UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERIA ELECTRICA

"Diseño de una estación de laboratorio de automatización y control automático para la Escuela de Ingeniería Eléctrica de la UES"

PRESENTADO POR:

SIXTO EDWIN ARGUETA DÍAZ

CARLOS ALEXANDER PALACIOS HENRÍQUEZ

MARIO ALEJANDRO VÁSQUEZ RAMÍREZ

PARA OPTAR AL TITULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, NOVIEMBRE DE 2010

UNIVERSIDAD DE EL SALVADOR

RECTOR :

MSc. RUFINO ANTONIO QUEZADA SÁNCHEZ

SECRETARIO GENERAL :

LIC. DOUGLAS VLADIMIR ALFARO CHÁVEZ

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO :

ING. MARIO ROBERTO NIETO LOVO

SECRETARIO :

ING. OSCAR EDUARDO MARROQUÍN HERNÁNDEZ

ESCUELA DE INGENIERIA ELECTRICA

DIRECTOR :

ING. JOSE WILBER CALDERÓN URRUTIA

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERIA ELECTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título :

"Diseño de una estación de laboratorio de automatización y control automático para la Escuela de Ingeniería Eléctrica de la UES"

Presentado por :

SIXTO EDWIN ARGUETA DÍAZ

CARLOS ALEXANDER PALACIOS HENRÍQUEZ

MARIO ALEJANDRO VÁSQUEZ RAMÍREZ

Trabajo de Graduación Aprobado por:

Docente Director :

ING. RICARDO ERNESTO CORTEZ

San Salvador, Noviembre de 2010

Trabajo de Graduación Aprobado por:

Docente Director :

ING. RICARDO ERNESTO CORTEZ

DEDICATORIA

DEDICO ESTE TRABAJO DE TESIS Y AGRADEZCO GRANDEMENTE A:

DIOS
POR DARME LA CAPACIDAD DE TOMAR LAS DECISIONES CORRECTAS, Y DE ENFRENTAR CADA NUEVO RETO CON VALOR, ENTUSIASMO Y DEDICACIÓN.

MIS PADRES JOSÉ ARGUETA Y ARGELIA DÍAZ
QUIENES CON SU ESFUERZO Y ENTREGA ME HAN AYUDADO A ALCANZAR CADA UNO DE MIS SUEÑOS Y A CRECER COMO PERSONA Y PROFESIONAL.

MIS HERMANOS
POR MOSTRARME CON SU EJEMPLO QUE EL ÉXITO SE CONSIGUE A BASE DE TRABAJO Y ESFUERZO; ADEMÁS POR ENSEÑARME A CREER EN MIS DECISIONES Y A TRABAJAR CADA DÍA POR LO QUE QUIERO.

MI NOVIA BRENDA FLORES
POR CAMINAR A MI LADO, DARME LAS GANAS Y LA FUERZA PARA SEGUIR ADELANTE CADA VEZ QUE ME FALTARON, Y POR RENOVAR MI AUTOCONFIANZA Y CREER EN MI TANTO COMO EN SI MISMA.

MIS AMIGOS
POR ACOMPAÑARME SIEMPRE Y HACER DE CADA UNO DE MIS ÉXITOS EL SUYO, POR NO DEJARME DESISTIR Y ENSEÑARME A VER LA VIDA MÁS ALLÁ DE CUALQUIER FRONTERA.

CARLOS PALACIOS Y MARIO VÁSQUEZ
POR SER COMPAÑEROS Y AMIGOS DURANTE CADA UNO DE LOS MESES DEDICADOS A ESTE TRABAJO.

Y ESPECIALMENTE AGRADEZCO A MI HIJA GISSELLE ARGUETA
POR DARME UN MOTIVO POR EL CUAL LUCHAR, A PESAR DE LOS TROPIESOS ELLA ME DIO LAS FUERZAS PARA SEGUIR ADELANTE Y LUCHAR POR MI Y ESPECIALMENTE POR ELLA Y ASI LOGRAR MIS METAS.

Sixto Edwin Argueta Díaz

DEDICATORIA

AGRADEZCO PRIMERAMENTE A DIOS POR SER QUIEN HA ESTADO A MI LADO EN TODO MOMENTO DÁNDOME LAS FUERZAS NECESARIAS PARA CONTINUAR LUCHANDO DÍA TRAS DÍA Y SEGUIR ADELANTE ROMPIENDO TODAS LAS BARRERAS QUE SE ME HAN PRESENTADO.

DEDICO ESTE TRABAJO DE TESIS Y AGRADEZCO GRANDEMENTE A MIS PADRES MARÍA CLEOTILDE HENRÍQUEZ DE PALACIOS Y RAMÓN DE JESÚS PALACIOS QUE SIEMPRE ME APOYARON EMOCIONAL Y ECONÓMICAMENTE PARA QUE ESTE SERVIDOR SEA EL PROFESIONAL EN EL QUE ME HE CONVERTIDO. A MIS HERMANOS RAMÓN PALACIOS, JOHANNA PALACIOS Y KATHERINE PALACIOS QUIENES ESTUVIERON CONMIGO EN TODO MOMENTO Y ME APOYARON INCONDICIONALMENTE.

EN FIN AGRADEZCO A TODA MI FAMILIA, AMIGOS Y PERSONAS QUE EN MI CREYERON, Y TERMINO CON ESTAS PALABRAS DICIENDO, CADA GOTA DE AYUDA QUE ME HAN BRINDADO SE LAS RETRIBUIRÉ A USTEDES Y A LA SOCIEDAD HASTA DONDE EL SEÑOR ASÍ ME LO PERMITA.

Carlos Alexander Palacios Henríquez

DEDICATORIA

A DIOS TODOPODEROSO POR DARME Y LLEVAR EL CONTROL DE MI VIDA, NO DEJARME SOLO EN NINGÚN MOMENTO, ESCUCHAR MIS ORACIONES Y PERMITIR LA CULMINACIÓN DE MIS ESTUDIOS.

A MIS PADRES LUCIA GREGORIA RAMIREZ DE VASQUEZ Y OSCAR ALBERTO VASQUEZ GUADRON POR BRINDARME SIEMPRE SU APOYO, SU PACIENCIA, SUS SABIOS CONSEJOS Y ALIENTOS DE ÁNIMO PARA LOGRAR MIS METAS PROFESIONALES Y PERSONALES.

A MIS HERMANOS OSCAR EDUARDO Y GERARDINA ELIZABETH POR BRINDARME SU COMPRENSIÓN, AMOR, BRINDARME SU APOYO INCONDICIONAL Y MOTIVARME A SEGUIR ADELANTE.

A TODA MI FAMILIA Y A KAREN CORNEJO POR APOYARME DURANTE TODO EL DESARROLLO DE MI CARRERA, POR ESTAR CONMIGO EN TODO MOMENTO, POR SU CARIÑO, TOLERANCIA, PALABRAS DE ALIENTO Y CONFIANZA EN MÍ.

A CARLOS PALACIOS Y EDWIN ARGUETA POR SER COMPAÑEROS Y AMIGOS DURANTE CADA UNO DE LOS MESES DEDICADOS A ESTE TRABAJO.

Mario Alejandro Vásquez Ramírez

ÍNDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

En un mercado global cada vez más exigente, una de las piezas claves que distingue y otorga competitividad a la industria moderna es la incorporación de procesos productivos que incluyan tecnologías de punta. Es así como la automatización se convierte en pieza fundamental para la optimización de los procesos y la utilización de recursos, incrementando las utilidades de las empresas. La capacitación en ésta área es una acción estratégica para el crecimiento y desarrollo profesional de los alumnos. Siendo el único camino para lograr una mayor competitividad. Para lo cual es importante contar con las herramientas y equipos necesarios para apoyar al estudiante a que comprenda la teoría y refuerce estos conceptos con la práctica.

Por otra parte en el campo del control automático, desarrollos recientes basados en software en el área de herramientas de prototipo rápido han dado lugar a un nuevo paradigma en el desarrollo de sistemas de control. En el sector industrial, estas herramientas han permitido a los desarrolladores un rápido y fácil diseño, simulación e implementación compleja en tiempo real de sistemas de control. Así esta tecnología puede ser explotada para realizar mejoras en sistemas de control educacionales.

El trabajo de graduación está enfocado en dos áreas, como lo son el automatismo industrial y el control automático. El automatismo industrial es es el uso de sistemas o elementos computarizados y electromecánicos para controlar maquinarias y/o procesos industriales sustituyendo a operadores humanos; en cambio, el control automático busca cambiar el comportamiento del sistema a controlar (es decir, su salida) mediante la manipulación de las magnitudes de entrada.

En el área de automatismo industrial se realizo un módulo de entrenamiento con el S7-200 CPU 224 XP, con el cual los alumnos puedan realizar diversas prácticas, ya que dispone de teclado, display y demás periféricos necesarios para aplicar los conocimientos sobre funciones básicas (contadores, timer y manipulación de registros), Funciones de rango medio (aritméticas, lógicas y de control de flujo) y funciones intermedio-avanzadas (manipulación de bits, subrutinas e interrupciones), para lo cual se realizaron nueve guías de laboratorio que servirán como una referencia para el alumno.

Además se creó un manual sobre el programa SCADA WinLog [(1)] (SCADA, acrónimo de Supervisory Control And Data Acquisition), enfatizando en el uso de las diferentes herramientas de desarrollo con las que cuenta WinLog para realizar un pequeño sistema SCADA. En este apartado se da un ejemplo de un pequeño sistema SCADA con fines didácticos en el que se simula un proceso industrial referido a la operación de una mini central hidroeléctrica, el cual es monitoreado con el programa WinLog utilizando el S7-200 como controlador automático del proceso.

Se realizo además una pequeña introducción a los buses de campo, y se dan las especificaciones y el diseño de una red AS-i que utiliza el S7-200 como maestro y como parte de uno de sus esclavos se encuentra el LOGO! 24RC formando una comunicación entre ambos MASTER-SLAVE.

En el área de control automático se realizo un sistema de prototipado rápido, utilizando una tarjeta de Procesamiento Digital de Señal (ezDSPF2812), una tarjeta de adecuación de señales y el modulo FeedBack 150F. Con este equipo, la velocidad y la posición del motor eléctrico están controladas por la tarjeta DSP, y el sistema de control completo es programado gráficamente con el software MATLAB Simulink. Una interfaz gráfica de usuario se utiliza para proporcionar una interacción en tiempo real entre el estudiante y el sistema, incluyendo la selección de velocidad y posición, registro de datos y visualización. Se hizo una guía para la realización de un control PID de velocidad y posición utilizando este sistema.

(1): Disponible en http://www.sielcosistemi.com/en/download/public/winlog_lite.html

OBJETIVOS

GENERAL

- Mejorar el equipamiento de los laboratorios de control automático y automatismo industrial en la Escuela de Ingeniería Eléctrica, para reducir la brecha entre la teoría y la práctica, empleando software y hardware para el rápido y fácil diseño, simulación e implementación compleja en tiempo real, de sistemas de automatización y control.

ESPECIFICOS

- Elaborar el módulo de entrenamiento para el S7-200 y sus respectivas guías de laboratorio a implementarse en la Escuela de Ingeniería Eléctrica.

- Crear un manual del programa WinLog, para que los alumnos puedan realizar pequeños sistemas SCADA.

- Proporcionar los conocimientos básicos en buses de campo, y específicamente de la red AS-i para que en un futuro se pueda contar con módulos de práctica en esta área.

- Elaborar la estación de trabajo de prototipo rápido para sistemas de control automático con su respectivo manual.

- Implementar como ejemplo de aplicación el control PID de velocidad y posición de un motor DC utilizando la estación de trabajo de prototipo rápido.

JUSTIFICACION

Los conocimientos sobre PLC's se estudian en la materia de automatismo industrial que actualmente se imparte en la Escuela de Ingeniería Eléctrica; los alumnos, para realizar sus prácticas y trabajos, se hacen valer de software de simulación como lo es PC_SIMU u otros, debido a que actualmente en la escuela de ingeniería eléctrica no se cuenta con un módulo de prácticas con PLC, lo que hace muy necesario proveer a la escuela de ingeniería eléctrica de un módulo de entrenamiento en base al S7-200, el cual es uno de los PLC más comunes en el campo industrial y con el cual se pueden poner en práctica todas las funciones con las que cuentan los PLC.

Los típicos equipos de laboratorio de sistemas de control automático son inflexibles, costosos y además las prácticas de laboratorio son excesivamente laboriosas. Estos inconvenientes son abordados con el sistema propuesto, la estación de trabajo ofrece un entrenador de sistemas de control flexible, de bajo costo que es capaz de soportar una variedad de demostraciones y actividades experimentales fomentando al mismo tiempo la experimentación estudiantil y haciendo hincapié en los aspectos prácticos del diseño, simulación e implementación en tiempo real de sistemas de control. A menudo los cursos típicos de sistemas de control se centran en la teoría con poco énfasis en la aplicación y la práctica. Herramientas de prototipo rápido proporcionan una transición sin problemas de sistemas de diseño a implementación, y por lo tanto puede ser empleada para reducir la brecha entre la teoría y la práctica en nuestro curso de sistema de control.

CAPITULO 1

PRACTICAS AUTOMATISMO CON EL PLC S7-200

CAPITULO 1.
PRACTICAS DE AUTOMATISMO CON EL PLC S7-200

Mediante las prácticas planteadas a continuación se pretende que el estudiante, al resolverlas, vaya adquiriendo conocimientos y habilidades que le permitan resolver situaciones y problemas de diseño y montaje con los que puede encontrarse un ingeniero en el ambiente industrial o laboral, comenzando con ejercicios sencillos y aumentando gradualmente el nivel de dificultad.

Las prácticas planteadas en la guía tratan de representar situaciones cotidianas para así observar el gran rango de aplicaciones que tiene la tecnología de la automatización en nuestra vida.

La implementación de los ejercicios de aplicación propuestos en cada práctica, mediante el uso de los módulos para PLC construidos, ofrece una gran garantía de que al haber realizado todas las practicas se hayan adquirido buenas bases sobre el tema, las cuales serán muy útiles a la hora de hacer frente a los diversos problemas y situaciones que conlleva el ámbito de la automatización industrial.

1.1 PRACTICAS DE FUNCIONES BÁSICAS

1.1.1 CONTROL DOMOTICO DE VIVIENDA FAMILIAR

Objetivos:

- Estudiar y utilizar localidades de memoria en el PLC como lo son las marcas (Mx.x) y las variables (Vx.x) que son considerados registros Holding, los cuales no afectan las entradas ni las salidas pero pueden ser utilizadas en cualquier operación interna del PLC.

- Comprender la forma en que el PLC trata las entradas y salidas, aplicando los conceptos de registros de entrada/salida individual y de grupo.

Materiales a utilizar:

- 1 PC
- 1 Cable PC/PPI
- 1 S7-200
- 1 maqueta simuladora para domótica
- 24 cables de conexión

Planteamiento:

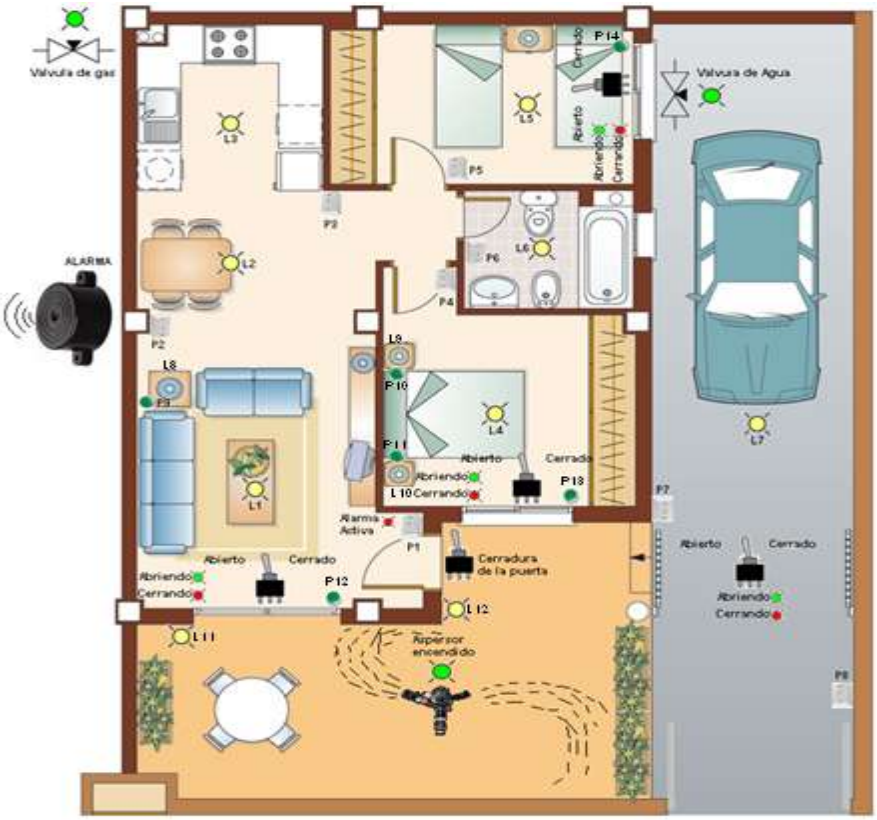


Fig. 1.1. Esquema de la vivienda

Se debe gestionar el uso de dispositivos de una vivienda usando el S7-200 con el fin de reducir el gasto de energía eléctrica, proporcionar seguridad y comodidad a la misma. La vivienda cuenta con pulsadores haciendo la vez de interruptores colocados en los sitios mostrados en el esquema. Se tienen 14 luces, tres ventanas que cuentan con persianas que puede cerrarse o abrirse automáticamente las cuales tiene sensores que indican persianas abiertas ó persianas cerradas. Se dispone de un portón en el garaje que es automático el cual al igual que las ventanas cuenta con sus respectivos sensores de portón abierto y portón cerrado; Además la casa cuenta con alarma.

Los dispositivos sensores de la vivienda están conectados a un bus de datos de 6 líneas, en las primeras 5 líneas se envía el número del sensor que se activo en determinado momento, para los pulsadores solo se utilizan estas líneas, pero para los interruptores final de carrera como lo son los sensores de ventanas abiertas o cerradas se utiliza la última línea que indica el estado en que se encuentra el interruptor; este dato llega directamente a las entradas del PLC para ser tratado. Los actuadores de la vivienda funcionan de la siguiente manera, todos están conectados a un bus de datos de 6 líneas, en las primeras 5 se envía el numero de dispositivo que se desea activar y en la última línea la acción que se quiere hacer (encender/1 o´ apagar/0). Si se desea más información sobre este simulador revisar el capitulo 2 que trata sobre la construcción del entrenador.

El automatismo funciona de la siguiente manera, al presionar algún pulsador una tan sola vez se enciende la luz respectiva y al presionarlo nuevamente se debe de apagar; por ejemplo, si se presiona P1 se debe encender L1. El portón se abre presionando P8 por fuera ó P7 por dentro, de igual manera si se presiona P8 ó P7 nuevamente el portón debe cerrarse. Con el portón abierto L7 está siempre apagada ó si está encendida se apaga al abrir el portón y se vuelve a encender al momento de dar la orden para cerrarlo solo si esta orden fue hecha con el pulsador P7. Las persianas se abren ó cierran con P12, P13 ó P14 para la sala, dormitorio 1 y dormitorio 2 respectivamente; con las persianas abiertas no se puede encender L1, L4 ó L5 según sea el caso, ó si cualquiera está encendida se apaga al abrir las persianas y se vuelve a encender al momento de dar la orden para cerrarlas.

Actuadores							Sensores						
Dispositivo	Código						Dispositivo	Código					
	Q0.5	Q0.4	Q0.3	Q0.2	Q0.1	Q0.0		I0.5	I0.4	I0.3	I0.2	I0.1	I0.0
L 1	0/1	0	0	0	0	1	**P 1**	0	0	0	0	0	1
L 2	0/1	0	0	0	1	0	**P 2**	0	0	0	0	1	0
L 3	0/1	0	0	0	1	1	**P 3**	0	0	0	0	1	1
L 4	0/1	0	0	1	0	0	**P 4**	0	0	0	1	0	0
L 5	0/1	0	0	1	0	1	**P 5**	0	0	0	1	0	1
L 6	0/1	0	0	1	1	0	**P 6**	0	0	0	1	1	0
L7	0/1	0	0	1	1	1	**P7**	0	0	0	1	1	1
L8	0/1	0	1	0	0	0	**P8**	0	0	1	0	0	0
L9	0/1	0	1	0	0	1	**P9**	0	0	1	0	0	1
L10	0/1	0	1	0	1	0	**P10**	0	0	1	0	1	0
ALARMA	0/1	0	1	0	1	1	**P11**	0	0	1	0	1	1
VS_UP	0/1	0	1	1	0	0	**P12**	0	0	1	1	0	0
VS_DOWN	0/1	0	1	1	0	1	**P13**	0	0	1	1	0	1
VD1_ UP	0/1	0	1	1	1	0	**P14**	0	0	1	1	1	0
VD1_DOWN	0/1	0	1	1	1	1	**Pta_Close**	0/1	0	1	1	1	1
VD2_UP	0/1	1	0	0	0	0	**VS_Close**	0/1	1	0	0	0	0
VD2_DOWN	0/1	1	0	0	0	1	**Vs_Open**	0/1	1	0	0	0	1
P_UP	0/1	1	0	0	1	0	**VD1_close**	0/1	1	0	0	1	0
P_DOWN	0/1	1	0	0	1	1	**VD1_Open**	0/1	1	0	0	1	1
AL_ON	0/1	1	0	1	0	0	**VD2_Close**	0/1	1	0	1	0	0
L11 Y L12	0/1	1	0	1	0	1	**VD2_Open**	0/1	1	0	1	0	1
VALVULAS	0/1	1	0	1	1	0	**P_Close**	0/1	1	0	1	1	0
ASPERSOR	0/1	1	0	1	1	1	**P_Open**	0/1	1	0	1	1	1

Tabla 1.1. Códigos para los actuadores y sensores en el simulador

Desarrollo:

A continuación se presentan algunos de los esquemas Grafcet a los cuales debe responder el automatismo, los Grafcet que hagan falta son una variante de los aquí presentados. Se debe de recalcar que los esquemas no son una concepción de programas independientes, si no que, forman un solo Grafcet que no fue colocado en su conjunto por ser muy grande, y además la manera en que se presentan a continuación es mas didáctica. A continuación se empieza mostrando el grafcet de macro etapas:

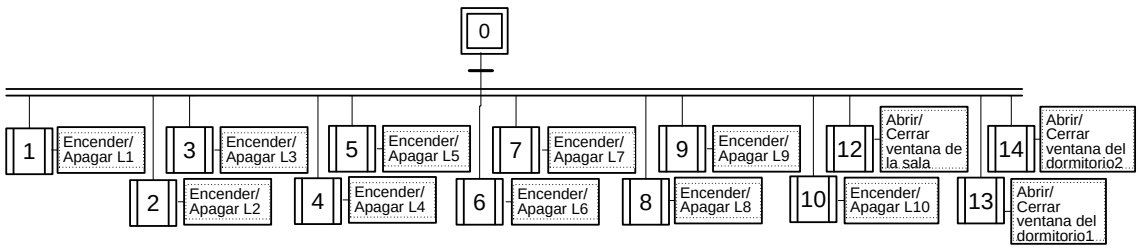


Fig. 1.2. Grafcet completo del automatismo en macro etapas.

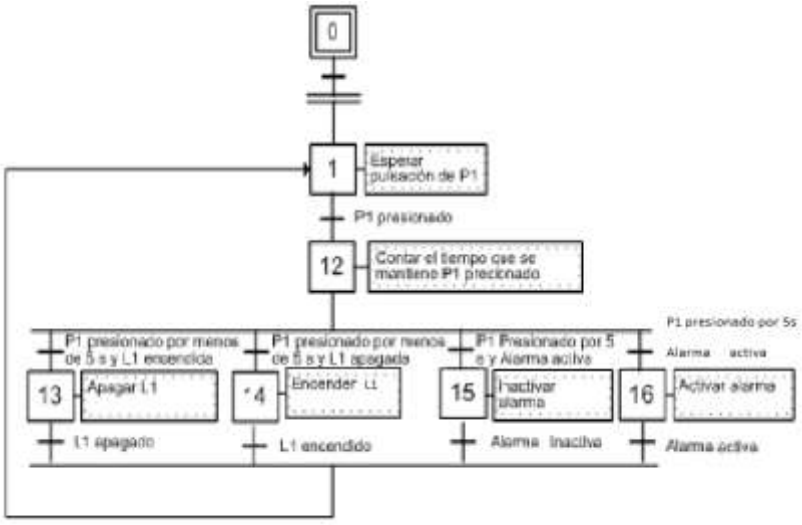


Fig. 1.3. Grafcet para ON/OFF L1 y Activar/Desactivar Alarma.

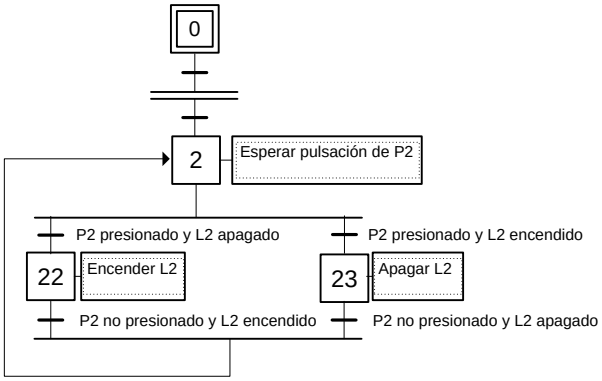


Fig. 1.4. Grafcet que se puede utilizar para Encender/Apagar L2, L3, L6, L8, L9, L10.

Nota: Solo debe modificarse cambiando el nombre de las etapas y usando el pulsador correspondiente en cada caso.

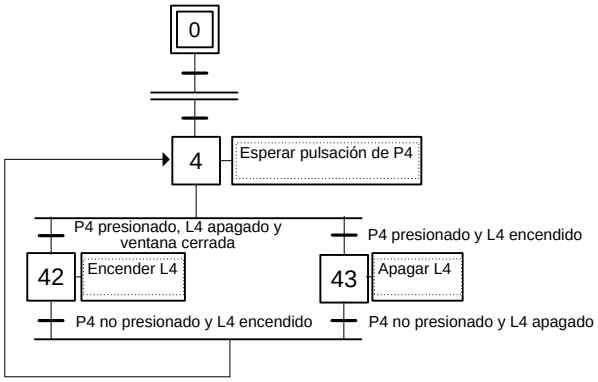


Fig. 1.5. Grafcet que se puede utilizar para Encender/Apagar L4, L5.

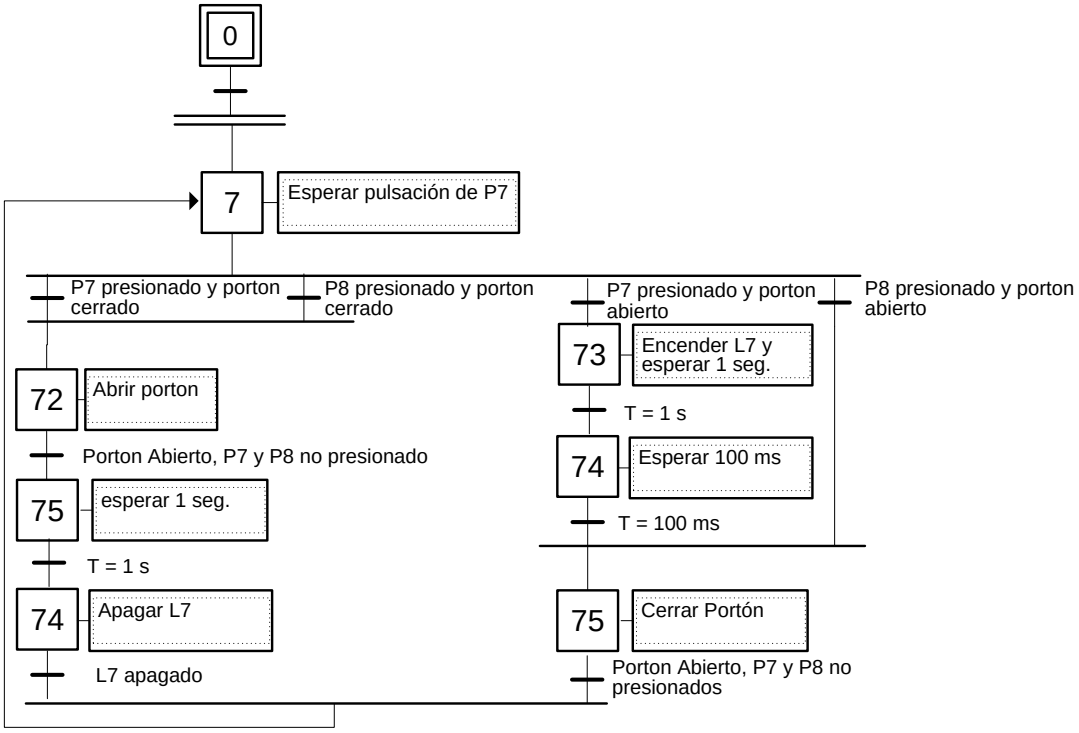


Fig. 1.6. Grafcet para Encender/apagar L7 y abrir/cerrar el portón.

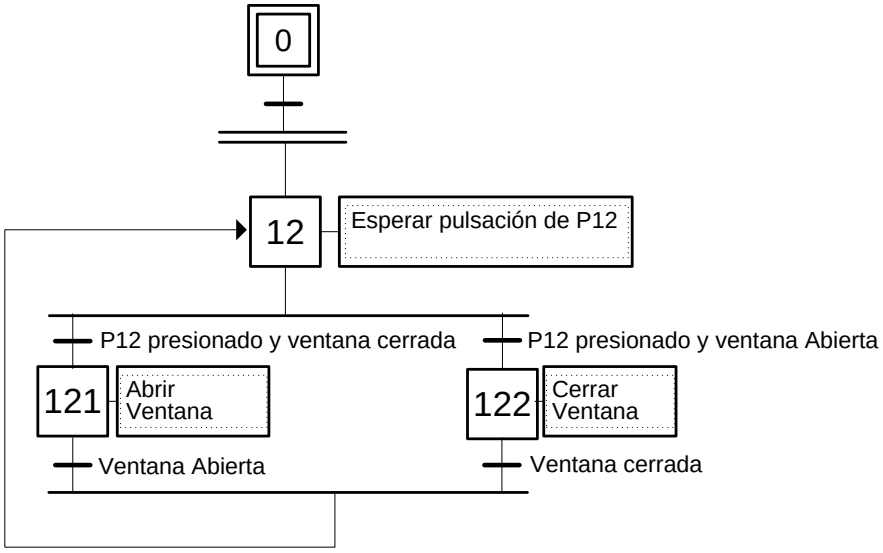


Fig. 1.7. Grafcet que se puede utilizar para abrir/cerrar ventana de la sala y cada uno de los dormitorios.

Descripción de las etapas:

ETAPAS	FUNCION
0	Etapa de inicio.
1	Etapa transitoria de entrada y de retorno, al proceso para Encender/apagar alarma y L1.
12	Se activa si P1 se presiona. Inicia el timer que cuenta el tiempo que se mantiene presionado P1.
13	Se entra a esta etapa si P1 no está presionado, L1 está apagada, T0 no se activo y la ventana de la sala está cerrada (VS_Close ON). Esta etapa activa L7.
14	Si se deja de presionar P1, la luz está encendida y el temporizador T0 no se activó, se entra en esta etapa y se apaga la luz L1.
15	Si P1 sigue presionado, el temporizador T0 se activa y la alarma está encendida, se entra en esta etapa y se apaga la alarma.
16	Si P1 sigue presionado, el temporizador T0 se activa y la alarma está apagada, se entra en esta etapa y se enciende la alarma.
2, 3, 6, 8, 9, 10	Etapas de entrada y de retorno, después del proceso de Encender/Apagar alguna de las siguientes luces L2, L3, L6, L8, L9, L10.
22, 32, 62, 82, 92, 102	Esta etapa enciende la luz correspondiente si su respectivo pulsador fue presionado.
23, 33, 63, 83, 93, 103	Esta etapa apaga la luz correspondiente si su respectivo pulsador fue presionado.
4, 5	Etapas de entrada y de retorno, después del proceso de Encender/Apagar alguna de las siguientes luces L4 y L5.
42, 52	Esta etapa enciende la luz correspondiente si su respectivo pulsador fue presionado y además si la ventana del cuarto está cerrada.
43, 53	Esta etapa apaga la luz correspondiente si su respectivo pulsador fue presionado.
7	Etapa de entrada y retorno del proceso de Encender/apagar L7 y Abrir/Cerrar el portón.
71	Se entra en esta etapa si P7 ó P8 fue presionado y además el portón está cerrado; la cual, Activa la salida para abrir el portón (P_UP).
72	Activa L7, si P7 fue presionado al momento de estar abierto el portón.
73	Activa la salida para cerrar el portón (P_DOWN).
74	Se entra en esta etapa si el portón esta abriéndose y se activa el sensor de portón abierto, en esta la salida de abrir portón se des energiza.
75	Desactiva L7
76	Se entra en esta etapa si el portón esta cerrándose y se activa el sensor de portón cerrado, en esta, la salida de cerrar portón se des energiza.

Tabla 1.2. Funciones realizadas por cada etapa en práctica de domótica.

Importante:

Para esta práctica es preciso crear un decodificador de entradas de 5 líneas a 23 marcas, como se muestra en el siguiente ejemplo, en este solo se han decodificado las señales provenientes de los primeros 4 pulsadores, se debe de recordar que los finales de carrera utilizan una línea más que indican el estado del mismo:

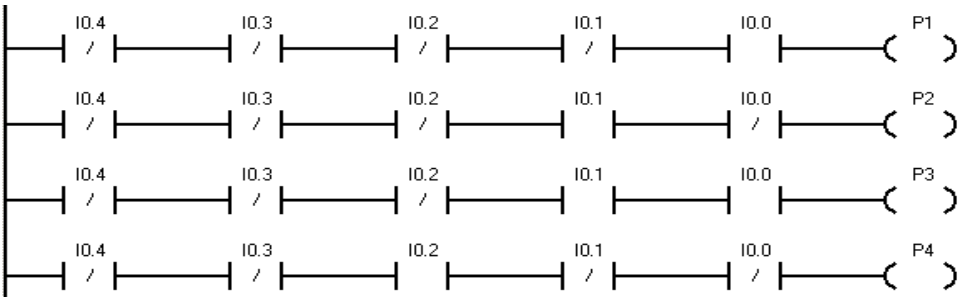


Fig. 1.8. Ejemplo de decodificación de 5 líneas para los primeros 4 pulsadores.

De igual forma se deben de codificar las salidas en código binario natural para poder controlar los actuadores de forma independiente, en esta parte se debe de tener en cuenta que la salida solo estará activa por un periodo corto de tiempo y que se debe incluir una línea más para indicar si se quiere apagar (0) ó encender (1) el dispositivo.

Con esto nos aseguramos de trabajar nuestra lógica de programa como si dispusiéramos de 23 entradas y 23 salidas en el PLC. Cuando se está creando una salida virtual (que nosotros consideraremos como una salida física) esta debe de ser momentánea y será considerada como el estado siguiente de la salida, deberá haber un registro que almacene el estado actual asociado a la salida y este será permanente; cambiará hasta que haya un nuevo estado siguiente. A continuación se muestra un ejemplo:

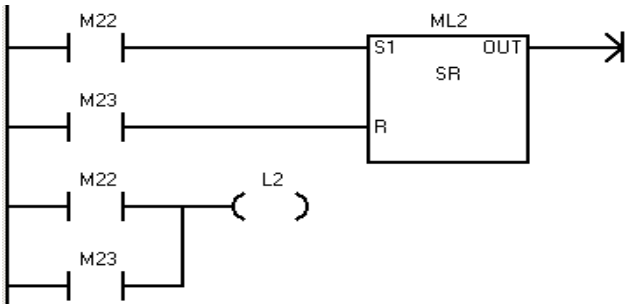


Fig. 1.9. Ejemplo de cómo escribir las salidas en el diagrama de escalera para esta práctica.

En este ejemplo ML2 será el estado actual y todos los contactos auxiliares correspondientes (en este caso) a L2 serán asociados a este registro. L2 será la marca utilizada en el bloque codificador de salidas ya que su activación es de carácter momentáneo.

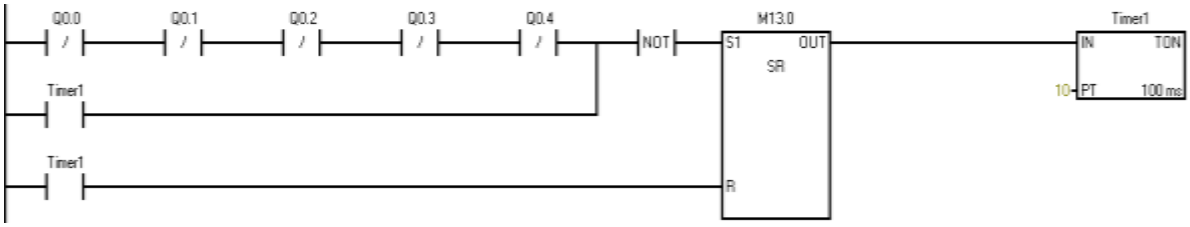


Fig. 1.10. Mantener las salidas activas durante 1 segundo.

Desarrollo de la práctica:

- Elaborar el Grafcet de segundo nivel completo del automatismo
- Obtener el correspondiente diagrama de escalera para el Grafcet presentado.
- Elaborar en Micro/Win el respectivo programa en KOP para esta práctica.
- Realizar las conexiones en el modulo entrenador como se muestra a continuación

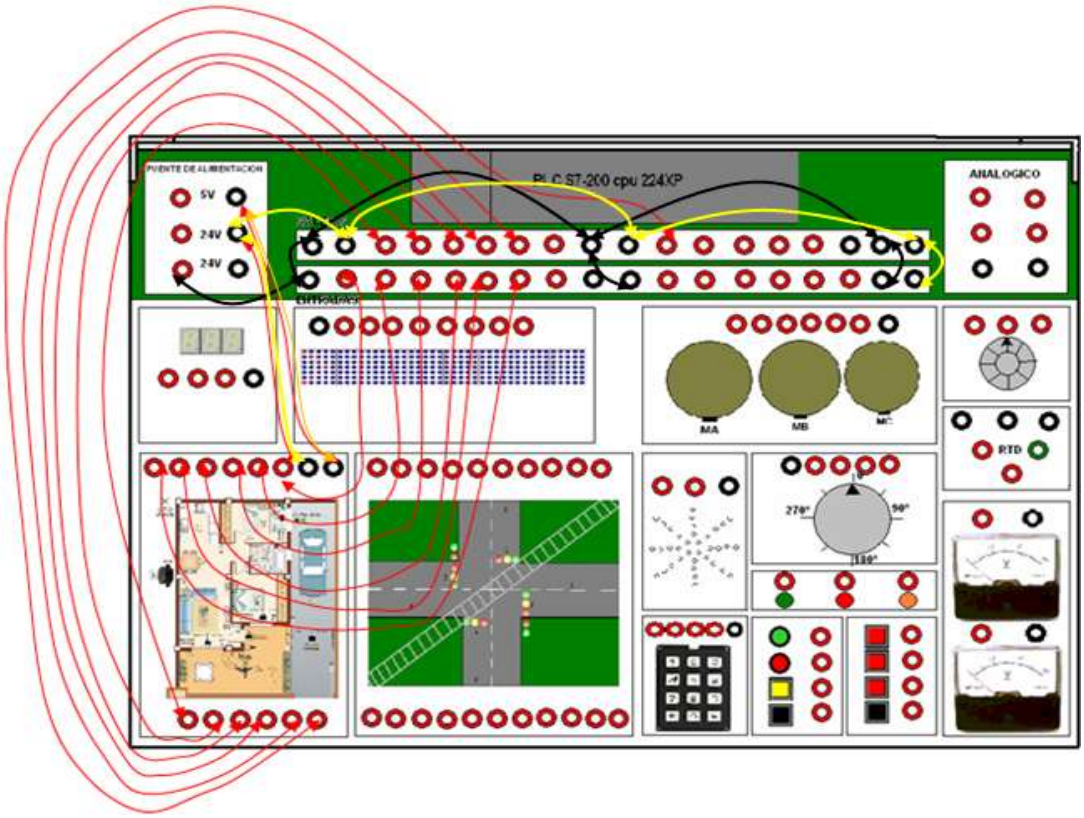


Fig. 1.14. Conexiones a realizar en el Trainer para la práctica de Domotica.

- Asegurarse que los interruptores on-off-on están en la posición OFF (su posición de en medio) y que el interruptor de la puerta este hacia la izquierda (OFF).
- Conectar el cable PC/PPI al S7-200.
- Energizar el modulo. El led Bi-color de alarma activa encenderá color verde.
- Programar el S7-200.
- Realizar las pruebas respectivas.
- Escriba sus conclusiones y/o observaciones

Actividad:

Si se mantiene presionado P1 durante 3 segundos se apagan todas las luces, se cierra el portón y las persianas si están abiertos, después de esto se espera a que se cierre la puerta principal para activar la alarma. Con la alarma activa si se abre el portón ó las persianas la bocina sonará inmediatamente, si se abre la puerta tiene 10 segundos antes de que esta suene. Para desactivar la alarma debe mantener presionado 3 segundos P1.

1.1.2 CONTROL DE SEMAFORO VIAL Y PEATONAL EN INTERSECCION.

Objetivos:

- Aprender lo básico del programa Micro/Win para poder realizar ediciones de programas en KOP y programar el PLC.

- Comprender y utilizar las más importantes funciones de temporización con las que cuenta el PLC S7-200.

Materiales a utilizar:

- 1 PC
- 1 Cable PC/PPI
- 1 S7-200
- 1 maqueta simuladora de semáforos en cruz calle.
- 31 cables de conexión

Planteamiento:

Se controlaran 4 semáforos viales que regulan el tráfico vehicular de una intersección, las vías de este a oeste cuentan con 5 sensores cada una que determinan el nivel de congestionamiento de las vías.

El funcionamiento es el siguiente. En modo normal la vía de sur a norte y de norte a sur tendrá un tiempo de paso de 15 segundos, y la vía de este a oeste y viceversa tendrán un tiempo de 5 segundos. Si los 5 sensores de alguna de las dos vías son activados, los tiempos de los semáforos de este a oeste y viceversa se modificarán y serán de 15 segundos. Si después de transcurrido el tiempo modificado de 15 segundos siguen los 5 sensores activos de cualquiera de la dos vías, se debe aumentar el tiempo de la vía este a oeste y viceversa a 20 segundos y se mantendrá así hasta que la vía se despeje.

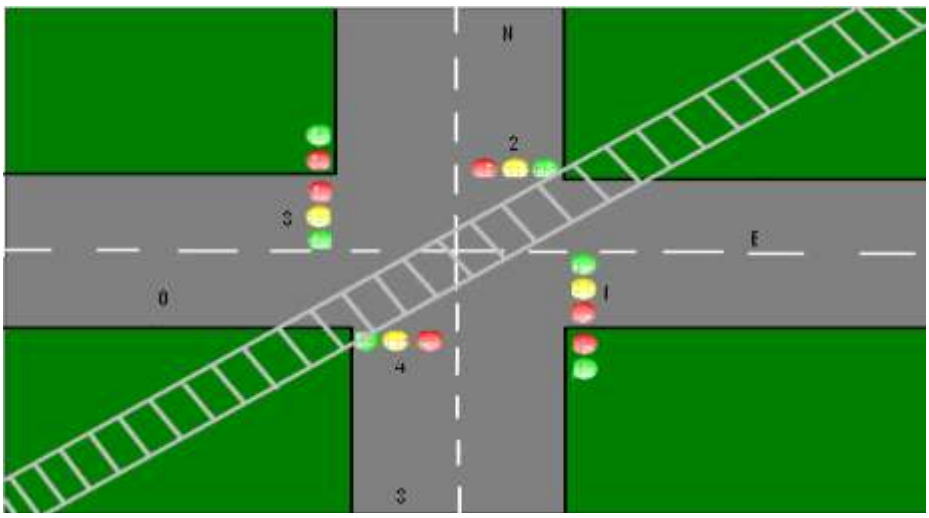


Fig. 1.12. Esquema para el control de semáforo vial en cruz calle.

Desarrollo:

El automatismo debe responder al siguiente esquema Grafcet.

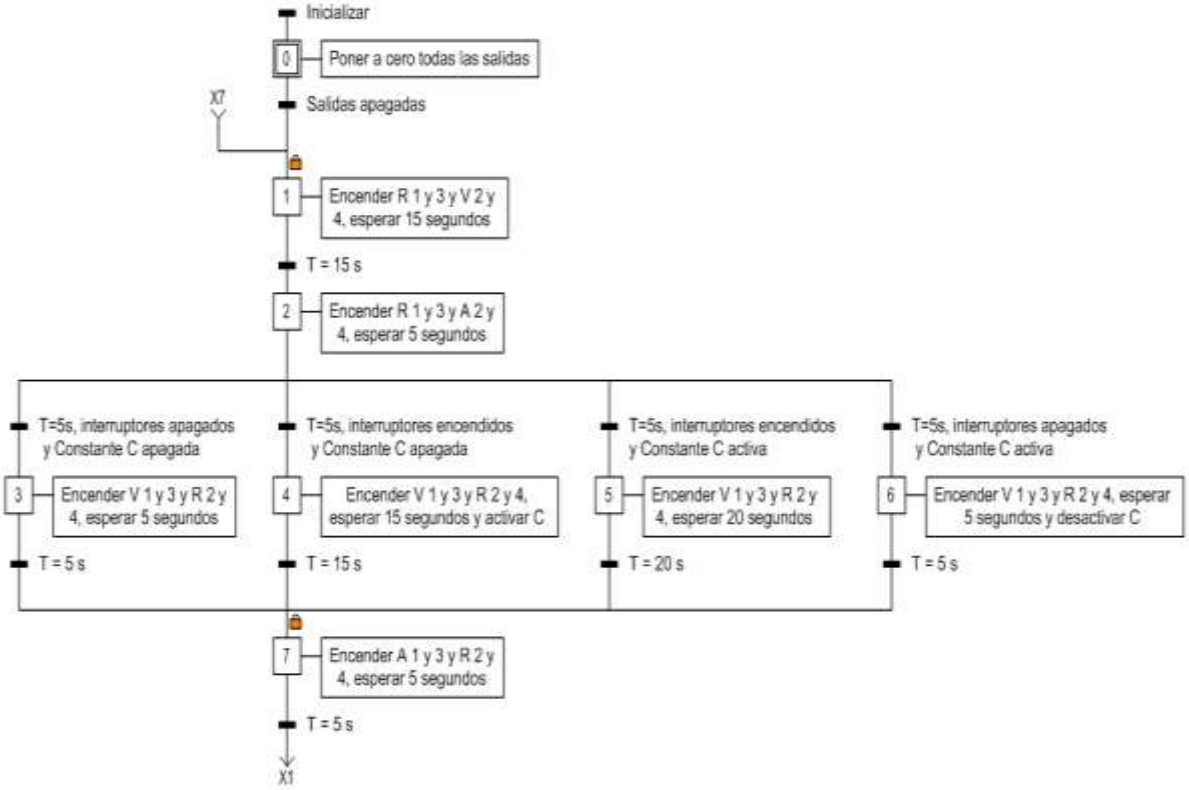


Fig. 1.13. Grafcet para el control de semáforos en cruz calle.

Descripción de las etapas:

ETAPA	*FUNCION*
0	Inicialización de todas las salidas.
1	Activación de Luz verde semáforo 2-4, luz roja semáforo 1-3, temporizador 1.
2	Activación de Luz amarilla semáforo 2-4, luz roja semáforo 1-3, temporizador 2.
3	Activación de Luz verde semáforo 1-3, luz roja semáforo 2-4, temporizador 3.
4	Activación de Luz verde semáforo 1-3, luz roja semáforo 2-4, temporizador 4, activación de C.
5	Activación de Luz verde semáforo 1-3, luz roja semáforo 2-4, temporizador 5.
6	Activación de Luz verde semáforo 1-3, luz roja semáforo 2-4, temporizador 3 y desactivación de C.
7	Activación de Luz amarilla semáforo 1-3, luz roja semáforo 2-4, temporizador 2.

Tabla 1.3. Funciones realizadas por cada etapa en práctica de semáforo.

Descripción de temporizadores:

Temporizador 1: **15 s**
Temporizador 2: **5 s**
Temporizador 3: **5 s**
Temporizador 4: **15 s**
Temporizador 5: **20 s**

Desarrollo de la práctica:

- Obtener el correspondiente diagrama de escalera para el Grafcet presentado.
- Elaborar en Micro/Win el respectivo programa en KOP para esta práctica.
- Realizar las conexiones en el modulo entrenador como se muestra a continuación:

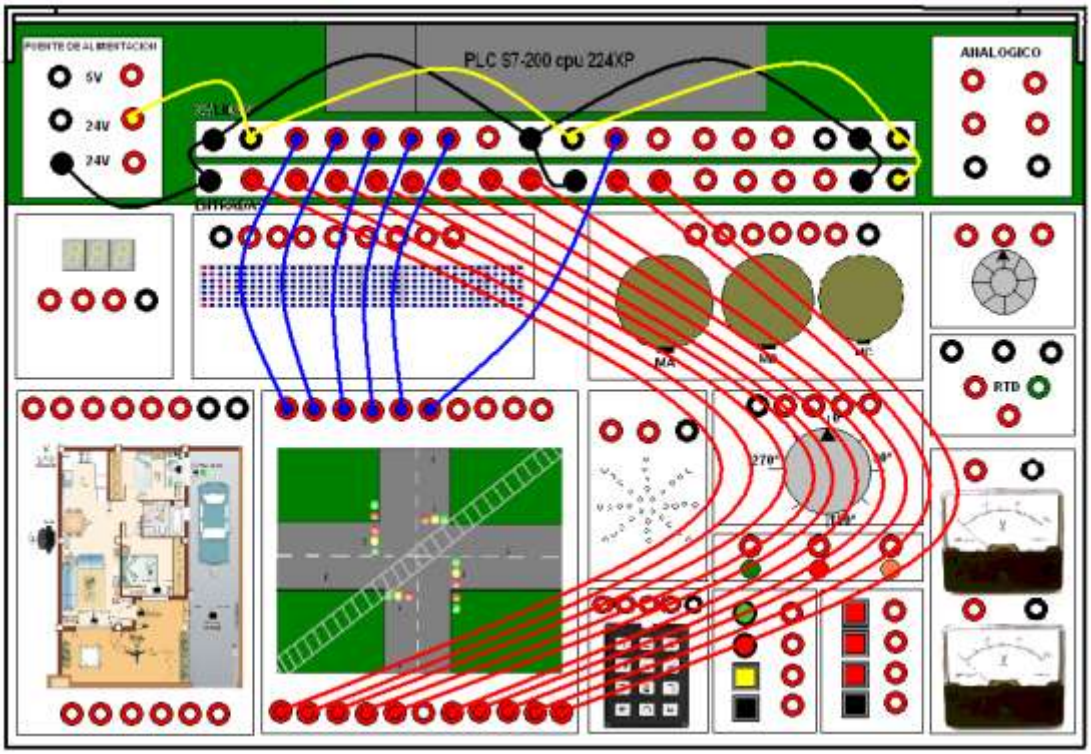


Fig. 1.14. Conexiones a realizar en el Trainer para la práctica de Semáforo.

- Conectar el cable PC/PPI al S7-200 y Programarlo.
- Escriba sus conclusiones y/o observaciones

Actividad:

Modificar el programa para que, al activar el interruptor que simula el cruce de un tren en la vía férrea, los 4 semáforos comiencen a parpadear cambiando de amarillo a rojo continuamente con un tiempo de 500ms cada luz, se volverá al estado normal al deshabilitar el interruptor de paso de tren; además, se debe de agregar el funcionamiento de los semáforos peatonales, los cuales se colocan en verde 1 segundo después de que se activa la luz roja en la vía correspondiente y se colocan en rojo 1 segundo antes de que se ponga el semáforo en verde de la vía correspondiente.

1.1.3 MAQUINA EXPENDEDORA DE LATAS DE SODA.

Objetivo:

- Aplicar los conocimientos de contadores en los PLC S7-200.

- Utilizar temporizadores para la generación de señales PTO ó PWM de baja frecuencia.

Materiales a utilizar:

- 1 PC
- 1 Cable PC/PPI
- 1 S7-200
- 3 motores DC
- 3 sensores ópticos
- 5 pulsadores
- 1 buzzer
- 22 cables de conexión

Planteamiento:

Esta máquina cuenta con 5 pulsadores, 3 motores DC, 3 sensores ópticos, 1 lámpara y 1 buzzer. Cuando se pulsa el botón de introducir moneda la lámpara se enciende, en ese momento se debe de presionar la bebida deseada (A, B ó C) y entonces la lámpara empezara a parpadear a razón de 1 destello por segundo y el motor correspondiente a la bebida seleccionada empezara a girar; cuando el motor haya dado 3 vueltas completas debe detenerse, la lámpara dejara de parpadear quedando encendida, se activara el buzzer emitiendo un sonido durante 1 segundo luego se apagará todo. Al haber sacado 7 sodas del mismo sabor el sistema no funcionara hasta que se presione el botón de Rearme.

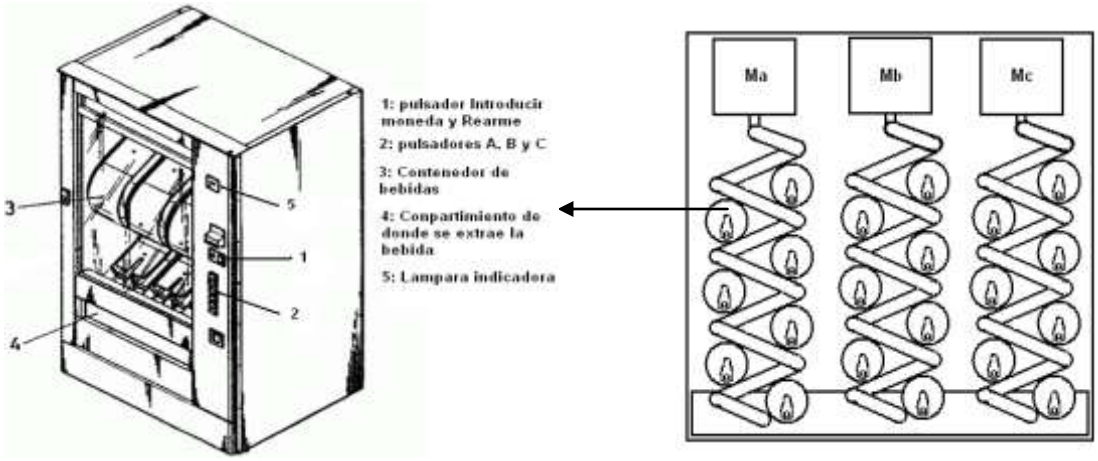


Fig. 1.15. Máquina Expendedora de sodas.

Desarrollo:

El automatismo debe responder al siguiente esquema Grafcet:

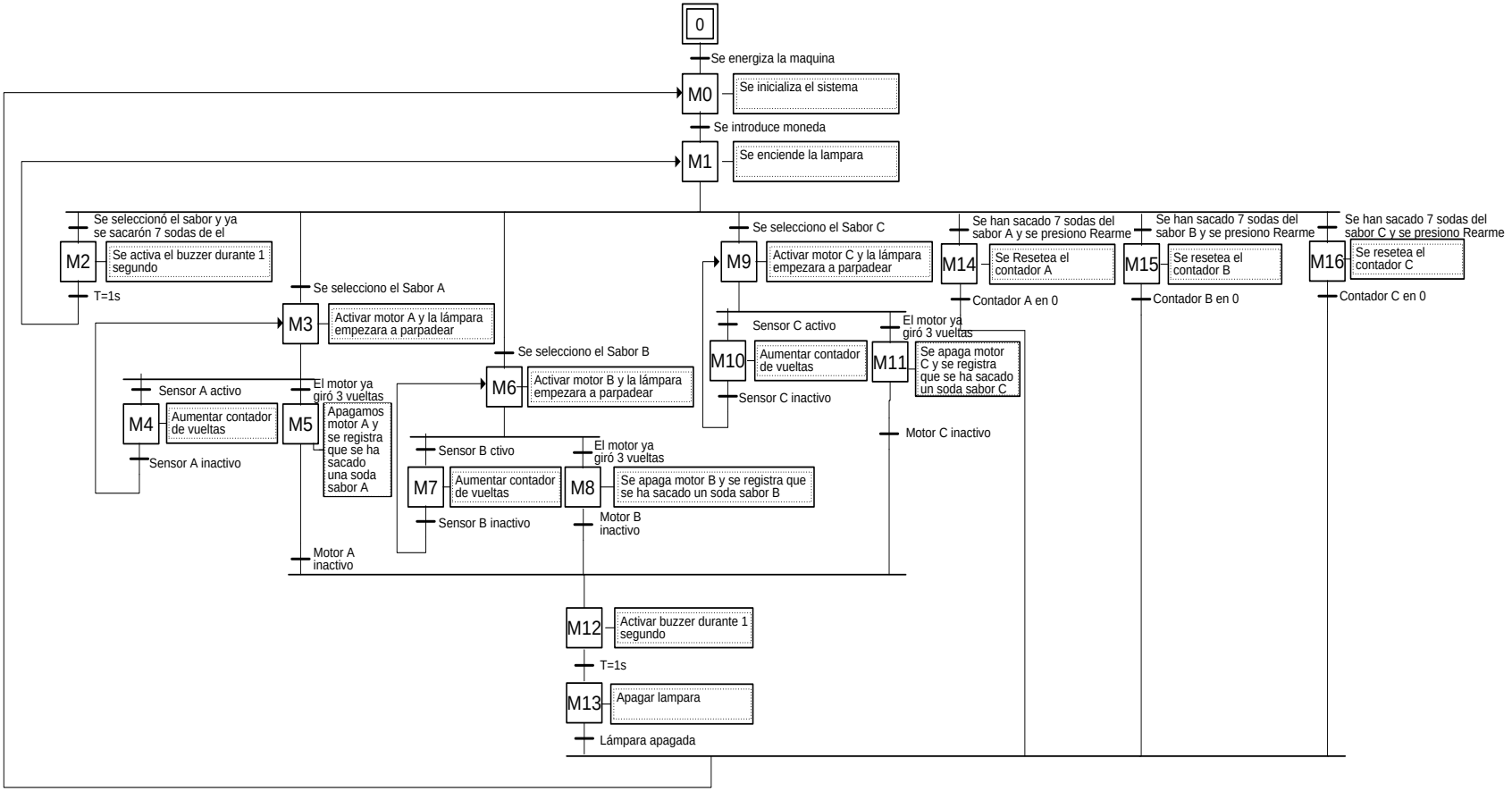


Fig. 1.16. Grafcet para la máquina expendedora de sodas.

Descripción de las etapas:

ETAPAS	FUNCION
0	Maquina energizada, todos los elementos apagados.
1	Lámpara Encendida, Buzzer apagado y Timer 1 OFF
2	Se mantiene encendido el Buzzer y el Timer 1 durante 1 segundo.
3, 6, 9	Activar motor. Ma, Mb ó Mc ON, Lámpara intermitente y contador de vueltas activo.
4, 7, 10	Incrementar contador de vueltas.
5, 8, 11	Incrementar contador de sodas y desactivar motor.
12	Activar buzzer durante 1 segundo. Timer 1 ON.
13	Desactivar Lámpara y Buzzer.
14,15,16	Reiniciar contador de sodas.

Tabla 1.4. Funciones realizadas por cada etapa en práctica de máquina de sodas.

Importante:

A continuación se presenta la forma de generar una señal PWM ó PTO con temporizadores, la máxima frecuencia que se puede generar por este método es de 500Hz debido a que 1ms es el minimo tiempo de retardo para los temporizadores, esto hace al método poco aplicable en el control de motores DC o otras aplicaciones similares en la industria. Los PLC Siemens de las series S7-200 y posteriores ya cuentan con rutinas dedicadas a generación de señales PWM y PTO con las cuales se pueden enviar señales hasta de 20KHz por alguna de las entradas Q0.0 ó Q0.1, estas rutinas se verán posteriormente cuando se estudien las funciones de rango avanzado en el PLC.

En la siguiente figura se presenta un ejemplo que activa y desactiva una lámpara cada segundo.

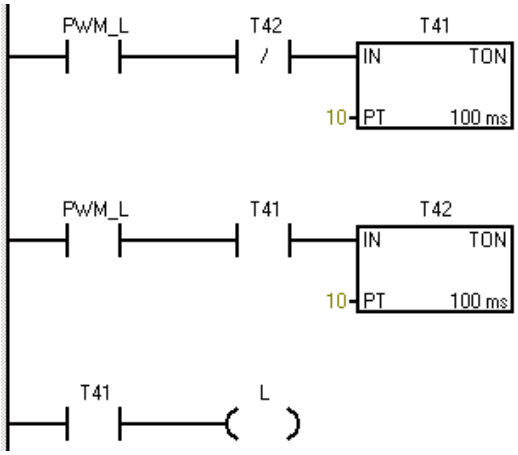


Fig. 1.17. Diagrama en KOP para la implementación de una señal PTO con temporizadores.

Desarrollo de la práctica:

- Obtener el correspondiente diagrama de escalera para el Grafcet presentado.
- Elaborar en Micro/Win el respectivo programa en KOP para esta práctica.
- Realizar las conexiones en el modulo entrenador como se muestra a continuación.

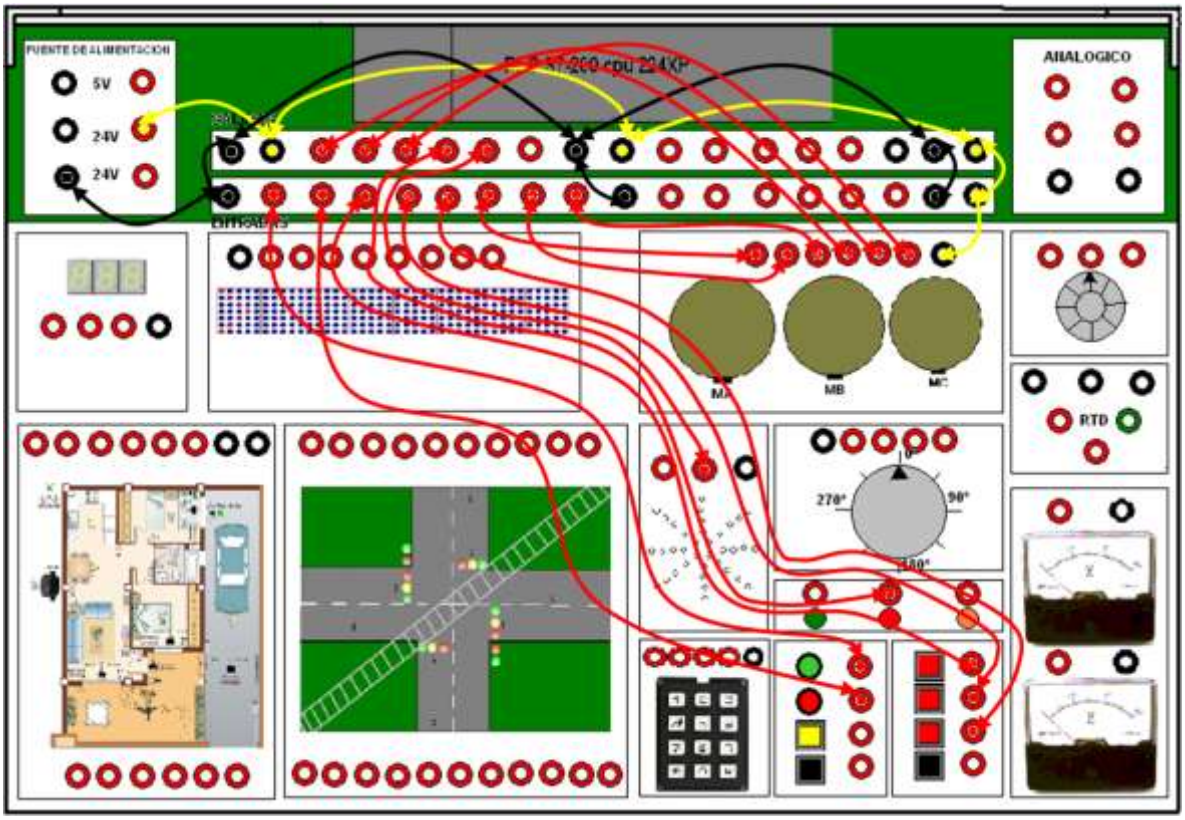


Fig. 1.18. Conexiones a realizar en el Trainer para la práctica de Maquina de Sodas.

- Conectar el cable PC/PPI al S7-200 y Programarlo.
- Escriba sus conclusiones y/o observaciones

Actividad:

- Agregar el estado GEMMA de “Marcha de preparación”, si el motor no está en la posición correcta se gira a una velocidad más lenta de lo normal hasta llegar a la posición adecuada para establecer que definitivamente se contarán 30 vueltas.

- Agregar los estados GEMMA necesarios para retornar de un “paro de emergencia” en el cual se entra si el motor está girando y de pronto durante 5 segundos no se detectan pulsos de conteo. El sistema se detiene, la lámpara deja de parpadear quedando encendida y el sistema no funciona nuevamente hasta que se presione el pulsador de rearme. El sistema empezara en la posición que se quedo antes que ocurriera la emergencia.

1.2 PRACTICAS DE FUNCIONES DE RANGO MEDIO

1.2.1 CHAPA ELÉCTRICA CON CONTRASEÑA

Objetivos:

- Introducir al estudiante a la utilización de las funciones de rango medio de los PLC.
- Aplicar los conocimientos de funciones aritméticas y de comparación en los PLC S7-200.

Materiales a utilizar:

- 1 PC
- 1 Cable PC/PPI
- 1 S7-200.
- 1 Teclado matricial.
- 2 Lámparas.
- 1 buzzer.
- 20 cables de conexión.

Planteamiento:

Este sistema contara con un teclado, 2 lámparas y un buzzer (modulo de alarma de dos tonos), con estos se realizara el control de una chapa eléctrica la cual debe ser abierta mediante una contraseña de cuatro dígitos. La contraseña debe ser fija y se almacenara en un registro de 32 bits (VD0), el teclado codifica sus teclas en binario de acuerdo a una tabla establecida, por lo cual cada tecla pulsada debe almacenarse en un registro de 8 bits. El sistema debe iniciar con una lámpara roja encendida, lo que indica que la chapa está cerrada, al introducir la contraseña correcta se apagara la lámpara roja y se encenderá una lámpara verde que indica que la chapa está abierta, si se introduce una contraseña errónea el sistema debe volver a su modo normal con la lámpara roja encendida para que la contraseña se introduzca nuevamente, si se introduce tres veces seguidas una contraseña errónea se debe activar el buzzer y este dejara de sonar hasta que la contraseña correcta sea ingresada al sistema.

Numero	0	1	2	3	4	5	6	7	8	9	*	#
Código en binario	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100

Tabla 1.5. Forma en que el teclado codifica sus datos según el número que se presione.

En la tabla anterior se puede observar que el número es mayor en 1 a la tabla de binario natural, por lo que el programa diseñado debe restar ese 1 a cada dato que se almacena para que se obtenga el valor real introducido.

Fig. 1.19. Partes del entrenador utilizadas en práctica de chapa con contraseña.

Desarrollo:

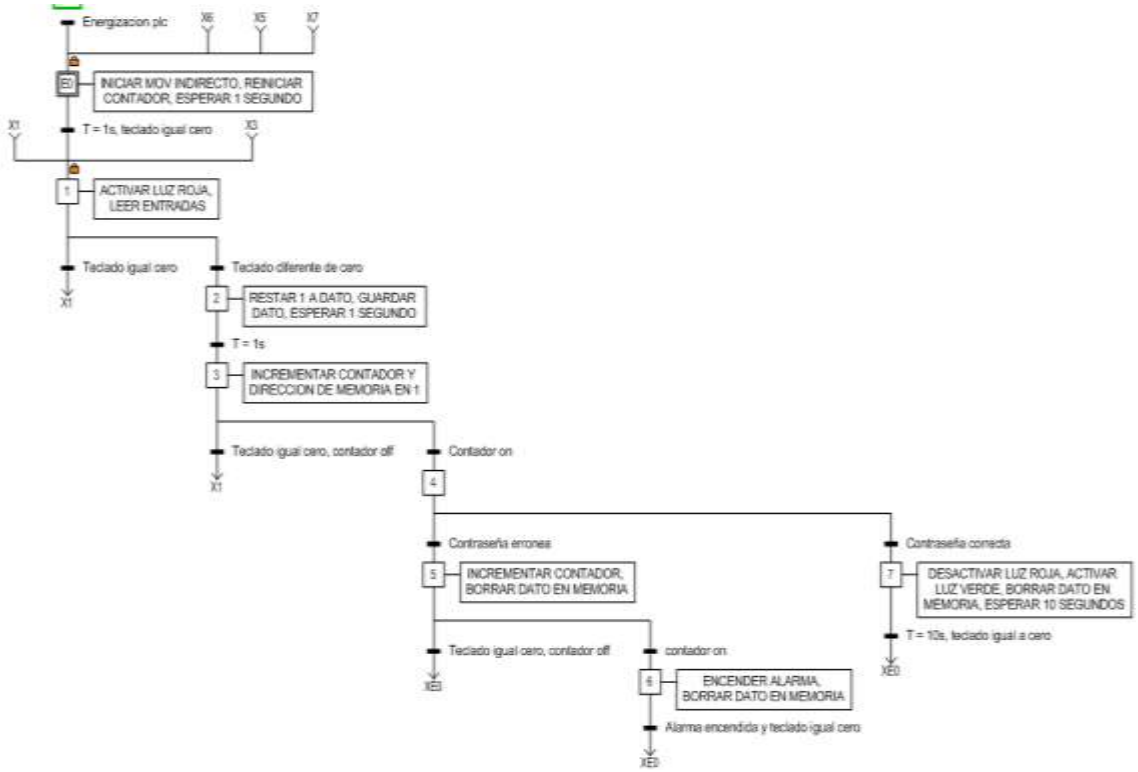


Fig. 1.20. Grafcet de chapa eléctrica con contraseña.

Descripción de las etapas:

ETAPA	FUNCION
0	Apunta LD10 (apuntador) a variable VB4, restablece Contador (C0) y activa un temporizador de 1 segundo.
1	Activa lámpara roja y desactiva la lámpara verde, lee byte IB0 (byte entrada) del PLC.
2	Resta 1 al dato de IB0, lo almacena en VB4 a través de LD10 y activa un temporizador de 1 segundo.
3	Incrementa en 1 LD10(apuntador) y Contador(C0)
4	Sin Acción.
5	Incrementa en 1 Contador de datos introducidos (C1) y pone a cero VD4(variable de doble palabra VB4- VB7)
6	Activa la alarma y pone a cero VD4.
7	Desactiva la lámpara roja y la alarma, activa la lámpara verde, resetea C1 y pone a cero el dato en memoria (VD4 = 0).

Tabla 1.6. Funciones realizadas por cada etapa en práctica de la flecha de desvió vial.

Desarrollo de la práctica:

- Obtener el correspondiente diagrama de escalera para el Grafcet presentado.
- Elaborar en Micro/Win el respectivo programa en KOP para esta práctica.
- Realizar las conexiones en el modulo entrenador como se muestra a continuación.

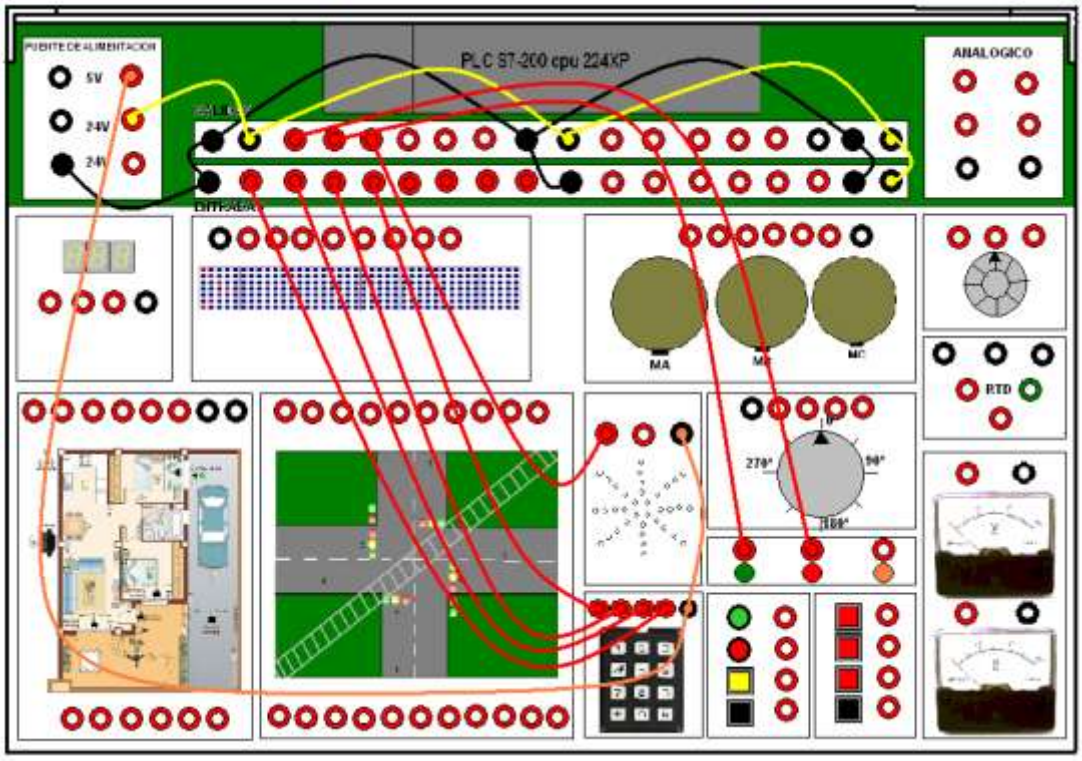


Fig. 1.21. Conexiones a realizar en el trainer para la práctica de la chapa.

- Conectar el cable PC/PPI al S7-200 y Programarlo.
- Escriba sus conclusiones y/o observaciones

Actividad:

Rediseñar el sistema anterior, para que se le pueda modificar la contraseña. Al presionar la tecla ‘ * ’ se debe entrar en el modo “cambio de contraseña”, este se identificara porque la lámpara roja deberá empezar a parpadear, seguidamente se debe ingresar la contraseña anterior, si esta se introdujo correctamente, mientras la lámpara roja sigue parpadeando se activara la lámpara verde la cual indica que la contraseña ingresada es correcta y que se puede introducir la nueva contraseña, si la contraseña original que se introdujo es errónea, la lámpara verde no se enciende y la roja deja de parpadear y se queda encendida lo que indicara que el sistema salió del modo cambio de contraseña y está en el modo normal.

1.2.2 BANDA TRANSPORTADORA DE OBJETOS.

Objetivos:

- Introducir al estudiante a la utilización de las funciones de rango medio de los PLC.

- Aplicar los conocimientos de funciones aritméticas, comparación, saltos y movimiento de datos en los PLC S7-200.

Materiales a utilizar:

- 1 PC
- 1 Cable PC/PPI
- 1 S7-200
- 1 Teclado matricial
- 1 maqueta de 2 bandas
- 16 cables de conexión
- 1 Buzzer
- 2 Lámparas indicadoras

Planteamiento:

El principio de funcionamiento es el siguiente: Se tiene una banda transportadora con un sensor óptico en la salida, su función será contar la cantidad de cajas que han sido transportadas y despachadas. El automatismo debe contar con un interruptor de marcha para iniciar su funcionamiento normal.

Con el teclado no matricial se debe especificar el número de unidades que deben ser transportadas, con un límite de dos dígitos en la cantidad de unidades, a continuación se debe presionar la tecla "#" para iniciar el funcionamiento de la banda o presionar la tecla "*" para anular la cantidad introducida por el teclado y esperar la nueva cantidad, el funcionamiento debe ser indicado por una lámpara, la cual se mantendrá encendida mientras la maquina este operando, hasta llegar al número establecido de unidades.

Se debe notar que las unidades son trasportadas en cajas con 10 unidades cada una, por lo cual se debe realizar la operación aritmética necesaria para poder obtener el número de unidades especificado con el teclado no matricial.

El funcionamiento del teclado se puede ver en la tabla 1.5 en la práctica anterior.
Cuando se detecten la cantidad de cajas necesarias para las unidades requeridas, se debe apagar la luz de funcionamiento y esperar que se vuelvan a introducir datos con el teclado, siendo este el ciclo normal de funcionamiento.

Además, el automatismo debe incluir una luz indicadora de error, cuando el usuario trate de introducir más de dos dígitos, debido a que se restringe a un máximo de 99 unidades; es decir, un máximo de 10 cajas.

Desarrollo:

El sistema debe corresponder al grafcet mostrado a continuación:

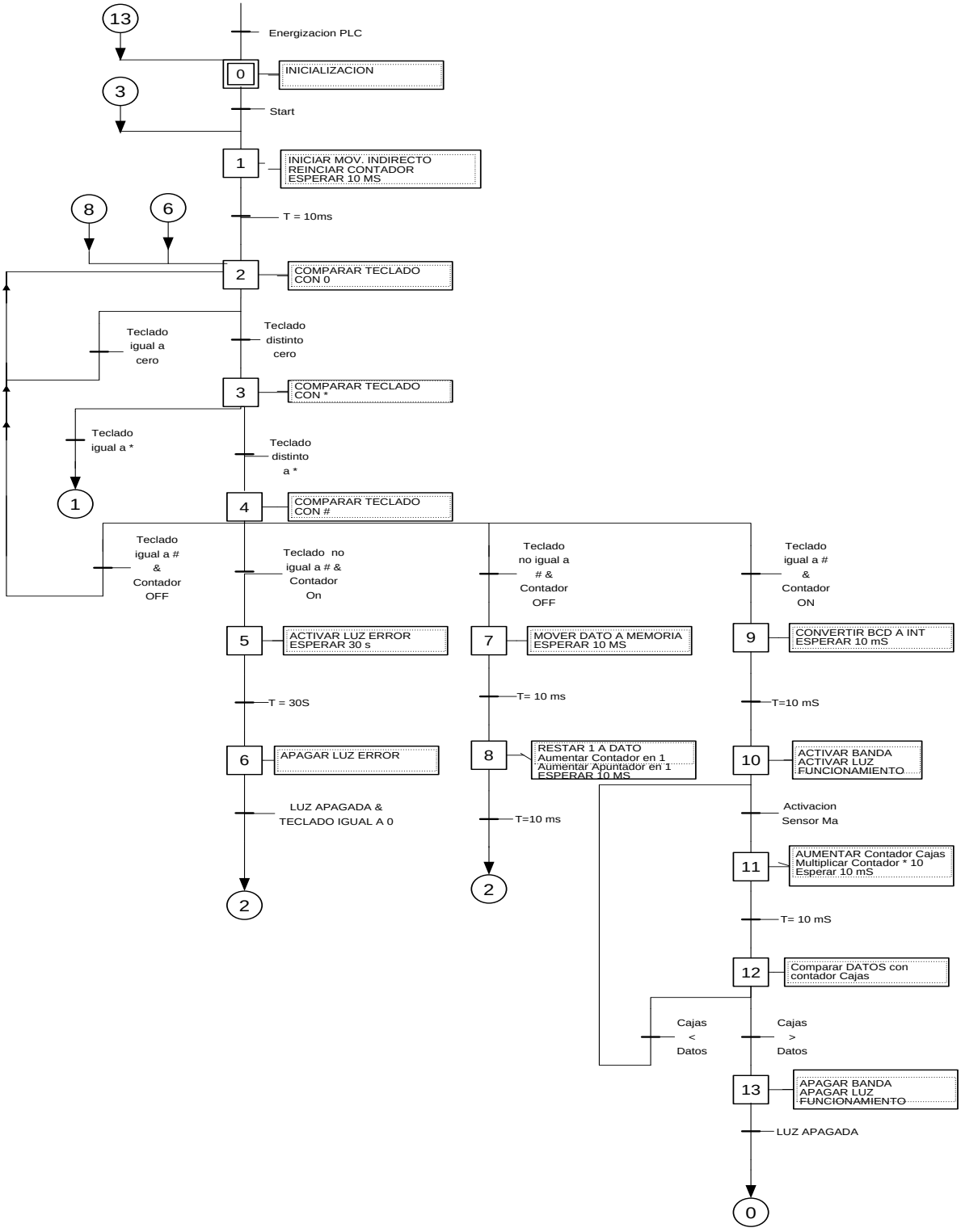


Fig. 1.22. Grafcet de banda transportadora de objetos.

Descripción de las etapas:

ETAPA	FUNCION
0	Inicialización de todas las salidas y sistema.
1	Inicialización movimiento indirecto hacia la posición VB1, activación temporizador
2	Detección de código en bit de entrada 0, inicialización de contador 0.
3	Validación de carácter "* ".
4	Validación de carácter "# ".
5	Activación de luz de error por introducir más de 2 dígitos desde el teclado, temporizador 1.
6	Desactivación de luz de error por introducir más de 2 dígitos desde el teclado.
7	Movimiento de datos a dirección apuntada por LD10, temporizador 1.
8	Resta 1 al dato introducido, Aumenta el contador y se aumenta el apuntador, temporizador 1.
9	Convertir BCD en palabras VW0 – VW2 en entero, temporizador 1.
10	Activación de motor para banda 1 y luz de funcionamiento normal.
11	Activación de contador 1 (# cajas), Multiplicación de numero de cajas * 10, temporizador 1.
12	Operación de comparación entre conteo y valor almacenado en VB3 (entero).
13	Desactivación de motor para banda 1 y luz de funcionamiento normal.

Tabla 1.7. Funciones realizadas por cada etapa en práctica de banda transportadora.

Descripción de temporizadores:

Temporizador 1: T31 → 10 ms.
Contador 0: conteo máximo 2.

Para la conversión del dato en BCD almacenado en MB1 y MB3 se sugiere utilizar las siguientes instrucciones:

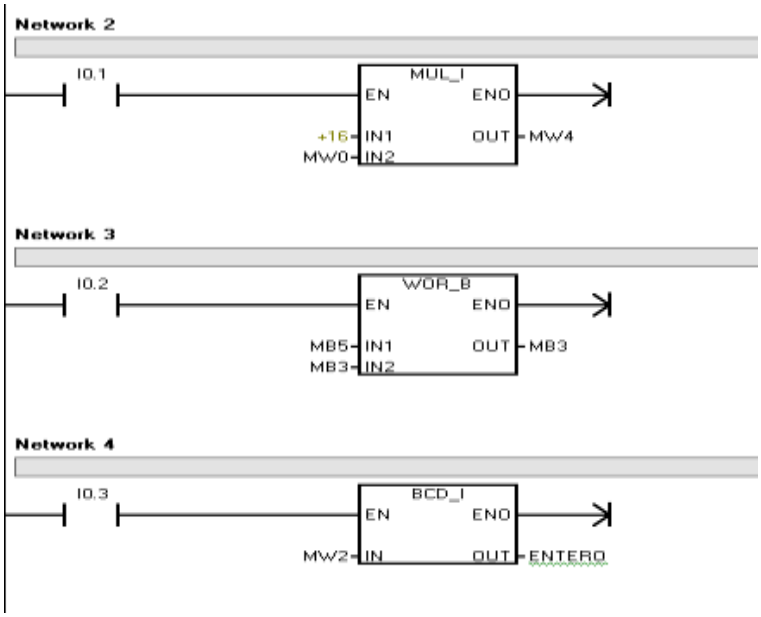


Fig. 1.23. Diagrama en KOP para la conversión de BCD a INT.

En la figura 1.23 se muestra el segmento de código destinado para convertir el BCD almacenado en MB1 y MB3, para lo cual se utiliza una marca intermedia como lo es MB5. Para la multiplicación por 16 (desplazar 4 bits a la izquierda) se utiliza la palabra MW0 (MB0 – MB1) y se almacena en MW4 (MB4-MB5). Luego se unen ambos bytes MB5 y MB3 a través de la función WOR_B. La función BCD_I (BCD a entero) requiere una palabra y la convierte a una INT de longitud de un byte.

Desarrollo de la práctica:

- Obtener el correspondiente diagrama de escalera para el Grafcet presentado.
- Elaborar en Micro/Win el respectivo programa en KOP para esta práctica.
- Realizar las conexiones en el modulo entrenador como se muestra a continuación.

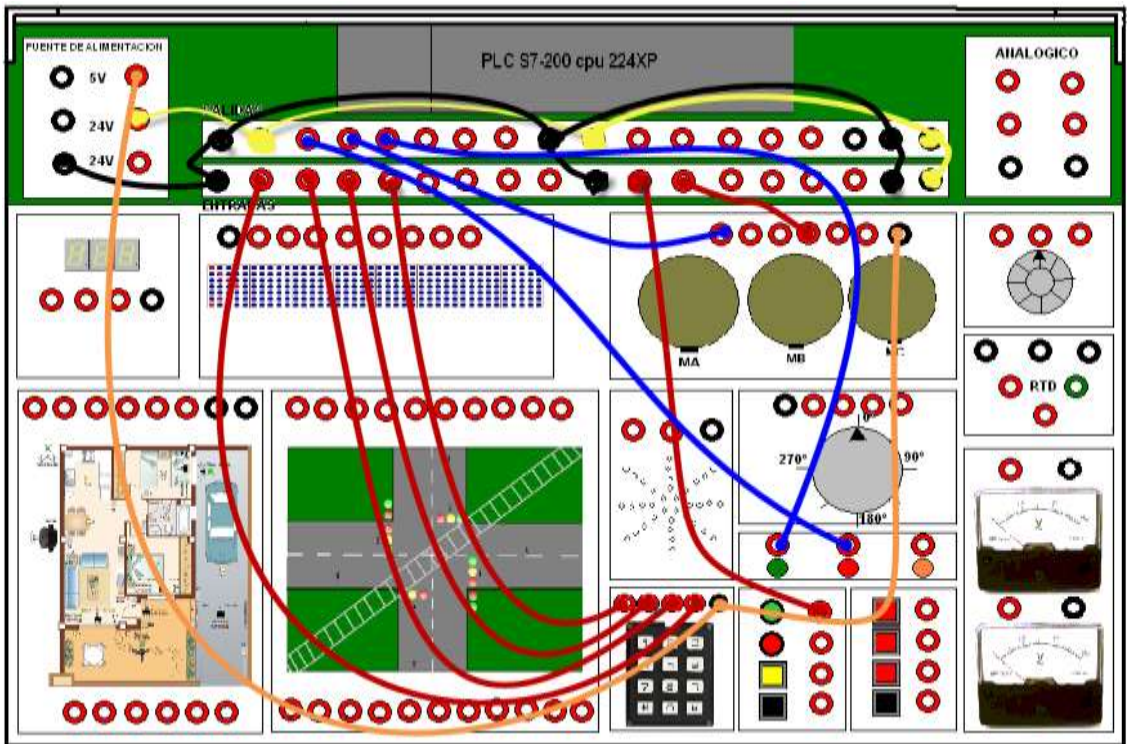


Fig. 1.24. Conexiones a realizar en el trainer para la práctica de la banda transportadora.

- Conectar el cable PC/PPI al S7-200 y Programarlo.
- Escriba sus conclusiones y/o observaciones

Actividad:

- Al sistema anterior debe de agregarse una segunda banda con su respectivo sensor óptico a la salida. La entrada de esta banda estará ubicada a la salida de la anterior.

- El funcionamiento será de la siguiente manera: la primera banda contara los objetos de entrada y la segunda los de salida, cuando la primera banda llegue al número establecido por el teclado numérico se detendrá, de igual manera cuando la segunda banda llegue al número establecido se detendrá.

- Agregar los estados GEMMA necesarios para retornar a la producción normal de un “paro de emergencia” en el cual se entra si después de 15 segundos de haberse detenido la banda 1 no se ha detenido la banda de salida, el sistema debe reportar el error a través de un buzzer activo por 15 s y una lámpara para indicar la emergencia y el sistema no funciona nuevamente hasta que se presione el pulsador de rearme o marcha.

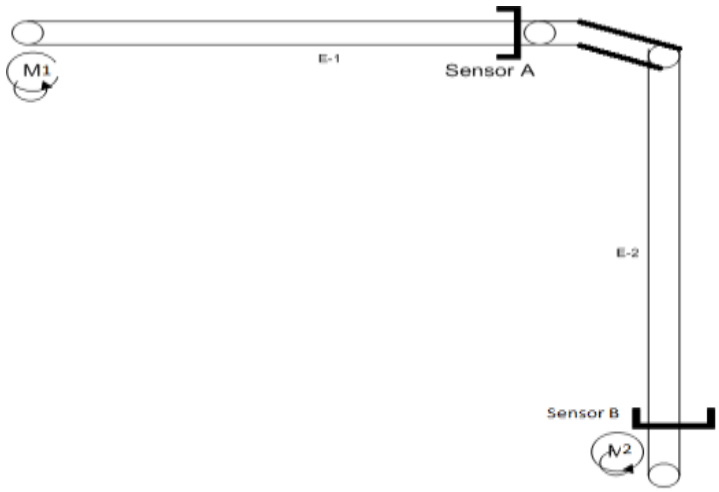


Fig. 1.25. Esquema de Bandas Transportadoras y sus respectivos sensores.

1.2.3 CONTROL DOMOTICO COMPLEJO EN VIVIENDA FAMILIAR.

Objetivos:

- Aprender a estructurar programas con la función SCR (Secuencial Control Relay)
- Conocer las estructuras que relacionan a los SCR con un GRAFCET.
- Conocer y comprender las distintas formas de utilización de la función 'MOV'.
- Poner en práctica las seis distintas funciones de comparación de número.

Materiales a utilizar:

- 1 PC
- 1 Cable PC/PPI
- 1 S7-200
- 1 maqueta simuladora para domótica
- 24 cables de conexión

Planteamiento:

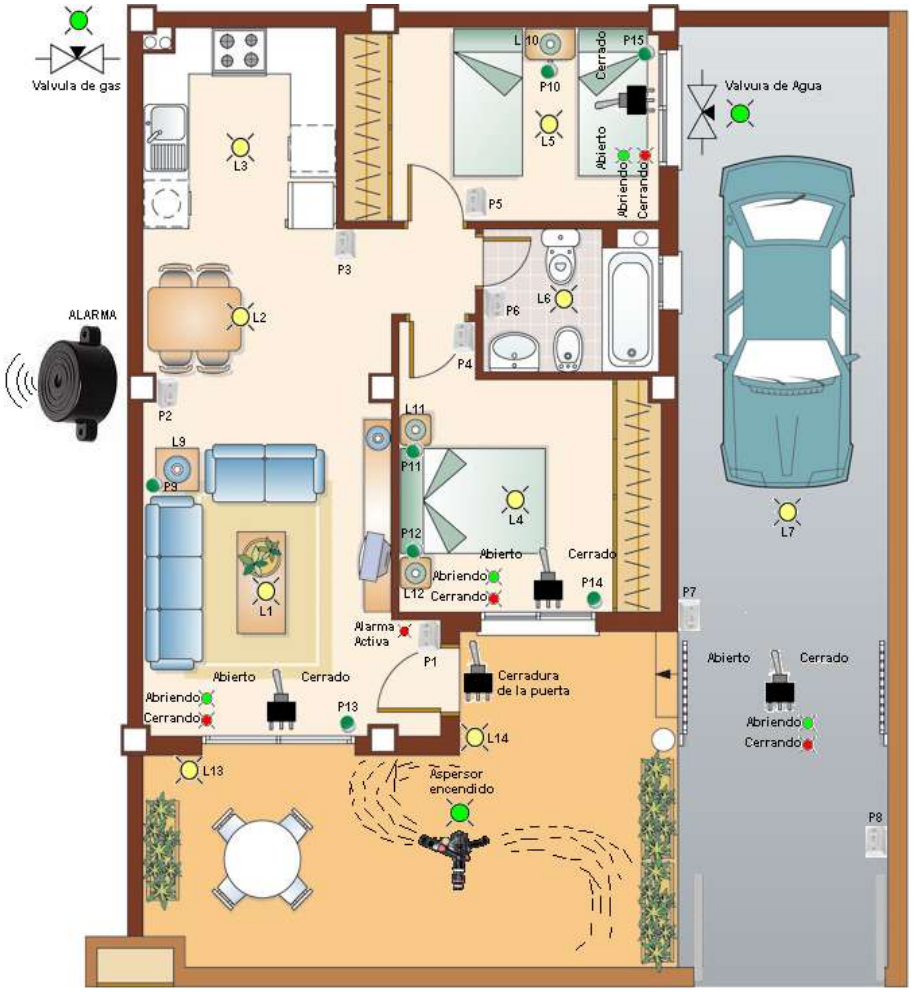


Fig. 1.26. Esquema de la vivienda.

Se debe automatizar una vivienda con el S7-200 la cual cuenta con pulsadores haciendo la vez de interruptores colocados en los sitios mostrados en el esquema. Se tienen 10 luces, tres ventanas una en la sala y una en cada cuarto, estas cuentan con persianas que puede cerrarse o abrirse automáticamente las cuales son operadas independientemente, además cada ventana cuentan con sus respectivos sensores de persianas abiertas y persianas cerradas. Se dispone de un portón en el garaje que es automático el cual al igual que la ventana cuenta con sus respectivos sensores de portón abierto y portón cerrado.

La automatización consiste en ajustar bandas horarias y dar más de una funcionalidad a cada pulsador; con lo cual se pretende limitar el funcionamiento de los dispositivos en su interior disminuyendo así el consumo de energía eléctrica.

El sistema debe funcionar de la siguiente manera:

Al presionar P1	
Nº de pulsaciones	***Acción***
1	Enciende L1
2	Enciende L2
3	Enciende L1 y L2
Al presionar P2	
Nº de pulsaciones	***Acción***
1	Encender L1
2	Encender L2
3	Encender L3
4	Encender L1, L2 y L3
Al presionar P3	
Nº de pulsaciones	***Acción***
1	Enciende L3
2	Enciende L2
3	Enciende L2 y L3
Al presionar P6	
Nº de pulsaciones	***Acción***
1	Enciende L6
Mantener durante 1 segundo	Se enciende durante 1 minuto
Al presionar P4	
Nº de pulsaciones	***Acción***
1	Enciende L4
2	Abre persianas
3	Enciende L10
4	Enciende L11
Al presionar P5	
Nº de pulsaciones	***Acción***
1	Enciende L5
2	Abre persianas

Tabla. 1.8. Resumen de funciones para cada pulsador.

Además, desde las 7:00 A.M. hasta las 5:00 P.M. no se pueden encender las luces L1, L4 y L5. Desde las 5:00 P.M. hasta las 7:00 A.M. se pueden encender todas las luces. El portón es abierto presionando P8 por fuera ó P7 por dentro, de igual manera si se presiona P7 ó P8 nuevamente el portón debe cerrarse. Con el portón abierto no se puede encender L7 ó si está encendida se apaga al abrir el portón y se vuelve a encender al momento de dar la orden para cerrar el portón solo si esta orden fue hecha con el pulsador P7.

Mapa de Memoria sugerido:

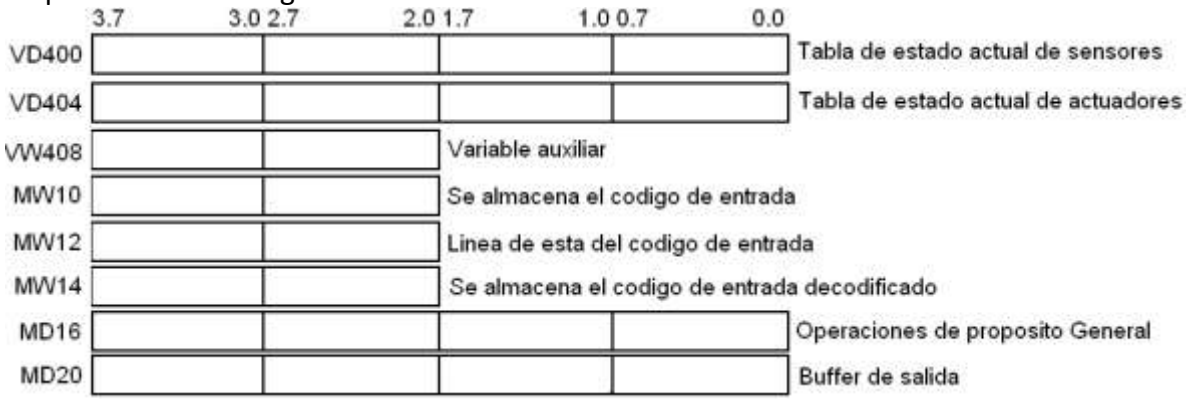


Fig. 1.27. Mapa de memoria.

Para una mayor explicación sobre los mapas de memoria consultar el anexo A sección "Acceso a los datos en memoria ".

