ScienceDirect
Disponible en www.sciencedirect.com
Revista Iberoamericana de Automática e Informática industrial 13 (2016) 322–329

Metodología para la elaboración de los programas a implementar en autómatas programables. MEPUS.

José M. Diez, Rafael Montoya*, Pedro A. Blasco

Universidad Politécnica de Valencia, Campus de Alcoy. Pl/ Ferrandiz i Carbonell, 2, 03801 Alcoy, Alicante, España.

Resumen

El objetivo de este trabajo es la presentación de una metodología para la elaboración del programa de usuario a la que hemos llamado MEPUS (Metodología para Estructurar el Programa de Usuario mediante Saltos). Con esta metodología se eliminan las aleatoriedades de programación, a la vez que se reduce de forma considerable el tiempo del ciclo de lectura o ciclo de scan, sobre todo en los procesos secuenciales. No es un nuevo método de diseño de automatismos programados. Es una herramienta gráfica para la redacción del programa de usuario, independiente del método de diseño utilizado para representar el modelo de secuencialización del proceso a automatizar, ya sea red de Petri, Grafcet, Teoría Binodal, etc. También, y aunque sea una ventaja menor, se optimizan los recursos internos del autómata (temporizadores, contadores, etc.), ya que esta metodología permite que se puedan volver a utilizar en un mismo programa. Basado en los resultados obtenidos durante más de 10 años, se concluye que con esta metodología el programa de usuario se obtiene de forma más sencilla, clara y sobre todo sistemática, lo que facilita el aprendizaje por parte del colectivo profesional y discente. Para este trabajo se ha elegido como modelo de secuencialización del proceso a automatizar el Grafcet, por ser el más utilizado por los programadores de PLC's.

Palabras Clave: Automática, Control, Metodología, Autómatas Programables, Programas, Aleatoriedades.

1. Introducción

Una vez elaboradas las condiciones de funcionamiento del proceso a automatizar, el diseñador se enfrenta a la elección de la tecnología a utilizar. En función de dicha tecnología se obtienen los esquemas (tecnologías cableadas) o programas (tecnologías programadas) que han de representar el funcionamiento deseado. En el caso de tecnologías programadas la forma de obtenerlos es muy variada, en efecto:

1) La mayoría de ingenieros y diseñadores deducen el esquema o programa de control mediante métodos puramente intuitivos, basados en la experiencia adquirida a lo largo de los años (Moreno y Peulot, 1996). Este procedimiento presenta dos grandes inconvenientes: el elevado tiempo de depuración y la utilización irracional de recursos.
2) El resto de ingenieros, diseñadores y grupos docentes, utilizan herramientas gráficas de diseño como redes de Petri, teoría binodal, grafcet, programación dirigida a objetos, etc. (IEC 60848, IEC 61131-3, IEC 61499 entre otros), la teoría de sistemas dinámicos de eventos discretos (Cassandras and Lafortune, 2006; Hrúz and Zhou, 2007; Daniel Gómez et al., 2011).

* Autor en correspondencia.
Correos electrónicos: rmontoya@die.upv.es (Rafael Montoya); jmdiez@die.upv.es (José M. Díez); pedblaes@die.upv.es (Pedro A. Blasco)

Sea cual sea el método de diseño utilizado, obtenemos ecuaciones o frases que constituyen líneas o comandos de programación, que dan lugar al programa de usuario a implementar en el PLC (Quezada-Quezada et al., 2014; Javier de las Morenas et al, 2015).

No obstante ningún método de los anteriores se indica en qué orden se deben escribir las líneas de programación. Ello puede dar lugar a la aparición de las llamadas aleatoriedades o ambigüedades de programación (Silva, 1985; Zaytoon et al., 1997; Daniel Gomez et al, 2008; F. Schumacher, 2011), debido al funcionamiento particular de los PLCs.

Su funcionamiento tan característico hace que el orden de situación de las instrucciones o líneas del programa sea un factor crucial para garantizar el funcionamiento deseado (Pérez et al., 1998). Por lo tanto, será necesario adaptar el modelo obtenido al funcionamiento de dichos dispositivos (Lhoste et al., 1997).

En efecto, aún siendo las mismas ecuaciones, basta con modificar su orden de ubicación en el seno del programa para que éste no funcione de forma satisfactoria. Las deficiencias detectadas son entre otras, la activación múltiple de etapas, deslizamientos, bloqueos parciales, etc. (Silva, 1985; Zaytoon et al., 1997; Frank Schumacher, 2011).

Por su amplia utilización, sobre todo en Europa, en este trabajo como herramienta de diseño se utiliza el grafcet. Este método apareció en 1977 y posteriormente fue establecido como norma IEC 60848 (International Electrotechnical Commission) en 1988.

http://dx.doi.org/10.1016/j.riai.2016.03.006

Han sido numerosos los trabajos que tratan la problemática "grafcet - autómatas programables". La forma de corregir estas deficiencias se agrupan básicamente en dos líneas de trabajo: en primer lugar, aquellas que no modifican el modelo diseñado y cuyo objetivo es encontrar un orden de ubicación adecuado y, en segundo lugar, aquellas que modifican el modelo de grafcet diseñado para adaptarlo al dispositivo.

