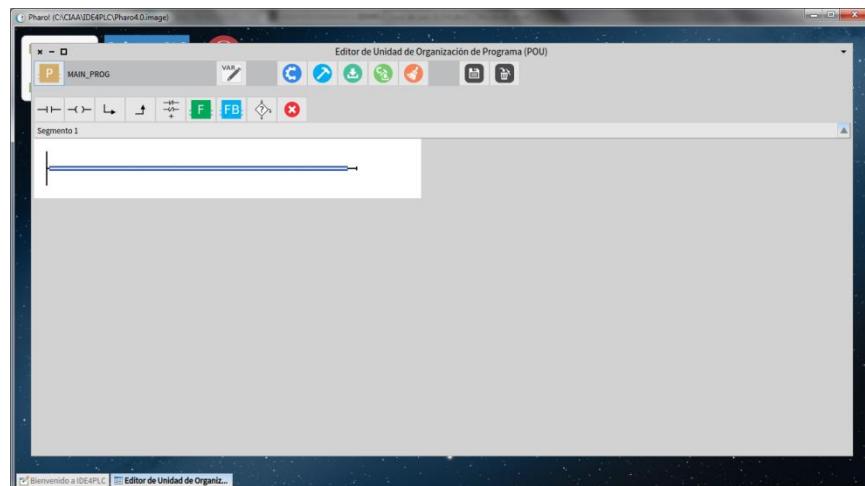


Figura 3-5: Ventana Editor de Organizador de Programa (POU)

En esta ventana se pueden ver varios iconos que permiten desarrollar el programa Ladder y un área de trabajo en donde se puede desarrollar el programa del PLC.

En la primera línea se encuentran los siguientes iconos:



Figura 3-6: Primer línea de iconos de la ventana de editor POU

De izquierda a derechas se pueden ver los siguientes iconos que corresponden a "Abrir declaración de variables", "Generar código C", "Compilar código C", "Descargar programa", "Generar código C, compilar y descargar", "Limpiar compilación de código C", "Grabar POU" y "Eliminar POU". A continuación se describe cada uno de ellos.

3.2.2.1 Declaración de variables



Abrir declaraciones de variables. Este ícono permite abrir una ventana “Editor de Declaraciones Variables de POU” en donde es posible ver las variables definidas con sus tipos, eliminar variables y crear otras.

Para cada tipo de placa hay una lista de variables predefinidas que corresponden a sus entradas y salidas.

Editor de Declaraciones de Variables de POU				
MAIN_PROG				
Variables de interfaz				
Categoría	Nombre	Tipo	Valor inicial	Descripción
VAR_INPUT	I0	BOOL	BOOL#false	Entrada 0 CIAA-NXP.
VAR_INPUT	I1	BOOL	BOOL#false	Entrada 1 CIAA-NXP.
VAR_INPUT	I2	BOOL	BOOL#false	Entrada 2 CIAA-NXP.
VAR_INPUT	I3	BOOL	BOOL#false	Entrada 3 CIAA-NXP.
VAR_INPUT	I4	BOOL	BOOL#false	Entrada 4 CIAA-NXP.
VAR_INPUT	I5	BOOL	BOOL#false	Entrada 5 CIAA-NXP.
VAR_INPUT	I6	BOOL	BOOL#false	Entrada 6 CIAA-NXP.
VAR_INPUT	I7	BOOL	BOOL#false	Entrada 7 CIAA-NXP.
VAR_OUTPUT	Q0	BOOL	BOOL#false	Salida 0 CIAA-NXP.
VAR_OUTPUT	Q1	BOOL	BOOL#false	Salida 1 CIAA-NXP.
VAR_OUTPUT	Q2	BOOL	BOOL#false	Salida 2 CIAA-NXP.
VAR_OUTPUT	Q3	BOOL	BOOL#false	Salida 3 CIAA-NXP.
VAR_OUTPUT	Q4	BOOL	BOOL#false	Salida 4 CIAA-NXP.
VAR_OUTPUT	Q5	BOOL	BOOL#false	Salida 5 CIAA-NXP.
VAR_OUTPUT	Q6	BOOL	BOOL#false	Salida 6 CIAA-NXP.

Variables Internas y Externas				
Categoría	Nombre	Tipo	Valor inicial	Descripción
VAR	var0	BOOL	BOOL#false	
VAR	var1	INT	INT#0	
VAR	var2	TIME	TIME#0	
VAR_TEMP	temp0	BOOL	BOOL#false	
VAR_TEMP	temp1	INT	INT#0	
VAR_TEMP	temp2	TIME	TIME#0	

Figura 3-7: Editor de Declaraciones de Variables de POU (para placa CIAA)

Editor de Declaraciones de Variables de POU				
MAIN_PROG				
Variables de interfaz				
Categoría	Nombre	Tipo	Valor inicial	Descripción
VAR_INPUT	TEC1	BOOL	BOOL#false	Botón 1 EDU-CIAA-NXP
VAR_INPUT	TEC2	BOOL	BOOL#false	Botón 2 EDU-CIAA-NXP
VAR_INPUT	TEC3	BOOL	BOOL#false	Botón 3 EDU-CIAA-NXP
VAR_INPUT	TEC4	BOOL	BOOL#false	Botón 4 EDU-CIAA-NXP
VAR_OUTPUT	LEDR	BOOL	BOOL#false	Led Rojo EDU-CIAA-NXP
VAR_OUTPUT	LEDG	BOOL	BOOL#false	Led Verde EDU-CIAA-NXP
VAR_OUTPUT	LEDB	BOOL	BOOL#false	Led Azul EDU-CIAA-NXP
VAR_OUTPUT	LED1	BOOL	BOOL#false	Led 1 EDU-CIAA-NXP
VAR_OUTPUT	LED2	BOOL	BOOL#false	Led 2 EDU-CIAA-NXP
VAR_OUTPUT	LED3	BOOL	BOOL#false	Led 3 EDU-CIAA-NXP

Variables Internas y Externas				
Categoría	Nombre	Tipo	Valor inicial	Descripción
VAR	var0	BOOL	BOOL#false	
VAR	var1	INT	INT#0	
VAR	var2	TIME	TIME#0	
VAR_TEMP	temp0	BOOL	BOOL#false	
VAR_TEMP	temp1	INT	INT#0	
VAR_TEMP	temp2	TIME	TIME#0	

Figura 3-8: Editor de Declaraciones de Variables de POU (para placa EDU-CIAA)

Esta ventana permite ver, editar, agregar y eliminar variables que podrán ser usadas en un programa Ladder. Existen algunas variables pre-definidas que corresponden a la interfaz de la placa EDU-CIAA. Estas variables de interfaz no pueden ser modificadas ni eliminadas:

- TEC1 a TEC3: Estas variables de entrada del tipo BOOL corresponden a las cuatro teclas incorporadas en la placa EDU-CIAA.
- LEDR, LEDG, LEDB y LED1 A LED3: Estas variables de salida del tipo BOOL corresponden al led RGB y a los 3 leds normales incorporados en la placa EDU-CIAA.

Las variables Internas y externas pueden ser modificadas, eliminadas y se pueden agregar nuevas. Son variables genéricas que no corresponden a variables que tienen una relación directa con las entradas o salidas de la placa. Pueden ser utilizadas internamente en el programa Ladder para realizar operaciones, establecer parámetros de bloques funcionales, mantener valores auxiliares, etc.

Al agregar algunos bloques de función se agregan variables que dichos bloques necesitan para operar correctamente. Los nombres de estas variables están en mayúscula, comienzan con el mismo nombre de la función que les dio origen y terminan con “_VAR_nn”, donde nn es un número correlativo para evitar la repetición de los nombres de variables, como muestra la Figura 3-9.

VAR	TP_TIME_VAR_31	TYPEPL	(BOOL#fal)
VAR	TP_TIME_VAR_32	TYPEPL	(BOOL#fal)
VAR	R_TRIG_BOOL_VAR_33	TYPEPL	(BOOL#fal)
VAR	SR_BOOL_VAR_34	TYPEPL	(BOOL#fal)
VAR	F_TRIG_BOOL_VAR_36	TYPEPL	(BOOL#fal)
VAR	TP_TIME_VAR_38	TYPEPL	(BOOL#fal)

Figura 3-9: Variables para Bloques de Función

3.2.2.1.1 Agregar nuevas variables



Añadir declaración: Este botón permite agregar nuevas variables.

Al pulsar este botón se abre la ventana de la Figura 3-10 para permitir declarar una nueva variable. Para crear una nueva variable complete los campos de la ventana y pulse Aceptar.

Categoría de variables: Las variables pueden ser

- VAR: Las variables de esta categoría son persistentes entre ejecuciones de la POU. Es decir que mantienen su valor entre una ejecución y la próxima.
- VAR_TEMP: las variables de esta categoría son locales y no mantienen su valor entre ejecución de la PAU.

Nombre: El nombre identifica a la variable. No puede tener el mismo nombre que una variable existente. Los nombres de variables no pueden tener espacios ni comenzar con número. Las variables son “case sensitive”, es decir que son distintas mayúsculas y minúsculas.

Tipo de datos: Pueden ser del tipo BOOL (Booleano), INT (Entero) o TIME (Duración). Para mayor detalle referente a tipos de datos ver 4.1.1.

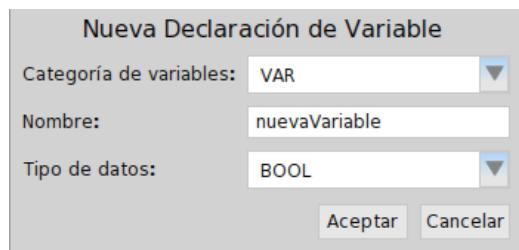


Figura 3-10: Ventana de Nueva declaración de variable

3.2.2.1.1.1 Variables del tipo VAR

Las variables del tipo VAR son variables persistentes, es decir que su valor se mantiene y no es inicializado cuando comienza a ejecutarse la POU (como ocurre con las variables del tipo VAR_TEMP). En el siguiente ejemplo se utiliza la variable var0 que es del tipo VAR.

En este circuito, mientras se mantenga pulsado TEC1 se mantendrá encendido el LED 1. Cuando el programa comienza a ejecutar LED1 está apagado, al pulsar TEC1 la variable var0 toma el valor true, cuando la POU vuelve a ejecutarse var0 tiene el valor true y por lo tanto LED1 se enciende. Cuando se deja de pulsar TEC1, var0 toma el valor false y cuando la POU vuelve a ejecutarse LED1 se apaga.

Categoría	Nombre	Tipo	Valor inicial	Descripción
VAR	var0	BOOL	BOOL#false	
VAR	var1	INT	INT#0	
VAR	var2	TIME	TIME#0	
VAR_TEMP	temp0	BOOL	BOOL#false	
VAR_TEMP	temp1	INT	INT#0	
VAR_TEMP	temp2	TIME	TIME#0	

Figura 3-11: Definición de variable del tipo VAR

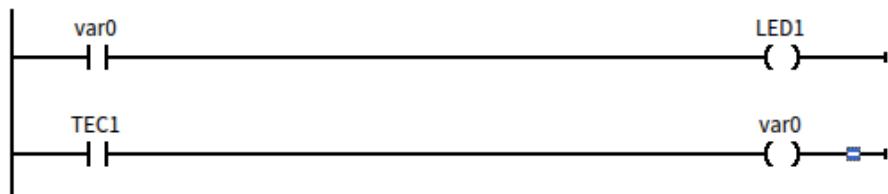


Figura 3-12: Circuito con variable del tipo VAR

3.2.2.1.1.2 Variables del tipo VAR_TEMP

Las variables del tipo VAR_TEMP son variables locales, es decir que su valor no se mantiene, se inicializan cuando comienza a ejecutarse la POU y no se mantienen (como ocurre con las variables del tipo VAR). En el siguiente ejemplo se utiliza la variable temp0 que es del tipo VAR_TEMP.

En este circuito al contrario del anterior, el LED1 no se enciende aunque se pulse TEC1. Cuando el programa comienza a ejecutar LED1 está apagado, al pulsar TEC1 la variable temp0 toma el valor true, cuando la POU vuelve a ejecutarse temp0 se inicializa en false y por lo tanto el LED1 no se enciende nunca.

Categoría	Nombre	Tipo	Valor inicial	Descripción
VAR	var0	BOOL	BOOL#false	
VAR	var1	INT	INT#0	
VAR	var2	TIME	TIME#0	
VAR_TEMP	temp0	BOOL	BOOL#false	
VAR_TEMP	temp1	INT	INT#0	
VAR_TEMP	temp2	TIME	TIME#0	

Figura 3-13: Definición de variable del tipo VAR_TEMP

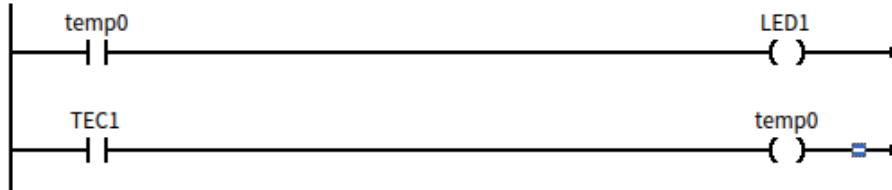


Figura 3-14: Circuito con variable del tipo VAR_TEMP

3.2.2.1.2 Eliminar variables



Remover declaración: Este botón permite eliminar una variable.

 Para eliminar una variable debe seleccionarla, pulsando el botón del mouse sobre la variable a eliminar, y pulsar el botón de Remover Declaración. La variable será eliminada y ya no aparecerá en la tabla de variables.

Las variables de Interfaz no pueden ser eliminadas. Si se intenta eliminar una Variable de Interfaz se presentará un mensaje de error indicando que la variable no puede ser eliminada.

3.2.2.2 Generar código C



Generar Código C: Este botón permite generar el código C correspondiente al programa Ladder creado en el área de programa.

Al pulsar este botón, si no se presenta ningún error, se genera el código C en el directorio .../IDE4PLC/IDE4PLC_user_projects/plc_application en el directorio donde fue instalada la aplicación (por ejemplo C:/CIAA/IDE4PLC/IDE4PLC_user_projects/plc_application) . Si el código C pudo generarse correctamente se presentará un mensaje como el siguiente.

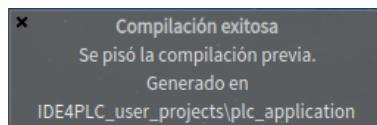


Figura 3-15: Mensaje Compilación exitosa

3.2.2.3 *Compilar Código C*



Compilar Código C: Este botón permite compilar el código C correspondiente al código generado con el botón “generar Código C”.

Este botón ejecuta el comando "make" que compila el código C.

Esta opción es muy útil para detectar la causa de un problema. Si, al compilar un circuito Ladder, se produce un error, las ventanas negras no se cerrarán, permitiendo verificar la causa del problema.

Al pulsar este botón se abrirán dos ventanas negras con texto similares a las siguientes. Si el proceso de Compilación concluye correctamente ambas ventanas se cerrarán automáticamente, si hay algún error en alguno de estos procesos las ventanas permanecerán abiertas permitiendo que se pueda analizar el error y deberá pulsar <Enter> para cerrarlas.

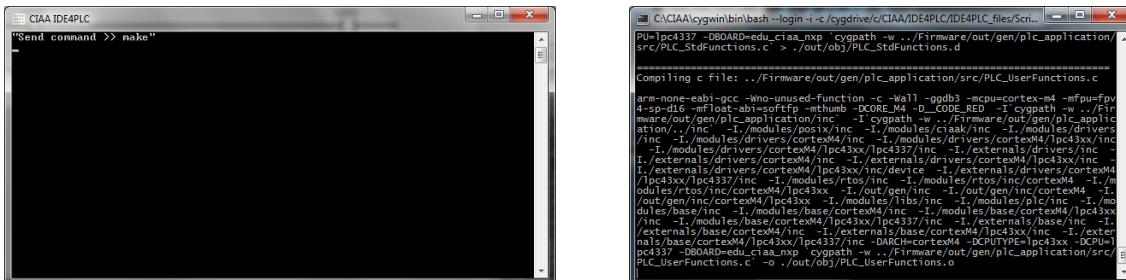


Figura 3-16: Ventanas de compilación

3.2.2.4 Descargar Programa



Descargar Programa: Este botón descarga el programa compilado al dispositivo seleccionado (EDU-CIAA-NXP o CIAA-NXP).

Este botón ejecuta el comando "make download" que descarga el código ejecutable a la memoria flash del microcontrolador. Al pulsar este botón se abrirán dos ventanas negras con texto similares a las siguientes. Si

el proceso de descarga del programa concluye correctamente ambas ventanas se cerrarán automáticamente, si hay algún error en alguno de estos procesos las ventanas permanecerán abiertas permitiendo que se pueda analizar el error y deberá pulsar <Enter> para cerrarlas.

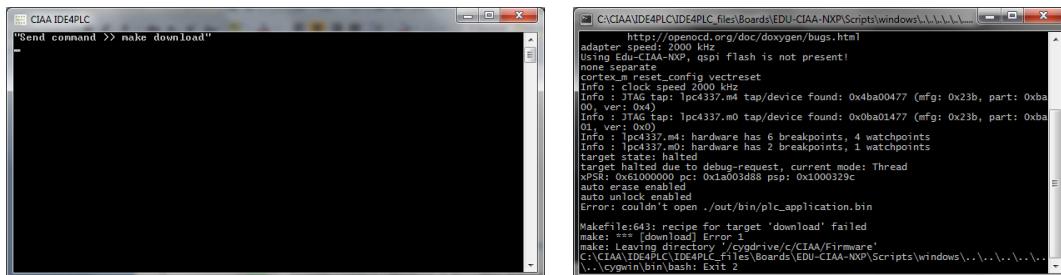


Figura 3-17: Ventanas de descarga de programa

3.2.2.5 Generar código C, Compilar y Descargar al dispositivo



Generar código C, Compilar y Descargar: Este botón permite generar el código C correspondiente al programa Ladder creado en el área de programa y descargarlo al dispositivo seleccionado (EDU-CIAA-NXP o CIAA-NXP).

Esta opción permite, con un solo click, realizar la generación del código C, la compilación y la descarga del programa compilado al dispositivo. Al pulsar este botón se abrirán dos ventanas negras con texto similares a las siguientes. Si el proceso de generación de código C, compilación y descarga al dispositivo (EDU-CIAA-NXP o CIAA-NXP) concluyó correctamente ambas ventanas se cerrarán automáticamente, si hay algún error en alguno de estos procesos las ventanas permanecerán abiertas permitiendo que se pueda analizar el error y deberá pulsar <Enter> para cerrarlas.

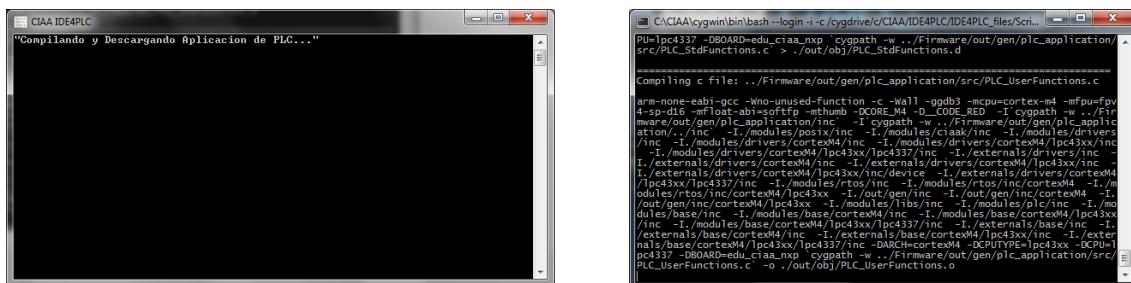


Figura 3-18: Ventanas de compilación

3.2.2.6 Limpiar Compilación de Código C



Limpiar Compilación de Código C: Este botón permite borrar completamente el directorio donde se encuentran los archivos C.

Este botón permite ejecutar el comando "make clean_generate" que borra los archivos generados por la compilación de C previa.

Al pulsar este botón se abrirán dos ventanas negras con texto similares a las siguientes.

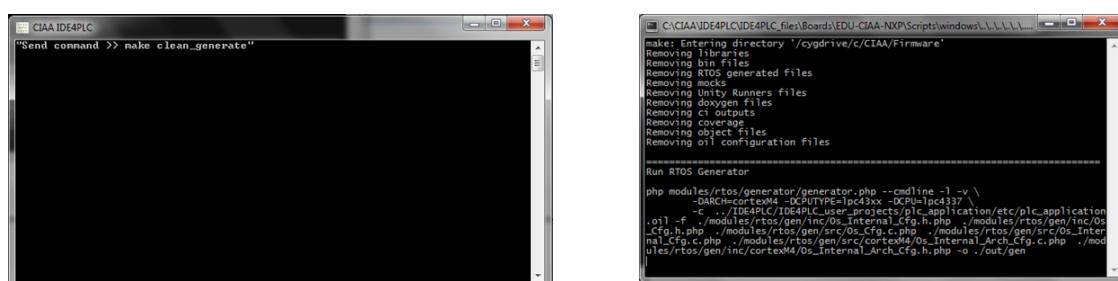


Figura 3-19: Ventanas de Limpieza de código C

3.2.2.7 Guardar POU



Guardar POU: Este botón permite guardar el programa generado y el entorno de programación.

Al pulsar el botón Guardar POU todo el entorno del IDE4PLC se guardará con el nombre Pharo4.0.image y cuando vuelva a ejecutar el programa todo estará de la misma manera que lo dejó.

Si se desea guardar el programa Ladder y seguir trabajando con otro se puede hacer una copia de los archivos Pharo4.0.image y Pharo4.0.changes con otro nombre, por ejemplo Pharo4.0.AAAA.image y Pharo4.0.AAAA.changes, donde AAAA es un nombre que identifica su programa Ladder.

