Un simulador del lenguaje IL del estándar IEC 61131-3 como apoyo a la asignatura de Automática Industrial

Rodrigo García Puente, César García-Osorio, Angel Peña Peña
Universidad de Burgos
Avda. Cantabría s/n, 09006, Burgos
rodrigo.garcia.puente@gmail.com, cgosorio@ubu.es, appena@ubu.es

Resumen

Los *Autómatas Programables Industriales* o *Controladores Lógicos Programables* (en adelante, PLCs, del término inglés *Programmable Logic Controllers*), son dispositivos electrónicos muy usados en el mundo de la automatización industrial. La programación de los autómatas se realiza con software desarrollados por los fabricantes propios dispositivos. La mayoría de estos entornos son propietarios y requieren el pago de una costosa licencia. Esto supone una dificultad no sólo para la migración de sistemas en la industria sino también para la docencia del funcionamiento de dichos sistemas.

En el año 1990 un grupo de trabajo de la *Comisión Electrotécnica Internacional* (*IEC*) definió el estándar *IEC 61131*, que es el primer paso para la estandarización de los autómatas programables y sus periféricos.

En este recurso docente se presenta un software que permite la edición, validación y simulación de programas para PLCs, en concreto, para programas escritos en lenguaje *Lista de Instrucciones* (*Instruction List*, en adelante, *IL*) dentro del estándar IEC 61131-3, un lenguaje muy parecido al lenguaje ensamblador.

El software desarrollado lleva a cabo el análisis léxico y sintáctico de dicho lenguaje. Además, presenta una interfaz de usuario para presentar los resultados de la simulación de los programas escritos en IL.

1. Introducción

Los PLCs son computadores diseñados específicamente para la tarea de control de procesos industriales discretos y analógicos. Están compuestos por una unidad central de proceso y de entradas y salidas para interactuar con el entorno industrial.

El diseño de los autómatas es modular para poder disponer del número de entradas y salidas necesarias para cada aplicación en particular. Los módulos se conectan a un bus que los comunica con la unidad central de proceso. Existen módulos (cartas) de entradas y salidas tanto digitales como analógicas, dependiendo de la aplicación que se trate. Si el autómata se va a dedicar al control de procesos discretos se añaden módulos de entradas y salidas discretas (8 ó 16 bits). Si el autómata se va a dedicar al control de procesos continuos se añaden módulos de entradas y salidas analógicas. Estos módulos realizan la conversión analógico/digital y digital/analógico.

La programación de los autómatas se realiza en alguno de los múltiples lenguajes que proporciona el fabricante del mismo. Son lenguajes de programación en modo texto o gráficos propietarios del fabricante. También existen una serie de lenguajes de programación especificados en el estándar IEC-61131-3. Este estándar se desarrolló para unificar la programación de autómatas, pues con anterioridad, cada vez que se cambiaba de fabricante había que aprender su lenguaje propietario. Ahora los entornos de programación que vienen con los autómatas soportan los lenguajes del estándar.

1.1. El estándar IEC 61131

El año 1990 un grupo de trabajo de la *Comisión Electrotécnica Internacional* (*International Electrotechnical Comisión*; *IEC*) definió el estándar IEC 61131 [3], que supone el primer paso hacia la estandarización de los autómatas programables y sus periféricos.

Se creó como un compendio y continuación de otros muchos estándares internacionales[1]. A lo largo de la década de los 90 esta norma fue ampliada hasta llegar a ser lo que es hoy, un documento dividido en 5 partes diferentes, incluyendo los lenguajes de programación que se deben utilizar:

- Parte 1: Vista general.
- Parte 2: Hardware.
- Parte 3: Lenguajes de programación.
- Parte 4: Guías de usuario.
- Parte 5: Comunicación.

La parte 3 del estándar, *IEC 61131-3*, es la base para estandarizar los lenguajes de programación en la automatización industrial. El estándar define cinco lenguajes de programación normalizados. Esto significa que su sintaxis y semántica ha sido definida, no permitiendo particularidades distintivas.

