PLCopen®
for efficiency in automation

Desarrollando la estructuración de un programa con IEC 1131-3

Direcciones generales

El papel del software ha cambiado. Cada vez toma mayor importancia en la calidad del producto. Los errores de software tienen efectos dramáticos, a menudo pueden llegar a arruinar todo el dinero invertido en investigación y desarrollo. Los requerimientos de la industria de control han crecido, extendiendo los códigos de software de 100 líneas a los 10.000 actuales. Esto no sólo los hace más propensos a los errores si no que testarlos al 100% se hace prácticamente imposible. La creación de este tipo de software no es ya el trabajo de un solo hombre: el programador convencional forma parte, ahora, de un equipo multidisciplinar.
Con el constante incremento de los requerimientos, instalación, mantenimiento, mayores cualificaciones y mejoras se han convertido en una parte esencial de la vida de los ciclos de control, y los bloques de software han tomado un papel preponderante.

Introducción

Los métodos de programación modernos proveen de herramientas para mejorar la calidad intrínseca del software, por ejemplo, su correcto funcionamiento en el sentido de realizabilidad, robustez, integridad, persistencia y seguridad. La norma internacional IEC 1131-3 mejora ampliamente como herramienta el reparto de la programación, la instalación y las fases de mantenimiento de proyectos de desarrollo de software en la industria de control.
Básicamente, la IEC 1131-3 consiste en dos partes: Elementos Comunes y Lenguajes de Programación.
Las herramientas de estructuración con la IEC 1131-3 están centradas en los elementos comunes, aunque claramente enlazan con los lenguajes de programación que se necesitan.
Este artículo muestra que utilizando la IEC 1131-3 de forma consistente, uno genera códigos de software comprensibles, reutilizables, verificables, y fáciles de mantener.

La esencia de la estructuración

Las líneas arriba mencionadas requieren una aproximación diferente. Un enfoque en estructuras provee de ventajas como:

- Una mejor visión global del sistema, no sólo importante para los programadores originales, sino también para la instalación y el personal de mantenimiento.
- Una mejor base para la comunicación interna del multidisciplinario equipo de desarrollo.
- Una base para la reutilización del software.
- Documentación inherente /automática.

En una sobrevista, la estructuración es hecha de modo que se divide el problema en partes más pequeñas, las cuales pueden ser sub-divididas. Existen límites para esto: no es práctico continuar dividiendo hasta las últimas consecuencias, porque el esfuerzo entonces se orientaría después hacia la integración de esas partes.
Con la IEC 1131-3 existen dos principios que cooperan, por motivos de claridad las llamamos:

- Modularidad
- Descomposición

Principios de Modularidad

Con los modernos métodos de desarrollo de software existen 5 principios asociados a la modularidad. Estos son:

1. El lenguaje de programación debería apoyar las unidades modulares
2. Las unidades deberían estar compuestas en tal parte / número que ellas tengan pocos interfaces y pocas interacciones.
3. Los interfaces deberían ser pequeños, necesitando pocos intercambios de datos.
4. El módulo de interacciones requiere una definición explícita, para incrementar su re-utilizabilidad.
5. Los módulos deberían mejorar la encapsulación de datos: los datos de aplicación son particionados, y cada partición sólo es accesible a través de un set de funciones las cuales estarían ocultas para usos indeseados.

Para fomentar esto IEC 1131-3 ha definido las Unidades de Organización de Programa, (Program Organisation Units, POU´s) consistentes en:

- Funciones
- Bloques Funcionales
- Programas

Estos serán explicados con mayor detalle más abajo.

