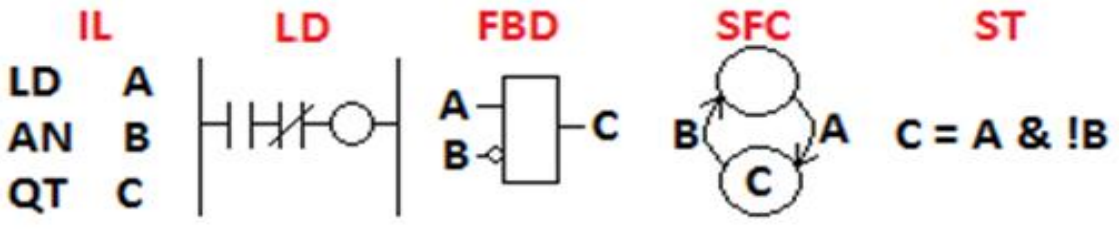
EL LENGUAJE DE LAS MÁQUINAS DE ESTADO CON MICROGRADES

INTRODUCCIÓN

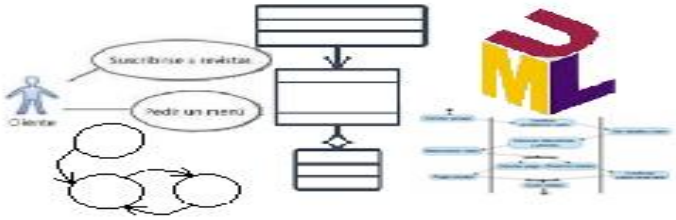
Los cortos tiempos para el desarrollo de soluciones tecnológicas, en la industria 4.0, exigen la existencia de herramientas ágiles y con curvas de aprendizaje cortas, que sean intuitivas para personas de diferentes áreas del conocimiento. La utilización de recursos tecnológicos, que aprovechan los logros en metodologías y lenguajes de programación, posibilita el desarrollo ágil de soluciones, observando las orientaciones de la norma IEEE61131-3, aplicable a diferentes estrategias de control para la automatización de máquinas y procesos productivos en el área de bienes y servicios.

MICROGRADES Y LA NORMA IEEE61131-3

La escritura de programas de control lógico y automatización, se desarrollaba, inicialmente en torno a lenguajes de contacto (LD) y a listas de instrucciones (IL). Con el avance de la informática y la popularización de diferentes tipos de control, la programación de dispositivos de control se hizo más rica en recursos, permitiendo la resolución de problemas más complejos. De esta forma, se hizo común el uso de bloques funcionales (FBD), diagramas de funciones secuenciales (SFC) y de texto estructurado (ST) como recursos emergentes de la norma IEEE61131-3.



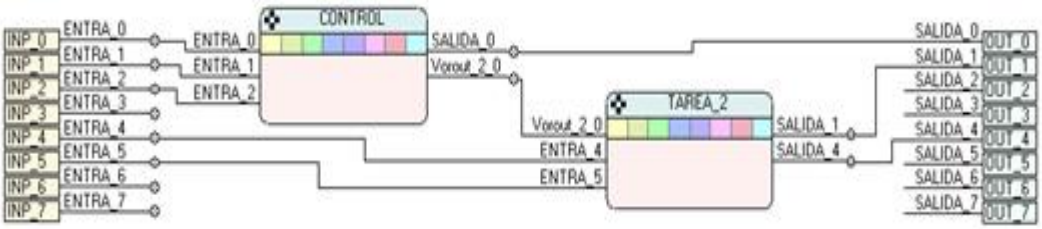
El desarrollo de nuevos métodos de modelado (UML) y de procesadores para el control automático, ha permitido el uso de herramientas que requieren de una curva de aprendizaje muy corta para llegar a desarrollar aplicaciones de control lógico y de variable continua.



La propuesta de Micrograde combina formas de representación UML con bloques funcionales (FBD) para presentar el flujo de información, también con diagramas de funciones secuenciales (SFC) para presentar el flujo de eventos y con texto estructurado (ST) para presentar la lógica de los programas.

