SIIL2, Lenguaje Textual del PLM3 para Control Lógico

Antonio Salvá*. Luis Altamirano.**

**Facultad de Ingeniería, Departamento de Control y Robótica.*
UNAM, D.F. 04510, México, Teléfono: (52)-55-56223108
salva@dctrl.fi-b.unam.mx
***Facultad de Ingeniería, Departamento de Control y Robótica.*
UNAM, D.F. 04510, México, Teléfono: (52)-55-56223108
altamirano@dctrl.fi-b.unam.mx

Resumen—*El control lógico tiene un amplio espectro de aplicaciones en la industria. Para la realización de una determinada automatización, se emplean autómatas programables mejor conocidos como Controladores Lógicos Programables (PLC por sus siglas en inglés). Los PLC cuentan con software para programación de los mismos para fines de una determinada aplicación. En lo fundamental, los lenguajes de programación propios de los PLC pueden ser de tres tipos a saber: Esquema de contactos o diagrama de escalera (LD por sus siglas en inglés), diagrama de bloques de funciones (FBD por sus siglas en inglés) y lista textual de instrucciones (IL por sus siglas en inglés). En este trabajo se describe, en lo fundamental, un lenguaje textual para control lógico denominado SIIL2 (Software de Interpretación de Instrucciones Lógicas versión 2), propio de un Controlador Lógico Secuencial básico (CLS) denominado PLM3 (Programador Lógico Modular versión 3). También se describe, en lo fundamental, la estructura del software de compilación de SIIL2, al que se le denominó COMSIIL2. En el desarrollo del PLM3 y del compilador COMSIIL2 se emplearon herramientas de software y hardware para aprendizaje y proyecto de aplicaciones basadas en microcontroladores. Tales facilidades fueron diseñadas en el Departamento de Control y Robótica de la Facultad de Ingeniería de la UNAM.*

Palabras Clave: *Automatización, Control Lógico, PLM, PLC, Microcontrolador.*

1. INTRODUCCIÓN

Como es bien sabido en el medio, el hardware funcional básico, que está detrás de la funcionalidad de un PLC para fines de control lógico, está integrado por los siguientes bloques:

- Computadora central
- Puertos binarios de entrada opto acoplados
- Puertos de salida binarios con actuación por relevador o TRIAC
- Fuente de poder

Actualmente la computadora central y los puertos binarios de entrada y salida se realizan mediante el empleo de un microcontrolador (MCU) comercial, siendo la computadora central la cpu del MCU implicado, y los puertos binarios de entrada y salida los propios del MCU. De esta forma, los bloques de un PLC para fines de control lógico podrían ser:

- MCU comercial
- Hardware de optoacoplamiento entre entradas físicas del PLC y puertos de entrada del MCU
- Interfaz entre puertos de salida del MCU y salidas físicas del PLC (relevadores o TRIACS)
- Fuente de poder

Considerando lo anterior, en el Departamento de Control y Robótica de la Facultad de Ingeniería de la UNAM, se han diseñado cuatro CLS físicos elementales y uno virtual a los que se les ha llamado genéricamente "Programador Lógico Modular" (PLM). Denominados como:

- PLM1, de 32 entradas y 16 salidas, basado en el MCU mc68hc11f1 de FREESCALE
- PLM2, de 16 entradas y 8 salidas, basado en el MCU mc68hc11f1 de FREESCALE
- PLM08, de 16 entradas y 8 salidas, basado en el MCU mc68hc908gp32 de FREESCALE
- PLM3, de 16 entradas y 8 salidas, basado en el MCU mc9s08gt60 de FREESCALE
- V-PLM, de 32 entradas y 16 salidas, éste es un simulador del PLM1 que corre bajo WINDOWS

Para el desarrollo de aplicaciones de control lógico basadas en el PLM1, PLM2 ó PLM08 se desarrolló un compilador cruzado de un lenguaje textual denominado SIIL1 (Software de Interpretación `de Instrucciones Lógicas versión 1). En lo que respecta al software para desarrollo con el PLM3, cabe señalar que se hicieron varias mejoras al lenguaje SIIL1, que

Reserva de Derechos No. EN TRÁMITE, ISSN. EN TRÁMITE

ya con éstas fue denominado SIIL2. El compilador cruzado correspondiente se hizo completamente nuevo y se le denominó COMSIIL2.