Se debe de crear el siguiente bloque de datos en el editor de bloque de datos de MicroWin:

```
//
//Tabla que contiene todos los registros
de //salida
//

VB0   0, 0, 0, 1          //P1
VB4   0, 0, 2, 0
VB8   0, 0, 2, 1
VB12  0, 0, 0, 0
VB16  0, 0, 0, 1          //P2
VB20  0, 0, 2, 0
VB24  0, 3, 0, 0
VB28  0, 3, 2, 1
VB32  0, 3, 0, 0          //P3
VB36  0, 0, 2, 0
VB40  0, 3, 2, 0
VB44  0, 0, 0, 0
VB48  0, 0, 0, 4          //P4
VB52  14, 0, 0, 0
VB56  0, 0, 9, 0
VB60  0, 10, 0, 0
VB64  0, 0, 0, 5          //P5
VB68  16, 0, 0, 0
VB72  0, 0, 0, 0
VB76  0, 0, 0, 0
VB80  0, 0, 0, 6          //P6
VB84  0, 0, 0, 0
VB88  0, 0, 0, 0
VB92  0, 0, 0, 0
VB96  0, 0, 0, 7          //P7
VB100 18, 0, 0, 0
VB104 0, 0, 0, 0
VB108 0, 0, 0, 0
VB112 0, 0, 0, 18         //P8
VB116 0, 0, 0, 0
VB120 0, 0, 0, 0
```

VB124 0, 0, 0, 0
VB128 0, 0, 0, 8 //P9
VB132 0, 0, 0, 1
VB136 0, 0, 0, 0
VB140 0, 0, 0, 0
VB144 0, 0, 9, 0 //P10
VB148 0, 10, 0, 0
VB152 0, 10, 9, 0
VB156 0, 0, 0, 4
VB160 0, 10, 0, 0 //P11
VB164 0, 0, 9, 0
VB168 0, 10, 9, 0
VB172 0, 0, 0, 4
VB176 0, 0, 0, 12 //P12
VB180 0, 0, 0, 0
VB184 0, 0, 0, 0
VB188 0, 16, 14, 12
VB192 0, 0, 0, 14 //P13
VB196 0, 0, 0, 0
VB200 0, 0, 0, 0
VB204 0, 17, 15, 13
VB208 0, 0, 0, 16 //P14
VB212 0, 0, 0, 0
VB216 0, 0, 0, 0
VB220 0, 0, 0, 0
VB224 0, 0, 0, 0 //Pta_Close
VB228 0, 0, 0, 0
VB232 0, 0, 0, 0
VB236 0, 0, 0, 0
VB240 0, 0, 0, 13 //VS_Close
VB244 0, 0, 0, 0
VB248 0, 0, 0, 0
VB252 0, 0, 0, 0
VB256 0, 0, 0, 12 //VS_Open

VB260 0, 0, 0, 0
VB264 0, 0, 0, 0
VB268 0, 0, 0, 0
VB272 0, 0, 0, 15 //VD1_Close
VB276 0, 0, 0, 0
VB280 0, 0, 0, 0
VB284 0, 0, 0, 0
VB288 0, 0, 0, 14 //VD1_Open
VB292 0, 0, 0, 0
VB296 0, 0, 0, 0
VB300 0, 0, 0, 0
VB304 0, 0, 0, 17 //VD2_Close
VB308 0, 0, 0, 0
VB312 0, 0, 0, 0
VB316 0, 0, 0, 0
VB320 0, 0, 0, 16 //VD2_Open
VB324 0, 0, 0, 0
VB328 0, 0, 0, 0
VB332 0, 0, 0, 0
VB336 0, 0, 0, 19 //P_Close
VB340 0, 0, 0, 0
VB344 0, 0, 0, 0
VB348 0, 0, 0, 0
VB352 0, 0, 0, 18 //P_Open
VB356 0, 0, 0, 0
VB360 0, 0, 0, 0
VB364 0, 0, 0, 0
VB368 0, 0, 0, 11 //ALARMA
VB372 0, 0, 0, 20 //AL_ON
VB376 0, 0, 0, 21 //L11 y L12
VB380 0, 0, 0, 22 //VALVULAS
VB384 0, 0, 0, 23 //ASPERSOR

Desarrollo:

El automatismo debe responder al siguiente esquema Grafcet:

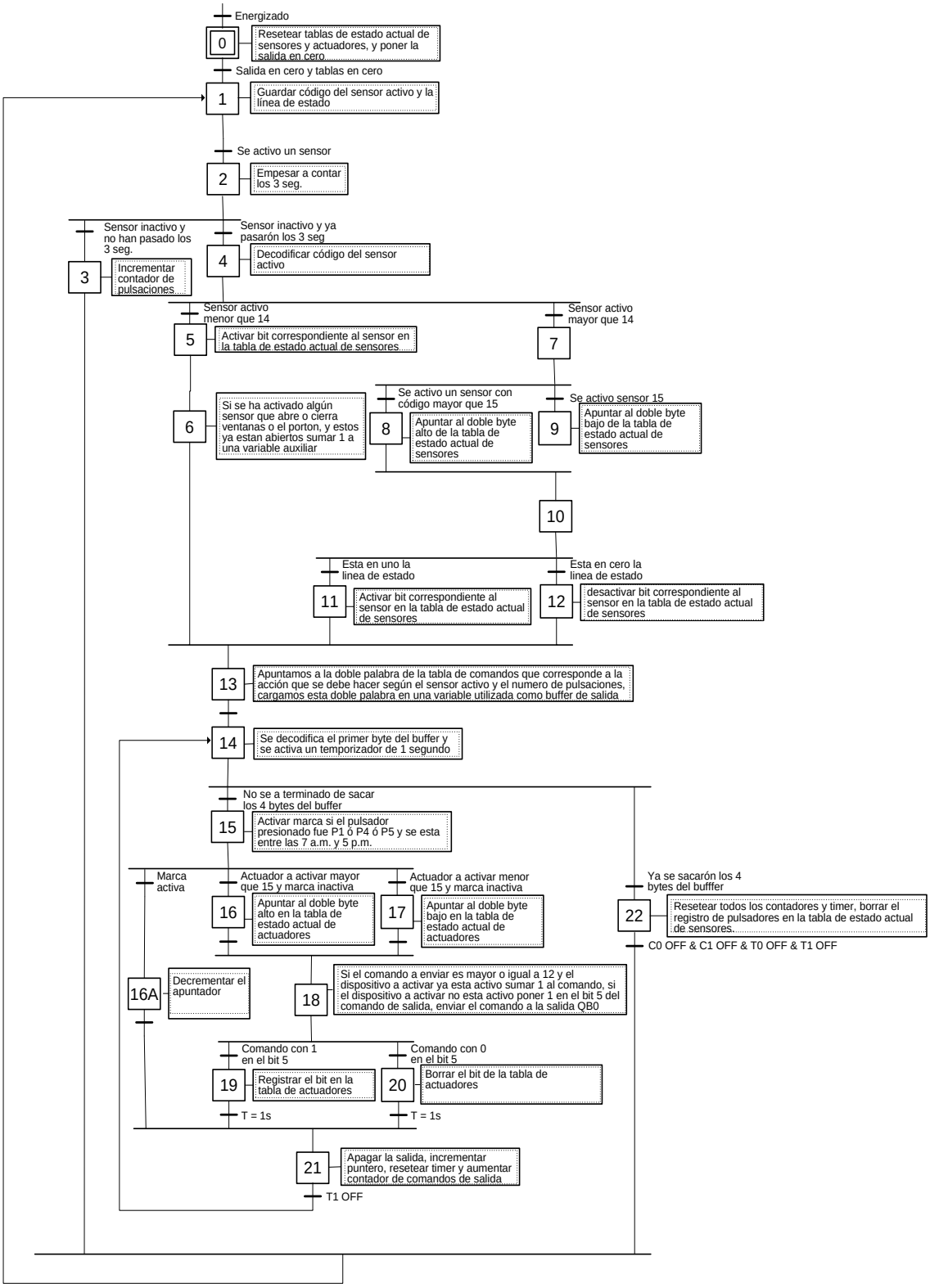


Fig. 1.28. Grafcet de la práctica.

Descripción de las etapas:

ETAPA	FUNCION
0	Se colocan los registros donde se almacena el estado de los actuadores a 0 y se apagan las salidas.
1	Se enmascaran las entradas para solo comprobar el estado del bit 6 el resultado se coloca en MB13. Nuevamente se enmascaran las entradas para comprobar el estado de los cinco primeros bit, si el resultado es cero se queda en la misma etapa si el resultado es distinto de cero pasa a 2.
2	Se empiezan a temporizar los 3 seg. Que tienen las personas para realizar una acción.
3	Se entra en esta etapa si no han pasado los tres segundos, si no se han contado el número máximo de pulsaciones y si las entradas están apagadas. En esta etapa se lleva el conteo de las teclas pulsadas.
4	Cuando pasan los tres segundos se decodifica el numero binario que se capturó en las entradas y se coloca en MW14. Se comprueba si el pulsador presionado está arriba de 14.
5	Si el pulsador presionado es menor que el 14 se registra en VW354 que fue presionado el pulsador.
6	En el caso que se desee cerrar una ventana, debido a que esta acción se realiza con el mismo número de pulsaciones, se comprueba que esta está abierta y se coloca 1 en VW350 la cual es una variable que se sumará después a un apuntador.
7	Etapa de transición. Se activa solo si el pulsador presionado es mayor del 14.
8	Si el pulsador presionado es del 16 en adelante se activa esta etapa, y en ella se apunta al byte VB352 para poder registrar la presión de pulsadores de presión continua como los del portón y las ventanas.
9	Se activa solo si el pulsador presionado fue el 14 para poder registrar este como el último de la palabra VW354.
10	Se comprueba si la entrada I0.6 estaba ó no activa al momento del comando de entrada.
11	Si estaba en uno se registra como activo el estado del sensor correspondiente, estableciendo a 1 el bit correspondiente.
12	Si estaba a cero se apaga el bit correspondiente al sensor.
13	En esta etapa se realiza la siguiente operación: LD10= (MW10-1)*16+ ((C0-1)+VW350)*4, esto hace que LD10 apunte a la doble palabra que contiene el registro de los cuatro byte a ser sacados. Estos registros se transfieren a MD20 y luego LD10 apunta al primer Byte de este registro.
14	Borrar el byte MB18 guardando el estado de las salidas 6 y 7, y colocando en este el primer byte a ser sacado. Se inicia el conteo del tiempo en que esta salida será mantenida.
15	Si no se han sacado los 4 byte, entonces se comprueba que no estamos entre las 7 am y las 5 pm, debido a que entre estas horas no se puede encender las luces L1, L4 y L5. Si este es el caso no se saca el byte.
16	Se comprueba si el actuador a ser accionado es mayor que 15, si es así, LD14 apunta al registro VW356.
17	Si el actuador es menor o igual que 15, LD14 apunta a VB358, en el registro VD356 se almacenan los actuadores activos (1) o no activos (0).
18	Se verifica si el actuador esta encendido o apagado, si este está encendido el comando enviado llevará la orden de apagarlo y si este estaba apagado el comando enviado llevará la orden de encenderlo.
19	Si el comando fue para encenderlo se registra en VD356 como encendido
20	Si el comando fue para apagarlo se registra en VD356 como apagado.
21	Se pone a cero las 6 primeras salidas y se conserva el estado de las 2 restantes.
22	Se inicializa todo y se borra el estado de los pulsadores momentáneos.

Tabla 1.9. Funciones realizadas por cada etapa en práctica de domótica.

Importante:

Para el desarrollo de esta práctica se debe crear un reloj con horas y minutos en el formato de 24 horas para que nos de la base de tiempos que necesitamos. Se propone el siguiente esquema:

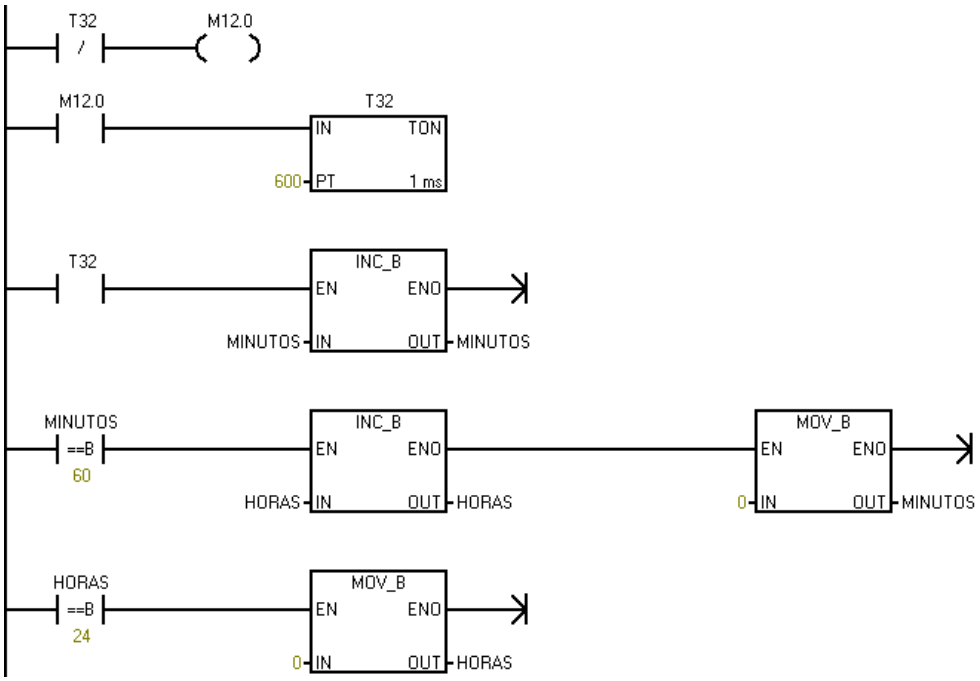


Fig. 1.29. Reloj a implementar para el desarrollo de esta practica.

Este bloque solo se debe agregar al programa. Por medio de un temporizador de 100ms se va a generar un pulso cada segundo el cual incrementa el valor de las variables “MINUTOS” y “HORAS”, para los minutos y horas del tiempo en curso respectivamente.

La base del esquema es el temporizador de 100 ms con un conteo máximo de 600 con lo cual se logra obtener 60 segundos (línea 2) , el cual se restablece por medio de la marca M12.0 y un contacto normalmente cerrado asociado al temporizador (línea 1).

La tercera línea utiliza la función INC_B para aumentar la variable MINUTOS en 1, cada vez que se cumplen 60 segundo; es decir, cada vez que el temporizador llega a la cuenta de 600.

La cuarta línea del esquema compara la variable MINUTOS con 6, cada vez que son iguales se utiliza la función INC_B para incrementar la variable HORAS y además para mover un valor de 0 a MINUTOS.

La última línea de código compara la variable HORAS con 24, cada vez que son iguales se hace un movimiento de 0 a la misma variable para restablecer el reloj.

Desarrollo de la práctica:

- Obtener el correspondiente diagrama de escalera para el Grafcet presentado. Utilizando para este caso la estructuración de programas con la función SCR.
  Para ver el programa en KOP completo, refiérase al archivo "*Domotica_Compleja.mwp*" que se encuentra en el CD adjunto en este documento.
- Realizar las conexiones en el modulo entrenador como se muestra a continuación

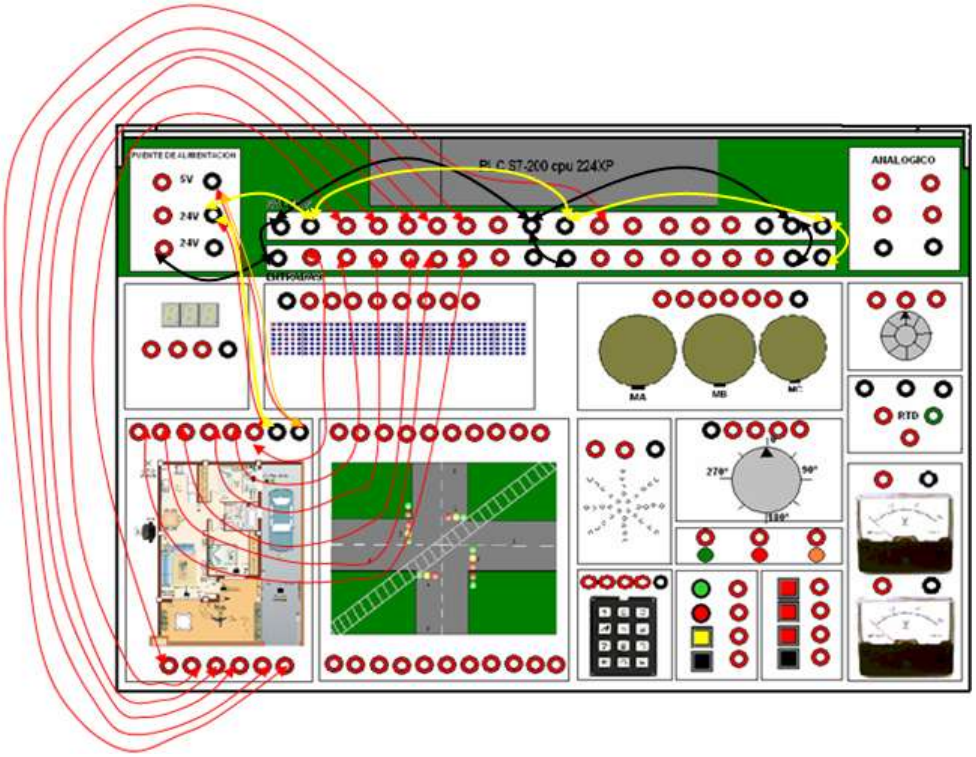


Fig. 1.30. Conexiones a realizar en el trainer para la práctica de domotica compleja.

- Asegurarse que los interruptores on-off-on están en la posición OFF (su posición de en medio) y que el interruptor de la puerta este hacia la izquierda (OFF).
- Conectar el cable PC/PPI al S7-200.
- Energizar el modulo. El led Bi-color de alarma activa encenderá color verde.
- Programar el S7-200.
- Realizar las pruebas respectivas.
- Escriba sus conclusiones y/o observaciones

Actividad:

Se debe de agregar al automatismo que las persianas se abran automáticamente a las 6:00 A.M. y se cierren completamente a las 7:00 P.M, de igual forma las luces exteriores se encenderán a las 6:00 pm y se apagarán a las 10:00 pm, El aspersor debe encenderse a las 9:00 am y debe de trabajar durante una hora, manteniendo presionado P11 ó P12 mas de 2 segundos se debe activar la alarma que se enciende automáticamente a las 5:00 am emitiendo tres tonos cortos de 2 segundos.

1.3 PRACTICAS DE FUNCIONES INTERMEDIO-AVANZADAS.

1.3.1 CONTROL ANGULAR DE UN MOTOR PASO A PASO.

Objetivos:

- Introducir al estudiante a la utilización de las funciones de rango medio-avanzado de los PLC.

- Aplicar los conocimientos de funciones lógicas, comparación y rotación en los PLC S7-200.

- Introducir al uso de funciones con tablas de memoria de datos en los PLC S7-200.

Materiales a utilizar:

- 1 PC
- 1 Cable PC/PPI
- 1 S7-200
- 1 Teclado matricial
- 1 maqueta simuladora para demótica
- 1 buzzer
- 10 cables de conexión

Planteamiento:

Este sistema contara con teclado numérico, con el cual se introduce el número de vueltas que se desea que el motor gire. Luego de introducir el valor de vueltas deseados se debe presionar la tecla "#" con la cual el motor paso a paso debe iniciar a girar la cantidad de vueltas indicadas anteriormente. El máximo valor de vueltas a introducir es de 999 vueltas; es decir, se debe restringir el número de entrada a 3 dígitos. Al tratar de introducir más de 3 dígitos se debe activar una luz de error, para luego esperar el "#" y empezar a girar.

Luego de haber introducido el valor de las vueltas se debe de activar un indicador de datos validos, y se debe mostrar el valor de vueltas deseadas en un display de 3 dígitos, para lo cual será necesario utilizar rotación (salida serial), la cual deberá sacar la palabra codificada en 7 segmentos tal como se muestra a continuación:

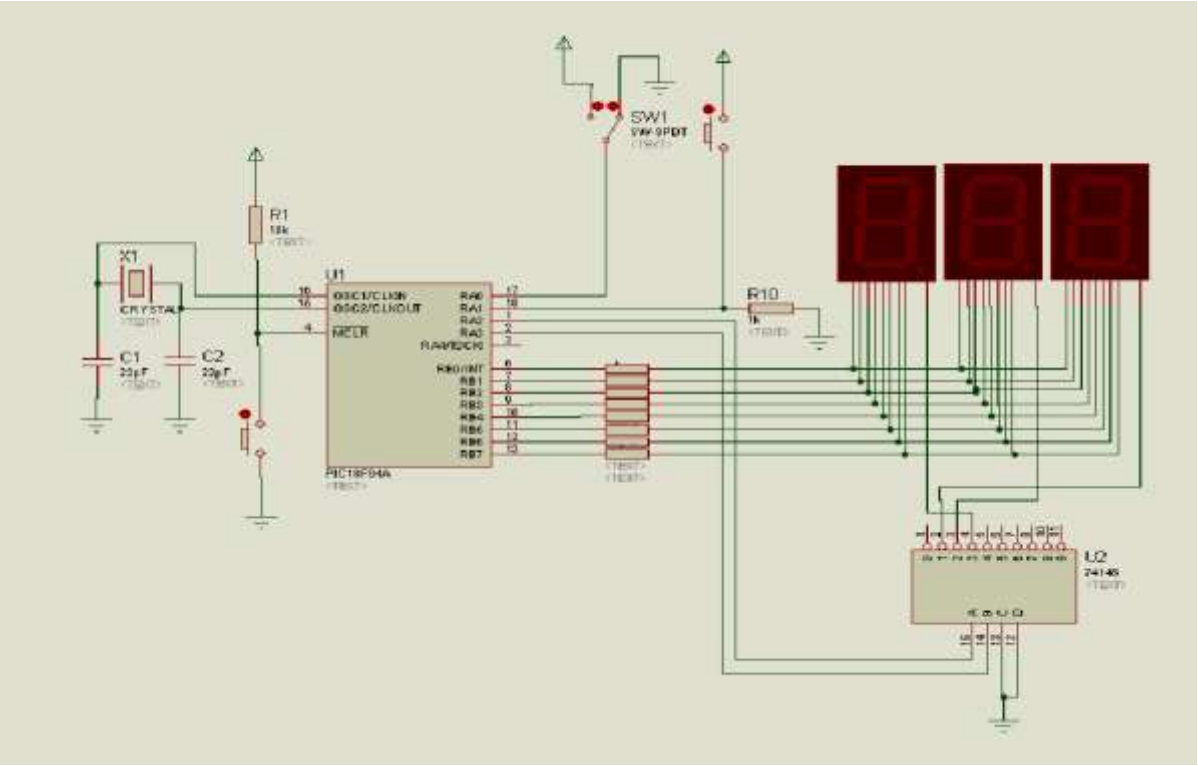


Fig. 1.31. Esquema del circuito para display de 3 dígitos.

Se debe notar que el RESET del modulo de display es activo en alto, y se debe mantener en bajo durante todo el funcionamiento del automatismo.

Para controlar un motor paso a paso unipolar se tiene la siguiente secuencia en una tabla de datos previamente establecida en el S7-200:

Paso	Secuencia			
	Bobina A	Bobina B	Bobina C	Bobina D
1	ON	OFF	OFF	OFF
2	OFF	ON	OFF	OFF
3	OFF	OFF	ON	OFF
4	OFF	OFF	OFF	ON

Tabla. 1.10. Combinación de entradas en las líneas del motor.

Esta Secuencia debe ser enviada un registro a la vez por medio de la función SEQUENCER, la cual no está disponible para el S7-200, por lo cual se debe utilizar direccionamiento indirecto para enviar toda la tabla las veces necesarias, en el caso de nuestro entrenador se tiene un motor PAP de 7.5°/paso por lo que se hace necesario repetir la secuencia 48 veces para completar una vuelta (360°).

Es decir, la tabla debe ser enviada por el puerto de salida la n vueltas que se introduzcan desde el teclado, con el funcionamiento descrito anteriormente.

Al haber completado el número de vueltas deseado y finalizado el ciclo, se debe esperar una orden de re- arranque a través de un pulsador para volver a iniciar el funcionamiento antes descrito.

Desarrollo:

El sistema debe corresponder al grafcet de macroetapas mostrado a continuación:

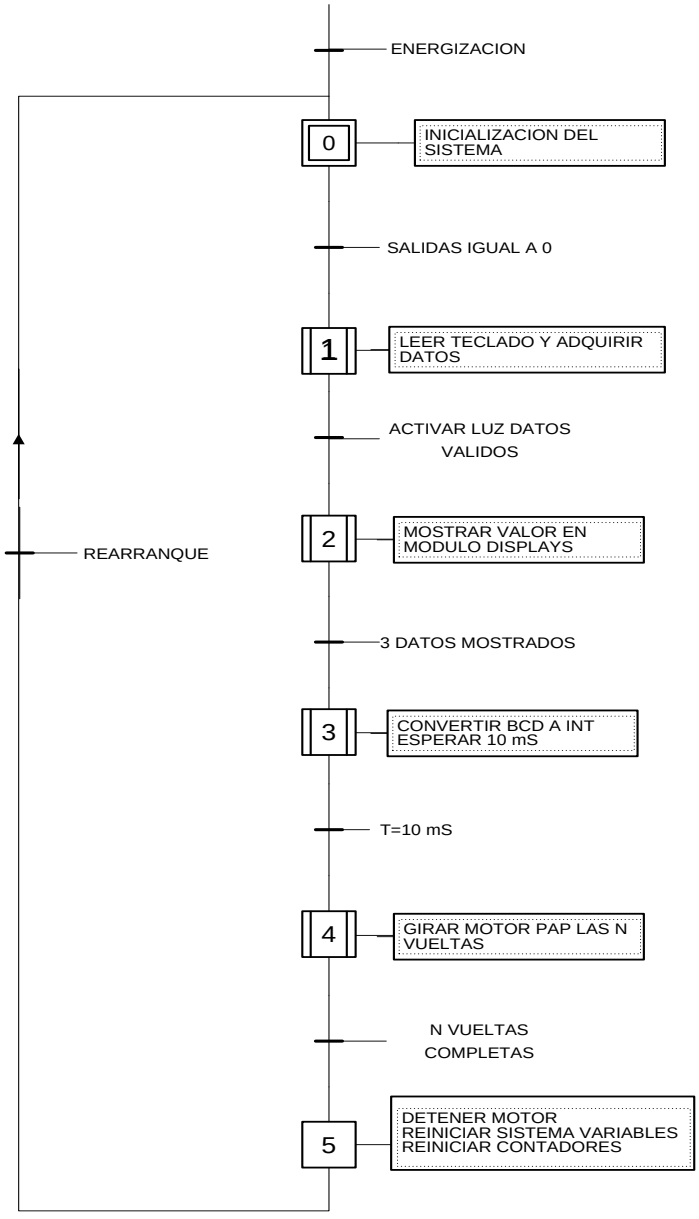


Fig. 1.32. Grafcet del automatismo.

Las macro etapas del automatismo se muestran detalladas a continuación:

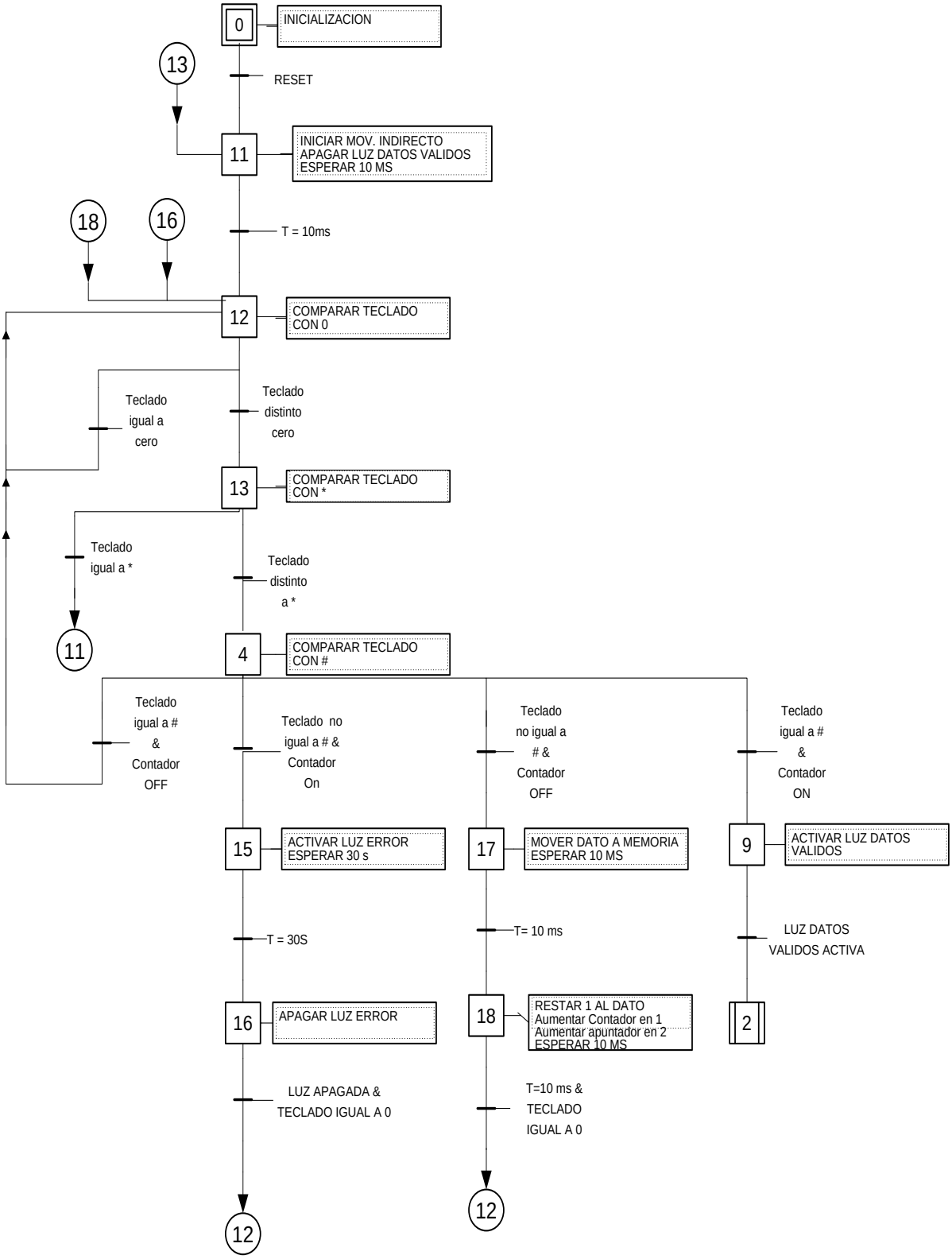


Fig. 1.33. Macroetapa 1.

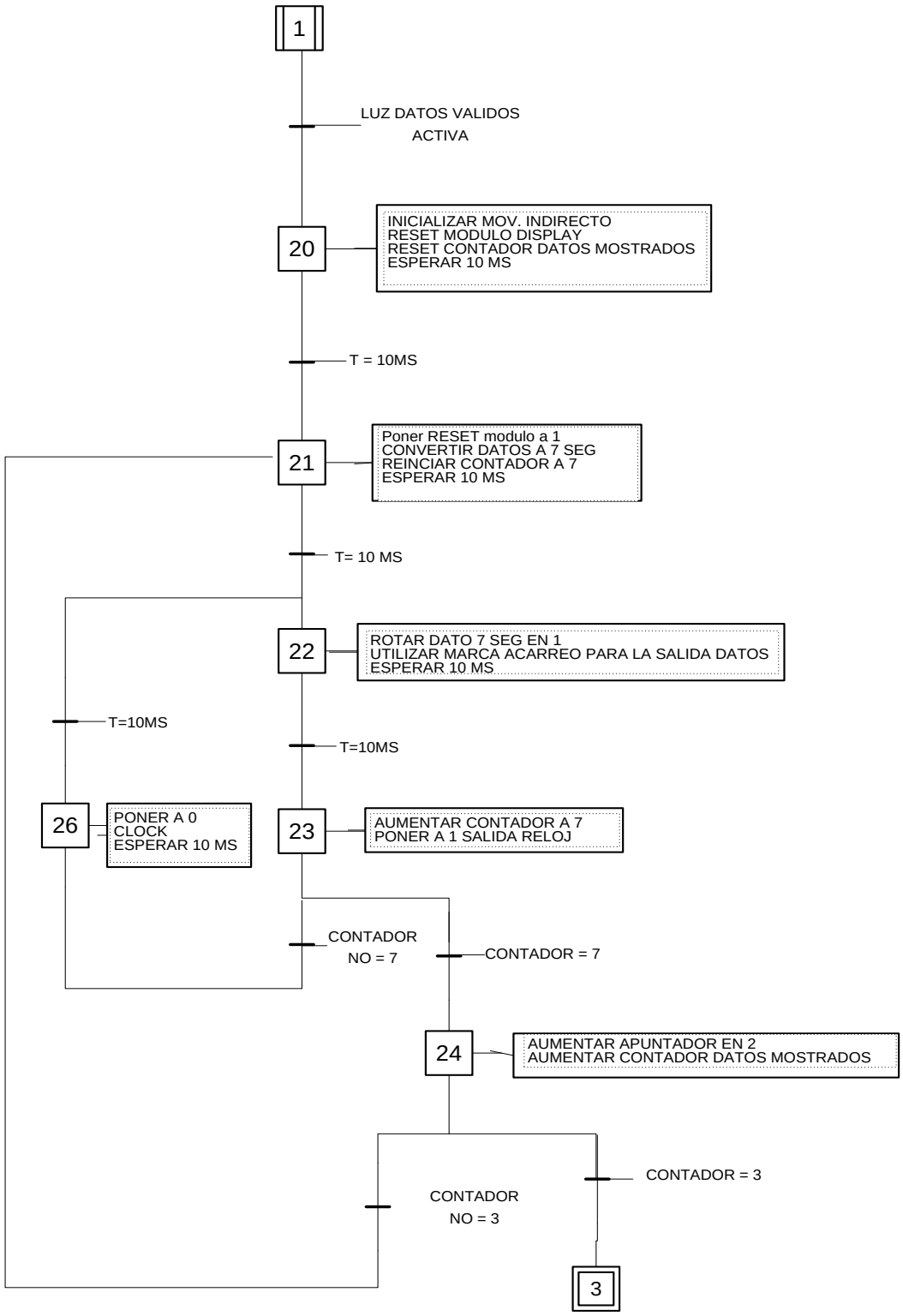


Fig. 1.34. Macroetapa 2.

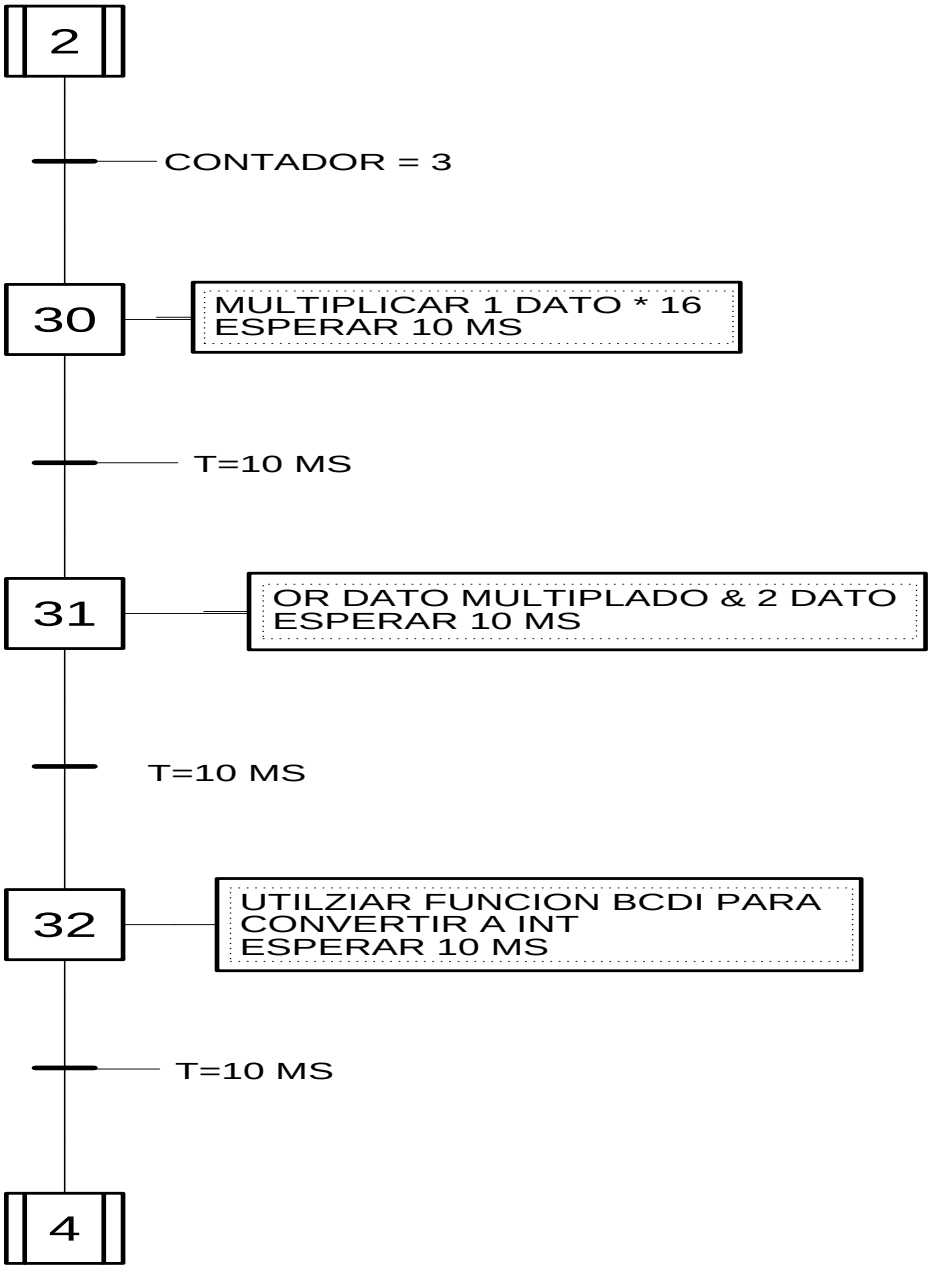


Fig. 1.35. Macroetapa 3.

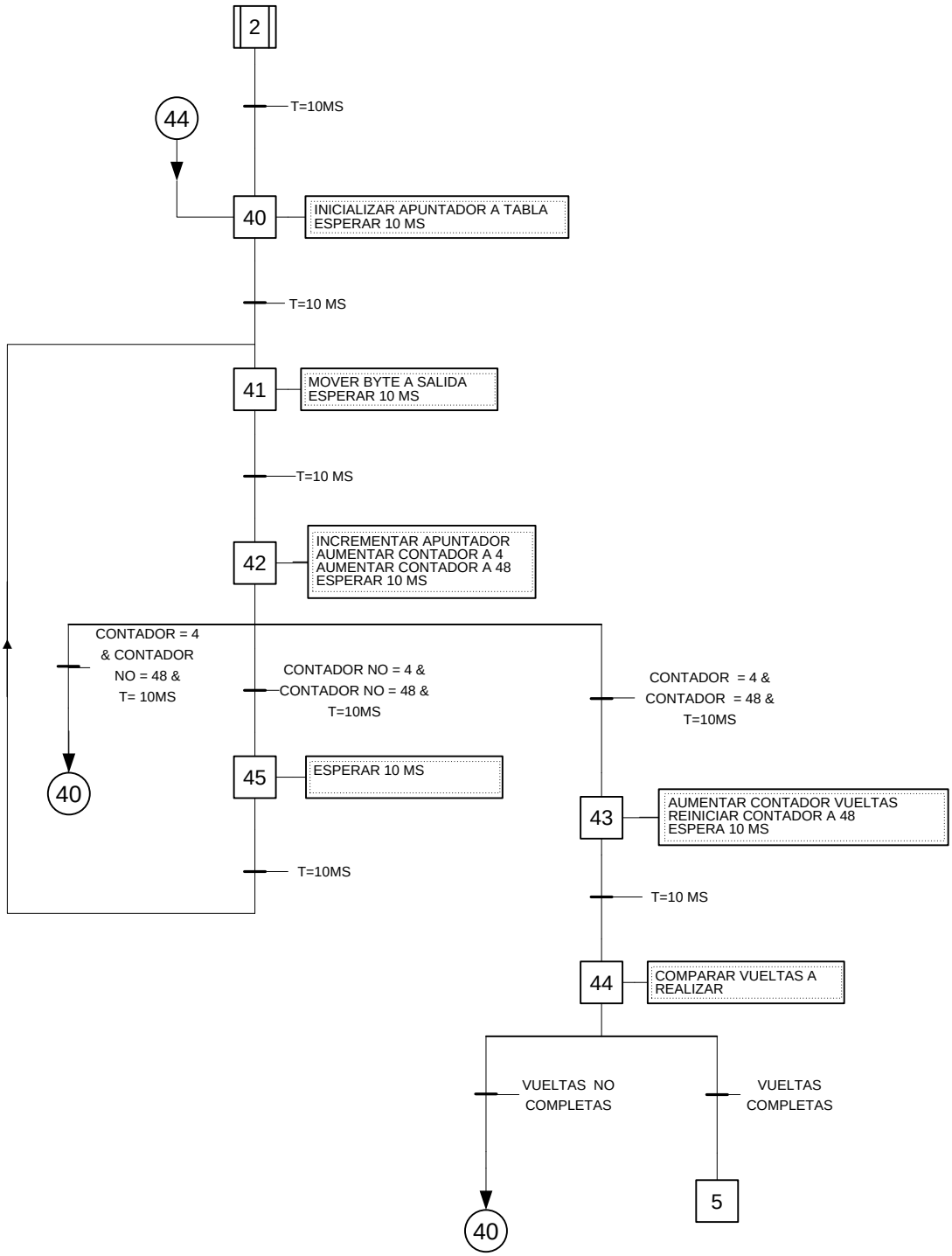


Fig. 1.36. Macroetapa 4.

NOTA:
Las operaciones de comparación se utilizan para comparar dos valores:
IN1 = IN2 IN1 >= IN2 IN1 <= IN2 IN1 > IN2 IN1 < IN2 IN1 <> IN2

Es de tomar en cuenta lo siguiente:

- Las comparaciones de bytes no llevan signo.
- Las comparaciones de enteros llevan signo.
- Las comparaciones de palabras dobles llevan signo.
- Las comparaciones de números reales llevan signo.

En KOP y FUP: Si la comparación es verdadera, la operación de comparación activa el contacto (KOP) o la salida (FUP).

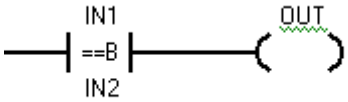


Fig. 1.37. Función Comparación.

Desarrollo de la práctica:

- Obtener el correspondiente diagrama de escalera para el Grafcet presentado.
- Elaborar en Micro/Win el respectivo programa en KOP para esta práctica.
- Realizar las conexiones en el modulo entrenador como se muestra a continuación.

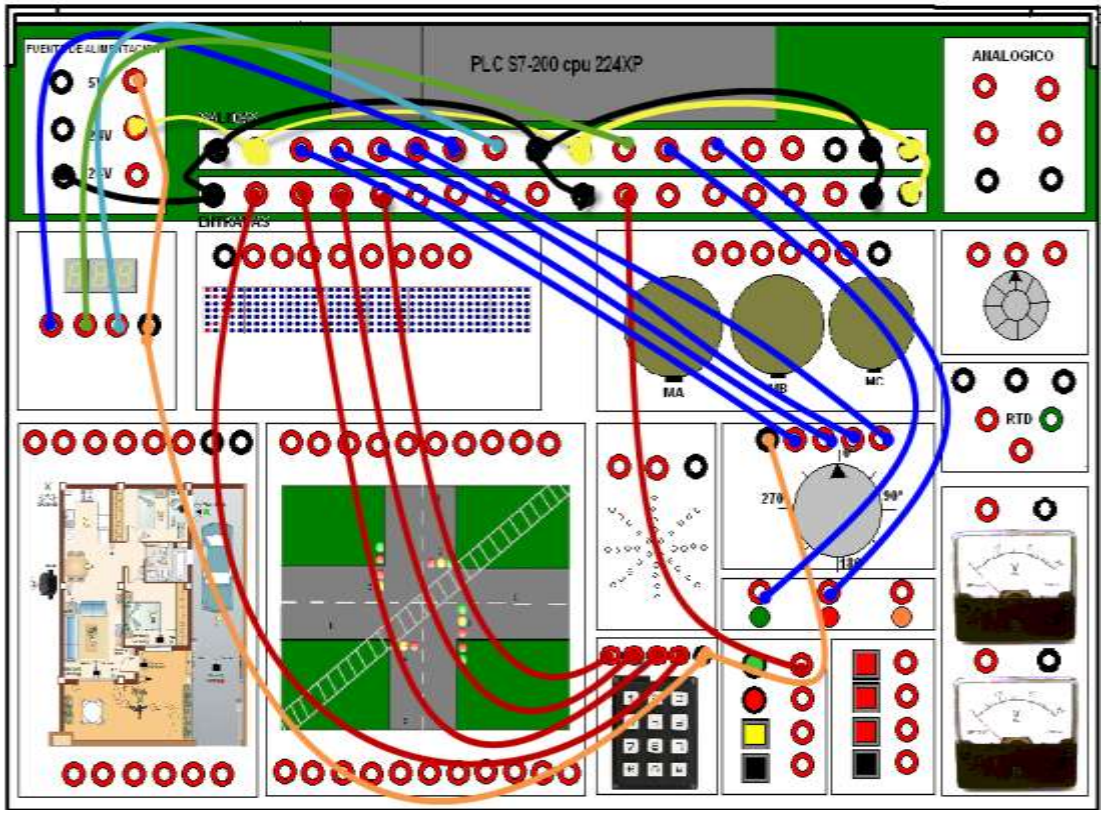


Fig. 1.38. Conexiones a realizar en la práctica del motor paso a paso.

- Conectar el cable PC/PPI al S7-200 y Programarlo.
- Escriba sus conclusiones y/o observaciones

Actividad:

- Al sistema anterior debe agregarse la opción de poder girar en sentido horario o anti horario el motor PAP, dependiendo de si el usuario presiona "* "o "# "respectivamente.

- El funcionamiento debe seguir los lineamientos antes establecidos, el mando para decidir en qué sentido debe girar el motor PAP se realizara en la macroetapa 4, utilizando el mismo grafcet para las demás macroetapas.

1.3.2 FLECHA DE DESVIO VIAL

Objetivos:

- Aprender a utilizar interrupciones en la programación del S7-200. *(1)

- Poner en práctica la utilización de subrutinas como una forma más ordenada de estructurar los programas.

- Utilizar el ciclo de trabajo del PLC para estructurar programas sin ningún tipo de transición en sus etapas.

Materiales a utilizar:

- 1 PC
- 1 Cable PC/PPI
- 1 S7-200
- 1 Pantalla de puntos 7x40
- 19 cables de conexión

Planteamiento:

En una pantalla matricial de puntos, se simulará una flecha de desvío vial, la cual cuenta con tres secciones que se encenderán de forma secuencial para simular el desplazamiento de la flecha a la derecha.

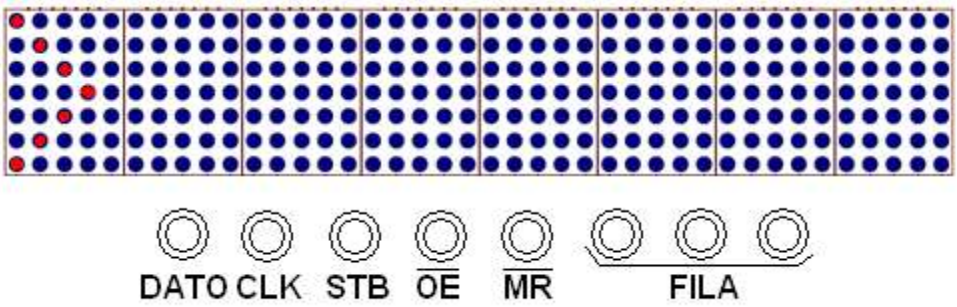


Fig. 1.39 Esquema de la pantalla matricial de puntos.

La pantalla funciona de la siguiente manera, en los bornes indicados por 'FILA' se debe de especificar el número de fila en binario a la cual se le enviarán los datos de forma secuencial; el estado de cada punto de la matriz se introduce de forma secuencial por el terminal 'DATO' empezando desde el ultimo que será mostrado, primero se debe colocar un nivel lógico en el borne 'DATO' y luego se produce un flanco de subida en el terminal 'CLK' para que el valor sea introducido, se debe repetir esto hasta el último dato requerido.

*(1) : Consultar anexo A para mayor información sobre interrupciones.

Un pulso en el terminal 'STB' carga los datos introducidos en los latch de salida y luego estos se deben mostrar colocando un nivel lógico bajo en el terminal 'OE', el terminal 'MR' siempre debe estar en nivel lógico alto a menos que se quiera restablecer todos los latch de entrada.

Para esta práctica se debe de crear una tabla de datos en el área de memoria 'Vx.x' empezando desde la dirección VB0 hasta VB77, estos datos serán sacados byte a byte y enviados a QB0. *Para mayor información sobre la edición de un bloque de datos consultar anexo A sección A1.8.3.*

El bloque de datos que se debe crear es mostrado a continuación.

```
VB0   2#10010, 2#10011, 2#11110100, 2#110000
VB4   2#11000, 2#11001, 2#11010100, 2#1010000
VB8   2#11000, 2#11001, 2#10110100, 2#10000, 2#1110000
VB13  2#11000, 2#11001, 2#10010100, 2#0
VB17  0

VB18  2#10010, 2#10011, 2#10000, 2#10001, 2#10000, 2#10001, 2#10000, 2#10001, 2#10000,
      >>2#10001, 2#11110100, 2#110000
VB30  2#11000, 2#11001, 2#11010100, 2#1010000
VB34  2#11000, 2#11001, 2#10110100, 2#10000, 2#1110000
VB39  2#11000, 2#11001, 2#10010100, 2#0
VB43  0

VB44  2#10010, 2#10011, 2#10000, 2#10001, 2#10000, 2#10001, 2#10000, 2#10001, 2#10000,
      >>2#10001, 2#10000, 2#10001, 2#10000, 2#10001, 2#10000, 2#10001, 2#10000, 2#10001,
      >>2#11110100, 2#110000
VB64  2#11000, 2#11001, 2#11010100, 2#1010000
VB68  2#11000, 2#11001, 2#10110100, 2#10000, 2#1110000
VB73  2#11000, 2#11001, 2#10010100, 2#0
VB77  0
```

Desarrollo:

El automatismo debe responder al siguiente esquema Grafcet:

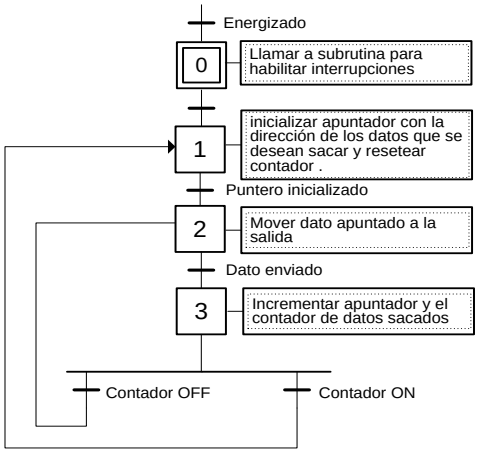


Fig. 1.40. Grafcet Principal de la práctica.

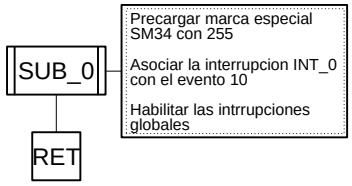


Fig. 1.41. Grafcet de la Subrutina 0.

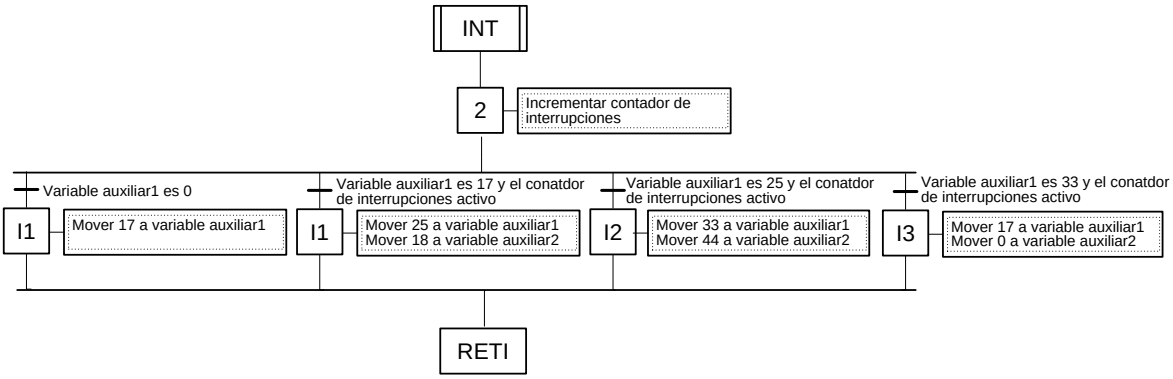


Fig. 1.42. Grafcet de la Rutina de interrupción.

Descripción de las etapas:

ETAPA	FUNCION
0	Se llama a la subrutina donde se habilitaran las interrupciones.
1	Se inicializa el apuntador con la dirección del primer dato que se desea sacar (VB0), además se fija el set del contador de datos a 17 y se resetea el contador de datos.
2	Se mueve el dato apuntado a la salida.
3	Se incrementa el apuntador hacia el segundo dato que se desea sacar y se incrementa el contador de datos sacados.
SUB_O	Sub rutina, en esta se carga la variable SM34 con un valor comprendido entre 0 y 255, este valor representa cuantos milisegundos pasarán entre cada interrupción, se asocia el evento 10 a la INT_0, este evento corresponde a una interrupción temporizada en la cual el valor fijado en SM34 es comparado constantemente con T32 y si ocurre una coincidencia la interrupción se activa. Además se habilitan las interrupciones globales.
INT	Se encarga de cambiar la posición de la flecha, después de cierto tiempo pasa al siguiente display con esto emulamos el movimiento hacia la derecha de la flecha.
I1	Si el puntero apunta a VB0 al momento de la interrupción se re direcciona el puntero a VB18 y se fija el set del contador de datos sacados a 25.
I2	Si el puntero apunta a VB18 al momento de la interrupción se re direcciona el puntero a VB44 y se fija el set del contador de datos sacados a 33.
I3	Si el puntero apunta a VB44 al momento de la interrupción se re direcciona el puntero a VB0 y se fija el set del contador de datos sacados a 17.

Tabla 1.11. Funciones realizadas por cada etapa en práctica de la flecha de desvió vial.

Desarrollo de la práctica:

- Obtener el grafcet de segundo nivel de la práctica.
- Obtener el correspondiente diagrama de escalera para el Grafcet presentado.

- Editar en Micro-Win el respectivo programa en KOP para esta práctica.
- Realizar las conexiones en el modulo entrenador.

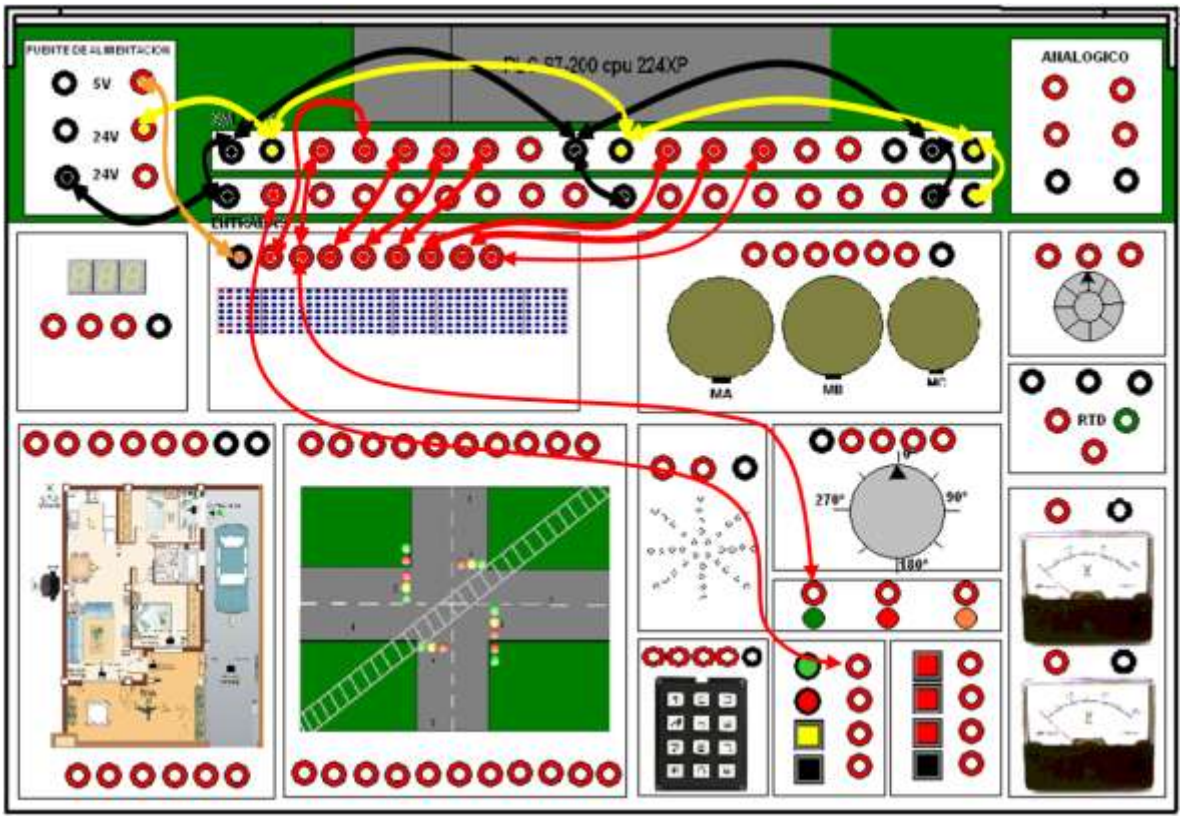


Fig. 1.43. Esquema de conexión de la pantalla de puntos en el modulo entrenador

Nota: *Para mejorar la señal de reloj que emite el PLC por la salida Q0.0 se debe conectar una carga a esta, como se muestra en el esquema, para esta práctica se utiliza de carga una de las lámparas del entrenador.*

- Conectar el cable PC/PPI al S7-200 y Programarlo.
- Escriba sus conclusiones y/o observaciones

Actividad:

Se debe hacer que la pantalla se cubra de flechas una a continuación de la otra apagándose y encendiéndose cada 3 segundos como se muestra en la imagen.

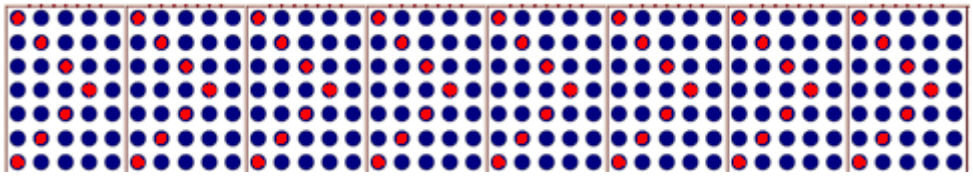


Fig. 1.44. Primera imagen a mostrar en la actividad.

Cuando se presione un pulsador P0 conectado a la entrada I0.0 se debe de borrar las flechas y pasar a mostrar la siguiente imagen que debe de quedar estática en pantalla.

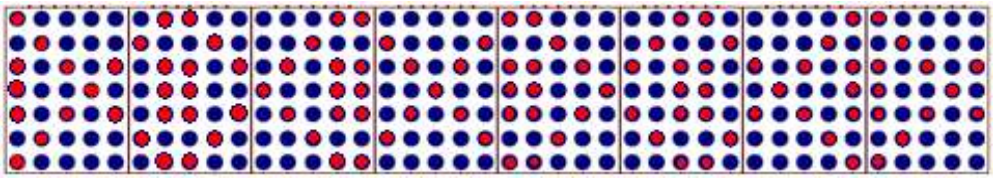
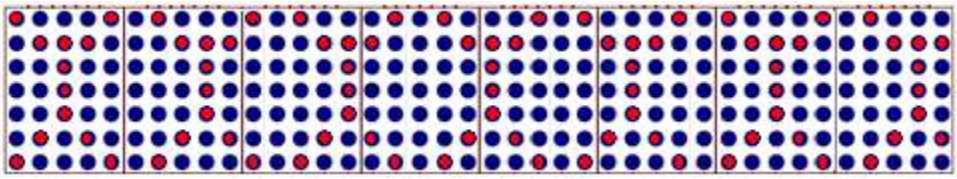
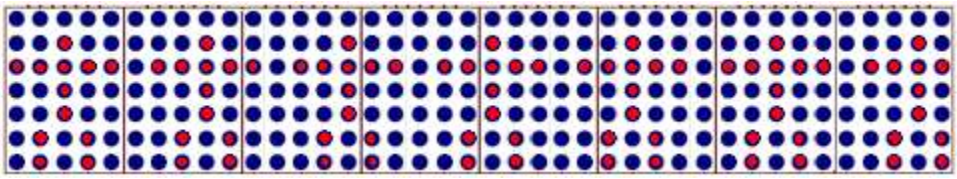


Fig 1.45. Segunda imagen a mostrar en la actividad.

Y con una nueva pulsación de P0 se borra esta imagen, y se pasa a mostrar dos imágenes que se alternarán de forma continua cada segundo simulando un movimiento animado continuo. Las imágenes se muestran a continuación.



a)



b)

Fig. 1.46. Animación con dos imágenes, usada en la actividad como ultima figura mostrada.

En esta posición con una nueva pulsación de P0 se vuelven a sacar las flechas para repetir nuevamente otro ciclo. Como una función adicional se debe conectar un pulsador a la entrada I0.1 el cual será el botón de paro del sistema, al presionar este se debe de apagar la pantalla y desasociar cualquier evento de interrupción.

Utilizar interrupciones de detección de flanco de subida vía la entrada I0.0 y I0.1 para producir los cambios de imagen en la pantalla y el paro del sistema respectivamente.

1.3.3 TERMOVENTILADOR

Objetivos:

- Introducir al estudiante a la utilización de las funciones intermedio-avanzadas de los PLC.

- Aprender a utilizar las entradas analógicas de los PLC S7-200.

- Que el estudiante descubra aplicaciones prácticas para la utilización de subrutinas y las funciones analógicas del PLC S7-200.

Materiales a utilizar:

- 1 PC.
- 1 Cable PC/PPI.
- 1 S7-200
- 3 Motores DC.
- 1 lámpara.
- 20 cables de conexión.
- Modulo de termómetro analógico.

Planteamiento:

Se pretende controlar el funcionamiento de un termoventilador (simulado por tres motores DC) de forma que responda a las variaciones de temperatura que se produzcan en el local donde se instale.

El autómata controlara el funcionamiento de tres motores DC que simularan 3 ventiladores y el de una lámpara que indicara un descenso en la temperatura o aumento en la temperatura. Las condiciones de funcionamiento serán:

- Cuando la temperatura sea inferior a 30°, los ventiladores 1 y 3 deben estar apagados y solamente debe funcionar el ventilador 2; la lámpara estará encendida.

- A partir de los 30°, los 3 ventiladores deben estar encendidos y la lámpara se apagara.

- Si se activa el interruptor de paro, los 3 ventiladores deben dejar de funcionar y la lámpara deberá estar parpadeando, al desactivarse este interruptor, se debe comenzar nuevamente el funcionamiento normal.

La tabla siguiente resume los posibles modos de funcionamiento del automatismo:

Funcionamiento	*Modo 1*	*Modo 2*
Temperatura	< 30°	>=30°
Ventiladores Operando	2	1, 2 y 3
Lámpara	Encendida	Apagada

Tabla 1.12. Modos de funcionamiento del automatismo.

Para simular el funcionamiento del termostato, detector de la temperatura del local, se utilizo una RTD con electrónica necesaria y salida variable entre 0 y 10 V (0° a 100°, 1° por cada 0.1 V) conectada a través del modulo analógico del autómata.

Desarrollo:
El sistema debe corresponder al grafcet mostrado a continuación:

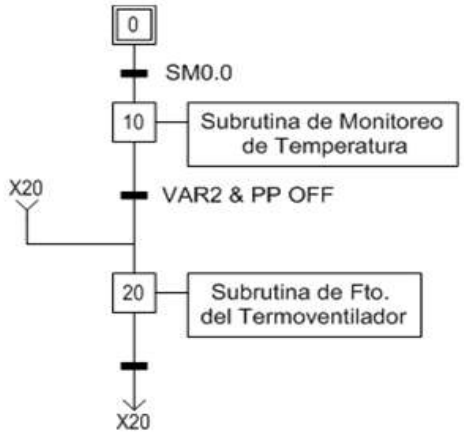


Fig. 1.47. Grafcet de Subrutinas.

Descripción de las etapas:

ETAPA	FUNCION
SUB_O	Sub rutina, en esta se lee la entrada analógica del PLC, este valor de entrada es transformado en un valor entre 0 y 32,738 (que representa un valor entre 0-10V), este valor se multiplica por 100, se convierte a real y se divide entre 32738 para obtener el valor en grados, este valor en grados se almacena en VD8
SUB_1	Se encarga comparar el valor en grados guardado con ciertos valores de temperatura definidos, si la temperatura es menor que 30 grados, activa el motor 2 y la lámpara y si es mayor o igual que 30 grados activa los tres motores y desactiva la lámpara, verifica también si se ha presionado el botón de paro, si esta activo, detiene los tres motores y la lámpara empieza a parpadear y se mantiene así hasta que el botón se desactive.

Tabla 1.13. Descripción de subrutinas.

Macro-etapa 10

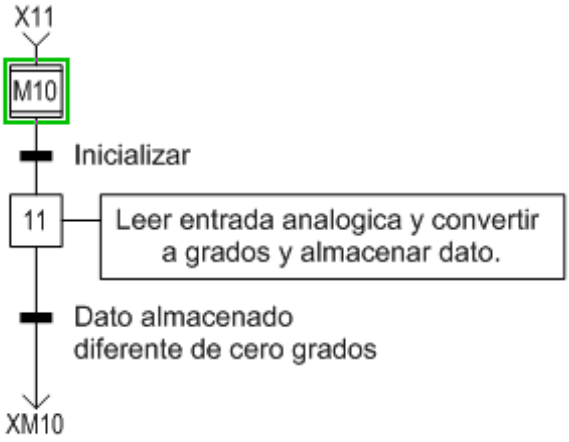


Fig.1.48. subrutina para lectura de entrada analógica.

Macro-etapa 20

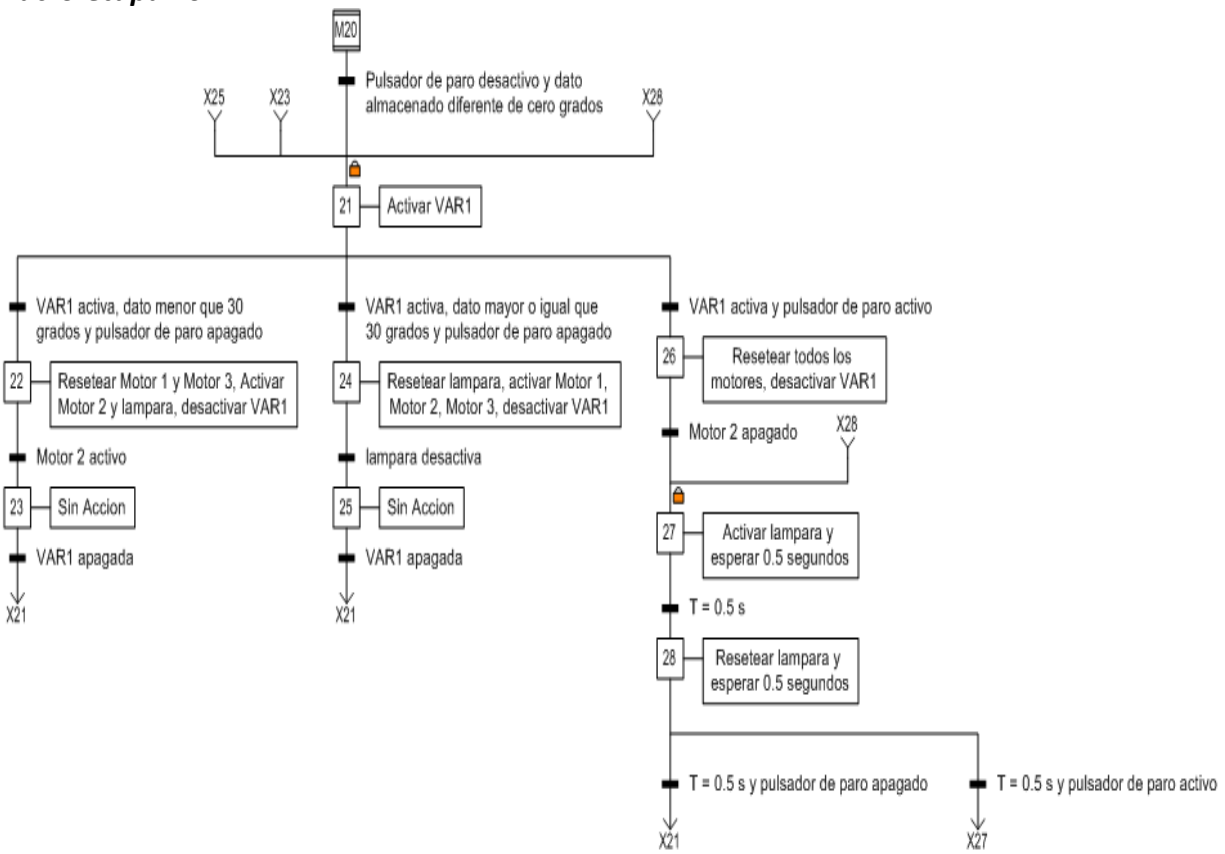


Fig. 1.49. Subrutina funcionamiento termoventilador.

Desarrollo de la práctica:

- Obtener el correspondiente diagrama de escalera para el Grafcet presentado.
- Elaborar en Micro/Win el respectivo programa en KOP para esta práctica.

- Realizar las conexiones en el modulo entrenador como se muestra a continuación.

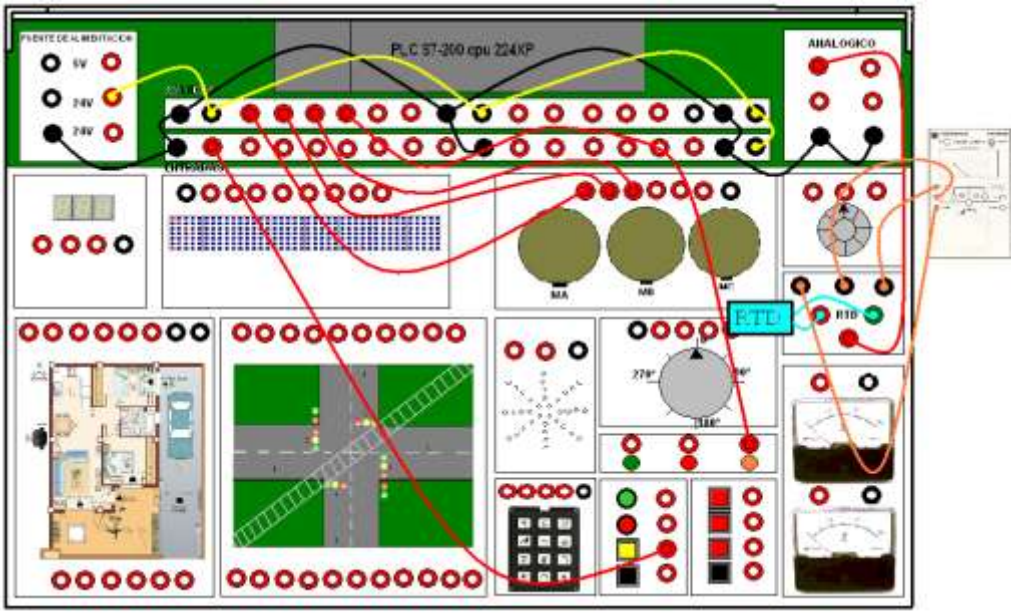


Fig. 1.50. Conexiones a realizar para la práctica del termoventilador.

- Conectar el cable PC/PPI al S7-200 y Programarlo.
- Escriba sus conclusiones y/o observaciones.

Actividad:

Rediseñar el sistema para que cumpla con las condiciones siguientes de funcionamiento:

- Cuando la temperatura sea inferior a 20°, el ventilador 1 estará funcionando, los ventiladores 2 y 3 estarán apagados y la lámpara estará apagada.
- A partir de los 20° y hasta 25°, los ventiladores 1 y 2 funcionaran, la lámpara estará encendida y el ventilador tres estará apagado.
- Entre 25° y 30°, los ventiladores 2 y 3 deben funcionar y el ventilador 1 debe estar apagado, la lámpara deberá estar encendida.
- A partir de 30°, funcionaran los 3 ventiladores y la lámpara deberá estar apagada.
- Si se activa el interruptor de paro, los 3 ventiladores deben dejar de funcionar y la lámpara deberá estar parpadeando, al desactivarse este interruptor, se debe comenzar nuevamente el funcionamiento normal.

La tabla siguiente resume los cuatro posibles modos de funcionamiento del automatismo:

Funcionamiento	*Modo 1*	*Modo 2*	*Modo 3*	*Modo 4*
Temperatura	<20°	20° - 25°	26° - 30°	>30°
Ventiladores operando	1	1 y 2	2 y 3	1, 2 y 3
Lámpara	Apagada	Encendida	Encendida	Apagada

Tabla 1.14. Modos de funcionamiento del automatismo.

CONCLUSIONES DEL CAPITULO 1.

Las prácticas cubren gran parte del área de aplicación de los PLC S7-200, al abordar estas guías de laboratorio propuestas se logra adquirir los conceptos básicos, medios y avanzados para programar cualquier tipo de micro PLC, debido a que los S7-200 son similares a los disponibles en el mercado y así poder adquirir los conceptos generales para programar cualquier controlador lógico.

CAPITULO 2

CONSTRUCCION ENTRENADOR PARA PLC S7-200

CAPITULO 2.
CONSTRUCCION DEL ENTRENADOR PARA PLC S7-200

Las prácticas planteadas en el capítulo 1 tratan de representar situaciones cotidianas, lo que hace necesario implementarlas físicamente, de donde surge la necesidad de diseñar el hardware para la implementación de cada una de ellas.

El presente capitulo tiene como objetivo detallar el diseño y la fabricación de el chasis para el entrenador; además, detallar el esquema eléctrico, el diagrama de pistas y el hardware utilizado en cada uno de los módulos de el entrenador para PLC S7-200.

Fig. 2.1. Entrenador para PLC S7-200.

2.1 DESCRIPCIÓN GENERAL

2.1.1 DATOS ESPECÍFICOS DE DISEÑO

Las dimensiones del entrenador de PLC se pueden observar en la fig. 2.2 mostrada a continuación:

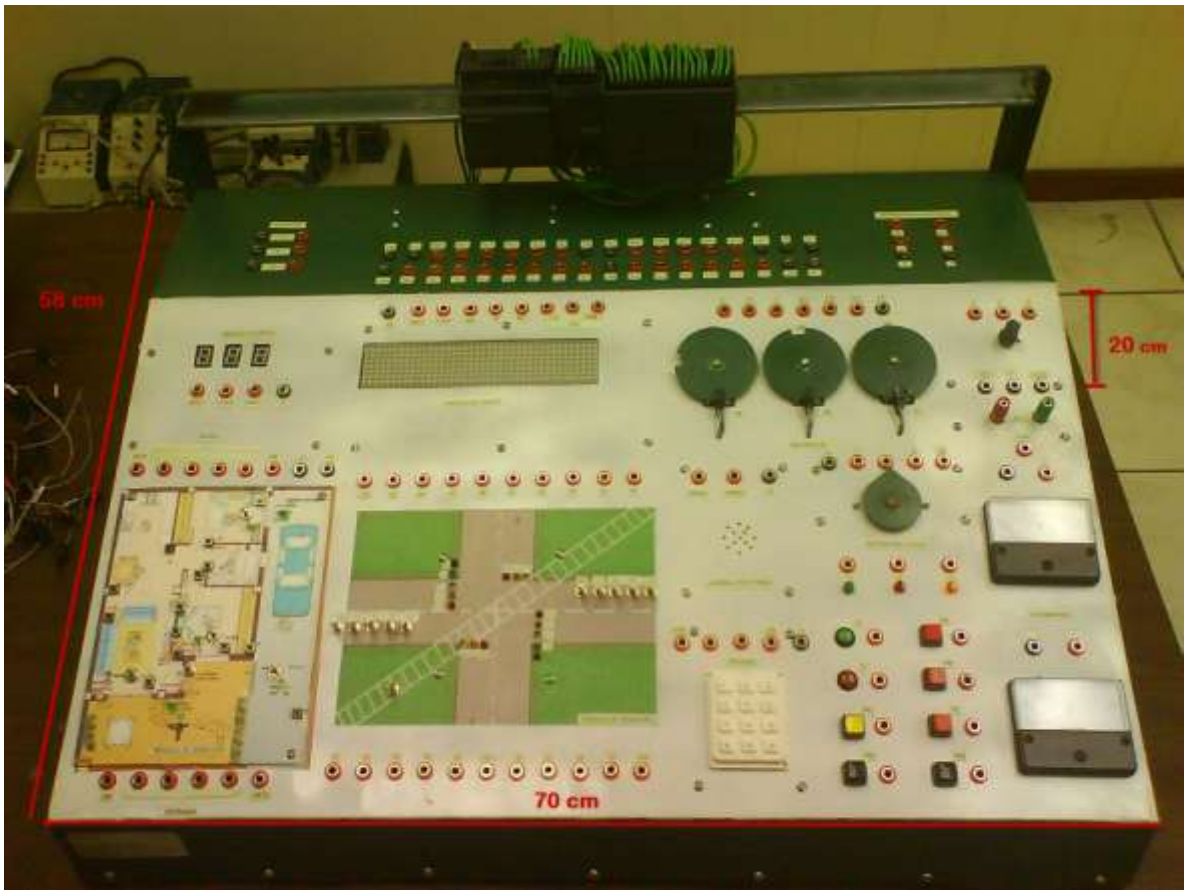


Fig. 2.2. Dimensiones Físicas Entrenador.

La vista frontal del entrenador es la que se muestra en la fig. 2.3, mostrada a continuación:

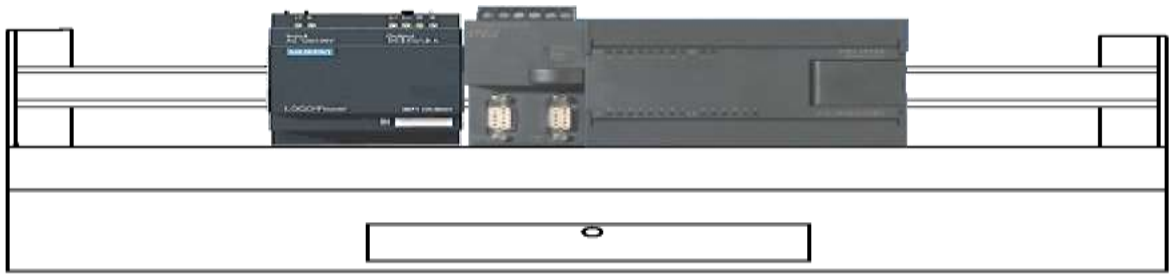


Fig. 2.3 Vista Frontal Entrenador.

Algunas características generales del entrenador son:

- 9 Módulos Simuladores.
- 3 Indicadores Luminosos a 24V (Verde, Rojo, Naranja).
- 5 pulsadores NA.
- 3 Push Button con auto retención.
- Voltaje Nominal: 120V
- Dimensiones:
  Ancho: 50 cm
  Largo: 60 cm
  Alto: 22 cm

2.2 CONSTRUCCIÓN DE MÓDULOS.

2.2.1 MODULO DE DOMÓTICA

El modulo de domótica fue necesario para resolver una necesidad en la simulación de una casa donde se puedan controlar las distintas partes de esta, para cumplir con las especificaciones de entradas/salidas de la CPU 224 XP; por lo cual se tuvo que reducir las entradas y salidas tal como se explica en el capítulo 1, practicas básicas.

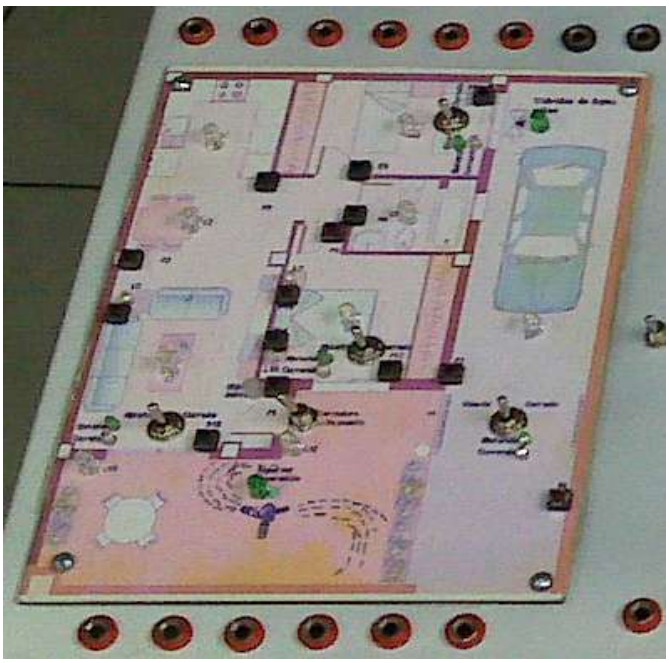
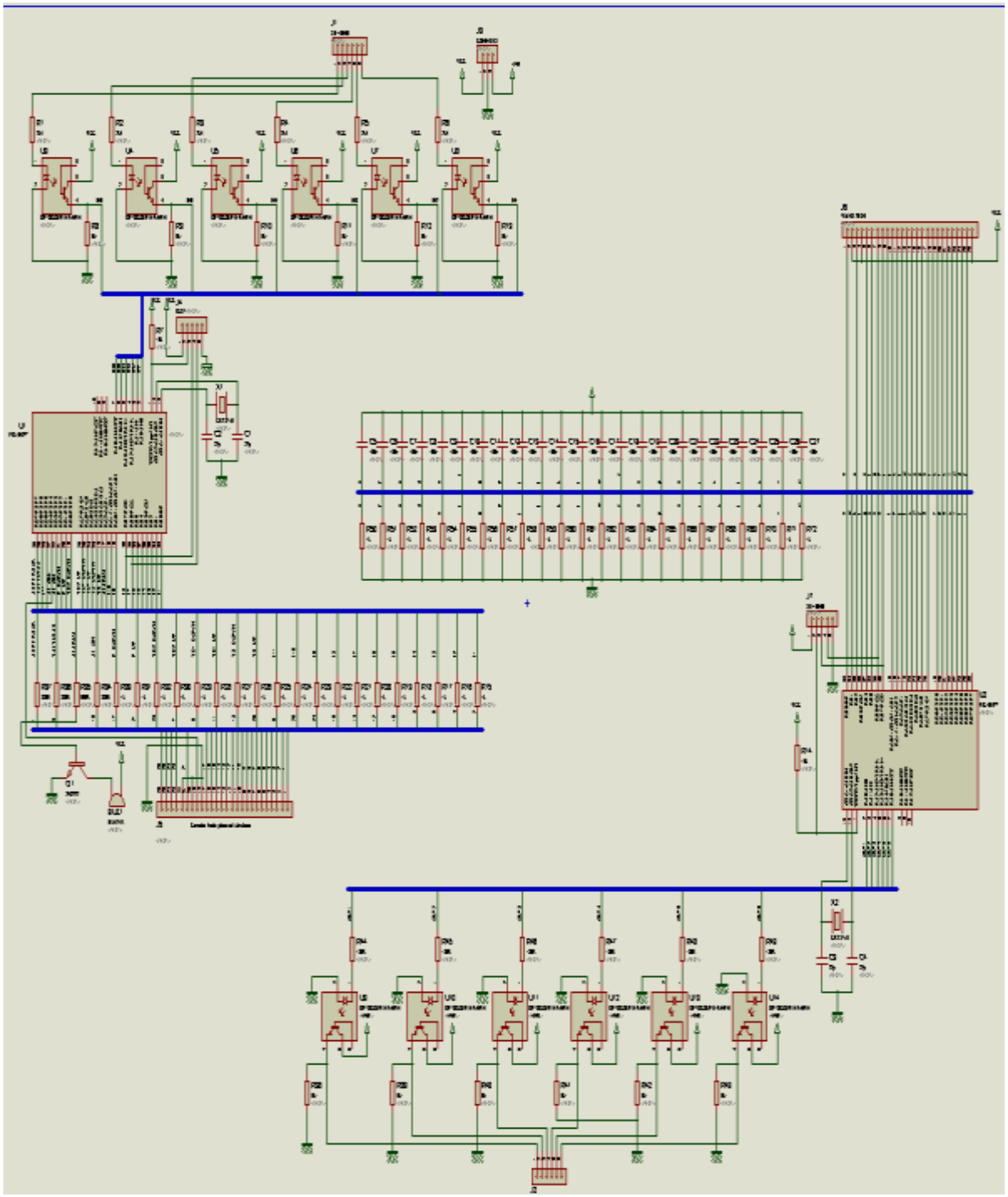


Fig. 2.4 Modulo Domótica.

El circuito que se implemento se muestra en la Fig. 2.5, donde se puede observar el uso de dos PIC 16F877, el primero para recibir los valores de entrada de los 15 pulsadores y reducirlo a 5 líneas de entrada para el PLC, 4 para indicar el pulsador presionado y la 5 línea para indicar el estado del pulsador.

El segundo PIC se utilizo para recibir las 5 salidas del PLC, 4 indican la luz o actuador a activar y la 5 el estado de la salida, ya sea 1 (encender) o 0 (apagar). Estas 5 salidas del PLC son recibidas por el PIC el cual decodifica y activa/desactiva la salida correspondiente.



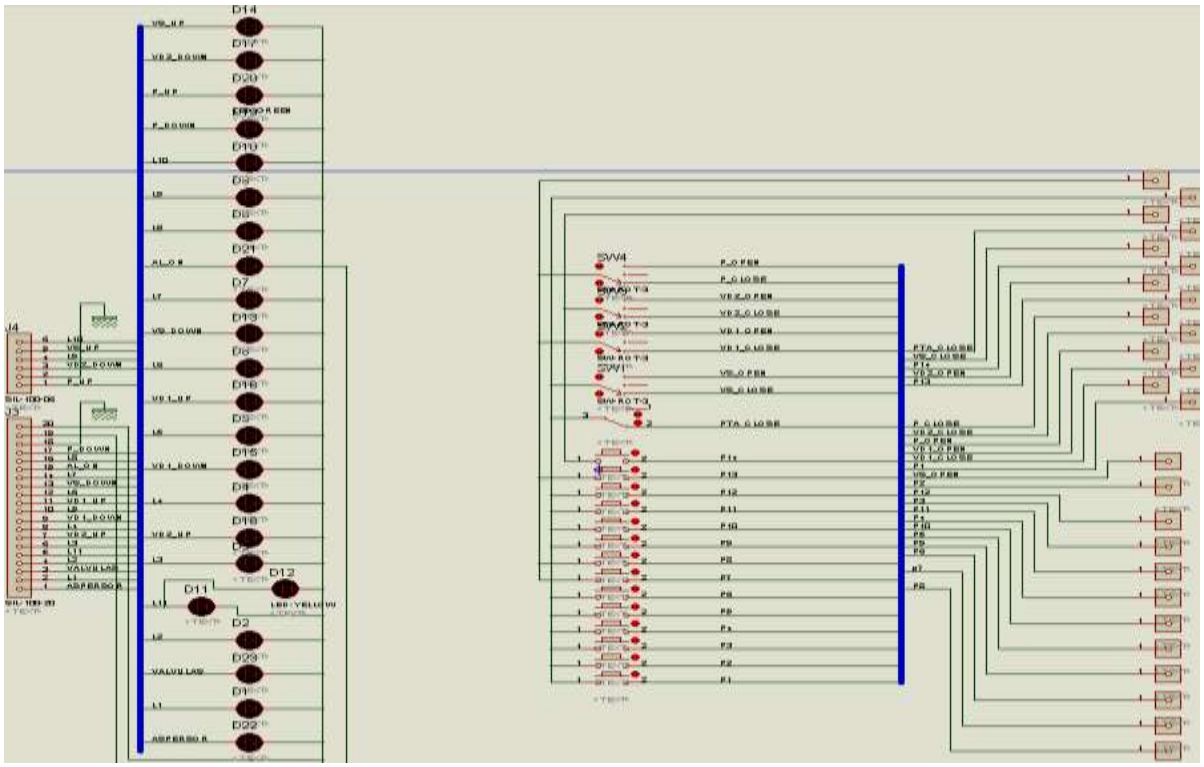


Fig. 2.5. Esquema Eléctrico Modulo Domótica.

Tal como se comento en la sección anterior el PCB se creó utilizando ARES del paquete PROTEUS, tal como se muestra en la figura siguiente:

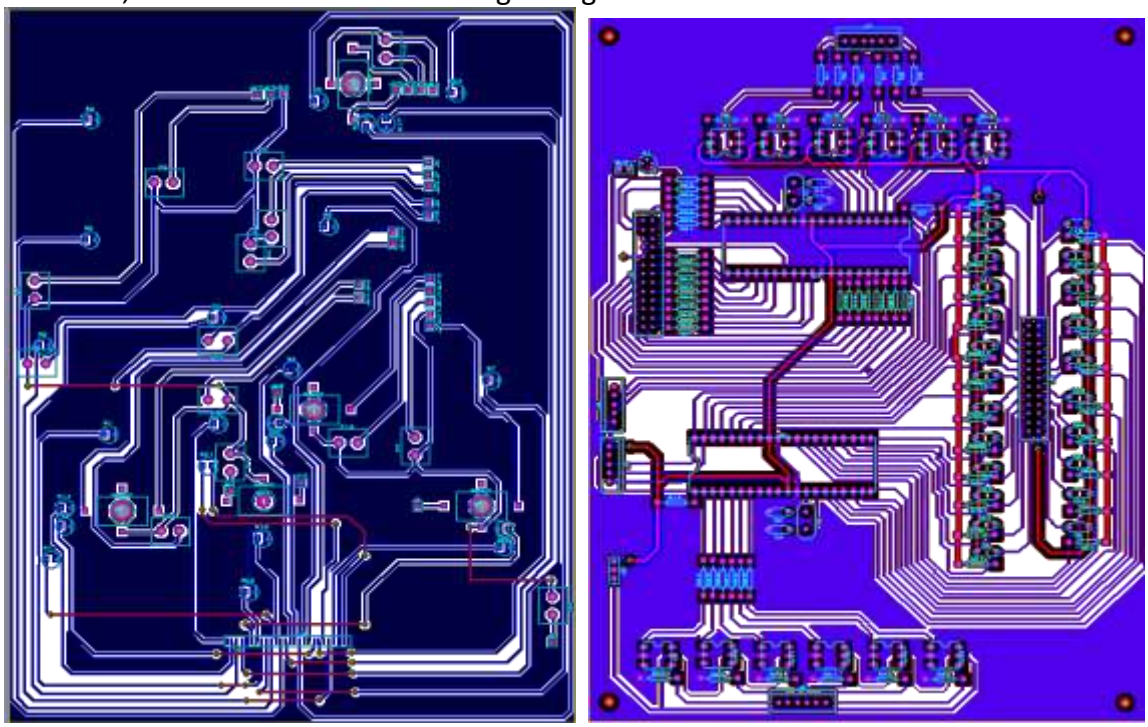


Fig. 2.6. PCB Modulo Domótica.

2.2.2 MODULO DE SEMÁFOROS

El modulo del semáforo consta de 4 semáforos, los cuales se activan simultáneamente los que pertenecen a la misma vía, es decir los semáforos norte – sur solo poseen 3 entradas, una para la luz verde , roja y amarilla ; de igual forma los semáforos de oriente-occidente posee sus propias 3 entradas, una para cada luz indicadora. La activación se debe realizar a 24V DC, es decir alimentado directamente del PLC. Tal como se muestra en la siguiente figura:

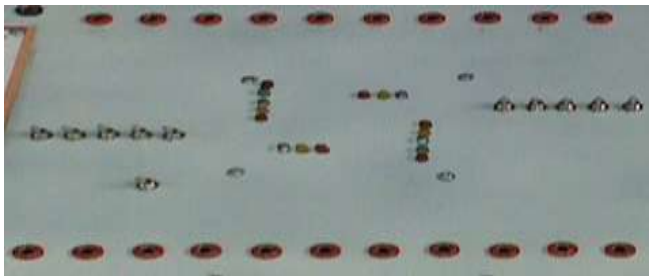


Fig. 2.7. Modulo Semáforo Vial

El modulo además consta con 10 interruptores que hacen la vez de sensores de tráfico, utilizados en la práctica de rango básico. Además, consta de un interruptor que cumple la función de activar el cruce de tren, también para la misma práctica. Las conexiones internas y circuito impreso se pueden ver en la figura 2.8.

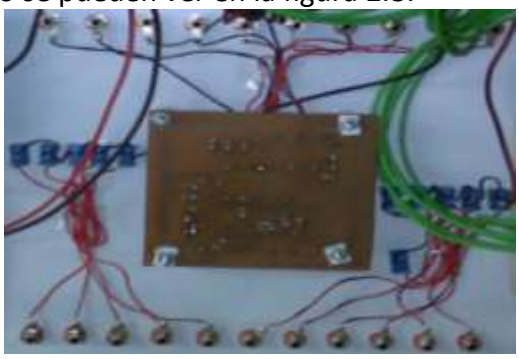


Fig. 2.8. Conexiones internas modulo semáforo Vial

2.2.3 MODULO DE TECLADO

El modulo del teclado permite reducir o codificar los 12 botones en 4 líneas a 24V DC, utilizando para este fin un PIC 16f876 el cual realiza la codificación en base a la siguiente tabla:

Numero	0	1	2	3	4	5	6	7	8	9	*	#
Código	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100

Tabla 2.1. Codificación Modulo de teclado.

El modulo del teclado se muestra en la figura 2.9, donde se puede observar que solo tiene disponibles las 4 conexiones de salidas, y una entrada de alimentación de 5V DC para el PIC.

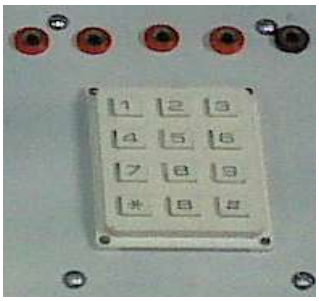


Fig. 2.9. Modulo de Teclado.

El diagrama eléctrico y circuito impreso del modulo, se pueden observar en la figura 2.10.

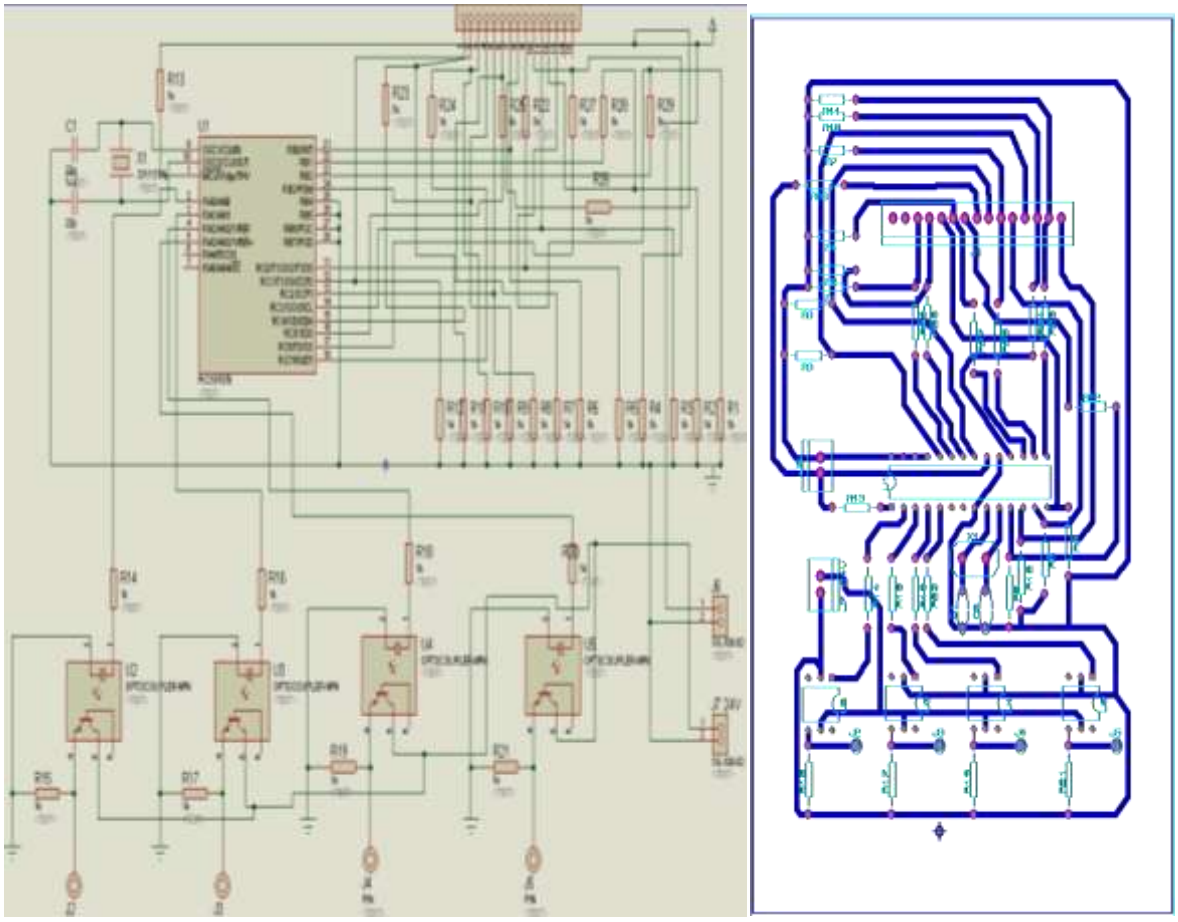


Fig. 2.10. Esquemático y PCB Modulo de Teclado.

2.2.4 MODULO DE SIRENA DE DOS TONOS

El modulo de la sirena, tal como se puede observar en la figura 2.11, consta de 3 entradas, una para alimentación de 5V DC (conexión negra), una para la activación del zumbador y una para la activación de la alarma. Es decir, si una entrada es alimentada con 24 V DC se activa un tono de la sirena, dependiendo de la entrada energizada respectivamente.

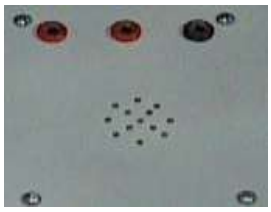


Fig. 2.11. Modulo de Sirena de dos Tonos.

El modulo ocupa internamente un IC 556 y un IC 555, con los cuales se logra modificar la frecuencia de la señal de salida para percibir dos tonos distintos, el esquematico del modulo se puede observar en la fig 2.12 mostrada a continuacion:

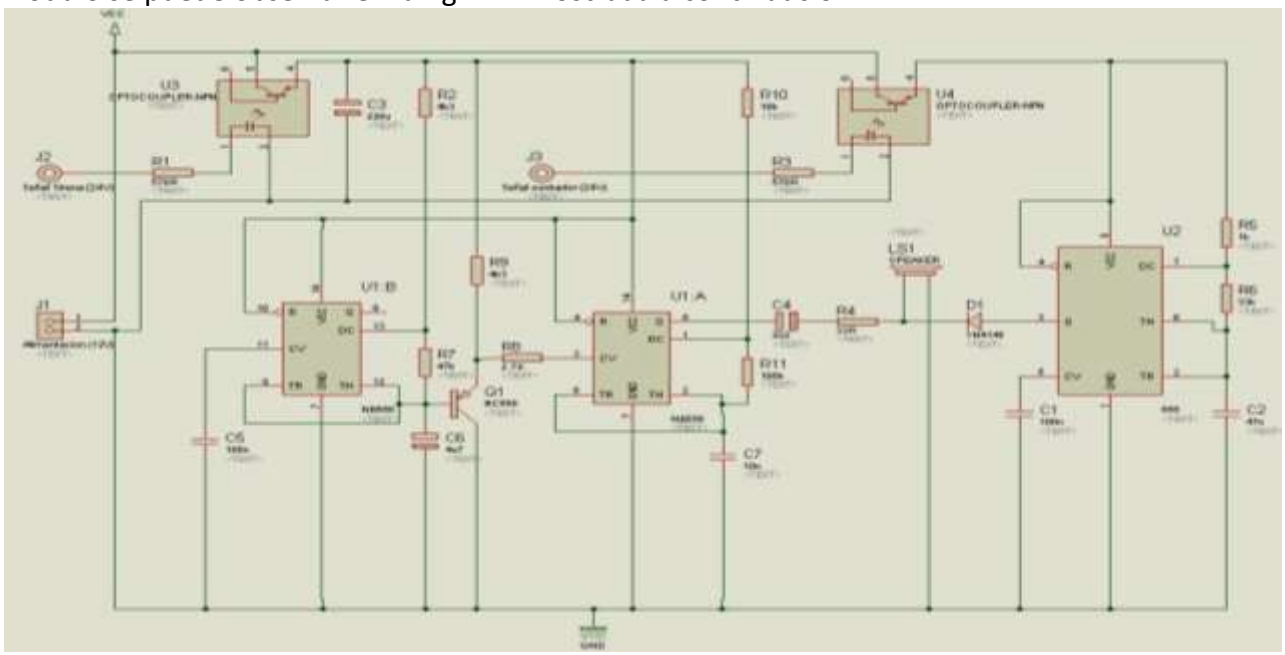


Fig. 2.12. Esquemático modulo de Sirena de dos Tonos.

El PCB del modulo, se diseño en ARES del paquete de programación PROTEUS, con el fin de obtener un diseño simple, confiable y reducido en tamaño, tal como se muestra a continuación:

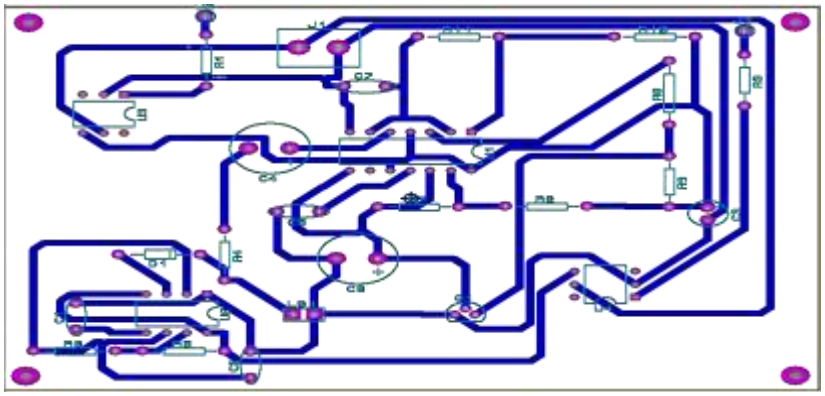


Fig. 2.13. PCB modulo de Sirena de dos Tonos.