BLOQUES FUNCIONALES (FBD)

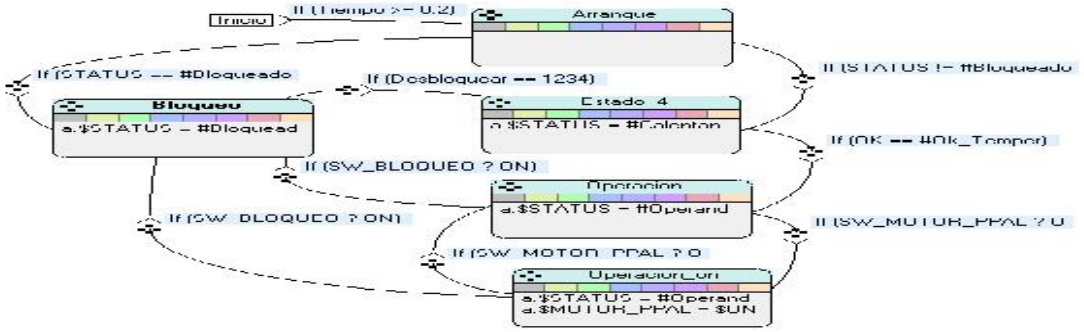
El bloque funcional se define como una unidad de organización del programa, la cual, al ser ejecutada por la CPU del autómata programable, retorna uno o más valores. Hay bloques funcionales que son llamados desde el RTOS (Tareas) y otros que se invocan desde las maniobras (Bloques de maniobras). Una aplicación de control es un programa compuesto por tareas, las cuales son contenedores de secuencias (SFC) que facilitan la interconexión de variables de forma estructurada y con proyección modular.



Las tareas constituyen una vista de los flujos de datos del programa y son un recurso que permite dividir secuencias complejas en grupos de secuencias simples. El sistema operativo (RTOS) del autómata programable, invoca las tareas con una periodicidad establecida en la configuración de cada tarea. La representación de las tareas corresponde, al análisis funcional del problema de automatización a través de cajas negras, con variables de entrada, variables de salida y la definición de la función que realizan.

DIAGRAMAS DE ESTADO (SFC)

Los diagramas de funciones secuenciales (SFC) representar el flujo de eventos en un sistema y se realizan a través de diagramas (máquinas) de estado, los cuales son una forma de representación que permite modelar las secuencias de operación de los sistemas automatizados. A la vez son una de las formas simples, para la interpretación de la operación de los sistemas y los procesos. Los diagramas de estado están compuestos por estados, que son intervalos de tiempo en los cuales las salidas del sistema están en un régimen estable y por transiciones, que son los cambios de un estado al otro.

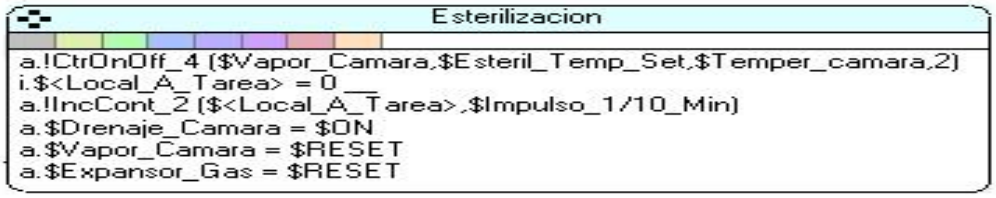


Los estados están compuestos por maniobras que son las acciones directas que se realizan sobre las salidas del controlador para actuar sobre el sistema. Las transiciones están compuestas por flechas que muestran la dirección del flujo de la operación y condiciones que permiten que ocurra la transición. Las condiciones y las maniobras se escriben a través de texto estructurado.

