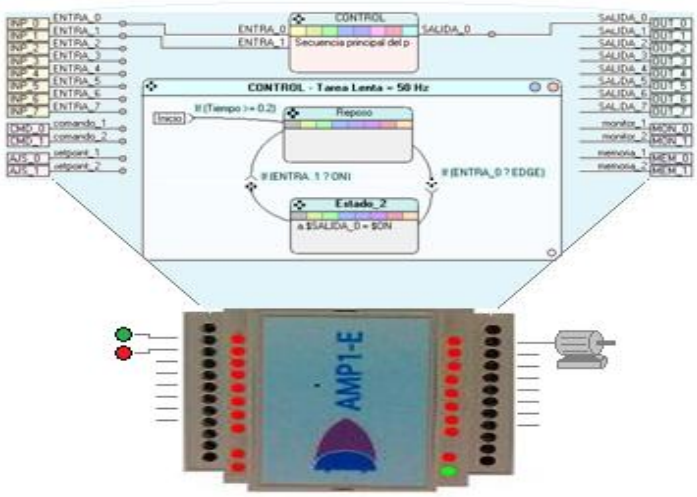
MANIOBRAS FUNCIONALES EN MGDMOD



INTRODUCCIÓN.

El desarrollo de soluciones de automatización de sistemas y procesos, demanda de recursos de programación que sean ágiles y que tengan curvas de aprendizaje cortas. Es por esto que la utilización de autómatas programables con características de interconexión versátil, debe ir acompañada de recursos de software que permitan explotar al máximo la potencia del autómata para lograr los mayores beneficios para el integrador de soluciones y sus clientes.

El control lógico con máquinas de estado, es una propuesta que responde a las necesidades de efectividad en la implementación de soluciones tecnológicas. Aun así estas no responden a los todas las necesidades de automatización. En este caso se hace fundamental el uso de maniobras funcionales para ampliar las posibilidades del autómata programable.

LAS MANIOBRAS DEL MGDMOD.

Las maniobras son las expresiones escritas en texto estructurado, encargadas de realizar acciones sobre las salidas de las tareas de las aplicaciones de automatización. Estas tienen una forma de edición, basada en la pulsación del botón izquierdo del mouse, sobre el término que se desea modificar. Existen maniobras de asignación simple y compuesta, de asignación por comparación, de ejecución condicional y cíclica.

LAS MANIOBRAS FUNCIONALES.

Las maniobras funcionales son la respuesta a las necesidades de extensión de características de los autómatas programables. Las maniobras funcionales permiten realizar cálculos complejos con baja sobrecarga de tiempos y código, esto en virtud a que son realizadas en el código nativo del procesador de los autómatas programables.

Existen maniobras funcionales de cálculo de resultados, de configuración de entradas, de configuración de salidas y de bloques de maniobras. Las maniobras funcionales Se componen del nombre de la función (terminado en la cantidad de argumentos) y los argumentos encerrados entre paréntesis (variables o constantes).

¡NombreFuncion_#(Variable_Salida, *Argumento_1*, … , *Argumento_N*)

Las maniobras funcionales han sido clasificadas, de acuerdo a su funcionalidad, en los siguientes grupos:

Funciones sobre las entradas. Configuración y filtrado de variables de las entradas de Amp.

Funciones de conteo y temporización. Realización de conteos de tiempo y eventos.

Funciones de control. Control y manipulación de variables continuas.

Funciones sobre las salidas. Configuración y envío de valores a las salidas del Amp.

Funciones con arreglos de operandos. Manipulación de arreglos de variables y constantes.

Funciones de interfaz hmi. Manejo de display LCD 2x16 con teclados matriciales.

Funciones de usuario con bloques de maniobras. Invocación y paso de parámetros a funciones conformadas por bloques de maniobras.

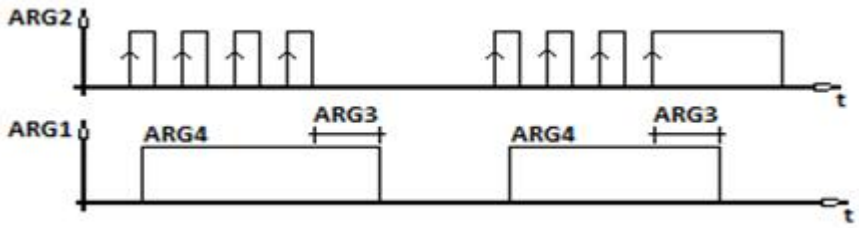
Funciones textuales con código de usuario. Invocación y paso de argumentos a funciones Codificadas en lenguaje C.

MANIOBRAS FUNCIONALES SOBRE LAS ENTRADAS.

PULSEDETEC_4. Detector de pulsos existentes en una entrada. Detecta que haya pulsos en la entrada ARG2 para cargar ARG1 con ARG4. Si durante ARG3 ciclos no hay pulsos en ARG2, se carga ARG1 con la negación de ARG4.

If (ARG2(i) != ARG2(i + ARG3)) ARG1 = ARG4.

ARG1 = Variable de salida.
ARG2 = Entrada a detectar.
ARG3 = Cantidad de ciclos.
ARG4 = Lógica de salida.

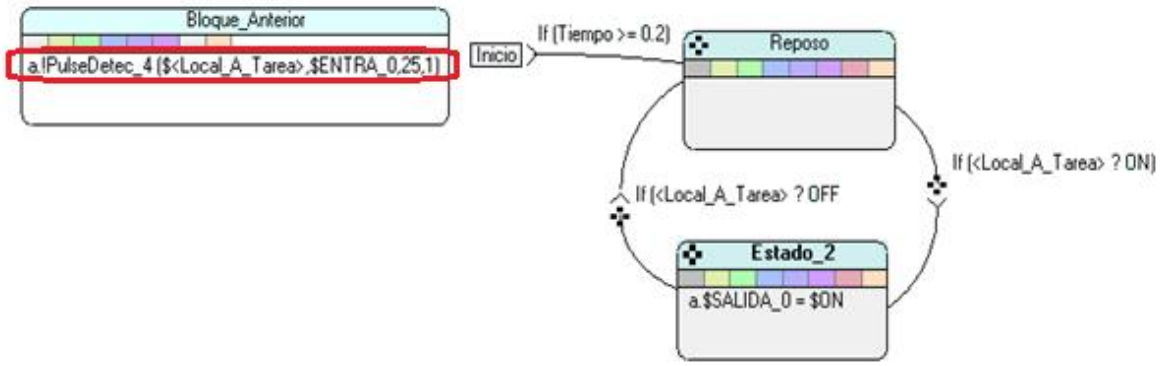


Aplicaciones: Se usa para detectar la integridad de los trenes de pulsos.

Activación de calentadores, mientras haya flujo de agua.

Operación de un molino mientras esté entrando materia prima.

Energizado de un motor de lubricación mientras un eje esté girando.

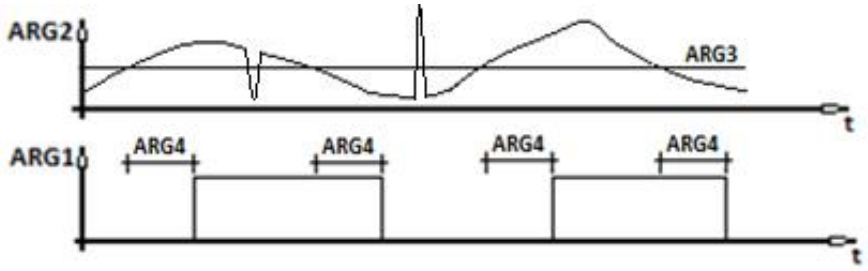


En el bloque anterior se detecta que haya pulsos en ENTRA_0, hacia <Local_A_Tarea>, cuando haya pulsos con un período menor a 25/50 de segundo. Mientras lleguen pulsos por ENTRA_0, se energiza SALIDA_0. (Se des energiza cuando cesan los pulsos)

DETCHANGE_4. Detección booleana temporizada de nivel. ARG1=1 si ARG2 >= ARG3 al cabo de ARG4 ciclos de la Tarea. ARG1=0 si ARG2 < ARG3 al cabo de ARG4 ciclos. La variable ARG1 conserva su estado en caso contrario. ARG3 está en unidades de ADC.

If (Abs(ARG2(i)-ARG2(i - ARG4))>ARG3) ARG1 = ON : OFF.

ARG1 = Variable de salida.
ARG2 = Variable a detectar.
ARG3 = Umbral de detección.
ARG4 = Cantidad de ciclos.

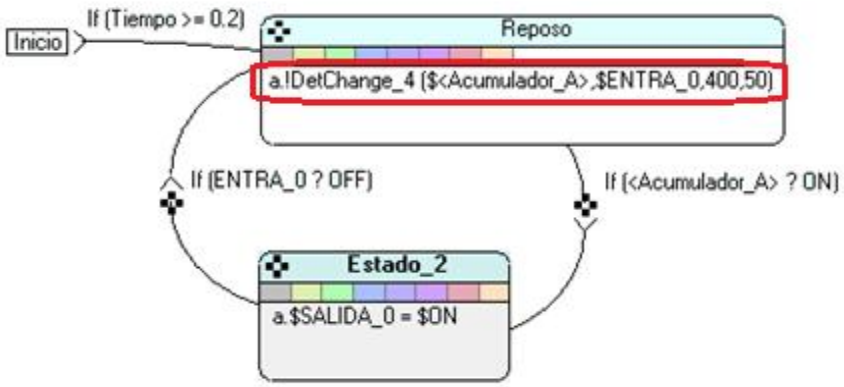


Aplicaciones: Se utiliza para filtrar en amplitud y frecuencia, entradas digitales ruidosas.

Temporización ON Delay (Retardo al trabajo)

Temporización OFF Delay. (Retardo al reposo)

Eliminación de picos de entrada tras el disparo de cargas inductivas.

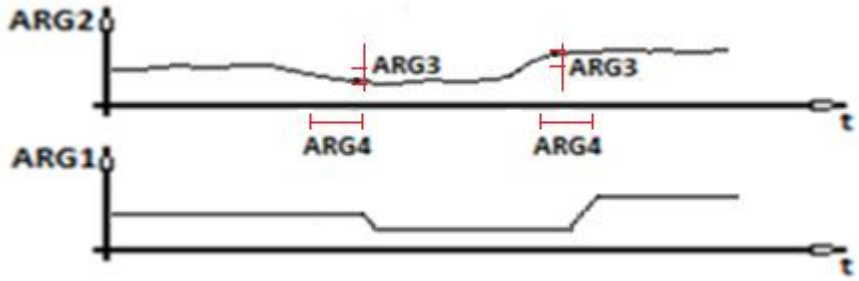


Cuando pasa un segundo con la variable ENTRA_0 mayor a 400, se activa la variable <Acumulador_A>, tras lo cual, la secuencia pasa al Estado_2 y se energiza la SALIDA_0. Tan pronto como se desactive la ENTRADA_0, se regresa al Estado de reposo. Esta secuencia constituye un retardo al trabajo.

CHNGFILTER_4. Filtro de detección de cambios en la variable ARG2 hacia ARG1 cuando el cambio de ARG2 es mayor que ARG3 al cabo de ARG4 ciclos de la Tarea. La variable ARG1 conserva su valor en caso contrario.

If (Abs(ARG2(i)-ARG2(i - ARG4))>ARG3) ARG1 = ARG2.

ARG1 = Variable de salida.
ARG2 = Entrada a filtrar.
ARG3 = Umbral de comparación.
ARG4 = Cantidad de ciclos.

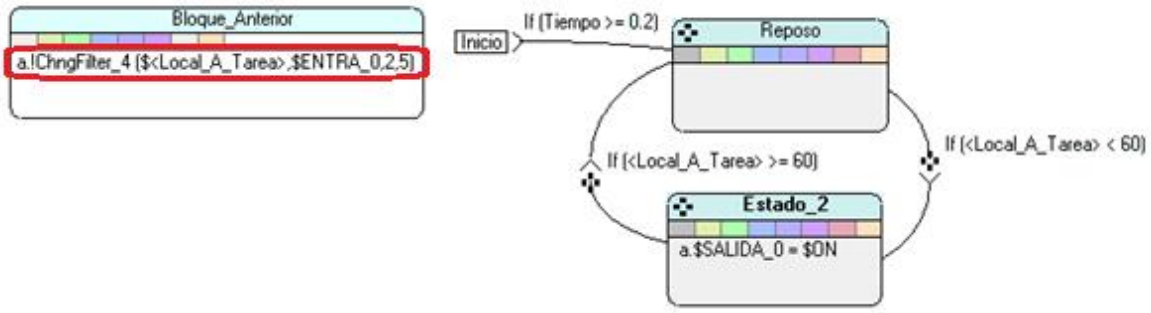


Aplicaciones: Se usa para filtrar cambios en variables con ruidos.

Eliminador de fluctuaciones en la presentación de una variable en HMI.

Adición de banda muerta para control ON-OFF por estados.

Filtrado de entradas desde selectores para selección de programas.

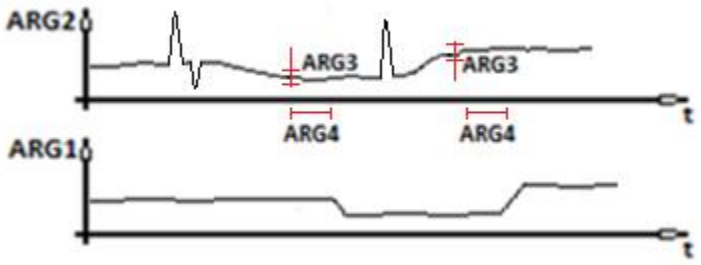


La variable <Local_A_Tarea> se carga con el valor de la entrada ENTRA_0, cada que esta última sufre cambios mayores a 2 unidades (banda muerta del control), durante 5 / 50 de segundo (0.1 segundos). La secuencia realiza un control ON-OFF sobre la SALIDA_0 comparando la variable <Local_A_Tarea> con el setpoint 60.

STABFILTER_4. Filtro de estabilización de la variable ARG2 hacia ARG1 cuando el cambio de ARG2 es menor que ARG3 al cabo de ARG4 ciclos de la Tarea. La variable ARG1 conserva su valor en caso contrario.

If (Abs(ARG2(i)-ARG2(i - ARG4))<ARG3) ARG1 = ARG2.

ARG1 = Variable de salida.
ARG2 = Entrada a filtrar.
ARG3 = Umbral comparación.
ARG4 = Cantidad de ciclos.

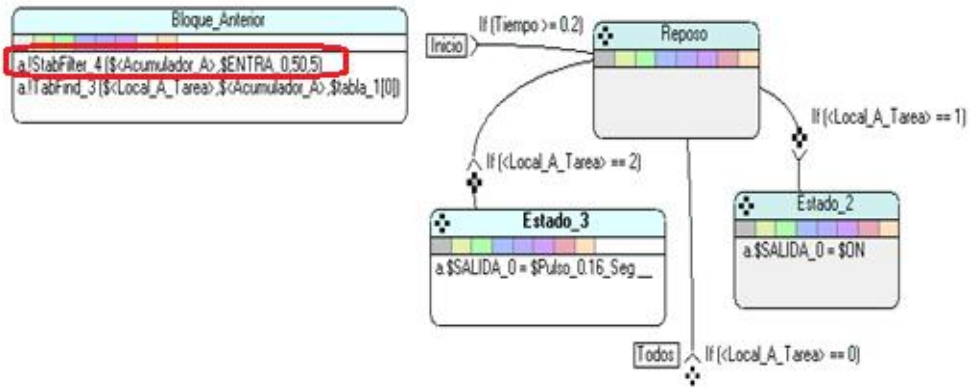


Aplicaciones: Se usa para estabilizar variables usadas en TabFind.

Eliminación de valores transitorios (picos) en las entradas del autómata.

Filtrado de entradas desde selectores para selección de programas.

Retardos de activación – desactivación de entradas digitales.

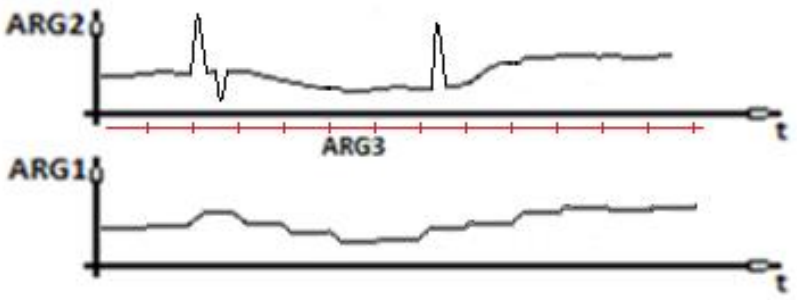


La función Stabfilter carga el valor de ENTRA_0, en la varible <Acumulador_A>, si los cambios en la primera, son menores a 50 unidades en un transcurso de una décima de segundo (5 / 50). El valor de la variable <Acumulador_A>, se utiliza en la función TabFind para cargar en <Local_A_Tarea> el índice del programa a seleccionar, el cual determina a cual estado de la secuencia, pasa el control.

PROMFILTER_3. Filtro de promedio de la variable ARG2 hacia ARG1 cada ARG3 ciclos de la Tarea. El producto del máximo valor de ARG2 y el valor de ARG3 debe ser menor que 65536.

ARG1 = (Suma(ARG2(i=[0, ARG3])) / ARG3.

ARG1 = Variable de salida.
ARG2 = Entrada a filtrar.
ARG3 = Cantidad de ciclos.

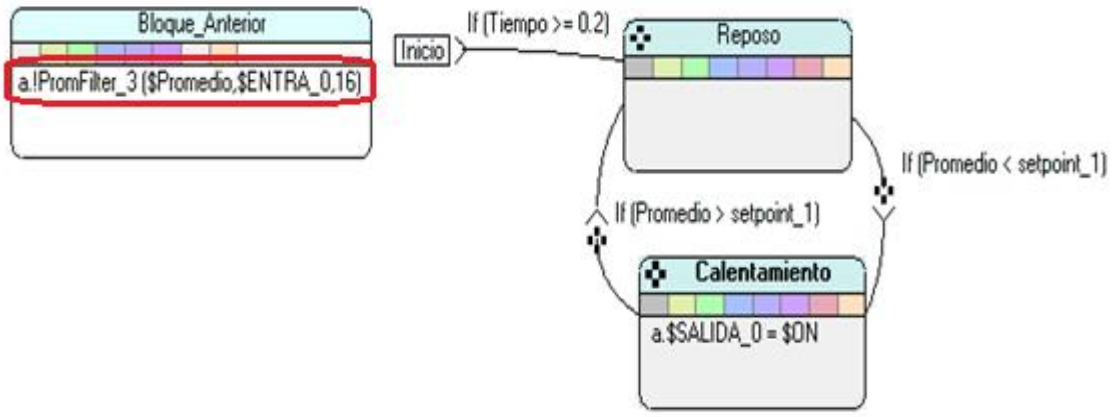


Aplicaciones: Se usa para filtrar variables con ruidos de baja frecuencia.

Eliminación de valores transitorios (picos) en las entradas del autómata.

Filtrado de entradas en aplicaciones de control de temperatura.

Filtrado de variables para su presentación a través de interfaces HMI.

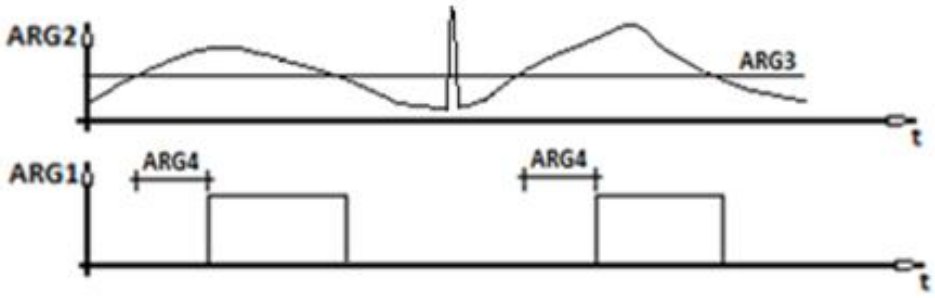


La función “PromFilter” carga la variable “Promedio” con el resultado de la suma de los valores de ENTRA_0, dividida entre 16. El cálculo se hace a una velocidad de 50 veces por segundo. Luego en la secuencia, se compara el promedio con un set Point para hacer un control a través de la variable SALIDA_0.

