PRÁCTICA

Tema: Instrucciones Add-On y subrutinas en Sombra digital

1. Objetivos

1.1. Objetivo general

Elaborar programación ladder, bloques de instrucciones y texto estructurado en subrutinas utilizando el controlador micro850 y el software Connected Component Workbench para una sombra digital de una estación didáctica clasificadora de cajas, comparando un proceso simulado en Factory I/O con su contraparte real en el laboratorio de automatización.

1.2. Objetivos específicos

1. Realizar funciones de bloques definidas por el usuario (UDFB) mediante programación de texto estructurado de instrucciones de comparadores y sentencias booleanas.
2. Utilizar funciones definidas por el usuario (UDF) mediante programación de bloques de instrucciones.
3. Integrar UDFB y UDF en el programa principal del proyecto para la realización de aplicaciones industriales.
4. Realizar un análisis comparativo entre el comportamiento del sistema simulado y el real, utilizando los datos recopilados por la sombra digital para identificar discrepancias, optimizar el proceso y proponer mejoras en la configuración del sistema.

2. Equipos y herramientas

- Micro850
- Connected Components Workbench
- RsLinx Classic
- Computadora
- Cables Ethernet.
- Switch Stratix
- Botoneras del tablero

3. Marco teórico

User-Defined Function

Funciones definidas por el usuario (UDF) son útiles para reutilizar la lógica del programa y hacer que sus programas sean más legibles. Es importante saber que se utiliza para un cálculo simple que requiere solo una salida.

Las UDF son similares a una subrutina ya que una UDF tiene parámetros de entrada y un único parámetro de salida. Las UDF no pueden acceder a las variables locales en el programa de llamada. Las variables locales en el programa de llamada deben pasarse a la UDF como parámetros de entrada.

Características de UDF

- Un parámetro de entrada definido por el usuario no se puede usar para habilitar o deshabilitar las UDF porque el parámetro de entrada solo puede habilitar o deshabilitar las instrucciones dentro de la UDF.
- Para habilitar o deshabilitar la ejecución de un UDF, seleccione la casilla de verificación EN / ENO en la ventana Selector de bloque de instrucciones para el UDF especificado. Por ejemplo, cuando EN es FALSO, el UDF no se ejecuta y los parámetros de salida no se sobrescriben.
- Si el programa que realiza la llamada ejecuta un UDF más de una vez por exploración del programa, no se recomienda utilizar una instrucción que requiera más de una exploración del programa para finalizar la ejecución. Esto incluye instrucciones que retienen el estado entre los escaneos del programa, tales como temporizador, movimiento, mensaje e instrucciones de contador.

User-Defined Function Block

Es un programa definido por el usuario que puede ser empaquetado dentro de un bloque de instrucciones. Este bloque puede reutilizarse dentro de un proyecto. Además, cuando se añade una UDFB se selecciona el lenguaje de programación vasado en el tipo de aplicación que se está desarrollando. Por ejemplo, el diagrama escalera para ejecutar lógicas booleanas simple, temporizadores y contadores. Diagrama de bloques y texto estructurado pueden ser más eficiente para procesos con instrucciones más avanzadas en esos lenguajes.

Ventajas de utilizar user-defined function blocks (UDFBs)

- *Reusar código*

  Usar UDFBs para algoritmos que se usan varias veces en el mismo proyecto. Añadir el código dentro de un UDFB para hacerlo modular y fácil de reutilizar.

- *Usar UDFB en lugar de UDF*

  Para cálculos complejos que tienen múltiples salidas.

  Al guardar valores de variables locales de ejecución en ejecución (se requiere guardar estado).

  Cuando se requieren varias instancias, el uso de un UDFB puede usar menos memoria que un UDF porque un UDFB en un proyecto no existe en un programa hasta que se instancia como una variable.

- *Proporcionar una interfaz más fácil de entender*

  Coloque algoritmos complicados dentro de un UDFB y luego proporciona una interfaz que sea más fácil de entender al mostrar solo los parámetros esenciales o requeridos.

  Reduzca el tiempo de desarrollo de la documentación insertando comentarios en cualquier lugar del UDFB. Los comentarios son solo para fines de documentación y el programa no actúa sobre ellos.

- *Simplificar el mantenimiento*

  Simplifica el mantenimiento del código porque la lógica de UDFB monitoreada en Connected Components Workbench muestra los valores de entrada y salida relativos a la instancia específica del UDFB.

- *Fácil restablecer valores iniciales de instancias*
  Use la ventana Refactorización para restablecer los valores iniciales para instancias de bloques de funciones definidas por el usuario

Utilizar un User-Defined Function en vez de un User-Defined Function Block

- Para un cálculo simple que requiere solo una salida como Y = MX + B
- Para instrucciones sin estado que no requieren guardar valores de variables locales de ejecución. Ejemplo: SIN es una instrucción sin estado.
- Cuando el parámetro de salida no requiere una matriz o tipo de datos estructurados.
- Siempre que sea posible porque un UDF consume menos memoria.
- Cuando se requieren varias instancias, el uso de un UDFB puede usar menos memoria que

un UDF porque un UDFB no existe en un programa hasta que se instancia como una variable.

