DESCRIPCIÓN E IMPLEMENTACIÓN DE SFC PARA SIMULACIÓN Y EJECUCIÓN DE AUTOMATISMOS EN LENGUAJE DE PROGRAMACIÓN GRÁFICA

Description and implementation of SFC for simulation and execution of automatisms in graphical programming language

RESUMEN

En la implementación de automatismos, cada vez es más común tener que integrar técnicas de modelamiento de sistemas de eventos discretos con lenguajes de programación orientados a PC. Dentro de las técnicas de mayor prestación en automatización se encuentra SFC (Sequential functional chart), la cual gracias a funcionalidades tales como descripción en forma de eventos discretos, acciones de forzado, control de acciones concurrentes y divergentes se ha convertido en estándar. En el presente artículo se presenta una descripción funcional desde el punto de vista de las ecuaciones que describen la operación de SFC tal que permitan su posterior implementación en lenguajes de programación por medio de algoritmos pertinentes. Se pretende fusionar el poder de SFC con lenguajes de programación gráfica que permitan pasar del modelo a la implementación física en pocos pasos y la menor adición posible de código.

PALABRAS CLAVES: Automatismos, eventos discretos, GEMMA, SFC, programación gráfica.

ABSTRACT

In the implementation of automation, it is increasingly common to have to integrate the modeling techniques for discrete event systems with PC programming languages. Among the techniques to provide greater automation is SFC (Sequential functional chart), which has been a standard thanks to features such as the description in the form of discrete events, forced control, concurrent and divergent actions. The present article shows a functional description from the viewpoint of the equations that describe the operation of the SFC to its implementation in programming languages by means of appropriate algorithms. It is intended to merge the power of SFC with graphical programming language to pass the model to physical implementation in a few steps and adding the smallest possible code.

KEYWORDS: *Automatisms, discrete events, GEMMA, SFC, graphical programming language.*

ÁLVARO ANGEL OROZCO GUTIÉRREZ
Doctor en Bioingeniería
Directos Grupo de Investigación en Control e Instrumentación
Profesor Titular
Universidad Tecnológica de Pereira
aaog@utp.edu.co

MAURICIO HOLGUÍN LONDOÑO
Ingeniero Electricista
Candidato a Máster en Ing. Eléctrica
Profesor Transitorio
Universidad Tecnológica de Pereira
ma_hol@ohm.utp.edu.co

EDUARDO GIRALDO SUÁREZ
Máster en Ingeniería Eléctrica
Profesor Auxiliar
Universidad Tecnológica de Pereira
egiraldos@ohm.utp.edu.co

1. INTRODUCCIÓN

Un automatismo es el sistema de control de cualquier tipo de proceso, cuyo objetivo es su ejecución con la mínima intervención humana [1, 2]. En la actualidad se utilizan varias formas de realizar control sobre procesos, las que pueden estar basadas en lógica binaria o programación mediante comandos, y cada una de ellas presenta ciertas ventajas y desventajas con respecto una hacia la otra [2, 3].

Las limitaciones de los lenguajes de control para realizar tareas robustas han sido el principal problema en el diseño de automatismos. Un lenguaje tradicional como lógica cableada, que describe sistemas con base en dispositivos electromecánicos, se ve incapacitado para realizar tareas simultáneas, presentando además problemas como imposibilidad de realizar cambios a las variables, dificultad en la adaptabilidad y mantenibilidad de un diseño, entre otros. El lenguaje SFC se introdujo con el fin de solucionar todos los problemas mencionados, entregando la posibilidad de implementar un sistema mediante una descripción por eventos discretos [1, 2, 3, 4].

SFC cuenta con características enriquecidas que permiten diferentes posibilidades y la implementación de tareas robustas. Entre estas características se tienen: ejecución de tareas concurrentes, ejecución de tareas divergentes, control de acciones, ejecución de acciones de forzado, implementación de automatismos según norma GEMMA [5, 6], etc.