POSFILTER_4. Filtro positivo de entrada analógica. Filtra la entrada ARG2 hacia la variable ARG1, activándola cuando la entrada sea mayor que el umbral superior durante los ciclos de retardo ARG4. (El umbral está en unidades de ADC)

If ((ARG2 > ARG3) && (Tiempo > ARG4)) ARG1=1.

ARG1 = Variable resultado.
ARG2 = Variable entrada.
ARG3 = Umbral superior.
ARG4 = Ciclos de retardo.

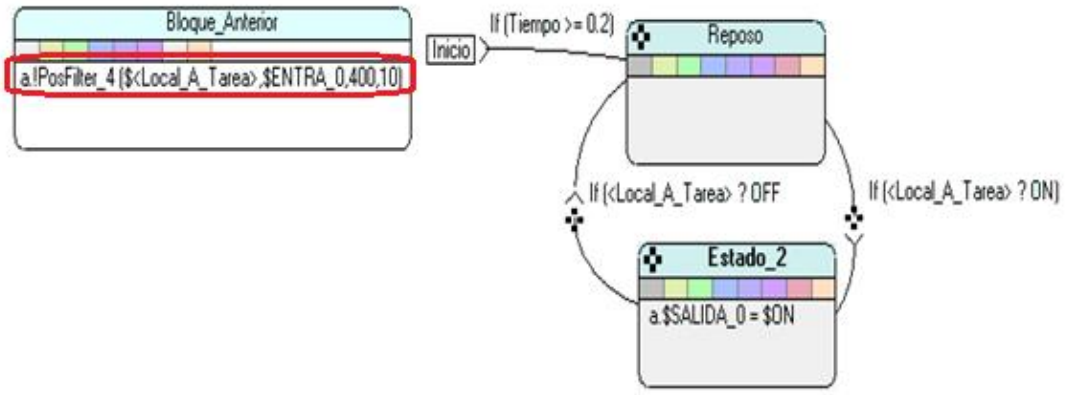


Aplicaciones: Se utiliza para filtrar en amplitud y en frecuencia entradas digitales ruidosas.

Eliminador de fluctuaciones en la presentación de una variable en HMI.

Eliminación de picos positivos en las entradas del autómata.

Temporización ON Delay (Retardo al trabajo)

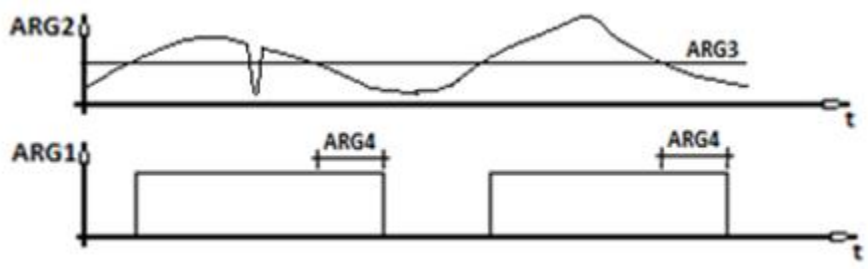


La función PosFilter activa la variable <Local_A_Tarea> cuando el valor de ENTRA_0 es mayor que 400 unidades de ADC durante 200 ms (10/50) ó más, y la desactiva en caso contrario de forma inmediata. La secuencia realiza una acción de ON – delay sobre la SALIDA_0 a través de transiciones con <Local_A_Tarea>.

NEGFILTER_4. Filtro negativo de entrada analógica. Filtra la entrada ARG2 hacia la variable ARG1, activándola cuando la entrada sea menor que el umbral inferior durante los ciclos de retardo ARG4. (El umbral está en unidades de ADC).

If ((ARG2 < ARG3) && (Tiempo > ARG4)) ARG1=1.

ARG1 = Variable resultado.
ARG2 = Variable entrada.
ARG3 = Umbral inferior.
ARG4 = Ciclos de retardo.

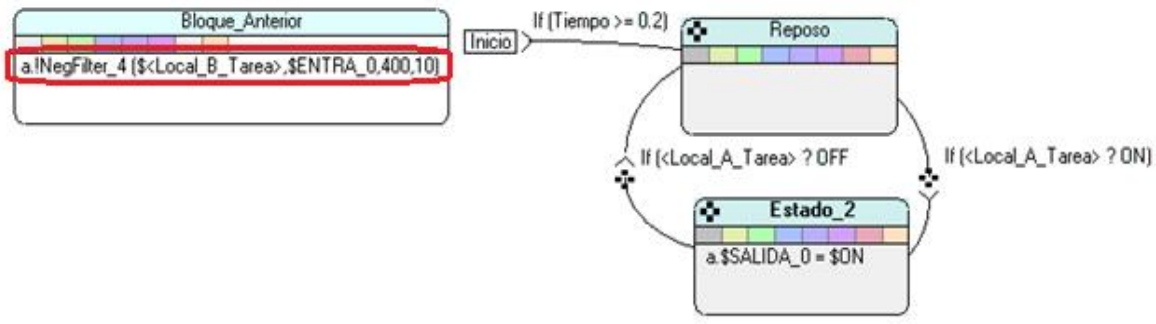


Aplicaciones: Se utiliza para filtrar en amplitud y en frecuencia entradas digitales ruidosas.

Evacuación de materiales en secuencias de control de procesos.

Eliminación de picos negativos en las entradas del autómata.

Temporización OFF Delay (Retardo al reposo)

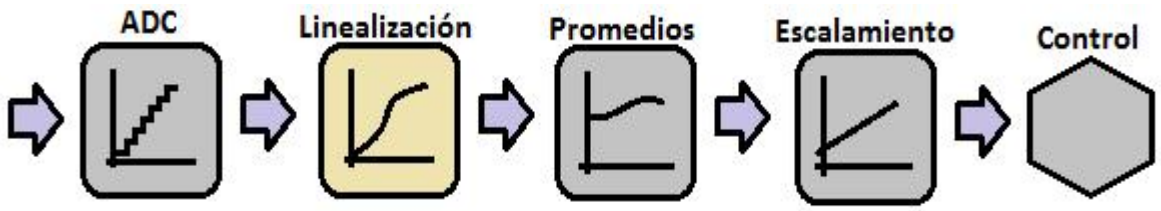


La función NegFilter activa la variable <Local_A_Tarea> cuando el valor de ENTRA_0 es mayor que 400 unidades de ADC y la desactiva cuando ocurre lo contrario durante 200 ms (10/50) ó más. La secuencia realiza una acción de OFF – delay sobre la SALIDA_0 a través de transiciones con <Local_A_Tarea>.

ASIGFILTERLIN_2. Asignar Filtro de linelización a puerto de entrada. Asigna un Filtro de linelización a través de la consulta de la tabla ARG2[Longitud], al puerto de entrada ARG1.

Variable filtrada=Tabla[Entrada / 32].

ARG1 = Puerto de entrada.
ARG2 = Tabla de filtrado.

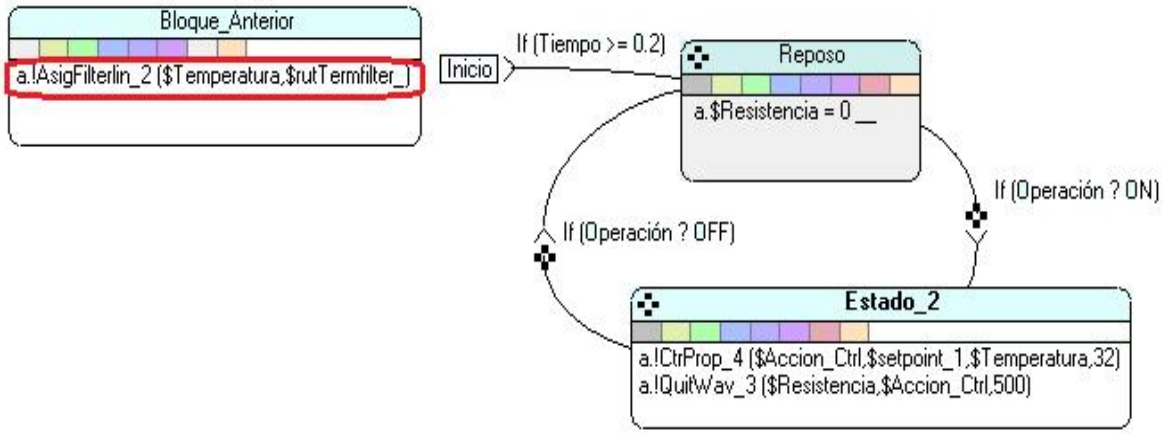


Aplicaciones: Se utiliza para linealizar entradas desde sensores no lineales.

Linealización de las entradas opto aisladas del autómata programable.

Medición y control de temperatura detectada con termistores.

Setpoint de procesos desde potenciómetros logarítmicos.

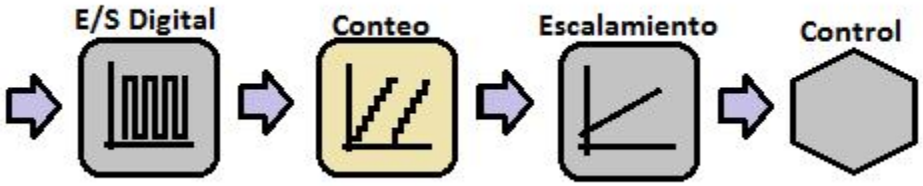


La función Asigfilterlin, la cual puede ser invocada en cualquiera de los estados o bloques de maniobras del programa, es una declaración del tipo de entrada, en virtud de la cual, el RTOS incluye un eslabón de filtrado para dicha entrada. En la invocación de la función se indica que se agrega una tabla de linealización para termistores. La variable de entrada linealizada (Temperatura), se utiliza para implementar un control proporcional con salida hacia una resistencia eléctrica de calentamiento.

ASIGCONT_2. Asignar a una Entrada un Contador de pulsos. Realiza un conteo ascendente en el contador ARG1 incrementándolo cada que hay flancos ascendentes con una frecuencia de hasta 3 kHz, en la entrada ARG2.

If (Edge(ARG2)) ARG1 += 1.

ARG1 = Variable contador.
ARG2 = Entrada de pulsos.

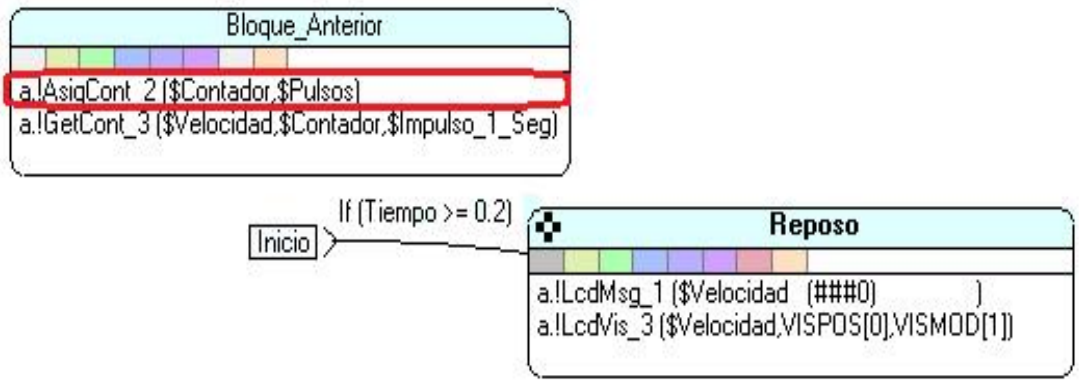


Aplicaciones: Se utiliza para contar eventos en los puertos de entrada.

Control de posición de actuadores con retro alimentación por pulsos.

Medición de velocidades mediante sensores con salida por pulsos.

Medición de flujo con sensores de paleta.



La función AsigCont, la cual puede ser invocada en cualquiera de los estados o bloques de maniobras del programa, es una declaración del tipo de entrada, en virtud de la cual, el RTOS realiza conteos de los pulsos que ingresan por dicha entrada. En la invocación de la función se indica una variable de conteo de pulsos. La variable de conteo (Contador), se utiliza para calcular la cantidad de pulsos por segundo (Velocidad).

ASIGENCOD_3. Asignar Contador Encoder a dos puertos del autómata. Asignar a las entradas "A" y "B" en cuadratura, la función de contador ascendente descendente.

If (ARG2->ARG3) ARG1++, If (ARG2<-ARG3) ARG1- -.

ARG1 = Variable de conteo.
ARG2 = Puerto entrada A.
ARG3 = Puerto entrada B.

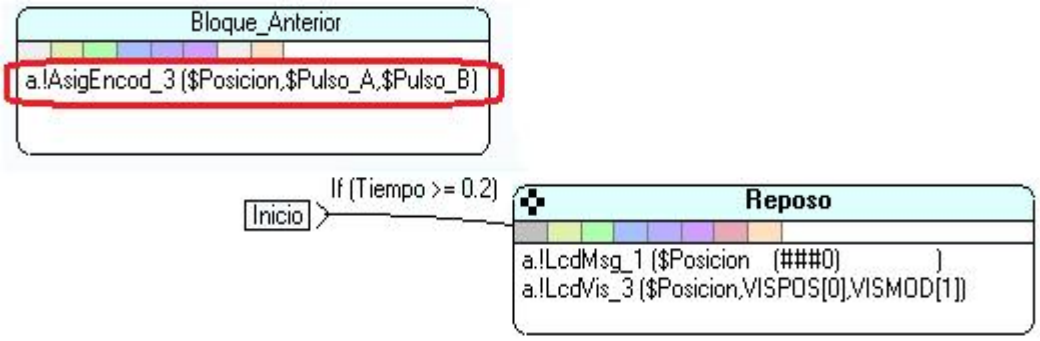


Aplicaciones: Se utiliza para realizar conteos absolutos con Encoder de cuadratura.

Control de posición de actuadores con retro alimentación por encoder.

Interfaces de potencia para sistemas de open CNC.

Ajuste de variables mediante encoder de giro infnito.

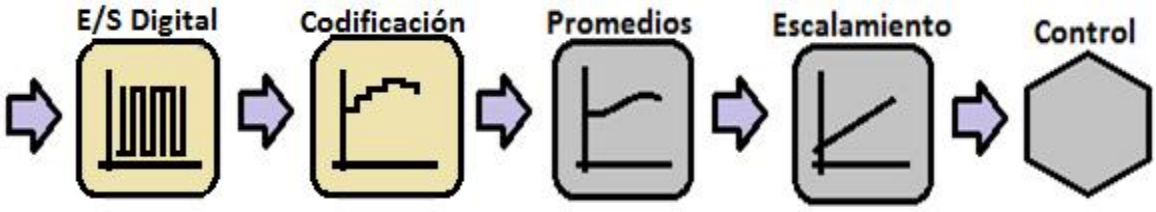


La función AsigEncod, la cual puede ser invocada en cualquiera de los estados o bloques de maniobras del programa, es una declaración del tipo de entrada, en virtud de la cual, el RTOS realiza conteos en cuadratura, de los pulsos que ingresan por dos entradas (Pulso_A, Pulso_B). La variable de conteo de posición (posición), se presenta en el display del Hmi y realiza conteos ascendentes y descendentes, de acuerdo con la sucesión de pulsos por las dos entradas. La variable Posición puede ser utilizada para realizar comparaciones en las transiciones y controles de posición en sistemas mecánicos.

ASIGDEVNET_1. Asignar Dispositivo Serial DevNet a Puerto de entrada. Asigna la funcionalidad de dispositivo serial al puerto de entrada ARG1 con posibilidades de calibración.

Puerto de entrada opera a través de DevNet.

ARG1 = Puerto de entrada.

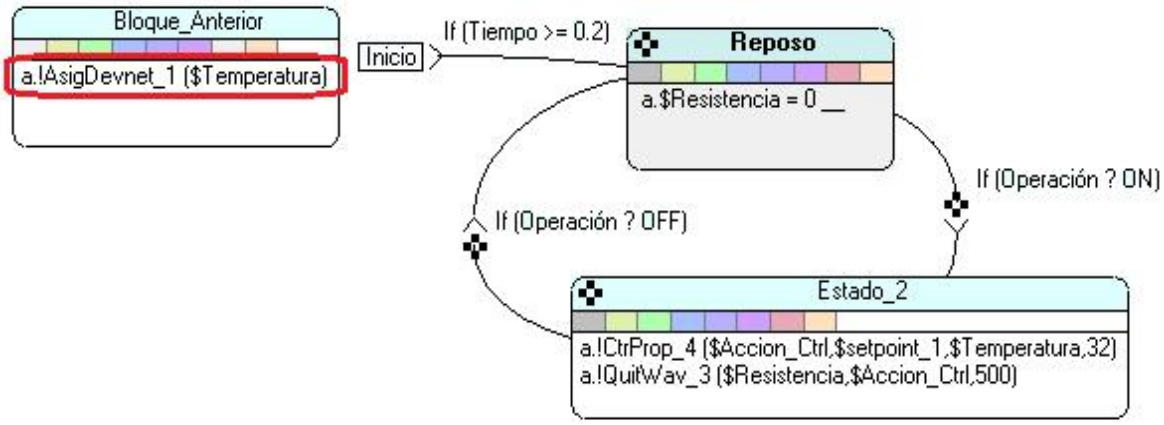


Aplicaciones: Se utiliza con dispositivos tipo DevNet con comunicación tipo Manchester.

Control de Temperatura con sensores infra rojos (Sensores I2C).

Control de procesos de pasaje continuo y por lotes Sensores SPI).

Medición precisa de voltajes y corrientes (Rangos 0 – 12 Vdc, 0-20 mA).



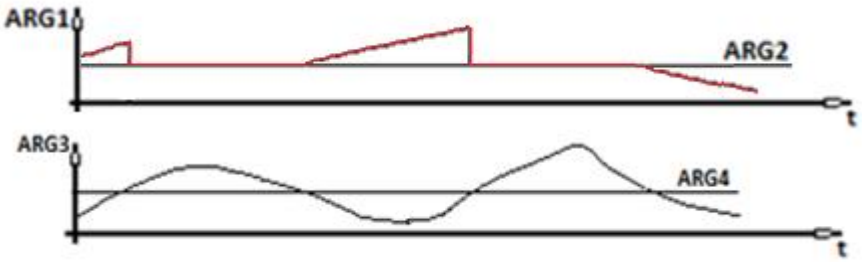
La función AsigDevnet, la cual puede ser invocada en cualquiera de los estados o bloques de maniobras del programa, es una declaración del tipo de entrada, en virtud de la cual, el RTOS recibe datos a través de la entrada indicada, mediante una comunicación con codificación Manchester, desde un dispositivo inteligente. La variable de entrada (Temperatura), se utiliza para implementar un control proporcional con salida hacia una resistencia eléctrica de calentamiento.

MANIOBRAS FUNCIONALES DE CONTEO Y TEMPORIZACIÓN.

SETVAL_4. Asigna el valor en ARG2 a la variable ARG1 cuando la variable ARG3 es mayor o igual que el valor ARG4.

If (ARG3 >= ARG4) ARG1 = ARG2

ARG1 = Variable a modificar.
ARG2 = Valor a asignar.
ARG3 = Variable de entrada.
ARG4 = Valor comparación.



Aplicaciones: Se usa para reiniciar o escribir en una variable al cumplirse una condición.

Escritura en variables tipo “Comandos”.

Reiniciar contadores de eventos IncCont y DecCont.

Cálculos de frecuencia y velocidad.