2.2.5 MODULO DE LÁMPARAS Y PULSADORES

Este modulo está formado por 3 indicadores luminosos (Verde, Rojo y Naranja), los cuales solo necesitan la alimentación de 24V DC para funcionar, la conexión del tierra del sistema es interna y no está a disponibilidad del usuario.

Además, el modulo tiene 5 pulsadores normalmente abiertos (1 Verde y 4 Rojos) y 3 pulsadores normalmente abiertos con autoretención, los cuales pueden ser ocupados directamente con la alimentación de todo el entrenador a 110V; es decir, no necesitan una conexión de alimentación, pueden ser conectadas directamente al PLC. El modulo se puede observar en la figura 2.14.

Fig.2.14. Modulo de lámparas y Pulsadores.

2.2.6 MODULO DE MOTOR PASO A PASO

El modulo del motor paso a paso está formado por un motor de 7.5 °/paso, el cual tiene 4 bobinas que controlan su funcionamiento, el modulo se puede ver en la figura 2.15, de donde se ve que la primera entradas (negra) corresponde a la de alimentación de 5V DC, las otras cuatro entradas corresponden a las bobinas del motor PAP, las cuales deben ser energizadas con 24 V DC.

Fig. 2.15. Modulo de Motor paso a paso.

El diagrama de circuito impreso se muestra en la figura 2.16, los componentes son opto-acopladores que convierten de un valor de voltaje de 24 V a 5V con lo cual se alimenta a las bobinas del motor PAP.

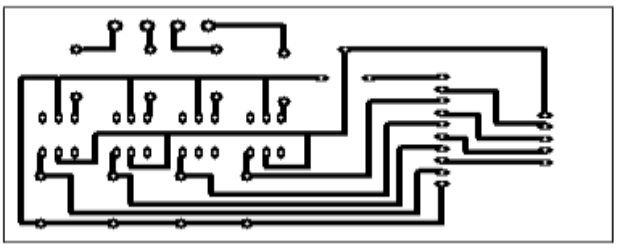


Fig. 2.16. PCB Modulo de Motor paso a paso.

2.2.7 MODULO PARA PRÁCTICAS CON SEÑALES ANALÓGICAS

El modulo para prácticas con señales analógicas, corresponde a un sensor de temperatura, el cual como se ve en la figura 2.17, necesita la conexión de un RTD y un voltaje de referencia de +13.5 V y – 13.5 V.

FIG 2.17. Modulo de Señal Analógica.

El circuito eléctrico implementado para convertir la señal analógica de temperatura a valores de voltaje entre el rango de 0-10V, se muestra a continuación:

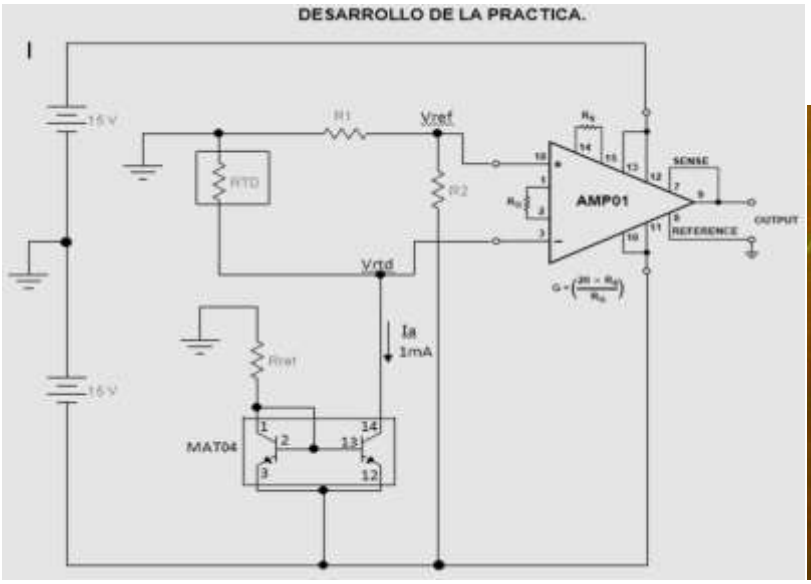


Fig. 2.18. Esquemático Modulo de Señal Analógica.

El diagrama de circuito impreso del modulo, se puede observar en la figura 2.19.

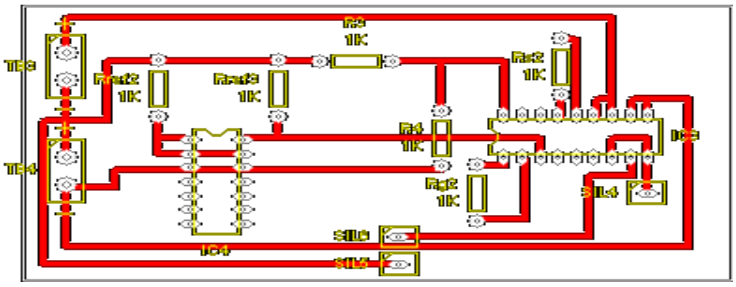


Fig. 2.19. PCB Modulo de Señal Analógica.

2.2.8 MODULO DE MOTORES DC

Este modulo está formado por 3 motores DC a 24 V, además de un sensor óptico por cada motor el cual detecta a través de una muesca del disco del motor, el momento en que se completa una vuelta. La conexión física en el entrenador de izquierda a derecha, los 3 primeros son las alimentaciones a 24 V para cada motor respectivamente; las siguientes 3 entradas del modulo son los estados de los sensores ópticos, la última entrada corresponde a la alimentación de 5 V DC para el modulo.

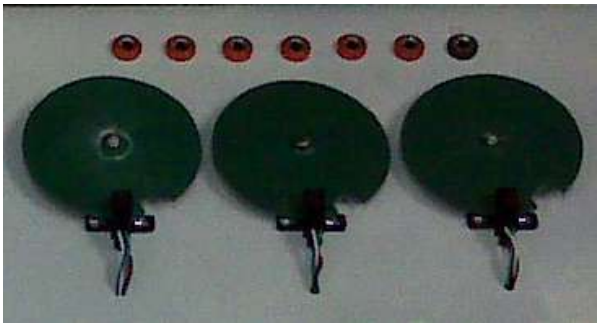


Fig. 2.20. Modulo de Motores DC.

2.2.9 MODULO DE PANTALLA DE PUNTOS

El modulo de la pantalla de puntos está conformada por 8 pantallas matriciales de puntos y 9 entradas para el funcionamiento del mismo. En la figura 2.21 se puede observar la función de cada una de las entradas del modulo de pantalla de puntos.

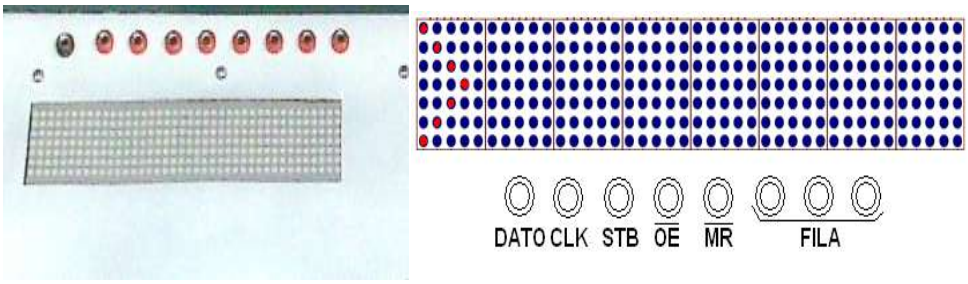
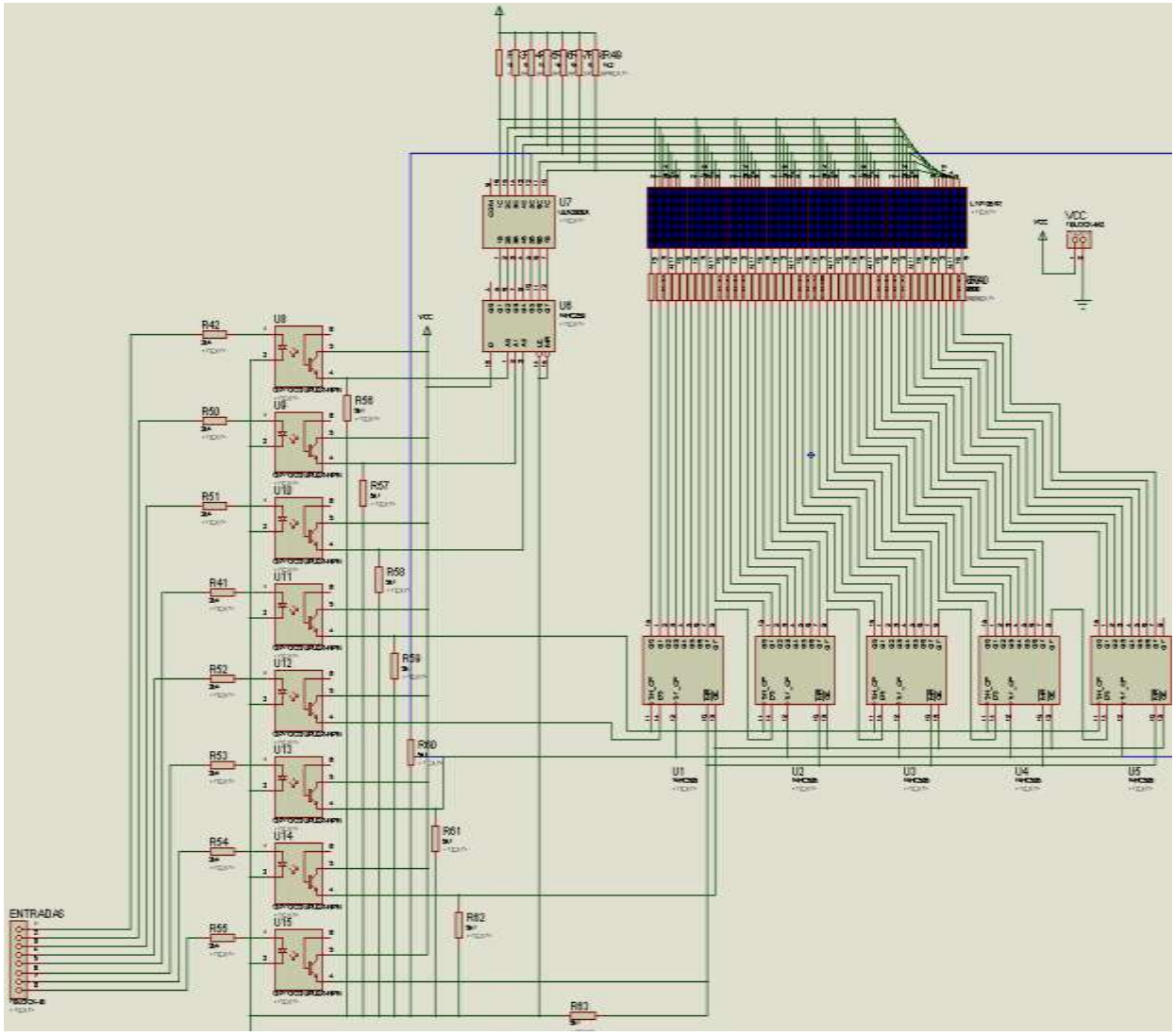


Fig. 2.21. Modulo de Pantalla de Puntos.

La pantalla de puntos funciona de la siguiente manera, en los bornes indicados por ‘FILA’ se debe de especificar el número de fila en binario a la cual se le enviarán los datos de forma secuencial; el estado de cada punto de la matriz se introduce de forma secuencial por el terminal ‘DATO’ empezando desde el ultimo que será mostrado, primero se debe colocar un nivel lógico en el borne ‘DATO’ y luego se produce un flanco de subida en el terminal ‘CLK’ para que el valor sea introducido, se debe repetir esto hasta el último dato requerido; un pulso en el terminal ‘STB’ carga los datos introducidos en los latch de salida y luego estos se deben mostrar colocando un nivel lógico bajo en el terminal ‘OE’, el terminal ‘MR’ siempre debe estar en nivel lógico alto a menos que se quiera restablecer todos los latch de entrada.

En el esquema eléctrico se puede observar que se utilizaron 4 IC, los cuales son registros de desplazamiento con latch de entrada y salida, que logran que el funcionamiento del modulo sea como el antes descrito. El esquema de circuito impreso se muestra en la figura 2.22, para completar la información del modulo.



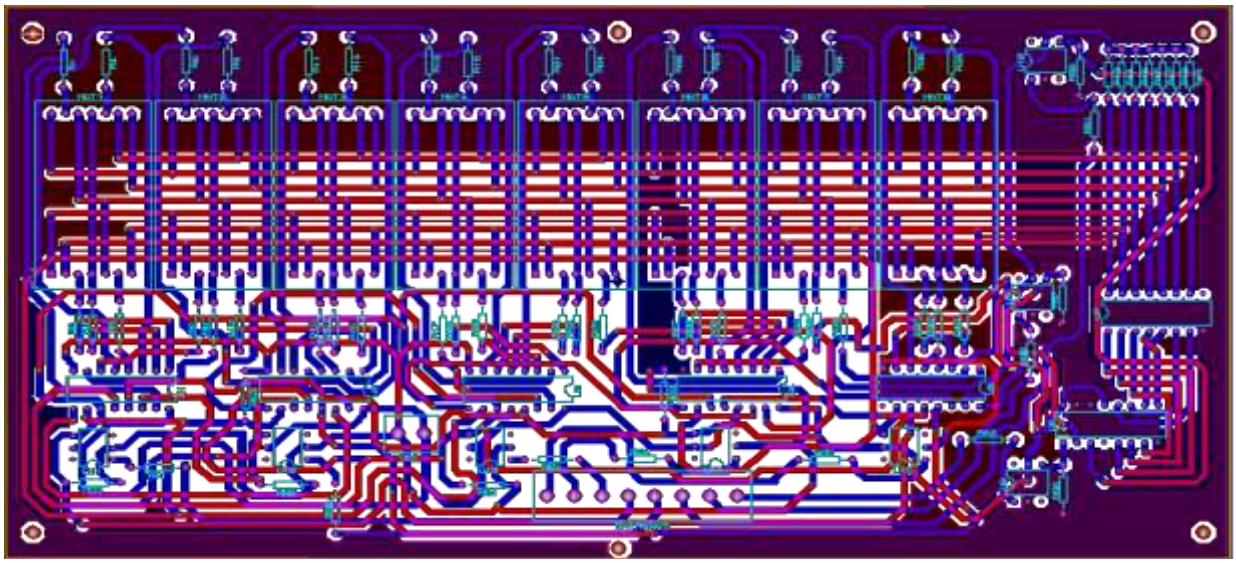


Fig. 2.22. Esquemático y PCB de modulo de Pantalla de Puntos.

2.2.10 MODULO DE DISPLAY DE 3 DÍGITOS

El modulo de display de 3 dígitos está formado por 3 display de 7 segmentos, 4 entradas; las cuales de izquierda a derecha son reset (activo alto), Reloj (flanco positivo), Dato serial y alimentación de 5V DC. Tal como se muestra en la figura 2.23.

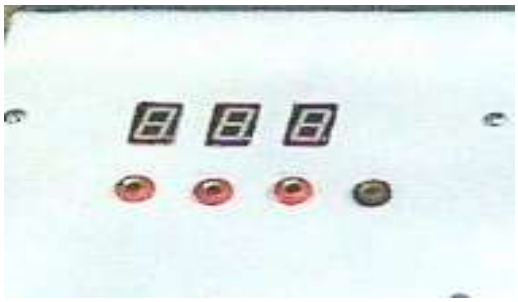


Fig. 2.23. Modulo de Display de 3 Dígitos.

En cuanto al esquema eléctrico se puede observar que se utilizaron dos IC uno es el PIC16F84A y el otro es un decoder 74145 con salidas activas bajas. El microcontrolador es el cerebro de todo el sistema y su funcionamiento es simple; al energizar el circuito este muestra en los display el valor 000 constantemente, luego si se desea cambiar el valor mostrado en los display se debe de introducir datos con su respectivo pulso de reloj por los pines RA0(Dato) y RA1(Reloj), al empezar a introducir datos los display se apagan y el PIC solo se queda esperando la recepción de 24 pulsos de reloj, al pasar los 24 pulsos se vuelven a encender los display pero ahora con los dígitos introducidos.

Debido a que solo se cuentan con 13 pines de entrada/ salida en el PIC, la manipulación de los 24 segmentos de los display se realiza por conmutación de digito, cada display está

conectado al puerto B del pic (un segmento por cada pin) y el común de cada display va conectado a una salida del 74145 para decidir a que display se están enviando los datos.
Por ejemplo, si se desea mostrar el numero 365 se envía el valor 5 codificado en BCD de 7 segmentos al puerto B del PIC luego se estable un 1 binario entre los pines RA2 y RA3 del PIC las cuales están conectadas a las entradas del decoder encendiendo el primer display, para encender el segundo display limpiamos el puerto B y cambiamos de display enviando 2 en binario entre los pines RA2 y RA3 del PIC luego se envía el numero 6 al puerto B y este es mostrado en el segundo display, es el mismo procedimiento para el tercer digito.

Para cambiar de dígitos, se debe de enviar en forma serie secuencial los dígitos codificados en BCD de 7 segmentos a la entrada RA0 del PIC empezando desde el segmento 'A' para cada digito, cada pulso de reloj es un dato introducido.
El esquema eléctrico y el diagrama de circuito impreso se muestran en la figura 2.24.

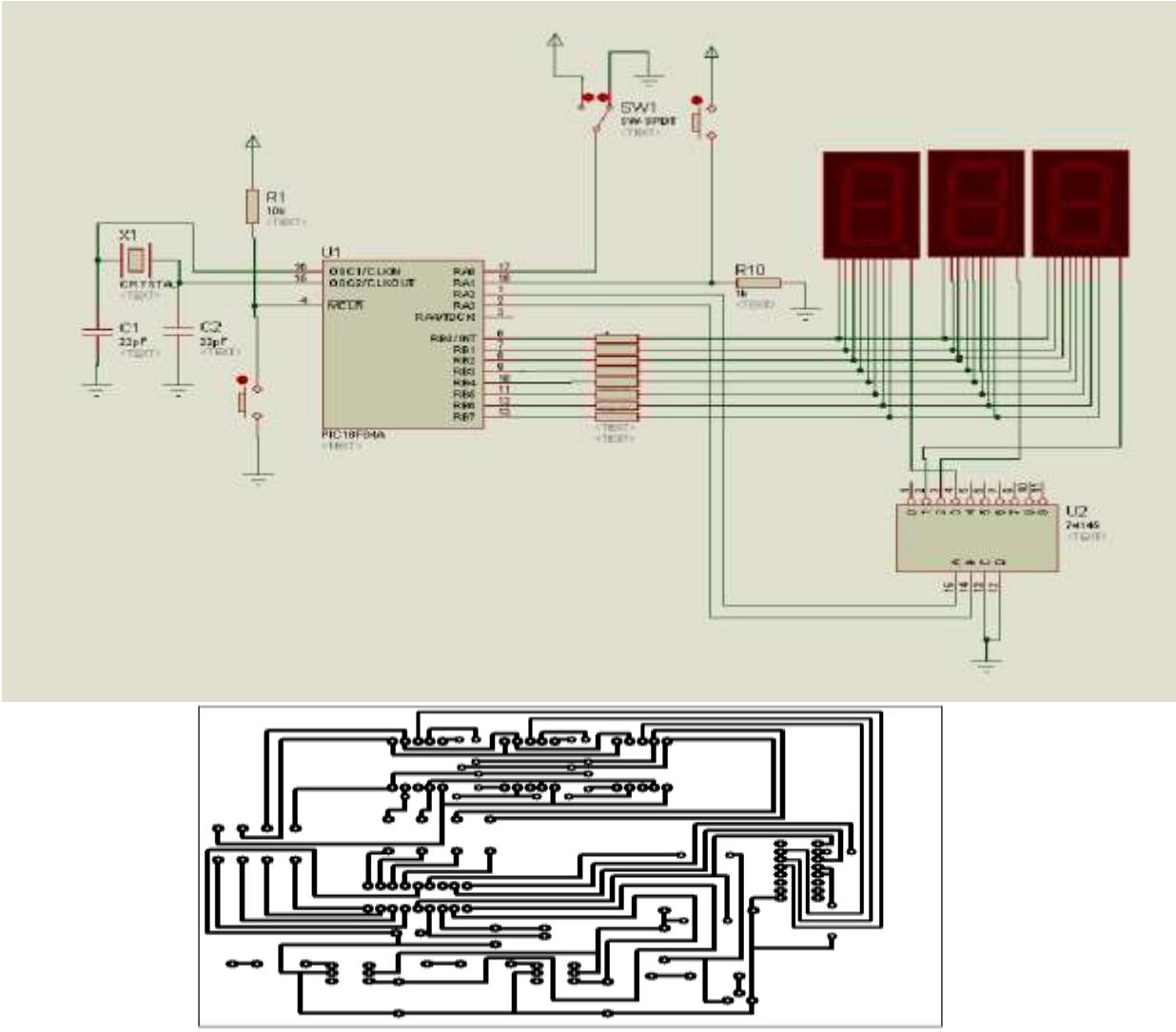


Fig. 2.24. Esquemático y PCB de modulo de display de 3 dígitos.

2.3 DISEÑO Y CONSTRUCCIÓN DEL CHASIS.

2.3.1 ETAPA DE DISEÑO

El diseño del chasis o estructura externa del entrenador, se realizo por medio de computadora, utilizando el software POWERSHAPE, donde se definió la forma y dimensiones del mismo, quedado el diseño como se muestra en la figura 2.25.

Dado que solo se contaba con la forma y alguna de las dimensiones POWERSHAPE fue esencial para que de una manera más fácil y exacta se pudieran obtener las plantillas de las piezas que se iban a cortar de la lámina de acero.

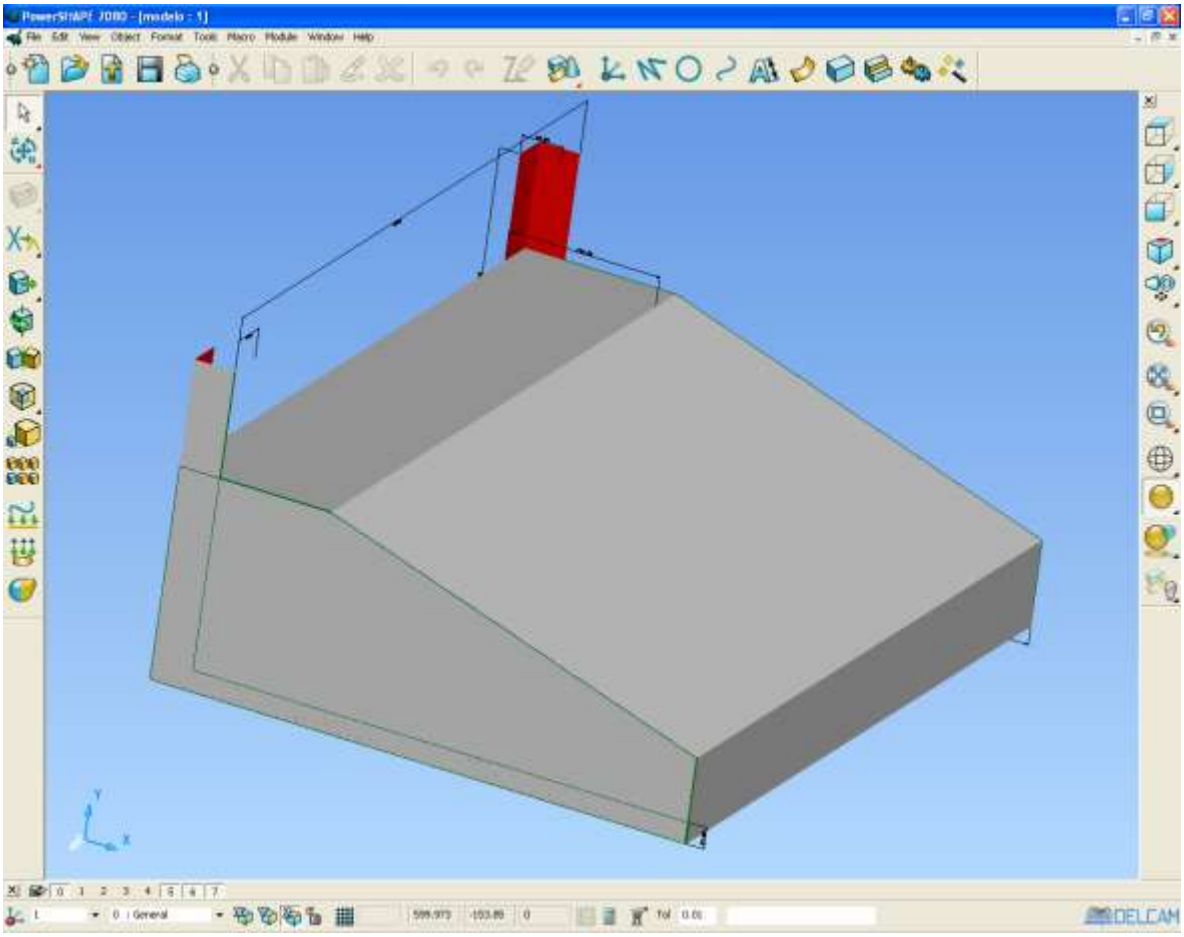


Fig. 2.25. Diseño del Chasis.

2.3.2 ETAPA DE CONSTRUCCIÓN

La siguiente fase luego de tener el diseño, fue la etapa de construcción del mismo, donde se realizaron distintas actividades entre ellas:

- Medición y Marcado de Laminas

Fig. 2.26. Marcado de la lámina.

- Cortado de Piezas de Lamina

Fig. 2.27. Cortado de la lámina.

- Marcado de Agujeros y Espacios de Apertura

Fig. 2.28. Distribución de elementos.

- Taladrado de Lamina

Fig. 2.29. Perforación de la lámina.

2.3.3 ETAPA DE ENSAMBLE

La etapa de ensamble conto con diversas actividades entre ellas están:

- Doblado de Piezas
- Tratamiento de Lijado
- Aplicación de Pintura

Fig. 2.30. Ensamble del chasis.

2.3.4 CABLEADO Y ARMADO

Como una última etapa en la construcción del entrenador, se tuvo que colocar todas las tarjetas y componentes del mismo en su posición, sujetarlo con tornillos y soldar todos los cables necesarios para los conectores hembras de cada modulo.

Fig. 2.31. Cableado de módulos y el PLC.

2.3.5 ETAPA DE PRUEBAS

La última etapa de la construcción del chasis, fue la realización de todas las pruebas necesarias para el funcionamiento completo de todos los módulos que forman parte del entrenador.

Fig. 2.32. Prueba del entrenador.

2.4 COSTOS DE MATERIALES Y EQUIPO UTILIZADO PARA LA CONSTRUCCIÓN DEL ENTRENADOR

En La siguiente tabla se muestran los equipos y elementos electrónicos utilizados para la construcción de los módulos:

Producto	Cantidad
Resistencias de 180 ohm	40
Resistencias de 1k	26
Resistencia 2.4k	23
Resistencias 5k	7
Resistencia 5.1k	35
Resistencias de 10k	3
Capacitores 22pf	6
Capacitores 0.1uf	23
Led 5mm color claro	9
Led 5mm color Rojo	6
Led 5mm color verde	8
Led 5mm color Amarillo	4
Led 3mm color claro	3
Led 3mm color verde	4
Led 3mm color rojo	5
Lámpara 24V color claro	1
Lámpara 24V color rojo	1
Lámpara 24V color verde	1
Display 7-segmentos cátodo común	3
Display con matriz de puntos 5x7	8
Transistor 2N2222	3
Diodo 1N4002	3
Sensor óptico CNA1007H	3
Opto acoplador KPC817 (ó 4N25)	24
IC 74LS164	4
IC 74LS259	1
IC 74HC595	5
IC ULN2003A	1
IC PIC16F84	1
IC PIC16F877	2
AMP01	1
MAT04	1
C1458	2

PU4119	1
Oscilador 4MHz	3
Bocina	1
MiniPulsadores	14
Interruptores on-off-on	4
Interruptores off-on	12
Push button a 24V	8
Teclado matricial 12 teclas	1
Motores DC 24V con caja reductora	3
Motor paso a paso 5V	1
Headers 5 pines macho	1
Headers doble fila 23 pines Macho	2
Housing 23 pines	2
Conectores hembra para terminal de banana Rojos	100
Conectores hembra para terminal de banana Negros	28
Tableta de cobre doble cara 11.8"x11.8"	1
Tableta de cobre una cara 11.8"x11.8"	2
Voltímetros DC 30V	2
Cable AWG #14	50 mts.
Conductor dúplex #26	20 mts.
Interruptor 220VAC 10A	1
Conector macho para alimentación de 120VAC	1
Extensión para alimentación de 120V	1
Porta fusible montaje en chasis	1
Fusible 220V 4.5A	1
Hojas Papel couche	30
Onzas de percloruro	9

Tabla 2.2. Lista de elementos electrónicos y varios para la construcción de los módulos.

Los materiales necesarios para la construcción del chasis son listados en la siguiente tabla:

Producto	Cantidad
Lamina de hierro de 3/64''	1
Riel DIN	1 mts.
Tornillo Goloso 1/2'' x 1/8 ''	50
Tornillo galvanizado 2''x 1/8''	60
Tuercas para tornillo galvanizado de 2''x1/8''	180
Pintura en aerosol	3

Tabla 2.3. Lista de materiales utilizados para la construcción del chasis.

La siguiente tabla muestra los productos siemens utilizados por el entrenador:

Cantidad	EQUIPO	Descripción	Código de catalogo
1	**SIMATIC S7-200**	CPU 224xp ap. compacto, alimentación DC, 14 ED dc/10 SD dc, 2 EA, 1 SA, /16 kb progr./10 kb datos, 2 puertos PPI/Freeport.	6ES7214-2AD23-0XB0
1	**Cable PC/PPI**	Cable pc/ppi MM Multimaster, para conexión de s7-200 a interfaz serie del pc, soporta freeport y modem GSM	6ES7901-3CB30-0XA0
1	**LOGO!POWER**	LOGO! power 24 v stabilized power supply input: 100-240 v ac output: 24 v dc/4 A	6EP1332-1SH51

Tabla 2.4. Equipos Siemens utilizados.

La siguiente tabla resume los costos en los que se incurrió para la construcción del entrenador:

Descripción	Monto ($)
Construcción y cableado de módulos	286.81
Construcción del chasis	36.00
Productos siemens	530.00
Transporte de equipo y elementos electrónicos	41.00
Total	893.81

Tabla. 2.5. Costo total de construcción del entrenador.

CONCLUSIONES CAPITULO 2.

Ahora con el entrenador para el S7-200 construido, se tienen las herramientas para que en el laboratorio de automatización se capacite a los alumnos en esta área. Este entrenador esta desarrollado pensando en todos los usos que el alumno le puede dar por lo cual cada modulo es independiente y puede ser utilizado con otros dispositivos externos si este así lo desea.

CAPITULO 3

HERRAMIENTA DE PROTOTIPADO RAPIDO CON eZdsp F2812

CAPITULO 3
HERRAMIENTA DE PROTOTIPADO RAPIDO CON eZdsp F2812.

Una herramienta de prototipado rápido es un sistema compuesto por una PC o sistema HMI y una tarjeta de procesamiento de señales (DSP), con la cual se pueden realizar experimentos físicos y obtener datos en tiempo real, esta herramienta permite que, por medio de algún software de computo, se pueda rediseñar el sistema sin necesidad de realizar largos procesos de conexión y desconexión de dispositivos.

En este capítulo se ha realizado una modulo controlador digital, el cual, por medio de una tarjeta eZdsp F2812, permite realizar experimentos de control de motores DC y/ó servomecanismos de posición o velocidad en tiempo real. Se utiliza el software Matlab Simulink para diseñar los sistemas y visualizar los datos. Este Modulo cuenta con cinco entradas analógicas y cuatro salidas analógicas, estas permiten realizar una variedad de experimentos, dejando los diversos usos a la imaginación del operario.

Con esta herramienta se han realizado dos controladores PID, uno de velocidad y el otro de posición, estos tienen como objetivo, que los alumnos de la EIE aprendan los conceptos básicos sobre herramientas de prototipado rápido, el diseño de sistemas en Simulink y el dominio de controladores PID en tiempo discreto.

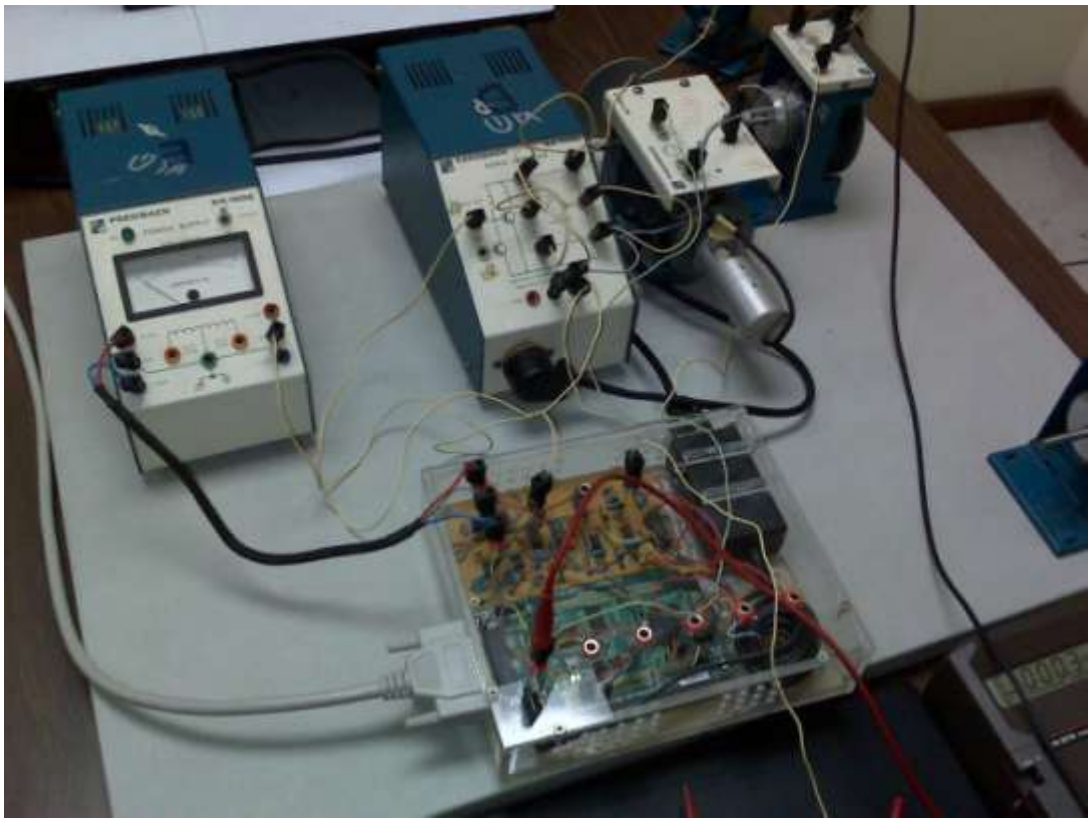


Fig. 3.1. Equipo necesario para controladores PID de velocidad y posición.

3.1 INFORMACION GENERAL

3.1.1 eZdsp F2812

La eZdsp F2812 es una tarjeta independiente de desarrollo que permite examinar a la DSP (Digital Signal Processor) TMS320F2812. Además es una excelente plataforma para el desarrollo de software con este procesador.

La eZdsp F2812 es una tarjeta de circuito impreso multicapa de 5.25 x 3.0 pulgadas, alimentada por una fuente de poder externa de 5 V, incluida en el equipo de laboratorio, la tarjeta requiere 500 mA, la energía es alimentada a través del conector P6.

La DSP TMS320F2812 de Texas Instrument posee las siguientes características:

- Generación: Controladores TMS320F281X.
- Velocidad de reloj de 150 MHz.
- Memoria: 256 Kb (expandible hasta 1Mb).
- Señales PWM (Pulse Widht Modulation): 16 canales.
- Conversión Análoga a Digital: 16 canales, resolución de 12 bit, tiempo de conversión de 80 ns.
- Pines Entrada/Salida: Hasta 56.
- Niveles de señal: (0, 3.3 V), (0, 3V) en pines ADC.

La eZdsp F2812 posee las siguientes características:

- TMS320F2812 Digital Signal Processor.
- Velocidad de operación de 150 MIPS (Millones de instrucciones por segundo).
- 18K palabras en memoria RAM.
- 128K palabras en memoria Flash.
- Reloj de 30 Mhz.
- 2 Conectores de Expansión (Análogo, I/O).
- Controlador JTAG IEEE 1149.1.
- Operación única a 5 V.
- Drivers para la aplicación TI F28XX Code Composer Studio.

La figura 3.2 muestra el diagrama de bloques para la configuración básica de la eZdsp F2812.

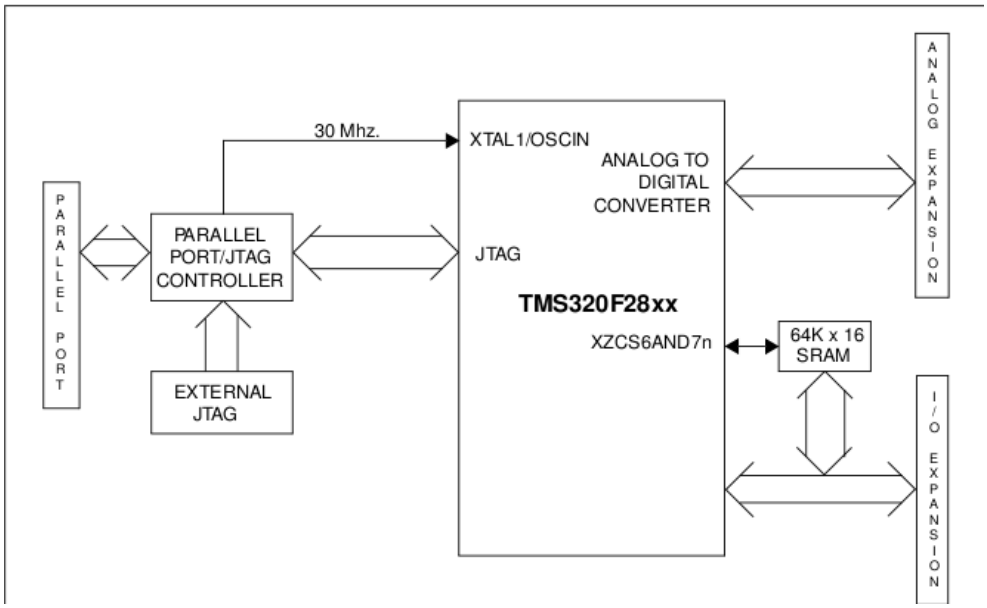


Fig. 3.2. Diagrama de bloques eZdsp F2812.

La eZdsp F2812 está compuesta de 4 bloques lógicos principales:

- Conector Interfaz Análoga
- Conector Interfaz I/O
- Interfaz JTAG
- Controlador Interfaz de Puerto Paralelo

La DSP se conecta a la computadora anfitrión a través de un puerto paralelo estándar usando un interfaz JTAG (Joint Test Action Group, IEEE 1149.1). La F281x ofrece JTAG en tiempo real, una característica que no está disponible en otros procesadores de la serie C2000. JTAG en tiempo real permite al usuario la opción de modificar el contenido de la memoria y periféricos mientras el procesador está corriendo. Debido a que los programas son a través de Simulink no se tomara ventaja de esta característica.

La figura 3.3 muestra la disposición de la tarjeta F2812 para ambas versiones, la que posee base de conexión y la que no.

La diferencia principal en la figura radica en que la tarjeta F2812 inferior no posee base para colocar el chip TMS320F2812, mientras que la tarjeta superior si posee dicha base, lo que facilita la instalación del chip.

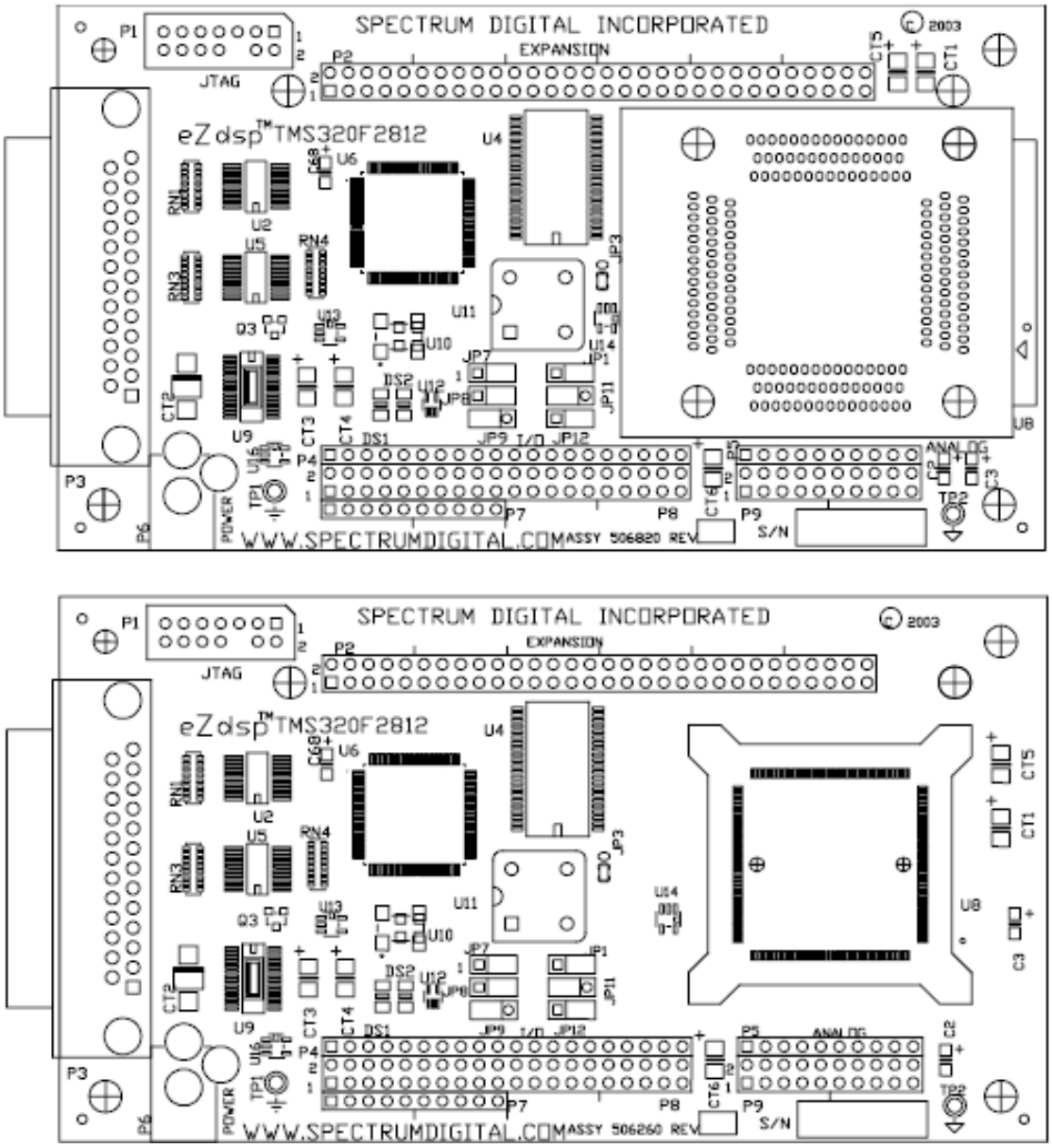


Fig. 3.3. Disposición física DSP F2812

La tarjeta eZdsp incluye las siguientes capacidades de memoria:

- ✓ 128 x 16 Flash.
- ✓ 2 bloques de 4K x 16 RAM (single Access RAM).
- ✓ 1 bloque de 8K x 16 SARAM.
- ✓ 2 bloques de 1K x16 SARAM.

La configuración del mapa de memoria de la eZdsp F2812 se muestra en la figura 3.4.

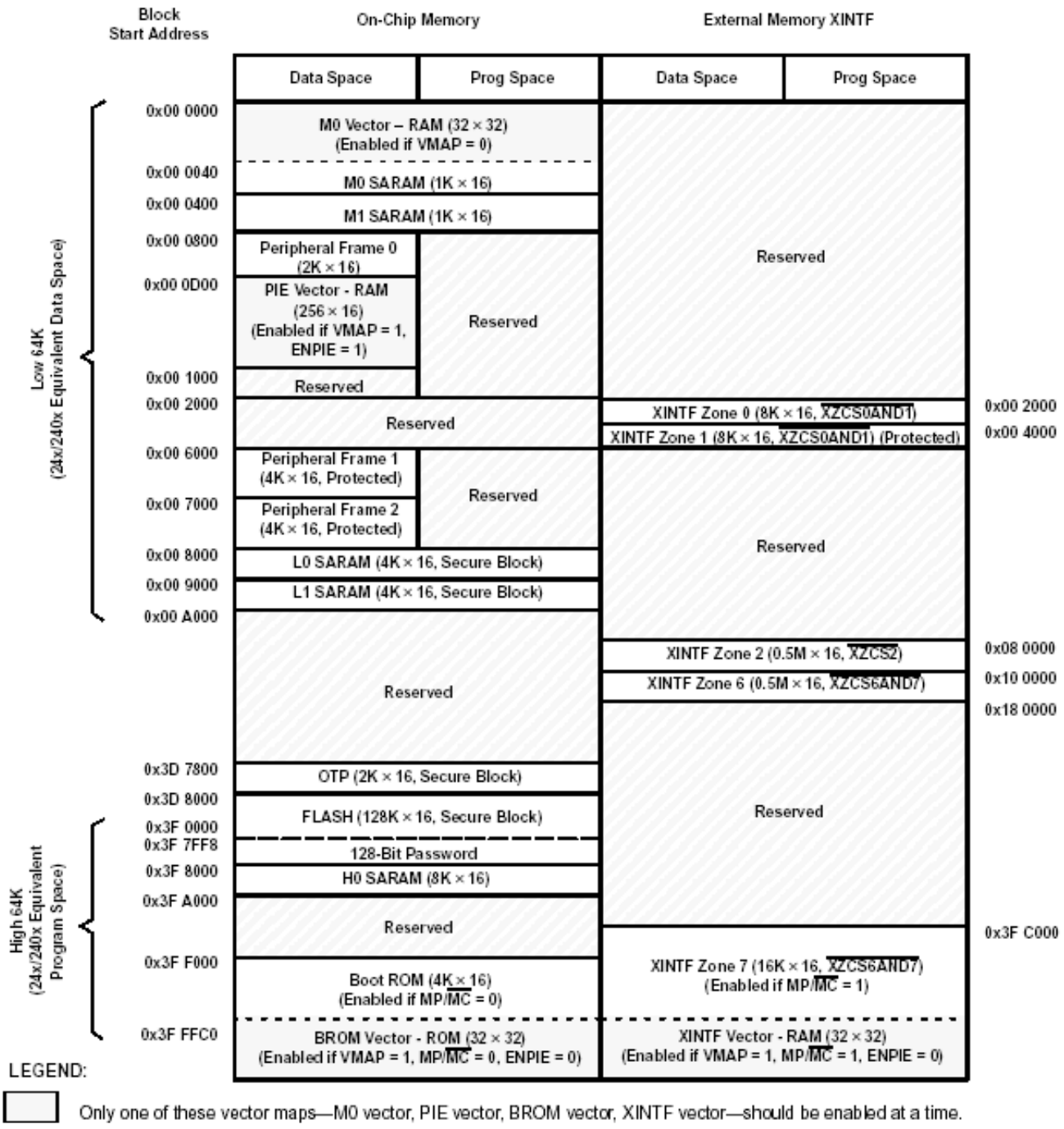


Fig. 3.4. Configuración de mapa de memoria.

La tarjeta eZdsp F2812 tiene 6 conectores tal como se muestra en la figura 3.5. El pin 1 de cada uno se identifica por la forma cuadrada de la pista para soldar. La Tabla 3.1 muestra la identificación de cada uno de los conectores.

Connector	Function
P1	JTAG Interface
P2	Expansion
P3	Parallel Port/JTAG Controller Interface
P4/P8/P7	I/O Interface
P5/P9	Analog Interface
P6	Power Connector

TABLA 3.1. Conectores de la eZdsp F2812.

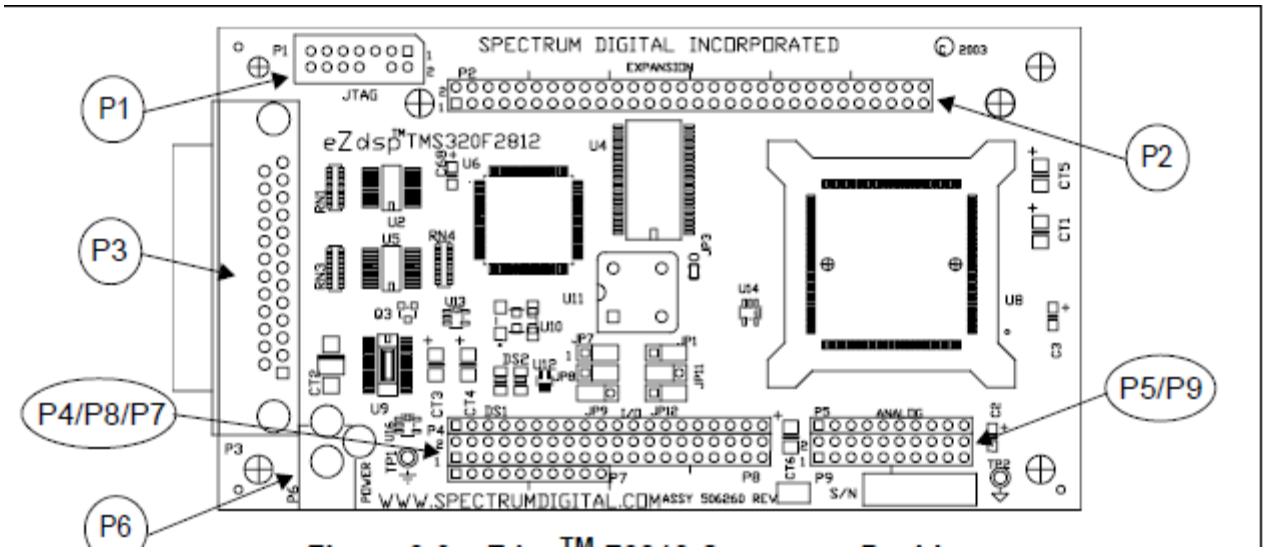


Fig. 3.5. Disposición física conectores eZdsp F2812

Los conectores usados para los controladores de velocidad y posición realizados son P3, P4/P8/P7, P5/P9 y P6 los cuales se describen a continuación:

- **P3, Interface Puerto Paralelo/JTAG**
  La tarjeta eZdsp F2812 usa un dispositivo de interface de puerto paralelo. Este soporta comunicaciones ECP, EPP y SPP8/bidireccional. Este dispositivo tiene acceso directo a la interfaz integrada JTAG.

- **P4/P8/P7, I/O interface**
  Este conector dispone de las señales de I/O desde la DSP. La disposición de estas conexiones se muestra en la figura 3.6.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	P4
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	P8
1	2	3	4	5	6	7	8	9	10	P7										

Fig. 3.6. Disposición conector P4/P8/P7

La descripción de cada pin del conector P4/P8 se muestra en la tabla 3.2. La descripción para el conector P7 se muestra en la tabla 3.3.

P4 Pin #	P4 Signal	P8 Pin #	P8 Signal	P8 Pin #	P8 Signal
1	+3.3V/+5V/NC *	1	+3.3V/+5V/NC *	2	+5 Volts
2	XINT2/ADCSOC	3	SCITXDA	4	SCIRXDA
3	MCLKXA	5	XINT1n/XBIOn	6	CAP1/QEP1
4	MCLKRA	7	CAP2/QEP2	8	CAP3/QEPI1
5	MFSXA	9	PWM1	10	PWM2
6	MFSRA	11	PWM3	12	PWM4
7	MDXA	13	PWM5	14	PWM6
8	MDRA	15	T1PWM/T1CMP	16	T2PWM/T2CMP
9	No connect	17	TDIRA	18	TCLKINA
10	GND	19	GND	20	GND
11	CAP5/QEP4	21	No connect	22	XINT1N/XBIOn
12	CAP6/QEPI2	23	SPISIMOA	24	SPISOMIA
13	T3PWM/T3CMP	25	SPICLKA	26	SPISTEA
14	T4PWM/T4CMP	27	CANTXA	28	CANRXA
15	TDIRB	29	XCLKOUT	30	PWM7
16	TCLKINB	31	PWM8	32	PWM9
17	XF/XPLLDISn	33	PWM10	34	PWM11
18	SCITXDB	35	PWM12	36	CAP4/QEP3
19	SCIRXDB	37	T1CTRIP/PDPINTAn	38	T3CTRIP/PDPINTBn
20	GND	39	GND	40	GND

TABLA 3.2. Descripción conector P4/P8

P7 Pin #	P7 Signal
1	C1TRIPn
2	C2TRIPn
3	C3TRIPn
4	T2CTRIPn/EVASOCn
5	C4TRIPn
6	C5TRIPn
7	C6TRIPn
8	T4CTRIPn/EVBSOCn
9	No connect
10	GND

TABLA 3.3. Descripción conector P7.

- **P5/P9 Interface.**

La posición de los 30 pines del conector P5/P9 se muestran a continuación en la figura 3.7. La descripción de cada pin se muestra en la tabla 3.4.

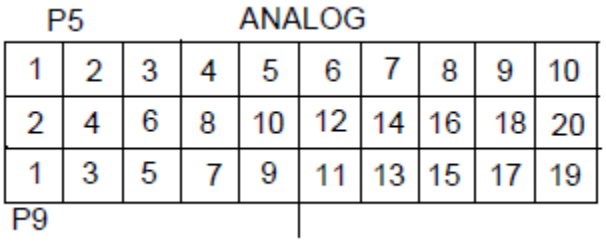


Fig. 3.7. Disposición conector P5/P9

P5 Pin #	Signal	P9 Pin #	Signal	P9 Pin #	Signal
1	ADCINB0	1	GND	2	ADCINA0
2	ADCINB1	3	GND	4	ADCINA1
3	ADCINB2	5	GND	6	ADCINA2
4	ADCINB3	7	GND	8	ADCINA3
5	ADCINB4	9	GND	10	ADCINA4
6	ADCINB5	11	GND	12	ADCINA5
7	ADCINB6	13	GND	14	ADCINA6
8	ADCINB7	15	GND	16	ADCINA7
9	ADCREFM	17	GND	18	VREFLO *
10	ADCREFP	19	GND	20	No connect

* Connect VREFLO to AGND or VREFLO of target system for proper ADC operation.

TABLA 3.4. Descripción conector P5/P9.

La tarjeta esta equipada de 16 canales de conversión de análogo a digital. Cada canal tiene una resolución de 12 bit y un tiempo de conversión de 80 ns. La máxima frecuencia de muestreo es de 25 Mhz. La señal de entrada debe estar entre 0-3V. Sin embargo, a través de un circuito dedicado de interfaz se logra el rango de -15 V a 15 V.

- **P6.**
  Este conector es utilizado para colocar la alimentación de la tarjeta.

3.1.2 GENERALIDADES MATLAB & SIMULINK.

3.1.2.1 MATLAB.

Matlab es un programa de gran aceptación en ingeniería destinado a realizar cálculos técnicos científicos y de propósito general. En él se integran operaciones de cálculo, visualización y programación, donde la interacción con el usuario emplea una notación matemática clásica.

Los usos y aplicaciones típicos de Matlab son:

- Matemáticas y cálculo.
- Desarrollo de algoritmos.
- Adquisición de datos.
- Modelado, simulación y prototipado.
- Análisis y procesado de datos.
- Gráficos científicos y de ingeniería.
- Desarrollo de aplicaciones.

Matlab consta de cuatro partes fundamentales:

1. *Entorno de desarrollo.* Se trata de un conjunto de utilidades que permiten el uso de funciones Matlab y ficheros en general. Muchas de estas utilidades son interfaces graficas de usuario. Incluye el espacio de trabajo Matlab y la ventana de comandos.

2. *La librería de funciones matemáticas Matlab.* Se trata de un amplio conjunto de algoritmos de cálculo, comprendiendo las funciones más elementales como la suma, senos y cosenos o la aritmética compleja, hasta funciones más sofisticadas como la inversión de matrices, el cálculo de autovalores, funciones de Bessel y transformadas rápidas de Fourier.

3. *Gráficos.* Matlab dispone de un conjunto de utilidades destinadas a visualizar vectores y matrices en forma de gráficos. Existe una gran cantidad de posibilidades para ajustar el aspecto de los gráficos, destacando la visualización tridimensional con opciones de iluminación y sombreado, y la posibilidad de crear animaciones.

4. *El Interfaz de Aplicación de Matlab (API).* Consiste en una librería que permite escribir programas ejecutables independientes en C y otros lenguajes, accediendo, mediante DLLs, a las utilidades de cálculo matricial de Matlab.

La gestión de complementos de Matlab se realiza mediante los denominados toolboxes (paquetes de herramientas). Un Toolbox de Matlab es un conjunto de funciones y algoritmos de cálculo especializados en un área de conocimiento: finanzas, tratamiento de señales, etc.

3.1.2.1.1 GUIDE (Graphical User Interface Development Environment).

GUIDE es un juego de herramientas que extiende por completo el soporte de MATLAB, diseñada para crear GUIs (Graphical User Interfaces) fácil y rápidamente ayudando en el diseño y presentación de los controles de la interfaz, reduciendo la labor al grado de seleccionar, tirar, arrastrar y personalizar propiedades.

Una vez que los controles están en posición se editan las funciones de llamada (Callback) de cada uno de ellos, escribiendo el código de MATLAB que se ejecutará cuando el control sea utilizado.

GUIDE está diseñado para hacer menos tedioso el proceso de aplicación de la interfaz grafica y obviamente para trabajar como herramienta de trazado de GUIs; entre sus poderosos componentes esta el editor de propiedades (property editor), este se encuentra disponible en cualquier momento que se esté lidiando con los controles de MATLAB, el editor de propiedades por separado se puede concebir como una herramienta de trazado y asistente de codificación (revisión de nombres y valores de propiedades).

Una de las tantas herramientas con la que cuenta Matlab, es la creación de GUI. La forma de implementar las GUI con Matlab es crear los objetos y definir las acciones que cada uno va a realizar. Al usar GUIDE se obtienen dos archivos:

- Un archivo FIG contiene la descripción de los componentes de la interfaz.
- Un archivo M contiene las funciones y los controles del GUI así como el Callback.

Un callback se define como la acción que llevara a cabo un objeto de la GUI cuando el usuario lo active. Para ejemplificarlo, suponga que en una ventana existe un botón el cual al presionarlo ejecutara una serie de acciones, a eso se le conoce como función callback.

A la herramienta GUIDE se accede de varias maneras, la primera de ellas es tecleando guide en la ventana de comando.

>> guide

Otra manera de entrar a GUIDE es través del menú “File” opción “New” y por último el “GUI”, (como se muestra en la figura 3.8).

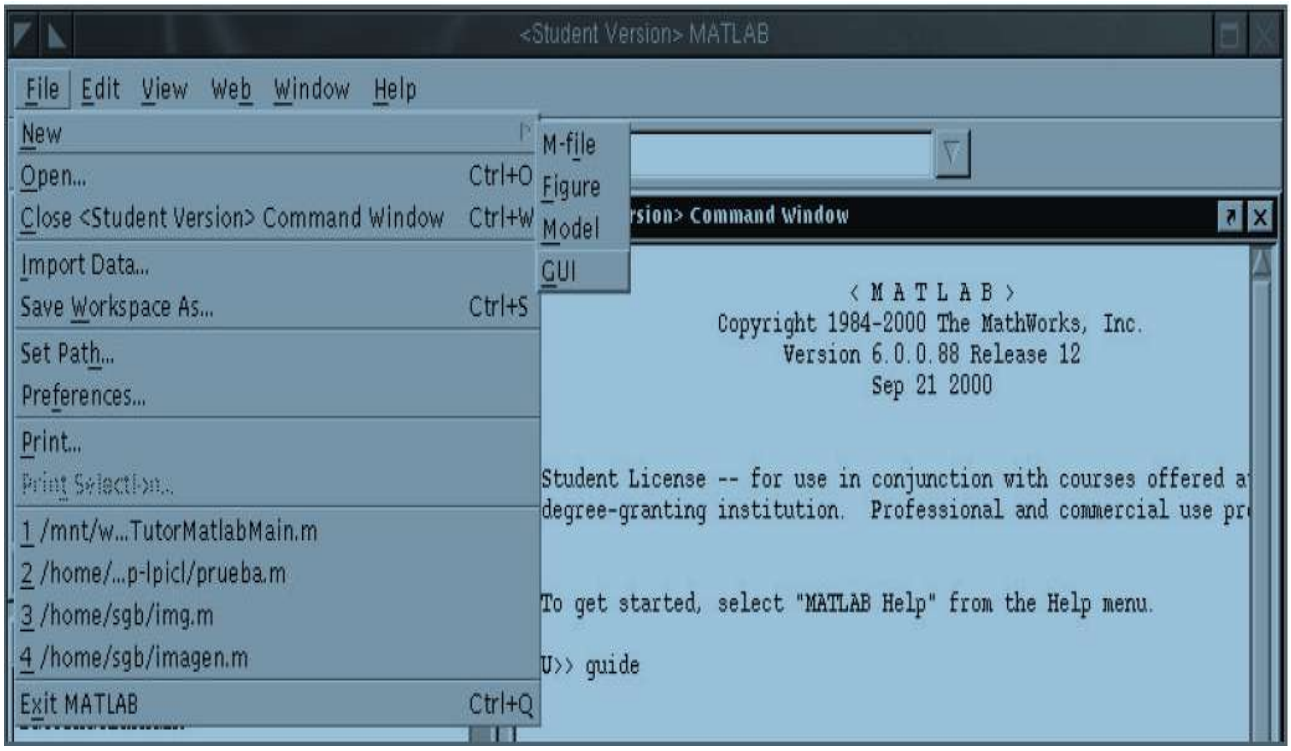


Fig. 3.8. Ejecución GUIDE usando ventana principal.

Las partes principales que conforman el toolbox GUIDE se pueden observar en la figura 3.9 mostrada a continuación:

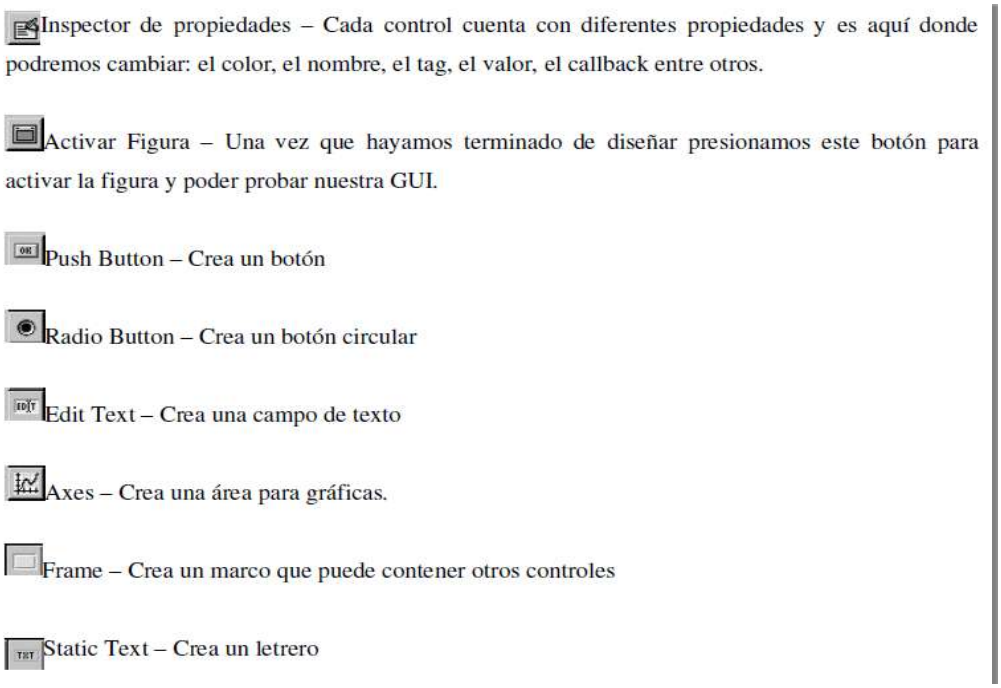


Fig. 3.9. Componentes principales de GUIDE.

Las propiedades varían dependiendo del control a utilizar, a continuación se describen las más comunes y que se han utilizando en las prácticas de control automático.

- *Background Color:* Cambia el color del fondo del control utilizado.
- *Callback:* La propiedad más importante del control, ya que le dice al control que hacer cuando se active.
- *Enable:* Activa o desactiva el control.
- *String:* En el caso de botones, cajas de texto, texto estático; es el texto que muestra el control.
- *Tag:* Otra propiedad importante ya que con este es posible regresar datos o identificar al control.

3.1.2.2 SIMULINK.

Simulink es una aplicación que permite construir y simular modelos de sistemas físicos y sistemas de control mediante diagramas de bloques. El comportamiento de dichos sistemas se define mediante funciones de transferencia, operaciones matemáticas, elementos de Matlab y señales predefinidas de todo tipo.

Simulink dispone de una serie de utilidades que facilitan la visualización, análisis y guardado de los resultados de simulación. Simulink se emplea abundantemente en ingeniería de control.

En primer lugar, se abre la aplicación escribiendo Simulink en la línea de comandos de Matlab, o abriendo desde el Explorador de Windows cualquier fichero con extensión *".mdl"*. En el primero de los casos se abrirá la ventana de la figura 3.10, esta ventana inicial no está destinada a crear modelos de simulación; su función principal consiste en navegar por la enorme librería de bloques disponibles para el modelado.

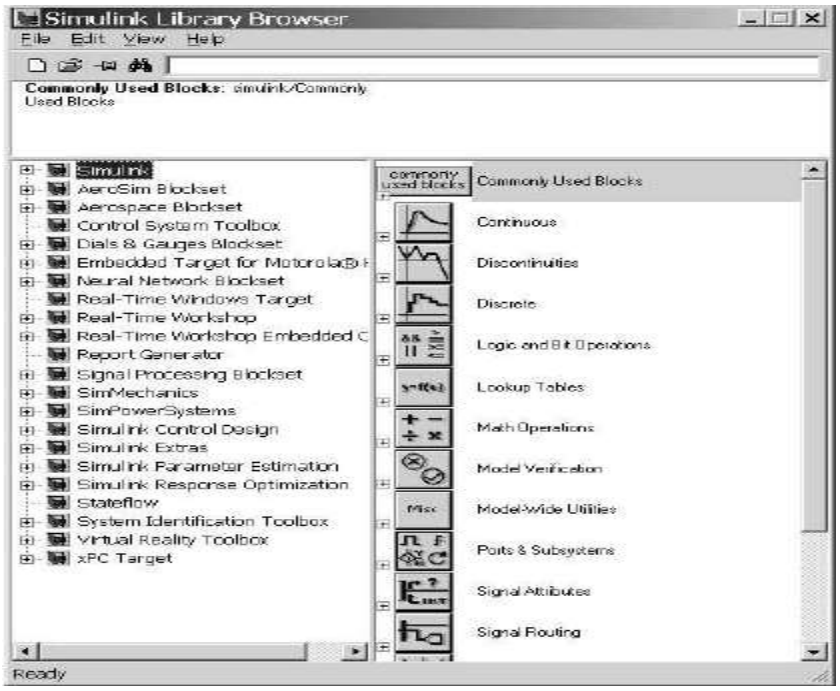


Fig. 3.10. Ventana navegación de bloque Simulink

En ella se distinguen dos partes: la izquierda contiene una visión en forma de árbol de todos los Toolboxes instalados que contienen bloques Simulink. La amplitud de este árbol dependerá de las opciones que se hayan activado al instalar Matlab.

Son de especial interés, los bloques "Real Time Workshop" e "IDE Link Embedded Target" destinados a generar automáticamente código de control para determinadas plataformas hardware comerciales en particular la que controla las familias C281X de Texas Instruments.

La parte derecha de la ventana de la figura 3.10 muestra los bloques Simulink contenidos en el Toolbox o nodo de la parte izquierda de la ventana. Estos bloques se deben arrastrar sobre el espacio de trabajo de Simulink para la creación de un modelo.

Por último, cabe indicar que en la parte superior de la ventana de inicio de Simulink hay varias herramientas, como la búsqueda de un bloque determinado a partir de su nombre, estas pueden resultar bastante útiles.

Si se pulsa en el icono superior izquierdo de la ventana de la figura 3.10 (página en blanco), se abre una ventana blanca sobre la que se iniciara la creación de un modelo de simulación. Dicha ventana se muestra en la figura 3.11.

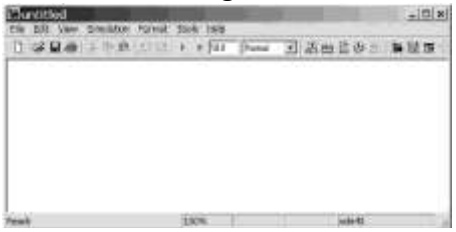


Fig. 3.11. Espacio Trabajo Simulink.

3.1.3 REAL TIME WORKSHOP.

Real Time Workshop genera y ejecuta código C independiente para el desarrollo y prueba de algoritmos modelados en Simulink y código interno de Matlab. El código resultante puede ser usado para muchas aplicaciones en tiempo real y en aplicaciones que no son en tiempo real, incluyendo aceleración de simulaciones, prototipado rápido y prueba de hardware.

Real Time Workshop es la base de la generación de código de Simulink. Este genera código C que cumple con el ANSI/ISO para un modelo entero o para un subsistema individual, habilitando el código para que corra en cualquier microprocesador.

Real Time Workshop es una parte integral del ambiente de trabajo de Simulink. Se puede interactuar con él a través del explorador de modelos, donde se puede configurar los parámetros para la generación de código en Simulink. A través del explorador de modelos se puede:

- Generar código a partir de modelos o subsistemas en Simulink.
- Seleccionar un objetivo para el Real Time Workshop.
- Configurar el objetivo para la generación de código.

Cuando el software de Real Time Workshop genera código, este produce los siguientes archivos:

- *model.c* o *model.cpp*: Código C o C++ generado del diagrama de bloques de Simulink. Este código implementa las ecuaciones del sistema del diagrama de bloques, así como también, inicializa y actualiza las salidas.
- *Model_data.c o model_data.cpp:* Archivo opcional que contiene los datos para parámetros y constantes del bloque de E/S, las cuales además deben ser declaradas externas.
- *Model.h:* Archivo cabecera que contiene los parámetros de simulación del diagrama de bloques, estructuras de E/S y otras definiciones.

Estos archivos son llamados en base al nombre del modelo de Simulink de donde se generan.

3.1.4 EMBEDDED IDE LINK.

Embedded IDE Link conecta a Matlab Simulink con el software para ambientes de desarrollo. Esta herramienta permite generar, ensamblar, probar y optimizar código para prototipado o producción. El automatiza la depuración, la generación del proyecto y la verificación del código en procesadores o simuladores.

Se puede utilizar Embedded IDE Link con Real Time WorkShop para generar proyectos completos en ambientes de desarrollo integrados (IDE) soportados por la herramienta, estos se pueden utilizar para realizar una aplicación en tiempo real para los procesadores soportados. El proyecto generado no solo crea código C para la aplicación, sino que genera el software del marco de trabajo requerido para inicializar y manejar la ejecución del código de la aplicación del procesador en tiempo real.

Se puede utilizar Matlab para analizar y depurar el código interactivamente que se ejecutara en el procesador a través del IDE (ambiente de desarrollo integrado), y visualizar los resultados de la ejecución usando los gráficos del mismo Matlab.

Las principales características son:

- Depuración, verificación y análisis automatizado de código escrito o generado usando Matlab Simulink.
- Capacidad de ejecución en tiempo real.
- Configuración personalizada de mapeo de memoria para IDE especificas.
- Soporte para IDEs y procesadores de la siguientes vendedores: Altium, Analog Devices, Freescale, Renesas, STMicroelectronics y Texas Instruments.

Embedded IDE Link permite que se usen las funciones de Matlab para comunicarse con el software Code Composer Studio, explicado en la sección siguiente. Además, se hace fácil la verificación de código en Code Composer Studio usando un modelo creado en Simulink. Una gran gama de DSPs de Texas Instrument son soportadas por esta herramienta, entre las que se encuentran:

- TI's C2000
- TI's C5000
- TI's C6000

3.1.4.1 RTDX

Real-Time Data Exchange (RTDX) de la Texas Instrument agrega la característica de tener visibilidad continua en el funcionamiento en modo real de la tarjeta DSP. RTDX permite transferir información entre la tarjeta destino y la computadora anfitrión sin interrumpir el funcionamiento de la tarjeta.

RTDX se utiliza con Embedded IDE Link y el Code Composer Studio para acelerar el desarrollo de sistemas para los procesadores de la familia C2000 de Texas Instruments. Por ejemplo, utilizando RTDX se puede:

- Mandar y recibir datos desde la memoria del procesador.
- Cambiar las características operativas del programa.
- Hacer cambios al algoritmo sin necesidad de detener el programa.

Es importante recordar la importancia de habilitar la interacción en tiempo real de un proceso o de uno algoritmo, y poder observar los resultados mientras se desarrollan.

3.1.4.2 BLOQUES EMBEDDED IDE LINK

Entre los bloques a destacar de la librería de este toolbox se encuentran los siguientes:

- **ADC**

  El bloque ADC maneja la configuración de cuales entradas tendrán la conversión análoga a digital activada. El usuario puede configurar el tiempo de muestreo así como el tipo de dato de salida preferido. Además, existen opciones para cuales módulos inicializar. El número de entradas puede estar entre 1 a 16. El bloque de Simulink se muestra en la figura 3.12.

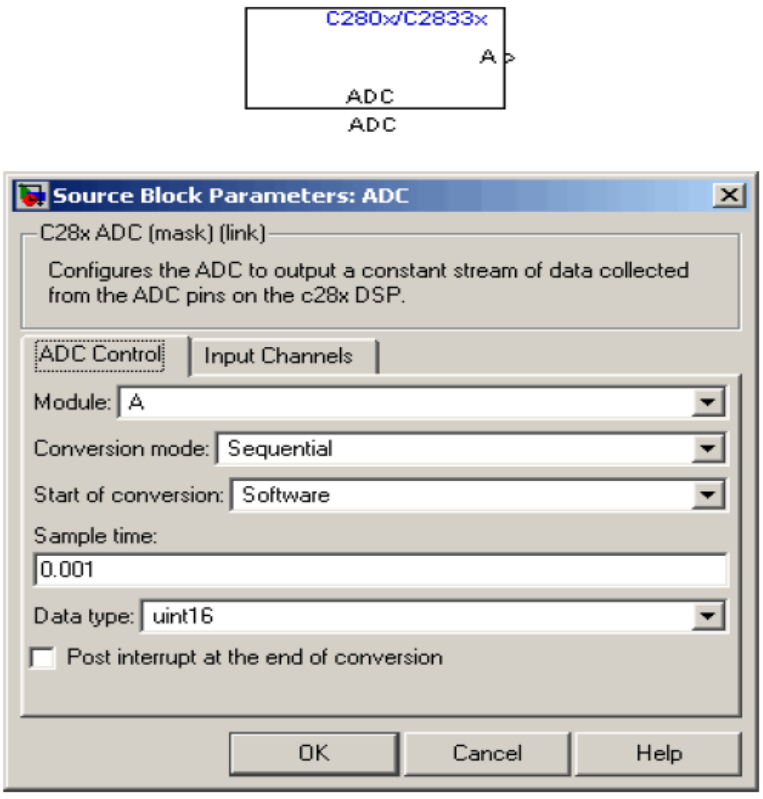


Fig. 3.12. Bloque ADC en Simulink.

- **PWM**

A diferencia del CCS (Code Composer Studio), configurar una señal PWM en Simulink es muy fácil. El bloque ofrece muchas opciones, por ejemplo, el periodo de la señal, si la señal es asimétrica o no, y la más importante cual salida habilitar.

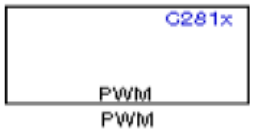


Fig. 3.13. Bloque PWM en Simulink.

Muchas opciones dependiendo de la lógica de control están disponibles, dándole al usuario control total sobre como la señal resultante debe comportarse. Se debe tener en cuenta que todas las entradas al bloque PWM deben ser valores escalares.

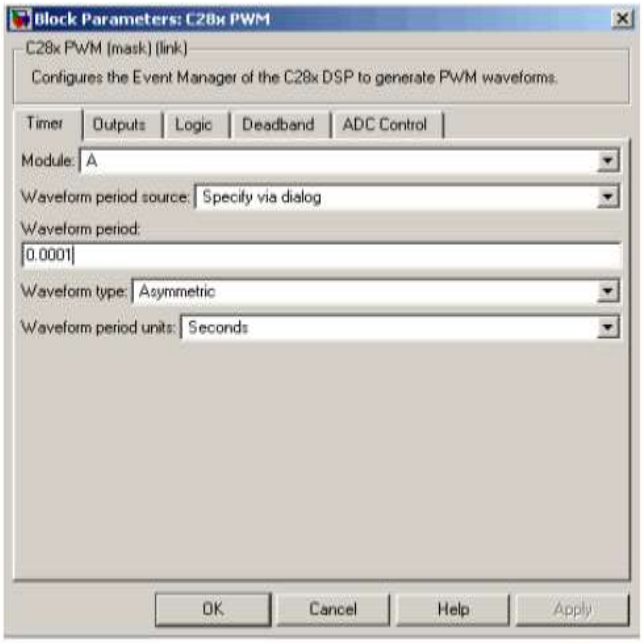


Fig. 3.14. Opciones bloque PWM en Simulink.

- **Bloques RTDX**

Los bloques RTDX son una gran herramienta que provee esta librería a los modelos en Simulink, tienen un gran potencial y son de suma importancia para el desarrollo de los controladores de velocidad y posición.

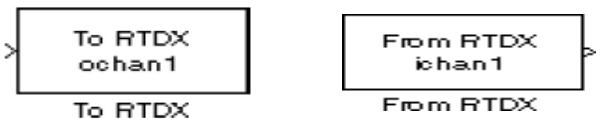


Fig. 3.15. Bloques RTDX en Simulink.

- **From RTDX**

Agrega un canal de entrada RTDX. Cuando se genera código desde Simulink utilizando Real Time Workshop y este incluye un bloque "From RTDX", la generación del código agrega los comandos en C para crear un canal de entrada RTDX. Los canales de entrada transfieren datos desde la computadora hacia la tarjeta. El comando que se incluye en el archivo C es:

RTDX_enableInput(&channelname); donde &channelname es el nombre asignado al canal de entrada.

Para usar bloques RTDX en un modelo, se debe descargar y correr el modelo en la tarjeta destino, se deben habilitar los canales RTDX desde Matlab o en el menú de dialogo del bloque se marca "Enable RTDX channel on start up". Las opciones disponibles en el menú de dialogo de bloque se detallan a continuación.

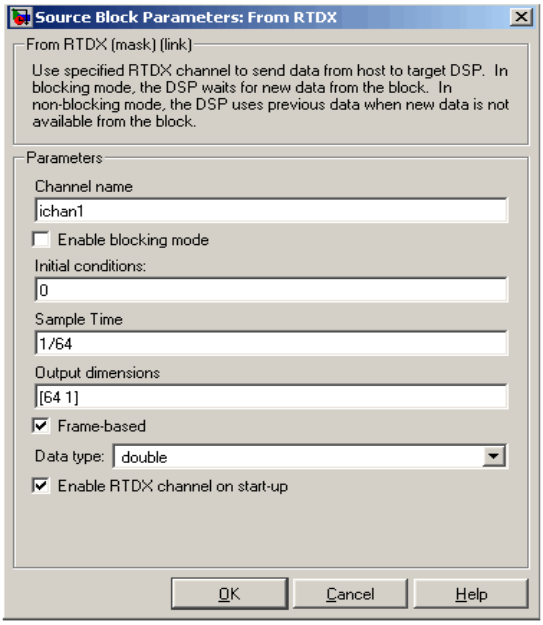


Fig. 3.16. Opciones bloque "From RTDX" en Simulink.

Channel Name: Nombre del canal de entrada que se creara por el código generado.

Enable blocking mode: El modo de bloqueo instruye al procesador a pausar el proceso hasta que nueva información esté disponible del bloque RTDX. Si se habilita y nuevos datos no están disponibles cuando el procesador lo requiera el proceso se detiene.

Initial Conditions: Es el dato que el procesador lee del el bloque RTDX en la primera lectura. Dejando esta opción en blanco causa un error en Real Time Workshop.

Sample Time: Tiempo entre muestras de la señal. Por defecto es 1 segundo.

Output dimensión: Dimensiones de la matriz para la señal de salida del bloque. El primer valor es el numero de filas y el segundo el numero de columnas.

Data Type: Tipo de datos que viene de el bloque. Se puede elegir entre “double”, “single”, “Uint8”, “Int16”, “Int32”.

Enable RTDX channel on start-up: Se habilita el canal RTDX cuando se inicializa el canal desde MATLAB. El canal siempre se debe de abrir en Matlab.

- **To RTDX**

Agrega un canal de salida RTDX. Cuando se genera código desde Simulink utilizando Real Time Workshop y este incluye un bloque “From RTDX”, la generación del código agrega los comandos en C para crear un canal de salida RTDX. Los canales de salida transfieren datos desde la tarjeta hacia la computadora. El comando que se incluye en el archivo C es:

RTDX_enableOutput(&channelname); donde &channelname es el nombre asignado al canal de entrada.

Para usar bloques RTDX en un modelo, se debe descargar y correr el modelo en la tarjeta destino, se deben habilitar los canales RTDX desde Matlab o en el menú de dialogo del bloque se marca “Enable RTDX channel on start up”. Las opciones disponibles en el menú de dialogo de bloque se detallan a continuación.

Fig. 3.17. Opciones bloque “To RTDX” en Simulink.

Channel Name: Nombre del canal de salida que se creara por el código generado.

Enable blocking mode: En el modo de bloqueo, la escritura de datos se suspende mientras el canal RTDX está ocupado, esto es, cuando datos están siendo escritos en cualquier dirección.

Enable RTDX channel on start-up: Se habilita el canal RTDX cuando se inicializa el canal desde MATLAB. El canal siempre se debe de abrir en Matlab.

3.1.5 CODE COMPOSER STUDIO.

El software Code Composer Studio es creado por Texas Instruments como una herramienta de desarrollo para las DSP fabricadas por ellos mismos, se utilizo la versión 3.3 la cual es compatible con la versión de Matlab Simulink R2008b.

La interfaz de usuario del programa se muestra en la figura 3.18, donde se pueden distinguir algunas barras de herramientas importantes.

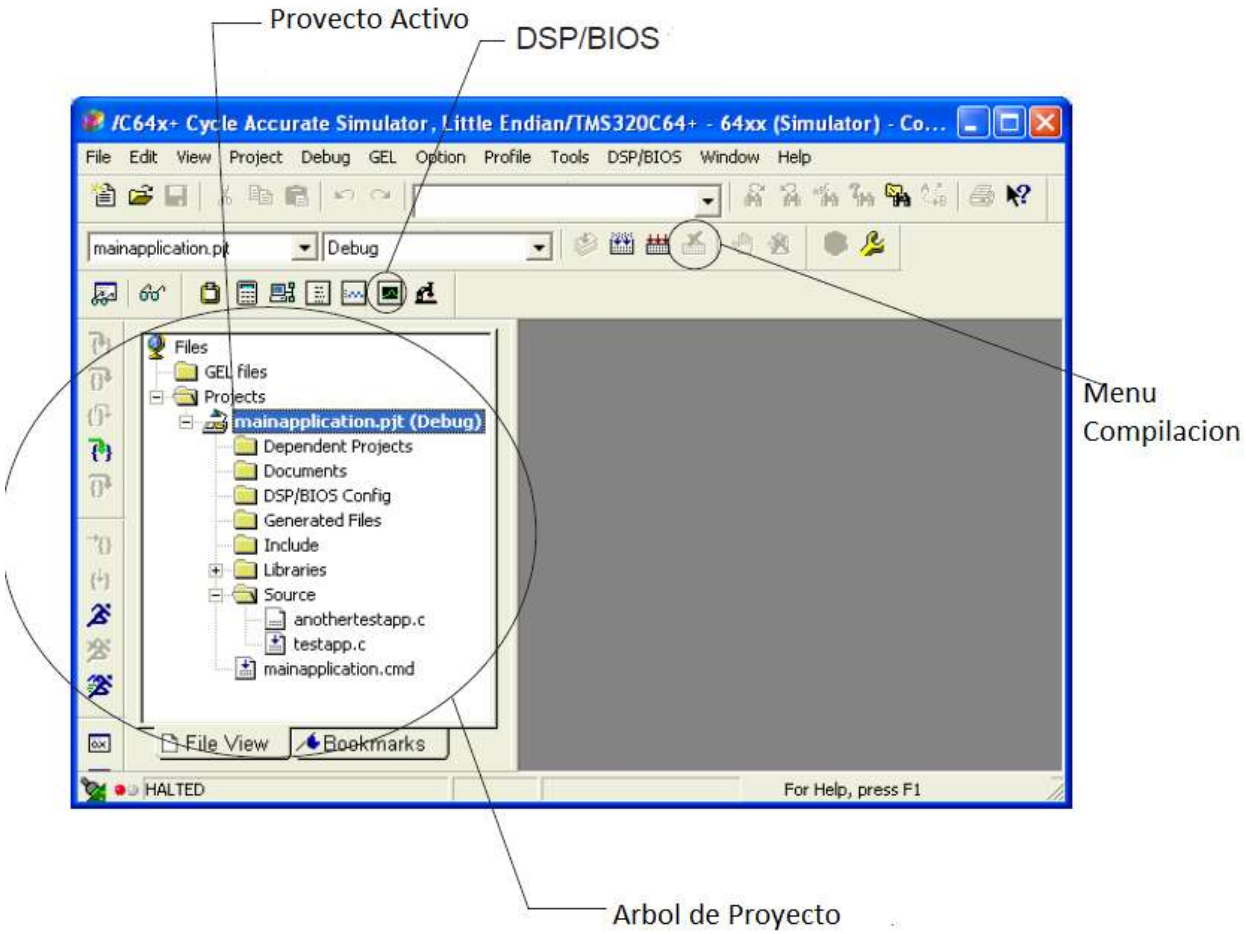


Fig. 3.18. Descripción Code Composer Studio.

Algunos iconos importantes dentro del programa se pueden observar junto con su descripción en la figura 3.19.

Fig. 3.19. Descripción de botones de Code Composer Studio.

Para compilar un proyecto o un archivo fuente individual existen varias opciones disponibles; desde el menú de proyecto o desde los iconos en la barra de compilación.

Compilar: Compila un archivo individual.
Compilar incrementalmente: Compila solamente los archivos que han cambiado.
Compilar Todo: Compila todo el proyecto.

El resultado de la compilación aparece en la pantalla de depuración en la parte inferior de la pantalla y notifica al usuario si hubiera errores o advertencias.

Cuando el programa ha sido generado o compilado, es necesario cargarlo a la DSP. Esto se puede hacer seleccionando “Cargar Programa” desde el menú archivo(o alternativamente, “Recargar Programa” si previamente se había cargado el mismo proyecto).

Al cargar un programa se genera un archivo de salida, visible en el menú proyecto, el cual es guardado en la carpeta llamada Debug. Cuando el programa es cargado el código en ensamblador se abre automáticamente.

Algunos comandos muy útiles para el software son:

F5: Ejecuta el programa.
Shift-F5: Detiene la ejecución del programa.
F8: Paso a través del programa (útil para depurar programas).

3.2 CONSTRUCCIÓN DEL CONTROLADOR DIGITAL CON eZdsp F2812.

3.2.1 DESCRIPCIÓN DEL CIRCUITO DE INTERFAZ.

Cuando se trabaja con DSP's casi siempre es necesario una etapa de acomodamiento de señales por diversas razones, como pueden ser:

- Adecuación de niveles de tensión.
- Eliminación del ruido.
- Conversión de señales digitales a analógicas.
- Incrementar la potencia de salida.
- Aislamiento del circuito de potencia con el de control.

Estas razones brindan los motivos necesarios para la construcción de un circuito de interfaz que comunique el área de potencia con el área de control.

A continuación se muestra el esquema de circuito necesario para comunicar la salida PWM de la DSP con el pin de salida correspondiente en el modulo.

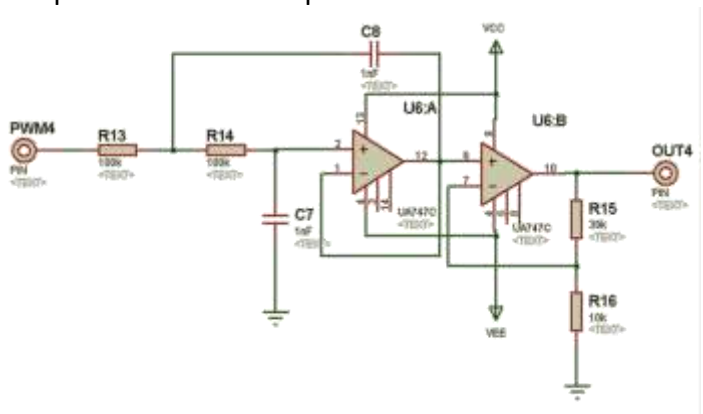


Fig. 3.20. Esquema de filtro pasó bajo y pre amplificador.

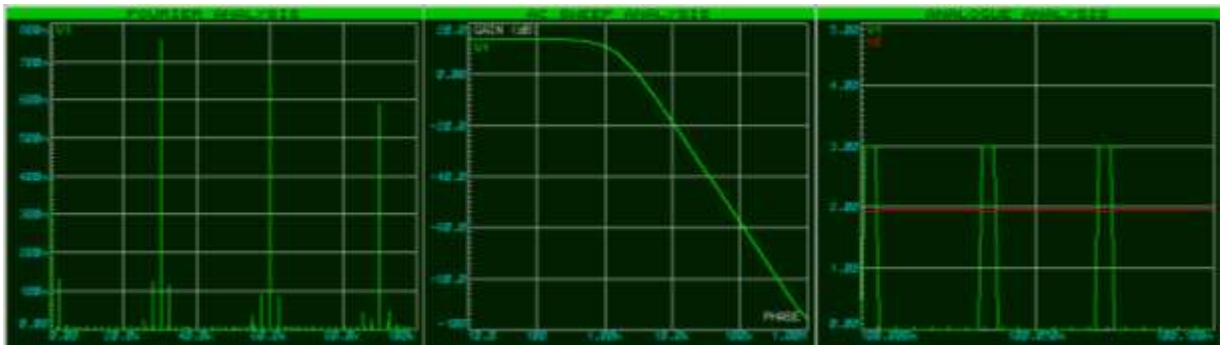
Este circuito se compone de bloques simples, el primero es un filtro paso bajo sintonizado a 1KHz el cual tiene la función de eliminar todos los voltajes armónicos de la señal PWM de entrada y solo dejar pasar la componente DC de la señal, el segundo bloque es un amplificador no inversor con ganancia 5 el cual amplifica la señal de 0 V a 3 V máximo que proporciona el LPF a un nivel de 0 V a 15 V máximos necesarios para poder alimentar el servoamplificador del KIT FEEDBACK 150.

El circuito fue simulado para poder determinar a qué frecuencias respondía mejor; los resultados se muestran en la figura 3.21. La grafica a) muestra las frecuencias en las que aparecen las primeras armónicas de la señal PWM de entrada, se ajusta la señal a una frecuencia de 20KHz y un ancho de pulso del 10%. Después de diversas pruebas se determino que a menor ancho de pulso aparecen armónicas más cerca de la fundamental, lo que quiere decir que con un ancho de pulso del 10% se simula la peor situación a esta frecuencia.

La grafica b) muestra la respuesta en frecuencia del circuito, en esta se puede destacar que, como se dijo anteriormente, el filtro esta sintonizado a 1KHz y tiene una pendiente de -40dB por década lo que indica que para poder darle una atenuación razonable a las armónicas estas no deben presentarse en frecuencias debajo de 10KHz.

La Figura c) muestra la entrada y salida en función del tiempo. Como se puede observar la señal de salida es bastante plana a pesar de que se puede observar una armónica debajo de 10KHz que sí está dejando pasar el filtro.

Se realizaron simulaciones con distintas frecuencias y anchos de pulsos, y se determino que el circuito empieza a operar de una manera aceptable a los 10KHz pero su uso recomendado es a frecuencias entre 20Khz y 40Khz.



a) b) c)

Fig. 3.21. Señales circuito salida PWM a) Graficas de Fourier, armónicas que tiene la señal de entrada a 20KHz y ancho de pulso de 10%. b) respuesta del circuito en el dominio de la frecuencia. c) Entrada (verde) y salida (rojo) en el dominio del tiempo.

A continuación se explicará el circuito que comunica las entradas analógicas externas del controlador digital con la entrada analógica de la tarjeta eZdsp F2812. Este no es más que un filtro paso bajo inversor de primer orden, el cual esta sintonizado a 1Hz, su propósito es eliminar todo el ruido que pueda contener la señal de entrada.

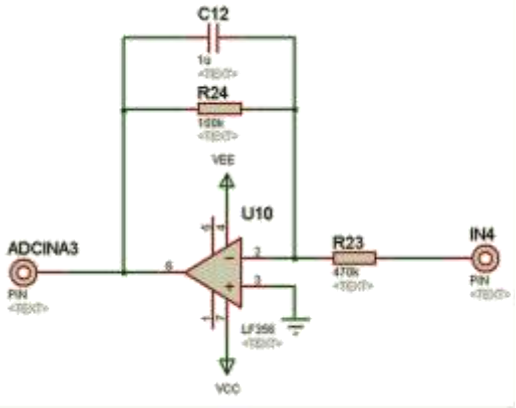


Fig. 3.22. Filtro paso bajo para entradas analógicas.

Las graficas de la figura 3.23 muestran la respuesta del circuito en la frecuencia y en el tiempo, la de la derecha representa la simulación en el dominio del tiempo para una entrada de señal con ruido la cual fue simulada utilizando un tono de 5Hz modulado a 30Hz con un índice de modulación de 0.7, una amplitud de 0.4V y con un offset de -15V, esta señal puede representar la realimentación de voltaje enviada por el tacómetro para medir la velocidad del motor. La salida del circuito se representa con la grafica de color rojo, se puede ver que elimina la mayor parte del ruido de la señal aunque siempre se presenta algo de distorsión en el voltaje de salida.

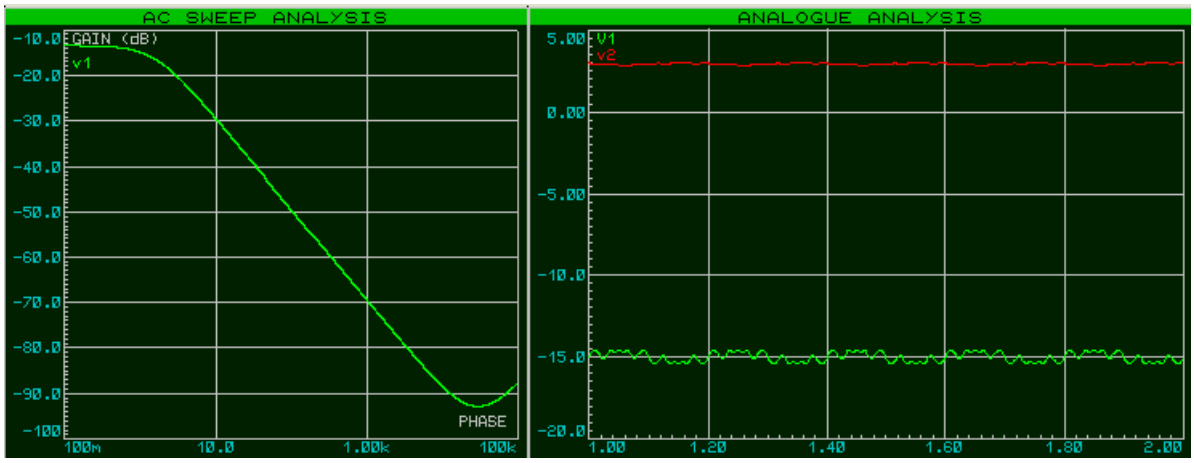


Fig. 3.23. Simulación en el dominio de la frecuencia y en el dominio del tiempo para el filtro paso bajo de la figura 3.20.

Esto se mejoraría moviendo el polo del sistema más a la izquierda, pudiendo haberse colocado un capacitor de 1uF en lugar de uno de 0.1uF, pero como se puede observar en la grafica de la figura 3.24, los tiempos de levantamiento del circuito aumentarían hasta aproximadamente 0.5 segundos. Utilizando el capacitor de 0.1uF para que la salida responda a un cambio en la entrada se tiene un retardo máximo de 50ms, lo que provoca algunos problemas de inexactitud en el sistema, pero se utilizará así porque se pretende lograr un equilibrio entre eliminación del ruido y los tiempos de levantamiento del sistema.

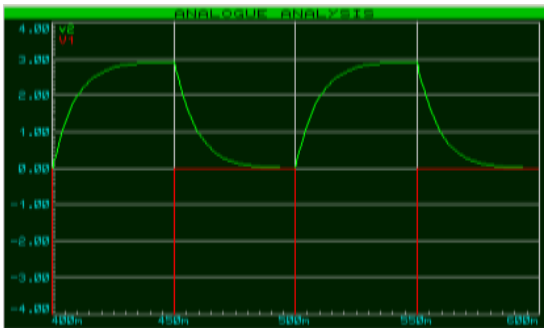


Fig. 3.24. Simulación de la respuesta a una entrada escalón del circuito de la figura 3.22.

Una entrada algo peculiar con la que cuenta este modulo controlador es un transductor que convierte una señal de corriente a una de voltaje que es fácilmente manejable por la

DSP. Este circuito esta hecho en base a un amplificador operacional AMP01 de Analog Devices el cual es un amplificador de instrumentación de alta precisión y bajo ruido; este amplificador toma la diferencia de potencial entre los dos terminales de las resistencias R27 y R28, y la amplifica para dar una equivalencia de 1V de salida por cada A de circulación de corriente entre las resistencias.

