HERRAMIENTA SOFTWARE, BASADA EN EL ESTANDAR IEC 1131, SOPORTE PARA EL DESARROLLO DE AUTOMATISMOS LOGICOS SECUENCIALES, ASISTIENDO EL DISEÑO Y GENERANDO CODIFICACIÓN (AUTOLOGIC 1131).

Por:
JAIME LEONARDO CACERES CELY
MARCO FIDEL DÍAZ QUIJANO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD INGENIERIAS FISICO-MECANICAS
ESCUELA DE INGENIERIA DE SISTEMAS E INFORMATICA
BUCARAMANGA
2007

HERRAMIENTA SOFTWARE, BASADA EN EL ESTANDAR IEC 1131, SOPORTE PARA EL DESARROLLO DE AUTOMATISMOS LOGICOS SECUENCIALES, ASISTIENDO EL DISEÑO Y GENERANDO CODIFICACIÓN (AUTOLOGIC 1131).

Por:
JAIME LEONARDO CACERES CELY
MARCO FIDEL DÍAZ QUIJANO

Proyecto de Grado para Optar al Título de Ingeniero de Sistemas

Director
PROFESOR JORGE HERRERA
Escuela Ingeniería de Sistemas e Informática

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD INGENIERIAS FISICO-MECANICAS
ESCUELA DE INGENIERIA DE SISTEMAS E INFORMATICA
BUCARAMANGA
2007

A mis Padres, hermano y hermana
Inspiración y apoyo, constante e incondicional.
Siempre juntos, por encima de las dificultades de la vida.

Leonardo.

A los de corazón y razón fuertes y presentes, Padres, Hermana
Y Ana; a los de lucha, compañeros; y a los de
la distancia.

Marco.

AGRADECIMIENTOS

A todos aquellos que ayudaron a ser lo que somos, por que gracias a ellos lo somos. Seguiremos viviendo según las enseñanzas dadas y las que nos otorgarán con el paso compartido de nuestras vidas.

RESUMEN

TITULO: HERRAMIENTA SOFTWARE, BASADA EN EL ESTANDAR IEC 1131, SOPORTE PARA EL DESARROLLO DE AUTOMATISMOS LOGICOS SECUENCIALES, ASISTIENDO EL DISEÑO Y GENERANDO CODIFICACIÓN (AUTOLOGIC 1131)*.

Autores: Jaime Leonardo Cáceres Cely**
Marco Fidel Díaz Quijano***.

Palabras Clave: Automatización, Lenguajes, Diseño, Estándar.

DESCRIPCIÓN

La dependencia hacia los fabricantes de los elementos de control, en el campo de la automatización, ha generado dificultades, debidas; a que cada PLC**** contiene su propio lenguaje(s) de programación, lo que otorga a cada desarrollo una atadura interpretativa, de mantenimiento y de reutilización. Aunado a esto el campo de la automatización ha evolucionado desde una perspectiva empírica que ha imposibilitado el establecimiento de un lenguaje de diseño.

Con este panorama se desarrolla por la Comisión Internacional Electrotécnica (IEC), el estándar IEC 1131, que propone una metodología de trabajo basada en el desarrollo de los proyectos de automatización, en cinco reconocidos lenguajes, de forma que; el campo de la automatización encuentre un mecanismo de comunicación efectivo y logre con mayor agilidad y menor costo en la interpretación, mantenimiento y reutilización de los desarrollos.

En el contexto del IEC 1131 y enfatizando su tercera parte*****, se desarrolla la investigación para establecer sobre los Automatismos Lógicos Secuenciales (A.L.S) una forma de trabajo que fortalezca la labor de diseño y apoye el puente comunicacional requerido, así pues; la investigación muestra; que el lenguaje Esquema Secuencial de Funciones (SFC) por sus elementos constitutivos y su amplitud descriptiva para los A.L.S, posibilita la denominación de lenguaje de diseño y partir de él, se logra generar una conexión con los otros lenguajes posibilitando la generación de la codificación en ellos a partir de lo diseñado.

Con este resultado de la investigación se construye la herramienta Autologic 1131, que es la portadora del sustento conceptual expuesto anteriormente, y que muestra una herramienta que asiste el diseño en SFC, permite la validación de lo diseñado en sintaxis, congruencia y genera la codificación respectiva en los lenguajes estandarizados por la IEC 1131-3 Lista de Instrucciones (IL), Texto Estructurado (ST), Diagrama de Bloques Funcionales (FBD) y Diagrama de Contactos (LD).

* Trabajo de Grado
** Facultad Ingeniería Físico-mecánicas, Escuela de Ingeniería de Sistema e Informática, Dir. Jorge Herrera.
*** Facultad Ingeniería Físico-mecánicas, Escuela de Ingeniería de Sistema e Informática, Dir. Jorge Herrera
**** PLC, Control Lógico Programable, hace referencia para este caso en particular al elemento hardware que gobierna la interfase entre lo programado en PC y la ejecución física del automatismo
***** La tercera parte del estándar es de gran importancia debida a que es en esta parte donde se expone la sintaxis, los elementos y la forma de trabajo de los lenguajes.

ABSTRACT

TITLE: SOFTWARE TOOL, CRADLE IN STANDARD IEC 1131, SUPPORT FOR THE DEVELOPMENT OF SEQUENTIAL LOGICAL AUTOMATISM, ATTENDING THE DESIGN AND GENERATING CODIFICATION (AUTOLOGIC 1131)*.

Authors: Jaime Leonardo Cáceres Cely**
Marco Fidel Díaz Quijano***.

Key Words: Automatization, language, design, standard.

DESCRIPTION

The dependency towards the manufacturers of the control elements, in the field of the automatization, has generated difficulties, which had; to that each PLC **** contains their own language (s) of programming, which grants to each development an interpretation fastening, of maintenance and reusability. Combined to this the field of the automatization it has evolved from an empirical perspective that has disabled the establishment of a design language.

With this panorama, is developed by International Electrotechnical Commission (IEC), the standard IEC 1131, that proposes a methodology of work based on the development of the automatization projects, in five recognized languages, so that; the field of the automatization finds a communication mechanism effective and obtains with greater agility and minor cost in the interpretation, maintenance and reusability of the developments.

In the context of IEC 1131 and emphasizing his third part*****, is developed the investigation to establish on Sequential Logical Automatism (S.L.A.) a work form that fortifies the design work and supports the required communicational bridge, therefore; the investigation shows; that the language Sequential Function Chart (SFC) by its constituent elements and their descriptive amplitude for the S.L.A., makes possible the denomination of design language and to start off of him, is managed to generate a connection with the other languages making possible the generation of the codification in them from the designed thing.

With this result of the investigation the tool Autologic 1131 is constructed, that is the carrier of the exposed conceptual sustenance previously, and

that shows a tool that attends the design in SFC, allows the validation of the designed thing in syntax, congruency and generates the respective codification in the languages standardized by the IEC 1131-3 Instruction List (IL), Structured Text (ST), Function Block Diagram (FBD) and Ladder Diagram (LD).

---

* Promotion Paper

** Facultad Ingeniería Físico-Mecánicas, Escuela de Ingeniería de sistemas e informática, Dir. Jorge Herrera.

*** Facultad Ingeniería Físico-Mecánicas, Escuela de Ingeniería de sistemas e informática, Dir. Jorge Herrera.

****PLC, Programmable Logical Control, makes reference for this case in individual to the element hardware that governs the interphase between programmed in PC and the physical execution of the automatism

*****The third part of the standard is of great importance due to that it is in this part where it exposes the syntax, the elements and the form of work of the languages.

CONTENIDO

INDICE DE TABLAS.

INDICE DE FIGURAS.

INTRODUCCIÓN

Es el campo de la automatización industrial, un escenario de crecimiento acelerado, y en el cual se han realizado avances importantes, en la consecución de un mejor rendimiento del conocimiento y de la tecnología, evidenciado esto; en los elementos de control cada vez mas robustos y con mayor funcionabilidad, de igual forma; la construcción de metodologías de trabajo que aportan a la solución de problemáticas, como la dificultad de comunicación, mantenimiento y reutilización de los proyectos de automatización. Que por la dependencia a los fabricantes se presentan.

En este escenario, surge un estándar emitido por la Comisión Internacional Electrotécnica (IEC) denominado IEC 1131, donde se muestra una metodología de trabajo para automatización industrial, que: Primero, reconoce el avance de las casas fabricantes, en relación con los lenguajes utilizados por los PLC's, y articula estos desarrollos, incluyendo los lenguajes mas utilizados en el mundo y dictando para ellos, la sintaxis, los elementos y su forma de trabajo. Segundo, posibilita, al independizar los desarrollos de los proyectos de automatización, la implementación en cualquier dispositivo PLC o en combinación de PLC's de diferentes fabricantes. Con esto; surge solución a una problemática, que permite seguir en expansión el campo de la automatización.

Ya en este contexto, este trabajo se interesa en investigar para los Automatismos Lógicos Secuenciales (A.L.S), la aplicabilidad del estándar desde la perspectiva proyectiva, de materializar el avance teórico en una herramienta que brinde la expansión de la norma, aunando a esto; la necesidad de indagar sobre el fortalecimiento del diseño en automatización industrial. Encontrando; que asumiendo el análisis del estándar en su tercera parte, donde se consigan los lenguajes y los cuales son Esquema Secuencial de Funciones (SFC), Lista de Instrucciones (IL), Texto Estructurado (ST), Diagrama de Bloques Funcionales (FBD) y Diagrama de Contactos (LD), el primer lenguaje por sus elementos constitutivos y por su capacidad descriptiva para los A.L.S, puede ser denominado lenguaje de diseño, y aunado a esto se encuentra la capacidad de conexión de lo diseñado con los demás lenguajes, lo cual posibilita la codificación en IL, ST, FBD y LD a partir de lo diseñado en SFC.

Con estos resultados, se materializa, Autologic 1131; que es la herramienta software, que condensa el sustento conceptual encontrado,

generando así, una herramienta software que asiste el diseño en SFC, valida en sintaxis y congruencia lo diseñado, y genera la codificación en los demás lenguajes estandarizados según la norma IEC 1131-3.

Es este un recuento del proceso de investigación que posibilito la construcción conceptual y software de una forma de trabajo para los A.L.S que fortalece el diseño, genera codificación y permite para este contexto aplicar el estándar y sus implicaciones.

1. PERESENTACIÓN DEL PROYECTO

El presente proyecto, esta enmarcado en un contexto de investigación, que en el ámbito de la automatización industrial, propone; fortalecer la labor del diseño, al igual que fortalecer una metodología que intenta dar solución a la necesidad de comunicación estándar en el campo de la automatización industrial, esto para los automatismos lógicos secuenciales, es decir; el presente proyecto expone un trabajo que, basándose en el estándar IEC 1131, ofrece una herramienta software que pone en prioridad el diseño de automatismos lógicos secuenciales y a partir de éste, genera codificación. Se entiende que el diseño es el enlace comunicacional estándar que posibilita en el campo de la automatización superar las dificultades de mantenimiento, interpretación y reutilización de automatismos.

Para el desarrollo del presente trabajo se contemplan cuatro capítulos donde se expone en primer lugar los objetivos y el problema planteado, seguido de una exposición del estándar IEC 1131, para continuar con la exposición de la investigación e interpretación realizada en este proyecto y finaliza con la muestra y explicación de la herramienta software Autologic 1131 versión 1.0.0.0.

1.1OBJETIVOS.

1.1.1 Objetivo General.

Desarrollar una herramienta software basada en el estándar IEC 1131, que permita asistir el diseño y a partir de este generar la codificación del automatismo lógico secuencial, disminuyendo el tiempo y aumentando la fiabilidad del diseño, posibilitando un adaptable sistema controlado resultante; en consecuencia una disminución en el gasto y el tiempo de implementación, con referencia a la forma de trabajo manual.

1.1.2 Objetivos Específicos.

- Generar una herramienta que permita:

- Asistir el diseño de automatismos lógicos secuenciales soportados en la sintaxis y semántica del SFC[1] (Grafico Funcional Secuencial) recogido en el estándar IEC 1131-3.
- A partir del diseño generar la codificación en los cuatro lenguajes del estándar IEC 1131-3, que son LD (Lenguaje de contactos), IL (Lenguaje de instrucciones), ST (Texto estructurado) y FBC (Diagrama de bloques funcionales).
- Generación de archivos contenedores tanto del diseño (SFC), como del código generado por la herramienta en los cuatro lenguajes (LD, IL, ST, FBC).

- Posibilitar la validación del diseño SFC en dos etapas, en primera instancia lo relacionado con la sintaxis[2] y en segunda instancia con la congruencia[3] del diseño.
- Realizar la verificación del código generado por la herramienta, mediante la trascripción del mismo en el software controlador del PLC específico a usar; para dar funcionamiento al montaje final.
- Rescatar el carácter interdisciplinario que fundamenta la labor del ingeniero de sistemas, desde la incidencia del presente proyecto en los campo de la automatización, fortaleciendo la labor investigativa en una temática que si bien no es nuestra, no es ajena a nosotros, queriendo con esto mostrar que el ingeniero de sistemas puede ir más allá del desarrollo software.

1.1.3 Planteamiento del problema.

En el campo de la automatización desde los múltiples años de desarrollo, se ha encontrado que la dinámica en el diseño de este tipo de proyectos responde al pragmatismo y empirismo de quienes los ejecutan, es por tanto que entendiéndose la dificultad en la interpretación, continuidad y actualización de los desarrollos realizados se han generado esfuerzos que intentan avanzar en la solución de este tipo de dificultades sacando al mercado numerosos lenguajes y metodologías que pretendían poner en sintonía a los programadores, diseñadores y operarios. Encontrado en este avance un nuevo obstáculo. La multiplicidad de lenguajes pues dependían de los fabricantes de los implementos a usar en el montaje, y continuando sin solución eficaz la labor de diseño en tanto no se consolidaba una estrategia que tuviese un lenguaje común. Después de muchos intentos y de recoger experiencias fructíferas se encuentra un estándar que primero: Define una metodología para la descripción de los

[1] SFC basado en Grafcet y las redes Petri, es un conjunto de elementos gráficos y normas de interconexión para el diseño de automatismos secuenciales.
[2] Se entiende por sintaxis la correcta disposición de los elementos de diseño del SFC.
[3] La congruencia hace referencia a que lo diseñado no posea operaciones problemáticas en la ejecución del automatismo.

automatismos basándose en los elementos comunes[4] de desarrollo de automatismos existentes, donde se encuentran contenidos avances como el del estándar IEC 60848[5]. Segundo: Se dan normas sintácticas y semánticas claras sobre los lenguajes de programación para los automatismos al igual que se define los lenguajes a utilizar recogiendo aquí los más utilizados en el mundo. Con este panorama se avanza en términos conceptúales para la solución de la problemática expuesta y es entonces la intención de este proyecto seguir en esta línea de desarrollo aportando una herramienta que permita materializar aun mas la intención, es por tanto que atendiendo la necesidad evidenciada del fortalecimiento a la labor del diseño de automatismos, especialmente los automatismos lógicos secuenciales y posibilitando la generación de código a partir del diseño realizado, se plantea el desarrollo de la herramienta fundamentada en el estándar IEC 1131.

[4] Comisión internacional de electrotécnica. IEC 61131-3, suiza. IEC 1995.
[5] Estándar que estructuró todo lo relacionado con el grafcet

2. ESTÁNDAR IEC 1131

2.1 INTRODUCCIÓN

El estándar IEC 1131, es un documento que expone una forma de trabajar en automatización industrial, recogiendo para esto, cinco lenguajes de programación, escogidos desde el conocimiento de la Comisión Internacional Electrotécnica (IEC) en el campo de la automatización y la utilización de estos en la mayoría de los PLC's; así pues, tenemos que el estándar es un documento que conteniendo los lenguajes, enuncia para cada uno su sintaxis, sus fortalezas y su mejor aprovechamiento.

Aunado a lo anteriormente expuesto, se encuentran otras partes en el estándar, que recoge desde las definiciones hasta cómo desarrollar la documentación de un proyecto de automatización, en este capitulo se expondrá de forma sintetizada todo el estándar IEC 1131, sin embargo; se aclara que para el presente trabajo si bien la globalidad del estándar es importante, se focalizan los esfuerzos en la tercera parte del mismo.

2.2 PROPUESTA DEL ESTÁNDAR.

Como se exponía en el primer capitulo de este trabajo, la automatización industrial ha sido desarrollada en con contexto de dependencia a los lenguajes definidos por los fabricantes de los PLC's, dejando esto como consecuencia: Primero, una atadura al PLC en el que se realizo el desarrollo. Segundo: la reutilización de ese proyecto no será posible en un dispositivo diferente al programado y finalmente la interpretación estará sujeta al lenguaje defino por la casa fabricante, es entonces; el estándar la respuesta a la necesidad de un mecanismo de comunicación efectivo en la automatización industrial, tratando de que la existencia de múltiples PLC's no siga siendo un obstáculo para el mantenimiento, interpretación y reutilización de los proyectos, debido a la multiplicidad de lenguajes, por el contrario se exhorta desde la IEC a que todos los fabricantes de PLC's acojan el estándar IEC 1131 y de esta forma saldar la dificultad.

2.2.1 Ventajas del estándar

- Reduce el esfuerzo humano en entrenamiento, depuración, mantenimiento y consultaría.
    - Una vez aprendido se puede utilizar en todos los sistemas
- Permite crear software reutilizable, minimiza:
    - El tiempo de desarrollo
    - El esfuerzo de codificación
    - Los errores de compilación y ejecución
- Coordina eficazmente diferentes componentes desde distintas localizaciones, compañías o proyectos.
    - Amplio campo de aplicación.
- Aumenta la conectividad, facilita la distribución del control.

2.2.2 Claves de desarrollo del estándar.

- Software estructurado a través de diseño, proyectos, tareas, programas y bloques.
- Descripción del comportamiento secuencial complejo de un proceso a través del SFC.
- Encapsulación del software a través de unidades de organización de programas, estructuras y tipos de datos.

2.3 COMPOSICIÓN DEL ESTÁNDAR.

El estándar esta compuesto por cinco partes que expresan su contenido e intención para lo cual fue creado y están expuestas de forma sintetizada así:

2.3.1 Parte 1. Información general

Contiene las definiciones generales usadas en la norma, de igual forma se identifican las características típicas de los sistemas de autómatas programables.

Algunas Definiciones contenidas son:

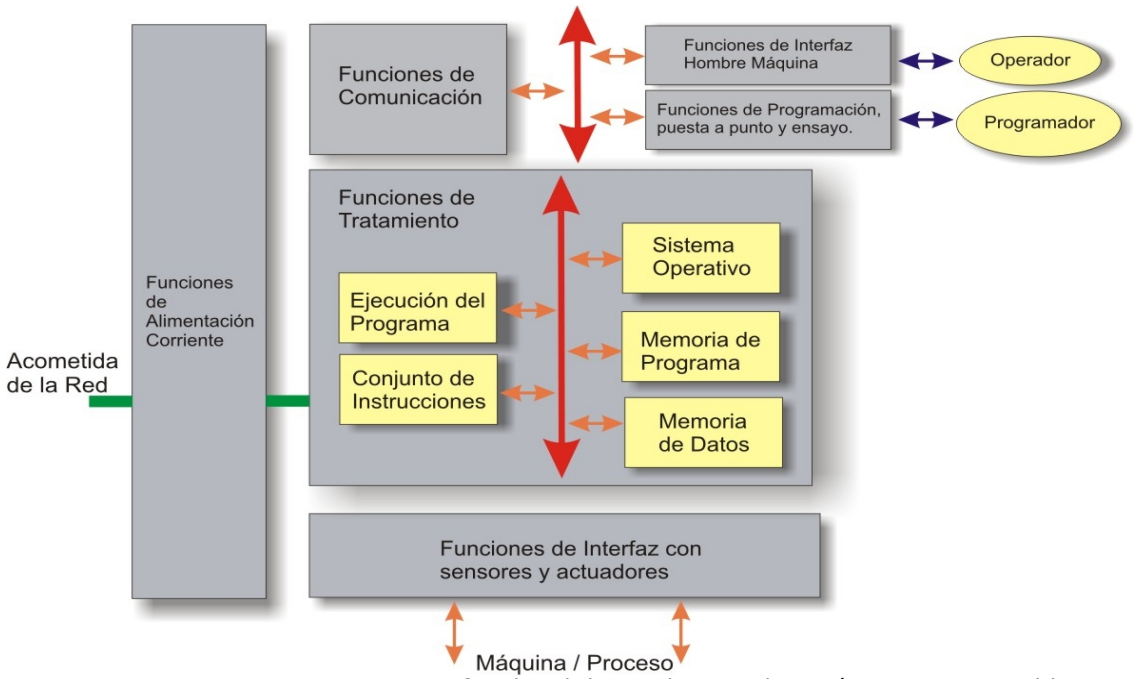
Programación de aplicación, lista de asignaciones, sistema automatizado, disponibilidad, datos boléanos, bus, componente, conexión/desconexión, lista de referencia cruzada, equipo de comunicación de datos (DCE), equipo terminal de datos (DTE), ejecución, parada de seguridad, imagen de entrada/salida, entrada, instrucción, fallo interno, diagrama de escalera , lenguaje (FBD, IL, LD, ST, SFC), sistema de control lógico, procesador principal (MPU), interfaces hombre-máquina (MMI), tiempo

medio entre fallos (MTBF), red, en línea, salida, programa, periférico, barras de corriente, autómata programable (AP ó PLC), equipo de programación y puesta a punto (PADT), estación de entrada/salida remota (RIOS), reinicio, sistema de control secuencial, transmisión de datos en serie, etc.

Dentro de la estructura funcional de un sistema de autómata programable tenemos:

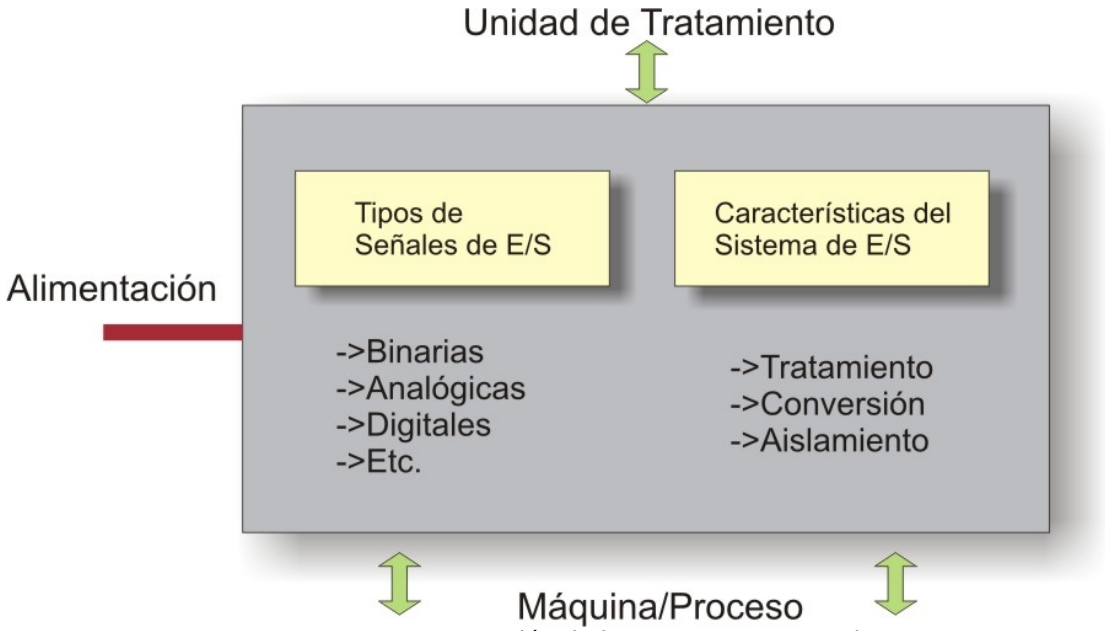
- Función de tratamiento de la señal.
- Función de interfaz con los sensores y actuadores.
- Función de comunicación.
- Función de interfaz hombre-máquina.
- Funciones de programación, puesta a punto, ensayo y documentación.
- Funciones de alimentación de corriente

Figura 1. Estructura funcional de un sistema de autómata programable



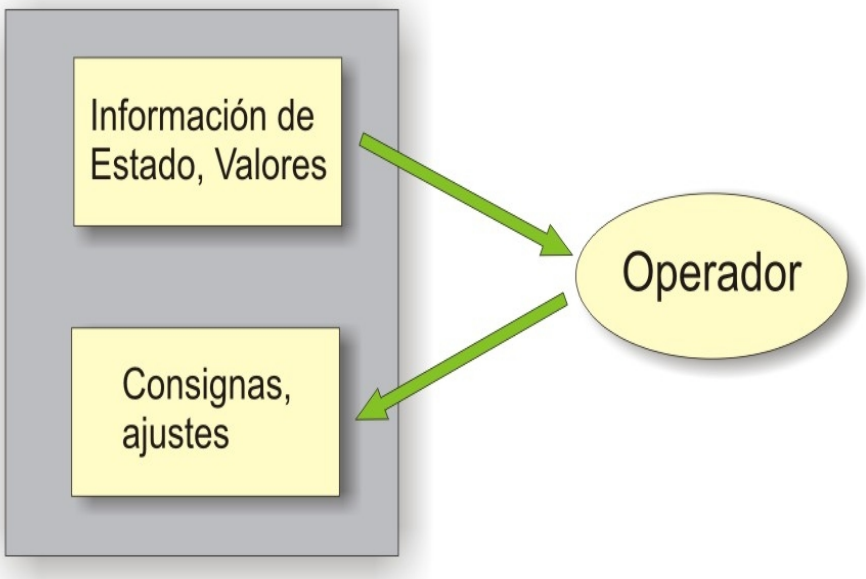
Fuente: IEC 61131-1, Estructura funcional de un sistema de autómata programable

Figura 2. Función de los sensores y actuadores



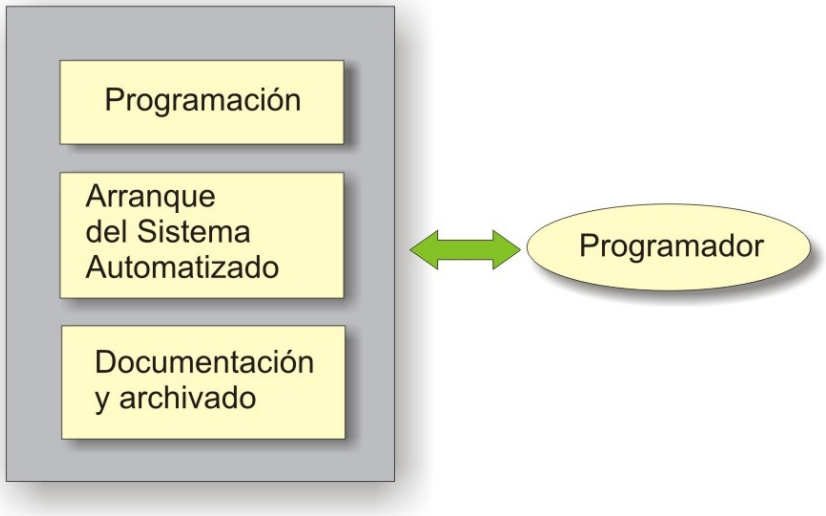
Fuente: IEC 61131-1, Función de los sensores y actuadores

Figura 3. Función de interfaz hombre-maquina MMI



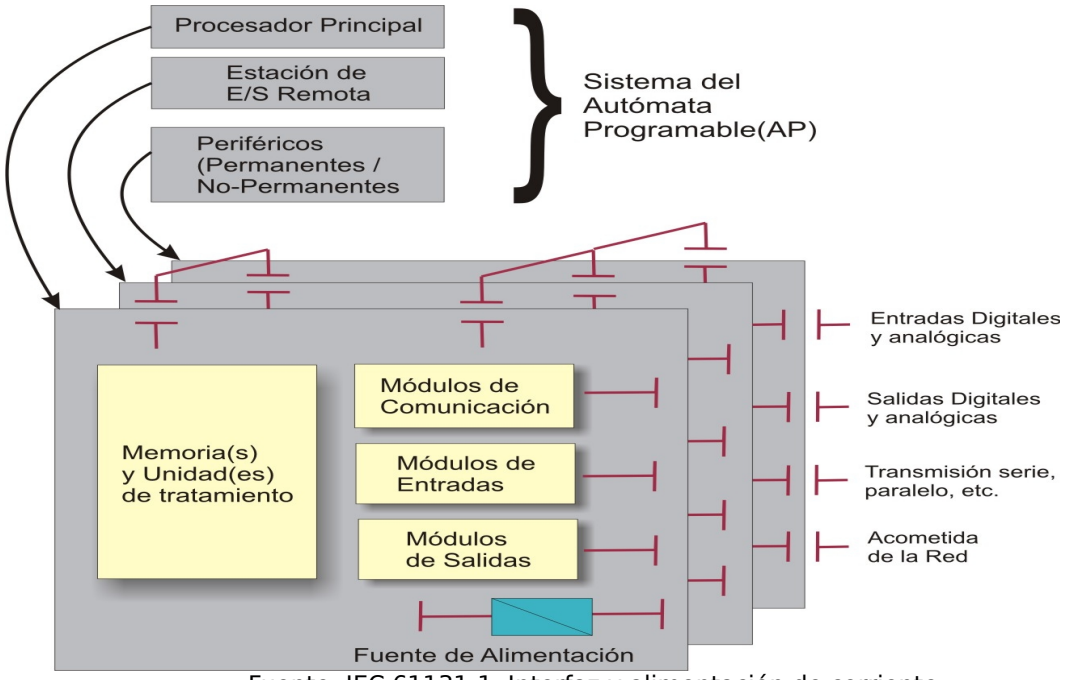
Fuente: IEC 61131-1, Función de interfaz hombre-maquina MMI

Figura 4. Programación, puesta a punto, ensayo y documentación.



Fuente: IEC 61131-1, Programación, puesta a punto, ensayo y documentación

Figura 5. Interfaz y alimentación de corriente



Fuente: IEC 61131-1, Interfaz y alimentación de corriente

2.3.2 Parte 2. Especificaciones y ensayos de equipos

Contiene los requisitos mecánicos, eléctricos y funcionales para los autómatas programables y los periféricos correspondientes, así como las

condiciones de servicio, almacenamiento y transporte aplicables. Contiene además la información que ha de suministrarse por el fabricante y los métodos y procedimientos de ensayo que han de utilizarse para la comprobación del cumplimiento de los requisitos por parte de los autómatas programables y sus periféricos.

Se tiene entonces que las condiciones de servicio y requisitos del entorno físico son:

- Condiciones del entorno físico: temperatura, humedad, contaminación, inmunidad a la corrosión y altitud.

- Condiciones de servicio y requisitos eléctricos: alimentaciones, ruido eléctrico, sobretensiones, etc.

- Condiciones de servicio y requisitos mecánicos: Vibraciones, choque y caída libre.

- Condiciones de servicio especiales. Polvo, humo, partículas radiactivas, vapores, sales, insectos, pequeños animales, etc.

- Requisitos para el transporte y almacenaje: temperatura, presión atmosférica y humedad relativa.

Los Requisitos eléctricos son:

- Alimentación de corriente alterna (c.a.) y continua (c.c.).
- E/S digitales.
- E/S analógicas.
- Interfaces de comunicación.
- Procesador(es) principal(es) y memoria(s) del sistema AP.
- Estaciones de entrada/salida remota (RIOS).
- Periféricos: PADT, TE, MMI.
- Inmunidad al ruido y ruido emitido.
- Propiedades dieléctricas.
- Autodiagnósticos y diagnósticos.

Requisitos mecánicos

- Protecciones contra el riesgo de choques eléctricos.
- Requisitos de distancias en el aire y líneas de fuga.
- Requisitos de inflamabilidad para materiales aislantes.

- Envolvente.
- Requisitos mecánicos de los materiales de conexión.
- Disposiciones para la tierra de protección.
- Tierra funcional.
- Cables y conectores de interconexión.
- Conexión/desconexión de unidades desmontables.
- Requisitos de la batería.
- Marcado e identificación.

La Información que debe facilitar el fabricante es:

El fabricante deberá facilitar a los usuarios la información necesaria para la aplicación, proyecto, instalación, puesta en marcha, funcionamiento y mantenimiento del sistema de autómata programable. Adicionalmente el fabricante puede ocuparse de la formación del usuario.

- Tabla resumen con la información que se debe facilitar.
- Tipo y contenido de la información escrita: Catálogos y hojas de características, manuales de usuario y documentación técnica.
- Información relativa al cumplimiento de esta norma.
- Información relativa a la fiabilidad.
- Información relativa a la seguridad.

Los ensayos y verificaciones son:

- Se define como ha de verificarse la conformidad del autómata programable y sus periféricos correspondientes con los requisitos fijados en las partes 1 y 2 de la norma.
- Estos ensayos no se refieren a los métodos de aplicación de los autómatas programables, para cumplir con los requisitos del sistema automatizado.
- Se dividen en ensayos de tipo y ensayos de rutina.

Los ensayos de tipo son:

- Equipos a ensayar.
- Procedimientos de verificación.
- Condiciones generales para los ensayos.
- Ensayos climáticos, mecánicos y eléctricos.
- Verificación de las características de la alimentación de c.a. y c.c.
- Verificación de las características de entrada/salida.
- Verificación de las características del procesador principal.

- Verificación de las estaciones de E/S remotas.
- Verificación de las características de los periféricos.
- Verificación del autodiagnóstico y diagnóstico.

Los ensayos de rutina son:

- Ensayo estándar de rigidez dieléctrica
- Ensayo de continuidad de la tierra de protección.

2.3.3 Parte 3. Lenguajes de programación

Contiene la definición de los lenguajes de programación de uso más común, las reglas sintácticas y semánticas. Y se divide para su compresión en elementos comunes y los lenguajes, interpretación esta de lectura denominada Top Down.

A. Elementos Comunes.

- Tipos de datos y variables.

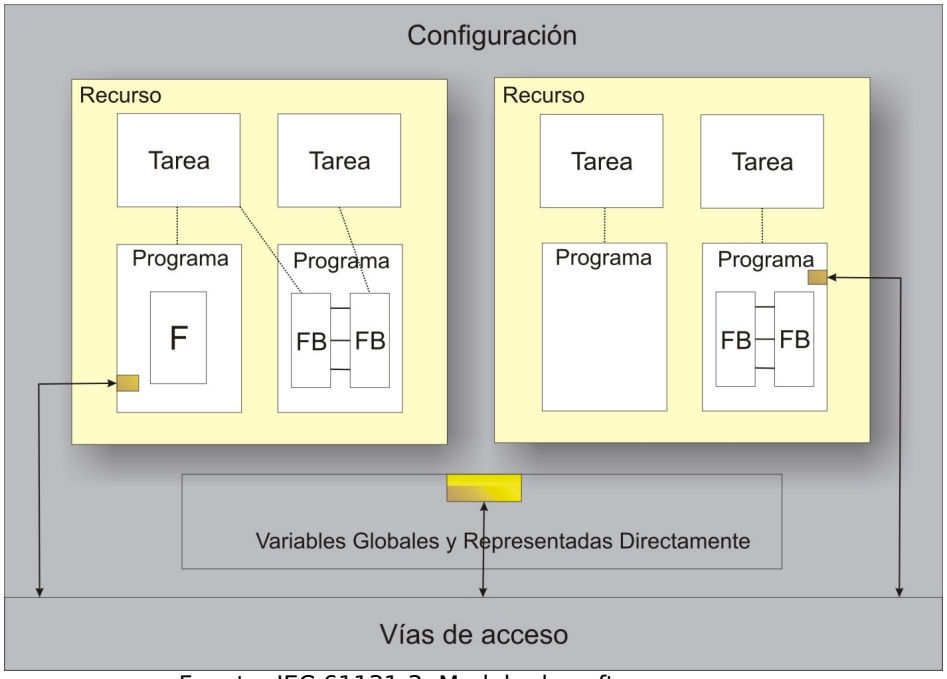
  Tipos de datos: boléanos, enteros, reales, byte, palabra, cadenas de caracteres, fecha, hora del día, canal analógico de entrada, tipos de datos derivados (creados por el usuario).

  BOOL, INT, REAL, BYTE, WORD, STRING, DATE, TIME_OF_DAY.

  Variables. Asignan direcciones del hardware: E/S, memoria y datos. Locales o globales. Hacen la programación independiente del hardware.