Actualmente, SFC es implementado por una gran variedad de sistemas de desarrollo todos con orientación PLC, sin embargo existe dificultad para su desarrollo en

Fecha de Recepción: 8 de julio de 2009
Fecha de Aceptación: 13 de Agosto de 2009

lenguajes tradicionales, donde la descripción de estos sistemas implica realizar modelos en técnicas de traducción que hagan el enlace entre los eventos discretos y la descripción por flujo.

Aprovechando el hecho que SFC fundamenta su operación en lógica secuencial [1, 3], la cual se puede implementar en un lenguaje de programación común, se parte de la necesidad de tener su descripción total comportamental y evolutiva con el fin de desarrollar los algoritmos necesarios para su ejecución directa en lenguajes de programación [7].

Actualmente existen lenguajes de programación gráfica (como LabVIEW™ o Matlab-Simulink™) que simulan dispositivos de lógica combinacional y a su vez tienen control en el procesamiento de datos con la posibilidad de hacer seguimiento en cada proceso de la simulación, características que son importantes para el desarrollo de SFC, gracias a esto se opta por hacer la implementación en lenguaje de programación gráfica [8].

LabVIEW™ servirá de plataforma para el diseño de un código gráfico que simule SFC y sus características enriquecidas. Al trabajar con LabVIEW™, se logra pasar de la simulación a la implementación de forma directa [9], ya que después de comprobar el buen funcionamiento de algún diseño, sólo será necesario agregar una pequeña parte de código para la comunicación con algún tipo de hardware.

2. CONTENIDO

2.1 Modulo Secuenciador de Etapa

Es el elemento tecnológico funcional en SFC capaz de interaccionar con sus etapas anterior y posterior [3]. Su descripción se muestra en la Figura 1.

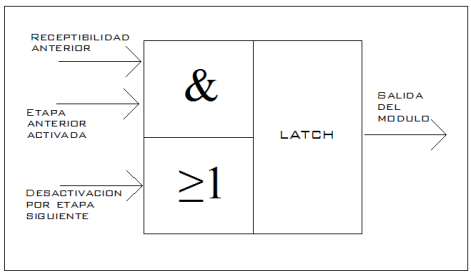

Figura 1. Modulo Secuenciador de Etapa

El módulo secuenciador de etapa posee tres entradas y una salida; y el objeto de cada una de ellas es el siguiente [3]:

- Receptibilidad anterior: Se utiliza para efectuar, junto con la señal de activación de la etapa previa, la activación de la etapa y proviene del disparo de la transición de entrada al módulo.
- Etapa anterior activada: Se utiliza para efectuar la validación del módulo presente, proveniente de la salida del módulos o módulos previos.
- Desactivación por etapa siguiente: Se utiliza para efectuar la desactivación del módulo presente y la señal proviene de la salida realimentada desde el módulo o módulos posteriores.
- Latch: Realiza tres funciones distintas y simultáneas que son: Validar el módulo o módulos posteriores al presente, desactivar el módulo o módulos previos y ejecutar algún tipo de acción prevista. Es la salida del módulo.

Una etapa de SFC puede ser implementada por medio de este dispositivo que contiene elementos de lógica combinacional y un elemento biestable (lógica secuencial). Las entradas se componen de un par de compuertas, AND (&) para la condición de activación y OR (≥1) para la condición de desactivación [3, 10, 11]. El biestable de tipo Latch funciona como una memoria que retiene el estado que se valida en algún instante de funcionamiento.

