Revista INGENIERÍA UC
ISSN: 1316-6832
revistaing@uc.edu.ve
Universidad de Carabobo
Venezuela

Díaz P., José Gregorio; Dorta Franceschi, René; Arteaga, Francisco J.
Diseño de un software simulador de un autómata programable que utiliza el lenguaje de lista de instrucciones
Revista INGENIERÍA UC, vol. 12, núm. 1, abril, 2005, pp. 36-47
Universidad de Carabobo
Valencia, Venezuela

REVISTA INGENIERÍA UC. Vol. 12, Nº 1, 36-47, 2005

Diseño de un software simulador de un autómata programable que utiliza el lenguaje de lista de instrucciones

José Gregorio Díaz P., René Dorta Franceschi, Francisco J. Arteaga
Unidad de Investigación en Automatización Industrial, Escuela de Ingeniería Eléctrica
Universidad de Carabobo, Valencia, Venezuela
Email: jpdiazp@cantv.net, rene@uc.edu.ve, farteaga@uc.edu.ve

Resumen

En esta investigación, los autores han desarrollado un software que simula la ejecución por parte de un autómata programable o PLC (Programmable Logical Controller) de programas de aplicación, desarrollados con el lenguaje de lista de instrucciones (Instructions List Language o ILL). El conjunto de instrucciones que abarca el simulador es muy amplio, comprendiendo instrucciones booleanas, bloques funcionales (temporizadores, contadores, registros LIFO/FIFO, programadores cíclicos o drum controllers, contadores de pasos y registros de desplazamiento de bits o shift bit registers), instrucciones de salto, subrutinas, instrucciones booleanas y aritméticas para el procesamiento numérico de variables analógicas y también de comparación y asignación de variables numéricas. La simulación se hace a partir de un archivo de texto que contiene el programa de aplicación elaborado mediante un ILL, cuyo conjunto de instrucciones es compatible con lo establecido en la norma IEC-61131-3, publicada en el año de 1.993 por la International Electrotechnical Comission (IEC) o Comité Electrotécnico Internacional. El simulador trabaja ejecutando un ciclo similar al que realizan los autómatas programables reales, caracterizado por la secuencia: lectura de entradas, procesamiento del programa de aplicación y actualización de las salidas.

Palabras clave: Autómata programable PLC, lenguaje de lista de instrucciones, simulador.

Design of a simulation software for a programmable logical controller that uses the instructions list language

Abstract

In this research, the authors have developed a software that simulates the Programmable Logical Controller (PLC) execution of an application program using the Instructions List Language (ILL). The set of instructions is very wide incending boolean instructions, function blocks (timers, counters, LIFO/FIFO registers, drum controllers, step counters, and shift bit registers), jump instructions, subroutine, boolean instructions and arithmetic, comparison and assignment instructions for analog variables. The program to be simulated must be written using an ILL compatible with International Electrotechnical Comission (IEC) standard IEC-61131-3 (1993), and saved as a text file. The simulation process follows the execution cycle of real PLCs: input reading, application program processing, and output writing.

Keywords: Programmable logic controller PLC, instructions list language, simulator.

1. INTRODUCCIÓN

Los autómatas programables constituyen uno de los dispositivos de control que mayor difusión han encontrado dentro del ambiente industrial, comercial e incluso residencial. En ello han influido algunas de sus favorables características, como son: bajo costo relativo, pequeño tamaño, escasa disipación térmica, versatilidad, robustez mecánica, compatibilidad electromagnética más que satisfactoria, capacidad de integración dentro de redes de comunicación de datos y facilidad de programación para las aplicaciones de control secuencial típicas.

La solución de un problema específico de control por parte de un autómata programable requiere de su programación, para lo cual se han desarrollado lenguajes especiales, muchas veces solamente aplicables a los PLCs de un fabricante en particular. En busca de una estandarización mundial, el IEC normalizó cinco

de estos lenguajes: lista de instrucciones, escalera o de contactos, bloques funcionales, GRAFCET o SFC y estructurado. Uno de los más utilizados es el de lista de instrucciones, ya que muchos de los autómatas comerciales admiten un lenguaje de este tipo, el cual presenta similitudes con los lenguajes ensambladores de los microprocesadores.

Por otra parte, una forma de ahorrar tiempo y dinero a la hora de desarrollar un programa de aplicación para un PLC consiste en la simulación del comportamiento del mismo en un computador personal, mediante un software simulador apropiado, ya que de esta manera el programador puede ensayar sus estrategias en la oficina, taller o laboratorio, sin necesidad de transferir el programa a la memoria de un PLC real. Por otra parte, un simulador de este tipo también resulta de gran utilidad para las Instituciones de educación técnica y universitaria, en donde la enseñanza de la automatización basada en autómatas programables alcanza cada vez mayor difusión.

2. AUTÓMATAS PROGRAMABLES

Un autómata programable es un equipo electrónico de control con un cableado interno (hardware) independiente del proceso a controlar, que se adapta a dicho proceso mediante un programa específico (software) [1].