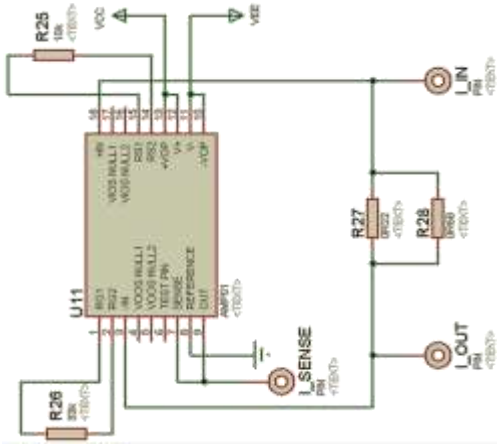
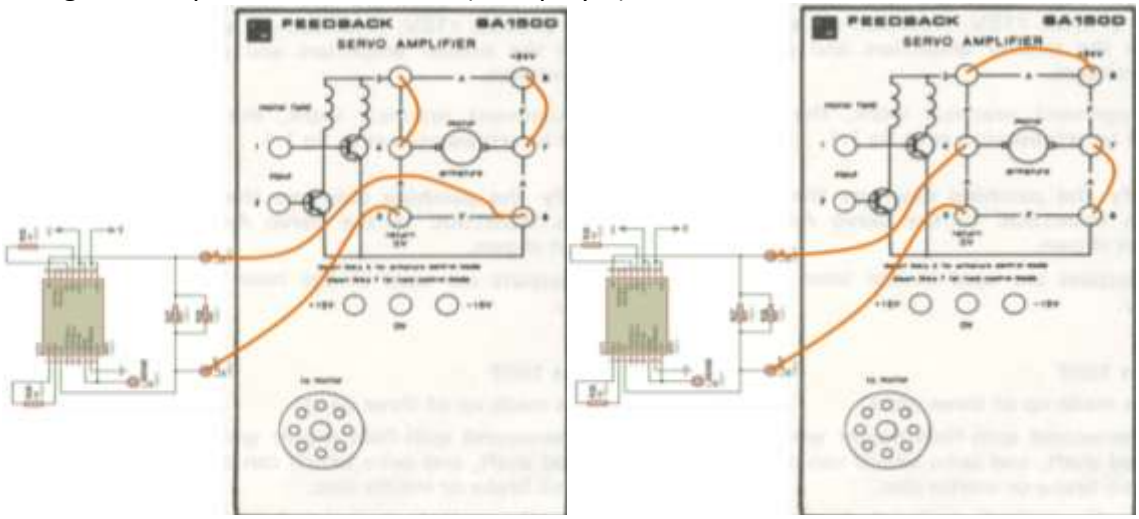


Fig. 3.25. Circuito transductor de corriente a voltaje.

Es importante la incorporación de un sensor de corriente porque es algo con lo cual no se cuenta en los módulos analógicos del KIT FEEDBACK 150 que se utilizan en los laboratorios de control automático. Esto da la capacidad de realizar prácticas en el campo de controladores PID de corriente en motores DC.

La forma de conexión del servoamplificador con el modulo controlador digital se muestra en la figura 3.26 para las dos formas a) campo y b) armadura.



a) **b)**

Fig. 3.26. PID de corriente a) conexión del servoamplificador en control de campo para la medición de corriente b) conexión del servoamplificador en control de armadura para medición de corriente.

Nota: En el anexo C se puede encontrar mayor información sobre la creación del PCB y el chasis para el Modulo Digital con eZdsp F2812.

3.2.2 CONEXIÓN ENTRE EL CIRCUITO DE INTERFAZ Y LA EZDSP F2812.

Anteriormente se describieron los circuitos que comunican el exterior con la DSP y viceversa, ahora se describe la conexión física de estos circuitos a la DSP. Anticipándose un poco a lo que viene en los siguientes temas se puede decir que para el desarrollo de un controlador PID de velocidad o posición utilizando los módulos necesarios del KIT FEEDBACK 150, solo se necesita disponer de dos A/D y de dos D/A para cualquiera de ambos. Pero pesando en que en un futuro a este modulo controlador digital se le pueden dar distintos usos como por ejemplo el controlar un motor síncrono trifásico ó el control de un ciclo convertidor y además sumando el hecho de no dejar muy subutilizado el hardware de la DSP, ya que esta cuenta con 12 salidas PWM y 16 convertidores analógico digital, se decidió colocar 4 salidas D/A y 5 entradas A/D entre las cuales se encuentra la señal del transductor corriente-voltaje.

El esquema de conexiones de todo el sistema se muestra en la figura 3.27.

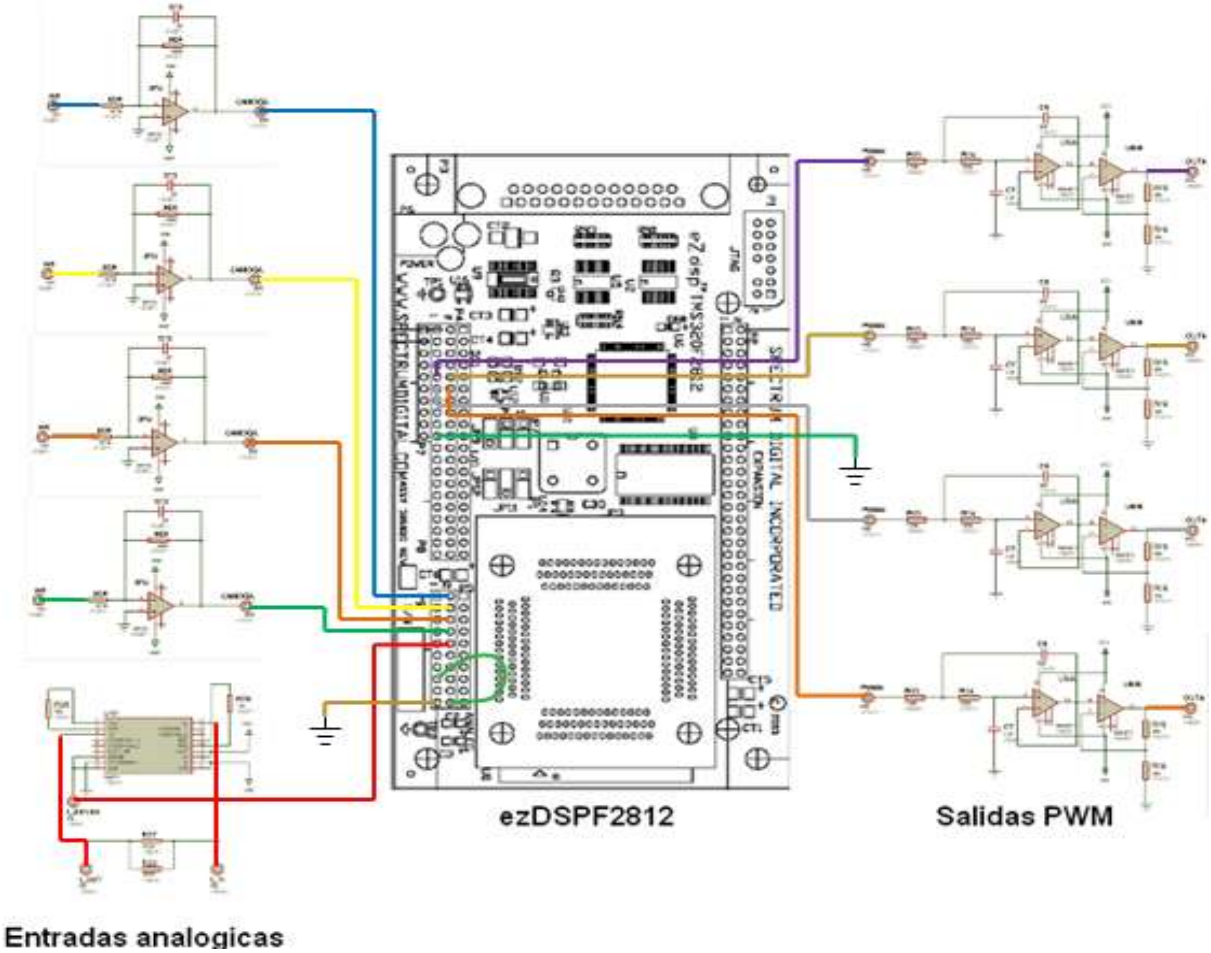


Fig. 3.27. Esquema de conexiones.

En el esquema anterior los colores de las líneas representan el color verdadero de los cables que conectan físicamente estas dos tarjetas, y también los bornes frontales del modulo controlador con la tarjeta de interfaz.

3.2.3 COSTOS DE MATERIALES Y EQUIPO UTILIZADO PARA LA CONSTRUCCIÓN DEL MODULO DIGITAL CON EZDSP F2812

La siguiente tabla resume los costos en los que se incurrió para la construcción del Modulo Digital con eZdsp F2812:

Descripción	Monto ($)
Construcción de circuito de interfaz	25.00
Construcción del chasis para circuito de interfaz y eZdsp F2812	15.00
eZdsp F2812	325.00
Transporte de eZdsp F2812	100.00
Total	465.00

TABLA 3.5. Costo total de construcción del Modulo Digital con eZdsp F2812.

3.3 CONTROL DE VELOCIDAD DE MOTOR DC MEDIANTE RTDX.

La construcción de este controlador se basa en el software Simulink de Matlab, este utiliza una serie de bloques que realizan las operaciones deseadas. En este controlador se muestra el uso de los periféricos C28x y los bloques de la librería DMC para controlar la velocidad de un motor de DC en lazo cerrado.

La velocidad deseada del motor es ajustada por el usuario en la interfaz gráfica (GUI). Este valor es alimentado al controlador (basado en la eZdsp-F2812) para cambiar la velocidad del motor. El lazo se cierra con un tacómetro. El controlador ajusta constantemente el valor del voltaje DC aplicado al motor para mantener la velocidad deseada.

3.3.1 ESPECIFICACIONES DE CONSTRUCCIÓN.

El controlador cuenta dos botones, uno para *construir, cargar y correr;* y otro para *ver un demo del script,* cuenta también con 3 subsistemas generales dentro de los cuales se encuentra una serie de bloques y otros subsistemas según la función que se desea realizar, todos los bloques con los que dispone Simulink se encuentran en el buscador de librerías (Library Browser).

3.3.1.1 Subsistema 1: Establecer Velocidad.

Para crear este subsistema se selecciona el bloque de subsistema, este se encuentra en: Simulink - *Ports & Subsystems* - *Subsystem.*

Este posee internamente una entrada y una salida, se elimina la entrada y se deja únicamente la salida, ya que la entrada será mediante el RTDX (Real Time Data eXange), se selecciona el bloque "From RTDX", este se encuentra en:
Target Support Package TC2 - RTDX Instrumentation – From RTDX

Los parámetros que se le definen a este bloque RTDX son los siguientes:

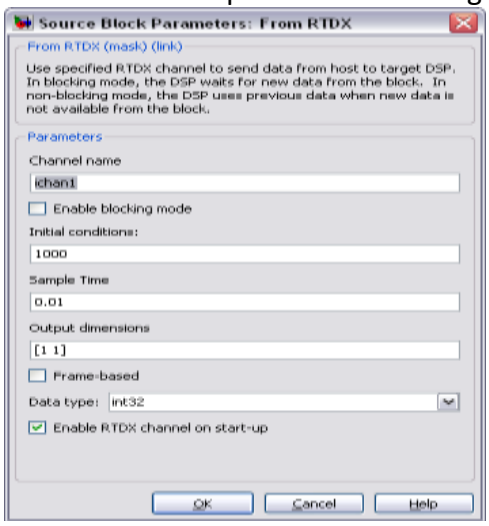


Fig. 3.28. Parámetros definidos al bloque "From RTDX".

Este será el bloque inicial de este subsistema, luego de este se coloca un bloque de "Conversión de Tipo de Dato", este se encuentra en:
Simulink - Commonly Used Blocks - Data Type Conversion.

Los parámetros de este bloque se definen de la siguiente manera:

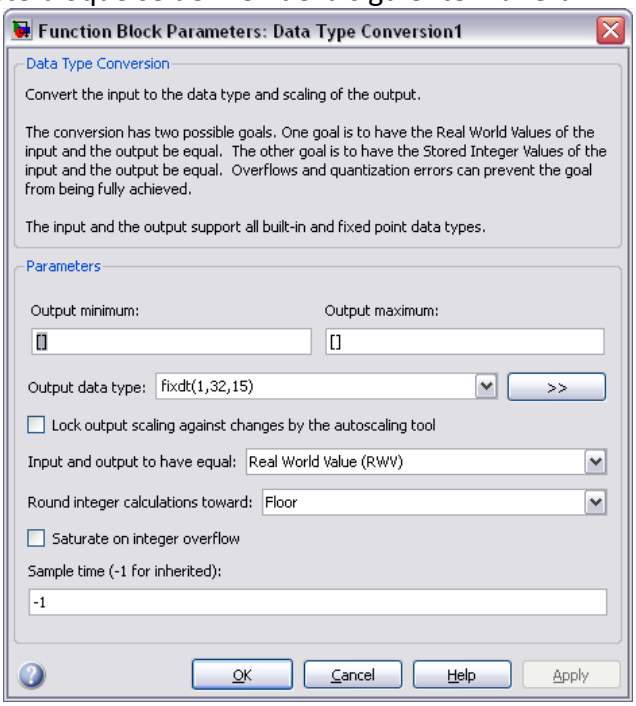


Fig. 3.29. Parámetros definidos al bloque "Data Type Conversion".

A la salida de este bloque se conecta el bloque de salida, a este se le cambia el nombre a "*Velocidad Deseada*", luego se le cambia el nombre al subsistema por "*Establecer Velocidad*" y se tiene completo el primer subsistema, este queda internamente de la siguiente manera:

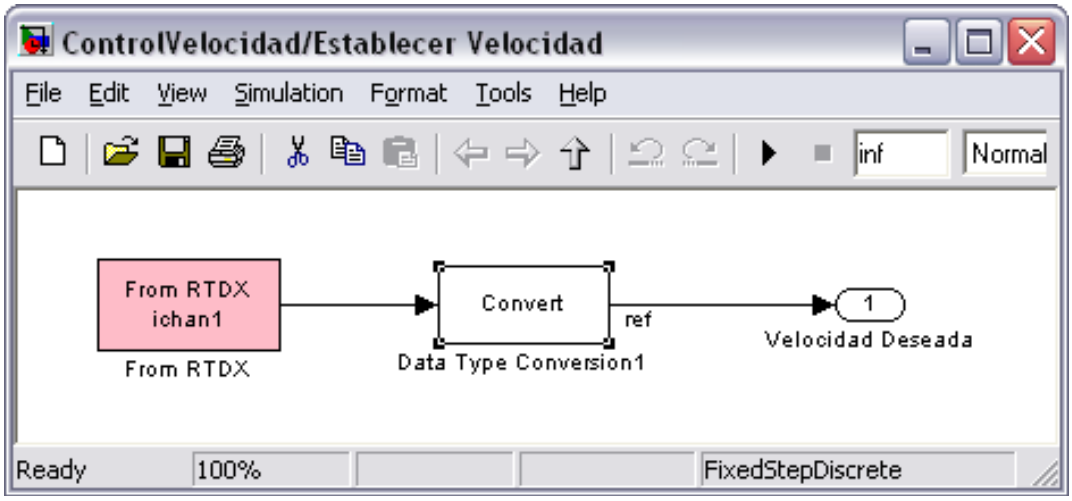


Fig. 3.30. Subsistema 1 "Establecer Velocidad".

3.3.1.2 Subsistema 2: PID.

Este posee internamente dos entradas y se le elimina la salida, la entrada extra se le puede agregar de la siguiente ubicación:
Simulink - Commonly Used Blocks - In1

Una de las entradas posee el nombre de "*Referencia*", la segunda entrada recibe el nombre "*Realimentación*". Ambas entradas van a bloques de conversión de tipo de datos, ambos definidos con los siguientes parámetros:

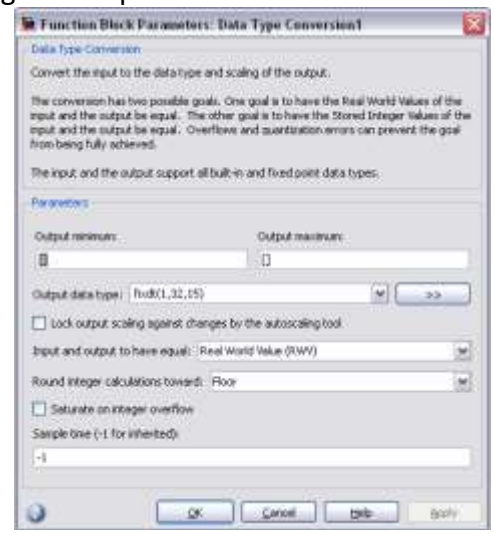


Fig. 3.31. Parámetros definidos a bloques "Data Type Conversion".

Las salidas de los bloques de conversión de tipo de dato van a un controlador PID, este se encuentra en:
Target Support Package TC2 – C28x DMC Library – PID Controller

A este se le definen los siguientes parámetros:

Fig. 3.32. Parámetros definidos al bloque "PID".

A la salida del controlador PID se conectan en paralelo dos bloques de Conversión de Tipo de Dato, uno de ellos convierte a entero con signo de 16 bits y el otro a entero sin signo de 32 bits, sus parámetros son los siguientes:

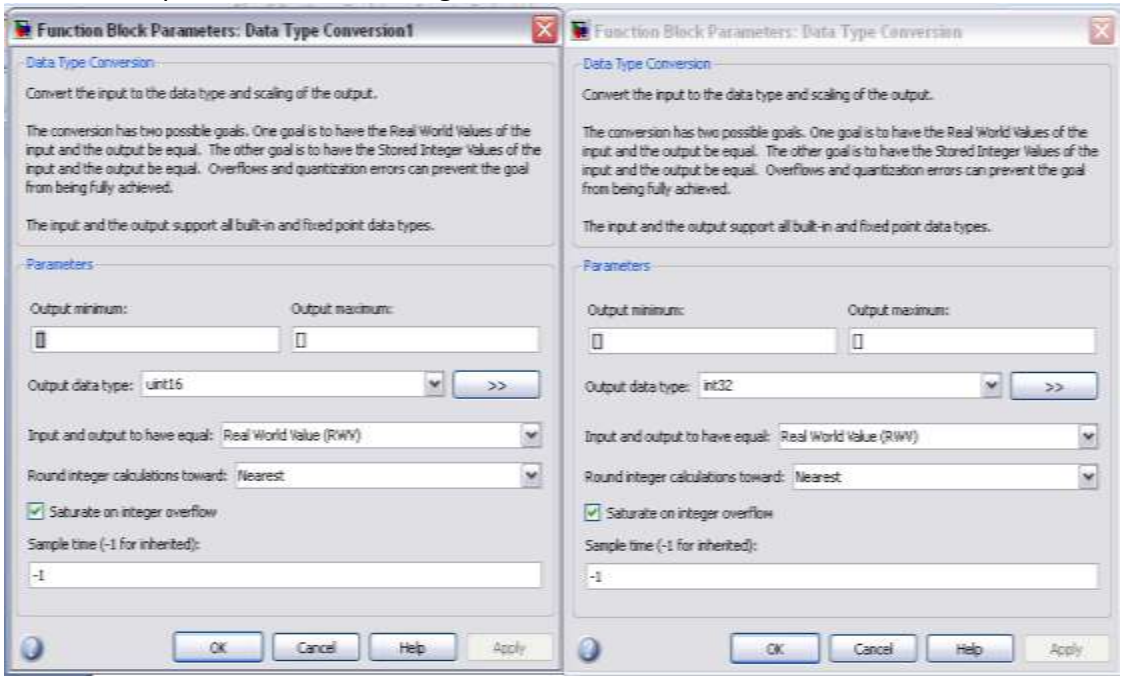


Fig. 3.33. Parámetros definidos a bloques "Data Type Conversion".

El bloque de conversión de tipo de datos a entero con signo de 16 bits se conecta con un bloque PWM, este se encuentra en:
Target Support Package TC2 – C281x DSP Chip Support – PWM

Los parámetros que se le definen a este bloque son los siguientes:

- Para la pestaña "Tiempo":

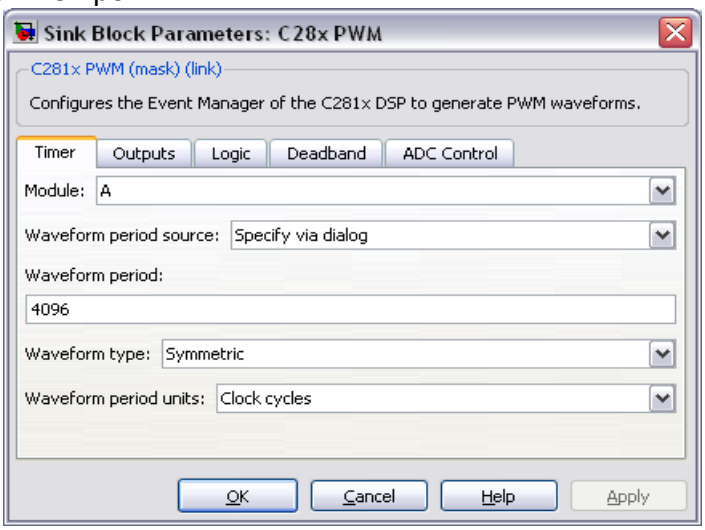


Fig. 3.34. Parámetros definidos a la pestaña "Timer" del bloque "PWM".

- Para la pestaña “Salidas”:

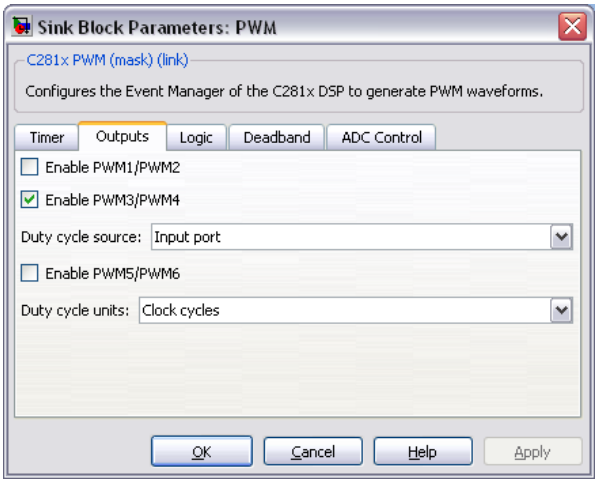


Fig. 3.35. Parámetros definidos a la pestaña “Outputs” del bloque “PWM”.

- Para la pestaña “Lógica”:

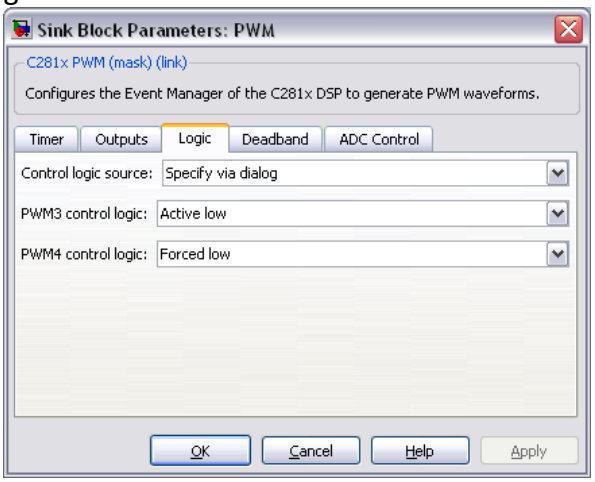


Fig. 3.36. Parámetros definidos a la pestaña “Logic” del bloque “PWM”.

- Para la pestaña “Banda Muerta”:

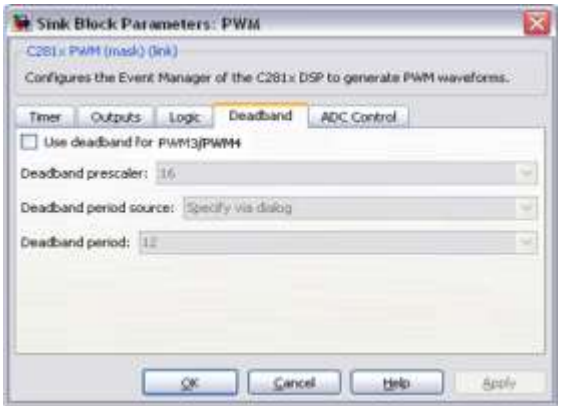


Fig. 3.37. Parámetros definidos a la pestaña “Deadband” del bloque “PWM”.

- Para la pestaña “Control ADC”:

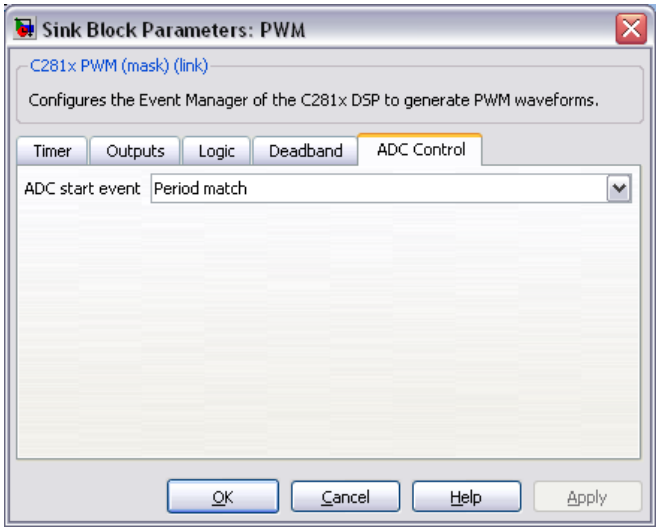


Fig. 3.38. Parámetros definidos a la pestaña “ADC Control” del bloque “PWM”.

El bloque de conversión de tipo de datos a entero de 32 bits se conecta con un nuevo subsistema, este sirve para tomar muestras en tiempo real, a este subsistema se le elimina la salida que posee por defecto ya que esta se sustituye por un bloque RTDX, la estructura de este subsistema esta diseñada de la siguiente manera:

La entrada del subsistema se conecta a un bloque “Downsample1”, este se encuentra en: *Signal Processing Blockset - Signal Operations - Downsample*

Los parámetros de este bloque son los siguientes:

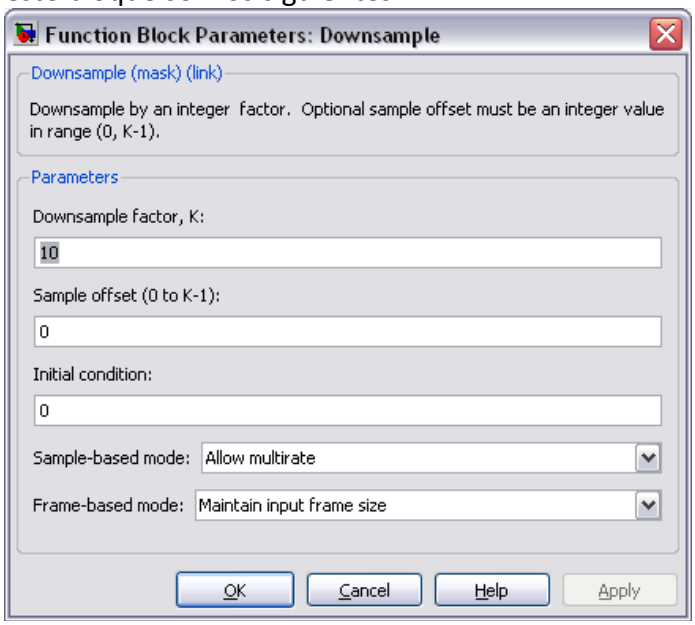


Fig. 3.39. Parámetros definidos al bloque “Downsample”.

A la salida de este se conecta un bloque de conversión de tipo de datos, los parámetros de este son los siguientes:

Fig. 3.40. Parámetros definidos al bloque "Data Type Conversion4".

A la salida de este bloque se conecta un Buffer, este bloque se encuentra en:
Signal Processing Blockset – Signal Management – Buffers – Buffer

Los parámetros definidos para este bloque son los siguientes:

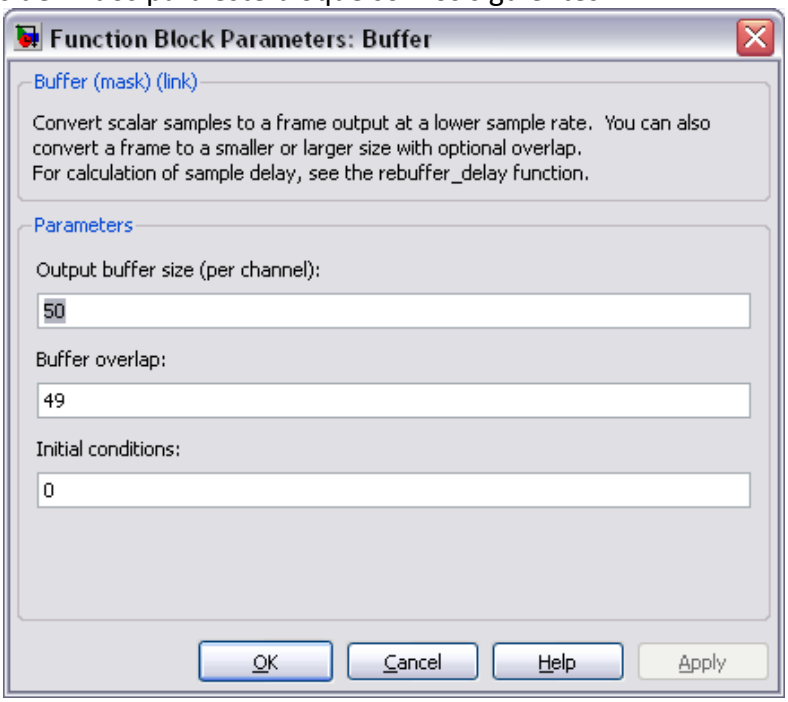


Fig. 3.41. Parámetros definidos al bloque "Buffer".

Para finalizar este subsistema se conecta un bloque “To RTDX” a la salida del buffer, la ubicación de este bloque es:
Target Support Package TC2 - RTDX Instrumentation – To RTDX

Los parámetros de este son:

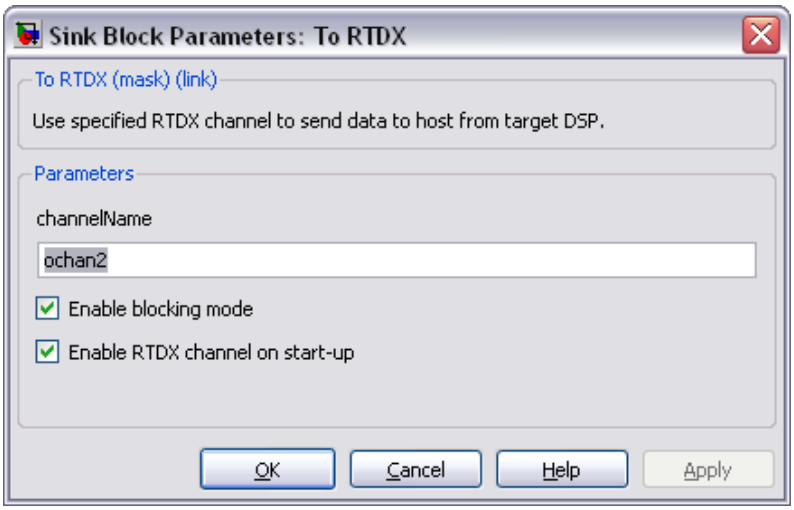


Fig. 3.42. Parámetros definidos al bloque “To RTDX”.

El subsistema completo queda de la siguiente forma:

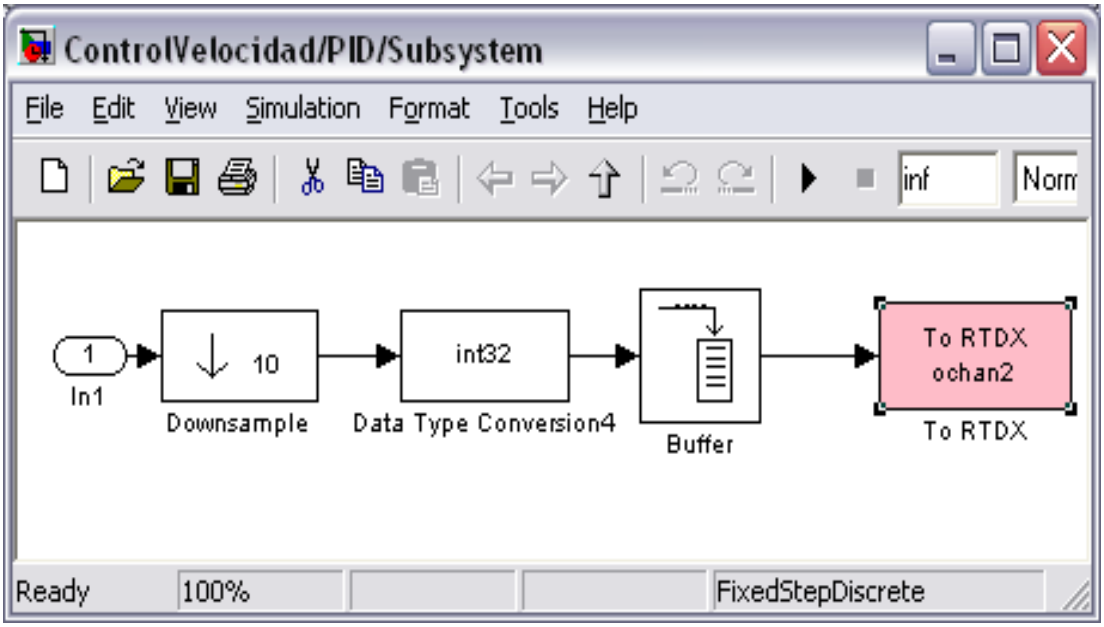


Fig. 3.43. Subsistema para graficar el “Ciclo de trabajo” de la señal.

Con esto se finaliza también el subsistema "*PID*", el cual queda de la siguiente forma:

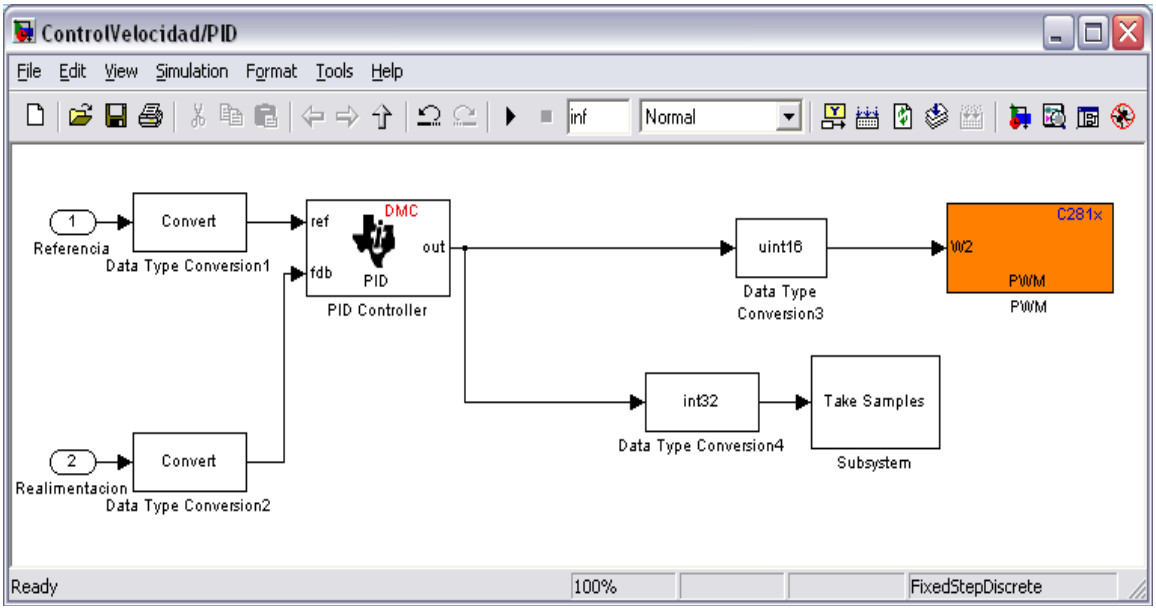


Fig. 3.44. Subsistema 2 "PID".

3.3.1.3 Subsistema 3: Medición de Velocidad.

Este subsistema posee como entrada un bloque ADC, este se encuentra en:
Target Support Package TC2 – C281x DSP Chip Support – ADC

Sus parámetros son los siguientes:
Para la pestaña *'Control ADC'*:

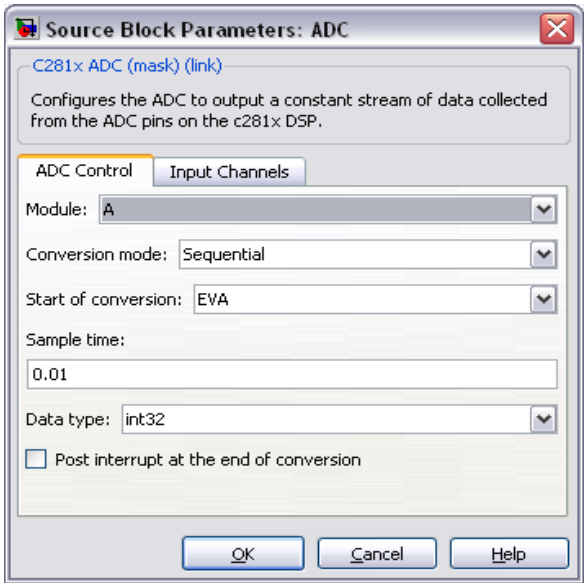


Fig. 3.45. Parámetros definidos a la pestaña "ADC Control" del bloque "ADC".

Para la pestaña *'Canales de entrada'*:

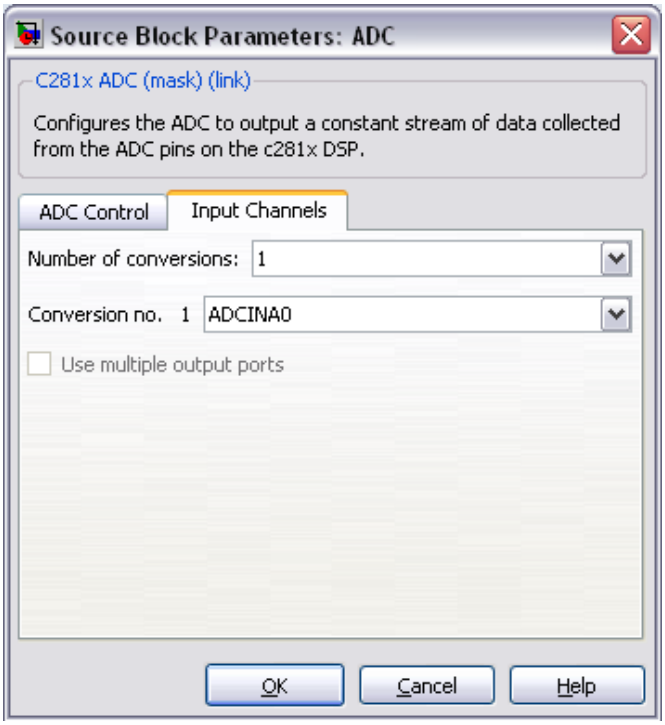


Fig. 3.46. Parámetros definidos a la pestaña "Input Channels" del bloque "ADC".

A la salida de este bloque, se conectan en paralelo un bloque de conversión de tipo de dato y un nuevo subsistema, los parámetros del bloque de conversión de tipo de dato son los siguientes:

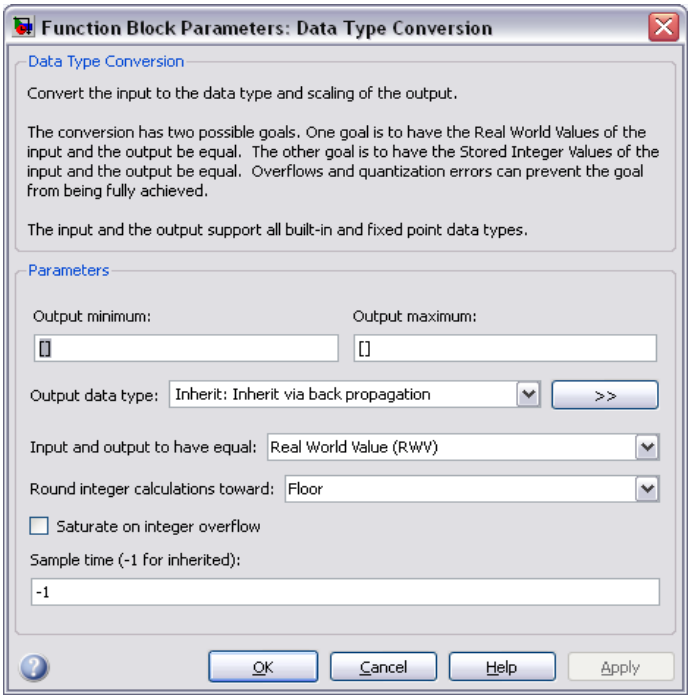


Fig. 3.47. Parámetros definidos al bloque "Data Type Conversion" del subsistema 3.

El nuevo subsistema que se ingresa es similar al subsistema que se introdujo dentro del subsistema '*PID*' a excepción de que este no posee el bloque de conversión de tipo de datos, los parámetros del bloque 'To RTDX' son los siguientes:

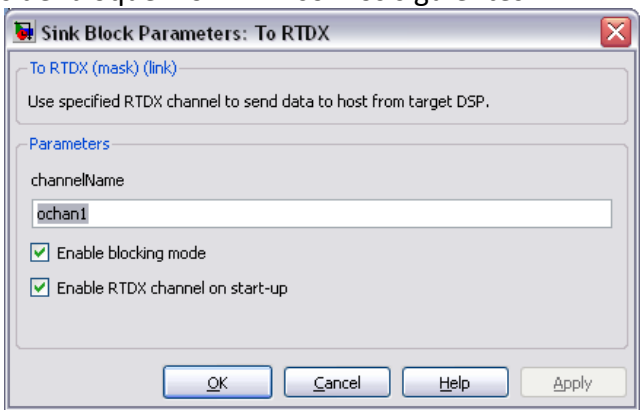


Fig. 3.48. Parámetros definidos al bloque "To RTDX".

El subsistema queda de la siguiente forma:

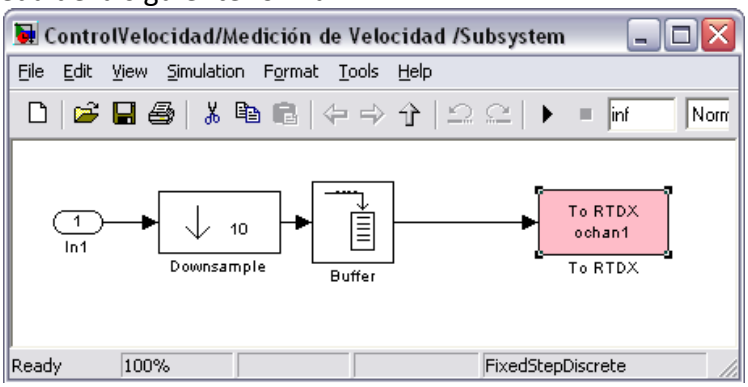


Fig. 3.49. Subsistema para graficar la señal "PID" del controlador.

El subsistema 'Medición de Velocidad' queda de la siguiente forma:

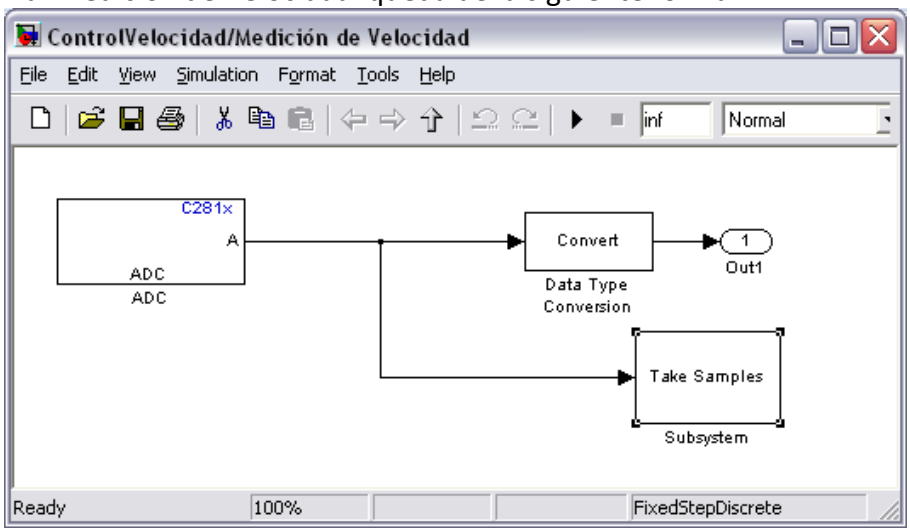


Fig. 3.50. Subsistema 3 "Medición de Velocidad".

Definición de bloques adicionales:

Dentro de la ventana del programa principal se agrega un bloque F2812 eZdsp este se encuentra en:
Target Support Package TC2 – C2000 Target Preferences – F2812 eZdsp

Hay que tener en cuenta que para definir sus parámetros, se debe tener conectada la DSP a la PC de lo contrario se produce el siguiente error:

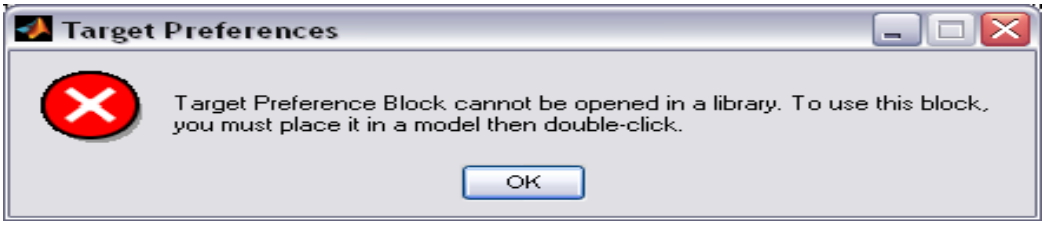


Fig. 3.51. Mensaje de error.

A este bloque se le definen los siguientes parámetros:

Para la pestaña “Board Info”:

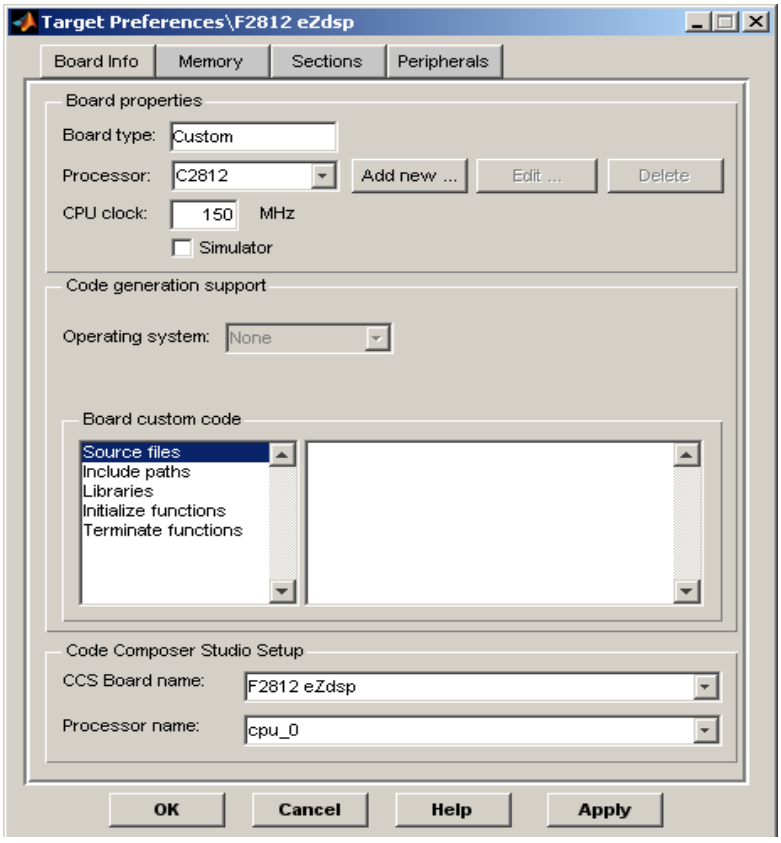


Fig. 3.52. Parámetros definidos en la pestaña “Board Info” del bloque “F2812 eZdsp”.

Para la pestaña “Memory”:

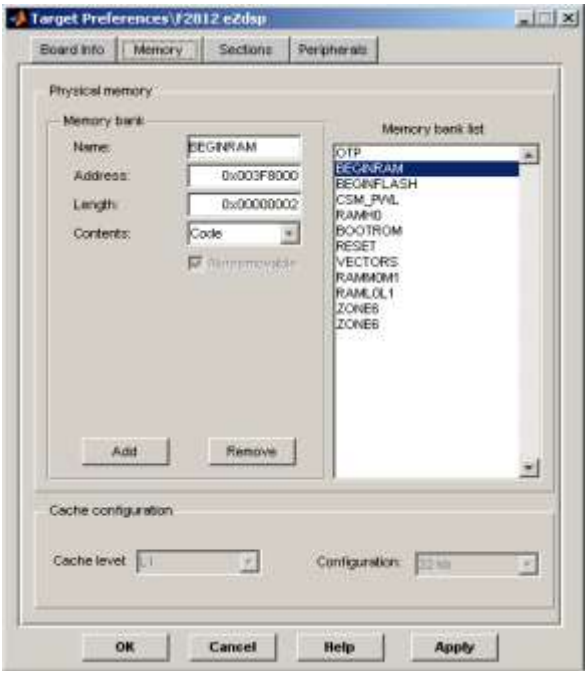


Fig. 3.53. Parámetros definidos en la pestaña “Memory” del bloque “F2812 eZdsp”.

Para la pestaña “Sections”:

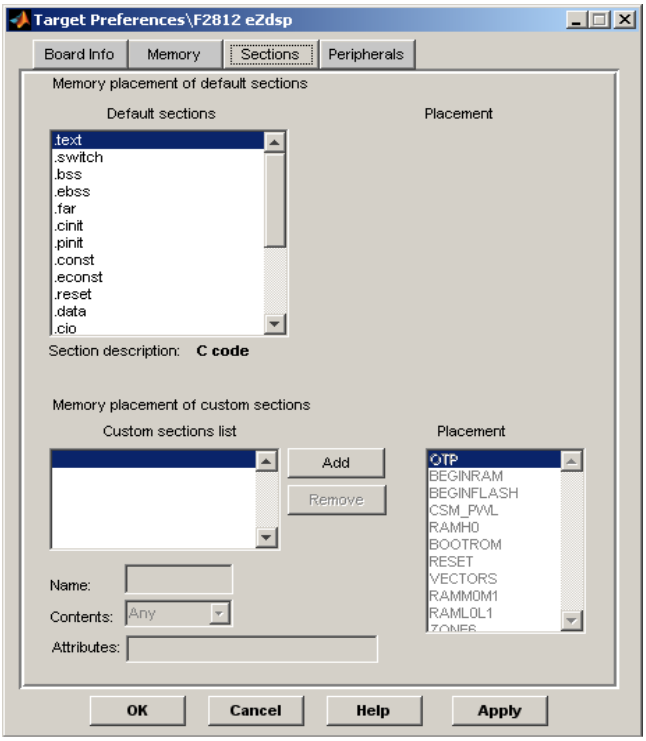


Fig. 3.54. Parámetros definidos en la pestaña “Sections” del bloque “F2812 eZdsp”.

Para la pestaña “Peripherals”:

Fig. 3.55. Parámetros definidos en la pestaña “Peripherals” del bloque “F2812 eZdsp”.

Para finalizar con la creación del controlador de velocidad, se deben agregar los botones, estos se construyen dando doble clic en la parte donde se quieren ubicar, luego se coloca el nombre que este botón poseera, como si se estuviera agregando una viñeta, cuando ya se tiene creado el texto del botón, se da clic derecho sobre el y se selecciona la opción “*Annotation Properties…*”.

En la parte inferior de esta se define el archivo al cual llamara el botón, en este caso se llama al archivo “*correrconrtrolvelocidadmotorDC*”. Para el caso del botón que llama el Script, lo que se coloca en este espacio es: “*edit correrconrtrolvelocidadmotorDC*” para que al dar clic sobre el botón, se abra la ventana del Script.

El controlador completo queda de la siguiente forma:

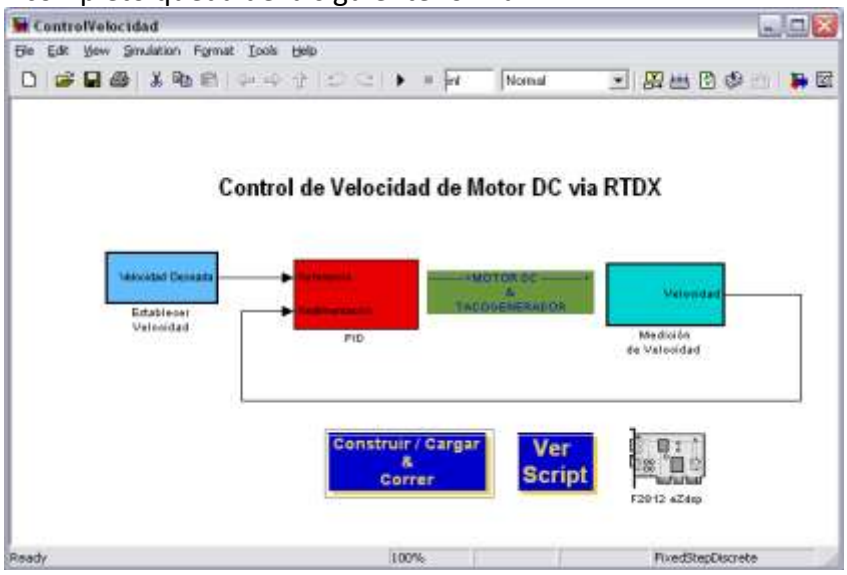


Fig. 3.56. Controlador de Velocidad.

3.3.2 DESCRIPCIÓN DE FUNCIONAMIENTO DEL CONTROLADOR DE VELOCIDAD VÍA RTDX.

El funcionamiento del controlador se basa en el uso de los bloques RTDX, estos sirven para un intercambio de datos en tiempo real entre Matlab y la DSP utilizando como medio de enlace el programa "*Code Composer Studio*", el proceso inicia en el subsistema "*Establecer Velocidad*".

3.3.2.1 Establecer Velocidad:

Este subsistema comienza su funcionamiento definiendo un valor digital inicial de 1000 mediante el bloque "*From RTDX*" con un tiempo de muestreo de 10 milisegundos, luego actualiza su valor según varié la barra deslizante de la GUI; este bloque RTDX es definido como 'ichan1' lo que indica que es el canal de entrada 1, luego, este valor es convertido a punto fijo de 32 bits (Fixed Point) mediante el bloque de conversión de tipo de datos y se envía a la salida la cual llega a la entrada del subsistema "*PID*".

3.3.2.2 PID:

Este subsistema posee dos entradas, en la entrada '*Referencia*' que recibe el dato de referencia proveniente del subsistema '*Establecer Velocidad'*, la entrada *'Realimentación'* recibe el dato proveniente del subsistema *'Medición de Velocidad'* por lo que su función es tomar el dato de realimentación del sistema.

Ambos datos llegan a bloques de conversión de tipo de datos, los cuales los convierten a punto fijo de 32 bits y de estos salen a un controlador PID, en este se definen los valores para las ganancias, proporcional, integral y derivativa, del sistema, así como, los valores de salida que tendrá el controlador, tanto el mínimo como el máximo, se le define una salida mínima de 1000 y una máxima de 4000; hay que tener en cuenta que con estos valores lo que se representa son las revoluciones del motor, ósea la velocidad del motor y se define un mínimo de 1000 ya que con valores menores el motor no gira.

La salida del controlador PID envía un valor para establecer la velocidad, siempre de punto fijo, pero este varía según sus parámetros de entrada, este compara la velocidad deseada, que es la velocidad de referencia, con la velocidad medida que es la velocidad de realimentación y genera un valor digital proporcional al valor DC que se aplica al motor, la salida va a dos bloques de conversión de tipo de datos conectados en paralelo, uno de ellos convierte a entero con signo de 16 bits y el otro a entero sin signo de 32 bits.

El bloque que convierte a entero con signo de 16 bits envía su salida a un bloque PWM, este tiene la función de convertir el valor digital que recibe a un voltaje entre 0 V y 3.3 V en uno de los pines PWM de la DSP, esto lo hace por medio de variaciones en el ciclo de trabajo de la señal que envía al pin de la DSP y un filtro paso bajo, el valor de voltaje generado se define por la siguiente ecuación:

$$V = D \cdot V_{ss}$$

Donde:

V	Voltaje generado
D	PWM Duty cicle (Ciclo de trabajo del PWM)
V_{SS}	Voltaje de alimentación de la DSP (3.3V)

TABLA 3.6: Parámetros utilizados en ecuación de voltaje generado.

El otro bloque de conversión de tipo de dato a entero sin signo de 32 bits se envía a un nuevo subsistema, este tiene la función de tomar muestras del valor digital de salida del controlador PID ya convertido a entero de 32 bits y estas muestras enviarlas a un bloque RTDX definido como 'ochan2' (canal de salida 2).

3.3.2.3 Medición de Velocidad:

Este subsistema tiene como función tomar, por medio de uno de los pines ADC o convertidores analógico-digital de la DSP, el valor analógico generado en el tacómetro del motor (que es un valor con signo negativo) y convertirlo en digital para obtener el valor de realimentación que le llega al controlador PID del subsistema de Corrección de Velocidad.

Para realizar este proceso, se utiliza un bloque ADC, este captura los datos ubicados en los pines de conversión analógica a digital de la DSP y convierte el valor negativo de voltaje a un valor digital que se encuentra en el rango de 0 a 4096 y este valor es enviado a su salida; a este bloque se le define como canal de entrada el 'ADCINA0' (Analogic to Digital Converter IN A0) que es el 'Convertidor Analógico-Digital Entrada A0'; se le define un tiempo de muestreo de 10 milisegundos y el tipo de dato como entero de 32 bits, la salida de este bloque se conecta a un bloque de conversión de tipo de dato y un subsistema, estos se encuentran en paralelo.

El bloque de conversión de tipo de dato tiene como función heredar el valor de la entrada a la salida para así poder enviar ese valor a la entrada "*Realimentacion*" del subsistema PID.

El subsistema tiene la función de tomar muestras del valor digital de salida del bloque ADC convertido a entero de 32 bits y estas muestras enviarlas a un bloque RTDX definido como 'ochan1' (canal de salida 1) para la comunicación en tiempo real.

3.3.2.4 Bloque F2812 eZdsp:

Este bloque tiene la función de realizar el enlace entre Simulink y el software Code Composer Studio (CCS) que es el software con el cual se realiza la comunicación con la DSP. La ausencia de este bloque provocaría que no se pueda dar la comunicación en tiempo real entre el diseño del controlador creado por medio de Simulink y la DSP que se comunica directamente con el circuito de interfaz.

3.3.2.5 Botones del controlador:

Ver Script:
Al dar clic en este botón, se abre una ventana en la cual se muestra el Script completo que se ha diseñado para hacer el llamado a todos los códigos utilizados para el controlador, en este se hace el llamado a los códigos que se utilizan para la creación de la Interfaz Grafica de Usuario (GUI) entre otros, la ventana tiene la siguiente forma:

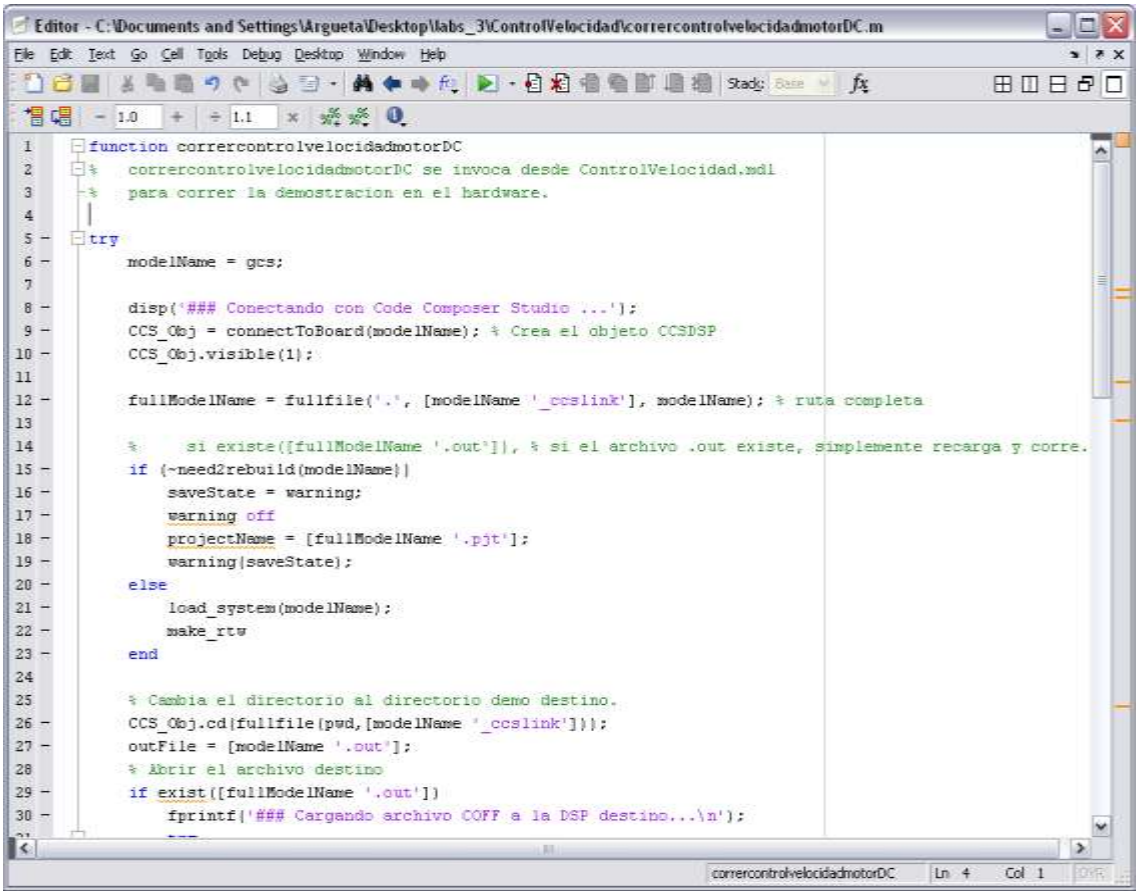


Fig. 3.57. Script principal del controlador de velocidad.

La creación del script se basa en uno de los script que posee Matlab como ejemplo, a este se le dio el nombre de *"correrconrtolvelocidadmotorDC.m".*

Este archivo llama a dos archivos .m que también se basan en los script de ejemplo que trae Matlab aunque se realizan algunas modificaciones a estos, uno de los archivos a los que se llama es el "*speedcontrolDC*", este se encarga de hacer el enlace entre la GUI y el controlador, el otro archivo es el "*speedcontrolDCLoop*", este archivo realiza la comunicación entre los bloques RTDX de Simulink y la GUI de Matlab, este actualiza los datos que se capturan en tiempo real en los bloques RTDX y los envía a las graficas de la GUI para que se muestren en tiempo real.

Construir/Cargar & Correr:

Al dar clic en este botón, se debe construir, cargar y correr en tiempo real el controlador, este proceso inicia abriendo la Interfaz Grafica de Usuario (GUI) que posee la siguiente forma:

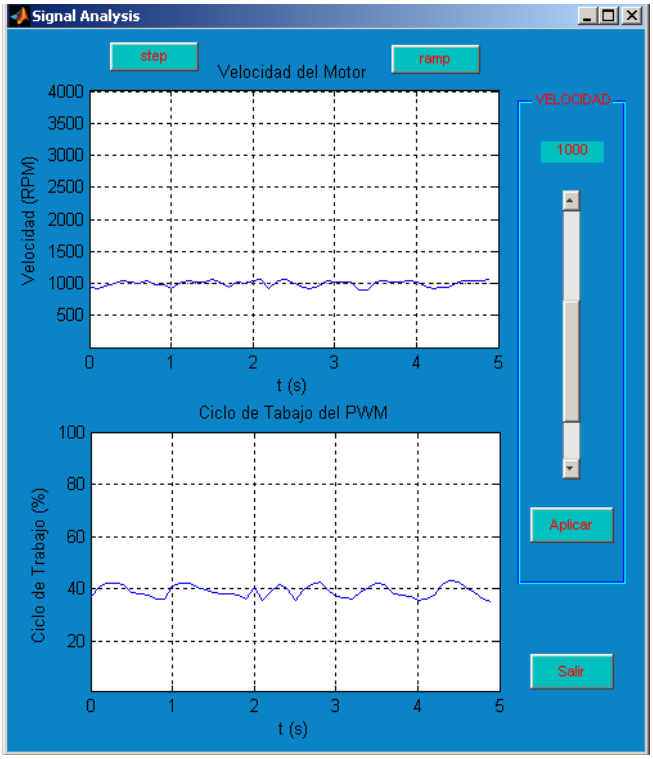


Fig. 3.58. Interfaz Grafica de Usuario (GUI) del controlador de velocidad.

En esta, la barra deslizante sirve para ajustar la velocidad deseada, como se puede ver esta inicia en un valor de 1000 que es el que se le definió en el subsistema de '*Establecer Velocidad*' al cambiar el valor se debe presionar el botón aplicar para que se de el control en tiempo real.

La grafica de "Velocidad de Motor" muestra las características del PID y como este ajusta la velocidad según el valor que se desee y la grafica del "Ciclo de Trabajo" muestra que tanto esta variando el ciclo de trabajo de la señal PWM que se envía a la DSP, hay que tener en cuenta que las variaciones en el ciclo de trabajo del PWM son las que producen las variaciones en el voltaje en el pin PWM de la DSP.

Con el botón "*Ramp*" se simula la función rampa, esta sirve para calcular la pendiente y el error en estado estable de dicha función, el botón "*Step*" simula la función escalón, esta tiene como propósito servir de ayuda para determinar los parámetros de rendimiento del PID como lo son, los tiempos de levantamiento, asentamiento, el sobrepaso máximo y el número de oscilaciones del sistema.

Con el botón "Salir" se finaliza la operación del controlador.

3.4 CONTROL DE POSICION DE MOTOR DC MEDIANTE RTDX.

En el apartado anterior se hablo de un control PID de velocidad implementado en Matlab Simulink en base a las librerías Real Time Workshop y Embedded IDE Link CC, en el cual se trató específicamente el tema de la configuración y bloques necesarios para su implementación, pero en esta descripción, no se habló de los archivos necesarios para la comunicación de Matlab y con Code Composer Studio que hacen posible la visualización de los datos en tiempo real.

Ahora se tratará el tema de un controlador PID de posición que básicamente tiene en su gran mayoría la misma estructura que el controlador PID de velocidad descrito anteriormente; pero para no redundar en lo ya descrito, solo se dirá que la configuración de los bloques en Simulink y la interfaz, es la misma, ya que los dos están basados en el mismo principio. Por ello en esta parte se trata más a fondo la descripción de los archivos de captura y visualización de datos en tiempo real, y los archivos de enlace de Matlab Simulink con Code Composer Studio.

3.4.1 MODIFICACIÓN PARA UN CONTROLADOR PID DE POSICIÓN.

3.4.1.1 Modificación al Subsistema PID.

Antes de entrar en detalle con los archivos de visualización y comunicación, se describirán las diferencias que existen entre los dos controladores. Esta diferencias radican en el hecho de que el control de velocidad solo se realiza en un solo sentido de giro del motor, y por lo tanto el PID de este controlador nunca proporcionará valores negativos, es mas hasta se le asignó un valor mínimo de salida de 1000 por que debajo de este valor el motor ya no giraba, por otra parte el control PID de posición debe de corregir la posición haciendo movimientos en ambos sentidos de giro del motor, por lo cual en el sub-sistema PID de la ventana principal se realizaron las modificaciones que se pueden observar en la figura 3.59.

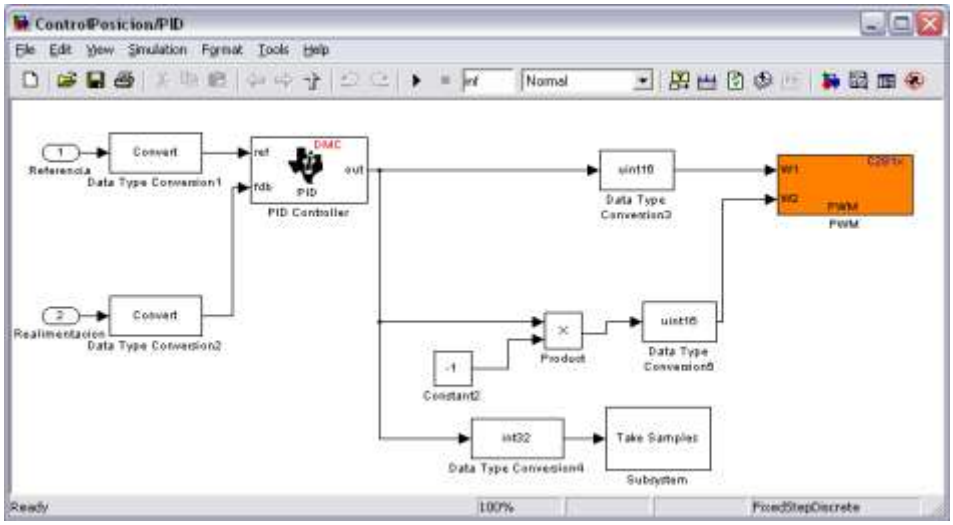


Fig. 3.59. Modificaciones al subsistema PID en el controlador de posición.

Se puede apreciar que en este caso el bloque de salida PWM tiene dos entradas que corresponden a las dos salidas PWM utilizadas para cambiar el sentido de giro del motor, cuando esta activa W1 el motor girará en sentido horario y con W2 en sentido anti horario.

W1 recibe directamente el dato que proviene del controlador PID a través de un bloque de conversión a entero de 16 bit, W2 recibe el dato con polaridad opuesta dado que el dato proveniente del PID es multiplicado por -1 antes de ser convertido a entero de 16 bit. Esta lógica se basa en el hecho de que si alguna de las entradas PWM recibe un valor negativo la salida correspondiente a esa entrada presentará un valor cero dado que no hay valores negativos de ancho de pulso y la otra salida sí presentará un valor, dado que recibe el dato opuesto a la entrada.

La configuración del bloque PWM se muestra en la siguiente figura.

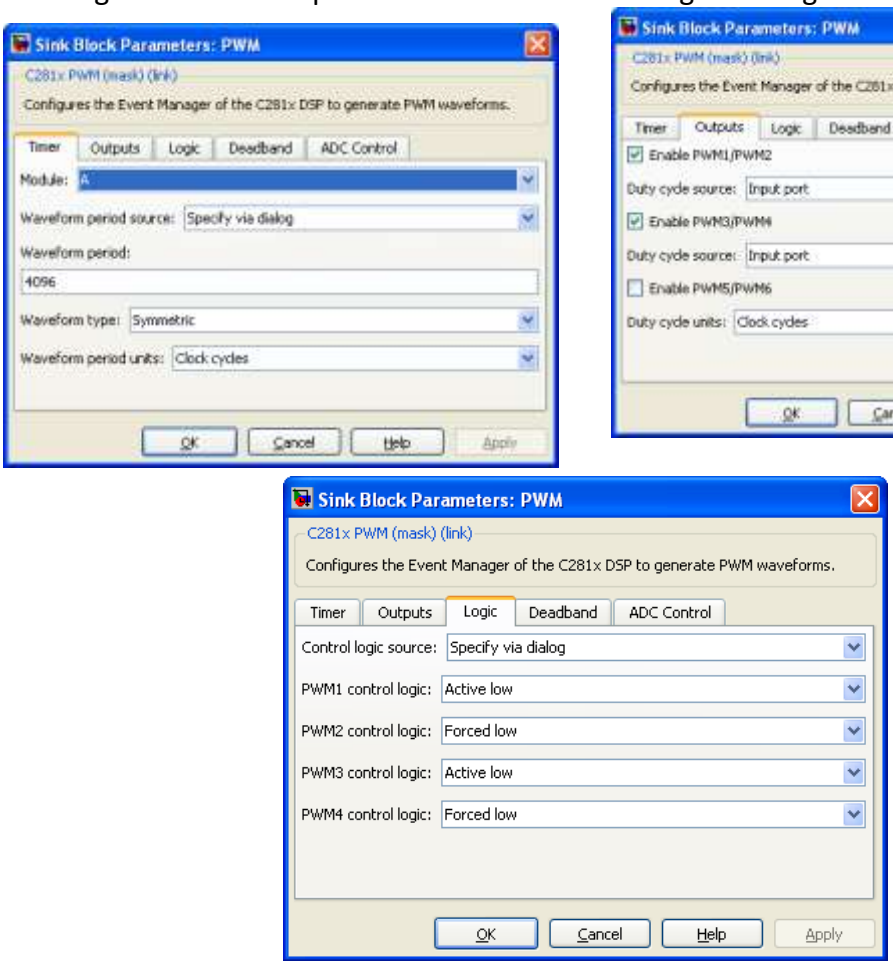


Fig. 3.60. Configuración del bloque PWM.

Como se puede observar se han utilizado las salidas PWM1 y PWM3, ya que Simulink trata las señales PWM como pares acoplados; por ejemplo, PWM1 y PWM2 son señales que comparten el mismo puerto de entrada lo único que las diferencia, es el tipo de lógica

definida, una puede ser de flanco de subida y la otra de bajada o viceversa, esto quiere decir que cuando una este activa la otra no lo estará. Esta situación también se da para las otras señales PWM, lo que da como resultado que a través de Simulink solo se pueden operar 6 señales PWM de forma independiente a pesar que la eZdsp F2812 cuente con 12.

3.4.1.2 Modificaciones al Subsistema Medición de Velocidad.

Otra modificación importante realizada al controlador se da en el subsistema Medición de velocidad, ya que en este, además de la realimentación de velocidad (señal del tacómetro), se le agrega una realimentación de posición (señal del potenciómetro).

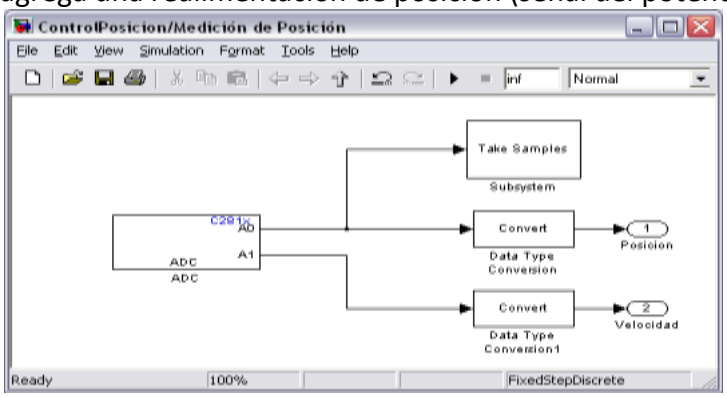


Fig. 3.61. Modificaciones al subsistema Medición de Posición.

La señal de realimentación de velocidad se envía a la entrada “Realimentación” del subsistema PID, la señal de realimentación de posición se envía a un bloque sumador dentro de la ventana principal y la salida de este bloque sumador va a un bloque de ganancia al cual se le ha dado una ganancia de 4; para finalizar la salida de este bloque de ganancia se envía a la entrada de referencia del subsistema PID.

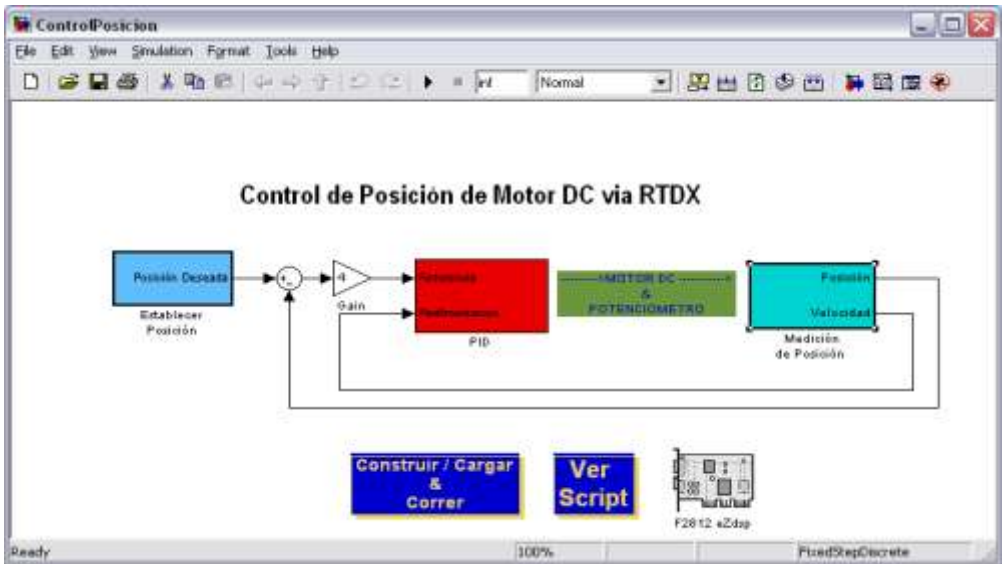


Fig. 3.62. Controlador de posición.

El principio de funcionamiento se basa en la utilización de un controlador proporcional, adicional al controlador PID utilizado. Se comienza restando de la posición de referencia, la posición medida por el sistema, esto genera una señal de error, la cual es multiplicada por un bloque con ganancia 4, ya que mediante la realización de pruebas es la que generaba menos error entre la señal de referencia y la señal medida, luego esta señal de error con ganancia 4 se envía a la entrada de referencia del subsistema PID, en donde es comparada con la señal de realimentación de velocidad, las cuales al ser iguales hacen que el motor se detenga en la posición que se deseaba.

3.4.2 DIAGRAMA JERÁRQUICO DE FICHEROS.

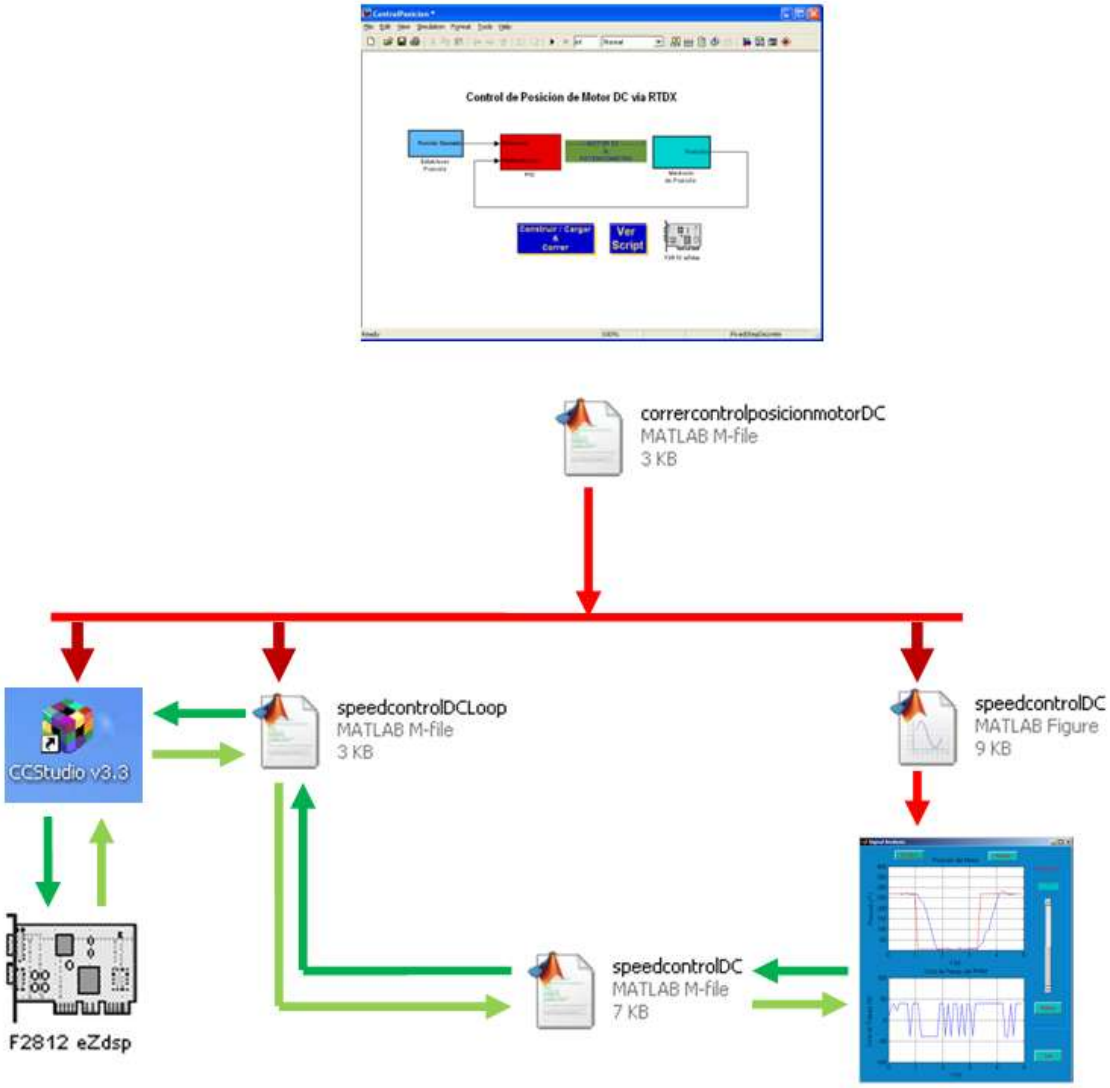


Fig. 3.63. Diagrama de flujo entre ficheros.

El esquema de la figura 3.63 muestra en forma general como se interrelacionan los distintos archivos que componen la ejecución de cualquiera de los dos controladores PID, en este caso el de posición.

3.4.3 DESCRIPCIÓN DE LOS ARCHIVOS.

La descripción de los archivos se hará de arriba hacia abajo y de izquierda a derecha empezando por el archivo "*controlposicion.mdl*", este es el archivo de modelado en Simulink el cual se describió con mucho detalle en el tema "CONTROL DE VELOCIDAD DE MOTOR DC VIA RTDX" y que en general es común para ambos controladores por lo cual no se detendrá mucho en su explicación, en esta ventana hay un botón etiquetado como "Construir Cargar & Correr", al presionarlo se ejecuta el script de Matlab "*correrconrolposicionmotorDC.m*", este script contiene las instrucciones necesarias para abrir CCS, conectarse a CCS, llamar al archivo de interfaz grafica, llamar al archivo que genera la interacción en lazo cerrado y finalizar la ejecución del programa cerrando los canales RTDX y terminando la ejecución del CCS.

A continuación se describen más detalladamente estos pasos.

correrconrolposicionmotorDC.m

Lo primero que hace este script es obtener el nombre del modelo de Simulink con el que se está trabajando actualmente con el comando 'gcs', luego se crea un objeto que cuenta con todas las funciones necesarias para generar una comunicación entre el modelo actual y Code Composer Studio. La función 'visible' de este objeto permite que cuando se conecte a CCS se muestre la ventana del CCS en la barra de tareas de Windows. Por último se genera en el directorio, por defecto de Matlab (generalmente 'C:\Documents and Settings\EIE\Mis documentos\MATLAB), una carpeta, 'nombre del modelo_ccslink', donde se guardan todos los archivos de código C generados por RTW, este directorio se guarda en la variable 'fullModelName'.

```
1  function correrconrolposicionmotorDC
2  %   correrconrolposicionmotorDC se invoca desde ControlPosicion.mdl
3  %   para correr la demostracion en el hardware.
4
5  try
6      modelName = gcs;
7
8      disp('### Conectando con Code Composer Studio ...');
9      CCS_Obj = connectToBoard(modelName); % Crea el objeto CCSDSP
10     CCS_Obj.visible(1);                  % hace visible la ventana del CCS en el escritorio
11
12     fullModelName = fullfile('.', [modelName '_ccslink'], modelName); % ruta completa
```

Fig. 3.64. Conectándose con CCS.

```
14     %     si existe([fullModelName '.out']), % si el archivo .out existe, simplemente recarga y corre.
15     if (~need2rebuild(modelName))
16         saveState = warning;
17         warning off
18         projectName = [fullModelName '.pjt'];
19         warning(saveState);
20     else
21         load_system(modelName);
22         make_rtw
23     end
```

Fig. 3.65. Comprobando la existencia del proyecto.

Si el proyecto fue creado con anterioridad existirá un archivo .out que indica que no es necesario volver a copilar el proyecto, pero si no existe el proyecto es copilado nuevamente por RTW con el comando 'make_rtw'. Esto se visualiza en la figura 3.65.

```
25        % Cambia el directorio al directorio demo destino.
26 -      CCS_Obj.cd(fullfile(pwd,[modelName '_ccslink']));
27 -      outFile = [modelName '.out'];
```

Fig. 3.66. Indicándole el directorio del proyecto a CCS.

Si la conexión con CCS fue exitosa, se procede a la carga y la ejecución del proyecto, para lo cual se tiene un bucle 'if' dentro del cual existen 3 sentencias 'try-catch', cada una de las cuales comprueba si existen errores en el código que se encuentra dentro de ella; la primera, prueba si se puede resetear la DSP, luego copilar el código y cargarlo a la DSP, si esto es posible no se ejecuta lo que está en la sentencia 'catch', si no que, se ejecuta la siguiente sentencia 'try'.

```
29 -      if exist([fullModelName '.out'])
30 -          fprintf('### Cargando archivo COFF a la DSP destino...\n');
31 -          try
32 -              CCS_Obj.reset;
33 -              CCS_Obj.load(outFile,100);
34 -          catch
35 -              clear CCS_Obj;
36 -              msg = lf_message('demos:runDemoLoadError');
37 -              errordlg(msg, 'Error');
38 -              return
39 -          end
40 -          try
41              % configura RTDX
42 -              r = CCS_Obj.rtdx;
43 -              r.disable;
44 -              r.configure(64000,4,'continuous');
45 -              r.open('ochan1', 'r');
46 -              r.open('ochan2', 'r');
47 -              r.open('ichan1','w');
48 -              r.enable;
49
50 -              CCS_Obj.run;
51
52              % Abre la figura GUIDE
53 -              figHandle = speedcontrolDC;
54 -              figHandle.CCS_Obj = CCS_Obj;
55 -              figHandle.sendRequest = 0; %inicializa la bandera sendRequest
56              %llama al lazo principal actualizado
57 -              speedcontrolDCLoop(CCS_Obj,figHandle);
58              %sale del lazo
59 -              delete(figHandle.output);
60 -              disp('Fin de Simulación del Control de Posición con Motor DC')
61 -          catch ME
62 -              errordlg(ME.message, 'Error');
63 -              return
64 -          end
65          %Cierra los canales RTDX
66 -          try
67 -              r.disable('ichan1');
68 -              r.disable('ochan1');
69 -              r.disable('ochan2');
70 -              r.disable;
71 -          catch
72              % si los canales no estan abiertos, no cierra nada
73 -          end
74 -          halt(CCS_Obj);
75 -          CCS_Obj.reset;
76 -      else
```

Fig. 3.67. Cargando y ejecutando el proyecto en la DSP.

La segunda sentencia ‘try’ hace lo siguiente, vincula ‘r’ con la funcionalidad RTDX que proporciona CCS, cada vez que se quiera acceder a esta función se hará referencia con el símbolo ‘r’, se deshabilitan los RTDX para poderlos configurar; configura los RTDX con 4 buffer de 64kbits de ancho cada uno este ancho puede ser variable pero no puede ser menor a 1024bits, abre tres canales RTDX, dos de salida (ochan1 y ochan2) y uno de entrada (ichan1) y habilita su utilización. Se corre la aplicación por medio de CCS en la DSP, se abre la interfaz grafica de usuario, esta interfaz y el objeto CCS_obj son pasados a la función ‘speedcontrolDCLoop.m’ que es la encargada de enviar y presentar los datos en tiempo real desde la DSP, en esta parte la ejecución de este script se detiene hasta que la función ‘speedcontrolDCLoop’ le retorne el mando, cuando esto sucede es porque la ejecución a finalizado y se procede a cerrar la interfaz grafica, deshabilitar los canales RTDX, detener la DSP y resetearla.

speedcontrolDCLoop.m

El archivo ‘speedcontrolDCLoop.m’ empieza con los siguientes comandos.

```
1   function speedcontrolDCLoop(CCS_Obj,handles)
2
3 -   guidata(handles.output, handles);
4 -   x_axis = 0:0.1:4.9;
5 -   senal_ref = ones(1,50);
6 -   r = CCS_Obj.rtdx;
7 -   new_position = 1000;
8 -   senal_ref = senal_ref .* new_position;
```

Fig. 3.68. Creando ventores para las graficas.

Con este código se guardan todos los datos de la GUI en la variable ‘handles’ cada vez que se desee capturar un evento se realizará atreves de este símbolo, además se crean los vectores que representan la escala en el eje x y la señal de referencia, se inicializa la variable ‘new_position’ con 1000 y a cada valor del vector ‘senal_ref’ se le coloca 1000.

Una sentencia ‘while’ es la que proporciona el lazo cerrado para la captura de datos de forma continua, se ejecuta mientras el objeto ‘CCS_obj’ este corriendo.

```
19 -    while (isrunning(CCS_Obj))
20 -        senal_ref(1:48)=senal_ref(3:50);
21 -        senal_ref(49)=new_position;
22 -        senal_ref(50)=new_position + 11.37*10;
23 -        if (get(handles.pushbutton6,'UserData'))
24 -            new_position = 11.37*get(handles.slider1, 'Value');
25 -            r.writemsg('ichan1',int32(new_position));
26 -            set(handles.pushbutton6,'UserData',0);
27 -        end
28 -        if (get(handles.pushbutton12,'UserData'))
29 -            new_position = 11.37*270;
30 -            r.writemsg('ichan1',int32(new_position));
31 -            set(handles.pushbutton12,'UserData',0);
32 -        end
33 -        if (get(handles.pushbutton13,'UserData'))
34 -            new_position = new_position + 11.37*10;
35 -            r.writemsg('ichan1',int32(new_position));
36 -            if (new_position >= 11.37*340)
37 -                set(handles.pushbutton13,'UserData',0);
38 -            end
39 -        end
```

Fig. 3.69. Captura de eventos ‘step’, ‘ramp’ y aplicar.

Las primeras tres líneas son para generar una señal de referencia actualizando siempre este vector, al valor de 'new_position', luego con cada 'if' se capturan los eventos de presionar botón aplicar, botón 'step' o 'ramp' respectivamente. Con el botón aplicar solo se establece una nueva posición que se captura desde la barra deslizante y se escribe en el canal de entrada 'ichan1', se posiciona la variable 'UserData' a '0' para que no vuelva a entrar en esta sentencia 'if'. El botón 'step' hace el mismo procedimiento solo que el valor enviado a 'ichan1' no es capturado desde la barra sino que está fijado en 270º. El botón 'ramp' corresponde al pushbotton 13, en este caso al valor de 'new_speed' se le suman 10 grados con cada interacción del lazo 'while', hasta que el valor de 'new_speed' supera 340 grados la variable 'UserData' es establecida a 0 y ya no se incrementa mas.

```
41 -          numMsgsOchan1 = r.msgcount('ochan1');
42 -          if (numMsgsOchan1 > 1),
43              % flush frames as necessary to maintain real-time display
44 -            r.flush('ochan1', numMsgsOchan1-1);
45 -          end
46 -          if (numMsgsOchan1)
47 -              position = r.readmsg('ochan1', 'int32');
48 -          end
49
50 -          numMsgsOchan2 = r.msgcount('ochan2');
51 -          if (numMsgsOchan2 > 1),
52              % flush frames as necessary to maintain real-time display
53 -            r.flush('ochan2', numMsgsOchan2-1);
54 -          end
55 -          if (numMsgsOchan2)
56 -              pid = r.readmsg('ochan2', 'int32');
57 -          end
```

Fig. 3.70. El proceso de extracción de datos vía RTDX y actualización de las graficas.

En la figura 3.71 se muestra la extracción de datos en tiempo real desde la RTDX. Primero se extrae el numero de mensajes del contador de mensajes para el canal de salida 'ochan1', luego si ese contador de mensajes tiene un número mayor que 1, se procede a extraer el último mensaje de la pila, luego se lee el mensaje de este canal interpretándolo como un vector, de enteros de 32bits y lo se guarda en la variable position; este será el valor de posición leído por el puerto analógico de la DSP, el mismo procedimiento se realiza para 'ochan2'.

```
58 -          if ((numMsgsOchan1 ~=0) && (numMsgsOchan2 ~= 0))
59 -              axes(handles.axes3);
60 -              plot(handles.axes3,x_axis, position/11.37,x_axis, senal_ref./11.37, 'r-');
61 -              title(handles.axes3,'Posición del Motor');
62 -              xlabel(handles.axes3,'t (s)');
63 -              ylabel(handles.axes3,'Posición ( ° )');
64 -              grid(handles.axes3,'on');
65 -              axis(handles.axes3,[0 5 1 400]);
66
67 -              axes(handles.axes4);
68 -              cycle = (double(pid).*100./4095);
69 -              plot(handles.axes4,x_axis, cycle);
70 -              title(handles.axes4,'Ciclo de Tabajo del PWM');
71 -              xlabel(handles.axes4,'t (s)');
72 -              ylabel(handles.axes4,'Ciclo de Trabajo (%) ');
73 -              grid(handles.axes4,'on');
74 -              axis(handles.axes4,[0 5 -100 100]);
75 -          end
```

Fig. 3.71. Actualización de las graficas.

```
76 -            if (get(handles.pushbutton7,'UserData'))
77 -                break
78 -            end
79
80 -        end
81 -    catch ME
82 -        errordlg(ME.message);
83 -    end
```

Fig. 3.72. Salir del lazo 'while' por medio de la pulsación de pushbutton7 (salir).

Si se presiona el botón salir se sale del lazo y se retorna el mando al programa 'correrconrolposicionmotorDC.m' y se termina la ejecución.

speedcontrolDC.m

En este archivo se definen todos los elementos de los que se compone la interfaz, cuando se abre la interfaz se corre la siguiente función '*figure_speed_OpeningFcn*', en esta se crea el objeto 'handles.output' que es requerido para cerrar la interfaz, además se especifican algunos parámetros de las graficas como lo son el titulo, nombre de los ejes, escala de los ejes y la activación del grid.

```
50    function figure_speed_OpeningFcn(hObject, eventdata, handles, varargin)
51    % This function has no output args, see OutputFcn.
52    % hObject    handle to figure
53    % eventdata  reserved - to be defined in a future version of MATLAB
54    % handles    structure with handles and user data (see GUIDATA)
55    % varargin   command line arguments to figure_speed (see VARARGIN)
56
57    % Choose default command line output for figure_speed
58 -  handles.output = hObject;
59
60    % timer('TimerFcn',{'update_figures',handles}, 'Period', 5);
61    %
62 -  guidata(hObject, handles);
63 -  axes(handles.axes3);
64 -  title('Posición del Motor');
65 -  xlabel('t (s)');
66 -  ylabel('Posicion ( ° )');
67 -  grid on;
68 -  axis([0 5 1 400]);
69 -  axes(handles.axes4);
70 -  xlabel('t (s)');
71 -  ylabel('Ciclo de Tabajo del PWM');
72 -  axis([0 5 -100 100]);
73 -  grid on;
```

Fig. 3.73. Función de apertura del archivo de interfaz.

Cada vez que se interactué con la barra deslizante se llama esta función que lo único que hace es actualizar el valor que se muestra en el cuadro texto; captura el valor, lo almacena en la variable 't', lo redondea, lo convierte a cadena de caracteres y lo envía a handles.text8.

```
88    function slider1_Callback(hObject, eventdata, handles)
89    % hObject    handle to slider1 (see GCBO)
90    % eventdata  reserved - to be defined in a future version of MATLAB
91    % handles    structure with handles and user data (see GUIDATA)
92
93    % Hints: get(hObject,'Value') returns position of slider
94    %        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
95 -  t = get(hObject,'Value');
96 -  set(handles.text8, 'string', num2str(ceil(t)));
```

Fig. 3.74. Función que se ejecuta cuando se interactúa con la barra deslizante.

```
117     function pushbutton6_Callback(hObject, eventdata, handles)
118     % hObject    handle to pushbutton6 (see GCBO)
119     % eventdata  reserved - to be defined in a future version of MATLAB
120     % handles    structure with handles and user data (see GUIDATA)
121
122 -   set(handles.pushbutton6,'UserData',1);
123
124     % --- Executes on button press in pushbutton7.
125     function pushbutton7_Callback(hObject, eventdata, handles)
126     % hObject    handle to pushbutton7 (see GCBO)
127     % eventdata  reserved - to be defined in a future version of MATLAB
128     % handles    structure with handles and user data (see GUIDATA)
129
130 -   set(handles.pushbutton7,'UserData',1);
170     function pushbutton12_Callback(hObject, eventdata, handles)
171     % hObject    handle to pushbutton12 (see GCBO)
172     % eventdata  reserved - to be defined in a future version of MATLAB
173     % handles    structure with handles and user data (see GUIDATA)
174 -   set(handles.pushbutton12,'UserData',1);
175
176     % --- Executes on button press in pushbutton13.
177     function pushbutton13_Callback(hObject, eventdata, handles)
178     % hObject    handle to pushbutton13 (see GCBO)
179     % eventdata  reserved - to be defined in a future version of MATLAB
180     % handles    structure with handles and user data (see GUIDATA)
181 -   set(handles.pushbutton13,'UserData',1);
```

Fig. 3.75. Funciones que se llaman cuando se presiona algún botón.

Esta es una de las funciones más simples, si se presiona cualquiera de los botones se llama la función respectiva y para cualquiera de ellas solo se establece la variable ‘UserData’ a ‘1’. Esta variable está definida dentro de cada objeto pushbutton, no se está cambiando la misma variable, solo se cambia el valor de la variable definida dentro de cada objeto.

Cuando quiere saberse si el botón fue presionado se comprueba así:

get(handles.pushbutton12,’UserData’)

La función ‘get’ lee el valor del atributo ‘UserData’ perteneciente al objeto pusbutton12.

3.5 PRACTICA 1. DETERMINACION DE PARAMETROS DEL SISTEMA.

OBJETIVOS:

1. Familiarizarse con el uso de Matlab – Simulink, específicamente con las librerías 'Real Time Workshop' y 'Embedded IDE Link' que este posee.

2. Aprender a simular sistemas de bloques en Simulink en tiempo real, utilizando el software Code Composer Studio y la tarjeta DSP F2812.

Equipo Requerido:

1 Modulo Controlador Digital con tarjeta eZdsp F2812.
1 Suministro de potencia PS150E.
1 Unidad Servo amplificador SA150D.
1 Motor DC con taco generador MT150F.

DISCUSION PRELIMINAR.

La función de transferencia de un motor DC se puede aproximar por un modelo de primer orden con constantes desconocidas. Estas constantes se pueden identificar experimentalmente.

El tacómetro proporciona la señal de realimentación para el sistema de control de velocidad. Un diagrama esquemático del tacómetro se da en la figura P1.1.

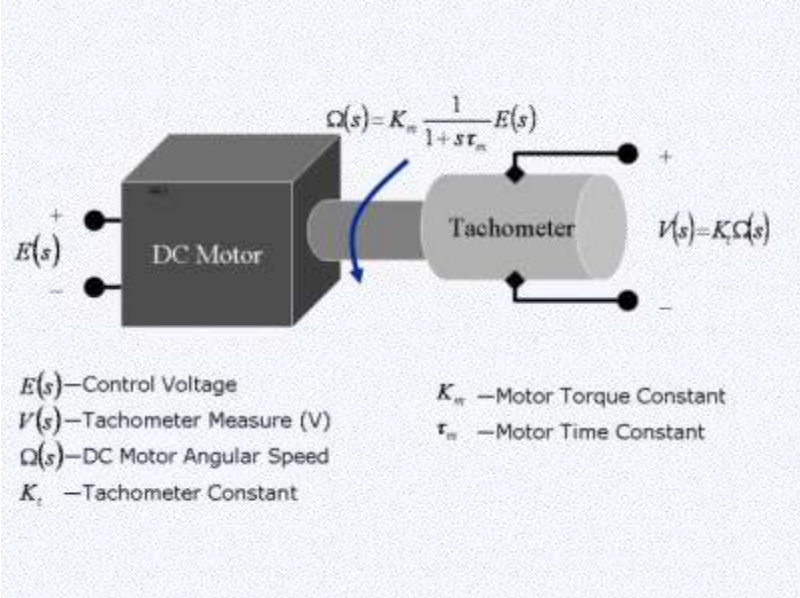


Fig. P1.1. motor DC con tacómetro.