El estándar distingue dos lenguajes de tipo textual y otros tres de tipo gráfico:

- Textuales o literales:
  - *Lista de Instrucciones* o *Instruction List* (*IL*): es similar a un lenguaje ensamblador basado en un acumulador simple.
  - *Texto Estructurado* o *Structured Text* (*ST*): es un lenguaje de alto nivel con orígenes en Ada, Pascal y C. Puede ser utilizado para codificar expresiones complejas e instrucciones anidadas.
- Gráficos:
  - *Diagrama Funcional Secuencial* o *Sequential Function Chart* (*SFC*): describe gráficamente el comportamiento secuencial de un programa de control. Ayuda a estructurar la organización interna de un programa, y a descomponer un problema en partes manejables, cada una de las cuales puede se programada en el resto de lenguajes que define el estándar.
  - *Diagrama de Contactos* o *Ladder Diagram* (*LD*): tiene sus orígenes en Estados Unidos y está basado en la representación gráfica de la lógica de relés.
  - *Diagrama de Bloques Funcionales* o *Function Block Diagram* (*FBD*): ampliamente utilizado en Europa y muy común en aplicaciones que implican flujo de información o datos entre componentes de control. Las funciones y bloques funcionales aparecen como circuitos integrados.

En la Figura 1 se puede visualizar un mismo programa representado en 4 de los 5 lenguajes que define el estándar.

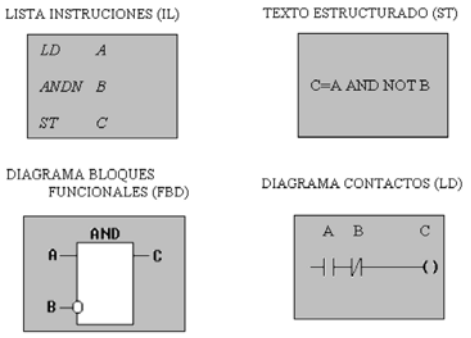

Figura 1. Un mismo programa en los 4 lenguajes del estándar IEC 61131-3..

2. Motivación

El simulador de PLC para el lenguaje IL del estándar IEC 61131-3 se ha desarrollado para las prácticas de la asignatura Automatización y Regulación de Procesos de 3º curso de Ingeniería Técnica Industrial, especialidad Mecánica, de la Universidad de Burgos. Los conocimientos impartidos en esta asignatura son transversales para esta especialidad. El plan de estudios de la titulación no esta diseñado para que sus egresados se dediquen al control de procesos en su vida profesional. En las empresas, las tareas de sintonización de controladores PID y de programación de autómatas son realizadas por los titulados en la rama de Electrónica.

La asignatura Automatización y Regulación de Procesos es una asignatura obligatoria de seis créditos. Dos terceras partes del periodo lectivo de la asignatura se dedican a la teoría de control

[1] IEC 50, IEC 559, IEC 617-12, IEC 617-13, IEC 848, ISO/AFNOR, ISO/IEC 646, ISO 8601, ISO 7185 e ISO 7498.

clásica y la parte restante a la automatización industrial. En esta parte se hace una breve introducción general a la materia y a continuación se dedica a la programación en el lenguaje IL. Los conocimientos impartidos de este lenguaje se limitan a operaciones lógicas, aritméticas y a la programación de contadores y temporizadores. La programación se realiza simultáneamente en el lenguaje textual IL y en el *gráfico de secuencia de bloques funcionales* (*Grafcet*) para que los alumnos adquieran los conceptos de estado y de transición de estado.

La programación en el lenguaje IL se empezó a impartir en el curso 2006-07 sobre papel al no disponer de un entorno de programación que lo soportase. En el curso 2007-08 se ha vuelto a impartir con la misma metodología por que la aplicación no ha estado disponible hasta febrero de 2008 y la asignatura se imparte en el primer cuatrimestre. A partir del curso 2008-09 las prácticas de programación en este lenguaje se realizaran utilizando el software descrito en este artículo.

En el laboratorio de Ingeniería de Sistemas y Automática, área de conocimiento que imparte esta asignatura, se disponen de autómatas programables de la compañía alemana Siemens junto con su entorno de programación Step7 v5.0. Este entorno de desarrollo es muy potente, permitiendo programar en lenguajes textuales como AWL (equivalente al lenguaje IL) y gráficos como KOP (diagramas de contactos) y FUP (diagramas de bloques funcionales).