Las ventajas del autómata programable con respecto a otras tecnologías de control pueden resumirse en:

- Flexibilidad: al ser un dispositivo programable, puede adaptarse a cualquier modificación del proceso del cual forma parte. Inclusive, el PLC que hoy se está utilizando con una determinada maquinaria, puede destinarse mañana al control de otra, ameritando tan sólo un nuevo programa.
- Versatilidad: existen autómatas capaces de trabajar con entradas y salidas de muy variados tipos, como termopares, RTDs, encoders, etc.
- Robustez: tanto desde el punto de vista mecánico y térmico, como de compatibilidad electromagnética, el PLC es extraordinariamente robusto, razón por la cual se adapta muy bien a los exigentes ambientes industriales.
- Economía: dado que el PLC se ha transformado en un equipo de utilización masiva, los precios de los autómatas programables han descendido de tal manera, que hoy en día se les encuentra inclusive dentro del ambiente doméstico.
- Confiabilidad: años de explotación han demostrado que los autómatas programables son dispositivos sumamente confiables.
- Reducido tamaño: la microelectrónica incorporada dentro de los autómatas programables ha permitido reducir su tamaño, de forma que ocupan mucho menos espacio que otros dispositivos de control.
- Escasa disipación térmica: es muy poco el calor que genera un PLC en operación; por lo tanto, su utilización dentro de un tablero eléctrico tiende a afectar muy poco a los demás dispositivos que se encuentran en el interior del mismo.
- Facilidad de programación: la programación de los PLCs es relativamente sencilla, cuando se limita a problemas de automatización industrial.
- Rapidez de instalación y puesta en marcha: dado que la comunicación del PLC con el mundo exterior se hace a través de borneras instaladas sobre él mismo, su conexión con los diferentes elementos que forman parte del sistema que se desea controlar es una tarea fácil y rápida. Además, la puesta en marcha del conjunto puede realizarse en un tiempo extremadamente breve, ya que la lógica de control reside en un programa (el cual se puede probar independientemente y con antelación), en lugar de ser cableada.
- Poca necesidad de mantenimiento: el PLC no incluye ningún tipo de partes móviles, por lo que sus requerimientos de mantenimiento son mínimos (limpieza, verificación de conexiones y condiciones de ventilación, etc.)
- Fácil y rápida ubicación de fallas: si la lógica de control de un proceso demuestra ser defectuosa, se puede ubicar rápidamente la falla visualizando en la pantalla de un computador o en el terminal de programación del PLC cómo es la evolución del programa, los sucesivos valores que van adquiriendo las variables, etc.
- Capacidad de comunicación con otros dispositivos: hoy en día, los autómatas programables se conciben como elementos integrados dentro de una red digital de comunicación de datos que puede abarcar otros autómatas, computadoras, terminales de diálogo hombre – máquina, controladores de procesos, pulsadores, detectores, etc.

3. CICLO DE FUNCIONAMIENTO DEL AUTÓMATA

Los autómatas programables son máquinas secuenciales que ejecutan correlativamente las instrucciones indicadas en el programa de usuario que ha sido almacenado en su memoria, generando unas órdenes o señales de mando a partir de las señales de entrada leídas de la planta o proceso bajo su control y de la historia pasada del mismo. Al detectarse cambios en las señales, el PLC reacciona según el programa, hasta obtener las órdenes de salida necesarias. Esta secuencia se ejecuta continuamente a fin de conseguir el control actualizado del proceso.

El ciclo de operación del autómata se puede dividir en tres fases principales [2], que se suceden en el tiempo y que se repiten periódicamente:

- Lectura de señales desde la interfaz de las entradas. Estos valores se almacenan en la memoria del PLC y son consultados por el programa cada vez que se hace referencia a una operación que involucre a una o más de las entradas.
- Procesamiento del programa a fin de obtener las señales de control necesarias, adaptadas al estado actual de la planta o proceso. Los valores generados para las variables internas y las de salida se almacenan en memoria.
- Escritura de señales en la interfaz de salidas. Los valores almacenados en memoria para las diferentes variables de salida se vacían en la bornera de salida.

Aparte de esto, el autómata también realiza una serie de acciones comunes que garantizan la seguridad de su funcionamiento, como son las verificaciones de la memoria y de la unidad central de procesamiento (CPU), comprobación del watchdog o perro guardián, etc. El PLC también puede establecer una comunicación periódica con periféricos exteriores, como terminales de diálogo hombre – máquina, otros autómatas, un computador personal, etc.

4. PROGRAMACIÓN DE LOS AUTÓMATAS

Se denomina programación del autómata al establecimiento de la secuencia de órdenes que fijan la ley general de mando, a partir de la cual se obtienen las variables de salida o de control [3-10].

Prácticamente desde su primera introducción al mundo industrial, se ha venido empleando una amplia gama de técnicas de programación aplicadas a los autómatas. Ello originó que, en un principio, prácticamente cada marca de PLC (e incluso cada serie) poseyera sus propios lenguajes de programación, lo que obligaba a los usuarios al estudio en profundidad de muchos de ellos. Frente a esta indeseable situación, la International Electrotechnical Comission (IEC) generó en 1.993 la norma IEC-61131-3, con la cual aspiraba a establecer unas bases mínimas comunes para los diferentes lenguajes de programación de PLCs.