Funciones

Todos nosotros conocemos las funciones como ADD, SQUARE ROOT, SIN, COS, GREATER THAN, etc. IEC tiene un enorme juego de ellas definidas. En suma, tú también puedes crear tus propias funciones como esta simple función que mostramos:

FUNCION SIMPLE_FUN : REAL

VAR_INPUT

A, B : REAL;

C : REAL := 1.0;

END_VAR

SIMPLE_FUN := A*B/C;

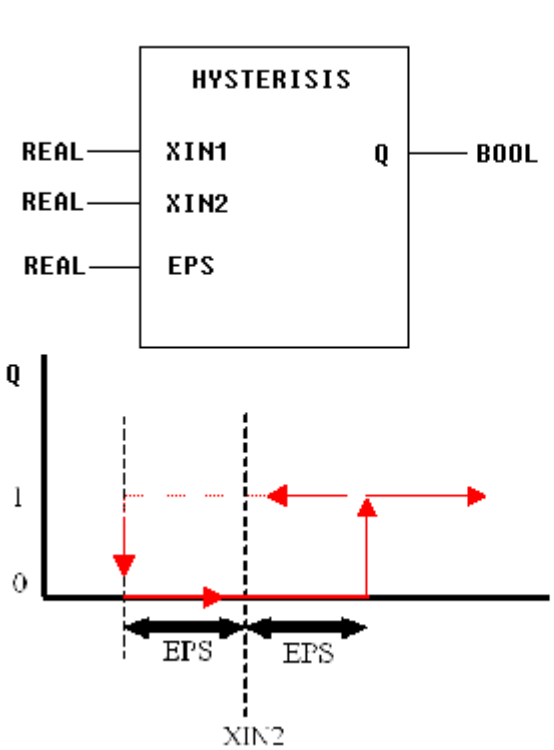
END FUNCTION

Una vez definida podrás utilizarla una y otra vez, en el mismo programa, en diferentes programas o en cualquier otro proyecto.

Bloques Funcionales, FB´s

Lo mismo es válido para los bloques funcionales: existen bloques funcionales normalizados y FB´s añadidos por el proveedor. Tú también puedes crear tus propios bloques funcionales y añadirlos a tu propia librería de bloques funcionales. Todos esos bloques funcionales son altamente re-utilizables en el mismo programa, nuevos programas o distintos proyectos. También puedes usarlos con algún otro lenguaje de programación IEC, dándote una clara separación entre los diferentes niveles de programadores o personal de mantenimiento.

Esta re-utilizabilidad incrementa tu eficiencia, y reduce el número de errores.
Vamos a ver un ejemplo:




```
FUNCTION_BLOCK HYSTERISIS
VAR_INPUT
   XIN1,  XIN2 :  REAL;
   EPS  :  REAL;   (* HYSTERISIS:
END_VAR
VAR_OUTPUT
   Q  : BOOL  := 0
END_VAR
IF Q THEN
  IF XIN1 < (XIN2-EPS) THE]
    Q  := 0 (XIN1 + DECREAS:
 END_IF;
ELSIF XIN1 > (XIN2 + EPS )
 Q := 1; ( * XIN1  INCREASIN
 END_IF;
END_FUNCTION_BLOCK
```