Para detalles acerca del hardware y software implicados para el PLM1, PLM2 y PLM08, pueden verse respectivamente las referencias: (Salvá, 1999), (Salvá, Sánchez, Salcedo y Ramírez, 2006) y (Salvá, Guzmán y Altamirano 2010). Por lo que toca acerca del V-PLM puede verse (Salvá, Altamirano, Dehesa y Hernández, 2003).

Cabe señalar, que los PLM están orientados únicamente al desarrollo de aplicaciones de Control Lógico, lo cual viene siendo sólo una de las funcionalidades con las que cuenta un PLC. Actualmente éstos últimos, además de poder realizar control lógico, cuentan con software y hardware que habilitan entre otras las siguientes facilidades:

- Capacidad de manejo de entradas y salidas analógicas
- Realización de control clásico (PID)
- Realización de control difuso
- Servidor web presente que habilita monitoreo remoto
- Software y hardware de interacción con MATLAB y SIMULINK

Sin embargo, potencialmente sería posible diseñar, software y hardware, que habilite para los PLM algunas de las funcionalidades propias de los PLC aquí mencionadas.

En este trabajo se describe en lo fundamental la sintaxis asociada con el lenguaje SIIL2 y se esboza como se diseño el compilador cruzado COMSIIL2. Aspectos acerca del hardware propio del PLM3 y un ejemplo de aplicación práctica pueden verse en (Salvá, Altamirano, Álvarez y Herrera, 2015).

2. ASPECTOS BÁSICOS DEL PLM3

En lo que toca al hardware del PLM3, actualmente se cuenta con un prototipo funcional. Éste se realizó a partir del PLM2, cambiando la computadora central de éste por una tarjeta denominada MINICON_08GT, que está basada en el MCU mc9s08gt60 de FREESCALE.

2.1 Variables binarias del PLM3

Las variables binarias manejadas por el PLM3 pueden ser de tres tipos a saber:

- Variables binarias de entrada física (VBE), contenidas en dos grupos de ocho de ellas. Denotadas como 'EJK', donde 'J' denota el grupo y 'K' denota el bit implicado. Por ejemplo, el string 'E16' indica a la VBE que es el bit seis del grupo uno de entradas.
- Variables binarias de salida física (VBS), contenidas en un grupo de ocho de ellas. Denotadas como 'SJK', donde 'J' denota el grupo y 'K' denota el bit implicado. Por ejemplo, el string 'S03' indica a la VBS que es el bit tres del grupo cero de entradas.
- Variables binarias intermediarias (VBI), contenidas en diez grupos de ocho de ellas. Denotadas como 'IJK', donde 'J' denota el grupo y 'K' denota el bit implicado. Por ejemplo, el string 'I47' indica a la VBI que es el bit siete del grupo cuatro de variables intermediarias.

2.2 Software para desarrollo con el PLM3

Para el desarrollo de aplicaciones de control lógico con el PLM3, se diseñó un ambiente manejador de éste, al que se le denominó SWMANPLM3, el cual contiene entre otras facilidades las siguientes:

- Compilador cruzado COMSIIL2
- Editor de texto plano para la introducción del programa fuente en SIIL2
- Capacidad de compilación del programa fuente presente sin ejecución en el PLM3
- Capacidad de compilación y ejecución del programa fuente presente en el PLM3