- Modelos de software.

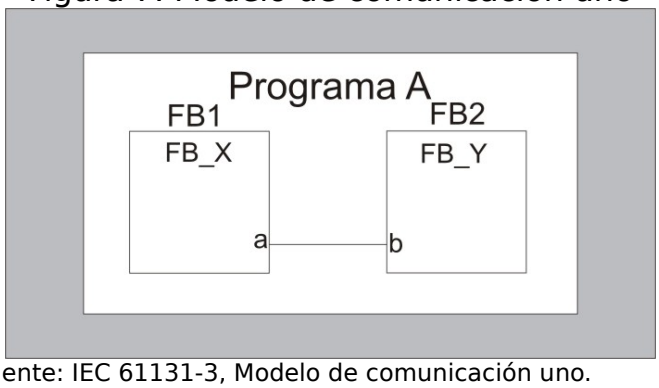
Figura 6. Modelo de software



Fuente: IEC 61131-3, Modelo de software

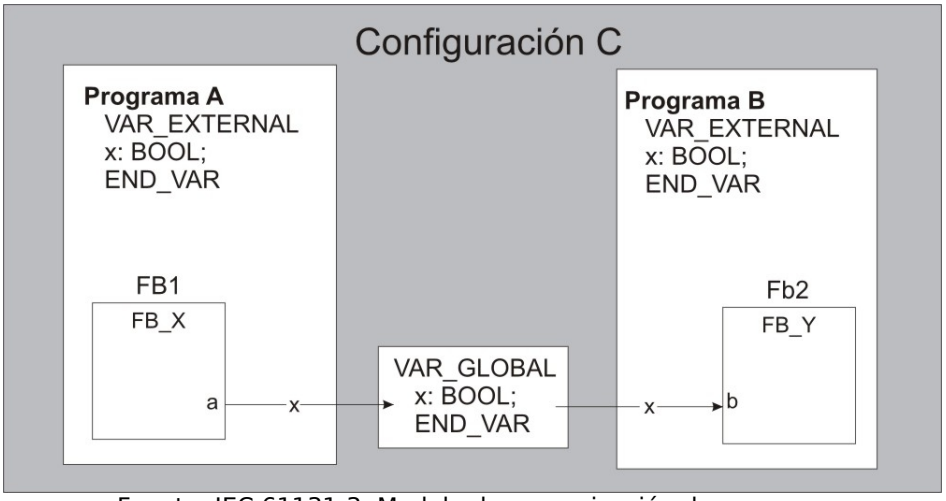
- Modelo de comunicación de datos

Figura 7. Modelo de comunicación uno



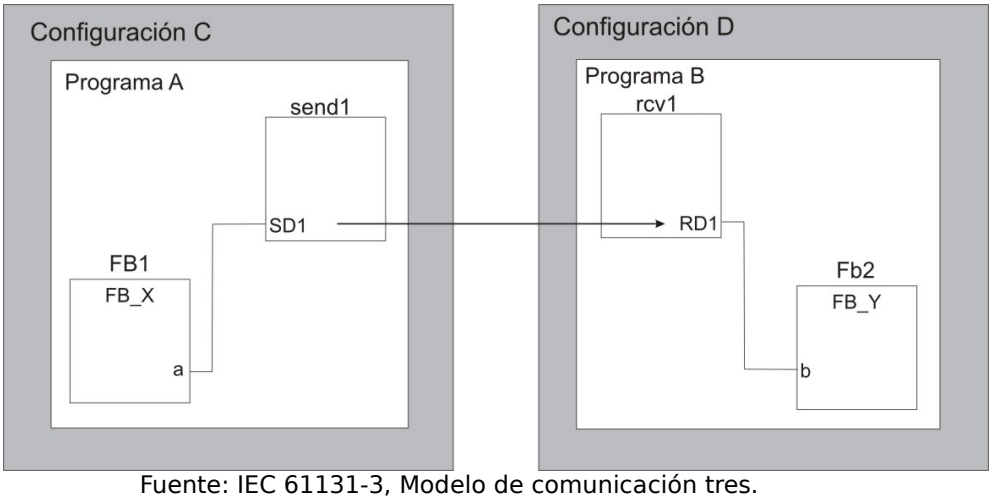
Fuente: IEC 61131-3, Modelo de comunicación uno.

Figura 8. Modelo de comunicación dos



Fuente: IEC 61131-3, Modelo de comunicación dos.

Figura 9. Modelo de comunicación tres



Fuente: IEC 61131-3, Modelo de comunicación tres.

- Modelo de programación

Tipos de datos derivados
Unidades de organización de programa:

- Funciones
- Bloques funcionales
- Programas

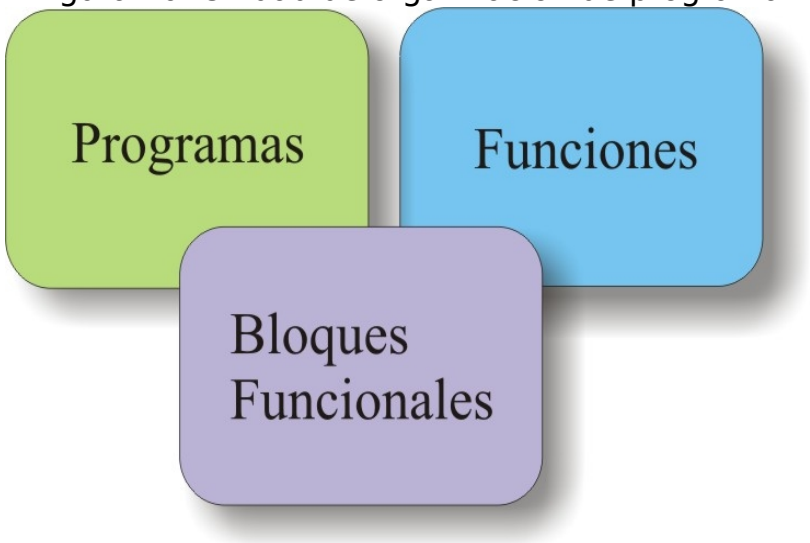
Elementos del diagrama secuencial (SFC)
Elementos de configuración:

- Variables globales

Recursos
Tareas
Vías de acceso

- Unidades de organización del programa

Figura 10. Unidad de organización de programa



Fuente: IEC 61131-3, Unidad de organización de programa.

Funciones:

Una función se define como una unidad de organización del programa que al ser ejecutada suministra exactamente un elemento de datos y cuya invocación se puede utilizar en lenguajes literales como operando en una expresión.

Las funciones no deben contener ninguna información de estado interno, es decir, que la invocación de una función con los mismos argumentos (parámetros de entrada) debe suministrar siempre el mismo valor (salida). Existen funciones estándar y Funciones definidas por el usuario

Bloques funcionales:

Un bloque funcional es una unidad de organización del programa que al ser ejecutada suministra uno o más valores. Existe la posibilidad de crear múltiplos (copias) de un bloque funcional, denominadas instancias.
Cada instancia llevará asociado un identificador (el nombre de la instancia) y una estructura de datos que contenga sus variables de salida e internas.

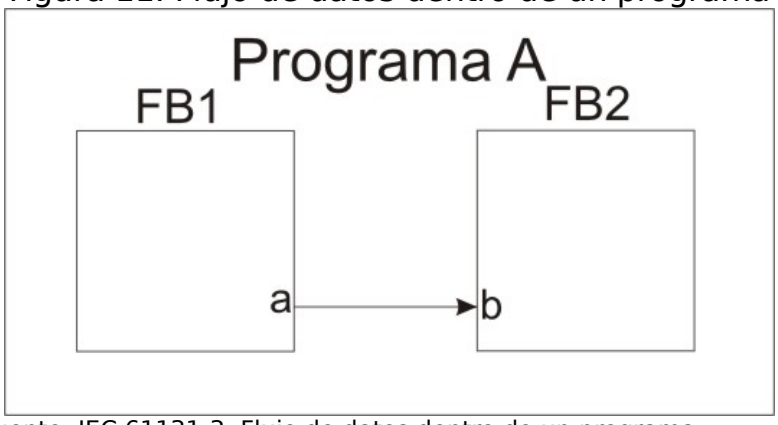
Todos los valores de las variables de salida e internas deberán persistir de una ejecución del bloque funcional al siguiente, por lo que la llamada de un mismo bloque funcional con los mismos argumentos (parámetros de entrada) no tiene por qué dar siempre los mismos valores de salida.

Programas:

Los programas son “un conjunto lógico de todos los elementos y construcciones del lenguaje de programación que son necesarios para el tratamiento de señal previsto que se requiere para el control de una máquina o proceso mediante el sistema de autómata programable”.

Flujo de datos dentro de un programa:

Figura 11. Flujo de datos dentro de un programa

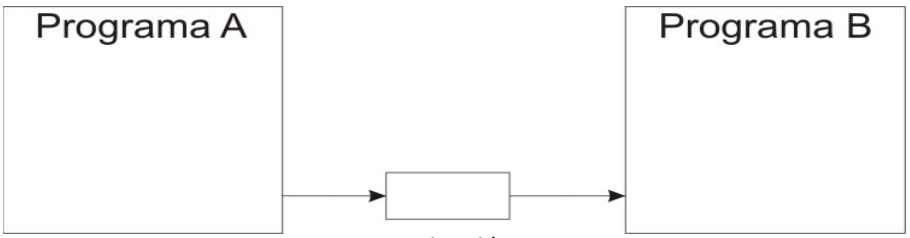


Fuente: IEC 61131-3, Flujo de datos dentro de un programa.

Comunicación entre programas.

En la misma configuración (variables globales), o en configuraciones distintas (vías de acceso, comunicaciones)

Figura 12. Comunicación entre programas



Fuente: IEC 61131-3, Comunicación entre programas.

- Elementos de configuración

Una configuración se compone de:

RECURSOS
TAREAS
VARIABLES GLOBALES
VIAS DE ACCESO

Esta es la exposición de los elementos comunes, que estructuran las generalidades de los lenguajes de programación. Se presenta a continuación un ejemplo que muestra grafica y textualmente la definición de bloques funcionales y programas.

Figura 13. Ejemplo de elementos de configuración

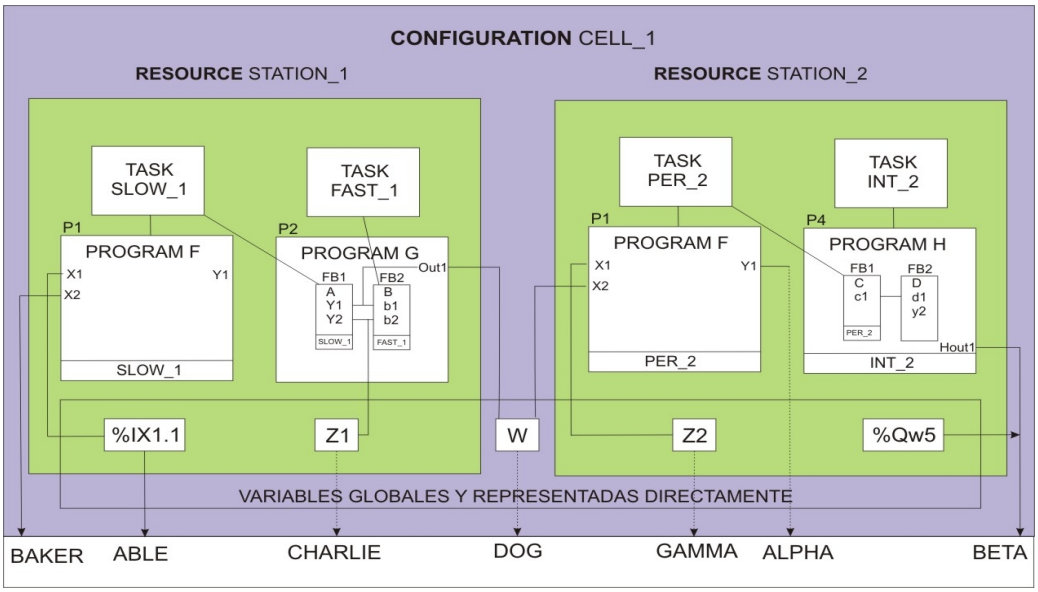


Figura 14. Declaración de Bloque funcionales y parámetros

```
FUNCTION_BLOCK A
VAR_OUTPUT
    y1  :  UINT;
    y2  :  BYTE;
END_VAR
END_FUNCTION_BLOCK
```

```
FUNCTION_BLOCK B
VAR_INPUT
    b1  :  UINT;
    b2  :  BYTE;
END_VAR
END_FUNCTION_BLOCK
```

```
FUNCTION_BLOCK C
VAR_OUTPUT
    c1  :  BOOL;
END_VAR
END_FUNCTION_BLOCK
```

```
PROGRAM F
VAR_INPUT
    x1  :  BOOL;
    x2  :  UINT;
END_VAR
VAR_OUTPUT
    y1  :  BYTE;
END_VAR
END_PROGRAM
```

```
PROGRAM G
VAR_OUTPUT OUT1  :  UINT;  END_VAR
VAR_EXTERNAL Z1  :  BYTE;  END_VAR
VAR FB1  :  A;  FB2  :  B;  END_VAR
FB1(...); OUT1 := FB1.Y1; Z1 := FB1.Y2;
FB2(B1 := FB1.Y1) ; B2 := FB1.Y2);
END_PROGRAM
```

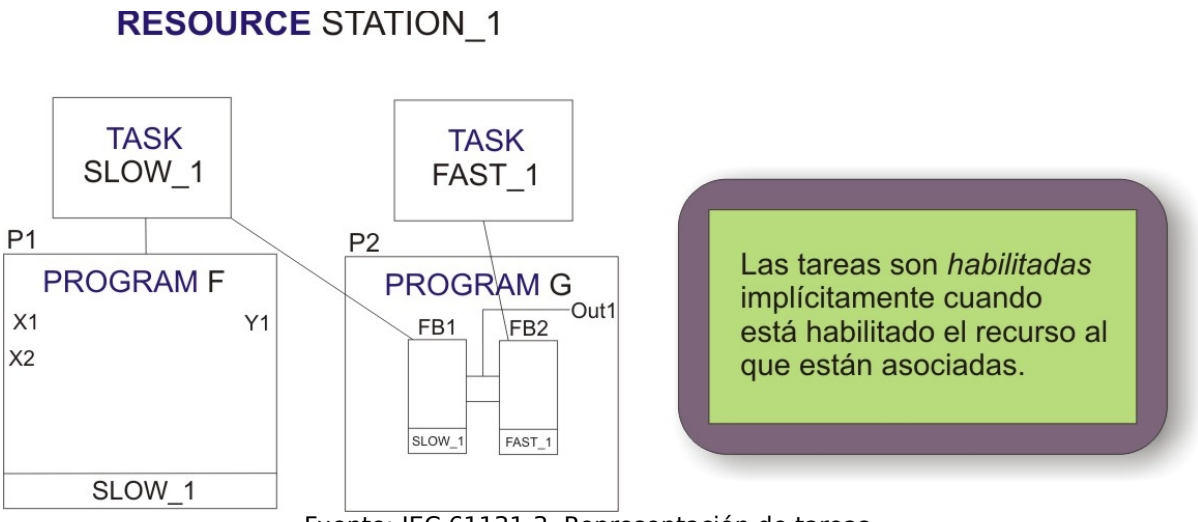
```
PROGRAM H
VAR_OUTPUT HOUT1  :  INT;  END_VAR
VAR FB1  :  C;  FB2  :  D;  END_VAR
FB1(...);
FB2(D1 := FB1.C1) ; HOUT1 := FB2.Y2;
END_PROGRAM
```

Figura 15. Configuración, recursos y variables

```
CONFIGURATION CELL_1
   VAR_GLOBAL w : UINT; END_VAR
   RESOURCE STATION_1 ON PROCESSOR TYPE_1
      VAR_GLOBAL z1 : BYTE ; END_VAR
      TASK SLOW_1(INTEGRAL := t#20ms, PRIORYTY := 2) ;
      TASK FAST_1(INTERVAL := t#10ms, PRIORYTY := 1) ;
      PROGRAM P1 WITH SLOW_1 ;
            F(x1 := %IX1.1) ;
      PROGRAM P2 : G(out1 => w,
            FB1 WITH SLOW_1,
            FB2 WITH FAST_1) ;
   END_RESOURCE
   RESOURCE STATION_2 ON PROCESSOR TYPE_2
      VAR_GLOBAL z2 : BOOL ;
         AT %QW5 : INT ;
      END_VAR
   TASK PER_2(INTERVAL := t#50ms, PRIORITY := 2) ;
   TASK INT_2(SINGLE := z2, PRIORITY := 1) ;
   PROGRAM P1 WITH PER2 :
      F(x1 := z2, x2 := w) ;
   PROGRAM P4 WITH INT 2 :
      H(HOUT1 => %QW5,
      FB1 WITH PER_2) ;
   END_RESOURCE
   VAR_ACCESS
      ABLE : STATION_1.%IX1.1 : BOO READ_ONLY ;
      BAKER : STATION_1.P1.X2 : BOO READ_ONLY ;
      CHARLIE : STATION_1.z1 : BOO READ_ONLY ;
      DOG : w : BOO READ_ONLY ;
      ALPHA : STATION_2.P1.y1 : BOO READ_ONLY ;
      BETA : STATION_2.P4.HOUT1 : BOO READ_ONLY ;
      GAMMA : STATION_2.z2 : BOO READ_ONLY ;
   END_VAR
END_CONFIGURATION
```

Por ultimo se tiene el elemento de configuración tareas, Tarea es un elemento de control de ejecución que es capaz de iniciar la ejecución de un conjunto de POU´s: programas y bloques funcionales, cuyas instancias están en la declaración de los programas.

Figura 16. Representación de tareas



Fuente: IEC 61131-3, Representación de tareas.

B. Lenguajes de programación

- **Lista de instrucciones (IL)**

Es un lenguaje textual, procedente de Alemania, sus características básicas son:

- Tipo de lenguaje ensamblador.
- Es una trascripción elemental e inmediata de las instrucciones del lenguaje maquina.
- Representación por expresiones nemotécnicas.
- Aplicación en pequeños controles.

Las instrucciones en IL están compuestas así:

Etiqueta : Operador/Nombre de función Operando Comentario

Las etiquetas son nombres asignados a partes de codificación para su identificación.

Tabla 1: Operadores IL.

Operador	Grupo de Operadores	Descripción
LD LDN	Creación	**Carga el operando, carga su negado respectivamente. (En el CR)**
AND ANDN AND(ANDN(	Proceso	**Operación lógica AND y su respectiva negación**
OR ORN OR(ORN(	Proceso	**Operación lógica OR y su respectiva negación**
XOR XORN XOR(XORN(	Proceso	**Operación lógica de OR-exclusivo y su respectiva negación**
ST STN	Salida	**Asigna lo que está cargado en el CR sobre el operando que lo sigue (STN negación)**
S	Salida	**Asigna un Verdadero al operando que lo sigue si el resultado del CR es 1.**
R	Salida	**Asigna un Falso al operando que lo sigue si el resultado del CR es 1**
)	Salida	**Paréntesis de Cierre, evalúa la operación que encierra**
ADD ADD(	Proceso	**Suma el operando con el valor del CR**
SUB SUB(	Proceso	**Sustrae al CR el operando**
MUL MUL(	Proceso	**Multiplica el operando con el CR**
DIV DIV(	Proceso	**Divide el CR por el operando**
GT GT(	Proceso	**CR > Operando (Resultado Booleano)**
GE GE(	Proceso	**CR >= Operando (Resultado Booleano)**
EQ	Proceso	**CR = Operando (Resultado**

Operador	Grupo de Operadores	Descripción
EQ(		**Booleano)**
NE NE(	Proceso	**CR <> Operando (Resultado Booleano)**
LE LE(	Proceso	**CR <= Operando (Resultado Booleano)**
LT LT(	**Proceso**	**CR < Operando (Resultado Booleano)**

Fuente: IEC 61131-3, Operadores.

El operando hace referencia a las variables sobre las que se opera y estas son de algún tipo de dato soportado por el lenguaje. Por ultimo tenemos el comentario debe empezar con (* y terminar con *).

Funciones y bloques funcionales. Se coloca el nombre en el campo del operando y los parámetros, si los lleva, entre paréntesis. Se emplea el operador CAL.

- **Texto estructurado (ST)**

Es un lenguaje textual denominado de alto nivel, por su completa funcionalidad y amplitud de operadores, posee grandes similitudes con lenguajes de programación de PC, como Pascal o C; sus características son:

  - Estructuración en bloques.
  - Es una construcción sintáctica que al ser evaluada proporciona un valor.
  - Esta compuesta por operadores y operandos.
  - Facilita la programación de procesos que requieren instrucciones complejas y cálculos grandes.

Un programa en ST consiste en un número de declaraciones separadas por punto y coma (;), lo cual significa que las declaraciones pueden ser extendidas en una línea o en múltiples líneas. De igual forma que en IL los comentarios empiezan con (* y terminan con *) y estos pueden estar en cualquier parte del programa ST.

Tabla 2: Estructuras ST.

Etiqueta	Descripción	Ejemplo	Explicación
:=	Asignación	d:=10;	**Asignación de un valor calculado en la derecha y es**

Etiqueta	Descripción	Ejemplo	Explicación
			asignado en el de la izquierda.
	Llamado de un FB	FBNombre(P1:= 10; P2:=20);	**Llamado de otra POU de tipo FB incluyendo sus parámetros.**
RETURN	Retorno	RETURN;	**Salida de un POU y retorna el valor de llamada del POU.**
IF	Condición	IF d<e THEN F:=1; ELSIF d=e THEN F:=2; ELSE f:=3; END_IF	**Selección de alternativas por alguna expresión booleana.**
CASE	Selección Múltiple	CASE f OF 1: g:=11; 2: g:=FunA(); ELSE g:=12; END_CASE	**Selección de un bloque de instrucciones, dependiendo del valor de la expresión "f".**
FOR	Bucle "Para"	FOR h:=1 TO 10 BY 2 DO F[h/2]:=h; END_FOR;	**Múltiples ciclos de un bloque de instrucciones, con condición tanto de inicio como de fin y un valor de incremento**

WHILE	Bucle "Mientras"	WHILE m>1 DO
		n := n / 2;
		END_WHILE;
			Múltiples ciclos de un bloque de instrucciones, con condición de finalización.

REPEAT	Bucle "Repetir hasta"	REPEAT I:=i*j; UNTIL i<10000 END_REPEAT;	**Múltiples ciclos de un bloque de instrucciones con condición de finalización.**
EXIT	Finaliza el Ciclo	EXIT;	**Terminación prematura de una iteración en un bucle**
;	**Final de Instrucció**	**; ;**	

Etiqueta	Descripción	Ejemplo	Explicación
	n		

Fuente: IEC 61131-3, Estructuras ST.

Dentro del lenguaje ST se encuentran los operandos y operadores, los primeros se pueden clasificar como sigue:

- Literales (numéricos, alfanuméricos caracteres, de tiempo)
- Variables (simples, multi-elementos)
- Llamado de funciones (Nombrefuncion(argumentos))
- Expresiones extendidas (10+20)

Tabla 3: Operadores ST.

Operador	Descripción	Ejemplo y Valor de la Expresión	Prioridad
( )	Paréntesis	(2 * 3) + (4 * 5) Valor: 26	Alta
	Llamado a una Función	CONCAT('AB','CD') Valor: 'ABCD'	
**	Exponenciación	3**4 Valor: 81	
-	Negación	-10 Valor: -10	
NOT	Complemento	NOT TRUE Valor: FALSE	
*	Multiplicación	10*20 Valor:200	
/	División	20/10 Valor: 2	
MOD	Módulo	17 MOD 10 Valor: 7	
+	Suma	1.4 + 2.5 Valor: 3.9	
-	Resta	3 – 2 - 1 Valor: 0	
<, >, <=, >=	Comparación	10 > 20 Valor: FALSE	
=	Igualdad	T#26h = T#1d2h Valor: TRUE	
<>	Desigualdad	8#15 <> 13 Valor: FALSE	
&, AND	AND Lógico	TRUE AND FALSE	

		Valor: FALSE	
XOR	OR exclusivo lógico	TRUE XOR FALSE Valor: TRUE	
OR	OR Lógico	TRUE OR FALSE Valor: TRUE	Baja

Fuente: IEC 61131-3, Operadores ST.

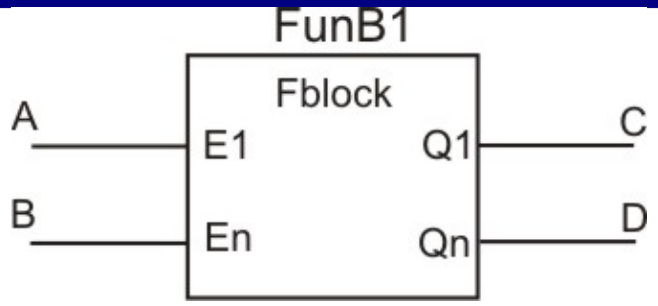
- **Diagrama de Bloques Funcionales (FBD)**

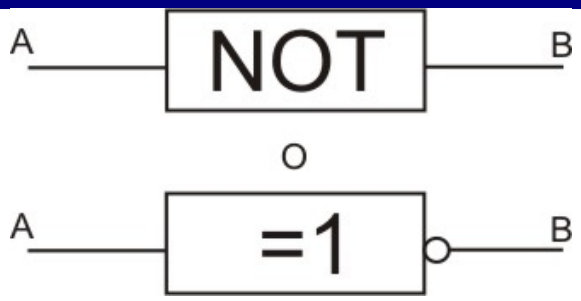
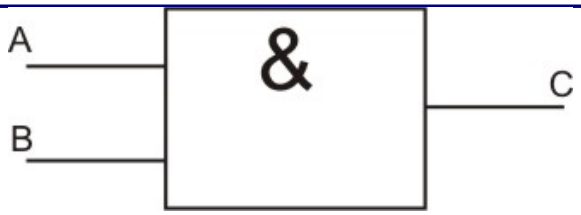
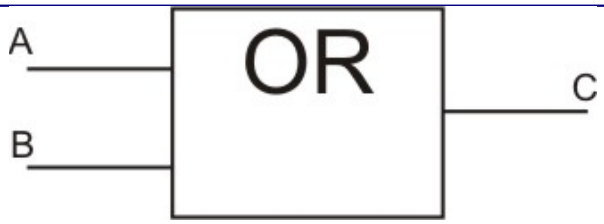
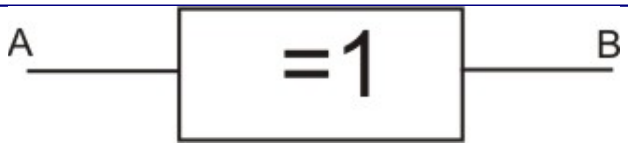
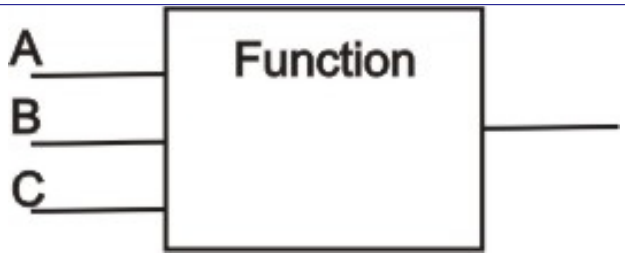
Es un lenguaje grafico, y sus representaciones se dan mediante caracteres ISO/CEI 646 o elementos gráficos o semigráficos. Sus características básicas son:

- Los programas son bloques cableados entre sí de forma análoga al esquema de un circuito.
- Posee una interfase de E/S bien definida y además posee un código interno oculto.
- Posee conjunto de funciones y bloques bien estructurados.
- Su programación es estructurada.

Dentro del FBD, para la estructuración de un programa, la parte de declaración puede ser definida de forma textual o grafica, dependiendo del soporte que se tenga para el lenguaje; siendo la más utilizada la declaración textual, la parte de codificación esta dividida en redes y las redes consisten en; etiqueta de la red, comentario de la red y el grafico de la red.

Tabla 4: Elementos FBD.

Descripción	Ejemplo
Llamado a un bloque funcional (Entre ellos: FlipFlop, Temporizador, Contador, detectores de flanco, etc)	

Descripción	Ejemplo
Complemento	
AND Lógico	
OR Lógico	
XOR Lógico	
Asignación	
Llamado de funciones (Entre ellas: ABS, MUL, DIV, MOD, CONCAT, etc.)	

Finalmente se tiene las reglas de ejecución del FBD

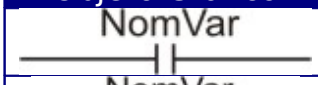
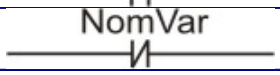
- El bloque se ejecuta cuando todas sus entradas han sido evaluadas
- El bloque se evalúa por completo cuando se han calculado todas sus salidas.
- La evaluación de un conjunto de bloques termina cuando se calculan todas y cada una de sus salidas.

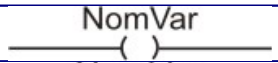
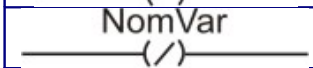
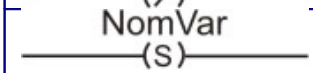
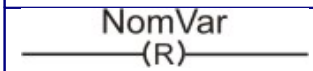
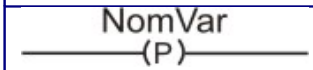
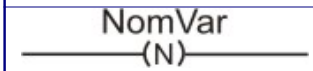
- **Esquema de Contactos (LD)**

Es un lenguaje grafico, es el lenguaje de más antigua utilización, de esto se deriva que presente similitud con los diagramas eléctricos de escalera utilizados por los técnicos, anteriormente a la programación del autómata. Sus características básicas son:

- Las decisiones de control son programas en ejecución.
- Se ejecuta por líneas, que contienen las entradas y salidas; las entradas permiten una comparación o test de las condiciones y se obtiene el resultado de la evaluación. La salida examina el resultado de la evaluación y si es verdadera ejecuta alguna operación o función.
- Las instrucciones de entrada pueden ejecutarse mediante relaciones lógicas AND y OR de forma sencilla; si las instrucciones están en serie se evalúa AND y en caso que estén en paralelo se evalúa OR, de igual forma la salida en paralelo permite activar varias operaciones o funciones con el mismo resultado de la evaluación.

Tabla 5: Elementos LD.

Objeto Gráfico	Nombre	Explicación
	Contacto Abierto	CD[a] := CI[b] AND nomVar
	Contacto Cerrado	CD := CI AND NOT nomVar
	Transición a Positivo	CD := TRUE, Cuando: • La variable tenía el valor de FALSE en el ciclo de ejecución anterior **y** • La variable ahora tiene el valor de TRUE (en el ciclo de ejecución actual) **y** • CI tiene como valor TRUE. := FALSE, En cualquier otro caso
	Transición a	CD := TRUE, Cuando:

Objeto Gráfico	Nombre	Explicación
	Negativo	• La variable tenía el valor de TRUE en el ciclo de ejecución anterior **y** • La variable ahora tiene el valor de FALSE (en el ciclo de ejecución actual) **y** • CI tiene como valor TRUE := FALSE, En cualquier otro caso
	Bobina	nomVar := CI
	Bobina Negada	nomVar := NOT CI
	Activación (Set)	nomVar := TRUE, si CI tiene el estado de TRUE, si esto no ocurre: nomVar no cambia de valor.
	Desactivación (Reset)	nomVar := FALSE, si CI tiene el valor de TRUE, si esto no ocurre: nomVar no cambia de valor.
	Bobina de transición a positivo	nomVar := TRUE, cuando • CI tenía el valor de FALSE en el ciclo de ejecución anterior **y** • CI tiene ahora el valor de TRUE (en el presente ciclo de ejecución) Si esto no ocurre: la variable no cambia de valor.
	Bobina de transición a negativo	nomVar := TRUE, Cuando: • CI tenia el valor de TRUE en el ciclo de ejecución anterior **y** • CI ahora tiene el valor de FLASE (en el presente ciclo de ejecución).

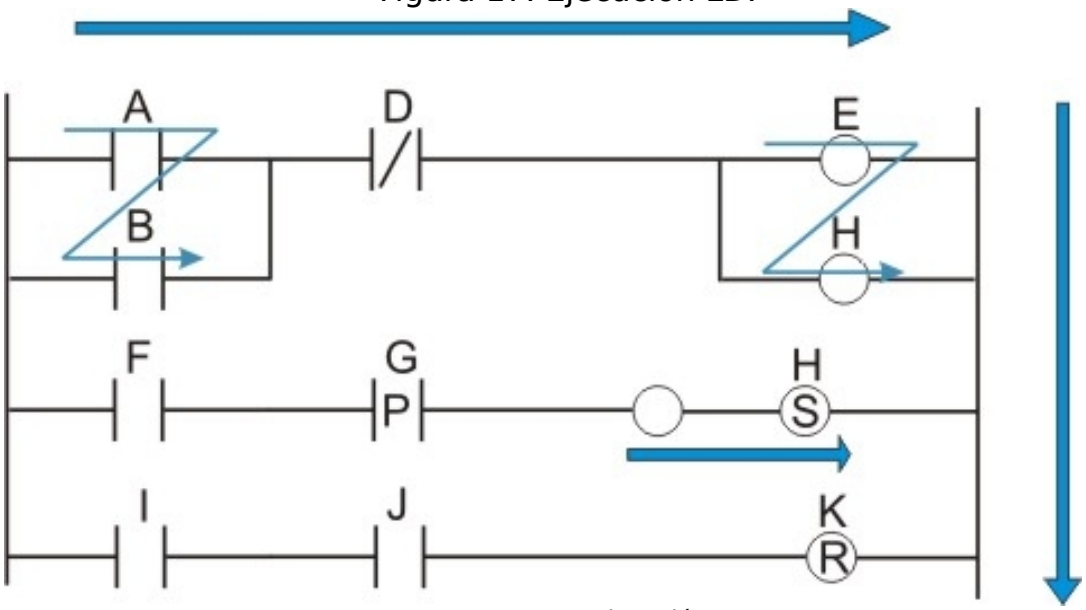
Fuente: IEC 61131-3, Elementos LD.

a. CD es la sigla equivalente para este caso de Conexión por Derecha

b. CI es la sigla equivalente para este caso de Conexión por Izquierda

La ejecución del LD se da de izquierda a derecha y de arriba abajo, las líneas con bifurcaciones se ejecutan de arriba izquierda a abajo derecha.

Figura 17. Ejecución LD.



Fuente: IEC 61131-3, Ejecución LD.

- **Esquema Secuencial de Funciones (SFC)**

Es SFC es un lenguaje, de representación gráfica primordialmente, sin embargo; posee la representación textual de sus elementos gráficos, proviene de un estándar anterior IEC 848 que fue construido basando en el Gráfico funcional de control etapa transición (Grafcet). Sus características básicas son:

- Describe el comportamiento secuencial de un proceso y programa.
- Usado para distribuir problemas de control.
- Permite un rápido diagnostico.
- Las etapas o estados implican acciones asociadas.
- Las transiciones gobiernan los cambios de estado
- Pueden darse esquema que poseen divergencia y convergencias, así como ciclos.
- Se pueden definir secuencias alternativas o paralelas.

Los elementos gráficos del SFC son:

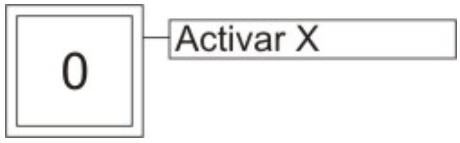
Figura 18. Etapa.

Estado específico del diagrama en el cual se da la orden de ejecutar sus acciones asociadas cuando se active y que permanece en ese estado hasta el momento en que la condición de alguna de sus transiciones salientes sea verdadera (La etapa inicial se especifica por medio de dos recuadros concéntricos).

Figura 19. Transición.

Pasaje de una etapa a otra, Tiene asociada una condición (función booleana combinacional), su condición es validada cuando alguna de las etapas anteriores está activa.

Figura 20. Acción

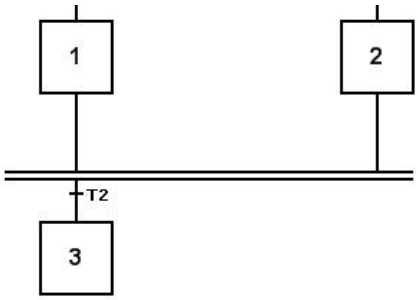


Operaciones a realizar sobre el sistema cuando la etapa a la que está asociada está activa.