TEXTO ESTRUCTURADO (ST)

El texto estructurado es un conjunto de sentencias que permiten construir la lógica de los programas, mediante la realización de operaciones que leen y escriben los operandos del autómata programable (variables y constantes). En la norma IEEE 61131-3 se utiliza el lenguaje Pascal como base de fundamentación. En el entrono Mgdmod, se utiliza la sintaxis de los lenguajes basados en Ansi (C, Java, C#, Php, Perl) ISO/IEC 9899, en virtud del código que genera y de la difusión que tiene entre los programadores.

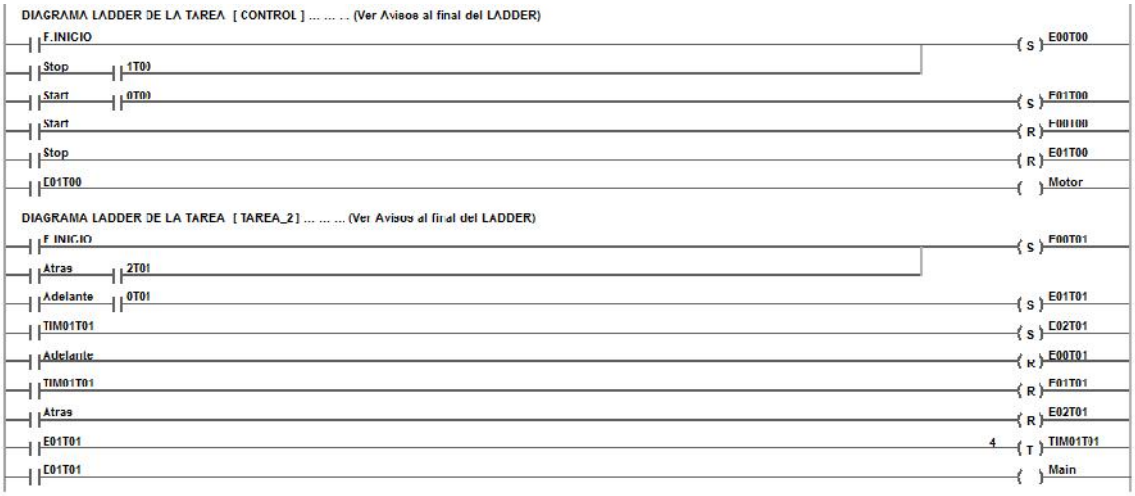
El texto estructurado se compone de diferentes estructuras de escritura como son las de asignación, las de comparación, las de repetición y las funcionales. En el entorno Mgdmod se arman sentencias de programación utilizando el click del mouse (sin necesidad de digitar texto), con la finalidad de evitar errores de sintaxis y de agilizar la creación de maniobras y condiciones.



Para editar el texto estructurado de las maniobras y condiciones, se debe hacer click (del mouse) sobre el término que se desee modificar, basados en que cada expresión de texto estructurado está compuesta por términos separados por espacios (variables, símbolos de operación, argumentos, constantes, etc). Al pulsar click sobre un término específico, se despliega un menú, el cual presenta las opciones apropiadas para cada caso.

LENGUAJE DE ESCALERA - LADDER (LD)

El diagrama de programación en lenguaje LADDER (LD) es generado por el entorno de desarrollo, a partir del programa creado con FBD, SFC y ST. Aunque no todas las expresiones del ST pueden ser convertidas a LD, sí se puede hacer con la mayoría de estas.



La creación del diagrama en LD se crea para cada bloque de FBD bajo la estrategia de SET-RESET. Se crea una bandera (marca) por cada estado. Se crean los SET de las banderas como el resultado de la conexión en serie, de la entrada consultada en las flechas (transiciones) del SFC que llegan al estado correspondiente y la bandera del estado del que sale la flecha (Por cada flecha se crea una rama en paralelo). Se crean los RESET de las banderas, como la conexión en paralelo de las entradas consultadas en las flechas (transiciones) del SFC, que salen del estado correspondiente. Para el conteo de tiempo dentro de un estado, se agrega un temporizador activado por la bandera del estado correspondiente. Las salidas se activan con la conexión en paralelo de los estados en los cuales está activada la salida.

