IEC 1131-3: UNA FUENTE DE PROGRAMACION STANDARD

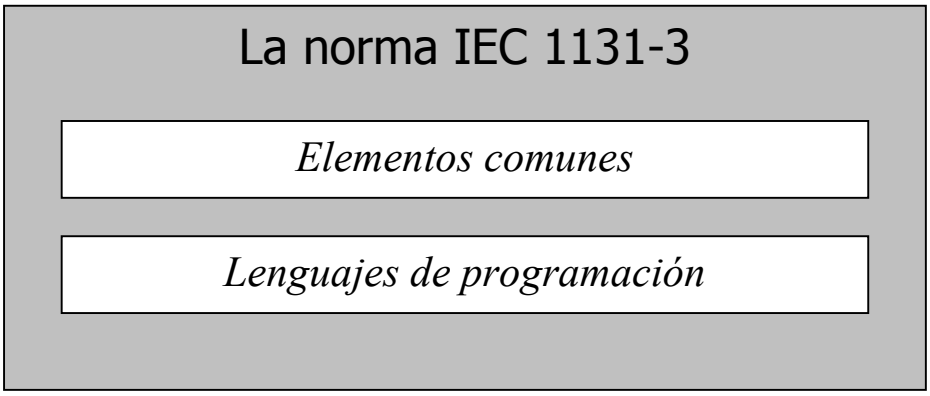
IEC 1131-3 es el primer intento real de estandarizar los lenguajes de programación en la automatización industrial, haciendo el trabajo independiente de cualquier compañía.

IEC 1131-3 es la tercera parte de la familia IEC 1131 que se divide en cinco partes:
- Parte 1: Vista general.
- Parte 2: Hardware.
- Parte 3: Lenguaje de programación.
- Parte 4: Guías de usuario.
- Parte 5: Comunicación.

Hay muchas maneras de enfocar el trabajo desarrollado en la tercera parte de la norma, indicaremos algunas de ellas:
- Los resultados de la Parte 3 en Lenguajes de Programación, dentro del estándar IEC TC65 SC65B
- El resultado del gran esfuerzo realizado por 7 multinacionales a los que se añaden 10 años de experiencia en el campo de la automatización industrial.
- 200 páginas de texto aproximadamente, con 60 tablas incluyendo hojas de características.
- Las especificaciones de la sintaxis y semántica de un lenguaje apropiado, que incluya una definición completa del modelo de software y una estructura del lenguaje.

Otra visión distinta es dividir la norma estándar en dos partes: (ver figura 1):

1. Elementos comunes.
2. Lenguajes de programación.



Vamos a ver en detalle cada una de estas partes:

ELEMENTOS COMUNES:

Introducción de datos

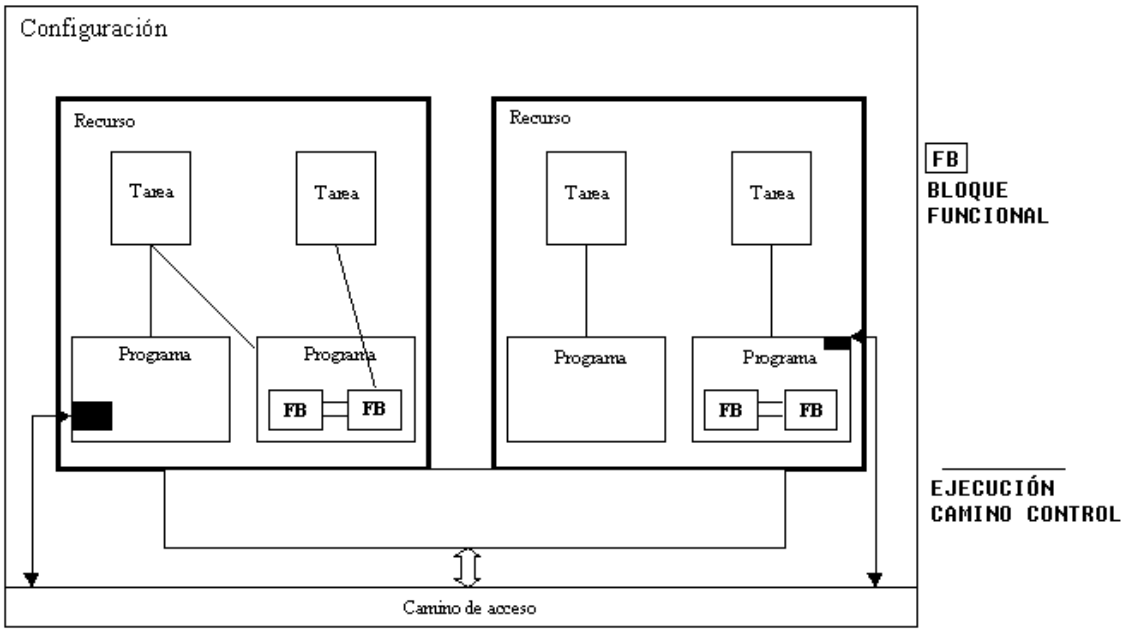
En los elementos comunes se definen los tipos de datos. La determinación del tipo de datos previene de errores en una fase inicial, como por ejemplo la división de un dato tipo fecha por un número entero.
Los tipos comunes de datos son: variables booleanas, número entero, número real, byte y palabra, pero también fechas, horas del día y cadenas (strings).
Basado en estos tipos de datos cada uno puede definir sus propios tipos de datos, conocidos como tipos de datos derivados. De este modo, se puede definir por ejemplo un canal de entrada analógica como un tipo de dato, y usar éste una y otra vez.

Variables

Las variables sólo están asignadas a direcciones de hardware explícitas (ej.: entradas y salidas) en configuraciones, recursos o programas. De este modo se crea un alto nivel de independencia con el hardware, favoreciendo la flexibilidad del software.
La extensión de las variables está normalmente limitada a la unidad de organización en la cual han sido declaradas como locales.
Esto significa que sus nombres pueden ser reutilizados en otras partes sin conflictos, eliminando una frecuente fuente de errores.. Si las variables debieran tener una extensión global, deberían ser declaradas como globales utilizando la palabra reservada VAR_GLOBAL.
Pueden ser asignados parámetros y valores iniciales que restablecen al inicio, para obtener la configuración inicial correcta.

Configuración, recursos y tareas

Para entender esto mejor, vamos a ver el modelo de software, que se define en la norma (ver figura).



Al más alto nivel, el elemento software requerido para solventar un problema de control particular puede ser formulado como una configuración. Una configuración es específica para un tipo de sistema de control, incluyendo las características del hardware: procesadores, direccionamiento de la memoria para canales de I/O y otras capacidades del sistema.
Dentro de una configuración, se pueden definir uno o más recursos. Se puede entender el recurso como el procesador capaz de ejecutar programas IEC.
Con un recurso, pueden ser definidas una o más tareas. La ejecución del control de tareas de un grupo de programas y/o bloques de función.
Cada una de éstas puede ser ejecutada periódicamente o por una señal de disparo especificada, como el cambio de estado de una variable.
Los *programas* están diseñados a partir de un diferente número de elementos de software, escrito en algunos de los diferentes lenguajes IEC definidos.
Típicamente, un programa es una interacción de *Funciones* y *Bloques Funcionales,* con capacidad para intercambiar datos. Funciones y Bloques Funcionales, son las partes de construcción básicas, conteniendo una estructura de datos y un algoritmo.

Vamos a comparar esto con un PLC convencional: éste contiene un recurso, ejecutando una tarea, controlando un programa, ejecutándose en un lazo cerrado. IEC 1131-3 añade mucho a esto, abriéndolo al futuro. Un futuro que incluye multi-procesamiento y dirección de eventos de programas. Y este futuro no está muy lejos: viendo simplemente los sistemas distribuidos o los sistemas de control de tiempo real. IEC 1131-3 está disponible para un amplio rango de aplicaciones, sin tener que aprender lenguajes de programación adicionales.

Unidades de Organización de Programa

Con la IEC 1131-3, los Programas, Bloques Funcionales y Funciones son llamadas Unidades de Organización de Programas, POU´s.