El siguiente programa batch de Windows permite hacer esta copia.

```
@ECHO OFF  
ECHO -----  
ECHO Este programa hace una copia de los archivos Pharo4.0.image  
ECHO y Pharo4.0.changes con el nombre indicado.  
ECHO -----  
SET /P destino=Ingrese el nombre del archivo destino:  
IF "%destino%"=="%" GOTO Error  
  
ECHO Copiando archivo image...  
copy Pharo4.0.image Pharo4.0.%destino%.image  
  
ECHO Copiando archivo changes...  
copy Pharo4.0.changes Pharo4.0.%destino%.changes  
  
GOTO End  
:Error  
ECHO Debe ingresar un nombre!!  
:End  
ECHO Pulse [Enter] para continuar...  
Pause
```

3.2.2.8 Eliminar POU



Eliminar POU: Este botón permite eliminar todo el programa y las variables agregadas por el usuario.

Al pulsar el botón Eliminar POU el programa Ladder y las variables volverán a estar como lo estuvieron originalmente (las variables creadas ya no estarán y las originales que fueron eliminadas volverán a aparecer), dejando la ventana inicial.

En la Figura 3-20 se puede apreciar cómo se muestra un programa Ladder al que aun no se le ha agregado ningún elemento, o después de ejecutar “Eliminar POU”. Una nueva POU posee por defecto tres elementos gráficos de programa, los cuales son “Barra de alimentación derecha” y “Barra de alimentación izquierda”, que se encuentran unidos por una “Conexión horizontal”.

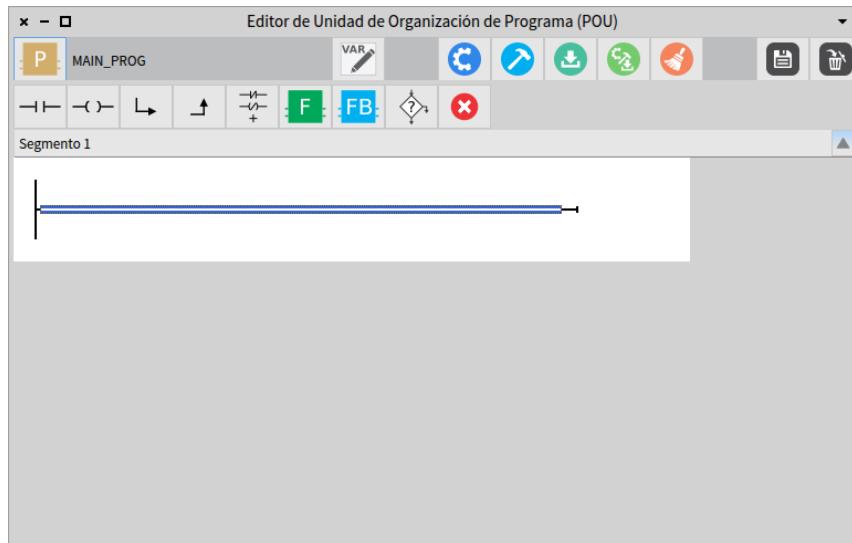


Figura 3-20: ventana cuerpo de POU

3.2.2.8.1 Haciendo un programa Ladder

En la segunda línea se encuentran los siguientes iconos que servirán para desarrollar el programa Ladder:

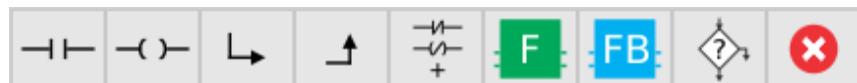


Figura 3-21: Línea de elementos de programa Ladder

De izquierda a derechas se pueden ver los siguientes iconos que corresponden a “Añadir contacto”, “Añadir bobina”, “Abrir rama”, “Cerrar rama”, “Añadir componentes Ladder”, “Añadir llamado a Función”, “Añadir llamado a bloque de Función”, “Añadir componente de Control de Ejecución” y “Remover Componente(s) seleccionado(s)”. A continuación se describirá cada uno de ellos.



Añadir contacto: Este ícono permite agregar un Contacto normal abierto en un programa Ladder.



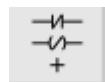
Añadir bobina: Este ícono permite agregar una bobina en un programa Ladder.



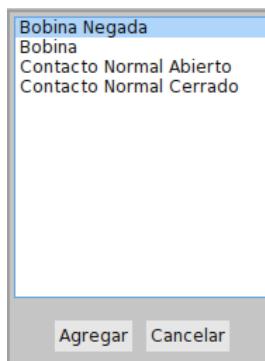
Abrir rama: Este ícono permite abrir una nueva rama de un programa Ladder.



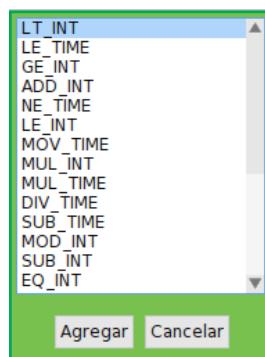
Cerrar rama: Este ícono permite cerrar una nueva rama de un programa Ladder.



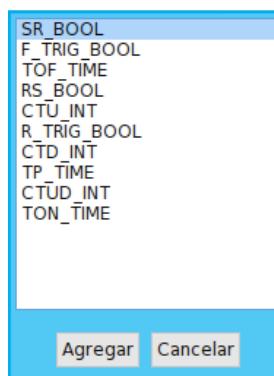
Añadir componentes Ladder: Este ícono presenta un menú desde el cual se puede seleccionar una nueva Bobina, Bobina Negada, Contacto normal Abierto o un Contacto Normal Cerrado para ser agregado a un programa Ladder. Seleccione el componente deseado y pulse Agregar.



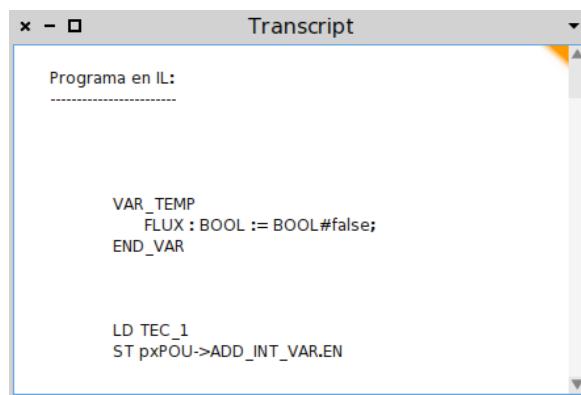
Añadir Llamado a Función: Este ícono presenta un menú desde el cual se puede seleccionar un llamado a Función (como Comparaciones, Funciones Aritméticas, etc.) para ser agregado a un programa Ladder. Seleccione el llamado a Función deseado y pulse Agregar.



Añadir Llamado a Bloque de Función: Este ícono presenta un menú desde el cual se puede seleccionar un Bloque de Función (como Contadores, Temporizadores, etc.) para ser agregado a un programa Ladder. Seleccione el Bloque de Función deseado y pulse Agregar.



Generar y ver código C o IL del segmento: Este ícono permite visualizar el programa Ladder como un programa en lenguaje IL y en lenguaje C, como se muestra a continuación.



```

x - □ Transcript
-----
Programa en IL:
-----
VAR_TEMP
FLUX : BOOL := BOOL#false;
END_VAR

LD TEC_1
ST pxPOU->ADD_INT_VAR.EN

```



Eliminar Seleccionado: Este ícono permite eliminar uno o varios elementos de un programa ladder.

3.2.2.8.1.1 Insertando un nuevo elemento Laddder

El usuario incorpora un elemento (Contacto, Bobina, Llamada a Función, Bloque de Función, etc.) a un segmento seleccionando el punto dentro del circuito definido para el segmento en el que debe insertarse el nuevo elemento, y luego eligiendo el ícono correspondiente al tipo de elemento a agregar (Contacto, Bobina, Llamada a Función, etc.) en la barra de herramientas.

En este momento, el editor de programas inserta el elemento seleccionado, haciendo las reconfiguraciones necesarias en las conexiones entre los elementos que forman parte del segmento, y reacomodando la ubicación de cada elemento en pantalla de acuerdo al nuevo grafo de conectividad. Esto significa que en un segmento en lenguaje Ladder es el entorno quien se encarga de distribuir en pantalla los elementos que componen el circuito.

En la Figura 3-22 se muestran las distintas etapas en el agregado de un Contacto dentro de un segmento.

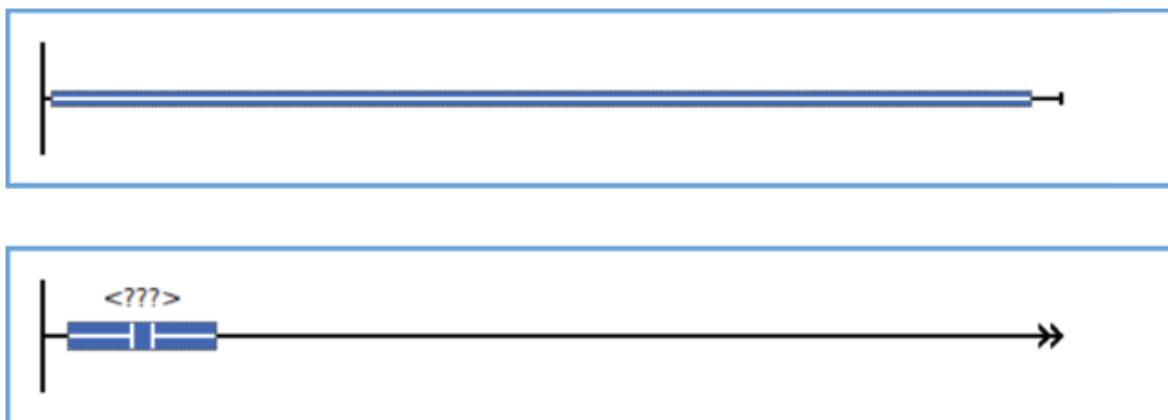


Figura 3-22: Diagrama Ladder. Agregado de un contacto sobre una conexión.

Estas etapas son:

1. Seleccionar el punto de inserción. Este debe ser una conexión entre dos elementos o un pin, como indica la Figura 3-22 parte superior.
2. Elegir elemento a insertar en la barra de herramientas (en este caso un Contacto normal abierto).
3. Se redibuja automáticamente el segmento mostrando el nuevo elemento insertado en el circuito, como indica la Figura 3-22 parte inferior.

Un efecto a destacar es el cambio del dibujo del elemento "Barra de alimentación derecha", indicando que el circuito no puede terminar en un Contacto y, en cambio, debe terminar en Llamada a Función, Llamado a Bloque de Función, Bobina o elemento de Bifurcación de programa. Es por ello que en la Figura 3-22 abajo

se muestra el elemento "Barra de alimentación derecha" con una doble punta de fecha. Esto indica que se producirá un error al compilar esta POU. De modo semejante, la Figura 3-23 expone las distintas etapas en el agregado de un llamado a Función suma ("ADD") dentro de un segmento.

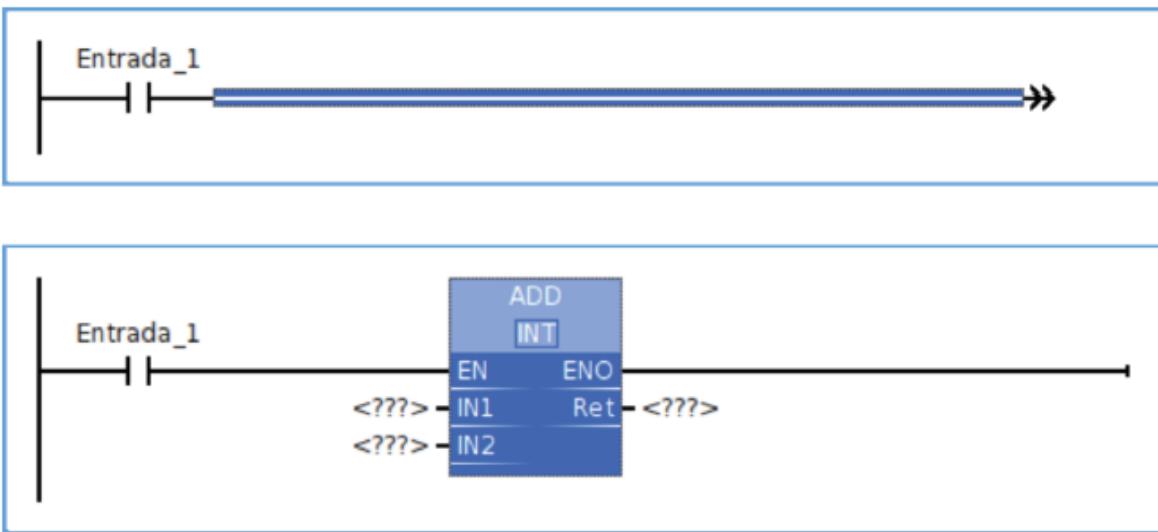


Figura 3-23: Diagrama Ladder. Agregado de un llamado a Función Suma ("ADD") sobre una conexión

3.2.2.8.1.2 Configurando un elemento Ladder

Los argumentos de los elementos pueden configurarse mediante la acción de "doble clic" sobre el mismo. Cada argumento admite como valor un literal o variable de cierto tipo. Las variables además, deben encontrarse en el "alcance" de la POU, esto es, declaradas en la sección de "Declaraciones de variables de POU" descripta anteriormente.

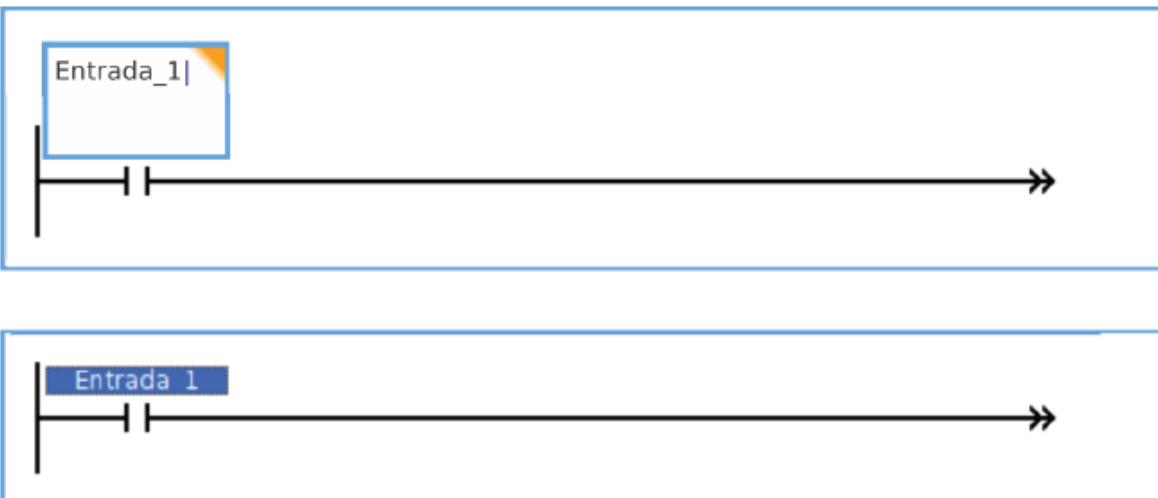


Figura 3-24: Diagrama Ladder. Edición del argumento de un contacto

3.2.2.8.1.3 Editando y eliminando un elemento Laddder

En la Figura 3-25 se expone la secuencia de edición del único argumento que posee un Contacto normal abierto. El editor de programas también permite la eliminación de un elemento en un segmento. Para realizarlo debe seleccionarse el elemento a eliminar, y luego, presionar el botón "Eliminar elemento". Esta secuencia se muestra en la Figura 3-25. En la parte superior de la misma se muestra seleccionado el componente Contacto y en la parte inferior el circuito resultante luego de su eliminación. Nótese como se reacomoda el Contacto que se encontraba conectado en paralelo con el que fue eliminado.

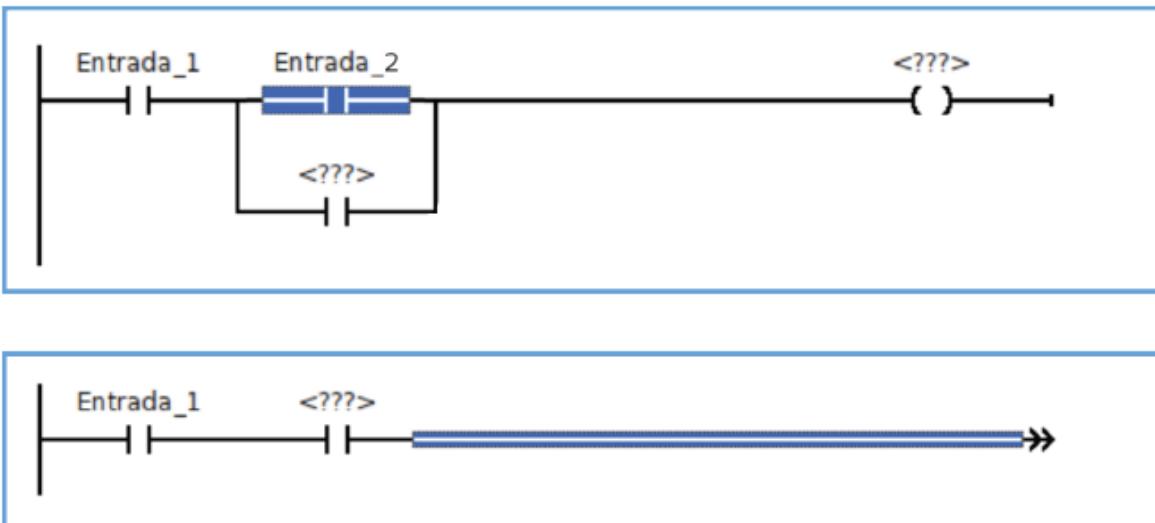


Figura 3-25: Diagrama Ladder. Borrado de contacto

3.2.2.8.1.4 Abriendo y cerrando una rama paralela

En un circuito Ladder se permiten conexiones en serie y paralelo de los elementos. Estas conexiones se realizan mediante los botones de "Apertura de rama paralela" y "Cierre de rama paralela" contenidos en la barra de herramientas. En ambos casos se inserta un componente "Enlace Vertical".

La Figura 3-26 muestra las dos acciones que producen como resultado la apertura de una rama paralela. En la parte superior de esta figura puede observarse la conexión seleccionada; mientras que en la parte inferior de la misma se da el resultado, luego de presionar el botón "Apertura de rama paralela".

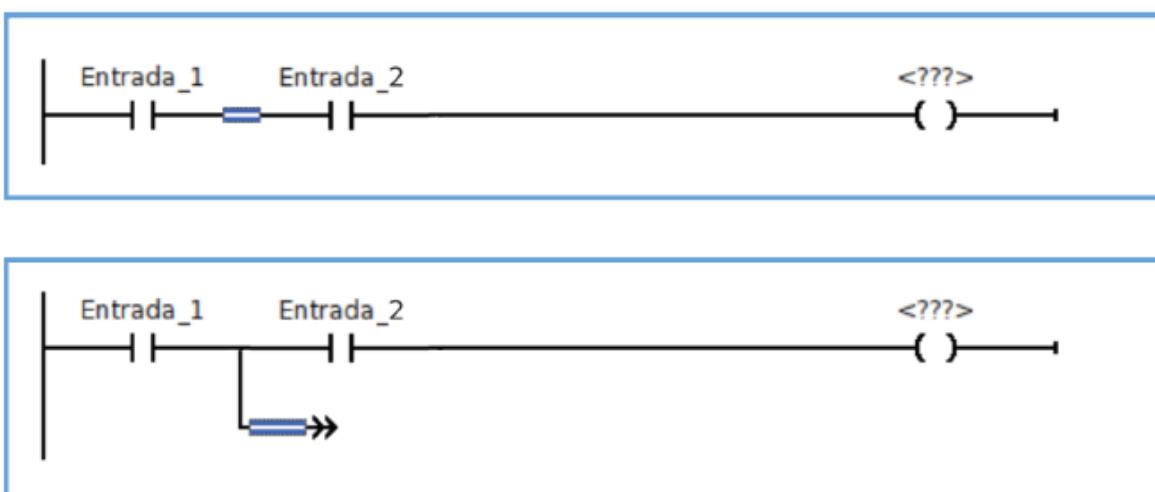


Figura 3-26: Diagrama Ladder. Abrir una rama paralela sobre una conexión

Una rama paralela al igual que la principal debe terminar en un llamado a Función, llamado a Bloque de Función, Bobina o elemento de Control de ejecución de programa. Para añadir un elemento en la misma, se procede de igual forma que en el caso de la rama principal como se expone en la Figura 3-27.

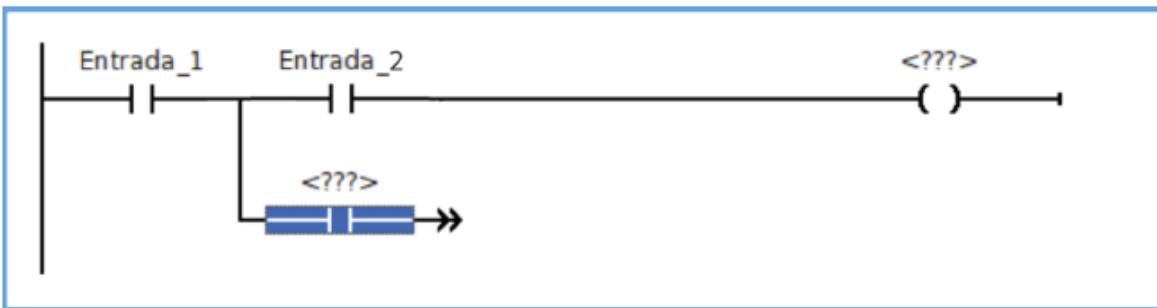


Figura 3-27: Diagrama Ladder. Agregado de un contacto en una rama paralela

Para poder cerrar una rama paralela, creando en consecuencia una conexión en paralelo, es requisito que exista un elemento en la misma para no producir un cortocircuito como el de la Figura 3-28.