SENTENCIAS DE PROGRAMACIÓN CON TEXTO ESTRUCTURADO

El texto estructurado de Mgdmod se compone de cinco tipos fundamentales de sentencias, las cuales se describen de manera detallada a continuación.

SENTENCIAS DE ASIGNACIÓN SIMPLE: Se componen de una variable (generalmente de salida de la tarea), de un símbolo de operación y de un argumento (variable o constante).

```
$Motor = $ON          La salida Motor está activa mientras la secuencia este en el estado.
$Motor = $SET         ($RESET, $TOGGLE)
$<Local_A_Tarea> = $ENTRA_0 __
$<Local_B_Tarea> += #Uno
```

SENTENCIAS DE ASIGNACIÓN COMPUESTA: Se componen de una variable (generalmente de salida de la tarea), de un símbolo de asignación, de un argumento (variable o constante), de un símbolo de operación y de un complemento (variable o constante).

```
$<Local_A_Tarea> = $ENTRA_0 + 24
```

SENTENCIAS DE COMPARACIÓN: Hay tres casos de utilización de las sentencias de comparación (en las transiciones, maniobras únicas y grupos de maniobras).

Las sentencias de comparación en las condiciones de las transiciones, son parte de un grupo de sub expresiones encerradas entre paréntesis y unidas por conectivos lógicos (an = AND y or = OR). Cada sentencia se compone de una variable a comparar, de un símbolo de comparación y de un argumento de comparación (variable o constante).

If (ENTRA_0 ? ON) *Consulta si la variable está activa (Verdadero)*

If (ENTRA_0 >= 20) *Consulta si la variable es mayor o igual a 20*

If (Tiempo >= 5) *Consulta han pasado 5 segundos en el estado ctual*

If (ENTRA_1 >= 6) an (Tiempo >= 2)

Las sentencias de comparación de expresión única se utilizan en las maniobras para asignar uno de dos posibles valores, tras el cumplimiento o no, de una condición. Están compuestas por una variable (generalmente de salida de la tarea), de un símbolo de asignación, de una comparación encerrada entre paréntesis, un argumento (variable o constante) que se asigna si la condición es verdadera, dos puntos y un argumento (variable o constante) que se asigna si la condición es falsa. La condición se compone de una variable a comparar, de un símbolo de comparación y de un argumento de comparación (variable o constante).

$SALIDA_0 =($ENTRA_0 > #Uno)? $ENTRA_1 : 0 *La salida_0 se carga con el valor de Entra_1 cuando Entra_0 es mayor que 1, o se carga con 0 en caso contrario*.

$SALIDA_1 =($Comando_1 == 3)? 1 : SALIDA_1 *La salida_1 se carga con el valor 1 cuando Comando_1 es igual a 3, o continúa con su valor en caso contrario*.

Las sentencias de comparación con grupos de maniobras habilitan la ejecución de un grupo de maniobras en caso de cumplirse una condición. Estas sentencias se componen de dos líneas (condición y cierre del grupo de maniobras). La condición se compone de una variable a comparar, de un símbolo de comparación y de un argumento de comparación (variable o constante), encerrados entre dos paréntesis con una llave de apertura“{” al final. La línea de cierra tiene una llave de cierre “}” indicando el final del grupo de maniobras.

If($ENTRA_0 >= 2){
$SALIDA_0 = $Impulso_1_Seg __
$<Local_A_Tarea> = 243 __
}

SENTENCIAS DE REPETICIÓN: Hay dos tipos de repeticiones (incrementales y decrementales), las cuales se componen de dos líneas (lanzamiento del ciclo y cierre del grupo de maniobras). El lanzamiento del ciclo se compone de una variable de conteo (generalmente de “Especiales”), de una igualación al valor inicial de conteo, de una comparación menor/mayor que el valor posterior/anterior al máximo/mínimo y de un valor de incremento/decremento del conteo. Estos términos están encerrados entre paréntesis, precedidos por la palabra clave “for” y terminado con una llave de apertura“{”. La línea de cierra tiene una llave de cierre “}” indicando el final del grupo de maniobras que se han de repetir.