En la figura 1 se muestra un esquema del ambiente para desarrollo con el PLM3

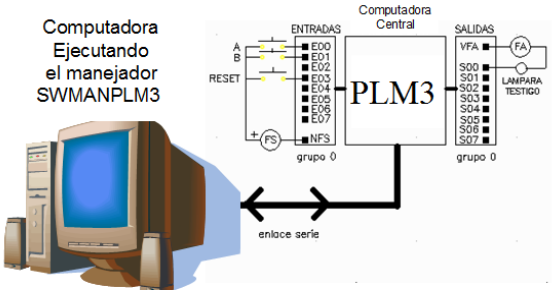

Fig. 1. Ambiente para desarrollo con el PLM3

3. ASPECTOS BÁSICOS DEL LENGUAJE SIIL2

El PLM3 puede realizar las siguientes funciones elementales de control lógico:

- Seguidores lógicos
- Inversores lógicos

- Compuertas lógicas AND, NAND, OR, NOR, XOR y XOR negada, de 2, 3 y 4 entradas
- Temporizadores monodisparo con disparo por nivel (one-shot)
- Temporizadores monodisparo con disparo por flanco (one-shot)
- Temporizadores con retardo a la activación (on-delay)
- Temporizadores con retardo a la desactivación (off-delay)
- Temporizadores astables
- Contadores de eventos
- Comparadores de cuenta de contador de eventos
- Flip-Flops RS asíncronos
- Secuenciadores de estados de un bit (drummers)

En la nomenclatura del PLM3 a las entidades que realizan las funciones de control lógico antes citadas se les denomina módulos lógicos (ML). Para una determinada aplicación, con excepción de los ML secuenciadores de estado, cada módulo lógico usado tendrá asociada una sentencia declaratoria en el programa fuente en SIIL2 asociado. Para los ML secuenciadores de estado habrá más de una sentencia asociada con éstos, ya que se usarían otros renglones en el programa fuente para indicar los estados implicados.

3.1 Formato de las sentencias declaratorias de los ML

La forma genérica para la declaración en SIIL2 de los ML que puede realizar el PLM3 es:

CODM#N E1,.. En,.. S1,.. Sm, D1,.., Dq, CADBI

Donde:

- CODM es una cadena de caracteres que simboliza la función efectuada por el módulo, por ejemplo, para una compuerta and de tres entradas CODM será ‘AND3’ y para un temporizador monodisparo con activación por nivel será ‘TEMPOC’.
- N es el número asociado con el módulo, ya que todos los ML de un mismo tipo de una aplicación, deben ser numerados.
- E1 a En son las designaciones asociadas con las n variables de entrada; n puede ser un valor de uno a cuatro.
- S1 a Sm son las designaciones asociadas con las m variables de salida; m puede ser uno ó dos.
- D1 a Dq son datos auxiliares que pudieran ser requeridos por algunos módulos. Estos podrían ser entre otros: el tiempo asociado con la duración de un pulso generado por un temporizador, tiempos asociados con el ciclo básico de un temporizador astable, etc. Hay módulos que no requieren de estas especificaciones, como las compuertas lógicas. Para los módulos que si requieren de estos datos, “q” puede ser uno o dos.
- CADBI es una cadena formada por unos y ceros, que especifica diversas características de funcionamiento, como podrían ser: qué entradas a una compuerta van a tener preinversión, a qué tipo de flanco responde una entrada de algún otro tipo de módulo, qué nivel de verificación tendrá la salida de un temporizador, entre otras funcionalidades

3.2 Ejemplo 1

Se desea realizar con el PLM3 una compuerta ‘and’ de dos entradas sin preinversión. Se requiere que las entradas sean las VBE E00 y E01, y la salida sea la VBI I00. Además, se le asignará a este ML el número uno. Véase la figura 2. En SIIL2 la forma genérica de la declaración de una compuerta ‘and’ de dos entradas es:

AND2#N E0, E1, SALIDA, AB

Donde:
N es el número que se le asigna a cada uno de los ML de tipo ‘and2’ empleados en la aplicación.
E0 y E1 son las dos entradas, éstas pueden ser variables binarias VBE o VBS o VBI.
SALIDA, puede ser una variable binaria VBS o VBI.
A es un dígito binario que debe ser cero si se desea que la entrada ‘E1’ tenga preinversión, en otro caso debe ser uno.
B es un dígito binario que debe ser cero si se desea que la entrada ‘E0’ tenga preinversión, en otro caso debe ser uno.