En esta sección se van a identificar experimentalmente las constantes en el modelo matemático de un motor DC y el tacómetro. El proceso consta de dos partes:

1. La medición de la ganancia del sistema.
2. La medición de la constante de tiempo del motor.

PRACTICA 1A. Medición de la ganancia del sistema.

La ganancia del sistema se puede medir mediante la generación de un voltaje de control, usando el bloque PWM de la DSP, obteniendo:

1. El voltaje de entrada al servo amplificador.
2. El voltaje a la salida del tacómetro.

La configuración se muestra en la Figura P1.2.

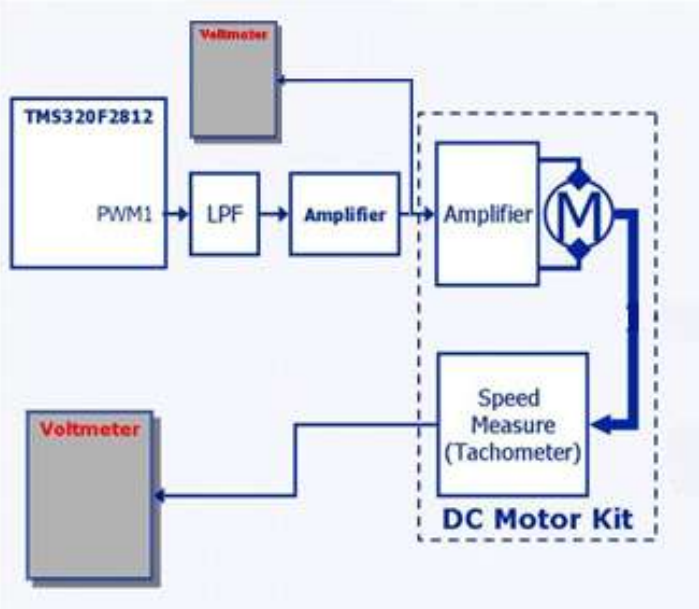


Fig.P1.2. Medición de voltaje del motor y tacómetro.

La figura P1.3 muestra el modelo que se utilizará para generar el voltaje de control.

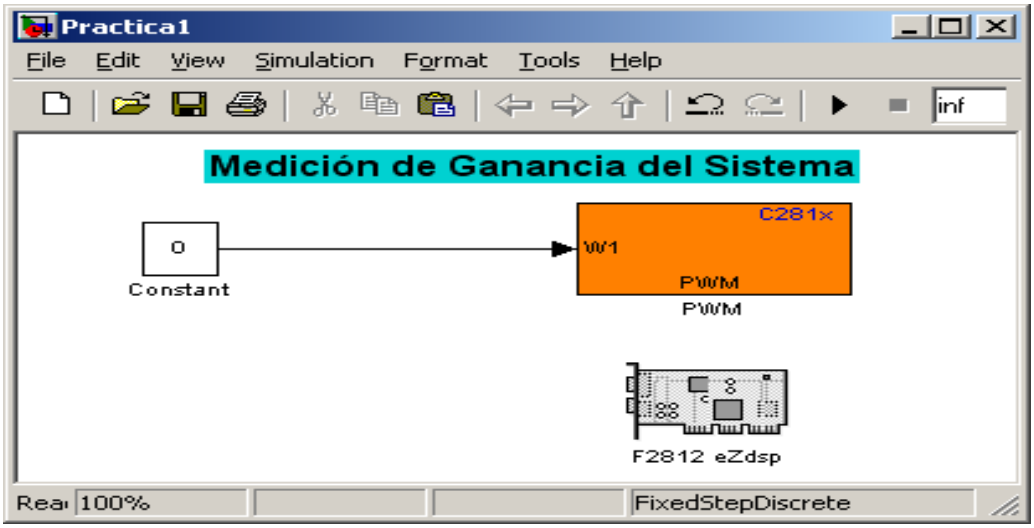


Fig. P1.3. Medición de ganancia del sistema.

El bloque PWM se puede configurar para generar una señal PWM con una frecuencia de 20 kHz (ver Figura P1.4).

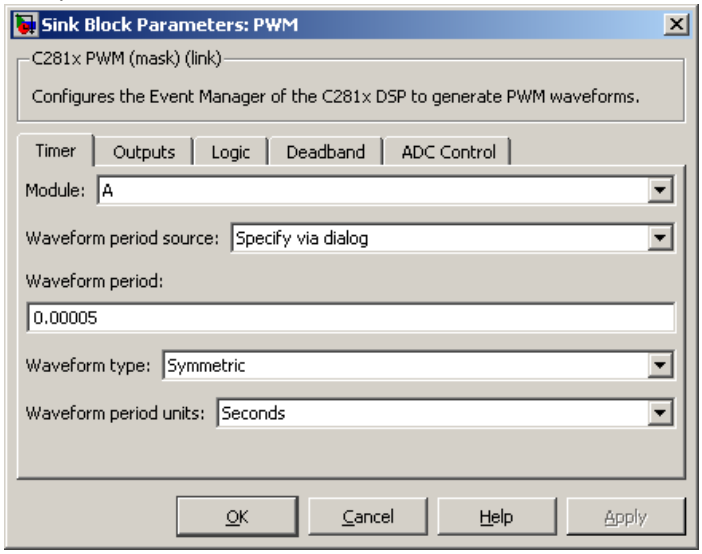


Fig. P1.4. configuración PWM

Procedimiento de medición:

1. Conecte el circuito que se muestra en la Figura P1.5.

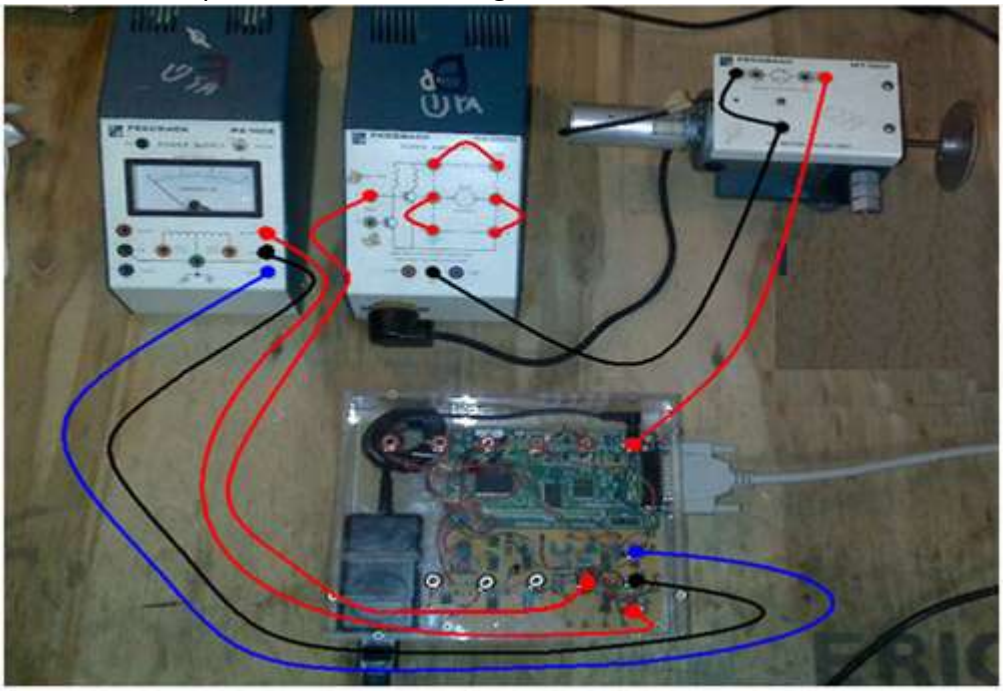


Fig. P1.5. Conexión del kid feedback al Modulo Digital eZdsp F2812.

2. Conectar el cable paralelo del CPU a la tarjeta DSP F2812 y energizar la tarjeta DSP.

3. Siga las siguientes instrucciones para ejecutar el proyecto:

- Dar clic sobre el icono F2812 eZdsp CCStudio v3.1, se abrirá la ventana de la figura P1.6.

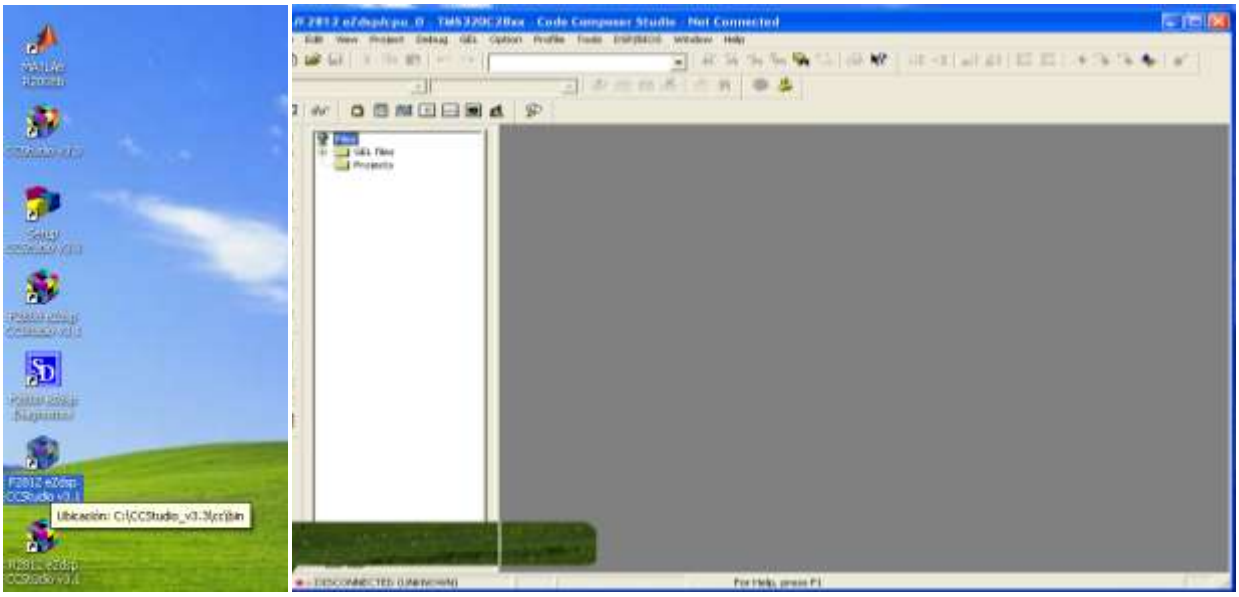


Fig.P1.6. Entorno de CCS.

- Dar clic en el menú 'Debug' y seleccionar la opción 'Connect' como se muestra en la figura P1.7.

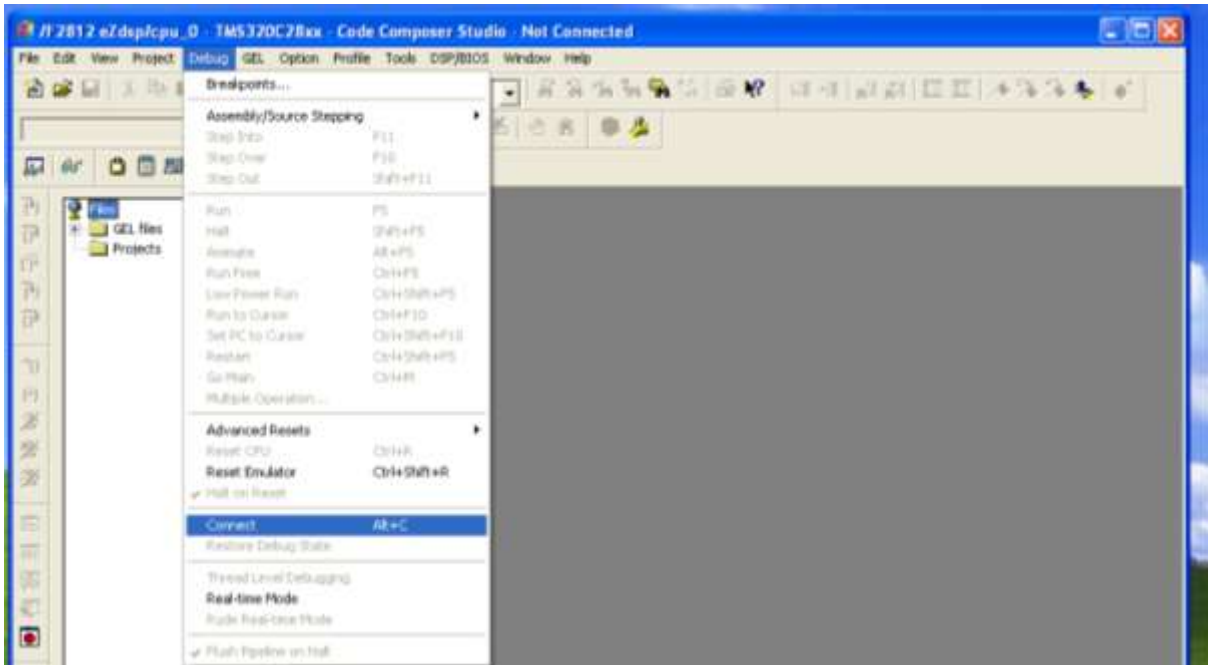


Fig. P1.7. Conectar DSP.

- En la barra de tareas se debe desplegar el texto ‘HALTED’ que indica que la DSP esta correctamente conectada, como se muestra en la figura P1.8.

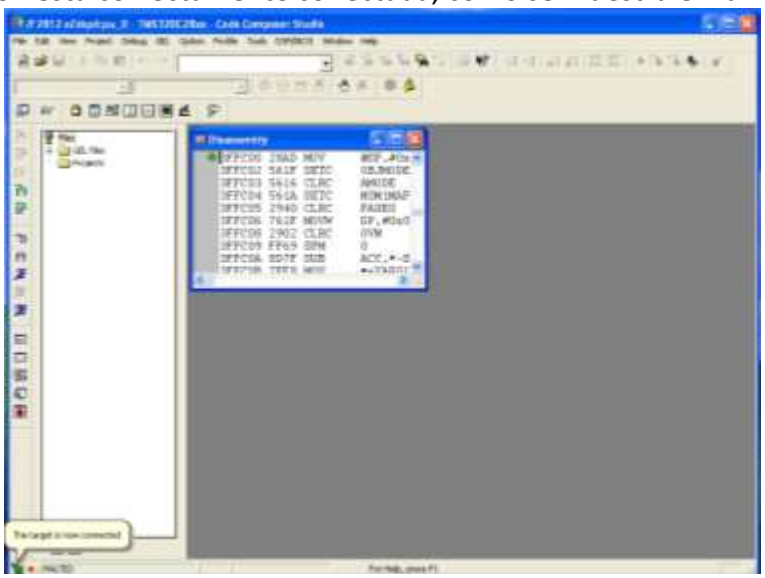


Fig. P1.8. DSP conectada.

- Dar clic en archivo ‘Practica1A.mdl’, se deben observar las ventanas mostradas en la figura P1.9.

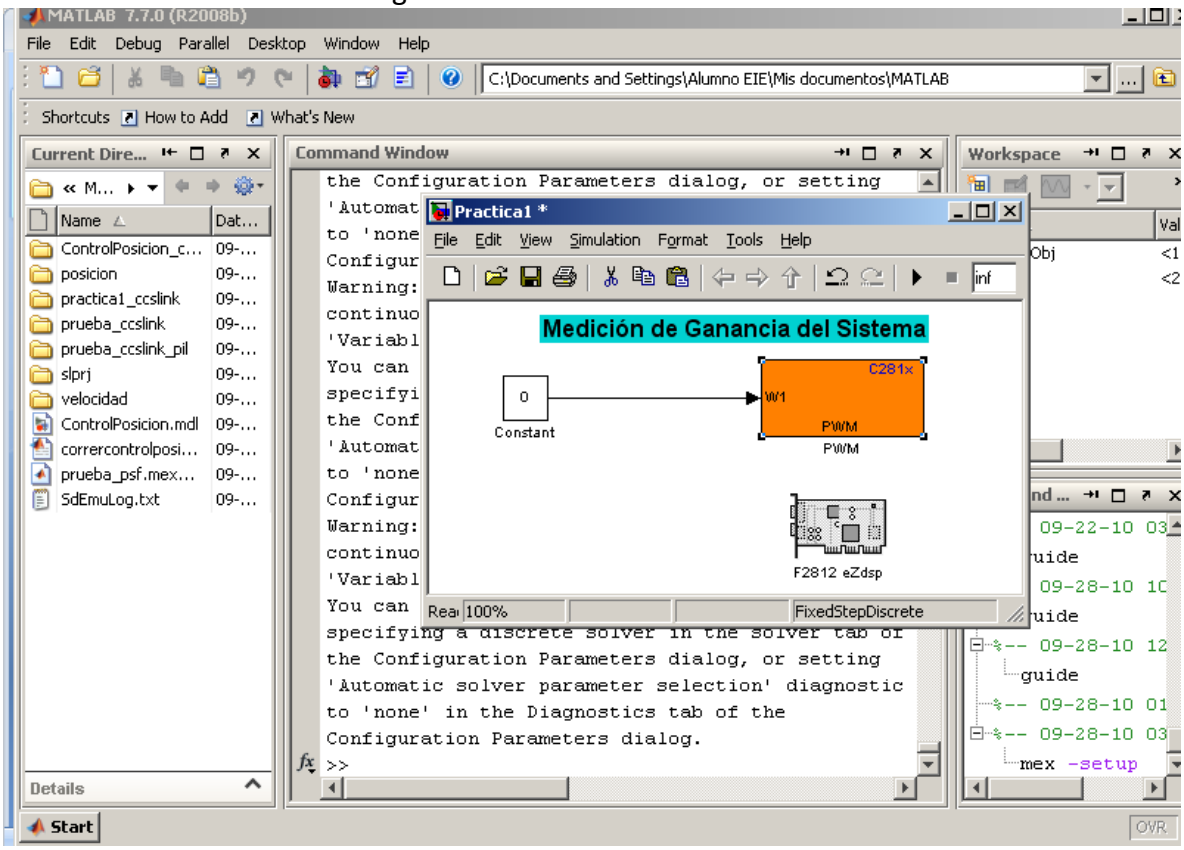


Fig. P1.9. Ventana de Matlab y Proyecto en Simulink.

4. Establecer el valor de la constante con un valor de 0 y seguir los siguientes pasos:
    - Dar clic en el menú 'Simulation' y seleccionar la opción 'Configuration Parameters' como se muestra en la figura P1.10.

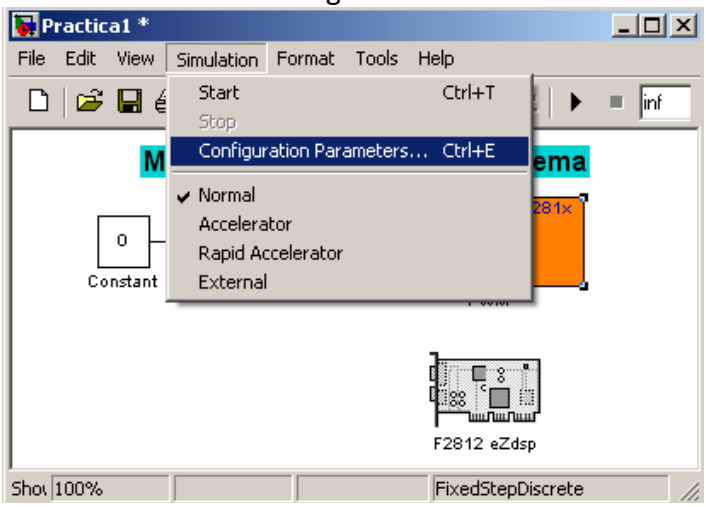


Fig.P1.10. Parámetros de Configuración.

- Se debe observar la ventana de la figura P1.11.

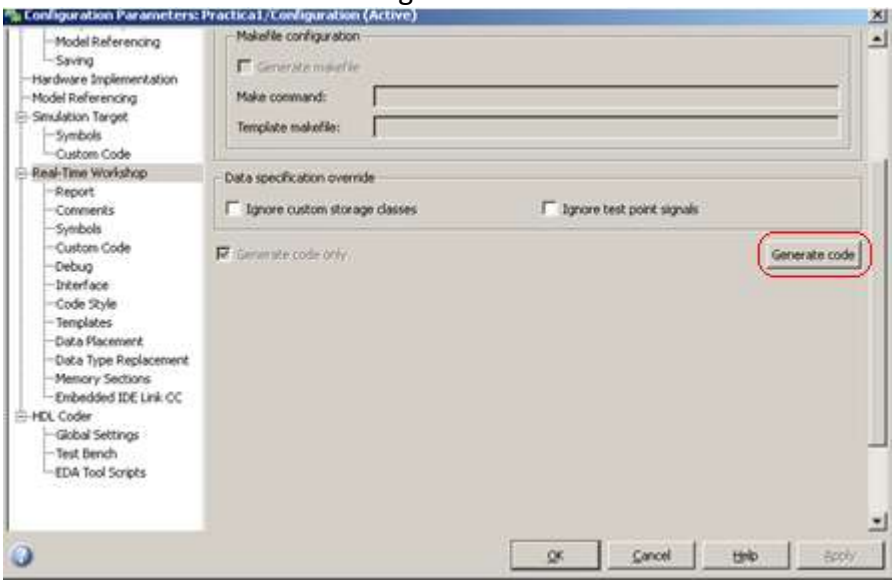


Fig. P1.11. Generar Código.

- Ir a la pestaña 'Real-Time Workshop' y dar clic en el botón 'Generate Code'.

- Cuando el código termine de generarse, se observara en línea de comandos de Matlab el nombre de una serie de archivos que se crearon y en la ventana del Code Composer Studio se debe observar el texto 'RUNNING' en la barra de tareas, se debe encender la fuente PS150E.

5. Medir el voltaje a la entrada del servo amplificador y el voltaje del tacómetro.

6. En la ventana del Code Composer Studio, dar clic en el menú '*Debug*' y seleccionar la opción '*Halt*' para pasar de modo RUNNING a HALT.

7. Repita los pasos 4, 5 y 6 para obtener valores de voltajes en el rango de 0-15 V. Cambiar el Ciclo de trabajo (Valor de la constante 0-100) de 0 a 100% en pasos de 10% y llenar la tabla P1.1.

Ciclo de Trabajo (%)	Entrada Servo amplificador (V)	Salida Tacómetro (V)
0		
10		
20		
30		
40		
50		
60		
70		
80		
90		
100		

Tabla P1.1. Valores para cálculo de ganancia del sistema.

8. De los datos anteriores, obtener una grafica del voltaje del tacómetro vrs. el voltaje de entrada del servoamplificador utilizando algún programa de computadora.

9. Anotar conclusiones y observaciones.

Medición de la constante de tiempo del motor.

La medición de la constante de tiempo se obtendrá mediante la generación de un pulso cuadrado y la medición de la respuesta motora a este estímulo. El principio de funcionamiento se muestra en la Figura P1.12, el montaje experimental se muestra en la Figura P1.13.

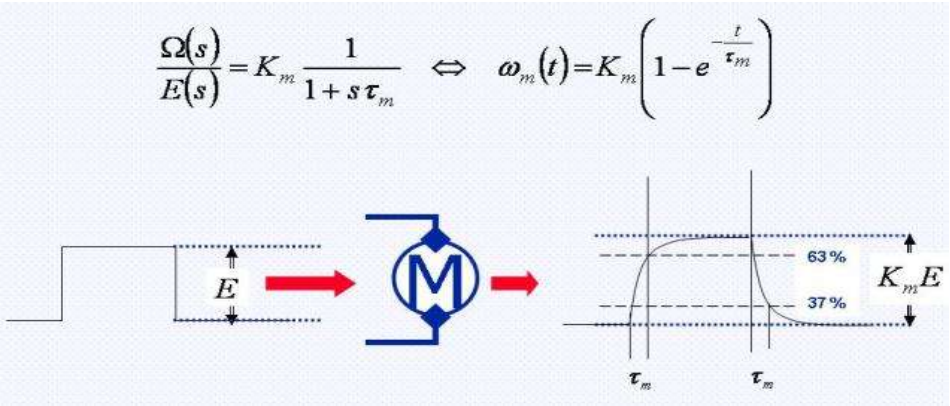


Fig. P1.12. medición de la constante de tiempo del motor.

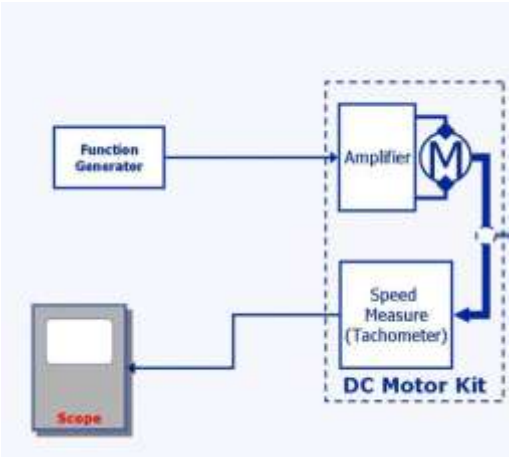


Fig. P1.13. medición de la constante de tiempo del motor.

El Generador de funciones genera pulsos, como se muestra en la Figura P1.14.

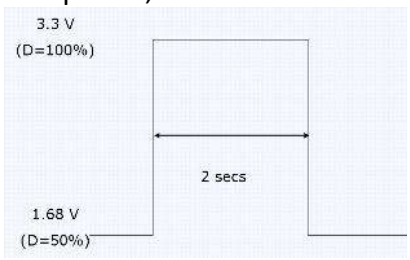


Fig. P1.14. parámetros de pulsos.

Como el propósito de esta práctica es familiarizarse con Simulink, se creó un modelo que representa el circuito de la figura P1.13, se crea una GUI para poder mostrar las graficas de las señales (la del generador de funciones y la del tacómetro).

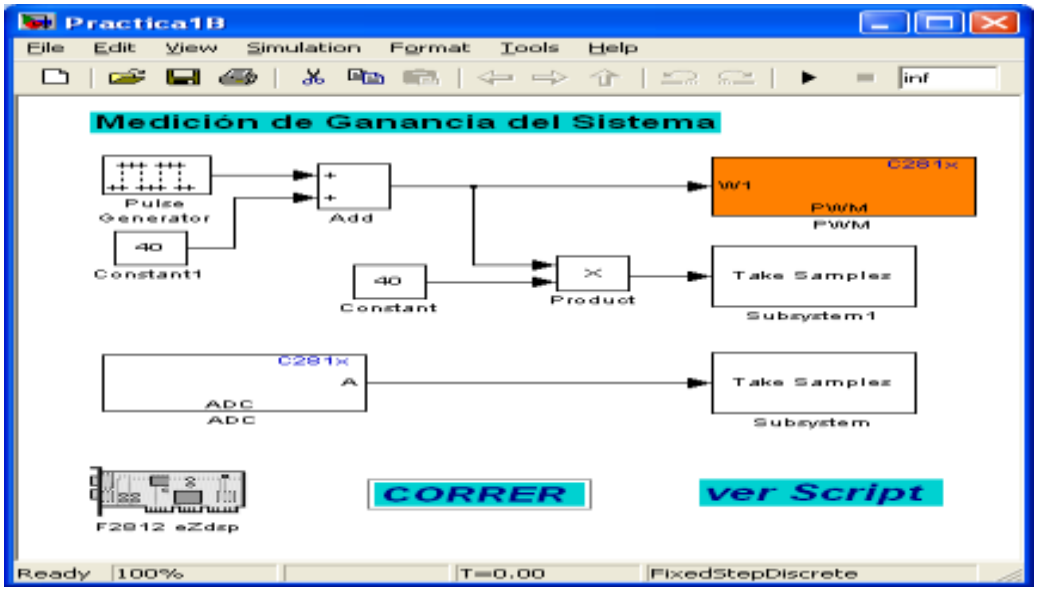


Fig. P1.15. Modelo para la medición de la constante de tiempo del motor.

Procedimiento.

1. Dentro de la carpeta Practica1 se encuentran 5 archivos necesarios para esta practica, los archivos "*Practica1B.mdl"* y "*correrpractica1B.m*" deben colocarse en la siguiente carpeta:

   C:\Documents and Settings\Alumno EIE\Mis documentos\MATLAB

   Si aparece el mensaje de sobrescribir archivos, dar clic en sí.

   Los archivos "*speedcontrolDC.m*", "*speedcontrolDC.fig*" y "*speedcontrolDCLoop.m*" deben copiarse en la siguiente carpeta:

C:\Archivos de programa\MATLAB\R2008b\toolbox\rtw\targets\tic2000\tic2000demos

   Si aparece el mensaje de sobrescribir archivos, dar clic en sí.

2. Siga los pasos 1, 2 y 3 de la práctica 1A. Con la diferencia que en el paso 3 se debe seleccionar *"Practica1B.mdl".*

3. Dar clic en el botón "*Correr*"; en la ventana de comandos de Matlab se podrá visualizar el proceso de construcción y carga, al finalizar este proceso se deberá observar la interfaz grafica, luego de esto se debe encender la fuente PS150E.

4. Mida los tiempos de subida (τ_{rise}) y bajada (τ_{fall}) de la respuesta del motor, recuerde que estos tiempos se deben medir entre el 37% y 63% de la señal, como se indica en la figura P1.12.

   El motor DC se modela como un sistema lineal de primer orden donde $\tau_{rise} = \tau_{fall,}$ sin embargo, el motor real no es lineal (debido a la fricción, por ejemplo), por lo tanto el valor a tomar para la constante de tiempo del motor τ_m, será el valor promedio de los dos tiempos medidos:
   τ_{rise} =
   τ_{fall} =

5. Sustituir los valores encontrados para Km y τ_m dentro de la ecuación:

$$\frac{\Omega(s)}{E(s)} = K_m \frac{1}{1 + s\tau_m}$$

6. Anotar conclusiones y observaciones.

3.6 PRACTICA 2. CONTROL PID DE VELOCIDAD.

OBJETIVOS:

- Que el alumno tenga contacto con un servomecanismo de velocidad real, mediante su identificación y control haciendo uso de programas de computadora.

- Que el alumno conozca, mediante la realización de pruebas al controlador, los problemas de implantación y limitaciones físicas que surgen al trabajar con equipos reales (ruidos, zonas muertas, saturaciones, …).

Equipo Requerido:

1 Modulo Controlador Digital con tarjeta eZdsp F2812.
1 Suministro de potencia PS150E.
1 Unidad Servo amplificador SA150D.
1 Motor DC con taco generador MT150F.
1 Unidad de freno magnético.

DISCUSION PRELIMINAR.

Un PID (Proporcional Integral Derivativo) es un mecanismo de control por realimentación que se utiliza en sistemas de control industriales. Un controlador PID corrige el error entre un valor medido y el valor que se quiere obtener calculándolo y luego sacando una acción correctora que se puede ajustar acorde al proceso. El algoritmo de cálculo del control PID se da en tres parámetros distintos: el proporcional, el integral, y el derivativo. El valor Proporcional determina la reacción del error actual. El Integral genera una corrección proporcional a la integral del error, esto asegura que aplicando un esfuerzo de control suficiente, el error de seguimiento se reduce a cero. El Derivativo determina la reacción del tiempo en el que el error se produce. La suma de estas tres acciones es usada para ajustar al proceso por medio de un elemento de control. Ajustando estas tres variables en el algoritmo de control del PID, el controlador puede proveer un control diseñado para lo que requiera el proceso a realizar, en este caso un control de velocidad.

Significado de las constantes del PID.

Kp constante de proporcionalidad: se puede ajustar como el valor de la ganancia del controlador o el porcentaje de banda proporcional.

Ki constante de integración: indica la velocidad con la que se repite la acción proporcional.

Kd constante de derivación: hace presente la respuesta de la acción proporcional duplicándola, sin esperar a que el error se duplique. El valor indicado por la constante de derivación es el lapso de tiempo durante el cual se manifestará la acción proporcional correspondiente a 2 veces el error y después desaparecerá.

Tanto la acción Integral como la acción Derivativa, afectan a la ganancia dinámica del proceso. La acción integral sirve para reducir el error estacionario, que existiría siempre si la constante Ki fuera nula.

Ajuste de parámetros del PID

El objetivo de los ajustes de los parámetros PID es lograr que el lazo de control corrija eficazmente y en el mínimo tiempo los efectos de las perturbaciones; se tiene que lograr la mínima integral de error. Si los parámetros del controlador PID (la ganancia del proporcional, integral y derivativo) se eligen incorrectamente, el proceso a controlar puede ser inestable, por ejemplo, que la salida de este varíe, con o sin oscilación, y está limitada solo por saturación o rotura mecánica.

Modificación de los valores de las constantes Kp, Ki y Kd para el controlador PID implementado.

Para modificar los valores de las constantes del PID y así poder observar el efecto que dicha modificación produce, se deben seguir los siguientes pasos.

- Abrir el archivo 'ControlVelocidad.mdl', se deben observar las ventanas mostradas en la figura P2.1.

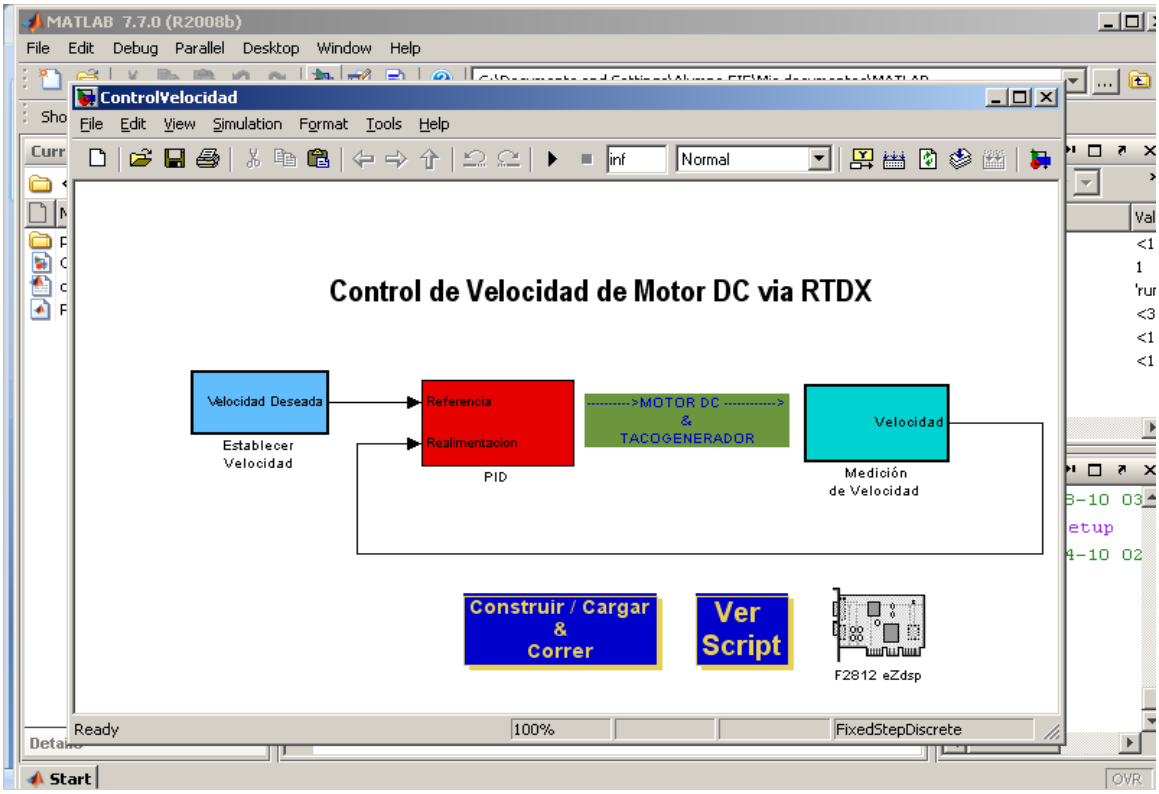


Fig. P2.1. Control de Velocidad de Motor DC

- Dar doble clic en el subsistema "*PID*", se deberá observar la figura P2.2.

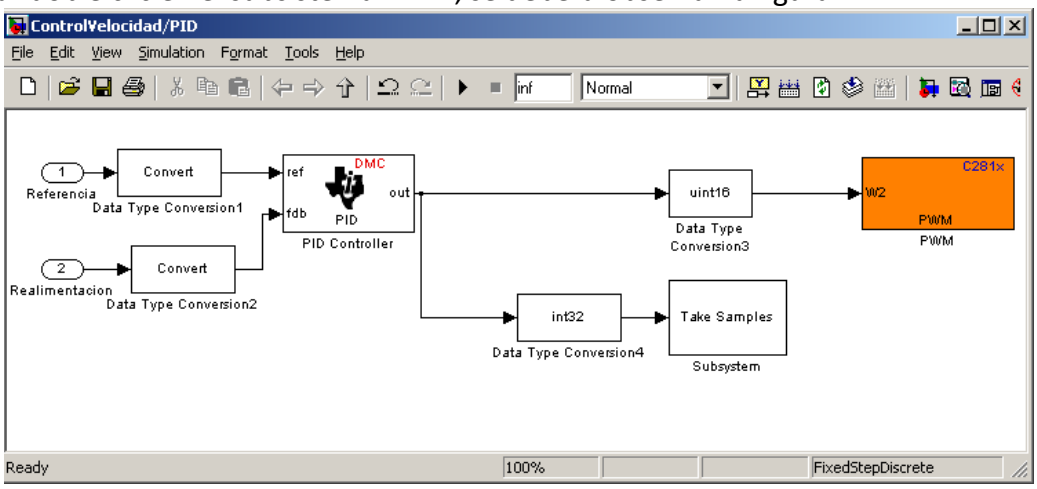


Fig.P2.2. Subsistema PID

- Dar doble clic en el bloque "*PID Controller*", se observara la ventana de configuración de parámetros del PID, esta se muestra en la figura P2.3.

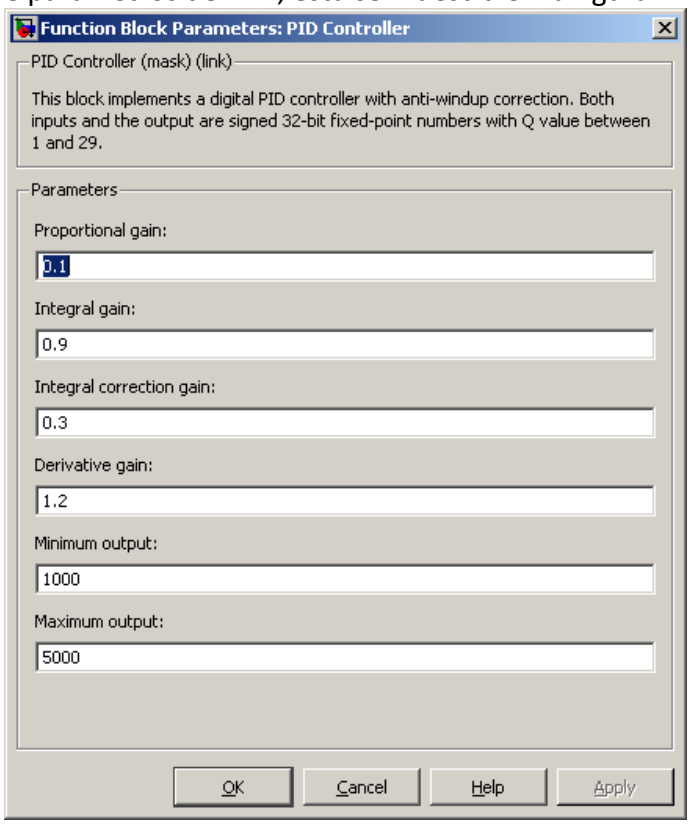


Fig. P2.3. Ventana de configuración de parámetros del bloque "PID Controller".

- Para modificar los valores de las constantes, basta con cambiar los valores de "*Ganancia Proporcional*", "*Ganancia Integral*" y "*Ganancia Derivativa*" que representan a Kp, Ki y Kd respectivamente, luego dar clic en Ok y guardar las modificaciones realizadas.

Procedimiento de la práctica:

1. Conecte el circuito que se muestra en la Figura P2.4.

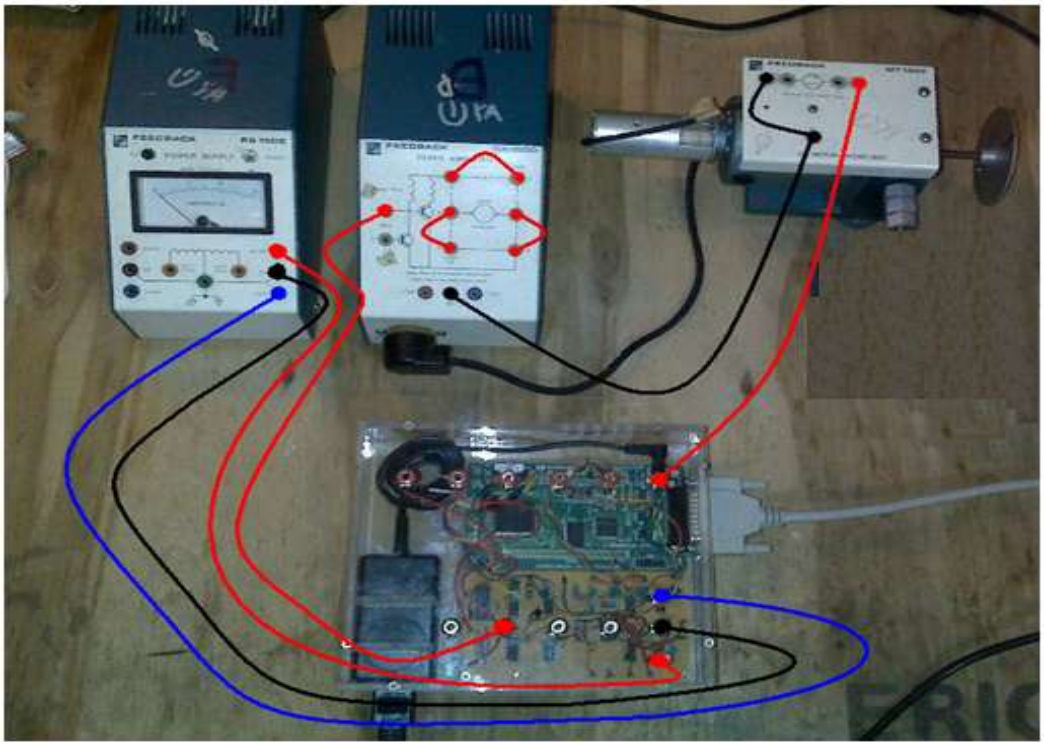


Fig. P2.4. Conexión del kid feedback al Modulo Digital eZdsp F2812.

2. Conectar el cable paralelo del CPU a la tarjeta DSP F2812 y energizar la tarjeta.

3. Siga las instrucciones de la practica 1 para conectar la DSP.

4. Dentro de la carpeta ControlVelocidad se encuentran 5 archivos, los archivos "*ControlVelocidad.mdl"* y "*correrconcontrolvelocidadmotordc.m*" deben colocarse en la siguiente carpeta:

   C:\Documents and Settings\Alumno EIE\Mis documentos\MATLAB

   Si aparece el mensaje de sobrescribir archivos, dar clic en sí.

   Los archivos "*speedcontrolDC.m*", "*speedcontrolDC.fig*" y "*speedcontrolDCLoop.m*" deben copiarse en la siguiente carpeta:

   C:\Archivos de programa\MATLAB\R2008b\toolbox\rtw\targets\tic2000\tic2000demos

   Si aparece el mensaje de sobrescribir archivos, dar clic en sí.

5. Dar clic en archivo ‘ControlVelocidad.mdl’, se deben observar las ventanas mostradas en la figura P2.5.

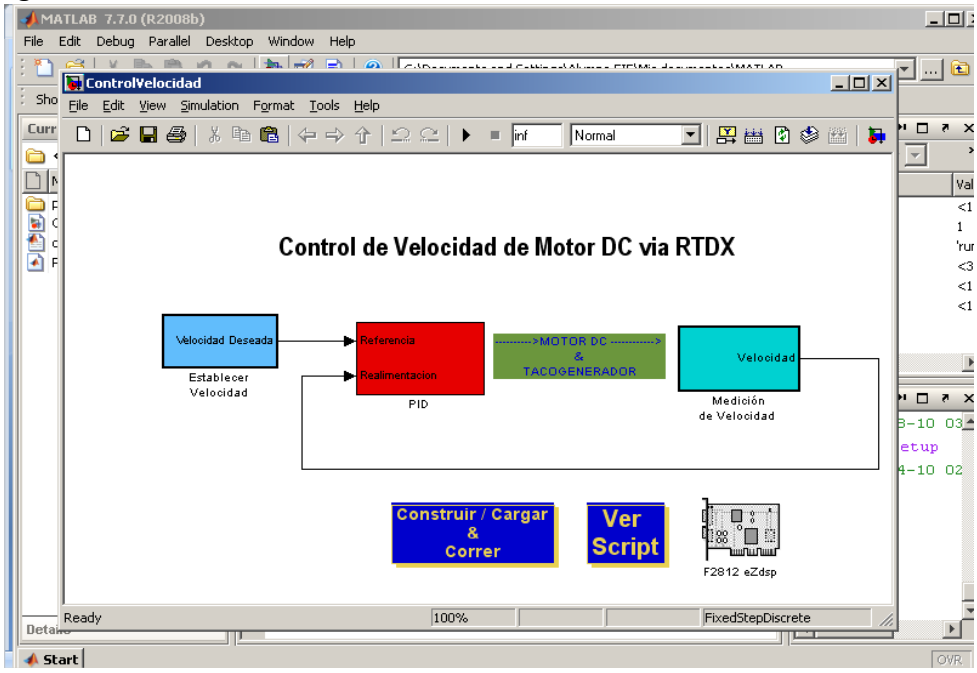


Fig. P2.5. Controlador de Velocidad en Simulink

6. Dar clic en el botón “*Construir, Cargar y Correr*”; en la ventana de comandos de Matlab se podrá visualizar el proceso de construcción y carga, al finalizar este proceso se deberá observar la interfaz grafica mostrada en la figura P2.6, luego de esto se debe encender la fuente PS150E.

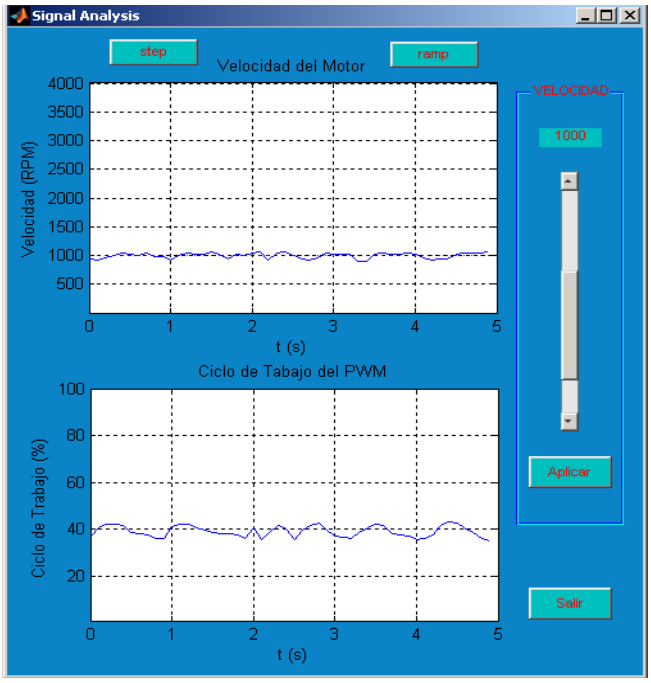


Fig. P2.6. GUI del Controlador de Velocidad.

7. Llenar la tabla P2.1 con los valores de velocidad medidos en la grafica, tomando en cuenta la velocidad de referencia y posición de freno establecidos.

	500 [RPM]		***1000 [RPM]***		***1500 [RPM]***		***2000 [RPM]***		***2500 [RPM]***		***3000 [RPM]***		***3500 [RPM]***	
Posición de freno	*RPM*	*Ciclo de Trabajo (%)*	*RPM*	*Ciclo de Trabajo (%)*	*RPM*	*Ciclo de Trabajo (%)*	*RPM*	*Ciclo de Trabajo (%)*	*RPM*	*Ciclo de Trabajo (%)*	*RPM*	*Ciclo de Trabajo (%)*	*RPM*	*Ciclo de Trabajo (%)*
0														
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														

Tabla P2.1. Velocidad Vrs. Carga.

Hacer una grafica con curvas de velocidad del motor Vrs. Posición del freno para cada valor de velocidad de referencia que se muestra en la tabla.

8. Colocar a cero la barra deslizante y presione aplicar, luego presione el botón rampa, capturar una imagen de la grafica y en base a esta calcular la pendiente y el error en estado estable para esta función.

9. Llevar la barra deslizante a la posición cero presionar el botón aplicar, luego presione el botón escalón, capturar una imagen de la grafica y en base a esta determinar el tiempo de asentamiento, tiempo de levantamiento, sobrepaso máximo y numero de oscilaciones. Llenar la tabla P2.2.

KP	**KI**	**KD**	**Te**	**Td**	**Velocidad max. (RPM)**	**Nº oscilaciones**	**Error EE (RPM)**
0.1	0.9	1.2					
1	10	0					
5	0.1	0.008					
10	1	0.22					
11	0.7	5					

Tabla P2.2. Determinación de los parámetros de rendimiento de PID.

10. Elaborar un reporte que contenga los resultados obtenidos y sus conclusiones.

3.7 PRACTICA 3. CONTROL PID DE POSICIÓN.

OBJETIVOS:

- Conocer las características más importantes en un mecanismo de posición, controlado en tiempo discreto por medio de una DSP.
- Determinar las características más importantes que diferencian los mecanismos de control de posición con los mecanismos de control de velocidad.

Equipo Requerido:

1 Modulo Controlador Digital con tarjeta eZdsp F2812.
1 Suministro de potencia PS150E.
1 Unidad Servo amplificador SA150D.
1 Motor DC con taco generador MT150F.
1 Unidad Potenciómetro de Salida OP150K.

Procedimiento de la práctica:

1. Conecte el circuito que se muestra en la Figura P3.1.

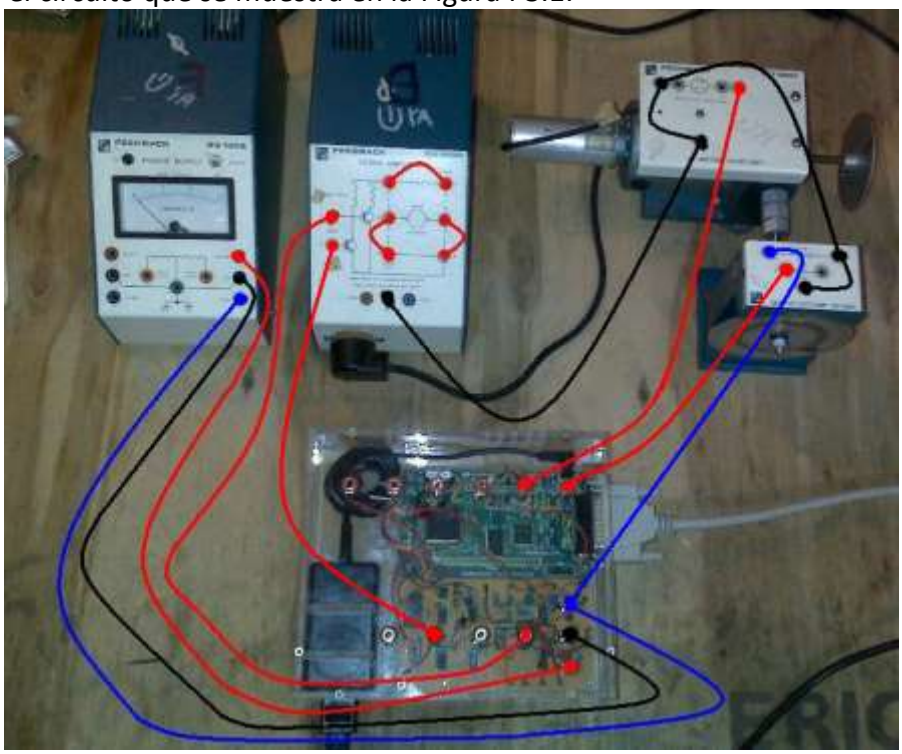


Fig. P3.1. Conexión del kid feedback al Modulo Digital eZdsp F2812.

2. Conectar el cable paralelo del CPU a la tarjeta DSP F2812 y energizar la tarjeta.

3. Siga las instrucciones de la practica 1 para conectar la DSP.

4. Dentro de la carpeta ControlPosicion se encuentran 5 archivos, los archivos "*ControlPosicion.mdl"* y "*correrconcontrolposicionmotordc.m*" deben colocarse en la siguiente carpeta:

   C:\Documents and Settings\Alumno EIE\Mis documentos\MATLAB

   Si aparece el mensaje de sobrescribir archivos, dar clic en sí.

   Los archivos "*speedcontrolDC.m*", "*speedcontrolDC.fig*" y "*speedcontrolDCLoop.m*" deben copiarse en la siguiente carpeta:

   C:\Archivos de programa\MATLAB\R2008b\toolbox\rtw\targets\tic2000\tic2000demos

   Si aparece el mensaje de sobrescribir archivos, dar clic en sí.

5. Dar clic en archivo 'ControlPosicion.mdl', se deben observar las ventanas mostradas en la figura P3.2.

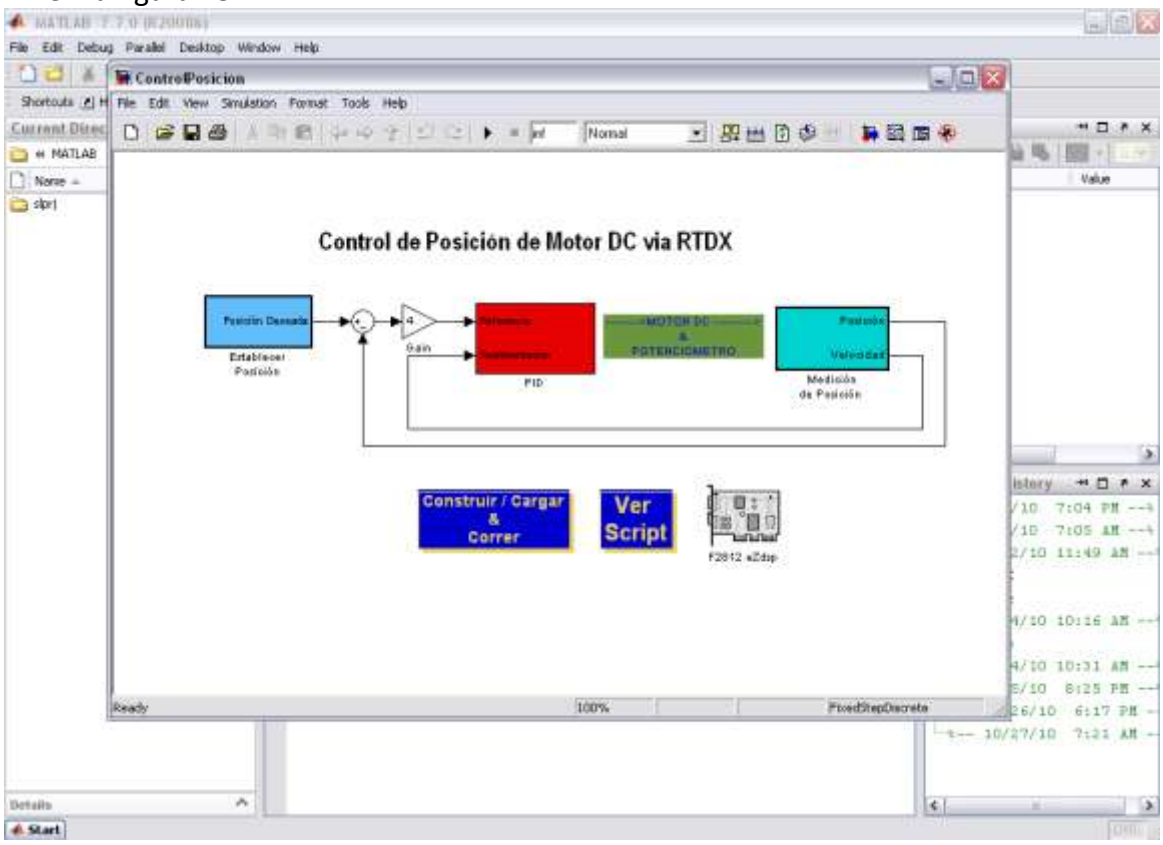


Fig.P3.2. Controlador de Posición en Simulink.

6. Dar clic en el botón "*Construir, Cargar y Correr*"; en la ventana de comandos de Matlab se podrá visualizar el proceso de construcción y carga, al finalizar este proceso se deberá observar la interfaz grafica mostrada en la figura P3.3, luego de esto se debe encender la fuente PS150E.

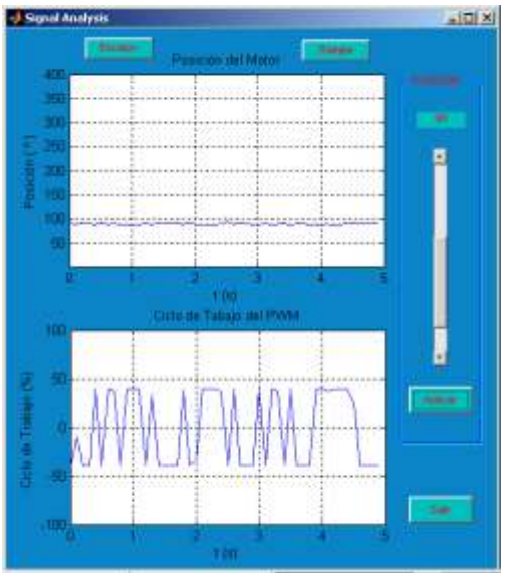


Fig. P3.3. GUI del Controlador de Posición.

7. A continuación llene la siguiente tabla para todos los valores de angulos dados, encuentre el error en estado estable de este sistema de posición.

Posición Deseada (º)	Posición Medida(º)	Error (º)
15		
30		
45		
60		
75		
90		
105		
120		
135		
150		
165		
180		
195		
210		
225		
240		
255		
270		
285		
300		
315		
330		
345		
360		

TablaP3. 1: Medición de ángulos.

De sus conclusiones sobre lo observado.

8. Colocar a cero la barra deslizante y presione aplicar, luego presione el botón rampa, capturar una imagen de la grafica y en base a esta calcular la pendiente y el error en estado estable para esta función.

9. Llevar la barra deslizante a la posición cero presionar el botón aplicar, luego presione el botón escalón, capturar una imagen de la grafica y en base a esta determinar el tiempo de asentamiento (Te), tiempo de levantamiento (Td), sobrepaso máximo (Velocidad Max.) y numero de oscilaciones. Llenar la tabla P3.2.

KP	**KI**	**KD**	**Tr**	**Ts**	**Sobrepaso max. (°)**	**Nº oscilaciones**	**Error EE (RPM)**
0.1	0.9	1.2					
1	10	0					
5	0.1	0.008					
10	1	0.22					
11	0.7	5					

Tabla P3.2. Determinación de los parámetros de rendimiento de PID.

10. Elaborar un reporte que contenga los resultados obtenidos y sus conclusiones.

CONCLUSIONES CAPITULO 3.

La estación de trabajo de prototípado rápido, proporciona grandes ventajas con respecto a los sistemas de bloques y cableados convencional que aun se utilizan en los laboratorios de control. Esta herramienta viene a reforzar y facilitar la compresión de sistemas de control prácticos, ya que toda la reestructuración del sistema se hace en base a diagramas de bloques en Simulink, lo cual tiene mayor similitud con las técnicas que se estudian en el curso de sistemas de control; a demás, se introduce al estudiante al diseño de sistemas de control en tiempo discreto que es algo que no se podía lograr en los laboratorios actuales.

Con los laboratorios propuestos en este documento se puede ver claramente los efectos de controladores PID en los sistemas, gracias a la interfaz grafica que se creó en Matlab. La interfaz grafica proporciona una señal de referencia, y muestra en la misma grafica la respuesta del sistema, esto ayudará grandemente a que el alumno entienda claramente conceptos básicos como sobre paso máximo, tiempo de asentamiento y levantamiento, etc.

A pesar de que la estación de trabajo de prototípado rápido está elaborada en base a un motor DC, el controlador digital creado con la eZdsp F2812 puede soportar un gran número de plantas como lo son: circuitos RC simples, ciclo convertidores u otra planta desarrollada por un estudiante. Se espera que en un futuro, más plantas puedan ser desarrolladas para experimentación en el laboratorio.

CAPITULO 4

MANUAL DE UTILIZACION DE WINLOG SCADA/HMI

CAPITULO 4
SISTEMA SCADA: MANUAL DE UTILIZACIÓN DE WINLOG LITE V.2.86

FIG. 4.1. WINLOG LITE

Los sistemas SCADA (del acrónimo **S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition) son interfaces HMI orientadas a la aplicación en computadoras. Proporcionan la comunicación con los dispositivos de campo (PLC's) y controlan el proceso global de forma automática desde la pantalla del ordenador. También provee toda la información que se genera en el proceso productivo a diversos usuarios, tanto del mismo nivel como de otros usuarios supervisores dentro de la empresa (supervisión, control calidad, control de producción, almacenamiento de datos, etc.).

En la industria las nuevas tecnologías están haciendo que las aplicaciones de automatización industrial cambien las condiciones del mercado y faciliten la aplicación de técnicas de automatización cada vez a más sectores. Los sistemas SCADA se están volviendo cada vez más esenciales en el control de sistemas que se vuelven más complejos.

Una opción muy viable para la realización de pequeños sistemas SCADA ó el aprendizaje básico en el diseño de estos, es la utilización del programa Winlog de la empresa Italiana Sielco Sistemi; esta empresa ha puesto a la disposición del público en general la versión gratuita Winlog Lite que cuenta con una gran cantidad de características de los programas de diseño de sistemas SCADA profesionales, siendo uno de los programas gratuitos para el diseño de sistemas SCADA mas completo del mercado.

En el presente capitulo se trata de dar una pequeña introducción al diseño de sistemas SCADA, para el cual se debe contar con el programa antes descrito, Winlog Lite. Se empieza dando una pequeña introducción al programa y posteriormente se redacta un manual detallado para su uso, que va desde la realización de las interfaces hasta la edición del código que se ejecutará en tiempo real; dando una descripción detallada en cada uno de los pasos.

Como último tema se da un ejemplo de un pequeño sistema SCADA, el cual consiste en la automatización y control de una mini central hidroeléctrica de la cual se resaltan las partes más importantes del proceso sin ahondar tanto en los detalles de la central para no entorpecer el verdadero objetivo del ejemplo. Se da una guía desde la fase de configuración y creación de los protocolos de comunicación hasta la creación de las plantillas, dando una descripción detallada en cada paso del proceso de diseño.

4.1 DESCRIPCIÓN GENERAL

Las principales características del software Winlog Lite son:

- ✓ Aplicaciones HMI (Human Machine Interface) multilenguaje.
- ✓ Soporte TCP/IP cliente-servidor.
- ✓ Arquitectura Intranet/Internet.
- ✓ Lenguaje de sistema SCADA integrado.
- ✓ Extensa librería de drivers.
- ✓ Símbolos y objetos gráficos complejos.
- ✓ Reportes accesibles desde Excel, Access, etc.

El fabricante describe al software como flexible, sencillo y económico, Winlog es un paquete de software SCADA/HMI para la supervisión de plantas industriales o civiles. Como un ambiente de desarrollo integrado provee de diferentes herramientas (Gate Builder, Template Builder, Code Builder) para el fácil e intuitivo desarrollo de aplicaciones. Una extensa librería de drivers y una interfaz OPC[1] cliente/servidor permite la comunicación con la mayoría de componentes electrónicos, tales como PLCs, controladores, manejadores de motores, módulos entradas/salidas, etc. Winlog además hace posible la instalación de una arquitectura cliente/servidor[2] con protocolo TCP/IP en una red Intranet/Internet o crear aplicaciones web accesibles desde navegadores estándar.

LIBRERÍA GRAFICA

Las herramientas de desarrollo de Winlog incluyen a *Symbol Factory 2.0,* la cual es una librería de objetos gráficos de automatización, con cerca de 4000 objetos de fabricación e industria tales como válvulas, bombas, motores, tanques, PLC, tuberías, símbolos ISA, etc.; un editor integrado permite cambiarles tamaño y color, orientación , etc. Además se incluyen potenciómetros lineales o circulares, indicadores lineales o punteros, termómetros, interruptores y selectores.

EJECUCIÓN DE APLICACIONES

Las aplicaciones multilenguaje pueden ser creadas en dos o más lenguajes, y con un simple comando pueden intercambiar el lenguaje de las aplicaciones. Reportes y datos históricos pueden ser guardados en formatos estándar tales como DBF (Formato de Base de Datos) o CSV (Valores Separados por Coma), los cuales pueden ser abiertos por otras aplicaciones de Windows (Excel, Access, etc.).

[1] OPC es un estándar de comunicación en el campo del control y supervisión de procesos.

[2] Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta.

En cuanto a los gráficos de tendencias históricas o en línea, estas pueden ser mostradas individualmente o en grupos; son posibles hasta un total de 10 tendencias al mismo tiempo, las cuales pueden referirse a diferentes tipos de etiquetas, con su propio color o escala. El apuntador se puede utilizar para acercar o alejar la vista, otras opciones de vista están disponibles.

Los eventos y alarmas pueden ser activados como una función de los valores tomados o por etiquetas asociadas, las alarmas son señaladas inmediatamente en un área de la pantalla reservada especialmente, y la confirmación es controlada también por el operador. La información de estado (eventos activos) y el record (datos, tiempo, evento) pueden ser accedidos de acuerdo a varias categorías libres, por ejemplo, prioridad o locación.

Los reportes con información resumida acerca de los datos de producción, algún proceso o alarmas. Pueden ser generados en un formato pre establecido y puede ser mostrado, impreso o enviado a cualquier dispositivo periférico. Winlog permite realizar reportes requeridos por el operador, o generarlos automáticamente en un ciclo básico (especificado por un intervalo de tiempo, un día particular de la semana, etc).

4.1.1 HERRAMIENTAS DE DESARROLLO

Project Manager (Manejador de proyectos) es un ambiente de desarrollo integrado que provee de diferentes herramientas, para la creación de cualquier aplicación con Winlog, entre las que se encuentran:

- ✓ Gate Builder: Es la herramienta para configurar las base de datos de compuertas (etiquetas), diferentes tipos de compuertas (numéricas, digitales, cadenas, combinados, eventos, alarmas) pueden ser definidas y asignarles las propiedades correctas (nombre, descripción, dirección, unidad de medida, factor de escala, etc.). Las compuertas pueden ser leídas desde dispositivos externos (controladores, PLCs, indicadores, módulos de adquisición de datos, etc.) o generados por el mismo software. El método de muestreo puede ser configurado para cada compuerta(o un juego de compuertas) para obtener una frecuencia de actualización de los datos sin necesidad de usar demasiado espacio en memoria (lectura de bloque, lectura en sistema, guardar solo en caso de una variación significativa, etc.).

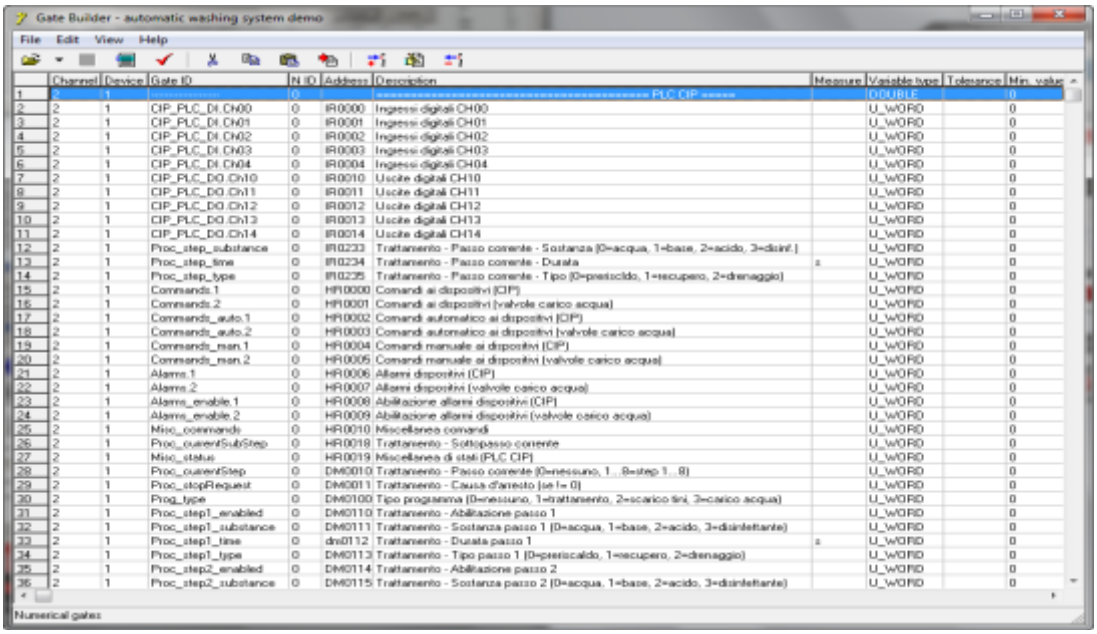


FIG. 4.2. Pantalla principal Gate Builder

- Template Builder: Esta es la herramienta para una fácil creación de plantillas y paginas de imágenes. Todo lo que se debe de hacer es arrastrar y arreglar en la pantalla los objetos (mapas de bits, textos, barras de estado, leds e iconos de control) y definir sus propiedades (dimensiones, estilos, etiquetas asociadas, etc.). Cada objeto de la plantilla puede ser asignado a un tipo de control que permite acceso solo a algunos operadores a través de contraseña.

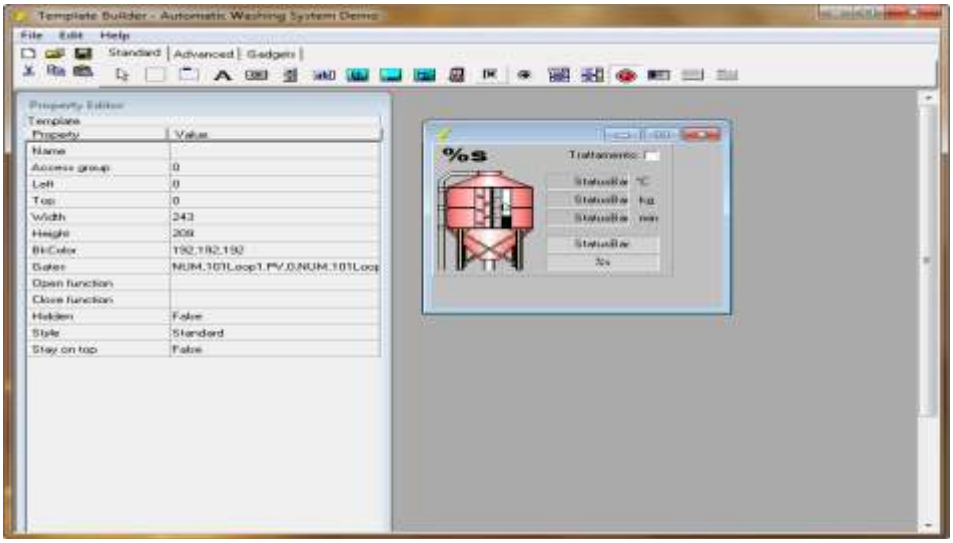


FIG. 4.3. Pantalla principal Template Builder

- Code Builder: Es el ambiente integrado de desarrollo que da la posibilidad de enriquecer y personalizar la aplicación; un lenguaje de programación parecido a c permite al programador interactuar con todos los elementos de Winlog (etiquetas,

plantillas, recetas, reportes, etc.), para definir lazos o condiciones “if-then-else”, para crear funciones (Macros) que puedan ser ejecutadas automáticamente o bajo el control de el operador.

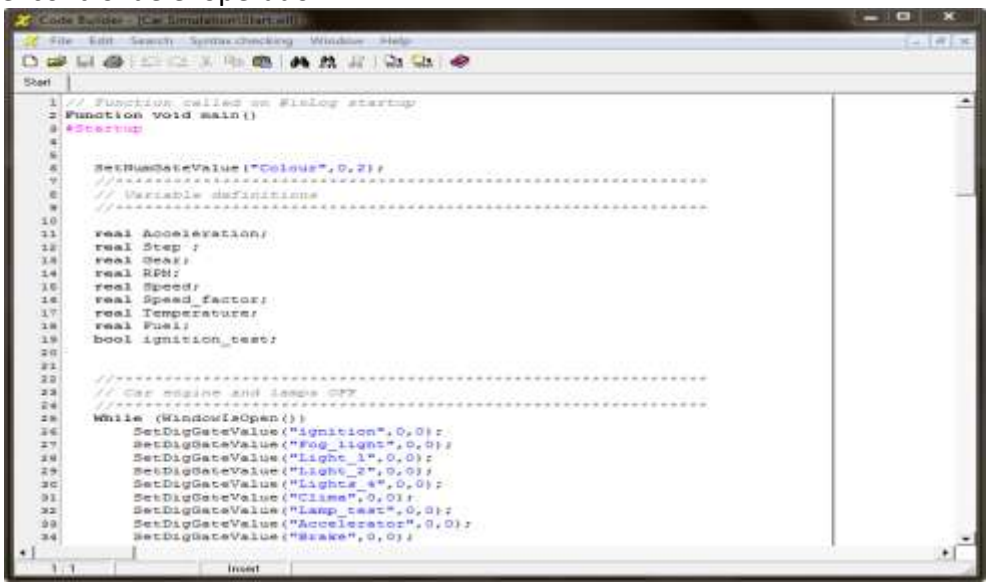


FIG. 4.4. Pantalla principal Code Builder

4.2 MANEJADOR DE PROYECTO (PROYECT MANAGER)

4.2.1 GENERALIDADES

A través del manejador de proyectos es posible crear, borrar, copiar y configurar los proyectos en una forma visual. La pantalla principal se muestra en la figura 4.5, donde se puede observar al lado izquierdo esta el menú donde se muestran todos los proyectos presentes con sus respectivas carpetas de configuración y edición, mientras en la parte derecha se muestran los componentes de la carpeta seleccionada.

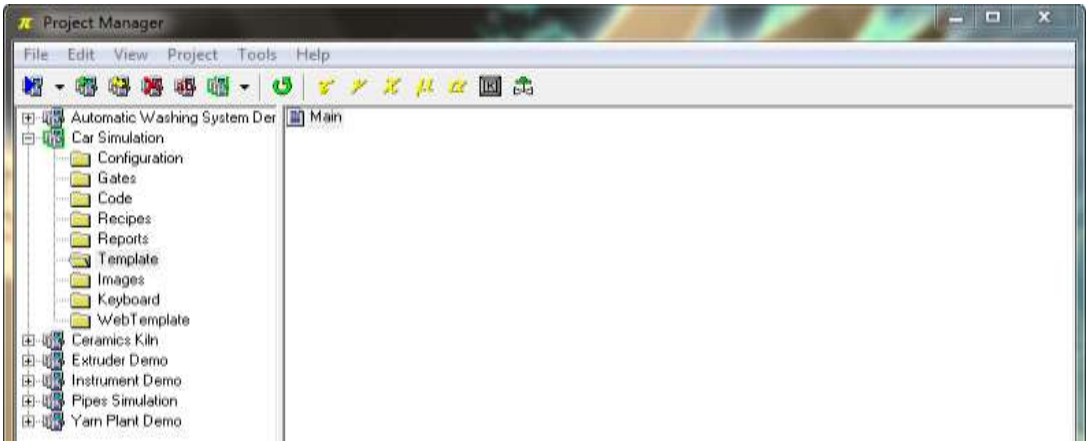


FIG. 4.5. Pantalla principal Project Manager

Las operaciones que se pueden realizar con un proyecto se muestran en la barra de herramientas del manejador de proyectos, tal como se muestra en la figura 4.6, de izquierda a derecha son:

FIG. 4.6. Barra de proyecto.

- Ejecutar Proyecto: Al hacer clic se despliega la lista de proyectos disponibles, al seleccionar uno se ejecuta y se llama automáticamente a *Run Time*.
- Crear Proyecto: Se crea un proyecto en blanco, al cual se le debe asignar un nombre y se agrega a la barra izquierda de proyectos disponibles.
- Copiar Proyecto: Para copiar un proyecto se debe de seleccionar y hacer clic en la barra, se debe de asignar un nombre al proyecto copiado.
- Eliminar Proyecto: Para eliminar un proyecto se debe de seleccionar y hacer clic en el icono de la barra de herramientas.
- Renombrar Proyecto: Seleccionar un proyecto y hacer clic en el icono de la barra de herramientas.
- Proyecto por defecto: Este selecciona el proyecto que se ejecuta cuando se abre el *Run Time* directamente sin el manejador de proyectos. El proyecto seleccionado se distingue por un icono sombreado de verde en la barra izquierda de la pantalla.

Todas las operaciones anteriores también se encuentran disponibles a través del menú *Project* de la barra de menús. Además, se dispone de otras operaciones tales como construir una copia con seguridad; la cual establece una contraseña para poder modificar el proyecto, importar el proyecto y exportarlo.

4.2.2 CONFIGURACIÓN DE PROYECTO

En el menú de la izquierda para cada proyecto se puede acceder a la carpeta de configuración la cual incluye: opciones, canales, dispositivos, plantillas y grupos de accesos. Para configurar y modificar estos elementos solo es necesario darle doble clic. La figura 4.7 muestra la carpeta de configuración y sus elementos.

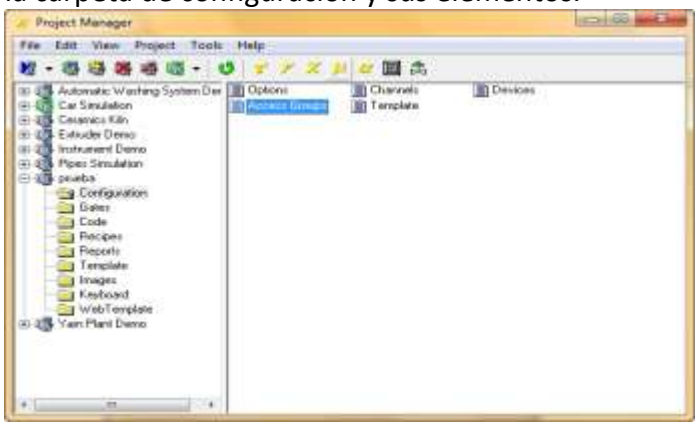


FIG. 4.7. Carpeta de configuración de proyecto

4.2.2.1 OPCIONES DE CONFIGURACIÓN

La ventana de dialogo de opciones permite configurar aspectos generales del proyecto, la ventana está dividida en 6 pestañas, dentro de las cuales se tienen distintas opciones, estas se pueden observar en la figura 4.8.

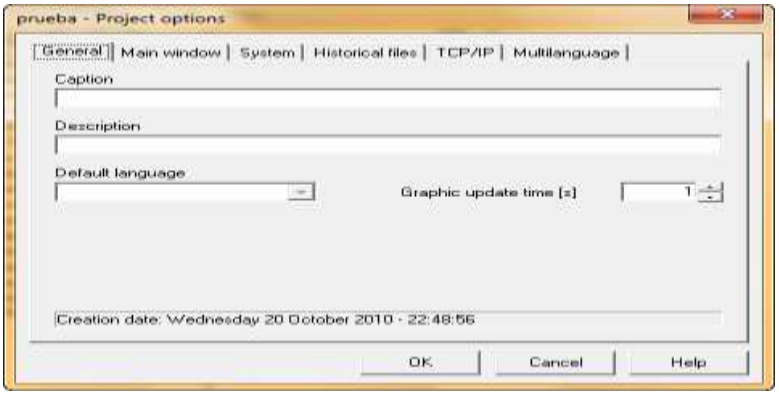


FIG. 4.8. Ventana de dialogo Opciones.

En la pestaña *general* podemos encontrar las siguientes opciones, tal como se puede observar en la figura anterior.

- Caption: Titulo que el proyecto mostrara durante la fase de ejecución.
- Description: Simple descripción del proyecto.
- Default Languaje: Permite seleccionar el idioma por defecto para la fase de ejecución.
- Grafic update time: Tiempo de actualización del flujo de gráficos.

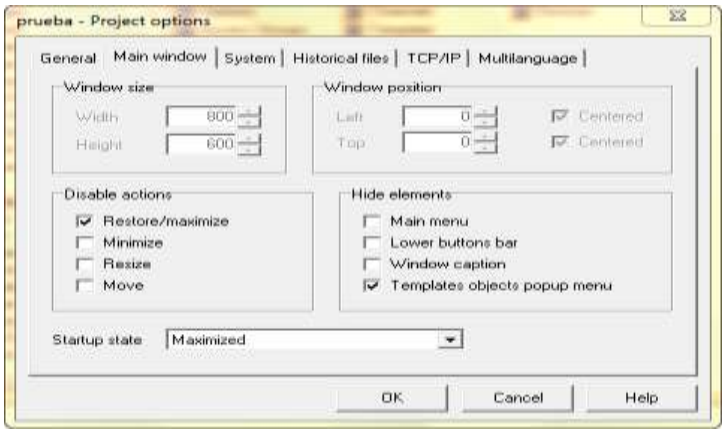


FIG. 4.9. Pestaña opciones de Ventana Principal

En la pestaña *Main window* se puede configurar la apariencia de la ventana principal en la fase de ejecución, por lo cual se pueden encontrar las siguientes opciones, tal como se puede observar en la figura 4.9.

- Window size: Permite configurar el tamaño de la ventana en modo normal(es decir, cuando no está maximizada o minimizada), estableciendo valores de ancho y alto en pixeles.
- Window Position: permite configurar la posición de la ventana en modo normal, esta se establece a través de coordenadas XY tomando como referencia la esquina superior izquierda.
- Disable Actions: Estas casillas de verificación permite deshabilitar acciones de usuario, las cuales son:
    - Restore/Maximize: no permite restaurar la ventana cuando es maximizado o niega la opción de maximizar la ventana cuando se encuentra en estado normal.
    - Minimize: No permite minimizar la ventana.
    - Resize: No permite modificar el tamaño de la ventana cuando se encuentra en estado normal.
    - Move: No permite mover la ventana cuando se encuentra en estado normal.
- Hide Elements: Estas casillas de verificación permiten ocultar elementos de la ventana principal. Entre las que se encuentran:
    - Main Menu: Oculta el menú principal de *Runtime.*
    - Lower buttons bar: Oculta los botones inferiores de la barra de *Runtime.*
    - Window caption: Oculta la barra de titulo de la ventana de *Runtime.*
    - Template object popup menú: Deshabilita el menú de los objetos de la plantilla.
- Startup state: Estado inicial de la ventana en la fase de ejecución.

En la pestaña de sistema se pueden encontrar ciertas opciones, las cuales se observan en la figura 4.10.

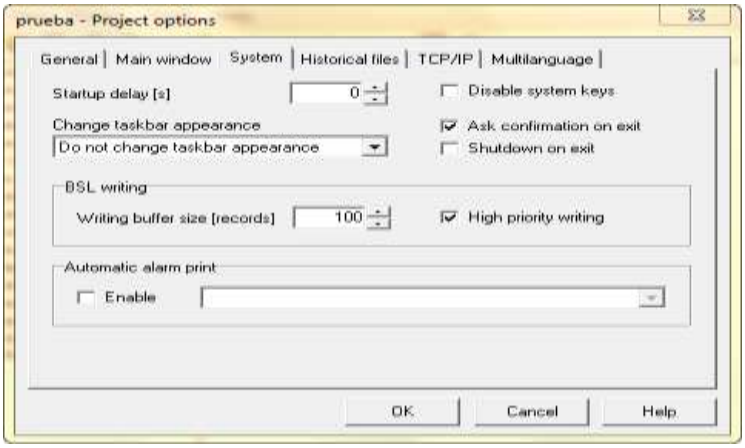


FIG. 4.10. Pestaña opciones de sistema

- Startup Delay: Especifica la cantidad de segundos a esperar para establecer la comunicación después del inicio de *Runtime*.
- Change Taskbar appearance: Permite modificar el comportamiento de la barra de tareas durante la fase de ejecución. Permite ocultarla para ganar espacio o

establecerla para no ocultar otras ventanas, esta última opción permite bloquear que los usuarios intercambien de ventana mientras se está ejecutando el proyecto.

- Disable system keys: Deshabilita las combinaciones de tecla Alt+TAB, Alt+Esc y Ctrl+Alt+Supr.
- Ask confirmation on exit: Solicita confirmación al cerrar la ventana de ejecución.
- Shutdown on exit: Permite apagar el sistema junto con el cierre de la ventana de ejecución.
- Writing buffer size: Todas las escrituras a dispositivos son almacenadas en esta cola de datos. Cada canal posee su propia cola de datos.
- High priority writing: Si está habilitada, toda escritura a dispositivos es realizada inmediatamente, de lo contrario se ejecuta luego de un ciclo de lectura.
- Automatic alarm print: Si está marcada, cada vez que una alarma sea verdadera esta se mandará al dispositivo de impresión seleccionado.

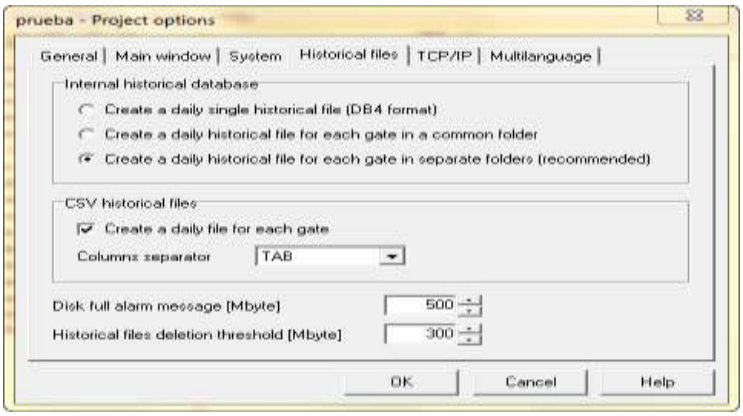


FIG. 4.11. Pestaña opciones de archivos.

La figura 4.11 muestra las opciones disponibles para la creación de archivos donde se guarda la información histórica de las compuertas. Estos son guardados en la subcarpeta DBTABLES del proyecto. La carpeta está organizada dependiendo de las opciones seleccionadas, entre las cuales se tienen:

- Internal Historical data: Permite seleccionar entre crear un único archivo diario en formato DB4 para cada tipo de compuerta, crear un archivo binario para cada compuerta que se ha definido como “record on file”, todas son guardadas en una misma carpeta y la última opción crea un archivo binario para cada compuerta que se ha definido como “record on file”, cada compuerta es guardada en una carpeta separada.

- CSV Historical Files: Este permite crear un archivo CSV diario para cada compuerta. Este tipo de archivo también es conocido como texto con formato y puede ser fácilmente manipulado por EXCEL. Este archivo está compuesto de dos campos, el campo de tiempo y el campo de valor. Un registro es agregado cada vez que una compuerta varía su valor. El separador de los campos puede ser elegido entre TAB, coma, etc.

- Disk full alarm message: Es el punto a partir del cual el programa informa al operador que el disco está quedando sin espacio libre.

- Historical files deletion threshold: Es el umbral a partir del cual el programa inicia a borrar archivos, los más antiguos hasta llegar al umbral establecido en la opción anterior.

Las opciones de TCP/IP como lo muestra la figura 4.12 son:

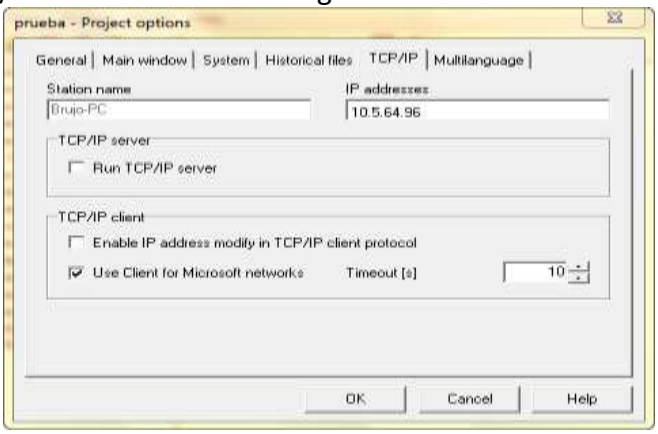


FIG. 4.12. Pestaña opciones TCP/IP

- Station name: Muestra el ID de red de la PC.

- IP addresses: Muestra la lista de direcciones IP de la PC.

- Run TCP/IP server: Permite a *Runtime* trabajar como un servidor TCP/IP para otra estación de trabajo que utilizan el mismo software de supervisión.

- Enable IP address modify in TCP/IP client protocol: Si es marcada permite en el caso de trabajar con el protocolo de cliente TCP/IP, modificar la dirección IP de el servidor durante la fase de ejecución del proyecto.

- Use client for Microsoft networks: Con los objetos de plantilla Chart, Histview y OperatorView, es posible acceder a archivos históricos ubicados en un servidor remoto. Esto es posible usando el protocolo TCP/IP.

4.2.2.2 CONFIGURACIÓN DE CANALES

Un canal es un puerto de comunicación utilizado para comunicarse con dispositivos externos. La ventana de configuración de canales permite modificar los parámetros de los canales de comunicación, tal como se puede observar en la figura 4.13.

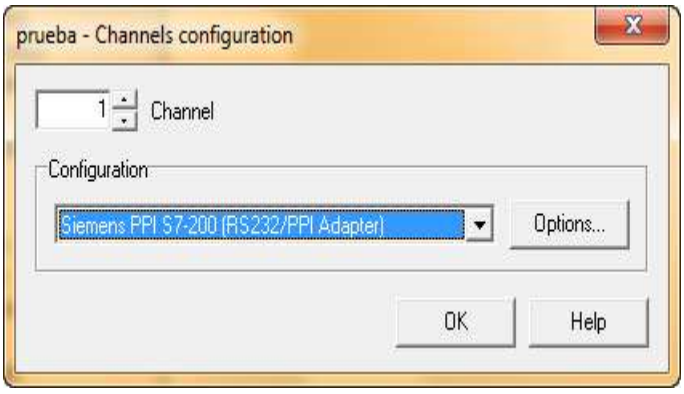


FIG. 4.13. Configuración de canales.

En esta se deben de seleccionar la cantidad de canales y el protocolo a utilizar para cada canal de comunicación. Cada protocolo tiene la facilidad de configurarse al presionar en el botón Options. Por ejemplo, en la figura 4.14 se muestran las opciones de configuración del protocolo PPI para la comunicación con un S7-200 a través de un cable multimaestro. Para mayor información sobre protocolos consultar la sección 4.6 Protocolos de comunicación.

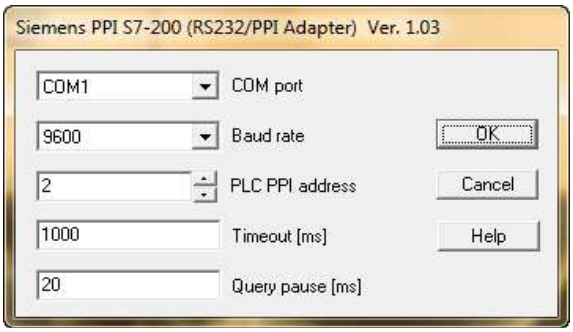


FIG. 4.14. Opciones de protocolo PPI

4.2.2.3 DISPOSITIVOS

Winlog permite monitorear las comunicaciones de todos los dispositivos conectados. Estos son identificados por el número de canal de comunicación y por una dirección. A través de la ventana de dispositivos, mostrado en la figura 4.15 es posible describir cada uno de los dispositivos para identificarlos más claramente.

FIG. 4.15. Descripción de Dispositivos

4.2.2.4 GRUPOS DE ACCESO

Winlog permite crear hasta 15 grupos de acceso (brindar privilegios según el usuario del proyecto), para lo cual es posible describir y darle nombre a cada uno de los grupos, para un manejo más fácil e intuitivo.

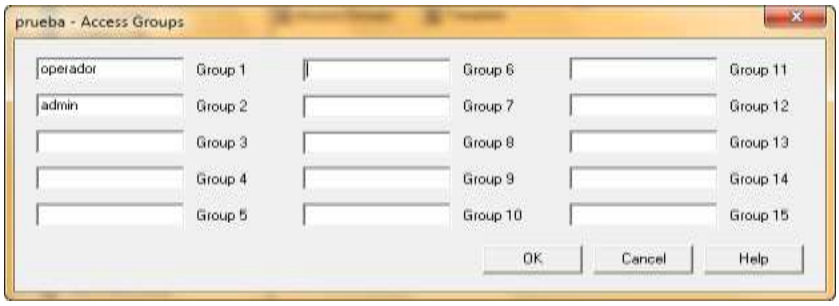


FIG. 4.16. Grupos de Acceso

4.2.2.5 CONFIGURACIÓN DE PLANTILLAS

La configuración de plantillas, tal como lo muestra la figura 4.17 posee las siguientes pestañas:

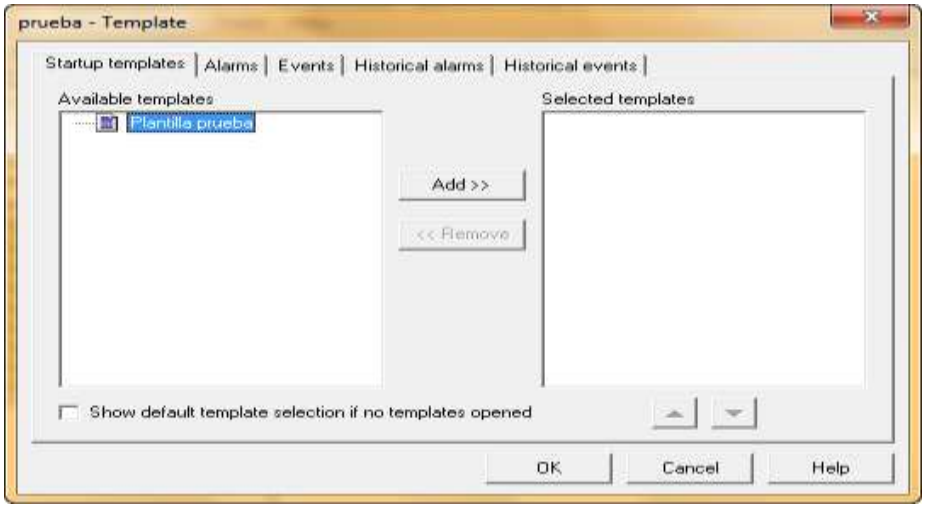


FIG. 4.17. Configuración de plantillas.

- Startup templates: Permite seleccionar la plantilla que será automáticamente cargada y abierta en la ejecución de *Runtime.* Para agregar solo se debe seleccionar y presionar el botón ADD con lo cual pasara a la lista de la derecha. La opción en la parte inferior de la pantalla, para el caso que no se halla seleccionado una plantilla, en la ejecución de Runtime mostrara una ventana para seleccionar las plantillas disponibles.

- Alarm, Event, Historical Alarm y Events: Estas pestañas se pueden observar en la figura 4.18, estas permiten seleccionar las columnas que se mostraran en las alarmas y eventos, durante la ejecución de *Runtime*. Las que poseen por nombre

estándar Class, pueden contener datos genéricos por lo cual es posible renombrarlos.

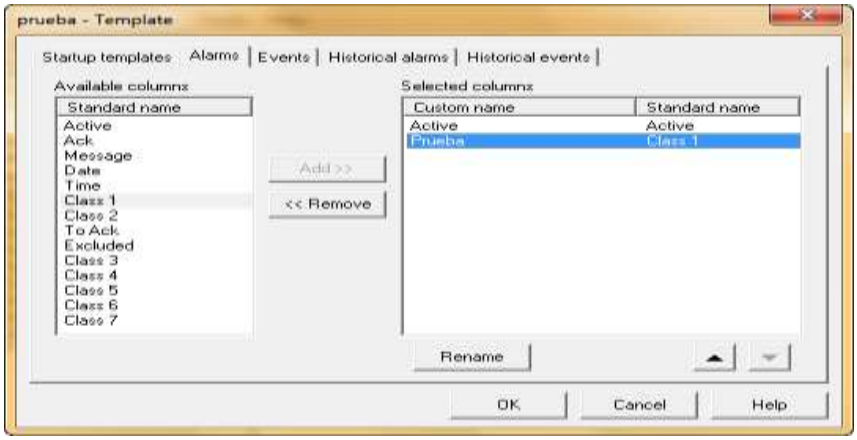


FIG. 4.18. Pestaña de Alarma y Eventos.

4.3 COMPUERTAS (GATE BUILDER)

4.3.1 GENERALIDADES

Una compuerta tiene como fin reflejar una variable del proceso. Para la configuración de las compuertas se utiliza el Gate Builder. Para acceder a él se puede realizar de dos maneras, a través de la barra de herramientas del Project Manager o a través de doble clic en alguno de los tipos de compuerta, los cuales se encuentran en la carpeta Gates de cada proyecto. La figura 4.19 muestra estas dos opciones.

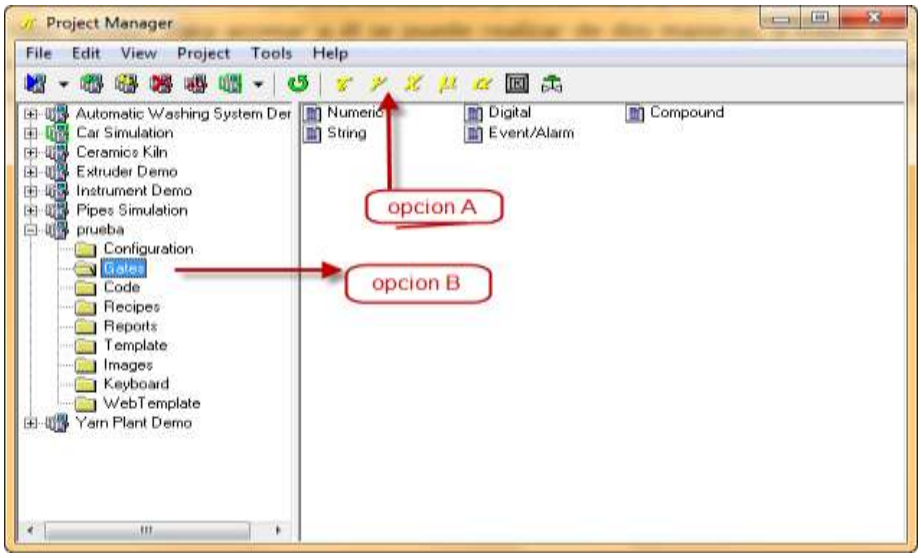


FIG. 4.19. Acceso a Gate Builder.

Al iniciar a través de la opción A, el empieza sin ningún archivo abierto, por lo cual para abrir algún tipo de compuerta se puede realizar a través de *File-Port*.

Si se inicia a través de la opción B, Gate Builder inicia con el tipo de compuerta seleccionado, tal como se muestra en la imagen 4.20.

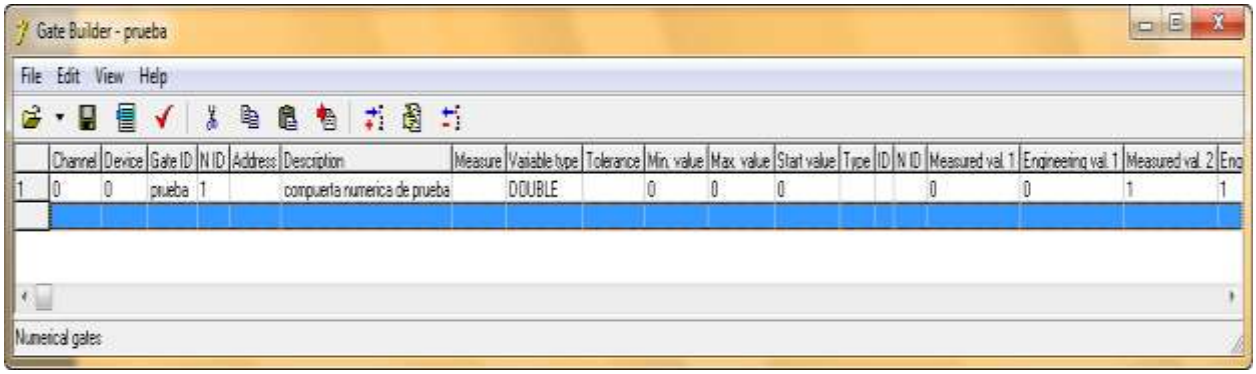


FIG.4.20.Gate Builder

En el programa se muestra en una tabla todas las compuertas del tipo seleccionado, en las columnas se muestra la información que permiten tener una visión global de las compuertas, y en las filas se listan las distintas compuertas que componen al proyecto. En la parte inferior de la ventana se muestra la información del tipo de compuertas mostradas.