For($<Acumulador_A> = 0; < 4; + 1){
$Monitor_1[$<Local_A_Tarea>] = 0 __
}
For($<Acumulador_B> = 4; > 0; - 1){
$Monitor_2[$<Local_B_Tarea>] = $Setpoint_1[$<Local_B_Tarea>] __
}

SENTENCIAS FUNCIONALES: Son llamados a funciones del RTOS y de usuario. Se componen del nombre de la función (terminado en la cantidad de argumentos) y los argumentos encerrados entre paréntesis (variables o constantes).

¡DecCont_4(*$<Local_A_Tarea>,$Impulso_1_Seg,#Cero,#Uno*)

¡AsigEncod_3(*$<Local_B_Tarea>],$ENTRA_0,$ENTRA_1*)

¡QuitStep_3(*$SALIDA_0,$ENTRA_1,50*)

EDICIÓN DE LAS CONDICIONES DE LAS TRANSICIONES

Las transiciones son los cambios de un estado a otro y son suscitados por el cumplimiento de una condición, la cual está compuesta por sentencias encerradas entre paréntesis con conectivos lógicos (an, or) entre ellos. Por defecto cada condición tiene por lo menos una sentencia encerrada entre paréntesis.

Las sentencias de las condiciones se agregan pulsando click sobre el símbolo "__" ubicado al final de las condiciones o sobre los conectivos (an, or). El menú que se despliega en tal caso ofrece opciones de cambio o adición del conectivo y de borrado de la sentencia ubicada a la izquierda o a la derecha del conectivo. Las condiciones se construyen sólo con sentencias conectadas en forma lineal, sin anidamientos, las cuales serán evaluadas de izquierda a derecha, sentencia por sentencia.

If (sentencia 1) an (sentencia 2)) or (sentencia 3)) an (sentencia 4)

Primero se revisa la sentencia 1 y luego la 2, para calcular el resultado parcial (booleano) de la operación AND (Y lógico) entre la 1 y la 2. Luego se revisa la sentencia 3 y se hace la operación OR entre el resultado parcial y la 3. Finalmente se revisa la sentencia 4 y se realiza la función AND como resultado final.

Cada sentencia se compone de una variable a comparar, de un símbolo de comparación y de un argumento de comparación (variable o constante). Hay dos grupos de comparaciones que son las booleanas, de magnitud y temporales. Las comparaciones booleanas utilizan el símbolo "?" para consultar si una variable está activa (ON), si está desactivada (OFF) o si ha sufrido un cambio de OFF a ON (EDGE = Flanco ascendente). Las comparaciones de magnitud revisan si la primer variable es > ; >= ; < ; <= ; == ; ¡= de la segunda. Las comparaciones temporales consultan el tiempo transcurrido en el estado de la transición, para lo cual consultan la variable "Tiempo" y comparan si es >= al tiempo programado en horas : minutos : segundos . centésimas.

If(Variable_a_comparar ? ON/OFF/EDGE)

If(Variable_a_comparar >/>=/</<=/==/¡= Variable/constante)

If(Tiempo >= Variable/constante)

EDICIÓN DE LAS MANIOBRAS

Las maniobras son las expresiones escritas en texto estructurado, encargadas de realizar acciones sobre las salidas del autómata programable. Estas están explicadas en la sección de “Sentencias de programación con texto estructurado” y tienen una forma de edición, basada en la pulsación de click sobre el término que se desee modificar.

Al pegar una maniobra nueva, aparece la expresión “No_Sale = No_Acción”, la cual es la base de edición (no realiza ninguna acción sobre las salidas). Haciendo click sobre la expresión aparece un menú desplegable con diferentes opciones, la lista de variables especiales y la lista de variables de salida de la tarea. Al pegar la variable aparece la sentencia simple “$Variable = $ON”, en la cual se puede hacer click sobre cada uno de los tres términos y en cada caso aparece un menú diferente (variables de salidas / símbolos de operación / variables de entrada o salida).