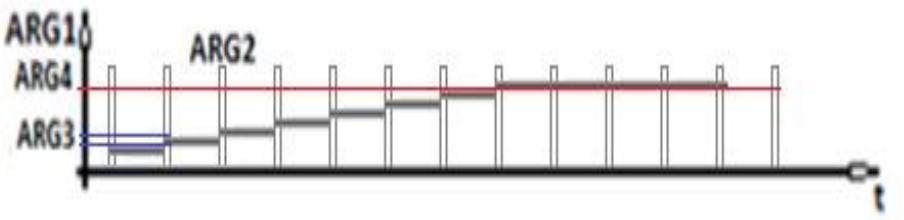


La función SetVal reinicia el contador de tiempo de estados con cero, cuando el valor de la entrada (Nivel) es mayor que 200. La secuencia genera una alarma durante tres segundos cada que la entrada Nivel sea menor o igual a 200 durante un período mayor o igual a dos segundos. La secuencia es útil cuando se necesita revisar permanentemente que una variable no baje de su valor crítico. (Nivel de agua, nivel de aceite, presión mínima, etc.)

INCCONT_4. Contador que incrementa la variable ARG1en el valor ARG3 al haber flanco positivo en la variable ARG2, hasta el valor ARG4. La variable contador puede ser una variable de RAM no volátil, para que al des-energizar y energizar de nuevo el autómata programable, no se pierda el valor actual del conteo.

If ((Edge(ARG2)) && (ARG1<ARG4)) ARG1 += ARG3

ARG1 = Variable contador.
ARG2 = Variable pulsos.
ARG3 = Valor de incremento (Paso).
ARG4 = Máximo valor de conteo.



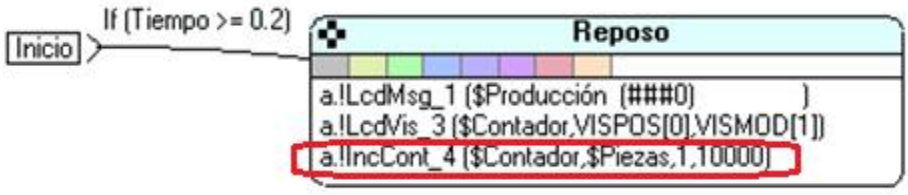
Aplicaciones: Se usa para contar eventos incrementalmente.

Conteos ascendentes de ciclos de producción en memoria no-volátil.

Conteos de tiempo utilizando bases de tiempo.

Medición de frecuencias y velocidades lentas.

Conteos de tiempo sin reinicio durante varios estados.

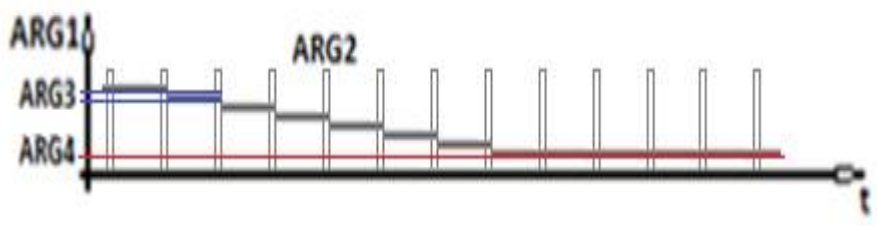


La función IncCont incrementa la variable contador en una unidad, cada que se detecta un flanco ascendente en la variable de entrada Piezas. Cuando el valor de Contador es mayor que 10000, este se conserva en 10000. En el estado se presenta el valor de conteo en el display de la interfaz Hmi. La secuencia es útil para llevar conteos de pulsos de baja frecuencia ingresados por las entradas del autómata programable.

DECCONT_4. Contador que decrementa la variable ARG1 en el valor ARG3 al haber flanco positivo en la variable ARG2, hasta el valor ARG4.

If ((Edge(ARG2)) && (ARG1>ARG4)) ARG1 -= ARG3

ARG1 = Variable contador.
ARG2 = Variable pulsos.
ARG3 = Valor de decremento (Paso).
ARG4 = Mínimo valor de conteo.



Aplicaciones: Se usa para contar eventos decrementalmente.

Conteos descendentes de ciclos de producción en memoria no-volátil.

Conteos de tiempo faltante utilizando bases de tiempo.

Medición de frecuencias y velocidades lentas.

Conteos de tiempo sin reinicio durante varios estados.

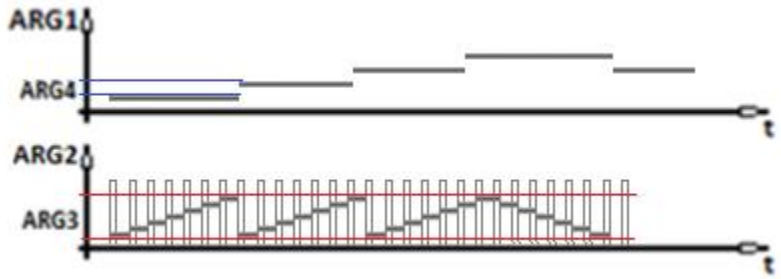


La función DecCont decrementa la variable contador en una unidad, cada que se detecta un flanco ascendente en la variable de entrada Piezas. Cuando el valor de Contador es igual a cero no sigue decrementando. En la secuencia se lleva un conteo decremental de la variable conteo desde 1000 hasta 0, a partir del instante en que llegue un flanco ascendente por la entrada operación. El conteo es presentado en el display del Hmi conectado al Amp1.

CONTPRESC_4. Conteo en variable desde valor acumulado en prescaler, comparando el valor absoluto de la variable ARG2 con ARG3, si es mayor positivo se incrementa la variable ARG1 en el valor ARG4, si es menor negativo se decrementa la variable ARG1 en el valor ARG4.

If (Abs(ARG2) >= ARG3) ARG1 = ARG1 +/- ARG4

ARG1 = Variable de conteo.
ARG2 = Variable prescaler.
ARG3 = Valor de comparación.
ARG4 = Valor de incremento de conteo.

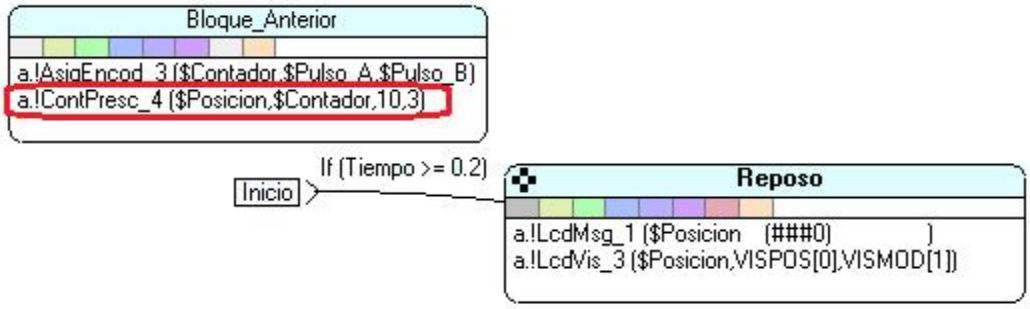


Aplicaciones: Se usa para cambiar la resolución de los conteos.

Control de posición con encoder de alta resolución.

Escalamiento de variables frecuenciales.

Eliminación de conteos por desajuste mecánico y vibración.



La función ContPresc, incrementa o decrementa la variable Posición en 3 unidades cada que la variable Contador incrementa o decrementa en 10 unidades. La variable Posición se presenta en el display del Hmi con una relación 3 / 10 con respecto a la cantidad de pulsos que ingresan por las entradas, lo que muestra que la función no solamente permite dividir la frecuencia, sino escalar la medición con fines de control en unidades de ingeniería.

GETCONT_3. Asignación a variable y reseteo de contador, asignando el valor en ARG2 a la variable ARG1 y borra ARG2, cuando la variable ARG3 es mayor que cero.

If (ARG3 > 0) ARG1 = ARG2; ARG2 = 0

ARG1 = Variable a modificar.
ARG2 = Variable de conteo.
ARG3 = Reloj de acción.



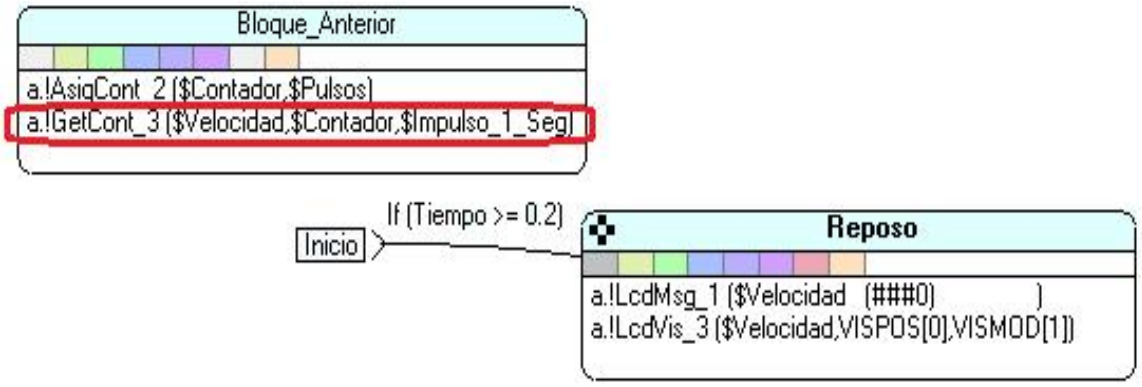
Aplicaciones: Se usa para calcular velocidad de un conteo.

Medición de frecuencia de entradas de conteo.

Control de flujo de líquidos y gases.

Medición de período de entradas de pulsos.

Medición de las r.p.m. de los motores eléctricos.



La función GetCont carga la variable Velocidad con el valor de la variable Contador y borra esta última, cada segundo. En la secuencia se tiene un bloque de maniobras con la asignación de un contador a la entrada y la función GetCont para calcular la frecuencia de los pulsos. La variable Velocidad se presenta en el display del Hmi, conectado al autómata programable y puede ser utilizada en secuencias de control de diferente tipo.

TIMERINC_2. Temporizador incremental de estado. Incrementa la variable ARG1 al haber flanco en ARG2. La variable ARG1 se carga con el valor cero cuando hay inicio de estado. Esta función permite contar tiempo o eventos al interior de un estado.

If (Edge(ARG2)) ARG1 += 1; If (Inicio_estado) ARG1 = 0.

ARG1 = Variable contador.
ARG2 = Base de tiempo conteo.

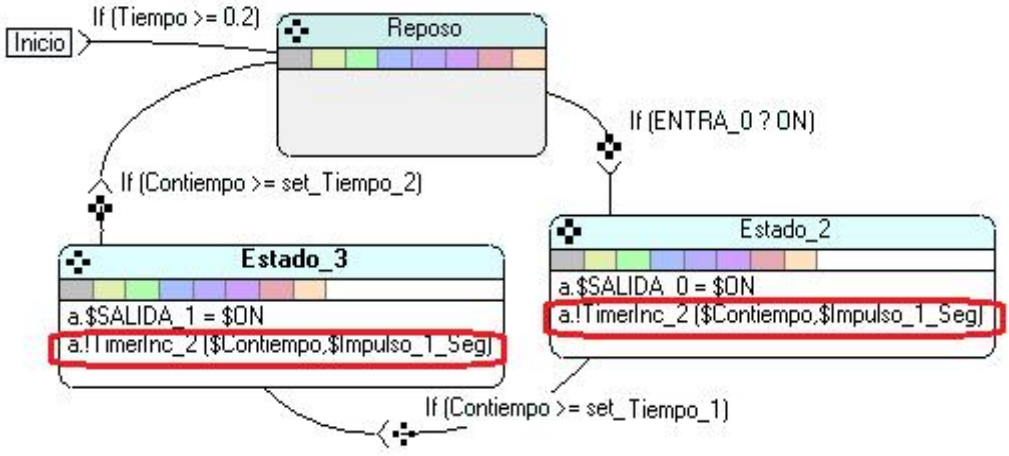


Aplicaciones: Se usa para contar tiempo con bases de tiempo en los estados.

Conteos de tiempo con set-point modificado desde HMI o SCADA.

Contador de eventos al interior de un estado.

Medición de lapsos de tiempo utilizando cambios de estado.



La función TimerInc es invocada en dos estados de la secuencia para contar segundos en la variable Contiempo, la cual se compara respectivamente con dos setpoint de tiempo para que operen las transiciones entre estados. Esto permite la realización de secuencias con temporización programable, desde una interfaz Hmi o desde un sistema remoto.

TIMERDEC_3. Temporizador decremental de estado. Decremental la variable ARG1 al haber flanco en ARG2, desde el valor ARG3 al iniciar el estado, hasta Cero.

If (Edge(ARG2)) ARG1 -= 1; If (Inicio_estado) ARG1=ARG3.

ARG1 = Variable contador.
ARG2 = Base de tiempo conteo
ARG3 = Valor inicial de conteo

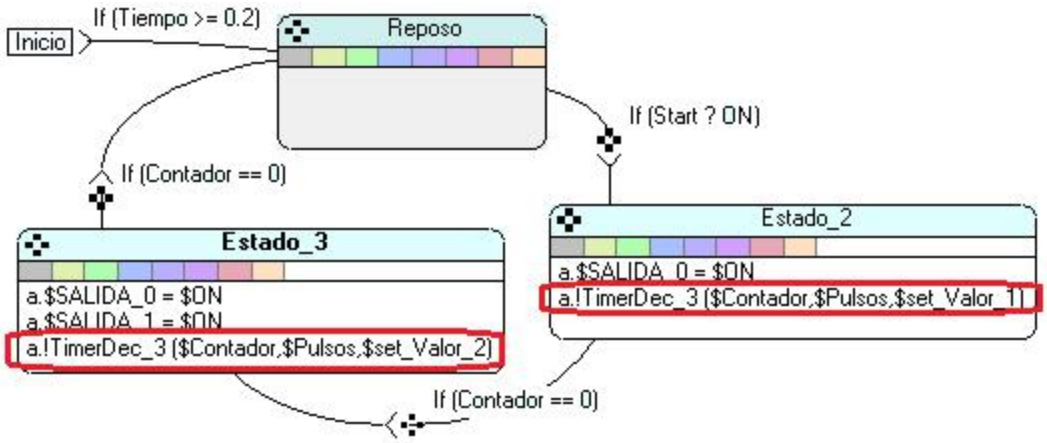


Aplicaciones: Contar tiempo con bases de tiempo en los estados.

Conteos de tiempo con set-point modificado desde HMI o SCADA.

Contador decremental de eventos al interior de un estado.

Presentación de tiempo faltante para salir de un estado.

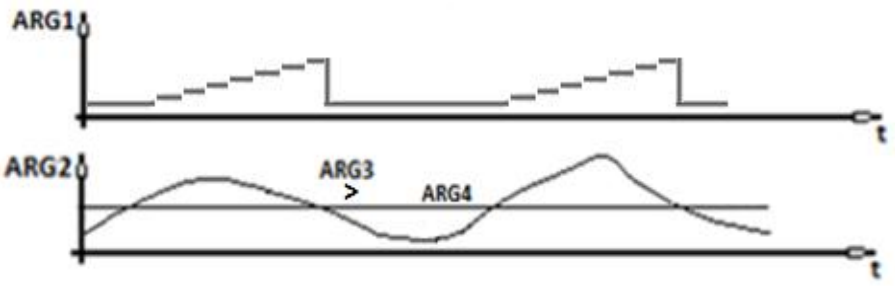


La función TimerDec es invocada en dos estados de la secuencia para descontar en la variable Contador, cantidades de piezas programadas en dos variables de setpoint. Al pulsar Start, se activa la SALIDA_0, tras lo cual se carga la variable Contador con Set_Valor_1 y se descuenta hasta cero, para pasar al siguiente estado, en el cual se activa adicionalmente la SALIDA_1 y se realiza un descuento desde Set_Valor_2 hasta cero, para regresar al estado de reposo, en el cual están desactivadas las salidas.

TIMERCMP_4. Temporizador incremental con comparación. Conteo incremetal de ARG1 al cumplirse que ARG2 se compara con ARG4 mediante ARG3, en caso contrario y al haber inicio de estado, ARG1 se carga con el valor.

If (ARG2 ¿ARG3? ARG4) ARG1++; else Ini_estado; ARG1=0.

ARG1 = Variable contador.
ARG2 = Valor a comparar.
ARG3 = Tipo de comparación.
ARG4 = Valor comparación.

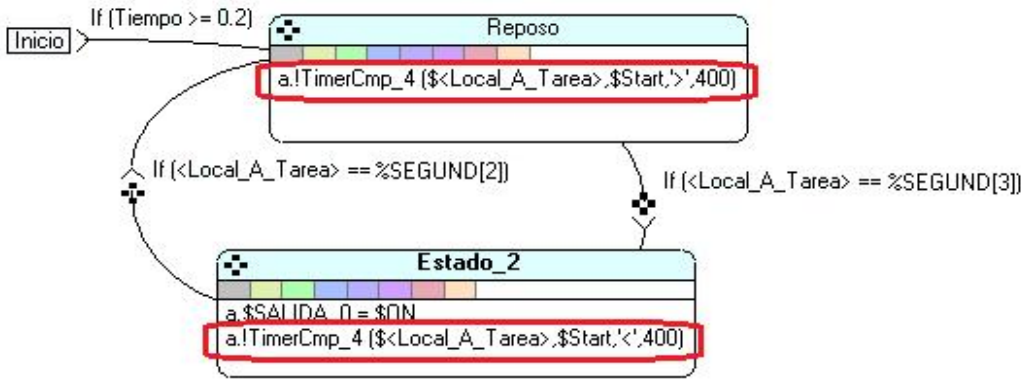


Aplicaciones: Se usa para contar el tiempo que lleva una variable activada.

Detección de variables ruidosas para realizar cambios de estado.

Retardo al trabajo / reposo de una variable de entrada.

Control on-off con tiempo muerto, de una variable continua.

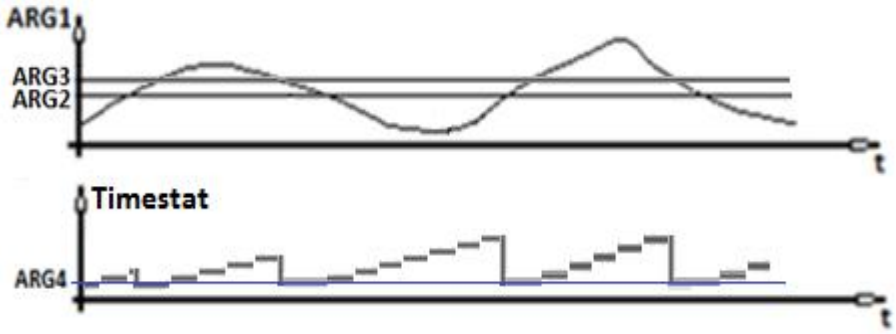


La función TimerCmp es invocada en el estado de Reposo para incrementar 50 veces por segundo la variable Local_A_Tarea, mientras la entrada Start sea > 400 y cambiar de estado luego de 3 segundos en esta condición (On-Delay). Se realiza un conteo similar cuando la entrada es < 400 y luego de 2 segundos, regresar a Reposo (Off-Delay).

TIMSTATINIC_4. Reiniciar el Contador de Tiempo de Estado con ARG4 en el momento en que la Variable ARG1 esté en el rango entre el valor ARG2 y el valor ARG3.

If(ARG1 == [ARG2 ... ARG3]) ContTiempo = ARG4.

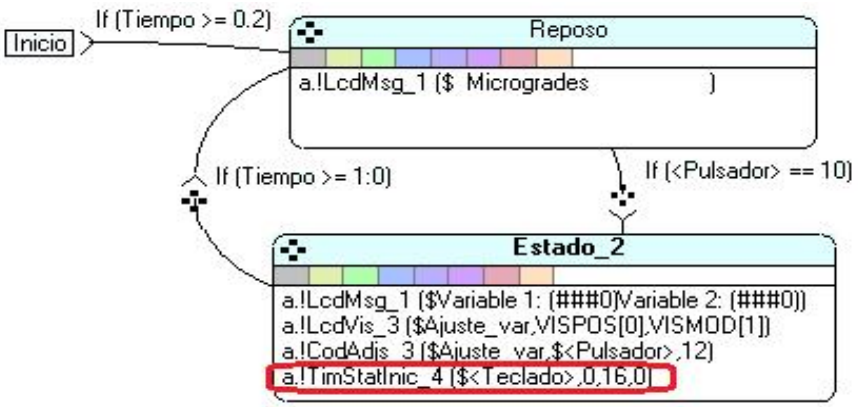
ARG1 = Variable a revisar.
ARG2 = Valor mínimo del rango.
ARG3 = Valor máximo del rango.
ARG4 = Valor de Reinicio.