Funciones

IEC ha definido las **funciones estándar** y **funciones definidas por usuario**. Las Funciones estándar son utilizadas para ADD (suma), ABS (valor absoluto), SQRT (raíz cuadrada), SIN (seno), y COS (coseno). Las funciones definidas por usuario, una vez definidas, pueden ser usadas una y otra vez en el programa.

Bloques Funcionales, FB´s

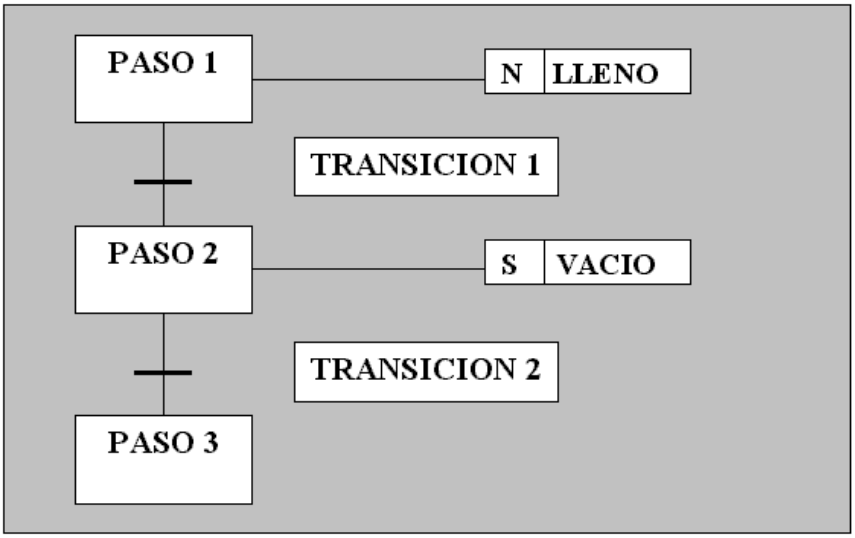
Los bloques funcionales son los equivalentes de los circuitos integrados, IC´s, representando una función especializada de control. Ellos contienen datos al igual que algoritmos, así que pueden guardar datos (que es una de las diferencias con las Funciones W.R.T.).? Tienen un interfaz bien definido y un interior oculto, como un circuito integrado o una caja negra.
De este modo, dan una clara separación entre los diferentes niveles de programadores, o personal de mantenimiento.

Un lazo de control de temperatura, PID, es un excelente ejemplo de Bloque Funcional. Una vez definido, puede ser usado una y otra vez, en el mismo programa, en diferentes programas o en distintos proyectos. Esto lo hace altamente reutilizable.
Los bloques funcionales pueden ser escritos en alguno de los lenguajes de la norma IEC, y en la mayoría de los casos, en C. Esto hace que puedan ser definidos por el usuario. Los Bloques Funcionales Derivados, están basados en los FB´s definidos por la norma, pero también es posible realizarlos completamente nuevos, diseñados, a la medida del usuario: Sólo esto mejora ya el trabajo.

Programas:

Con los bloques de función básicos mencionados anteriormente, se puede decir que un programa es un conjunto formado por unas Funciones y unos Bloques Funcionales. Un programa puede ser escrito en alguno de los lenguajes de programación definidos.

Gráfica de Función Secuencial (Sequential Function Chart, SFC)



SFC describe gráficamente el comportamiento secuencial de un programa de control. Está derivado de Petri Nets e IEC 848 Grafcet, con los cambios oportunos para convertir la representación de un documento estándar en un set de elementos de control de ejecución.
SFC estructura la organización interna de un programa, y ayuda a descomponer un problema en partes manejables, manteniendo simultáneamente una visión global.
SFC consiste en pasos, encadenados con Bloques de Acción y Transiciones. Cada paso representa un estado particular del sistema que está siendo controlado. Una transición está asociada con una condición, que cuando se cumple, causa la desactivación de la transición anterior y la activación de la siguiente. Los pasos están unidos a los Bloques

de Acción, realizando un control del proceso. Cada elemento puede ser programado en alguno de los lenguajes IEC, incluyéndose el SFC.
Se pueden usar secuencias alternativas y paralelas, como en la mayoría de las aplicaciones que se emplean. Pongamos por caso, una secuencia que es usada para el proceso primario y una segunda para monitorizar globalmente el desempeño de las operaciones.
Debido a su estructura general, SFC mejora también como herramienta de comunicación, combinando diferentes personas, lugares o países.

Lenguajes de Programación

Se definen cuatro lenguajes de programación normalizados. Esto significa que su sintaxis y semántica ha sido definida, no permitiendo particularidades distintivas (dialectos). Una vez que los has aprendido podrás usar una amplia variedad de sistemas basados en esta norma.

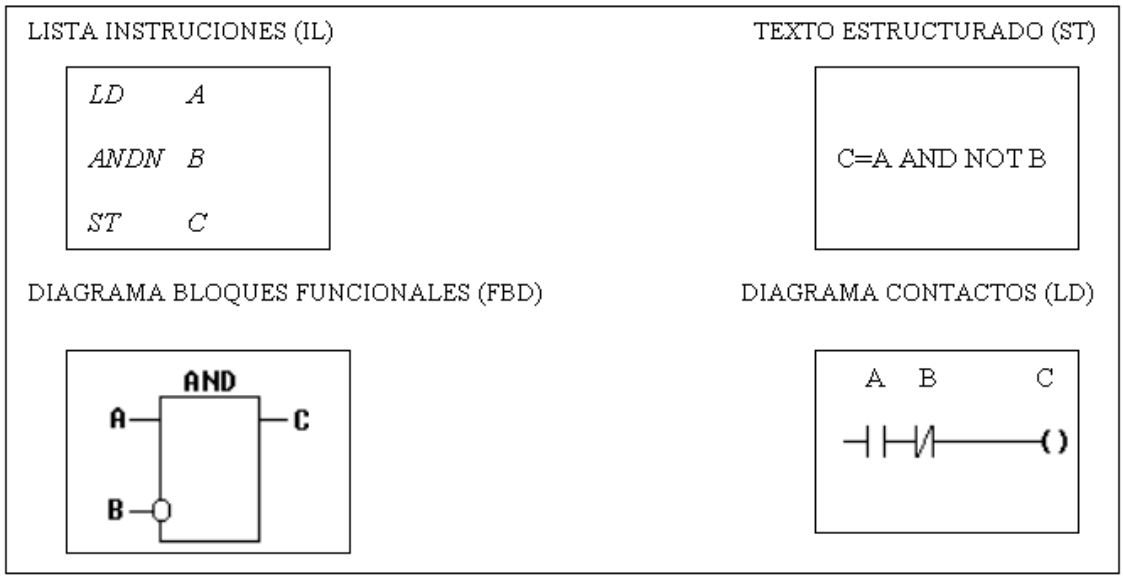
Los Lenguajes consisten en dos versiones que son de texto y dos que son gráficas:

De texto:

- **Lista de instrucciones**, (Instruction List, IL)
- **Texto estructurado**, (Structured Text, ST)

Gráficos:

- **Diagrama de contactos**, (Diagram Ladder, LD)
- **Diagrama de bloques funcionales**, (Function Block Diagram) FBD



En la figura superior, los cuatro programas describen la misma acción.