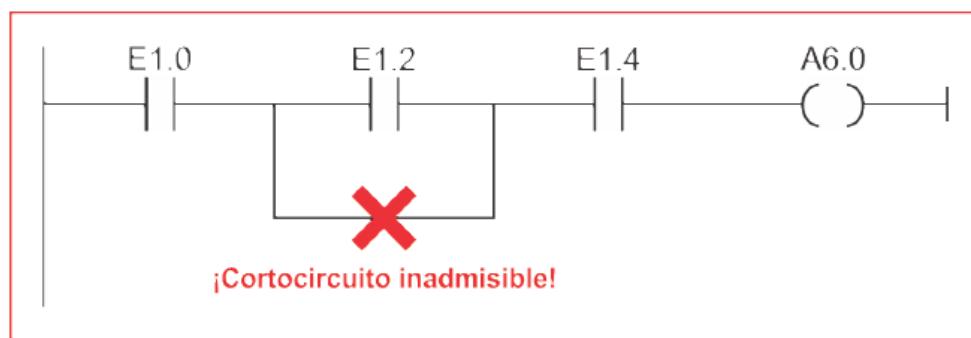


Figura 3-28: Diagrama Ladder. Circuito inválido por cortocircuito

El entorno chequea que exista al menos un elemento en la rama paralela cuando el usuario intenta realizar esta acción. Los pasos para cerrar una rama paralela se ofrecen en la Figura 3-29. Estos son:

1. Seleccionar el componente "Barra de alimentación derecha" que se quiere utilizar como origen para cerrar la rama (Figura 3-29 parte superior).
2. Presionar la tecla shift, que permite selección múltiple de elementos, y mientras se encuentra esta tecla presionada seleccionar una conexión destino hacia dónde se cerrará la rama (Figura 3-29 parte central).
3. Teniendo estos dos elementos seleccionados, presionar el botón "Cierre de rama paralela" de la barra de herramientas.
4. En caso de permitir el cerrado de la rama, el entorno redibuja automáticamente el segmento, mostrando el resultado de la conexión en paralelo entre ambos Contactos (Figura 3-29 parte inferior).

Otra opción consiste en cerrar parcialmente dos ramas seleccionando las conexiones que deben unirse mediante un "Enlace Vertical", creando un cierre de rama paralela utilizando el mismo botón. Se muestra un ejemplo en la Figura 3-30.

Al eliminar una rama paralela el entorno la borra automáticamente al suprimir todos los elementos situados en las mismas.

La norma IEC 61131-3 indica que "en un circuito Ladder la corriente debe circular de izquierda a derecha" tomando como polos positivo y negativo a los componentes "Barra de alimentación izquierda" y "Barra de alimentación derecha" respectivamente. Un ejemplo de circuito inválido se da en la Figura 3-31. Aquí un estado de señal "0" en E1.4 causará un flujo de corriente de derecha a izquierda en E6.8, lo cual no es admisible.

Para cumplir con la condición anterior el entorno solo permite conexiones en serie o paralelo de elementos. Este chequeo es realizado junto con los anteriores al presionar el botón "Cierre de rama paralela".

El entorno no permite crear en pantalla elementos redundantes; por ejemplo en la Figura 3-32 se indica el comportamiento que tendrá el sistema en el caso que el usuario requiera cerrar una rama paralela antes de la apertura de otra rama creada previamente.

Aquí en vez de crear otro elemento "Enlace Vertical" solamente debe conectarse al mismo. Esto puede considerarse como una absorción de un "Enlace Vertical" con respecto al otro formando un solo "Enlace Vertical".

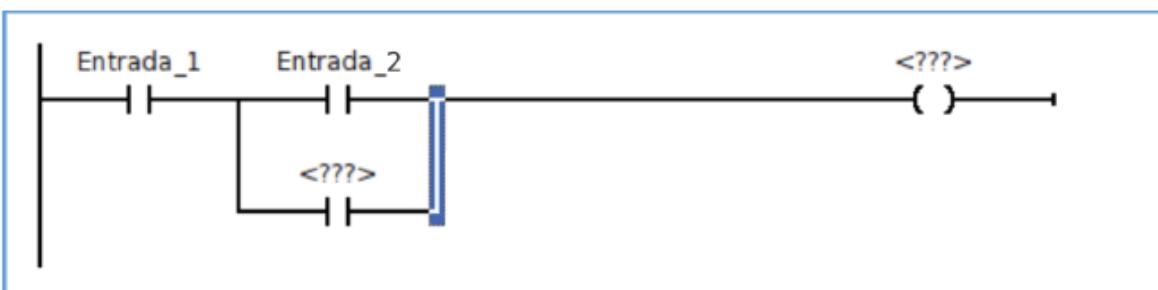
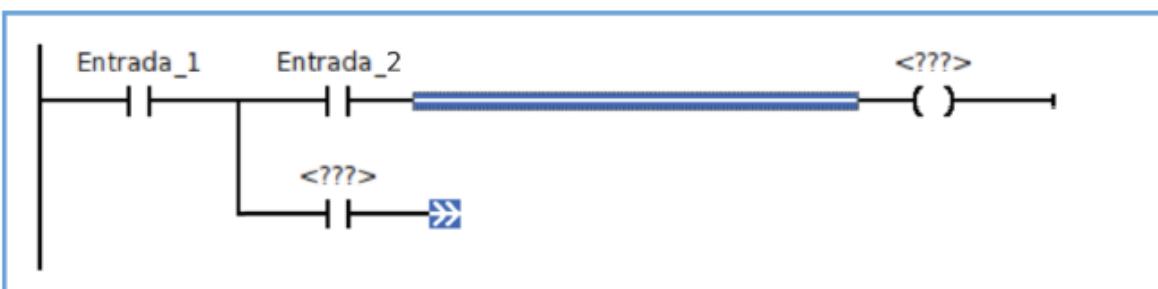
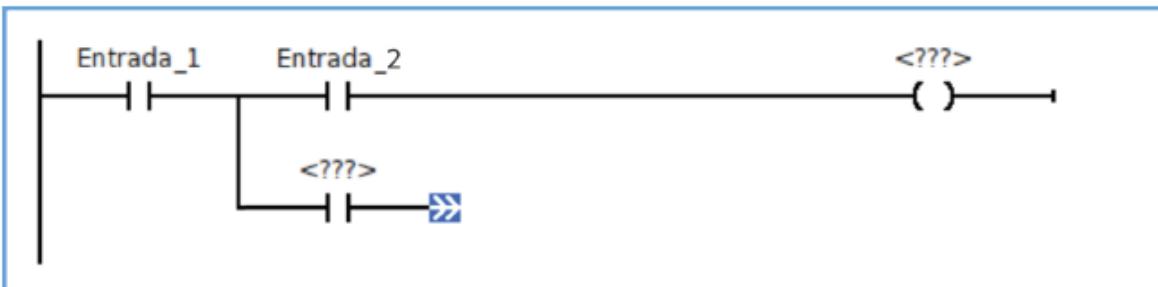


Figura 3-29: Diagrama Ladder. Cerrar una rama paralela para formar un circuito paralelo

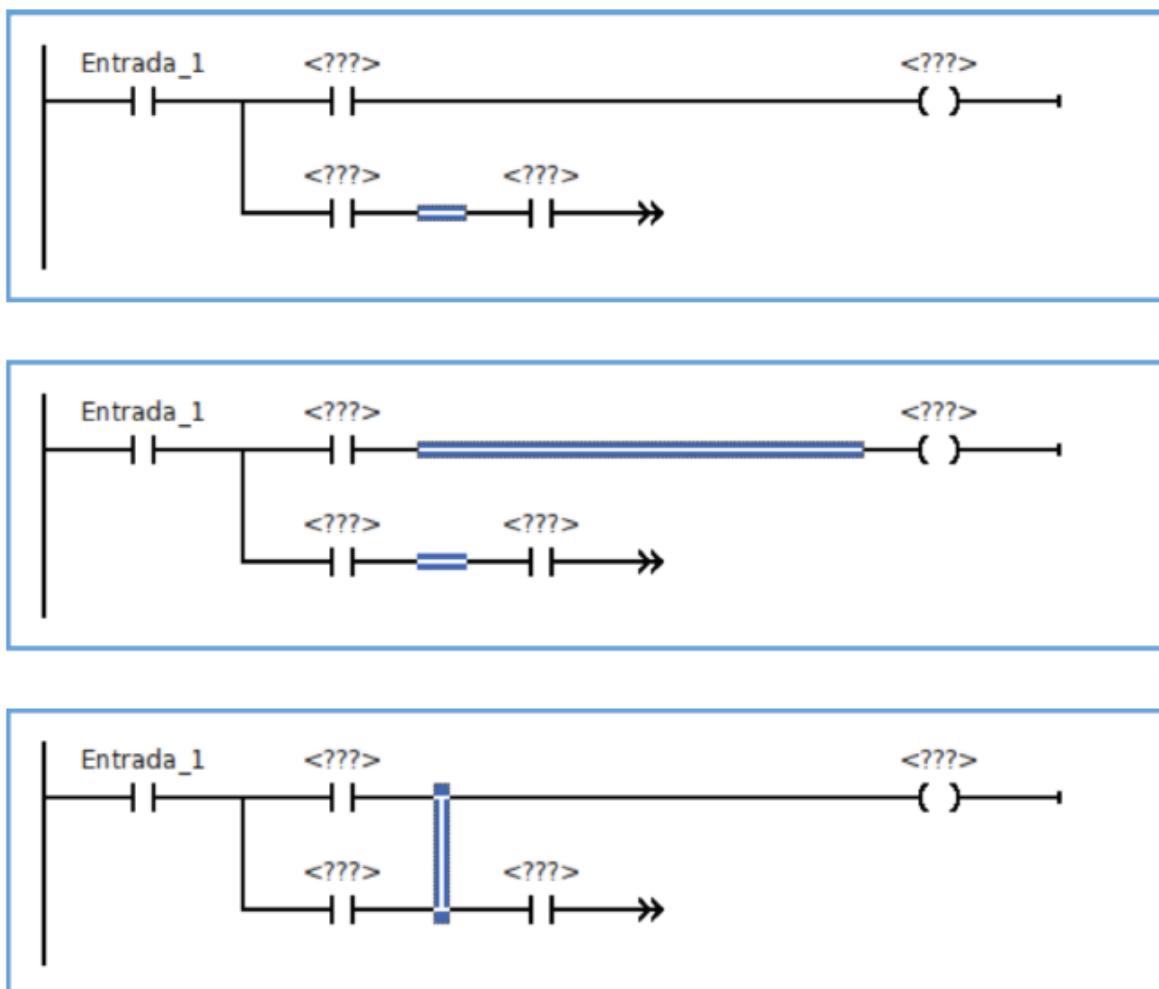


Figura 3-30: Diagrama Ladder. Cerrar rama paralela entre dos conexiones

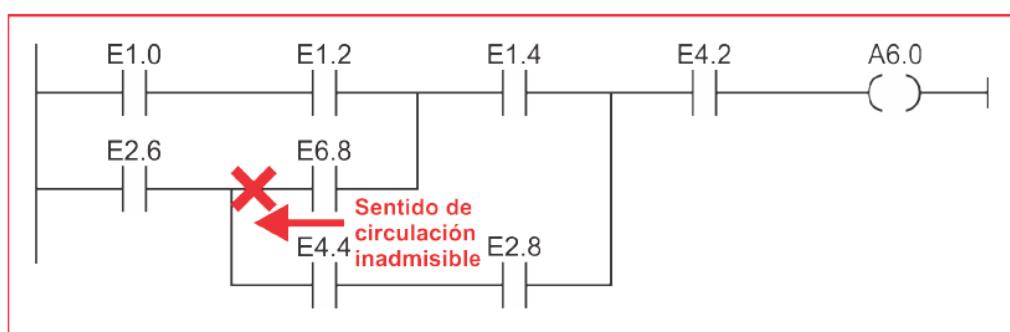


Figura 3-31: Diagrama Ladder. Circuito inválido por flujo de corriente de derecha a izquierda

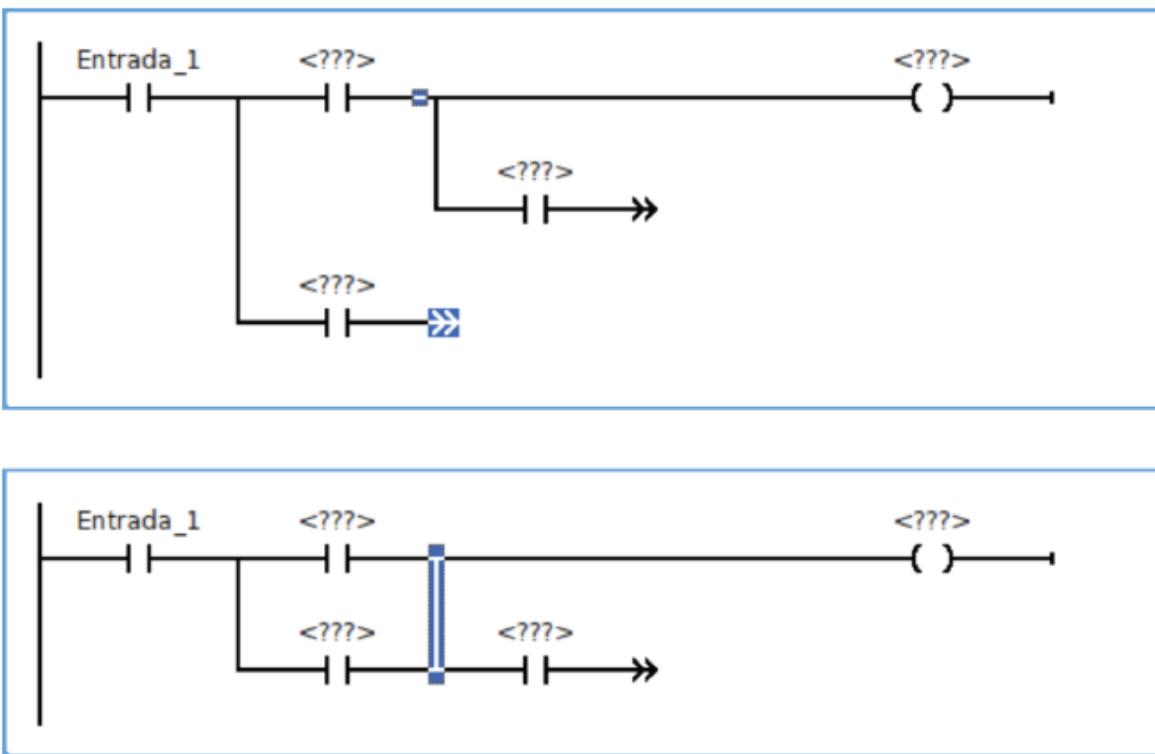


Figura 3-32: Diagrama Ladder. Cerrar una rama antes de una rama abierta.

Finalmente, en la Figura 3-33 se ilustra un ejemplo más complejo de circuito Ladder para indicar el comportamiento de la interfaz. En el circuito se muestran varias aperturas de rama en cascada y además, cómo “empujan” los Contactos al Enlace Vertical para no pisarse en pantalla.

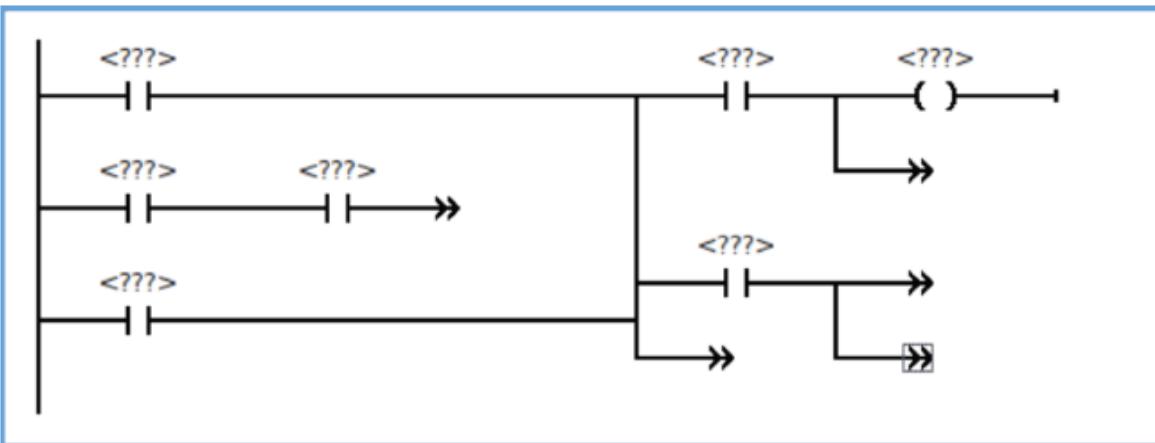


Figura 3-33: Diagrama Ladder. Circuito Completo

3.2.2.8.1.5 Accesos directos

IDE4PLC provee accesos directos para agregar contactos, bobinas y conexiones en forma rápida. Para usar un acceso directo hay que posicionar el cursor en el lugar donde se quiere insertar la bobina o contacto y pulsar la tecla correspondiente, por ejemplo “C”.

Para abrir una nueva rama se debe marcar el lugar a donde se desea abrirla y pulsar la tecla “O”.

Para cerrar una rama se debe marcar los dos extremos de la rama utilizando la tecla shift (como se indica en 3.2.2.8.1.4-Abriendo y cerrando una rama paralela) y pulsar la tecla “P”.

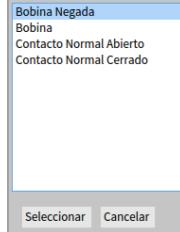
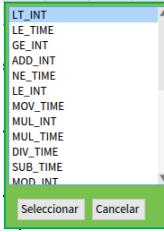
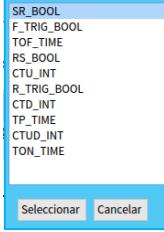
Elemento	Descripción	Acceso directo
	Contacto normal abierto	C
	Contacto normal cerrado	V
	Bobina	B
	Bobina negada	N
	Crea una rama	O
	Cierra una rama	P
	Presenta el menú de Componentes Ladder	D
	Presenta el menú de Funciones. ⁸	F
	Presenta el menú de Bloques de Funciones. ⁹	G

Figura 3-34: Accesos directos para agregar contactos y conexiones

⁸ El acceso directo F puede no estar disponible en todas las versiones⁹ El acceso directo G puede no estar disponible en todas las versiones

4 Elementos del Lenguaje Ladder

4.1 Elementos del lenguaje Ladder.

Los siguientes son los elementos del lenguaje Ladder que el programa IDE4PLC admite. Antes de describir los elementos se describen los tipos de datos soportados por el entorno.

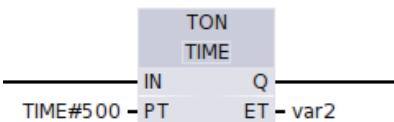
4.1.1 Tipos de datos soportados por IDE4PLC

Los tipos de datos soportados por IDE4PLC son Duración (TIME), Entero (INT) y booleano (BOOL). A continuación se describe cada uno de ellos y, a manera de ejemplo, se muestra un bloque que usa el tipo de dato.

4.1.1.1 Tipo de dato TIME

TIME es el tipo de datos que hace referencia a duración, expresado en milisegundos.

Este tipo de datos se utiliza principalmente en temporizadores pero también los soportan los bloques funcionales Aritméticos y de Comparación.



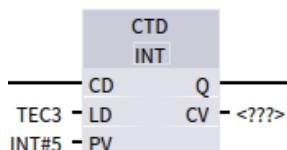
Los valores ingresados con este tipo de datos se deben ingresar como "TIME#nnn", donde nnn es el valor del tiempo en milisegundos. Por ejemplo TIME#200, para establecer un tiempo de 200 milisegundos.

El tipo de dato TIME tiene un rango de -2147483648 ms a 2147483647.

IMPORTANTE: Se debe escribir todo el texto (por ej. TIME#200) para que sea tomado adecuadamente por el bloque en el cual se usará. Si solo escribe el valor, por ejemplo 200, IDE4PLC lo tomará como un valor incorrecto y seguramente la aplicación no funcionará de acuerdo a lo esperado.

4.1.1.2 Tipo de dato INT

INT es el tipo de dato que hace referencia un dato entero. Este tipo de datos se utiliza en bloques Comparación, Aritméticos y Contadores.



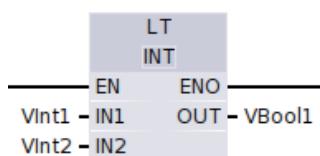
Los valores ingresados con este tipo de datos puede ingresarse solo con el valor (por ej. 5) y IDE4PLC completará el valor con "INT#".

Por ejemplo, si se ingresa solo el valor 5, IDE4PLC completará el dato con "INT#" formando "INT#5" y el dato será tomado en forma correcta.

El tipo de dato INT tiene un rango de -32768 a 32767 .

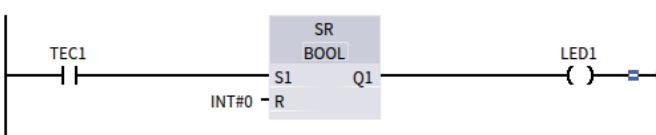
4.1.1.3 Tipo de dato BOOL

BOOL es el tipo de dato que hace referencia a un dato booleano o lógico. Si bien existen elementos, como



Bobinas y Contactos, y bloques funcionales que son del tipo BOOL, como bloques Biestables y detección de Flancos, todos los bloques usan este tipo de datos para habilitar la operación del bloque.

En este ejemplo, la entrada EN y las salidas ENO y OUT son del tipo BOOL.



Los valores booleanos verdadero ("1") y Falso ("0") deben ingresarse escribiendo el valor 1 o 0 y IDE4PLC lo completará INT#, como se ve en el ejemplo.

4.1.2 Datos generales de los elementos Ladder

Los siguientes son datos generales que aplican a todos los elementos Ladder.