Las primeras requieren un gran tiempo de depuración y en determinadas ocasiones el orden obtenido no es del todo fiable, siendo necesaria una modificación sobre el propio modelo. Las segundas, suponiendo que sean posibles, destruyen gran parte de la potencia descriptiva que tiene el Grafcet, obteniendo modelos definitivos que dificultan analizar a través de ellos el funcionamiento del proceso.

Esta serie de problemas se evitaría fácilmente si en un cierto ciclo de lectura, sólo se procesara la parte del programa que le corresponde evolucionar en ese momento, según el modelo grafcet diseñado.

Mediante la metodología, denominada MEPUS "Metodología de Estructuración del Programa de Usuario mediante Saltos" (Diez, 2000), se consigue de forma sistemática y sencilla, mediante la utilización de instrucciones u operaciones de salto, elaborar el programa de usuario en el PLC directamente, a partir de las ecuaciones obtenidas de los grafcets.

En este trabajo se presenta la metodología MEPUS después de su aplicación durante más de 10 años. Durante este tiempo se ha implementado en todo tipo de casos reales. Además, también se ha impartido en docencia durante este tiempo, observando que el grado de asimilación por parte de los alumnos ha crecido hasta niveles máximos.

Todo ello nos da garantía suficiente para decir que se ha conseguido un procedimiento sencillo, sistemático y validado que nos indica en que orden debemos escribir las líneas de programa para evitar las posibles aleatoriedades de programación.

2. Descripción general del método.

MEPUS consigue mediante la utilización de instrucciones de salto, realizar una lectura parcial del programa de usuario del PLC, de tal forma que en cada "ciclo de scan o de lectura" se procesa únicamente la parte o porción del programa que en dicho instante es necesaria, no leyendo el resto.

Pero, ¿cómo puede conseguirse que el programa se desarrolle de esta forma?. La respuesta es doble:

a) Utilizar la variable interna asociada a cada etapa o etapas, con objeto de discriminar la porción de lectura a la cual se debe acudir para preguntar sí se ha de pasar a la etapa o etapas siguientes.
b) Utilizar adecuadamente los saltos, tanto condicionales como absolutos, para ir leyendo las diferentes instrucciones de usuario.

En la Figura 1, se describe el planteamiento del método donde se distinguen: situaciones, condiciones, inicialización, porciones asociadas y porción de lectura permanente, las cuales se relacionan a través de instrucciones de salto. Para ayudar a relacionar estos nuevos conceptos se utilizan elementos gráficos.

Hay que destacar que las condiciones, porciones asociadas y porción de lectura permanente, se pueden estructurar de varias formas, aunque en este trabajo se presentan numeradas de forma consecutiva por columnas de izquierda a derecha, dejando para el final la porción denominada "Porción de Lectura Permanente" (PLP).

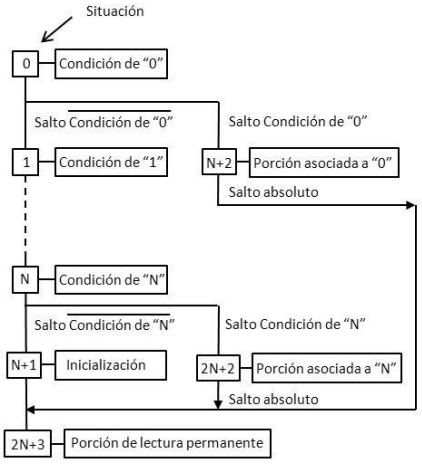

Figura 1: Esquema general del método.

El significado de la simbología representada en la Figura 1 es el siguiente:

- Situación.

Se representa mediante un cuadrado con dos segmentos (superior e inferior) tal como se indica en la Figura 2. "N" representa un número que se corresponde con el orden de ubicación de las líneas de programación asociadas a la etapa "N", en el seno del programa. En la Figura 1 se tienen las situaciones: 0, 1, N, N+1, N+2, 2N+2 y 2N+3. Es imprescindible respetar el orden de colocación establecido.

Figura 2: Anotación de una situación.

- Condición.

Una condición "CN" viene dada por el contenido de programa asociado a una situación "N", que al ser evaluado, determina si debe procederse o no a la lectura de su porción asociada "PN". Conviene destacar que el número de la condición CN se corresponde con la etapa del grafcet "N". Se representa según la Figura 3.

Dentro de esta condición se preguntará, en principio, por el estado activo de la variable interna que define la etapa del grafcet "N". Además se debe incluir las instrucciones de salto para indicar cuál es el siguiente grupo de instrucciones que ha de procesarse.

Figura 3: Anotación de una condición.