Para facilitar el uso del programa, todas las funciones se pueden acceder desde la barra de herramientas ubicadas debajo de la barra de menú, o a través del menú desplegable al hacer clic con el botón secundario del mouse en la tabla. Al cargar las compuertas es posible realizar con ellas las operaciones de agregar, borrar, modificar, guardar en disco o imprimir la tabla con las compuertas.

Para modificar los parámetros de una compuerta solo es necesario dar doble clic sobre la compuerta a modificar, para agregar una nueva compuerta se puede hacer a través del menú *Edit-Insert* o haciendo doble clic en la última fila de la tabla.

Las modificaciones se pueden hacer para una compuerta individual o para un grupo de compuertas, para estos solo es necesario seleccionar las compuertas a modificar y dar clic en el menú *Edit-Edit selected Gates,* en la ventana que se despliega aparecen las propiedades comunes de las compuertas seleccionadas, las propiedades que son distintas son dejadas en blanco y se deben modificar una a una.

La impresión de la tabla se realiza a través del menú File, donde se puede seleccionar imprimir toda la tabla o imprimir solo las filas seleccionadas. Luego de haber seleccionado el tipo de impresión se despliega una ventana (Fig. 4.21) donde es posible seleccionar las columnas y la información que deseamos imprimir para las compuertas.

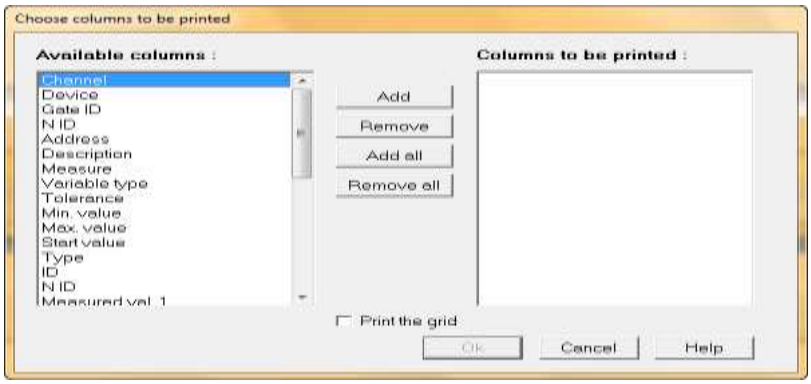


FIG.4.21.Opciones de impresión

4.3.2 DEFINICIÓN DE COMPUERTAS

En Winlog existen 5 tipos o clases de compuertas, las cuales son:

- Numerical: Todas las compuertas que pueden tener valores en coma flotante.
- Digital: Compuertas que solo pueden tener un valor de 0 o 1.
- String: Compuertas relacionadas a variables que contiene cadenas de caracteres.
- Compound: Compuertas cuyo valor no es muestreado, pero es el resultado de operaciones matemáticas entre dos puertos distintos.
- Event/Alarm: Compuertas usadas para definir alarmas y eventos de acuerdo a las condiciones de otras compuertas.

4.3.2.1 COMPUERTAS NUMÉRICAS

Al insertar una nueva compuerta de tipo numérica se puede configurar en 5 pestañas, tal como lo muestra la figura 4.22. Estas son: General, Muestreo, Valor, Conversión y Tolerancia.

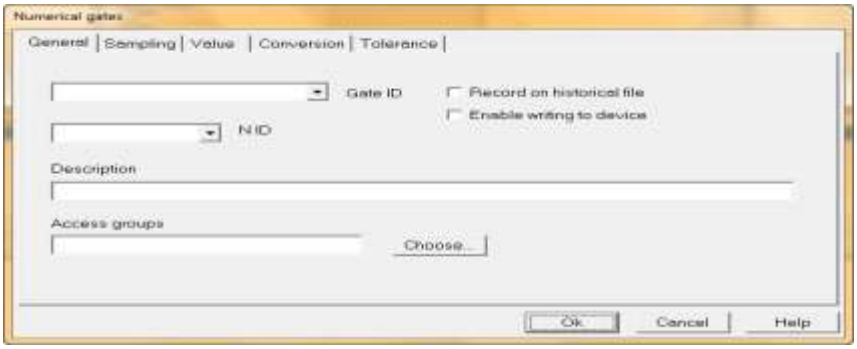


FIG.4.22. Compuertas numéricas.

En la pestaña general, se puede insertar los siguientes datos:

- Gate ID y N ID: Estas identifican a la compuerta dentro del proyecto y debe ser único. La ID debe ser un máximo de 20 caracteres y el N ID debe ser un valor entre 0 y 65535.

- Record on historical file: Si esta opción es activada, el valor proveniente del campo y leído a través de la compuerta es guardado en un archivo de texto, dependiendo de la configuración escogida para llevar el registro.

- Enable writing to device: Esto indica que cualquier cambio de valor en la compuerta debe ser transmitido al dispositivo externo.

- Description: Este campo sirve para describir a la compuerta.

- Access groups: Indica a través de un número, que grupo de usuario podrán modificar los parámetros de la compuerta.

En la pestaña sampling, tal como se puede observar en la figura 4.23, se pueden insertar los siguientes datos:

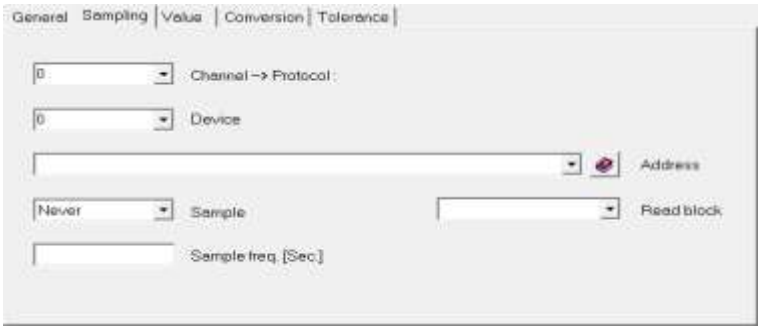


FIG.4.23. Pestaña Sampling compuertas numéricas.

- Channel: Indica a través de cual canal hacer el muestreo.

- Device: Indica el numero de dispositivo a través del cual hacer el muestreo.

- Address: Usualmente indica el tipo y el número de registro del dispositivo. Para mayor información consultar la sección de protocolo.

- Sample: Indica la forma en que se puede hacer el muestreo. Existen 3 opciones las cuales son:
  - Never: La compuerta no es muestreada.
  - Always: La compuerta siempre es muestreada.
  - If in monitor: La compuerta es muestreada solo si es necesario.

- Read Block: Permite definir bloques de compuertas adjuntas. Por lo cual un bloque puede ser leído en un solo ciclo de lectura.

- Sample Freq: Indica la diferencia en segundo entre dos muestreos. Un valor 0 indica un muestreo continuo, por lo cual su uso debe ser estrictamente si es necesario.

La pestaña value, se muestra en la figura 4.24 y los datos a insertar son:

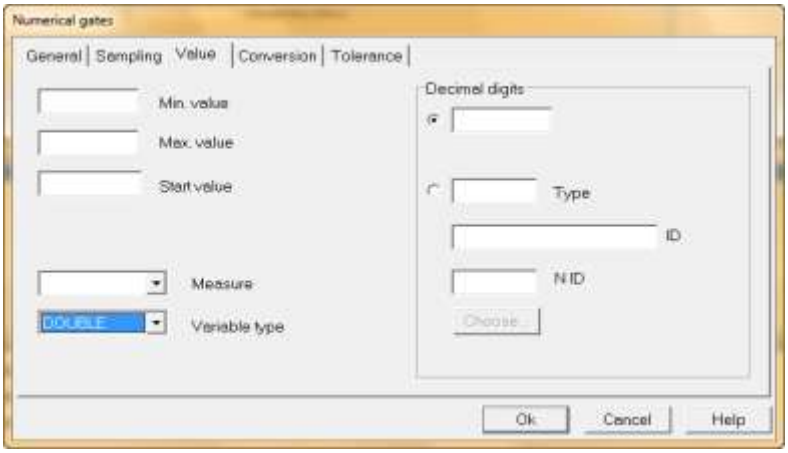


FIG.4.24. Pestaña Value compuertas numéricas.

- Min, Max Value: Los limites a aplicar al valor de la compuerta. (si ambos son 0, los límites son ignorados).

- Start Value: El valor dado a la compuerta al inicio de la aplicación.

- Mesaure: Este permite asignarle una unidad de medida a la compuerta. Esto es puramente indicativo y no afecta al valor realmente.

- Variable Type: Aquí se puede decidir entre 9 tipos de datos, dependiendo del proyecto o aplicación a realizar, estas son:
  - S_BYTE: Entero con signo de un byte (-128 // +127)
  - U_BYTE: Entero sin signo de un byte (0 // 255)
  - S_WORD: Entero con signo de dos bytes (-32768 // 32767)
  - U_WORD: Entero sin signo de dos bytes (0 // 65535)
  - S_INT32: Entero con signo de cuatro bytes (-2147483648 // 2147483647)
  - U_INT32: Entero sin signo de cuatro bytes (0 // 4294967295)
  - FLOAT: Valor decimal (3.4 // 10 -38 // 3.4 //10 +38)
  - DOUBLE: Valor decimal con doble precisión (1.7 //10-308 // 1.7 //10+308)
  - BCD: valor codificado en decimal de dos bytes sin signo (0 // 65535)

- Decimal Digits: Numero de decimales al cual debe ser redondeado el valor medido a un valor en formato ingenieril.

La pestaña Conversión se muestra en la figura siguiente y los datos a introducir son:

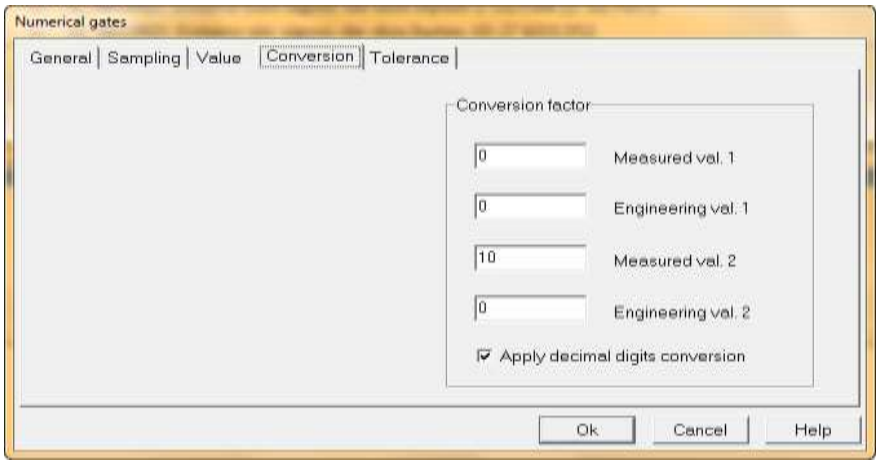


FIG.4.25. Pestaña Conversión compuertas numéricas.

- Conversión Factor: Define la conversión a aplicar a los valores leídos por el dispositivo. La conversión es realizada a través de la interpolación de un par de valores *Measured val – Engineering val.*

- Apply decimal digits conversión: Si esta opción es habilitada, el valor medido es dividido entre el valor de 10 elevado a el valor de dígitos decimales, establecidos en la pestaña value.

La pestaña Tolerance se puede observar en la figura 4.26, y los valores disponibles para insertar son:

FIG.4.26. Pestaña Tolerance compuertas numéricas.

- Tolerance: valor de tolerancia a aplicar al valor de la compuerta numérica. Esto permite filtrar los datos a incluir en los reportes o en la página de supervisión; es decir, la compuerta cambia de valor solamente si la variación es mayor o igual a la tolerancia establecida.
- Tolerance Gate: Junto con el valor de tolerancia, permite no aplicar el valor de tolerancia si el valor leído es igual al valor de la compuerta establecida como tolerancia.

4.3.2.2 COMPUERTAS DIGITALES

Al insertar una nueva compuerta de tipo digital se puede configurar en 3 pestañas, tal como lo muestra la figura 4.27. Estas son: General, Muestreo y Valor.

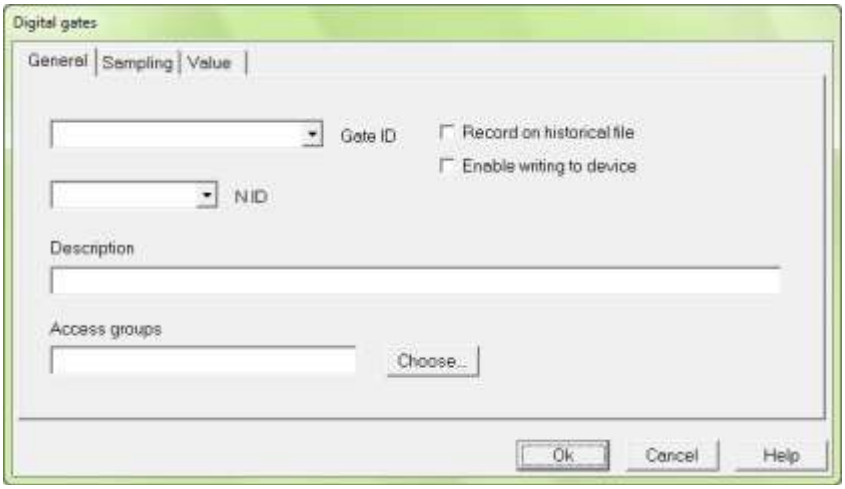


FIG.4.27. Pestaña General compuertas digitales.

En la pestaña general, se puede insertar los siguientes datos:

- Gate ID y N ID: Estas identifican a la compuerta dentro del proyecto y debe ser único. La ID debe ser un máximo de 20 caracteres y el N ID debe ser un valor entre 0 y 65535.
- Record on historical file: Si esta opción es activada, el valor proveniente del campo y leído a través de la compuerta es guardado en un archivo de texto, dependiendo de la configuración escogida para llevar el registro.
- Enable writing to device: Esto indica que cualquier cambio de valor en la compuerta debe ser transmitido al dispositivo externo.
- Description: Este campo sirve para describir a la compuerta.
- Access groups: Indica a través de un número, que grupo de usuario podrán modificar los parámetros de la compuerta.

En la pestaña sampling, tal como se puede observar en la figura 4.28, se pueden insertar los siguientes datos:

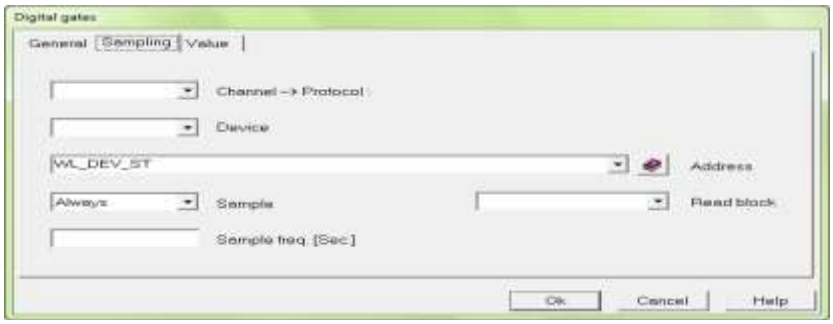


FIG.4.28. Pestaña Sampling compuertas digitales.

- Channel: Indica a través de cual canal hacer el muestreo.
- Device: Indica el numero de dispositivo a través del cual hacer el muestreo.
- Address: Usualmente indica el tipo y el número de registro del dispositivo. Para mayor información consultar la sección de protocolo.
- Sample: Indica la forma en que se puede hacer el muestreo. Existen 3 opciones las cuales son:
    - Never: La compuerta no es muestreada.
    - Always: La compuerta siempre es muestreada.
    - If in monitor: La compuertas es muestreada solo si es necesario.
- Read Block: Permite definir bloques de compuertas adjuntas. Por lo cual un bloque puede ser leído en un solo ciclo de lectura.
- Sample Freq: Indica la diferencia en segundo entre dos muestreos. Un valor 0 indica un muestreo continuo, por lo cual su uso debe ser estrictamente si es necesario.

La pestaña value, se muestra en la figura 4.29, donde se puede observar que la única configuración es la de establecer el valor inicial de la compuerta digital (0 o 1 por ser una compuerta digital).

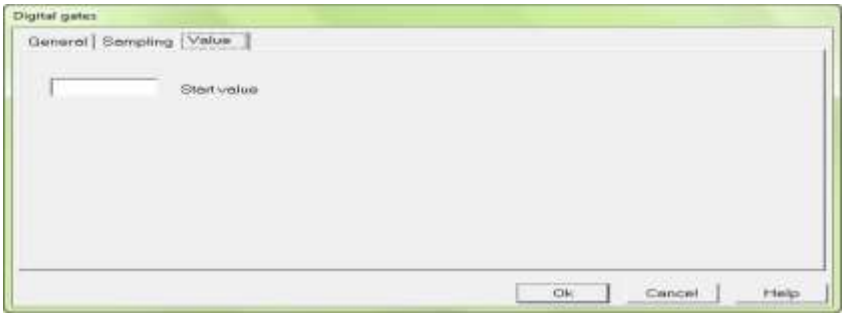


FIG.4.29. Pestaña Value compuertas digitales.

4.3.2.3 COMPUERTAS DE CARACTERES

Al igual que las compuertas digitales, las compuertas string o de caracteres se configuran en 3 pestañas las cuales son General, Muestreo y Valor, tal como lo muestra la figura 4.30.

FIG.4.30. Pestaña General compuertas String.

En la pestaña general, se puede insertar los siguientes datos:

- Gate ID y N ID: Estas identifican a la compuerta dentro del proyecto y debe ser único. La ID debe ser un máximo de 20 caracteres y el N ID debe ser un valor entre 0 y 65535.
- Record on historical file: Si esta opción es activada, el valor proveniente del campo y leído a través de la compuerta es guardado en un archivo de texto, dependiendo de la configuración escogida para llevar el registro.
- Enable writing to device: Esto indica que cualquier cambio de valor en la compuerta debe ser transmitido al dispositivo externo.
- Description: Este campo sirve para describir a la compuerta.
- Access groups: Indica a través de un número, que grupo de usuario podrán modificar los parámetros de la compuerta.

En la pestaña sampling, tal como se puede observar en la figura 4.31, se pueden insertar los siguientes datos:

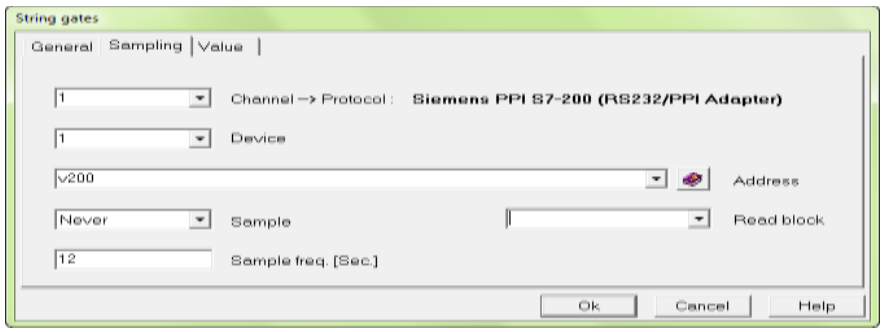


FIG.4.31. Pestaña Sampling compuertas String.

- Channel: Indica a través de cual canal hacer el muestreo.
- Device: Indica el numero de dispositivo a través del cual hacer el muestreo.
- Address: Usualmente indica el tipo y el número de registro del dispositivo. Para mayor información consultar la sección de protocolo.
- Sample: Indica la forma en que se puede hacer el muestreo. Existen 3 opciones las cuales son:
  - Never: La compuerta no es muestreada.
  - Always: La compuerta siempre es muestreada.
  - If in monitor: La compuertas es muestreada solo si es necesario.
- Read Block: Permite definir bloques de compuertas adjuntas. Por lo cual un bloque puede ser leído en un solo ciclo de lectura.
- Sample Freq: Indica la diferencia en segundo entre dos muestreos. Un valor 0 indica un muestreo continuo, por lo cual su uso debe ser estrictamente si es necesario.

La pestaña value, se muestra en la figura 4.32, donde se puede configurar lo siguiente:

- Max Dimension: Indica la máxima longitud de la cadena de la compuerta. El software maneja como máximo un valor de 80 caracteres.

- Start Value: La cadena a asignarle a la compuerta en el inicio.

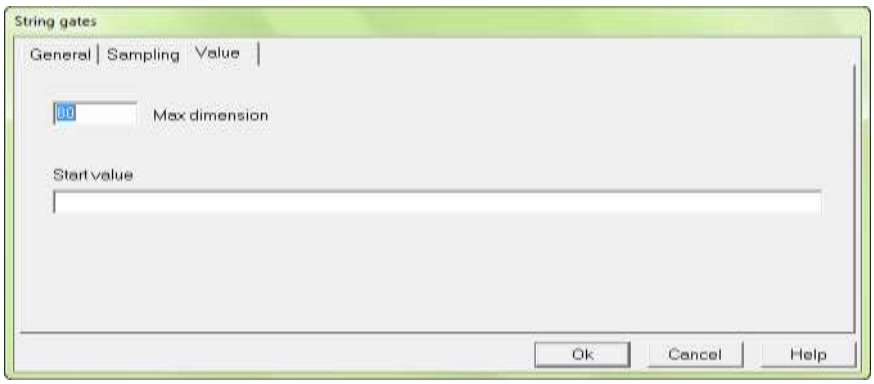


FIG.4.32. Pestaña Value compuertas digitales.

4.3.2.4 COMPUERTAS COMPUESTAS

Las compuertas compuestas son compuertas ficticias, que no corresponden con ningún registro de ningún dispositivo. Su valor es un resultado matemático de la operación con el valor de otra compuerta. Sin embargo, para cada compuerta es posible especificar un registro de un dispositivo donde escribir el valor al cambiar su estado, estas compuertas tienen como propósito realizar operaciones entre compuertas directamente, sin tener que hacerlo dentro del código.

Las propiedades se pueden configurar usando 4 pestañas, las cuales son General, Writing, Operation y Value, tal como se observa en la figura 4.33.

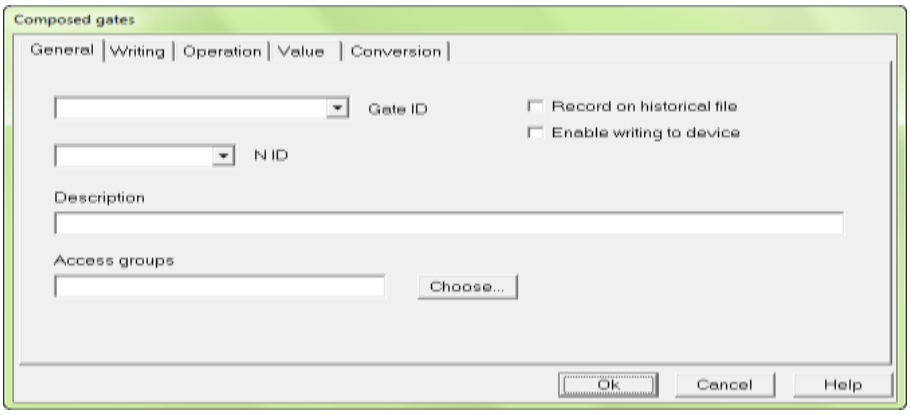


FIG.4.33. Pestaña General compuertas compuesta.

En la pestaña general, se puede insertar los siguientes datos:

- Gate ID y N ID: Estas identifican a la compuerta dentro del proyecto y debe ser único. La ID debe ser un máximo de 20 caracteres y el N ID debe ser un valor entre 0 y 65535.

- Record on historical file: Si esta opción es activada, el valor proveniente del campo y leído a través de la compuerta es guardado en un archivo de texto, dependiendo de la configuración escogida para llevar el registro.
- Enable writing to device: Esto indica que cualquier cambio de valor en la compuerta debe ser transmitido al dispositivo externo.
- Description: Este campo sirve para describir a la compuerta.
- Access groups: Indica a través de un número, que grupo de usuario podrán modificar los parámetros de la compuerta.

En la pestaña writing, tal como se puede observar en la figura 4.34, es posible indicar el dispositivo y registro donde escribir, por lo cual se pueden insertar los siguientes datos:

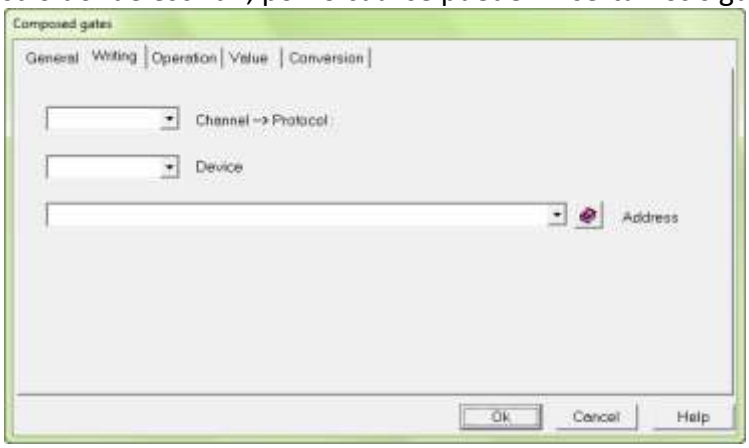


FIG.4.34. Pestaña Writing compuertas compuestas.

- Channel: Indica a través de cual canal hacer la escritura.
- Device: Indica el numero de dispositivo en el cual escribir las variaciones.
- Address: Usualmente indica el tipo y el número de registro del dispositivo. Para mayor información consultar la sección de protocolo.

En la pestaña Operation se especifica las compuertas con las cuales se contribuirá para obtener el valor de la compuerta. Las opciones de configuración se muestran en la figura 4.35 y estas son:

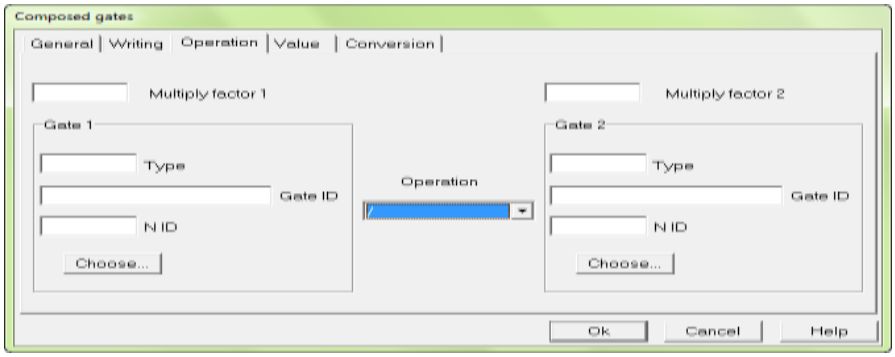


FIG.4.35. Pestaña Operation compuertas compuestas.

- Gate 1: Es la primera compuerta involucrada en la operación. Para elegir la compuerta solo se debe presionar en Choose, y se desplegara la ventana mostrada en la figura 4.36, desde donde se puede escoger la compuerta correspondiente.

  Para escoger la compuerta se debe escoger primero el tipo de compuerta, esta puede ser numérica, digital, compuesta (Generar jerarquía) y de eventos. Luego de seleccionar una, los datos de la compuerta se muestran en los campos de Gate ID y N ID.

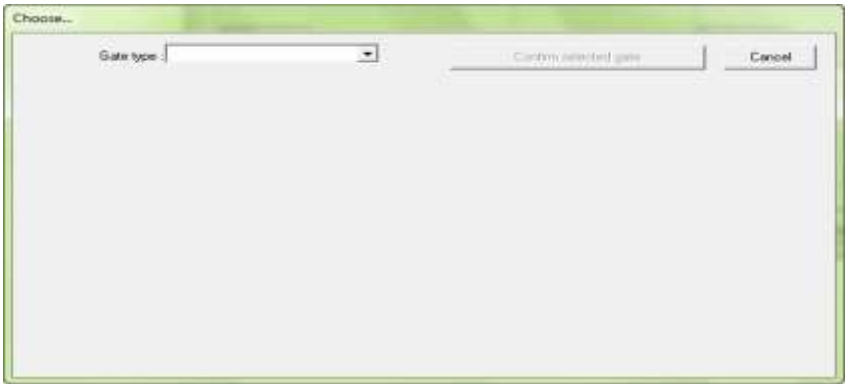


FIG.4.36. Ventana selección de compuertas.

- Multiply Factor 1 &2: Indica el factor por el cual se multiplicara el valor de la compuerta 1 o 2 respectivamente.
- Gate 2: Es la segunda compuerta con la cual se conforma el valor de la compuerta compuesta. La forma de escogerla es igual a la descrita para la compuerta 1.
- Operation: Es la operación que involucra a las compuertas seleccionadas para conformar el valor de la compuerta compuesta. Estas pueden ser:
  - + = Compuesta =Gate 1 + Gate 2
  - - = Compuesta =Gate 1 - Gate 2
  - * = Compuesta =Gate 1 * Gate 2
  - / = Compuesta =Gate 1 / Gate 2
  - Discard HI: si Gate 1 > Gate 2 el valor no es calculado de otra manera el valor = Gate 1.
  - Discard LO: si Gate 1 < Gate 2 el valor no es calculado de otra manera el valor = Gate 1.
  - Limit Hi: Si Gate 1 > Gate 2 entonces el valor = Gate 2 de otra manera el valor =Gate 1.
  - Limit Lo: Si Gate 1 < Gate 2 entonces el valor = Gate 2 de otra manera el valor =Gate 1.
  - Reset If: si Gate 1=Gate2 entonces el valor = 0 de otra manera el valor = 1.

La pestaña value, se muestra en la figura 4.37, donde se puede configurar el valor asignado mientas se lee. Es posible establecer el número de decimales a redondearlo.

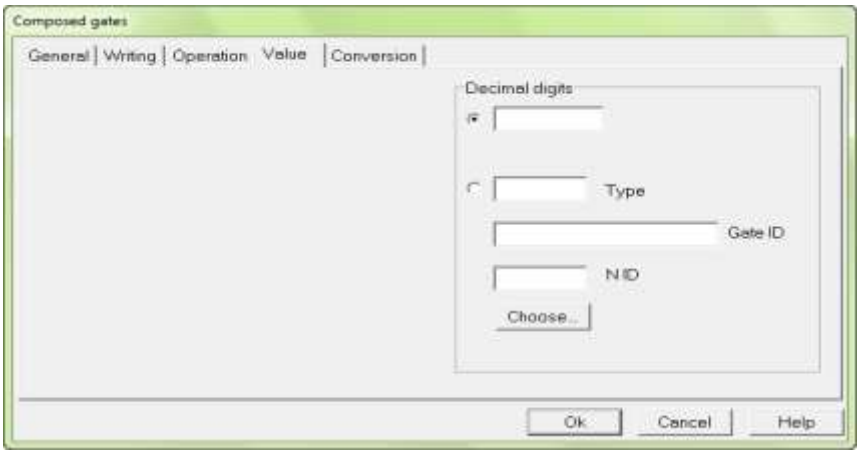


FIG.4.37. Pestaña Value compuertas compuestas.

4.3.2.4 COMPUERTAS DE EVENTO

Las compuertas de evento permiten mantener bajo control algunas variables críticas del proyecto. Una compuerta de evento de hecho solo se activa si la condición por la cual ha sido creada toma lugar. Esta condición está establecida específicamente a un tipo de comparación entre el valor leído y el valor establecido.

Durante la etapa de supervisión todas las compuertas de evento son guardadas para mantener el registro de funcionamiento. Para tener mayor flexibilidad es posible crear compuertas de evento especiales como son las compuertas de alarma. Estas son idénticas con la diferencia que en la etapa de supervisión las compuertas de alarma avisan al usuario sobre su activación en la barra de estado.

La configuración de las compuertas se lleva a cabo a través de 4 pestañas, las cuales son General, Condition, Message y Class tal como se observa en la figura 4.38.

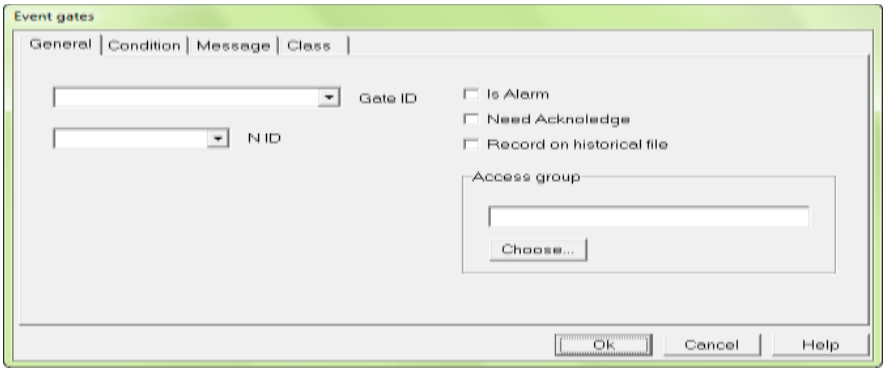


FIG.4.38. Pestaña General compuertas de evento.

La información genérica de este tipo de compuertas, se introduce en la pestaña General, donde se puede definir:

- Gate ID y N ID: Estas identifican a la compuerta dentro del proyecto y debe ser único. La ID debe ser un máximo de 20 caracteres y el N ID debe ser un valor entre 0 y 65535.
- Record on historical file: Si esta opción es activada, el valor proveniente del campo y leído a través de la compuerta es guardado en un archivo de texto, dependiendo de la configuración escogida para llevar el registro.
- Access groups: Indica a través de un número, que grupo de usuario podrán modificar los parámetros de la compuerta.
- Is Alarm: si es seleccionada, el evento es considerado una alarma, para poder tener un aviso inmediato durante la etapa de supervisión.
- Need Acknoledge: Seleccionada esta opción, se mantiene el evento en la tabla de eventos activos hasta que es confirmado por el operador, incluso cuando la condición que produce la activación ya no esté presente.

La pestaña Condition, se muestra en la figura 4.39, donde se establece la condición con la cual se creara el evento.

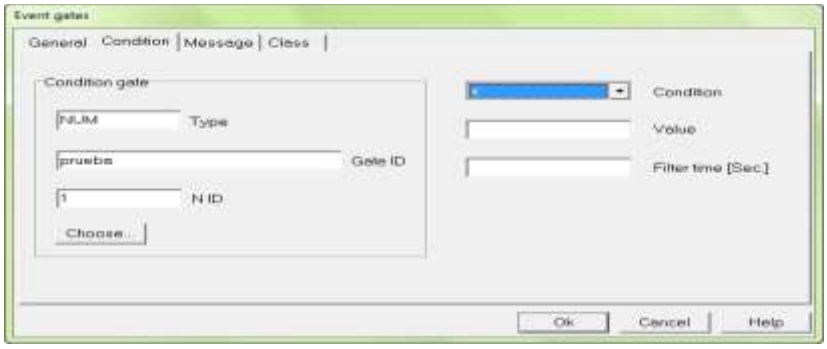


FIG.4.39. Pestaña Condition compuertas de evento.

- Condition Gate: Es la compuerta de la cual leer el valor que provocara la verificación de la condición. Presionando el botón aparecerá la pantalla de selección de compuerta con la que se comprueba la igualdad.
- Condition: Este indica la condición con la cual se activara el evento. La condición debe ser escogida entre las disponibles. Estas dependen del tipo de compuerta escogida como referencia. En la tabla siguiente se pueden observar las distintas condiciones disponibles.

Condition Gate	Condición de Activación de Evento	Significado
Numérica	=	El valor de la condición Gate debe ser igual al valor
	<>	El valor de la condición Gate debe ser diferente al valor
	>	El valor de la condición Gate debe ser mayor al valor
	>=	El valor de la condición Gate debe ser mayor o igual al valor

Compuesta	<	El valor de la condición Gate debe ser menor al valor
	<=	El valor de la condición Gate debe ser menor o igual al valor
	OR	La condición OR bit a bit entre la condición Gate y el valor debe ser diferente de cero.
	Bit n°/	El valor del bit i del valor de la condición Gate debe ser igual al del valor.
	=	El valor de la condición Gate debe ser igual al valor
	<>	El valor de la condición Gate debe ser diferente al valor
Digital	Variation	El valor de la condición Gate debe cambiar
	->1	El valor de la condición Gate debe cambiar de cero a uno
	->0	El valor de la condición Gate debe cambiar de uno a cero
String	=	El valor de la condición Gate debe ser igual al valor
	<>	El valor de la condición Gate debe ser diferente al valor

TABLA 4.1. Condiciones compuertas de evento.

- Value: Este indica el valor con el cual se compara la compuerta de referencia (dependiendo de la condición escogida), por lo cual puede ser numérica o cadena dependiendo de la compuerta de referencia.
- Filter Time: Este indica el tiempo mínimo que la condición se debe de cumplir para crear el evento. Estableciendo correctamente este valor es posible que el software evite activarse con evento cortos.

La pestaña Message, se muestra en la figura 4.36, donde se establece la leyenda a mostrar con la activación del evento, en la tabla de eventos activos o en la barra de estado (si el evento es una alarma).

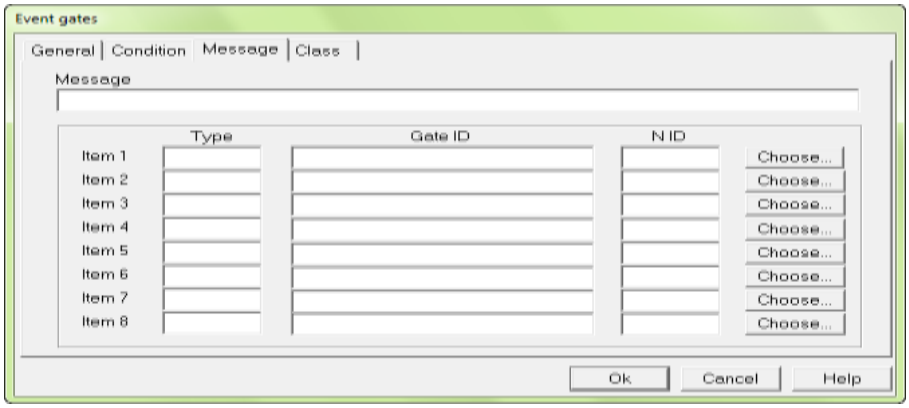


FIG.4.40. Pestaña Message compuertas de evento.

Se usa % en el mensaje para especificar el punto de inserción de el valor de la compuerta, luego se especifica en la lista inferior (presionando el botón choose) la compuerta cuyo valor se necesita.

- %s: es usado para insertar el valor de una compuerta de caracteres.
- %lf: es usado para insertar un valor de una compuerta numérica.

El parámetro %lf puede ser usado además para indicar el número máximo de dígitos y el número de decimales de esta forma:

- %x.ylf:

  Donde X es el número de dígitos a mostrar, todos los dígitos disponibles se mostrara si este parámetro no es especificado. Y es un parámetro opcional para indicar los decimales.

Los valores de compuertas son insertados en el mensaje donde se encuentren parámetros % y en el mismo orden de la lista.

La pestaña Class, se muestra en la figura 4.41, donde se establece las etiquetas para mostrar las compuertas de evento en la tabla de eventos activos.

FIG.4.41. Pestaña Class compuertas de evento.

- Class 1: Es posible establecer un valor numérico con el cual se puede ordenar la tabla de eventos activos.

- Class 2: Es posible establecer una cadena de caracteres con el cual se puede ordenar y buscar en la tabla de eventos activos.

- Class 3-9: Es posible vincular clases con compuertas. Cada vez que el evento se activa estas clases capturan el valor de las compuertas vinculadas a ellas.

4.4 CODIGO (CODE BUILDER)

4.4.1 GENERALIDADES

El Code Builder es un ambiente de desarrollo diseñado para hacer más fácil la escritura de funciones en este lenguaje. Este permite escribir código, guardarlo permanentemente, crear nuevo y corregir la sintaxis. Además, para reducir la cantidad de errores el texto se diferencia por colores dependiendo del rol que desarrollo dentro de la función.

Para abrir el software se puede realizar de dos maneras, a través de la barra de herramientas del Project Manager o a través de doble clic en alguna de las funciones ya creadas, las cuales se encuentran en la carpeta Code de cada proyecto. La figura 4.42 muestra estas dos opciones.

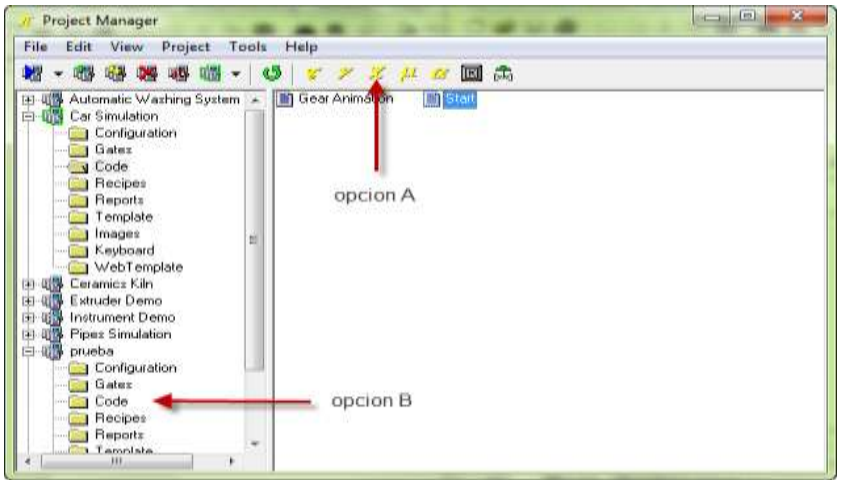


FIG.4.42. Inicio de Code Builder

La pantalla de inicio del programa dependiendo de la opción seleccionada para acceder al software; puede ser en blanco o con una función ya abierta. Para este caso se trabaja con la opción A; es decir, con el programa sin ninguna función abierta, tal como se ve en la figura 4.43.

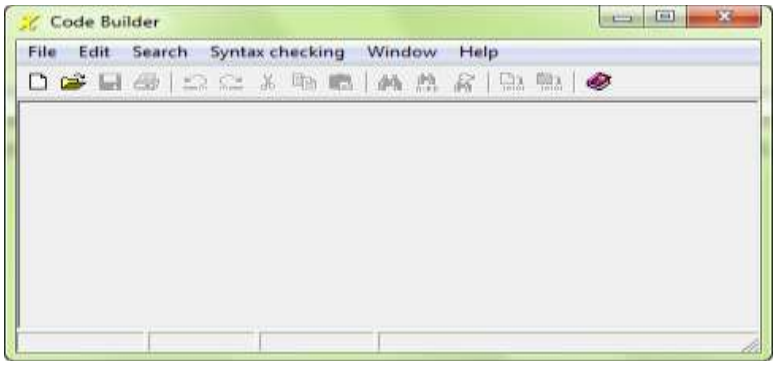


FIG.4.43. Code Builder

Para acceder a una función se utiliza el botón abrir o desde el menú *File/Open*, donde se puede navegar a través de las carpetas para abrir la función deseada, en el caso, las funciones de los proyectos se almacenan individualmente en la carpeta Code tal como se muestra en la figurar 4.44.

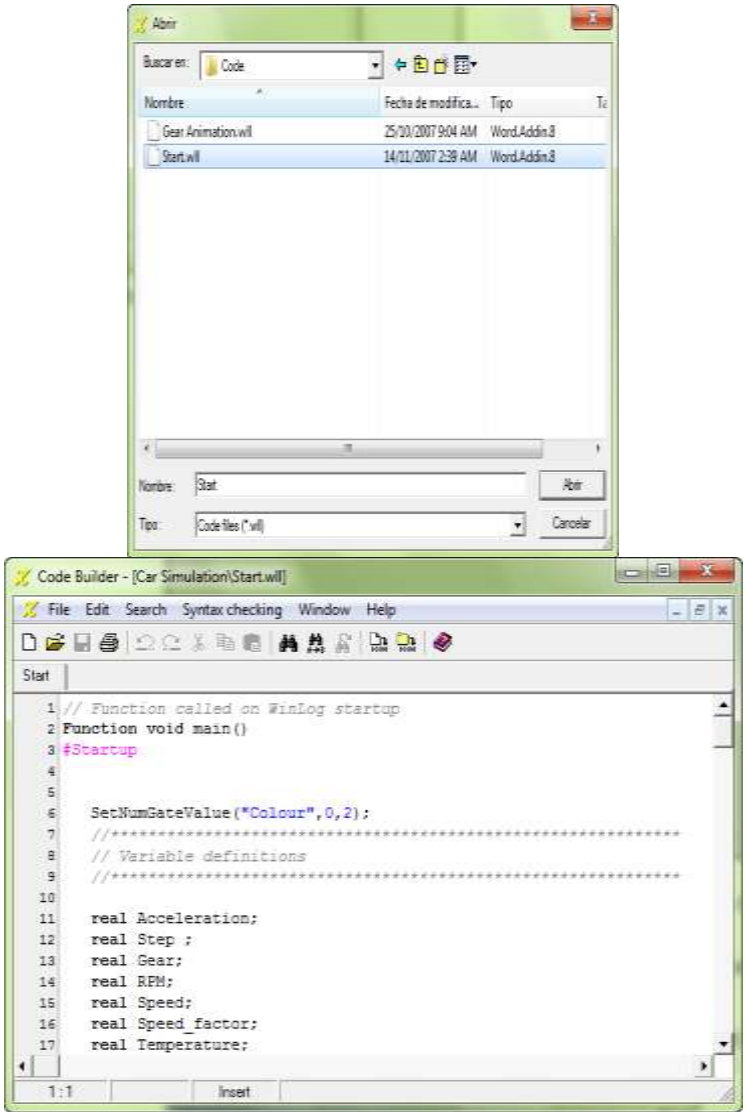


FIG.4.44. Abrir Función en Code Builder

Se puede notar, los diferentes colores en el código de ejemplo. El código tiene esos colores de acuerdo al análisis hecho por el identificador de Code Builder que categoriza los elementos de texto. Los atributos de cada categoría se pueden cambiar, mientras que las reglas que diferencias las clases de elementos no pueden ser modificadas y tienen su origen en la sintaxis del lenguaje.

Con el teclado es posible acceder a muchas funciones que están en los menús. Sin embargo, hay muchas funciones que solo pueden ser accedidas a través del teclado. La tabla 4.2 muestra estas funciones.

	Keys	Function
Also present in the menu	CTRL + N	Creates a new code window
	CTRL + O	Opens a WinLog code file
	CTRL + S	Saves the working file
	CTRL + Z	Undo
	CTRL + C	Copy to clipboard the selected text
	CTRL + X	Deletes the selected text
	CTRL + V	Pastes from the clipboard
	CTRL + F	Search in the text
	CTRL + R	Search and replaces text
	CTRL + F9	Check the syntax of the working code
	F3	Repeats the last search
	F9	Checks the syntax of the entire project
	F1	Calls up the contextual help
Available only with keystrokes	INS	Changes the insert state
	CTRL + Y	Deletes the row at the cursor
	ALT + N	Brings up the next code window
	ALT + P	Brings up the previous code window
	CTRL + LEFT	Move the cursor to the next left word
	CTRL + RIGHT	Move the cursor to the next right word
	CTRL + HOME	Move the cursor to the first line
	CTRL + END	Move the cursor to the last line

TABLA 4.2. Funciones a través del teclado.

El ratón tiene una función especial en el Code Builder, ya que el botón derecho abre un menú contextual que ayuda al usuario a escribir código.

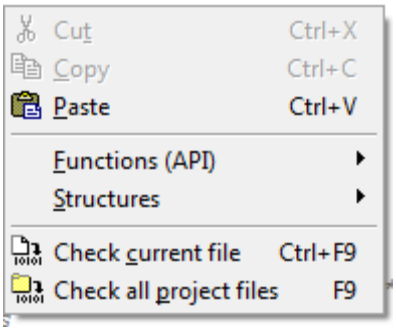


FIG.4.45. Menú Contextual Botón Derecho

Las funciones API se refieren a todas las funciones que hay en las librerías, estas se pueden navegar a través de este menú. Si se le da clic en una función, el Code Builder agrega el prototipo en la posición del ratón en la ventana activa.

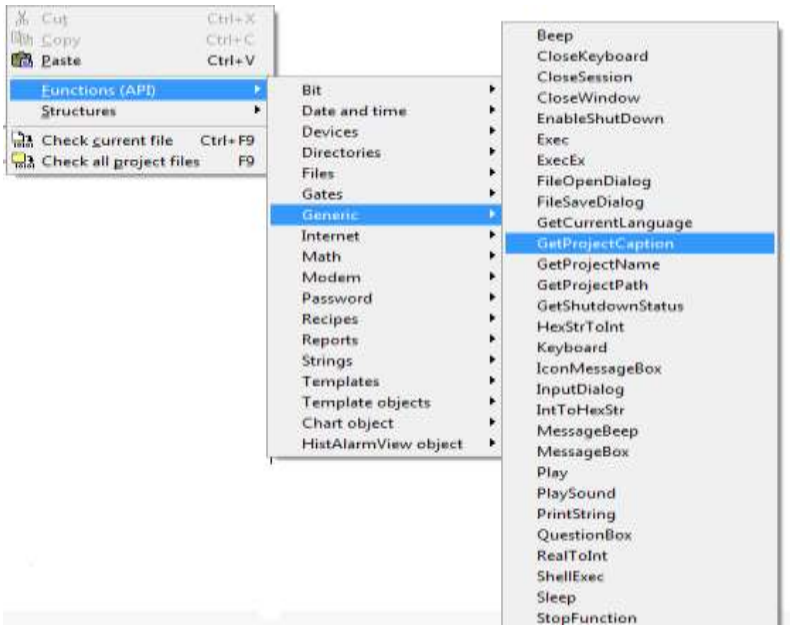


FIG.4.46. Menú Contextual Funciones API

Al igual que las funciones, se pueden navegar y agregar estructuras de lenguaje.

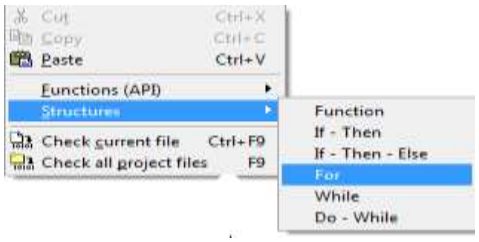


FIG.4.47. Menú Contextual Estructuras.

4.4.2 PREFERENCIAS

Hay algunos aspectos integrados del ambiente que se pueden cambiar. Estos se pueden lograr a través del menú *Edit* seleccionando *Preference.* En la ventana de preferencias se tienen 2 pestañas: Color y Editor.

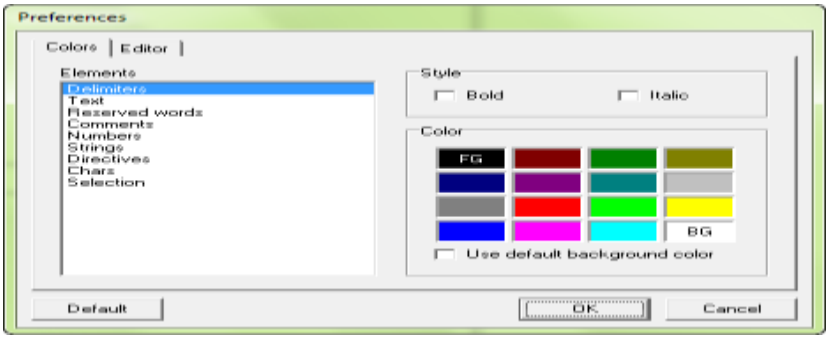


FIG.4.48. Menú Preferencias

Es posible configurar el color y el texto de cualquier forma deseada. Las reglas que diferencian al texto vienen directamente de la sintaxis del lenguaje.

Escogiendo un elemento de la lista de la ventana mostrada en la figura 4.49, la pestaña es actualizada automáticamente mostrando la configuración para ese elemento. Para modificar la configuración de un elemento, solo es necesario marcar la casilla del estilo deseado. Similarmente, para escoger el color se debe dar clic en la casilla del color deseado: haciendo clic con el botón izquierdo se escoge el color del texto (FG), mientras, usando el botón derecho se escoge el color de fondo (BG).

En la pestaña Editor tal como se observa en la figura siguiente, se puede configurar la ventana de código. Escogiendo si mostrar o no el fin del archivo, la columna derecha indica el margen de la pagina, etc.

FIG.4.49. Pestaña Editor Menú Preferencias

4.4.3 FUNCIONES API

Las funciones API son aquellas que ya están definidas. Estas pueden ser llamadas desde cualquier programa. En esta sección se describirán las funciones más usadas, separándolas por tema.

BIT

- BitAnd: Realiza la operación AND bit a bit entre Value1 y Value2.
  Sintaxis: *int BitAnd (int Value1, int Value2)*
  Parámetros: Value 1: Primer número a procesar
  Value 2: Segundo numero a procesar
  Retorno: El resultado de la operación AND.

- BitMask: Realiza la operación AND bit a bit entre un numero y unas mascara.
  Sintaxis: *int BitMask (int Num, int Mask)*
  Parámetros: Num: Primer número a procesar
  Mask: mascara (posición de bit desde 0 a 31)
  Retorno: El resultado de la operación AND.

- BitNot: Realiza la negación de bit de Value
  Sintaxis: *int BitNot (int Value)*
  Parámetros: Value: Valor a procesar
  Retorno: El resultado de la operación NOT.

- BitOR: Realiza la operación OR bit a bit entre Value1 y Value2.
  Sintaxis: *int BitOr (int Value1, int Value2)*
  Parámetros: Value 1: Primer número a procesar
  Value 2: Segundo numero a procesar
  Retorno: El resultado de la operación OR

- BitXor: Realiza la operación XOR bit a bit entre Value1 y Value2.
  Sintaxis: *int BitXor (int Value1, int Value2)*
  Parámetros: Value 1: Primer número a procesar
  Value 2: Segundo numero a procesar
  Retorno: El resultado de la operación XOR

- GetBit: Retorna un bit de un numero.
  Sintaxis: *int GetBit (int Num, int Bit)*
  Parámetros: Num: Numero a procesar
  Bit: Bit a comprobar (0 – 31)
  Retorno: Bit Requerido

- ResetBit: Resetea (pone a 0) un bit de un número.
  Sintaxis: *int ResetBit(int Num, int Bit)*
  Parámetros: Num: Numero a procesar
  Bit: Bit a resetear (0 – 31)
  Retorno: Numero con el bit a 0

- SetBit: Establece (pone a 1) un bit de un número.
  Sintaxis: *int SetBit (int Num, int Bit)*
  Parámetros: Num: Numero a procesar
  Bit: Bit a establecer (0 – 31)
  Retorno: Numero con el bit a 1

TIEMPO Y FECHA

- DataTimeToSeconds: Convierte la fecha de entrada a un total de segundos desde 1 de Enero de 1901.
  Sintaxis: *Unsigned DateTimeToSeconds (int Day, int Month, int Year, int Hour, int Minute, int Second)*
  Retorno: Tiempo en segundos desde 1901.

- GetDayOfMonth: Retorna un numero entero representando el dia actual.
  Sintaxis: *Int GetDayOfMonth()*
  Retorno: El dia actual como un numero entero.

- GetDayOfWeek: Retorna un numero entero representando el dia actual de la semana.(0-7)
  Sintaxis: *Int GetDayOfWeek()*
  Retorno: El dia actual como un numero entero.

- GetHour: Retorna un numero entero representando la hora actual.
  Sintaxis: *Int GetHour()*
  Retorno: La hora actual como un numero entero.

- GetMinute: Retorna un numero entero representando el numero de minutos de la hora actual.
  Sintaxis: *Int GetMinute ()*
  Retorno: La cantidad de minutos de la hora actual como un numero entero.

- GetSecond: Retorna un numero entero representando el numero de segundos de la hora actual.
  Sintaxis: *Int GetSecond()*
  Retorno: La cantidad de segundos de la hora actual como un numero entero.

DISPOSITIVOS

- DeviceEnableCommunication: Habilita o deshabilita las comunicaciones con el dispositivo.
  Sintaxis: *Bool DeviceEnableCommunication(Int Channel, Int DevNum, Bool ToEnable, Bool SaveChanges)*
  Parámetros: Channel: Numero de canal
  DevNum: Numero de dispositivo
  ToEnable: si es "true" la comunicación se habilita
  Si es "false" la comunicación se deshabilita.
  SaveChanges: si es "true" los cambios serán permanentemente guardados.
  Si es "false" los cambios solo serán para esa sesión.
  Retorno: Retorna un true si se ha ejecutado con éxito.

- GetDeviceRXErrors : Retorna el numero de errores de comunicación ocurridos durante toda la lectura de datos del dispositivo.
  Sintaxis: *Int GetDeviceRxErrors(Int Channel, Int DevNum)*
  Parámetros: Channel: Numero de canal
  DevNum: Numero de dispositivo
  Retorno: Numero de errores ocurridos.

- GetDeviceTxErrors: Retorna el numero de errores de comunicación ocurridos durante toda la escritura de datos al dispositivo.
  Sintaxis: *Int GetDeviceTxErrors(Int Channel, Int DevNum)*

Parámetros: Channel: Numero de canal
DevNum: Numero de dispositivo
Retorno: Numero de errores ocurridos.

COMPUERTAS

Las funciones que involucran compuertas se agrupan de acuerdo al tipo de la compuerta, aquí se muestran las funciones más utilizadas para cada tipo.

Compuertas Numéricas

- GetNumGateComunnicationStatus: Retorna el estado de la compuerta especificada.
  Sintaxis: *Bool GetNumGateCommunicationStatus(String Name, Int Id)*
  Parámetros: Name: nombre de la compuerta a leer
  Id: Identificador de la compuerta a leer
  Retorno: El estado de la compuerta True (si la compuerta esta OK) o False (si la compuerta esta KO).

- GetGateNumValue: Retorna el valor de la compuerta especificada a través del nombre y el identificador numérico.
  Sintaxis: *Real GetNumGateValue(String Name, Int Id)*
  Parámetros: Name: nombre de la compuerta a leer
  Id: Identificador de la compuerta a leer
  Retorno: Valor de la compuerta especificada.

- GetGateNumValueAsString: Retorna el valor de la compuerta especificada como una cadena de caracteres.
  Sintaxis: *String GetNumGateValueAsString(String Name, Int Id,String Format)*
  Parámetros: Name: nombre de la compuerta a leer
  Id: Identificador de la compuerta a leer
  String Format: Formato del texto a retornar
  Retorno: Valor de la compuerta en texto con el formato establecido.

- SetNumGateValue: Establece un nuevo valor a la compuerta especificada.
  Sintaxis: *Bool SetNumGateValue(String Name, Int Id, Real Value)*
  Parámetros: Name: nombre de la compuerta a leer
  Id: Identificador de la compuerta a leer
  Value: Valor nuevo a darle a la compuerta
  Retorno: True si la compuerta existe y su valor ha cambiado
  False si no se ha realizado

Compuertas Digitales

- GetDigGateComunnicationStatus: Retorna el estado de la compuerta especificada.

Sintaxis: *Bool GetDigGateCommunicationStatus(String Name, Int Id)*
Parámetros: Name: nombre de la compuerta a leer
Id: Identificador de la compuerta a leer
Retorno: El estado de la compuerta True (si la compuerta esta OK) o False (si la compuerta esta KO).

- GetDigGateValue: Retorna el valor de la compuerta especificada a través del nombre y el identificador numérico.
  Sintaxis: *Int GetDigGateValue(String Name, Int Id)*
  Parámetros: Name: nombre de la compuerta a leer
  Id: Identificador de la compuerta a leer
  Retorno: Valor de la compuerta especificada.

- SetDigGateValue: Establece un nuevo valor a la compuerta especificada.
  Sintaxis: *Bool SetDigGateValue(String Name, Int Id, int Value)*
  Parámetros: Name: nombre de la compuerta a leer
  Id: Identificador de la compuerta a leer
  Value: Valor nuevo a darle a la compuerta
  Retorno: True si la compuerta existe y su valor a cambiado
  False si no se ha realizado

Compuertas Compuestas

- CmpGateExists: Comprueba si una compuerta existe.
  Sintaxis: *Bool CmpGateExists(String Name, Int Id)*
  Parámetros: Name: nombre de la compuerta a buscar
  Id: Identificador de la compuerta a buscar
  Retorno: True si la compuerta existe y está definida en la aplicación
  False si no se ha definido en la aplicación.
- GetCmpGateValue: Retorna el valor de la compuerta especificada a través del nombre y el identificador numérico.
  Sintaxis: Real *GetDigGateValue(String Name, Int Id)*
  Parámetros: Name: nombre de la compuerta a leer
  Id: Identificador de la compuerta a leer
  Retorno: Valor de la compuerta especificada.

Compuertas String

- GetStrGateComunnicationStatus: Retorna el estado de la compuerta especificada.
  Sintaxis: *Bool GetStrGateCommunicationStatus(String Name, Int Id)*
  Parámetros: Name: nombre de la compuerta a leer
  Id: Identificador de la compuerta a leer
  Retorno: El estado de la compuerta True (si la compuerta esta OK) o False (si la compuerta esta KO).
- GetStrGateValue: Retorna el valor de la compuerta especificada a través del nombre y el identificador numérico.

Sintaxis: *String GetStrGateValue(String Name, Int Id)*
Parámetros: Name: nombre de la compuerta a leer
Id: Identificador de la compuerta a leer
Retorno: Valor de la compuerta especificada.

- SetStrGateValue: Establece un nuevo valor a la compuerta especificada.
  Sintaxis: *Bool SetNumGateValue(String Name, Int Id, String Value)*
  Parámetros: Name: nombre de la compuerta a leer
  Id: Identificador de la compuerta a leer
  Value: Valor nuevo a darle a la compuerta
  Retorno: True si la compuerta existe y su valor a cambiado
  False si no se ha realizado

Compuertas de Evento

- EvnGateExists: Comprueba si una compuerta de estado/alarma existe.
  Sintaxis: *Bool EvnGateExists(String Name, Int Id)*
  Parámetros: Name: nombre de la compuerta a buscar
  Id: Identificador de la compuerta a buscar
  Retorno: True si la compuerta existe y está definida en la aplicación
  False si no se ha definido en la aplicación.

- GetEvnGateAckedStatus: Este retorna el estado de adquisición de la alarma a través de nombre y número de identificación.
  Sintaxis: *Bool EvnGateExists(String Name, Int Id)*
  Parámetros: Name: nombre de la compuerta a buscar
  Id: Identificador de la compuerta a buscar
  Retorno: True si la alarma ha sido atendida por el operador
  False si la alarma no ha sido atendida por el operador

- GetEvtGateMsg: Retorna el mensaje de una compuerta de evento a través de número y nombre de ID.
  Sintaxis: *String GetEvnGateValue(String Name, Int Id)*
  Parámetros: Name: nombre de la compuerta a buscar
  Id: Identificador de la compuerta a buscar
  Retorno: Mensaje del evento.

PLANTILLA

- TPageClose: Cierra la plantilla especificada.
  Sintaxis: *void TPageClose(Integer Page)*

- TPageOpen: Abre la plantilla dando un nombre especifico
  Sintaxis: *int TPageOpen(String Name)*

Para obtener más información sobre las funciones API disponibles en el Code Builder, se puede consultar la referencia bibliográfica del capítulo.

4.4.4 Elementos del Lenguaje

El lenguaje esta hecho básicamente de elementos básicos (letras, números, puntuación, etc.), reglas de semántica y sintaxis que especifica la forma correcta en que puede ser creado.

Las instrucciones son elementos que involucran algunas variables y funciones gracias a operadores provistos por el lenguaje en sí.

Las variables son áreas de memoria donde se puede almacenar información necesaria. Existen varios tipos de variables y dependen de la cantidad de información a almacenar.
Las funciones son un grupo de instrucciones que están agrupadas, como una forma de volver a escribirlas y porque representan una forma de instrucción más compleja.

De manera de simplificar la escritura de código en el lenguaje hay algunas estructuras como los son las condiciones, ciclos, ciclos condicionados, etc.

Los archivos interpretados por *Run Time* tienen como la extensión *"WLL".* Para el intérprete del lenguaje no hay diferencia entre mayúsculas y minúsculas. La doble pleca // indica el inicio de un comentario.

4.4.4.1 VARIABLES

En cada archivo se pueden declarar variables y funciones globales, las cuales son visibles a todas las funciones. Estas son declaradas al inicio de cada archivo, la visibilidad indica el poder acceder a los datos de esa variable.

Las variables son elementos que contienen los valores a manipular durante la ejecución de un programa o función. Dentro del lenguaje existen dos tipos de variables las cuales son:

- Global: Son visibles para todos las funciones incluso si no están en el mismo archivo.
  Sintaxis: GLOBAL Tipo Nombre (=Default Value);

- Local: Son variables definidas en cada función. Solo pueden ser usadas en el contexto de la función.
  Sintaxis: Tipo Nombre (=Default Value);

Las variables pueden contener distintos tipos de datos por lo cual se dividen en categorías o tipos entre los que se encuentran:

- Int: Son valores enteros de 32 bits con signo.
- Real: Son valores con punto flotante que puede almacenar números con gran precisión.
- Bool: Son valores booleanos por lo cual solo pueden ser verdadero (True) o falso(False).
- String: Están son cadenas de caracteres con distintas longitudes.

4.4.4.2 FUNCIONES

Las funciones están caracterizadas por:

- Valor de retorno: puede ser de un tipo o Void; lo cual indica que la función no retorna nada.
- Un nombre: El cual identifica a la función dentro del programa.
- Una lista de parámetros (opcional): Son los valores que las funciones necesitan en orden para ser operativos.
- Directivas: Son comandos especiales que enriquecen a la función.
- Secuencia de instrucciones

Sintaxis:

Function Type Name([Parameter List)

[Directives]

[Local Vars]

Instructions

End

Las directivas indican:

- *#Macro:* El nombre de la función es insertado en el menú Macro de Run Time; para que la función pueda ser llamada por el usuario.
- *#Startup:* La función es ejecutado en el inicio de Run Time en segundo plano. Así es posible crear funciones cíclicas realizando revisiones o acciones determinadas por ciertas condiciones.
- *#Shutdown:* la función es ejecutada cuando se cierra Run Time
- *#Modal:* Run Time espera por que la ejecución de la función termine.

4.4.4.3 INSTRUCCIONES

Toda instrucción debe terminar con ";", las distintas estructuras e instrucciones del lenguaje son:

- Llamada a una función
  Sintaxis: *FunctionName ([Parameter Expressions]);*
- Asignación: Se usa cuando se le da otro valor a una variable.
  Sintaxis: *VarName = Expression;*
- Retornar Valor: Esta palabra reservada es usada cuando se necesita salir de una función a una función superior y retornar un valor.
  Sintaxis: *Return Expression;*
- Condición If then Else: Ejecuta una parte de código solamente si una condición es verdadera.
  Sintaxis: If (*Condition*) Then *Instructions*
  *[*Else *Instructions]*
  End

- For Next Cicle: Es un ciclo cuyas interacciones esta definidas, es necesario un entero como contador.
  Sintaxis: For *VarName* = *Expression* To *Expression* Do
  [Instructions]
  End
- While Cicle: Es un ciclo donde la condición es evaluada primero para ejecutar el código interno.
  Sintaxis: While (*Comparison*)
  [Instructions]
  End
- Do While Cicle: En este ciclo la condición es evaluada luego de ejecutar las instrucciones.
  Sintaxis: Do
  [Instructions]
  While (*Comparison*);

4.4.4.4 OPERADORES

Los operadores son parte del lenguaje que vincula diferentes expresiones que permiten la creación de expresiones más complejas. Existen operadores lógicos (And, Or,..) cuyo resultado es booleano y operadores algebraicos cuyo resultado es numérico.

OPERADORES LOGICOS

==	Igualdad
¡=	Desigualdad
>=	Mayor o igual
<=	Menor o igual
>	Mayor
<	Menor
&&	And
||	Or

OPERADORES ALGEBRAICOS

+	Suma
-	Resta
*	Multiplicación
/	División

4.5 REPORTES

Los reportes son representación de datos del proceso. Esta carpeta del proyecto contiene reportes. Para crear uno nuevo, solo se debe de dar clic derecho en la parte derecha luego de seleccionar la carpeta Reports. Para modificar uno solo es necesario dar doble click en el archivo en la parte derecha. Tal como se muestra en la figura 4.50.

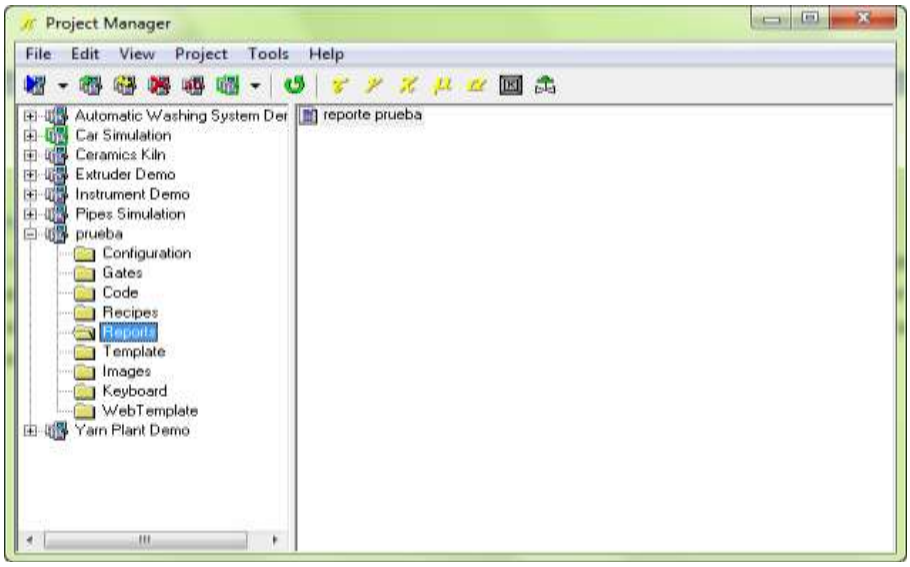


FIG.4.50. Carpeta de Reportes.

Primero, se debe escoger el tipo de archivo, se selecciona usando el cuadro mostrado en la figura 4.51.

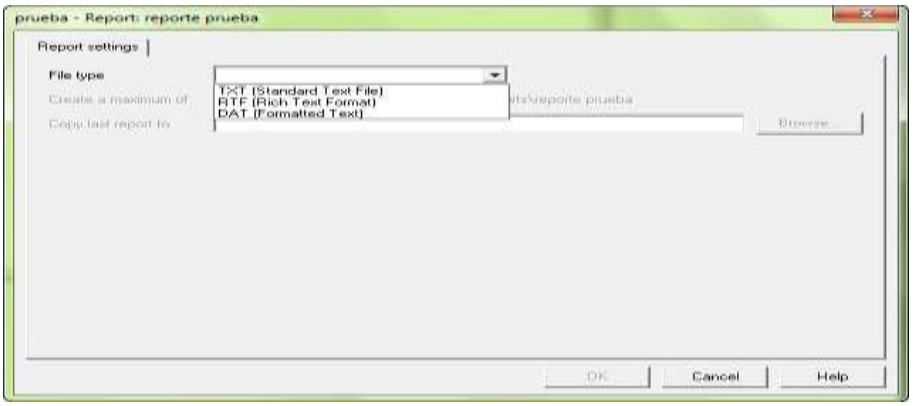


FIG.4.51. Configuración de reportes.

- TXT (Estándar Text File): Solo puede contener datos en forma simple.
- RTF (Ritch Text Format): Puede contener datos en texto con formato (con forma y color en texto) e imágenes. Además, puede ser convertido en formato PDF.
- DAT (Texto con formato): Contiene los datos de información en con formato compatible con CVS, cada informe es una lista de variables que se pueden seleccionar de la lista de variables disponibles.

4.6 PROTOCOLOS DE COMUNICACIÓN

Winlog provee comunicación con la mayoría de dispositivos de automatización (PLC, Controladores, Indicadores, Contadores, manejadores de motores, etc.) gracias a un amplio catalogo de drivers y a una interfaz OPC cliente servidor.

OPC (OLE for Process Control), en el control de procesos, se ha necesitado un lenguaje común desde hace buen tiempo, este es OLE para Control de Procesos (OPC) que es uno de los más promisorios. OPC es un conjunto normalizado de interfaces, propiedades y métodos que se definen cómo componentes individuales del programa que pueden interactuar y compartir información.

Los protocolos disponibles en Winlog son:

- Allen Bradley DF1 FULL DUPLEX PROTOCOL PARA PLC3
- Allen Bradley DF1 FULL DUPLEX PROTOCOL PARA PLC5
- Allen Bradley DF1 FULL DUPLEX PROTOCOL PARA SCL5/MICROLOGIX
- AVEBUS
- DATASTREAM
- DECOM-CONTREX
- EUROTHERM BISYNCH ASCII
- EV2001
- GEFRAN-CEMCAL
- IDEC IZUMI FA
- KLOCKNER MOELLER SUCOM-A
- KLOCKNER MOELLER SUCOM-A PARA PS4
- MATSUSHITA MEWTOCOL –COM
- MITSHUBISHI FR-CU03
- MODBUS ASCII- MODBUS RTU
- ODBC CLIENT
- OMRON FINS
- OMRON FINS IN HOST LINK Protocol
- OMRON SYSMAC
- OPC CLIENT
- RED LION PAXI-1/8 DIN COUNTER/RATE METER
- SAIA P800
- SAIA S-BUS
- SIEMENS MPI(Prodave MPI Mini)
- SIEMENS MPI
- SIEMENS PLC-SIMATIC S5
- TCP/IP
- TUTONDO
- PROFIBUS MASTER DP
- PROFIBUS MPI E S7
- PROFIBUS PPI S7-200
- PPI S7-200 (PPI ADAPTER)

El protocolo utilizado y del cual se amplía la información en este manual es el PPI S7-200 para mayor información de los demás protocolos consultar la referencia bibliográfica del capitulo.

4.6.1 PROTOCOLO PPI S7-200

Este protocolo requiere el siguiente hardware:

- Siemens RS232-PPI Multi-Master Cable
  Configurado(los dip switch) como:
  - PPI/Freeport
  - Local/DCE
  - 11 bit

De esta forma y a través del cable multi-maestro es posible acceder a los siguientes campos del PLC:

- Entradas
- Salidas
- Entradas Análogas
- Salidas Análogas
- Banderas o Marcas
- Marcas Especiales
- Área V

Las compuertas numéricas pueden ser Entradas (I), Salidas (Q), Marcas (M), marcas especiales (SM), Entradas Análogas (AI) y Salidas Análogas (AQ). Para todos los tipos de datos, excepto los análogos, es posible especificar diferentes formatos de datos: Byte, Word, Double Word y Float.

Debe notarse que la dirección del dato se refiere a un byte. Es decir, tal como se ve en la tabla 4.3:

PLC MEMORY	DATA TYPE "B"	DATA TYPE "W"	DATA TYPE "D"	DATA TYPE "F"
V0	VB0	VW0	VD0	VF0
V1	VB1			
V2	VB2	VW2		
V3	VB3			
V4	VB4	VW4	VD4	VF4
V5	VB5			
V6	VB6	VW6		
V7	VB7			
V8	VB8	VW8	VD8	VF8
V9	VB9			
V10	VB10	VW10		
V11	VB11			
V12	VB12	VW12	VD12	VF12

TABLA 4.3. Dirección y Tipos de datos.

La tabla 4.4 muestra las direcciones de compuertas numéricas tales como las mencionadas anteriormente:

Description	Type	Format	Address	Gate read	Gate write	Block read	Block write
V BYTE	V	B	0...99999	Yes	Yes	Yes	Yes
V WORD	V	W	0...99999	Yes	Yes	Yes	Yes
V DOUBLE WORD	V	D	0...99999	Yes	Yes	Yes	Yes
V FLOAT	V	F	0...99999	Yes	Yes	Yes	Yes
INPUT BYTE	I	B	0...99999	Yes	No	Yes	No
INPUT WORD	I	W	0...99999	Yes	No	Yes	No
INPUT DOUBLE WORD	I	D	0...99999	Yes	No	Yes	No
INPUT FLOAT	I	F	0...99999	Yes	No	Yes	No
OUTPUT BYTE	Q	B	0...99999	Yes	Yes	Yes	Yes
OUTPUT WORD	Q	W	0...99999	Yes	Yes	Yes	Yes
OUTPUT DOUBLE WORD	Q	D	0...99999	Yes	Yes	Yes	Yes
OUTPUT FLOAT	Q	F	0...99999	Yes	Yes	Yes	Yes
MERKER BYTE	M	B	0...99999	Yes	Yes	Yes	Yes
MERKER WORD	M	W	0...99999	Yes	Yes	Yes	Yes
MERKER DOUBLE WORD	M	D	0...99999	Yes	Yes	Yes	Yes
MERKER FLOAT	M	F	0...99999	Yes	Yes	Yes	Yes
SPECIAL MERKER BYTE	SM	B	0...99999	Yes	Yes	Yes	Yes
SPECIAL MERKER WORD	SM	W	0...99999	Yes	Yes	Yes	Yes
SPECIAL MERKER DOUBLE WORD	SM	D	0...99999	Yes	Yes	Yes	Yes
SPECIAL MERKER FLOAT	SM	F	0...99999	Yes	Yes	Yes	Yes
ANALOG INPUT WORD	AI	W	0...99999	Yes	Yes	Yes	Yes
ANALOG OUTPUT WORD	AQ	W	0...99999	Yes	Yes	Yes	Yes

TABLA 4.4. Direcciones de compuertas numéricas

Para aumentar la velocidad de comunicación entre la PC y el PLC, es recomendado usar lectura en bloque.

Los bloques de compuertas deben estar compuestos de compuertas con el mismo tipo pero pueden ser compuertas con distinto formato y tener una dirección consecutiva en orden creciente (en relación al formato de datos).

El tamaño del bloque está relacionado al formato de datos de las compuertas agrupadas en el bloque. El máximo tamaño del bloque dependiendo del formato se puede observar en la tabla 4.5.

Data format	Maximum block lenght
BYTE	200
WORD	100
LONG	50
FLOAT	50

TABLA 4.5. Máximo tamaño del bloque.

Por ejemplo, en la tabla 4.6 se pueden observar las compuertas que pueden ser agrupadas en bloque y las que no tienen como referencia las entradas y área V del PLC.

Numeric gates that can be grouped on block:	Numeric gates that can be grouped on block:	Numeric gates that CAN'T be grouped on block:	Numeric gates that CAN'T be grouped on block:
IB0	VW0	IB0	VW0
IB1	VW2	IB3	VW3
IB2	VD4	IB5	VW5
IB3	VD8	IW4	VW8
IW4	VF12	IW8	VW7
IB6	VB16	IB10	VD8
IB7	VB17	IB12	QB10

TABLA 4.6. Ejemplo de bloques compuertas numéricas.

Las compuertas digitales pueden estar relacionadas a entradas (I), salidas (Q), área V (V), marcas (M) y marcas especiales(SM).

Las direcciones deben estar alineadas solamente como bytes. La posición del bit dentro del byte debe ser especificada. Por ejemplo*: IB13.7 se refiere al Byte de entrada 13 y al bit 7 del mismo*.

Para aumentar la velocidad de comunicación entre la PC y el PLC, es recomendado usar lectura en bloque.

Los bloques de compuertas deben estar compuestos de compuertas con el mismo tipo y formato y tener una dirección consecutiva en orden creciente (en relación al formato de datos). No es necesario que el bit dentro del byte sea en forma consecutiva.

La tabla 4.7 muestra un ejemplo de lectura en bloque valida y no valida.

Digital gates that can be grouped in block	Digital gates that can be grouped in block	Digital gates that CAN'T be grouped in block	Digital gates that CAN'T be grouped in block
IB0.0	VB0.3	IB0.1	VB0.0
IB0.1	VB0.4	IB3.3	VB3.0
IB0.5	VB0.5	IB5.2	VB5.0
IB1.1	VB1.3	IB5.3	VB8.0
IB1.2	VB1.4	IB5.4	VB7.4
IB1.7	VB1.6	IB5.5	VB8.3
IB2.0	VB1.7	IB6.0	VB8.4

TABLA 4.6. Ejemplo de bloques compuertas digitales.

El protocolo se debe configurar en Winlog, a través del Project Manager en la sección de canales tal como se explica en la sección 4.2.2.2, los parámetros se deben establecer cómo se muestra en la figura 4.52.

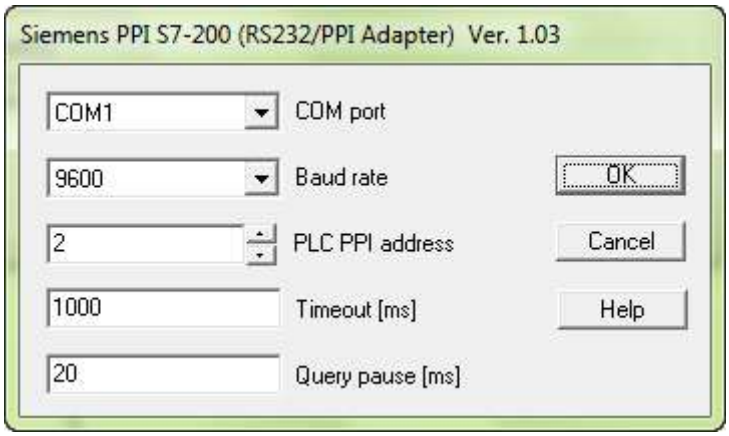


FIG. 4.52. Configuración del protocolo PPI s7-200.

Donde:

- Port: Puerto Serie de la PC (COM) a utilizar en la comunicación con el PLC.
- BaudRate: Velocidad de la comunicación serial.
- PLC PPI Address: Dirección de PLC.
- TimeOut: Tiempo antes de contestar un mensaje.
- QueryPause: Tiempo de espera entre 2 mensajes.

4.7 PLANTILLAS (TEMPLATE BUILDER)

4.7.1 GENERALIDADES

Template Builder es el que permite crear las páginas de supervisión o plantillas, donde se puede usar todos los componentes disponibles en el software (botones, etiquetas, barra de estado, etc.). Cada uno de ellos es configurable de distintas maneras, para poder ajustarlas de la mejor manera a la aplicación en construcción.

El software se puede iniciar de dos maneras, a través del menú de herramientas de Project Manager o a través de una plantilla ya diseñada, ubicada en la carpeta *Template* de cada proyecto. Al iniciar desde la barra de herramientas de Project Manager se inicia una plantilla en blanco tal como se puede observar en la figura 4.53.

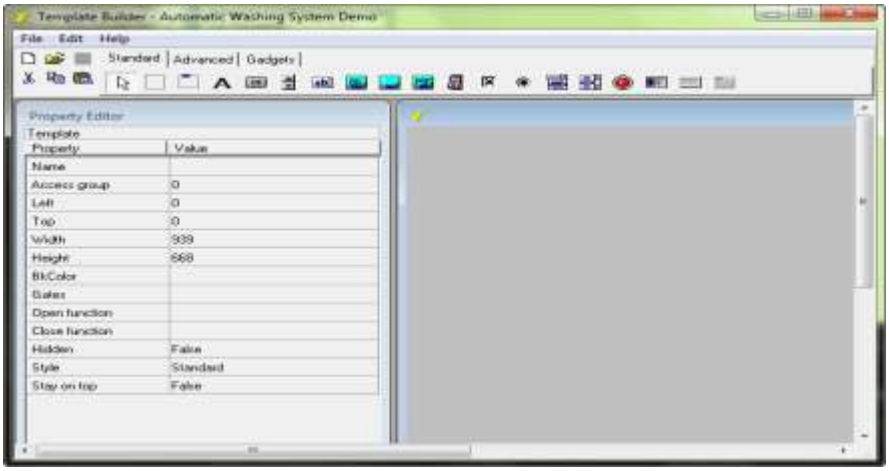


FIG. 4.53. Template Builder.

En la pantalla inicial de Template Builder se puede observar tres secciones principales: en la parte superior la barra de herramientas, en la parte izquierda el editor de propiedades y en la parte derecha la plantilla en blanco.

La barra de herramientas, mostrada en la figura 4.54, está dividida en 2 partes, la primera de la parte izquierda contiene los comandos para crear una nueva plantilla, abrir una existente, guardar la plantilla y el uso del portapapeles, estos comandos también se pueden acceder a través del menú File. La segunda parte de la derecha contiene los objetos que se pueden ubicar en la plantilla.

FIG. 4.54. Barra de herramientas

Los objetos están divididos en tres secciones: Estándar, Avanzados y Gadget. En la sección estándar se encuentran los objetos más utilizados entre ellos botones, etiquetas, cuadros,

barras deslizantes, etc. En la sección avanzada se encuentran objetos más complejos como gráficos, vista de operador, de alarmas, etc. En la sección gadget se encuentran barras deslizantes horizontales, verticales, dial, etc.

Para ubicar un objeto en la plantilla primero se debe dar clic en el botón del objeto y luego un clic en la posición a ubicarlo en la plantilla, al ubicarlo se marco automáticamente el botón de puntero con el cual es posible modificar la posición del objeto y al mismo tiempo modificar sus propiedades a través del editor de propiedades de la parte izquierda de la pantalla.

4.7.2 VENTANA DE PLANTILLAS

La ventana de plantillas muestra el borrado de la página de supervisión del proyecto, una vez que ha sido mostrada por el software.

En esta ventana van ubicados los objetos que conforman la plantilla, simplemente ubicándolos en ella. Al ubicarlos en la ventana, Template Builder ubica en los 8 cuadros alrededor del objeto, y en el editor de propiedades se muestran todas las propiedades modificables del objeto. Las dimensiones y ubicación del objeto se pueden modificar directamente en esta ventana sin la necesidad de usar el editor de propiedades.

Para mover un objeto, simplemente se le da un clic y se arrastra hacia su nueva ubicación, es de notar, que para mantener la jerarquía entre los objetos (ver en la sección 4.7.4 la descripción de objetos que pueden incluirse en otros objetos), un objeto hijo no puede ser movido fuera de los bordes del objeto padre.

Para modificar el tamaño de un objeto, simplemente se le da un clic y se utilizan los cuadros alrededor del objeto para modificar su tamaño, las filas correspondientes al tamaño en el editor de propiedades se actualizan automáticamente. Para mantener la proporcionalidad entre los objetos, el cambio de tamaño solo se puede realizar en múltiplos de cinco.

Existe una herramienta llamada Multiplantilla la cual permite crear plantillas a través de otras plantillas como fuente de objetos.

Para utilizarla se debe hacer a través del menú *File-New-Multitemplate*, luego de seleccionarlo aparecerá la ventana mostrada en la figura 4.55, desde donde a través de las casillas de selección se decide que plantillas que existen en el proyecto se utilizara para generar la plantilla de salida.

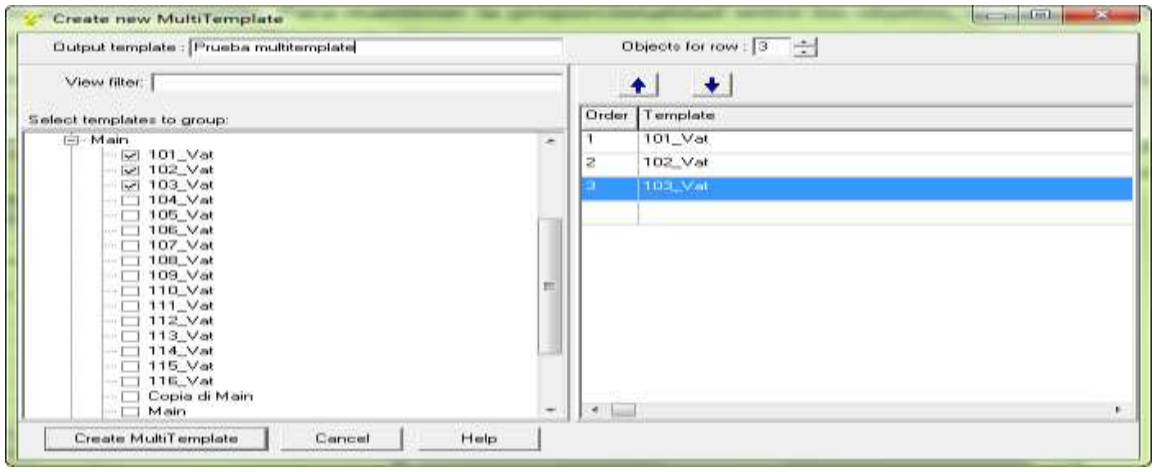


FIG. 4.55. MultiPlantillas

4.7.3 EDITOR DE PROPIEDADES

Cada objeto de la plantillas tiene distintas propiedades. Estas propiedades pueden ser modificadas según el gusto u objetivos del mismo. Al hacer clic en un objeto sus propiedades se ubican en el editor, donde para modificarlas se debe hacer clic en la fila de la propiedad a modificar.

Básicamente existen 3 tipos de filas en el editor de propiedades, las filas editables, las filas de selección múltiple y las filas con lista desplegable. La figura 4.56 muestra los tipos de filas antes mencionados.

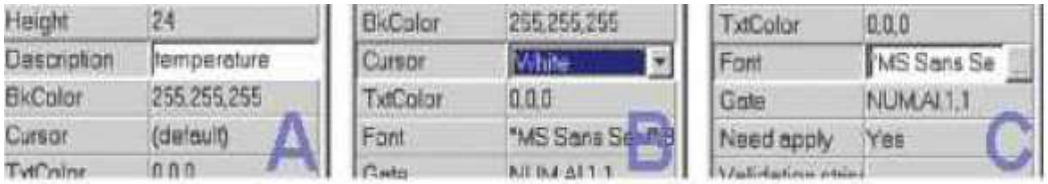


FIG. 4.56. Tipo de Filas.

4.7.3.1 DESCRIPCION DE OBJETOS

BITMAPS

A diferencia de los BKbitmaps, que representan imágenes estáticas, los Bitmaps permite tener en la plantilla imágenes que cambian durante la etapa de supervisión, para mostrar gráficamente alguna condición presente en el sistema. Con cada bitmaps se pueden asociar distintas imágenes cada uno de ellas con su respectiva condición, así durante la etapa de supervisión si una condición se cumple la imagen respectiva se muestra.

Las propiedades modificables en el editor de propiedades se muestran en la imagen 4.57, junto con un Bitmap.

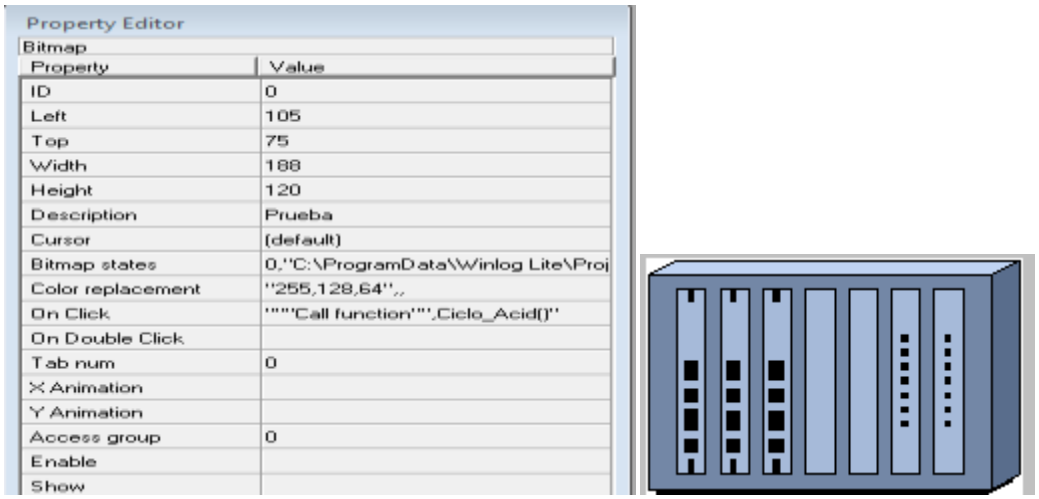


FIG. 4.57. Propiedades Objeto Bitmap.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.
- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Description: Descripción del objeto.
- Cursor: Indica la forma del cursor al ubicarse sobre el objeto durante la etapa de supervisión.
- Bitmap States: Este indica las imágenes y estados del bitmap. Al hacer clic en la fila y en el botón aparecerá la figura 4.49, donde se pueden agregar, editar o borrar estados y configurar la condición que indica la activación de cada estado.

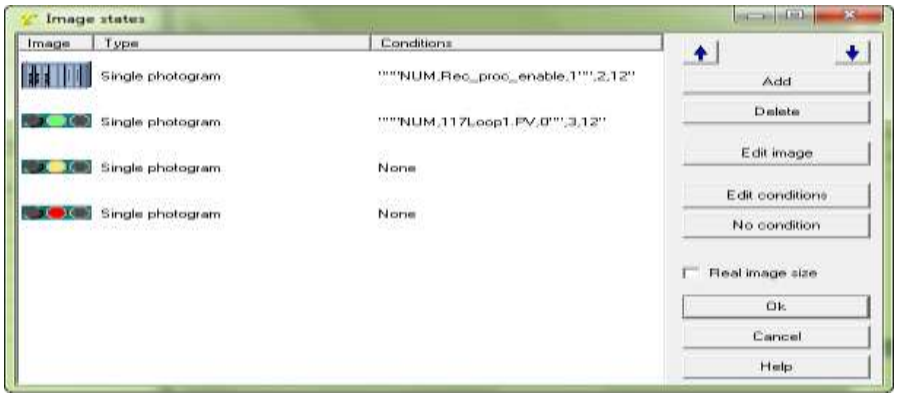


FIG 4.58. Editor de Estados

Además es posible indicar para cada estado una sola imagen o una serie de imágenes para crear una animación, a la cual se le puede configurar el tiempo de cambio de imágenes, tal como se muestra en la imagen 4.59. Además, en la misma figura se puede observar el acceso a la librería de Symbol Factory desde donde se puede escoger más imágenes que las disponibles en Winlog.

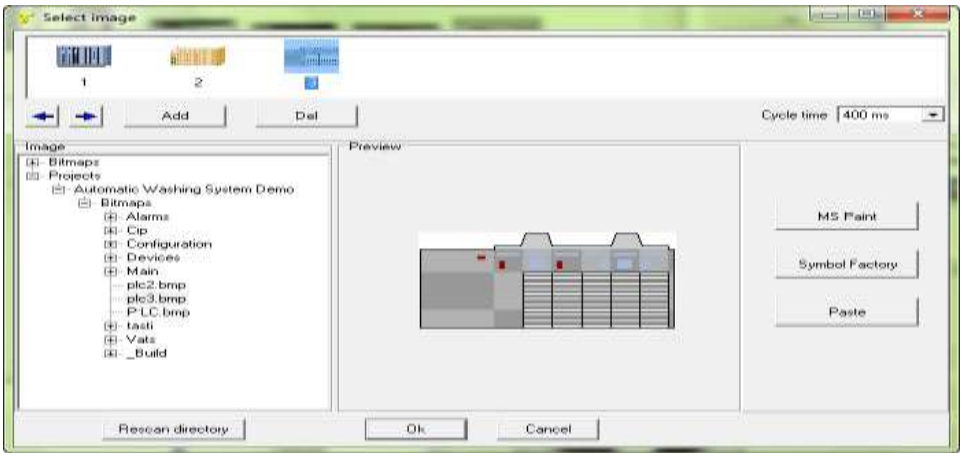


FIG 4.59. Selección de imágenes

Luego para la edición de condiciones se debe presionar en el botón edit conditions, donde se mostrarla la ventana en la figura 4.60. Desde donde es posible agregar, o editar condiciones, al agregar una nueva o editar una existente se mostrara la imagen de la figura 4.61.

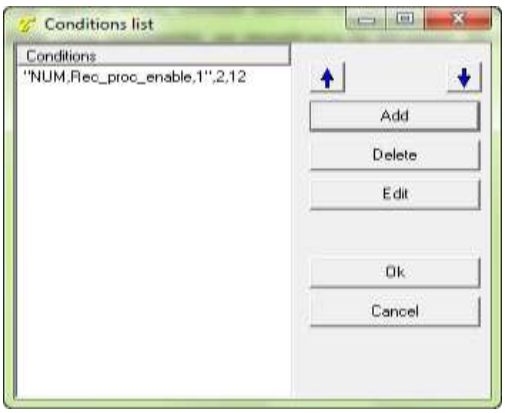


FIG 4.60. Editor de condiciones.

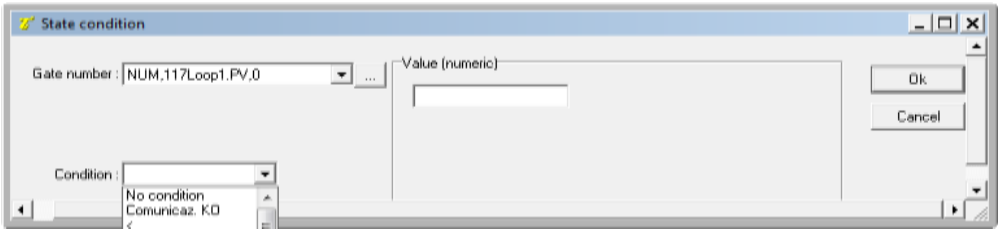


FIG 4.61. Editor de condiciones.

Primero se debe escoger la compuerta de la plantilla donde se verificara la condición, Luego se deberá escoger la condición a verificar de la lista disponible y el valor con el cual se debe hacer la comparación. La lista de condiciones depende del tipo de compuerta escogida previamente y el valor de comparación depende

de ambos. Una lista de condiciones dependiendo del tipo de compuerta se muestra en la tabla 4.7.

Condition	Description	Numeric gates	Digital gates	String gates	Composed gates	Event gates
Never	Condition never active	3	3	3	3	3
Comm. KO	Condition active only if the communication is broken	3	3	3	3	3
<	Condition active if the value of the gate is less than the value	3		3	3	
>	Condition active if the value of the gate is greater than the value	3		3	3	
<=	Condition active if the value of the gate is less than or equal the value	3		3	3	
>=	Condition active if the value of the gate is greater than or equal the value	3		3	3	
==	Condition active if the value of the gate is equal to the value	3	3	3	3	3
!=	Condition active if the value of the gate is different to the value	3	3	3	3	3
&&	Condition active if the logical AND between the value of the gate and the value is different to zero	3			3	
&	Condition active if the bitwise AND between the value of the gate and the value is different to zero	3	3		3	3
||	Condition active if the logical OR between the value of the gate and the value is different to zero	3			3	
|	Condition active if the bitwise OR between the value of the gate and the value is different to zero	3	3		3	3
^	Condition active if the bitwise XOR between the value of the gate and the value is different to zero	3	3		3	3
*	Condition active if the value of the gate corresponds to the mask (composed of 32 bits with value 0, 1 or x for the irrelevant bits)	3			3	

TABLA 4.7. Condiciones Disponibles.

- On clic : Esta indica la operación a realizar cuando el usuario presiona el objeto. En la figura 4.62 se muestra la ventana donde se puede seleccionar la operación. Entre las cuales tenemos:
  - Call Funtion: Presionar para llamar a la función indicada.
  - Stop Funtion: Presionar para detener la función indicada si esta activa.
  - Open Template: Presionar para abrir la plantilla indicada.
  - Close Template: Presionar para cerrar la plantilla indicada.
  - Apply Changes: Presionar el botón para aplicar los cambios a todos los componentes del mismo nivel o en nivel superior que necesiten confirmación por este.