- Todos los elementos Ladder tienen su entrada por la izquierda, del lado de la conexión de Fase (o línea positiva) y su salida por la derecha, del lado de la conexión de Neutro (o línea negativa).
- Todos los elementos Ladder, incluso las Bobinas, tienen al menos una entrada y una salida digital.
- La primera entrada de un Bloque de Función y la primera salida son siempre digitales.

4.1.3 Resumen de los elementos del lenguaje Ladder

Primeramente se hará un resumen de los elementos soportados por IDE4PLC y luego se describirán cada uno de ellos en forma particular.

4.1.3.1 Resumen de Contactos y Bobinas

Los Contactos y Bobinas son los elementos más típicos de los diagramas Laddder, ya que las Bobinas hacen referencia directa a las bobinas de relés y los Contactos a los contactos correspondientes a dichos relés.

En la Tabla 4-1 se listan los elementos Contactos y Bobinas incluidos en IDE4PLC.

Elemento	Descripción
Contactos y Bobinas	
	Nombre: Bobina Se activa al recibir tensión desde la línea de fase. Ver 4.1.4.1
	Nombre: Bobina negada. Se desactiva al recibir tensión desde la línea de fase. Ver 4.1.4.2
	Nombre: Contacto normal abierto. Se cierra cuando la bobina del relé asociado, o la entrada física, se activa. Ver 4.1.4.3
	Nombre: Contacto normal cerrado. Se abre cuando la bobina del relé asociado, o la entrada física, se activa. Ver 4.1.4.4

Tabla 4-1: Elementos Ladder. Contactos y Bobinas

4.1.3.2 Llamado a funciones



Funciones son aquellas que no tienen memoria, es decir que sus salidas están definidas totalmente por la combinación de sus entradas. Se pueden dividir en Funciones de Comparación y Aritméticas.

4.1.3.2.1 Funciones de Comparación

En la Tabla 4-2 se listan los elementos incluidos en IDE4PLC que permiten hacer comparaciones por igual, distinto, menor, mayor, etc. Cada elemento está disponible para ser usado con datos Enteros (INT) y datos del tipo Duración (TIME).

Elemento	Descripción
Funciones de comparación	
	Nombre: EQ Función: igual. Dato soportado: Entero (INT) y Duración (TIME). EQ_INT, EQ_TIME Ver 4.1.4.5
	Nombre: NE Función: distinto. Dato soportado: Entero (INT) y Duración (TIME). NE_INT, NE_TIME Ver 4.1.4.5
	Nombre: LT Función: menor. Dato soportado: Entero (INT) y Duración (TIME). LT_INT, LT_TIME Ver 4.1.4.5

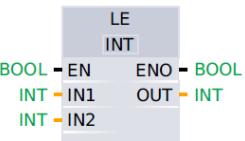
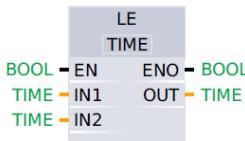
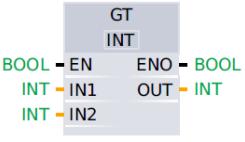
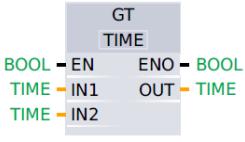
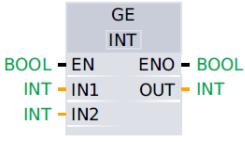
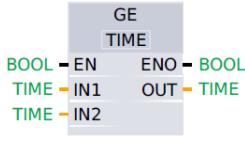
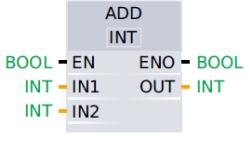
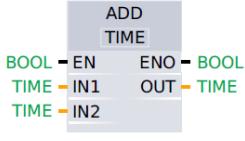
Elemento	Descripción
 	Nombre: LE Función: menor o igual. Dato soportado: Entero (INT) y Duración (TIME). LE_INT, LE_TIME Ver 4.1.4.5
 	Nombre: GT Función: mayor. Dato soportado: Entero (INT) y Duración (TIME). GT_INT, GT_TIME Ver 4.1.4.5
 	Nombre: GE Función: mayor o igual. Dato soportado: Entero (INT) y Duración (TIME). GE_INT, GE_TIME Ver 4.1.4.5

Tabla 4-2. Elementos Ladder. Funciones de comparación

4.1.3.2.2 Resumen de Funciones de Aritméticas

En la Tabla 4-3 se listan los elementos incluidos en IDE4PLC que permiten hacer funciones Aritméticas como Suma, Resta, Multiplicación, División, etc. Cada elemento está disponible para ser usado con datos Enteros (INT) y datos del tipo Duración (TIME).

Elemento	Descripción
Funciones Aritméticas	
 	Nombre: ADD Función: suma. Dato soportado: Entero (INT) y Duración (TIME). ADD_INT, ADD_TIME Ver 4.1.4.6

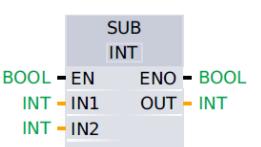
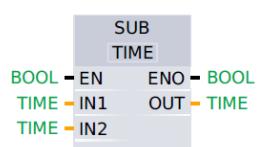
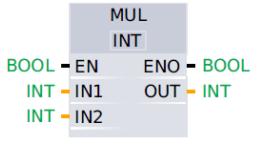
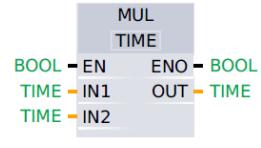
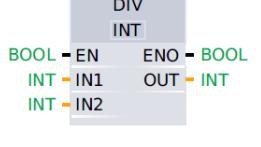
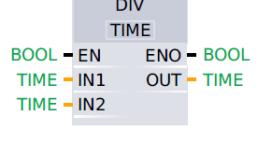
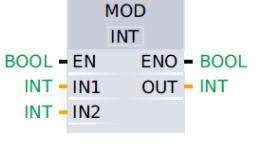
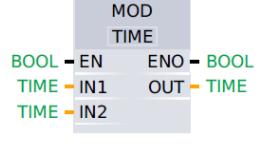
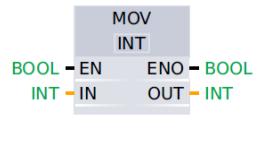
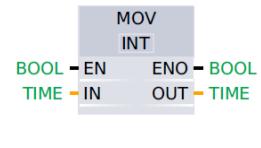
Elemento	Descripción
 	Nombre: SUB Función: resta. Dato soportado: Entero (INT) y Duración (TIME). SUB_INT, SUB_TIME Ver 4.1.4.6
 	Nombre: MUL Función: producto. Dato soportado: Entero (INT) y Duración (TIME). MUL_INT, MUL_TIME Ver 4.1.4.6
 	Nombre: DIV Función: división. Dato soportado: Entero (INT) y Duración (TIME). DIV_INT, DIV_TIME Ver 4.1.4.6
 	Nombre: MOD Función: resto de división Dato soportado: Entero (INT) y Duración (TIME). MOD_INT, MOD_TIME Ver 4.1.4.6
 	Nombre: MOV Función: movimiento Dato soportado: Entero (INT) y Duración (TIME). MOD_INT, MOD_TIME Ver 4.1.4.6

Tabla 4-3: Elementos Laddder. Funciones Aritméticas

4.1.3.3 Llamado a Bloque de Funciones



Bloques de Funciones son aquellas que tienen memoria, es decir que sus salidas están definidas por sus entradas y por el estado anterior. Sus estados internos y sus salidas se mantienen inalterables de un ciclo de scan al siguiente. Se pueden dividir en Bloques de Funciones de circuitos Biestables, Detección de Flancos, Contadores y Temporizadores.

4.1.3.3.1 Resumen de Funciones de Circuitos Biestables

En la Tabla 4-4 se listan los elementos incluidos en IDE4PLC que permiten hacer funciones biestables, también llamadas Flip-Flop en electrónica. Estos elementos cambian y mantienen su salida ante determinadas secuencias de entradas. Son indispensables para hacer circuitos de los denominados secuenciales, en los cuales no solo importa la combinación de determinadas variables sino también la secuencia en las cuales se presentan o cambian. Estos elementos usan datos del tipo booleano (BOOL).

Elemento	Descripción
Bloque de Función - Biestables	
	Nombre: SR Función: biestable con Set prioritario Dato soportado: Booleano (BOOL). SR_BOOL Ver 4.1.4.7
	Nombre: RS Función: biestable con Reset prioritario Dato soportado: Booleano (BOOL). RS_BOOL Ver 4.1.4.7

Tabla 4-4: Elementos Ladder. Bloque de Función – Biestables

4.1.3.3.2 Resumen de Funciones de detección de Flancos

En la Tabla 4-5 se listan los elementos incluidos en IDE4PLC que permiten detectar cambios en una variable o un contacto booleano. Estos elementos son indispensables cuando es necesario detectar un cambio de estado en una variable o contacto. Estos elementos usan datos del tipo booleano (BOOL).

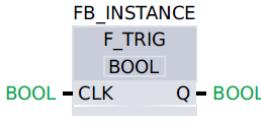
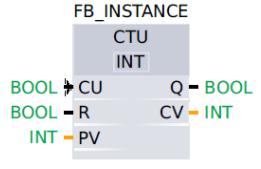
Elemento	Descripción
Bloque de Función – detección de Flancos	
	Nombre: R_TRIG Función: detector de flancos ascendentes Dato soportado: Booleano (BOOL) R_TRIG_BOOL Ver 4.1.4.8
	Nombre: F_TRIG Función: Detector de flancos descendentes Dato soportado: Booleano (BOOL) F_TRIG_BOOL Ver 4.1.4.8

Tabla 4-5: Elementos Ladder. Bloque de Función – detección de Flancos

4.1.3.3.3 Resumen de Funciones Contadores

En la Tabla 4-6 se listan los elementos incluidos en IDE4PLC que permiten contar cambios de estado en una variable booleana o un Contacto. Estos elementos son indispensables cuando es necesario contar transiciones de estados en una variable o contacto. Cada elemento está disponible para ser usado con datos Enteros (INT).

Elemento	Descripción
Bloque de Función - Contadores	
	Nombre: CTU Función: contador ascendente Dato soportado: Entero (INT) CTU_INT Ver 4.1.4.9.1

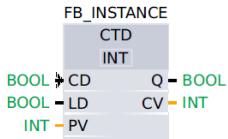
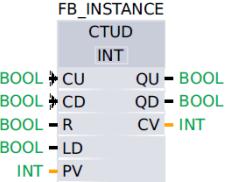
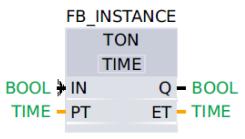
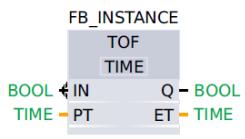
Elemento	Descripción
	Nombre: CTD Función: contador descendente Dato soportado: Entero (INT) CTD_INT Ver 4.1.4.9.2
	Nombre: CTUD Función: Contador descendente/descendente Dato soportado: Entero (INT) CTUD_INT Ver 4.1.4.9.3

Tabla 4-6: Elementos Ladder. Bloque de Función – Contadores

4.1.3.3.4 Resumen de Funciones Temporizadores

En la Tabla 4-7 se listan los elementos incluidos en IDE4PLC que permiten realizar temporizaciones. Estos elementos son indispensables cuando se requiere realizar retardos, anular entradas por un tiempo determinados o funciones similares en un circuito. Cada elemento está disponible para ser usado con datos Duración (TIMER).

Elemento	Descripción
Bloque de Función - Temporizadores	
	Nombre: TON Función: retardo a la conexión Dato soportado: Duración (TIME) TON_TIME Ver 4.1.4.10.1
	Nombre: TOF Función: retardo a la desconexión Dato soportado: Duración (TIME) TOF_TIME Ver 4.1.4.10.2

Elemento	Descripción
	Nombre: TP Función: temporizador de impulsos Dato soportado: Duración (TIME) TP_TIME Ver 4.1.4.10.3

Tabla 4-7: Elementos Ladder. Bloque de Función - Temporizadores

4.1.4 Descripción de Elementos Ladder

A continuación se describe cada elemento del lenguaje Ladder. En cada caso se detalla el elemento y se da un ejemplo de uso. Estos ejemplos pueden resultar triviales pero ayudarán a entender mejor cada elemento. Ejemplos mas completos, combinando varios elementos Ladder, se darán en el siguiente capítulo.

En el caso de aquellos elementos que tengan funciones similares (ej. Funciones de comparación como Igual, Menor, Menor o Igual, etc. o Funciones Aritméticas como Suma, Diferencia, Producto, etc.) se explicará uno y las diferencia que pueda tener con los elementos del mismo tipo.

4.1.4.1 Descripción de Bobina



Para agregar una Bobina al circuito pulse el icono indicado.



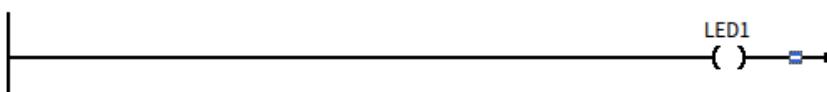
Este es el elemento de salida digital típico del lenguaje Ladder. Representa la bobina de un relé físico.

Si, por la rama izquierda, recibe tensión, la bobina se activará.

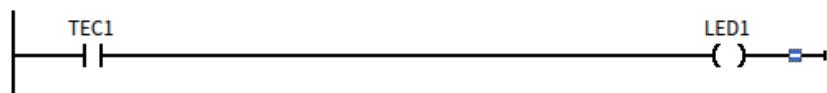
Este elemento puede representar una variable interna, es decir que no está asociada con una salida física, o bien con una salida Digital física de la EDU-CIAA (por ejemplo con LED1, LEDR, etc.). En ambos casos la Bobina tiene asociada uno o mas Contactos normal cerrado y/o normal abierto.

Este elemento tiene como salida un valor que repite la entrada. Esto es muy útil ya que permite conectar en serie varias Bobinas, simplificando el circuito, lo cual sería imposible hacerlo eléctricamente.

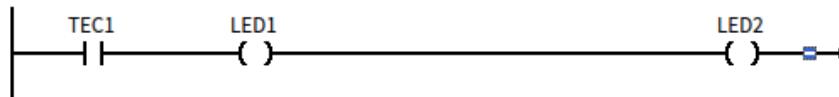
En el siguiente ejemplo se ve una Bobina que siempre recibe tensión desde la barra de alimentación izquierda (también llamada Left power rail), por lo cual siempre está activada, es decir que el LED1 (LED 1 de la placa EDU-CIAA) siempre estará encendido.



En el siguiente ejemplo se ve una Bobina que recibe tensión desde la barra de alimentación izquierda (también llamada Left power rail) por intermedio de un Contacto normal abierto (TEC1), es decir que solo recibirá tensión si el contacto TEC1 (pulsador TEC 1 de la placa EDU-CIAA) se cierra. Así el LED 1 solo se enciende cuando se pulsa el TEC 1 de la EDU-CIAA.



En el siguiente ejemplo dos salidas lógicas (Bobina LED1 y LED2, correspondientes a los leds LED 1 y LED 2) se activan cuando el pulsador TEC 1 de la EDU-CIAA se pulsa (se cierra).



Nota: para realizar cualquiera de estos ejemplos puede referirse al punto 3.2.2.8.1 (Haciendo un programa Ladder)

4.1.4.2 Descripción de Bobina Negada



Para agregar una Bobina Negada al circuito pulse el icono indicado y seleccione “Bobina Negada” del menú que se presenta.

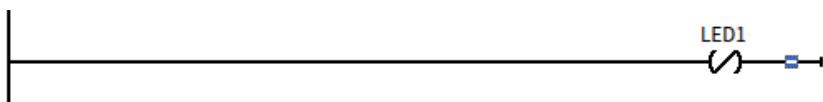


BOOL
Este elemento es similar a la Bobina común (del punto 4.1.4.1) solo que en este caso la Bobina se activa sin tensión y se desactiva con tensión. Este elemento no existe físicamente ya que se activa sin tensión, pero es muy útil ya que puede ayudar a simplificar circuitos. Si, por la rama izquierda, recibe tensión la bobina se desactiva y si no recibe tensión la Bobina se activa.

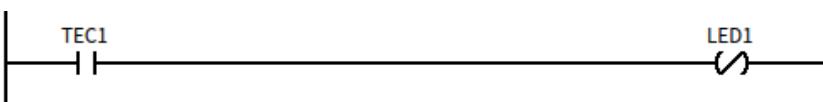
Este elemento puede representar una variable interna, es decir que no está asociada con una salida física, o bien con una salida Digital física de la EDU-CIAA (por ejemplo con LED 1, LED R, etc.). En ambos casos la bobina tiene asociada uno o mas Contactos normal cerrado y/o normal abierto.

Este elemento tiene como salida un valor que repite la entrada. Esto es muy útil ya que permite conectar en serie varias Bobinas, simplificando el circuito, lo cual sería imposible hacerlo eléctricamente.

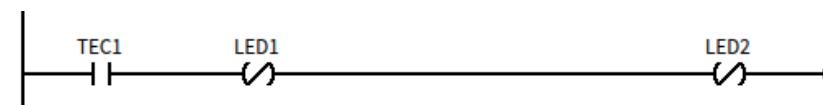
En el siguiente ejemplo se ve una Bobina que siempre recibe tensión desde la barra de alimentación izquierda (también llamada Left power rail), por lo cual siempre está desactivada, es decir que el LED 1 siempre estará apagado.



En el siguiente ejemplo se ve una Bobina que recibe tensión desde la barra de alimentación izquierda (también llamada Left power rail) por intermedio de un Contacto normal abierto (TEC1), es decir que solo recibirá tensión si el contacto TEC 1 se cierra. Así el LED 1 está normalmente encendido (ya que no recibe tensión por estar abierto el contacto) y solo se apaga cuando el pulsador TEC1 de la EDU-CIAA se pulsa (se cierra).

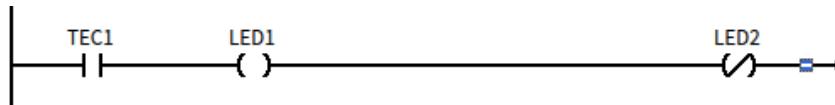


En el siguiente ejemplo dos salidas lógicas (Bobina LED1 y LED2, correspondientes a los leds LED 1 y LED 2) se desactivan cuando el pulsador TEC1 de la EDU-CIAA se pulsa (se cierra).



En el siguiente ejemplo la Bobina LED1 se activa (LED 1 se enciende) y la Bobina negada LED2 se desactiva (LED 2 se apaga) cuando el pulsador TEC1 de la EDU-CIAA se pulsa (se cierra). Mientras TEC1 no esté pulsado

la Bobina LED1 estará desactivada (LED 1 apagado) y la Bobina negada LED2 estará activada (LED 2 encendido).



Nota: para realizar cualquiera de estos ejemplos puede referirse al punto 3.2.2.8.1 (Haciendo un programa Ladder)

4.1.4.3 Descripción de Contacto Normal Abierto



Para agregar un Contacto Normal Abierto al circuito pulse el icono indicado.



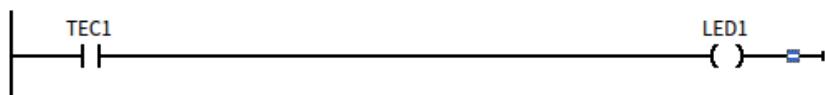
Este elemento es uno de los más comunes en los diagramas Ladder. Representa el contacto normal abierto de un relé o un contacto de una entrada física.

Cuando está asociado a un relé, este contacto se cierra cuando dicho relé está activado, de lo contrario permanece abierto.

Si está asociado a una entrada física, el contacto se cierra cuando la entrada se energiza y se abre cuando la entrada no tiene energía.

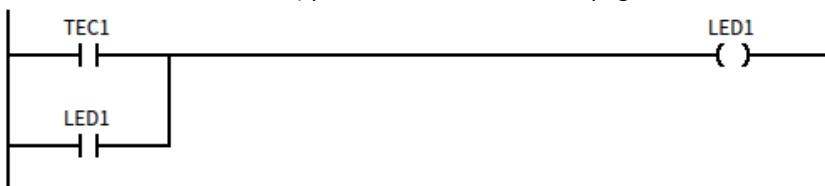
Este elemento puede estar asociado a una Bobina interna, un salida física de la EDU-CIAA (por ejemplo LED1) o un con una entrada Digital física de la EDU-CIAA (por ejemplo con TEC1, TEC2, etc.).

En el siguiente ejemplo se ve un contacto normal abierto (asociado a la entrada digital TEC1, el pulsador TEC 1 de la EDU-CIAA) que energiza una Bobina (asociada a la salida digital LED1, LED 1 de la EDU-CIAA). El led LED 1 se encenderá cuando el pulsador TEC 1 se pulse (se cierra).



En el siguiente ejemplo se muestra un circuito de retención permanente cuando se cierra TEC1 (se pulsa el pulsador TEC1 de la EDU-CIAA).

Cuando TEC1 se cierra (se pulsa TEC 1) el LED1 se activa (led LED 1 se enciende) y el contacto asociado a la Bobina LED1 (que no existe en la realidad ya que LED1 es una salida física) se cierra manteniendo el circuito cerrado a pesar de soltar el pulsador TEC 1. Este circuito no tiene un circuito de liberación (el cual se hará cuando se vea contactos normales cerrados) por lo cual LED 1 solo se apagará reseteando la EDU-CIAA.



4.1.4.4 Descripción de Contacto Normal Cerrado



Para agregar un Contacto Normal Cerrado al circuito pulse el icono indicado y seleccione "Contacto Normal Cerrado" del menú que se presenta.



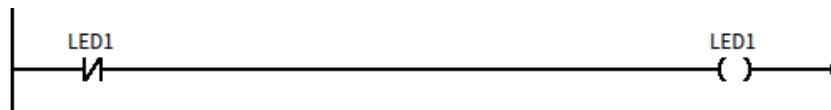
Este elemento es uno de los más comunes en los diagramas Ladder. Representa el contacto normal cerrado de un relé o un contacto de una entrada física.