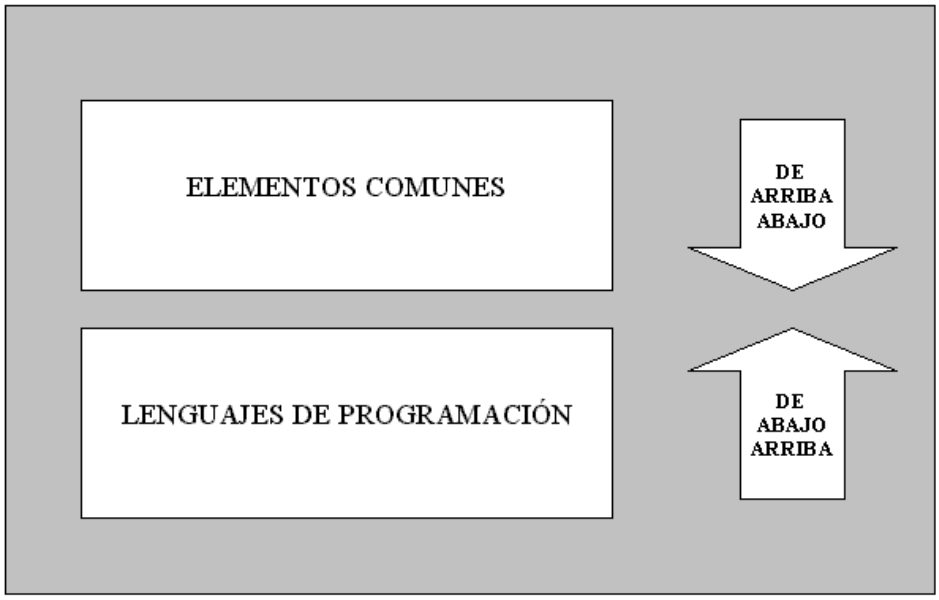
La elección del lenguaje de programación depende de:

- Los conocimientos del programador
- El problema a tratar
- El nivel de descripción del proceso
- La estructura del sistema de control
- El interfaz con otras personas o departamentos.

Los cuatros lenguajes están interrelacionados: ellos permiten su empleo en común, con un enlace a una experiencia existente. En este modo también provén de una herramienta de comunicación combinando gente de distinta formación.

El Diagrama de contactos tiene sus orígenes en los Estados Unidos. Está basado en la presentación gráfica de Relay Ladder Logic.
Lista de Instrucciones es la parte correspondiente en Europa. Como lenguaje de texto, se parece al ensamblador.
Diagramas de Bloques Funcionales es muy común en los procesos industriales. Expresa el comportamiento de funciones, bloques funcionales, como en diagramas de circuitos electrónicos. Se comporta como un sistema en cuanto a flujo de señales entre elementos de procesamiento.
Texto estructurado es un lenguaje muy poderoso con sus orígenes en el Ada, Pascal y C. Puede ser utilizado excelentemente para la definición de complejos bloques funcionales, los cuales pueden ser utilizados con algunos de los otros lenguajes.

Arriba-abajo vs. Abajo-arriba



La norma también permite dos formas de desarrollar tu programa: De arriba abajo y de abajo arriba. Puedes especificar tu modo de aplicación y dividirlo en sub-partes, declarar tus variables y demás. O tú puedes comenzar la programación de tu aplicación desde abajo, por ejemplo, por medio de funciones derivadas y bloque funcionales.

Para cualquiera que tu escojas, el medio de desarrollo te ayudará durante todo el proceso.

Implementaciones

Cumplir todos los requerimientos de la norma IEC 1131-3 no es fácil. Por esta razón, la norma permite implementaciones parciales en varios aspectos. Esto cubre el número de lenguajes que soporta, funciones y bloques funcionales. Esto deja libertad a la hora de escoger, pero el usuario debe tener cuidado durante este proceso de selección. También una revisión puede tener un nivel dramáticamente alto de implementación.

Muchos entornos de programación IEC actuales ofrecen todas las cosas que tú esperas de un entorno moderno:
Ratón, menús de persiana, pantallas de programación gráfica, posibilidad de tener abiertas múltiples ventanas, funciones construidas en hipertexto, verificación durante el diseño. Por favor, observe que esto no está especificado por la norma por sí misma: ésta es una de las partes donde los proveedores pueden diferenciarse.

Conclusión

Las implicaciones técnicas de la norma IEC 1131-3 son altas, dejando bastante espacio para el crecimiento y la diferenciación. Esto la hace apta para entrar óptimamente en el próximo siglo.
La IEC 1131-3 tendrá un gran impacto en el mundo del control industrial. Ciertamente no se restringe al mercado convencional de los PLC´s. Ahora mismo, se puede ver adoptada en el crecimiento de los mercados de control, sistemas distribuidos y sistemas de control basados en PC / softlogigc, incluidos los formatos SCADA. Y las áreas están todavía en crecimiento...
Teniendo una norma sobre el área de aplicación, se consiguen numerosos beneficios para usuarios y programadores. Los beneficios de su adopción son varios, dependiendo de las áreas de aplicación. Vamos a nombrar sólo algunos de ellos:

- Se reduce el gasto en recursos humanos, enseñanza, entrenamiento, mantenimiento y asesoramiento.
- Solventando focos de problemas por el alto nivel de flexibilidad y re-usabilidad del software.
- Reducción de malas interpretaciones y equivocaciones.
- Técnicas de programación utilizables en un amplio entorno de trabajo: Control industrial general.
- Combinando diferentes componentes provenientes de diferentes programas, proyectos, lugares compañías y/o países.

Para más información, por favor, contacte con PLCopen directamente
O vía Internet WWW.plcopen.org.
Nuestra dirección es:
PO Box 2015
NL 5300 CA Zaltbommel
Holanda
Tel.: 31-418-541139 /Fax.: 31-418-516336