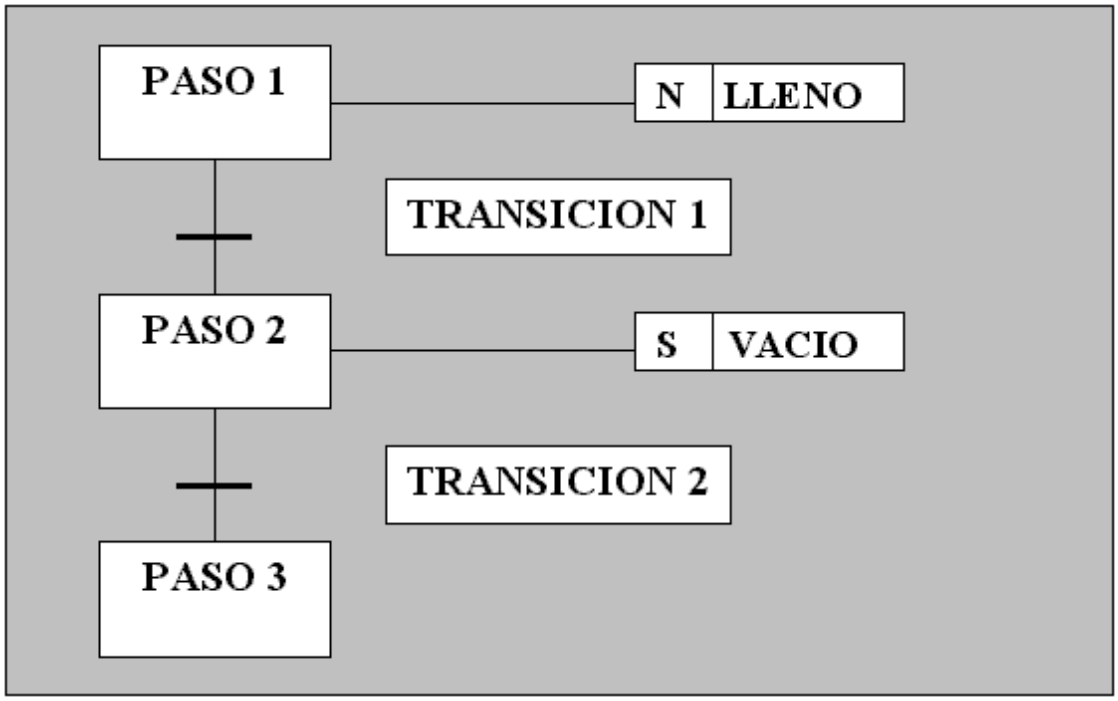
El Bloque Funcional arriba mostrado, (en el lado izquierdo), está representado aquí en el lenguaje de programación de Diagramas de Bloques Funcionales. El Bloque Funcional tiene el nombre de Histéresis, Tiene tres entradas a la izquierda, denominadas XIN1, XIN2 y EPS, todas del tipo de variable real. Tiene una salida, a la derecha, llamada Q, del tipo variable booleana (BOOL). Internamente, el FB contiene un código de programa, como el mostrado en el lado derecho. En este ejemplo, el código está escrito en Texto Estructurado, ST. La primera parte configura los datos de la estructura, la segunda parte con el algoritmo, usa las entradas, realiza los cálculos, y modifica las salidas. El algoritmo está oculto para el usuario del Bloque Funcional, quien sólo ve la funcionalidad del Bloque como se muestra en la izquierda. Esto crea un diferente nivel de acceso, mostrando la encapsulación como nos referíamos en el punto 5 de los principios de modularidad.
Los nombres usados en el bloque funcional son locales al FB. No hay problema si el nombre es usado en un FB para datos locales, no habrá conflicto si el mismo nombre es usado de diferente manera en otro Bloque Funcional o en algún otro lugar del programa.

Programas: interacción jerárquica

Con estas funciones y bloques funcionales, tú puedes abordar el programa como una interacción de estos bloques de construcción básicos.
De este modo complejos programas pueden ser descompuestos en bloques funcionales, los cuales pueden ser de nuevo descompuestos en bloques funcionales más pequeños. Esto te ayuda a incrementar tu eficacia.

Descomposición: ¿cómo hace que se vea en la IEC 1131-3?

Como herramienta de descomposición, IEC 1131-3 provee de Cartas de Función Secuencial (Sequential Function Charts, SFC). Las SFC describen gráficamente el comportamiento secuencial de un programa de control. En este modo estructura la organización interna de un programa descomponiendo el problema de control en partes manejables, mientras se mantiene una visión de conjunto. Esto lo hace muy apto en la realización de diagnósticos.
Las SFC consisten en pasos, enlazados con bloques de acción y transiciones (ver figura de la página siguiente).
Cada paso representa un estado del sistema. Los pasos están enlazados con las acciones, realizando un control cierto de la acción.
Una transición está unida a una condición, la cual, cuando es cierta, causa el paso previo a la desactivación y el paso siguiente a la activación.
Cada bloque de acción o transición puede ser programado en alguno de los lenguajes IEC, Diagrama de Contactos, Diagrama de Bloques Funcionales, Lista de Instrucciones, y Texto Estructurado, y cualquier inclusión de SFC a sí mismas, para una ulterior descomposición.



Las SFC apoyan las secuencias alternativas y paralelas, tal como son requeridas en las aplicaciones batch. Por poner un caso, una secuencia es usada para el proceso primario y una segunda monitorea las acciones de operación general del programa. Y esto ocurre con la misma sobre visión y estructura.