Cuando está asociado a un relé, este contacto se abre cuando dicho relé está activado, de lo contrario permanece cerrado.

Si está asociado a una entrada física, el contacto se abre cuando la entrada se energiza y se cierra cuando la entrada no tiene energía.

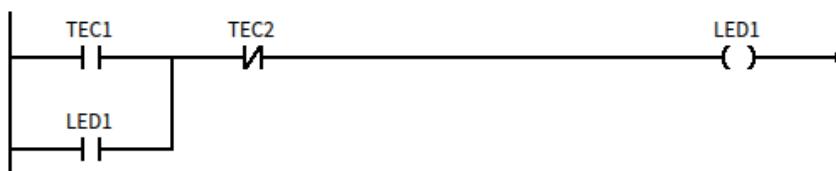
Este elemento puede estar asociado a una Bobina interna, una salida física de la EDU-CIAA (por ejemplo LED 1) o a una entrada Digital física de la EDU-CIAA (por ejemplo con TEC 1, TEC 2, etc.).

En el siguiente ejemplo se ve un contacto normal cerrado (asociado a la entrada digital TEC 1, un pulsado de la EDU-CIAA) que energiza una Bobina (asociada a la salida digital LED1, led LED 1 de la EDU-CIAA). El led LED 1 permanece encendido y cuando se pulsa TEC 1 el LED 1 se apaga.



En el siguiente ejemplo se muestra un circuito de retención con liberación. TEC 1 actúa como retenedor y TEC 2 como liberador.

Cuando TEC1 se cierra (se pulsa TEC 1) el LED1 se activa (led LED 1 se enciende) y el contacto asociado a la Bobina LED1 (que no existe en la realidad ya que LED1 es una salida física) se cierra manteniendo el circuito cerrado a pesar de soltar el pulsador TEC 1. Mientras lo anterior ocurre el contacto TEC2 se mantiene cerrado porque no está energizado (el pulsador TEC 2 no está pulsado), cerrando el circuito. La Bobina se libera cuando se abre el contacto TEC2 (cuando se pulsa TEC 2).



4.1.4.5 Descripción de las Funciones de Comparación



Para agregar un bloque de Comparación al circuito pulse el icono indicado y seleccione el Comparador deseado por su nombre y tipo de dato (EQ_INT, EQ_TIME, LT_INT, LT_TIME, etc.)

Existen funciones para comparar por Igual (EQ), Distinto/No Igual (NE), Menor (LT), Menor o Igual (LE), Mayor (GT) y Mayor o Igual (GE). La Tabla 4-8 describe las funciones disponibles.

Estos llamados a funciones realizan la comparación entre dos valores enteros (INT) o dos valores Duración (TIME).

Las entradas y salidas de todas estas funciones tienen funciones equivalentes, solo cambia la operación realizada por el bloque.

Entradas		
ID	Tipo	Función
EN	Booleana (BOOL)	Cuando está en "1" el bloque realiza la comparación y se genera la salida correspondiente. Esta entrada se activa por nivel.
IN1	Entero (INT) o Duración (TIME)	Es el primer argumento de la comparación.
IN2	Entero (INT) o Duración (TIME)	Es el segundo argumento de la comparación.

Salidas		
ID	Tipo	Función
ENO	Booleana (BOOL)	Esta salida repite el valor de la entrada EN.
OUT	Booleana (BOOL)	Es el resultado de la comparación realizada por el bloque.

La entrada EN habilita la función del bloque, es decir que el bloque solo realiza la operación mientras su entrada EN está energizada.

La Comparación realizada es del tipo IN1 cmp IN2, donde *cmp* es la comparación correspondiente al bloque usado. Por ejemplo, para menor (LE) la comparación es: IN1 < IN2.

La salida OUT se energiza cuando el resultado de la comparación entre IN1 e IN2 es verdadero. Si este resultado es falso la salida OUT no se energiza.

Elemento	Descripción
Funciones de comparación	
	Nombre: EQ Función: igual. Dato soportado: Entero (INT) y Duración (TIME). EQ_INT, EQ_TIME
	Nombre: NE Función: distinto. Dato soportado: Entero (INT) y Duración (TIME). NE_INT, NE_TIME
	Nombre: LT Función: menor. Dato soportado: Entero (INT) y Duración (TIME). LT_INT, LT_TIME
	Nombre: LE Función: menor o igual. Dato soportado: Entero (INT) y Duración (TIME). LE_INT, LE_TIME
	Nombre: GT Función: mayor. Dato soportado: Entero (INT) y Duración (TIME). GT_INT, GT_TIME

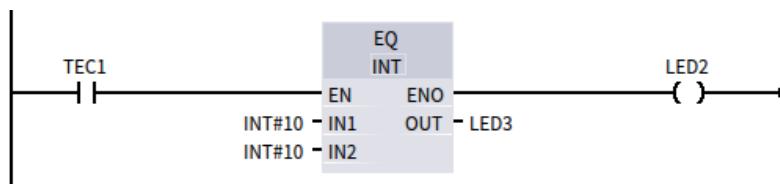
Elemento	Descripción
<p>Nombre: GE Función: mayor o igual. Dato soportado: Entero (INT) y Duración (TIME). GE_INT, GE_TIME</p>	

Tabla 4-8: Elementos Ladder. Funciones de comparación

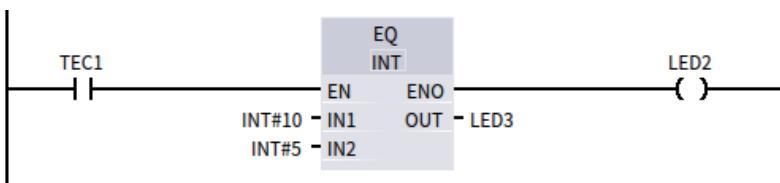
En el siguiente ejemplo los LED 2 y LED 3 se encenderán cuando se pulsa TEC 1.

LED 2 encenderá porque la salida ENO repite la entrada EN.

LED 3 encenderá porque IN1 es igual a IN2 (ambos valen 10).



Si se cambia el valor de IN2, por ejemplo a 5, se verá que al pulsar TEC 1 se enciende LED 2 pero no LED 3, puesto que no se cumple con la condición de igualdad.



4.1.4.6 Descripción de las Funciones Aritméticas



Para agregar un bloque Aritmético al circuito pulse el icono indicado y seleccione la función Aritmética deseada por su nombre y tipo de dato (ADD_INT, ADD_TIME, SUB_INT, SUB_TIME, etc.)

Existen funciones Aritméticas de Suma (ADD), resta (SUB), Multiplicación (MUL) División (DIV), resto de División (MOD) y Movimiento de datos (MOV). La Tabla 4-9 describe las funciones disponibles.

Estos llamados a funciones realizan la Operación Aritmética entre dos valores enteros (INT) o dos valores Duración (TIME).

Las entradas y salidas de todas estas funciones tienen funciones equivalentes, solo cambia la operación realizada por el bloque.

Entradas		
ID	Tipo	Función
EN	Booleana (BOOL)	Cuando está en "1" el bloque realiza la Operación Aritmética y se genera la salida correspondiente. Esta entrada se activa por nivel.
IN1	Entero (INT) o Duración (TIME)	Es el primer argumento de la Operación Aritmética.
IN2	Entero (INT) o Duración (TIME)	Es el segundo argumento de la Operación Aritmética.

Salidas		
ID	Tipo	Función
ENO	Booleana (BOOL)	Esta salida repite el valor de la entrada EN.
OUT	Entero (INT)	Es el resultado de la Operación Aritmética realizada por el bloque.

La entrada EN habilita la función del bloque, es decir que el bloque realiza la Operación Aritmética solo cuando se activa su entrada EN.

La Operación Aritmética realizada es del tipo IN1 *OpArit* IN2, donde *OpArit* es la Operación Aritmética correspondiente al bloque usado. Por ejemplo, para resta (SUB) la Operación Aritmética es: IN1 - IN2.

La salida OUT da el resultado de la Operación Aritmética entre IN1 e IN2.

Elemento	Descripción
Funciones Aritméticas	
	Nombre: ADD Función: suma. Dato soportado: Entero (INT) y Duración (TIME). ADD_INT, ADD_TIME
	Nombre: SUB Función: resta. Dato soportado: Entero (INT) y Duración (TIME). SUB_INT, SUB_TIME
	Nombre: MUL Función: producto. Dato soportado: Entero (INT) y Duración (TIME). MUL_INT, MUL_TIME
	Nombre: DIV Función: división. Dato soportado: Entero (INT) y Duración (TIME). DIV_INT, DIV_TIME

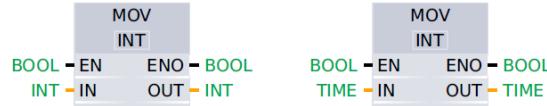
Elemento	Descripción
	Nombre: MOD Función: resto de división Dato soportado: Entero (INT) y Duración (TIME). MOD_INT, MOD_TIME
	Nombre: MOV Función: movimiento Dato soportado: Entero (INT) y Duración (TIME). MOV_INT, MOV_TIME

Tabla 4-9: Elementos Laddder. Funciones Aritméticas

En el siguiente circuito se ve un ejemplo del uso del bloque de función suma (ADD).

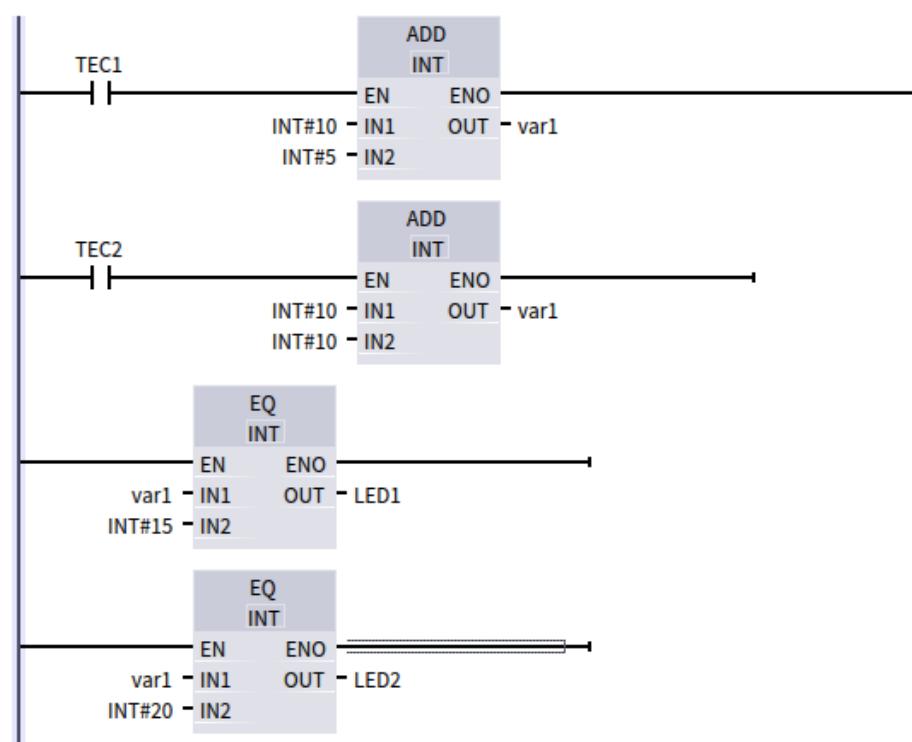
El primer bloque de suma realizar la suma cuando TEC 1 se cierra.

El segundo bloque de suma realiza la suma cuando TEC 2 se cierra.

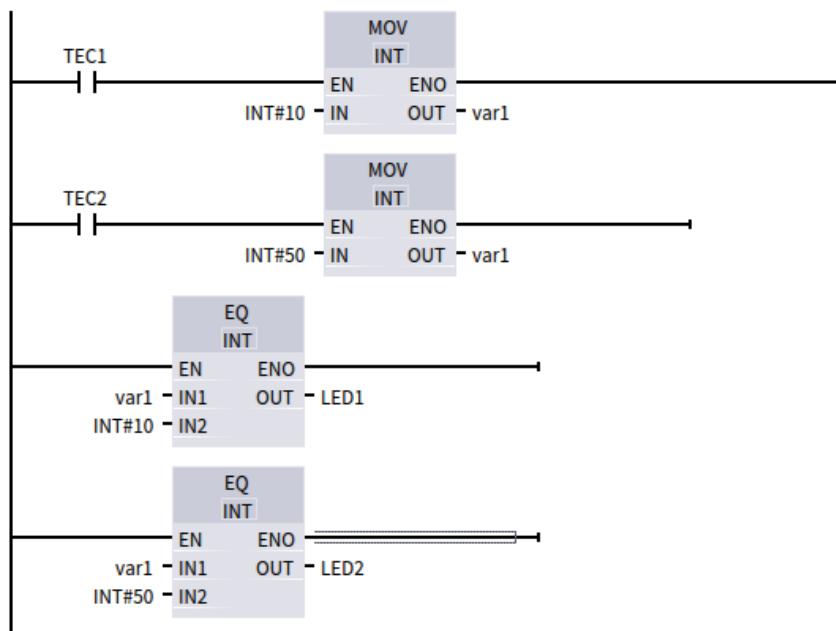
Si TEC 1 se cierra, var1 tomará el valor 15 (10 + 5). En este caso la salida Q del primer bloque EQ se activará y se encenderá el LED 1.

Si TEC 2 se cierra, var1 tomará el valor 20 (10 + 10). En este caso la salida Q del segundo bloque EQ se activará y se encenderá el LED 2.

Si ambos TEC 1 y TEC 2 se pulsan simultáneamente, el último bloque de suma es el que establece el valor final de var1 (20), por lo tanto el led que se encenderá será LED 2.



En el siguiente circuito se utiliza el bloque de Movimiento (MOV) en lugar del bloque ADD, pero el efecto es el mismo.



4.1.4.7 Descripción de las Funciones de Circuitos Biestables



Para agregar un bloque Biestable al circuito Ladder pulse el icono indicado y seleccione el Biestable deseado por su nombre (SR_BOOL o RS_BOOL)

Existen Circuitos Biestables del tipo Set/Reset (SR) con Set prioritario y con Reset prioritario (RS). La Tabla 4-10 describe las funciones disponibles.

Estas Funciones Biestables operan con valores booleanos (BOOL) tanto de entrada como salida.

Las entradas y salidas de estas funciones tienen funciones equivalentes, solo cambia la prioridad de las entradas en cada bloque. El bloque SR es de Set prioritario mientras que el bloque RS es de Reset prioritario.

Entradas		
ID	Tipo	Función
S S1	Booleana (BOOL)	Es la entrada Set del circuito Biestable.
R R1	Booleana (BOOL)	Es la entrada Reset del circuito Biestable.

Salidas		
ID	Tipo	Función
Q1	Booleana (BOOL)	Es la salida del Circuito Biestable.

La salida Q1 se activa cuando en la entrada S (o S1) hay un “1” y en la entrada R (o R1) hay un “0”. Si ambas entradas están en “0” la salida Q no se activa (“0”). Esto ocurre de la misma manera en ambos Biestables. La diferencia entre el Biestable SR y el RS se ve cuando ambas entradas están en “1”.

En el caso del Biestable SR, cuando tanto S1 como R1 están en “1”, la salida se activa puesto que es de Set prioritario.

En el caso del Biestable RS, cuando tanto S como R1 están en “1”, la salida se desactiva puesto que es de Reset prioritario.

Elemento	Descripción
Bloque de Función – Biestables	
<p>Nombre: SR Función: biestable con Set prioritario Dato soportado: Booleano (BOOL). SR_BOOL</p>	
<p>Nombre: RS Función: biestable con Reset prioritario Dato soportado: Booleano (BOOL). RS_BOOL</p>	

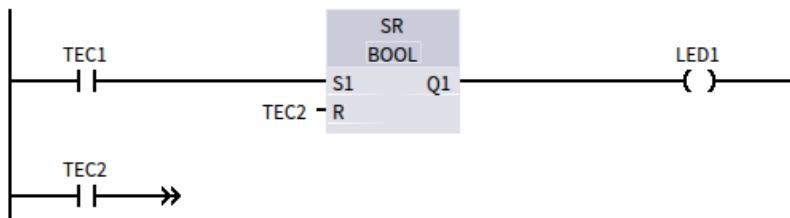
Tabla 4-10: Elementos Ladder. Bloque de Función – Biestables

En el siguiente circuito se muestra el funcionamiento del Biestable SR.

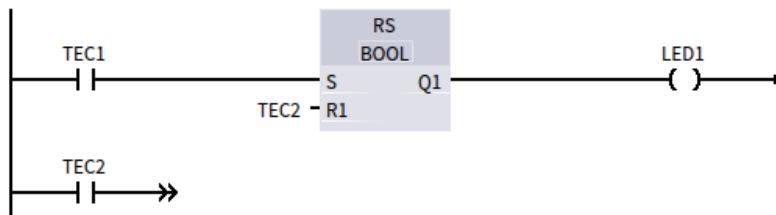
Si se pulsa TEC 1 se enciende el led LED 1.

Si se pulsa TEC 2 el Biestable se resetea y LED 1 se apaga.

Si se pulsan simultáneamente TEC 1 y TEC 2 el led LED 1 se enciende ya que este Biestable es de Set prioritario.



El siguiente circuito se comporta como el anterior, excepto que al pulsar simultáneamente TEC 1 y TEC 2 el led LED 1 se apaga ya que este Biestable es de Reset prioritario.



4.1.4.8 Descripción de las Funciones de Detección de Flancos



Para agregar un bloque de Detección de Flancos al circuito Ladder pulse el icono indicado y seleccione el Detector de Flancos deseado por su nombre (R_TRIG_BOOL o F_TRIG_BOOL).

Existen Circuitos de Detección de que permiten detectar Flancos Ascendentes (R_TRIG) y detección de Flancos Descendentes (F_TRIG)

Estas Funciones de Detección de Flancos operan con valores booleanos (BOOL) tanto de entrada como salida.

Las entradas y salidas de estas funciones tienen funciones equivalentes, solo cambia el tipo de Flanco que el bloque detecta.

Entradas		
ID	Tipo	Función
CLK	Booleana (BOOL)	Es la entrada de Flancos.

Salidas		
ID	Tipo	Función
Q	Booleana (BOOL)	Es la salida del Circuito de detección de Flancos.

Cuando se presenta un flanco ascendente o descendente (para R_TRIG o F_TRIG según el bloque usado) la salida Q se activa y permanece activada hasta el próximo scan. Es decir que, si se detecta un flanco, la variable de salida del bloque estará activa desde que se resuelve el bloque hasta que termina un barrido (scan) completo del programa.

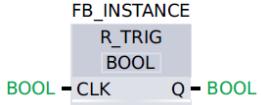
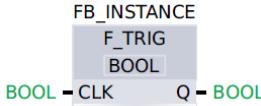
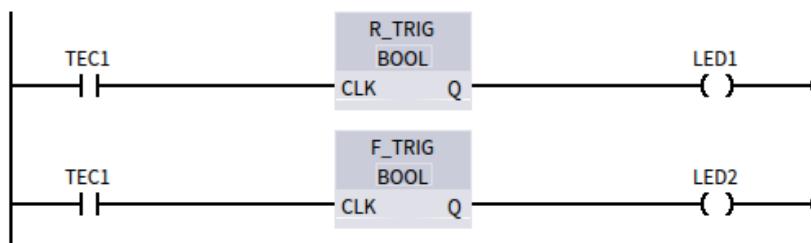
Elemento	Descripción
Bloque de Función – detección de Flancos	
	Nombre: R_TRIG Función: detector de flancos ascendentes Dato soportado: Booleano (BOOL) R_TRIG_BOOL
	Nombre: F_TRIG Función: detector de flancos descendentes Dato soportado: Booleano (BOOL) F_TRIG_BOOL

Tabla 4-11: Elementos Ladder. Bloque de Función – detección de Flancos

En el siguiente circuito se demuestra el funcionamiento de los detectores de flancos.

Cuando se pulsa TEC 1 se enciende el led LED 1 por un periodo de scan (solo se alcanza a ver un parpadeo).

Cuando se suelta TEC 1 se enciende el led LED 2 por un periodo de scan (solo se alcanza a ver un parpadeo).



4.1.4.9 Descripción de las Funciones Contadores



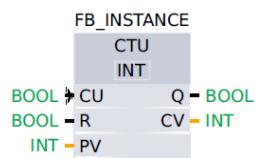
Para agregar un Contador al circuito Ladder pulse el icono indicado y seleccione el Contador deseado por su nombre (CTU_INT, CTD_INT o CTUD_INT).

Existen Circuitos Contadores que realizan cuentas ascendentes (CTU), cuentas descendentes (CTD) o cuentas ascendentes/descendentes (CTUD).

Estas Funciones Contadores operan con valores enteros (INT) tanto de entrada como salida.

Las entradas y salidas del Contador ascendente (CTU) y del contador descendente (CTD) tienen funciones equivalentes, solo cambia el sentido de la cuenta.

4.1.4.9.1 Contador ascendente (CTU)



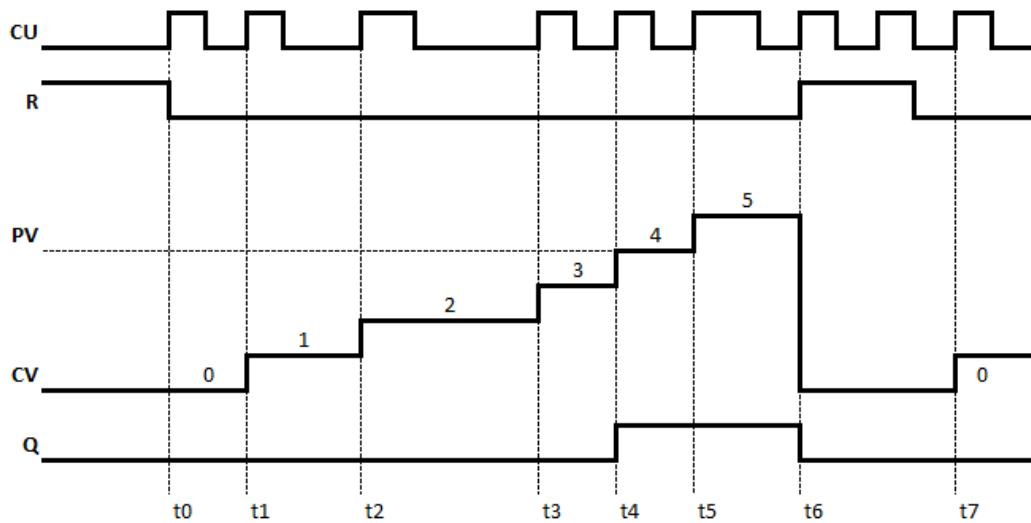
En el Contador ascendente (CTU) la cuenta inicia desde cero, cuando el contador se resetea (entrada R), y la salida Q se activa cuando llega al valor Preseleccionado (PV).