2.2. Ecuaciones de Activación/Desactivación

Básicamente, existen dos tipos de ecuaciones asociadas al concepto de memoria binaria, que se denominan:

- Ecuación de memoria binaria de set o enclavamiento prioritario, cuya expresión es:

$$E_{(t+\Delta t)} = S + \overline{R} * E_{(t)} \qquad (2.1)$$

- Ecuación de memoria binaria de reset o disparo prioritario, cuya expresión es:

$$E_{(t+\Delta t)} = \overline{R} * (S + E_{(t)}) \qquad (2.2)$$

Para un módulo secuenciador de reset prioritario y con condiciones de activación y desactivación se puede escribir:

$$En_{(t+\Delta t)} = Cd * (Ca + En_{(t)}) \qquad (2.3)$$

Donde, para las tres expresiones anteriores:

$E_{(t)}$ = Estado actual de la etapa.

$E_{(t+\Delta t)} = En_{(t+\Delta t)}$ = Estado siguiente de la etapa.

$S = Ca$ = Condición de activación de la etapa.

$\overline{R} = Cd$ = Condición de desactivación de la etapa.

Las expresiones concretas de *S* y *R* dependerán de las estructuras lógicas específicas resultantes del modelo realizado; o más concretamente, dependerá de la estructura básica y lógica del diagrama SFC. En general serán funciones OR y AND donde intervienen las variables representativas de etapas y transiciones implicadas [3].

La implementación del módulo secuenciador se hace por medio de un biestable asíncrono, Latch, con enclavamiento prioritario reset, debido a que la prioridad de una orden de forzado impide que la red evolucione. En la Figura 2 se describe el módulo secuenciador con reset prioritario.

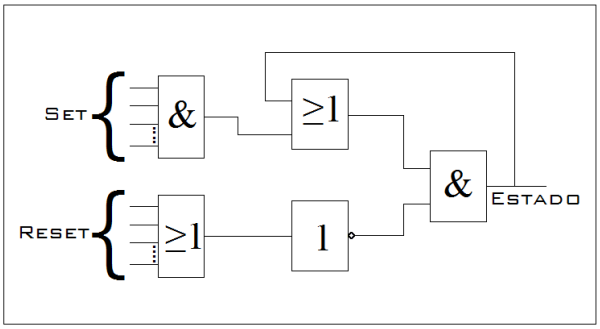

Figura 2. Modulo Secuenciador con Reset Prioritario.

Para obtener la función general de activación/desactivación se debe de tener en cuenta la acción de forzado, para ésta es necesario obtener cuatro datos:

- Red a ser forzada: $R_{Forzado}$
- Red que genera el forzado: $RG_{Forzado}$
- Etapa que genera el forzado: $EG_{Forzado}$
- Etapa a ser forzada: $E_{Forzado}$

Las variables $RG_{Forzado}$ y $EG_{Forzado}$ son las encargadas de realizar asignación a las otras dos variables de forma respectiva, cuando éstas tienen un valor booleano asertivo, las otras dos variables que son de tipo entero pasarán de tener un valor por defecto de cero a tener los números correspondientes de las redes y etapas a las que se les aplica la acción de forzado.

Sintetizando estos datos se obtiene:

Si $R_{Forzado} = Rn$ y $E_{Forzado} = En$

Entonces $F_{\mathrm{Red}} = 1$ y $F_{Etapa} = 1$

De lo contrario $F_{\mathrm{Red}} = 0$ y $F_{Etapa} = 0$

Donde:

$Rn =$ Número de red propia.
$En =$ Número de etapa propia.

Con los datos anteriores, se puede construir la ecuación general de estado activación/desactivación del módulo, según la ecuación (2.3), de la siguiente forma:

$$En_{(t+\Delta t)} = Cd * (F_{Etapa} + \overline{F_{\mathrm{Red}}}) * (P_{inicio} + F_{Etapa} * F_{\mathrm{Red}} + Ca + En_{(t)}) \quad (2.4)$$

Donde:

F_{Red} = Condición de forzado hacia una red.
F_{Etapa} = Condición de forzado hacia etapa de una red.
P_{inicio} = Condición inicial.
Ca = Condición de activación.
Cd = Condición de desactivación.