En este estándar se acogen cinco diferentes lenguajes de programación:

- Lista de instrucciones. Se trata de un lenguaje muy parecido a un assembler o ensamblador de bajo nivel, por lo que sus instrucciones son bastante básicas, lo cual lo hace relativamente difícil y engorroso de aplicar. Es el más próximo al verdadero lenguaje de máquina que emplea el PLC durante su operación, por lo que el examen de un programa realizado con este lenguaje proporciona la mejor visión de cómo es realmente la secuencia de operaciones e instrucciones que va ejecutando el autómata.
- Escalera o de contactos. Es un lenguaje gráfico basado en la lógica de relés, si bien admite añadidos (como bloques de función, comparación y asignación), inexistentes dentro del campo de los relés. Su principal ventaja radica en su carácter visual, lo que lo hace muy poderoso a la hora de localizar errores, identificar la lógica de un programa sencillo, etc. Sin embargo, resulta engorroso a la hora de programar aplicaciones de mediana y alta complejidad, ya que no se presta para la programación jerárquica y estructurada. A pesar de todo, es el lenguaje para programación de autómatas más popular.
- Diagrama de bloques funcionales. Se trata de otro lenguaje gráfico en donde se representa el flujo de señales y datos a través de bloques funcionales que incluyen las operaciones booleanas básicas, biestables, etc. Los programas elaborados con este lenguaje tienen el aspecto de un plano de compuertas lógicas digitales.
  GRAFCET (Gráfico Funcional de Control de

Etapas y Transiciones) o SFC (Sequential Flow Chart o Carta de Flujo Secuencial). Es otro lenguaje gráfico, muy útil para representar el comportamiento secuencial de un sistema de control. Se le utiliza en la definición de secuencias de control gobernadas por eventos o el paso del tiempo. Brinda soluciones muy sencillas y elegantes para los sistemas de secuencia única; sin embargo, en la medida en que se presentan alternativas condicionales, pierde parte de su efectividad, ya que sus diagramas se hacen exageradamente engorrosos.

- Lenguaje estructurado. Se trata de un lenguaje textual de alto nivel que se presta muy bien para la programación estructurada. Su sintaxis es muy parecida a la de Pascal o C y soporta una amplia gama de funciones y operadores estandarizados. Lamentablemente, son pocos los autómatas que admiten en la actualidad la programación realizada con este lenguaje.

5. LENGUAJE DE LISTA DE INSTRUCCIONES UTILIZADO POR EL SIMULADOR

En las Tablas 1, 2, 3, 4, 5, 6, 7, 8 y 9 se identifican los operandos y operaciones que admite el simulador desarrollado durante esta investigación.

6. ESPECIFICACIONES BÁSICAS PARA EL DISEÑO DEL SIMULADOR

A fin de proceder con la programación del simulador se decidió desde un principio que éste debía contar con las siguientes características:

1. Debía funcionar en el ambiente Windows.
2. Los programas de aplicación a ser simulados estarían contenidos en archivos .TXT, generados por cualquier procesador de textos.
3. Sin embargo, resulta preferible que los archivos .TXT provengan de un editor de programas adaptado al lenguaje de instrucciones (ILL) de IEC, como los utilizados para la programación de muchos autómatas comerciales. De esta manera se aprovecharían los excelentes depuradores o debuggers que acompañan a estos editores. Inclusive, con algunos PLCs es posible programar mediante el lenguaje escalera, traducir el programa a lista de instrucciones y luego exportar el mismo como un archivo .TXT.
4. El conjunto de instrucciones aceptadas por el simulador sería el contenido en la sección 5.
5. Tal como sucede con muchos autómatas comerciales, el simulador dispondría de un único acumulador y de un stack o pila de ocho niveles.
6. La simulación realizada con el software seguiría el ciclo de funcionamiento del PLC esbozado en la sección 3.
7. Las cantidades de entradas y salidas (booleanas y analógicas), palabras constantes, palabras digitales y bits internos y bloques funcionales serían similares a las de un autómata comercial de la gama baja.
8. Para realizar una simulación sería necesario realizar los siguientes pasos:

   a. Proceder a ejecutar el archivo .EXE del simulador.

   b. Desde la barra de menús del simulador (opción Archivo), proceder a cargar el archivo.TXT que contiene el programa del PLC que se desea simular. La lista del mismo debe aparecer en la ventana principal del simulador.

   c. Dado que los bloques funcionales deben ser configurados antes de proceder con la simulación (por ejemplo, en el caso de los temporizadores es necesario definir el tipo de temporizador, su base de tiempo y preset, con los contadores hay que especificar el preset, etc.), una vez cargado el archivo .TXT, se habilita en la barra de menús la opción Configurar, sobre la cual se debe hacer clic antes de continuar. El simulador determina automáticamente si el programa contiene elementos que deben ser configurados. En caso afirmativo, abre una o varias ventanas que permitan realizar esta actividad. De lo contrario, simplemente aparece un mensaje en pantalla, advirtiendo al usuario que no se va a realizar ninguna configuración previa.

   d. Una vez realizado lo anterior, se habilita en la barra de menús la opción de Simular. Haciendo clic en ella aparece una ventana adicional que permitirá: 1) seleccionar el tipo de simulación (continua o paso a paso), 2) forzar los valores de las entradas booleanas y analógicas, 3) observar los valores en memoria de las salidas booleanas y analógicas,

Tabla 1 Lista de bits operandos.

Ítem	Tipo	Ejemplo	Descripción
1	Valores inmediatos	0 ó 1	Verdadero o falso
2	Bits de entrada	%I0.0, %I0.1, etc.	Entradas booleanas del PLC
3	Bits de salida	%Q0.0, %Q0.1, etc.	Salidas booleanas del PLC
4	Bits internos	%M0, %M1, etc.	Variables booleanas internas del PLC
5	Bits de bloques funcionales	%TM0.Q, %DR0.F, %C1.D, etc.	Bits propios de contadores, temporizadores, etc.
6	Bits de bloques funcionales (versión reversible)	E, D, F, Q, etc.	Ídem al anterior (expresado de forma diferente)
7	Bits extraídos de palabras	%IW0.0:X.4, %MW12:X.6, %KW5:X.0, etc.	Bits extraídos a partir de palabras digitales de entrada, internas, de salida, etc.