Para este ejemplo, la declaración en SIIL2 requerida será:

AND2#1 E00, E01, S00, 11

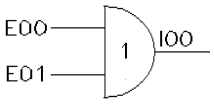

Fig. 2. Módulo lógico del ejemplo 1 como bloque funcional.

Para una descripción detallada acerca de la sintaxis asociada, con cada una de las sentencias declaratorias en SIIL2 requeridas, por cada uno de los ML realizables con el PLM3, véase (Salvá, 2015). Ya que ésta información no se incluye aquí por razones de espacio.

3.3 Estructura de un programa en SIIL2

Un programa en SIIL2 estará integrado por dos subprogramas, a uno se le llamó ‘subprograma principal’ (SPP), y al otro se le denominó ‘subprograma temporizado’ (SPT). El SPP en lo básico inicializa recursos del MCU y

pasa a un lazo que se ejecuta cíclicamente. El flujo de ejecución de éste se muestra en la figura 3.

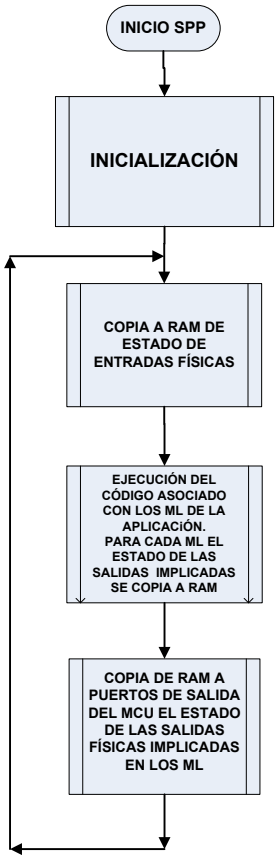

Fig. 3. Flujo de ejecución del SPP

Por otra parte, el SPT debe ejecutarse periódicamente cada 10 [ms]. Para lograr esto, el código de éste será parte de una rutina de interrupción por overflow del MCU que realiza la computadora central del PLM3.
En general, cada ML podrá tener uno o dos tramos de código en ensamblador asociados. Cuando es uno solo, éste deberá ser parte del SPP. Entre otros, este es el caso de las compuertas lógicas. Para los ML que requieren dos tramos de código en ensamblador, uno de ellos estará en el SPP, y el otro en el SPT; por ejemplo, tal es el caso de los temporizadores.
Para fines ilustrativos, a continuación se muestra el tramo de código en ensamblador asociado con un inversor lógico cuya entrada es la VBE E00 y su salida es la VBI I23. Además, a este ML se le asignó el número cinco. El código implicado es:

```
                brclr 0, gpe0, invl_uno005
                bclr 3, gpi2
                bra invl_sal005
invl_uno005:    bset 3, gpi2
invl_sal005:    nop
```

Donde: gpe0 y gpi2 son localidades de RAM, asociadas con los grupos de la entrada y la salida del inversor lógico. Para otros ML los tramos de código son más largos y complicados.
El armado de los tramos de código en ensamblador, la configuración del MCU para fines de la interrupción de overflow requerida, y otras acciones diversas para la generación del código en ensamblador, asociado con el programa en SIIL2 implicado, es trabajo del compilador COMSIIL2. Todo esto es transparente para el programador en SIIL2, éste simplemente debe estructurar su programa fuente en SIIL2 bajo la forma genérica que se muestra a continuación:

```
inprog ‘Sentencia delimitadora de inicio del subprograma principal
Sentencias declaratorias de los ML implicados en la aplicación
finpp  ‘Sentencia delimitadora del final del subprograma principal
‘Sentencias delimitadoras de código a ejecutarse con
‘una temporalidad base de 10 ms.
inmodi ‘Inicio del subprograma temporizado
finmodi ‘Fin del subprograma temporizado
```

Nótese el uso del carácter apostrofe para delimitar comentarios.
Las sentencias inmodi y finmodi deberán ir siempre colocadas en la forma mostrada. El compilador COMSIIL2 las usa para colocar el código asociado con el SPT cuando esto es requerido.