Las sentencias simples que tienen un vínculo doble al final, permiten agregar un símbolo de operación y un operando adicional, para construir sentencias compuestas.

$<Local_A_Tarea> = $ENTRA_0 __ → $<Local_A_Tarea> = $ENTRA_0 + 45

Las maniobras de asignación booleana cargan valores en las variables, cargando la variable con 0 cuando se asigna $RESET (False), un valor de 65535 al asignar $SET o $ON (True).

Cuando se asigna $ON, la salida permanece activa durante el estado de la maniobra y se desactiva automáticamente al salir del estado. Al utilizar $SET, la variable estará activada hasta que aparezca una asignación $RESET sobre la misma variable.

Las variables enteras utilizadas en las maniobras, soportan valores no-signados entre 0 y 65535. Las entradas del autómata operan por defecto entre 0 y 1023 (escala que pueden ser modificada para cada entrada independientemente).

Las maniobras tienen al lado izquierdo una letra que determina su contexto de operación. “a.” (allways) indica que la maniobra opera todo el tiempo, mientras el estado esté activo. “i.” (initializing) indica que la maniobra solamente opera en el momento de inicio del estado. “f.” (finishing) indica operación sólo en el momento de finalización del estado.

OPERANDOS UTILIZADOS EN EL TEXTO ESTRUCTURADO

El texto estructurado de las maniobras está constituido por sentencias, en las cuales se modifican las variables de salida de la tarea (también las especiales) mediante operaciones con variables y constantes, las cuales son los operandos, con los cuales se construyen las sentencias de las maniobras. Los operandos tienen una longitud de 16 bits y manipulan valores no-signados en el rango [0 – 65535], operables en forma numérica o booleana y con posibilidad de agrupación en arreglos.

A continuación se presentan algunas definiciones relacionadas con los operandos utilizados en el texto estructurado de Micrograde s.

VARIABLES. Son los operandos que pueden ser modificados en las maniobras, generalmente salidas de las tareas (salidas del autómata, variables de monitoreo, variables no volátiles y especiales). También son variables los operandos que reciben valores desde afuera (entradas físicas del autómata, comandos y variables de monitoreo del bus Modbus).

Existe un tipo particular de variables (ajustes en memoria flash), los cuales sólo son modificables desde el bus Modbus, y desde las maniobras, con funciones de grabación.

CONSTANTES. Son los operandos que no son modificables en las maniobras (valores numéricos, constantes booleanas, definiciones de constantes, mensajes de HMI). Los operandos constantes son globales a todo el programa, o sea que se pueden utilizar en todas las tareas, sin necesidad de conectarlas a cada tarea. Las constantes solamente pueden ser modificadas al editar el programa que se descargará hacia el autómata programable.

Cuando es necesario modificar constantes, durante el funcionamiento del programa del autómata, se debe utilizar un operando tipo “Ajuste” en lugar de una constante.

ARREGLOS. Algunas variables (comandos, monitores y no-volátil) y constantes (ajustes y tablas) pueden ser configuradas como arreglos, con el fin de crear programas más flexibles, que permitan construir acciones iterativas, procesamiento de recetas, programación de procesos, colas y filas de órdenes. Los arreglos se crean pulsando click derecho sobre los operandos y seleccionando la opción “Presentar como arreglo”. Las tablas se crean o modifican desde el menú de variables globales.

En el grupo de operandos utilizados para desarrollar proyectos con Micrograde, se destaca una serie de variables y estrategias que facilitan el desarrollo de soluciones de automatización con los autómatas programables.

COMANDOS. Son variables que son modificadas por las comunicaciones externas (Modbus, bluetooth, expansión). Se utilizan como entradas de las tareas para la recepción de órdenes remotas y setpoints volátiles.

AJUSTES. Son constantes modificables (no-globales), las cuales pueden ser modificadas desde las comunicaciones externas o a través de funciones de grabación. Se utilizan como entradas de las tareas para la recepción de setpoints no-volátiles.

MONITORES. Son las variables internas del programa (soportan lectura desde las comunicaciones externas), utilizadas como salidas de las tareas, para guardar valores de los procesos y para enviar mensajes de una tarea a la otra.