Figura 21. Divergencia en Y

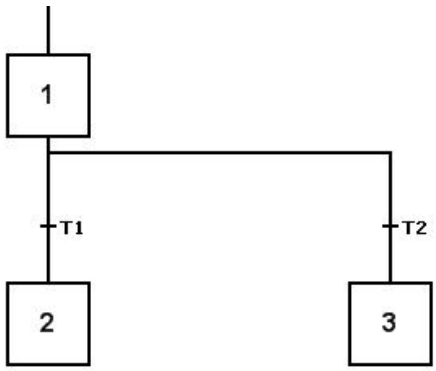
Asocia varios caminos diferentes agrupándolos con una condición común que al cumplirse desata una ejecución simultanea de las etapas a las cuales están enlazadas. Toda Divergencia en Y debe terminar en una Convergencia en Y.

Figura 22. Convergencia en Y.



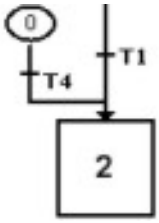
Es la manera de cerrar dos o más caminos diferentes con ejecución simultanea gracias a una divergencia en Y asociando una condición como común a las líneas que se van a cerrar, de tal forma que si se cumple estos caminos terminan y convergen en una única etapa de destino que continua con el proceso.

Figura 23. Divergencia en O.



Es cuando una etapa tiene asociadas mas de una transición saliente, con condiciones combinacionales lógicas excluyentes entre si para evitar conflictos e incongruencias. Cada una de esas condiciones tiene como destino un camino diferente lo que hace que haya una continuación de proceso de manera alternativa.

Figura 24. Convergencia en O



Es cuando una etapa tiene asociadas mas de una transición entrante, cuyas condiciones combinacionales lógicas son excluyentes entre si. Convirtiéndose en un destino alternativo para varios caminos del proceso. Cuando una de las condiciones se cumple y la etapa inmediatamente superior está activa se produce un flanqueamiento a la etapa destino de la convergencia.

La ejecución del SFC se da en virtud de las líneas de evolución, es decir; después de la etapa de doble rotulo, llamada etapa inicial, se desprenden líneas que se conectan a otra etapa, el orden esta dado por estas líneas y las condiciones consignadas en las transiciones, que permiten la ejecución del grafico diseñado, así la ejecución de las acciones asociadas a las etapas estará sujeta al cumplimiento de la condición de la transición y al orden secuencial de las líneas de evolución.

2.3.4 Parte 4. Guía de usuario

La cuarta parte se piensa como una guía para ayudar al usuario de PLC en todas las fases del proyecto de la automatización. La información Práctica-orientada se da en los asuntos que se extienden de análisis del sistema y de la opción del correcto equipo para el mantenimiento.

2.3.5 Parte 5. Especificación de comunicaciones

Esta última parte se refiere a la comunicación entre PLC's de diferente fabrica con ellos y con otros dispositivos. En cooperación con ISO 9506 (especificación del mensaje de la fabricación; las clases de la conformidad de MMS) serán definidas para permitir que PLC's se comunique, por ejemplo, vía redes. Éstos cubren la función de la selección del dispositivo, del intercambio de datos, del proceso de alarmar, del control de acceso y de la administración de la red.

3. IEC 1131-3 PARA AUTOMATISMOS LÓGICOS SECUENCIALES

3.1 INTRODUCCIÓN

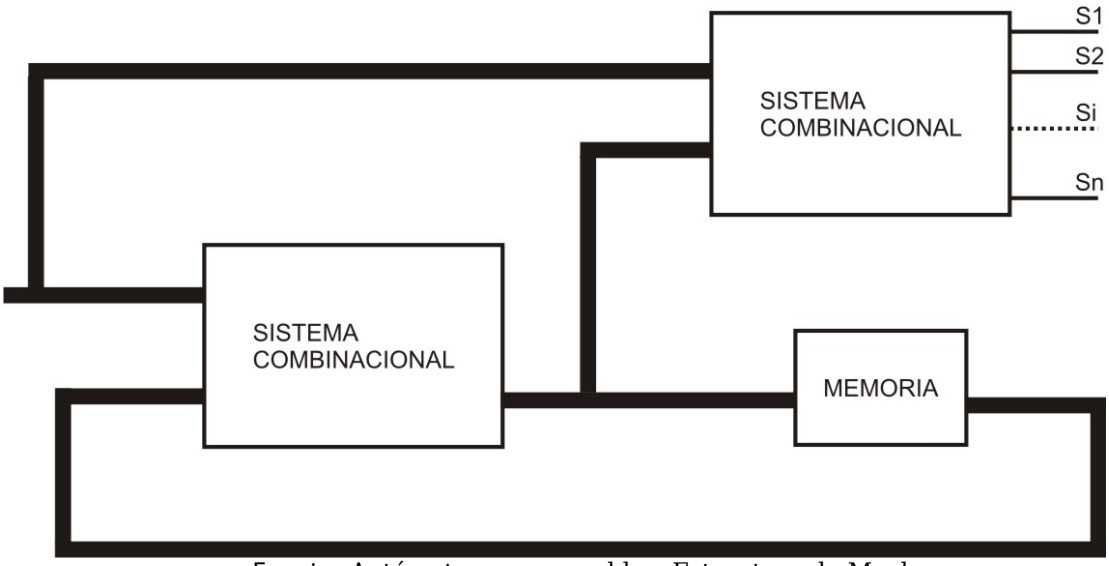
En el presente capítulo se encuentra condensada la interpretación y sustento conceptual que posibilitó el desarrollo del proyecto, como se manifestó en la introducción del capitulo anterior, aunque la globalidad del entendimiento del estándar es importante, donde centra la atención el estándar para el proyecto se encuentra consignada en la tercera parte, por tanto, será en contexto del IEC 1131-3 y para automatismos lógicos secuenciales que se desarrolla este capitulo.

La investigación que se realizó, desde una perspectiva de una investigación proyectiva, mostrara que frente al problema planteado lo más acertado es interpretar el IEC 1131-3, desde la fortaleza que evidencia éste, sobre el SFC y para este lenguaje construir los recursos necesarios de forma tal que sea el diseño mediante él, el que logre establecer un puente comunicacional para el campo de la automatización industrial.

3.2 CONCEPTOS E INTERPRETACIONES

Se ha enunciado que para este trabajo su rango de acción lo dan los automatismos lógicos secuenciales, se tiene entonces; que un automatismo secuencial es aquel cuyas salidas dependen de las variables de entrada y del propio estado inicial del sistema, que puede ser cualquiera, entonces debe tener la capacidad de memorizar todos y cada uno de los estados posibles. El concepto de lógico implica, para este caso en particular, lo concerniente a la bivalencia de cero y uno, por composición tenemos pues lo referente a los automatismos lógicos secuenciales.

Figura 25. Estructura Mealy



Fuente: Autómatas programables, Estructura de Mealy.

De igual forma también se ha enunciado que es la fuente de prioridad para la investigación el IEC 1131-3 en lo concerniente a los lenguajes que éste expone, en relación a su sintaxis y su forma de trabajo, y aquí se hace necesario aclarar que la forma de trabajo asumida Top Down permitido evidenciar a la investigación que la posibilidad para la resolución del problema planteado, estaba en recuperar un lenguaje con la suficiente capacidad de diseño y la independencia de fabricantes que colocara al campo de la automatización con la facilidad de convergencia a él.

Por último, y antes de entrar en contacto con la descripción de los lenguajes y la separación que se ha realizado del SFC para ubicarlo dentro de una labor de diseño del automatismo lógico secuencial, se precisa necesario mostrar: Primero, que el SFC en proveniente de un estándar anterior (IEC 848), estándar que nace con la intención de mostrar un metodología clara para la descripción y el diseño del control, independientemente de la tecnología a usar; Segundo, que estando el SFC en la capacidad de mostrarse gráfica y textualmente, lo convierte en un lenguaje de mayor amplitud a la hora de su interpretación, lo cual posibilita un mayor rango comunicacional, con lo cual se tiene un lenguaje que por sus elementos constitutivos se muestra apropiado para el diseño y además con la capacidad, al no estar atado a tecnología particular alguna, de generar un puente comunicacional efectivo.

3.3 MANEJO DEL DISEÑO.

En el campo de la automatización es frecuente encontrar procesos que implican la realización de una serie de actividades u operaciones de forma secuencial. La ejecución de éstas actividades convoca, para el caso de los automatismos lógicos secuenciales, señales binarias. El desarrollo del proceso es una sucesión encadenada de operaciones, cuya evolución se controla mediante unas condiciones de tipo lógico, que indican si el proceso continua y como debe hacerlo.

Ahora, los automatismos que controlan estos procesos, poseen bloques combinacionales y células de memoria biestable interconectados (ver Figura 25. Estructura de Mealy). De esta forma se tiene que para cualquier lenguaje que aspire realizar el diseño de un automatismo lógico secuencia debe tener en cuenta estas condiciones, por tanto debe ser observable que el SFC recoge estas condiciones.

El manejo que se da al diseño, es en consecuencia con el cumplimento de las condiciones necesarias para los procesos en los que confluyen el bloque combinacional y las células de memoria y que el SFC los maneja desde sus elementos.

Con lo anteriormente expuesto, queda claro que es el SFC un lenguaje que posee las características para ser denominado un lenguaje de diseño y es en virtud de esta interpretación que la investigación avanza hacia la solución de: Primero, el fortalecimiento de la labor de diseño desde la focalización del SFC como el lenguaje destinado para que mediante él se diseñe y en gracia de lo diseñado se genere la codificación en los otros lenguajes y Segundo, que siendo el SFC un lenguaje de independencia tecnología posibilita la comunicación en el campo de la automatización industrial, ahora todo esto dentro del contexto de los automatismos lógicos secuenciales.

En adelante se mostrara como el SFC, cumple con las condiciones para el diseño de automatismos lógicos secuenciales y su importancia.

3.3.1 Esquema Secuencial De Funciones (SFC)

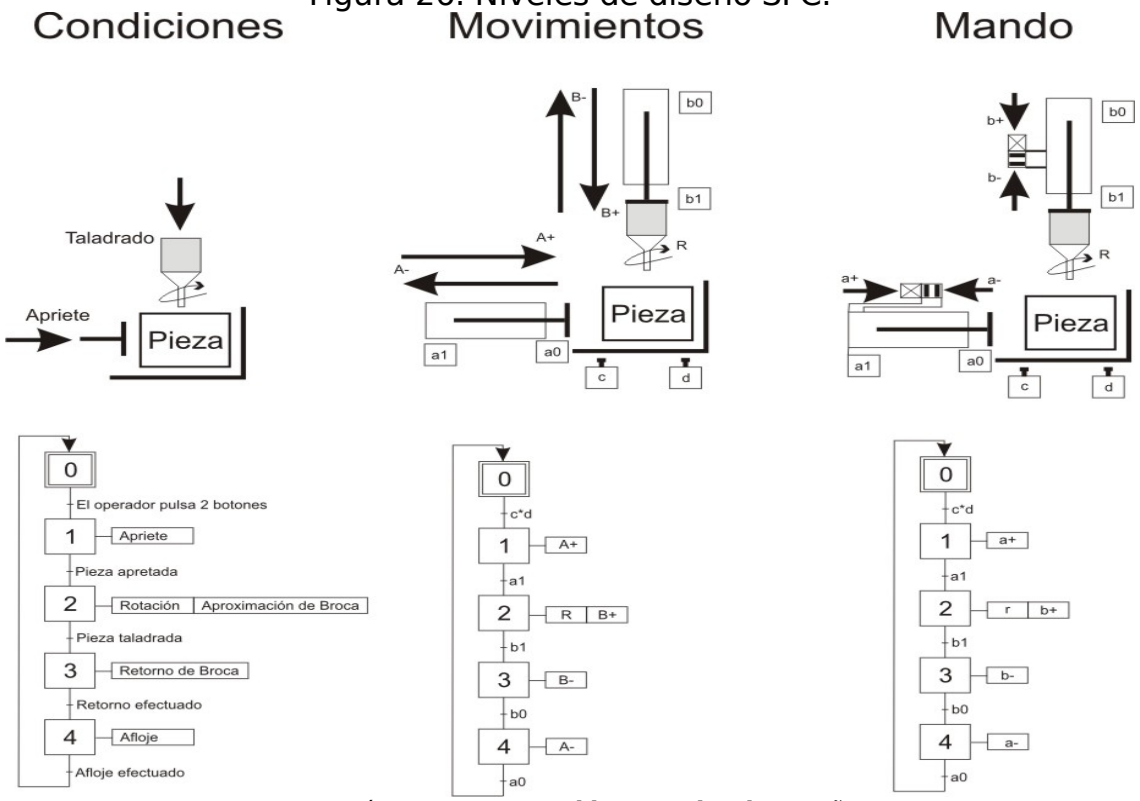
El SFC se hace importante, debido a que se ha diagnosticado que en el campo de la automatización industrial existe una debilidad entorno al diseño, por tanto es el diseño una labor importante que apoyar, esto; por que en el contexto de ingeniería, son los diseños los que marcan la pauta del desarrollo de cualquier proyecto, de un buen diseño se desprende un

buen proyecto, y además en la medida de la existencia de un diseño se posee la capacidad de interpretación, mantenimiento y reutilización de éste, ahora; debido a la existencia de un lenguaje que proporciona la solución de esta problemática, se deriva el fortalecimiento a esta labor mediante la generación de un mecanismo que focaliza sus esfuerzos en el trabajo mediante SFC.

Por otra parte, y en sintonía con lo anterior, el SFC en la forma de estructurar sus elementos permite unificar la forma de descripción del proceso para técnicos de distintas campos, desde el ingeniero industrial o de producción, que define las necesidades del automatismo, pasando por el de diseño, que debe implementar el sistema de control y los accionamientos, hasta el técnico de mantenimiento, que debe cuidar de su funcionamiento o introducir alguna modificación.

El SFC, posee la capacidad mediante su representación de presentar varios niveles, entre los cuales se puede encontrar. El nivel de condiciones, el de movimientos y el de mando (ver Figura 26. Niveles de diseño SFC), sin embargo; para la generación de codificación que pueda ser ejecutada por un PLC, deberá diseñarse en el último de los niveles, pues es en este nivel donde se relacionan los accionamientos y condiciones.

Figura 26. Niveles de diseño SFC.



Fuente: Autómatas programables, Niveles de Diseño SFC

Para la realización de un diseño mediante SFC para automatismos lógicos secuenciales es necesario tener en cuenta los siguientes aspectos:

- Caracterizar el funcionamiento del automatismo independientemente de los componentes con los que se vaya a construir.
- El elemento fundamental es la operación, entendida como una acción ha realizar por el automatismo, que pueden ser complejas o simples, las complejas deben poderse subdividir en otras mas elementales.
- Las acciones ha realizar deben ser elementales, de forma tal que solo dependan de relaciones combinacionales entre entradas y salidas.
- Se debe establecer un grafico de evolución que indique la secuencia de operaciones y las condiciones lógicas para pasar de una a otra, así se obtendrán las ecuaciones lógicas de las variables de estado. Y con esto queda establecida la parte secuencial del automatismo.

Ahora, como se mostró en el capitulo anterior, donde se expuso los elementos del lenguaje, que por la interpretación adoptada son elementos de diseño, el diseño se maneja entorno a ellos, por lo tanto; es necesario precisar que implicaciones tiene cada elemento y como se ajustan al trabajo para automatismo lógicos secuenciales.

A. **Elementos de diseño**

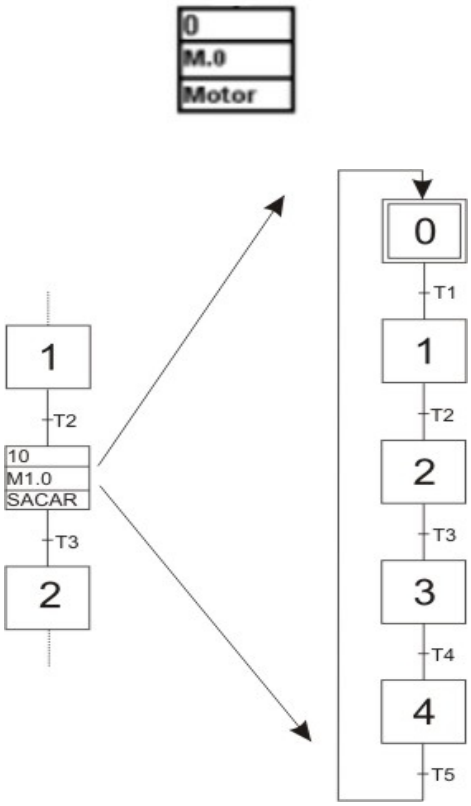
Estos elementos constituyen los símbolos a partir de los cuales se diseña mediante el grafico funcional. Los elementos son:

- Las etapas: representan cada uno de los estados del sistema, cada etapa tiene asociada por lo menos una acción, y ésta debe darse mediante una relación combinacional. Cada etapa corresponde a un estado del sistema por lo cual cada etapa tiene asociada una variable de estado[6] que es la encargada de generar la ejecución de la acción, de igual forma esta asociada a la etapa un número de identificación para ella, que dependerá del orden de agregación al diseño.
- Las macroetapas: Como se mostró en el capitulo anterior las etapas pueden albergar subrutinas o subprocesos que pueden tener la misma magnitud de complejidad que un diseño normal, pues se ha introducido para el cubrimiento de estas situación un elemento que tiene las mismas características

[6] Variable de estado o interna de estado son llamadas a las variables que elabora el sistema de las entradas y eventualmente de otras variables internas. Para el caso las variables de estado asociadas a las etapas son variables que dependen de las condiciones combinacionales y de otras variables de estado de otras etapas, todas estas relacionadas con los elementos de memoria biestables.

básicas de la etapa, en relación a las variables de estado, sin embargo; se ha otorgado la posibilidad que con la introducción de este elemento de diseño se logre poseer procesos interiores con el grado de complejidad que se requiera, haciendo mucho mas organizado la elaboración del diseño y del proyecto de automatización lógica secuencial.

Figura 27. Macroetapa



- Las líneas de evolución: son líneas que enlazan o unen a las etapas o macroetapas (ver definición macroetapa) que representan actividades consecutivas; su orientación siempre se entenderá de arriba hacia abajo, a menos que con flechas se indique lo contrario.
- Las transiciones: representan la condición lógica necesaria para que finalice la actividad de una etapa o macroetapa y se inicie la de la o las etapas o macroetapas inmediatamente consecutivas.
- Los reenvíos: Son símbolos en forma de flecha que indican la procedencia o destino de las líneas de evolución. Estas flechas permiten fraccionar el grafico de forma tal que no se dificulte su entendimiento.
- Divergencia y convergencia en “O”: Conjuntamente a la divergencia y convergencia se le llama bifurcación en “O”,

consiste en varios caminos que se pueden dar de forma alternativa, cada uno con su propia condición de transición; esta estructura debe ser globalmente cerrada, implica que; los caminos abiertos por la divergencia deben tener su correspondiente camino de cierre en la convergencia.

- Divergencia y Convergencia en “Y”: Conjuntamente a la divergencia y convergencia se le llama bifurcación en “Y”, consiste en la existencia de varios caminos que se inician simultáneamente cuando se cumpla una condición de transición común, esta estructura debe ser globalmente cerrada, es decir; el número de camino abiertos por la divergencia deben ser cerrados por la convergencia.

Algunas consideraciones de sintaxis para el SFC son:

- Cuando se recorre el grafico de evolución, por cualquier camino posible, deben alternarse siempre una etapa y una transición.
- Entre dos etapas debe existir una y solo una condición de transición, que puede venir expresada por una función combinacional con la complejidad que se requiera siempre que de cómo resultado un bit que indique uno para condición verdadera y cero para condición falsa.
- Los números de las etapas nada indican con relación al orden de ejecución, simplemente tiene un carácter de identificación.
- El diseño SFC debe poseer para cada etapa una(s) transición(es) que la conecte con otra u otras etapas o macroetapas.

Con la exposición realizada se ha mostrado, el argumento conceptual del SFC, entorno al manejo del diseño para automatismo lógicos secuenciales, en el capitulo cuatro se precisara la aplicación de lo expuesto conceptualmente en la herramienta Autologic 1131.

3.4 MANEJO DE LOS LENGUAJES IEC 1131-3

Con el manejo dado para el diseño ha quedado expuesto, la importancia de éste, en función de la necesidad de generar una fortaleza para el campo de la automatización entorno a él, de igual forma; que es el SFC, el lenguaje que focaliza la atención para el desarrollo de esta intención. Con lo cual queda que para el manejo de los lenguajes textuales y gráficos expuestos en el capítulo anterior, lo que corresponde es mostrar como se conectan con el SFC, en términos de sus elementos y cuales son los

elementos que los constituyen entorno al trabajo con los automatismos lógicos secuenciales y en relación con el IEC 1131.
De igual forma se debe hacer claridad, que el POU programa, para los lenguajes se manejara en dos segmentos, el primero que consignara la parte secuencial y el segundo que contendrá la parte combinacional, para una organización del código que permita una mejor visualización de lo diseñado. Debido a que; como se ha expuesto la parte secuencial que esta asociada a los elementos de memoria y estos a las etapas; con lo cual el primer segmento consignara las condiciones de transición que activaran o desactivaran las etapas y el segundo segmento contendrá, lo que deben ejecutar las variables de estado asociadas a las etapas, es decir; la ejecución de las acciones.

3.4.1 Lista de Instrucciones (IL)

El lenguaje IL en su forma de trabajo se ha mostrado similar a lenguaje ensamblador, mediante operadores, para el caso de la conexión con el SFC, se realiza teniendo en cuenta, que la correspondencia se debe dar: Primero, en la parte de declaración, teniendo claro que las variables concernientes a las transiciones son variables de entrada para el A.L.S, de igual forma que las variables asociadas a la acciones son variables de salida, aunque; puede darse el caso en que una variable asociada a las acciones sea requerida en la condición de transición, por lo cual esta variable será de entrada y salida y deberá declararse así. Segundo, que en la parte de codificación se deben disponer los operadores correspondientes para establecer la función combinacional (ver tabla 6. Operadores IL para A.L.S.), y finalmente que se debe realizar los llamados a los bloques funcionales estándar (ver apartado 3.4.5 Bloque Funcionales Estándar) y los definidos mediante el diseño con las macroetapas, antes de utilizar los resultados de estos para la ejecución de las acciones.

Con esta disposición en el orden de conexión con el SFC, se da la generación basada en el diseño para el IL.

A. **Elementos del lenguaje**

En lo anteriormente expuesto, se enunciaban la necesidad de la utilización de los operadores correspondientes, para la representación de la parte de declaración, así como la parte de codificación dentro de la generación del IL, para esto se dispone entonces de los siguientes operadores.

Tabla 6. Operadores IL para A.L.S.

Operador	Grupo de Operadores	Descripción
LD LDN	Creación	**Carga el operando, carga su negado respectivamente. (En el CR)**
AND ANDN AND(ANDN(	Proceso	**Operación lógica AND y su respectiva negación**
OR ORN OR(ORN(	Proceso	**Operación lógica OR y su respectiva negación**
ST STN	Salida	**Asigna lo que está cargado en el CR sobre el operando que lo sigue (STN negación)**
S	Salida	**Asigna un Verdadero al operando que lo sigue si el resultado del CR es 1.**
R	Salida	**Asigna un Falso al operando que lo sigue si el resultado del CR es 1**
)	**Salida**	**Paréntesis de Cierre, evalúa la operación que encierra**

Fuente: IEC 61131-3, Etiquetas IL.

Esta tabla muestra los operadores que se utilizan para el trabajo con A.L.S dentro del IEC 1131-3, sin embargo; no solo se necesitan operadores, debido a la necesidad de cubrir aspectos como: unidades o células de memoria (biestables), temporizadores, contadores y detectores de flanco; todos estos denominados bloques funcionales estándar (ver 3.4.5 Bloques Funcionales Estándar) y de gran importancia en los A.L.S

3.4.2 Texto Estructurado (ST)

El ST tiene un alcance de programación muy amplio debido a las estructuras que contempla, para el caso de la generación en ST a partir del SFC se debe contemplar: Primero, en la parte de declaración, teniendo claro que las variables concernientes a las transiciones son variables de entrada para el A.L.S, de igual forma que las variables asociadas a la acciones son variables de salida, aunque; puede darse el caso en que una variable asociada a las acciones sea requerida en la condición de

transición, por lo cual esta variable será de entrada y salida y deberá declararse así. Segundo, en la parte de codificación se deberá disponer los operandos y operados (ver tabla 7. Operadores ST para A.L.S) de la forma como se muestra en el capitulo dos para cubrir las funciones combinacionales consignadas en las transiciones, y estas funciones combinacionales deben estar asignadas a los parámetros de los bloques funcionales para su correcta ejecución y las variables asociadas a los bloques funcionales serán las que generen la ejecución de las acciones, de esta forma se tendrá que las funciones combinacionales estarán en la parte derecha de la sentencia y el bloque funcional estará en la parte izquierda, recibiendo la asignación de la evaluación de la función combinacional.

Así se tendrá, la codificación en ST de lo diseñado en SFC.

A. **Elementos del lenguaje**

Los elementos utilizados para generar la conexión entre el SFC y el ST se pueden observar en la tabla siguiente, donde se relaciona el operador y su utilización.

Tabla 7: Operadores ST para A.L.S.

Operador	Descripción	Ejemplo y Valor de la Expresión	Prioridad
( )	Paréntesis	(2 * 3) + (4 * 5) Valor: 26	Alta
	Llamado a una Función	**CONCAT**(‘AB’,’CD’) Valor: ‘ABCD’	
NOT	Complemento	NOT TRUE Valor: FALSE	
&, AND	AND Lógico	TRUE AND FALSE Valor: FALSE	
OR	OR Lógico	TRUE OR FALSE Valor: TRUE	
:=	Asignación	A := 25 Valor de A: 25	Baja

Fuente: IEC 61131-3, Operadores ST.

Al igual que la necesidad de estos operadores, es necesario para generar la conexión con el SFC, la utilización bloques funcionales estándar consignados en el apartado 3.4.5.

Con estos elementos se constituye para A.L.S en relación al IEC 1131-3 la generación en ST partiendo del diseño realizado en SFC.

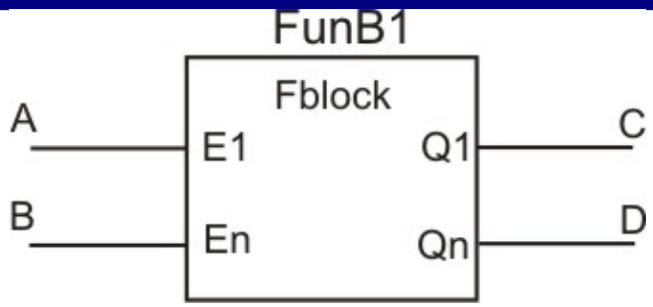
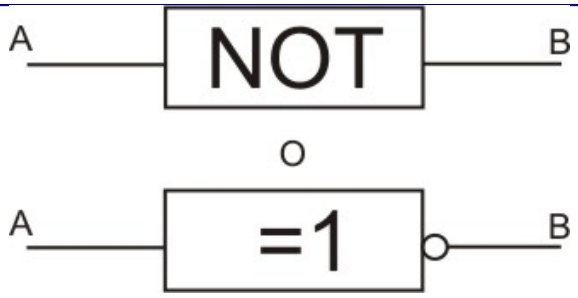
3.4.3 Diagrama de Bloques Funcionales (FBD)

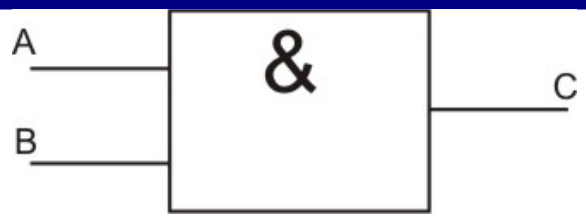
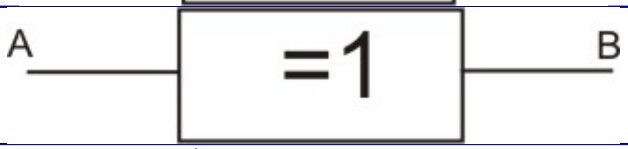
Con relación al FBD, para la generación según lo diseñado en SFC se debe tener en cuenta: Primero, en la parte de declaración, teniendo claro que las variables concernientes a las transiciones son variables de entrada para el A.L.S, de igual forma que las variables asociadas a la acciones son variables de salida, aunque; puede darse el caso en que una variable asociada a las acciones sea requerida en la condición de transición, por lo cual esta variable será de entrada y salida y deberá declararse así. Segundo, en la parte de codificación se deben utilizar las representaciones graficas de las compuertas (ver tabla 8. Operadores FBD para A.L.S), para cubrir la parte combinacional que se encuentra en las transiciones, estas combinaciones deben estar acopladas a las representaciones de los bloques funcionales, y de las salidas de estos bloque funcionales se generan las ejecuciones de las acciones.

A. **Elementos del lenguajes**

Los elementos para la representación de las compuertas en FBD están expresados en la siguiente tabla.

Tabla 8. Operadores FBD para A.L.S.

Operador	Descripción y *Ejemplo ST*	Ejemplo
	Llamado a un bloque funcional (FunB1 (E1 := A, En := B); C := FunB1.Q1; D := FunB1.Qn;)	
NOT	Complemento (B := NOT A;)	

Operador	Descripción y *Ejemplo ST*	Ejemplo
&, AND	AND Lógico (C := A AND B;)	
OR	OR Lógico (C := A OR B;)	
	Asignación (B := A)	

Fuente: IEC 61131-3, Operadores FBD.

De igual forma para el completo cubrimiento de los A.L.S se requiere de bloques funcionales que se relacionan en el apartado 3.4.5.

Así queda expuesto, los elementos que posibilitan la generación, a partir del diseño realizado mediante el SFC de la codificación en FBD.

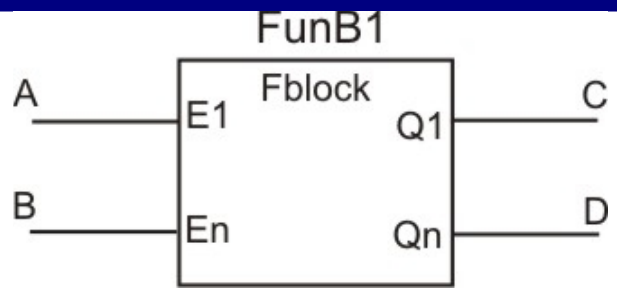
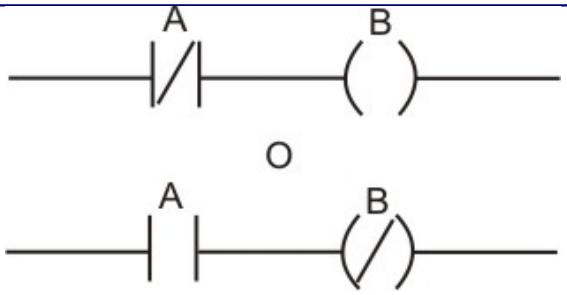
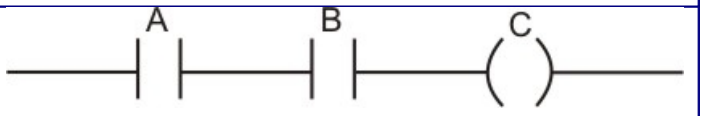
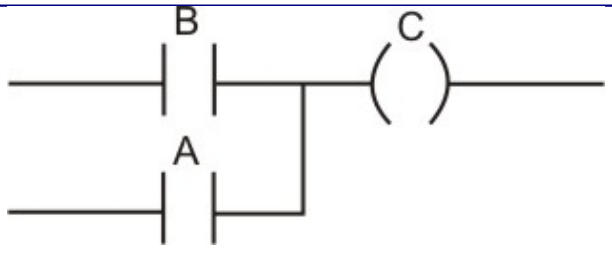
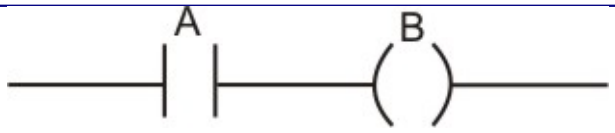
3.4.4 Diagrama de Contactos (LD)

El diagrama de contactos posee bastantes similitudes con el FBD, en lo relacionado a la representación de los bloques funcionales, así que para el caso de la generación del LD a partir del SFC se debe tener en cuenta: Primero, en la parte de declaración, teniendo claro que las variables concernientes a las transiciones son variables de entrada para el A.L.S, de igual forma que las variables asociadas a la acciones son variables de salida, aunque; puede darse el caso en que una variable asociada a las acciones sea requerida en la condición de transición, por lo cual esta variable será de entrada y salida y deberá declararse así. Segundo, para la parte de codificación, se tendrá la organización en las líneas en serie o paralelo (ver tabla 9. Operadores LD para A.L.S) para cubrir la parte combinacional consignada en las transiciones, esta parte combinacional serán las entradas a los bloques funcionales, encargados de generar las salidas para la ejecución de las acciones.

A. **Elementos del lenguaje**

Las representaciones en LD, para los operadores de compuertas se relacionan en la siguiente tabla.

Tabla 9. Operadores LD para A.L.S.

Operador	Descripción y *Ejemplo ST*	Ejemplo
	Llamado a un bloque funcional (FunB1 (E1 := A, En := B); C := FunB1.Q1; D := FunB1.Qn;)	
NOT	Complemento (B := NOT A;)	
&, AND	AND Lógico (C := A AND B;)	
OR	OR Lógico (C := A OR B;)	
	Asignación (B := A)	

Fuente: IEC 61131-3, Operadores LD.

De igual forma para el cubrimiento de la generación de codificación para los A.L.S se debe poseer los bloques funcionales relacionados en el apartado 3.4.5.

3.4.5 Bloques Funcionales Estándar

La norma IEC 1131-3 define algunos bloques funcionales estándar que cubren funcionalidades importantes para el trabajo de los PLC's, los cuales se enuncian a continuación.

Elementos biestables (flipflops)
Detectores de flancos
Contadores
Temporizadores

La tabla 10 lista los bloques funcionales estándar utilizados en el presente proyecto, como parte importante del mismo y del manejo de los Automatismos Lógicos Secuenciales

Tabla 10. Bloques funcionales estándar.

Nombre del Bloque Funcional estándar y sus entradas	Parámetros de salida	Pequeña descripción
Biestables		
SR (S,R,	Q) Q)	Prioridad al Set Prioridad al Reset
RS (S,R,		
Detectores de Flanco		
R_TRIG{->} (CLK,	Q) Q)	Flanco Ascendente Flanco Descendente
F_TRIG{-<} (CLK,		
Contadores		
CTU (CU,R,PV,	Q,CV) Q,CV)	Ascendente Descendente
CTD (CD,LD,PV,		
Temporizadores		
TP (IN,PT,	Q,ET)	A desconexión memorizado
	Q,ET)	A conexión
TON {T---0} (IN,PT,	Q,ET)	A desconexión
TOF {0---T}		

Nombre del Bloque Funcional estándar y sus entradas	Parámetros de salida	Pequeña descripción
(IN,PT,		

Fuente: IEC 61131-3, Bloques Funcionales.

Los tipos de datos correspondientes a estas variables de entrada y de salida, son listados en la tabla 11, referenciados por su nombre.

Tabla 11. Tipos Variables, bloques funcionales estándar.

Entradas / Salidas	Representación	Tipo de dato
R	Entrada para reset	BOOL
S	Entrada para set	BOOL
Q	Salida por excelencia estándar	BOOL
CLK	Variable con sensor para cambio de estado	BOOL
CU	Entrada de pulso para contar uno	R_EDGE
CD	Entrada de pulso para descontar uno	R_EDGE
LD	Carga el valor al contador	INT
PV	Entrada del valor a contar	INT
CV	Valor actual del contador	INT
IN	Entrada del temporizador	BOOL
PT	Valor a temporizar	TIME
ET	Tiempo que ha transcurrido	TIME

Fuente: IEC 61131-3, Tipos de variables bloques funcionales.

Ahora por medio de una serie de ejemplos de código se intentará dar una idea del manejo en general de los bloques funcionales estándar que propone la norma IEC 1131-3.