Estructurando: 7 pasos a seguir

La estructuración con el IEC1131-3 está presentada aquí como la combinación de la Modularidad y la Descomposición. Los 7 pasos siguientes mejoran el camino a seguir para la estructuración del software:

1. Identificación de los interfaces externos al sistema de control.
2. Definición de las principales señales intercambiadas entre el sistema de control y el resto de la instalación.
3. Definición de todas las interacciones del operador, prioridades y datos de supervisión.
4. Análisis de la descomposición del problema de control de nivel superior en particiones lógicas.
5. Definición de las POU´s, por ejemplo, Programas y Bloques Funcionales.
6. Definición de los requerimientos del tiempo del ciclo de escaneo para las diferentes partes de la aplicación.
7. Configuración del sistema por definición de programas fuente, enlazando programas, con entradas y salidas físicas y asignando Programas y Bloque Funcionales a las tareas.

La IEC 1131-3 te ayuda especialmente en los últimos pasos 4-7, donde ocurre la translación a software.

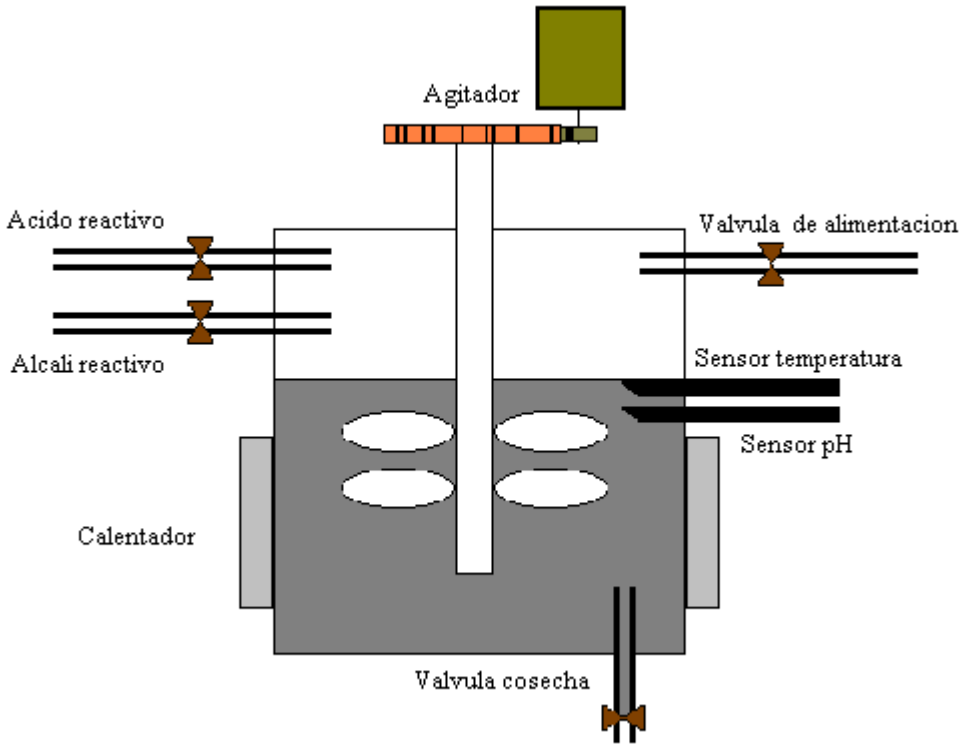
Además de estos 7 pasos, existen algunos principios más que deberían ser usados para optimizar el método de estructuración:
Estos principios son:

- Trabajo puramente simbólico: sin direccionamiento absoluto (esto sólo en la parte de declaración). Ventajas: fácil adaptación a los cambios del entorno, mayor nivel de re-utilización del código, menores efectos colaterales.
- Las partes de programa pertenecientes a otra deberían ser unidas también en el código fuente.
- No usar saltos: Ventajas: mayor transparencia, mayor nivel de re-utilización, reducción de efectos colaterales.
- Nombrar adecuadamente las variables y los bloques funcionales incrementa la transparencia y la lectura global.