Aplicaciones: Se usa para alargar el tiempo de los estados de edición tipo “xxxAdjs”.

Estadía en un estado, mientras esté activa una variable durante un tiempo.

Estadía en un estado, mientras estén llegando pulsos por una entrada.



La función TimStatInic se utiliza en el Estado_2 para reiniciar el tiempo de estado cada que pulsen alguna tecla (códigos entre 0 y 16). La secuencia pasa del estado de Reposo al Estado_2 cuando se pulsa la tecla 10 (A). En el Estado_2 se presenta y edita la Variable Ajuste_Var y al pasar un minuto sin pulsar ninguna tecla se retorna al estado de Reposo, en el cual se presenta un mensaje fijo.

MANIOBRAS FUNCIONALES DE CONTROL.

VARLIMIT_3. Limitador de variable en un rango de valores. Compara la variable ARG1, si es menor que el valor ARG2 se hace igual a ARG2, si es mayor que ARG3 se hace igual a ARG3.

ARG1 = [ARG2 ... ARG3].

ARG1 = Variable a limitar.
ARG2 = Mínimo valor de la variable.
ARG3 = Máximo valor de la variable.

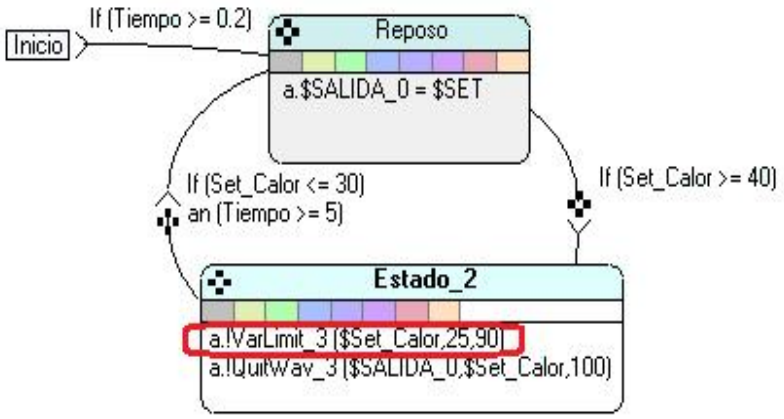


Aplicaciones: Se usa para limitar el valor de una variable en un rango de valores.

Limitación de los valores de las acciones de control PID.

Enmarcado de una variable en el rango de trabajo eliminando valores espurios.

Eliminación de zonas no lineales en sistemas de control.

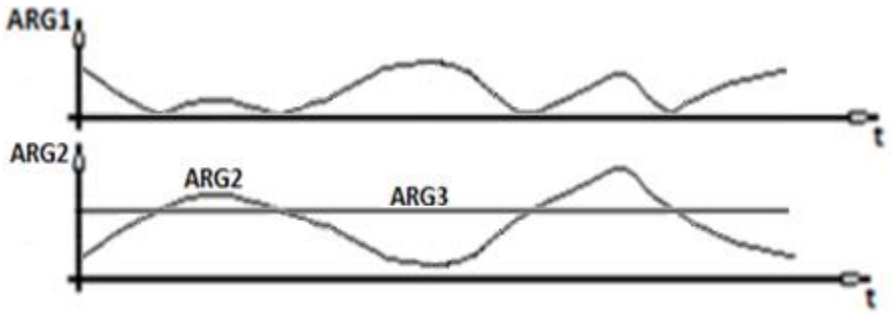


Cuando el valor en la entrada SET_CALOR es menor que 30% durante 5 segundos, la salida se activa permanentemente y cuando el valor en la entrada SET_CALOR es mayor que 40%, se envía un tren de pulsos a través de la salida, el cual tiene anchos entre el 25% y el 90% (rango limitado por la función Varlimit).

ABSDIF_3. Valor absoluto de la diferencia entre dos valores. Carga en la variable ARG1 el valor absoluto de la diferencia entre las variables ARG2 y ARG3.

$$ARG1 = Abs(ARG2 - ARG3)$$

ARG1 = Variable de salida.
ARG2 = Variable minuendo.
ARG3 = Variable sustraendo.

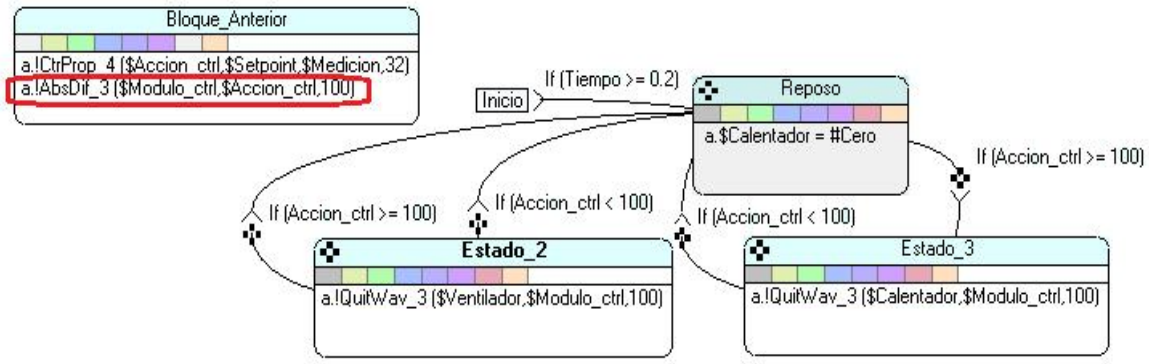


Aplicaciones: Se usa para determinar si dos variables son similares.

Transiciones por similitud de valor en sistemas altamente ruidosos.

Tratamiento de variables de corriente alterna.

Cálculo de la salida hacia sistemas de control con actuadores suplementarios.

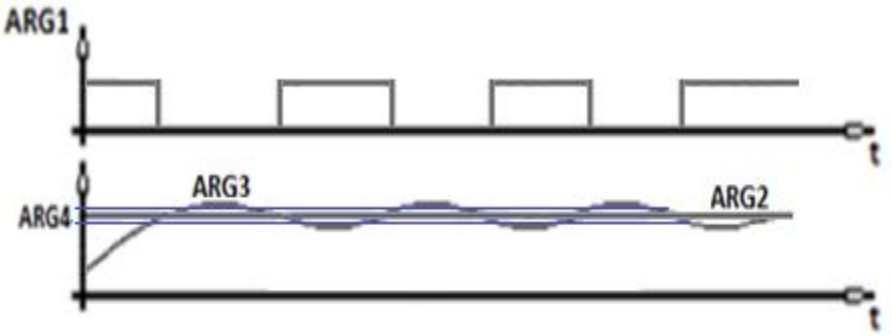


El control proporcional entrega la salida ACCION_CTRL, la cual varía entre 0 y 200. La función AbsDif entrega, en la variable MODULO_CTRL, el valor absoluto de la variable ACCION_CTRL. La secuencia de control, envía un tren de ondas, con ancho de pulso MODULO_CTRL, a la salida VENTILADOR, cuando la ACCION_CTRL es menor que 100 y a la salida CALENTADOR, cuando la ACCION_CTRL es mayor a igual a 100, haciendo que el control opere en cuatro cuadrantes.

CTRONOFF_4. Control ON-OFF. ARG1se activa si ARG3 es menor que ARG2 - ARG4/2. ARG1 se desactiva si ARG3 es mayor que ARG2 + ARG4/2.

if(ARG3<(ARG2-ARG4)) ARG1=1

ARG1 = Variable de salida.
ARG2 = Variable Set point.
ARG3 = Entrada medición.
ARG4 = Banda muerta.

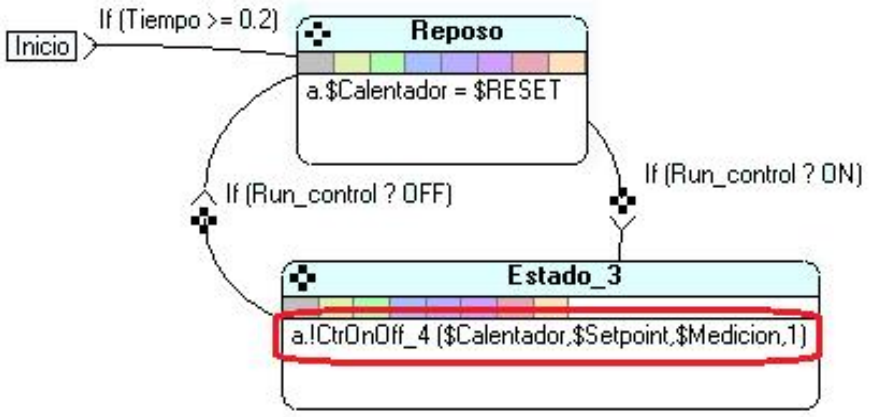


Aplicaciones: Se utiliza para controlar variables continuas mediante actuadores digitales.

Control On-Off de temperatura con entrada continua y salida digital.

Control de nivel de líquidos con sensor continuo y válvulas On-Off.

Control de presión con sensor de 4-20 mA y salida tipo On –Off.

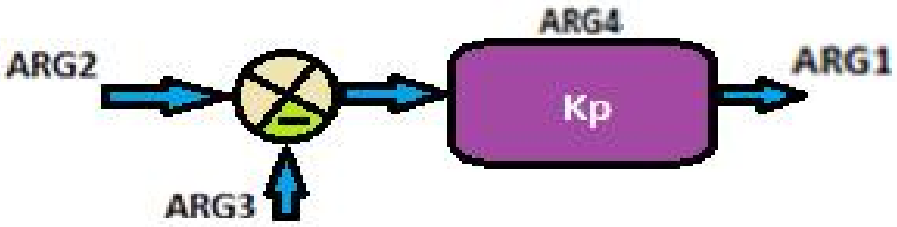


La función CtrlOnOff activa la salida CALENTADOR cuando la MEDICIÓN es menor que SETPOINT – 1 y desactiva la salida CALENTADOR cuando la MEDICIÓN es mayor que SETPOINT + 1. Lo anterior mientras esté activada la entrada RUN_CONTROL, en caso contrario, la salida CALENTADOR se desactiva.

CTRPROP_4. Control proporcional. ARG1 recibe el producto de la ganancia ARG4/32 multiplicada por el Error (ARG2 - ARG3) < 4000.

ARG1=(ARG4/32) * (ARG2 - ARG3).

ARG1 = Variable de salida.
ARG2 = Variable Set point.
ARG3 = Entrada medición.
ARG4 = Valor Ganancia * 32.

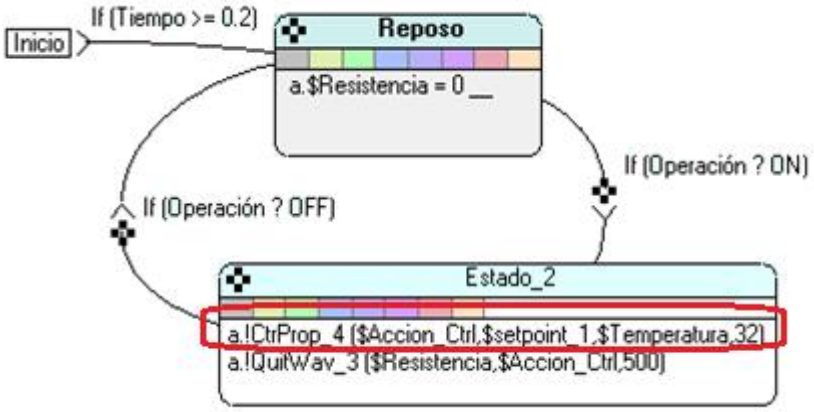


Aplicaciones: Se utiliza para controlar variables continuas mediante salidas continuas.

Aplica a salidas QuitPwm, QuitWav, QuitPhase, QuitServo, QuitStep, QuitBldc.

Control proporcional de temperatura con entrada y salida continuas.

Control proporcional de nivel con sensor 4-20 mA y salida tipo “QuitXxx”

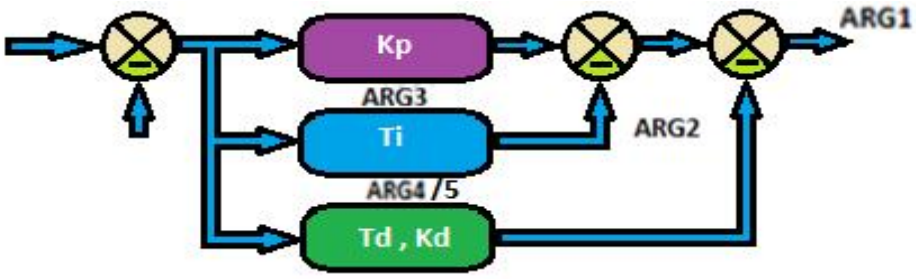


La función CtrlProp entrega, en la variable ACCION_CTRL, el resultado de un control proporcional con setpoint en la variable SETPOINT, medición en TEMPERATURA y ganancia igual a 32 / 32. La ACCION_CTRL se envía a la salida RESISTENCIA, como el ancho de pulso de un tren de ondas con período de 500 / 1250 de segundo. El control proporcional debe estar sintonizado para un rango de salida entre 0 y 500.

CTRINTDER_5. Complemento I.D. a CtrProp, para realizar Control PID. ARG1 recibe la suma de las acciones proporcional, integral y derivativa, calculadas a partir del Error en CtrProp. Utiliza el vector ARG2[0] para cálculos internos.

ARG1=PID(Proporcional, ARG3/256, ARG4/16 : ARG5).

ARG1 = Variable de salida.
ARG2 = Vector[0] de control.
ARG3 = Constante Integral * 256.
ARG4 = Constante derivativa * 16.
ARG5 = Ciclos tiempo derivativo.



Aplicaciones: Se utiliza luego de CtrProp para hacer control PID de variable continua.

Aplica a salidas QuitPwm, QuitWav, QuitPhase, QuitServo, QuitStep, QuitBldc.

Control PID de temperatura con sensor continuo y salida tipo “QuitXxx.

Control PID de nivel con sensor 4-20 mA y salida tipo “QuitXxx.



Las funciones CtrlProp y CtrlIntDer entregan el resultado de una función PID con ganancia proporcional de 32 / 32, constante integral de 128 / 256 (cada ciclo de la tarea) y constante derivativa de 16 / 16 (cada 5 ciclos de la tarea), en la variable ACCIÓN_CTRL, la cual establece la posición de una servo-válvula con rango de operación entre 0 y 1000, conectada en la salida SERVOMOTOR.

SETSAVE_3. Guardar variables Ram en Flash tipo “Ajsute”, la cual se conecta a las tareas como entrada. Guardar la cantidad ARG3 de Variables ARG2 en Ajuste ARG1 al iniciar el estado que contiene la maniobra.

If(Inicio_Estado) ARG1[ARG3] = ARG2[ARG3].

ARG1 = Ajuste de destino.
ARG2 = Variables a guardar.
ARG3 = Cantidad variables grabar.

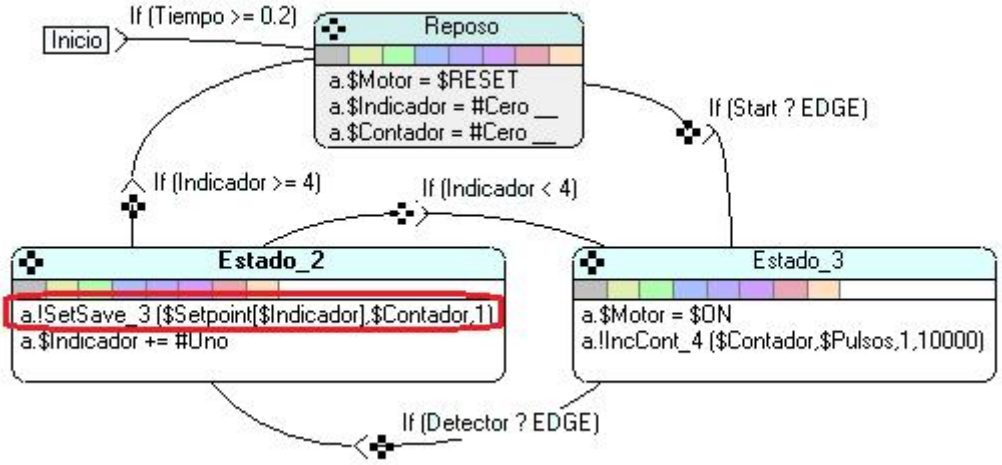


Aplicaciones: Se usa para guardar datos en la memoria Flash tipo “AJS”

Ajustar variables no-volátiles en tiempo de ejecución de las secuencias.

Aplicaciones con aprendizaje de setpoints de posición.

Aplicaciones de control con ciclos de auto sintonización (auto-tuning).



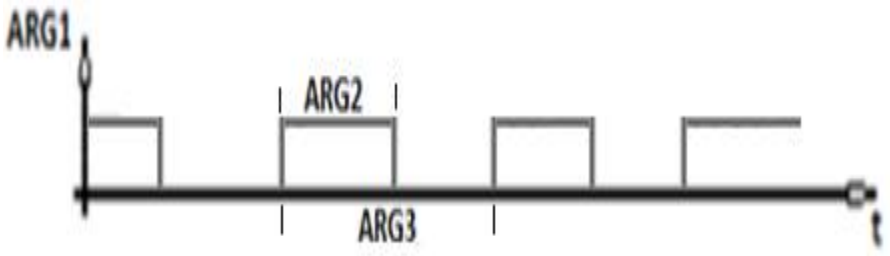
La secuencia realiza un aprendizaje de posiciones en un sistema de ubicaciones pre-programadas. Al activarse la entrada START se energiza el MOTOR de avance, tras lo cual empiezan a contarse PULSOS con la función IncCont, en la variable CONTADOR. Cada que se activa la entrada DETECTOR, Se guarda la ubicación correspondiente en el vector de Flash SETPOINT, en la posición apuntada por INDICADOR (hasta 4 posiciones).

MANIOBRAS FUNCIONALES SOBRE LAS SALIDAS.

GENPULSE_3. Generación de tren de pulsos de internos. ARG1 = Conteo_Interno cuando éste es menor que ARG2, ARG1 = 0 en caso contrario. El conteo_interno se incrementa hasta ARG3 y luego se reinicia en 0.

If (Cont++ >= ARG3) Cont = 0; If (Cont < ARG2) ARG1 = Cont.

ARG1 = Variable de salida.
ARG2 = Ancho de pulso.
ARG3 = Período de pulsos.

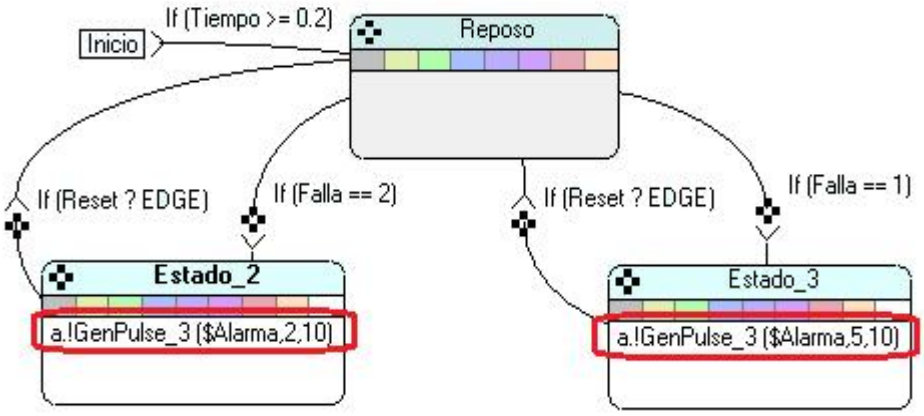


Aplicaciones: Se usa para generar pulsos internos configurables.

Indicadores de alarmas con modulación particular y/o progresiva.

Creación de modulaciones para salidas tipo “QuitXxx”.