Desarrollando la estructuración de un programa con IEC 1131-3

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

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
```

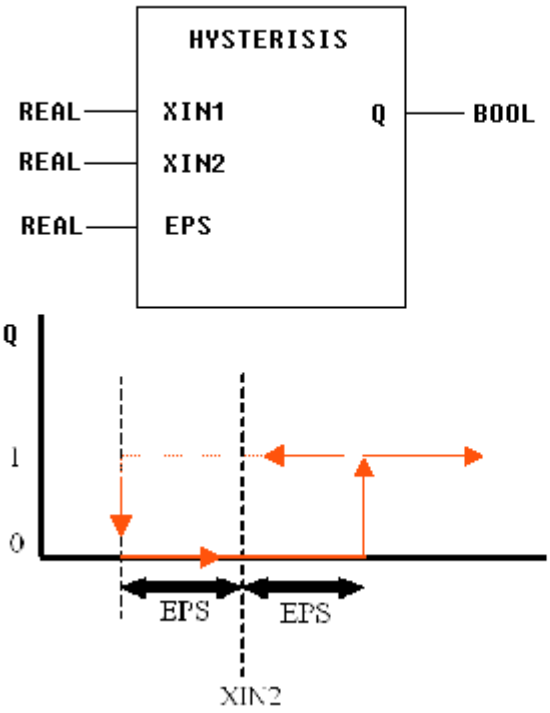
Una vez definida podrás utilizarla una y otra vez, en el mismo programa, en diferentes programas o en cualquier otro proyecto.

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?

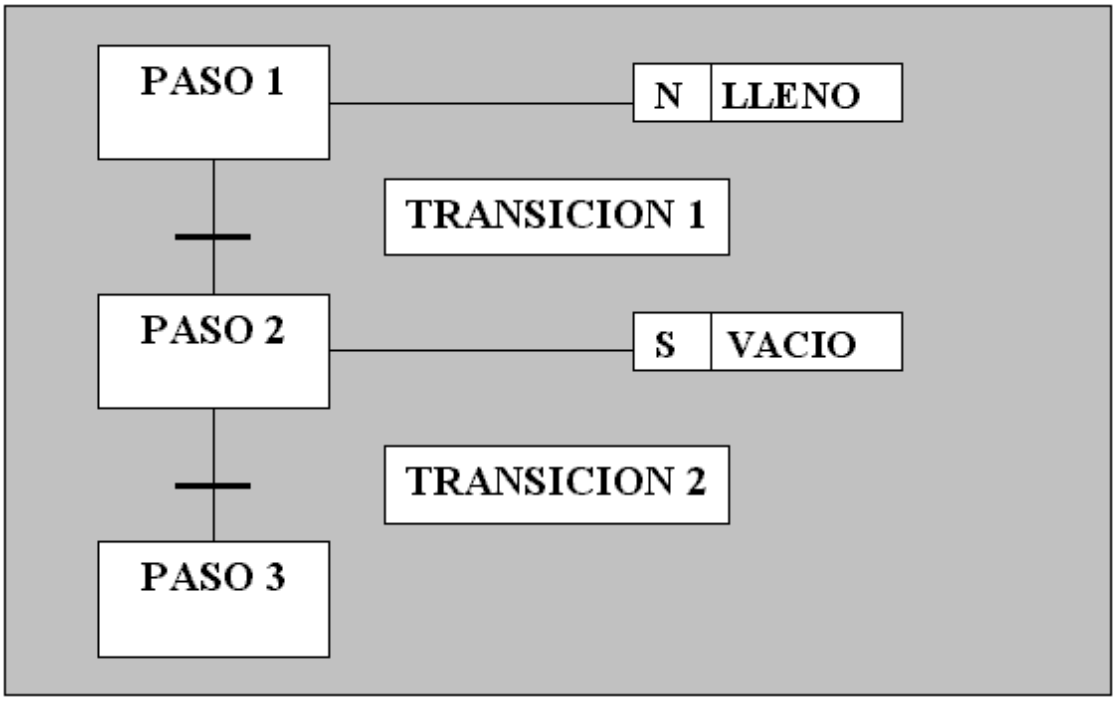
Como herramienta de descomposición, IEC 1131-3 provee de Cartas de Función Secuencial (Sequential Function Charts, SFC). Las SFC describen gráficamente el comportamiento secuencial de un programa de control. En este modo estructura la organización interna de un programa descomponiendo el problema de 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.

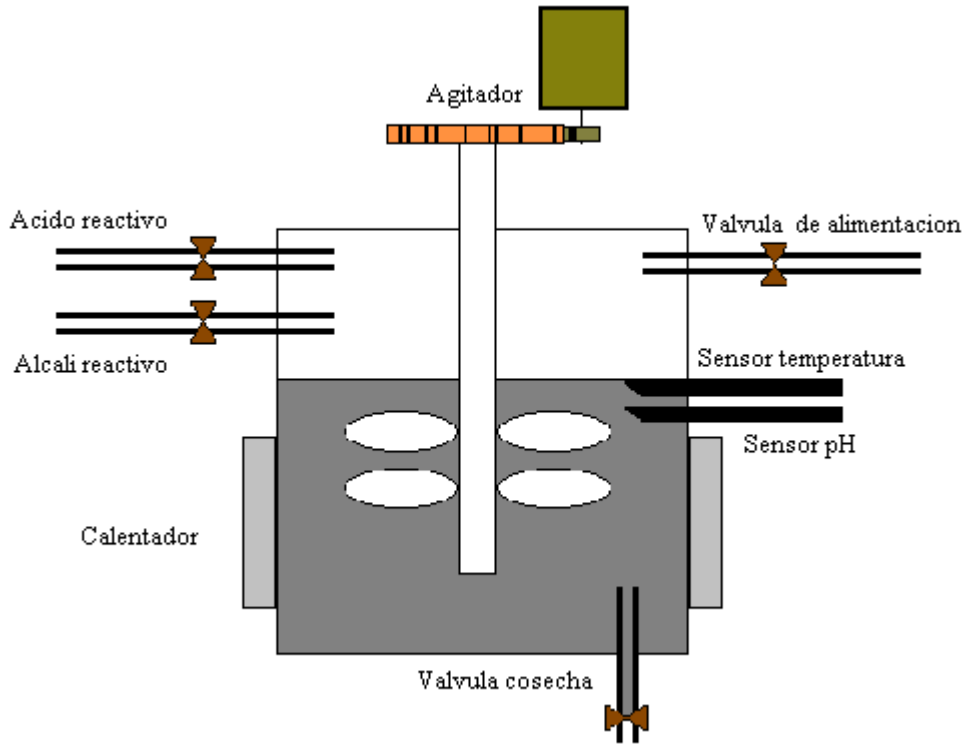
Además de estos 7 pasos, existen algunos principios más que deberían ser usados para optimizar el método de estructuración:
Estos principios son:

- 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)

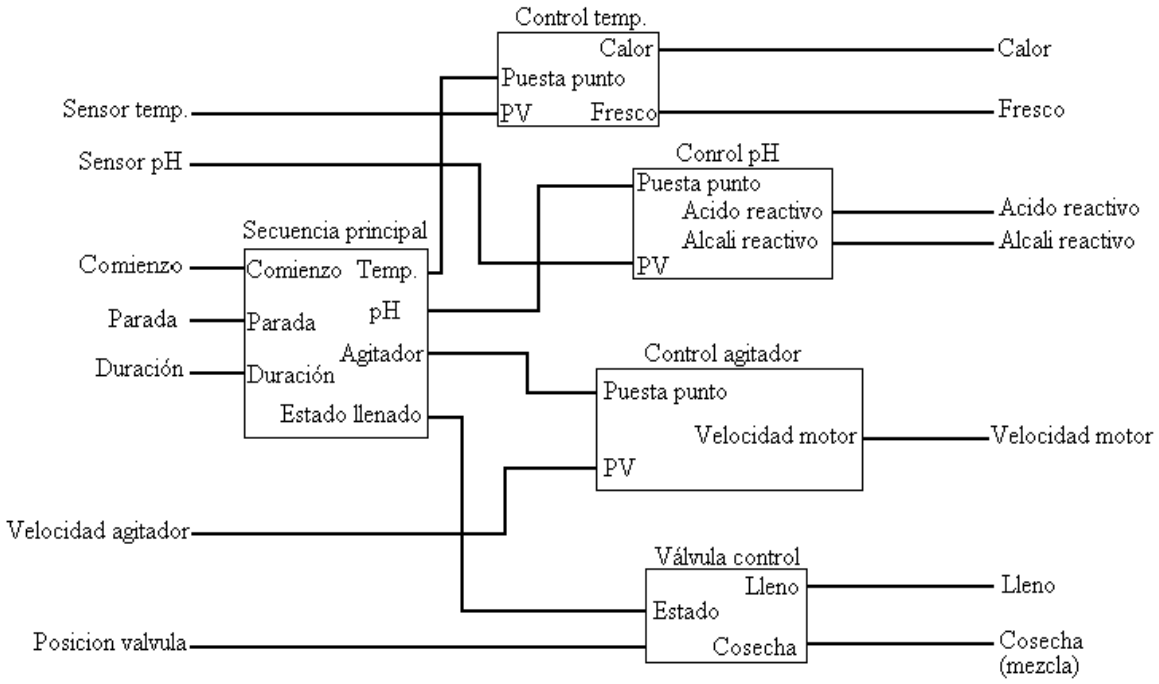
Un ejemplo dice más que mil palabras..., así que vamos a ver un proceso de fermentación y su control, como se muestra más abajo.



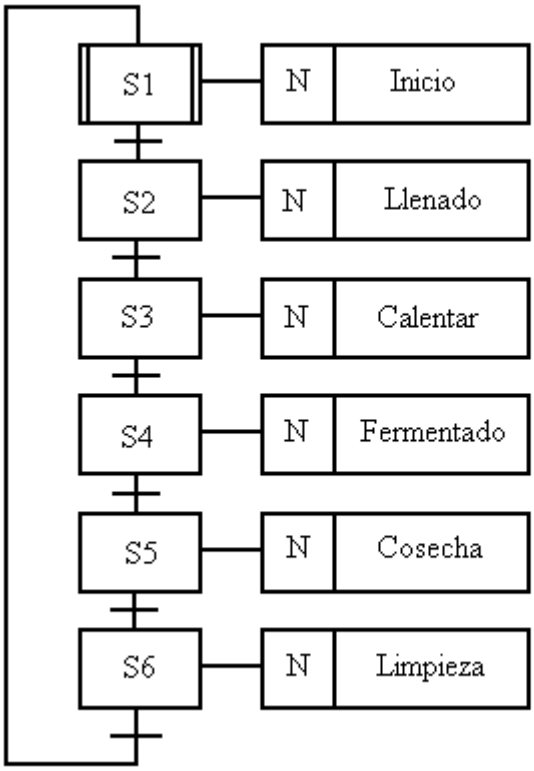
Todos los interfaces externos están definidos aquí (paso 1). Hay un gran recipiente, que puede llenarse (válvula de alimentación) con líquido, puede calentarse con el calentador (enfriamiento por convección), puede agitarse con un motor, y donde el ácido y el álcali pueden ser añadidos dentro del recipiente.
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.

Ventajas para los usuarios

Dentro de la IEC1131-3

IEC 1131-3 es una norma a escala mundial. Armoniza la manera de ver de la gente el control industrial, estandarizando el interfaz de programación. Esto incluye la definición del lenguaje de Cartas de Función Secuencial (SFC), usadas en la organización interna de un programa, y cuatro lenguajes inter-operables: Lista de Instrucciones (IL), Diagrama de Contactos (LD), Diagrama de Bloques Funcionales (FBD) y Texto Estructurado (ST). Por medio de la modularización y la declaración de variables cada programa está adicionalmente estructurado, incrementando su re-utilizabilidad, reduciendo errores e incrementando la eficiencia. En adición, IEC 1131-3 estructura el modo en el que un sistema de control está configurado.

Ventajas generales

Como la IEC 1131-3 está en proceso de expansión, esto favorece a los usuarios de la norma. Ventajas generales son:

- Se reduce el gasto en recursos humanos.
- El alto nivel de re-utilizabilidad del software soluciona una fuente de problemas.
- Reduce las malas interpretaciones y los errores.
- Las técnicas de programación provistas están disponibles para un amplio área: La industria general de control.
- Combina armoniosamente diferentes componentes provenientes de diferentes localizaciones, países y proyectos.

Ventajas para los usuarios

Usuarios: ¿Qué usuarios?

"Usuarios de la IEC 1131-3" es un término muy genérico. Pueden ser encontrados en un amplio área, cubriendo todos los aspectos del mercado industrial. En el entorno de las cadenas de distribución, uno puede distinguir entre diferentes tipos de usuarios:

- proveedores de software (independientes)
- proveedores de sistemas y hardware de control industrial
- programadores
- fabricantes de unidades de producción (máquinas)
- Integradores de sistemas
- Proveedores de SCADA y DCS
- Usuarios finales de la industria de equipamiento, en manufacturación y procesos de control
- Secciones de instalación
- Personal de mantenimiento (independiente)
- Profesores (independientes) / nivel educacional

Beneficios para proveedores de software (independientes)

- IEC 1131-3 provee una definición de norma para los entornos de desarrollo de software.
- La norma cada vez está más y más aceptada por sus clientes, creando un gran mercado potencial.

- El mercado es amplio: Control industrial general.
- Posibilidad de desarrollo de software independiente: las adaptaciones de hardware están limitadas sólo a partes específicas. Esto crea la posibilidad de cumplir plenamente los requerimientos del cliente.
- Posibilidad de obtener una mayor eficiencia de manera sensible, usuarios satisfechos, acoplable a otros programas y herramientas, estructura óptima, apoyo, entrenamiento, documentación de usuario, disponibilidad de lenguaje, librerías de bloques funcionales, etc.
- Apoyo de múltiples proveedores independientes a la aceptación de la norma.

Beneficios para los proveedores de sistemas

También como fabricante, la IEC 1131-3 ofrece beneficios generales:

- No se necesitan especificaciones internas.
- El mercado que cumple la norma se incrementa: a menudo es un requisito imprescindible.
- Reconocimiento y aceptación de la compañía.
- Posibilidad de presentarlo como un valor añadido.
- Acceso a módulos y herramientas de 3ª generación.
- Incremento de la totalidad del mercado disponible entre amplias áreas de aplicación.
- Como camino para diferentes líneas de productos.
- Mejora el enlace entre el pasado y el futuro (ejemplo, softlogic)

Ventajas para los programadores

- Desarrollo de un programa bien estructurado de "arriba-abajo" o de "abajo-arriba".
- Mejora de las herramientas por descomposición del problema en unidades manejables.
- Beneficia el desarrollo en un grupo con diferente formación y/o niveles.
- Alto nivel de re-utilización del código de programa.
- Tipos de datos sólidos que impiden los errores de programa.
- Apoyo para un pleno control de ejecución.
- Apoyo en la descripción de complejos comportamientos secuenciales (SFC)
- Estructuras de datos con fácil intercambio de elementos de datos.
- Posibilidad de selección de un lenguaje flexible, jugando con diferentes estilos.
- Posibilidad de la existencia de un vendedor que desarrolle software independiente.

Ventajas para los fabricantes de unidades de producción (máquinas)

- Mejora la independencia hacia los proveedores en materia de control.
- Mejora las bases para la reutilización del software en las plataformas.
- Mayor flexibilidad en personal: no estarán limitados a un sólo sistema.
- Se requiere menor entrenamiento.
- Alta eficiencia en la programación.
- Un alto nivel de re-utilización reduce costos e incrementa la confidencialidad

- Mejora las herramientas para requerimientos específicos de alto nivel.
- Mejor posibilidad de comparación entre las diferentes ofertas, especialmente en el desarrollo de software.
- Mejora la compenetración entre el personal existente.
- Aplicable en todo el mundo una y otra vez.
- Mejora de una herramienta de comunicación entre ingenieros en diferentes localizaciones.

Ventajas para los integradores de sistemas

- Mejora la independencia hacia los proveedores al nivel de control.
- Mejora las bases para la reutilización del software en las plataformas.
- Mayor flexibilidad en personal: no estarán limitados a un sólo sistema.
- Se requiere menor entrenamiento.
- Una alta eficiencia en la programación reduce los costes.
- Un alto nivel de re-utilización reduce costos e incrementa la confidencialidad
- Mejora las herramientas para requerimientos específicos de alto nivel, mejorando la comunicación con los clientes.
- Aplicable en todo el mundo una y otra vez.
- Mejora de una herramienta de comunicación entre ingenieros pertenecientes a la organización.

Ventajas para los proveedores de SCADA y DCS

- Clarifica los contenidos en el mercado disponible por la integración de la parte del control.
- Inexistencia de la necesidad de especificaciones internas propias.
- Gran independencia para escoger dentro de los proveedores de software, creando la base para soluciones óptimas.
- Enlace con una amplia parte de los productos y entrenadores de 3ª generación.

Ventajas para el usuario final de la industria de equipo

- Mayor independencia hacia los proveedores al nivel de control.