- Porción asociada a una condición.

Una porción "PN" la constituye una parte del programa de usuario que está relacionada con su condición asociada "CN". Esta parte debe leerse únicamente si se verifica dicha condición "CN". En caso contrario, aún formando parte del programa establecido, no podrá evaluarse, ya que no es el momento de proceder a su lectura. "X" representa el orden de ubicación del programa ligado a la porción "PN", en el seno del programa y que evidentemente tendrá un número de orden superior a "N". Se representa según la Figura 4.

El contenido relativo a una porción "PN" vendrá dado por, y en el orden que se expresan, los siguientes coceptos:

1) Receptividad de la transición correspondiente.
2) Activación de la etapa o etapas posteriores.
3) Desactivación de la etapa "N".
4) Eventos internos asociados a la etapa "N", por ejemplo, incrementar y decrementar valores, sumar y restar valores, comparación de valores, creación de variables temporizadas, de contaje, flancos, etc.
5) Además, de forma opcional, ya que puede incluirse en la porción de lectura permanente, la activación y desactivación de salidas y variables internas, ya se materialicen con instrucciones de memoria (Set y Reset) o de forma combinacional (igualdades).

Figura 4: Anotación de una porción asociada.

- Porción de lectura permanente.

Se entiende por porción de lectura permanente (PLP) al contenido del programa que debe ser evaluado continuamente, es decir, en cada ciclo de lectura. Estará ubicada en la última situación y se representa según la Figura 5.

Su contenido vendrá dado por:

a) la activación y desactivación de salidas y variables internas, ya se materialicen con instrucciones de memoria (Set y Reset) o de forma combinacional (igualdades).
b) Generación de variables temporizadas, de contaje, tipo flanco, etc.

Figura 5: Anotación de la PLP.

- Saltos condicionados.

A cada situación asociada a una condición le precede una selección de dos secuencias cuyas transiciones están definidas por dos saltos condicionados (S CN), ligados al cumplimiento o no de la mencionada condición asociada. Se representa según Figura 6.

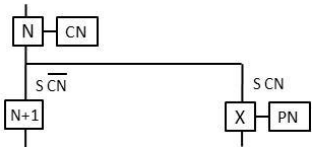

Figura 6: Anotación de los saltos condicionados.

El primer salto se produce cuando no se verifica el contenido de la condición "CN" y se direcciona hacia la situación siguiente "N+1"; caso de verificarse "CN" provocaría el otro salto hacia la porción asociada "PN".

- Saltos absolutos.

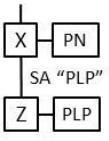

Figura 7: Anotación de saltos absolutos.

A cada situación ligada a una porción asociada le precede un único salto de tipo absoluto (SA).

Este salto podrá realizarse hacia la porción de lectura permanente, tal como muestra la Figura 7, o bien hacia cualquier otra situación que lo requiera.

3. Desarrollo del método.

- Gráfico de Estructuración del Programa.

La base principal de MEPUS es la representación del Gráfico de Estructuración del Programa de usuario (GEP). El GEP representa gráficamente el contenido del programa de un sistema de control basado en grafcet, en este caso.

La Figura 8 muestra la representación general del GEP, donde se aprecia los símbolos utilizados y finalidad de cada uno de ellos.

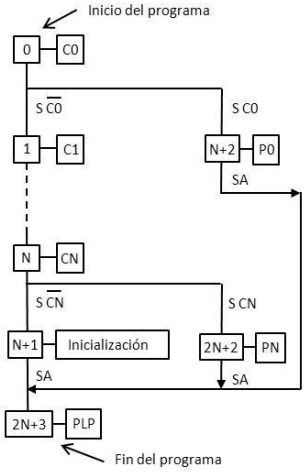

Figura 8: Estructura del GEP.

Del mismo, y teniendo en cuenta que "N" es el número de la etapa del Grafcet, se deduce que:

1) Cada situación que identifica una parte del programa se anota mediante un cuadrado numerado internamente. La numeración comienza con "0" y en este caso identifica la primera parte del programa de usuario a desarrollar.
2) La situación "N+1" está reservada para colocar la inicialización del Grafcet. Este contenido sólo se procesa en el primer ciclo de lectura del PLC.
3) A partir de la situación "N+2" comienza la ubicación de las porciones asociadas, siguiendo también un orden creciente hasta la situación "2N+2" que contendrá la porción asociada a la condición "CN".
4) Al final del programa, en la situación "2N+3", se ubica la PLP.

En el programa las situaciones están ligadas mediante instrucciones de saltos condicionados y absolutos. Por ejemplo, en el G.E.P. representado en la Figura 8, se observa:

1) En la situación "0" se ubica como contenido la condición "C0" (¿está activada la etapa 0?):

a) Si se verifica "C0", se ejecuta un salto condicionado (SC0) hacia la situación "N+2" donde se encuentra el contenido de la porción asociada "P0". Una vez procesado el contenido de "P0", se ejecuta un salto absoluto "SA" hacia la situación "2N+3", donde se encuentra el contenido de la porción de lectura permanente "PLP".

b) Si no se verifica "C0", el programa prosigue a la situación "1" donde se procesa el contenido definido por "C1".