3. ALGORITMO ACTIVACIÓN/DESACTIVACIÓN

El funcionamiento de una etapa se basa en la implementación de las funciones de activación/desactivación. El procesamiento de los datos de entrada, salida y realimentación permite generar las variables de control de estado de las etapas, ahora bien, a partir de la función general de estado de la etapa con reset prioritario se pueden diseñar algoritmos que describan las condiciones de estado de las etapas en un lenguaje de programación gráfica.

El algoritmo general se emplea para construir las acciones de activación/desactivación y la de forzado, en el lenguaje de programación gráfico. El algoritmo se muestra en la Figura 3, donde:

E_{n+1}: representa la etapa siguiente
E_{n-1}: representa la etapa anterior
T_{n-1}: representa la transición previa

El algoritmo propuesto puede ser implementado en cualquier tipo de lenguaje de programación, ya sea gráfico o basado en texto, sólo se requiere ajustar los tipos de variables dependiendo del entorno de programación empleado, ya que la condición de activación o desactivación de la ecuación (2.4) es independiente del contexto.

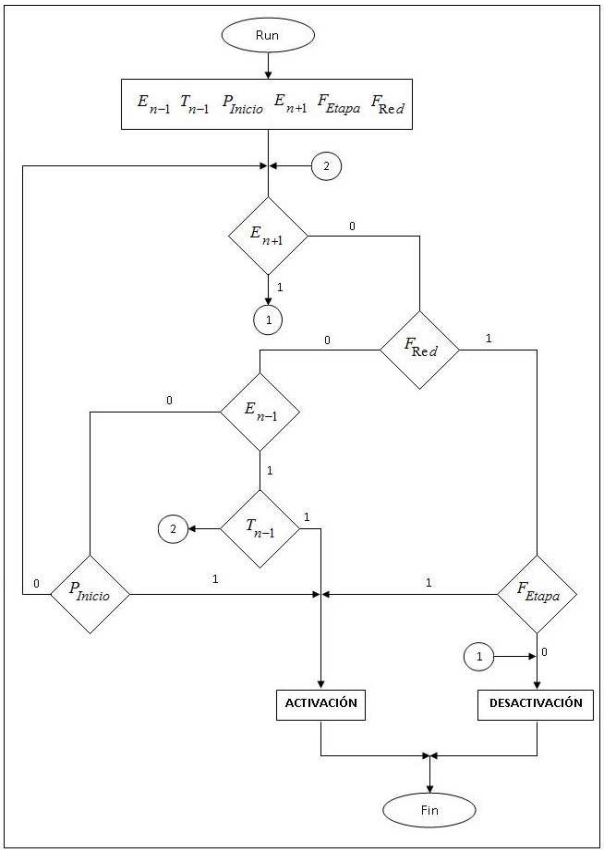

Figura 3. Algoritmo general de activación/desactivación.

4. RESULTADOS

El resultado final es un conjunto de herramientas en LabVIEW™ para la implementación de sistemas de eventos discretos, el cual contiene los diferentes elementos de SFC y que se ha denominado SFC Kit. La simulación de SFC se logra por medio de un paquete de VIs que se instalan dentro del menú de programación en el diagrama de bloques de LabVIEW™ y en el cual se tiene acceso a todas las posibles estructuras, controles de acciones y acciones de forzado de un modelo SFC.

4.1 Paleta de VIs de SFC Kit

Dentro del ambiente LabVIEW™, se encuentra el menú principal, que consta de submenús que a su vez contienen los elementos constitutivos de una red SFC. En la Figura 4 se muestra el menú principal con sus submenús.

En cada uno de los submenús se encuentran VIs encargados de funcionalidades específicas de una red SFC tales como: Etapas normales e Etapas iniciales; Divergencias AND y OR, Convergencias AND y OR; calificadores básicos de acciones [7] y un VI especial de control necesario para la coordinación y manipulación de información entre cada una de las redes que conforman un sistema de control de eventos discretos. Las transiciones se encuentran incluidas en el interior de las etapas, con el fin de obtener una representación esquemática más compacta.