3.4 Ejemplo 2

Se desea realizar con el PLM3 el sistema lógico mostrado en la figura 4. El temporizador *one-shot* debe generar un pulso de dos minutos verificado en alto, esto cuando se tenga un flanco de subida en su entrada de disparo. El nivel de verificación de la entrada de reset debe ser alto, el nivel de verificación del pulso de salida debe ser alto. Para fines de claridad en este ejemplo, a continuación se detalla la sintaxis asociada con la sentencia declaratoria de los temporizadores *one-shot* con disparo por flanco. Ésta es:

TEMPOF#N ED, ER, STM, MM.SS.CC, ABC

Donde:
N es el número que se le asigna a cada uno de los ML de tipo ‘tempof’ empleados en la aplicación.
ED es la entrada de disparo. Puede ser una VBE o VBS o VBI
ER es la entrada de reset. Puede ser una VBE o VBS o VBI
STM es la salida. Puede ser una variable binaria VBS o VBI.
MM.SS.CC es la duración del pulso de salida en minutos, segundos y centésimas.
A es un dígito binario que debe ser cero, si se desea que el disparo sea por flanco de bajada, en otro caso debe ser uno.

B es un dígito binario que debe ser cero, si se desea que el nivel de verificación de la entrada de reset sea bajo, en otro caso, debe ser uno.
C es un dígito binario que debe ser cero, si se desea que el nivel de verificación de la salida sea bajo, en otro caso, debe ser uno.

Considerando las formas sintácticas de las declaraciones de los ML implicados en este ejemplo, el programa fuente requerido en SIIL2 es:

```
inprog
and2#1  e00, e01, i00, 11
tempof#1  i00, e03, s00, 02.00.00, 111
finpp
inmodi
finmodi
```

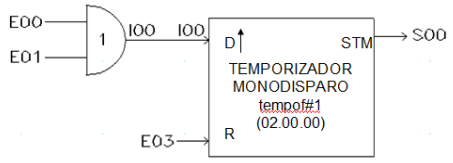

Fig. 4. Sistema lógico del ejemplo 2

4. ASPECTOS BÁSICOS DE COMSIIL2

Para el desarrollo del compilador COMSIIL2 se empleó infraestructura de software y hardware para manejo de microcontroladores previamente diseñada en el Departamento de Control y Robótica de la Facultad de Ingeniería de la UNAM. Aquí se presentan aspectos básicos acerca de su funcionalidad y las etapas de software que lo integran.

4.1 Funcionalidad COMSIIL2

COMSIIL2 es parte de un ambiente para desarrollo con el PLM3 denominado SWMANPLM3, éste se desarrolló a partir un software manejador de microcontroladores HC08 diseñado previamente. Éste contiene, entre otras facilidades, un ensamblador cruzado denominado ENS08 y un cargador y ejecutor de código de máquina HC08, previamente generado por el ensamblador ENS08, véase (Salvá, 2011).
En la figura 5 se muestra la ventana principal del SWMANPLM3, en el área de edición se aprecia el programa del ejemplo 2, al que se le denominó 'ejsil_amca_2015'.