Entradas		
ID	Tipo	Función
CU	Booleana (BOOL)	Es la entrada de Flancos a contar. Con cada flanco ascendente en CU la cuenta se incrementa en 1.
R	Booleana (BOOL)	R es la entrada de Reset del contador ascendente. Pone el contador a cero.
PV	Entero (INT)	Es la entrada del valor de Preselección.

Salidas		
ID	Tipo	Función
Q	Booleana (BOOL)	Es la salida Contador. En el Contador ascendente se activa cuando se llega al valor de preselección.
CV	Entero (INT)	Valor actual de la cuenta.

En el siguiente gráfico se puede ver el diagrama temporal del contador ascendente (CTU).

Hasta t0 el contador está reseteado (en cero) por efecto del valor “1” en la entrada R, a partir de t0 R pasa a cero y el contador queda liberado para contar. A partir de t0 por cada flanco ascendente en CU (t1, t2, t3, t4, t5) el valor de CV se incrementa en uno. Cuando el contador llega al valor preestablecido (en este caso 4) por PV (lo que ocurre en t4) la salida Q se activa hasta que se presenta el próximo Reset (R) en t6. Si no hay un Reset el contador sigue contando mas allá del valor PV.



En el siguiente circuito se ve el funcionamiento del contador ascendente (CTU).

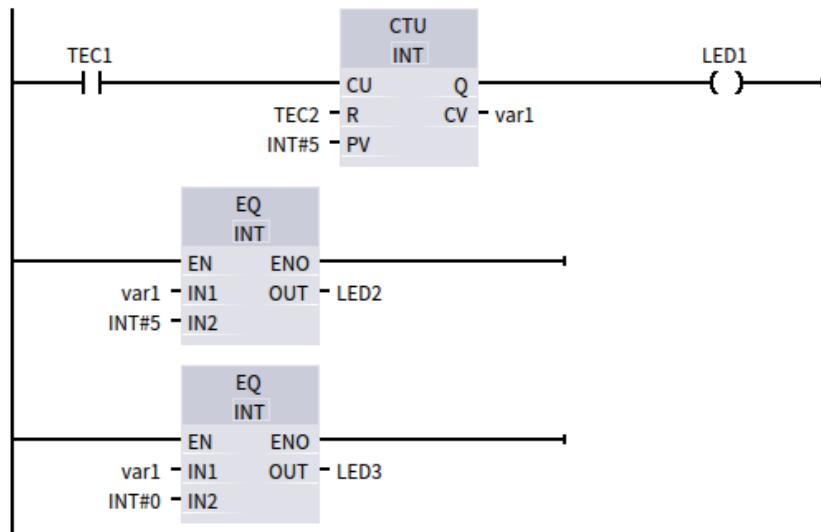
Cuando el circuito se descarga a la placa EDU-CIAA el led LED 3 se enciende ya que el contador (salida CV) está en cero.

Si se pulsa TEC 1 una vez el LED 3 se apaga porque var1 pasó a valer 1. LED 1 permanece apagado porque el contador no llegó a la cuenta preestablecida por PV (en este caso 5). LED 2 permanece apagado porque var1 no vale 5.

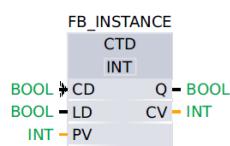
Si se pulsa TEC 1 cuatro veces más el LED 1 se enciende porque la salida Q se activa ya que el contador llegó a la cuenta preestablecida por PV (en este caso 5). El led LED 2 se enciende porque var1 llegó a 5.

Si se pulsa una vez mas TEC 1 se ve que el LED 1 permanece encendido mientras que el LED 2 se apaga puesto que ahora var1 vale 6.

Si se pulsa TEC 2 (entrada de Reset) se vuelve a la situación inicial en la cual el LED 3 se enciende porque la salida Q del contador vuelve a valer cero.



4.1.4.9.2 Contador descendente (CTD)



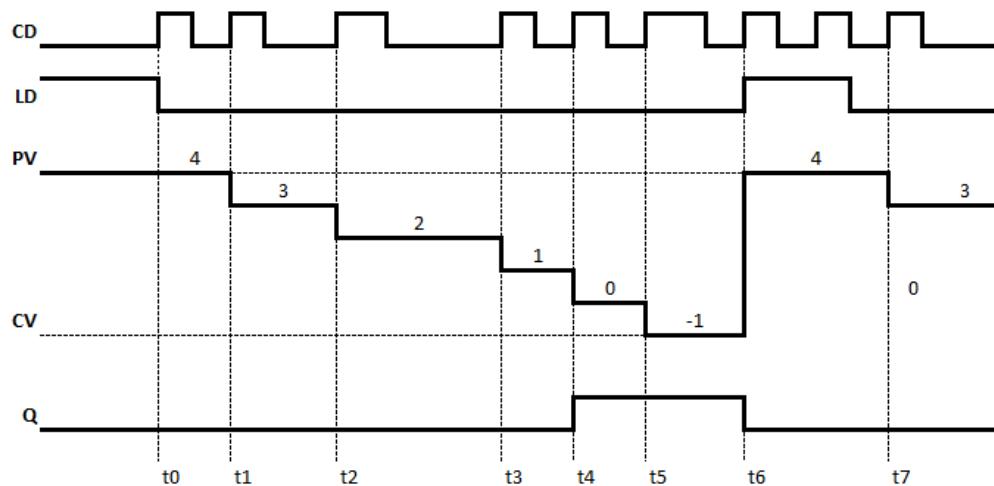
En el Contador descendente (CTD) la cuenta inicia desde el valor Preseleccionado, cuando el contador se preselecciona (entrada LD), y la salida Q se activa cuando llega al valor cero.

Entradas		
ID	Tipo	Función
CD	Booleana (BOOL)	Es la entrada de Flancos a contar. Con cada flanco ascendente en CD la cuenta se decremente en 1.
LD	Booleana (BOOL)	LD es la entrada de carga del contador descendente. Pone el contador en el valor de preselección.
PV	Entero (INT)	Es la entrada del valor de Preselección.

Salidas		
ID	Tipo	Función
Q	Booleana (BOOL)	Es la salida Contador. En el Contador descendente se activa cuando se llega cero.
CV	Entero (INT)	Valor actual de la cuenta.

En el siguiente gráfico se puede ver el diagrama temporal del contador descendente (CTD).

En t0, por efecto del valor "1" en LD, el contador se preestablece con el valor de PV (en este caso 4). A partir de t0 la entrada LD pasa a cero y el contador se libera, y por cada flanco ascendente CD (t1, t2, t3, t4, t5, t6) el valor de CV se decrementa en uno. Cuando el contador llega a cero la salida Q se activa hasta que se presenta el próximo Load (LD) en t6. Si no hay un Load el contador sigue contando en forma descendente mas allá del valor PV.



En el siguiente circuito se ve el funcionamiento del contador descendente (CTD).

Cuando el circuito se descarga a la placa EDU-CIAA el led LED 3 se enciende ya que el contador (salida CV) está en cero. También se enciende el LED 1 porque el contador inicia en cero.

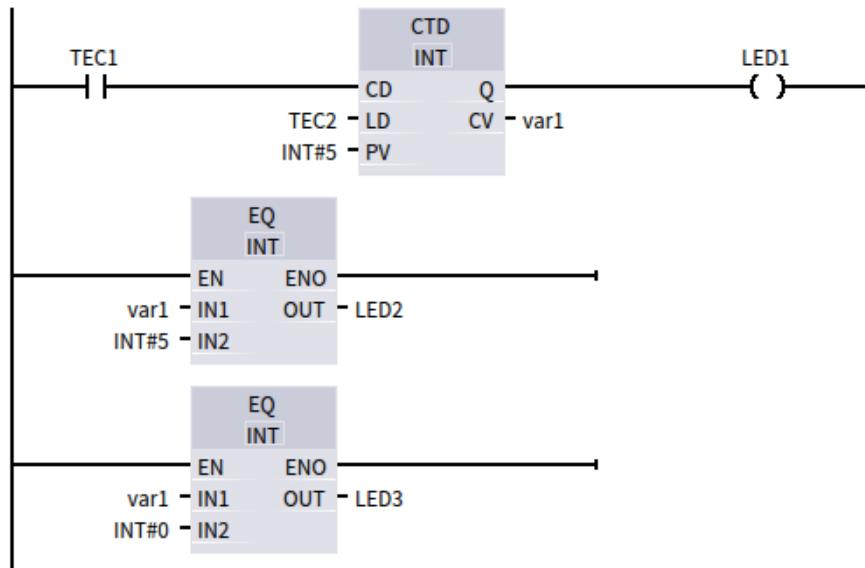
Si se pulsa TEC 2 para que se cargue el valor inicial presente en la entrada LD (en este caso 5) los LED 1 y 3 se apagan porque ahora la salida CV se establece en el valor 5 (valor preestablecido por la entrada PV) y por la misma causa se enciende el LED 2.

Si se pulsa TEC 1 una vez, el LED 2 se apaga porque ahora la salida Q tiene el valor 4. LED 1 y LED 3 permanece apagado.

Si se pulsa TEC 1 cuatro veces más el LED 1 se enciende porque la salida Q se activa ya que el contador llegó a cero. El led LED 3 se enciende porque la var1 llegó a cero.

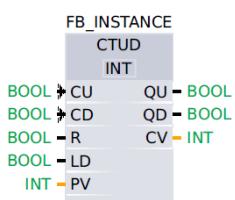
Si se pulsa una vez mas TEC 1 se ve que el LED 1 permanece encendido mientras que el LED 2 se apaga puesto que ahora var1 vale -1.

Si se pulsa TEC 2 se vuelve a la situación inicial en la cual el LED 2 se enciende porque la salida Q del contador vuelve a valer 5 (valor de preestablecimiento).



4.1.4.9.3 Contador descendente/descendente (CTUD)

El Contador ascendente/descendente (CTUD) cuenta incrementando o decrementando su valor actual (CV) dependiendo de la entrada por la cual se reciben flancos ascendentes.



Si el contador recibe flancos positivos por la entrada CU se produce una cuenta ascendente y el valor de la salida CV se incrementa.

Si el contador recibe flancos positivos por la entrada CD se produce una cuenta descendente y el valor de la salida CV se decrementa.

Si, al contar en forma ascendente, el valor de la salida CV llega al valor Preseleccionado, la salida QU se activa.

Si, al contar en forma descendente, el valor de la salida CV llega al valor cero, la salida QD se activa.

El valor Preseleccionado (PV) se carga en el contador cuando se activa la entrada LD.

Si la entrada de Reset (R) se activa el valor de la cuenta actual (CV) se pone en cero.

Entradas		
ID	Tipo	Función
CU	Booleana (BOOL)	Es la entrada de Flancos a contar. Con cada flanco ascendente en CU la cuenta se incrementa en 1.
CD	Booleana (BOOL)	Es la entrada de Flancos a contar. Con cada flanco ascendente en CD la cuenta se decrementa en 1.
R	Booleana (BOOL)	R es la entrada de Reset del contador ascendente. Pone el contador a cero.
LD	Booleana (BOOL)	LD es la entrada de carga del contador descendente. Pone el contador en el valor de preselección.
PV	Entero (INT)	Es la entrada del valor de Preselección.

Salidas		
ID	Tipo	Función
QU	Booleana (BOOL)	En el Contador ascendente/descendente se activa cuando la cuenta llega al valor de Preselección (PV).
QD	Booleana (BOOL)	En el Contador ascendente/descendente se activa cuando la cuenta llega a cero.
CV	Entero (INT)	Valor actual de la cuenta.

4.1.4.9.4 Resumen de las Funciones Contadores

Elemento	Descripción
Bloque de Función - Contadores	
	Nombre: CTU Función: Contador ascendente Dato soportado: Entero (INT) CTU_INT
	Nombre: CTD Función: Contador descendente Dato soportado: Entero (INT) CTD_INT
	Nombre: CTUD Función: Contador descendente/ascendente Dato soportado: Entero (INT) CTUD_INT

Tabla 4-12: Elementos Ladder. Bloque de Función – Contadores

4.1.4.10 Descripción de las Funciones Temporizadores



Para agregar un temporizador al circuito Ladder pulse el icono indicado y seleccione el temporizador deseado por su nombre (TON_TIME, TOF_TIME o TP_TIME).

Existen Circuitos Temporizadores en donde su salida se activa un tiempo después de recibir un flanco ascendente y mientras se mantiene energizada la entrada (TON), en donde su salida se desactiva un tiempo después de recibir un flanco descendente y mientras se mantiene desenergizada su entrada (TOF) y en donde su salida se activa un tiempo después de recibir un flanco ascendente aunque su entrada ya se haya desenergizado (TP).

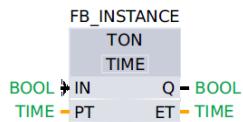
Estas Funciones Temporizadores operan con valores Duración (TIME).

Las entradas y salidas de los temporizadores tienen funciones equivalentes, solo cambia el modo de trabajo.

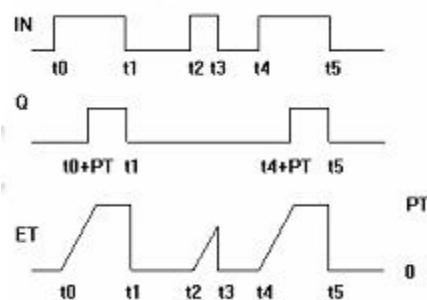
Entradas		
ID	Tipo	Función
IN	Booleana (BOOL)	Es la entrada de Flancos que habilita el temporizado.
PT	Duración (TIME)	Es la entrada que establece el tiempo del temporizado.

Salidas		
ID	Tipo	Función
Q	Booleana (BOOL)	Es la salida que se activa o se desactiva (dependiendo el tipo de Temporizador) cuando el tiempo llegó a valor preestablecido en PT.
ET	Duración (TIME)	Es la salida que indica el valor actual del temporizado.

4.1.4.10.1 Temporizador con retardo a la conexión (TON)



El Temporizador con retardo a la conexión (TON) comienza a contar el tiempo desde el momento en que se recibe un flanco ascendente por su entrada IN y mientras la entrada siga energizada. Cuando la cuenta llega al valor preestablecido por PT, la salida Q se activa hasta que el nivel de la entrada IN vuelve a cero. Si la entrada IN está en estado alto por un tiempo menor que el establecido por PT (como entre t2 y t3), la salida Q no se activa.



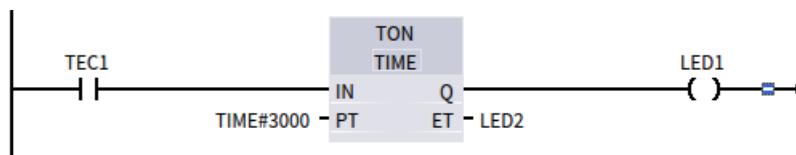
En este diagrama de tiempos se ve que en t0 se presenta un flanco ascendente en la entrada IN, esto hace que comience la cuenta de tiempo interna. Cuando el tiempo llega al valor PT (en $t_0 + PT$) la salida Q se activa hasta que la entrada IN vuelve a cero (en t1).

En el tiempo t2 el temporizador recibe otro flanco ascendente. Nuevamente el contador comienza a contar, pero en este caso la entrada IN vuelve a cero (en t3) antes de que se llegue al tiempo PT, por lo cual la salida Q no se activa.

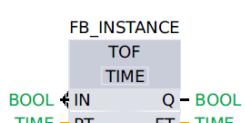
En el tiempo t4 nuevamente se presenta un flanco

ascendente. Nuevamente el contador comienza a contar. En $t_4 + PT$ llega al tiempo predefinido, entonces la salida Q se activa. En t5 la entrada IN se desactiva y la salida Q va a cero.

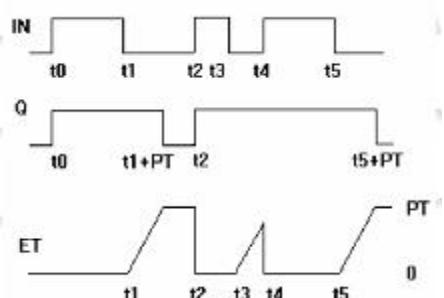
El siguiente circuito demuestra el funcionamiento de temporizador con retardo a la conexión (TON). Si TEC 1 se mantiene pulsado por mas tiempo que lo que tiene la entrada PT (en este caso 3 segundos, es decir 3000 ms) el LED 1 se enciende. Cuando TEC 1 se suelta LED 1 se apaga. Si TEC 1 se pulsa por menos de 3 segundos el LED 1 no enciende.



4.1.4.10.2 Temporizador con retardo a la desconexión (TOF)



El Temporizador con retardo a la desconexión (TOF) comienza a contar el tiempo desde el momento en que se recibe un flanco descendente por su entrada IN y mientras la entrada siga desenergizada. Cuando la cuenta llega al valor preestablecido por PT, la salida Q se desactiva hasta que el nivel de la entrada IN vuelve a energizarse. Si la entrada IN está en estado bajo por un tiempo menor que el establecido por PT, la salida Q no se desactiva.

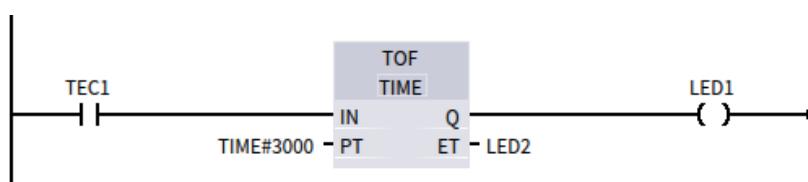


En este diagrama de tiempos se ve que en t_0 se presenta un flanco ascendente en la entrada IN, esto hace que la salida Q se active. En t_1 se presenta un flanco descendente en IN lo que provoca que comience la cuenta de tiempo interna. Cuando el tiempo llega al valor PT (en $t_1 + PT$) la salida Q se desactiva y vuelve a cero.

En el tiempo t_2 el temporizador recibe otro flanco ascendente. Nuevamente la salida Q se activa. En t_3 llega un flanco descendente, el contador comienza a contar, pero en este caso la entrada IN vuelve a activarse (en t_4) antes de que se llegue al tiempo PT, por lo cual la salida Q no se desactiva.

En el tiempo t_5 nuevamente se presenta un flanco descendente. Nuevamente el contador comienza a contar. En $t_5 + PT$ llega al tiempo predefinido, entonces la salida Q se desactiva.

El siguiente circuito demuestra el funcionamiento de temporizador con retardo a la desconexión (TOF). Cuando TEC1 se pulsa, la salida Q se activa. Cuando TEC1 se suelta el temporizador comienza a correr y después del valor de PT (en este caso 3 segundos, es decir 3000 ms) LED1 se apaga.

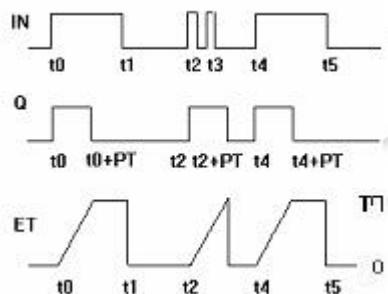


4.1.4.10.3 Temporizador de impulsos (TP)

FB_INSTANCE
TP
TIME

BOOL → IN	Q = BOOL
TIME → PT	ET = TIME

El Temporizador de impulsos (TP) comienza a contar el tiempo desde el momento en que se recibe un flanco ascendente por su entrada IN, sin importar si la entrada sigue energizada o no. Cuando la cuenta llega al valor preestablecido por PT, la salida Q se desactiva. Si la entrada IN recibe otro flanco positivo nuevamente se reinicia la cuenta.



En este diagrama de tiempos se ve que en t_0 se presenta un flanco ascendente en la entrada IN, esto hace que la salida Q se active. En ese momento el temporizador comienza su cuenta. Cuando la cuenta llega al valor TP (lo que ocurre en $t_0 + PT$) la salida Q se desactiva. Cuando la entrada IN pasa a cero (en t_1) el contador interno vuelve a cero pero la salida no se modifica puesto que ya estaba en cero.

En el tiempo t_2 se presenta un flanco ascendente en IN. La salida Q se activa y el contador interno comienza a contar. En t_3 llega otro flanco ascendente pero como el contador interno todavía no llegó a PT, la salida no cambia. En $t_2 + PT$ el contador llega al valor predefinido, por lo cual la salida Q se desactiva.

En t_4 el temporizador recibe un flanco ascendente que hace que el temporizado comience nuevamente, cuando el contador llega al tiempo predefinido por TP (en el instante $t_4 + PT$) la salida Q se desactiva. En t_5 la entrada pasa a cero con lo cual el contador interno del temporizador pasa a cero.