2) En la situación "1" el procedimiento es análogo al establecido para la situación "0".

- Elaboración del programa de usuario

A partir del G.E.P. obtenido se elabora el programa de usuario, donde se establecen de forma ordenada el contenido asociado a cada una de las situaciones del G.E.P., según el lenguaje de programación elegido. El orden de las partes del programa se realizará tal y como se indica en la Tabla 1.

Tabla 1: Estructura general del programa.

Condición "C0" Salto condicionado "S C0" hacia "P0" ... Condición "CN" Salto condicionado "S CN" hacia "PN" Inicialización Salto absoluto "SA" hacia la PLP
Porción "P0" Salto absoluto "SA" hacia la PLP ... Porción "PN"
Porción de lectura permanente "PLP"

Se observa que no es necesario incorporar en el programa los saltos condicionados y absolutos que ligan dos situaciones consecutivas. En nuestra estructura corresponde a todos los saltos condicionados a la no verificación de cualquier condición y al último salto absoluto desde "PN" hacia "PLP".

4. Ejemplo de aplicación 1 (Secuencia única).

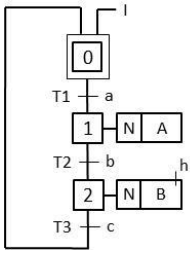

Figura 9: Grafcet de tres etapas y secuencia única.

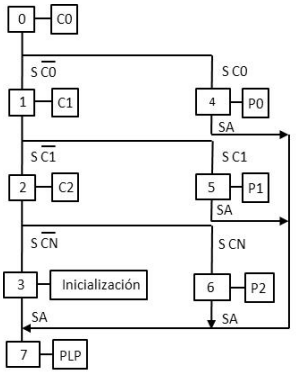

Figura 10: G.E.P. del grafcet de la Figura 9.

La Figura 9 ilustra un grafcet de secuencia única constituido por tres etapas. La acción A está activa cuando lo esté la "etapa 1" y la acción B está activa cuando lo esté la "etapa 2" siempre que esté presente la señal "h". La condición de inicialización "I" es necesaria para poner operativo el grafcet cuando se ejecuta por primera vez.

- Expresiones lógicas.

Las expresiones lógicas relativas al contenido de las condiciones, porciones asociadas y porción de lectura permanente se muestran en la Tabla 2. El orden de ubicación de cada una de ellas se realizará según el G.E.P. de la Figura 10.

Tabla 2: Expresiones lógicas (S = Set; R = Reset).

C0: Si está %M0 ➔ SALTA A (P0)
C1: Si está %M1 ➔ SALTA A (P1)
C2: Si está %M2 ➔ SALTA A (P2)
Inicialización: Si no está %M0 ➔ S(%M0) SALTA A (PLP)
P0: Si está "a" ➔ S(%M1), R(%M0) SALTA A (PLP)
P1: Si está "b" ➔ S(%M2), R(%M1) SALTA A (PLP)
P2: Si está "c" ➔ S(%M0), R(%M2) SALTA A (PLP)
PLP: Si está %M1 = %QA; si está %M2·"h" = %QB

- Programa de usuario.

Tabla 3: Programa de usuario del ejemplo 1.

Nº	tipo	instrucciones
0	C0:	LD "%M0"
	SC P0:	JMPC P0
1	C1:	LD "%M1"
	SC P1:	JMPC P1
2	C2:	LD "%M2"
	SC P2:	JMPC P2
3	Inicialización:	LDN "%M0" S "%M0"
	SA:	JMP PLP
4	P0:	P0: LD "a" S "%M1" R "%M0"
	SA:	JMP PLP
5	P1:	P1: LD "b" S "%M2" R "%M1"
	SA:	JMP PLP
6	P2:	P2: LD "c" S "%M0" R "%M2"
7	PLP:	PLP: LD "%M2" ST "%QA" LD "%M3" AND "h" ST "%QB"

En la Tabla 3 el programa de usuario ha sido desarrollado mediante el lenguaje de instrucciones. El significado de cada una de estas columnas es el siguiente:

1) La primera columna indica el orden de ubicación de los contenidos.
2) La segunda columna especifica el tipo de contenido: condiciones, saltos, porciones asociadas o porción de lectura permanente.
3) La tercera columna especifica el programa vinculado a cada uno de los contenidos.

5. Ejemplo de aplicación 2 (Secuencias alternativas).

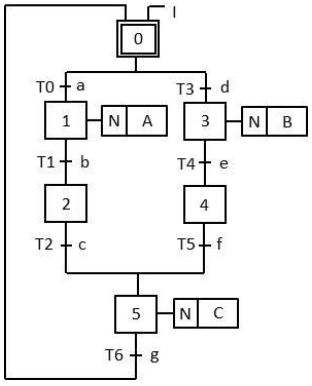

Figura 11: Grafcet con secuencias alternativas y seis etapas.