Creación de bases de tiempo ajustadas desde entradas continuas.

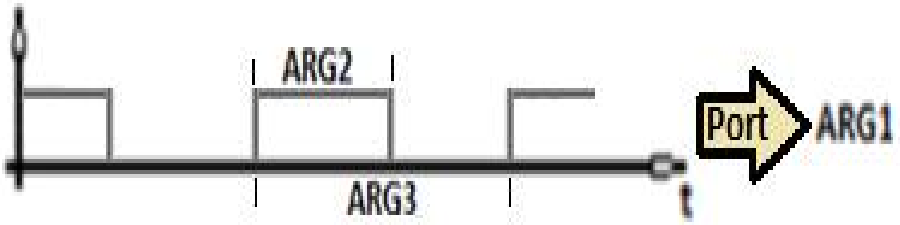


La secuencia utiliza la función GenPulse para enviar a la salida ALARMA, un tren de pulsos con período de 10 ciclos de la tarea y ancho de pulso de 5 ciclos de la tarea cuando ocurre una falla con código 1. El ancho de pulso es de 2 ciclos de la tarea cuando ocurre una falla con código 2. La función puede generar los pulsos hacia variables internas.

QUITWAV_3. Generador de Tren de Ondas hacia una salida. Genera un tren de Ondas hacia la salida ARG1 con ancho de pulso ARG2 y período ARG3/1250. La frecuencia base es de 1250 Hz. El ancho de pulso se varía entre 0 y ARG3. Esta función sólo puede ser utilizada con las salidas del autómata programable, como primer argumento.

Promedio salida=ancho de pulso / período de onda.

ARG1 = Puerto de salida.
ARG2 = Ancho de pulso.
ARG3 = Valor Período * 1250.

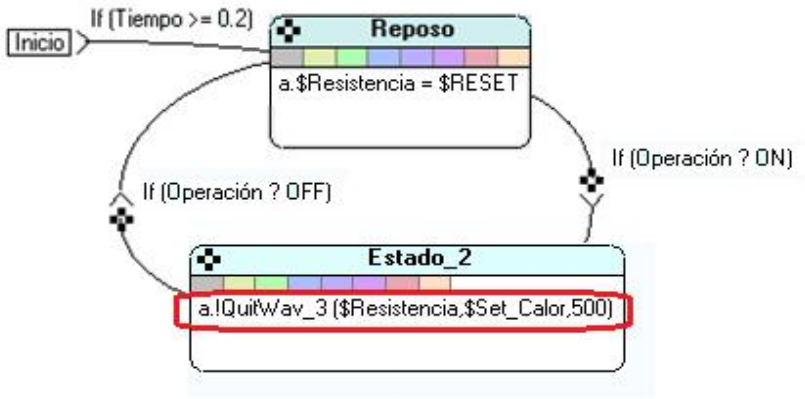


Aplicaciones: Se utiliza para enviar variables continuas a través de actuadores digitales.

Control proporcional / PID de temperatura con salida hacia resistencias.

Control en lazo abierto de variables altamente inerciales.

Indicación de diferentes tipos de alarmas con modulación variable.

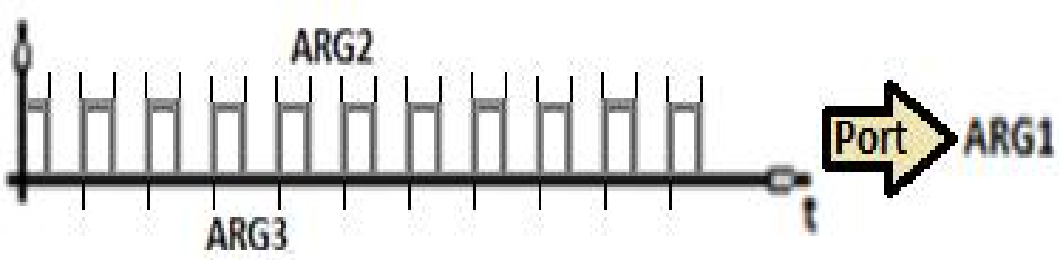


La función QuitWav envía a la salida RESISTENCIA, un tren de ondas (pulsos) con período de 500 / 1250 de segundo y un ancho de pulso establecido en la entrada SET_CALOR (en 1 / 1250 de segundo). La función de salida del tren de ondas opera mientras la entrada OPERACIÓN está activada, en caso contrario, la salida RESISTENCIA está desactivada. La función envía la orden al RTOS para que envíe los pulsos a la salida.

QUITPWM_3. Genera PWM hacia un puerto modulado. Genera PWM en el puerto ARG1 con ancho de pulso ARG2 y período ARG3/(1.25 MHz). El ancho de pulso se varía entre 0 y ARG3. Si el puerto no es de tipo Pwm, la salida opera en modo Tren de Ondas con frecuencia base de 1250 Hz.

Promedio salida=ancho de pulso / período.

ARG1 = Puerto de salida.
ARG2 = Ancho de pulso.
ARG3 = Divisor desde 1.25 MHz.

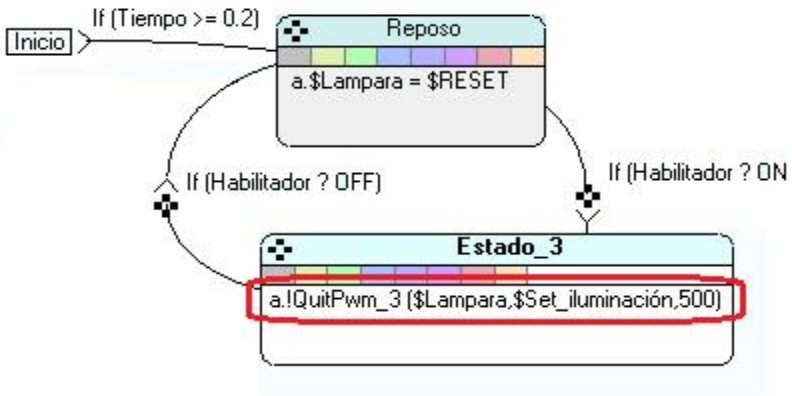


Aplicaciones: Se utiliza para actuar sobre variables continuas.

Enviar consignas de velocidad hacia variadores de velocidad de motores.

Control de iluminación con luces LED conectadas directamente al autómata.

Control de velocidad de motores de corriente directa utilizando DC-SSR.

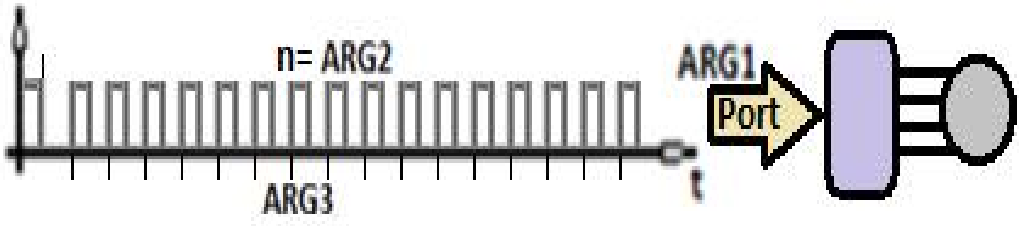


La secuencia realiza una dimerización de alta frecuencia sobre una lámpara de corriente directa. La función QuitPwm genera un tren de pulsos de alta frecuencia, utilizando los módulos de pwm del hardware del autómata programable. El período de los pulsos es de 500 / 2500000 de segundo y el ancho de pulso es igual al valor de la variable SET_ILUMINACIÓN / 2500000 de segundo.

QUITSTEP_3. Generación de pulsos hacia motor de paso. Genera Pulsos para Motor de Paso a través del puerto ARG1 a una frecuencia de 6250/ARG3. La función se debe usar con modo "i" o "f". Si se utiliza en modo "a" el movimiento del motor es continuo.

Frecuencia de pulsos=6250 Hz/Divisor.

ARG1 = Puerto de salida.
ARG2 = Cantidad de pulsos.
ARG3 = Divisor desde 6250 Hz.

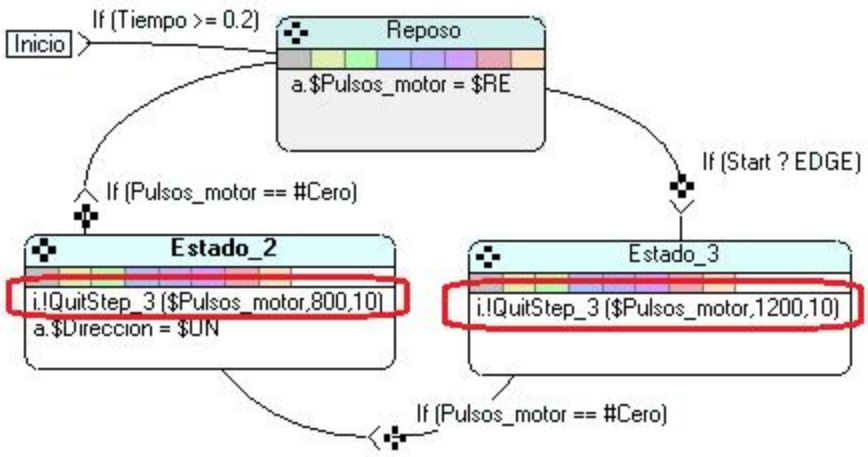


Aplicaciones: Se utiliza con driver que reciba pulsos y dirección (TB6600, A4988).

Control de posición de motores de paso a paso, mediante recorridos programados.

Ajuste de velocidad de motores de paso a paso (de aproximación y de trabajo).

Interfaz de control numérico con ejecución de comandos básicos G01 y G02.

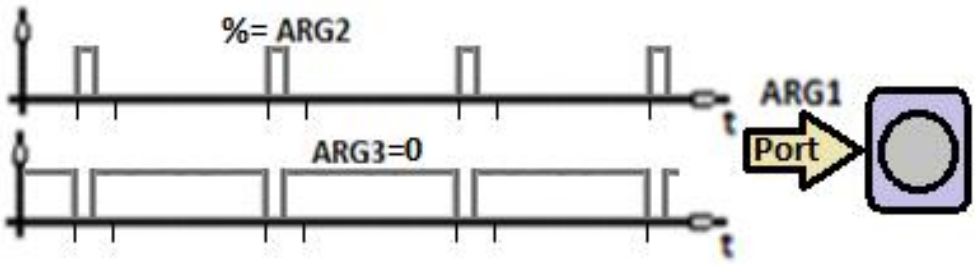


La secuencia realiza un recorrido de ida y vuelta utilizando un motor de paso a paso controlado a través de un driver conectado a dos salidas del autómata programable PULSOS_MOTOR y DIRECCIÓN. Al activarse la entrada START, el motor avanza un recorrido de 1200 pulsos con una frecuencia de 1250 / 10 Hz. Luego regresa un recorrido de 800 pulsos con la misma frecuencia.

QUITSERVO_3. Envío de órdenes hacia servomotor con pulsos de 1-2 ms. Genera pulsos para Servomotor a través del puerto ARG1 con lógica en ARG3, anchos de pulso entre 1 y 2 ms y un período de 20 ms. 1ms = ángulo cero y 2ms = ángulo máximo.

Décimas de porcentaje de posición 0=1 ms, 1000=2 ms.

ARG1 = Puerto de salida.
ARG2 = Décima de porcentaje posición.
ARG3 = Polaridad de la salida.

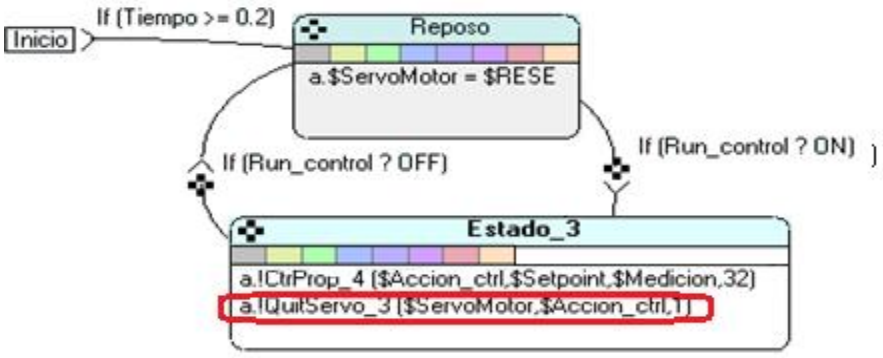


Aplicaciones: Se utiliza con servomotores de posición por ancho de pulso.

Control continuo de apertura y cierre de válvulas motorizadas.

Control continuo de apertura y cierre de compuertas de sistemas de ventilación.

Control de posición de compuertas de salida en tolvas de almacenamiento.

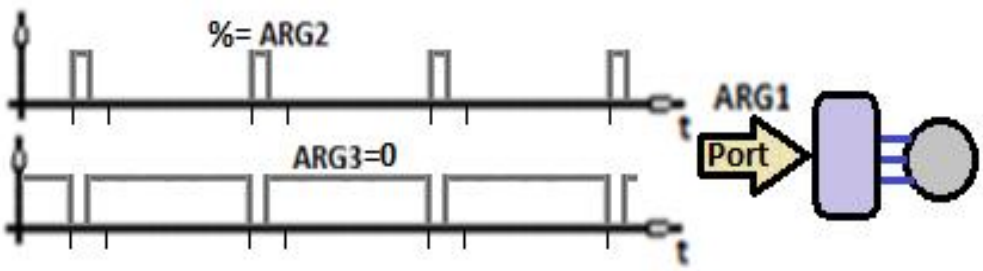


El control de flujo con un actuador tipo moto-válvula, opera mientras la entrada RUN_CONTROL está activada. El resultado de la función de control proporcional ACCIÓN_CTRL sirve de indicador de apertura de la válvula en un rango entre 0 y 1000 (válvula cerrada – válvula abierta). La función QuitServo envía los parámetros al RTOS del Amp1 para que controle la posición angular del servomotor.

QUITBLDC_3. Envío de órdenes hacia motor BLDC con pulsos de 1-2 ms. Genera pulsos para motor BLDC a través del puerto ARG1 con lógica en ARG3, anchos de pulso entre 1 y 2 ms y un período de 20 ms. (1ms = velocidad cero y 2ms = velocidad máxima)

Décimas de porcentaje de velocidad 0=1 ms, 1000=2 ms.

ARG1 = Puerto de salida.
ARG2 = Décimas de porcentaje velocidad.
ARG3 = Polaridad de la salida.

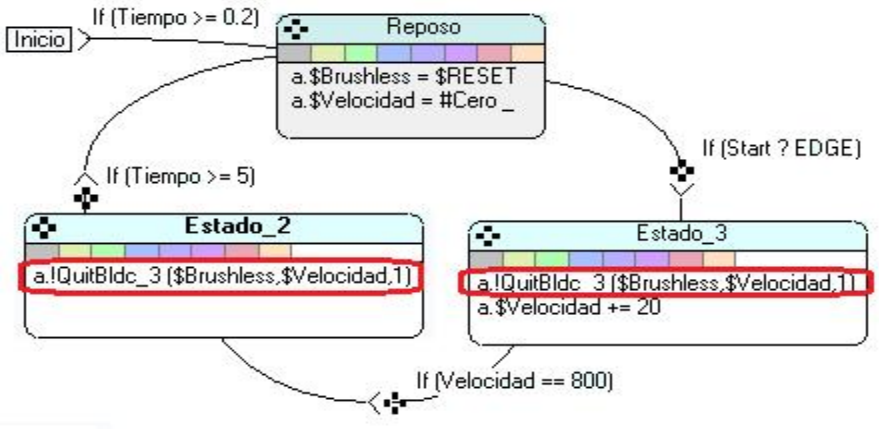


Aplicaciones: Se utiliza con motores brushless con driver por ancho de pulso.

Control de aplicación de calor y frío mediante ventiladores.

Control de velocidad y sentido de giro del husillo de máquinas herramientas.

Sistemas de secado de materiales con curvas de velocidad de inyección de aire.

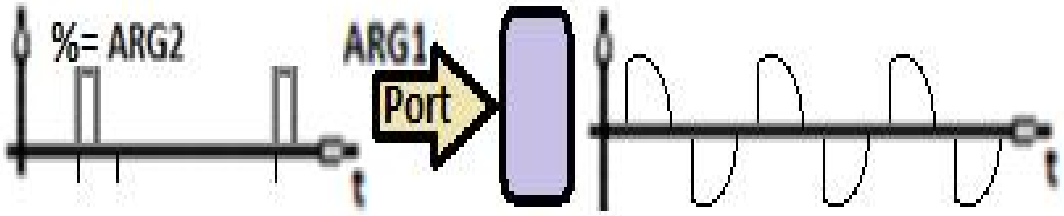


La secuencia realiza el arranque en rampa, de un motor brushless mediante la utilización de la función QuitBldc. Al activarse la entrada START, se envían pulsos al driver del motor brushless a través de la salida BRUSHLESS, con un ancho de pulso, el cual crece en 2.0% cada ciclo de la tarea, hasta llegar al 80.0% de la velocidad, en la cual se conserva durante otros 5 segundos antes de parar el motor brushless.

QUITPHASE_2. Envío de modulación hacia relevos dimerizables. Genera pulsos para relevo dimerizable a través del puerto ARG1 con décimas de porcentaje de apertura en ARG2.

Ángulo de apertura = 0 deg para 100.0% y 170 deg para 0.1%.

ARG1 = Puerto de salida.
ARG2 = Décimas de Porcentaje de apertura.

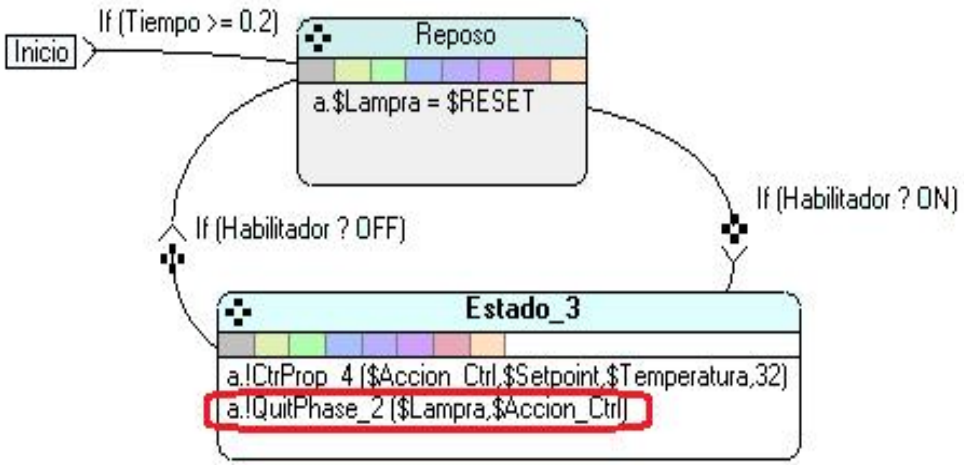


Aplicaciones: Se utiliza con relevos D-SSR para realizar control por ángulo de fase.

Dimerización de voltajes alternos mediante el control por ángulo de fase.

Control de velocidad de motores universales por ángulo de fase.

Control de calentamiento mediante el uso de lámparas de baja inercia térmica.



La secuencia realiza un control de temperatura que utiliza como actuador una lámpara de rápido calentamiento. Mientras esté activada la entrada HABILITADOR se realiza un control proporcional de la variable TEMPERATURA con respecto a un SETPOINT con ganancia de 32 / 32 y salida hacia la variable ACCIÓN_CTRL, la cual corresponde a un valor entre el 0.0% y el 100.0% de la potencia que entrega el relevo dimerizable conectado en la salida LÁMPARA del autómata programable.

QUITDEVNET_2. Envío de valores continuos hacia dispositivos de salida mediante codificación Manchester. Genera pulsos con codificación Manchester de 12 bits con checksum de 4 bits hacia autómatas programables y manejadores de actuadores especiales. No disponible para Amp1-E ni Amp1-H.

Valor enviado = 0 para 0% y 1000 para 100%.

ARG1 = Puerto de salida.
ARG2 = Valor a enviar.