Tabla 2 Instrucciones booleanas básicas.

Instrucción	Operando[(1)]	Descripción[(2)]	Ejemplo[(3)]
LD	Ítems 1, 2, 3, 4, 5, 6 ó 7	Carga el bit operando	LD 0, LD %I0.4, etc.
LDN	Ídem al anterior	Carga el complementario del bit operando	LDN %Q0.5
LDR	Ítem 2	Carga un 1 cuando se detecta un flanco ascendente	LDR %I0.2
LDF	Ídem al anterior	Carga un 1 cuando se detecta un flanco descendente	LDF %I0.7
ST	Ítems 3, 4, 5 ó 7	Copia el acumulador. en el operando	ST %Q0.3
STN	Ídem al anterior	Copia el complementario del acumulador en el operando	STN %Q0.6
S	Ídem al anterior	Si el acumulador es 1, hace *set* sobre el operando	S %M19
R	Ídem al anterior	Si el acumulador es 1, hace *reset* sobre el operando	R %M19

(1) Los ítems indicados se refieren a los de la Tabla 1.
(2) La carga consiste en la copia del valor en el acumulador.
(3) Los bits de entrada y de salida siempre tendrán el dígito 0 antes del punto.

4) detener la simulación, y 5) cerrar esta ventana de simulación. Cuando se esté simulando bajo la modalidad de paso a paso (es decir, que la simulación hace una pausa en cada instrucción), se debe destacar en el listado del programa aquella instrucción que está siendo procesada por el simulador.

e. Mientras se esté realizando la siventanalación, en la ventana principal estará activa la opción de Visualizar. Haciendo clic sobre ella, se abre una ventana adicional, la cual permitirá: 1) seleccionar una o más variables booleanas o analógicas internas que se desea visualizar (palabras digitales o bits internos, de temporizadores, contadores, programadores cíclicos, etc.), y 2) visualizar los valores actuales de estas variables.
f. En la barra de menús se tendrá una opción de Ayuda con información general acerca del simulador y de su operación.
g. En cualquier momento se podrá cerrar el archivo que se está simulando o cerrar el programa simulador.

Tabla 3 Operaciones booleanas básicas.

Instrucción	Operando[1]	Descripción[2]	Ejemplo
AND	Ítems 1, 2, 3, 4, 5, 6 ó 7	Realiza el AND entre el operando y el acumulador	AND %M17
ANDN	Ídem al anterior	Realiza el AND entre el complementario del operando y el acumulador	ANDN %Q0.2
ANDR	Ítem 2	Realiza el AND entre el eventual flanco ascendente del operando y el acumulador	ANDR %I0.1
ANDF	Ídem al anterior	Realiza el AND entre el eventual flanco descendente del operando y el acumulador	ANDF %I0.1
OR	Ítems 1, 2, 3, 4, 5, 6 ó 7	Realiza el OR entre el operando y el acumulador	OR %M17
ORN	Ídem al anterior	Realiza el OR entre el complementario del operando y el acumulador	ORN %Q0.2
ORR	Ítem 2	Realiza el OR entre el eventual flanco ascendente del operando y el acumulador	ORR %I0.1
ORF	Ídem al anterior	Realiza el OR entre el eventual flanco descendente del operando y el acumulador	ORF %I0.1
XOR	Ítems 1, 2, 3, 4, 5, 6 ó 7	Realiza el XOR entre el operando y el acumulador	XOR %TM0.Q
XORN	Ídem al anterior	Realiza el XOR entre el complementario del operando y el acumulador	XORN %M12
XORR	Ítem 2	Realiza el XOR entre el eventual flanco ascendente del operando y el acumulador	XORR %I0.3
XORF	Ídem al anterior	Realiza el XOR entre el eventual flanco descendente del operando y el acumulador	XORF %I0.4
N	Sin operando	Complementa el valor del acumulador. El resultado se almacena en éste.	N
AND(	Ítems 1, 2, 3, 4, 5, 6 ó 7	[3]	AND(%M56
AND(N	Ídem al anterior	[3]	AND(N %Q0.5
AND(R	Ítem 2	[3]	AND(R %I0.7
AND(F	Ídem al anterior	[3]	AND(F %I0.7
OR(	Ítems 1, 2, 3, 4, 5, 6 ó 7	[3]	OR(%TMO.Q
OR(N	Ídem al anterior	[3]	OR(N %M34
OR(R	Ítem 2	[3]	OR(R %I0.0
OR(F	Ídem al anterior	[3]	OR(F %I0.4
)	Sin operando	Cierre de paréntesis	)

[1] Los ítems indicados se refieren a los de la tabla 1.
[2] El resultado de la operación se copia en el acumulador.
[3] La presencia de paréntesis modifica el orden de ejecución de las instrucciones del programa, ya que siempre se deben ejecutar en primer lugar aquéllas que se encuentren enmarcadas por los paréntesis más internos.

Tabla 4 Instrucciones booleanas especiales. [1]

Instrucción	Descripción
MPS	Copia el acumulador en el tope del *stack*. Los demás elementos de la pila son empujados hacia abajo
MRD	Copia en el acumulador el valor que está en el tope del *stack*
MPP	Descarga en el acumulador el valor que está en el tope del *stack*. Los demás elementos de la pila suben un nivel.