- Undo Changes: Presionar el botón para deshacer los cambios a todos los componentes del mismo nivel o en nivel superior que necesiten confirmación por este.

FIG 4.62. Operaciones con un clic.

- On Double Clic: indica el nombre de la plantilla a abrir con un doble clic del ratón.
- X/Y Animation: Selecciona la compuerta a leer para la posición X/Y del bitmap. A través de esta propiedad es posible mover el bitmap a través de la plantilla.
- Show: El objeto se muestra si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre es visible.

BKBITMAPS

Los BkBitmaps son bitmaps de fondo, los cuales representan imágenes que pueden insertarse en una plantilla para mostrar gráficos estáticos.

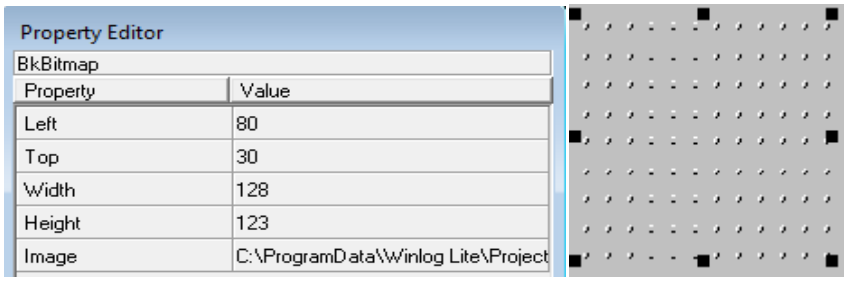


FIG. 4.63. Propiedades Objeto BkBitmaps.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Image: Selecciona la imagen a mostrar en el BkBitmaps.

BUTTON

La función principal del botón es realizar una operación (durante la fase de supervisión) cuando es presionado por el usuario.

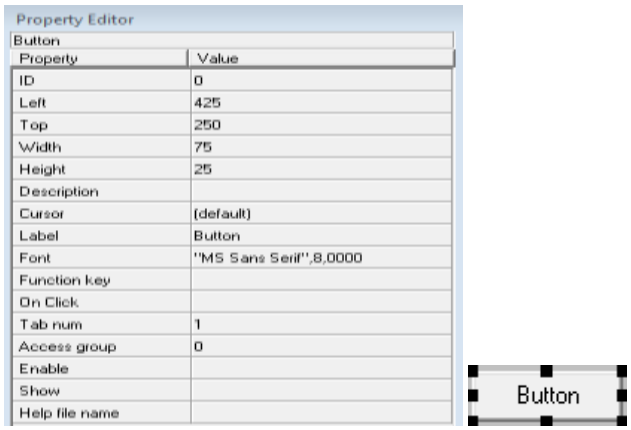


FIG. 4.64. Propiedades Objeto Button.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.
- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Description: Descripción del objeto.
- Cursor: Indica la forma del cursor al ubicarse sobre el objeto durante la etapa de supervisión.
- Label: Texto que el botón mostrara durante la fase de supervisión.
- Funtion Key: Indica el vínculo del botón con alguna tecla del teclado.
- On clic: Esta indica la operación a realizar cuando el usuario presiona el objeto. En la figura 4.53 se muestra la ventana donde se puede seleccionar la operación. Entre las cuales tenemos:
    - Call Funtion: Presionar para llamar a la función indicada.
    - Stop Funtion: Presionar para detener la función indicada si esta activa.
    - Open Template: Presionar para abrir la plantilla indicada.
    - Close Template: Presionar para cerrar la plantilla indicada.
    - Apply Changes: Presionar el botón para aplicar los cambios a todos los componentes del mismo nivel o en nivel superior que necesiten confirmación por este.
    - Undo Changes: Presionar el botón para deshacer los cambios a todos los componentes del mismo nivel o en nivel superior que necesiten confirmación por este.
- Access group: Indica los usuarios permitidos para modificar o ejecutar la operación del botón.
- Show: El objeto se muestra si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre es visible.
- Enable: El objeto está habilitado si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre está habilitado.

CHART

El objeto Chart permite mostrar una tabla de un grupo de compuertas. Es posible especificar el grupo de compuertas a mostrar y el tiempo que la tabla se considerara.

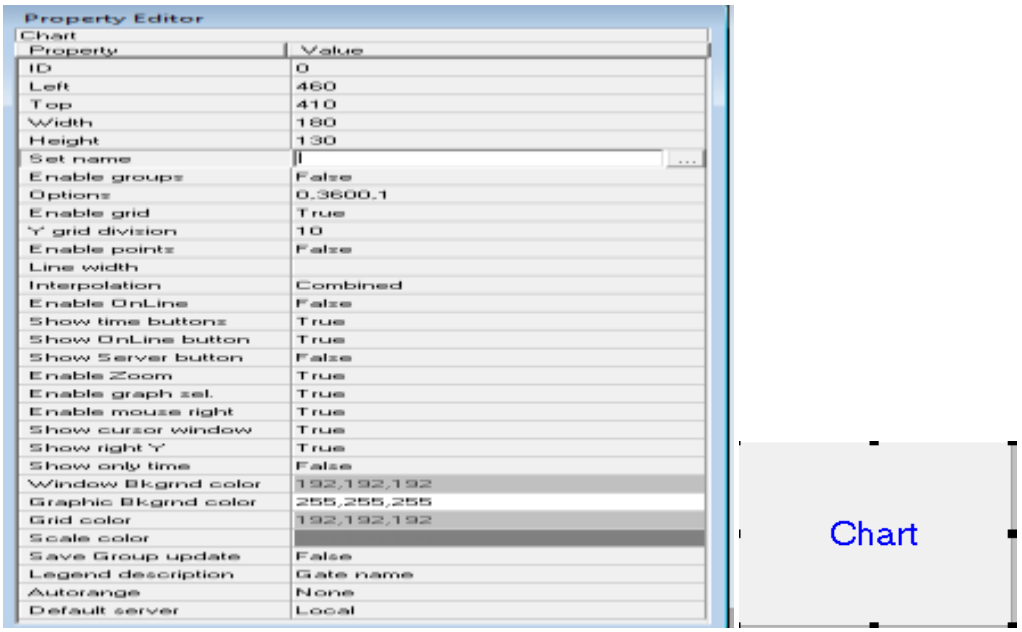


FIG. 4.65. Propiedades Objeto Chart.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.
- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Set Name: indica el grupo de compuertas a mostrar en la tabla.
- Enable Groups: Indica si es permitido modificar al usuario las compuertas a mostrar durante la etapa de supervisión.
- Option: Permite especificar el estilo y el rango de tiempo de la tabla. Ya sea normal, o a través de funciones como ChartSetTimeRange.
- Interpolation: Indica la interpolación a usar para dibujar la grafica.

EDIT

Permite al usuario modificar el valor de una compuerta en el modo de supervisión. Se puede editar para que solo se puedan ingresar ciertos valores.

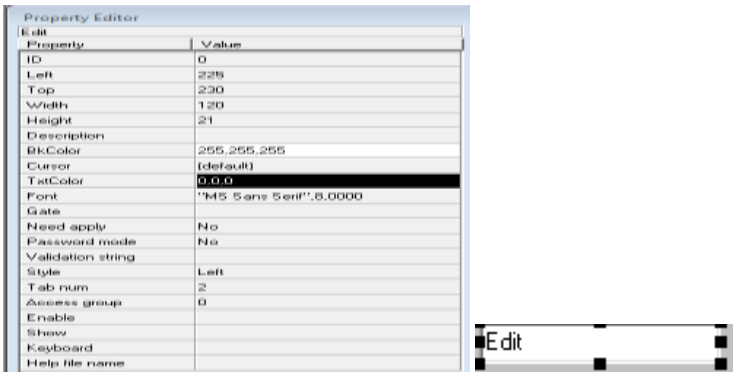


FIG. 4.66. Propiedades Objeto Edit.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.
- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Description: Descripción del objeto.
- Cursor: Indica la forma del cursor al ubicarse sobre el objeto durante la etapa de supervisión.
- Gate: Indica la compuerta a la cual escribir el valor ingresado en el editor.
- Need Apply: indica cuando el valor escrito debe ser aplicado a la compuerta. Hay dos opciones si es NO el valor es aplicado luego que el usuario presiona TAB o RETURN. Si la opción es SI el valor es escrito cuando el usuario da la confirmación desde afuera.
- Validation String: Permite definir una cadena de caracteres que el usuario deberá seguir para poder ingresar datos a la compuerta.
- Show: El objeto se muestra si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre es visible.
- Enable: El objeto está habilitado si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre está habilitado.

FRAME

Un cuadro es usualmente utilizado para contener otros objetos, por lo cual este objeto puede contener objetos hijos. Para insertar un hijo se debe insertar primero el cuadro y luego seleccionar el objeto hijo y darle clic en adentro del cuadro así se creara como hijo del objeto cuadro.

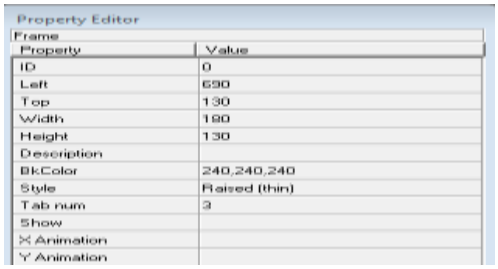


FIG. 4.67. Propiedades Objeto Frame.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.
- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Description: Descripción del objeto.
- Tab Num: Es un número que indica el orden que los objetos se vuelven activos en Run Time al presionar TAB.
- X/Y Animation: Selecciona la compuerta a leer para la posición X/Y del bitmap. A través de esta propiedad es posible mover el bitmap a través de la plantilla.
- Show: El objeto se muestra si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre es visible.

- Style: Indica el estilo del cuadro. Hay siete estilos disponibles los cuales se puede observar en la tabla 4.8.

Frame Style	Appearance
Plain	
Raised (thin)	
Recessed (thin)	
Embossed	
Grooved	
Raised (thick)	
Recessed (thick)	

TABLA 4.8. Estilos de Cuadros.

GAUGE

Es un componente que permite mostrar gráficamente el valor de una compuerta. De hecho, con un indicador está asociado una compuerta y un valor máximo y mínimo. Durante la fase de supervisión este mostrara el valor a través de una barra medidor entre los limites.

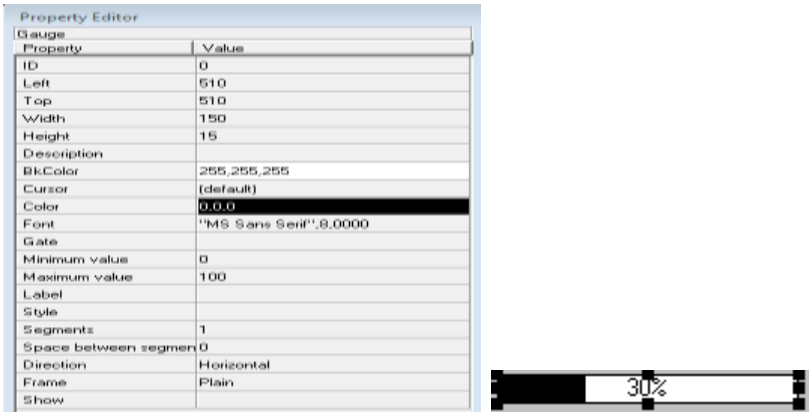


FIG. 4.68. Propiedades Objeto Gauge.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.

- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Description: Descripción del objeto.
- Cursor: Indica la forma del cursor al ubicarse sobre el objeto durante la etapa de supervisión.
- Gate: Indica la compuerta desde la cual leer el valor a mostrar en el indicador durante la fase de supervisión.
- Máximum Value: Valor máximo a mostrar en el indicador durante la fase de supervisión.
- Mínimum Value: Valor mínimo a mostrar en el indicador durante la fase de supervisión.
- Label: Texto que el objeto mostrara durante la fase de supervisión.
- Style: Indica el estilo del objeto.
- Show: El objeto se muestra si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre es visible.

GROUP BOX

El groupbox es una especie de cuadro, donde se aplica un nombre en la parte superior del mismo. Como el cuadro este también puede contener objetos.

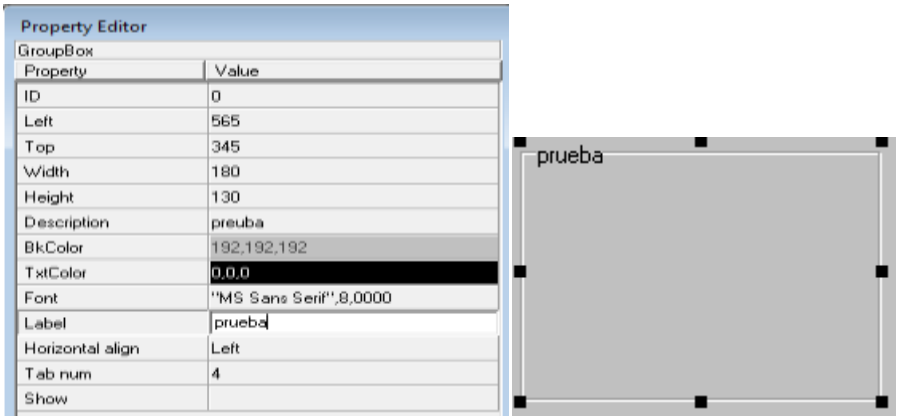


FIG. 4.69. Propiedades Objeto GroupBox.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.
- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Description: Descripción del objeto.
- Label: Texto que el objeto mostrara durante la fase de supervisión.
- Horizontal Align: Alineación que el texto mostrara durante Run Time, puede ser izquierda, centro o derecha.
- Tab Num: Es un número que indica el orden que los objetos se vuelven activos en Run Time al presionar TAB.

- X/Y Animation: Selecciona la compuerta a leer para la posición X/Y del bitmap. A través de esta propiedad es posible mover el bitmap a través de la plantilla.
- Show: El objeto se muestra si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre es visible.

HISTORICAL VIEW

Con este objeto se pueden mostrar los valores históricos de los record de los datos de cada compuerta.

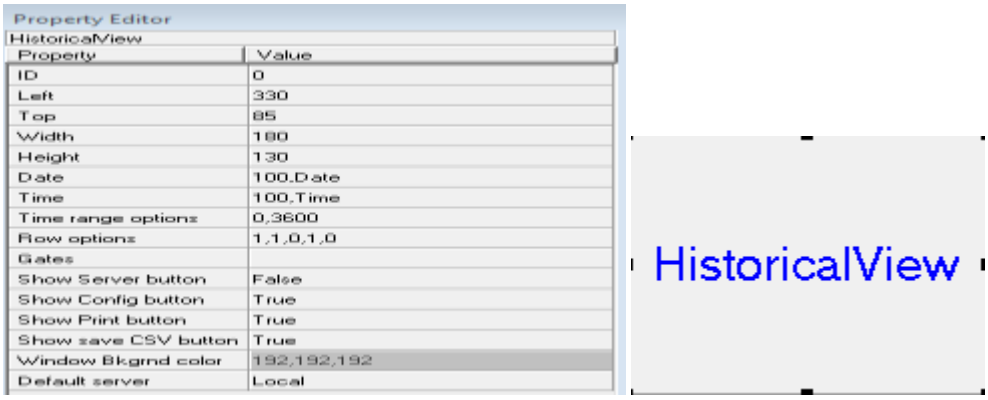


FIG. 4.70. Propiedades Objeto HistoricalView.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.
- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles
- Date: Indica el ancho y el texto de la columna Date.
- Time: Indica el ancho y el texto de la columna Time.
- Gates: Define la lista de compuertas que deben ser mostradas en el objetos. Pueden ser compuertas de tipo numérica, digital, cadena y compuesta.
- Show Config Button: habilita/deshabilita el botón config para establecer el rango de tiempo.
- Show Print Button: habilita/deshabilita el botón print.
- Show Save CSV button: habilita/deshabilita el botón Save para exportar los datos a un archivo de texto con formato.

LABEL

Las etiquetas son cuadros de texto que puede usarse para comunicar importante información al usuario. Por ejemplo, puede usarse para mostrar texto o para mostrar el valor de una compuerta.

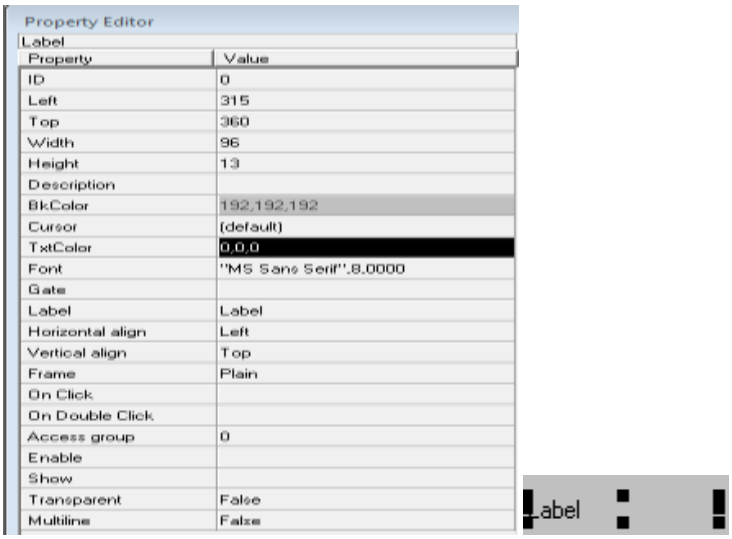


FIG. 4.71. Propiedades Objeto Label.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.
- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Description: Descripción del objeto.
- Cursor: Indica la forma del cursor al ubicarse sobre el objeto durante la etapa de supervisión.
- Gate: Indica la compuerta desde la cual leer el valor a mostrar en la etiqueta.
- Label: Texto que el objeto mostrara durante la fase de supervisión.
- Horizontal Align: Alineación que el texto mostrara durante Run Time, puede ser izquierda, centro o derecha.
- Tab Num: Es un número que indica el orden que los objetos se vuelven activos en Run Time al presionar TAB.
- Frame: Indica el estilo del cuadro. Hay siete estilos disponibles los cuales se puede observar en la tabla 4.8.
- On clic: Esta indica la operación a realizar cuando el usuario presiona el objeto. En la figura 4.62 se muestra la ventana donde se puede seleccionar la operación. Entre las cuales tenemos:
    - Call Funtion: Presionar para llamar a la función indicada.
    - Stop Funtion: Presionar para detener la función indicada si esta activa.
    - Open Template: Presionar para abrir la plantilla indicada.
    - Close Template: Presionar para cerrar la plantilla indicada.
    - Apply Changes: Presionar el botón para aplicar los cambios a todos los componentes del mismo nivel o en nivel superior que necesiten confirmación por este.
    - Undo Changes: Presionar el botón para deshacer los cambios a todos los componentes del mismo nivel o en nivel superior que necesiten confirmación por este.

- Show: El objeto se muestra si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre es visible.
- Enable: El objeto está habilitado si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre está habilitado.

LED

Es un objeto grafico que permite mostrar el estado de algunas compuertas. Con cada condición está vinculado un LED, si esta activa el Led está en On. Es posible indicar la imagen deseada o escoger de las disponibles.

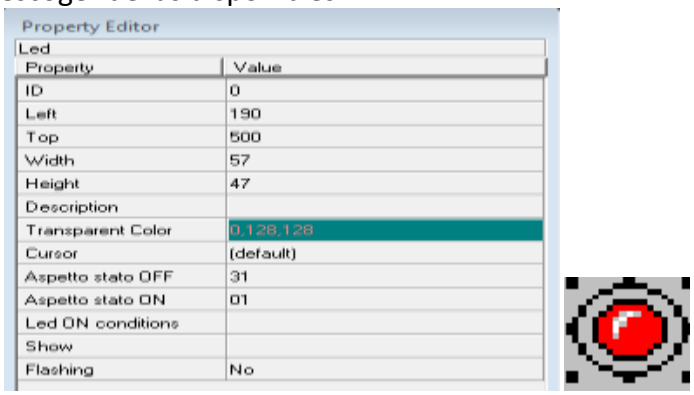


FIG. 4.72. Propiedades Objeto Led.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.
- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Description: Descripción del objeto.
- Cursor: Indica la forma del cursor al ubicarse sobre el objeto durante la etapa de supervisión.
- OFF Apareance: Indica la apariencia del Led cuando está apagado. Las imágenes disponibles se pueden observar en la imagen 4.73.
- ON Apareance: Indica la apariencia del Led cuando esta encendido.

Resource	Led	Resource	Led
01	(red)	02	(red)
11	(green)	12	(green)
21	(blue)	22	(blue)
31	(gray)	32	(gray)
41	(yellow)	42	(yellow)

FIG. 4.73. Imágenes LED disponibles.

- Led On Condition: Indica las condiciones que indican la activación de el Led. Es posible agregar nuevas, editar o remover condiciones como es usual.

- Show: El objeto se muestra si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre es visible.
- Flashing: Si esta activo el Led parpadeara cuando este activo.

SWITCH

Es un objeto que actúa como un interruptor. Por lo cual hay un estado apagado y un encendido. Durante la fase de supervisión el usuario puede activar o desactivar. El interruptor puede programarse para reaccionar al cambio de estado modificando el valor de una compuerta. Al igual Run Time vigila el valor de la compuerta para actualizar el estado del interruptor en caso de variación.

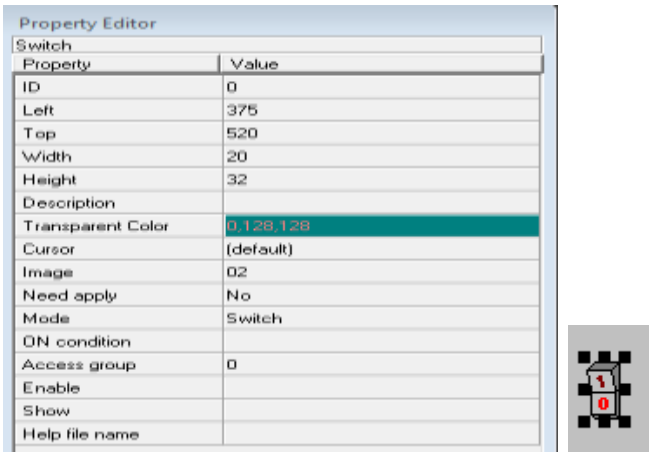


FIG. 4.74. Propiedades Objeto Switch.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.
- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Description: Descripción del objeto.
- Cursor: Indica la forma del cursor al ubicarse sobre el objeto durante la etapa de supervisión.
- Image: Indica la apariencia del interruptor. Pueden escogerse de las disponibles o de un archivo externo. Las imágenes disponibles se muestran en la figura 4.75.

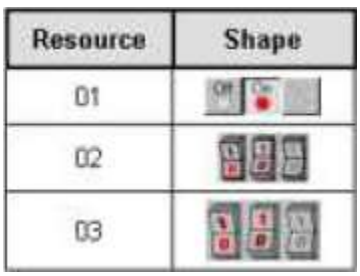


FIG. 4.75. Imágenes Switch disponibles.

- Need Apply: indica cuando el valor escrito debe ser aplicado a la compuerta. Hay dos opciones si es NO el valor es aplicado luego que el usuario presiona TAB o RETURN. Si la opción es SI el valor es escrito cuando el usuario da la confirmación desde afuera.
- Mode: Indica si es modo Switch entonces cambia con un clic. Si es modo Button entonces el objeto cambia solo mientras se mantenga presionado.
- ON condition: indica la compuerta a escribir cuando se dé el cambio en el objeto. La ventana de selección se muestra en la figura 4.76.

FIG. 4.76. Selección de Compuerta.

- Show: El objeto se muestra si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre es visible.
- Enable: El objeto está habilitado si la condición especificada en este campo es verdadera. Si no se especifica condición el objeto siempre está habilitado.

TAB SHEET

Con este objeto es más fácil organizar la plantilla ya que se puede dividir en varias hojas. Para cambiar de una página a otra solo es necesario darle clic tanto en la etapa de supervisión como en la plantilla.

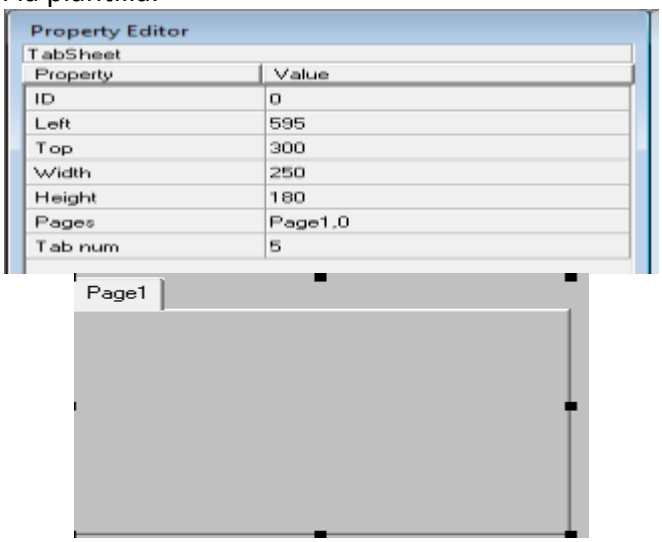


FIG. 4.77. Propiedades Objeto TabSheeet.

Entre las propiedades más importantes a modificar en este objeto se tienen:

- ID: Numero que identifica al objeto.

- Left, Top: Posición del objeto en base a la esquina superior izquierda del objeto.
- Widht, Height: Tamaño del objeto en pixeles.
- Pages: Indica la lista de páginas de la cual está formada el objeto. Aquí se puede agregar, modificar o eliminar paginas.
- Tab Num: Es un número que indica el orden que los objetos se vuelven activos en Run Time al presionar TAB.

Estos son solo algunos de los objetos disponibles en Winlog y son los de mayor uso en las plantillas, para consultar mayor información sobre otros objetos revisar la referencia bibliográfica 1.

4.8. EJECUCION (RUN TIME)

4.8.1 GENERALIDADES

El Run Time permite el control y supervisión de un proyecto. Para ejecutar un proyecto es posible hacerlo directamente desde el Run Time abriéndolo directamente o a través del Project Manager tal como se puede ver en la figura 4.78.

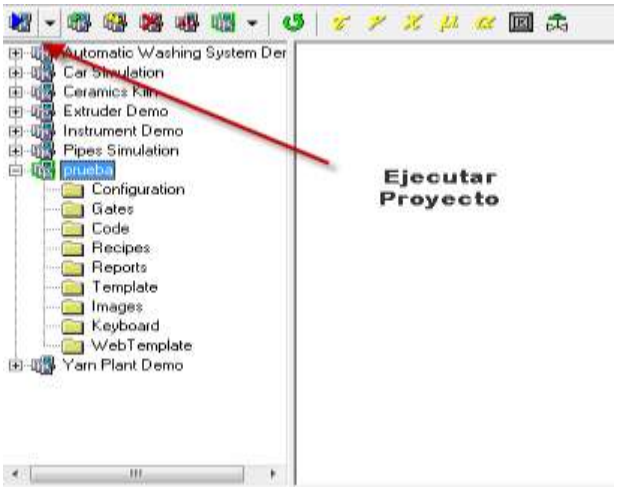


FIG. 4.78. Ejecutar Run Time

Al ejecutar un proyecto se abre la venta de supervisión con la plantilla y todas sus funciones que se le han habilitado. La ventana de Run Time y sus partes se pueden observar en la figura 4.79 mostrada a continuación.

El menú principal permite acceder a todas las partes de Run Time; existen dos formas de acceder a los menús, uno es a través de un clic del ratón o a través de la tecla Alt + la primera letra del menú.

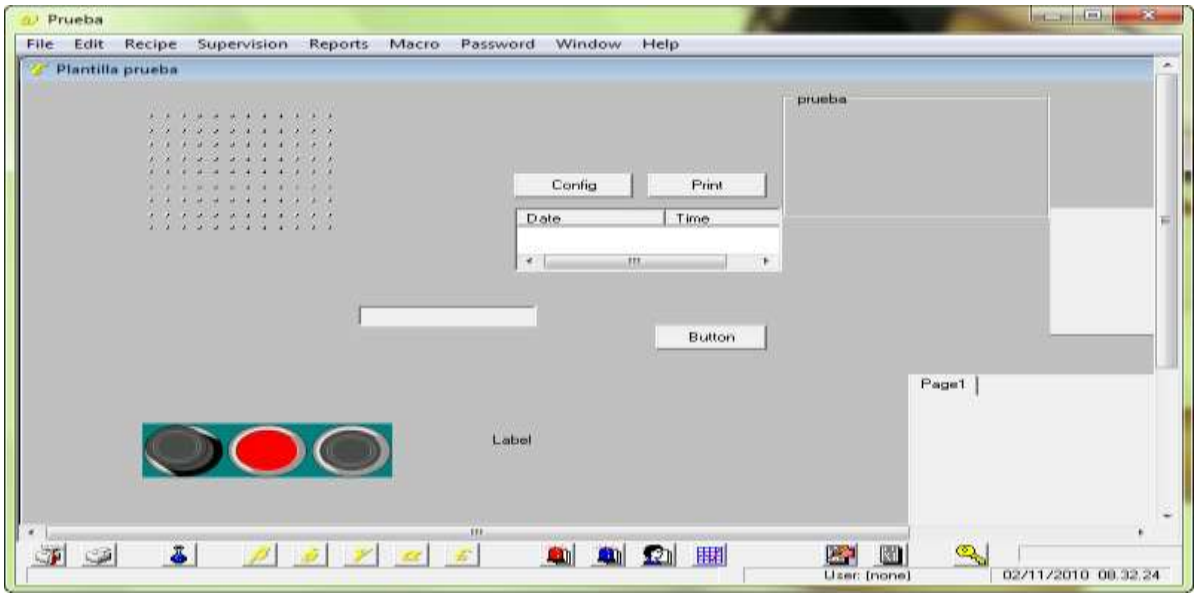


FIG. 4.79. Ventana principal Run Time

Los botones de uso rápido permiten acceder a la página deseada con solo un clic. El significado de cada uno de ellos es el siguiente:


Configuración de la impresora: permite especificar la impresora a usar.


Print: Imprime la pagina actual.


Recipe: Abre la pagina del manejador de medios o recetas.


System Status: Abre la página con el estado del sistema. La página se puede observar a continuación.

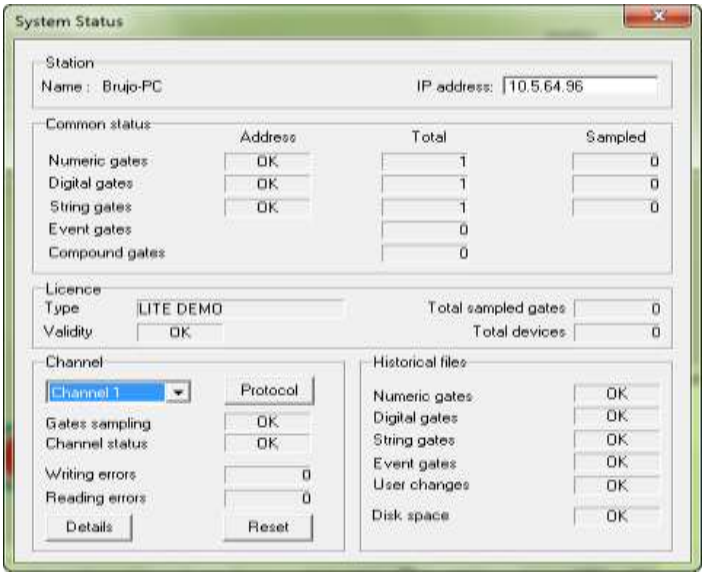


FIG. 4.80. Estado del Sistema.

Device status: Abre la página con el estado de dispositivo. La página se puede observar a continuación.

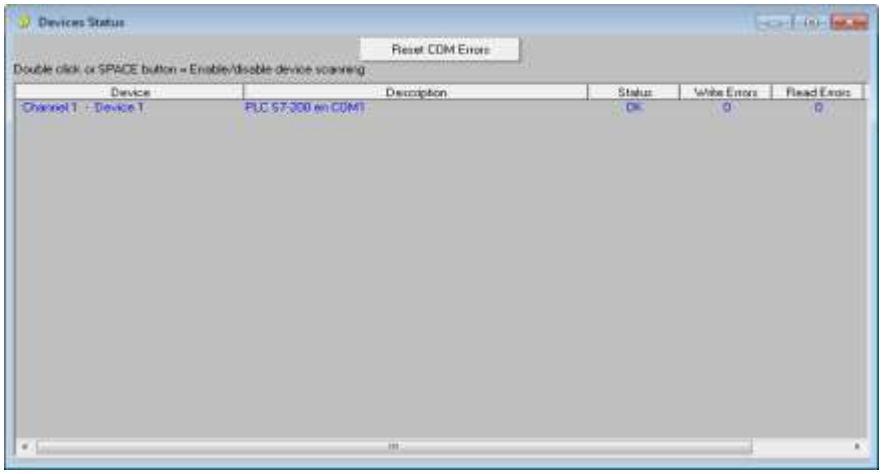


FIG. 4.81. Estado del Dispositivo.

Gate Status: Abre la página con el estado de las compuertas. La página se puede observar a continuación. A través de Gate Property es posible ver toda la descripción de la compuerta seleccionada de la lista.

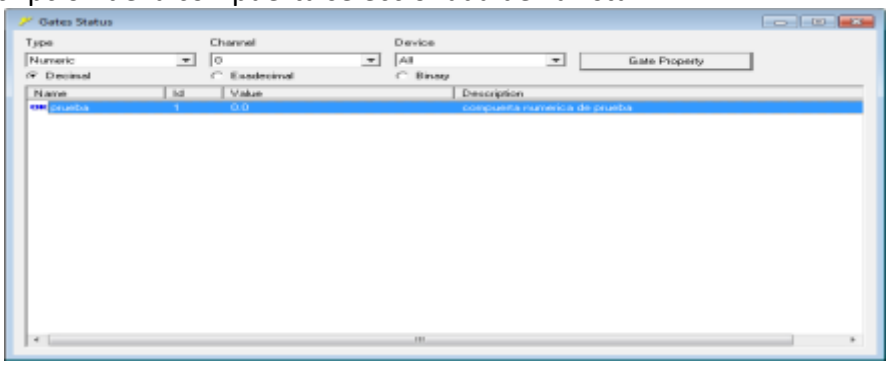


FIG. 4.82. Estado de Compuertas.

Alarms status: Abre la página con el estado de las alarmas. La página se puede observar en la figura siguiente.

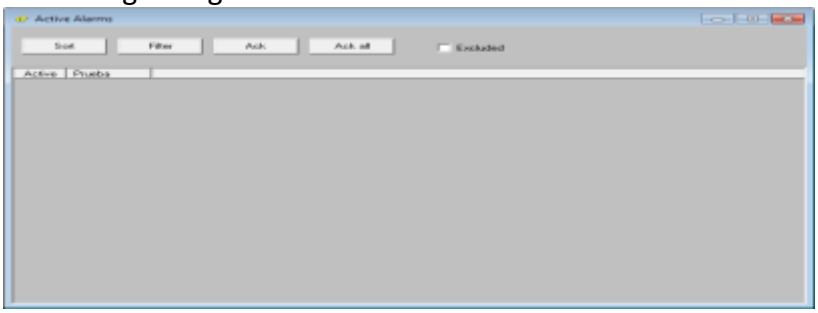


FIG. 4.83. Estado de Alarmas

Event Status: Abre la página con el estado de los eventos. La página se puede observar en la figura siguiente.

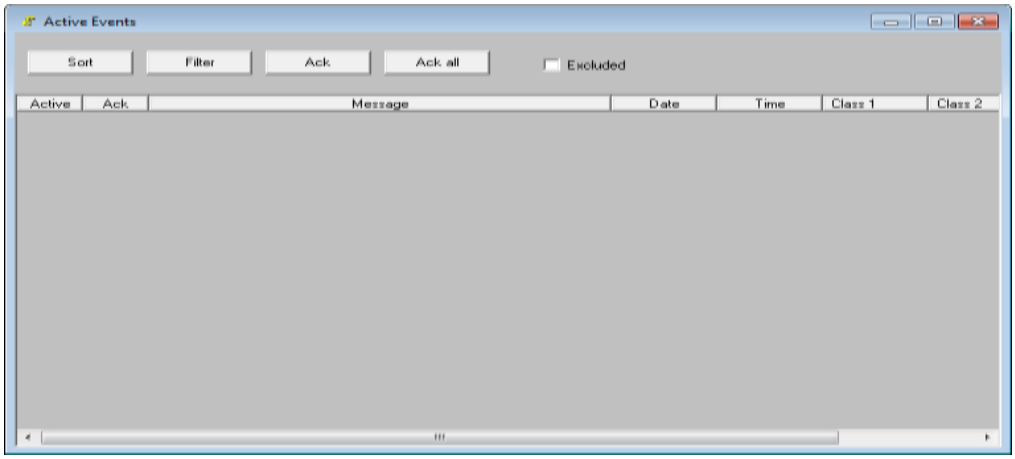


FIG. 4.84. Estado de Eventos.

Historical Alarms: Muestra la pagina con la historia de las alarmas.

Historical Events: Muestra la pagina con la historia de los eventos.

Users Changes: Muestra la pagina con las operaciones hechas por los usuarios. La página se puede observar en la figura siguiente.

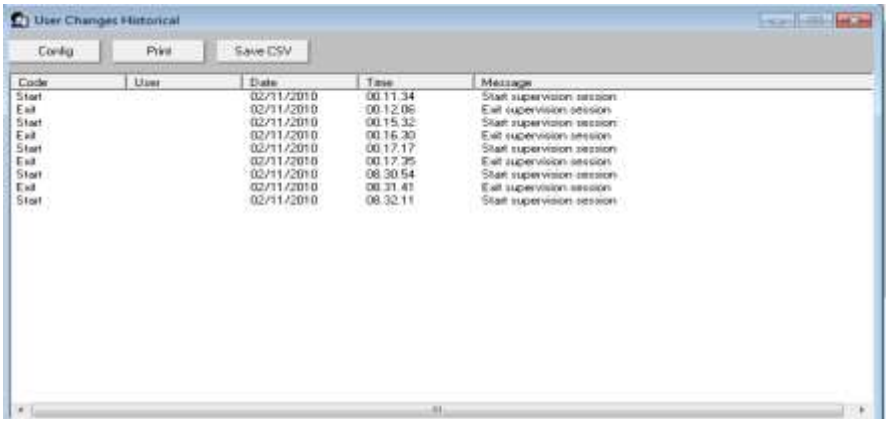


FIG. 4.85. Cambio hechos por usuarios.

Chart: Muestra la pagina para el manejo de las tablas y graficas. La página se puede observar en la figura siguiente.

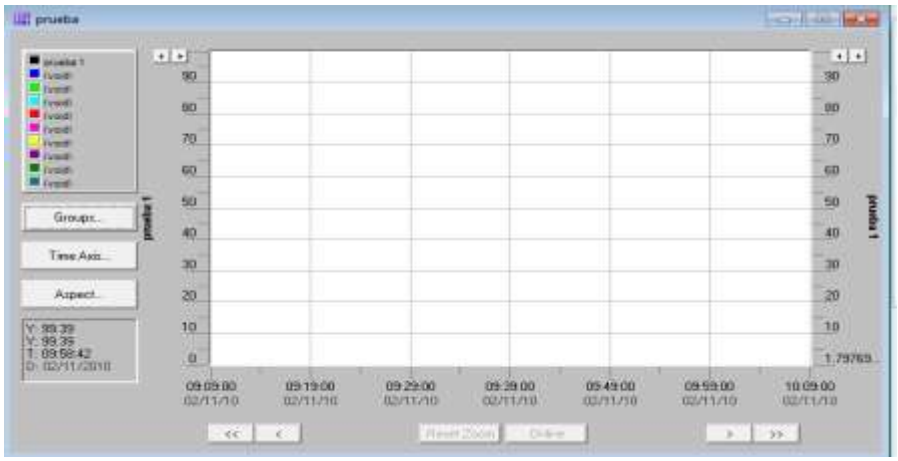


FIG. 4.86. Chart.

Make Report: Permite crear un reporte.

View Report: Permite abrir los reportes.

Password: Permite cambiar de usuario.

Un signo de exclamación rojo a la par de un icono indica que ha sucedido una anormalidad.

La barra de alarmas muestra el mensaje de la última alarma activada. Para eliminar estos mensajes se tiene que confirmar la activación o esperar la confirmación a través de las páginas de alarmas.

En la parte inferior derecha de la pantalla esta el tiempo y la fecha actual del sistema y el nombre del último operador que se ha registrado. Todas las operaciones realizadas aquí se registran con este nombre.

Run Time permite restringir el acceso y los cambios en las páginas solo a personal autorizado, gracias a su propio sistema de defensa. De hecho, es posible asignarles un usuario y contraseña a los operadores, y restringir los grupos a los cuales podrá acceder.

Para acceder un usuario ya definido se hace en Run Time a través de password, el cual se ubica en la esquina inferior derecha, donde se desplegara la ventana mostrada en la figura 4.87.

FIG. 4.87. Código de Acceso.

Para reiniciar la autorización introducida, solo se debe abrir la página de ingreso y sin ingresar nada se da clic en el botón OK.

Es posible definir un número ilimitado de usuarios, todos identificados por una contraseña y un grupo de acceso. Para crear un nuevo usuario se hace a través del *menú Password* en la pestaña *Define*, donde se escoge la opción Users. Al hacerlo aparecerá la ventana mostrada a continuación.

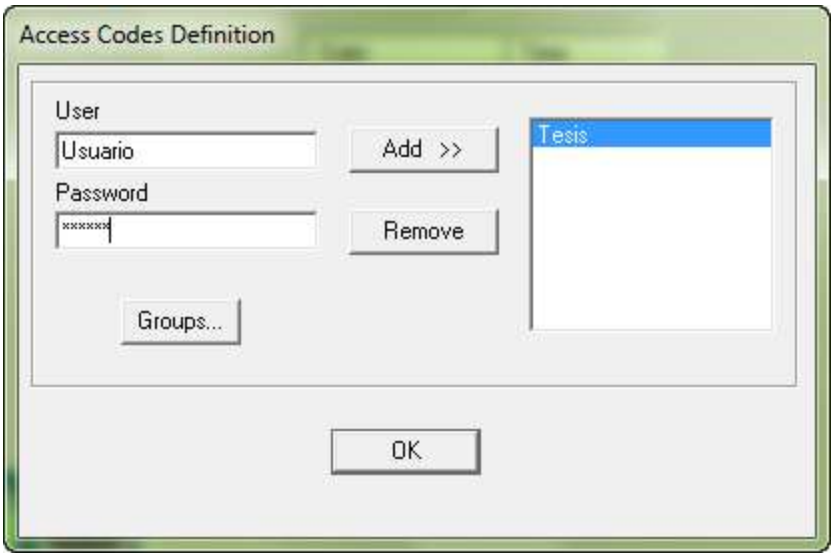


FIG. 4.88. Creación de usuarios.

Para definir grupos de accesos y establecerles sus propios nombres, es posible hacerlo a través del *menú Password* en la pestaña *Define,* donde se escoge la opción Group Names. Al hacerlo aparecer la ventana mostrada en la figura 4.89.

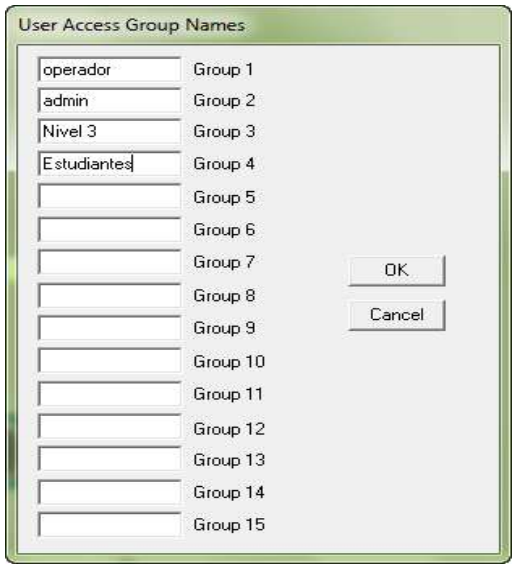


FIG. 4.89. Creación de grupos de usuarios.

Para definir los accesos a las páginas, y si un grupo podrá modificar ciertas características se hace a través del menú *Password* en la pestaña *Define,* se escoge la opción Pages Access, donde se desplegara la ventana mostrada en la figura 4.90. A través de las casillas de selección para cada página del Run Time se puede definir que grupos pueden acceder y que grupos pueden modificar los parámetros en las distintas páginas que conforman el Run Time y que ha sido explicada previamente.

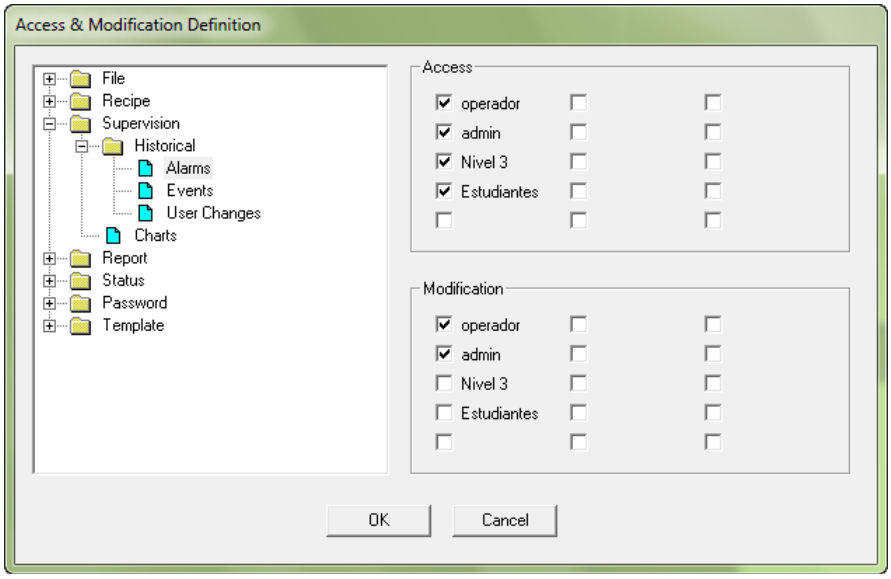


FIG. 4.90. Asignación de privilegios a los usuarios.

4.9 EJEMPLO SCADA: MINICENTRAL HIDROELECTRICA

A continuación se ha desarrollado una pequeña aplicación de un sistema SCADA utilizando el programa Winlog y el PLC S7-200 CPU 224XP, la comunicación se hace a través de un cable PC/PPI directamente entre la PC y el PLC.

Es importante mencionar que no se cuenta con la versión completa del programa, lo que limita más aun el diseño, dentro de las limitaciones más importantes están las siguientes:

LIMITACIÓN	DESCRIPCIÓN
Límite en el monitoreo de variables	Limita la posibilidad de desarrollar y ejecutar aplicaciones de hasta un máximo de 24 etiquetas.
Limite en la velocidad de muestreo	La velocidad de muestro está establecida a un mínimo de 1 segundo, sin la posibilidad de mejorar la velocidad de repuesta del sistema SCADA aumentando la velocidad de comunicación, que por defecto es de 9600 baudios.
Límite de tiempo de ejecución	Winlog Lite se puede ejecutar tanto en modo de demostración (sin necesidad de registro) y el modo completo, en modo de demostración, la comunicación con dispositivos externos y toma de muestras de variables externas se detiene automáticamente después de 15 minutos (si es necesario, se puede reiniciar manualmente). En modo completo la comunicación continúa sin límite de tiempo.

Reducida biblioteca de símbolos	Se dispone de una versión reducida de la biblioteca "Simbol factory 2.0" que cuenta con un máximo de 200 símbolos solamente. La versión completa cuenta con una biblioteca de más de 5,000 símbolos y con la posibilidad de seguirlos ampliando en posteriores actualizaciones.
Sin soporte WEB	No se cuenta con la posibilidad de soporte técnico vía WEB, la empresa no da soporte a individuos que no cuentan con una licencia, y las actualizaciones para ellos no están disponibles.

TABLA 4.8 Restricciones del uso de la versión lite incompleta de winlog.

4.9.1 DESCRIPCIÓN DEL PROCESO

El ejemplo consiste en el Monitoreo y control de una Mini central hidroeléctrica del tipo "pie de presa", la cual cuenta con un embalse de más o menos 3000 m^3 y una represa de 7 m de alto, la diferencia de alturas entre la compuerta de servicio y la turbina es de 43 m, la longitud de la tubería forzada es de 190 m, el caudal maximo sobre la tubería forzada es de más o menos 5 m^3/s, la potencia generada a plena carga es de 750 kW, el nivel de tensión del generador es 480V/240V en configuración estrella con neutro, cuenta con un transformador elevador de 480V a 23 KV que alimenta una carga a distancia completamente aislada. Los datos anteriores no son relevantes para nuestro objetivo pero se proporcionan para que se tenga una idea del tipo de proceso industrial que se desea controlar.

La central cuenta con:

- *Compuerta de servicio:* Se sitúa a la entrada de la tubería forzada y se encarga de interrumpir el flujo de agua cuando no se desea producir.
- *Sistema de válvulas de pre llenado de la tubería:* se encargan de llenar el tubo completamente de agua a un caudal más reducido de cómo se haría con la compuerta de servicio, evitando así los posibles daños de la turbina. El sistema cuenta con un sensor de tubería llena.
- *Sistema de lubricación:* Se compone de una bomba de levantamiento y una de lubricación que hacen que el cojinete y el eje del generador se mantengan casi libres de fricción. Cuando la presión adecuada se ha alcanzado, el sensor de presión localizado en el cojinete se activa.
- *Sistema de excitación de las bobinas de campo:* en base a un banco de baterías y un rectificador, este sistema alimenta con corriente directa las bobinas del generador para la producción de la energía eléctrica.
- *Servo mecanismo de apertura y cierre de alabes:* El control de potencia generada se realiza por medio de la apertura o cierre de un sistema de alabes que regulan el caudal de agua que llega a la turbina. La posición de los alabes se controla por medio de un mecanismo que envía un pulso de voltaje cada vez que los alabes incrementan su apertura el 1 % de su apertura total.
- *Sistema de sincronismo con la red:* Este sistema verifica el correcto acoplamiento de nuestro generador a la red, en base a tres parámetros nivel de tensión, frecuencia y ángulo de desfase de las señales de voltaje con respecto a la de la red.

En nuestro SCADA solo se controlan 2, el ángulo de desfase en base a la emulación de un sincronoscopio y el acople de voltajes; esto debido a las limitaciones del programa ya antes mencionadas.

- Interruptor de conexión al sistema: Es un interruptor de potencia que se encuentra en el lado de alta del transformador y da el paso de nuestro sistema a la red.
- *Sistema SCADA:* una PC de escritorio con el software Winlog instalado.
- *PLC:* S7-200 CPU 224 Xp

El control de cada uno de estos elementos puede hacerse de manera independiente pero en algunos casos deben seguir algunas reglas de validación; además, se cuenta con dos botones que inician una secuencia automática de encendido o apagado.

A continuación se describe el inicio automático. El automatismo empieza con el llenado de la tubería forzada, para este paso debe estar únicamente activa la válvula de pre llenado de la tubería, en el SCADA se muestra este proceso con la simulación de un movimiento constante y cíclico de la tubería forzada llenándose y la válvula parpadeando.

Cuando el agua ha llegado a su nivel, se apaga la válvula, esto se sabe porque el sensor de tubería llena se activará, el SCADA en esta etapa mostrará una imagen estática de la tubería llena. En este punto se empieza a abrir la compuerta de servicio, esta se detiene cuando el sensor de compuerta abierta este presionado, al instante que la compuerta se detiene empieza a trabajar la bomba de lubricación de los cojinetes, esta se apaga cuando el sensor de presión de los cojinetes se active, un led verde encenderá en el SCADA indicando la presión adecuada de los cojinetes.

Automáticamente los alabes de la turbina empezarán a abrir hasta un 50%, esto se muestra en el SCADA por un movimiento cíclico de los alabes, sin movimiento de la turbina, esta empezará a girar hasta que los alabes hayan alcanzado un 10% de su apertura y girarán más rápido cada vez que se les aumente un 30%. La apertura de los alabes se controla por una serie de pulsos a la entrada del PLC, cada pulso equivale a un 1% de apertura de los alabes, cuando los alabes hayan alcanzado la apertura especificada se visualizaran en el SCADA completamente abiertos y su apertura real se mostrará en un cuadro texto al pie de la imagen.

En este momento el sistema está listo para ser excitado, y efectivamente, este se conecta indicándose en el SCADA por el cambio de color de gris a verde de los elementos que componen este sistema (Rectificador, batería, generador); al monto de conectar la excitación se empieza el sincronizado del sistema, este paso es solo una simulación ya que no se cuenta con algún dispositivo externo que ejecute esta función; esto se realiza colocando en una variable un valor distinto de cero, este valor se muestra en el SCADA a través de una caratula tipo analógica de aguja que esta graduada de -100 a 100 e indica el grado de desfase de la seña de voltaje del generador con respecto a la de la red, tres botones permiten el valor del ajuste a cero y posteriormente la conexión.

El mismo proceso se puede realizar de forma manual realizando cada uno de los pasos antes descritos. Debido a que cualquier error se puede cometer en el proceso manual, se han declarado algunas reglas de validación y alarmas.

Algunas alarmas declaradas, son por ejemplo; cuando se quiere abrir la compuerta de bocatoma sin llenar antes la tubería, el sistema no lo permitirá y al mismo tiempo se mostrara una alarma que dice “no es permitido abrir la compuerta si antes llenar la tubería”, otra alarma se muestra cuando se desconecta la excitación mientras se está conectado a la red, también se muestra una alarma cuando la apertura de los alabes es muy pequeña para mantener girando la turbina, en nuestro caso para aperturas de los alabes menores al 10%.

En total son nueve alarmas que se han declarado en el sistema para mas detalles se puede ver la tabla 4.11 que muestra una lista completa de todas las alarmas declaradas en el sistema.

4.9.2 ESQUEMA GENERAL DEL PROCESO

La figura 4.91 muestra un diagrama de bloques general del automatismo, en este se pueden identificar claramente dos bloques esenciales, uno es la planta la cual está compuesto de todos los sensores y actuadores que ejecutan y se involucran en el proceso. Se cuenta con 8 actuadores y 8 sensores, dentro de los actuadores tenemos: la válvula de pre llenado, los activadores de los dos sentidos de giro del motor que abre y cierra la compuerta, la activación del sistema de lubricación, el interruptor que excita las bobinas de campo, los interruptores que activan los dos servomecanismos de posición de los alabes y el interruptor de conexión a la red.

Dentro de los sensores tenemos: el monitor de nivel máximo del embalse, los sensores de compuerta cerrada o abierta, el sensor de la tubería llena, el sensor de presión adecuada en los cojinetes, pulsos de entrada en la apertura o cierre de los alabes, monitor de voltaje y monitor de corriente.

En el S7-200 salen y entran las mismas señales, además se introducen dos señales de control, la de “inicio manual” y “paro general” las cuales son dos pulsadores con los que puede interactuar el usuario con un inicio o paro externo al sistema SCADA.

El esquema presentado es típico de pequeños sistemas SCADA en la cual los controladores automáticos en este caso el S7-200, funciona en lazo cerrado, lo que quiere decir que para cada combinación de las salidas hay una particular combinación en las entradas.

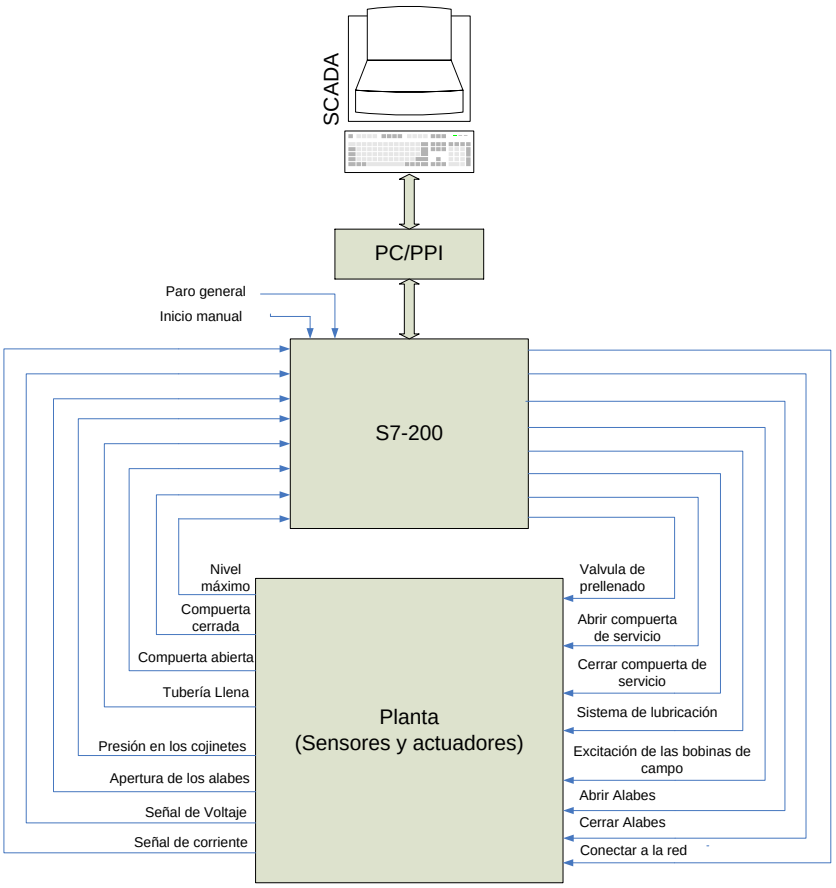


FIG. 4.91 Diagrama de bloques del sistema.

4.9.3 COMUNICACIÓN FÍSICA PC-PLC

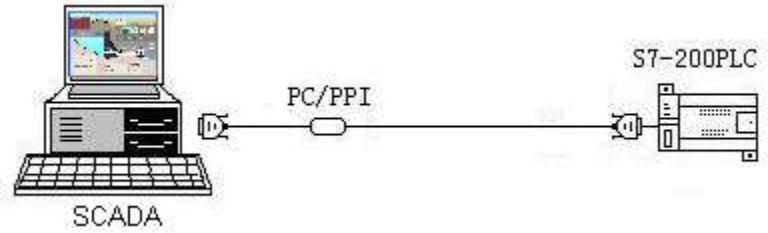


FIG. 4.92 Conexión Física PC-PLC.

Para la comunicación entre la computadora y el PLC es necesario un cable denominado PPI-PC. La denominación surge dado que utiliza el protocolo del sistema S7-200: *interfece point to point (PPI)* que se basa en la comunicación de sistemas abiertos (OSI) de la arquitectura de siete capas. El protocolo PPI es un protocolo *maestro/esclavo* implementado en un bus testigo, con niveles de señal RS-485. A pesar de que el protocolo PPI es un protocolo *maestro/esclavo* se le puede operar también en modo *Freepor;* y este es el modo que se utilizará ya que el programa Winlog requiere operar en este modo para poder leer y escribir las variables internas del PLC.

La velocidad de transferencia se puede ajustar a 9.600 ó 19.200 bits/s. Antes de programar el PLC o antes de utilizar el SCADA, se lo debe de comunicar con la PC mediante el cable PC/PPI. Para ello se realizan los siguientes pasos:

a. Conectar el extremo RS232 en el puerto COM1 de la PC.
b. Conectar el extremo RS485 en el puerto de comunicación del PLC.
c. Configurar los interruptores del cable como en la figura 4.79.
d. Energizar el PLC.

Colocar los interruptores del cable de la siguiente manera para poder realizar la comunicación entre el PLC y el Programa Winlog.

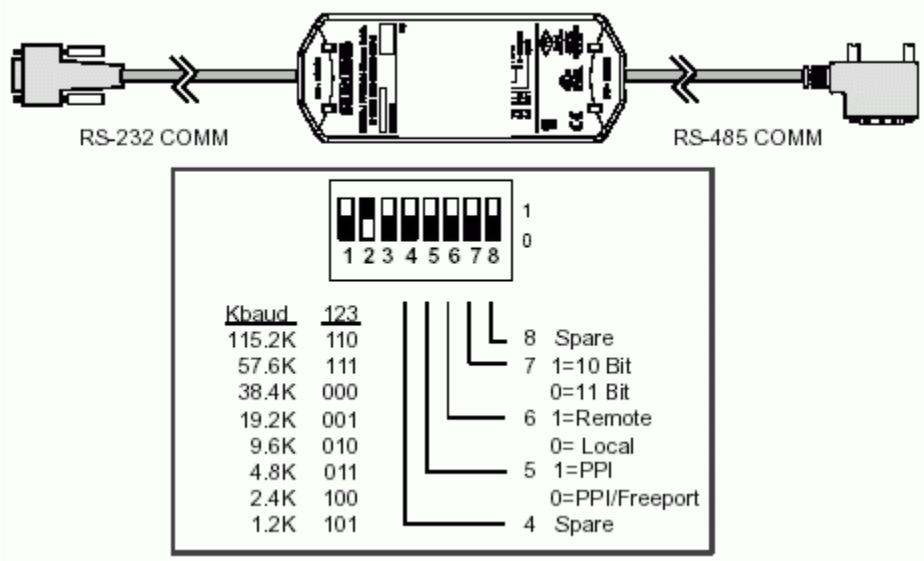


FIG. 4.93. Configuración del cable PC/PPI en modo Freeport.

4.9.4 CONSTRUCCIÓN DEL SCADA

4.9.4.1 Configuración de la comunicación e inicio del sistema.

Lo primero que hay que hacer es crear nuestro nuevo proyecto, por lo cual hay que dirigirse al menú *"Project"* y dar clic en *"New"*, con lo cual se nos pedirá el nombre de nuestro proyecto y se debe escribir *"Mini Central Hidroeléctrica"*. Con esto aparecerá en el árbol de proyecto un nuevo proyecto con el nombre especificado.

A continuación hay que realizar la configuración del protocolo y medio de comunicación que se usara para comunicarse con el PLC. Para este efecto se va a la carpeta *"Configuration"* de nuestro proyecto y en el ítem *"Channels" se* configuran dos canales de comunicación como se muestran en la figura 4.94 y 4.95. El primero es el canal de comunicación entre el SCADA y el S7-200 y el segundo es un canal que no afecta ningún

dispositivo externo y deja crear variables que tengan como único propósito comunicarse entre la plantilla y el código del proyecto.

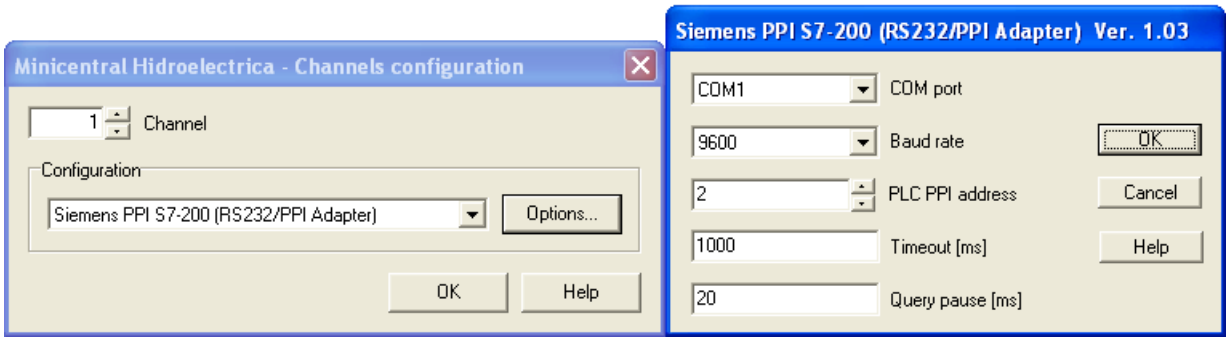


FIG. 4.94. Creación y configuración del canal de comunicaciones entre el S7-200 y Winlog.

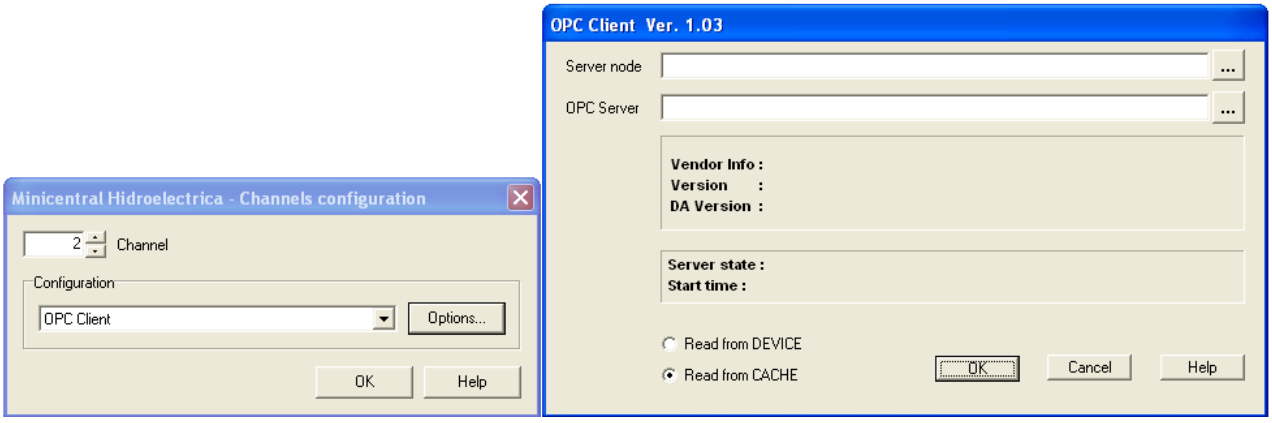


FIG. 4.95. Creación de un canal de comunicaciones auxiliar para variables internas en la PC.

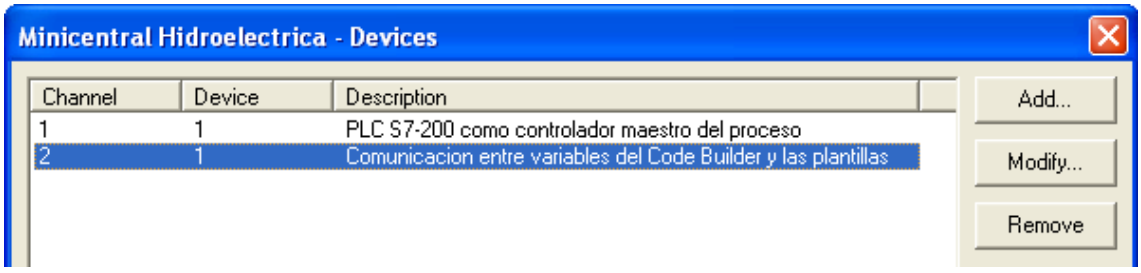


FIG. 4.96. Los dos canales de comunicación creados.

El siguiente paso es crear dos dispositivos uno por cada canal. En nuestro caso la creación de más de dos dispositivos es innecesaria ya que esta opción tiene su utilidad cuando por ejemplo el SCADA monitorea un PLC que tiene conectado varios módulos de expansión como por ejemplo un modulo de entradas salidas analógicas o un arrancador de motor, en este caso sería útil crear un dispositivo por cada modulo de expansión el cual maneje su propio conjunto de variables.

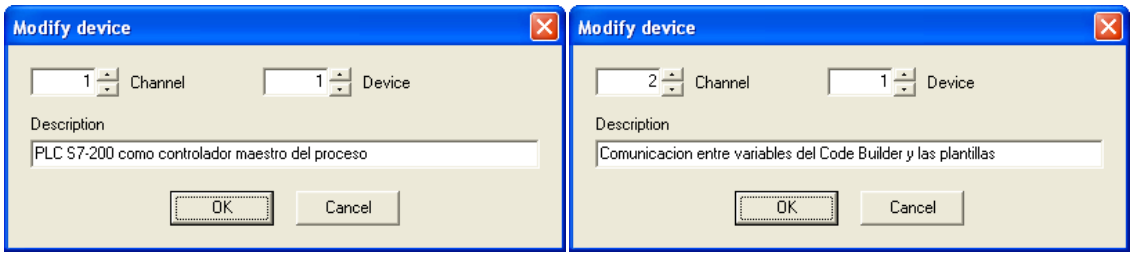


FIG.4.97. Creación de dos dispositivos, uno por cada canal de comunicación.

4.9.4.2 Creación de las compuertas de enlace

El siguiente paso es crear todas las variables del sistema, en nuestro caso muchas de ellas se leerán directamente del PLC, las otras solo servirán de comunicación interna entre las plantillas y el código.

La tabla 4.9. Lista todas las variables numéricas del sistema y muestra los parámetros más importantes que se deben configurar a la hora de crearlas.

Entre las más importantes de esta lista se tienen las compuertas voltaje y corriente que permiten leer desde las variables VW104 y VW108 los niveles de tensión y corriente que se manejan en el generador respectivamente, para posteriormente ser visualizados en una grafica de voltaje y corriente en el SCADA.

	Channel	Device	Gate ID	N ID	Address	Description	Measure	Variable type	Tolerance	Min. value	Max. value	Start value
1	2	1	ImgTubo	1		Determina que imagen se va a presentar en la tuberia forzada.		U_INT32		0	10	0
2	2	1	ImgCompSer	1		Determina que imagen se va a presentar en la compuerta de voca toma		U_INT32		0	10	0
3	2	1	EstadoTurbina	1		Se especifica el tipo de imagen que se mostrara en la turbina		S_INT32		0	10	0
4	2	1	Potencia_Mecanica	1		Potencia del eje calculado en base al caudal	[KW]	DOUBLE		0	1000	400
5	2	1	Caudal	1		Caudal efectivo del caracol de la turbina	[m^3/s]	DOUBLE		0	10	2.3
6	2	1	VoltajeExitacion	1		voltaje de exitacion del generador	[Volt]	DOUBLE		0	100	37.5
7	2	1	Frecuencia	1		Frecuencia de la linea de 480 a la salida del generador	[Volt]	DOUBLE		0	100	60
8	1	1	Voltaje	1	VW104	Monitorea el valor de la entrada analogica AIW0 del S7-200	[Volt]	S_INT32		0	65535	0
9	1	1	Corriente	1	VW108	Monitorea el valor de la entrada AIW2 del S7-200	[Amp]	S_INT32		0	65535	0
10	1	1	Sincronoscopio	1	VW110	Angulo de desfase del voltaje de la red con el generador en %	[%]	S_WORD		-100	100	-40
11	1	1	ApertAlabes	1	VW100	Contiene el valor de apertura de los alabes	[%]	U_WORD		0	100	0
12	1	1	PulsosAlabes	1	VW112	Llev el conteo de los pulsos que da el senor de los alabes	[%]	U_WORD		0	100	0
13	2	1	VoltajeTrafo	1		Voltaje a la salida del transformador	[Volt]	FLOAT		0	30000	0
14	2	1	CorrienteTrafo	1		Corriente en el primario del Transformador	[Amp]	FLOAT		0	200	0
15	2	1	PotenciaEntregada	1		Potencia inyectada a la red	[KW]	FLOAT		0	1000	0

TABLA 4.9. Variables Numéricas del sistema.

A continuación la tabla 4.10. Muestra todas las variables digitales que se utilizan en el SCADA, las primeras que se declaran en la tabla son las entradas y salidas directas del PLC, se han asignado una variable para cada entrada o salida que se ocupa del mismo.

La declaración de variables digitales con el nombre de alarmas se realiza porque cuando se crea una alarma como las que lista la tabla 4.11. Solo permiten asignar una condición de activación, que particularmente corresponde a una sola compuerta. Por lo que es necesario si las condiciones son más complejas que eso; escribir en el código una sentencia de validación que establezca a uno la compuerta que activará la alarma.

Gate Builder - minicentral hidroelectrica

File Edit View Help

	Channel	Device	Gate ID	N ID	Address	Description	Start value	Sample	Sample freq. [Sec.]	Read block	Record on historical file
1	1	1	VaPreAct	1	QB0.0	Estado de la valvula de pre llenado de la tuberia forzada	0	Always	1	salida	T
2	1	1	AbrCompServ	1	QB0.1	Activar la salida para abrir la compuerta de servicio	0	Always	1	salida	T
3	1	1	CrrCompServ	1	QB0.2	Activar la salida para cerrar la compuerta de servio	0	Always	1	salida	T
4	1	1	CompServAbt	1	IB0.2	Compuerta de servicio esta abierta	0	Always	1	entrada	T
5	1	1	CompServCrd	1	IB0.1	Compuerta de servicio esta cerrada	0	Always	1	entrada	T
6	1	1	TuboLleno	1	IB0.0	Es 1 cuando la tuberia esta llena, representa la entrada I0.0 del S7-200	0	Always	1	entrada	T
7	1	1	NivelMaximo	1	IB0.3	Representa el nivel maximo de la represa	0	Always	1	entrada	T
8	1	1	ParoPC	1	MB0.1	Detiene el funcionamiento de la planta normalmente	0	Always	1		T
9	1	1	BombaLevantamiento	1	QB0.3	Activa la bomba de levantamiento que da presion de aceite a los cojinetes	0	Always	1	salida	T
10	1	1	SensorPresion	1	IB0.5	Se activa cuando la presion de los cojinetes es la adecuada	0	Always	1	entrada	T
11	1	1	Exitacion	1	QB0.6	Activa la exitacion de campo	0	Always	1	salida	T
12	1	1	ConectarRED	1	QB0.7	Interruptor de conexion a la red	0	Always	1	salida	T
13	1	1	AbrAlabes	1	QB0.4	Abre los alabes	0	Always	1	salida	T
14	1	1	CrrAlabes	1	QB0.5	Cierra los alabes	0	Always	1	salida	T
15	1	1	InicioPC	1	MB0.0	El comandode inicio automatico desde el PC	0	Always	1		T
16	1	1	Emergencia	1	IB1.0	Pulsador externo de paro de emergencia	0	Always	1	entrada	T
17	2	1	Alarma1	1		El tubo esta Lleno y se quiere abrir la valvula de PreLlenado	0	Never			F
18	2	1	Alarma2	1		La compuerta ya esta abierta y se quiere volver a abri	0	Never			F
19	2	1	Alarma3	1		La compuerta esta cerrada y se quiere volver a cerrar	0	Never			F
20	2	1	Alarma4	1		La bomba de lebantamiento esta activa y se quiere activar	0	Never			F
21	2	1	Alarma5	1		Cuando se quiere conectar a la red y no esta sincronizado	0	Never			F
22	2	1	Alarma7	1		La tuberia no esta llena y se quiere abrir la compuerta de bocatoma	0	Never			F
23	2	1	Alarma8	1		Se pierde inesperadamente la exitacion de campo	0	Never			F
24	2	1	Alarma9	1		la turbina dejo de girar y se esta conectado a la red	0	Never			F

TABLA 4.10. Variables digitales del sistema.

Gate Builder - minicentral hidroelectrica

File Edit View Help

	Gate ID	N ID	Message	Type	Gate ID	N ID	Condition	Value	Filter time [Sec.]	Class 1	Class 2	Is Alarm	Need Acknoledge
1	Alarma1	1	El tubo esta Lleno no es permitido activar la valvula en este estado	DIG	Alarma1	1	1	1	0	1	Alarma1	T	F
2	Alarma2	1	La compuerta ya esta abierta	DIG	Alarma2	1	1	1	0	2	Alarma2	T	F
3	Alarma3	1	La compuerta ya esta cerrada	DIG	Alarma3	1	1	1	0	3	Alarma3	T	F
4	Alarma4	1	La presion en los cojinetes ya es la adecuada	DIG	Alarma4	1	1	1	0	4	Alarma4	T	F
5	Alarma5	1	No se esta sincronizado con la red la conexion no se puede realizar	DIG	Alarma5	1	1	1	0	5	Alarma5	T	F
6	Alarma6	1	La represa esta a punto de desbordarse el nivel maximo ha sido alcanzado	DIG	NivelMaximo	1	1	1	5	6	Alarma6	T	F
7	Alarma7	1	No es permitido abrir la compuerta si la tuberia esta basia	DIG	Alarma7	1	1	1	0	7	Alarma7	T	F
8	Alarma8	1	Se ha desconectado de la red debido a un fallo en el sistema de exitacion	DIG	Alarma8	1	1	1	0	8	Alarma8	T	F
9	Alarma9	1	Se ha desconectado de la red debido a un bloqueo de la turbina	DIG	Alarma9	1	1	1	0	9	Alarma9	T	F

Tabla 4.11. Declaración de las alarmas del sistema.

4.9.4.3 Elaboración de las plantillas

El proceso de elaboración de plantillas en el Template builder es muy intuitivo, mucho más sencillo que el de cualquier programa de diseño de SCADA's profesional. La barra de herramientas del Template Builder proporciona una reducida pero basta y suficiente cantidad de utilidades que se pueden utilizar para nuestros diseños.

A continuación se muestra el proceso que se siguió para la elaboración de las plantillas para nuestro sistema SCADA.

Hoja Principal

La pantalla principal tiene una reducida cantidad de elementos en su gran mayoría bitmaps. Para la imagen de fondo se eligió desde la barra de herramientas un *"bitmap"* se coloco en la posición 0,0 y se expandió hasta alcanzar un tamaño de 949 x 642. Los 5 botones del frame Acciones son necesarios para ejecutar cada una de las acciones con las que cuenta la pantalla, el frame de estado de dispositivos, cuenta con 5 leds, tres de ellos

indican el estado de la tubería forzada, dos el estado de la compuerta y uno el estado de la válvula pre llenado.

El frame *"inicio automático del sistema"* cuenta con dos botones que permiten el inicio o paro del proceso automático. Los indicadores de estado del proceso son cinco bitmaps que corresponde, cada uno, a una etapa del proceso que más a delante se mostrará como están configurados.

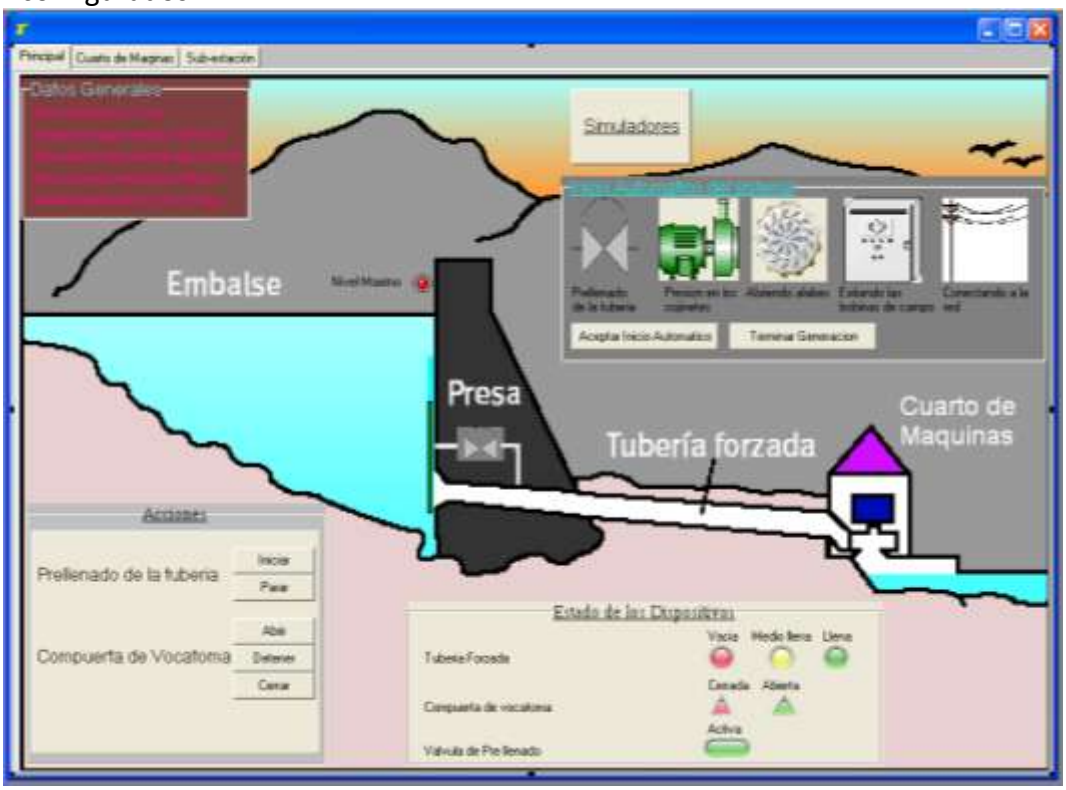


FIG. 4.98. Vista de la hoja principal del Template Builder.

A continuación desde la tabla 4.12 hasta la tabla 4.19 se muestra la secuencia de imágenes hecha para la animación de los diferentes elementos de la hoja. En la columna condiciones se especifica con que valor de la variable se muestra determinada secuencia y el periodo de cambio entre imágenes para cada secuencia, para aquellas que son animaciones, cada fila de la tabla corresponde a una imagen o animación. son imágenes fijas cuando solo se agrega una imagen a la secuencia y es animación cuando la secuencia de imágenes que se va a mostrar tiene más de una imagen.

Es posible en un mismo bitmap cambiar la imagen que se verá, declarando una gran cantidad de secuencias, estas se mostrarán asignándole una condición diferente a cada una. Es importante decir que no se le puede asignar la misma condición a dos imágenes por que el programa entra en conflicto y muestra la primera secuencia editada ignorando las demás condiciones.

Compuerta de servicio	
Condición	***Bitmap states***
ImgCompSer=0	
Ciclo=0 seg.	
ImgCompSer=1	
Ciclo=800 seg.	
ImgCompSer=2	
Ciclo=0 seg.	
ImgCompSer=3	
Ciclo=0 seg.	
ImgCompSer=4	
Ciclo=800 seg.	

TABLA 4.12. Condición y secuencia de imágenes de la compuerta de servicio.

Válvula Pre llenado	
Condición	***Bitmap states***
ValPreAct=0	
Ciclo=0 seg.	
ValPreAct=1	

TABLA 4.13. Condición y secuencia de imágenes de la válvula de pre llenado.

Tubería Forzada	
Condición	***Bitmap states***
ImgTubo=0	
Ciclo=0 seg.	
ImgTubo=1	
Ciclo=1600 seg.	
ImgTubo=2	
Ciclo=0 seg.	
ImgTubo=3	
Ciclo=0 seg.	
ImgTubo=4	
Ciclo=600 seg.	

TABLA 4.14. Condición y secuencia de imágenes de la tubería forzada.

Inicio automático del sistema (Válvula Pre llenado)	
Condición	***Bitmap states***
ImgTubo=0	
Ciclo=0 seg.	
ImgTubo =1,2	
Ciclo=1000 seg	
ImgTubo =3,4	
Ciclo=0 seg.	

TABLA 4.15. Condición y secuencia de imágenes en el apartado inicio automático del sistema de la válvula de pre llenado.

Inicio automático del sistema (Presión en los cojinetes)				
Condición	***Bitmap states***			
Sensorpresion=1 Ciclo=0 seg.				
BombaLevantamiento =0 Ciclo=0 seg				
BombaLevantamiento =1 Ciclo=1000 seg.				

TABLA 4.16. Condición y secuencia de imágenes en el apartado inicio automático del sistema de la Bomba de levantamiento.

Inicio automático del sistema (Abriendo Alabes)				
Condición	***Bitmap states***			
EstadoTuberia=0 Ciclo=0 seg.				
EstadoTuberia=1,3,5,7 Ciclo=1000 seg				
EstadoTuberia=2,4,6,8 Ciclo=0 seg.				

TABLA 4.17. Condición y secuencia de imágenes en el apartado inicio automático del sistema de la apertura de los alabes.

Inicio automático del sistema (Excitando las bobinas de campo)				
Condición	***Bitmap states***			
Excitación=0 Ciclo=0 seg.				
Excitación=1 Ciclo=0 seg				

TABLA 4.18. Condición y secuencia de imágenes en el apartado inicio automático del sistema de la excitación de las bobinas de campo.

Inicio automático del sistema (conectado a la red)				
Condición	***Bitmap states***			
ConectarRED=0 Ciclo=0 seg.				
ConectarRED=1 Ciclo=0 seg				

TABLA 4.19. Condición y secuencia de imágenes en el apartado inicio automático del sistema de la conexión a la red.

La tabla 4.20 muestra la función que se llama al presionar cada uno de los botones descritos en la columna de la derecha.

Etiqueta de botón	Función a llamar
Iniciar (Pre llenado de la tubería)	Abrir_valvula_prellenado()
Parar (Pre llenado de la tubería)	Cerrar_Valvula_Prellenado()
Abrir (compuerta de bocatoma)	Abrir_compuerta_servicio()
Detener (compuerta de bocatoma)	Detener_compuerta_servicio()
Cerrar (compuerta de bocatoma)	Cerrar_compuerta_servicio()
Aceptar inicio automático	InicioAutomatico()
Terminar generación	Terminar_Generacion()
Simuladores	Abrir plantilla simuladores

TABLA 4.20. Correspondencia de funciones que se llaman por cada botón.

El estado de los leds que se describen en la tabla 4.21 es asignado por la condición que se muestra en la columna de la derecha de esta misma tabla.

Etiqueta de led	Condición ON
Tubería forzada bacía	ImgTubo=0
Tubería forzada medio llena	ImgTubo=2
Tubería forzada llena	ImgTubo=3
Compuerta de servicio cerrada	CompServCrd=1
Compuerta de servicio abierta	CompServAbt=1
Nivel máximo	NivelMaximo=1
Válvula de pre llenado activa	ValPreAct=1

TABLA 4.21. Condición de encendido para cada led de la hoja principal.

Hoja Cuarto de maquinas

La imagen de la pestaña cuarto de maquinas se muestra en la figura 4.99 el Frame *"parámetros eléctricos del generador"* está constituido por cuatro cuadros texto en los que se muestran los valores de voltaje, corriente, frecuencia y excitación del generador, estos cuadros texto están asociados a unas compuertas como se describe en la tabla 4.28. El frame bomba de levantamiento está constituido por un botón, un led que se ilumina si el sensor de presión esta activo y un bitmap que cambia entre gris y verde cuando la bomba esta activa. El frame sistema de excitación está compuesto de un interruptor y cuatro bitmaps que cambian como se especifica según las tablas 4.23 y 4.24. el sistema de alabes y turbina está constituido de una barra deslizante asociada directamente a la variable ApertAlabes, tres cuadros texto que muestran el valor de la apertura de los alabes, el caudal y la potencia mecánica, estos datos se calculan internamente en el SCADA ya que no se tiene dispositivos externos que midan estos parámetros. La imagen de la turbina es un solo bitmap que está diseñado como se muestra en la tabla 4.25.

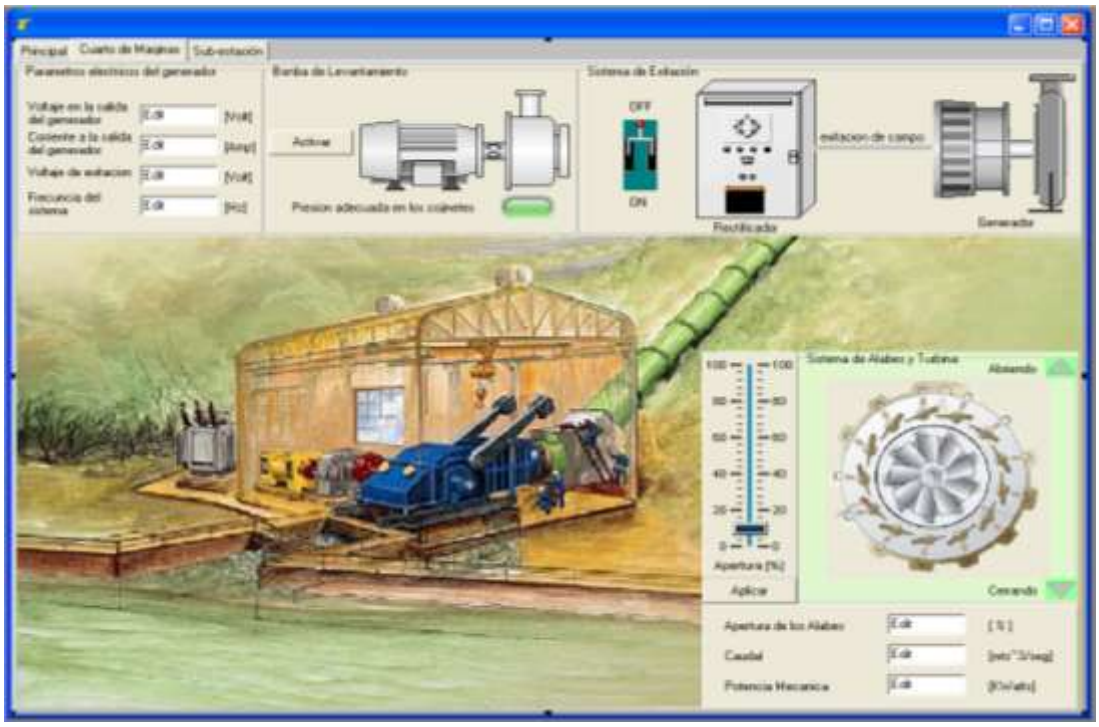


FIG. 4.99. Vista de la hoja cuarto de maquinas del Template Builder.

Bomba de levantamiento				
Condición	***Bitmap states***			
BombaLevantamiento =0				
Ciclo=0 seg				
BombaLevantamiento =1				
Ciclo=1000 seg.				

TABLA 4.22. Condición y secuencia de imágenes de la Bomba de levantamiento.

Rectificador y banco de baterias				
Condición	***Bitmap states***			
Excitación =0				
Ciclo=0 seg				
Excitación =1				
Ciclo=0 seg.				

TABLA 4.23. Condición y secuencia de imágenes del rectificador y banco de baterías en la excitación.

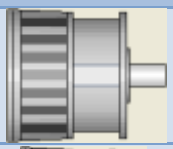
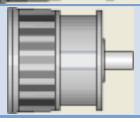
Generador				
Condición	***Bitmap states***			
Excitación =0 Ciclo=0 seg				
Excitación =1 Ciclo=1000 seg.				

TABLA 4.24. Condición y secuencia de imágenes del generador en la excitación.

Sistema de alabes y turbina				
Condición	***Bitmap states***			
EstadoTurbina=0 Ciclo=0 seg.				
EstadoTurbina =1 Ciclo=1600 seg.				
EstadoTurbina =2 Ciclo=0 seg.				
EstadoTurbina =3 Ciclo=800 seg.				
EstadoTurbina =4 Ciclo=800 seg.				
EstadoTurbina =5 Ciclo=400 seg.				
				
EstadoTurbina =6 Ciclo=400 seg.				

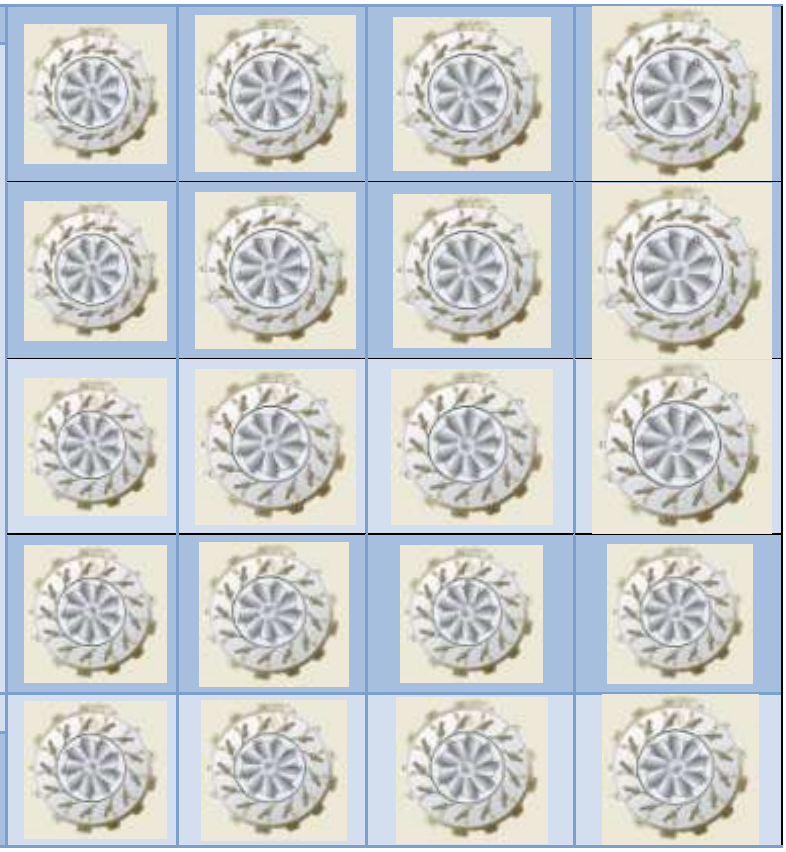
EstadoTurbina =7 Ciclo=200 seg.	
EstadoTurbina =8 Ciclo=200 seg.	

TABLA 4.25. Condición y secuencia de imágenes de la turbina en el sistema de alabes y turbina.

Etiqueta de botón	**Función o puerta a llamar**
Activar (bomba de levantamiento)	BombaLevantamiento()
Aplicar (apertura de los alabes)	Alabes()
Interruptor de excitación	Excitación

TABLA 4.26. Correspondencia de funciones a llamar para cada botón.

Etiqueta de led	**Condición ON**
Presión adecuada en los cojinetes	SensorPresion=1

TABLA 4.27.condicion de activación para cada led de la plantilla.

Etiqueta del cuadro texto	**Puerta de enlace**
Voltaje en la salida del generador	Voltaje
Corriente en la salida del generador	Corriente
Voltaje de excitación	VoltajeExitacion
frecuencia del sistema	frecuencia
Apertura de alabes	PulsosAlabes
Caudal	Caudal
Potencia Mecánica	Potencia_Mecanica

TABLA 4.28. Variable asociada a cada cuadro texto de la plantilla.

Hoja de la subestación

En las Hojas anteriores se han visto elementos como lo son bitmaps, botones, leds, etc. Estos son considerados objetos básicos. En esta hoja también se han agregado; pero además, también se han incorporado algunos elementos avanzados, no por la dificultad de su uso si no por la complejidad con la que Winlog los trata, dentro de los cuales tenemos HistoricalView que nos permite guardar en un archivo texto, CVS o RTF, el valor de una o más variables a un determinado intervalo de tiempo, en este ejemplo se está usando para monitorear las entradas de voltaje y corriente guardando un valor cada 15 minutos, con lo cual se puede exportar a un archivo texto un perfil de valores de las variables monitoreadas. Otro objeto que se ha incorporado son dos graficas en las que se visualiza en tiempo real el comportamiento de las variables de voltaje y corriente.

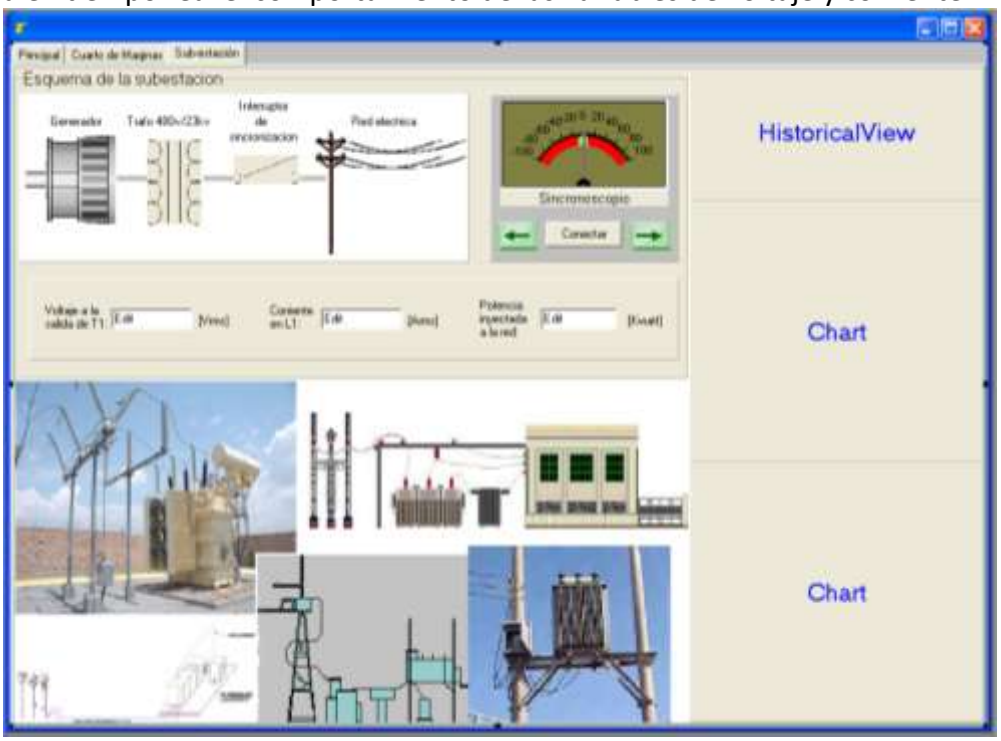


FIG.4.100. Vista de la hoja Subestación del Template Builder.

Las graficas de la figura 4.100 no se configuran completamente en el template Builder, el parámetro más importante, el cual es la variable a visualizar, se asigna cuando el SCADA ya está corriendo dando clic derecho y creando grupos de variables a visualizar.

Por último se han incorporado en nuestro diseño dos Gadgets, el primero es la barra deslizante que ya lo habíamos mencionado, y el segundo es un medidor analógico, estos objetos son elementos muy bien trabajados con los que el usuario puede interactuar, estos se pueden operar con la simplicidad con la que se hace con cualquier objeto básico, simple y sencillamente asignándole una compuerta a la cual escribirá o de la cual leerá valores.

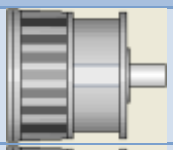
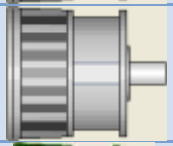
Generador				
Condición	***Bitmap states***			
EstadoTrubina<3 Ciclo=0 seg				
EstadoTurbina=3,5,7 Ciclo=1000 seg.				
EstadoTurbina=4,6,8 Ciclo=0 seg.				

TABLA 4.29. Condición y secuencia de imágenes del generador en el sistema conexión a la red.

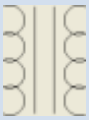
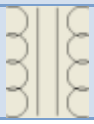
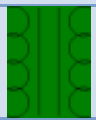
Transformador				
Condición	***Bitmap states***			
EstadoTurbina <3 Ciclo=0 seg				
EstadoTurbina>=3 Ciclo=1000 seg.				

TABLA 4.30. Condición y secuencia de imágenes del transformador en el sistema de conexión a la red.

Interruptor de Sincronización				
Condición	***Bitmap states***			
ConectarRED =0 Ciclo=0 seg				
ConectarRED=1 Ciclo=0 seg.				

TABLA 4.31. Condición y secuencia de imágenes del interruptor de sincronismo en el sistema de conexión a la red.

Red electrica				
Condición	***Bitmap states***			
ConectarRED =0 Ciclo=0 seg				
ConectarRED=1 Ciclo=800seg.				

TABLA 4.32. Condición y secuencia de imágenes de la red en el sistema de conexión a la red.

Etiqueta del cuadro texto	Puerta de enlace
Voltaje de salida del trafo	VoltajeTrafo
Corriente de salida del trafo	CorrienteTrafo
Potencia inyectada a la red	PotenciaEntregada

TABLA 4.33. Variable correspondiente a cada cuadro texto.

Etiqueta de botón	Función a llamar
Flecha izquierda	Sincronizar_abajo()
Flecha derecha	Sincronizar_arriba()
Conectar	Conectar()

TABLA 4.34. Correspondencia de funciones a llamar para cada botón.

Después de creadas las plantillas se debe de volver a la pestaña *"Configuration"* y en el ítem *"Template"* asignar la plantilla que queremos que aparezca al momento de iniciar la aplicación en el *"Rumtime"* . en la imagen 4.101 se muestran dos plantillas, la que se creo llamada *"Central_Hidroelectrica"* y una llamada *"simuladores"*, de esta plantilla no se entrara en detalle ya que se elaboro con el propósito de simular los interruptores externos para no estar en comunicación constante con el PLC y solo fue hecha para efectos de pruebas en el proceso de diseño.