La Figura 11 ilustra un grafcet de secuencias alternativas constituido por siete etapas. En él se puede observar que después de la etapa 0, existen dos alternativas. Si se verifica la transición "a" se activará la secuencia de la izquierda. Mientras que si se verifica la transición "d" se procesará la secuencia de la derecha.

- Gráfico de Estructuración del Programa.

A partir del grafcet de la Figura 11 se representa el G.E.P. de la figura 12.

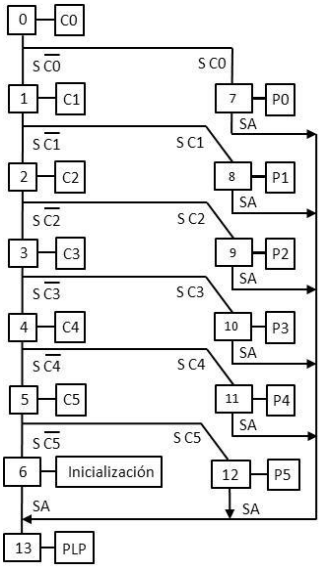

Figura 12: G.E.P. del grafcet de la Figura 11.

Se observa que se sigue la misma metodología. Se redactan primero las condiciones con sus saltos condicionados y a continuación la inicialización, porciones y por último la PLP.

- Expresiones lógicas.

Tabla 4: Expresiones lógicas.

C0: Si está %M0 ➔ SALTA A (P0)
C1: Si está %M1 ➔ SALTA A (P1)
C2: Si está %M2 ➔ SALTA A (P2)
C3: Si está %M3 ➔ SALTA A (P3)
C4: Si está %M4 ➔ SALTA A (P4)
C5: Si está %M5 ➔ SALTA A (P5)
Inicialización: Si no está %M0 ➔ S(%M0) SALTA A (PLP)
P0: Si está "a" ➔ S(%M1), R(%M0) Si está "d" ➔ S(%M3), R(%M0) SALTA A (PLP)
P1: Si está "b" ➔ S(%M2), R(%M1) SALTA A (PLP)
P2: Si está "c" ➔ S(%M5), R(%M2) SALTA A (PLP)
P3: Si está "e" ➔ S(%M4), R(%M3) SALTA A (PLP)
P4: Si está "f" ➔ S(%M5), R(%M4) SALTA A (PLP)
P5: Si está "g" ➔ S(%M0), R(%M2), R(%M4) SALTA A (PLP)
PLP: Si está %M1 = %QA; Si está %M3 = %QB; Si está %M5 = %QC

Para comparar la longitud de programa a procesar, frente al caso de utilizar la metodología Grafcet en este ejemplo 2, obtendremos primero el programa en dicha metodología. Se puede observar en la Tabla 6, donde se han numerado las líneas de programa necesarias.

Tabla 6: Programa de usuario del ejemplo 2 con Grafcet.

Nº Líneas	instrucciones	Nº Líneas	instrucciones
1	LD "%M5"	22	LD "%M2"
2	AND "g"	23	AND "c"
3	OR "I"	24	OR("%M4
4	S "%M0"	25	AND "f"
5	R "%M5"	26	)
6	LD "%M0"	27	S "%M5"
7	AND "a"	28	R "%M2"
8	S "%M1"	29	R "%M4"
9	R "%M0"	30	LD "%M1"
10	LD "%M1"	31	ST "%QA"
11	AND "b"	32	LD "%M3"
12	S "%M2"	33	ST "%QB"
13	R "%M1"	34	LD "%M5"
14	LD "%M0"	35	ST "%QC"
15	AND "d"	36	LDN "%M0"
16	S "%M3"	37	LDN "%M1"
17	R "%M0"	38	LDN "%M2"
18	LD "%M3"	39	LDN "%M3"
19	AND "e"	40	LDN "%M4"
20	S "%M4"	41	LDN "%M5"
21	R "%M3"	42	ST "I"

Según la Tabla 6, el autómata siempre procesará, en cada ciclo de scan, 42 líneas de comando. En cambio en el programa de la Tabla 5, tendremos en el primer ciclo de scan, que sólo se produce la primera vez que conectamos el PLC, 20 líneas.

Cuando el proceso se encuentra en la etapa "0", 15 líneas y cuando se encuentra en el resto de etapas, de la "1" a la "5", 12 líneas.

De ello es fácil deducir que cuanto más largo sea el proceso de secuencia única, o más secuencias alternativas tenga el programa a implementar, mayor será la reducción del tiempo de ciclo de scan del PLC.

- Programa de usuario.

Tabla 5: Programa de usuario del ejemplo 2.