4.1.4.10.4 Resumen de Temporizadores

Elemento	Descripción
Bloque de Función – Temporizadores	
 <p>The timing diagram illustrates the TON function. The input IN is active at times t0, t2, t4, and t5. The output Q is active from t0+PT to t1, and again from t4+PT to t5. The elapsed time ET is zero during inactive periods and increases during active periods, starting from t0 and reaching zero again at t5.</p>	Nombre: TON Función: Retardo a la conexión Dato soportado: Duración (TIME) TON_TIME
 <p>The timing diagram illustrates the TOF function. The input IN is active at times t0, t2, t3, t4, and t5. The output Q is active from t1+PT to t2, and again from t5+PT to t5. The elapsed time ET is zero during inactive periods and increases during active periods, starting from t1 and reaching zero again at t5.</p>	Nombre: TOF Función: Retardo a la desconexión Dato soportado: Duración (TIME) TOF_TIME

Elemento	Descripción
<p>FB_INSTANCE</p> <p>TP</p> <p>TIME</p> <p>BOOL IN Q - BOOL</p> <p>TIME PT ET - TIME</p> <p>IN</p> <p>t0 t1 t2 t3 t4 t5</p> <p>Q</p> <p>t0 t0+PT t2 t2+PT t4 t4+PT</p> <p>ET</p> <p>t0 t1 t2 t4 t5 0</p>	<p>Nombre: TP</p> <p>Función: Temporizador de impulsos</p> <p>Dato soportado: Duración (TIME)</p> <p>TP_TIME</p>

Tabla 4-13: Elementos Ladder. Bloque de Función - Temporizadores

5 Programas Ladder ejemplos

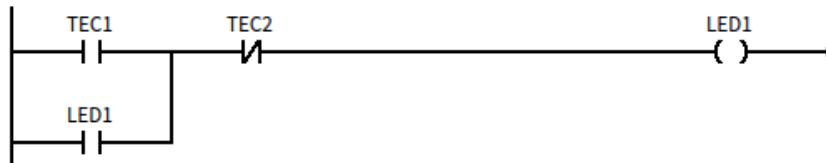
5.1 Circuitos de ejemplo

Los circuitos presentados en este capítulo se proponen a manera de ejemplos. Seguramente el lector podrá verificar su funcionamiento, experimentar con ellos, mejorarllos y/o usarlos como base para otros circuitos mas complicados.

5.1.1 Circuito con retención con contactos

En el siguiente ejemplo se muestra un circuito de retención con liberación. TEC 1 actúa como retenedor y TEC 2 como liberador.

Cuando TEC1 se cierra (se pulsa TEC 1) el LED1 se activa (led LED 1 se enciende) y el contacto asociado a la Bobina LED1 se cierra manteniendo el circuito cerrado a pesar de soltar el pulsador TEC 1. Mientras lo anterior ocurre el contacto TEC2 se mantiene cerrado porque no está energizado (el pulsador TEC 2 no está pulsado), cerrando el circuito. La Bobina se libera cuando se abre el contacto TEC2.

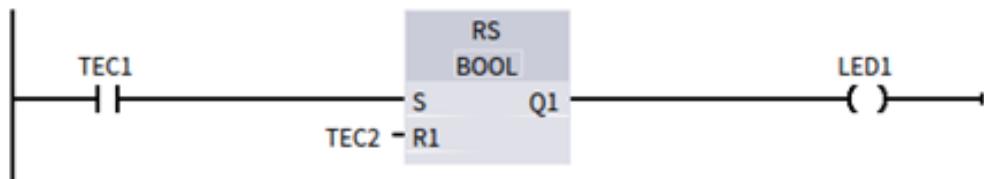


Este circuito puede ser usado, por ejemplo, para comandar con dos botones el encendido de una luz, una bomba, abrir y cerrar una válvula, etc.

5.1.2 Circuito con retención con biestables

En el siguiente ejemplo se ve un circuito de retención hecho con un biestable RS. TEC 1 actúa como retenedor y TEC 2 como liberador.

Cuando TEC1 se cierra (se pulsa TEC 1) el LED1 se activa (led LED 1 se enciende). El circuito permanece en ese estado hasta que la entrada R del biestable se energiza (cuando se cierra el pulsador TEC 2) y la salida Q1 se desenergiza apagando el LED 1.



Este circuito puede ser usado, por ejemplo, para comandar con dos botones el encendido de una luz, una bomba, abrir y cerrar una válvula, etc.

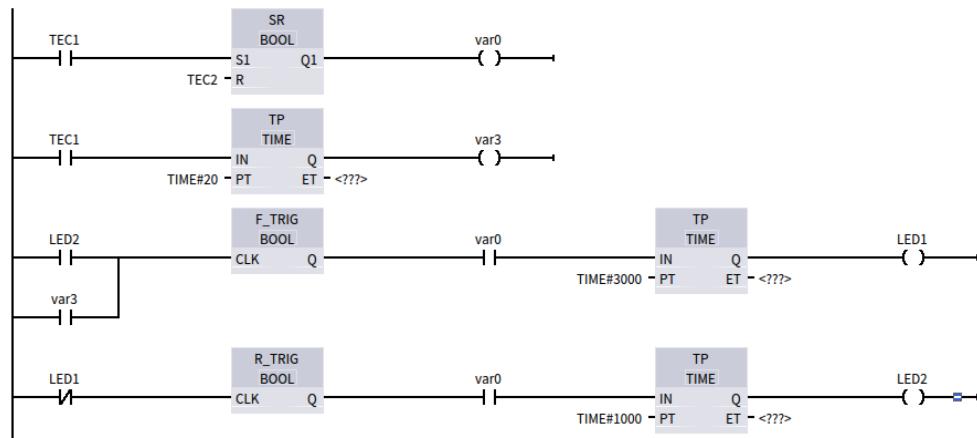
5.1.3 LEDs intermitentes

Para que este circuito funcione correctamente hay que crear la variable var3 bajo la categoría VAR y del tipo BOOL.

Este circuito enciende los leds LED 1 y LED 2 en forma alternativa. LED 1 enciende por 3 segundos, cuando se apaga se enciende LED 2 por 1 segundo, cuando se apaga vuelve a encenderse LED 1 y así sucesivamente. La secuencia comienza desde el LED 1 cuando se pulsa TEC 1 y se detiene (apagándose los dos leds) al pulsar TEC 2.

Inicialmente var0 y var3 están desenergizados. En estas condiciones var0 bloquea todos los disparos de los temporizadores TP.

Al pulsar TEC 1 var0 se mantiene energizado y var3 se energiza por 20 ms haciendo, con el flanco descendente, que el F_TRIG se dispare y el pulso a su salida Q llegue al primer TP encendiendo el LED 1 por 2 segundos. Al final de los 3 segundos el contacto normal cerrado de LED 1 se cierra originando un flanco ascendente en R_TRIG que dispara el segundo temporizador TP haciendo encender el LED 2 por 1 segundo. Al final del temporizado de un segundo la bobina LED2 se desenergiza originando un flanco descendente en la entrada del F_TRIG para que la secuencia vuelva a comenzar.



5-1: Circuito Ladder. Leds intermitentes

Este circuito puede ser usado, por ejemplo, para señalización haciendo encender alternativamente dos luces o, reemplazando una de las salidas (LED1 o LED2) por una variable interna, para hacer parpadear una luz.

5.1.4 Encendido escalonado por cuenta de señales

Este circuito enciende los LED 1, LED 2 o LED 3 dependiendo de la cantidad de pulsadores TEC 1, TEC 2 y TEC 3 que se pulsen simultáneamente. Si ningún pulsador está pulsado todos los leds estarán apagados. Si hay un pulsador pulsado (cuálquiera sea) se encenderá LED 1, si hay dos pulsadores pulsados (cualesquieras sean) se encenderá el LED 2, si hay tres pulsadores pulsados (cualesquieras sean) se encenderá el LED 3.

El bloque de movimiento (MOV) inicializa la variable var1 en cero.

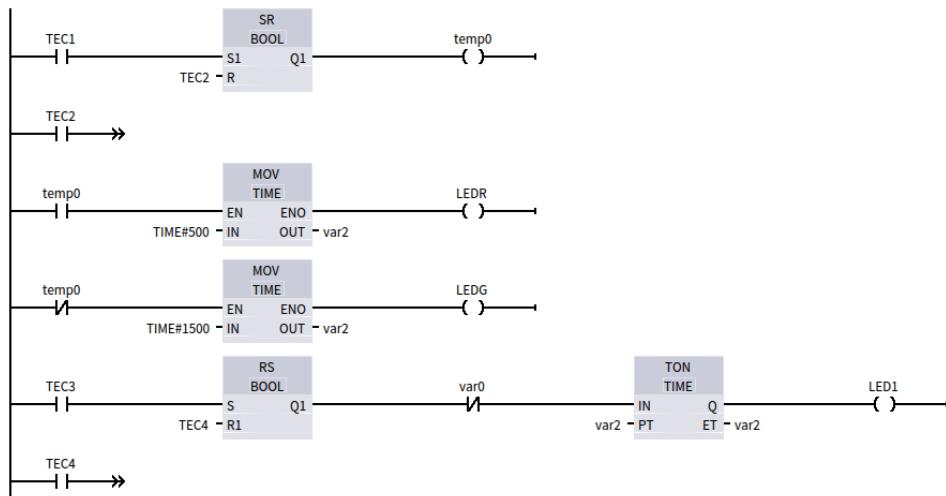
Por cada pulsador TEC pulsado se habilita un bloque ADD y se suma 1 a la variable var1. Luego de los bloques de suma, tres bloques de comparación por igual (EQ) comparan la variable var1 con 1, 2 o 3.

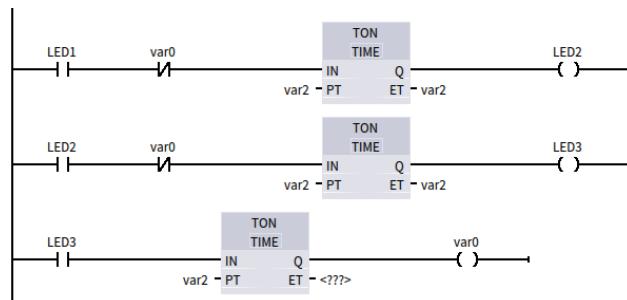
Si el primer bloque de comparación por igual (que compara con 1) da verdadero, el LED 1 se enciende.

Si el segundo bloque de comparación por igual (que compara con 2) da verdadero, el LED 2 se enciende.

Si el tercer bloque de comparación por igual (que compara con 3) da verdadero, el LED 3 se enciende.

Nótese que var1 tiene la función de acumular o contar cada uno de los pulsadores, TEC 1 a TEC 3, cerrados.





Círculo 5-2: Circuito Ladder. Encendido escalonado.

Este circuito puede ser usado, por ejemplo, para encender tres bombas dependiendo de la cantidad de válvulas abiertas o de los requerimientos de bombeo.

5.1.5 Secuenciador con permanencia

Este circuito hace una secuencia de encendido de los leds LED 1, LED 2 y LED 3 (manteniendo encendido el led anterior hasta terminar la secuencia, momento en el cual se apagan todos los leds). Adicionalmente cuando se enciende LED 1 se enciende en color Rojo (Red) el led RGB, cuando se enciende LED 2 se enciende en color Azul (Blue) el led RGB, cuando se enciende LED 3 se enciende en color Verde (Green) el led RGB.

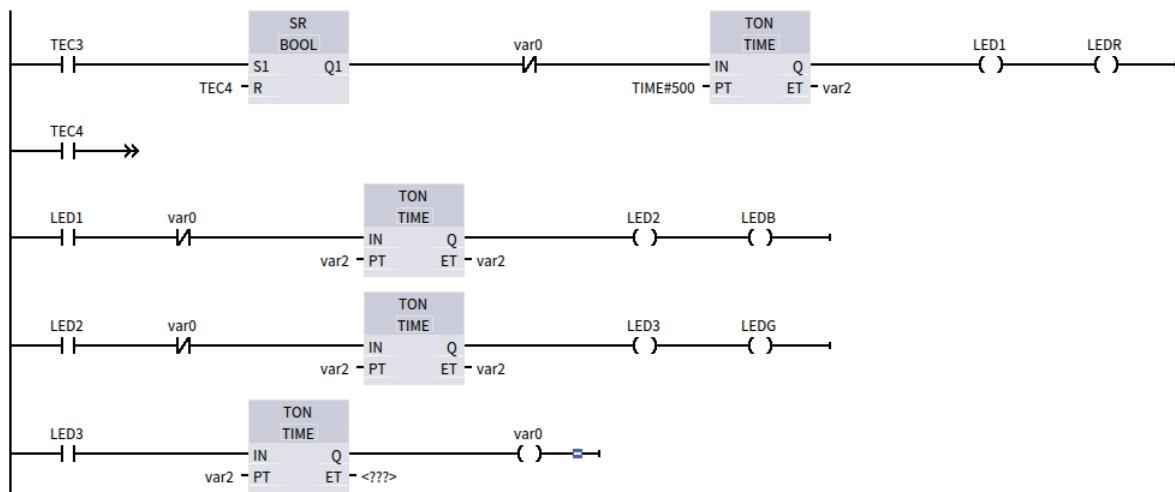
Inicialmente var0 tiene el valor cero y todos los pulsadores TEC están abiertos. La salida Q1 del biestable RS está en cero.

Cuando se pulsa TEC 3 la salida del biestable RS pasa a “1”, por intermedio del contacto normal cerrado var0 la entrada IN del primer temporizador TON se activa y comienza a contar de cero hasta 500 ms. Al llegar a los 500 ms su salida Q se activa, el LED 1 se enciende y el led Rojo del RGB se enciende. Cuando esto pasa, el contacto normal abierto LED1 se cierra y el segundo temporizador TON comienza a contar de cero hasta 500 ms. Al llegar a los 500 ms el LED 2 se enciende y también lo hace el led Azul del RGB. Cuando esto pasa el contacto normal abierto LED3 se cierra y el tercer temporizador TON comienza a contar de cero hasta 500 ms. Cuando la cuenta del tercer TON llega a 500 ms, la variable var0 pasa a “1” y todos los contactos normal cerrados de var0 se abren haciendo que todos los contadores pasen a cero con lo cual todos los leds se apagan y la secuencia vuelve a comenzar.

Si se pulsa TEC 4 el biestable RS se resetea, su salida Q1 pasa a “0” por lo cual LED 1 se desactiva y eso hace que, en cadena, el segundo, tercero y cuarto temporizador se desactiven.

La secuencia se reinicia si se pulsa TEC 3.

Nótese que var2 establece el tiempo de cada uno de los temporizadores a partir del primero, es decir que si se desea cambiar el tiempo de encendido de cada led, solo se debe cambiar el valor de la entrada PT del primer TON.



Círculo 5-3: Circuito Ladder. Secuenciador con permanencia

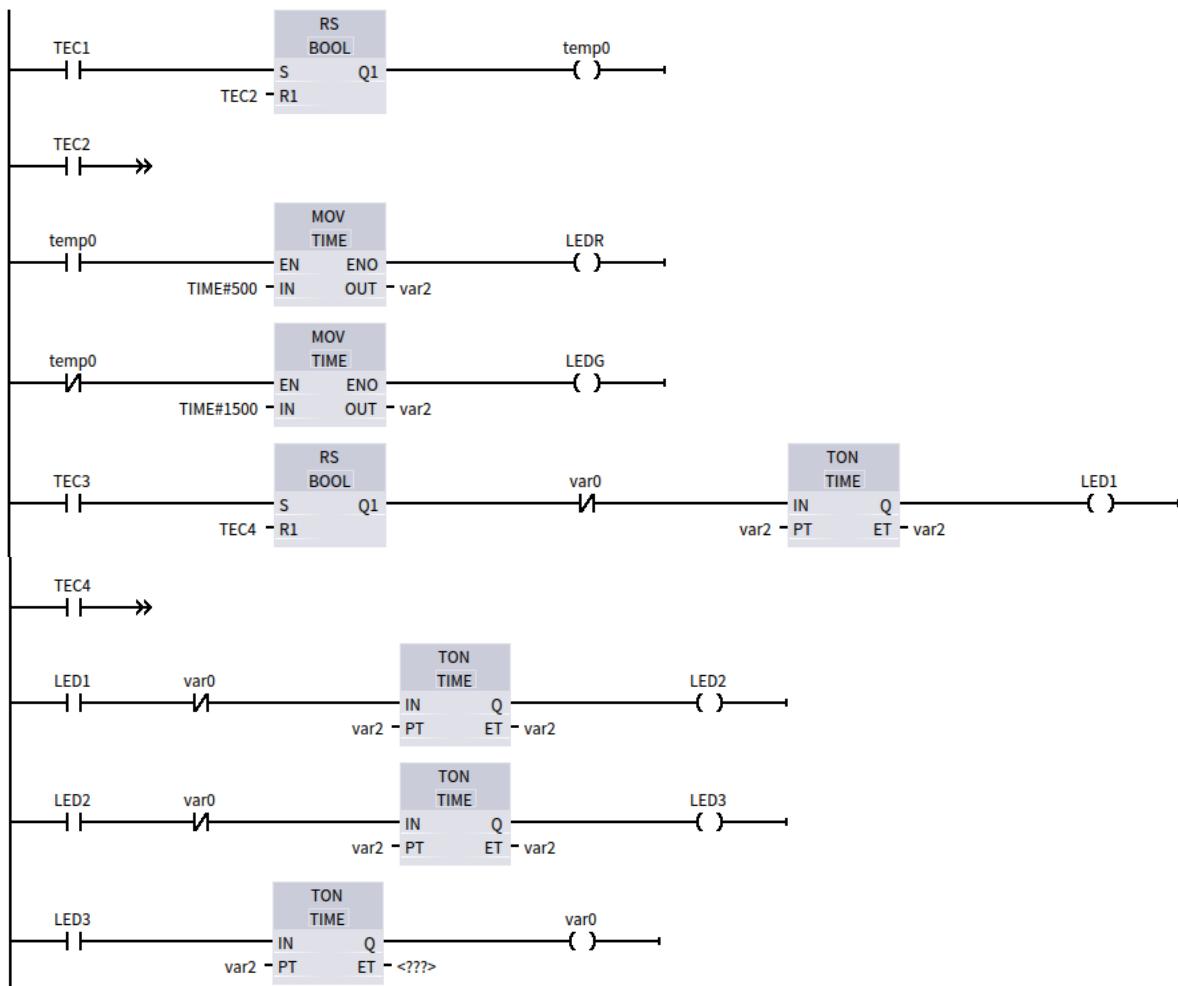
Este circuito puede ser usado, por ejemplo, para señalización de tres luces para indicar el sentido de circulación.

5.1.6 Secuenciador con permanencia y selección de velocidad

Este circuito hace una secuencia de encendido de los leds LED 1, LED 2 y LED 3 como en el Circuito 5-3, con el adicional que, en este caso, es posible cambiar la velocidad del secuenciado utilizando los pulsadores TEC 1 y TEC 2.

Al pulsar TEC 1 la salida Q1 del primer biestable RS pasa a valer "1" y la variable temp0 (variable booleana) pasa a "1". El contacto normal abierto de temp0 se cierra y el contacto normal cerrado se abre. En esta situación el primer bloque MOV se activa y la variable var2, que mantiene el tiempo de cada temporizador, toma el valor 500 (500 ms). Adicionalmente el led Rojo del RGB se enciende.

Al pulsar TEC 2 la salida Q1 del primer biestable RS pasa a valer "0" y la variable temp0 (variable booleana) pasa a "0". El contacto normal abierto de temp0 se abre y el contacto normal cerrado se cierra. En esta situación el segundo bloque MOV se activa y la variable var2, que mantiene el tiempo de cada temporizador, toma el valor 1500 (1500 ms). Adicionalmente el led verde del RGB se enciende.



Círculo 5-4: Círculo Ladder. Secuenciador con permanencia y selección de velocidad

Este circuito puede ser usado, por ejemplo, para señalización de tres luces para indicar el sentido de circulación con control de velocidad para dar una indicación adicional.

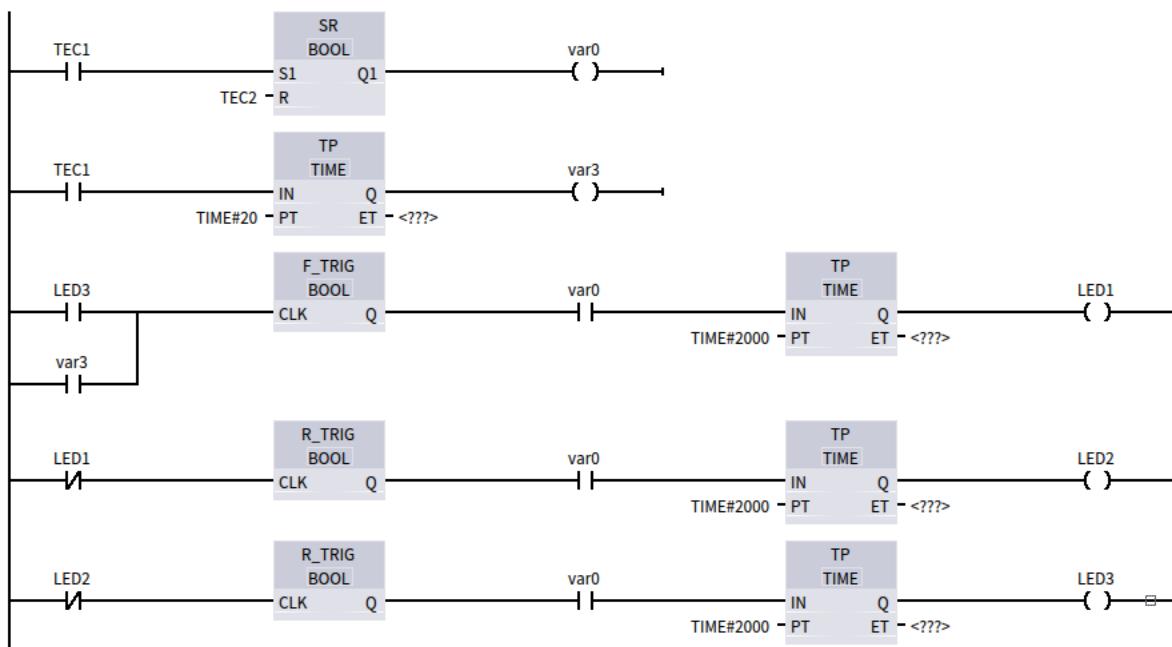
5.1.7 Secuenciador automático de tres leds sin permanencia

Para que este circuito funcione correctamente hay que crear la variable var3 bajo la categoría VAR y del tipo BOOL.