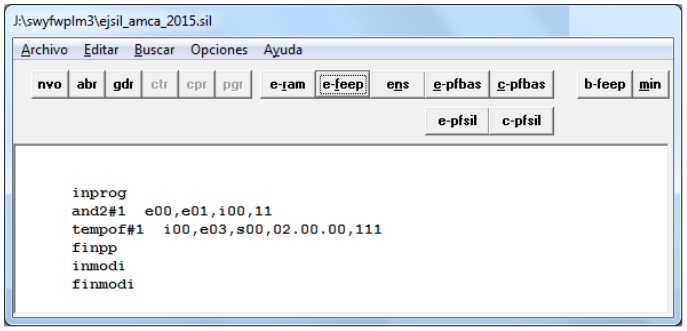

Fig. 5. Ventana principal del SWMANPLM3

El editor cuenta con facilidades para la captura, guardado y recuperación de programas fuente en SIIL2. Para la funcionalidad elemental desde el punto de vista del programador, existen tres botones denotados <c-pfsil>, <e-pfsil> y <b-feep>. A continuación se describen las acciones que se efectúan al oprimirse éstos.

Botón <c-pfsil>, al oprimirse se compila el programa fuente presente en la ventana de edición. Si no se detectan errores se notifica al usuario que la compilación fue exitosa; en otro caso, se abre una ventana donde se muestran éstos.
Botón <e-pfsil>, al oprimirse se compila el programa fuente presente en la ventana de edición. Si no se detectan errores se carga y ejecuta en el MCU del PLM3 el código objeto asociado; en otro caso, se abre una ventana donde se muestran los errores que se hayan presentado.
Botón <b-feep>, al oprimirse se borra la memoria de programa del MCU del PLM3; cabe señalar que antes de grabar un programa nuevo, el anterior debe ser borrado en su totalidad.

4.2 Etapas de software de COMSIIL2

En la figura 6 se muestra un diagrama de flujo simplificado donde se muestran las diversas etapas de software que comprenden al compilador COMSIIL2.

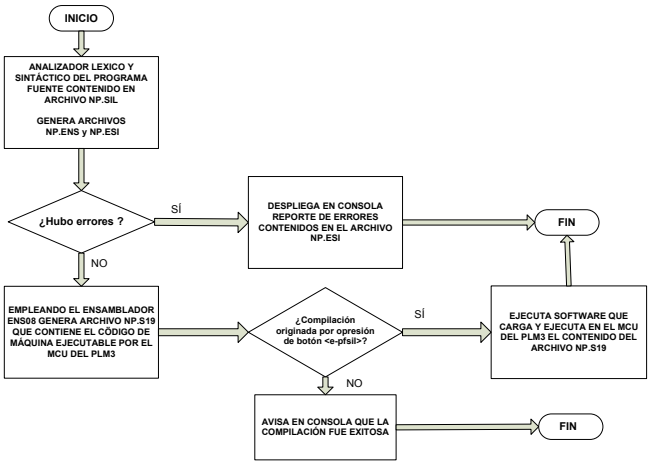

Fig. 6. Diagrama de flujo simplificado de COMSIIL2

Como se aprecia en la figura 6, el compilador procesa un archivo denominado ahí genéricamente como np.sil. Éste contiene al código fuente en SIIL2. NP es el nombre que el usuario le haya querido dar al programa fuente implicado. Al procesarse éste por el analizador sintáctico y léxico, esta etapa genera dos archivos de salida, uno denominado np.esi, que contiene un reporte de errores, si los hubo, y otro denominado np.ens, que contiene el código en ensamblador propio del MCU del PLM3, asociado con el código fuente original. En caso de que haya habido errores, el reporte de éstos se muestra en la consola y termina el proceso de compilación. Si no hubo errores, el código en ensamblador es procesado por el ensamblador ENS08, generando éste el archivo np.s19 que contiene el código de máquina ejecutable por el MCU del PLM3. Después de esto, se pregunta si la compilación se originó por la opresión del botón <e-pfsil> presente en la ventana principal del SWMANPLM3; si este es el caso, se procede a cargar y ejecutar en el PLM3 el programa implicado. En otro caso, simplemente se avisa al usuario que la compilación fue exitosa.