[1] Estas instrucciones gobiernan el funcionamiento del stack o pila del PLC. Ninguna de ellas admite operando.

Tabla 5 Bloques funcionales.

Bloque	Bits asociados[1]	Palabras asociadas[2]
Temporizador [3]	%TMi.Q: Salida del temporizador	%TMi.P: *Preset*
		%TMi.V: Tiempo transcurrido (en bases de tiempo)
Contador [4]	%Ci.D: Igual a 1 cuando la cuenta se hace igual al *preset*	%Ci.P: *Preset*
	%Ci.E: Igual a 1 cuando la cuenta pasa de 0 a 9999 (con cuenta decreciente)	%Ci.V: Valor actual de la cuenta
	%Ci.F: Igual a 1 cuando la cuenta pasa de 9999 a 0 (con cuenta creciente)	
Registro LIFO/FIFO	%Ri.E: Registro vacío	%Ri.I: Palabra de entrada al registro
	%Ri.F: Registro lleno	%Ri.O: Palabra de salida del registro
Programador cíclico [5]	%DRi.F: 1 cuando el paso actual es igual al último	%DRi.S: Paso actual
Registro de desplazamiento de bits [6]	%SBRi.j: bit j del registro de desplazamiento de bits i	
Contador de Pasos [7]	%SCi.j: bit j del contador de pasos i	

[1] La letra i corresponde a la identificación numérica del bloque en particular.
[2] Estas palabras pueden ser leídas desde el programa de aplicación y, en algunos casos, modificadas desde el mismo.
[3] Se dispone de tres tipos de temporizadores: On delay, Off delay y tipo pulso.
[4] Los contadores son ascendentes/descendentes. Además, admiten set y reset.
[5] Los pasos del programador pueden variar de forma ascendente o descendente
[6] Almacena datos boléanos que pueden ser desplazados en una u otra dirección.
[7] El contador puede avanzar o retroceder. A cada paso se le asigna un bit.

Tabla 6 Instrucciones de salto y subrutina.

Instrucción	Descripción	Ejemplo
JMP	Salto incondicional a la instrucción identificada con la etiqueta	JMP %L6
JMPC	Se salta a la instrucción identificada con la etiqueta sólo si el acumulador es igual a 1	JMPC %L12
JMPCN	Se salta a la instrucción identificada con la etiqueta sólo si el acumulador es igual a 0	JMPCN %L3
SRi	Salto a la subrutina i	SR3
SRi:	Inicio de la subrutina i	SR3:
RET	Final de una subrutina	RET
%Li:	Identifica a la siguiente instrucción con la denominación i	%L1:

Tabla 7 Instrucciones de final de programa.

Instrucción	Descripción
END	Final incondicional del programa de aplicación
ENDC	Final del programa si el acumulador es igual a 1
ENDCN	Final del programa si el acumulador es igual a 0

Tabla 8 Lista de palabras operandos.

Ítem	Tipo	Ejemplo	Descripción
1	Valores Inmediatos (1)	458 16#4FA	Constantes
2	Palabras internas	%MW23	Palabras digitales internas del PLC
3	Palabras constantes	%KW4	Palabras constantes definidas por el programador
4	Palabras de bloques funcionales	%TMi.P (2) %Ci.V	Palabras de los temporizadores, contadores, programadores, etc.
5	Palabras de entrada	%IW0.0	Palabras digitales de entrada al PLC
6	Palabras de salida	%QW0.1	Palabras digitales de salida del PLC

(1) Pueden estar expresados en base decimal o hexadecimal.
(2) La letra i corresponde a la identificación numérica del bloque funcional.

Tabla 9 Instrucciones con palabras. (1)