Figura 4. Menú principal de SFC Kit en LabVIEW™.

4.2 Ejemplo de Aplicación

A modo de ejemplo de aplicación, se muestra la implementación de la secuencia A→BC→D controlada por un pulsador monoestable T. Su funcionamiento básico se describe a continuación:

- La red inicia con las acciones sin ejecutarse.
- Se oprime T y se enciende A hasta que se libere nuevamente T.
- Cuando se presione nuevamente T se encienden B y C hasta que se libere T.
- Cuando se presione nuevamente T se enciende D hasta que se libere T.
- La secuencia se repite.

Implementación:

- Se realiza el diseño en SFC para la secuencia propuesta, ver Figura 5.

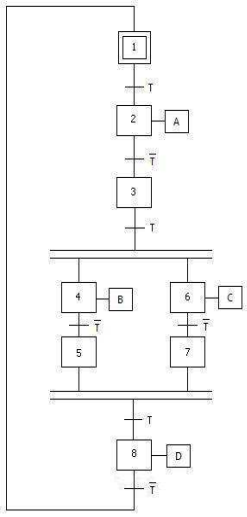

Figura 5. Diseño Secuencia A-BC-D

- Se procede a implementar el modelo diseñado con la asistencia de SFC Kit, ver Figura 6.

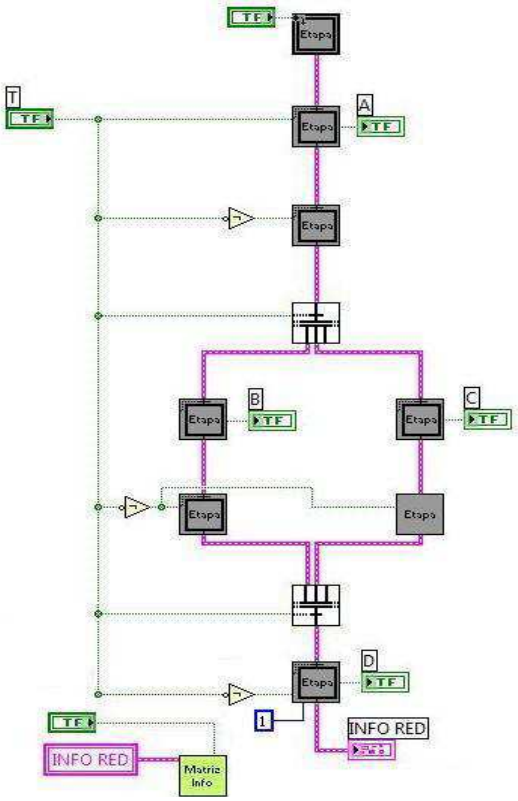

Figura 6. Secuencia A→BC→D

- De lo anterior se obtiene el siguiente resultado en el panel frontal, ver Figura 7.

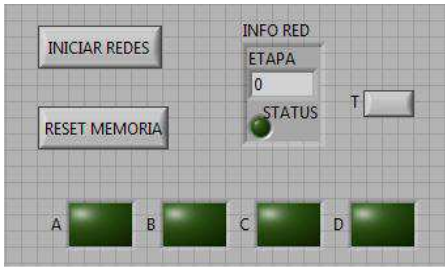

Figura 7. Secuencia A-BC-D Panel Frontal.

Ahora se modifica el automatismo básico con el fin de permitir que la activación de la carga A dure sólo 5s, que la carga C retrase su ingreso por 3s y que la carga B se active sólo por 2 segundos, tal como se muestra en la Figura 8. Estas acciones corresponden a calificadores básicos de acción.

Finalmente, se adicionan redes nuevas que implementan acciones de forzado correspondientes a un botón que retiene o congela el automatismo en su estado actual (Red G2) y la acción de forzado para un paro de emergencia (Red G3), tal como se observa en la Figura 9.