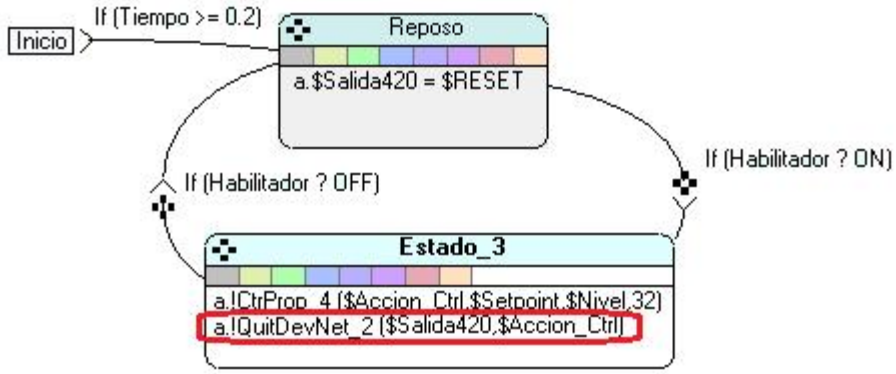
Aplicaciones: Se utiliza con dispositivos externos para enviar variables por las salidas.

Envío de consigna a dispositivos con actuadores tipo “Inverter”.

Salida de señales 4 – 20 mA desde el autómata programable.

Salidas de impedancia variable (potenciómetro digital).

.



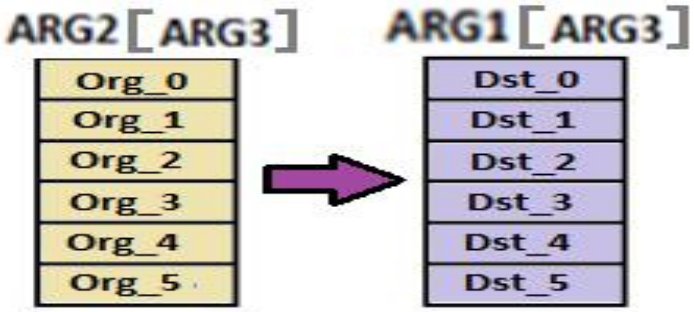
La secuencia realiza un control de nivel en un tanque, utilizando como actuador una válvula proporcional controlada a través de una señal de 4 – 20 miliamperios. Mientras esté activada la entrada HABILITADOR se realiza un control proporcional de la variable NIVEL con respecto a un SETPOINT con ganancia de 32 / 32 y salida hacia la variable ACCIÓN_CTRL, la cual corresponde a un valor entre el 0.0% y el 100.0% de la corriente de salida del dispositivo inteligente conectado a la salida del autómata programable

MANIOBRAS FUNCIONALES CON ARREGLOS DE OPERANDOS.

BLOKMOV_3. Copiar bloque de variables de memoria con dirección ARG2 hacia bloque de variables con dirección ARG1 con cantidad de datos ARG3.

ARG1[ARG3] = ARG2[ARG3].

ARG1 = Dirección de destino.
ARG2 = Dirección de origen.
ARG3 = Cantidad de variables.

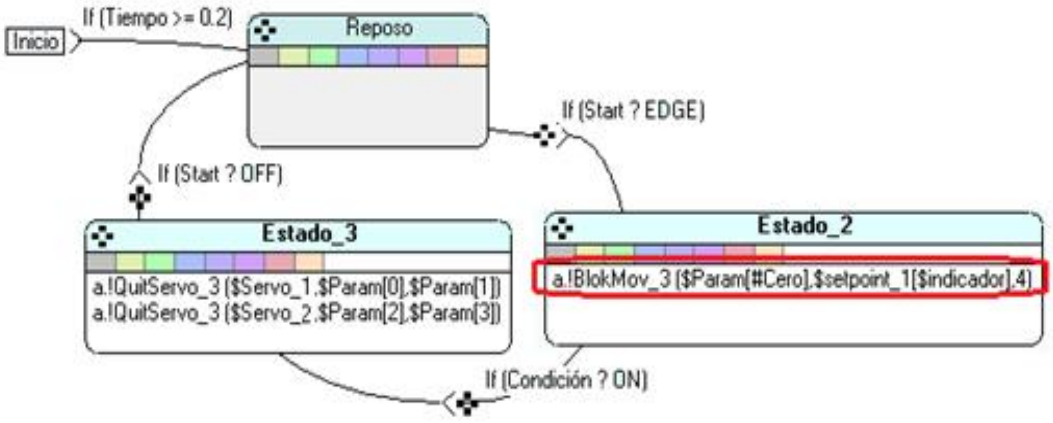


Aplicaciones: Se debe usar al inicio o al final de los estados para mover bloques de datos.

Carga de recetas en programadores de curvas de variables.

Transmisión de mensajes de contenido fijo a través de los buses de comunicación.

Carga de arreglos de datos para el ajuste mediante funciones de HMI.

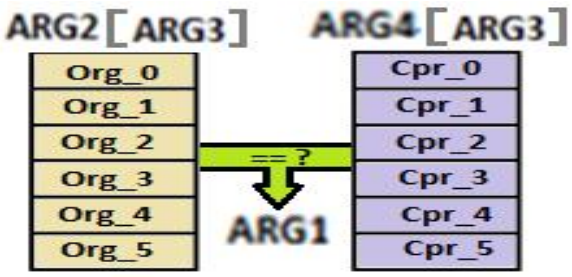


Cada que se activa la entrada START se cargan los parámetros de salida (porcentaje de apertura y lógica de salida).a dos servomotores (SERVO_1 y SERVO_2). La función BlokMov carga cuatro parámetros apuntados por la variable de entrada INDICADOR, desde el vector SETPOINT hacia el vector PARAM.

BLOKCMPR_4. Comparación de dos bloques de datos. ARG1 se activa cuando el contenido del bloque de datos ARG2[ ] con longitud ARG3 es igual al bloque de datos ARG4[ ], ARG1 se desactiva en caso contrario.

ARG1=(ARG2[ARG3] == ARG4[ARG3])?

ARG1 = Variable de salida.
ARG2 = Vector[0] de datos.
ARG3 = Longitud de datos.
ARG4 = Vector[0] compara.

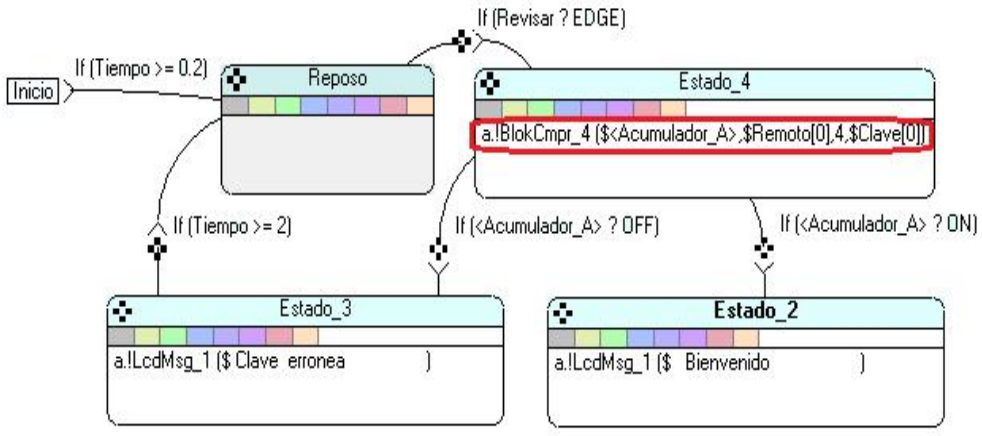


Aplicaciones: Se utiliza para comparar un arreglo de datos con otro.

Validación de claves de acceso en sistemas con seguridad.

Reconocimiento de comandos remotos para cambio de modo.

Validación de identidad con lectores de tarjetas RFID (Modbus).

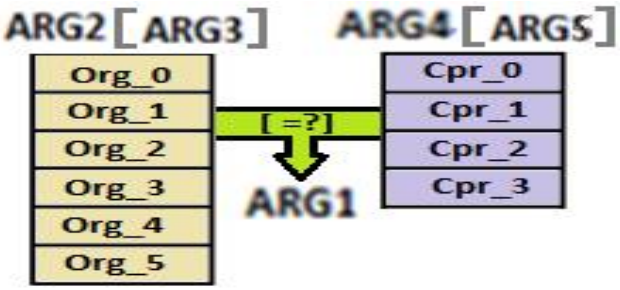


En un sistema de acceso por clave, cada que llega desde el dispositivo REMOTO una cadena de datos de validación, esta se compara, mediante la función BlokCmpr, con una CLAVE previamente guardada en memoria flash. En caso de ser válida la clave, se presenta el mensaje “Bienvenido” y en caso contrario el mensaje “Clave erronea”.

BLOKREVS_5. Revisión de bloque de datos en otro. ARG1 recibe la posición en la cual el bloque de datos ARG2[ ] con longitud ARG3 coincide con parte del bloque de datos ARG4[ ] con longitud ARG5. Retorna la longitud ARG5 en caso contrario.

ARG1(ARG4[ARG5] Contiene ARG2[ARG3])?

ARG1 = Variable de salida.
ARG2 = Vector[ ] de datos.
ARG3 = Longitud de datos.
ARG4 = Vector[ ] compara.
ARG5 = Longitud compara.

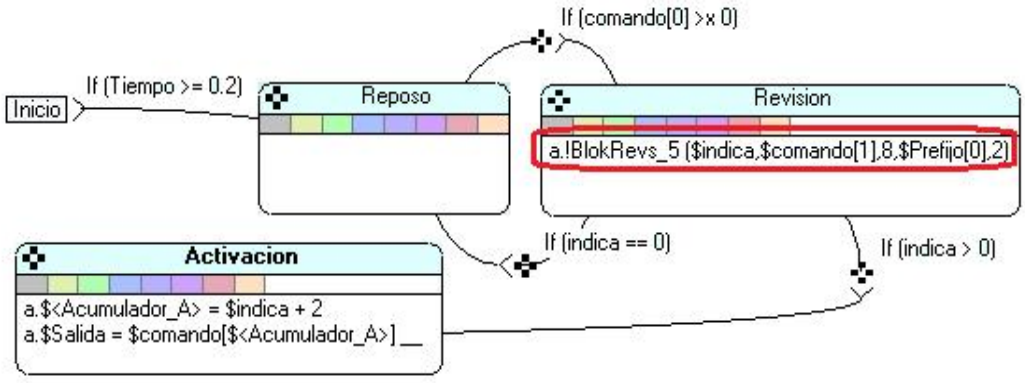


Aplicaciones: Se utiliza para detectar sub arreglos contenidos en arreglos de datos.

Selección de comandos remotos por comparaciones parciales.

Detección del inicio de una sub trama de datos de comunicación.

Procesamiento de comandos AT presentes en módem Wifi y Gprs.

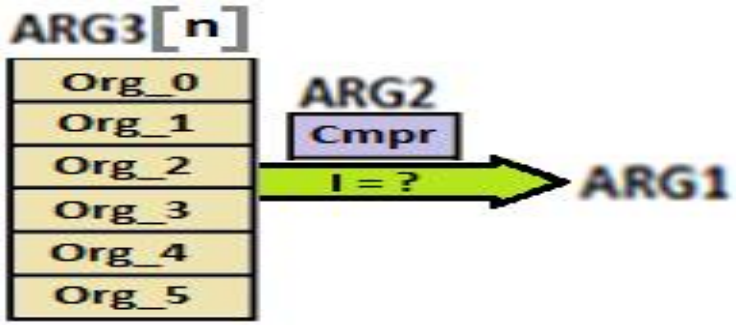


En un sistema de control con órdenes remotas, que llegan al vector COMANDO con longitud 8, al llegar una cadena de datos (detectado cuando COMANDO[0] > cero), con un encabezado igual al contenido del vector PREFIJO con longitud 2 (“V=”), asignado a la variable INDICA por la función BlokRevs, se carga el valor del dato de la cadena COMANDO apuntado por INDICA + 2 en la variable de salida del control

TABFIND_3. Buscar indicador de entrada en tabla. Buscar y cargar en ARG1 el índice del ítem de la tabla ARG3[Longitud] que contiene un dato similar a ARG2.

If (ARG2 >= ARG3[Indice]) ARG1=Indice

ARG1 = Indice obtenido.
ARG2 = Variable entrada.
ARG3 = Tabla búsqueda.

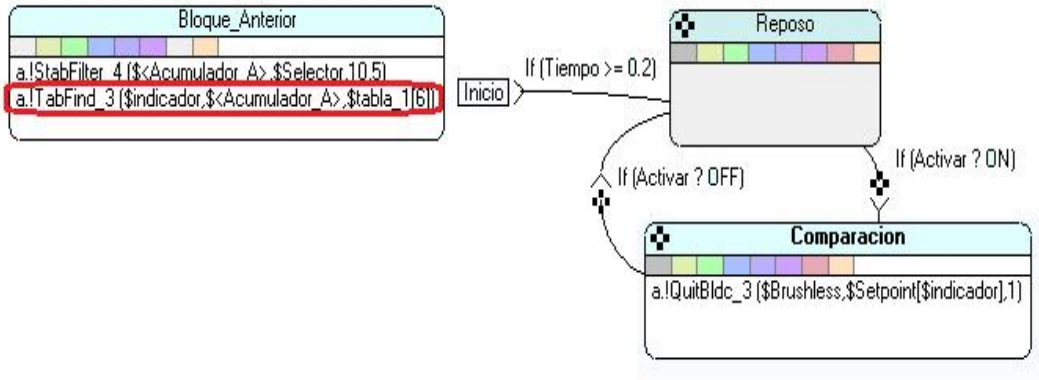


Aplicaciones: Se utiliza para discretizar entradas continuas no lineales.

Interpretación de selectores con resistencias en entradas analógicas.

Lectura de varios pulsadores por una entrada y procesamiento con FixKeyb.

Transformaciones no lineales de curvas de variables.

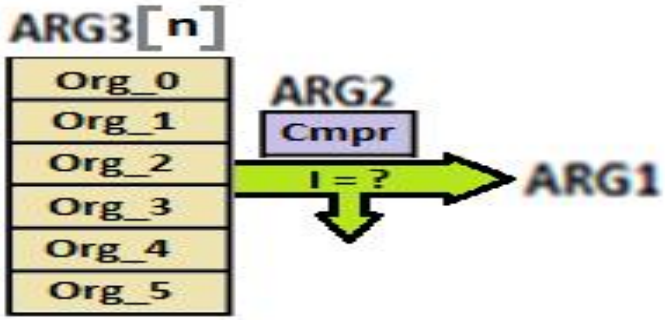


Un sistema de control con múltiples velocidades, seleccionadas mediante un SELECTOR conectado a la entrada (estabilizada para cambios menores a 10 unidades durante 5 ciclos de la tarea), utiliza la función TabFind para buscar en el vector TABLA_1 la variable estabilizada y entregar su posición en la variable INDICADOR, la cual apunta al dato del vector SETPOINT usado como consigna de velocidad en la salida BRUSHLESS.

FINDNEXT_3. Buscar indicador siguiente en tabla. Buscar con avance positivo y cargar en ARG1 el índice del ítem de la tabla ARG3[Longitud] que contiene un dato similar a ARG2.

If (ARG2 = ARG3[Indice ++]) ARG1= Índice.

ARG1 = Índice obtenido.
ARG2 = Variable entrada.
ARG3 = Tabla búsqueda.

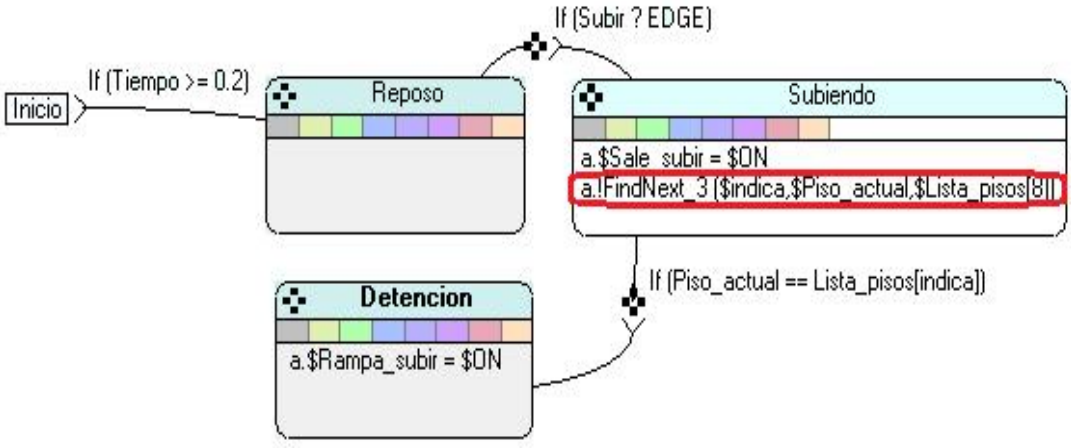


Aplicaciones: Se utiliza para hacer búsquedas con avance positivo.

Buscar pisos en el track ascendente de ascensores.

Buscar la posición por exceso de un objeto en un almacenamiento.

Búsqueda de mínimo / máximo valor contenido en un arreglo.

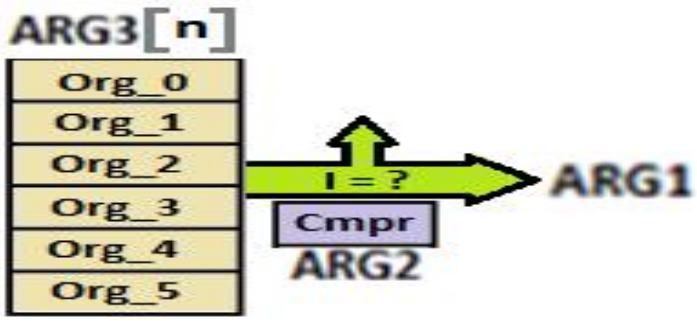


En el control de un ascensor, al recibir la orden SUBIR, se energiza la salida SALE_SUBIR y se utiliza la función FindNext para entregar en la variable INDICA, la posición del dato del vector LISTA_PISOS inmediatamente mayor a la variable PISO_ACTUAL. Cuando esta última variable es igual al piso siguiente (PISO_ACTUAL=LISTA_PISOS[INDICA]), el ascensor entra en régimen de frenado.

FINDPREV_3. Buscar indicador previo en tabla. Buscar con avance negativo y cargar en ARG1 el índice del item de la tabla ARG3[Longitud] que contiene un dato similar a ARG2.

if(ARG2 = ARG3[Indice --]) ARG1=Índice.

ARG1 = Índice obtenido.
ARG2 = Variable entrada.
ARG3 = Tabla búsqueda.

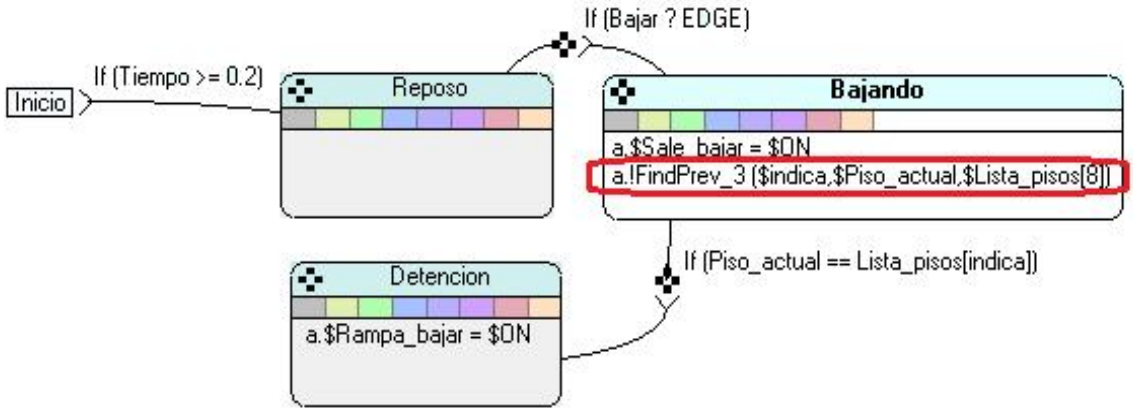


Aplicaciones: Se utiliza para hacer búsquedas con avance negativo.