Nº	tipo	instrucciones
0	CONDICIÓN, C0:	LD "%M0"
	SALTO CONDICIONADO, SC C0:	JMPC P0
1	CONDICIÓN, C1:	LD "%M1"
	SALTO CONDICIONADO, SC C1:	JMPC P1
2	CONDICIÓN, C2:	LD "%M2"
	SALTO CONDICIONADO, SC C2:	JMPC P2
3	CONDICIÓN, C3:	LD "%M3"
	SALTO CONDICIONADO, SC C3:	JMPC P3
4	CONDICIÓN, C4:	LD "%M4"
	SALTO CONDICIONADO, SC C4:	JMPC P4
5	CONDICIÓN, C5:	LD "%M5"
	SALTO CONDICIONADO, SC C5:	JMPC P5
6	Inicialización:	LDN "%M0" S "%M0"
	SALTO ABSOLUTO, SA:	JMP PLP
7	PORCIÓN, P0:	P0: LD "a" S "%M1" R "%M0" LD "d" S "%M3" R "%M0"
	SALTO ABSOLUTO, SA:	JMP PLP
8	PORCIÓN, P1:	P1: LD "b" S "%M2" R "%M1"
	SALTO ABSOLUTO, SA:	JMP PLP
9	PORCIÓN, P2:	P2: LD "c" S "%M5" R "%M2"
	SALTO ABSOLUTO, SA:	JMP PLP
10	PORCIÓN, P3:	P3: LD "e" S "%M4" R "%M3"
	SALTO ABSOLUTO, SA:	JMP PLP
11	PORCIÓN, P4:	P4: LD "f" S "%M5" R "%M4"
	SALTO ABSOLUTO, SA:	JMP PLP
12	PORCIÓN, P5:	P5: LD "g" S "%M0" R "%M2" R "%M4"
13	PORCIÓN, PLP:	PLP: LD %M1" ST "%QA" LD "%M3" ST "%QB" LD "%M5" ST "%QC"

6. Ejemplo de aplicación 3 (Secuencias simultáneas).

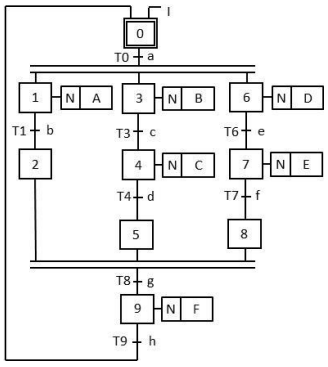

Figura 13: Grafcet con secuencias paralelas y diez etapas.

La Figura 13 ilustra un grafcet de secuencias paralelas constituido por diez etapas. En él se puede observar que existen tres secuencias simultáneas que se activan desde la etapa "0" y con la transición "a". Para salir de dichas secuencias el proceso se debe encontrar en las etapas 2, 5 y 8 y verificarse la transición "g".

- Gráfico de Estructuración del Programa.

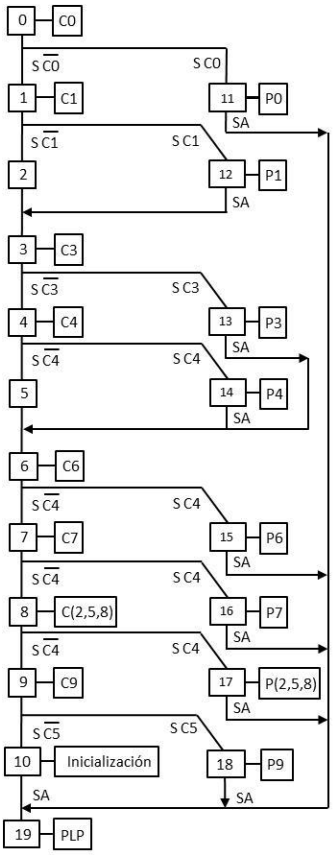

Figura 14: G.E.P. del grafcet de la Figura 13.

A partir del Grafcet de la Figura 13 se representa el G.E.P. Hay que tener presente que las tres secuencias simultáneas se deben leer en cada ciclo de lectura del PLC. Así pues, el G.E.P. resultante se muestra en la figura 14.

Tabla 8: Programa de usuario del ejemplo 3.