NO-VOLÁTIL. Es un grupo pequeño de variables (soportan lectura desde las comunicaciones externas), las cuales se guardan en flash al des-energizar el autómata y se recuperan desde flash al energizar de nuevo. Se utilizan como salidas de las tareas para llevar conteos y guardar el status de los procesos. El almacenamiento, en flash, se habilita desde las propiedades de la aplicación, en la casilla “Activar la detección de baja tensión”.

CONSTANTES ENTERAS. Se utilizan directamente como valores numéricos en las maniobras y las transiciones. Pueden ser declaradas desde la opción “Editar constantes” para ser utilizadas desde cualquiera de las tareas (operandos globales).

CONSTANTES BOOLEANAS EN LAS CONDICIONES. Son los casos de $ON=Entrada activada, $OFF=Entrada desactivada, $EDGE=Cambio de OFF a ON (el instante durante un ciclo de programa). $ON es “>= umbral” para las entradas físicas y “> 0” para las demás variables.

CONSTANTES BOOLEANAS EN LAS MANIOBRAS. Son los casos de $ON=Salida activa durante el estado de la maniobra y desactivada por fuera de éste, $SET=Salida activada hasta que aparezca una asignación $RESET sobre la misma variable. $ON y $SET asignan el valor 65535 y $RESET asigna el valor 0.

TABLAS CONSTANTES. Son grupos de constantes agrupadas en tablas, las cuales pueden ser leídas mediante expresiones con la forma “tabla_1[$Indice]”. Se utilizan para linealizar entradas, buscar índices y hacer comparaciones indexadas.

REDIRECCIONES. Son accesos a operandos en forma indexada, los cuales permiten agrupar en un mismo arreglo, diferentes variables (aunque no estén contiguas), posibilitando su utilización en ciclos para crear programas flexibles. Se crean desde el menú “Editar globales”, seleccionando la opción “Editar Redirecciones”. La sentencia “*$redir_2[1] = 7”* asigna el valor 7 a la variable que esté en la posición 1 en la tabla redir_2.

VARIABLES ESPECIALES. Son un grupo de variables que ofrece el sistema operativo (RTOS), sin necesidad de conectarlas a las tareas (unas son propias de cada tarea y otras globales). Los acumuladores (<Acumulador_A> y <Acumulador_B>) se utilizan para pasar valores entre maniobras de un mismo estado o bloque de maniobras. Las variables locales (<Local_A_Tarea>, <Local_B_Tarea> y <Local_C_Tarea>) se utilizan para pasar valores entre estados de una misma tarea y para llevar conteos. Adicionalmente hay otras variables que permiten revisar el estado del RTOS y enviarle órdenes especiales.

ENTRADAS FÍSICAS. Son variables de sólo lectura, las cuales se conectan a las entradas de las tareas para ser consultadas por las condiciones de las transiciones y por las maniobras. Las entradas físicas son escalables individualmente. Se puede configurar el umbral que establece cuando está activada o desactivada y la lógica positiva o negativa. Pueden ser configuradas para operar como entradas rápidas y de encoder, a través de funciones.

SALIDAS FÍSICAS. Son variables de lectura-escritura, las cuales se conectan a la salida de las tareas, para ser modificadas desde las maniobras, en forma booleana o con valores enteros. Su comportamiento puede ser modificado desde funciones tipo QuitYyy, para el manejo de modulaciones, motores de paso, servo-motores y relevos dimerizables por ángulo de fase.

VARIABLES DE TIEMPO. Cada secuencia (máquina de estado) cuenta el tiempo de cada estado, utilizando una variable de conteo de tiempo de la tarea. Las transiciones consultan por el tiempo transcurrido a través de la sentencia *“if(Tiempo >= 2)”*. El conteo de tiempo se reinicia con los cambios de estado o mediante la utilización de la función “TimStatInic”. La variable de conteo de tiempo está en “Especiales” como <Tiempo_Tarea>.