El lenguaje AWL (ver Figura 2) es bastante críptico por que utiliza nemotécnicos con el menor número de caracteres posible. Por ejemplo, en la nemotécnica alemana, que es la utiliza por defecto, la letra U (UND) representa el operador lógico AND, O (ODER) por el operador lógico OR, E (EINGABE) por INPUT o A (AUSGABE) por OUTPUT [11].

```
U  E 0.0
U  E 0.1
O
U  E 0.2
U  E 0.3
=  A 0.0

U  (
O  E 1.0
O  E 1.1
)
U  (
O  E 1.2
O  E 1.3
)
=  A 1.0
```

Figura 2. Ejemplo de programa escrito en el lenguaje AWL.

El lenguaje AWL también puede utilizar la nemotécnica inglesa, A (AND), O (OR), I (INPUT) o Q (OUTPUT), haciéndolo más intuitivo a las personas con conocimientos de inglés. Sin embargo, el problema persiste por que el código en este lenguaje es difícil de leer por que sus nemotécnicos tienen extremadamente pocos caracteres. Cuando se intenta hacerlo se hace necesario traducir mentalmente los nemotécnicos por el operador que representan. Un programador experto no tiene ningún problema por su habituación al lenguaje pero para un alumno de una especialidad tan ajena a la programación como es la de Mecánica puede dificultar su aprendizaje.

Por el contrario, el lenguaje IL (ver Figura 3) dispone de nemotécnicos cuyo número de caracteres no esta limitado a uno o dos caracteres. Por ejemplo, los nemotécnicos de los operadores lógicos AND, OR y XOR son los mismos operadores. Esta característica del lenguaje IL lo hace fácilmente legible.

```
LD      %I0.0
AND     %I0.1
OR (    %I0.2
AND     %I0.3
)
ST      %Q0.0

LD      %I1.0
OR      %I1.1
AND (   %I1.2
OR      %I1.3
)
ST      %Q1.0
```

Figura 3. Ejemplo de programa escrito en IL (equivalente al de la Figura 1).

Aprender lenguajes de programación estandarizados es una ventaja para los alumnos.

En el caso que durante su vida profesional necesiten programar un autómata, es cada vez más probable que el fabricante soporte total o parcialmente el estándar IEC 61131-3. Allen-Bradley, uno de los más importantes fabricantes de autómatas del mercado, soporta el estándar IEC 61131-3 desde hace varios años. Siemens ha adaptado su lenguaje AWL al estándar IEC 61131-3, denominándolo *STL* (*STatement List*) [9], pero sin cumplirlo estrictamente. STL continúa utilizando los nemotécnicos de AWL aunque en su versión en inglés.

También consideramos que no es necesario un entorno de desarrollo profesional para introducir a los alumnos en la programación de autómatas al nivel requerido en esta asignatura. Con un sencillo programa como el descrito en este artículo, en el que puedan escribir el código, validarlo y ejecutarlo de forma interactiva, es más que suficiente.

3. El simulador de IL

El simulador de IL, *IL Paser Simulator*, se ha programado usando Microsoft® Visual Studio 2005 y el lenguaje C# como entorno de desarrollo y lenguaje de programación respectivamente. Como herramienta para el análisis léxico y sintáctico se ha usado ANTLR[2] [5]; principalmente porque esta herramienta se integra a la perfección con el entorno de desarrollo y porque es una herramienta de análisis en continua evaluación.

Se ha diseñado la aplicación de manera que fuese similar a otros entornos de desarrollo basados en el sistema Windows (ver Figura 4). Todas las áreas que componen el entorno así como sus elementos son sensibles al contexto en el que se usan, esto quiere decir que los elementos de la pantalla se activarán o desactivarán en función de la posibilidad de uso de los mismos.

A continuación se dan más detalles de todos los elementos de la aplicación.

3.1. Barra de menús

Situada en la parte superior de la ventana, permite el acceso a todas las opciones de la aplicación a través de varios menús que agrupan la funcionalidad de la aplicación (ver Figura 5).

Se han incorporado 5 menús: Archivo, Editar, Ver, Depurar y Ayuda. Algunas de las opciones de menú disponen de códigos de atajos de teclado para ejecutar la opción directamente sin acceder al menú. De igual manera, la barra de herramientas permite también el acceso a las acciones más usuales.

3.2. Barra de herramientas

Situada justo debajo de la barra de menús, permite el acceso rápido a las funciones más usuales de la aplicación a través de una serie de iconos.