Nº	tipo	instrucciones
0	CONDICIÓN, C0:	LD "%M0"
	SALTO CONDICIONADO, SC C0:	JMPC P0
1	CONDICIÓN, C1:	LD "%M1"
	SALTO CONDICIONADO, SC C1:	JMPC P1
3	CONDICIÓN, C3:	LD "%M3"
	SALTO CONDICIONADO, SC C3:	JMPC P3
4	CONDICIÓN, C4:	LD "%M4"
	SALTO CONDICIONADO, SC C4:	JMPC P4
6	CONDICIÓN, C6:	LD "%M6"
	SALTO CONDICIONADO, SC C6:	JMPC P6
7	CONDICIÓN, C7:	LD "%M7"
	SALTO CONDICIONADO, SC C7:	JMPC P7
8	CONDICIÓN, C(2,5,8):	LD "%M2"
		AND "%M5"
		AND "%M8"
	SALTO CONDICIONADO, SC C(2,5,8):	JMPC P(2,5,8)
9	CONDICIÓN, C9:	LD "%M9"
	SALTO CONDICIONADO, SC C9:	JMPC P9
10	Inicialización:	LDN "%M0" S "%M0"
	SALTO ABSOLUTO, SA:	JMP PPR
11	PORCIÓN, P0:	LD "a" S "%M1"; S "%M3" S "%M6"; R "%M0"
	SALTO ABSOLUTO, SA:	JMP PPR
12	PORCIÓN, P1:	LD "b" S "%M2"; R "%M1"
	SALTO ABSOLUTO, SA:	JMP C3
13	PORCIÓN, P3:	LD "c" S "%M4"; R "%M3"
	SALTO ABSOLUTO, SA:	JMP C6
14	PORCIÓN, P4:	LD "d" S "%M5"; R "%M4"
	SALTO ABSOLUTO, SA:	JMP C6
15	PORCIÓN, P6:	LD "e" S "%M7"; R "%M6"
	SALTO ABSOLUTO, SA:	JMP PLP
16	PORCIÓN, P7:	LD "f" S "%M8"; R "%M7"
	SALTO ABSOLUTO, SA:	JMP PLP
17	PORCIÓN, P(2,5,8):	LD "g" S "%M9"; R "%M2" R "%M5"; R "%M8"
	SALTO ABSOLUTO, SA:	JMP PLP
18	P9:	LD "h" S "%M0"; R "%M9"
19	PLP:	LD "%M1"; ST "%QA" LD "%M3"; ST "%QB" LD "%M4"; ST "%QC" LD "%M6"; ST "%QD" LD "%M7"; ST "%QE" LD "%M9"; ST "%QF"

Se observa que se sigue la misma metodología. Se redactan primero las condiciones y a continuación la inicialización, porciones y por último la PLP.

Como característica particular, y debido a la singularidad de este tipo de procesos, las etapas finales de las líneas simultáneas no poseen condiciones particulares, sino que se reúnen en una sola en la última etapa de la línea situada más a la derecha. En este caso las etapas 2 y 5, no poseen condiciones, y la etapa 8 recoge la suya y las anteriores, es decir hay una condición C(2,5,8). Lo mismo ocurre con sus porciones.

Otra variable a tener en cuenta es que desde las etapas de las líneas simultáneas no se salta a la PLP, sino a las primeras etapas de las líneas simultáneas situadas a su derecha. Sólo desde las etapas de la línea simultánea situada más a la derecha se salta a la PLP.

En este ejemplo desde las etapas 1 y 2 se salta a la etapa 3; de las etapas 3, 4 y 5 a la etapa 6; y por último desde las etapas 6, 7, 8 y 9 se salta a la PLP.

En la tabla 8 se observa la secuencia del programa de usuario.

Tabla 7: Expresiones lógicas.

C0: Si está %M0 ➔ SALTA A (P0)
C1: Si está %M1 ➔ SALTA A (P1)
C3: Si está %M3 ➔ SALTA A (P3)
C4: Si está %M4 ➔ SALTA A (P4)
C6: Si está %M6 ➔ SALTA A (P6)
C7: Si está %M7 ➔ SALTA A (P7)
C(2,5,8): Si está %M2, %M5 y %M8 ➔ SALTA A (P(2,5,8))
C9: Si está %M9 ➔ SALTA A (P9)
Inicialización: Si no está %M0 ➔ S(%M0)
P0: Si está "a" ➔ S(%M1), S(%M3), S(%M6), R(%M0) SALTA A (PLP)
P1: Si está "b" ➔ S(%M2), R(%M1) SALTA A (C3)
P3: Si está "c" ➔ S(%M4), R(%M3) SALTA A (C6)
P4: Si está "d" ➔ S(%M5), R(%M4) SALTA A (C6)
P6: Si está "e" ➔ S(%M7), R(%M6) SALTA A (PLP)
P7: Si está "f" ➔ S(%M8), R(%M7) SALTA A (PLP)
P(2,5,8): Si está "g" ➔ S(%M9), R(%M2), R(%M5), R(%M8)
PLP: Si está %M1 = %QA; Si está %M3 = %QB; Si está %M4 = %QC; Si está %M6 = %QD; Si está %M7 = %QE; Si está %M9 = %QF;

7. Conclusiones.

En este trabajo se han analizado las causas de las aleatoriedades de programación, concluyendo que son debidas a la forma de procesar los programas de usuario introducidos en los PLCs (Pérez, 1988; Silva, 1985; Daniel Gomez et al, 2008, Frank Schumacher, 2011).

Para eliminar estos problemas, se ha desarrollado una nueva metodología, que permite elaborar de forma estructurada el programa de usuario.