Ejemplos de uso:

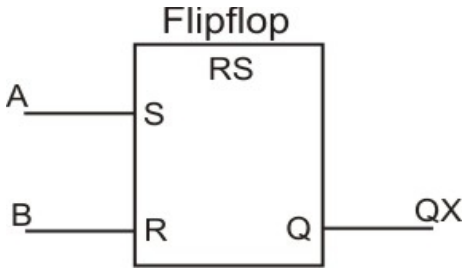
Biestable (Flipflop):

- Declaración:

```
VAR
        Flipflop    :    RS;
        A           :    BOOL;
        B           :    BOOL;
        QX          :    BOOL;
END_VAR
```

- Lenguajes Gráficos (FBD, LD):

Figura 28. Biestable.



- Lista de instrucciones (IL):
  LD A
  ST Flipflop.S
  LD B
  ST Flipflop.R
  CAL **Flipflop**
  LD Flipflop.Q
  ST QX

- Texto Estructurado (ST):
  Flipflop(S := A, R := B);
  QX := Flipflop.Q;

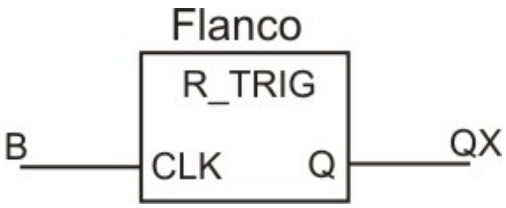
En este ejemplo se muestra como se hace una declaración correcta de las variables implicadas, como se enlazan las variables A y B a las entradas S y R del biestable (con prioridad al RESET) respectivamente y como se asigna a la variable QX la salida del bloque funcional, es decir; en cualquier momento del programa se podrá saber el estado de este Flipflop solo consultando QX. El valor del biestable será 1 siempre y cuando reciba una señal de 1 por la entrada S a través de A y si el valor de B es cero en ese momento. Por otra parte tomará el valor de 0 siempre que reciba una señal de 1 por B sin importar el valor que tenga A debido a su prioridad al RESET.

Detección de flanco:

- Declaración:
  VAR
  Flanco : RS;
  A : BOOL;
  B : BOOL;
  QX : BOOL;
  END_VAR

- Lenguajes Gráficos (FBD, LD):

Figura 29. Detector de flanco.



- Lista de instrucciones (IL):
  LD B
  ST Flanco.CLK
  CAL **Flanco**
  LD Flanco.Q
  ST QX

- Texto Estructurado (ST):
  Flanco (CLK := B);
  QX := Flanco .Q;

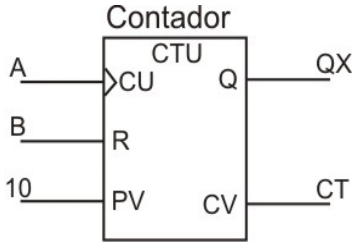
En este ejemplo se muestra como se declara de manera acertada las variables implicadas y como se enlazan las variables tanto de entrada como de salida de un detector de flanco ascendente a las respectivas variables internas del POU. Al detector de flanco ascendente se le asocia la variable B y el tomará el valor de 1 cuando el valor de B cambie de 0 a 1 en los otros casos será 0.

Contador:

- Declaración:
  VAR
  Contador : CTU;
  A : BOOL;
  B : BOOL;
  QX : BOOL;
  CT : INT;
  END_VAR

- Lenguajes Gráficos (FBD, LD):

Figura 30. Contador.



- Lista de instrucciones (IL):
  LD A
  ST Contador.CU
  LD B
  ST Contador.R
  LD 10
  ST Contador.PV
  CAL **Contador**
  LD Contador.Q
  ST QX
  LD Contador.CV
  ST CT

- Texto Estructurado (ST):
  Contador (CU := A, R := B, PV := 10);
  QX := Contador.Q;
  CT := Contador.CV;

En este ejemplo se muestra la declaración de un contador ascendente, de las variables internas del POU que se utilizaran en el caso y como estas se enlazan a las entradas y salidas de este bloque funcional. Este contador deberá contar desde 0 hasta 10, aumentando la cuenta de uno en uno cada ves que reciba una señal de 1 por su entrada CU a través de la variable A, también se debe tener en cuenta que cada vez que se reciba una señal de 1 por la entrada R a través de B la cuenta volverá a cero. En la salida CV se tendrá la información de la cuenta que se lleva, en todo momento y cuando esta cuenta sea igual al número especificado por la entrada PV (Para el caso 10), se activará la salida Q.

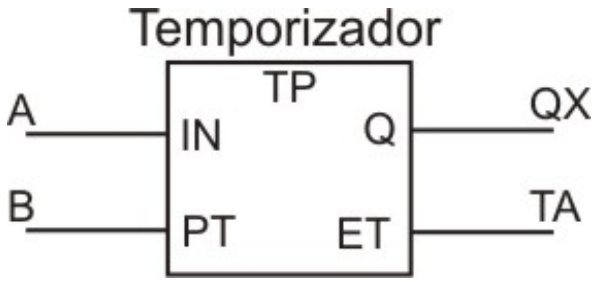
Temporizador:

- Declaración:
  VAR
  Temporizador : TP;
  A : BOOL;
  B : TIME := T#6S;
  QX : BOOL;

```
        TA              :       TIME;
END_VAR
```

- Lenguajes Gráficos (FBD, LD):

Figura 31. Temporizador.



- Lista de instrucciones (IL):

```
LD    A
ST    Temporizador.IN
LD    B
ST    Temporizador.PT
CAL   Temporizador
LD    Temporizador.Q
ST    QX
LD    Temporizador.ET
ST    TA
```

- Texto Estructurado (ST):

```
Temporizador ( IN := A, PT := B );
QX := Temporizador.Q;
TA := Temporizador.ET;
```

En este ejemplo se muestra la declaración de un temporizador, de las variables internas del POU que se utilizaran en el caso y como estas se enlazan a las entradas y salidas de este bloque funcional. Para el caso se usó un temporizador con retardo a la desconexión memorizado, es decir cuando se reciba una señal de 1 por IN se activará la salida Q y comenzará la cuenta, hasta que alcance el tiempo especificado a través de la entrada PT. Cuando alcance este tiempo se desconectará el temporizador, desactivando la salida Q. Este temporizador en particular, una vez recibe la señal de activación por IN termina su función sin importar si esta señal se cae o no (Por su característica de memorización).

3.5 IMPACTOS

Con el recorrido que se ha realizado a lo largo de los capítulos expuestos, se ha mostrado la importancia de establecer una forma de trabajo que logre solucionar las dificultades que se presentan en el campo de la automatización industrial, esta forma de trabajo la consiga el estándar IEC 1131 en su contenido, es entonces observable que el aporte es impactante, en la medida que atiende necesidades y soluciona problemáticas.

En términos, del cubrimiento del estándar es claro que su campo de aplicación es amplio, por lo tanto; el acotamiento realizado en este trabajo esta fundamentalmente basado en recorrer un trayecto de un proceso que requiere seguir fortaleciendo la propuesta del estándar, con el acogimiento y desarrollo de trabajos similares.

Ahora, en el contexto de los A.L.S, el aporte del estándar sigue siendo de gran importancia, debido; a que su propuesta sigue sin desviación (ver capitulo dos), entonces se sigue en el camino del fortalecimiento de de una forma de trabajo independiente de fabricantes y con prioridad al diseño.

3.5.1 Técnico

Dentro de las ventajas que posibilita el estándar se encuentra reducir el esfuerzo humano en entrenamiento, depuración, mantenimiento y consultoría, de igual forma la posibilidad de crear proyectos reutilizables e independientes del fabricante, que generara una disminución en los tiempos de desarrollo, en el esfuerzo de codificación, y en los errores de compilación, montaje e implementación. Estas entre otras ventajas se deducen de la norma, ahora; dentro del contexto del presente trabajo se recogen estas ventajas llevándolas a un plano de materialización en una herramienta software (Ver capitulo cuatro Autologic 1131) donde los procesos de entrenamiento, depuración, mantenimiento y consultaría al igual que la creación de proyectos reutilizables se logran de una manera mas ágil, mediante la herramienta software basada en el estándar IEC 1131-3, aunando a esto; tomando la forma de trabajo Top Down del estándar se focaliza el esfuerzo en la labor de diseño generando así la apropiación del SFC como el lenguaje de diseño y a partir de éste generar codificación en los cuatro lenguajes expuestos en este capitulo.

3.5.2 Económico

El impacto económico se obtiene, en la medida que los fabricantes converjan al estándar, como lo vienen haciendo empresas como

Siemens, debido; a que poseer PLC con el reconocimiento del estándar, posibilita que para un sistema de control los dispositivos puedan ser de casas diferentes, y con esto; que los diseños no estén atados a cada fabricante, es decir; poseer un diseño, la codificación y poder ejecutarlo en un número indefinido de industrias sin importar que PLC tengan, sin tener que invertir en un nuevo diseño, una nueva codificación, en la capacitación y en el mantenimiento es un impacto muy importante.

Por otra parte el costo en tiempo de capacitación y implementación se vera reducido, esto debido; a que quien entienda y aprenda sobre el estándar podrá desempeñar sobre cualquier PLC.

3.6 VIABILIDAD

Con lo anteriormente expuesto, queda claro que el estándar posee amplias ventajas para que usuarios exijan a las casas fabricantes la aplicación de éste en sus dispositivos de control. Así pues; la viabilidad del estándar y las aplicaciones como el presente trabajo que proponen una forma de trabajo basados en él, poseen un campo de expansión apreciable, además; que no solo se implementa una norma, sino que con ella y en virtud de un conocimiento de problemáticas en el campo de la automatización se realiza una propuesta de solución.

3.6.1 Técnica

La viabilidad queda expresada, en términos que el trabajo alcanza el cumplimiento de sus objetivos, es decir; posibilita, un marco conceptual y una herramienta software para el trabajo del IEC 1131-3, con esto la difusión, ventajas y propuesta descritas por la norma tienen un despliegue puntual en un contexto preciso de los A.L.S, así se hace viable, en la medida que su utilización esta colocando en escena la reproducción de un cúmulo de información que posibilita el alcance de un estado ideal; de independencia de fabricantes, fortalecimiento del diseño y con esto un puente comunicacional en el campo de la automatización.

3.6.2 Económica

Se consigna en la aplicación de la herramienta y la investigación para los estudiantes de la Universidad Industrial de Santander sin costo alguno, con lo cual la difusión de lo importante esta garantizada.

3.7 RESULTADOS DE LA INVESTIGACIÓN

En síntesis podemos enumerar los resultados obtenidos así:

- El SFC por los elementos constitutivos, por su riqueza interpretativa y su funcionalidad, además de los fundamentos que se consignan en el estándar de donde proviene, es denominado un lenguaje de diseño.
- La conexión realizada entre el diseño SFC y los demás lenguajes expuesto en el estándar para los A.L.S.
- La estructuración de las macroetapas como bloques funcionales que posibilitan un despliegue del diseño, y una encapsulación del mismo.
- La aplicabilidad para el contexto de los A.L.S de la propuesta de la norma IEC 1131-3.
- La estructuración de las unidades de organización de programa (POU), en relación al programa, en la parte de codificación para los lenguajes IL, ST, FBD y LD como segmento secuencial y segmento combinacional.

Con la síntesis realizada queda concluida la exposición del marco conceptual que sustenta el trabajo desarrollado.

4. AUTOLOGIC 1131

4.1 INTRODUCCIÓN

Al haber evidenciado una problemática y habiendo llevado a cabo un ordenamiento de ideas como producto de una exploración de conocimiento, se realiza la propuesta para aportar a la solución de dicha situación; siguiendo los lineamientos de la investigación proyectiva. Dicha propuesta se presenta como el Autologic 1131, que materializa los conceptos producto de la investigación, expuestos en el capítulo anterior; en una herramienta software que brinda al usuario la posibilidad de tener un punto de partida para el desarrollo de proyectos sobre automatismos lógicos secuenciales; focalizando sus esfuerzos en el diseño, como afirman los resultados de la investigación.

Siguiendo el orden de ideas que se proponen y exponen en el capítulo anterior, el Autologic 1131 posibilita la escritura de programas de control en el lenguaje denominado SFC (Gráfico Secuencial de Funciones), siguiendo las pautas que presenta el estándar IEC 1131-3 y concibiéndolo como un diseño a bajo nivel. Asistiendo este diseño, para que el usuario lo lleve a cabo casi que de una manera intuitiva, haciéndolo mas agradable y ordenado que con la forma de trabajo manual. Con la posibilidad de generar los códigos fuentes del programa de control (previamente diseñado en el modulo de diseño en SFC), en cualquiera de los lenguajes restantes pertenecientes y descritos en el IEC 1131-3 que son FBD (Diagrama de Bloques Funcionales), LD (Diagrama de Contactos), ST (Texto Estructurado) e IL (Lista de Instrucciones).

4.2 ALCANCES TÉCNICOS Y METODOLÓGICOS

Habiendo explorado el estándar IEC 1131 de una manera general y habiendo focalizado específicamente el interés del presente proyecto en los automatismos lógicos secuenciales; el Autologic 1131 al ser producto materializado de dicha investigación, contextualiza su área de acción en este tipo de automatismos, basándose específicamente en la tercera parte del estándar (IEC 1131-3) cuyo interés se centra en todo lo que a manejo de lenguajes de programación se refiere (operandos, operadores, sintaxis, declaración de variables, etc.).

El Autologic 1131 plantea que en primera instancia se realice el diseño del automatismo de una manera ordenada y congruente, guardando cuidadosamente las normas sintácticas de la norma. Posibilitando para esto la edición de diagramas SFC manejando como elementos de diseño: etapas, transiciones, acciones, bifurcaciones en Y, bifurcaciones en O y macroetapas (Como elemento acoplado para el entendimiento de subrutinas en forma de bloques funcionales).

En segunda instancia propone hacer una verificación del diseño que en forma, debe guardarse consistente. Brindando la opción de hacer el estudio de todos y cada uno de los elementos del diagrama, con solo dar un mando. Cuando esta orden se hace, se presentan los resultados al usuario, basándose en las pautas preestablecidas para del diseño (ver 4.4.1 Esquema Secuencial de Funciones, C. Validación, Especificaciones de validación). Todo esto con el fin de que el usuario verifique que su diseño cumple en sintaxis y congruencia según el estándar, y que está listo para ser llevado bien sea directamente a un PLC, o a uno de los otros lenguajes de programación del IEC 1131-3.

Por último si se requiere, propone hacer el paso del diseño en SFC a cualquiera de los otros lenguajes propuestos en la tercera parte de la norma IEC 1131, brindando para esto, la posibilidad de concretarlo con simplemente hacer un mando (siempre y cuando el gráfico sea verificado satisfactoriamente). Y así tener por lo menos un prototipo inicial de la codificación del programa de control que será llevado al PLC en todos los lenguajes del estándar IEC 1131.

Siendo el Autologic 1131, en síntesis, una herramienta software que: Se contextualiza en los automatismos lógicos secuenciales, basándose en la norma IEC 1131. Que asiste y valida el diseño de estos automatismos en el lenguaje SFC, que a partir de este diseño; genera codificación en IL, LD, ST y FBD y que brinda la posibilidad de portar todos sus contenidos ya sea generando archivos contenedores tanto del diseño, como de la codificación generada en cualquiera de los otros lenguajes, o imprimiendo los mismos.

4.3 FUNCIONALIDAD GENERAL

El Autologic 1131 estructura su desarrollo y su forma de desempeño en 5 módulos funcionales.

1. Módulo de diseño: Encargado de la generación visual del gráfico funcional correspondiente al SFC, guardando que este siempre sea consistente y legible, implementando funcionalidades como las de

eliminar, seleccionar, editar, etc. Que dan paso a una interfaz interactiva y amigable, sin que el usuario tenga que preocuparse por la presentación y el orden del diseño.

2. Módulo de Verificación: Encargado de la validación del diseño bajo las especificaciones que están dadas para el SFC en el estándar IEC 1131-3 en 2 etapas:

    1. En primera instancia lo relacionado con la sintaxis prestando guía con asistentes y mensajes al usuario durante el diseño.
    2. En segunda instancia se puede verificar cuando se guste la congruencia del gráfico, de tal manera; que si todo está en orden permita llevar a cabo la generación del código fuente en los 4 lenguajes de programación restantes de la especificación IEC 1131-3.

3. Módulo de Generación: Encargado de la generación de la codificación a partir del gráfico SFC previamente verificado satisfactoriamente, en los cuatro lenguajes de programación restantes especificados en la norma IEC 1131-3. Estableciendo las interfaces entre el primero, y cada uno de de ellos; para hacerlo y por último presentarlos de una manera congruente.
4. Módulo de Portabilidad: Como su nombre lo indica es el encargado de brindar la posibilidad tanto al diseño en SFC, como a cualquiera de los otros lenguajes generados a partir de este de ser portados por el usuario. Permitiendo guardar esta información en archivos XML, con todas las ventajas que esto trae para el transporte de los mismos; posibilitando de manera sencilla, abrir todos o cada uno de manera independiente en cualquier terminal de trabajo donde se encuentre el Autologic 1131.

    Como funcionalidades complementarias:

    1. La posibilidad de imprimir toda la información consignada en la herramienta, pudiendo visualizar en una vista preliminar la misma para mayor seguridad sobre la configuración del área de impresión y demás detalles de interés.
    2. La posibilidad de exportar en archivos planos de texto (con extensión “.txt”), la información correspondiente a los lenguajes textuales de la norma IEC 1131-3; el IL (Lista de Instrucciones) y el ST (Texto Estructurado). Permitiendo ser visualizados en cualquier editor de texto compatible con esta extensión.

    Todo esto facilitando su transporte y reutilización, ya sea sobre el mismo Autologic 1131 o sobre algún software específico de algún fabricante para dar consolidación al montaje previamente diseñado.

5. Módulo de Presentación: Encargado de presentar una interfaz gráfica agradable y amigable para el usuario, que le permite un manejo del programa sencillo e intuitivo.

4.3.1 Especificación de variables

Debido a que las especificaciones de los PLC's varían según los modelos y los fabricantes, la representación de direcciones de memoria se plantea que sea especificada en el momento en el cual se lleve el código fuente al sistema del PLC y que para efectos del diseño y muchas veces también de codificación (dependiendo de las funcionalidades del PLC), se usen para la representación de variables, etiquetas representativas, que sirvan de guía para una mejor concepción y un mejor entendimiento del sistema de control resultante.

El Autologic 1131 maneja este tipo de variables guardando como normas esenciales, las siguientes especificaciones:

- Las variables no deben contener espacios en blanco intermedios.
- Las variables aceptan todos los caracteres alfanuméricos, es decir; cualquier combinación entre letras y números.
- Las variables además de los caracteres alfanuméricos, aceptan los caracteres "%" (Porcentaje), "." (Punto) y "_" (Guión de Piso).
- Las variables deben tener por lo menos una letra en su nombre.
- Las variables pueden tener como máximo una longitud de 23 caracteres.

Implementando una validación al ingreso de los nombres de las variables al sistema, guardado que solamente se acepten variables cuyo nombre obedezca a las normas antes expuestas.

4.3.2 Especificación de macroetapas

Cuando se aplican las técnicas del diseño en SFC para la representación de procesos complejos, se hace necesario encapsular ciertos subprocesos, que según el estándar pueden hacerse como bloques de acción asociados a una etapa, es decir; existen etapas a las cuales se les asocia un bloque de acción que contiene el subproceso secuencial, representado con un gráfico SFC el cual permanecerá en ejecución de manera cíclica mientras la etapa a la que pertenece permanece activa.

Estos subprocesos conservan todas las funcionalidades brindadas por el SFC para la escritura de programas, tanto así que muchas veces los subprocesos llegan a ser mucho más complejos que el mismo diagrama principal. Además estos subprocesos pueden a su vez albergar en su

diagrama otros subprocesos, que a su vez pueden contener otros y otros sucesivamente; lo que hace que sea posible un anidamiento con un nivel indeterminado y ramificado en infinidad de formas.

El Autologic 1131 recoge este tipo de etapas incluyéndolas como macroetapas; uno de sus elementos esenciales de diseño. Estas en el área de diseño son representadas de una manera especial con relación a las etapas comunes (Ver Figura 27).

Cuando se incluye una macroetapa en el área de diseño del Autologic 1131, debe ser editada definiendo desde una etiqueta que describe el subproceso, hasta el diseño de la subrutina que la compete, brindando para esto un área de diseño totalmente independiente y que salvo un par de condiciones de congruencia se estructura de la misma manera que el área de diseño SFC del editor principal; permitiendo utilizar todas las funcionalidades de diseño para este nuevo gráfico en particular.

La verificación de sintaxis del gráfico definido para el subproceso de la macroetapa, se rige por las siguientes normas:

- La etapa inicial puede tener si se requiere transiciones entrantes lo que llevaría a procesos cíclicos (También es válido si no las tiene).
- No puede haber más de una etapa que no tenga transiciones salientes, pues es incongruente que el proceso tenga dos finales de proceso distintos.
- Es válido que las todas las etapas tengan transiciones salientes lo que haría al proceso cíclico.
- También es válido (y es lo recomendado) que en el gráfico solo haya una etapa sin transiciones salientes, lo que la haría la etapa única de fin del proceso.

La concepción que en el autologic 1131 se le da a las macroetapas para su representación en el código a generar, es mediante bloques funcionales (Ver 2.3.3 Unidades de organización del programa); debido a que las características que presentan las macroetapas, encajan en estas unidades de organización, por ejemplo: Cantidad de variables de salida (Solo en casos muy remotos será una única salida como lo tienen las funciones), que al llamarse con las mismas entradas puede presentar salidas distintas y, por último algo muy importante; que los bloques funcionales son compatibles con todos los lenguajes del IEC 1131 mientras que las funciones no son soportadas en IL y en algunos casos tampoco en ST.

Cuando se llevan las macroetapas a los códigos generados por el Autologic 1131, se les asigna una variable de entrada llamada “IN” la cual

es la señal de entrada para la iniciación del proceso y en los casos en los que el proceso no tiene una ejecución cíclica, sino; que tiene una única etapa de terminación; también controla la finalización del proceso. La variable “IN” tiene una correspondencia directa con la variable de estado asociada a la macroetapa; lo que hace que cuando esta resulte activa, el proceso comience su ejecución.

4.3.3 Especificación de funciones combinacionales para las condiciones

En elementos básicos del diseño de SFC como lo son las transiciones, las divergencias y convergencias; se especifican funciones lógicas combinacionales, las cuales representan las condiciones para dar paso de una(s) etapa(s) a otra(s). Estas funciones combinacionales pueden tener la complejidad que sea necesaria para la condición. Sin embargo; el autologic 1131 plantea que estas se especifiquen en una manera reducida y llevadas hacia una forma similar a las famosas listas de mintérminos[7], es decir; utilizar bien sea el algebra de Boole[8] o alguno de los procedimientos para la reducción de funciones lógicas combinacionales y obtener su equivalente en suma de multiplicaciones. En el diálogo donde se ingresa esta mencionada condición, solo se aceptarán funciones del tipo anteriormente mencionado (suma de multiplicaciones) y también si tiene en cuenta las siguientes pautas:

Tabla 12. Operadores función combinacional, condición transición.

Operador	Descripción	Ejemplo de uso (Autologic 1131)	Prioridad
" ' " la comilla sencilla	que representa la única operación unaria que es aceptada en la función, la cual niega el valor de la variable que la precede	Var1 ’	Alta
" * " el asterisco	Representa la operación lógica **AND** o conjunción	Var1 * Var2	Intermedia
" + " el	representa la	Var1 + Var2	Baja

[7] M. MORRIS, Mano. Diseño Digital. Productos estándar entre variables lógicas que representan una parte del diagrama de Venn y comúnmente son unidos por medio de sumas.

[8] M. MORRIS, Mano. Diseño Digital. Sistema deductivo que define reglas, teoremas y propiedades para el tratamiento sistemático de la lógica de manera algebraica.

Operador	Descripción	Ejemplo de uso (Autologic 1131)	Prioridad
símbolo suma	operación lógica **OR** o suma		

- Además la negación no puede ser asociativa, se puede negar una variable a la vez.

Ejemplos de funciones:

Tabla 13: Ejemplos Función Combinacional.

Ejemplo	Validez
Var1	Válido
Var1’	Válido
Var1 ’ ’	Inválido
(Var1) ’	Inválido
Var1 ’ + Var3 + Var2 ’ + Var4	Válido
Var1 ’ * Var2 ’ * Var3 * Var4	Válido
(Var1 + Var2) * Var3	Inválido
Var1' * Var2 * Var3 ' + Var4 * Var5 + Var6 ' + Var3 * Var7 * Var1	Válido
Var1 + +Var4	Inválido

4.3.4 Especificación de las acciones

Como se ha mencionado antes, en cada una de las etapas de un gráfico SFC son asociadas acciones, las cuales; son operaciones que se ejecutarán en el momento en el que la etapa a la que estén asociadas este activa (Básicamente es donde se realizan los mandos que dan las salidas del sistema).
El Autologic 1131 plantea para el manejo de estas operaciones, algunas acciones predefinidas, que posibilitan que el usuario pueda asociarlas a las etapas, solo con personalizar ciertos parámetros que complementan su definición y que se explican más detenidamente a continuación:

- Accionamiento Directo: Este tipo de acción, cumple asignando una señal de uno en la variable especificada, mientras la etapa permanezca activa. Cuando la etapa en la que está asociada la acción se desactiva, esta señal se cae. Esto debido a que está asociada directamente a la variable de estado correspondiente a la etapa.

- Accionamiento Inverso: Este tipo de acción, cumple asignando una señal de cero en la variable especificada, mientras la etapa permanezca activa. Cuando la etapa en la que está asociada la acción se desactiva, esta señal cambia a uno. Pues; está asociada inversamente a la variable de estado correspondiente a la etapa (es el complemento al estado de la etapa).
- Activación: Este tipo de acción busca asignar una señal de uno a la variable especificada, y que permanezca de esa manera independientemente del estado de la etapa asociada. Para esto; se define una célula de memoria (FlipFlop), que en últimas es a la que se le asocia la variable de estado correspondiente a la etapa actual y que a su vez; es la encargada ya sea de mantener esta señal o de hacerla caer; si se especifica en una acción asociada en una etapa posterior sobre el proceso.
- Desactivación: Este tipo de acción busca asignar una señal de cero a la variable especificada, y que permanezca de esa manera independientemente del estado de la etapa asociada. Para esto; se define una célula de memoria (FlipFlop), que en últimas es a la que se le asocia la variable de estado correspondiente a la etapa actual y que a su vez es la encargada ya sea de mantener esta señal o de hacerla uno; si se especifica en una acción asociada en una etapa posterior sobre el proceso.
- Temporizar: Para este tipo de acciones el usuario define; que tipo de temporizador se utilizará y la variable a la que se asociará la salida del mismo (Ver Tabla 11). La variable especificada presentará un comportamiento, que depende de cual de los tres tipos de temporizadores se decida usar:
    - Retardo a la conexión: Cuando se elige esta opción, la cuenta comienza cuando se activa la etapa, y al cabo del tiempo estipulado la variable asociada se activará en uno (Funciona mientras está la etapa activa).
    - Retardo a la Desconexión**:** Cuando se elige esta opción se activará en uno la variable asociada, comenzando la cuenta en el momento en el que la etapa asociada se active, y cuando expire el tiempo estipulado se desactivará la variable asociada es decir; se pondrá en cero (Funciona mientras está la etapa activa).
    - Retardo a Desconexión con Memoria: Cuando se elige esta opción las cosas funcionan de la misma manera que con la opción retardo a la desconexión; salvo que esta funciona independientemente del estado de la etapa a la cual está enlazada, gracias a sus propiedades de memorización.
- Contadores: El trabajo con contadores es un poco diferente, debido a que con el mismo contador se puede trabajar desde diferentes

etapas. Por esto se deben definir a priori a su utilización en las acciones. Para llevar a cabo la definición de contadores, el Autologic 1131 provee un asistente en el se adjunta el contador que requiera definir, que puede ser de dos tipos:

- Incremental: Este contador deberá contar desde cero hasta el número especificado, aumentando la cuenta de uno en uno, cada vez que reciba una señal de uno en la entrada de aumento a la cuenta. Cada vez que se inicialice el contador, la cuenta volverá a ser cero.
- Decremental: Este contador deberá contar desde el número especificado hasta cero, disminuyendo la cuenta de uno en uno, cada vez que reciba una señal de uno en la entrada de disminución a la cuenta. Cada vez que se inicialice el contador la cuenta volverá a ser el número especificado.

Una vez definido el contador; se permitirá trabajar con este desde cualquier etapa, posibilitado así; el trabajo con dos nuevos tipos de acción:

- Inicializar: Que envía una señal por la línea del reset al contador seleccionado para así ponerlo en su estado inicial.
- Impulso: Que envía una señal de cuenta para incrementar o decrementar la misma, en el contador elegido. En este tipo de acción también se asocia la variable a la cual, al concluir la cuenta se le enviará la señal de salida del contador (Con el uso de el mismo contador desde diferentes etapas, es permitido; asociar varias variables a la salida del contador).

Por último se debe tener en cuenta que no es sintacticamente permitido que una etapa no contenga acciones asociadas.

4.4 TRATAMIENTO DEL DISEÑO

Una de las partes más importantes tanto conceptualmente como en aplicabilidad del Autologic 1131, es la parte de diseño. En la cual se realizó una fuerte labor de ingeniería, con el fin de conseguir materializar un asistente de diseño, que permitiese incluir, seleccionar, editar, modificar y eliminar elementos en el gráfico SFC de manera interactiva, coherente, intuitiva, y amigable. Que siguiendo una serie de pautas de diseño permite a un gráfico con una estructura tan flexible, siempre permanecer legible, ordenado y bien presentado.

4.4.1 Esquema Secuencial de Funciones (SFC)

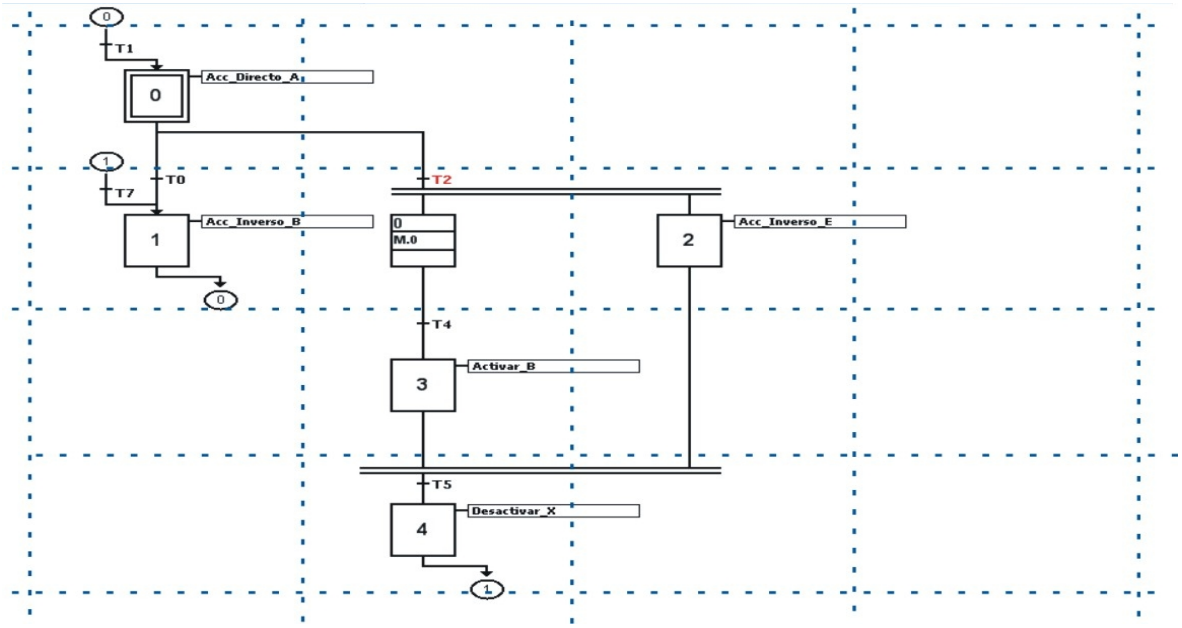
A. Interfaz

El área de diseño, esencialmente se aloja en un formulario que se carga sobre el formulario principal, llamado editor principal. Sin embargo; al ser incluidas macroetapas en el gráfico SFC, cada una de ellas posee su propia área de diseño, en un editor para el diseño de la subrutina que la compete, el cual se presenta en un formulario totalmente independiente.

Como punto de partida para el desarrollo del diseño, tanto en el editor principal como en el editor de las macroetapas; cuando se crea un nuevo proyecto o se incluya una nueva macroetapa respectivamente, se inicializa por defecto en el gráfico, la etapa inicial del mismo (representada por dos cuadros concéntricos) que para el Autologic 1131 siempre tendrá el cero como identificador.

Para lograr tener un diseño permanentemente ordenado se utilizó como estrategia la división imaginaria del área de diseño, en casillas con dimensiones preestablecidas que alojarán cada uno de los elementos.

Figura 32. División Imaginaria del área de diseño



Lo que posibilitó referenciar los elementos y reordenarlos, cuando ocurran inclusiones o eliminaciones de ellos. Acciones que demandan

una exploración y un re acomodamiento dinámico del gráfico; que se hace a través de un manejo de índices de matriz, lo que resulta más sencillo que trabajar con elementos gráficos y sus coordenadas estáticas.

B. **Asistencia**

Como se ha mencionado en repetidas ocasiones el Autologic 1131 brinda las herramientas apropiadas, para que el usuario realice un diseño de manera ordenada e intuitiva y guía el diseño hacia su realización sólida y coherente. Todo esto a través de una serie de asistentes, que organizan las especificaciones de los elementos a incluir o editar; en forma de parámetros, para que el usuario con solo llevar a cabo la edición de estos en los debidos asistentes, estos mismos organicen la información y la entreguen al módulo de diseño para que se genere su representación en el gráfico SFC.

Además la asistencia del diseño se complementa con la validación del SFC. Siendo la asistencia una de las herramientas vitales que usa la validación, con el fin de alcanzar las metas de solides y congruencia no solo para el diseño, sino para todo el desarrollo del proyecto de automatización concerniente.

C. **Validación**

El Autologic 1131, realiza una validación del diseño en dos etapas; la validación sintáctica y la validación de congruencia del gráfico. Las cuales se llevan a cabo, por medio de asistencias y procesos de análisis, que se realizan sobre el diseño en diferentes tiempos de ejecución, materializando el uso de las especificaciones de validación evidenciadas, evaluadas y sintetizadas en la investigación realizada.

- **Validación de sintaxis**

  En esta etapa básicamente se valida la correcta disposición de los elementos de diseño del SFC.

  Como primera medida, en la inserción y eliminación de elementos gráficos en el SFC, se realiza un acomodamiento automático del gráfico, mediante algoritmos que analizan los espacios y las coordenadas de los elementos. Además; de respetar las secuencias elementales de etapas – transiciones – etapas para garantizar el cumplimiento de las normas básicas de la sintaxis gráfica del SFC.

Por otra parte, tanto en la edición de las funciones combinacionales para las condiciones (en las transiciones y bifurcaciones), como cuando se asocian las variables a las acciones, se realizan validaciones basadas en expresiones regulares, con el fin de que se usen tanto las variables como las expresiones, con los formatos y las especificaciones sintácticas propuestas para el Autologic 1131 (Ver 4.3.1 Especificación de variables).

Por último se brinda la opción de realizar una inspección general del gráfico, la cual; revisa si el SFC actual aún presenta errores sintácticos, presentando los resultados de estos al usuario con su respectiva ubicación y posibles causas. Evitando que: existan etapas sin acciones asociadas, transiciones o bifurcaciones sin sus debidas funciones lógicas de condición y que si hay líneas de evolución creadas por divergencias, tengan una estructura cerrada con sus debidas convergencias.

- **Validación de congruencia**

En esta etapa básicamente se valida que lo diseñado no posea operaciones problemáticas en la ejecución del automatismo.

Como primera medida, cuando se insertan transiciones, divergencias en O, divergencias en Y o sus respectivas convergencias, el Autologic 1131 presenta listas dinámicas que informan y exponen exclusivamente las opciones válidas para que el usuario elija sobre ellas y evitar incongruencias como: el enlace de una etapa mediante una transición consigo misma, que se mezclen líneas de evolución de procesos independientes como se pueden dar cuando hay divergencias en Y, que entre dos etapas exista más de una transición, etc.