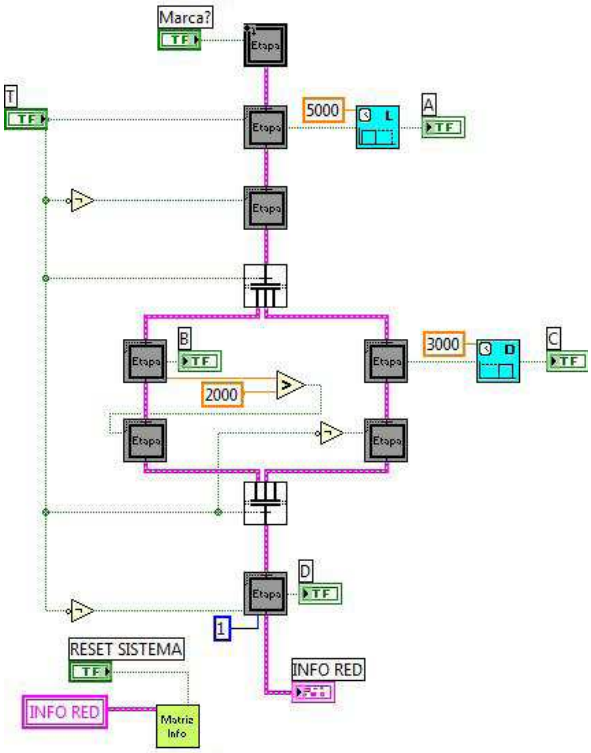

Figura 8. Empleo de calificadores de acción.

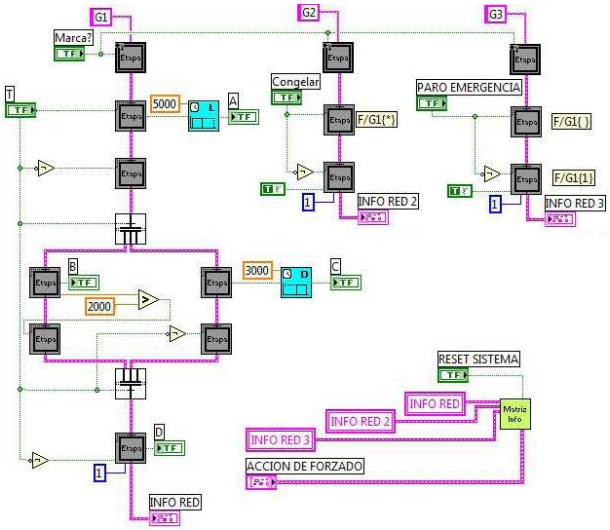

Figura 9. Empleo de acciones de forzado.

La adición de los calificadores de acción y de las acciones de forzado de las Figuras 8 y 9, es un procedimiento directo y con una topología prácticamente igual a como se realiza en implementaciones de PLC que usan *Sequential Functional Chart*.

El panel frontal, con los controles e indicadores recién agregados, se muestra en la Figura 10.

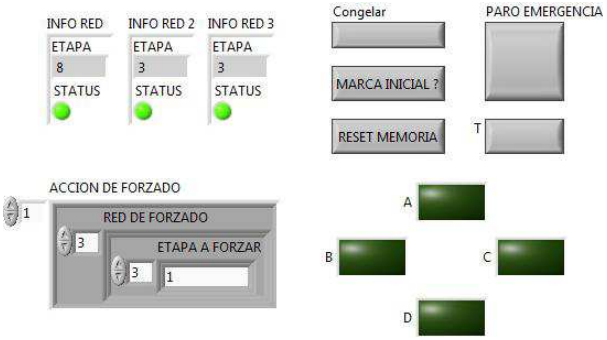

Figura 10. Panel frontal final.