Instrucción	Ejemplo	Operando	Descripción
Asignación	[Op1 := Op2]	Op1: Ítems 2, 4 y 6 Op2: Ítems 1, 2, 3, 4 y 6	Asigna a Op1 el valor de Op2
Comparación	[Op1 > Op2] [Op1 = Op2] [Op1 < Op2] [Op1 >= Op2] [Op1 <= Op2] [Op1<> Op2]	Op1 y Op2: Ítems 1, 2, 3, 4, 5 y 6	El resultado es 1 si la comparación es verdadera y es 0 en caso contrario
Suma	[Op1 := Op2 + Op3]	Op1: Ítems 2, 4 y 6 Op2 y Op3: Ítems 1, 2, 3, 4, 5 y 6	Asigna a Op1 la suma de Op2 y Op3
Resta	[Op1 := Op2 – Op3]	Op1: Ítems 2, 4 y 6 Op2 y Op3: Ítems 1, 2, 3, 4, 5 y 6	Asigna a Op1 la diferencia de Op2 y Op3
Multiplicación	[Op1 := Op2 * Op3]	Op1: Ítems 2, 4 y 6 Op2 y Op3: Ítems 1, 2, 3, 4, 5 y 6	Asigna a Op1 el producto de Op2 y Op3
División	[Op1 := Op2 / Op3]	Op1: Ítems 2, 4 y 6 Op2 y Op3: Ítems 1, 2, 3, 4, 5 y 6	Asigna a Op1 el cociente de Op2 dividido por Op3
Residuo	[Op1 := Op2 REM Op3]	Op1: Ítems 2, 4 y 6 Op2 y Op3: Ítems 1, 2, 3, 4, 5 y 6	Asigna a Op1 el residuo resultante de la división de Op2 entre Op3
Raíz cuadrada	[Op1 := SQRT(Op2)]	Op1: Ítems 2, 4 y 6 Op2: Ítems 1, 2, 3, 4, 5 y 6	Asigna a Op1 la raíz Cuadrada de Op2
Incrementar	[INC(Op1)]	Ítems 2, 4 y 6	Incrementa Op1 en una unidad
Disminuir	[DEC(Op1)]	Ítems 2, 4 y 6	Disminuye Op1 en una unidad
AND (2)	[Op1 := Op2 AND Op3]	Op1: Ítems 2, 4 y 6 Op2 y Op3: Ítems 1, 2, 3, 4, 5 y 6	Hace un AND bit a bit entre Op2 y Op3 y el resultado lo asigna a Op1
OR (2)	[Op1 := Op2 OR Op3]	Op1: Ítems 2, 4 y 6 Op2 y Op3: Ítems 1, 2, 3, 4, 5 y 6	Hace un OR bit a bit entre Op2 y Op3 y el resultado lo asigna a Op1
XOR (2)	[Op1 := Op2 XOR Op3]	Op1: Ítems 2, 4 y 6 Op2 y Op3: Ítems 1, 2, 3, 4, 5 y 6	Hace un XOR bit a bit entre Op2 y Op3 y el resultado lo asigna a Op1
NOT (2)	[Op1 := NOT(Op2)]	Op1: Ítems 2, 4 y 6 Op2: Ítems 1, 2, 3, 4, 5 y 6	Complementa cada bit de Op2. El resultado lo asigna a Op1
SHL	[Op1 := SHL(Op2,i)] (3)	Op1: Ítems 2, 4 y 6 Op2: Ítems 1, 2, 3, 4, 5 y 6	Desplaza i bits de Op2 hacia la izquierda. El resultado se asigna a Op2
SHR	[Op1 := SHR(Op2,i)] (3)	Op1: Ítems 2, 4 y 6 Op2: Ítems 1, 2, 3, 4, 5 y 6	Desplaza i bits de Op2 hacia la derecha. El resultado se asigna a Op2
ROL	[Op1 := ROL(Op2,i)] (3)	Op1: Ítems 2, 4 y 6 Op2: Ítems 1, 2, 3, 4, 5 y 6	Desplaza i bits de Op2 hacia la izquierda, pero en redondo. El resultado se asigna a Op2
ROR	[Op1 := ROR(Op2,i)] (3)	Op1: Ítems 2, 4 y 6 Op2: Ítems 1, 2, 3, 4, 5 y 6	Desplaza i bits de Op2 hacia la derecha, pero en redondo. El resultado se asigna a Op2

(1) Op1, Op2 y Op3 se refieren a Operando 1, Operando 2 y Operando 3, respectivamente.
(2) Las operaciones lógicas sobre palabras digitales se ejecutan bit a bit.
(3) La letra i representa la cantidad de bits a desplazar.