Además se realiza una inspección en el momento en el que se asocian acciones a las etapas con el fin de evitar incongruencias como: que se intente hacer accionamiento directo e inverso a la misma variable en la misma etapa y así mismo que se intente activar y desactivar la misma variable en la misma etapa.

Por último se brinda la opción de realizar una inspección general del gráfico, la cual revisa e informa al usuario de las incongruencias gráficas que se presenten en el diseño en el momento en el que esta se realiza. Evitando que existan etapas en el gráfico principal sin transiciones salientes, entre otras incongruencias.

- **Especificaciones de validación**

Es importante resaltar; que algunas inspecciones realizadas para los procesos de validación se implementan en rutinas recursivas, debido al anidamiento a nivel indeterminado y con infinidad de ramificaciones que puede ser resultado del uso de macroetapas.

Teniendo en cuenta ciertas pautas que se evidenciaron en la investigación que materializa esta herramienta, se sintetizan las especificaciones para hacer la validación del diseño, las cuales se enumeran a continuación:

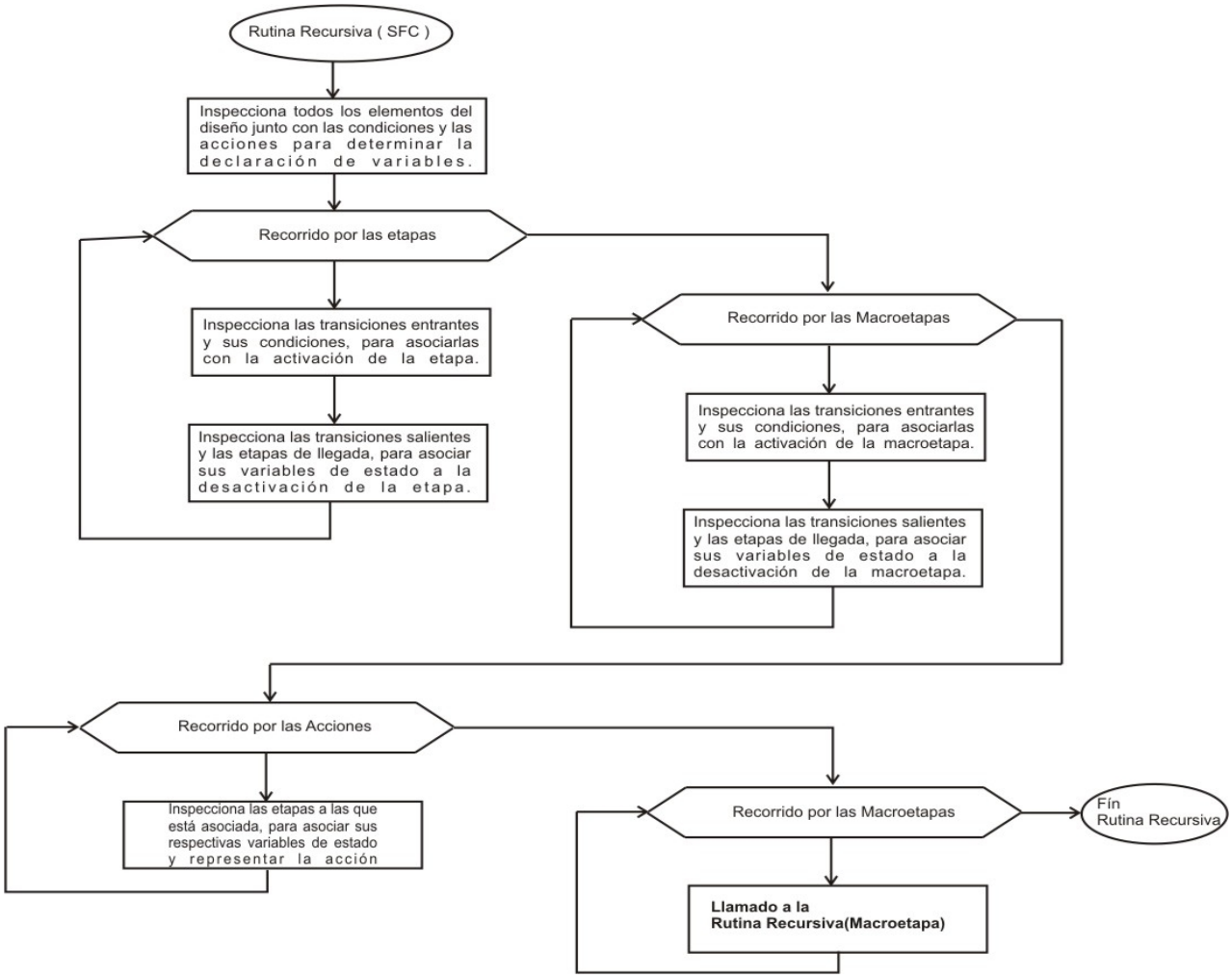
- En el gráfico principal la única etapa que puede que no tenga transición entrante es la etapa inicial (Distinguida por tener un doble recuadro en su representación gráfica y por que siempre tendrá el cero como identificador).
- En el gráfico principal absolutamente todas las etapas deben tener por lo menos una transición saliente de forma tal que el gráfico siempre deba permanecer en un proceso cíclico.
- En el diseño de las macroetapas la única etapa que puede que no tenga transición entrante es la etapa inicial (Distinguida por tener un doble recuadro en su representación gráfica y por que siempre tendrá el cero como identificador).
- En el diseño de las macroetapas es posible que una etapa no tenga transiciones salientes, pues esta será tomada como etapa de fin del subproceso.
- Todas las etapas deben tener, por lo menos una acción asociada.
- Todas las transiciones deben tener su función combinacional válida (incluyendo Divergencias y Convergencias).
- Todas las Líneas de evolución de las divergencias deben estar cerradas con sus debidas convergencias.
- Dos etapas no pueden estar en lazadas entre si con más de una transición.
- Un elemento (Etapa o Macroetapa) que pertenezca a una línea de evolución creada por una divergencia en Y, puede ser enlazada mediante transiciones únicamente con elementos que pertenezcan a la misma línea de la misma bifurcación.
- Las líneas de evolución de las divergencias solo pueden tener un único elemento (etapa o macroetapa) del cual parta la convergencia que la cierre.

- Es incongruente que en una misma etapa se asigne a la misma variable, mediante acciones ya sea de accionamiento o de activación valores de uno y cero al mismo tiempo.
- Es incongruente que una etapa se enlace a si misma mediante una transición, creando un bucle de una sola etapa.
- Tanto el diseño principal como los diseños de las subrutinas de las macroetapas son validados y se tendrán en cuenta todas las condiciones de congruencia para todos.

D. Generación

La generación de codificación en lenguajes IL, LD, ST y FBD a partir de un diseño en SFC, se presenta como una de las funcionalidades más importantes que brinda el Autologic 1131. Para esto; se llevo a cabo el diseño de interfaces mediante rutinas que toman el gráfico SFC satisfactoriamente validado y generan cada una de las codificaciones en los otros lenguajes, siguiendo una secuencia recursiva que bajo su estructura permite realizar un recorrido completo por todos y cada uno de los elementos pertenecientes al diseño y sus subrutinas (Ver figura 33).

Figura 33. Diagrama de flujo, algoritmo de generación de código



Son procesos que de manera general se pueden dividir en cuatro partes:

- Parte de declaración de variables: En esta se exploran todos los elementos del diseño, junto con las funciones de condición y las acciones asociadas a las etapas, para determinar la declaración de variables tanto de entrada como de salida, además de la declaración de las instancias de los bloques funcionales a utilizar en el POU.
- Parte secuencial: En esta se exploran tanto las etapas como las macroetapas (En ese orden), identificando tanto las opciones de activación como las de desactivación de las mismas. Se hace el llamado a los bloques funcionales biestables representativos de cada uno de estos elementos y

se asocia a sus variables de entrada las condiciones antes evidenciadas.

- Parte combinacional: En esta se exploran la lista de acciones y las etapas a las que pertenecen, para así; enlazar las variables de estado de las etapas directamente con los respectivos accionamientos, dependiendo del tipo de cada una de las acciones.
- Parte recursiva: En esta se exploran las macroetapas y desde cada una, se vuelve a llamar el proceso completo para el gráfico que les compete, definiendo el bloque funcional para el mismo. Y así realizar un recorrido en la totalidad del diseño del sistema automatizado.

4.5 TRATAMIENTO Y PRESENTACION DE LOS LENGUAJES

Los procesos de generación de código presentan una marcada estructuración en cuatro partes, las cuales se ven reflejadas en la presentación del los mismos, la cual se estructura básicamente en: parte de declaración, parte secuencial, parte combinacional y parte recursiva. Cabe aclarar que para cada bloque funcional definido en la parte recursiva, se realiza su respectiva declaración de variables, que se presentan como variables locales del proceso, pero; al llevarse el programa a un PLC y probar su funcionamiento, cabe la posibilidad de especificar la misma dirección de memoria física para una variable del programa principal y para cierta variable local de un bloque funcional (Dependiendo del PLC).
Básicamente la representación del código fuente generado en todos los lenguajes es muy similar, cada uno responde a la misma estructura, aunque con pequeñas modificaciones forzadas, por la diferente índole de cada uno de ellos.

Como se expuso en el capitulo anterior, para cada uno de los lenguajes la declaración de sus variables se realiza de forma textual, por lo cual; el Autologic 1131 implementa un formulario donde se encuentra contenida la declaración de variables del programa principal, que es común para todos los lenguajes, logrando con esto; un mejor ordenamiento y mayor claridad en la presentación de lo diseñado (Ver figura 34).

El Autologic 1131 para los códigos IL, ST, FBD y LD conserva los operadores, operandos y reglas sintácticas expuestas en el estándar IEC 1131-3 (Ver capitulo 3). Y los presenta sobre formularios independientes que se cargan sobre el formulario principal, los cuales permiten que se

exploren cada uno en su totalidad de manera ordenada y congruente (Ver Figuras 35-38).

Figura 34. Parte declaración Autologic 1131

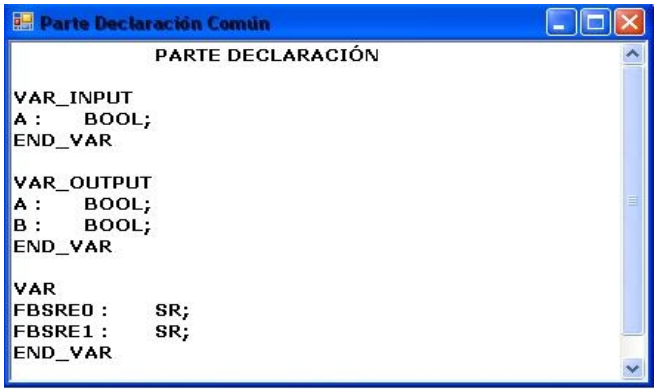


Figura 35. Parte codificación IL Autologic 1131

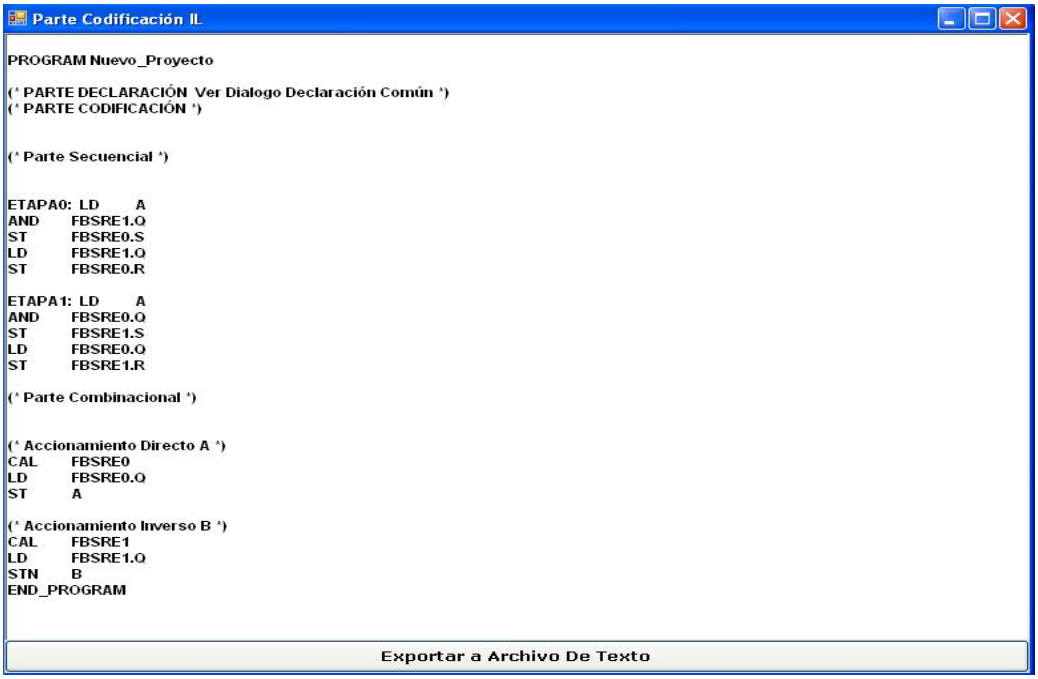


Figura 36. Parte codificación FBD Autologic 1131

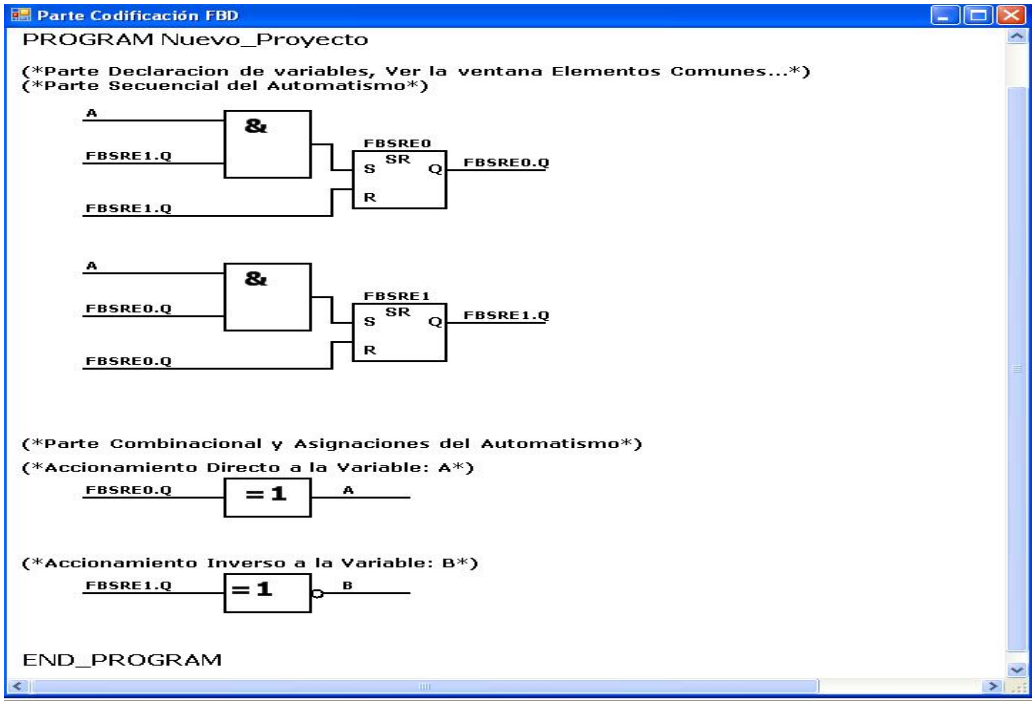


Figura 37. Parte codificación LD Autologic 1131

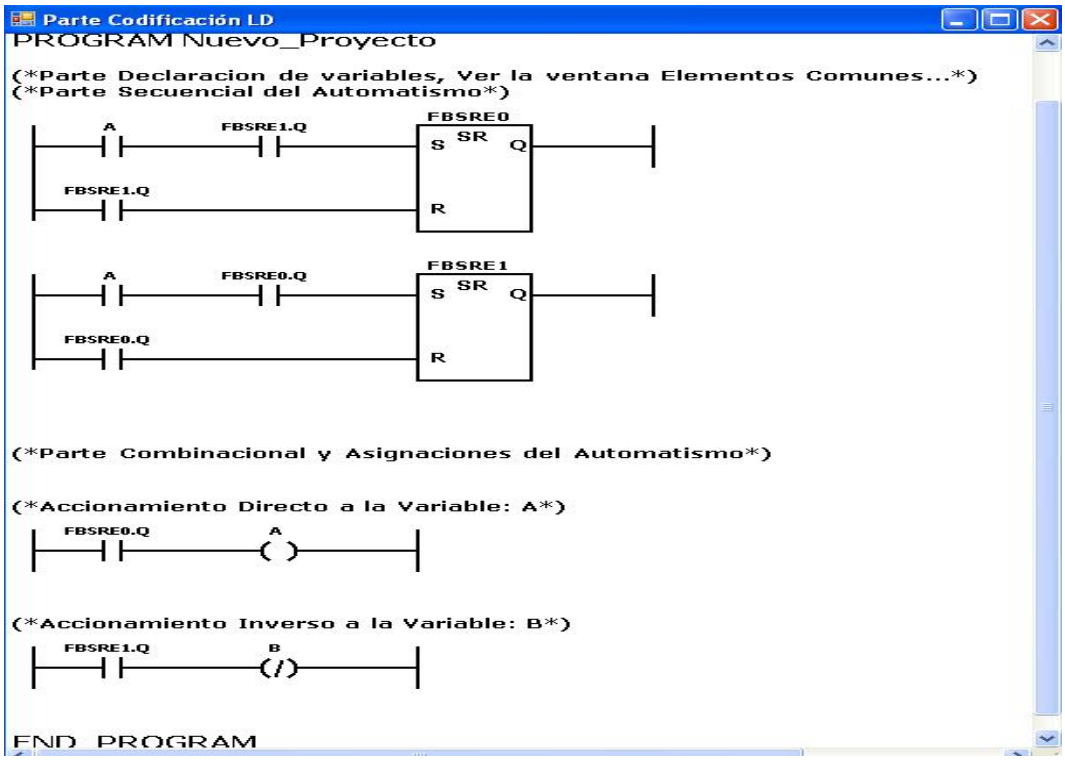
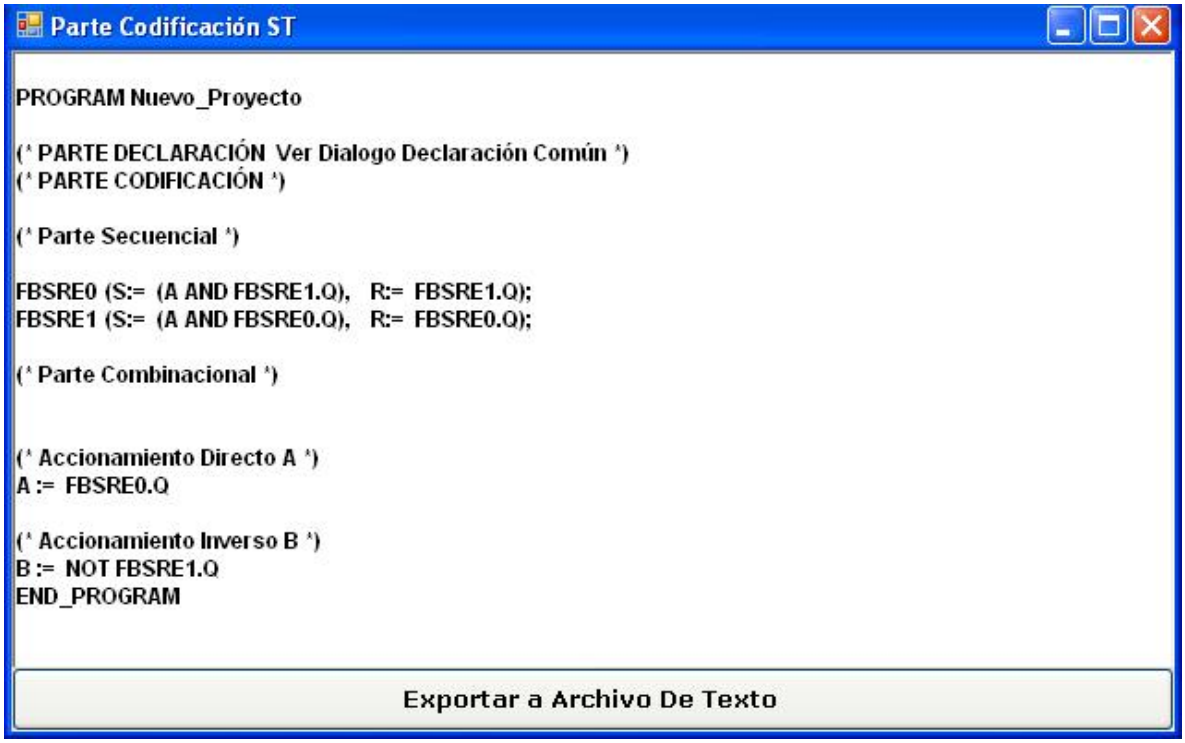


Figura 38. Parte codificación ST Autologic 1131



4.6 Portabilidad.

Como se menciono anteriormente, el Autologic 1131 siendo fiel a su metodología de trabajo; brinda la posibilidad de hacer portable por el usuario todos los contenidos consignados en él, es decir; permite portar tanto el diseño en SFC, como los códigos fuentes generados por la herramienta en los lenguajes IL, ST, LD y FBD.

Todo esto a gracias a que posibilita la generación de archivos contenedores para cada uno de los elementos de su contenido, por separado. Estos archivos contenedores son generados en XML[9], gracias a que la estructura de organización que presentan estos, es adaptable a las necesidades que demanda la estructura de datos maestra del Autologic 1131. Además el uso de XML para estos archivos, aporta a los mismos todas las bondades de orden, estructura, presentación y transporte que esto implica. Entre ellas algunas como:

- Una estructura ordenada de presentación por etiquetas.

[9] Lenguaje de marcas extendido, que permite definir la gramática de lenguajes específicos y expresar información estructurada de la manera más abstracta y reutilizable posible.

- Facilidad en su transporte, puesto que los archivos XML son equiparables a archivos planos, por esto; los filtros Web casi nunca los rechazan.
- posibilita su visualización ya sea en un navegador Web o en un editor de texto convencional.

Además como funcionalidades complementarias, el Autologic 1131 brinda la posibilidad al usuario de:

- Llevar tanto el gráfico de diseño, como cada uno de los códigos generados a partir de este; al papel físico, por medio de la funcionalidad de impresión, permitiendo la preparación de la hoja de impresión y mostrando en pantalla una vista previa de la misma.
- Exportar a archivos planos de texto (con extensión ".TXT"), los archivos de código correspondientes a los lenguajes textuales presentados en la norma IEC 1131-3.

Todo esto con el fin de brindar la posibilidad al usuario de transportar y reutilizar toda la información de trabajo, ya sea sobre el mismo Autologic 1131 o sobre algún software específico de algún fabricante; para dar consolidación al montaje previamente diseñado.

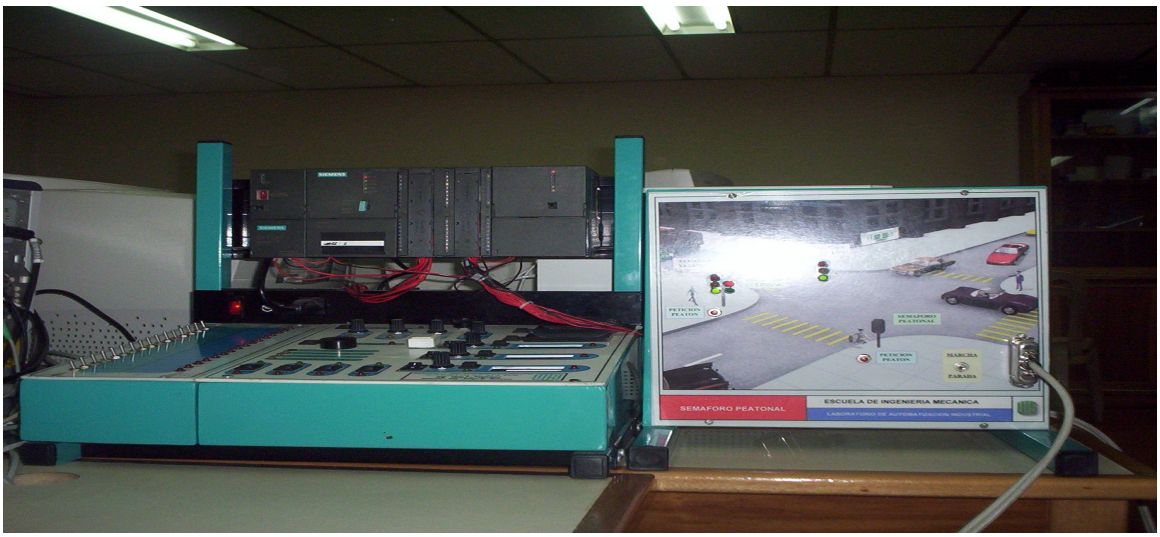
4.7 VERIFICACIÓN DEL CODIGO GENERADO

Con lo expuesto, queda por mostrar, la verificación del código generado; esto en relación a un ejemplo puntual, que permita visualizar que esta generación posee la correcta sintaxis y congruencia, a tal punto que es posible, que un PLC ejecute su codificación y presente los resultados diseñados mediante SFC.

Se ha escogido para la presentación de la verificación, el PLC Siemens S7-300[10] (ver Figura 39), aunque este PLC no esta bajo el estándar; es posible, realizar una verificación debido a los lenguajes gráficos que posee, KOP y FUP, que son lenguajes que el estándar a incluido y son los denominados en la norma LD y FBD respectivamente, claro esta, deberá realizarse determinados ajustes para su correcto funcionamiento, como se vera mas adelante. De igual forma, se ha escogido la situación de un semáforo peatonal para el planteamiento del diseño y su posterior funcionamiento.

[10] Por los implementos tecnológicos que posee la Universidad, solo se tiene acceso a este equipo, ya que no se cuenta con ningún otro PLC mas sofisticado o que se encuentre bajo el estándar, como se advirtió en anteriores escritos (ver plan de proyecto).

Figura 39. PLC Siemens S7-300 y mímico semáforo peatonal



Entrando en materia, el autómata tiene como fin controlar, los semáforos para el transito de vehículos y el semáforo para transito de peatones; para los semáforos de vehículos se tienen tres luces, rojo, amarillo y verde; para el semáforo peatonal se tiene una luz roja y otra verde, además un pulsador para que los peatones pidan paso libre (luz verde). De igual manera se posee un pulsador de dos posiciones (marcha/parada) que indicará la activación o no de los semáforos.

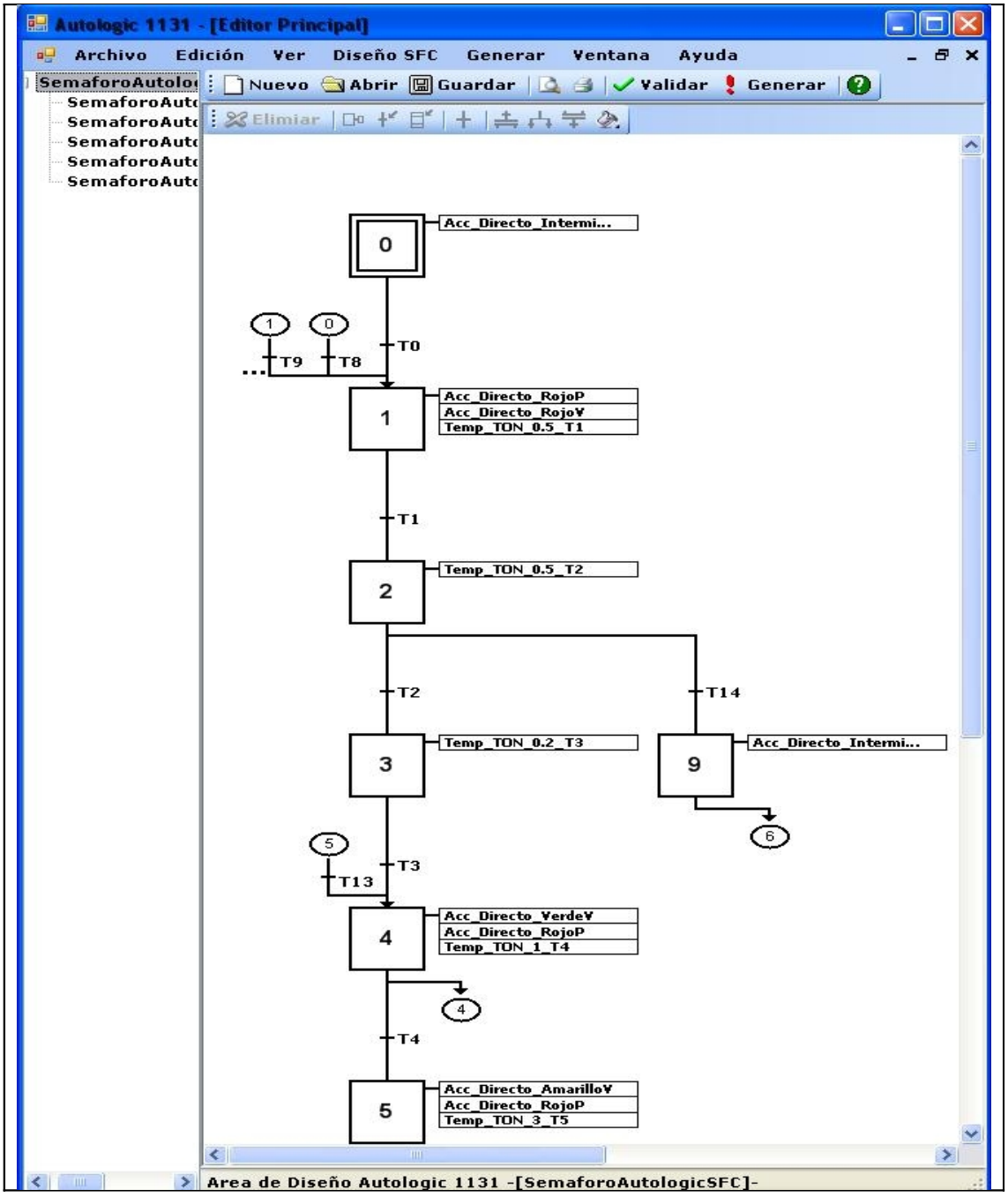
Los requisitos de funcionamiento para los semáforos son:

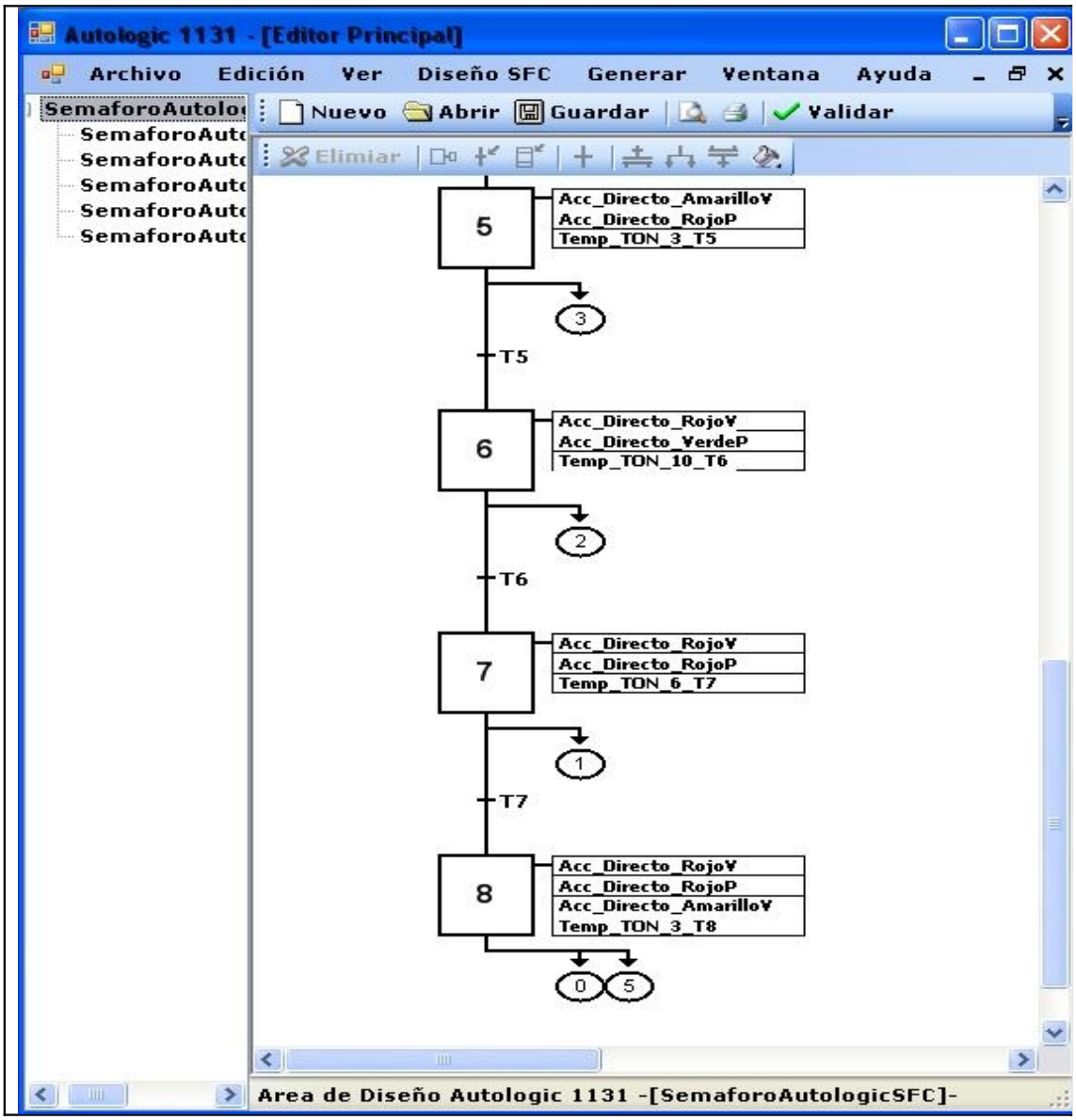
- Cuando el pulsador de dos posiciones, se encuentre en parada; la luz roja para vehículos y peatones se debe mostrar intermitente a una escalara de quinientos milisegundos.
- Cuando el pulsador de dos posiciones, se encuentre en marcha; la intermitencia se detiene, se debe producir una espera de doscientos milisegundos y quedar en verde para vehículos y rojo para peatones, con esto se empieza el ciclo que contiene:
    1. El amarillo para vehículos debe durar tres segundos
    2. el rojo para vehículos debe durar 6 segundos y deberá arrancar simultáneamente con el verde para peatones
    3. el verde para peatones deberá durar diez segundos
    4. tan pronto como finalice el verde para peatones, el semáforo para peatones deberá conmutar a rojo
    5. la fase rojo/amarillo para vehículos deberá durar tres segundos
    6. la siguiente solicitud para verde de peatones deberá esperar un segundo.

Estos son los requerimientos para el autómata, para el funcionamiento del semáforo.