- Un alto nivel de re-utilizabilidad reduce los costos y aumenta la confidencialidad de los diseños.
- Mejora las herramientas para requerimientos específicos de alto nivel.
- Mayor independencia hacia los proveedores de sistemas /servicios.
- Mejor posibilidad de comparación entre las diferentes ofertas, especialmente en el desarrollo de software.
- Mejora la compenetración entre el personal existente.
- Aplicable en todo el mundo una y otra vez.
- Mejora de una herramienta de comunicación entre ingenieros en diferentes localizaciones.

Ventajas para las secciones de instalación

- Facilita la comisión dentro de la programación estructurada.

- Menos errores, reducen los costes de instalación y ofrecen periodos de instalación más cortos.
- Soporta la integración de sistemas de control de diferentes marcas.
- Menor entrenamiento de personal, fomentando la potencia del grupo de trabajo y la distribución de los conocimientos.
- Comienzo, en muchos de los mercados, a considerar la norma como un pre-requisito imprescindible.

Ventajas para el personal de mantenimiento (independiente)

- Posibilidad de atender múltiples marcas sin necesidad de formación específica.
- Diferentes niveles de separación son posibles por medio de los bloques funcionales.
- La selección de lenguajes mejora a menudo diferentes puntos de vista para los distintos países.
- Con la estructuración de las herramientas se mejora la visión global.
- A menudo, en casos determinados, funciones de ayuda provistas y posibilidad de múltiples ventanas, facilitan la intervención.

Ventajas para proveedores de software (independientes) / nivel educacional

- Posibilidad de la organización de cursos independientemente del proveedor.
- Posibilidad de cursos para productos independientes.
- Menor diferencia entre la teoría y la práctica especialmente en escuelas /nivel educacional.
- Basado en técnicas de programación actuales y pensamiento estructural.

Usuarios: ¿Qué usuarios?
Beneficios para proveedores de software (independientes)
Beneficios para los proveedores de sistemas
Ventajas para los programadores
Ventajas para los fabricantes de unidades de producción (máquinas)
Ventajas para los integradores de sistemas
Ventajas para los proveedores de SCADA y DCS
Ventajas para el usuario final de la industria de equipo
Ventajas para las secciones de instalación
Ventajas para el personal de mantenimiento (independiente)
Ventajas para el personal de mantenimiento (independiente)
Ventajas para proveedores de software (independientes) / nivel educacional

¿Cómo reconocer un sistema de programación IEC 1131-3 en 8 fáciles pasos?

1. Existe un rango sólidamente definido de tipos de datos normalizados, que especifica exactamente el contenido de cómo tiene que ser interpretada una variable.
Ventaja: Es allí donde cada tipo de dato opera sólo donde le está permitido (ejemplo: operaciones matemáticas para tipos de datos numéricos y no para bits), Así se mejora la seguridad del programa y la visión general.

2. Existe la posibilidad de definir tipos de datos derivados como campos y estructuras.
Ventaja: El sistema de programación mejora las posibilidades y el uso de lenguajes de computador de alto nivel. Viendo los datos pueden ser plenamente agrupados por su significado y usados fácil y fiablemente.

3. Los programas de usuario pueden ser subdivididos exactamente en elementos de estructuración definidos, las unidades de organización de programa (POU), programas, funciones y bloques funcionales.
Ventaja: El problema puede ser estructurado en sub-tareas, haciéndolo mucho más claro que los "programas-spaghetti". La descomposición de las sub-tareas recurrentes en los propios bloques funcionales ahorra bastante código de programación.

4. Todas las unidades de organización de programa (POU´s) pueden contener datos locales, por ejemplo, datos, los cuales son sólo conocidos y utilizables con esta POU. Este principio de encapsulamiento de datos proviene también de los modernos lenguajes de programación.
Ventajas: Es posible una descomposición real del trabajo, porque los programadores no necesitan estar coordinados en las variables. La sobreescritura errónea de variables no es posible de ninguna de las maneras. Las POU´s son direcciones independientes y pueden ser re-utilizadas sin ningún problema (vea también los puntos 5 y 6).

5. Para el intercambio de datos entre POU´s existen unos interfaces bien definidos.
Ventajas: Funciones recurrentes pueden ser transferidas a librerías de funciones o bloques funcionales. Se pueden considerar como cajas negras, sin conocimiento de su interior como circuitos integrados con conexiones.

6. Funciones y bloques funcionales se hacen puramente simbólicos, por ejemplo direcciones y módulos independientes de programas.
Ventaja: En la conexión entre los datos locales y la programación simbólica pura no se producirá ninguna interferencia entre los programas de usuario. Las funciones y los bloques funcionales son independientes de la tarjeta de sistema, y en muchos casos independientes del proveedor y, por tanto, re-utilizables. (Vea también el punto 8).

7. Aquellos lenguajes de programación que cumplen con las especificaciones de la IEC 1131-3, no tienen ambigüedades en su sintaxis y semántica. En suma, se beneficia la normalización de funciones y los bloques funcionales.
Ventaja: Los usuarios, habiendo aprendido los lenguajes y comandos de la norma, pueden usar estos conocimientos en diferentes sistemas de programación. Se reducen los cursos de formación necesarios. Los usuarios y especialmente los servicios de personal encuentran un sistema de programación comprensible y unificado a escala mundial. Los programas y las partes de programa pueden ser usados en los distintos sistemas.

8. El sistema de programación está certificado a los niveles necesarios de certificación definidos por PLCopen, por ejemplo, por un instituto de prueba independiente en conformidad con la norma IEC 1131-3.
Ventaja: El usuario está seguro de que el sistema de programación cumple con la norma, que él puede usar sus conocimientos de programación de la IEC y que el sistema quedará a prueba para el futuro. Con la certificación y el nivel de transportabilidad, el programador puede cambiar las funciones y los bloques funcionales, escritas cumpliendo la norma, con el formato de archivos de intercambio comunes con otros sistemas certificados, ahorrando tiempo y dinero.

Evaluación de software: Entorno de desarrollo de software Autorizado /comprado

Si tú estas en el proceso de compra de un entorno de desarrollo IEC 1131-3, existe hoy en día un gran número de proveedores de software independientes para elegir. Para facilitar tu proceso de elección, las siguientes consideraciones pueden ayudarte en tu proceso de selección. No hay muchos detalles técnicos, pero sí consideraciones adicionales que deberían ser tenidas en cuenta. La primera: no existe el producto ideal. Un producto debería amoldarse a tus necesidades, lo que significa que tú debes evaluarlo. Más abajo existen unas líneas de evaluación.

Cualquier producto considerado como el mejor ahora, puede ser sobrepasado por una nueva versión de un competidor. También, la propia categoría del producto, puede ser un factor de menor importancia, pues posiblemente una nueva versión esté a la vuelta de la esquina.

Puntos de atención:

- Costes de adaptación: ¿Cumple tu hardware los requisitos mínimos del nuevo software?.¿Qué costes se derivarán de la inclusión de un hardware adicional y/o nuevas librerías?.
- Los costes iniciales son diferentes. En la mayoría de los casos el entorno del software requiere adaptaciones. Esto puede abarcar un rango muy amplio:
    - El nombre de producto que aparece en la pantalla.
    - La adaptación a tu entorno específico de hardware.
    - La adaptación a los manuales de usuario que necesites.
    - La creación de manuales de usuario bajo tu propio nombre.
    - La inclusión de requerimientos, como enlaces a tu compilador específico.