Permite el acceso rápido a las siguientes opciones: Nuevo proyecto o archivo, Abrir proyecto, Abrir proyecto reciente, Cerrar proyecto, Guardar, Guardar todos los archivos, Cortar, Copiar, Pegar, Deshacer y Rehacer última acción sobre el texto, Validar programa, Simular programa, Continuar ejecución, Detener ejecución, Siguiente instrucción, Establecer o quitar punto de interrupción y Salir.

En la Figura 6 se ve la barra de herramientas de la aplicación.

3.3. Árbol de proyecto

Situado en la parte lateral izquierda de la pantalla principal, muestra la estructura de ficheros del proyecto de programa que esté abierto (si es que está abierto algún proyecto).

Cada proyecto consta de un archivo de programa principal (extensión **`.plcm`**), el acceso a la configuración de simulación y luego tres carpetas que pueden contener de cero a *n* archivos, categorizados en:

- Funciones (Archivos *Function*, con extensión **`.plcf`**),
- Bloques de función (Archivos *Function Block*, con extensión **`.plcb`**)
- Texto (Archivos *Text*, con extensión **`.plct`**).

Toda la información del proyecto se almacena en el un archivo de proyecto (extensión **`.plcp`**). En la Figura 7 (izquierda) se puede ver el árbol de un proyecto de ejemplo.

[2] ANTLR es una herramienta que permite generar analizadores o intérpretes para un lenguaje dado. Es lo que se denomina un compilador de compiladores.

3.4. Página de propiedades

Situada en la parte inferior izquierda de la pantalla, bajo el árbol de proyecto, permite la visualización y edición de las propiedades (metadatos) de los archivos del proyecto.

Únicamente se permiten editar las propiedades nombre y descripción. Si no está seleccionado ningún archivo del proyecto, la página de propiedades se muestra vacía. Ver ejemplo en la Figura 7 (derecha).

3.5. Área de edición del programa

Situada en el centro de la pantalla, es el elemento que más área de la pantalla ocupa.

Permite la edición de los archivos de programa que componen el proyecto. Cada uno de los archivos de programa abiertos se representan dentro de una "pestaña" diferente. La activación de una pestaña produce la selección automática del archivo en el árbol de proyecto (y, en consecuencia, la visualización de sus propiedades en la página de propiedades).

Durante la edición de un archivo el usuario puede seleccionar porciones de texto y llevar a cabo las típicas operaciones de copiado, cortado y pegado. Estas accesibles a través del menú edición o con los típicos atajos de teclado. También, es posible deshacer y rehacer las operaciones de edición y realizar búsquedas.

3.6. Paneles

Situados bajo el área de edición (Figura 8), se divide en 4 subpaneles:

- *Análisis y resultados*: Dónde se muestran los mensajes del proceso de validación y se muestran mensajes de error en caso que estos se produzcan.
- *Examinador de variables*: Es el subpanel más interesante. En él se muestran los cambios de valor en las variables, que tienen lugar durante el proceso de simulación. Asimismo, se muestran los bloques de entrada, de salida y de memoria. El usuario puede utilizar los *cuadro de validación* (*check boxes*) en estos bloques para simular las distintas entradas al PLC o forzar determinadas situaciones durante la simulación. Para configurar el número de bloques de entrada y de salida, y el número de bits que tendrá cada bloque, se debe acceder a la "Configuración de la simulación" (Figura 9) del programa en el menú "Depurar" (Figura 5). Por defecto, cuando se crea el proyecto, se establecen unos parámetros por defecto, que son tres bloques, uno de entrada, otro para salida y otro para memoria, y, 8 bits por bloque.
- *AST*: En este subpanel aparece el *árbol de sintaxis abstracta* (*Abstract Syntax Tree*) del programa IL analizado.
- *Break Points*: Muestra la lista de *puntos de interrupción* que están establecidos en el programa con la línea y la instrucción a la que corresponde el punto de interrupción. Cuando el programa está siendo simulado y se alcanza un punto de interrupción en el programa, la ejecución queda suspendida hasta que el usuario tome una decisión: pasar a la siguiente instrucción, continuar o parar. Se pueden eliminar todos los puntos de interrupción del programa directamente accediendo a la opción "Depurar > Eliminar" todos los puntos de interrupción, previa confirmación serán eliminados todos los puntos de interrupción del programa.