El Diseño generado en Autologic 1131 para la solución de mencionado enunciado es:

Figura 40. Diseño SFC para el semáforo, realizado en Autologic 1131

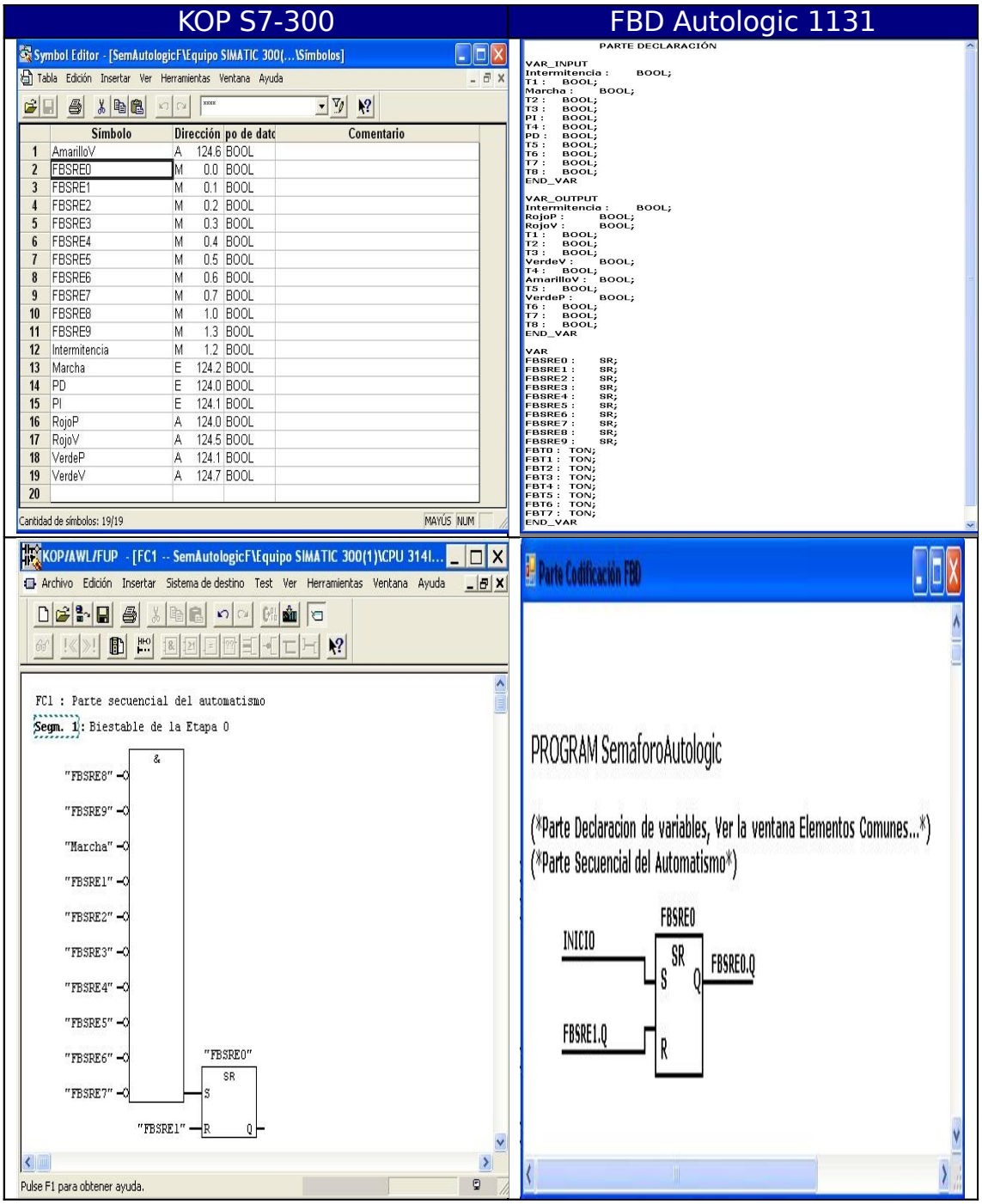


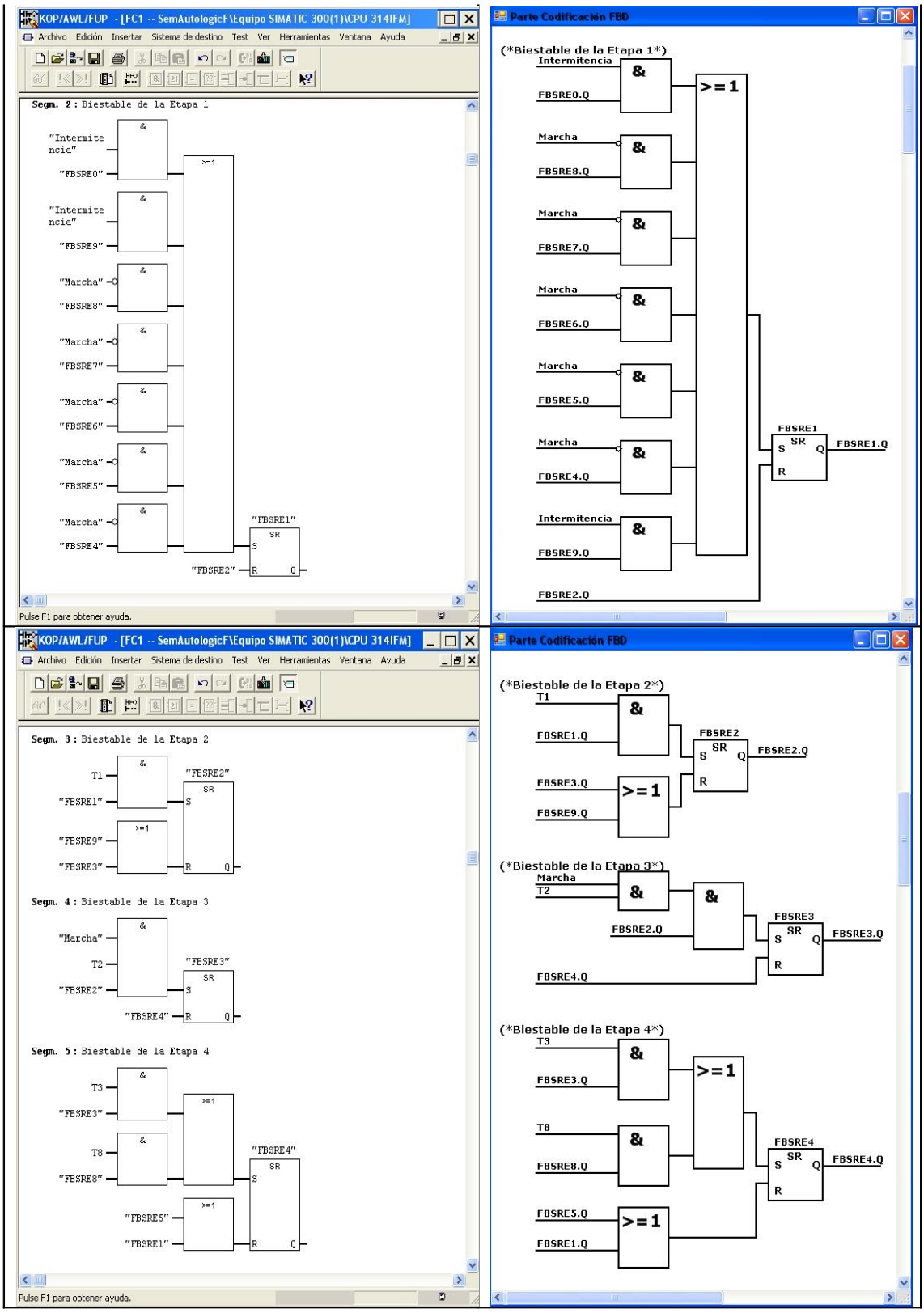


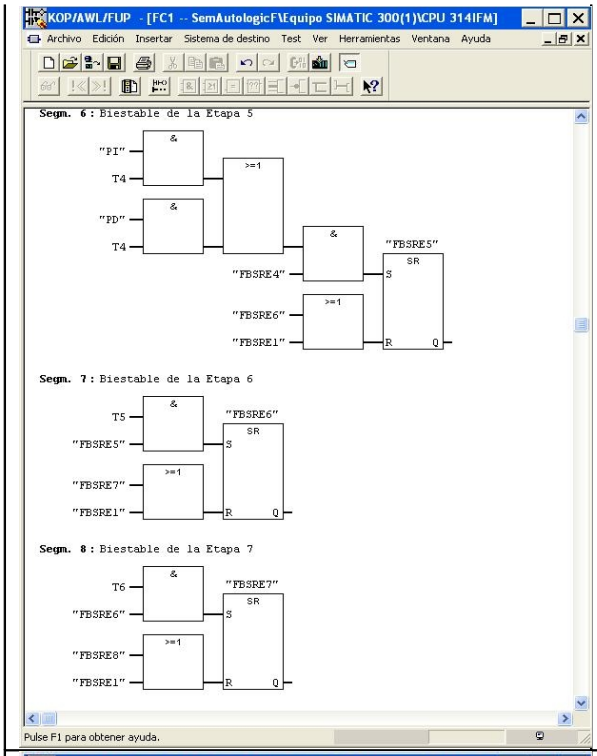
El anterior diseño (ver Figura 40), cuenta con diez etapas y dieciséis transiciones, comenzando en la etapa cero con el accionamiento directo de la variable intermitencia, que dará paso al encendido del rojo para vehículos y peatones, posteriormente las etapas dos y nueve se encargan de mantener el ciclo de intermitencia, cuando se active el pulsador a marcha, la transición T2 cumplirá su condición y la etapa tres, contara los dos segundos, para colocar el estado inicial del ciclo (verde vehículos y rojo peatones), las etapas cinco a ocho son las encargadas de realizar el funcionamiento, al realizar las peticiones los peatones. Este es un diseño que cumple con las condiciones de funcionamiento y genera la codificación en los lenguajes del estándar.

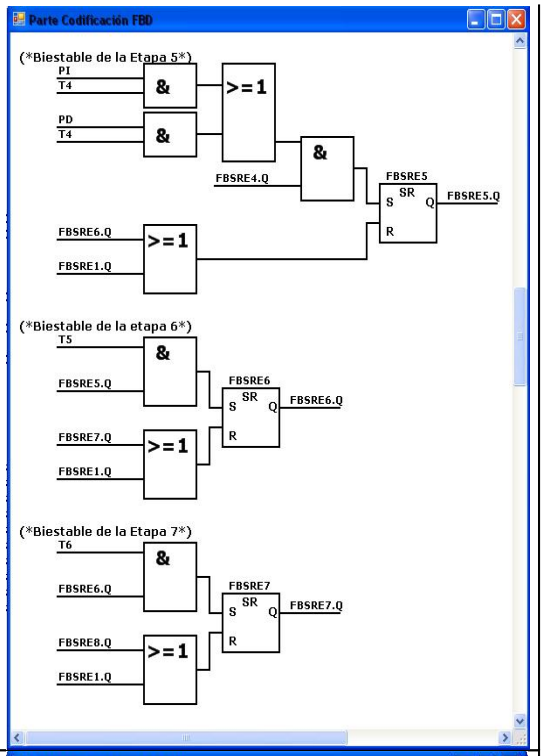
Con lo anteriormente expuesto, solo queda; transcribir la codificación de los lenguajes gráficos generados por Autologic 1131 al software controlador del PLC S7-300, lo cual se puede apreciar en las figuras siguientes.

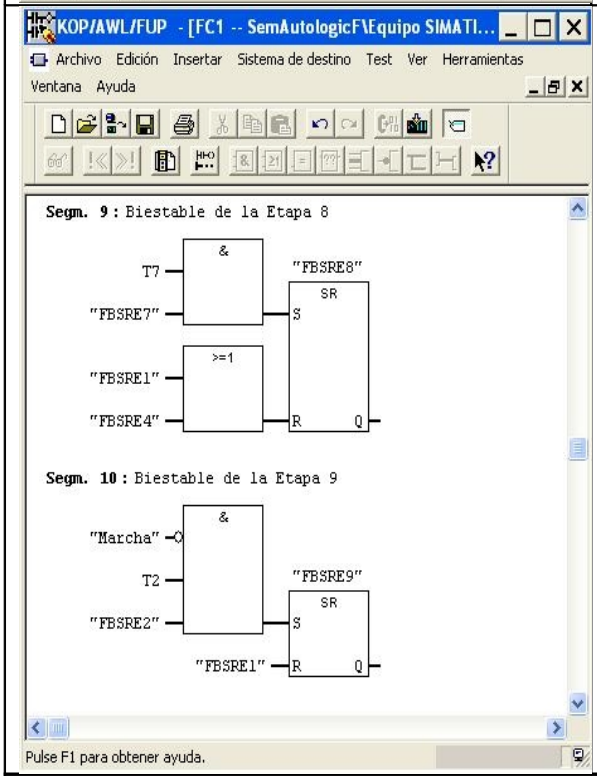
Tabla 14. Codificación en FUP del S7-300 y FBD del Autologic 1131.

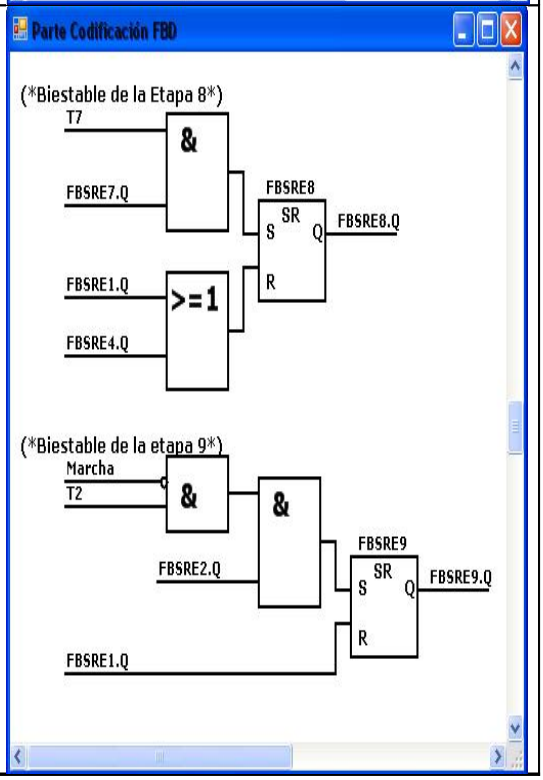


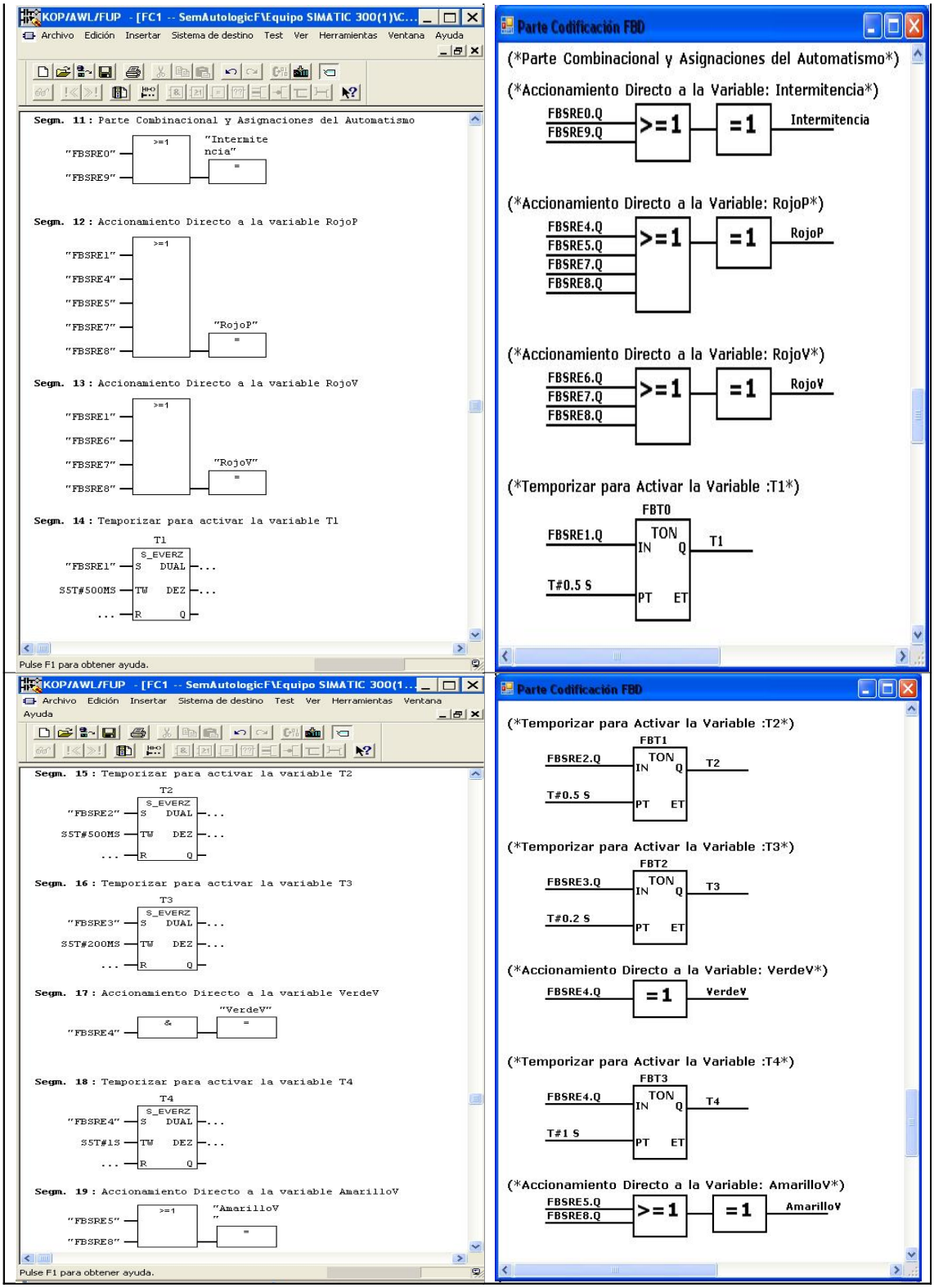

KOP/AWL/FUP - [FC1 -- SemAutologicF\Equipo SIMATIC 300(1)\CPU 314IFM]
Archivo Edición Insertar Sistema de destino Test Ver Herramientas Ventana Ayuda
Segm. 2 : Biestable de la Etapa 1
"Intermitencia"
"FBSRE0"
"Intermitencia"
"FBSRE9"
"Marcha"
"FBSRE8"
"Marcha"
"FBSRE7"
"Marcha"
"FBSRE6"
"Marcha"
"FBSRE5"
"Marcha"
"FBSRE4"
>=1
"FBSRE1"
SR
S
R
Q
"FBSRE2"
Pulse F1 para obtener ayuda.
Parte Codificación FBD
(*Biestable de la Etapa 1*)
Intermitencia
FBSRE0.Q
Marcha
FBSRE8.Q
Marcha
FBSRE7.Q
Marcha
FBSRE6.Q
Marcha
FBSRE5.Q
Marcha
FBSRE4.Q
Intermitencia
FBSRE9.Q
FBSRE2.Q
FBSRE1
FBSRE1.Q
KOP/AWL/FUP - [FC1 -- SemAutologicF\Equipo SIMATIC 300(1)\CPU 314IFM]
Archivo Edición Insertar Sistema de destino Test Ver Herramientas Ventana Ayuda
Segm. 3 : Biestable de la Etapa 2
T1
"FBSRE1"
"FBSRE2"
"FBSRE9"
"FBSRE3"
Segm. 4 : Biestable de la Etapa 3
"Marcha"
T2
"FBSRE2"
"FBSRE3"
"FBSRE4"
Segm. 5 : Biestable de la Etapa 4
T3
"FBSRE3"
T8
"FBSRE8"
"FBSRE4"
"FBSRE5"
"FBSRE1"
Pulse F1 para obtener ayuda.
Parte Codificación FBD
(*Biestable de la Etapa 2*)
T1
FBSRE1.Q
FBSRE3.Q
FBSRE9.Q
FBSRE2
FBSRE2.Q
(*Biestable de la Etapa 3*)
Marcha
T2
FBSRE2.Q
FBSRE4.Q
FBSRE3
FBSRE3.Q
(*Biestable de la Etapa 4*)
T3
FBSRE3.Q
T8
FBSRE8.Q
FBSRE5.Q
FBSRE1.Q
FBSRE4
FBSRE4.Q

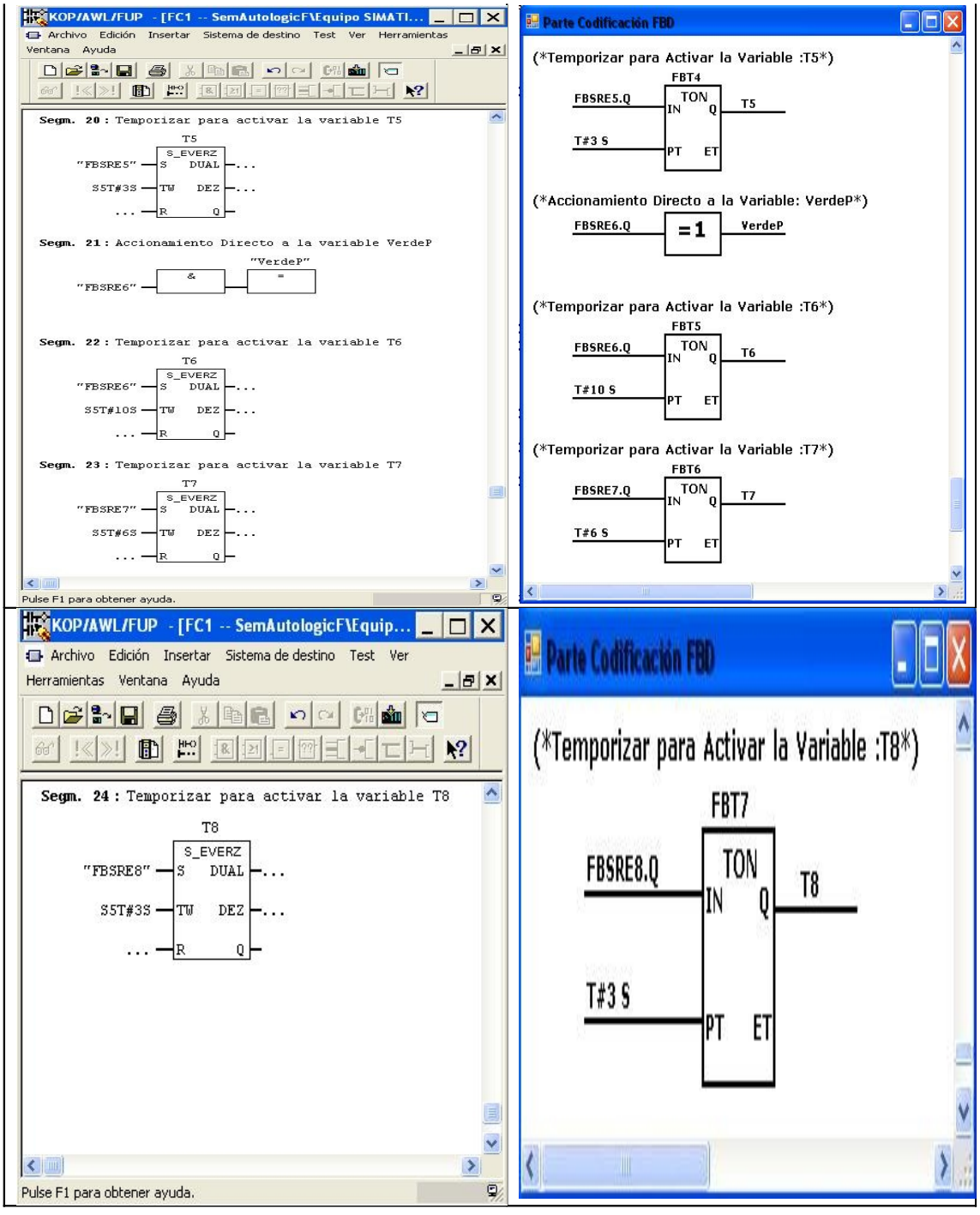

KOP/AWL/FUP - [FC1 -- SemAutologicF\Equipo SIMATIC 300(1)\CPU 314IFM]
Archivo Edición Insertar Sistema de destino Test Ver Herramientas Ventana Ayuda
Segm. 6 : Biestable de la Etapa 5
"PI"
T4
"PD"
T4
&
>=1
"FBSRE4"
"FBSRE6"
"FBSRE1"
"FBSRE5"
SR
S
R
Q
Segm. 7 : Biestable de la Etapa 6
T5
"FBSRE5"
"FBSRE7"
"FBSRE1"
"FBSRE6"
Segm. 8 : Biestable de la Etapa 7
T6
"FBSRE6"
"FBSRE8"
"FBSRE1"
"FBSRE7"
Pulse F1 para obtener ayuda.


Parte Codificación FBD
(*Biestable de la Etapa 5*)
PI
T4
PD
T4
&
>=1
FBSRE4.Q
FBSRE5
SR
S
R
Q
FBSRE5.Q
FBSRE6.Q
FBSRE1.Q
(*Biestable de la etapa 6*)
T5
FBSRE5.Q
FBSRE6
FBSRE6.Q
FBSRE7.Q
FBSRE1.Q
(*Biestable de la Etapa 7*)
T6
FBSRE6.Q
FBSRE7
FBSRE7.Q
FBSRE8.Q
FBSRE1.Q


KOP/AWL/FUP - [FC1 -- SemAutologicF\Equipo SIMATI...
Archivo Edición Insertar Sistema de destino Test Ver Herramientas
Ventana Ayuda
Segm. 9 : Biestable de la Etapa 8
T7
"FBSRE7"
"FBSRE1"
"FBSRE4"
"FBSRE8"
SR
S
R
Q
Segm. 10 : Biestable de la Etapa 9
"Marcha"
T2
"FBSRE2"
"FBSRE1"
"FBSRE9"
Pulse F1 para obtener ayuda.


Parte Codificación FBD
(*Biestable de la Etapa 8*)
T7
FBSRE7.Q
FBSRE8
SR
S
R
Q
FBSRE8.Q
FBSRE1.Q
FBSRE4.Q
>=1
(*Biestable de la etapa 9*)
Marcha
T2
FBSRE2.Q
FBSRE9
FBSRE9.Q
FBSRE1.Q


KOP/AWL/FUP - [FC1 -- SemAutologicF\Equipo SIMATIC 300(1)\C...
Archivo Edición Insertar Sistema de destino Test Ver Herramientas Ventana Ayuda
Segm. 11 : Parte Combinacional y Asignaciones del Automatismo
"FBSRE0"
"FBSRE9"
>=1
"Intermitencia"
Segm. 12 : Accionamiento Directo a la variable RojoP
"FBSRE1"
"FBSRE4"
"FBSRE5"
"FBSRE7"
"FBSRE8"
"RojoP"
Segm. 13 : Accionamiento Directo a la variable RojoV
"FBSRE1"
"FBSRE6"
"FBSRE7"
"FBSRE8"
"RojoV"
Segm. 14 : Temporizar para activar la variable T1
T1
S_EVERZ
"FBSRE1"
S5T#500MS
S DUAL
TW DEZ
R Q
Pulse F1 para obtener ayuda.
Parte Codificación FBD
(*Parte Combinacional y Asignaciones del Automatismo*)
(*Accionamiento Directo a la Variable: Intermitencia*)
FBSRE0.Q
FBSRE9.Q
>=1
=1
Intermitencia
(*Accionamiento Directo a la Variable: RojoP*)
FBSRE4.Q
FBSRE5.Q
FBSRE7.Q
FBSRE8.Q
RojoP
(*Accionamiento Directo a la Variable: RojoV*)
FBSRE6.Q
FBSRE7.Q
FBSRE8.Q
RojoV
(*Temporizar para Activar la Variable :T1*)
FBT0
FBSRE1.Q
TON
IN Q
T1
T#0.5 S
PT ET
KOP/AWL/FUP - [FC1 -- SemAutologicF\Equipo SIMATIC 300(1..
Archivo Edición Insertar Sistema de destino Test Ver Herramientas Ventana Ayuda
Segm. 15 : Temporizar para activar la variable T2
T2
S_EVERZ
"FBSRE2"
S5T#500MS
Segm. 16 : Temporizar para activar la variable T3
T3
"FBSRE3"
S5T#200MS
Segm. 17 : Accionamiento Directo a la variable VerdeV
"VerdeV"
&
"FBSRE4"
Segm. 18 : Temporizar para activar la variable T4
T4
"FBSRE4"
S5T#1S
Segm. 19 : Accionamiento Directo a la variable AmarilloV
"AmarilloV"
"FBSRE5"
"FBSRE8"
Pulse F1 para obtener ayuda.
Parte Codificación FBD
(*Temporizar para Activar la Variable :T2*)
FBT1
FBSRE2.Q
T#0.5 S
T2
(*Temporizar para Activar la Variable :T3*)
FBT2
FBSRE3.Q
T#0.2 S
T3
(*Accionamiento Directo a la Variable: VerdeV*)
FBSRE4.Q
VerdeV
(*Temporizar para Activar la Variable :T4*)
FBT3
FBSRE4.Q
T#1 S
T4
(*Accionamiento Directo a la Variable: AmarilloV*)
FBSRE5.Q
FBSRE8.Q
AmarilloV



Finalmente se obtienen los resultados pretendidos por el enunciado y diseñados mediante SFC en Autologic 1131, como lo muestran las figuras siguientes, donde se aprecian los diferentes estados diseñados.

Figura 41. Pulsador en parada, semáforos intermitencia (apagados)

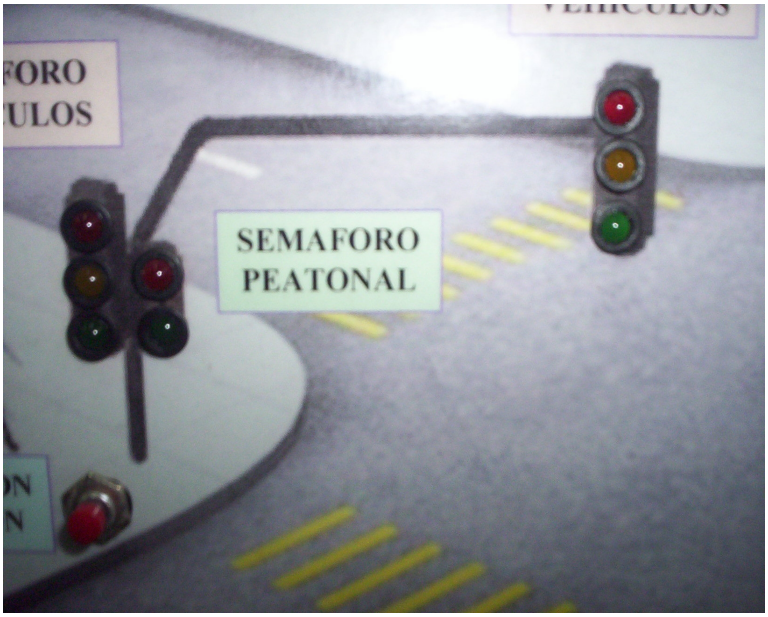


Figura 42. Pulsador en parada, semáforos intermitencia (encendidos)

Figura 43. Pulsador marcha, condición inicial (verde vehículos, rojo peatones)

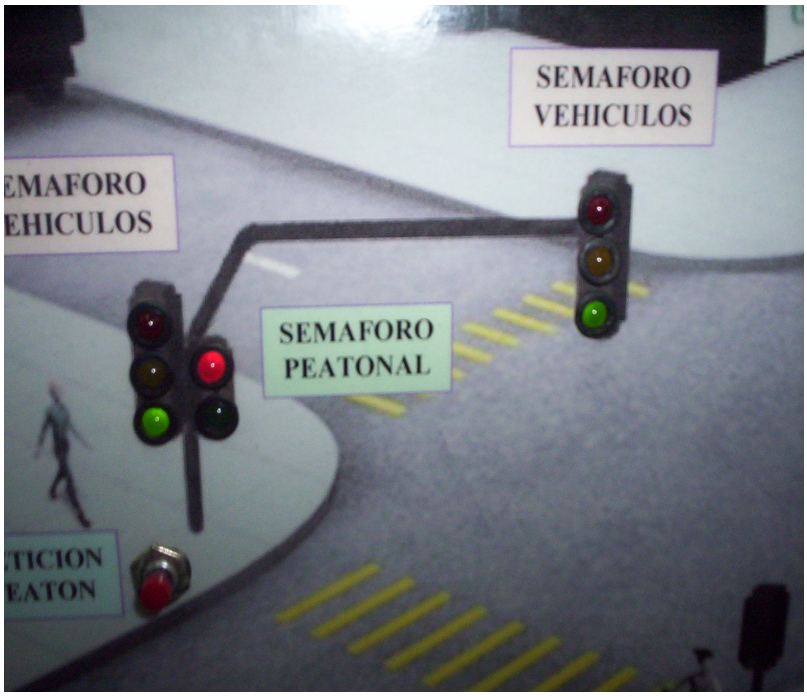
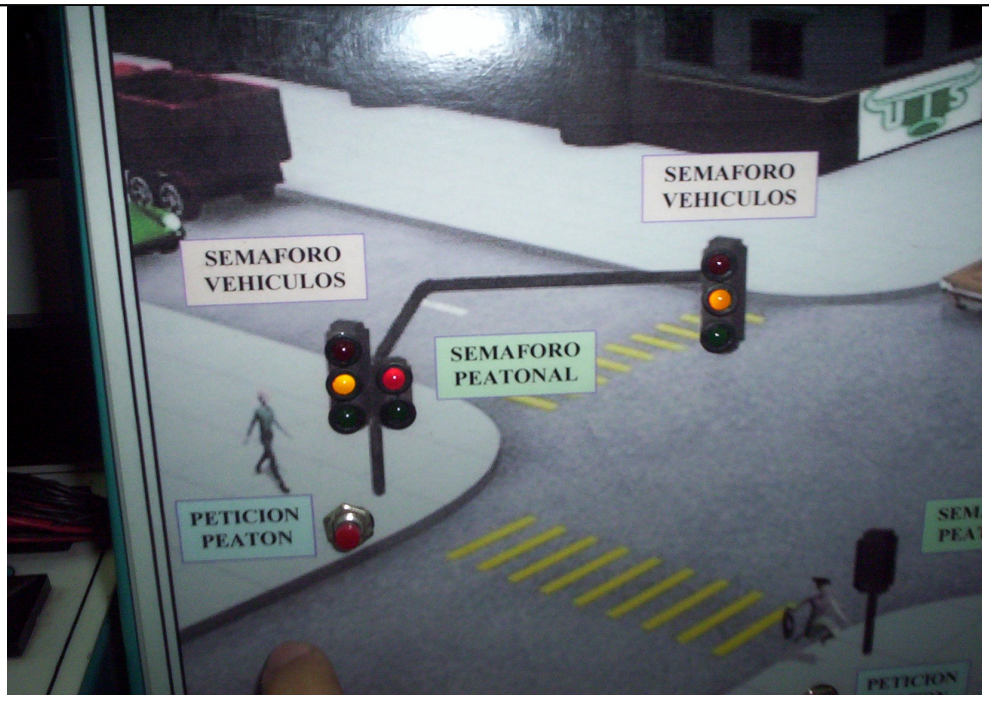
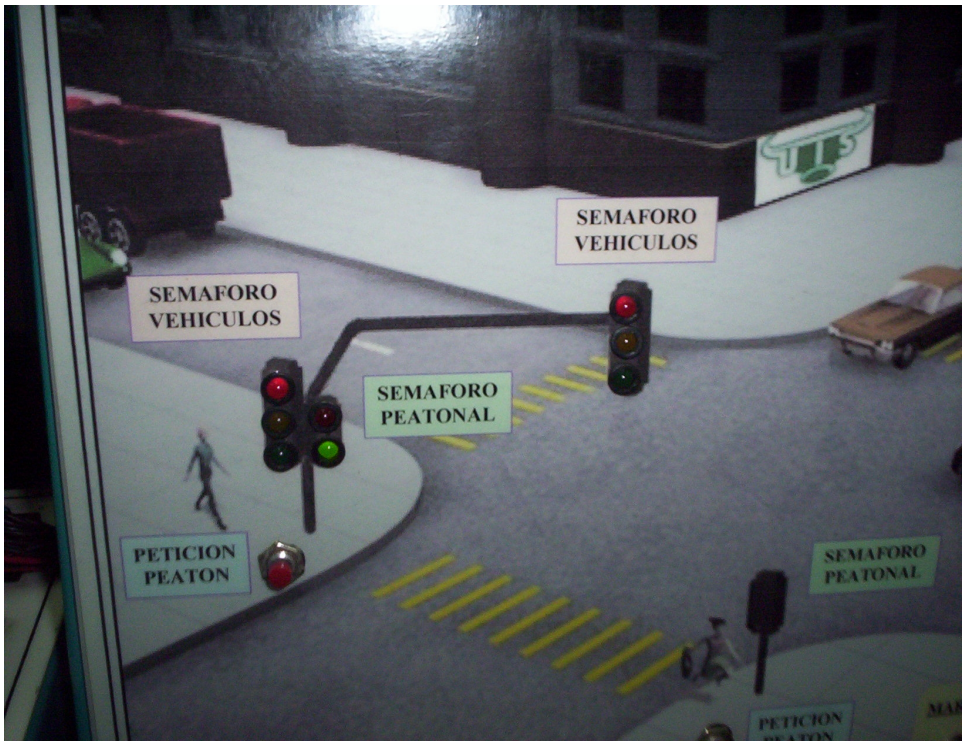
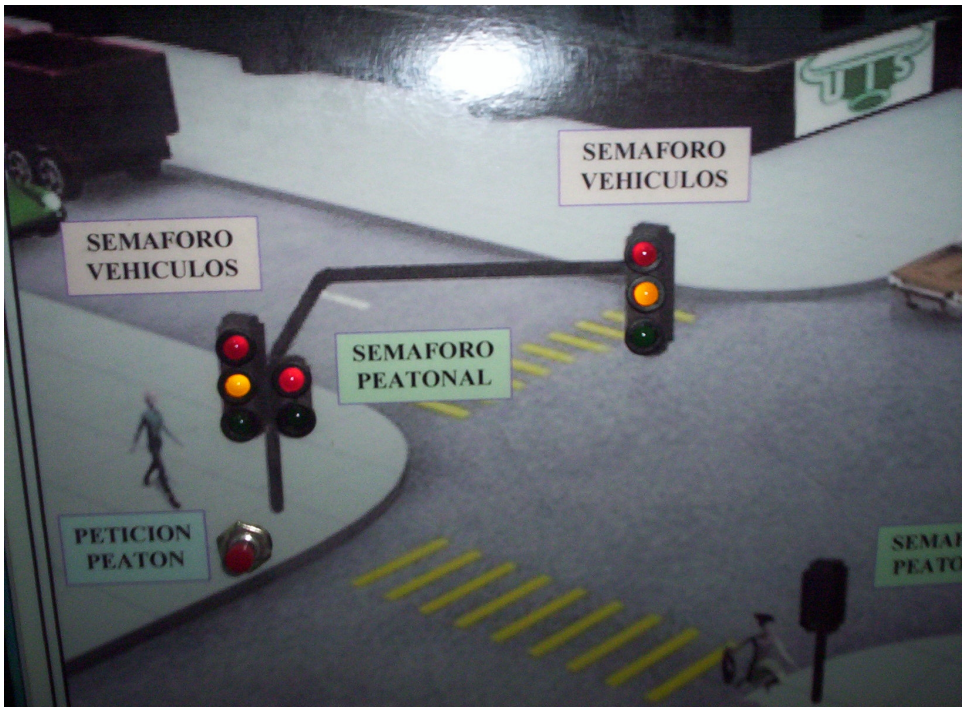


Figura 44. Pulsador en marcha, Ciclo petición peatón




SEMAFORO
VEHICULOS
SEMAFORO
VEHICULOS
SEMAFORO
PEATONAL
PETICION
PEATON
SEMAFORO
PEATONAL


SEMAFORO
VEHICULOS
SEMAFORO
VEHICULOS
SEMAFORO
PEATONAL
PETICION
PEATON
PETICION
PEATON


SEMAFORO
VEHICULOS
SEMAFORO
VEHICULOS
SEMAFORO
PEATONAL
PETICION
PEATON

CONCLUSIONES

Con el análisis realizado en la presente investigación, se puede afirmar; que el estándar IEC 1131 posee los contenidos necesarios y suficientes, para el desarrollo sólido de una metodología de trabajo y para la solución de comunicación y evolución de los proyectos de automatización industrial, consiguiendo con esto unos impactos técnicos y económicos que redundan en mayor agilidad, menor costo, menor complejidad en el mantenimiento, entendimiento y reutilización de los desarrollos.