- Licencia: Además de los costes iniciales de adaptación, la obtención de la licencia puede ser aplicable. ¿Cuánto cargará en cuenta?¿Cuánto cuesta sacarla para un período de tiempo?¿Se necesitarán futuras revisiones?.
- Estrategia para distribuir más o menos adaptaciones.
- La calidad del software y los manuales de entrenamiento, y la disponibilidad en los distintos idiomas.
- ¿Incluye el producto por sí mismo funciones de ayuda en línea, disponibles en los distintos lenguajes?.
- Servicio: Ellos pueden proclamarlo, pero ¿quién provee el mejor y en tu lenguaje?. ¿Y a qué precio?. ¿Cuál es su política con respecto a la garantía?
- Formación: ¿Pueden ellos formar al personal en su propio lugar de trabajo?. ¿Pueden ellos ayudar a tus usuarios?. ¿Cómo están sus manuales y en qué lenguaje?. ¿Puedes usar su propio material como base para tu propio desarrollo futuro?
- Adaptaciones: ¿Cómo distribuyen ellos sus adaptaciones?, ¿Cómo distribuyes tú, tus adaptaciones?

- ¿Cómo es el sistema de ayuda en línea?. ¿En qué lenguaje?. ¿Cubre tus necesidades?.
- ¿Es la compañía financieramente estable?.
- ¿Qué referentes / instalaciones posee la compañía?. ¿Trabaja con tus competidores?. ¿Te sirve eso de ayuda?. ¿Puedes obtener referencias?.
- ¿Cómo encaja la compañía en tus planes de futuro?. ¿Distribuirían distintos sistemas si fuera necesario?.
- Si tuvieras un código que desearas incluir, ¿qué apoyo te darían?. ¿Recibirías apoyos del entorno?. ¿Cómo trabajan?. ¿Cuánto esfuerzo es estimado por ellos y por ti para hacer el trabajo?.¿Están ellos dispuestos a hacerlo (a costes fijos), mostrando confidencialidad y dándote una garantía?, ¿a qué precio?, ¿en cuánto tiempo?.
- ¿Pueden ellos proveerte de un software de evaluación?
- ¿Cómo es la respuesta de rápida?
- ¿Hablan tu idioma, no sólo en tu zona de vida, sino que conocen las costumbres del ambiente que te rodea?
- ¿Está el producto certificado por PLCopen?, ¿a qué nivel?, ¿en qué lenguajes de programación?, ¿Cuántas adaptaciones han tenido después de registrarse?. ¿Pueden enseñarte una copia del certificado?.
- ¿Pueden ellos darte una prueba de que cumplen la norma mandándote las tablas de la IEC 1131-3?
- ¿Cuáles son tus principales lenguajes de programación para tu entorno?.¿Qué longitud de programa soporta?. ¿Qué versión tienen?.

Recuerda: no te engañes: elegir acertadamente cuesta tiempo y dinero.

Un buen punto de partida:

1. Describe tus requerimientos iniciales claramente en papel, incluyendo procedimientos y puntos críticos.
2. Envíalo a tus proveedores potenciales, mínimo 5, preferiblemente en el mismo día(fax).
3. Anota la fecha de llegada de les respuestas, ello te dará una primera impresión de la velocidad de respuesta.
4. Compara la calidad global de cada una de las ofertas.
5. Compara el cumplimiento de tus requerimientos.
6. Estudia las diferencias.
7. Contacta personalmente con al menos 3 compañías.

Estado de la norma:

¡Nueva versión inminente!

El documento de la norma fue preparado por el sub-comité IEC SC65B/WG7 (Controladores Programables) Grupo de Trabajo 3 (TF3) de lenguajes de programación. El documento en sí mismo, fue publicado en 1993, está disponible en el local de las oficinas de normalización, o en la central de IEC en Geneva, Suiza. La norma tendrá 5 años de vida y será revisada en 1998.
En el próximo encuentro IEC TC65B, de los días 3 y 4 de noviembre de 1997, se propondrán nuevas versiones de la IEC 1131-3, con un Comité de Votación.
El Comité IEC ha aceptado una nueva manera para denominar la norma. Esto significa que la IEC 1131-3 será en el futuro la IEC 61131.
En adición a la norma, el Grupo de Trabajo ha producido o está trabajando en los siguientes documentos:

Comisión Técnica

Durante las implementaciones de la norma, uno encuentra deficiencias, omisiones, y lagunas. Para solventarlo en una fase inicial, se creó la Comisión Técnica. Este documento ha sido propuesto a la IEC SC65 para su inclusión.

Enmiendas a la IEC 1131-3

El Grupo de Trabajo 3 ha recibido notificaciones para la inmediata inclusión de diversos cambios en la IEC 1131-3. Estos cambios, mientras no sean de la misma urgencia como los actualmente propuestos por la Comisión Técnica, deberían incrementar sustancialmente su corrección y seguridad antes de su implementación definitiva en los lenguajes de programación de la 1131-3. Allí antes, estos cambios son investigados para ser correctamente completados antes de la revisión de la IEC 1131-3, fijada para 1998.
La prioridad del documento es fijada por el Comité.

Información técnica Tipo 2: Extensiones propuestas a la 1131-3

Esta información técnica propone características adicionales las cuales pueden ser requeridas en el futuro para favorecer una implementación y aplicación eficiente de los lenguajes de programación definidos en la IEC 1131-3. La meta es llegar a un consenso en un conjunto de características comunes que serán definidas en la próxima edición de la IEC 1131-3.
Este documento está viendo la luz como un “prospecto normalizado para la aplicación provisional” en el campo de la programación porque hay una urgente necesidad de guiar en como la norma debería ser usada en este campo para solucionar esta necesidad bien identificada. No se mira como una Norma Internacional. Está propuesta como una aplicación provisional, así que puede reunirse información y experiencia de su uso.

Información técnica Tipo 3: Líneas de aplicación e implementación

Este documento está siendo clasificado como información técnica de tipo 3 en las series de publicaciones (acorde a la subcláusula G.4.2.3 de la parte 1 de las directivas de la IEC/ISO) en orden de ofrecer unas líneas para la implementación y aplicación de los lenguajes de programación definidos en la IEC 1131-3 (Controladores Programables, Parte 3 – Lenguajes de Programación).
Este documento es por completo informativo por naturaleza y no es contemplado como una "Norma Internacional". Contiene respuestas a un tipo frecuente de preguntas sobre las aplicaciones e implementaciones proyectadas para la normativa IEC 1131-3. Su publicación será como IEC 61131-8.
Para las últimas versiones de estos documentos, por favor consulte este sitio WEB o el sitio de Allen Bradly en:
ftp://ftp.cle.ab.com/stds/iec/sc65bwg7tf3.

DCS = IEC 1131-3 + Buses de campo

RenéSimon / Matthias Riedl
IFAK e. V., Barleben, Alemania

El modelo de software y los lenguajes de programación descritos en la IEC 1131-3 son aplicables en un amplio rango de Sistemas de Control Distribuidos (Distributed Control Sistems DCS´s), no sólo para PLC´s. Sobre todo allí, es necesario disponer de un sistema de comunicación de bus de campo que sea capaz de manejar los Bloques Funcionales (FB´s). Este artículo muestra una aplicación que fue desarrollada en el Instituto para Automatización y Comunicación e. V. (Institut für Automation und Kommunikation e.V., IFAK). Introduce el concepto de comunicación de la IEC 1131-5, describe las necesidades del usuario (no necesariamente ya demandadas) e intenta realizar un primer enfoque.
A fin de ilustrar los conceptos técnicos, el ejemplo mostrado en la figura 1 será usado en este artículo.

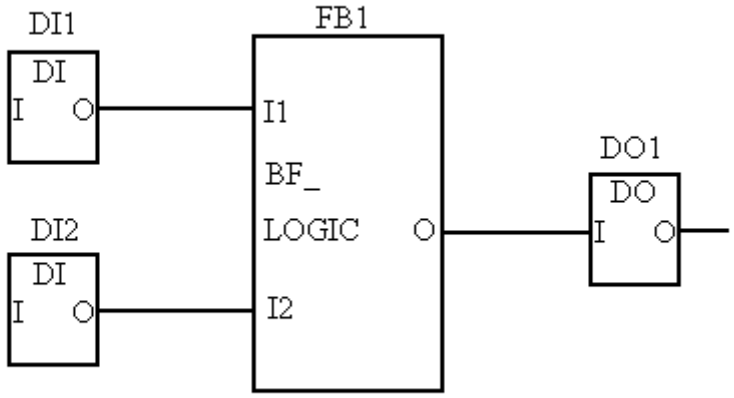


Fig. 1 Ejemplo técnico

El ejemplo consiste en 4 Bloques Funcionales y sus conexiones. Hay 2 entradas digitales (DI1 y DI2) que sirven de interfaz para el proceso. Ellas controlan una salida digital (DO1) por medio de una combinación lógica (FB_LOGIC).
Observe que los bloques funcionales no están circunscritos a un recurso o dispositivo físico concreto. Sólo se describe la funcionalidad desde el punto de vista del usuario. El comportamiento interno de los bloques funcionales no es relevante aquí.
Asumiendo que el bloque funcional FB_LOGIC está situado en un dispositivo físico (configuration A) y todos los otros bloques funcionales a un segundo (configuración B), la figura 2 describe una posible realización de nuestro ejemplo utilizando la sintaxis de la IEC 1131-3.