3.7. Barra de estado

Situada en la parte inferior de la pantalla, muestra información adicional. Está dividida en tres secciones, la primera de ellas muestra una barra de progreso cuando hay algún proceso en ejecución; la segunda muestra la línea en la que está situado el cursor dentro de un archivo de programa y la tercera muestra mensajes hacia el usuario.

4. El proceso de simulación

El primer paso será abrir un proyecto ("Archivo > Abrir proyecto"). Una vez abierto hay que validar el programa ("Depurar > Validad programa"). Si se produce errores se mostrarán en el subpanel de análisis de resultados. En caso contrario, estamos en disposición de iniciar la simulación. Existen dos modos principales de simulación, la ejecución paso a paso, o ejecución indefinida.

El usuario puede establecer puntos de ruptura para detener la ejecución de la simulación y poder consultar los valores de las variables o simular la modificación de las entradas al PLC. Para reanudar la ejecución puede decidir continuar

paso a paso o continuar hasta el siguiente punto de ruptura.

Los puntos de ruptura puede establecerse en cualquier momento del proceso de ejecución en el que la simulación este detenida. También es posible deshabilitar puntos de control previos o desactivar la totalidad de los puntos de control.

5. Conclusión

Se ha presentado una aplicación que permite la simulación de programas escrito en IL. Este simulador permite el aprendizaje de un lenguaje de programación de PLCs independientemente del fabricante. Que sepamos, la única aplicación de similares características es MatPLC[3] [10], pero ni se trata de un proyecto orientado a la docencia, ni presenta una interfaz gráfica.

Debido a la reciente finalización de la aplicación presentada en este artículo, todavía no se ha empezado a utilizar en las prácticas. Dado que en este momento todos los ejercicios de programación en IL se hacen exclusivamente con "lápiz y papel" parece incuestionable la gran ventaja que supondrá la utilización de esta herramientas. Los beneficios que aportará este simulador están avalados por el éxito de otras herramientas similares utilizadas como recurso docente [1, 2, 4, 6, 7, 8].

En el futuro esperamos ampliar la aplicación con la inclusión del lenguaje ST. La parte de declaración de datos es común con IL y ya esta presente en la actual versión, con lo que el primer paso en esta dirección ya esta dado.

Otra idea de desarrollo futuro es la inclusión de alguno de los lenguajes gráficos. Estos lenguajes requieren la traducción a IL como paso previo a su simulación, con lo que la funcionalidad desarrollada para la presente versión de la aplicación será punto de partido indispensable.

La herramienta descrita en este artículo puede descargarse de la siguiente página web: http://pisuerga.inf.ubu.es/cgosorio/ILsimulator/.

Referencias

[1] Concheiro, R., Loureiro, M., Amor, M., y González, P. Simula3MS: simulador pedagógico de un procesador. *XI Jornadas de Enseñanza Universitaria de Informática*, 489-495, Villaviciosa de Odón (Madrid), 2005.

[2] Jáuregui Vidal, A., Ruiz Vázquez, T., Etxeberria Uztarroz, I. Simulador de una unidad de control microprograma. *X Jornadas de Enseñanza Universitaria de Informática*, 501-504, Alicante, 2004.

[3] John, K.-H., Tiegelkamp, M. *IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Aids to Decision-Making Tools*. Springer, 2001.

[4] Moreno, L., González, E.J., Popescu, B., Torres, J., Toledo, J., González, C.. Simuladores de jerarquía de memoria en el contexto de un proceso de investigación-acción. *XIII Jornadas de Enseñanza Universitaria de Informática*, 393-400, Teruel, Julio 2007.

[5] Parr, T. *ANTLR Reference Manual. 2.7.7,* (http://www.antlr.org/, última visita 19/feb/2008).

[6] Perles, A., Martínez, J.M., Asan, H., Alvadalejo, J., Domínguez, C. El Simulador SimSenv en la Innovación Docente de la Asignatura Informática Industrial. *VII Jornada de Enseñanza Universitaria de la Informática*, 166-172, Palma de Mallorca, 2001.

[7] Prieto, M., Vicente, A.J., Vargas, J.A. Simulador de dispositivos de entrada/salida programables. *VIII Jornadas de Enseñanza Universitaria de Informática*, 595-598, Cáceres, 2002.