Dentro del campo de la automatización industrial, los procesos lógicos secuenciales representan un espacio de aplicación amplio y de gran desarrollo, por lo tanto la aplicación del estándar para este sector es un avance en la construcción de una forma de trabajo con independencia de fabricantes y con la focalización puesta en el diseño.

La capacidad evidenciada en el lenguaje SFC para su denominación como un lenguaje de diseño, y la realización de la conexión de los lenguajes IL, ST, FBD y LD con el diseño, para la generación coherente de codificación en ellos.

La estructuración de una herramienta software basada en la norma, con la capacidad de asistir el diseño mediante el lenguaje SFC, generando para esto un despliegue dinámico, en el área de diseño y validación, otorgando fiabilidad de lo diseñado, que hace conductista; el desarrollo de un diseño para un A.L.S, de igual forma; la capacidad de la herramienta software para la generación de codificación ejecutable en PLC que reconozcan el estándar, de los cuatro lenguajes que expone la norma IL, ST, FBD y LD consiguiendo una disminución en el tiempo de desarrollo e implementación del proyecto del A.L.S con referencia a la forma de trabajo manual.

La realización de la investigación y la generación a partir de ella de una herramienta software, sustenta el carácter interdisciplinario del ingeniero de sistemas, desde la incidencia del trabajo en el campo de la automatización industrial, en lo referente de los A.L.S, mostrando así que la elaboración de software debe ser fruto de una labor mas profunda de investigación y relación con temáticas que si bien, no son nuestras, no son ajenas.

TRABAJOS FUTUROS

Para la investigación seria un paso siguiente, avanzar en la caracterización del SFC con un lenguaje de diseño universal, incluyendo para él, el análisis de los demás bloques funcionales que quedan por fuera de esta investigación por su carácter lógico secuencial

Generar para la herramienta software Autologic 1131 un módulo, con la capacidad de conversión de los lenguajes IL, ST, FBD y LD generados, en archivos de ejecución en PLC’s con reconocimiento del estándar, en sus formatos específicos.

Ampliar la herramienta software Autologic 1131 para contemplar automatismo más allá de los lógicos secuenciales, de forma que se introduciría la investigación que se enuncio en el primer párrafo.

BIBLIOGRAFÍA

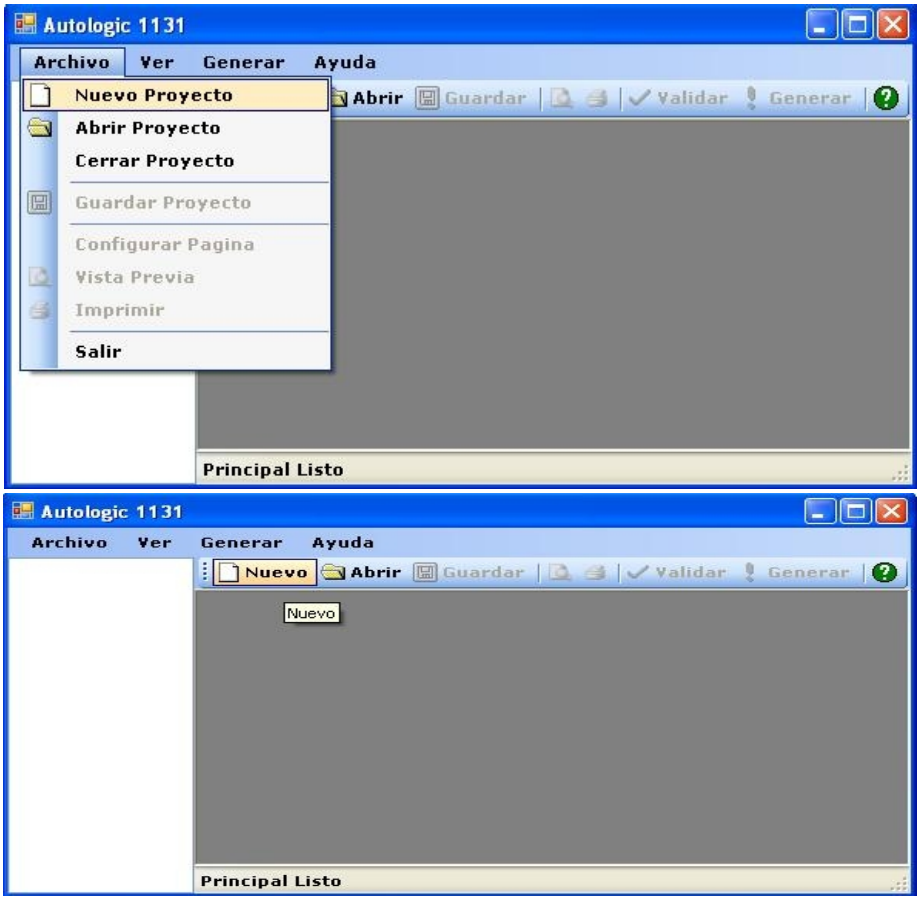
- BALCELLS, Joseph y ROMERAL, José Luís. Autómatas programables. México. Alfaomega. 1998.
- CALDERON, Jesús, ABRIL, Tilsmary y TELLO, Jesús. Iniciación en CONCEPT V2.5. Venezuela. Universidad de los Andes. 2003.
- COMISIÓN INTERNACIONAL DE ELECTROTÉCNICA. IEC 61131, Suiza. IEC 1995.
- DE CASTRO LOZANO, Carlos y ROMERO MORALES, Cristóbal. Introducción a SCADA. 2001.
- HURTADO DE BARRERA, Jacqueline. Metodología de la Investigación Holística. Caracas, Venezuela. Fundación Sypal. 1998.
- JOHN, Kart-Heinz y TIEGELKAMP, Michael. IEC 61131-3 Programming Industrial Automation Systems. New York. Springer. 2000.
- MATEUS MARTIN, Felipe. Autómatas Programables: Introducción al Estándar IEC 61131. España. Universidad de Oviedo Grupo de investigación Genia. 2000.
- M. MORRIS, Mano. Diseño Digital. Los Ángeles. Universidad del estado de California. Pentrice Hall. 1987
- PAPPAS, Chris H y MURRAY III, William H. Visual C++.Net, Mc Graw Hill, España. 2002.
- PLCOPEN. leguajes de Programación. Departamento. Países Bajos. PLCopen de sistemas electrónicos y de control. 2004.
- Siemens. Manual de usuario PLC S7-300. España. Siemens. 2000.
- UNIVERSIDAD DE ORIENTE. Tecnologías Emergentes En Automatización. Venezuela. Facultad de Ingeniería Eléctrica. 2002.

ANEXO A

1. Creando un Proyecto Nuevo

Arranque el **Autologic 1131,** con solo hacer un doble click sobre el archivo ejecutable o sobre su acceso directo en su defecto, e iniciará la pantalla principal del **Autologic 1131.**

Para crear un nuevo proyecto: haga click sobre el menú principal en **Archivo>Nuevo Proyecto** o sobre la barra de Herramientas principal en el botón ***Nuevo*.**

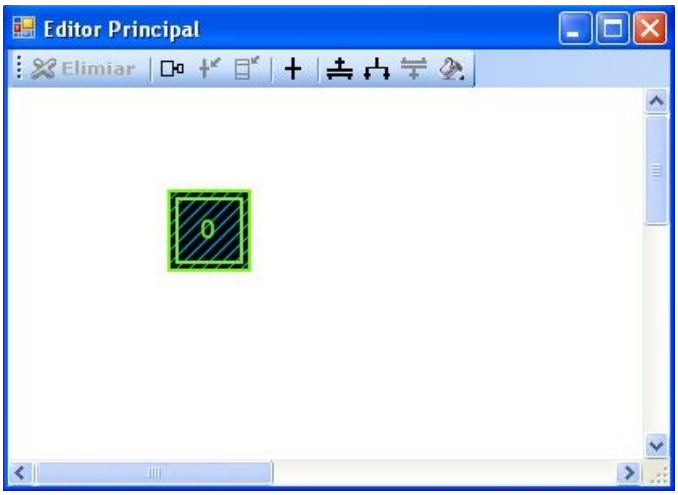


Así se abrirá el diálogo asistente para la creación de un nuevo proyecto.

En el cuadro de texto ***nombre*** se especifica el nombre que llevará el proyecto.

En el cuadro de texto ***ubicación*** se especifica la ruta de acceso en la que se alojará el proyecto, para una elección más sencilla de una ubicación; si se hace click sobre el botón ***Examinar,*** se mostrará un cuadro de diálogo que permite la navegación sobre el sistema de archivos. Cuya elección final será la ubicación que tendrá el proyecto, afirmando sobre la casilla de ***Ubicación*** la ruta elegida.

Habiendo definido el nombre y la ruta de ubicación del proyecto, se hace click sobre el botón ***Aceptar*** para que se cree y se muestre el formulario de Editor Principal; que contiene el área de diseño en SFC, la cual por defecto dibuja la etapa inicial.



Teniendo así; el editor listo para comenzar con el trabajo de diseño.

2. Diseñando un SFC

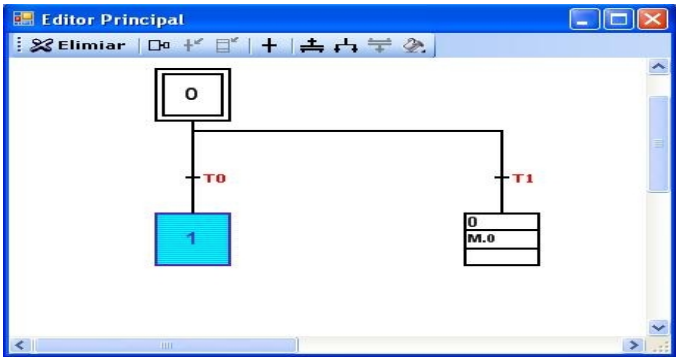
2.1 Seleccionando un elemento: Para seleccionar cualquier elemento en el área de diseño basta con hacer un click sobre el mismo. Cuando esto ocurra el elemento cambiará de color indicando que efectivamente ha sido seleccionado.

2.2 Insertando Elementos Básicos

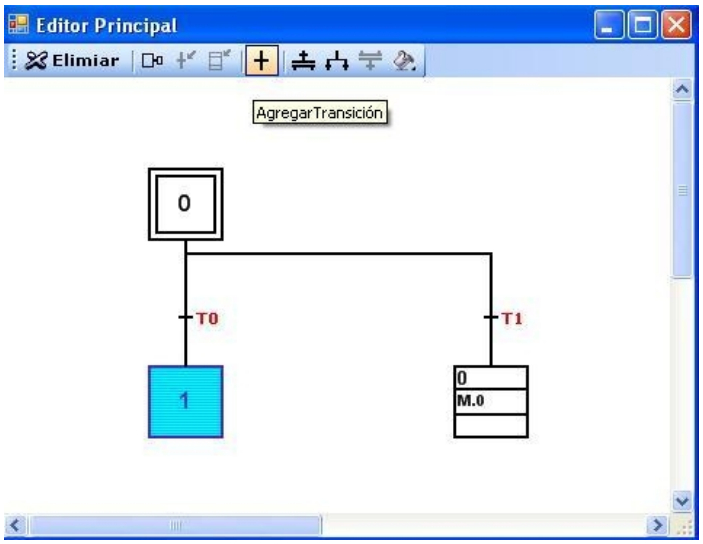
Como necesidad esencial para cada uno de los siguientes pasos se debe tener abierta y seleccionada el área de diseño.

2.2.1 Insertar una Transición

Para introducir una nueva transición al gráfico es necesario tener seleccionada la etapa o macroetapa que será la de partida para la transición.



Luego seleccione en el menú principal ya sea la opción **Diseño SFC>Agregar Transición** o en la barra de herramientas del formulario ***Editor Principal*** el botón de ***Agregar Transición.***

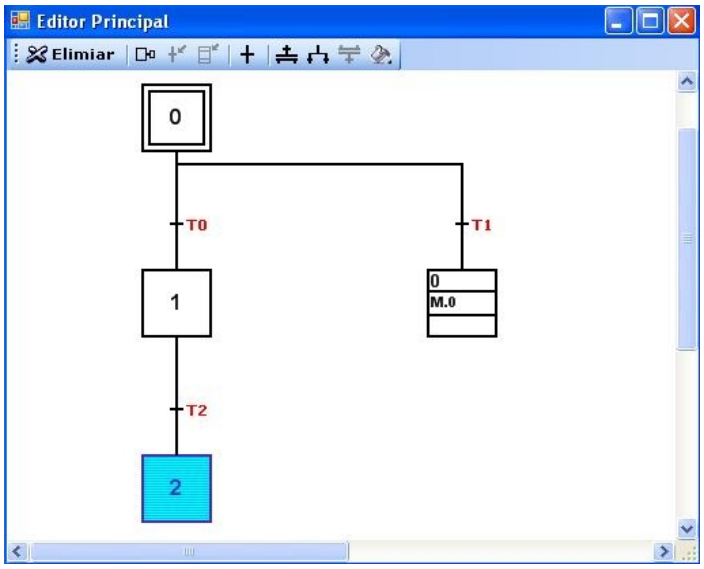


Al hacer esto se abrirá el dialogo asistente ***Enlace de Nueva Transición*** para elegir; de que manera será enlazada la nueva transición en el gráfico.

En este dialogo se presentan cuatro opciones de selección para el usuario, cada una representa una forma de enlace para el destino de la nueva transición del diseño.

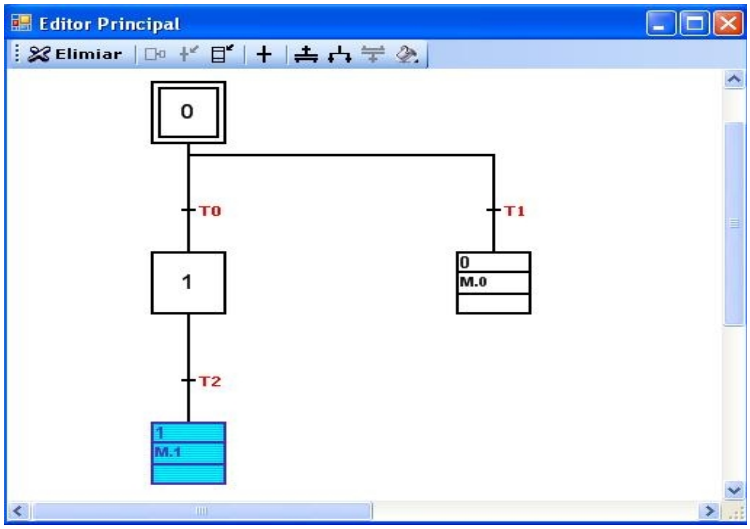
Una nueva Etapa: Si se selecciona ésta opción se creará una etapa nueva para el diseño y termina por enlazar la nueva transición en esta etapa que se crea.

Importante: La numeración de las etapas se genera y asigna de manera automática, el usuario no tiene por que preocuparse por eso. Cuando se inserte una nueva etapa por defecto no tiene ninguna acción asociada.

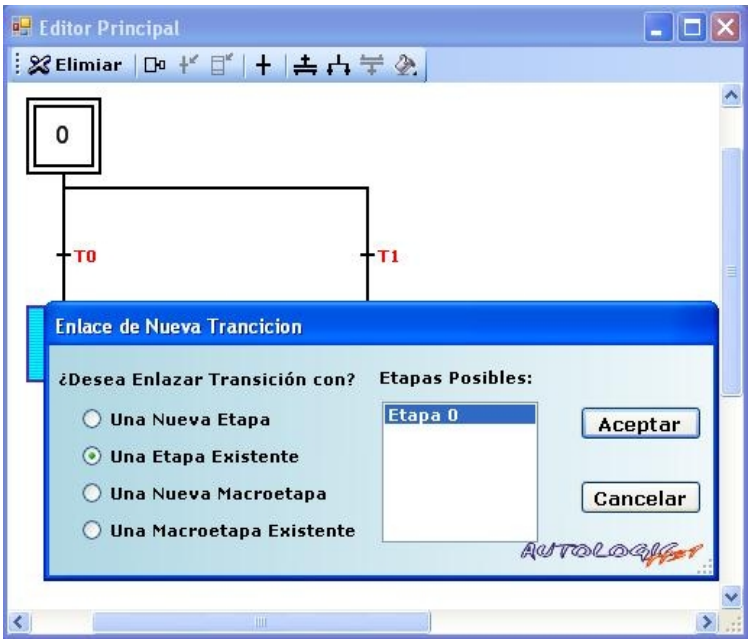


Una nueva Macroetapa: Si se selecciona ésta opción se creará una Macroetapa nueva para el diseño y termina por enlazar la nueva transición en esta Macroetapa que se crea.

Importante: El manejo de los identificadores que por defecto son visualizados sobre la macroetapa al crearse la misma, son manejados de manera automática por el software. Al crearse una nueva macroetapa se inicializa el gráfico correspondiente al diseño de la subrutina que la compete, de manera tal; que se define por defecto la etapa inicial.



Una Etapa Existente: Si se selecciona esta opción; el asistente desplegará una lista de las posibles etapas a seleccionar, para enlazar como etapa de destino para la nueva transición. Guardando las reglas de congruencia estipuladas en el presente proyecto. Al aceptar habiendo seleccionado se traza la transición como un reenvió hacia la etapa que fue seleccionada.



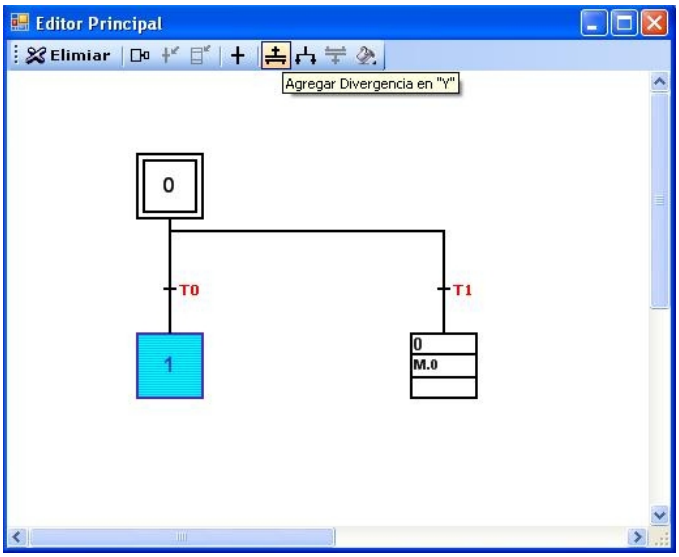
Una Macroetapa Existente: Si se selecciona esta opción se llevará a cabo basicamente el mismo proceso de la opción anterior, con la salvedad que en este caso se trabaja con Macroetapas.

Importante: Cuando un elemento se encuentra en una de las líneas de una bifurcación en Y, solo puede asociarse mediante transiciones con elementos única y exclusivamente pertenecientes a la misma bifurcación y línea de evolución, por esto cuando se quiere enlazar una nueva transición a un elemento existente se mostrarán solo en los que sería congruente asociar dicha transición.

2.2.2 Insertar una Divergencia en Y

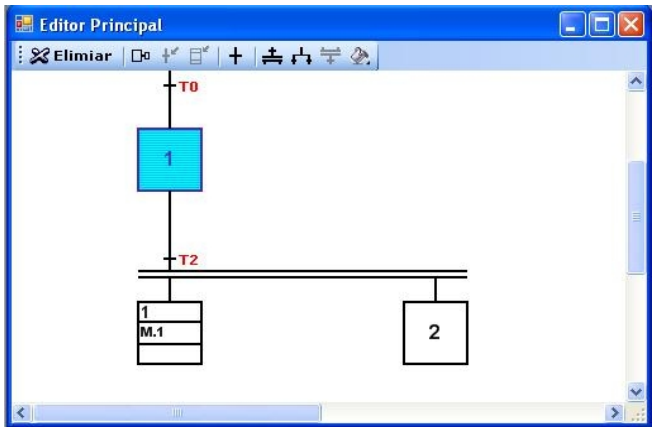
Para introducir una nueva Divergencia en Y al gráfico es necesario tener seleccionada la etapa o macroetapa que será la de partida para la misma.

Luego seleccione en el menú principal ya sea la opción **Diseño SFC>Agregar Divergencia en "Y"** o en la barra de herramientas del formulario ***Editor Principal*** el botón de ***Agregar Divergencia en "Y".***



Al hacer esto, se abrirá el dialogo asistente ***Agregar Divergencia en "Y"***, en el que podemos elegir el número de caminos por los cuales se hará la ejecución simultanea; especificándolo en el campo ***cantidad de transiciones,*** y con que tipo de elemento se desea comenzar cada una de esas líneas de ejecución; ya sea una etapa o una macroetapa; navegando en la lista de transiciones y especificando la opción deseada para cada una (Por defecto Una Nueva Etapa).

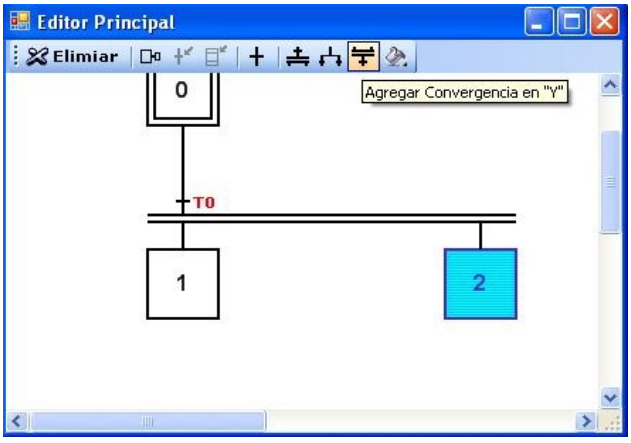
Al hacer click en el botón ***Aceptar*** se da paso a la creación de la divergencia, haciendo los enlaces pertinentes.



2.2.3 Insertar Convergencia en Y

Para que sea posible llamar al asistente de Agregar Convergencia en Y se debe tener seleccionada ya sea una etapa o una macroetapa que: no tenga transiciones salientes y que en su línea de evolución no exista otra etapa o macroetapa con esta característica, pues esta sería el único fin del camino. o que si tiene transiciones salientes, ninguna haya sido utilizada para crear una nueva etapa o macroetapa, es decir; solo puede tener transiciones representadas gráficamente como reenvíos, siempre y cuando no existan en su misma línea de evolución etapas o macroetapas que no tengan ninguna transición saliente; pues la prioridad se les da a ellas.

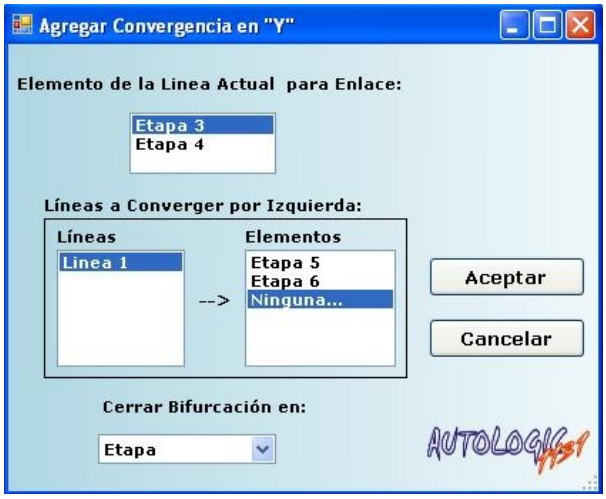
Si una de estas características se cumple, se produce un llamado a el asistente antes mencionado; desde el menú principal **Diseño SFC> Agregar Convergencia Y** o en su defecto sobre la barra de herramientas en el botón ***Agregar Convergencia Y***.



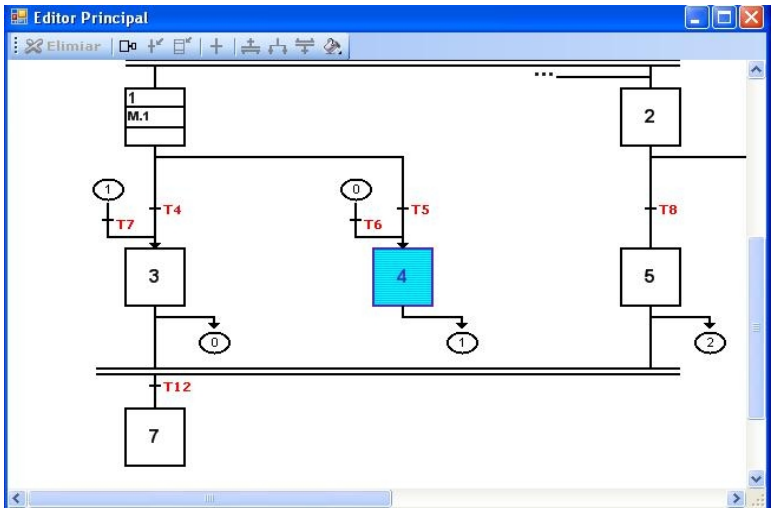
El cual presenta en primera instancia una lista que contiene los elementos a los cuales es congruente asociar la nueva convergencia y que a su vez

pertenecen a la misma línea de evolución del elemento seleccionado en el momento en el que se produjo el llamado. También presenta dos listas más, en las cuales se puede navegar por las líneas de evolución pertenecientes a la bifurcación del caso, y por los elementos a los cuales es congruente asociar la nueva convergencia y que a su vez pertenecen a la línea de evolución seleccionada en la lista anterior.

De esta manera se seleccionan uno por uno los elementos a los cuales se les asociará la convergencia y también el tipo de elemento al cual se convergerá, creando ya sea una nueva etapa o una nueva macroetapa.



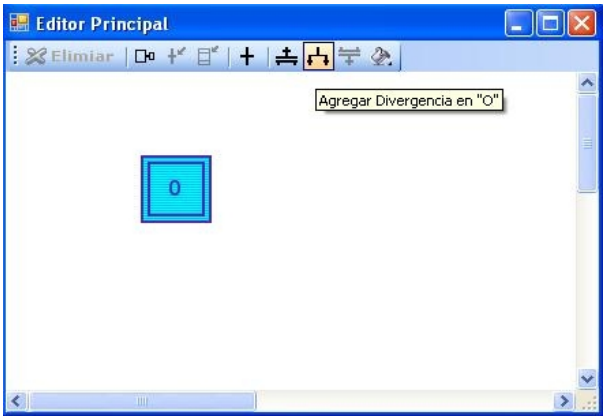
Con el click sobre el botón ***Aceptar*****,** se da paso a la creación de la convergencia; respondiendo gráficamente al siguiente patrón: Se converge debajo de la línea de evolución que se encuentre más a la izquierda y las líneas asociadas a esta que sean gráficamente consecutivas, tendrán trazadas sus correspondientes líneas cerradas con las que se enlazaran al elemento en el que se cerrará la bifurcación, mientras que las que no sean consecutivas, se cerrarán con reenvíos hacia el elemento en el que se cerrará la convergencia.



Importante: Cuando una línea de evolución es cerrada por una convergencia en Y es inhabilitada y no se podrá trabajar más gráficamente sobre ella, salvo la asociación de acciones a las etapas, la edición de condiciones combinacionales de las transiciones y la edición completa de las subrutinas en las macroetapas. Una vez cerradas las líneas para volver a estar activas; se necesita eliminar la convergencia que las cierra.

2.2.4 Insertar una Divergencia en O

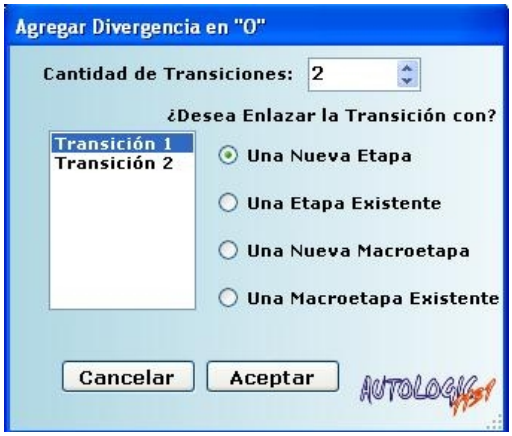
Primero se selecciona la etapa o macroetpa de la cual partirá la bifurcación (Haciendo click sobre ella), seguido de esto; para hacer el llamado al asistente de divergencia en O, se debe hacer click sobre la opción en el menú principal **Diseño SFC>Agregar Divergencia en "O"** o en su defecto sobre el botón de la barra de herramientas ***Agregar Divergencia en "O".***



En el diálogo asistente para el enlace de las transiciones asociadas a la divergencia, se especifica la cantidad de líneas en las cuales se abrirá la línea de evolución; en el campo ***cantidad de transiciones***, y se especifica la manera de enlazar cada una de las transiciones pertenecientes a la divergencia; navegando en la ***lista de las transiciones*** y para cada una se presentan cuatro opciones.

- **Una Nueva Etapa:** Crea una nueva etapa a partir de esta con la transición seleccionada.
- **Una Etapa Existente:** Enlaza la transición como transición entrante, en una etapa de las mostradas para seleccionar en la lista de etapas disponibles para el enlace (Esto se hace de manera dinámica)
- **Una Nueva Macroetapa:** Crea una nueva macroetapa a partir de esta con la transición seleccionada en la lista de transiciones
- **Una Macroetapa Existente:** Enlaza la transición como transición entrante, en una etapa macroetapa de las mostradas para seleccionar en la lista de macroetapas para el enlace (Esto se hace de manera dinámica)

Importante: Si en la opción de etapa existente o de macroetapa existente se selecciona un elemento este dejará de ser mostrado en la lista de disponibles para otra transición de la misma divergencia hasta que se quite la selección que hay sobre el.



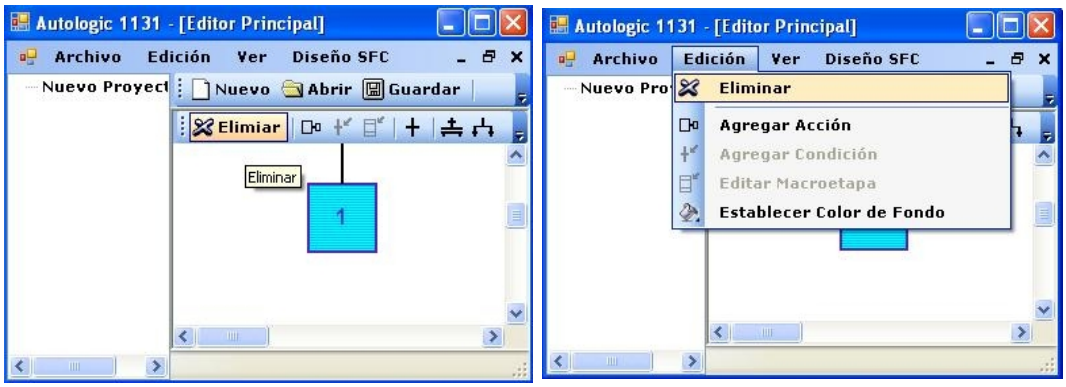
Al estar todo enlazado de manera congruente se hace click sobre el botón **Aceptar** y se lleva a cabo la creación de la divergencia.

2.3 Eliminando Elementos del Diseño

Importante:

- Para que sea posible eliminar una **etapa o macroetapa** del gráfico; esta debe carecer de transiciones salientes.
- Cuando se elimina una **etapa o macroetapa** del gráfico se borran automáticamente las transiciones entrantes de la misma.
- Cuando se elimina una etapa automáticamente se borran las acciones asociadas a la misma.
- Cunado se elimina una macroetapa automáticamente se borra en cascada el diseño de las subrutinas que contenga la misma y todos los elementos que las componen.
- Las únicas transiciones que permiten ser eliminadas son las transiciones secundarias (Representadas como reenvíos).
- Para que sea posible Eliminar una Convergencia en Y, el elemento en el que convergen las líneas de evolución no debe tener transiciones salientes.
- Para que sea posible eliminar una Divergencia en Y, debe tener solo los primeros elementos de cada línea de evolución ya sea la etapa o macroetapa donde comienza cada una.

Para eliminar cualquier elemento seleccionable de diseño, se hace click ya sea sobre el elemento del menú principal **Edición>Eliminar** o sobre el botón de la barra de herramientas ***Eliminar***.



2.3.1 Eliminar una Etapa: Para eliminar una etapa solo basta con seleccionar la misma haciendo click sobre ella y luego haciendo click sobre el botón ***Eliminar***.

2.3.2 Eliminar una Macroetapa: Para eliminar una macroetapa solo basta con seleccionar la misma haciendo click sobre ella y luego haciendo click sobre el botón ***Eliminar***.

2.3.3 Eliminar una Transición: Como se dijo antes se debe tener en cuenta que las únicas transiciones que se permiten eliminar son las que están representadas con reenvíos. Para eliminar cualquiera de estas

transiciones, solo basta con seleccionarla haciendo click sobre ella y luego haciendo click sobre el botón ***Eliminar***.

2.3.4 Eliminar una Divergencia en Y: Para eliminar una divergencia en Y solo basta con seleccionarla haciendo click sobre ella y haciendo click sobre el botón de ***Eliminar***.