5. CONCLUSIONES Y RECOMENDACIONES

Con la implementación del módulo secuenciador de etapa, que es el dispositivo básico de una red SFC, se puede llegar a la implementación del estándar de programación SFC en cualquier tipo de plataforma, ya sea en hardware o software.

Emplear LabVIEW™ como plataforma de programación para la implementación de SFC, ofrece grandes ventajas, ya que LabVIEW se caracteriza por ser una herramienta útil para construir sistemas de adquisición y control de forma más sencilla y el diseño de un sistema de control sólo requiere de una pequeña parte de código para realizar la interfaz entre software y hardware.

SFC permite describir paso a paso, el funcionamiento de un autómata, haciendo que éste sea altamente coherente y riguroso, permitiendo además su mantenibilidad en todo instante sin afectar otras partes del proceso [7]. SFC puede ser considerado como un lenguaje síncrono, que obedece a la filosofía del flujo de datos (Data Flow) que también es utilizada en LabVIEW™, haciendo que su unión sea exitosa e imponente.

El paquete resultado de este proceso, denominado SFC Kit, es una poderosa herramienta que permite la simulación e implementación de sistemas de eventos discretos al poder pasar desde un simple modelo SFC directamente a la simulación en el lenguaje gráfico, y desde éste a la aplicación real con la adición de muy poco código.

6. AGRADECIMIENTOS

Los autores agradecen a los estudiantes Jhon Fredy Castañeda Londoño y Cesar Augusto Castañeda Ramírez del programa de Ingeniería Eléctrica de la Universidad Tecnológica de Pereira por toda su colaboración en la elaboración de algoritmos y durante el proceso de validación de la herramienta SFC Kit.

7. BIBLIOGRAFÍA

[1] OROZCO, Álvaro. GUARNIZO, Cristian. HOLGUÍN, Mauricio. Automatismos Industriales. 1ed. PEREIRA: Taller de Publicaciones-Universidad Tecnológica de Pereira, 2008. ISBN: 978-958-8272-99-3

[2] BALCELLS, Joseph. ROMERAL, José Luis. Autómatas Programables. México: Alfaomega grupo editorial, 1998. ISBN 9701502477.

[3] GARCÍA MORENO, Emilio. Automatización de Procesos Industriales. México: Alfaomega grupo editorial, 2001. ISBN 9701506588.

[4] MARTÍNEZ, José L. TUOKKO, Reijo. Sistemas De Control Basados en Tecnología PC, Estado Del Arte y Tendencias a Corto Plazo. Revista: Automática E Instrumentación No. 299 [1999].

[5] LE PARC, Philippe. L'HER, Dominique. SCHARBARG, Jean-Luc. MARCÉ, Lionel. Grafcet Revisited with a Synchronous Data-Flow Language. IEEE Transactions On Systems, Man, And Cybernetics Part A: Systems And Humans, Vol. 29 No. 3 [1999] Pag. 284.

[6] RODRÍGUEZ, José A. Iniciación A La Guía GEMMA. Escuela Técnica Superior De Ingeniería ICAI (UPCO). Madrid.

[7] International Electrotechnical Commission SC65B/WG7/TF3 Committee Draft – IEC 61131-3, 2nd Ed. "Programmable Controllers - programming languages" Committee Draft, Houston, 10/1998.

[8] García, E. Morant F. Salt J. Correcher, A. Herramientas de modelado para el diseño jerárquico de sistemas de automatización industrial. ESPAÑA: Departamento de Ingeniería de Sistemas y Automática, Universidad Politécnica de Valencia.

[9] National Instrument Corporation. LabVIEW™ User Manual: LV_Fundamentals, LV_Getting_Started.

[10] TOCCI, Ronald J. WIDMER, Neal S. Sistemas Digitales: Principios y aplicaciones. Colombia: PRENTICE-HALL INC., 1977.

[11] WAKERLY, John F. Diseño Digital: Principios y prácticas. 3ed. México: Pearson Educacion de México, 2001. ISBN 9702607205.