Buscar pisos en el track descendente de ascensores.

Buscar la posición por defecto de un objeto en un almacenamiento.

Búsqueda de mínimo / máximo valor contenido en un arreglo.



En el control de un ascensor, al recibir la orden BAJAR, se energiza la salida SALE_BAJAR y se utiliza la función FindPrev para entregar en la variable INDICA, la posición del dato del vector LISTA_PISOS inmediatamente menor a la variable PISO_ACTUAL. Cuando esta última variable es igual al piso que sigue descendentemente (PISO_ACTUAL=LISTA_PISOS[INDICA]), el ascensor entra en régimen de frenado.

QUESPAC_3. Espacio disponible en cola. ARG1 recibe la cantidad de variables disponibles en la cola ARG2[ ] con longitud ARG3. La cola ARG2[ ] es un buffer circular.

ARG1 = Len(ARG2[ARG3]).

ARG1 = Índice devuelto.
ARG2 = Vector[ ] de datos.
ARG3 = Longitud de la cola.

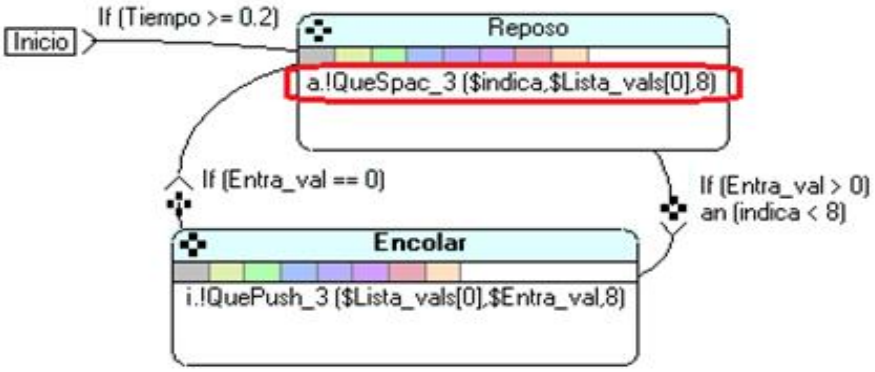


Aplicaciones: Se utiliza con las funciones que encolan datos para su posterior extracción.

Ubicar el espacio disponible en un parqueadero.

Determinar la cantidad de paradas que deberá realizar un ascensor.

Determinar la cantidad de ítems que faltan por atender en un sistema.



La función QueSpac entrega en la variable INDICA, la posición siguiente a llenar en la cola LISTA_VALS. En caso de haber espacio en la cola (INDICA < 8) y llegar un dato de entrada (ENTRA_VAL > 0), se encola el dato ENTRA_VAL en el vector LISTA_VALS, con longitud 8. La secuencia regresa al estado Reposo (en el cual revisa el espacio disponible en la cola) cuando la variable ENTRA_VAL es igual a cero de nuevo.

QUEPUSH_3. Encolamiento de valor en vector de datos. Encola el valor ARG2 en la cola ARG1[ ] con longitud ARG3, en caso de no haber espacio disponible no se encola el valor. La cola ARG2[ ] es un buffer circular.

ARG1[índice cola <= ARG3] = ARG2.

ARG1 = Vector[ ] de datos.
ARG2 = Variable a encolar.
ARG3 = Longitud de la cola.

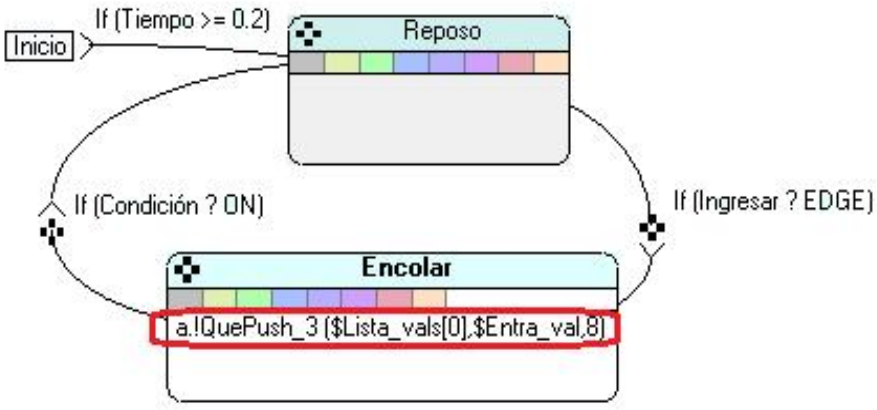


Aplicaciones: Se utiliza para encolar datos para su posterior extracción.

Agregar ítems a la lista de llamados de un ascensor.

Encolar llamados de servicio en un dispositivo de mantenimiento.

Encolar eventos ocurridos en un sistema para su posterior transmisión.



La función QuePush guarda en la cola LISTA_VALS con longitud de 8 datos, el valor de la variable ENTRA_VAL, en el momento en que la variable de entrada INGRESAR pasa de desactivada a activada. La secuencia regresa inmediatamente al estado de Reposo (“Condición ? ON” indica transición incondicional).

QUEPULL_3. Desencola valor desde vector de datos. Desencola un valor de la cola ARG2[ ] con longitud ARG3, hacia la variable ARG1. En caso de no haber datos para desencolar, retorna el último dato desencolado. La cola ARG2[ ] es un buffer circular.

ARG1 = ARG2[índice cola <= ARG3].

ARG1 = Valor devuelto.
ARG2 = Vector[ ] de datos.
ARG3 = Longitud de la cola.

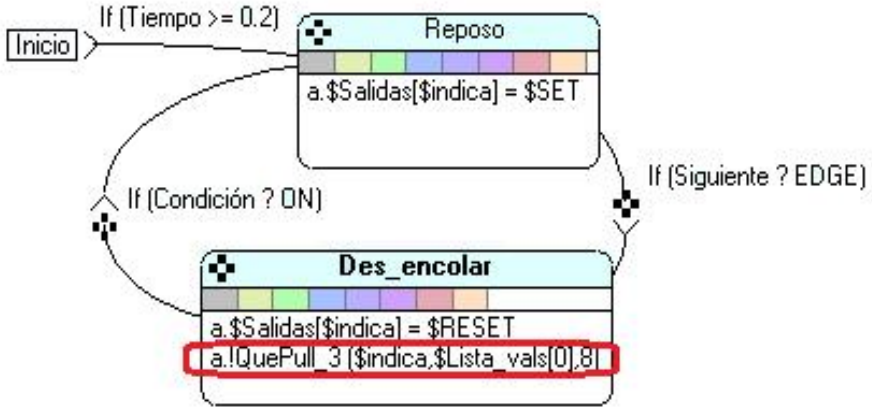


Aplicaciones: Se utiliza para desencolar datos previamente almacenados.

Atender el siguiente llamado en un sistema de turnos de operaciones.

Ejecutar la siguiente rutina de servicio en un sistema de mantenimiento.

Presentar la siguiente alarma sucedida en un sistema de vigilancia.



La secuencia activa la salida del autómata programable (mediante el vector indirecto SALIDAS), apuntada por la variable INDICA, la cual es entregada por la función QuePull al ser extraída de la cola LISTA_VALS con longitud de 8 datos, en el momento en el cual hay un flanco de activación de la entrada SIGUIENTE.

STKPUSH_3. Apilamiento de valor en vector de datos. Apila el valor ARG2 en la pila ARG1[ ] con longitud ARG3, en caso de no haber espacio disponible no se apila el valor. La pila ARG2[ ] es un buffer lineal.

ARG1[índice pila <= ARG3] = ARG2.

ARG1 = Vector[ ] de datos.
ARG2 = Variable a apilar.
ARG3 = Longitud de la pila.

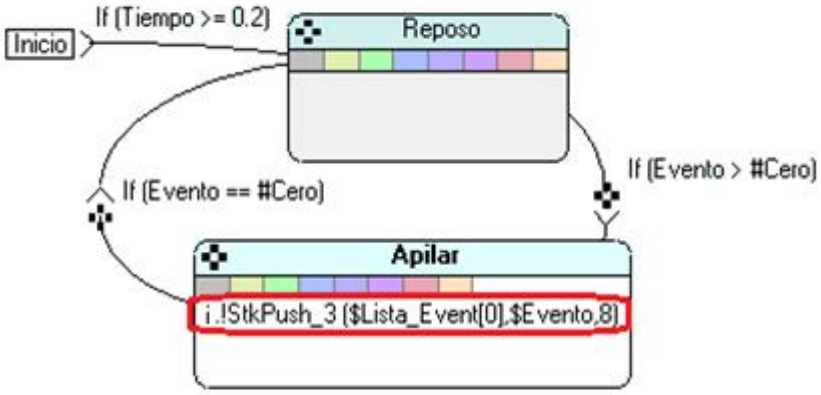


Aplicaciones: Se utiliza para apilar datos para su posterior extracción

Agregar ítems a la lista de llamados de un ascensor.

Apilar posiciones en el avance de un sistema de ida y vuelta.

Almacenar eventos ocurridos en un sistema para su posterior transmisión.

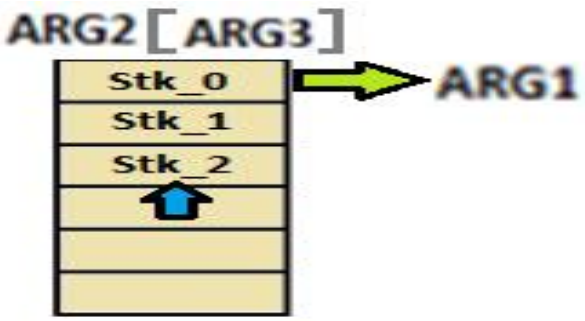


La función StkPush guarda en la pila LISTA_EVENT con longitud de 8 datos, el valor de la variable de entrada EVENTO, en el momento en que dicha variable es mayor que cero (la función opera solamente al iniciar el estado). La secuencia regresa al estado de Reposo, en el momento en el cual la entrada EVENTO pasa de nuevo al valor cero.

STKPULL_2. Des apila valor desde vector de datos. Des apila un valor de la pila ARG2[ ] hacia la variable ARG1. En caso de no haber datos para des apilar, retorna el último dato des apilado. La pila ARG2[ ] es un buffer lineal.

ARG1 = ARG2[índice pila].

ARG1 = Valor devuelto.
ARG2 = Vector[ ] de datos.

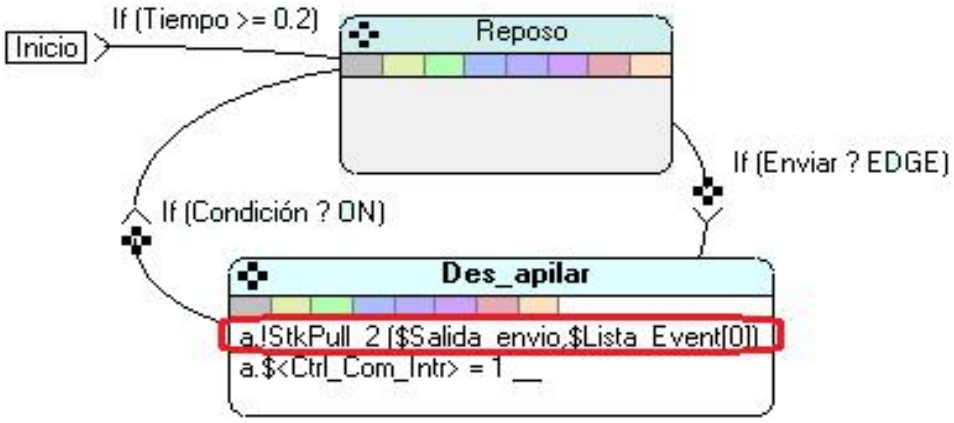


Aplicaciones: Se utiliza para des apilar datos previamente almacenados.

Regresar a las posiciones apiladas previamente por un sistema de ida y vuelta.

Presentar alarmas ocurridas desde la más reciente hasta la más lejana.

Deshacer operaciones en orden invertido a su ocurrencia.

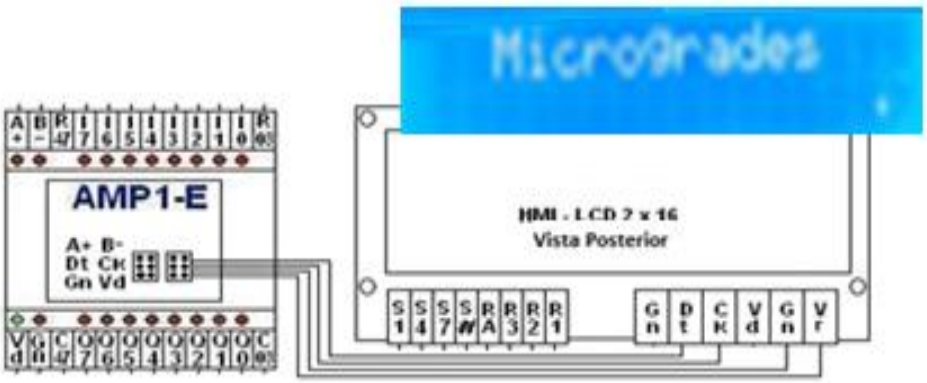


La secuencia envía un dato por el bus interno del autómata programable (cargando la variable del RTOS Ctrl_Com_Intr con el valor 1). El dato a enviar es cargado por la función StkPull en la variable tipo monitor SALIDA_ENVIO, desde la pila LISTA_EVENT, en el momento en el cual hay un flanco de activación de la entrada ENVIAR. La secuencia regresa inmediatamente al estado Reposo.

MANIOBRAS FUNCIONALES DE INTERFAZ HMI.

LCDMSG_1. Visualizar mensajes en LCD. Esta función presenta el mensaje ARG1 en un display LCD 2x16. La función envía al RTOS del autómata programable el formato mediante el cual se presentan mensajes y variables. Los formatos de presentación tienen la siguiente estructura: "Mensaje (###0) mensaje (###0)"\.

ARG1 = Formato de presentación en el display.

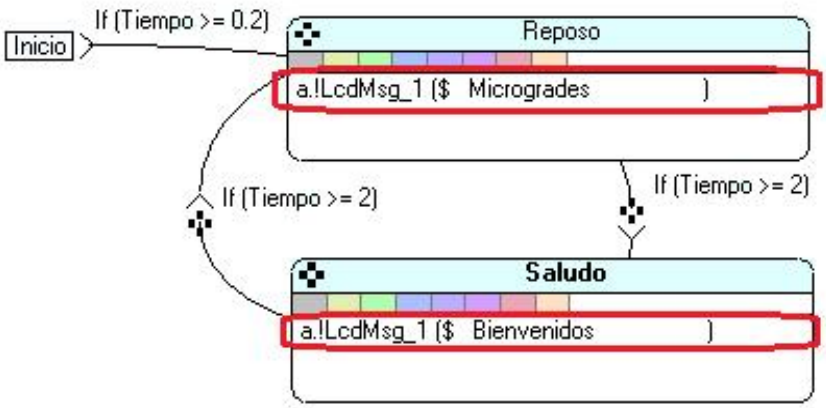


Aplicaciones: Se usa para presentar variables conjuntamente con la función LcdVis.

Presentar mensajes de saludo en sistemas con display LCD 2 x 16.

Presentar las variables de un proceso, en tiempo real.

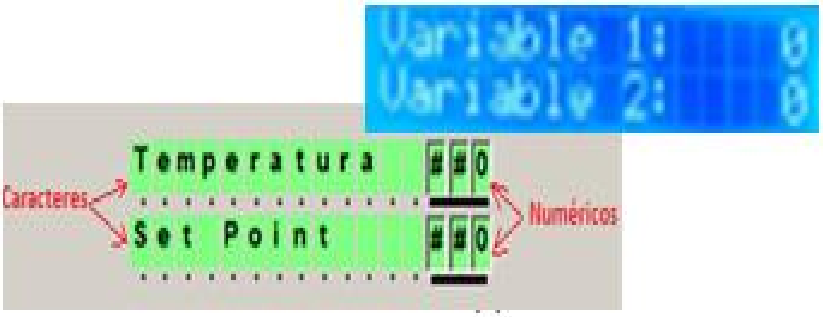
Interfaz para el ajuste de datos de las aplicaciones, en campo.



La secuencia presenta dos mensajes (a través del display LCD 2 x 16) a intervalos de tiempo determinados. El mensaje “Microgrades” es presentado durante 2 segundos y el mensaje “Bienvenidos” es presentado durante otros 2 segundos.

LCDVIS_3. Presentar variable en visualizador (###0) del display LCD 2 x 16 con orden ARG2 en Modo ARG3. Visualizador[ARG2] = &ARG1. Las variables se presentan en tiempo real y pueden tener formato decimal, hexadecimal, entero o tipo byte. Cada formato de mensaje puede contener hasta 8 visualizadores de presentación de variables, los cuales pueden presentar valores con justificación o sin ella, caracteres especiales en ASCII y signos de puntuación.

ARG1 = Variable a presentar.
ARG2 = Número del visualizador.
ARG3 = Modo visualización.

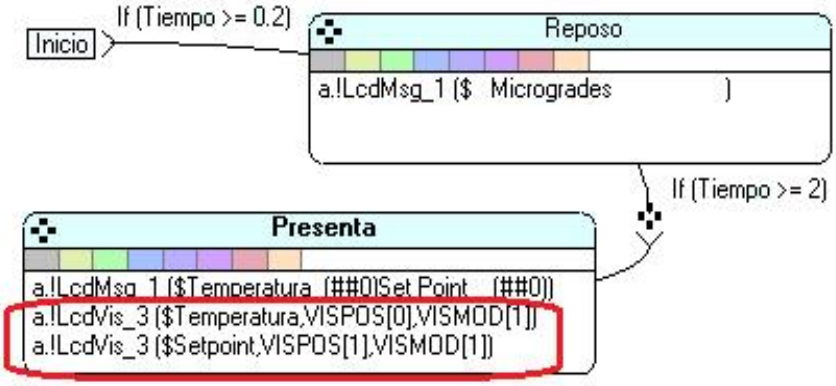


Aplicaciones: Se usa para presentar variables conjuntamente con la función LcdMsg.

Presentar las variables de un proceso, en formatos decimal y hexadecimal.

Interfaz para el ajuste de datos de las aplicaciones, en campo.

Presentación de tiempos y conteos de los procesos controlados.

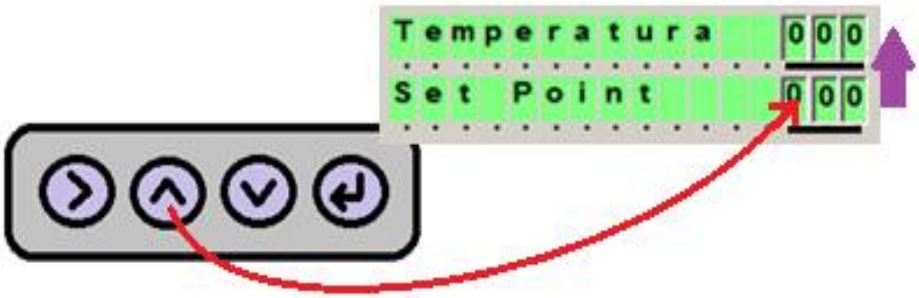


La secuencia de interfaz HMI LCD 2 x 16, inicia presentando el mensaje de saludo “Micrograde s” durante 2 segundos, luego de lo cual, se presentan las variables TEMPERATURA y SETPOINT en formato de entero decimal, utilizando la función LcdVis para indicar al RTOS como ha de ser visualizada cada variable.

INCADJS_4. Ajuste incremental de variable ARG1 en valor ARG2 hasta el valor ARG3 al pulsar la tecla con código ARG4.

If(Teclado == ARG4 And ARG1 < ARG3) ARG1 += ARG2.

ARG1 = Variable a ajustar.
ARG2 = Valor de incremento (Paso).
ARG3 = Máximo valor de la variable.
ARG4 = Código tecla incrementar.