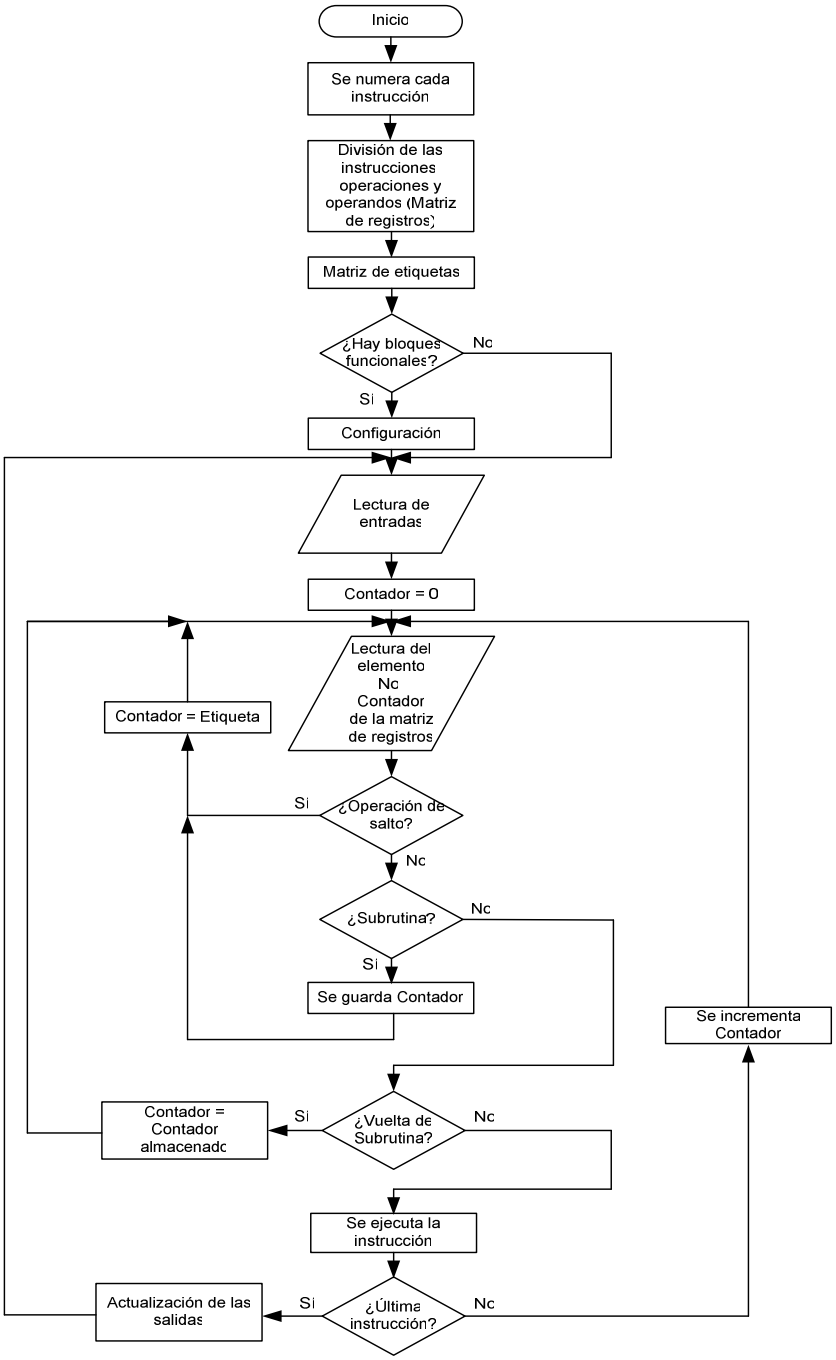

Figura 1. Diagrama de flujo del algoritmo.

7. ALGORITMO BÁSICO DE SIMULACIÓN

1. Al cargar el archivo .TXT, automáticamente se coloca un número delante de cada instrucción y se habilita la opción de Configurar en la barra de menús.
2. Haciendo clic en Configurar, se procede a formar una matriz unidimensional de records o registros, donde cada registro contiene campos diferentes para las operaciones, signos, funciones y operandos que puede contener una instrucción. El índice

de la matriz se corresponde con el número asignado a la instrucción.

3. Seguidamente, se forman matrices con las etiquetas que identifican instrucciones relacionadas con saltos y subrutinas.
4. A continuación, se verifica la presencia de bloques funcionales dentro de la matriz de registros, para proceder con el paso c) de la sección 6.
5. Después de completado el paso de configuración, al hacer clic en Simular y escoger la opción apropiada, se leen los valores de las entradas, se inicializa el contador del programa (que corresponde al número identificador de cada instrucción), se leen todos los campos de ese elemento de la matriz de registros, se decodifica (n) la (s) operación (ones) y operando (s) involucrado (s), se ejecuta lo especificado en la instrucción, se incrementa el contador del programa y se repite lo anterior. Por supuesto, si existen saltos o llamadas a subrutinas, se ejecutarán estas instrucciones según lo establecido en el programa de aplicación. Cuando el contador del programa supera la última instrucción, se procede a actualizar las salidas.
6. Mientras no se detenga la simulación, se repite el paso 5.

En la Figura 1 se muestra el diagrama de flujo correspondiente a este algoritmo.

Es necesario aclarar que, si bien el algoritmo básico del simulador no es complicado, la programación del mismo sí lo es, debido a la gran cantidad de instrucciones y operandos diferentes con que puede trabajar el autómata elegido como modelo.

8. RESULTADOS DE LA INVESTIGACIÓN

El simulador fue desarrollado con el lenguaje Object Pascal, utilizando el ambiente Delphi 6, siendo el resultado totalmente satisfactorio. Se trata de un programa con más de quince mil líneas de programación, lo cual da idea de la complejidad de su desarrollo. El mismo fue probado exhaustivamente, ensayando cada tipo de instrucción por separado. En las Figuras 2 a la 9 se muestran las principales ventanas del simulador, las cuales permiten una cómoda interacción con el usuario.

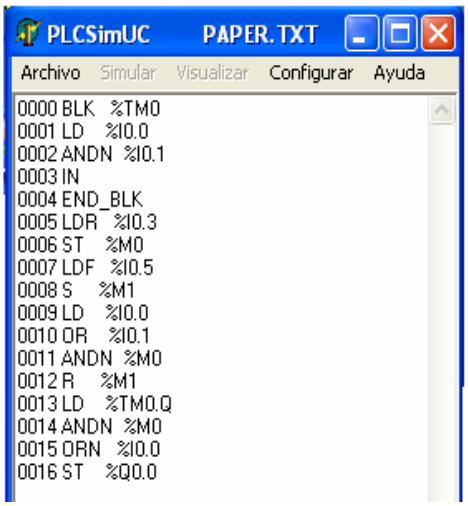

Figura 2. Ventana principal.

Figura 3. Ventana de simulación (Entradas / Salidas).

Figura 4. Ventana de visualización.

Figura 5. Ventana para la configuración de temporizadores.

Figura 6. Ventana para la configuración de contadores.

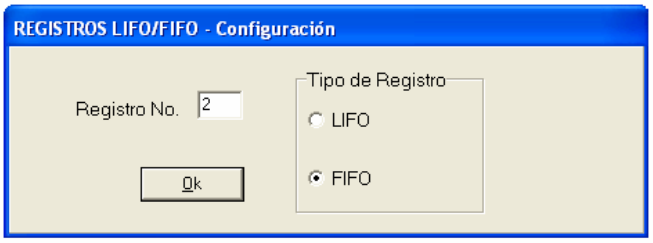

Figura 7. Ventana para la configuración de registros LIFO/FIFO.