2.3.5 Eliminar una Convergencia en Y: Para eliminar una convergencia en Y solo basta con seleccionarla haciendo click sobre ella o seleccionando ya sea la etapa o la macroetapa en la que se converge, y haciendo click sobre el botón ***Eliminar*.**

2.4 Editando Elementos del Diseño

2.4.1 Editando una Etapa

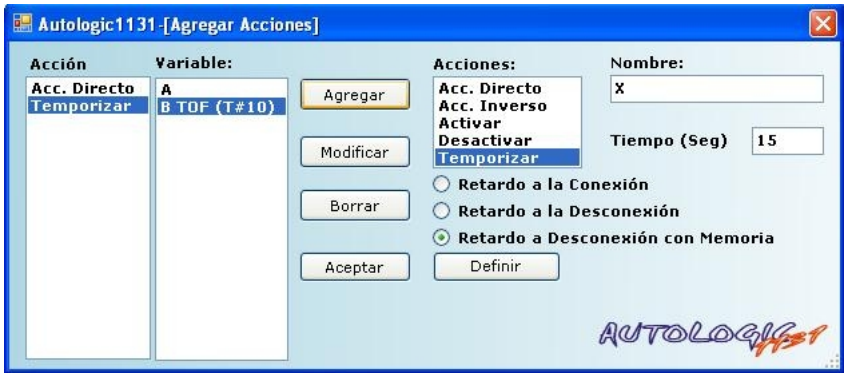
Cuando se habla de editar una Etapa se puede contextualizar en diferentes ámbitos.

1. **Editar las Acciones Asociadas:** Para agregar borrar o modificar las acciones asociadas a una etapa, se usa la ayuda del diálogo asistente **Agregar Acciones** el cual se puede llamar de varias formas; la primera es haciendo click derecho sobre la etapa y eligiendo la opción ***Agregar Acciones*** en el menú contextual, la segunda haciendo click en el elemento del menú principal **Edición>Agregar Acciones,** la tercera haciendo click sobre el botón de la barra de herramientas ***Agregar Acciones*** o simplemente haciendo doble click sobre la etapa.

   Al hacer esto, aparece en pantalla el asistente en mención el cual permite agregar la acción que se desee. Para el caso de **Acceso Directo, Acceso Inverso, Activar y Desactivar** alguna variable, se selecciona cualquiera de estos tipos de acción en la ***lista de Acciones***, se especifica el nombre de la variable en el campo ***Nombre***, por último; se hace click sobre el botón Agregar y la acción se agregará a la lista.

Para el caso de acciones con temporización, cuando se elige el elemento de la lista ***Acciones (Temporizar)***, aparecerán de forma dinámica tres opciones de temporización y un campo para asignar el tiempo de temporización en segundos. Se deberá decidir cualquiera de las opciones que significan:

- **Retardo a la conexión:** Cuando se elige esta opción, al cabo del tiempo estipulado la variable se activará en uno (Funciona mientras está la etapa activa).
- **Retardo a la Desconexión**: Cuando se elige esta opción se activará en uno la variable y en el momento en el que expire el tiempo estipulado se desactivará es decir; se pondrá en cero (Funciona mientras está la etapa activa).
- **Retardo a Desconexión con Memoria:** Cuando se elige esta opción las cosas funcionan de la misma manera que con la opción Retardo a la Desconexión salvo que esta funciona independientemente del estado de la etapa a la cual está enlazada.



Para trabajar con contadores, estos primero se deben definir. El asistente ***Agregar Acciones*** permite definirlos directamente. Haciendo click sobre el botón ***Definir,*** se desplegarán algunos controles para la definición de contadores. Se muestran dos opciones para seleccionar que tipo de contador usar en el caso:

- **Incremental CTU:** Define un contador incremental que se inicializa en cero y cuando alcanza el número de veces que recibe en la definición envía activa una señal.
- **Decremental CTD:** Define un contador decremental que se inicializa en el numero especificado en la definición y cuando llega a cero activa una señal para enviarla.

En el campo de **valor a contar** como su nombre lo indica se especifica el número de cuentas que llevará el contador. Para terminar de definirlo solo basta con pulsar sobre el botón **Adjuntar.**

Al haberse adjuntado contadores, en la lista de **Acciones** se habilitaran dos nuevas acciones: Inicializar envía una señal por la línea del reset del contador seleccionado para así ponerlo en su estado inicial e Impulso; que envía una señal por la línea de cuenta para incrementar o decrementar la misma y a su vez asociar la variable a la cual al concluir la cuenta se le enviará la señal de salida del contador.

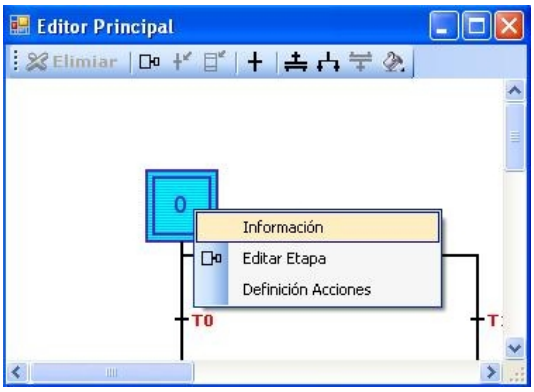
Al pulsar sobre el botón **Borrar**, eliminará el contador seleccionado y por ende todas las acciones asociadas a el.

Los contadores se pueden definir en cualquier punto del gráfico siempre y cuando sea llamado el dialogo asistente ***Agregar Acciones*** y una vez se agreguen, en cualquier punto se podrá trabajar con ellos, es decir desde cualquier etapa del gráfico funcional se podrá enviar una señal de inicialización o de cuenta a todos los contadores previamente definidos.

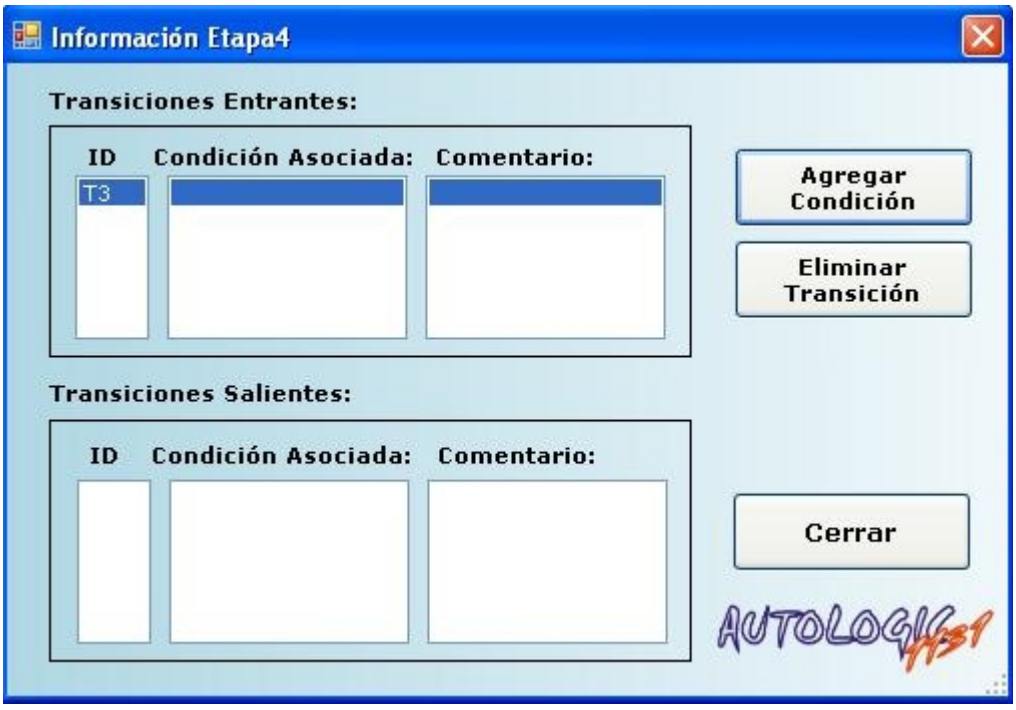
Por otra parte, al pulsar el botón Borrar en el ámbito de las acciones, elimina la acción asociada de la lista. Por último para modificar una acción es necesario tener llenos los campos de manera coherente y verificada para definir la acción que cambiará por la existente y haber seleccionado previamente una de las acciones de la lista.

Al adjuntar las acciones necesarias para la etapa se pulsa en el boton ***Aceptar*** y se concretará el enlace de las mismas a la etapa correspondiente.

2. **Editar las Transiciones Asociadas:** Es posible editar tanto las transiciones entrantes como las salientes a una etapa, al pulsar click derecho sobre una etapa y seleccionar la opción de ***Información*** en el menú contextual.



Se visualizará el dialogo asistente ***Información Etapa*** en el cual se listarán las transiciones asociadas a la etapa seleccionada; sobre las cuales, podemos asociar las condiciones lógicas combinacionales solo con seleccionarlas y pulsando sobre el botón ***Agregar Condición*** y también posibilitando eliminar las que sean posibles según las reglas de eliminación; solo con seleccionarlas y pulsando sobre el botón **Eliminar Transición**.



Al pulsar el botón ***Cerrar*** se guardarán los cambios y se visualizarán en el gráfico del diseño.

2.4.2 Editando una Macroetapa

Para abrir una ventana independiente del diseño del gráfico principal donde se podrá diseñar el gráfico correspondiente a la macroetapa y hacer la edición del mismo; primero se debe seleccionar la macroetapa pulsando un click sobre la misma. El siguiente paso se puede hacer de varias formas: Seleccionando en el menú principal la opción **Edición>Editar Macroetapa**, pulsando el botón de la barra de herramientas ***Editar Macroetapa*****,** haciendo click derecho sobre la macroetapa y eligiendo la opción ***Editar Macroetapa*** en el menú contextual o haciendo doble click sobre la macroetapa.

Con esto se visualizará el editor anteriormente mencionado y sus cambios se guardarán sobre el gráfico funcional cuando este se cierre.

También es posible editar las transiciones asociadas a la macroetapa de la misma manera que se hace con la etapa.

Por otro lado a las macroetapas también se les puede editar su etiqueta; esto se usa para proporcionar (si se desea), una etiqueta que represente alguna descripción de la subrutina del proceso que encierra la macroetapa; se puede ingresar pulsando click derecho sobre la macroetapa y eligiendo la opción ***Editar Etiqueta*** en el menú contextual y se visualizará un dialogo donde se especifica el texto que se visualizará en la etiqueta inferior de la macroetapa.

2.4.3 Editando una transición

Cuando una nueva transición se crea, sea cual sea su tipo; por defecto el campo de su función lógica combinacional es vacío. Para cambiar este valor y asignar una función combinacional pertinente se hace el llamado al dialogo asistente ***Agregar Función*** el cual se puede llamar de varias maneras: Pulsando sobre la opción del menú principal **Edición>Agregar Condición,** pulsando sobre el botón de la barra de herramientas del editor ***Agregar condiciones,*** pulsando click derecho sobre la transición

pertinente y eligiendo la opción ***Editar Transición*** del menú contextual o haciendo doble click sobre la transición pertinente.

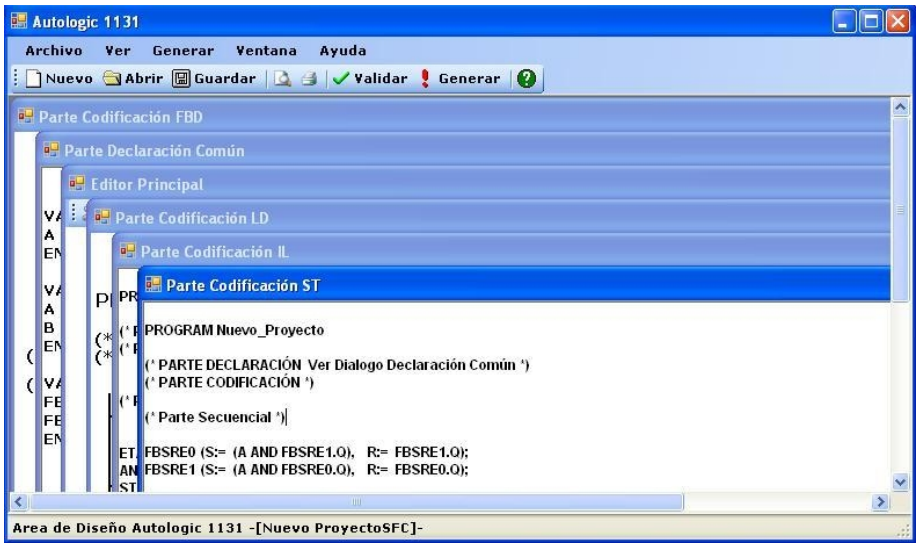
2.4.4 Editando una Divergencia o una Divergencia en Y

Las maneras de editar estos elementos son las mismas y tienen llamados idénticos a la edición de transiciones.

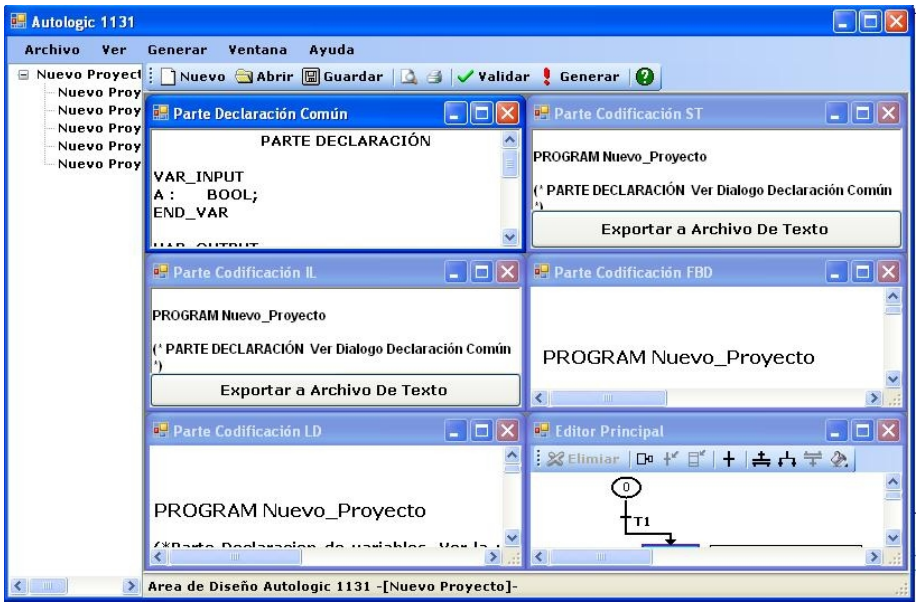
3. Formas de Visualización

En el menú principal es posible seleccionar una serie de opciones con las cuales se organizarán las ventanas de diferentes maneras automáticamente.

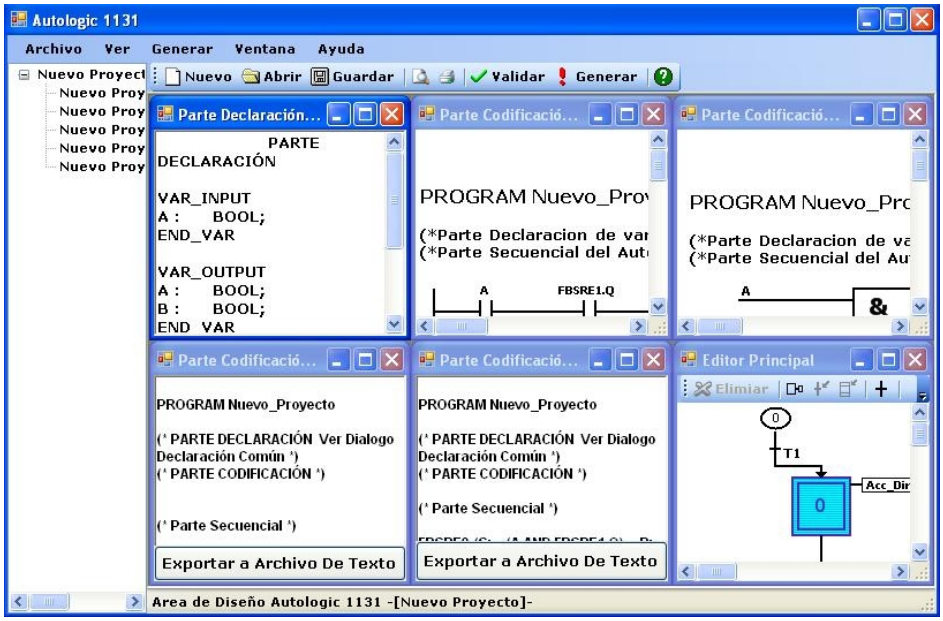
3.1 Visualización en Cascada: Se obtiene seleccionando **Ventana>Cascada**



3.2 Visualización en Título Horizontal: Se obtiene seleccionando **Ventana>Título Horizontal**



3.3 Visualización en Título Vertical: Se obtiene seleccionando **Ventana>Título Vertical**

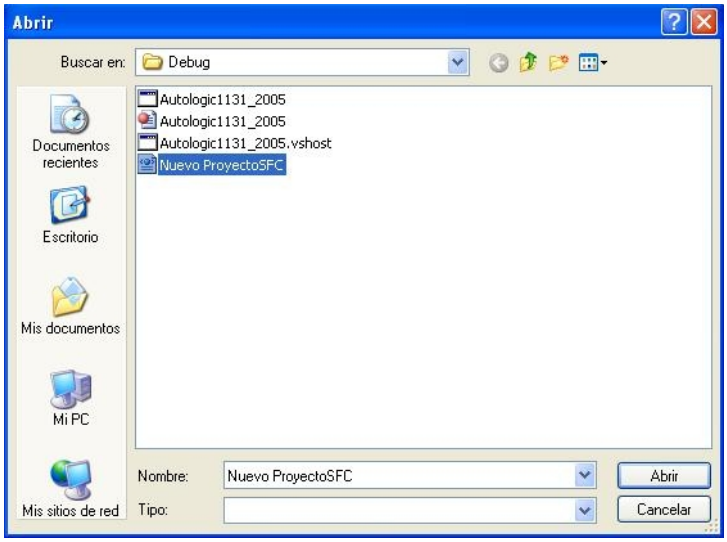


4. Abriendo un Proyecto

Para abrir un proyecto, primero se selecciona ya sea la opción **Archivo>Abrir Proyecto** en el menú principal o el botón ***Abrir*** en la barra de herramientas principal.

Con esto se visualizará al diálogo ***Abrir*** el cual permite navegar en el sistema de archivos del equipo, para seleccionar el archivo del proyecto que se abrirá.

El archivo seleccionado como requisito esencial debe mantener el **nombre del proyecto** y las siglas del final **"SFC",** Cuando se abre este archivo la herramienta verifica si este fue guardado junto con los leguajes generados y al ser así; carga todos estos en el área de trabajo, si es el caso contrario simplemente abrirá el archivo correspondiente al diseño SFC.



Los archivos de codificación generada también pueden ser abiertos independientemente según el gusto del usuario, simplemente siguiendo los pasos antes especificados para abrir un proyecto salvo que en este

caso se selecciona para abrir el archivo correspondiente al programa en el lenguaje requerido (Este debe guardar los mismos requisitos de Integridad que los especificados para el archivo de diseño).

5. Cerrar Proyecto

Para cerrar todo y dejar la ventana de la aplicación tal y como se encuentra recién se ejecuta la misma se selecciona la opción en el menú principal **Archivo>Cerrar Proyecto**.

6. Imprimir

Para imprimir un documento ya sea con el gráfico del diseño o con la codificación en alguno de los lenguajes generados debe:

- Seleccionar la ventana en la cual está la información que se va a imprimir.
- Seleccionar **Archivo>Imprimir**.
- Especifique las opciones de impresión y haga click sobre **Aceptar**.

Para una mejor configuración de lo que será la impresión resultante es aconsejable que realice primero los siguientes pasos:

- **Configurar Página**

  Puede configurar las propiedades de la página de presentación e impresión. Para hacerlo simplemente debe:

  - Seleccionar **Archivo>Configurar Página.**
  - Realizar la Configuración deseada y hacer click en aceptar.

- **Vista Previa**

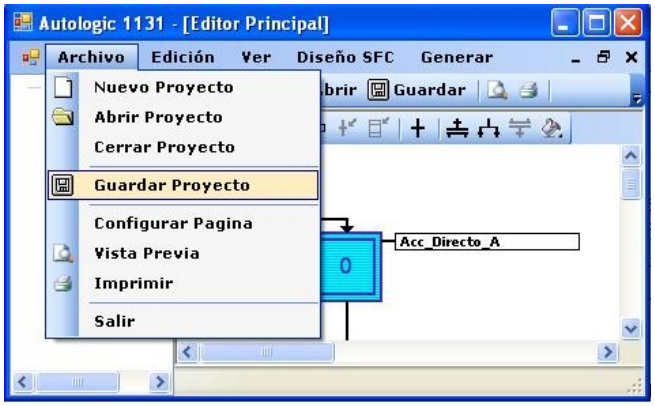
Tanto el diseño, como cada uno de los códigos generados, tienen su vista previa de manera independiente. Para acceder a ella debe:

- Seleccionar la ventana a la cual se requiere la presentación de vista previa.
- Seleccionar ya sea **Archivo>Vista Previa** o el botón ***Vista Previa*** en la barra de herramientas principal.

En la ventana de *Vista Previa* se puede ejecutar el mando de **Impresión**.

7. Guardando un Proyecto

Para ser guardada en el disco la toda información pertinente al trabajo en el **Autologic 1131,** basta con hacer click sobre la opción del menú principal **Archivo>Guardar Proyecto** o hacer click sobre el botón ***Guardar*** de la barra de herramientas principal.



La información se grabará en la ubicación elegida en el dialogo asistente ***Nuevo Proyecto*** usado si el presente proyecto fue iniciado desde cero. Si el proyecto actual era un proyecto previamente existente, la información se grabará en la ubicación desde la cual el mismo fue abierto.

La operación guardar puede ser solicitada en cualquier momento.

Dependiendo de la información existente en el proyecto en el momento que se guarda, los archivos que se crearán o sobre escribirán. Si no se ha generado código a partir del SFC, solo se creará un archivo que lleva el nombre del proyecto seguido de SFC, es decir; si el proyecto se llama **NuevoProyecto,** este archivo llevará como nombre **NuevoProyectoSFC.** Si se ha hecho una generación de código válida y está cargada en la aplicación, se creará un archivo para cada uno de los lenguajes descritos en el estándar IEC 1131-3, cuyo nombre será el

nombre del proyecto actual, seguido de las siglas representativas para cada uno de los leguajes (ST, IL, SFC, FBD y LD), es decir; si el proyecto actual se llama **NuevoProyecto**, los archivos creados se llamarán **NuevoProyectoSFC, NuevoProyectoLD, NuevoProyectoFBD, NuevoProyectoIL y NuevoProyectoST**.

8. Explorador de Ventanas

El Autologic 1131 presenta un explorador de ventanas en forma de árbol desplegable, que muestra las ventanas que se encuentran cargadas en la aplicación.

Esta permite navegar sobre ellas, seleccionando cada una con solo hacer un click sobre la etiqueta que represente la ventana pertinente en el explorador.

Así se pueden cerrar y abrir las ventanas que se deseen en el momento que se desee sin perder información.

Este explorador puede ser habilitado o viceversa seleccionando **Ver>Explorador de Ventanas.**

9. Establecer Color de Fondo

Para establecer un color para el fondo de todas las ventanas que tengan área de diseño se debe:

- Seleccionar ya sea **Edición>Establecer Color de Fondo** o el botón ***Establecer Color*** en la barra de herramientas del editor SFC principal.
- Navegar sobre las paletas de color, elegir el más indicado y hacer click en Aceptar.

10. Editar Descripción del Programa

Para un mejor entendimiento y facilitar la interacción de diferentes diseñadores sobre el problema, el **Autologic 1131** brinda un campo en el cual se podrá redactar la descripción de este.

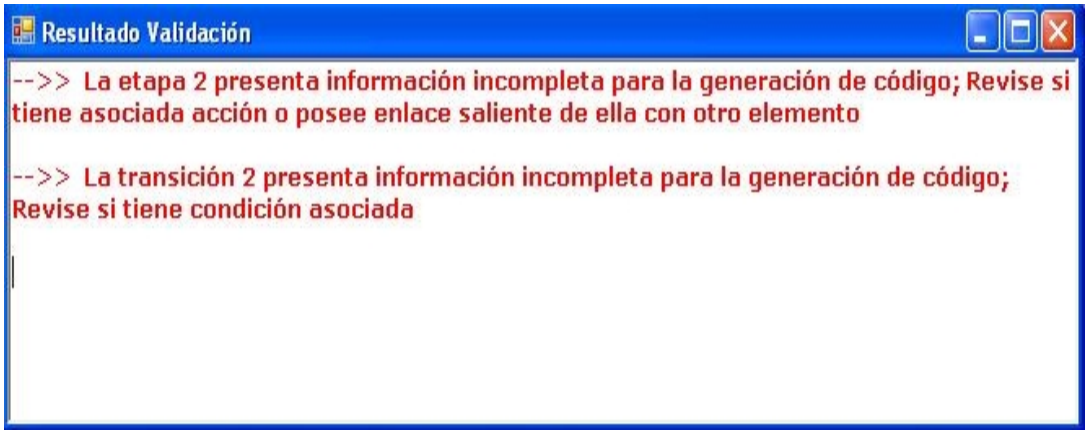
Para acceder al diálogo ***Descripción del Problema*** basta con seleccionar la opción **Ver>Descripción del Programa**, se redactará la misma y al oprimir aceptar se guardaran los cambios sobre el programa.

11. Validación

Para realizar la validación del diseño (SFC) basta con: Seleccionar la opción **Generar>Validación** o haciendo click sobre el botón ***Validar*** en la barra de herramientas.

Esto deberá tener dos tipos de respuestas:

- Un diálogo de mensajes; en el cual se le informan al usuario la falta de datos o la información errónea o incongruente que existe en el diseño, con el fin de que el usuario tenga conocimiento previo tanto de la naturaleza como la ubicación del problema.

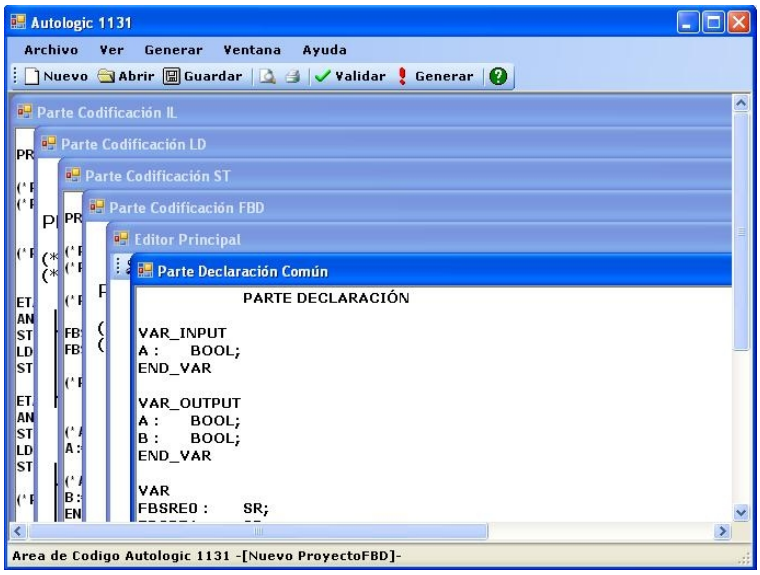


- O bien un diálogo de mensaje; en el cual se le comunica al usuario que su diseño esta completamente congruente y cumple con las condiciones necesarias para la generación de código.

12. Generación de Código

Para llevar a cabo la generación del código:

- Seleccione la opción **Generar>Codificación** o haga click sobre el botón ***Generar*** en la barra de herramientas principal.
- En seguida se produce automáticamente una validación del Diseño.
- Si la respuesta de la validación es satisfactoria se generarán automáticamente cinco formularios; ***Codificación en IL, Codificación en ST, Codificación en FBD, Codificación en LD*** y la parte de ***declaración de variables*** para el diseños principal que es común para todos.



13. Requerimientos del sistema

13.1 Hardware

- Procesador AMD Athlon XP de 1000 megahertz o superior.
- Memoria RAM 256 MB o mayor.
- Disco duro de 20 GB o mayor.
- Monitor a color de 15 pulgadas.
- Unidad de CD-R 42X o mayor.
- Unidad de disquete 3 ½.
- Mouse.
- Teclado.

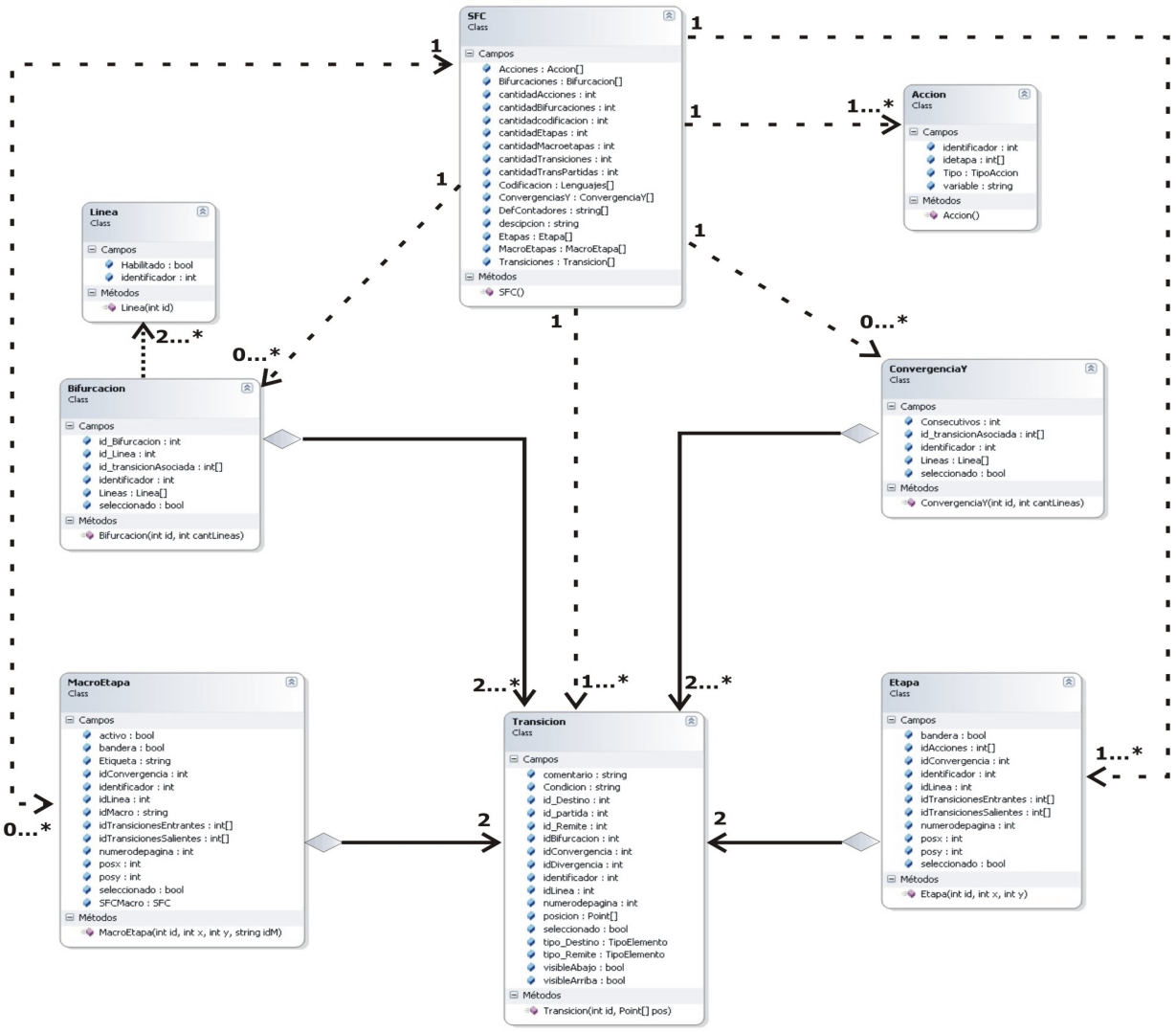
13.2 Software

- Sistema Operativo Windows XP, o superior.
- .NET Framework 2.0 o superior.

ANEXO B

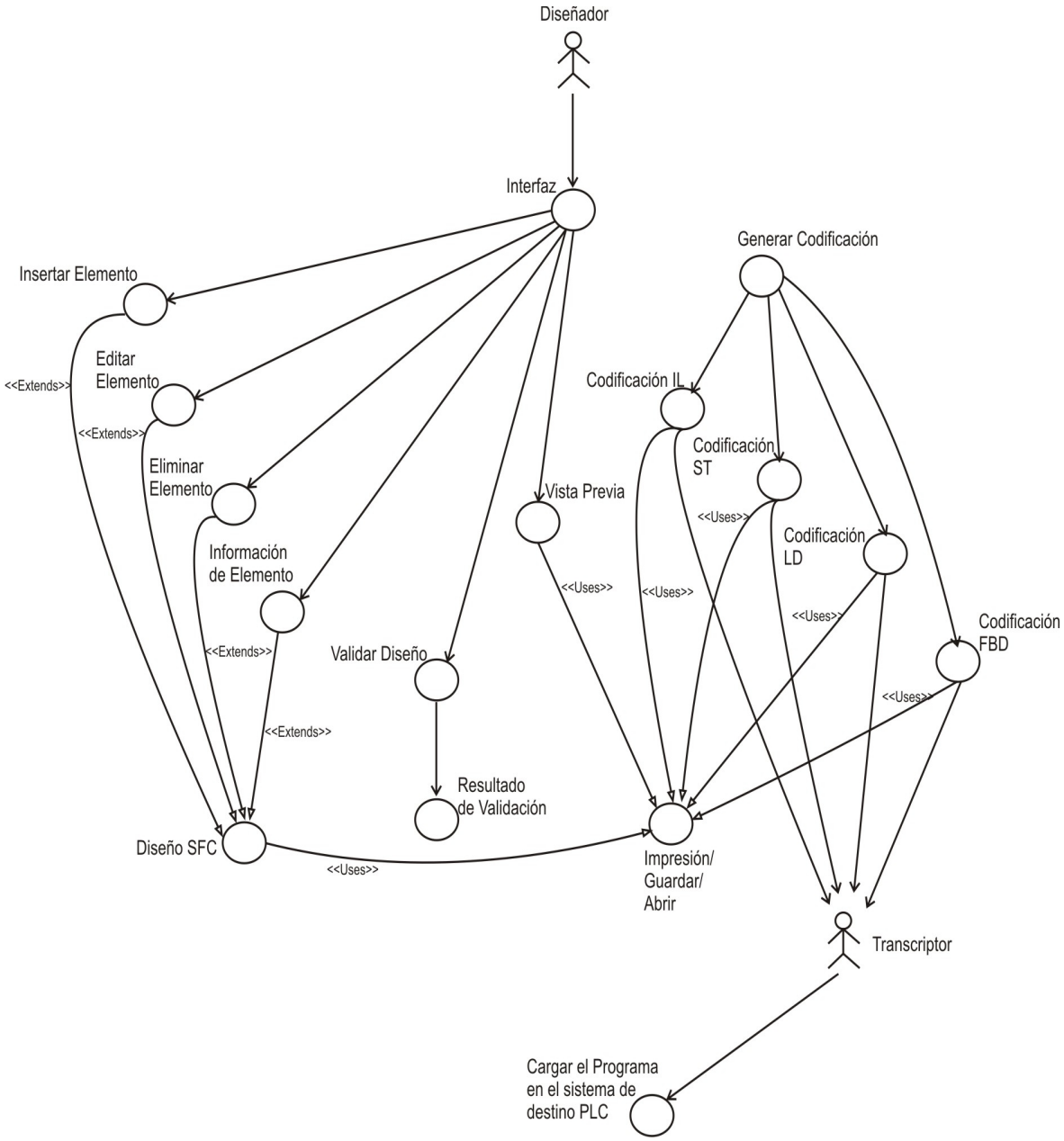
A continuación, se presenta el diagrama de clases; que representa la estructura de datos utilizada para el desarrollo de la herramienta software Autologic 1131. Esta estructura permite evidenciar el manejo dado a los datos para la construcción del diseño y su codificación, según el estándar IEC 1131.

1. DIAGRAMA DE CLASES



2. DIAGRAMA DE CASOS DE USO

Este diagrama representa, los casos de uso, en términos generales; que maneja la herramienta software Autologic 1131.



Los diagramas anteriormente expuestos, se obtienen mediante el uso del lenguaje de marcas unificado UML.