5. CONCLUSIONES

El lenguaje SIIL2 aquí presentado es funcional para fines del diseño de un amplio espectro de aplicaciones de control lógico. Una motivación, entre otras, para emprender las aventuras técnicas que han sido el desarrollo de los controladores lógicos de tipo PLM y sus lenguajes de programación, fue el mostrar una aplicación práctica funcional, hecha desde cero, de los conocimientos básicos acerca de software y hardware de microcontroladores, que son parte del contenido de asignaturas de Ciencias de la Ingeniería presentes en planes de estudio para la formación de ingenieros.

En esencia, SIIL2 es parte del binomio hardware/software constituido por el PLM3 y el SWMANPLM3. Puede decirse, a grosso modo, que en cuanto al peso en horas hombre requerido en el diseño de estas dos componentes, el hardware requirió alrededor del 20% y por lo tanto, el software requirió un 80% del esfuerzo.

Cabe señalar que la literatura acerca de los PLC está focalizada a explicar como desarrollar aplicaciones de control lógico empleando para ello algún PLC comercial, esto desde el punto de vista de un usuario final de esta tecnología. Por lo regular no se habla de cómo se diseñó el software y hardware implicado, ya que esto es parte del *know-how* propietario de cada empresa fabricante.

Para fines de la programación de los PLC, es más popular el lenguaje de programación de tipo diagrama de escalera (LD), que los lenguajes de tipo textual como lo es SIIL2. Trabajo futuro alrededor del software manejador del PLM3, podría ser el crear un software que genere a partir de un ambiente gráfico tipo LD el código fuente en SIIL2 que corresponda, y de esta forma potencialmente se podría programar el PLM3 bajo un ambiente gráfico de tipo LD.

6. REFERENCIAS

Bolton, W. *PROGRAMMABLE LOGIC CONTROLLERS* (2015). Newnes. Estados Unidos de América.

Dunning, G. *INTRODUCTION TO PROGRAMMABLE LOGIC CONTROLLERS* (2000). Delmar Publishers, ITP. Estados Unidos de América.

Mandado, E. Marcos, J. Pérez, S. (1999). *CONTROLADORES LÓGICOS Y AUTÓMATAS PROGRAMABLES*. 2ª Edición, Alfaomega Grupo Editor. Colombia.

Salvá, A. (1999). *PROGRAMADOR LÓGICO MODULAR*. México D. F., Tesis de maestría, División de Estudios de Posgrado, Facultad de Ingeniería, UNAM.

Salvá, A, L. Altamirano, Erick Dehesa, M. Hernández. (2003). *V-PLM, SOFTWARE ENTRENADOR DE CONTROL LÓGICO*. Congreso AMCA 2003.

Salvá, A, V. Sánchez, L. Salcedo y J. Ramírez (2006). *MANUAL DE USUARIO DEL PLM2*. Reporte del proyecto PAPIME PE101205, patrocinado por la DGAPA UNAM. Archivo muplm2v1.pdf, descargable desde http://dctrl.fi-b.unam.mx/~salva

Salvá, A, F. Guzmán y L. Altamirano (2010). *PLM08, CONTROLADOR LÓGICO PROGRAMABLE BASADO EN EL MICROCONTROLADOR 68HC908GP32*.(2010) MEMORIA DE ELECTRO 2010, realizado en Chihuahua en octubre de 2010.

Salvá, A (2011). *MANUAL DE USUARIO DEL SISTEMA AIDA08*. Archivo muaida08ve2011.pdf, descargable desde http://dctrl.fi-b.unam.mx/~salva

Salvá, A. (2015). *GUÍA BÁSICA DE SIIL2*. Archivo gbsiil2.pdf, descargable desde http://dctrl.fi-b.unam.mx/~salva

Salvá, A, L. Altamirano, L. Álvarez, y A. Herrera (2015). *PLM3, Controlador Lógico Secuencial para Auxilio Didáctico*. Memoria del CNCA AMCA 2015.