[8] Rodríguez Álvarez, M., Gómez García, A.M., Mesas García, P., Ruiz Núñez, J.I., Prieto, A. Desarrollo de un simulador de programación del microprocesador Intel 8085. *IX Jornadas de Enseñanza Universitaria de Informática*, 577-580, Cádiz, 2003.

[9] *SIMATIC - Statement List (STL) for S7-300 and S7-400 Programming*, Siemens, 2006. (http://support.automation.siemens.com/WW/llisapi.dll/csfetch/18653496/AWL_e.pdf?func=cslib.csFetch&nodeid=18653995&forcedownload=true, última visita 19/feb/2008).

[10] de Sousa, M., Carvalho, A. An IEC 61131-3 compiler for the MatPLC. IEEE Conference

[3] http://mat.sourceforge.net/.

on Emerging Technologies and Factory Automation, 1: 485-490, 2003.

[11] *TIA Manual de formación para soluciones generales en automatización*, Siemens, 2001 (http://www.automation.siemens.com/fea/ftp/module_sp/a08/a08_online.pdf, última visita 19/feb/2008).

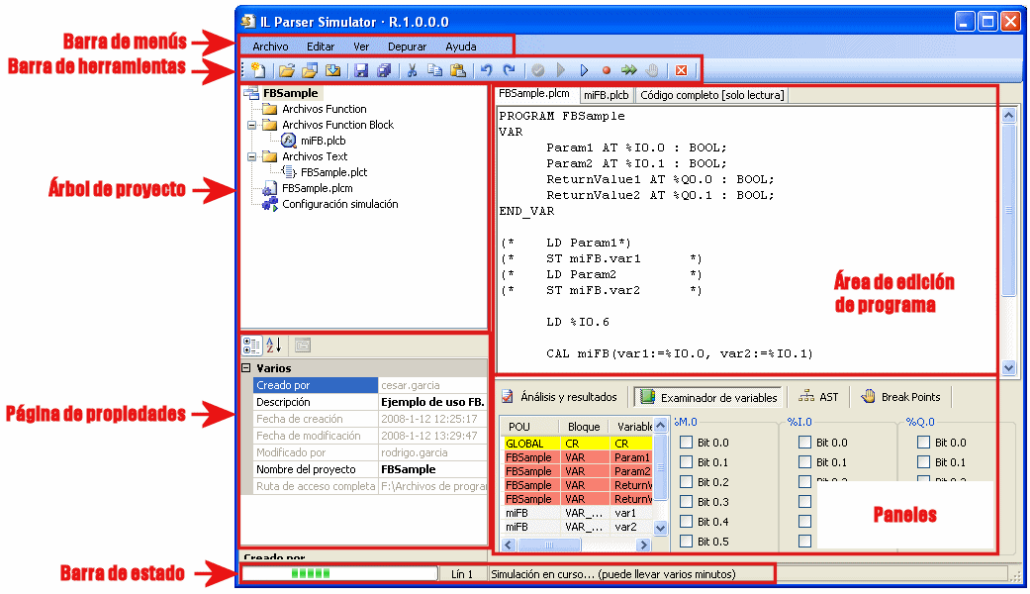

Figura 4. La ventana de la aplicación *IL Parser Simulator*.