```
CONFIGURATION A
      RESOURCE A1
            PROGRAM A11
                  VAR
                        SEND1 : SEND;
                        RCV1 : RCV;
                        FB1 : FB_LOGIC;
                  END_VAR
                  RCV1 (EN := TRUE, ID := `CH1´,
                  R_ID := `B.B1.B11.SEND1´);
                  FB1 (I1 := RCV1.RD_1, I2 := RCV1.RD_2);
                  SEND1 (REQ := TRUE´, ID := `CH1´,
                  R_ID := `B.B1.B11.RCV1´, SD_1 := FB1.0);
            END_PROGRAM
      END_RESOURCE
END_CONFIGURATION

CONFIGURATION B
      RESOURCE B1
            PROGRAM B11
                  VAR
                        SEND1 : SEND;
                        RCV1 : RCV;
                        DI1, DI2 : DI;
                        DO1 : DO;
                  END_VAR
                  DI1(%IX1);
                  DI2(%IX2);
                  SEND1 (REQ :=TRUE,  ID := `CH1´,
                  R_ID := A.A1.A11.RCV1´,
                  SD_1 := D11.=, SD_2 := DI2.0);
                  RCV1(EN := TRUE, ID := `CH1´,
                  R_ID := A.A1.A11.SEND1);
                  DO1(RCV1.RD_1, %QX1) ;
            END_PROGRAM
      END_RESOURCE
END_CONFIGURATION
```

Fig. 2 :ejemplo de uso de sintaxis de IEC 1131-3

Para intercambiar datos entre dispositivos físicamente separados, la IEC 1131-5 ofrece diversas posibilidades, aquí se usan los bloques funcionales para comunicación SEND y RCV. Aunque sólo se utiliza un ejemplo muy simple, el algoritmo correspondiente es complicado, especialmente para un programador que no conozca mucho acerca de modelos servidor / cliente o servicios a bajo nivel.

Este modelo de comunicación es sólo apropiado para aplicaciones que no tienen una conexión demasiado compleja. No es posible realizar de este modo, una base de datos distribuida que provea de la misma funcionalidad en un entorno distribuido como hace el PLC con la imagen normal de proceso. Además, no es soportada la sincronización entre los bloques funcionales.

Para intensificar la facilidad de comunicación en DCS´s usando la IEC 1131-3 es necesario aumentar primero el modelo de software de la norma. Está propuesto aquí

introducir una nueva categoría: “Sistema”. Una primera definición de sistema podría ser como sigue:

```
(D1) System
: configuration
| configuration system
;
```

Un sistema consiste en una lista no vacía de configuraciones. La definición de la propia categoría de configuración es equivalente a la norma actual sin los bloques funcionales de comunicación (CFB´s) y los caminos (Paths) de acceso. La comunicación con un sistema puede entonces ser descrita de la misma manera como una configuración descrita hoy. El programador estará capacitado no sólo para especificar dispositivos únicos sino para la totalidad de DCS, usando el método anteriormente descrito.
Después de este primer paso, la descripción de un DCS usando la nueva categoría “System”, se debería realizar un segundo paso: dividir el proceso de diseño en dos partes.

1. Descripción funcional
2. Instrumentación

Eso significa que el programador primero describe su problema desde el punto de vista expuesto en la figura 1 (descripción funcional). Después de que la asignación de las unidades funcionales (programas, bloques funcionales) a sus configuraciones (recursos) toma lugar (herramientas-soportadas, librerías). La Figura 3 muestra nuestro ejemplo después de la descripción funcional:

```
SYSTEM PROPOSAL
      PROGRAM P1
            VAR
                  FB1 : FB_LOGIC;
                  DI1 : DI;
                  DI2 :DI;
                  DO1 : DO;
            END_VAR
            DI1 (%IX1);
            DI2 (%IX2);
            FB1 (I1: =B.DI1.0, I2: =B.DI2.0);
            DO1 (I: =A.FB1.0, %QX1);
      END_PROGRAM
END_SYSTEM
```

Fig. 3: ejemplo de ampliación de uso de la sintaxis de IEC 1131-3

Por favor, observe que la configuración de categorías y recursos no se ha usado aquí, los cuales podrían ser necesarios en el “lenguaje ordinario 1131”. Una nueva definición de sistema debe ser provista:

```
(D2) System
: program
| program System
;
```

En lugar del programa de bloques funcionales, podría aparecer también en esta definición.
Mejorar la sintaxis de la IEC 1131-3 en la descripción de DCS´s es una cosa, el desarrollo de las herramientas necesarias y del software de comunicación, otra, pero no menos importante. Los buses normalizados de campo alemanes PROFIBUS están siendo extendidos con funcionalidad, la cual fue originalmente especificada por el InterOperable Systems Project (ISP). Este será hecho en una nueva guía adicional (PNO-Richtline) y en la versión europea de la norma PROFIBUS.
La distribución de la transferencia de datos (DDT) es especialmente importante para aplicaciones en el área del proceso de control. Esta función fue originalmente descrita por la especificación de los buses normalizados de campo franceses FIP. Los dispositivos emisores proporcionan datos a otros dispositivos que los demandan de una manera cíclica.

El uso de un Emisor Patrón permite enlazar dispositivos, cuya conexión no está inicialmente dispuesta para iniciar una transferencia de datos con alguno o todos los dispositivos del bus. La función es ayudada por un especial concepto de redundancia. El fallo del Emisor Patrón es reconocido y otro Dispositivo (Maestro) toma su función sin terminación de una conexión o una reconfiguración de un dispositivo o de un entorno de bus. El rol cambia Receptortor /Emisor Patrón, y la determinación de nuevos derechos de acceso ocurre durante el ciclo de transferencia de datos sin ninguna actividad por la aplicación. La aplicación está sólo informada sobre el cambio, por la dirección de red.

Dirección de Sistema

La Dirección de Sistema (SM) es una funcionalidad, distribuida en la red, la cual gestiona las tareas como el control del tiempo, validez de los bloques funcionales y la inclusión en programa.

#Control de tiempo
Una base de tiempos común en el entorno de sistema es necesario para la realización de DCS´s. La sincronización de los relojes locales toma lugar por los mensajes provenientes de la dirección de sistema (SM). En todo momento existe un reloj maestro (Time Master) por sistema, que sincroniza los demás relojes locales. En caso de fallo de este dispositivo otro potencial Reloj Maestro (Time Master) tomaría su función.

#**Localización de bloques funcionales**
La SM (Dirección de Sistema) provee la función para localizar un bloque funcional en un sistema para mejorar la conexión de los bloques funcionales, cuyos parámetros de entrada /salida han sido definidos. La función gestiona información sobre las relaciones de comunicación, las cuales permiten el intercambio de datos on-line.

#**Inclusión**
La co-operación de diversos bloques funcionales ha de ser sincronizada, si están dispuestos en red. La operación de los bloques funcionales está dirigida en concordancia a una inclusión fija que consiste en distintas fases de la misma longitud.
El tráfico de operación (Intercambio de datos entre bloques funcionales) y el tráfico de fondo (todos los otros intercambios de datos) están organizados por el mismo método de la dirección de sistema (SM).
La funcionalidad descrita más arriba capacita el procesamiento de los cuatro bloques funcionales de nuestro ejemplo acordes al diagrama mostrado en la figura 5.

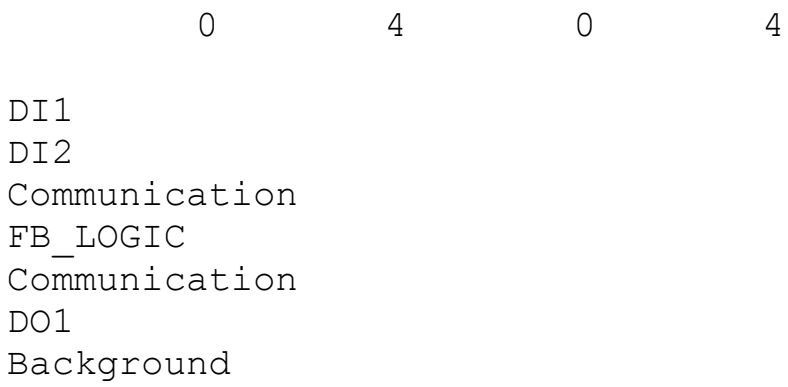


Fig. 5: diagrama de tiempos del ejemplo

El artículo mostrado arriba muestra algunas aproximaciones para construir Sistemas de Control Distribuidos usando una norma existente y aprobada. Esto ofrece ventajas tanto a los usuarios como a los vendedores. Las funcionalidades de los DDT y SM están especificadas, ellos pueden estar integrados en la intensificación del modelo de software de la IEC 1131-3. Una extensión de la norma existente parece posible y podría ser realizada siguiendo las preparaciones de TC4 y TC1 de PLCopen. Herramientas poderosas podrían ser construidas y usadas con esta base, capacitando a los usuarios a diseñar Sistemas de Control Distribuidos en el modo más efectivo posible.