Este circuito hace una secuencia de encendido de LED 1, a continuación se apaga el LED 1 y se enciende el LED 2, a continuación se apaga el LED 2 y se enciende el LED 3, luego se apaga el LED 3 y se enciende el LED 1 y la secuencia vuelve a comenzar. La secuencia comienza desde el LED 1 cuando se pulsa TEC 1 y se detiene (apagándose todos los leds) al pulsar TEC 2.

Inicialmente var0 y var3 están desenergizados. En estas condiciones var0 bloquea todos los disparos de los temporizadores TP.

Al pulsar TEC 1 var0 se mantiene energizado y var3 se energiza por 20 ms haciendo, con el flanco descendente, que el F_TRIG se dispare y el pulso a su salida Q llegue al primer temporizador TP encendiéndolo el LED 1 por 2 segundos. Al final de los 2 segundos el contacto normal cerrado de LED 1 se cierra originando un flanco ascendente en el primer R_TRIG que dispara el segundo temporizador TP haciendo encender el LED 2. Al final de los 2 segundos el contacto normal cerrado de LED 2 se cierra originando un flanco ascendente en el segundo R_TRIG que el dispara el tercer temporizador TP haciendo encender el LED 3. Al final de los 2 segundos la bobina LED3 se desenergiza originando un flanco descendente en la entrada del F_TRIG para que la secuencia vuelva a comenzar.



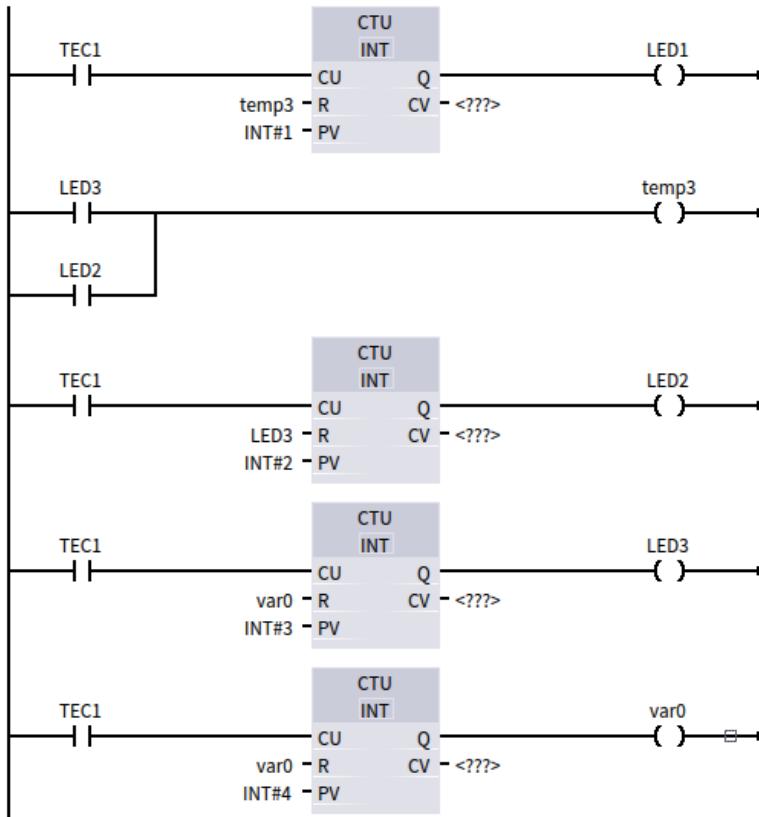
Círculo 5-5: Círculo Ladder. Secuenciador de tres leds sin permanencia

Este circuito puede ser usado, por ejemplo, para señalización de tres luces para indicar el sentido de circulación.

5.1.8 Secuenciador manual de tres leds sin permanencia

Para que este circuito funcione correctamente hay que crear la variable temp3, si no se encuentra generada, bajo la categoría VAR y del tipo BOOL.

Este circuito hace una secuencia de encendido manual pulsando TEC 1. El LED 1 se enciende, a continuación se apaga el LED 1 y se enciende el LED 2, a continuación se apaga el LED 2 y se enciende el LED 3, luego se apaga el LED 3 y todos los leds permanecen apagados hasta que se vuelva a pulsar TEC1 que hace que la secuencia vuelva a comenzar.



Círculo 5-6: Círculo Ladder. Secuenciador de tres leds sin permanencia

5.1.9 Secuenciador semiauto encendido/apagado de tres leds con permanencia

Para que este circuito funcione correctamente hay que crear las variable var3, var4 y var5, bajo la categoría VAR y del tipo BOOL.

Al pulsar TEC 1, este circuito hace una secuencia de encendido de LED 1, a continuación se enciende el LED 2 y después el LED 3. Los tres LEDs quedan encendidos hasta que se vuelve a pulsar TEC1, en este momento se apaga LED 3, luego se apaga LED 2 y finalmente LED 1. Para reiniciar la secuencia de encendido se debe pulsar nuevamente TEC 1.

Con TEC 2 se resetea la secuencia y todos los LEDs se apagan.

La variable var5 actúa como un reset controlado por TEC2 o la variable var4, que indica que la secuencia de apagado finalizó.

Al pulsar TEC 1 la salida del primer bloque RS se activa y se mantiene así hasta que var5 se active por pulsar TEC2 o porque var4 se activó porque la secuencia de apagado terminó.

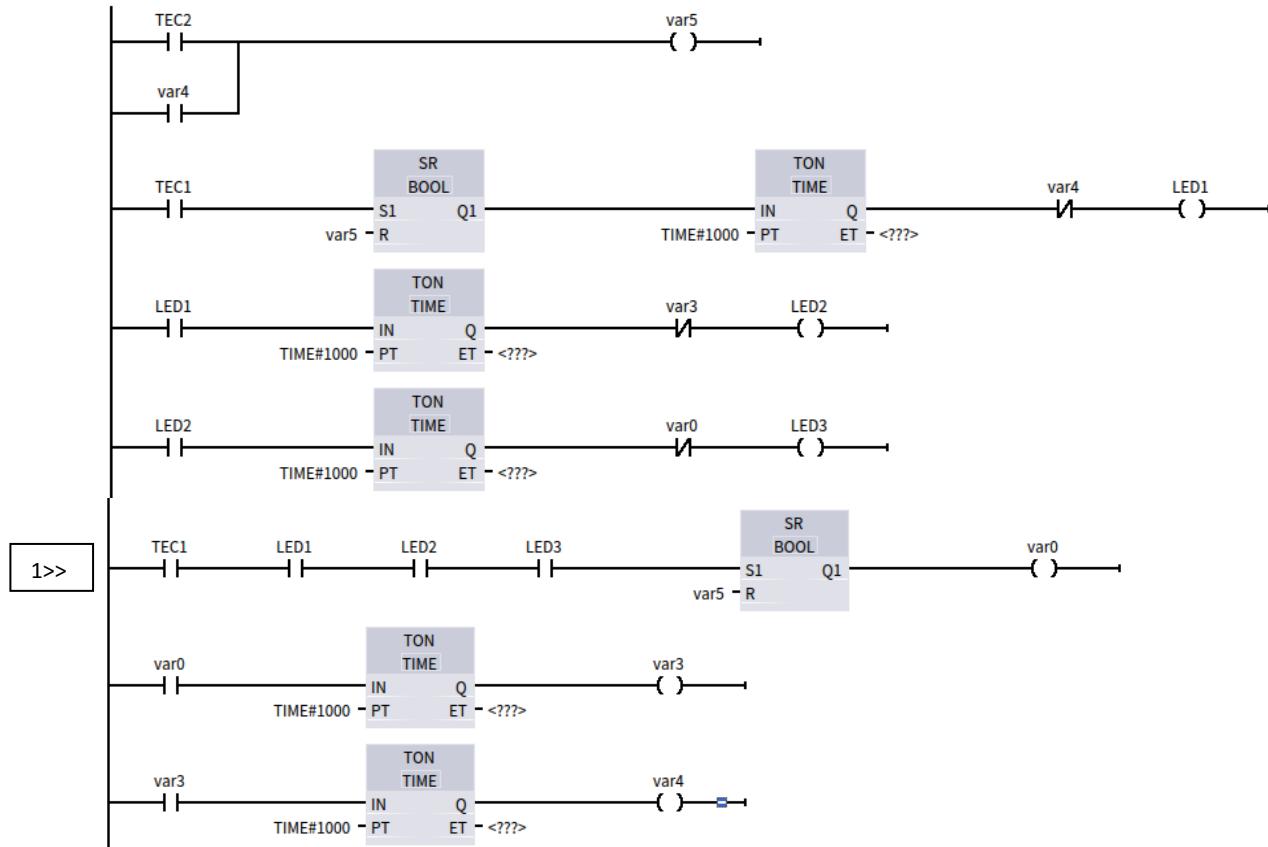
Cuando la salida del primer bloque RS se activa se inicia la cuenta de 1 segundo (1000 milisegundos) del temporizador a la conexión TON, por lo cual después de ese segundo se encenderá LED1 (ya que var4 está desactivada y el contacto normal cerrado var4 está cerrado). Cuando LED1 se enciende se inicia la cuenta del segundo bloque TON (también de 1 segundo), al cabo del cual se enciende LED2. Al encenderse LED2 el tercer bloque TON inicia el retardo de un segundo, encendiéndose el LED3 al cabo de dicho tiempo.

Una vez que LED1, LED2 y LED3 se encienden, quedan encendidos hasta que se vuelve a pulsar TEC1.

Cuando LED1, LED2 y LED3 están encendidos y se pulsa TEC1 la entrada Set del bloque SR se activa originando que la variable var0 se active. Al activarse var0, en el próximo scan, se apaga LED3 ya que un contacto negado de var0 está en serie con el LED3. Al activarse var0 se activa la entrada de otro bloque TON, que al cabo de un segundo va a activar var3 cuyo contacto negado está en serie con LED2, por lo cual este último se apagará. Al activarse var3 se activa la entrada de otro bloque TON, que al cabo de un segundo va a activar var4 cuyo contacto negado está en serie con LED1, por lo cual este último se apagará.

Adicionalmente cuando se activa var4 se activa var5, como se indicó al comienzo de la explicación, lo cual origina la activación de var5, cumpliendo el mismo efecto que pulsar TEC2.

Si se quiere usar otra tecla, por ejemplo TEC3, para comenzar el ciclo de apagado, se debe cambiar TEC1 por TEC3 en la línea “1>>”.



Circuito 5-7: Circuito Ladder. Secuenciador semiautomático de encendido/apagado de tres leds con permanencia

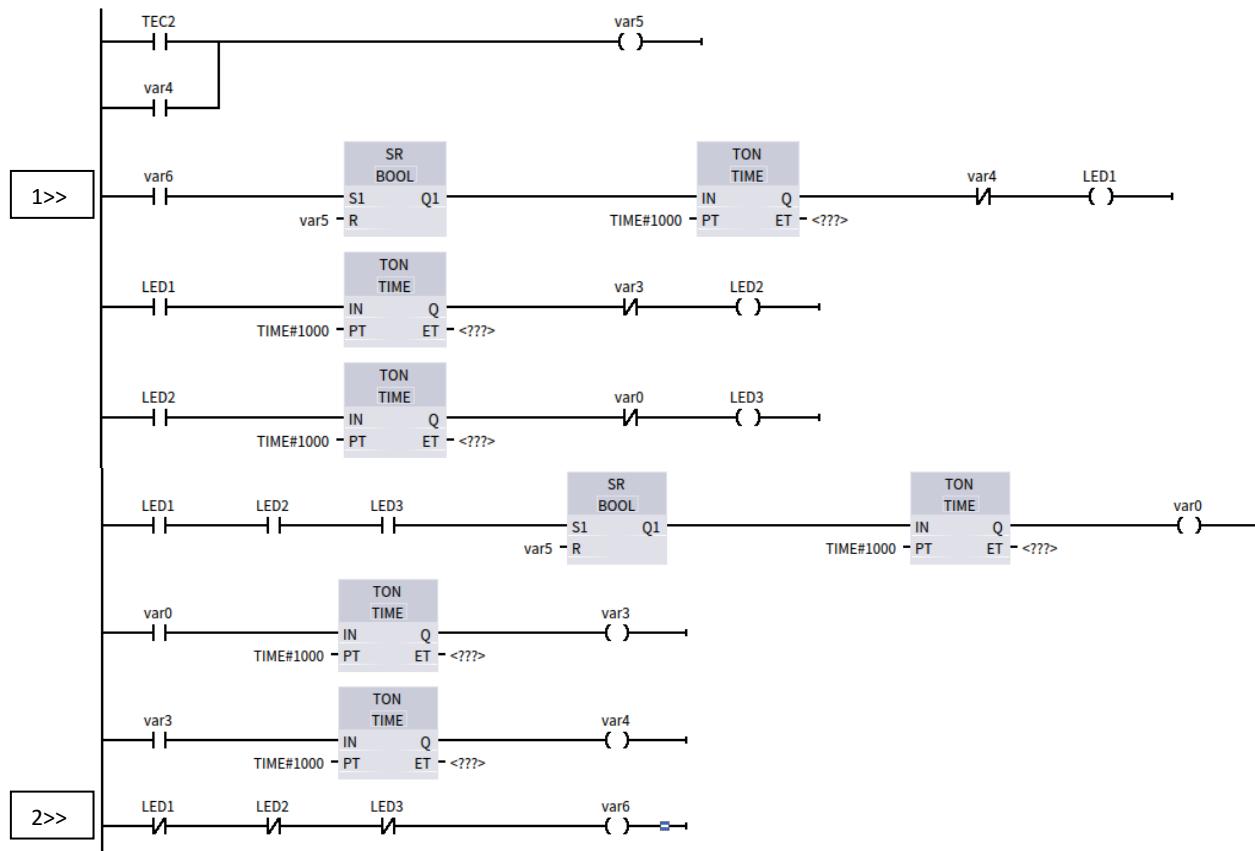
5.1.10 Secuenciador automático de encendido/apagado de tres leds con permanencia

Para que este circuito funcione correctamente hay que crear las variables var3, var4, var5 y var6, bajo la categoría VAR y del tipo BOOL.

Este circuito es similar al anterior con la única diferencia que en este la secuencia se inicia automáticamente, encendiéndo el LED1, luego el LED2 y finalmente el LED3, cuando los tres leds están encendidos se apaga el LED3, luego el LED2 y finalmente el LED1. Cuando los tres leds están apagados se reinicia automáticamente la secuencia.

Los únicos cambios en este circuito respecto al anterior son:

- Agregado de la línea “2>>”. Esta línea hace que cuando LED1, LED2 y LED3 estén apagadas se active var6 que reinicia la secuencia.
- El primer contacto de la línea “1>>” se cambia de TEC1 a var6 para que haga el reset cuando todos los leds estén apagados.



Círculo 5-8: Círculo Ladder. Secuenciador automático de encendido/apagado de tres leds con permanencia

5.1.11 Secuenciador auto de enc/apag de tres leds con permanencia e inicio/parada

Para que este circuito funcione correctamente hay que crear las variables var3, var4, var5, var6 y var7, bajo la categoría VAR y del tipo BOOL.

Este circuito es similar al anterior con la única diferencia que en este la secuencia se puede parar por TEC1 y reiniciar con TEC3. La secuencia se detiene cuando todos los leds están apagados. Es decir que si TEC1 se pulsa en el medio de la secuencia, se detendrá al llegar a al estado de los tres leds apagados.

Los únicos cambios en este circuito respecto al anterior son:

- Agregado de la línea “1>>”. Esta línea hace que la variable var7 se active cuando se pulsa TEC1 y se desactive cuando se pulsa TEC3.
 - En la línea “2>>” se agregó un contacto normal cerrado de la variable var7 para que el circuito se interrumpa, y la secuencia no vuelva a iniciarse, cuando este contacto se abra.
- Nota: el contacto var7 es normal cerrado para que la secuencia inicie en forma automática.

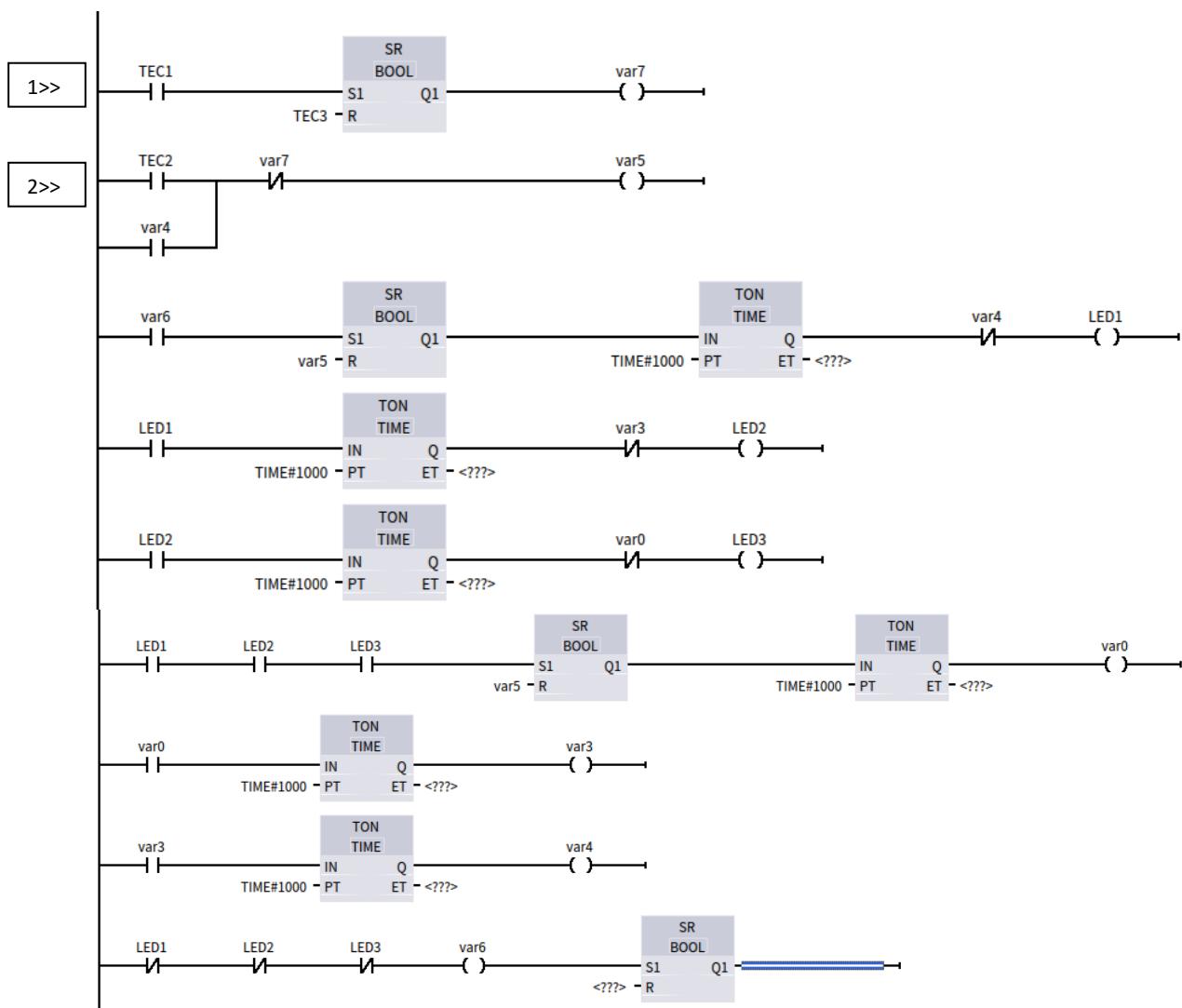


Figura 5-1: Circuito Ladder. Secuenciador auto de encendido/apagado de tres leds con permanencia e inicio/parada

6 Historial de versiones

En este apartado se describen las actualizaciones de cada una de las versiones de IDE4PLC.

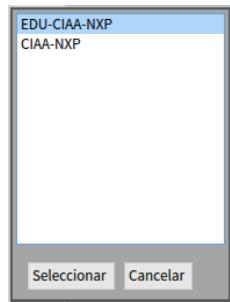
6.1 Versión 1.0.4

Fecha de publicación de la versión:

Versión de firmware requerido: 1.0.0 UPA LTS (release date 2015.12.23)

Esta versión incorpora las siguientes mejoras:

- 1) Soluciona algunos bugs relacionados con gráficos en Ladder y con la generación del programa que, en versiones anteriores, generaban inconsistencias en los programas.
- 2) Se actualizó la versión de Pharo de 3.0 a Pharo 4.0
- 3) Los identificadores de entradas de teclas (TEC) y de salida led (LED) se modificaron.
 - a. En versiones anteriores las teclas se identificaban como TEC_1, TEC_2, etc. Ahora se identifican como TEC1, TEC2, etc.
 - b. En versiones anteriores los led se identificaban como LED_1, LED_2, LED_R, etc. Ahora se identifican como LED1, LED2, LEDR, etc.
- 4) Se incorpora la opción de poder seleccionar la placa CIAA-NXP, además de la placa EDU-CIAA-NXP con la que se podía trabajar en las versiones anteriores.



- 5) Se incorporan algunas funciones para :



Generar código C.



Compilar código C.



Descargar programa.

- 6) Se reemplazaron algunos iconos por otros de representación mas clara:
Se reemplaza la primera línea del editor de POU



Se reemplaza la segunda línea del editor de POU





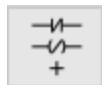
Se reemplazaron los iconos de “Guardar POU” y “Eliminar POU”.



“Generar Código C” y “Compilar Código C y Descargar al dispositivo”.



“Abrir declaraciones de variables”.



“Añadir componentes ladder”.