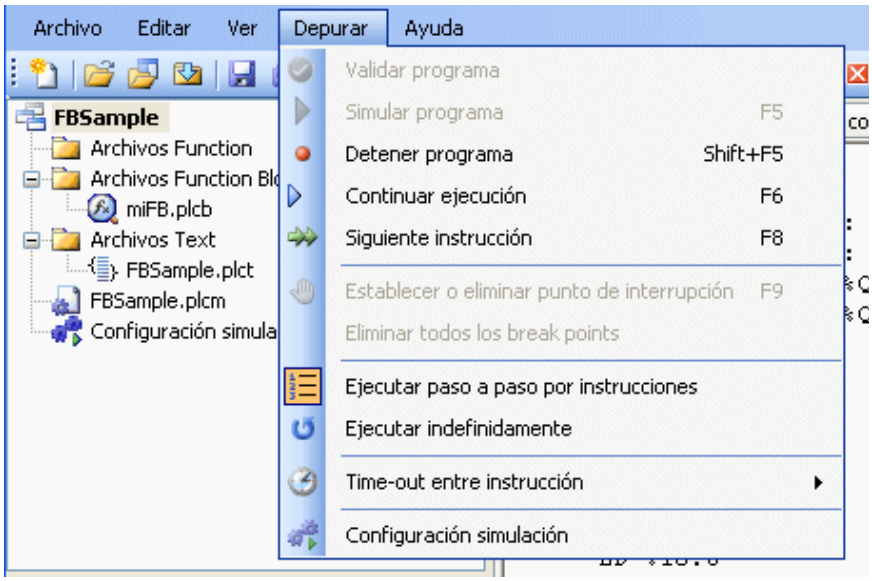

Figura 5. Barra de menús, con el menú "Depurar" desplegado.

Figura 6. Barra de herramientas.

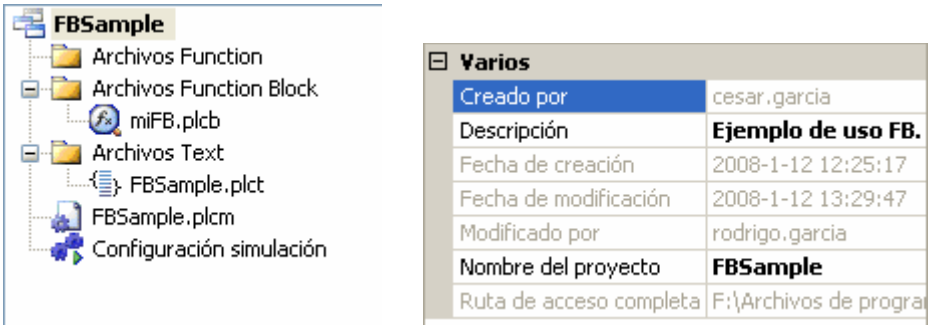

Figura 7. Izquierda: Árbol de proyecto. Derecha: Página de propiedades

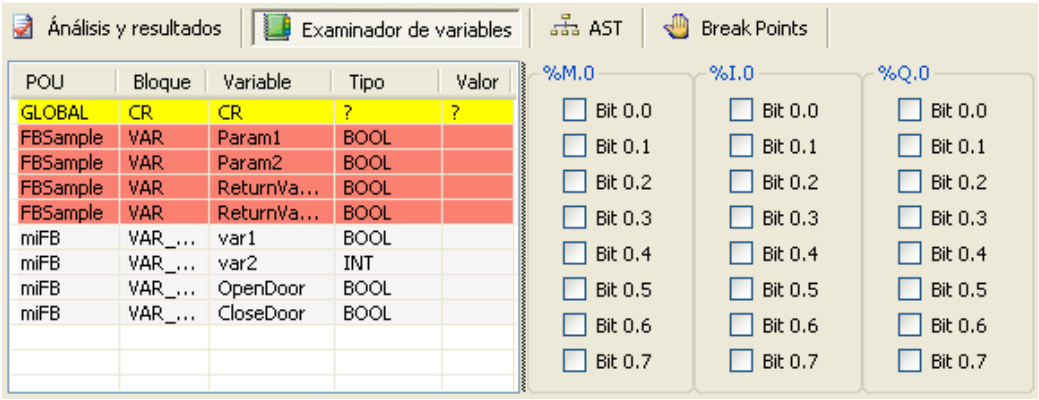

Figura 8. Izquierda: Árbol de proyecto. Derecha: Página de propiedades

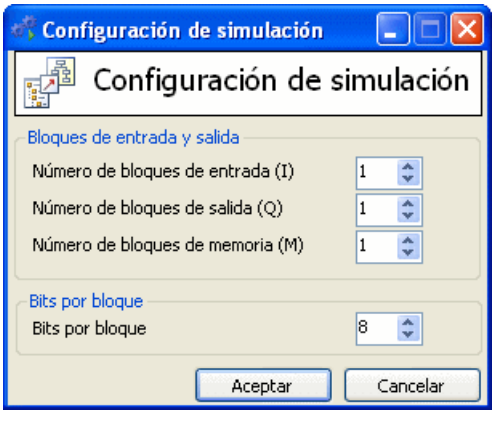

Figura 9. Configuración de la simulación.