Un ejemplo: Sistema de control de fermentación

(cortesía de Omron Electronics)

Un ejemplo dice más que mil palabras..., así que vamos a ver un proceso de fermentación y su control, como se muestra más abajo.



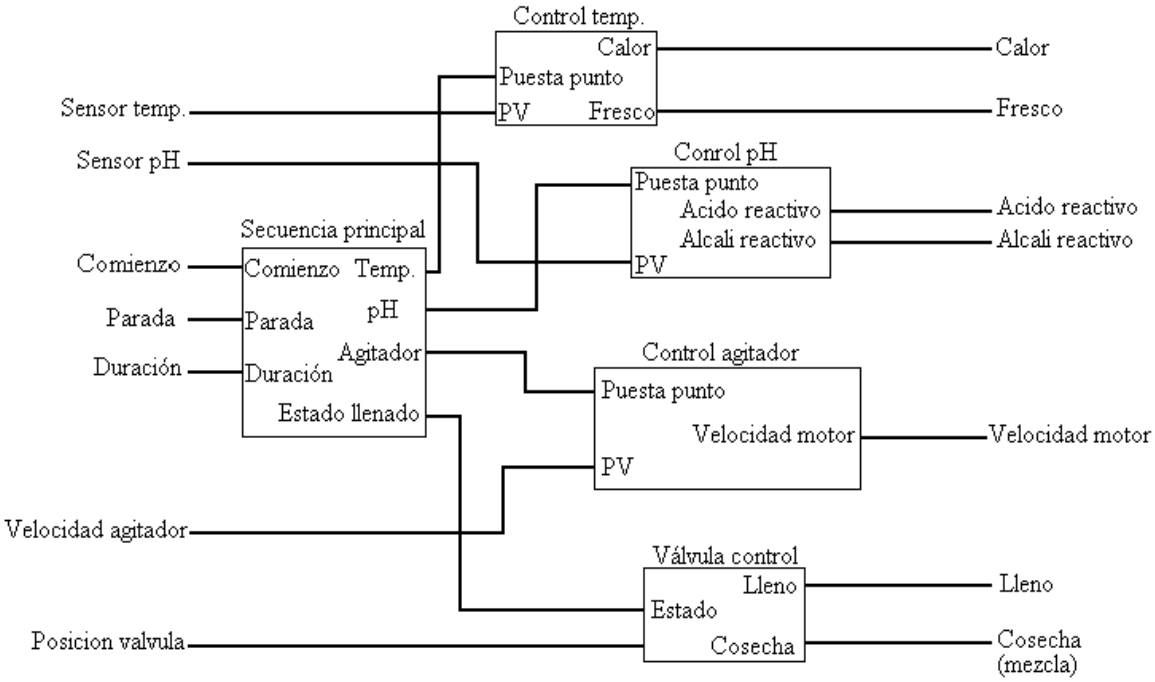
Todos los interfaces externos están definidos aquí (paso 1). Hay un gran recipiente, que puede llenarse (válvula de alimentación) con líquido, puede calentarse con el calentador (enfriamiento por

convección), puede agitarse con un motor, y donde el ácido y el álcali pueden ser añadidos dentro del recipiente.
Observando el paso 4, que se refería al análisis del problema de control de nivel superior para descomponerlo en particiones lógicas, podemos identificar fácilmente 5 funciones:

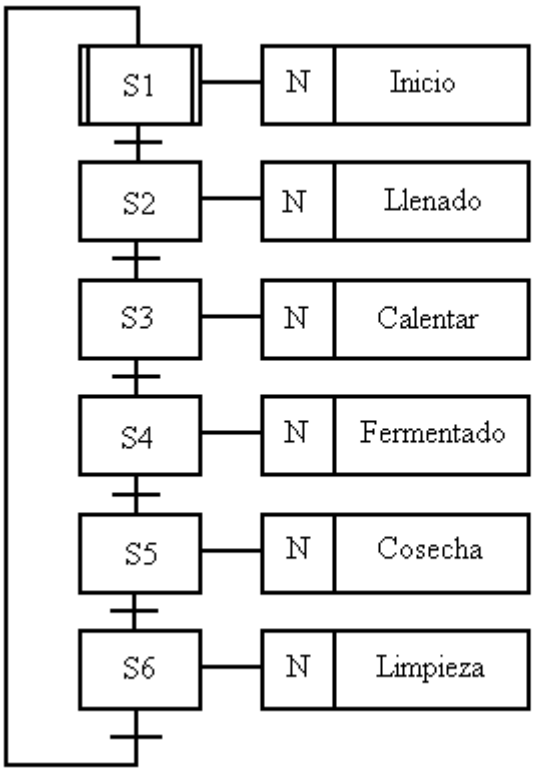
1. **Secuencia principal**, ej. Pasos de proceso de nivel superior – llenado, calentamiento, agitación, fermentación, cosecha del producto, y limpiado.
2. **Válvula de control**, ej. La operación de las válvulas usadas para llenar y vaciar el depósito.
3. **Control de temperatura** para monitorizar la temperatura del depósito de calentamiento.
4. **Agitador de control** para activar el motor del agitador según demande la secuencia de proceso principal.
5. **Control de pH** para monitorizar la acidez de los fluidos de fermentación, añadiendo reactivo ácido o álcali según sea necesario.

Paso 5: definición de las POU´s requeridas, por ejemplo, Programa y Bloques Funcionales.

Presentando éstas en el Diagrama de Bloques Funcionales del lenguaje de programación, la visión global del programa de control de fermentación podría parecerse a éste: (Lea de izquierda a derecha. A la izquierda están las entradas; a la derecha las salidas).



Si observamos únicamente la secuencia principal, nosotros podríamos estructurarla con Cartas de Función Secuencial, SFC, tal como sigue:



Nosotros empezamos arriba con la inicialización: sin conocer el estado del sistema, cuando lo encendemos, nosotros debemos testar el estado de las válvulas, etc.
Entonces nosotros comenzamos el llenado hasta el nivel requerido.
La siguiente fase consiste en el calentamiento hasta que comience el proceso de fermentación.
Entonces, nos movemos a la siguiente fase: la parte del control del proceso de fermentación.
Después de que se haya completado, nosotros recogemos el fluido fermentado y limpiamos el depósito, estando listos para reiniciar arriba de nuevo.
Esta descomposición ofrece a todos una clara visión de las secuencias involucradas, para una ulterior modularización en Bloques Funcionales que pueden ser programados en alguno de los cuatro lenguajes.
Dicho de otro modo: ¡el requerimiento de especificación del usuario está prácticamente hecho!.
El trabajo de programación que debe hacerse ahora es en al nivel de los bloques de acción. Aquellos deberían estar divididos entre diferentes personas, con diferente formación. Para esto, la norma IEC define dos lenguajes de programación gráficos y dos textuales, Lista de Instrucciones, Texto Estructurado, Diagrama de Contactos y Diagrama de Bloques Funcionales, para suplir de la mejor manera las necesidades y el problema a tratar. También una descomposición adicional de los bloques de acción puede ser hecha por medio de las SFC, si es necesario.
El sistema de desarrollo te ayudará en los dos pasos finales:

6. Definición de los requerimientos del tiempo del ciclo de escaneo para las diferentes partes de la aplicación.
7. Configuración del sistema definiendo sus fuentes, enlazando programas con las entradas y salidas físicas y asignando programas y bloques funcionales a las tareas.

Conclusión

El proceso de desarrollo de software ha cambiado:

- Más requerimientos
- Más funcionalidad
- Programas más largos
- Más gente involucrada
- ... mayores requerimientos /deseos

Estructuración, Modularidad y Descomposición son elementos esenciales en el desarrollo del software moderno y la IEC 1131-3 ofrece la base correcta para un pleno desempeño de tus requerimientos.

Eelco van der Wal, Director General de PLCopen (evdwal@plcopen.org)

Documento traducido para PLCopen por el miembro educacional:

Área de Ingeniería de Sistemas y Automática
Universidad de Oviedo
Campus de Viesques s/n
33204 - Gijón
España