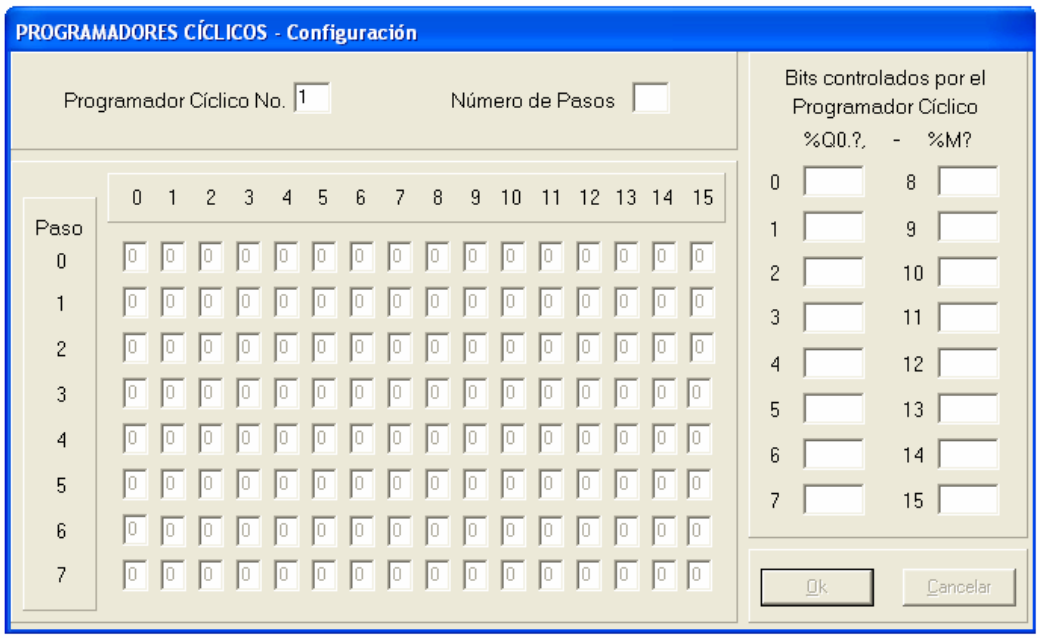

Figura 8. Ventana para la configuración de programadores cíclicos.

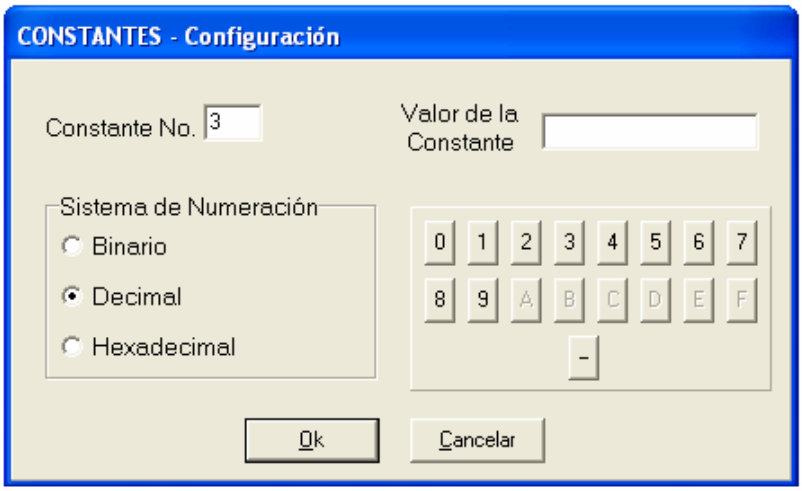

Figura 9. Ventana para la configuración de constantes.

9. CONCLUSIONES

En este trabajo se ha presentado el algoritmo general para la elaboración de un simulador de programas de aplicación para autómatas que hayan sido realizado con un lenguaje de lista de instrucciones, compatible con la norma IEC 61131-3 (1993). El mismo ha sido aplicado a un caso particular, correspondiente a un amplio conjunto de instrucciones.

La investigación puede ser utilizada como base para el desarrollo de simuladores de la actuación en red de varios autómatas, tanto con fines didácticos como de ensayo de aplicaciones industriales.

REFERENCIAS

[1] Ramón Piedrafita. Ingeniería de la Automatización Industrial. Alfaomega Grupo Editor, Méjico, 2001.

[2] Joseph Balcells y José Romeral. Autómatas Programables. Alfaomega Grupo Editor, Méjico, 1998.

[3] Emilio García. Automatización de Procesos Industriales. Alfaomega Grupo Editor, Méjico, 2001.

[4] Borland Software Corporation, Borland Delphi 6 for Windows Quick start, California, EE.UU

[5] Cantú, Marco, La Biblia de Delphi 6. Ediciones Amaya, Madrid, España, 2002.

[6] René, David y Hassane, Allá. Du Grafcet aux Receaux de Petri. Ediciones Hermes, Paris, Francia, 1997.

[7] Geller, David. Programmable Controllers using the Allen-Bradley SLC-500 Family. Prentice Hall, New Jersey, EE.UU. 2000

[8] Groover Mikell, Automation, Production Systems, and Computer Integrated Manufacturing, 2da Edición, Prentice-Hall, New Jersey, EE.UU. 2001

[9] Stenerson, Jon, Fundamentals of Programmable Logic Controllers, Sensors and Communications, 2da edición, Prentice Hall, New Jersey, EE.UU. 1999.

[10] Porras, A. Montanero, A.P., Autómatas Programables, McGraw Hill