En comparación con los métodos planteados por los autores que se han comentados en párrafos anteriores, el método desarrollado en este trabajo, además de resolver la problemática de las aleatoriedades de programación, presenta las siguientes ventajas:

a) Disminuye, todavía más, el tiempo de ciclo de lectura, ya que el contenido de programa que se procesa es menor que en los casos anteriores.
b) Reduce los recursos utilizados, ya que no se añaden ni etapas fantasmas, ni se definen nuevas variables para

identificar cada una de las condiciones de tránsito entre etapas del grafcet. Además se puede reducir el número de temporizadores y/o contadores a utilizar, ya que la estructura permite el poder utilizarlos en las porciones y como no se procesan simultáneamente se pueden utilizar los mismos, aunque los tiempos o contajes sean distintos.

c) Estructura el programa de una forma más clara, sencilla y sistemática, ya que se ha hecho coincidir los números de las etapas del Grafcet con las condiciones del G.E.P., que permite al diseñador tener una visión global de todo el programa de usuario más rápida.

English Summary

Methodology for developing programmes to implement in programmable logic controllers (PLCs). MEPUS.

Abstract

The aim of this paper is to present a methodology for the development of the user program to which we have called MEPUS (Methodology for Structuring the User Program by Jumping, in Spanish). This methodology eliminates the randomness of programming, while considerably the time of the reading cycle or scan, especially in the sequential processes. Furthermore, it is independent of the design method used to represent the secuencialization model of the process to automate, be it Petri net, Grafcet, Binodal Theory, etc.

It is not a new method of design of scheduled automations. It is a graphical tool for the drafting of the user program. Also, and even it is a minor advantage, the PLC's internal resources are optimized (timers, counters, etc.), as this methodology allows them to be reused in the same program. Based on the results obtained during more than 10 years, it is concluded that this new methodology the user program is obtained more easily, especially way clear and systematic, which facilitates learning by the learner. For this work we have chosen as model of secuentialization of the process to automate the Grafcet, because it is the most used by PLC programmers.

Keywords:

Automatic, Control, Methodology, PLC's, Programs, Random processes.

Referencias

Cassandras, C., Lafortune, S., 2006. Introduction to Discrete Event Systems. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Daniel Gómez, Enrique Baeyens, Clemente Cárdenas, Eduardo J. Moya, 2008. Supervisión de sistemas lógicos de control utilizando el diagrama de evolución del estado. Revista Iberoamericana de Automática e Informática industrial 8 (2011) 196–203

Frank Schumacher, Alexander Fay. Requirements and obstacles for the transformation of GRAFCET specifications into IEC 61131-3 PLC programs. IEEE ETFA'2011.

Hrúz, B., Zhou, M., 2007. Modeling and Control of Discrete-event Dynamic Systems: with Petri Nets and Other Tools. Springer Publishing Company, Inc.

Javier de las Morenas, Andrés García, Fernando Martínez, Pablo García Ansola. Implementación del Control en Planta de un Centro de Distribución Automatizado mediante Agentes Físicos y RFID. Revista Iberoamericana de Automática e Informática industrial 12 (2015) 25–35

J.L. Pérez, F. Mateos y M.A. Marcos, Normalización del diagrama funcional (Grafcet) para su aplicación en autómatas programables industriales, SAAEI´98, 103-106 (1998).

J.M. Diez, Nueva metodología para la programación de PLCs con sistemas de control basados en Grafcet, Tesis Doctoral U.P.V, Valencia (2000).

J. M. Diez, R. Montoya y J. Giner, MEPUS: Nueva metodología para la elaboración de los programas a implementar en Autómatas Programables. Revista Información Tecnológica – Vol. 14 Nº 3 – 2003.

J. Zaytoon, J.J. Lesage, L. Marcé, J.M. Fauré y P. Lhoste, Vérification et validation du Grafcet, RAIRO-APII-JESA, Vol.31-4, 713-740 (1997).

M. Silva, Las Redes de Petri: en la automática y la informática. Editorial AC. Madrid (1985).

P. Lhoste, J.M. Fauré, J.J. Lesage y J. Zaytoon, Comportement temporel du Grafcet, RAIRO-APII-JESA, Vol.31-4, 695-711 (1997).

Quezada-Quezada José Carlos, Bautista-López Jorge, Flores-García Ernesto y Quezada-Aguilar Víctor. Diseño e implementación de un sistema de control y monitoreo basado en HMI-PLC para un pozo de agua potable. Ingeniería Investigación y Tecnología, volumen XV (número 1), enero-marzo 2014: 41-50

R. Montoya, Algoritmos y teoría para el diseño y programación de sistemas de control industrial materializados con PLCs. Tesis Doctoral U.P.V. Valencia (2001).

S. Moreno y E. Peulot, Le Grafcet: Conception-Implantation dans les Automates Programmables Industriels. Editions Casteilla, Paris (1996).