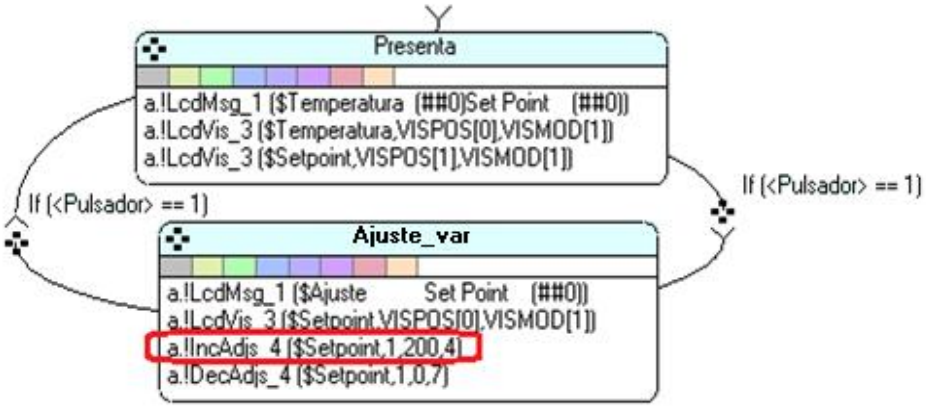


Aplicaciones: Se usa para editar datos conjuntamente con decAdjs y AdjSave.

Ajuste ascendente de setpoints de control en aplicaciones.

Recorrido ascendente por menús de variables.

Ajuste fino y grueso de variables temporales durante la marcha de un sistema.

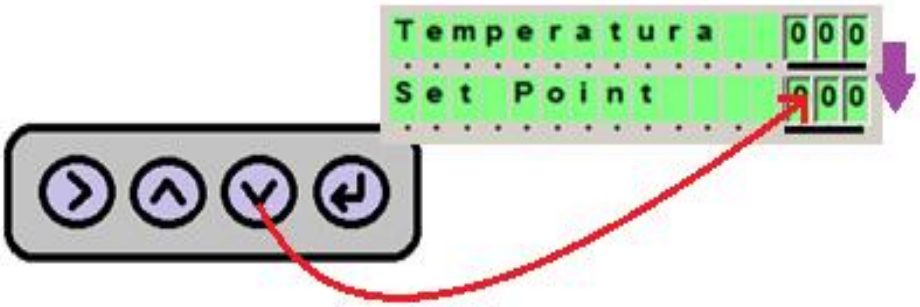


La secuencia visualiza las variables TEMPERATURA Y SETPOINT en el estado Presenta. Al ser detectado el pulsador del Hmi Lcd 2 x 16 con código 1, repetidamente, se pasa de un estado al otro. En el estado Ajuste_var, se visualiza y ajusta la variable SETPOINT en forma incremental, mediante la función IncAdjs, de 1 en 1 hasta 200 cada que se detecte el pulsador del Hmi Lcd 2 x 16 con código 4.

DECADJS_4. Ajuste decremental de variable ARG1 en el valor ARG2 hasta el valor ARG3 al pulsar la tecla con código ARG4.

If(Teclado == ARG4 And ARG1 > ARG3) ARG1 -= ARG2.

ARG1 = Variable a ajustar.
ARG2 = Valor de decremento (Paso).
ARG3 = Mínimo valor de la variable.
ARG4 = Código tecla decrementar.

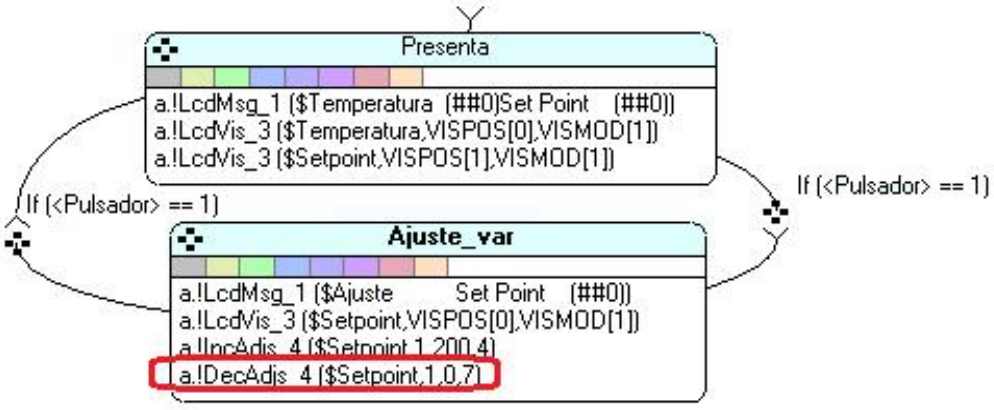


Aplicaciones: Se usa para editar datos conjuntamente con IncAdjs y AdjSave.

Ajuste descendente de setpoints de control en aplicaciones.

Recorrido descendente por menús de variables.

Ajuste fino y grueso de variables temporales durante la marcha de un sistema.

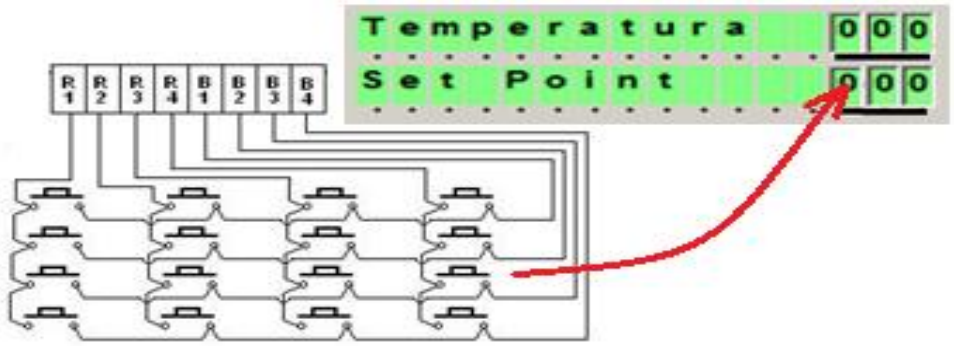


La secuencia visualiza las variables TEMPERATURA Y SETPOINT en el estado Presenta. Al ser detectado el pulsador del Hmi Lcd 2 x 16 con código 1, repetidamente, se pasa de un estado al otro. En el estado Ajuste_var, se visualiza y ajusta la variable SETPOINT en forma decremental, mediante la función DecAdjs, de 1 en 1 hasta 0 cada que se detecte el pulsador del Hmi Lcd 2 x 16 con código 7.

CODADJS_3. Ajuste por codificación de variable ARG1 con el valor en ARG2 (Al pulsar una tecla o al cambiar valor de ARG2). Cuando la tecla es igual a ARG3 se carga 0 en ARG1.

If(ARG2 < 10) ARG1 = ARG1 * 10 + ARG2.

ARG1 = Variable a ajustar.
ARG2 = Variable de teclado.
ARG3 = Código tecla borrar.

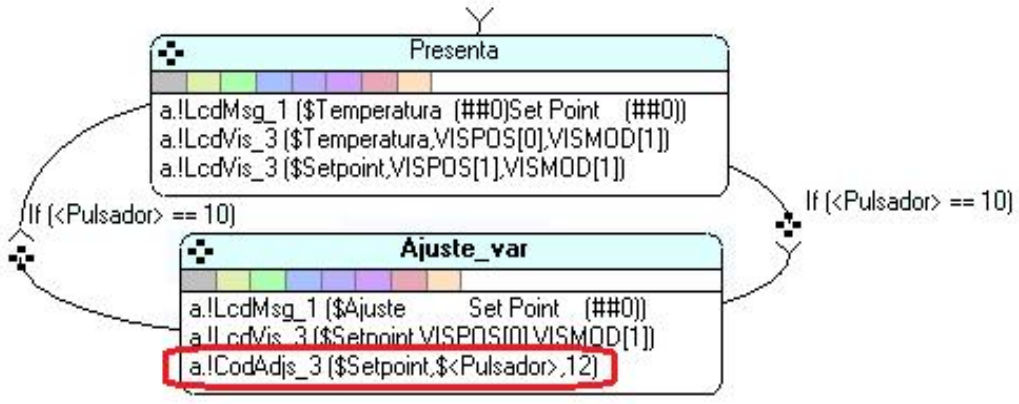


Aplicaciones: Se usa para editar datos conjuntamente con IncAdjs, DecAdjs y AdjSave.

Ajuste numérico de setpoints de control en aplicaciones.

Selección de ítems en menús de variables.

Entrada de claves de validación en un sistema de producción.

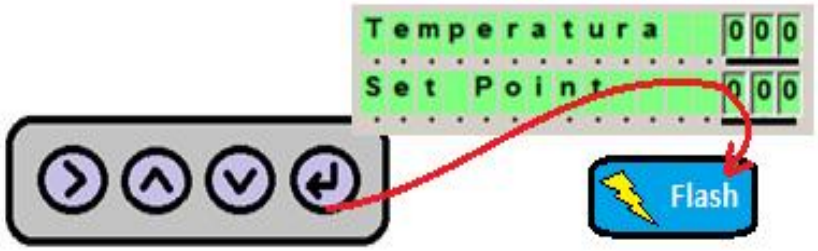


La secuencia visualiza las variables TEMPERATURA Y SETPOINT en el estado Presenta. Al ser detectado el pulsador del Hmi Lcd 2 x 16 con código 1, repetidamente, se pasa de un estado al otro. En el estado Ajuste_var, se visualiza y ajusta la variable SETPOINT digitando el valor deseado, mediante la función CodAdjs, en la cual se especifica que al ser detectado el pulsador del Hmi Lcd 2 x 16 con código 12, se borra el valor digitado.

ADJSAVE_3. Guardar variable Ram en Flash Ajsute. Guardar Variable ARG1 en Ajuste ARG2 al pulsar Tecla con código ARG3\n\nSe usa para guardar datos editados con IncAdjs y DecAdjs.

If(Edge(Teclado) == ARG3) ARG2 = ARG1.

ARG1 = Variable a guardar.
ARG2 = Ajuste de destino.
ARG3 = Código tecla grabar.

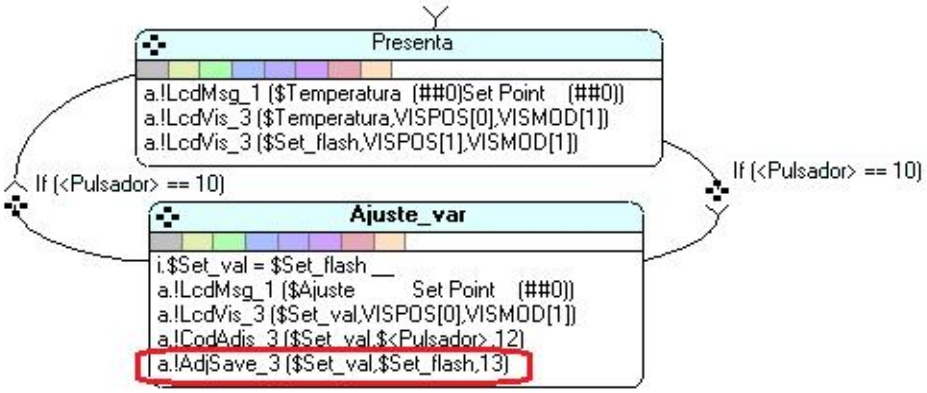


Aplicaciones: Guardar variables en menús de ajuste con HMI LCD 2 x 16.

Ajuste de setpoints de control en aplicaciones con menú.

Menú de ajuste de tiempos para programadores de secuencias.

Menús de calibración de las entradas de un autómata programable.

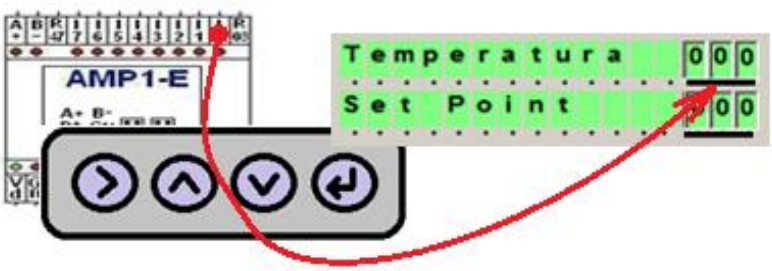


La secuencia visualiza las variables TEMPERATURA Y SET_FLASH en el estado Presenta. Al ser detectado el pulsador del Hmi Lcd 2 x 16 con código 1, repetidamente, se pasa de un estado al otro. En el estado Ajuste_var, se visualiza y ajusta la variable SET_FLASH, la cual se carga temporalmente en la variable SET_VAL. La función AdjSave almacena el valor ajustado en SET_VAL, hacia la variable SET_FLASH, al ser detectado el pulsador del Hmi Lcd 2 x 16, con código 13.

FIXKEYB_4. Asigna el valor ARG4 al teclado cuando la variable ARG1 esté en el rango [ARG2 - ARG3]. Permite la revisión de entradas analógicas como pulsadores de HMI, para realizar el ajuste de variables conjuntamente con IncAdjs, DecAdjs y AdjSave.

If((ARG1 >= ARG2) && (ARG1 =< ARG3)) Teclado = ARG4.

ARG1 = Variable a detectar.
ARG2 = Valor de comparación bajo.
ARG3 = Valor de comparación alto.
ARG4 = Código tecla a asignar.

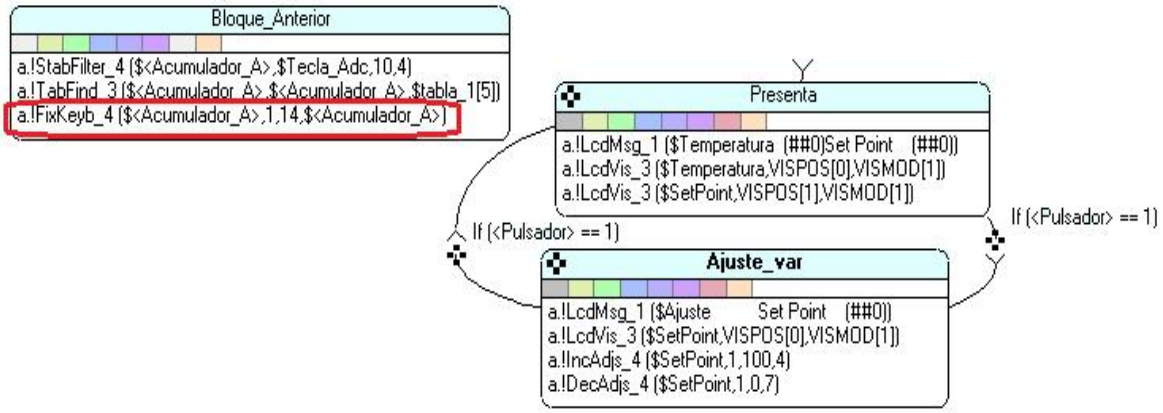


Aplicaciones: Se usa para revisar las entradas como pulsadores del HMI.

Ajuste ascendente-descendente de setpoints de control en aplicaciones.

Recorrido ascendente-descendente por menús de variables.

Ajuste fino y grueso de variables temporales durante la marcha de un sistema.



Las maniobras ubicadas en el Bloque Anterior, estabilizan la entrada TECLA_ADC en una banda de 10 unidades durante 4 ciclos de la tarea. El resultado de la estabilización se pasa a través del registro <Acumulador>, a la función que busca el indicador, el cual es transferido al RTOS mediante la función FixKeyb, cada que el indicador tiene un valor entre 1 y 4, para que sea utilizado por las secuencias de ajuste de variable con Hmi Lcd 2 x 16.

FUNCIONES DE USUARIO CON BLOQUES DE MANIOBRAS.

Son llamados a bloques de maniobras creados al interior de las tareas. Los bloques de maniobras se agregan a cada tarea desde el menú desplegable que aparece al hacer click derecho sobre el fondo de la tarea. Las maniobras de estos bloques utilizan las variables de la tarea, las constantes y tablas (globales) y los argumentos pasados en la invocación de la función, desde los estados y otros bloques de maniobras.

Las funciones de usuario con bloques de maniobras se componen del nombre de la función (ManFunc_5) y los argumentos encerrados entre paréntesis (variables o constantes). Los argumentos no utilizados se pasan como “void”.

MANFUNC_5. Función en Bloque de maniobras. Invocación a función compuesta por varias maniobras, hasta con cuatro argumentos pasados por referencia. La función puede retornar valores en los acumuladores locales.

Bloque_Maniobras_ARG1(ARG2, ARG3, ARG4, ARG5).

ARG1 = Nombre de la función.
ARG2 … ARG5 = Primer argumento … Cuarto argumento (opcionales).

Aplicaciones: Se usa con grupos de maniobras, repetidos en estados.

Ahorrar espacio en la memoria de programa cuando hay funciones repetidas.

Disponer de bloques de programa parametrizables para mayor versatilidad.



La secuencia energiza durante 2 segundos la SALIDA_0 si la variable ENTRA_0 es mayor que cero. Así mismo, energiza durante 3 segundos la SALIDA_1 si la variable ENTRA_1 es mayor que cero. La secuencia utiliza en cada estado, la función ManFunc para invocar el bloque de maniobras “Incrementos”, el cual realiza la comparación y activación a través de las variables ARGUM_2 y ARGUM_3 respectivamente.

FUNCIONES TEXTUALES DE USUARIO

Son llamados a funciones creadas por el programador en lenguaje C. Para crear el código de estas funciones se debe revisar el manual de compilador SDCC para la adaptación S08 de Motorola (Freescale, Nxp). Al digitar “v.” aparece la lista de variables y argumentos utilizables en las funciones textuales. Al digitar “f.” aparece la lista de funciones invocables. Las funciones textuales se componen del nombre de la función (userfunc_5) y los argumentos encerrados entre paréntesis (variables o constantes). No se recomienda invocar funciones de las librerías estándar del SDCC (provocan fallos en el RTOS).

USERFUNC_5. Función de Usuario en lenguaje textual. Invocación a función escrita en lenguaje textual, hasta con cuatro argumentos pasados por referencia. La función puede retornar valores en los acumuladores locales.

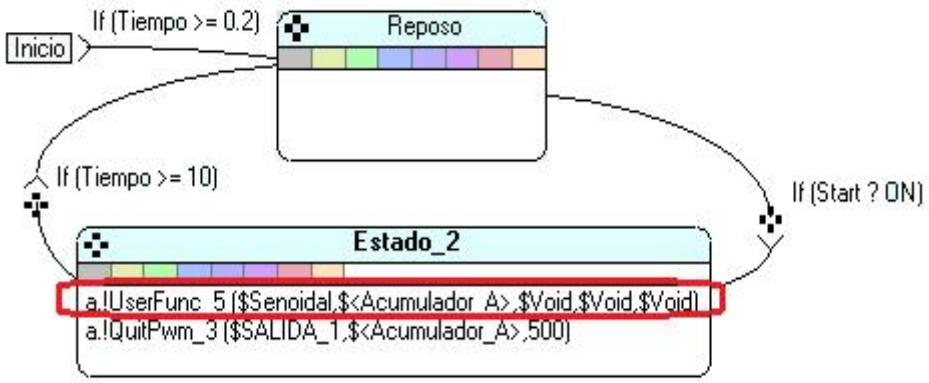
Función_Textual_ARG1(ARG2, ARG3, ARG4, ARG5).

ARG1 = Nombre de la función.
ARG2 … ARG5 = Primer argumento … Cuarto argumento (opcionales).

Aplicaciones: Se usa para realizar operaciones que no estén contenidas en Mgdmod.

Funciones de cálculo de CRC y algoritmos similares.

Llamado a funciones internas del sistema operativo del autómata programable.



La secuencia inicia en el estado Reposo y al detectar la activación de la entrada START, pasa al Estado_2, en el cual utiliza la función UserFunc para invocar la rutina “Senoidal”, la cual entrega en la variable <Acumulador> los valores de una sucesión sinusoidal, los cuales serán modulados por el RTOS través de la SALIDA_1, en la modalidad PWM.