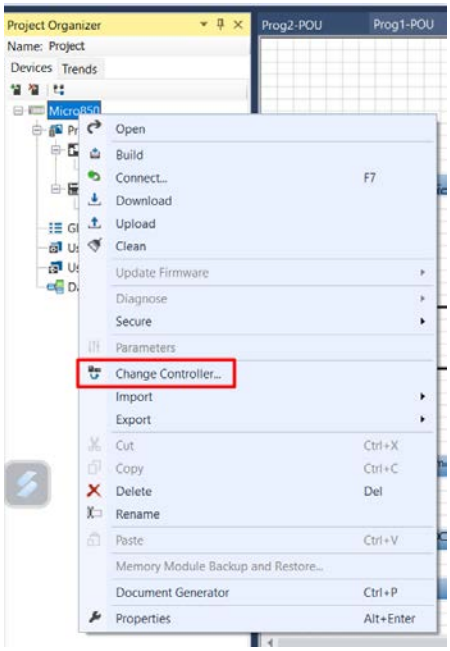
Utilice un bloque de funciones definido por el usuario en lugar de una función definida por el usuario

- Para cálculos complejos que tienen múltiples salidas.
- Al guardar valores de variables locales de ejecución en ejecución (se requiere guardar estado).
- Cuando se requieren varias instancias, el uso de un UDFB puede usar menos memoria que un UDF porque un UDFB en un proyecto no existe en un programa hasta que se instancia como una variable.
- Cuando el parámetro de salida requiere una matriz o un tipo de datos estructurados.
- Cuando se envía más de un mensaje simultáneamente, un UDFB podría ser una mejor opción que un UDF. Cuando un UDF contiene una instrucción de mensaje como MSG_CIPGENERIC, el UDF solo envía un mensaje a la vez, incluso si se llama al UDF varias veces por exploración de programa.

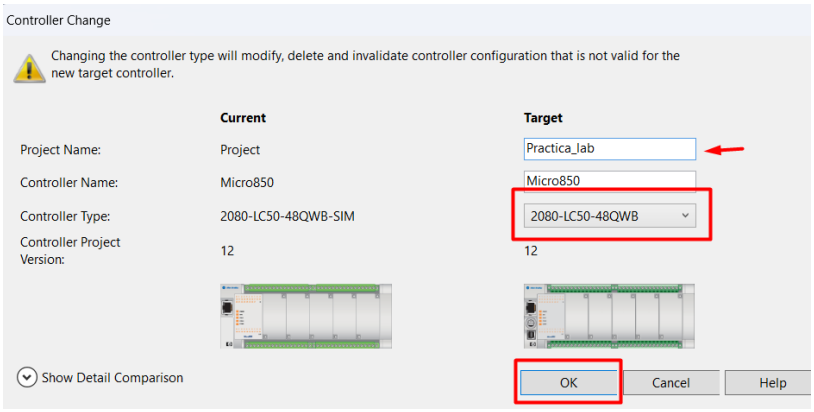
4. Actividades a desarrollar

4.1. Cambiar el controlador simulado del proyecto de la prepráctica al micro 850 con su respectiva IP del tablero utilizando el software Connected Component Workbench y RSLinx. Además, cambiar al menos el direccionamiento de una entrada y una salida del programa del proyecto para que se utilice botones/switches/luces piloto del tablero del laboratorio de automatización.

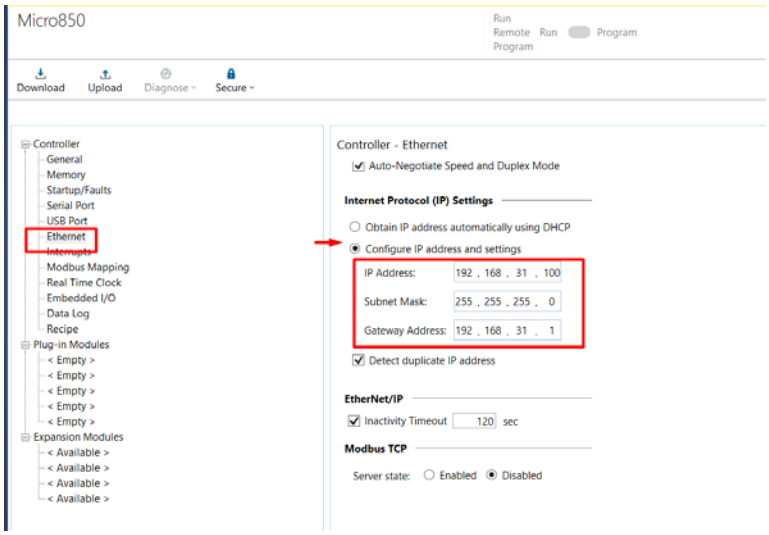
1. Para cambiar el controlador, dar clic derecho en el organizador del proyecto y seleccionar **Change Controller.**



2. Cambiar el nombre del proyecto y seleccionar el tipo de controlador de su tablero de acuerdo con el mostrado en RSLinx.



3. Luego, en la opción **Ethernet** configurar la IP de su controlador de acuerdo con la IP de su tablero. Por último, dar clic en **Download**.



4.2 Descargar la HMI de la prepráctica en un panelView con su respectiva IP.