Seleccionamos *"Central_Hidroelectrica"* y la agregamos presionando el botón *"Add"*.

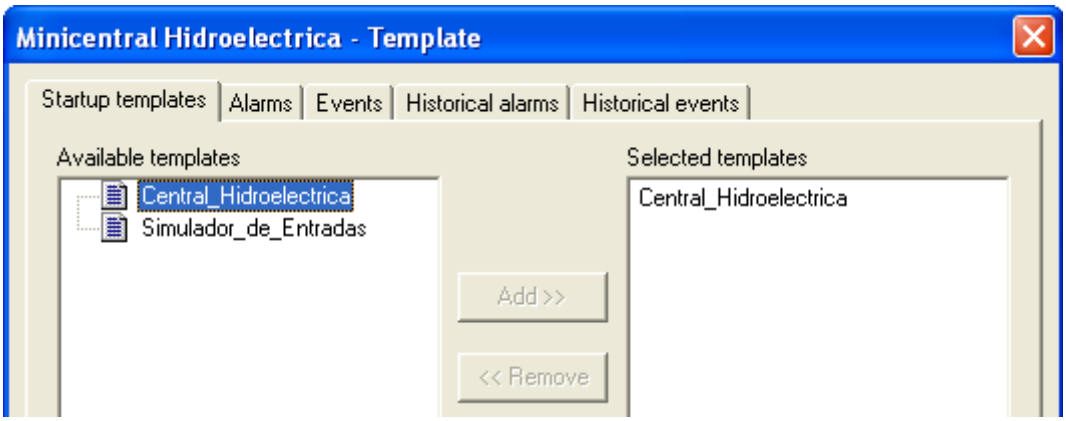


FIG. 4.101. Selección de la plantilla que se mostrará al iniciar el Runtime builder.

4.9.4.4 Edición de los códigos que se ejecutarán en tiempo real

El cerebro de todo el sistema SCADA es el código editado en el *"Code Builder"* el cual tiene una gran similitud en sintaxis con el código C estándar, con la diferencia que no se soportan todas las características del lenguaje C, muchas de las ventajas se han perdido pero se gana simplicidad y legibilidad del código.

Se han creado tres archivos, dos de los cuales son esenciales y el tercero trabaja directamente con la plantilla de simuladores de la cual ya se hablo anteriormente.

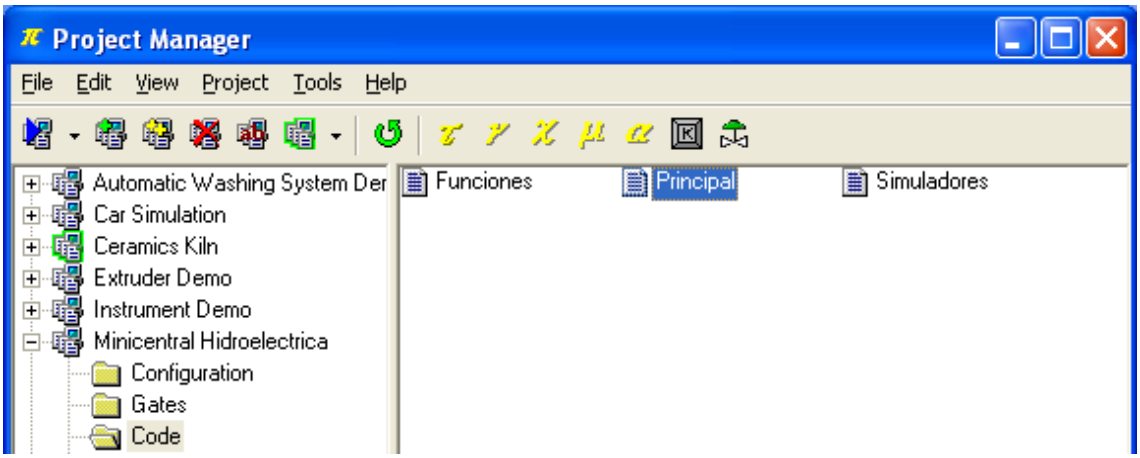


FIG. 4.102. Los tres archivos editados en Code Builder.

Código Principal

Esta es la función main que todo conocedor del lenguaje C sabe que no puede faltar, esta es una de las principales diferencias entre esta sintaxis y la del lenguaje C, ya que en el code builder no es necesario declarar una función main y si se declara no es necesario que se llame "main" dado que esta no es una palabra recervada en este lenguaje. Para hacer que el SCADA al momento de iniciar ejecute las sentencias de esta función basta con colocar la directiva "#Startup" y básicamente no importa como se llame la función cada vez que se coloque esta directiva el programa iniciará en esa función. Un ejemplo de cómo declarar la directiva #Startup se ve en la figura 4.103.

```
6  //Funcion principal, esta se inicia desde el momento de correr
7  //el SCADA
8  Function void main()
9  #Startup
10
11     real Caudal=0;
12     real Potencia=0;
13     int  retraso=UnsignedGetTickCount()/1000;
14     real  VoltajeTrafo=0;
15     real  CorrienteTrafo=0;
16     real  PotenciaEntregada=0;
17
18    //*****************************
19    // Monitoreo de sensores
20    //*****************************
21    while (WindowIsOpen())
```

FIG. 4.103. Porción de código del programa principal.

Otra diferencia con respecto al código C es que si se inicia en la función con la directiva #Startup, el proceso solo se ejecutara una vez, para hacerlo cíclico se debe de colocar un lazo while con la condición "WindowIsOpen" esto mantiene ejecutando todas las sentencias que estén dentro del lazo while mientras se ejecuta el programa SCADA.

Con relación a lo que hace la función “main” se puede decir que, monitorea el estado de todos los sensores y variables del sistema respondiendo así a algún cambio en alguna de ellas.

Para ver el código completo y las sentencias declaradas en esta función refiérase al anexo D.

Código de funciones

Todas las funciones que efectúan cambios en las imágenes de la plantilla o en las entradas y salidas del PLC, Como por ejemplo el abrir o cerrar la compuerta de bocatoma, o abrir/cerrar los alabes y girar la turbina, están declaradas en este archivo.

Es un archivo solo de funciones cada vez que se agregue una función en este archivo estará a la disposición de poderla utilizar en la llamada desde algún objeto de las plantillas o desde otro archivo, sin necesidad de incluirlo en los encabezados de programa. Esta es otra gran diferencia con respecto al lenguaje C.

Si se desea una vista detalla de todo con lo que cuenta este archivo refiérase al anexo D, para ver l código completo con sus comentarios respectivos.

4.9.5 DISEÑO DEL PROGRAMA DE CONTROL DEL PLC

El PLC ejecuta en su interior un programa, este es el inicio y paro automático de todo el sistema, en la figura 4.104 se presenta el Grafcet de todo el automatismo. Si se ejecuta el inicio automático, el SCADA pasa a ser un simple monitor de las variables ya que para que el sistema se detenga correctamente se debe hacer por medio de los pulsadores de paro ubicados en el exterior y en el interior del SCADA.

A parte del programa principal, se ha configurado la interrupción por tiempo, que se ejecuta cada 100ms con el propósito de monitorear regularmente las entradas analógicas cada 100ms y convertir ese dato a un valor entre 0 V y 480V para el voltaje y 0 Amp a 3500 Amp para la corriente, véase él grafcet de la interrupción en la figura 4.107.

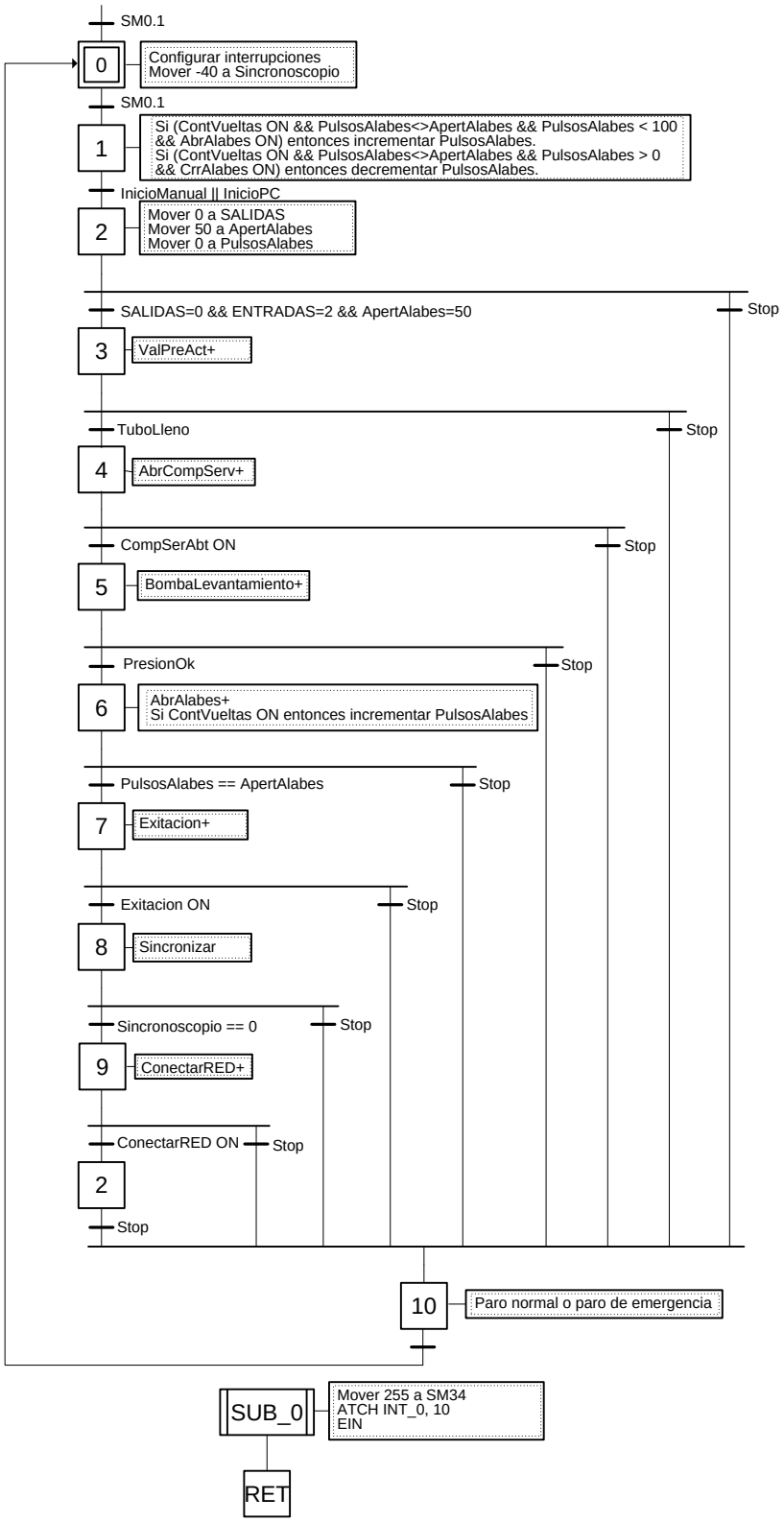


FIG. 4.104. Grafcet Sistema y Subrutina 0 Configurar interrupciones.

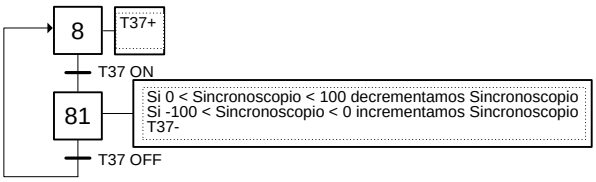


FIG. 4.105. Subrutina 8 proceso de sincronización.

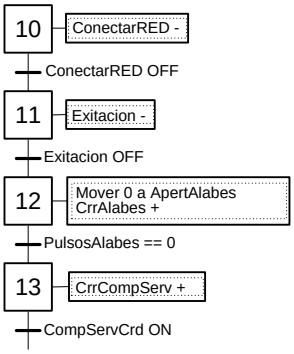


FIG. 4.106. Subrutina 10 Paro normal del sistema.

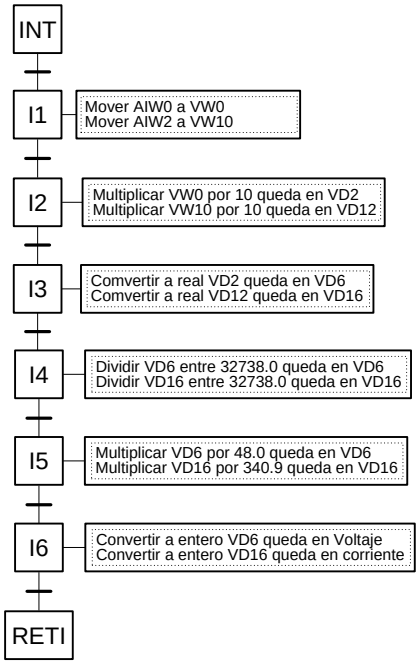


FIG.4.107. Interrupción, monitoreo y lectura de los analógicos.

CONCLUSIONES CAPITULO 4.

La herramienta de supervisión de plantas industriales Winlog Lite puesta a disposición gratuita por la empresa italiana Sielco sistema, cuenta con la gran mayoría de características de su versión pagada Winlog Pro. Prueba de ello es el sistema SCADA que brindamos como ejemplo en este capítulo, el cual monitorea los procesos de una mini central hidroeléctrica, con esto queda demostrado las capacidades de las cuales dispone este software libre.

Unos de los inconvenientes más grandes de trabajar con la versión gratuita de este programa son:

- Limitación en el tiempo de ejecución de la aplicación, reducido hasta 15 minutos.
- Limite de variables que se pueden leer desde dispositivos externos. Hasta un máximo de 24.
- Limite en la velocidad de muestreo, como mínimo se permite 1 segundo.

Estas son las razones por las cuales la versión gratuita no puede ser implementada para uso directamente profesional; pero para propósitos demostrativos y de aprendizaje es más que suficiente.

Este software presenta grandes similitudes con muchos de los sistemas para diseño de sistemas SCADA profesionales entre los cuales se encuentra, la disposición de una biblioteca de símbolos, cuenta con una herramienta de edición y compilación de código que se ejecutará en tiempo real, el lenguaje de programación presenta grandes similitudes con el lenguaje de alto nivel C, para la elaboración de la interfaz grafica cuenta con un gran número de objetos como lo son bitmap, led, interruptores, gráficos, etc. Y cuenta con la pestaña de propiedades de cada elemento que hace muy fácil su configuración, entre otras muchas más similitudes se encuentra la disposición de muchos controladores que permiten la comunicación con dispositivos de un gran número de fabricantes.

A pesar de sus muchas ventajas, presenta algunos inconvenientes; como por ejemplo la dificultad de comunicación entre el código y la interfaz grafica, ya que no se pueden acceder las variables globales declaradas en el código desde las propiedades de cada objeto. Siendo esta para nuestro propósito una de las desventajas más grandes que este programa presenta.

CAPITULO 5

RED AS-i DENTRO DE LOS BUSES DE CAMPO

CAPITULO 5
RED AS-I DENTRO DE LOS BUSES DE CAMPO.

Un bus de campo es un sistema de transmisión de información (datos) que simplifica enormemente la instalación y operación de máquinas y equipamientos industriales utilizados en procesos de producción. El objetivo de un bus de campo es sustituir las conexiones punto a punto entre los elementos de campo y el equipo de control.

Típicamente son redes digitales, bidireccionales, multipunto, montadas sobre un bus serie, que conectan dispositivos de campo como PLCs/PACs, transductores, actuadores y sensores. Cada dispositivo de campo incorpora cierta capacidad de proceso, que lo convierte en un dispositivo inteligente, manteniendo siempre un costo bajo. Cada uno de estos elementos será capaz de ejecutar funciones simples de diagnóstico, control o mantenimiento, así como de comunicarse bidireccionalmente a través del bus.

Ejemplos:

- Ethernet Industrial
- PROFIBUS
- PROFInet
- AS-interface

Ethernet Industrial.

Ethernet es el nombre de una tecnología de redes de área local (LANs) basada en tramas de datos. Ethernet define características de cableado y señalización de nivel físico y los formatos de trama del nivel de enlace de datos del modelo OSI. Además se refiere a las redes de área local y dispositivos bajo el estándar IEEE 802.3 que define el protocolo CSMA/CD.

Ethernet Industrial es similar a Ethernet Convencional, pero esta rediseñada para ser utilizada en plantas tanto de procesos continuos como de manofactura. La misma utiliza chips, componentes y cableado Ethernet estándar para reemplazar a los protocolos especializados tradicionalmente empleados en las redes industriales. Y, para satisfacer los requerimientos de estos entornos, incorpora características de robustez, redundancia y durabilidad, que permiten a los dispositivos seguir conectados a pesar de las condiciones agresivas en que a menudo se trabaja en planta.

Ethernet Industrial permite a las empresas tomar datos de una línea de manofactura y utilizarlos en el software corporativo, como por ejemplo las aplicaciones de control de inventarios y de gestión de activos; estos datos en tiempo real se ofrecen vía navegador Web a los encargados de las tareas de diagnostico y monitorización remota.

Todos los dispositivos de entrada salida pueden trabajar con la Web: con solo dar una dirección IP a los sistemas, toda la información de diagnostico sobre los dispositivos esta disponible en tiempo real.

En general, la versión de Ethernet Industrial supera a los viejos protocolos alternativos, casi siempre en velocidad de los componentes y el cableado, que puede ser el típico par trenzado estándar.

Con todo el despliegue de Ethernet Industrial en los sistemas de control, se enfrenta un obstáculo que es la existencia de brechas de seguridad, ya que las viejas redes de manofactura mantienen por separado los tráficos relativos de información, control y comunicaciones, Ethernet los integra en una troncal única, lo que, conlleva riesgos de seguridad inherentes.

PROFIBUS.

PROFIBUS es el potente, abierto y robusto sistema de bus para la comunicación de proceso y de campo en redes de célula con pocas estaciones y para la comunicación de datos según IEC 61158/EN 50170. Deriva de las palabras PROcess FIeld BUS.

Los dispositivos de automatización tales como PLC, PC, HMI, sensores u actuadores pueden comunicarse a través de este sistema de bus. La normativa IEC 61158/EN 50170 sienta además las bases para garantizar la proyección de futuro de sus inversiones ya que permite ampliar con componentes conformes a normas las instalaciones existentes.

PROFIBUS puede aplicarse, entre otros, en las siguientes aéreas:
• Automatización manufacturera
• Automatización de procesos
• Automatización de edificios

De acuerdo a los posibles sectores de aplicación se diferencia en PROFIBUS entre las siguientes variantes:

Comunicación de proceso o campo
- PROFIBUS DP: (Periferia Descentralizada; Decentralized Peripherals) para un intercambio de datos rápido y cíclico con aparatos de campo. Está orientada al control a nivel sensor/actuador.

- PROFIBUS PA: para aplicaciones de automatización de proceso en zonas que exigen seguridad intrínseca.

Comunicación de datos
- PROFIBUS FMS: diseñada para control a nivel de célula, para la comunicación de datos entre dispositivos de automatización y aparatos de campo.

PROFInet.

PROFInet se desarrolló con el objetivo de favorecer un proceso de convergencia entre la automatización industrial y la plataforma de tecnología de la información de gestión corporativa y redes globales de las empresas. PROFInet se aplica a los sistemas de automatización distribuida basados en Ethernet que integran los sistemas de bus de campo existentes, por ejemplo PROFIBUS, sin modificarlos.

PROFInet es una solución de automatización distribuida: el modelo de componentes PROFInet divide el sistema general en módulos tecnológicos. El modelo de E/S de PROFInet contribuye a la integración de periféricos sencillos distribuidos. En este caso se mantiene la visualización de datos de entrada y salida de PROFIBUS. Visualización de componentes. Visualización de datos de E/S.

Dependiendo de los requisitos concretos, PROFInet ofrece tres modelos de comunicación con distintas prestaciones:

- **Modelo TCP/IP y DCOM** para aplicaciones en las que el tiempo no es crítico.

- **Tiempo real flexible (SRT, del inglés Soft Real Time)** para aplicaciones típicas de automatización en tiempo real (ciclo de tiempo de 10 ms):

  La funcionalidad de tiempo real se utiliza para datos de proceso donde el tiempo resulta crítico, es decir, con datos de usuario cíclicos o alarmas controladas por eventos. PROFINET utiliza un canal de comunicaciones en tiempo real optimizado para las necesidades de tiempo real de los procesos de automatización.

  De este modo se minimizan los tiempos de ciclo y se mejora el rendimiento a la hora de actualizar los datos de proceso. Se permiten unos tiempos de respuesta de entre 1 y 10 ms. Al mismo tiempo se reduce considerablemente la potencia de procesador necesaria en el dispositivo para la comunicación.

- **Tiempo real isócrono (IRT)** para aplicaciones de control de movimiento (ciclos de 1 ms):
  La comunicación en tiempo real asistida por hardware, conocida como Isochronous Real-Time (IRT), está disponible para aplicaciones especialmente exigentes, como el control de movimiento, y aplicaciones de alto rendimiento en automatización manufacturera. Con IRT se consigue un tiempo de ciclo inferior a 1 ms con una fluctuación de menos de 1 µs. Para ello, el ciclo de comunicaciones se divide en una parte determinista y otra abierta.

  La aceptación de PROFInet en el mercado depende, entre otras cosas, de si los sistemas de bus de campo existentes pueden o no ampliarse con PROFInet sin incurrir en grandes costes.

AS-interface

5.1 GENERALIDADES

El bus AS-interface o interfaz de Actuador/Sensor fue creado en 1990 y es un estándar internacional abierto según EN 50295 e IEC 62026-2 para la comunicación mediante buses de campo, surge para la sustitución de la gran cantidad de señales provenientes de sensores y dirigidos hacia los actuadores desde el controlador y como una alternativa económica al cableado tradicional.

El bus AS-i es un sistema de enlace para el nivel mas bajo de procesos en instalaciones de automatización. AS-i reemplaza grandes cantidades de cables por un único cable eléctrico, el cable AS-i. Por medio del cable AS-i y del maestro AS-i se acoplan sensores y actuadores binarios de la categoría más simple a las unidades de control a través de módulos AS-i en el nivel de campo.

AS-i se sitúa en la parte más baja de la pirámide de control mostrada en la figura 5.2, conectando los sensores y actuadores con el maestro del nivel de campo. Los maestros pueden ser autómatas o PC situados en los niveles bajos de control, o pasarelas que comuniquen la red AS-Interface con otras redes de nivel superior, como Profibus o ProfiNet.

5.1.1 CARACTERÍSTICAS PRINCIPALES:

- Ideal para la interconexión de sensores y actuadores binarios. A través del cable AS-i tienen lugar tanto el intercambio de datos entre sensores/actuadores (esclavos AS-i) y el maestro AS-i como la alimentación eléctrica de los sensores y actuadores.
- Cableado sencillo y económico; montaje fácil con técnica de perforación de aislamiento.
- El cable específico para AS-i, el Cable Amarillo, es autocicatrizante y está codificado mecánicamente para evitar su polarización incorrecta.
- Gran flexibilidad de topologías, que facilita el cableado de la instalación.
- Reacción rápida, el maestro AS-i necesita como máximo 5 ms para el intercambio de datos cíclico con hasta 31 esclavos conectados con direccionamiento estándar y 10 ms con direccionamiento extendido.
- Sistema monomaestro, con un protocolo de comunicación con los esclavos muy sencillo.
- Los esclavos AS-i conectados al cable AS-i pueden ser sensores/actuadores con conexión AS-i integrada o módulos AS-i a cada uno de los cuales se pueden conectar hasta ocho sensores/actuadores binarios convencionales.
- Permite la conexión de sensores y actuadores No AS-i mediante módulos activos.
- Hasta 124 sensores y 124 actuadores binarios con direccionamiento estándar.
- Si se utilizan módulos AS-i con un espacio de direccionamiento extendido, es posible la operación de hasta 248 sensores y 186 actuadores binarios con un maestro extendido.
- Longitud máxima de cable de 100 m uniendo todos los tramos, o hasta 300 m con repetidores.
- La revisión 2.1 del estándar facilita la conexión de sensores y actuadores analógicos.

- Transmisión por modulación de corriente que garantiza un alto grado de seguridad.
- Detección de errores en la transmisión y supervisión del correcto funcionamiento de los esclavos por parte del maestro de la red.
- Cables auxiliares para la transmisión de energía: Cable Negro (24 VDC) y Rojo (220 VAC).
- Grado de Protección IP-65/67 para ambientes exigentes.
- Cumple la normativa IP-20 para aplicaciones en cuadro.
- Temperaturas de funcionamiento entre –25°C y +85°C.

Las especificaciones eléctricas y mecánicas para AS-i han sido creadas por la Asociación AS-interface formada inicialmente por 11 empresas del área de los sensores y los actuadores.

Fig. 5.1. Miembros de AS-interface.

5.1.2 AS-I DENTRO DE LA COMUNICACIÓN INDUSTRIAL:

El bus AS-interface es una red estándar de mercado que cumple con todos los requerimientos para un bus de comunicación industrial y esta específicamente diseñada para el nivel mas bajo del proceso de control.

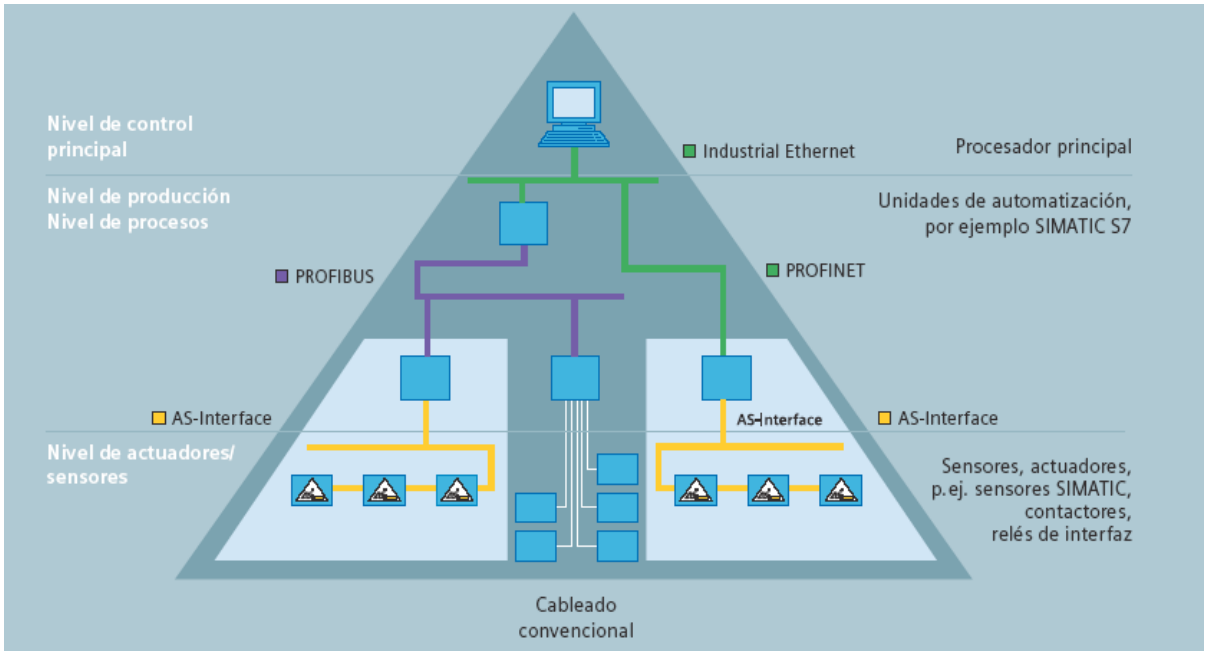


Fig. 5.2. Pirámide del proceso de control.

5.1.3 PRINCIPAL VENTAJA DE LA APLICACIÓN DE BUS AS-I.

Cuando se quiere automatizar un proceso, es necesario utilizar una gran cantidad de sensores y actuadores y el cableado de cada uno de estos va directamente al PLC de control, según la tecnología tradicional, por lo que es necesario utilizar una gran cantidad de cables. En la tecnología actual se emplea el bus AS-interface para la conexión de sensores y actuadores a la red mediante un único cable, el cable AS-i.

Fig. 5.3. Ventaja de la red AS-i

Las principales ventajas son:

- El montaje tan sencillo garantiza un funcionamiento muy simple.
- La transmisión de datos y energía por el mismo cable ahorra costes en las conexiones y el montaje.
- Alta seguridad de funcionamiento gracias a la continua supervisión de los esclavos conectados en la red.
- Puesta en marcha rápida y sencilla.
- Armarios de distribución mas pequeños, ya que se necesitan menos módulos de E/S y menos bornes.
- El grado de protección IP67 de los módulos de usuario ahorra la colocación de armarios en el campo.
- No se necesita ningún software adicional. Se utiliza la programación en STEP 7.
- Tiempos de parada más pequeños en casos de fallo, gracias al intercambio de módulos sin necesidad de reconfiguración.

5.1.4 PRINCIPALES DATOS TÉCNICOS:

La tabla 5.1 muestra los principales datos técnicos de este tipo de red de comunicación.

Método de acceso	Maestro – Esclavo
Tiempo de ciclo	Máximo 5 / 10 ms (31 / 62 esclavos)
Medio de transmisión	Cable a dos hilos sin pantalla y con protección contra cambio de polaridad. Datos y energía por el mismo cable.
Numero máximo de esclavos	62 Esclavos (máx. 496 bits E, 496 bits S con 62 esclavos)
Extensión de la red	Máximo 300 metros de red (con repetidor / extensor)
Topologías	Bus, Árbol, Estrella, Anillo, Rama
Protocolo	Según AS-interface
Aplicación	Comunicación a nivel de proceso o campo.

Tabla 5.1. Principales datos técnicos de la red AS-i

5.1.5 TOPOLOGÍAS.

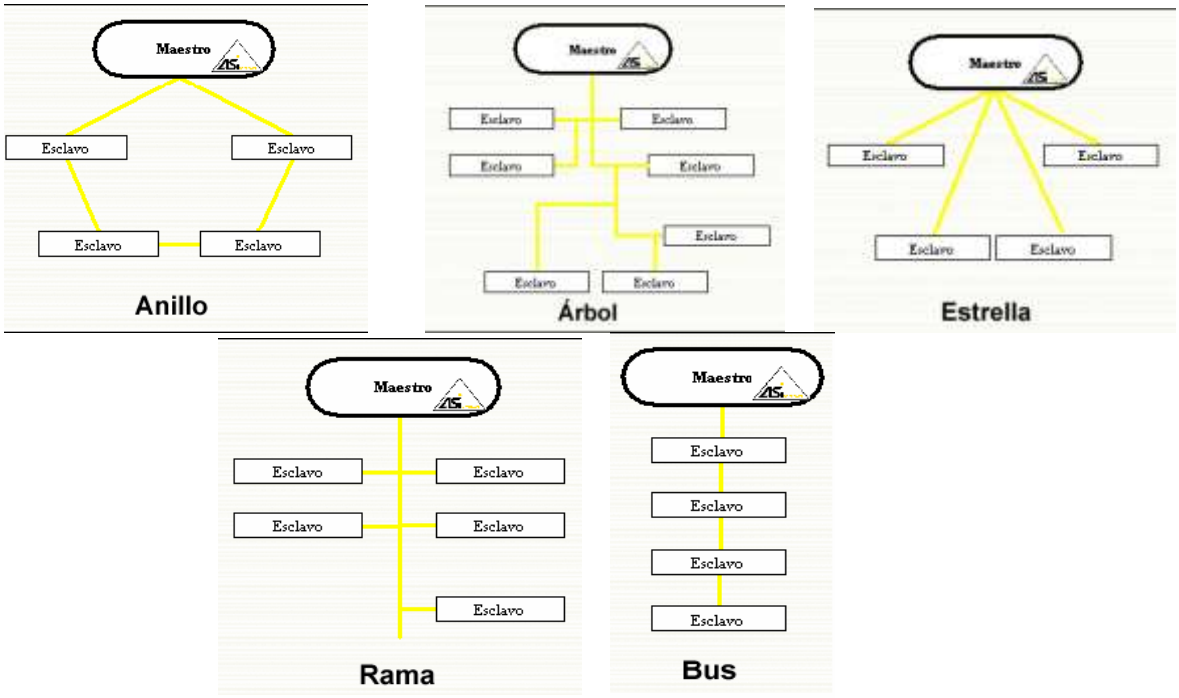


Fig. 5.4. Topologías que puede adoptar la red AS-i

5.1.6 COMPARACIÓN ENTRE VERSIONES.

Las versiones operativas de AS-interface son las versiones 2.0 y la 2.1, existen algunas diferencias entre ellas, aunque son totalmente compatibles, actualmente se comienza a utilizar la versión 3.0, en este documento el enfoque va orientado a las versiones 2.0 y 2.1.

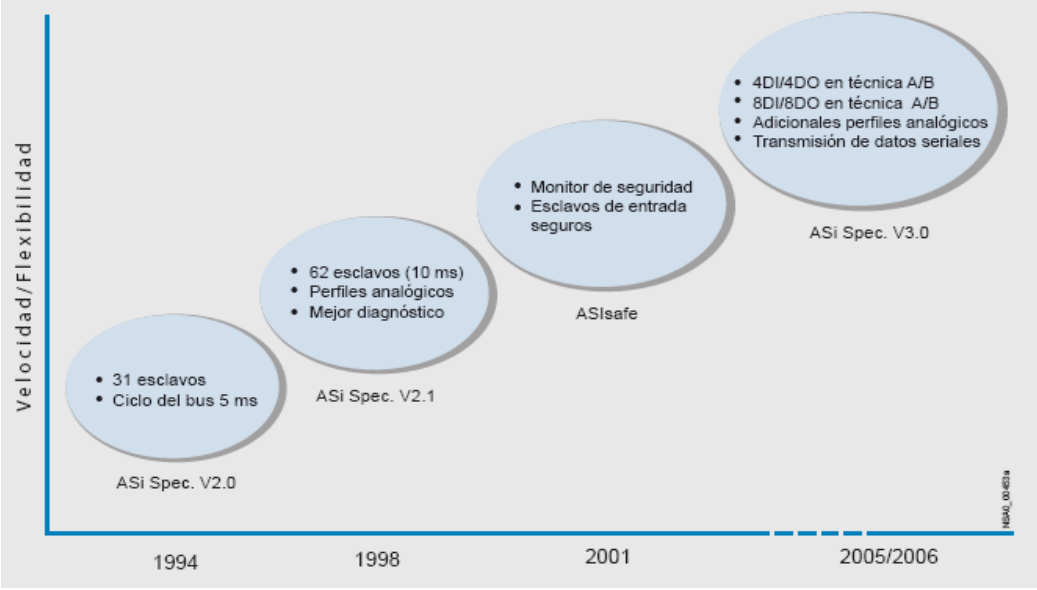


Fig. 5.5. Versiones operativas de AS-interface.

En la versión 2.0 cada esclavo puede contener como máximo 4 entradas + 4 salidas; por tanto un maestro así en la versión 2.0 puede controlar una red formada como máximo por 31 esclavos (dirección 1 a 31) con un total de 124 señales de entrada + 124 señales de salida.

A estos esclavos se les denomina esclavos "únicos".

En la versión 2.1 pueden existir dos esclavos con la misma dirección, el esclavo "A" y el esclavo "B", cada uno de ellos puede contener un máximo de 4 señales de entradas + 3 señales de salidas.

Por tanto, un maestro de la versión 2.1 podrá controlar una red formada como máximo por 62 esclavos (dirección 1A a 31A y 1B a 31B) con un total de 248 señales de entradas y 186 señales de salidas.

Ambas versiones son compatibles entre si, es decir que esclavos de la versión 2.0 se pueden conectar con maestros de la versión 2.1 y viceversa, solo que en el caso contrario en maestro no entiende de esclavos A ni B, por tanto se pierde esta ventaja.

En cualquier caso no puede haber en una misma red un esclavo único con la dirección "x" y otro esclavo A o B con la misma dirección "x".

Las tablas 5.2 y 5.3 resumen las principales diferencias entre las versiones.

	Versión 2.0	Versión 2.1
Nº de esclavos	Máximo 31 esclavos	Máximo 62 esclavos
Nº máximo de E/S	124 E + 124 S	248 E + 186 S
Tipo de transmisión	Datos y energía hasta 8 A	Datos y energía hasta 8 A
Medio Físico	Doble cable sin apantallar 2 x 1.5 mm^2	Doble cable sin apantallar 2 x 1.5 mm^2
Máximo tiempo de ciclo AS-i	5 ms	10 ms
Gestión de datos analógicos	Con bloques de función FC's	Integrada en la CP maestra
Nº de datos analógicos	16 Bytes para datos digitales y analógicos	124 datos analógicos disponibles
Método de acceso	Maestro/Esclavo	Maestro/Esclavo
Máxima longitud del cable	100 m, extensión con repetidores hasta 300 m	100 m, extensión con repetidores hasta 300 m

Tabla 5.2. Diferencias principales entre V2.0 y V2.1

Especificación AS-Interface	Número máximo de esclavos			Número de entradas digitales	Número de salidas digitales
	digitales	analógicas	ASIsafe		
Versión 2.0	31	31	31	31 × 4 = 124	31 × 4 = 124
Versión 2.1	62	31	31	62 × 4 = 248	62 × 3 = 186
Versión 3.0	62	62	31	62 × 8 = 496	62 × 8 = 496

Tabla 5.3. Diferencias principales entre V2.0, V2.1 y V3.0

5.1.7 CICLO DE LECTURA Y ESCRITURA EN LOS ESCLAVOS.

El ciclo de lectura/escritura en una red AS-i esta basado en el sistema “polling”, en donde el maestro realiza en primer lugar una llamada a todos los esclavos tipo “A” en donde copia el estado de sus entradas y les fuerza las salidas al estado indicado por el programa en cada momento, desde el esclavo 1 o 1A hasta el 31 o 31A uno tras otro y en ese orden, luego realiza el mismo proceso con los esclavos tipo “B”.

En 5 ms actualiza los datos en los 31 esclavos “A y en otros 5 ms actualiza los 31 esclavos “B”, lo que indica que con los 62 esclavos conectados en una misma red, el maestro AS-i habrá actualizado los datos en 10 ms.

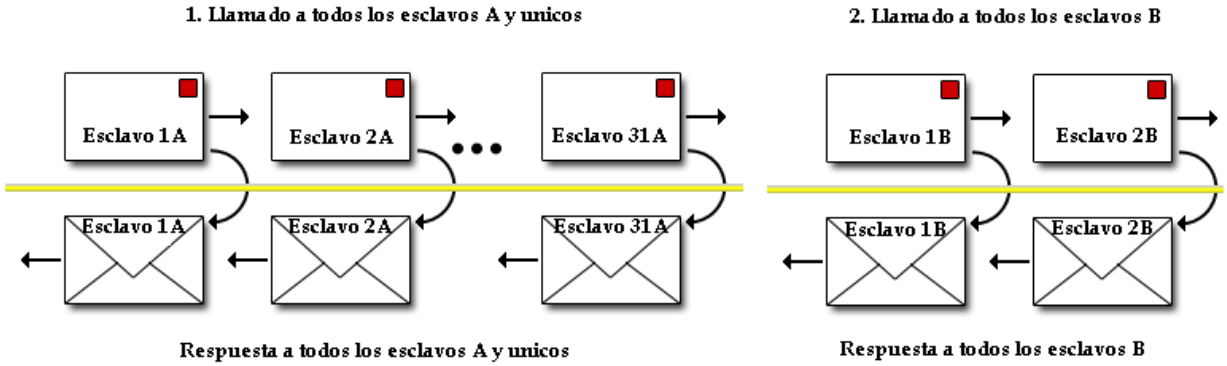


Fig. 5.6. Estructura del ciclo de lectura/escritura desde el maestro a los esclavos.

La estructura del sistema esta formada por un autómata programable que integra la CPU y el maestro AS-i (CP). De este, parte la red AS-i donde se conectan los diferentes esclavos.

El maestro dispone de su propio procesador. Realiza la función de actualizar todos los datos de los esclavos conectados en la red y guardarlos en su propia memoria no volátil. Por tanto, lee el estado de las señales de entrada de cada esclavo y las copia en su memoria, así como asigna a la salida de cada esclavo el estado que este registrado en su memoria.

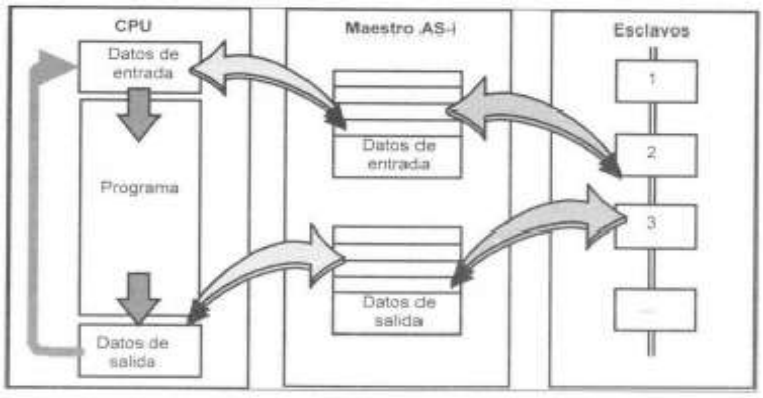


Fig. 5.7. Lectura y escritura de datos en los esclavos.

La CPU traslada los datos del estado actual de las entradas, así como del estado en que se deseen poner las salidas de cada esclavo conectado en la red. Para ello asigna un espacio en la memoria de datos del PLC. Puede ser el área de marcas (M) aunque lo más común es reservar un área del bloque de datos (DB). Por tanto, el programa del PLC, primero copia los datos registrados en la memoria no volátil del maestro AS-i que corresponden al estado de las entradas de cada esclavo, luego hace uso de estos datos realizando el programa correspondiente a la aplicación y por ultimo envía los datos que corresponden al estado en que deben estar las salidas de los esclavos al maestro AS-i.

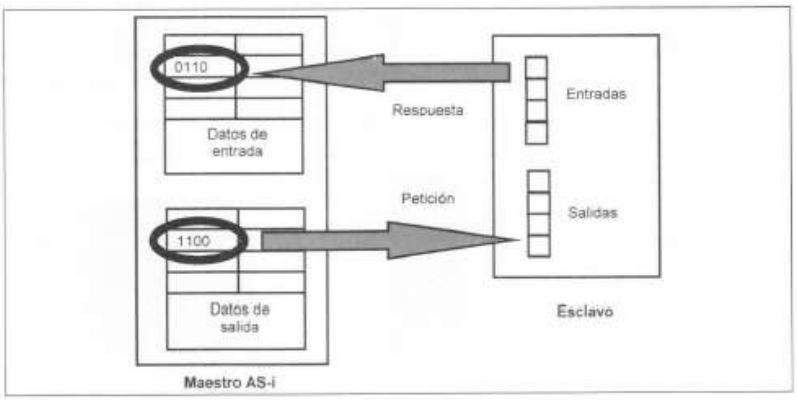


Fig. 5.8. Memoria del estado de los datos de entrada y salida de los esclavos.

5.2 EQUIPOS PARTICIPANTES EN EL BUS AS-I.

Los equipos que pueden participar en un bus AS-i se engloban en diferentes grupos:

- Fuente de alimentación AS-i.
- Maestros.
- Esclavos.
- Fuente de alimentación estándar.
- Cables y conectores.

5.2.1 FUENTE DE ALIMENTACIÓN AS-I.

Es específica y superpone una tensión aproximada de 31VDC a la tensión de los datos que circulan por el bus. Suministra energía a las estaciones conectadas al cable AS-i.

Pueden ser de diferentes tipos, clasificadas por la potencia de la misma, aunque existen otras diferencias como el grado de protección IP que incorpora. Normalmente son resistentes a cortocircuitos y sobrecargas.

Cada segmento de la red (si se utilizan repetidores) requiere su propia fuente de alimentación. Si se utiliza un módulo extensor, la fuente deberá conectarse en el extremo del extensor no conectado al maestro, ya que es en ese tramo de la red donde se

conectarán los esclavos. El otro extremo, al no poder conectarse esclavos en él, no requiere de fuente de alimentación.

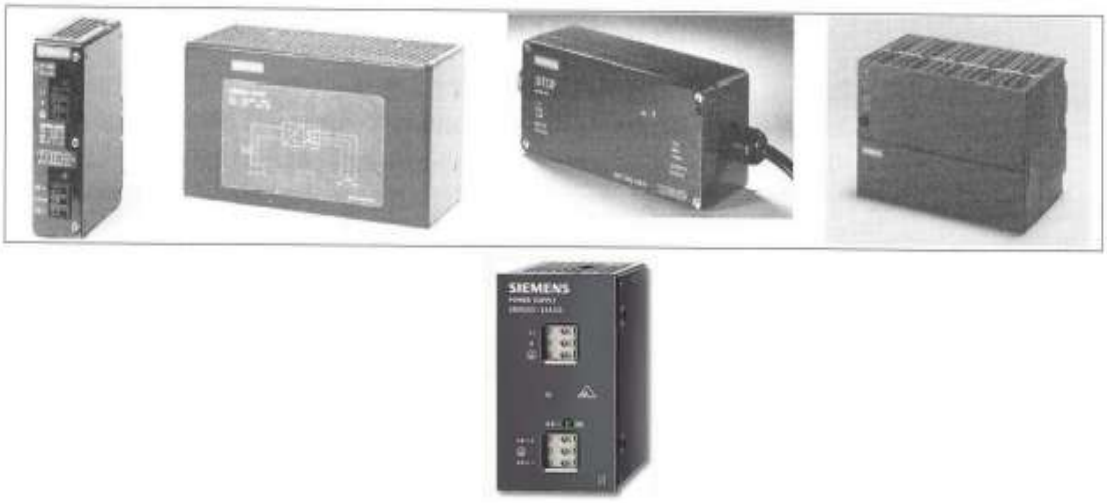


Fig. 5.9. Tipos de fuente de alimentación AS-i

5.2.2 MAESTROS AS-I.

Solo el PLC no es capaz de controlar una red AS-i, ya que no dispone de la conexión correspondiente. Por lo que se le agrega una tarjeta de expansión conectada a este que realice las funciones de maestro de la red. Esta tarjeta es conocida como CP (Communication Processor), aunque también hay maestros AS-i en formato de pasarela o Gateway. Sólo permite la existencia de un maestro en la red. Esto posibilita que el protocolo de comunicación de la red sea mucho más sencillo, simplificando la electrónica de red.

El maestro de una red AS-Interface es el encargado de recibir todos los datos que viajan a través de la red y enviarlos al PLC correspondiente. También es el que organiza todo el tráfico de datos y en caso de que fuera necesario pone los datos de los sensores y actuadores a disposición del PLC o de un sistema de bus superior (por ejemplo, PROFIBUS), a través de las pasarelas.

Además de todo esto, los maestros envían parámetros de configuración a los esclavos y supervisan la red constantemente suministrando datos de diagnóstico, por lo que son capaces de reconocer fallos en cualquier punto de la red, indicar de qué tipo de fallo se trata y determinar el esclavo que lo originó.

5.2.2.1 Maestro AS-i estándar.

Al maestro AS-i estándar se pueden conectar hasta 31 esclavos AS-i estándar o esclavos con espacio de direcciones extendido (sólo esclavos A).

Para sistema	Maestro AS-i estándar
SIMATIC S5 SPS	CP 2433 para AG S5-90U, AG S5-95U, AG S5-100U, CP 2430 para S5-115U, S5-135U, S5-155U
SIMATIC S7 SPS	CP 242-2 para S7-200 CP 242-8 para S7-200 CP 342-2 para S7-300
SIMATIC C7	C7-621 AS-i
Periferia descentralizada	DP/AS-interface Link 20 (grado de protección IP20) CP 242-8 para S7-200 CP 2433 para ET 200U CP 342-2 para ET 200M CP 142-2 para ET 200X DP/AS-interface Link (grado de protección IP 65)
PC compatible con IBM	CP 2413 para PC-AT

Tabla 5.4. Maestros AS-i estándar.

5.2.2.2 Maestros AS-i extendidos.

- ✓ Espacio de direcciones.
  Los maestros AS-i extendidos soportan 31 direcciones, que se pueden utilizar para esclavos AS-i estándar o para esclavos AS-i con espacio de direcciones extendido (extended addressing mode). Esclavos AS-i con espacio de direcciones extendido se pueden conectar por parejas (programados como esclavos A o B) con la misma dirección a un maestro AS-i extendido. Con esto aumenta el número de esclavos AS-i direccionables a 62.

  En el caso de esclavos con espacio de direcciones extendido, con la necesaria extensión de direcciones se reduce el número de salidas binarias a 3 por cada esclavo AS-i.

- ✓ Transmisión integrada de valores analógicos para esclavos AS-i según perfil 7.3/7.4
  Los maestros extendidos de SIMATIC NET soportan la transmisión integrada de esclavos analógicos AS-Interface que trabajen según el perfil 7.3/7.4 de la especificación de AS-Interface. Esclavos analógicos que trabajen con este perfil pueden ser actuados de un modo especialmente sencillo a través del programa de usuario.

Para sistema	Maestro AS-i extendido
SIMATIC S7 SPS	CP 243-2 para S7-200 CP 343-2 para S7-300
Periferia descentralizada	DP/AS-interface Link 20E (grado de protección IP20) CP 343-2 para ET200M

Tabla 5.5. Maestros AS-i extendidos.

Los maestros AS-i pueden tener alguna de las siguientes formas físicas:

Fig. 5.10. Tipos de maestros AS-i.

5.2.3 ESCLAVOS AS-I.

Se puede encontrar gran variedad de modelos diferentes en cuanto a formas, tipos y número de entradas/salidas, función, etc. Y que puede ir desde un esclavo para E/S estándar, hasta esclavos en forma de célula fotoeléctrica, parando por arrancadores, balizas de señalización, botonera de pulsadores, etc. Los esclavos intercambian cíclicamente sus datos con un maestro, el cual será el encargado de gestionar el tráfico de datos a través de la red.

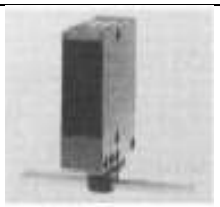
 **Serie K45** (Modelo con 2 entradas digitales y 2 salidas digitales)	 **Serie K60** (Modulo con 4 entradas digitales y 4 salidas digitales)	 **Serie Slimline** (Modelo con 4 entradas digitales y 4 salidas digitales)
 **Botonera** (Modelo con tres pulsadores y un piloto)	 **Baliza luminosa** (Modelo con 4 pilotos)	 **Serie Slimline** (Modelo con 2 entradas digitales y 2 salidas digitales)
 **Célula fotoeléctrica.**	 **Serie IP20 (F90)** (Modelo con 4 entradas digitales y 4 salidas digitales)	 **Detector de proximidad**

Fig. 5.11. Tipos de esclavos AS-i

5.2.3.1 LOGO! En el sistema AS-i. Esclavo inteligente.

Fig. 5.12. LOGO! Y modulo de expansión CM AS-interface.

Descripción.
El módulo de interfaz AS-i para LOGO permite integrar un esclavo inteligente a la red AS-i.

Con AS-i para LOGO! La interfaz modular permite que las diferentes unidades básicas se integren al sistema dependiendo de la funcionalidad requerida. Además, la funcionalidad puede ser rápida y fácilmente adaptada a los requerimientos de cambio mediante la sustitución de la unidad básica.

5.2.3.2 Módulo de comunicación.

CM AS-Interface-esclavo

El módulo de expansión tiene cuatro entradas y salidas virtuales, y actúa como una interfaz entre un sistema AS-i y un sistema logo!. El módulo permite cuatro bits de datos para transferirse desde el LOGO! al sistema AS-i y viceversa.
• Tensión de alimentación 12/24 V DC
• 4 DE/4 DA interfaces a un maestro AS-i.

Datos técnicos

Tensión de alimentación en V	24 V DC	
Entradas/salidas	4 / 4 (entradas / salidas virtuales)	
Conexión del bus	AS-Interface según especificación	
Temperatura ambiente en °C	0 ... +55	
Grado de protección	IP20	
Montaje	sobre perfil	
Dimensiones de montaje (An x Al x Pr) en mm	36 x 90 x 58	
Indicadores de los LEDs	*LED*	*Estado*
	verde	correcto
	rojo	no hay tráfico de datos
	parpadeo rojo/amarillo	dirección cero

Tabla 5.6. Datos técnicos del CM AS-i.

5.2.3.3 Qué direcciones han de darse al módulo LOGO! en la red AS-i?
Para la conexión a la red AS-i de un LOGO! como esclavo se necesita tener el módulo básico LOGO! + Módulo de expansión CM AS-i (esclavo).

Estos equipos vienen ajustados de fábrica con la dirección "0". Si se cambia un equipo ya existente por un módulo nuevo, no hace falta ajustar ninguna dirección al equipo nuevo. Simplemente hay que conectar el cable de bus al conector de red respetando la polaridad e insertarlo en la interfaz AS-i del LOGO! (identificador).

En cuanto el equipo esté conectado al cable de bus, el maestro AS-i reconoce la dirección del LOGO!. El maestro le asigna en este instante la dirección que tenía el módulo sustituido.

Si se inserta un nuevo LOGO! (es decir, hay uno nuevo adicional), entonces sí que hay que direccionarlo a través del maestro. Esto se puede hacer con la consola de programación AS-i o con un PC mediante Micro/Win.

Áreas de aplicación

Las áreas de aplicación preferidas son, donde:

- Se requieren esclavos inteligentes en la red AS-i (para pre-procesamiento de las señales).
- Son necesarios puntos independientes de trabajo de automatización, tales como:
  - Cintas transportadoras
  - Áreas de trabajo (por ejemplo, volver a trabajar en la industria manufacturera) o
  - Bombas
- Son necesarios informes de costo-eficiente de fallos.
- La inteligencia distribuida en el sitio, en los puntos de automatización individual, es una ventaja.
- La intercomunicación del LOGO! aumenta los beneficios.

Beneficios

- Durante un error, sólo falla la estación de trabajo afectada, ya que las células individuales trabajan de forma independiente.
- El bus AS-i se puede parametrizar muy fácilmente usando el asistente de AS-i de STEP7 Micro/Win (a partir de V3.2).
- Se puede ampliar de forma flexible y sencilla.
- Bajo nivel de gastos de cableado (en comparación con el cableado común sin sistema de bus)
- Reportes centrales baratos a través de la lámpara de señal AS-i.

5.2.3.4 Conectar el bus AS-interface.

Para ajustar la dirección del modulo en el bus AS-i por medio de la consola de programación, se debe utilizar una dirección valida comprendida entre 1 y 31; cada dirección solo puede ser utilizada una vez.

La dirección puede ajustarse en el bus AS-i antes o después del montaje. Si el modulo montado se direcciona a través del conector hembra de direccionamiento, la tensión del

AS-interface debe desconectarse antes. Esta medida es necesaria por razones de seguridad.

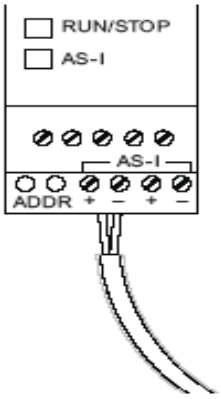


Fig. 5.13. Conexión al cable AS-i del modulo CM AS-interface.

LOGO! Solo puede actuar de esclavo en el bus AS-i. Esto significa que dos dispositivos LOGO! No pueden intercambiar datos directamente. El intercambio de datos se efectúa siempre a través del maestro AS-i.

Asignación lógica:

Sistema LOGO!		Sistema ASInterface
Entradas		Bits de datos de salida
I_n		D0
I_{n+1}		D1
I_{n+2}		D2
I_{n+3}		D3
Salidas		Bits de datos de salida
Q_n		D0
Q_{n+1}		D1
Q_{n+2}		D2
Q_{n+3}		D3

"n" depende de la posición de inserción del módulo de ampliación con respecto a LOGO! Basic. Indica el número de la entrada o salida en el código de programa de LOGO!.

Tabla 5.7. Asignación lógica de E/S.

Las E/S del modulo CM AS-interface son reconocidas dentro de LOGO como las E/S posteriores a las E/S propias del LOGO! (p. ej. El LOGO! 24RC posee 8 entradas y 4 salidas, por lo que si se conecta el CM como único modulo de expansión, las entradas del CM toman los valores 9 al 12 y las salidas los valores 5 al 8).

Nota: Asegúrese de que haya suficiente espacio disponible para las E/S de AS-interface en el área de direccionamiento de LOGO!. Si ya esta utilizando más de 12 salidas físicas o más de 20 entradas físicas, el CM AS-interface no podrá funcionar.

CM AS-interface, estados de comunicación.

El CM AS-i tiene tres estados de comunicación: el LED esta encendido en verde o rojo, o bien parpadea en rojo/amarillo.

LED ASI encendido		
Verde	Rojo	Rojo/amarillo
Comunicación ASInterface correcta	Fallo de la comunicación AS-Interface	El esclavo tiene la dirección "0".

Tabla 5.8. Estados de comunicación.

CM AS-interface, comportamiento en caso de fallo de comunicación.

- Si falla la tensión del AS-interface, se interrumpe la comunicación entre el sistema LOGO! Y los módulos de expansión dispuestos a la derecha del modulo de expansión LOGO! CM AS-i.
  Recomendación: Coloque el LOGO CM AS-i en el extremo derecho.

- Si se interrumpe la comunicación, las salidas conmutables se desactivan tras aproximadamente 40 a 100 ms.
-

5.2.4 FUENTE DE ALIMENTACIÓN ESTÁNDAR DE 24 VDC.

Algunos esclavos necesitan alimentación de 24 VDC estándar, para dar mayor potencia a los sensores/actuadores conectados a el. Para identificar que esclavos necesitan dicha alimentación se realiza una inspección ocular, fijándonos en dos aspectos:

- Dispone de bornes de conexión en donde haga referencia a algo igual a POWER EXT.

- Dispone de un led indicador de fallo con referencia a algo igual a AUX POWER.

5.2.5 CONECTORES Y CABLES.

Para diferenciar las distintas aplicaciones que pueden tener cada uno de los hilos que pueden integrar la red, nos podemos encontrar con cables perfilados con los siguientes colores:

Fig. 5.14. Tipos de cables perfilados AS-i según su aplicación.

En cuanto a los conectores, estos se utilizan cuando se quiere conectar un dispositivo estándar, ya sea sensor o actuador a esclavos del bus AS-i.

Estos conectores están formados por una carcasa y cuatro conexiones. Estas conexiones pueden tener una finalidad diferentes según el componente aplicado, sensor a dos o a tres hilos, sensor digital o analógico, etc.

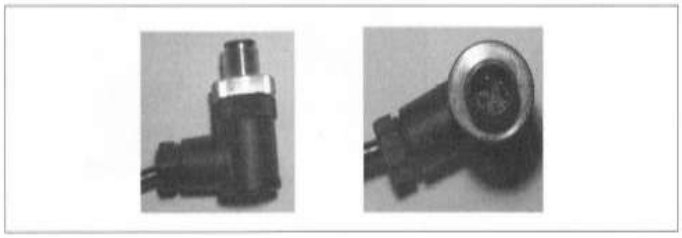


Fig. 5.15. Conector M12 para esclavos AS-i

5.3 FUNCIONAMIENTO DE LA CONSOLA DE CONFIGURACIÓN Y DIAGNOSTICO.

5.3.1 DESCRIPCIÓN TÉCNICA DE LOS EQUIPOS.

Función de cada posición del selector.
La función de cada una de las posiciones del selector giratorio se describe a continuación:

Posición	Función
OFF	Desconexión manual – Posición del selector en OFF.
ASI V=	Indicación de la tensión AS-i y el consumo de corriente AS-i.
ADDR	Direccionamiento sin almacenar la configuración del bus.
ADDR+MEM	Direccionamiento con almacenamiento de la configuración del bus.
Profile	Leer y escribir el perfil de un esclavo.
Data	Leer y escribir datos de/al esclavo.
Parameter	Leer y escribir parámetros de esclavo en formato HEX.
Memory	Cargar, almacenar, borrar y copiar juego de datos.
COM	Parametrizar y activar interface por computadora.

Tabla 5.9. Funciones de la consola de configuración.

Función de cada botón.

Descripción de cada uno de los botones que posee el instrumento frontalmente:

Fig. 5.16. Descripción de la consola de configuración.

Significado de los posibles avisos.

Seguidamente se muestra un listado de avisos y la función con su significado:

Aviso	Función	Significado
coMErr	COM	Error de transmisión.
dbLAdd	ADDR	Se ha encontrado una dirección duplicada.
Echo	Parameter	Parámetro recibido XH.
EMPty	Memory (Copy)	Copia realizada de la instalación.
Error	Data, Parameter	Dirección 0: datos y parámetros ilegibles.
Found	ADDR	Se han reconocido las direcciones de los esclavos.
Hi LoAd	ASI V=	Carga de corriente por aparato de direccionamiento demasiado alta.
intErr	ASI V=	Ninguna fuente de alimentación AS-i en el bus.
MASter	ADDR, Profile, Data, Parameter, Memory	Máster activo en el bus.
no ASi	ADDR	No se ha encontrado ninguna dirección de esclavo.
no out	Data	El tipo de esclavo no tiene ninguna salida (no OUTPUT).
no inP	Data	El tipo de esclavo no tiene ninguna entrada (no INPUT).
npPArA	Parameter	No se ha encontrado ningún parámetro.
notEQu	Memory (Copy)	No se ha encontrado ningún perfil de esclavo adecuado.
oL	ASi V=	Sobrecarga $V_{ALI} > 35$ V.
-PoL	ASi V=	Tensión < -2 V (-PoL parpadea).
PrGErr	ADDRE, Profile, Data, Parameter, Memory	Error de programación.

ProG	Data, Parameter, Memory (Copy)	Los datos se transfieren hacia el esclavo.
rEAd	ADDR, Profile, Data, Parameter, Memory	Leyendo Datos.
SEArch	ADDR, Data	Búsqueda de direcciones de esclavos.
uALbit	Data	Validación de bit correcto.
uSEnot	Data, Profile	Dirección 0 no admisible.
uSEonE	Memory (Copy)	Solo se puede copiar un esclavo, utilizar conector de direcciones.

Tabla 5.10. Significado de los mensajes que aparecen en la consola de configuración.

5.3.2 LECTURA DEL PERFIL DE UN ESCLAVO.

El perfil de un esclavo sirve como identificador de los diferentes tipos de módulos. Este identificador esta formado por códigos de E/S e ID. Además, los módulos con la especificación ampliada (AS-i V2.1) disponen de un código ID1 e ID2. Por tanto, si aparecen dos valores en hexadecimal quiere decir que el modulo es de especificación no ampliada y si aparecen cuatro valores en hexadecimal, es con especificación ampliada.

Fig. 5.17. Identificación de los esclavos.

5.3.3 LECTURA Y ESCRITURA DE DATOS DE LOS ESCLAVOS.

La posición del selector “DATA” permite visualizar directamente el estado de las entradas de cada esclavo y también permite forzar el estado de las salidas de cada esclavo.

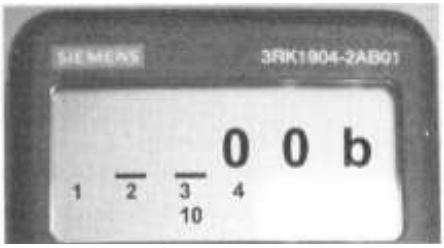


Fig. 5.18. Escritura de datos a un esclavo.

En este caso, el display indica que el esclavo examinado dispone de dos entradas, indicadas por “0”, mostrando que en este momento están desactivadas. Estas dos entradas se pueden visualizar y en su estado se puede encontrar los siguientes casos:

0 – entrada desactivada.

1 – entrada activada.

Para poder visualizar y forzar el estado de las salidas se debe poner la consola en el estado EDIT.

5.4 CONFIGURACIÓN Y PROGRAMACIÓN DE UNA RED AS-I.

5.4.1 DIRECCIONAMIENTO DE LOS ESCLAVOS MEDIANTE CONSOLA.

Cada esclavo que intervenga en la red AS-i debe tener asignada una dirección distinta entre ellos y comprendida entre los valores 1 y 31.

El direccionamiento de los esclavos puede realizarse de tres formas diferentes, que son:

- ✓ Mediante la conexión directa entre la consola de direccionamiento en un esclavo.
- ✓ Mediante la conexión de la consola de direccionamiento en el propio bus AS-i.
- ✓ A través de un bloque de función especial programado en el propio PLC.

5.4.1.1 Mediante la conexión directa entre la consola de direccionamiento en un esclavo.

Con esta conexión se asigna una dirección concreta al esclavo que se conecte de forma directa con la consola. Se puede conectar la consola teniendo el esclavo conectado en el bus y que el bus este o no conectado al maestro, pudiendo encontrarse el maestro en funcionamiento o parado, o totalmente independiente del propio bus.

Conexión de la consola a esclavos antiguos.

Se pueden encontrar esclavos de una versión anterior. Para este caso la consola trae una base con conexión externa. Se debe colocar el esclavo en la base y de esta forma poder conectarlo a la consola.

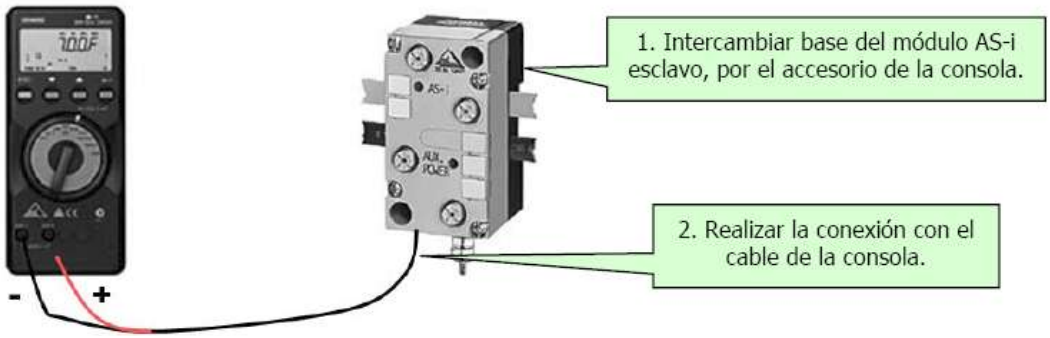


Fig. 5.19. Conexión de consola a esclavos antiguos.

Conexión de la consola a esclavos modernos.
En este caso los esclavos ya disponen de un conector para enlazar con la consola.

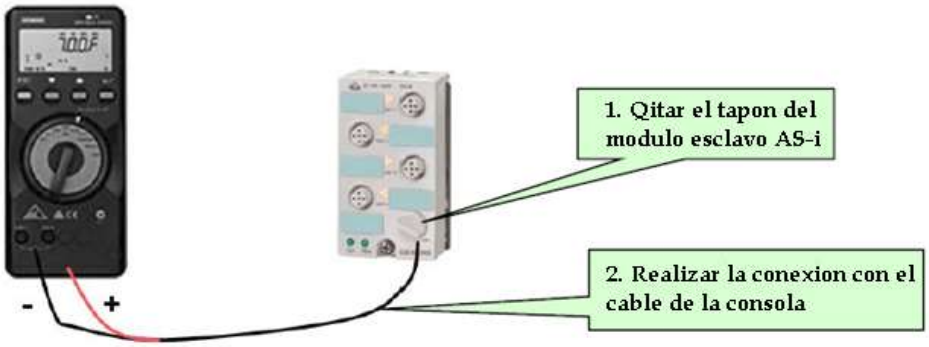


Fig. 5.20. Conexión de consola a esclavos modernos.

5.4.1.2 Mediante la conexión de la consola en el propio bus AS-i.

Si se esta en el caso de que no se pueda acoplar la base que acompaña la consola, ni el conector, la solución es conectar la consola directamente en los bornes que dispone para la conexión del bus AS-i.

Fig. 5.21. Conexión de consola directamente a los bornes.

5.4.2 DIRECCIONAMIENTO DE LOS ESCLAVOS CON CONEXIÓN DIRECTA A LA CONSOLA.

Para asignar una dirección a un esclavo, se pueden dar los siguientes casos:

- Esclavo sin conectar en el bus AS-i. en este caso la consola se conecta directamente al esclavo.

- Esclavo integrado en el bus. En este caso se tienen tres opciones:

  - ✓ Conectar la consola directamente al esclavo que se le quiere cambiar la dirección. Aquí, en el momento de la conexión de la consola, el esclavo queda desconectado del bus momentáneamente mientras la consola esta conectada.

- ✓ Conectar la consola directamente en un punto del bus, con el que se tiene la opción de modificar la dirección de cualquier esclavo conectado al bus AS-i.
- ✓ Mediante el software STEP 7 a través de comandos dirigidos sobre el propio maestro AS-i.

En el momento de querer cambiar la dirección del esclavo, se puede encontrar en los siguientes casos:

- ➢ Esclavo nuevo, por tanto con la dirección por defecto que es la "0".
- ➢ Esclavo que ya tiene una dirección diferente de "0" y que se quiere cambiar.

5.4.2.1 Mediante la conexión de la consola directamente al esclavo:

El proceso a seguir será:

- ✓ Conectar la consola de direccionamiento al esclavo o al bus, según sea el caso, mediante el cable incorporado.
- ✓ Colocar el selector de la consola en la posición "ADDR".
- ✓ Accionar el pulsador.

- ✓ En el display se visualiza la palabra "SEARCH", que quiere decir que se encuentra en estado de búsqueda.
- ✓ Se espera unos segundos y aparece en el display "SET x", donde "x" es la dirección actual del esclavo. Ahora, y con las teclas:

Elegimos la dirección que le queremos asignar y una ves se visualiza la nueva dirección, se confirma presionando:

Seguidamente, se visualiza "PROG" y a continuación "ADDRES x", en donde "x" es la nueva dirección asignada.

5.4.2.2 Mediante la conexión de la consola en el propio bus AS-i.

Con esta conexión se consigue asignar una dirección concreta de uno en uno a todos los esclavos que se encuentren conectados en el bus (cable amarillo).

Si se quiere modificar la dirección de un esclavo conectado al bus, se siguen los siguientes pasos:

- ✓ Dejar sin alimentación el maestro CP AS-i, apagando la fuente de alimentación del PLC.
- ✓ Conectar la consola de direccionamiento directamente en los cables del bus.
- ✓ Colocar el selector de la consola en la posición “ADDR”
- ✓ Accionar el pulsador:

- ✓ En el display se visualiza la palabra “SEARCH”, que quiere decir que se encuentra en estado de búsqueda.
- ✓ Se espera unos segundos y aparece en el display “USE x”, además de las direcciones de todos los esclavos encontrados en el bus, “x” es la dirección del primer esclavo encontrado. Ahora, y con las teclas:

Se elige el esclavo al cual se le quiere modificar su dirección. Se confirma accionando el pulsador:

- ✓ La dirección del esclavo aparece intermitente en el campo de direcciones, lo que quiere decir que aparece en el display “SET x”, donde “x” es la dirección actual del esclavo. Ahora, y con las teclas:

Se elige la dirección que se quiere asignar y una ves se visualice la nueva dirección, se confirma presionando:

Seguidamente, se visualiza “PROG” y a continuación “USE x” donde “x” es la dirección nueva asignada.

Si se encuentran dos o más esclavos con la misma dirección en el bus, el display de la consola muestra “DBLADD”, y la dirección correspondiente aparece de forma intermitente.

5.4.2.3 Mediante el envió de comandos desde STEP 7 al maestro AS-i.

En este caso se debe tener completamente formada y configurada, en el maestro, la red AS-i y utilizando un bloque protegido FC7 “ASI_3432” de Siemens especifico para su utilización en el bus AS-i, enviado al PLC desde STEP 7, cabe la posibilidad de poder, además de otras muchas aplicaciones, modificar el numero de estación de cualquier esclavo conectado en el bus.

5.4.3 MONTAJE DE LA RED AS-I.

5.4.3.1 Conexión del maestro AS-i.

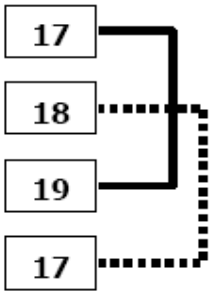
El maestro AS-i CP dispone de cuatro bornes que son los únicos a utilizar, estos son:

Conexión + bus AS-i, cable marrón

Conexión - bus AS-i, cable azul

Conexión + bus AS-i, cable marrón

Conexión - bus AS-i, cable azul



Internamente en el CP se encuentran puenteados dos a dos los bornes. Esto permite poder conectar por un lado el bus AS-i al que se conectan todos y cada uno de los esclavos y por otro lado el bus As-i para la conexión de la fuente de alimentación AS-i.

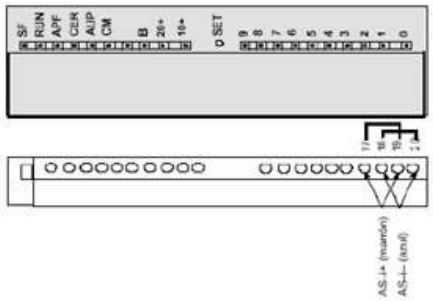
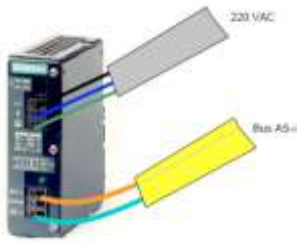


Fig. 5.22. Conexión del cable AS-i

5.4.3.2 Conexión de la fuente de alimentación AS-i.

La conexión de la fuente de alimentación AS-i se compone de los bornes de entrada para la conexión a 220 VAC y unos bornes de salida con la tensión correspondiente a la del bus AS-i, que es aproximadamente de 31 V.

El cable amarillo del bus AS-i que corresponde a datos + alimentación parte de la fuente de alimentación AS-i, en donde ese cable compuesto de dos hilos, se conectará:
Hilo Marrón: Borne positivo (+)
Hilo Azul: Borne negativo (-)



5.4.3.3 Conexión de una fuente de alimentación estándar de 24 VDC.

En algunos casos, los esclavos necesitan ser alimentados independientemente al bus As-i, de una alimentación auxiliar de 24 VDC, para ello se utiliza una fuente de alimentación estándar de 24 VDC de tensión de salida.

Puede ser que el esclavo que necesite de esta alimentación auxiliar tenga un tipo de conexión:

- Por sistema vampiro.
- Mediante bornes de conexión.

En cualquier caso se utiliza el cable perfilado de color negro en donde se respeta la polaridad la cual es idéntica del bus AS-i, es decir:
Cable marrón. Polaridad positiva [+].
Cable azul. Polaridad negativa [-].

5.4.3.4 Conexión de una tensión auxiliar de 220 VAC.

También se puede dar el caso en el que algún esclavo necesite ser alimentado independientemente al bus AS-i, de una alimentación auxiliar de 220 VAC. Para ello se utiliza cualquier punto en donde se tenga una tensión como la necesitada. Puede ser que el esclavo que necesite de esta alimentación auxiliar tenga un tipo de conexión:

- Por sistema vampiro.
- Mediante bornes de conexión.

En cualquier caso se utiliza el cable perfilado de color rojo.

5.4.3.5 Conexión de los esclavos.

La conexión de los esclavos al bus AS-i puede ser diferente dependiendo de los esclavos con los que se cuente. Hay dos tipos de conexión:

- Directa por el sistema vampiro.
- Mediante bornes de conexión.

En los módulos compactos, tanto de la serie K45 como de la serie K60, y otros como la botonera, el final de carrera, pueden realizar su conexión al bus mediante el sistema vampiro. Mientras que otros como los modelos Slimline, módulo contador, monitor de seguridad, etc. su conexión pasa a ser mediante bornes.

Fig. 5.23. Conexión de los esclavos al bus AS-i.

En cualquier caso se debe respetar la polaridad desde la fuente de alimentación AS-i.

En el caso de los esclavos con conexión por bornes, tan solo habrá que fijarse en la parte frontal del mismo en donde aparece la conexión que corresponde a cada borne.

Para en caso de esclavos con el sistema vampiro de la nueva versión, se observa, en la parte posterior del mismo, la existencia de una serie de clavos que son los que pinchan sobre el cable perfilado, dependiendo del esclavo elegido hay dos tipos, estos son los que necesitan:

- Únicamente cable de datos (cable amarillo).
- Cable de datos (cable amarillo), más cable de alimentación auxiliar (color negro).

Al fijarse en la parte posterior de un esclavo se puede observar la disposición de los diferentes pinchos para clavar en el cable, tanto del bus AS-i (amarillo) como para el de alimentación auxiliar (cable negro).

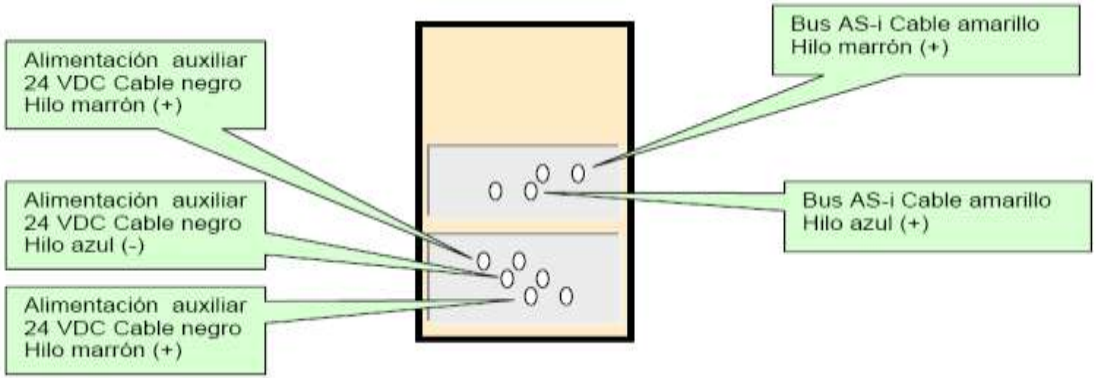


Fig. 5.24. Conexión de los esclavos al cable AS-i por método vampiro.

A continuación y mediante gráficos se refleja las diferentes posibilidades de conexión con las que puede contar un esclavo AS-i.

- ✓ Esclavos con necesidad del cable de datos únicamente.

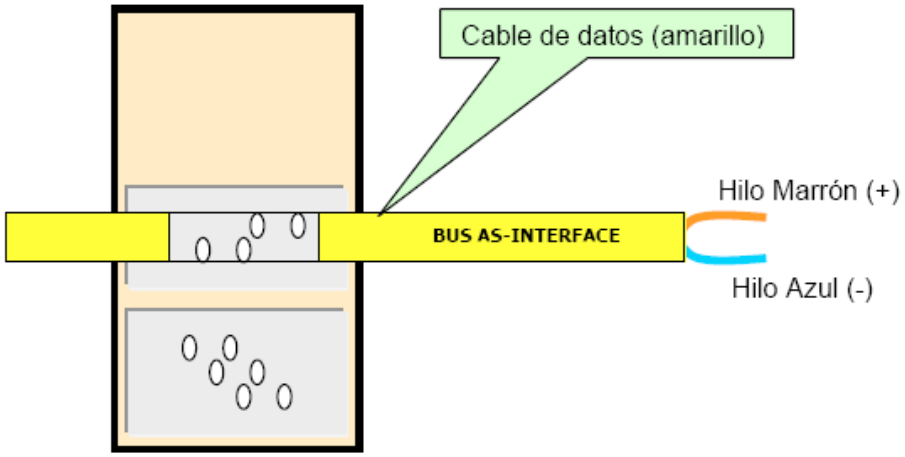


Fig. 5.25. Posición del cable AS-i para conexión a esclavo.

✓ Esclavos con necesidad del cable de datos y cable de alimentación auxiliar.

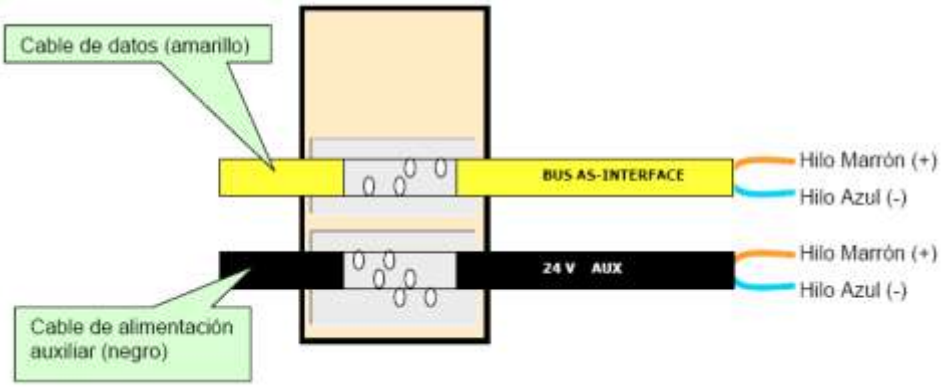


Fig. 5.26. Posición del cable AS-i y de alimentación para conexión a esclavo.

Una vez que se conoce el tipo de conexión que se debe realizar a cada esclavo, se procede a su montaje, para ello se siguen los siguientes pasos:

a) Pasar por la vía correspondiente tanto el cable de bus (amarillo), como el cable de alimentación auxiliar (negro) sobre la base del esclavo.

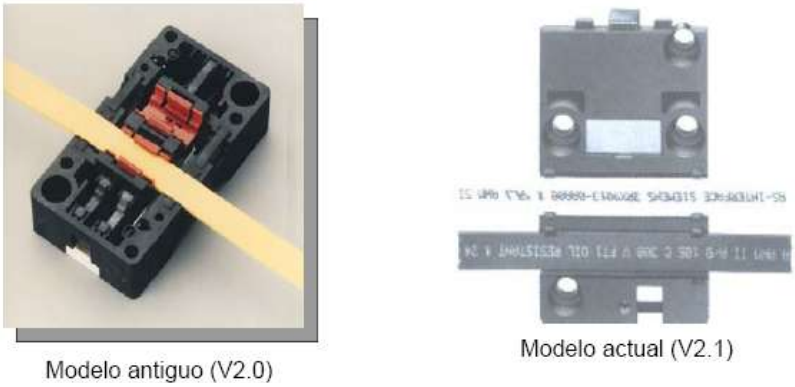


Fig. 5.27. Comparación de conexión de cable AS-i entre V2.0 y V2.1.

b) Colocar el esclavo sobre la base, apretando levemente sobre la base.

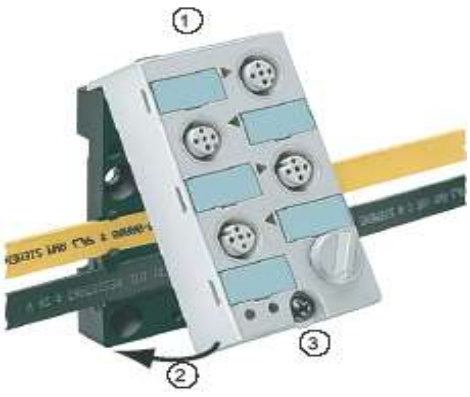


Fig. 5.28. Colocación de base.

c) Apretar los tornillos para fijar el esclavo a la base y para asegurar las conexiones tipo vampiro sobre los cables.

Fig. 5.29. Fijación de cable con esclavo.

5.5 INSTALACIÓN Y CONFIGURACIÓN DEL MAESTRO AS-I.

5.5.1 FUNCIONAMIENTO BÁSICO DEL CP 243-2.

El modulo CP 243-2 es una tarjeta maestro AS-i para el PLC S7-200, posee protección IP20. Ambos, el CP y el PLC pueden ser usados independientemente del otro.

El CP 243-2 se puede instalar en el sistema de automatización S7–200 (CPUs 22x) en todos los slots (ranuras) previstos para módulos de expansión.

El CP 243–2 es considerado por la CPU S7–22x como dos módulos de expansión (un módulo digital 8ED/8SD y un módulo analógico 8EA/8SA).

La técnica en que está estructurado el CP 243–2 equivale a la de un módulo de expansión estándar para S7–200.

5.5.2 PANEL FRONTAL.

Zona de conexiones, de visualización y de mando.

A través del panel frontal se tiene acceso a todos los elementos de conexión, visualización/ indicación y mando del CP 243–2. Las zonas de conexiones y mando están cubiertas por una tapa frontal durante el funcionamiento.

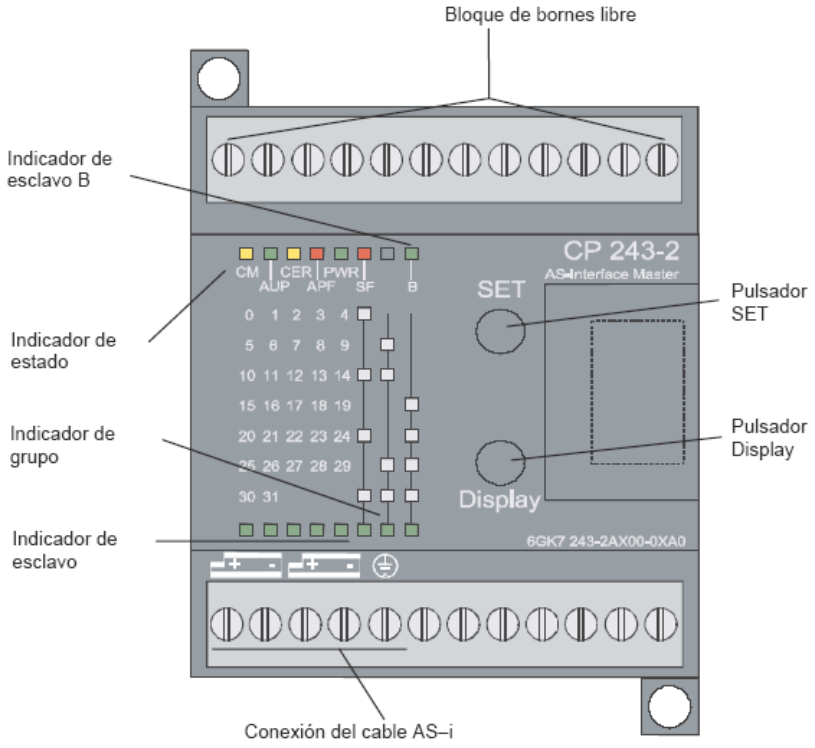


Fig. 5.30. Panel frontal de CP 243-2.

Conexiones:

El CP 243-2 posee las siguientes conexiones:

- ✓ Dos conexiones al cable AS-i (Puenteadas internamente).
- ✓ Una conexión para tierra de funcionamiento.

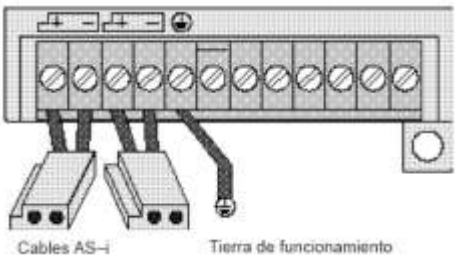


Fig. 5.31. Conexión de cable AS-i a modulo CP 243-2

5.5.3 MODOS DE FUNCIONAMIENTO DEL CP 243–2.

5.5.3.1 Modo estándar.

El programa de usuario accede a los datos útiles de los esclavos AS–i y a los datos de diagnóstico del CP 243–2. La programación es sencilla y resulta suficiente para la mayoría de las soluciones de automatización. En el modo estándar no se transmiten a los esclavos comandos ni parámetros especiales. Este modo se ajusta al perfil M0 de la especificación de maestro AS–i.

5.5.3.2 Modo extendido.

El programa de usuario utiliza la interfaz de comandos del CP 243–2. Con esto está a disposición del programador de PLC todo el volumen de funciones posible con el sistema AS-i. En especial están disponibles las llamadas de maestro AS–i (p. ej., parametrización de los esclavos). Este modo se ajusta al perfil M1 de la especificación de maestro AS–i.

5.5.4 SIGNIFICADO DE LOS LEDs.

En la parte frontal del CP 243–2 se encuentran dos filas de diodos luminiscentes. Los diodos CM, AUP, CER, APF, PWR y SF de la fila superior representan la indicación de estado. El LED B señaliza esclavos B. Se enciende adicionalmente cuando, estando activada la indicación de esclavos, se indican esclavos B.

Los primeros cinco diodos de la fila inferior indican los esclavos conectados (indicación de esclavos). Los tres diodos restantes señalizan el grupo de esclavos.

Si están apagados todos los diodos de indicación de grupos, está activa la indicación de estado, lo que significa que los diodos CM, AUP, CER, APF, PWR y SF indican el estado del CP 243–2. Si está encendido al menos uno de los diodos de indicación de grupos, se apaga la indicación de estado (excepción: diodo “PWR” encendido) y está activa la indicación de esclavos.Los diodos tienen el siguiente significado:

Diodo (color)	STATUS	Significado
CM (amarillo)	Configuration Mode	Esta indicación señaliza el modo de funcionamiento del CP 243–2. • Indicador encendido: modo de configuración • Indicador apagado: modo protegido El modo de configuración se necesita sólo para la puesta en servicio del CP 243–2. En el modo de configuración, el CP 243–2 activa todos los esclavos AS–i conectados e intercambia datos con ellos.
AUP (verde)	Autoprog available	En el modo protegido del CP 243–2, indica que es posible una programación automática de dirección de un esclavo AS–i. La programación automática de dirección simplifica la sustitución de un esclavo AS–i averiado en el cable AS–i
CER (amarillo)	Configuration Error	El diodo indica si la configuración de esclavos identificada en el cable AS–i coincide con la configuración teórica (LPS) del CP 243–2. En caso de diferencias se enciende el indicador CER. El indicador CER se enciende en los siguientes casos: • si un esclavo AS–i configurado no está en el cable AS–i (p. ej. fallo del esclavo). • si en el cable AS–i está un esclavo AS–i que no se ha configurado antes. • si un esclavo AS–i conectado tiene datos de configuración (configuración de E/S, ID–Code, Extended ID1–Code, Extended ID2–Code) diferentes a los del esclavo AS–i configurado en el CP 243–2. • si el CP 243–2 se encuentra en la fase Offline.
APF (rojo)	AS–i Power Fail	Indica que la tensión suministrada por la fuente de alimentación AS–i al cable AS–i es demasiado baja o falta.
PWR (verde)	Power	El diodo PWR (Power) señaliza que el CP 243–2 es abastecido de tensión.
SF (rojo)	Error del sistema.	El diodo se enciende si: • el CP 243–2 detecta un error interno (p. ej. defecto en EEPROM). • el CP 243–2 no puede realizar de momento el cambio de modo exigido durante un accionamiento de pulsador (p. ej. existe un esclavo AS–i con la dirección 0).

Tabla 5.11. Significado de los diodos LED del CP 243-2.

5.5.5 INDICACIÓN DE ESCLAVOS.

El cambio al modo de indicación de esclavos tiene lugar apretando el pulsador DISPLAY; el paso de un grupo a otro se consigue apretando de nuevo el pulsador DISPLAY.

5.5.5.1 Propiedades de la indicación de esclavos.

Si el CP 243–2 se encuentra en el modo de configuración, se indican todos los esclavos AS–i detectados. Si se encuentra en el modo protegido, se indican todos los esclavos AS–i activos.

En el modo protegido, los esclavos AS–i fallados o existentes pero no configurados son indicados al destellar el diodo correspondiente.

5.5.5.2 Estados de indicación en detalle.

Los esclavos AS–i son indicados en grupos de cinco. Los tres diodos señalizadores de grupo indican con una codificación binaria, cuál de los grupos de cinco es visualizado. Los cinco diodos de la indicación de esclavos indican entonces los esclavos AS–i detectados o activos dentro de este grupo.

Para constatar qué esclavos están activos, busque el grupo de cinco (la fila) que presente casillas en correspondencia con los diodos de grupo encendidos. Los diodos de la iniciación actualmente encendidos determinan qué esclavos está justamente activos dentro de ese grupo.

Si se visualiza un grupo de esclavos B, se enciende adicionalmente el diodo “B”.

5.5.5.3 Identificación de esclavo conectado mediante los LED.

Puede ver lo siguiente en la representación:

- el 2º diodo de grupo esta encendido, lo que significa la 2ª fila de arriba (equivale a 21 = 2DEC; 2º grupo de cinco; esclavos 5–9).

- Si dentro de la indicación de esclavos están encendidos además el 2º y el 4º diodo, esto significa que los esclavos 6 y 8 están activos.

- Si además está encendido el diodo “B”, esto significa en este ejemplo que están activos los diodos 6B y 8B.

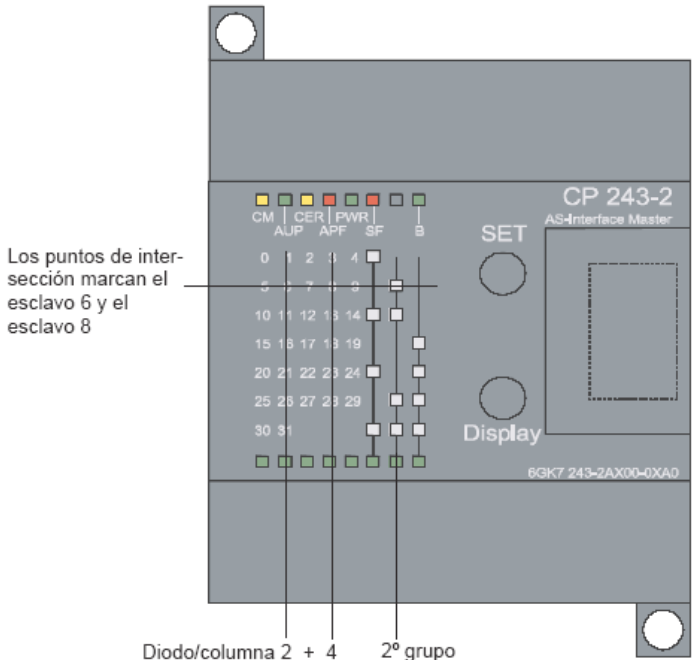


Fig. 5.32. Verificación de esclavos conectados a la red.

5.5.6 MODOS DE OPERACIÓN.

El CP 243-2 dispone de dos modos de operación, estos son:

- Modo configuración.
- Modo protegido.

5.5.6.1 Modo configuración.

El modo de configuración sirve para la puesta en servicio de una instalación AS–i. Si el CP 243–2 se encuentra en el modo de configuración (indicador CM encendido), puede intercambiar datos con cualquier esclavo AS–i conectado al cable AS–i (a excepción del esclavo AS–i con la dirección ‘0’). Nuevos esclavos AS–i agregados son detectados inmediatamente por el maestro, son activados y se incorporan al intercambio cíclico de datos.

Una vez terminada la fase de puesta en servicio, el CP se puede conmutar al modo protegido por medio del pulsador SET. Con esto se configuran al mismo tiempo los esclavos AS–i activos en ese momento. Los datos siguientes se almacenan entonces en forma no volátil en el CP:

- Las direcciones de los esclavos AS–i.
- Los ID–Codes (ID–Code, Extended ID1–Code, Extended ID2–Code).
- La configuración de E/S (entradas/salidas).
- Los parámetros actuales de los esclavos.

5.5.6.2 Modo protegido.

En el modo protegido, el CP 243-2 sólo intercambia datos con los esclavos AS-i configurados. “Configurado” significa que las direcciones de esclavos y los datos de configuración almacenados en el CP 243-2 coinciden con los valores de los esclavos AS-i existentes.

5.5.7 PREPARAR LA CONFIGURACIÓN DEL CP 243-2.

Asegure los estados siguientes:

- La CPU S7–22x tiene que estar en STOP (PLC_RUN=0).
- El CP 243–2 y todos los esclavos AS–i han de estar conectados a AS–Interface y tienen que ser abastecidos de tensión por la fuente de alimentación AS–i.

5.5.7.1 Realizar la configuración.

1. Con el pulsador DISPLAY, conmute el visualizador del CP al modo “Indicación de estado” (estado básico).
2. Compruebe si el CP se encuentra en el modo de configuración. (Diodo ”CM” encendido). De no ser así, conmute el CP 243–2 al modo de configuración con el pulsador SET.
3. Por conmutación a la indicación de esclavos con el pulsador DISPLAY puede comprobar si existen todos los esclavos conectados a AS–Interface.
4. Apriete el pulsador SET. Con esto se configura el CP 243–2.
   Al mismo tiempo, el CP se conmuta al modo protegido; se apaga el diodo ”CM”.
   El diodo “CER” también se apaga, ya que, después de configurar, la “configuración teórica” almacenada en el CP coincide con la “configuración real” existente en AS–Interface.
5. Pasar la CPU del PLC de nuevo al modo RUN/RUN-P, observando que si todo es correcto no se tendrá ningún fallo de sistema (indicador SF apagado) ni en la CPU ni en el CP.

Otras consideraciones:

- Tan solo es posible cambiar del modo de configuración al modo protegido si no hay conectado al bus AS-i ningún esclavo AS-i con la dirección ‘0’.
- Si está conectado un esclavo con la dirección ‘0’, al accionar el pulsador SET el CP enciende el led SF (System fail) de error del sistema.
- Si se realiza una configuración por pulsador cuando no hay tensión en el bus AS-i, el CP enciende el led APF de fallo en la tensión del bus AS-i.

5.5.8 SEÑALIZACIÓN DE ESCLAVOS.

Una vez realizado todo el montaje del cableado de la red, y haber configurado el maestro AS-i de forma que reconozca a todos los esclavos conectados y conocer si existe algún error en alguno de ellos o del bus o del cable de alimentación auxiliar, puede que aparezca algún error que puede ser reflejado tanto en el CP 243-2 como en el propio esclavo.

Si existiera algún error en alguno de los esclavos localizados, el indicador del número correspondiente a la dirección del esclavo erróneo se iluminará de forma intermitente, esto puede suceder en casos como:

- Esclavos con dirección duplicada.
- Esclavos defectuosos.

Si apareciera algún error, antes de continuar se deberá reparar el mismo y que se deberá actuar de forma diferente según sea el error producido.

También puede darse el caso de que al accionar el pulsador SET el maestro AS-i no pase al modo configuración (CM), si esto sucediera es que hay algún esclavo con la dirección "0", por tanto se deberá de modificar la dirección con la consola antes de accionar de nuevo el pulsador SET del maestro AS-i.

En el caso de existir un error en el esclavo, éste lo indicará mediante sus propios indicadores luminosos, indicadores que pueden ser diferentes en cada esclavo:

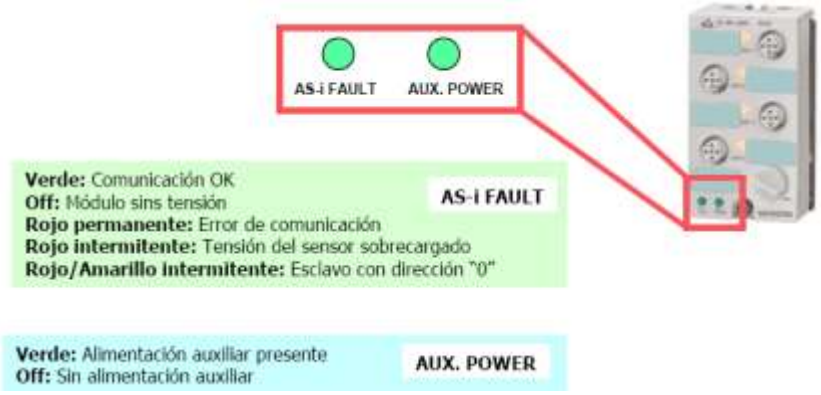


Fig. 5.33. Indicadores de error en los esclavos.

Esto indica que, como norma general los diferentes indicadores de los que disponga el esclavo deben lucir de color verde, indicando que no existen errores en el esclavo.

5.6 CONEXIONADO DE DISPOSITIVOS DE E/S ESTÁNDAR A LOS ESCLAVOS AS-I.

Según sea la aplicación a realizar se necesita una serie de componentes que se tendrán que conectar a los esclavos, elementos como pueden ser:

- Pulsadores.
- Detectores.
- Finales de carrera.
- Pilotos de señalización.
- Electro válvulas.
- …/…

La distribución de contactos de los conectores hembra M12 en módulos de usuario cumple la norma IEC 947-5-2.

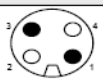
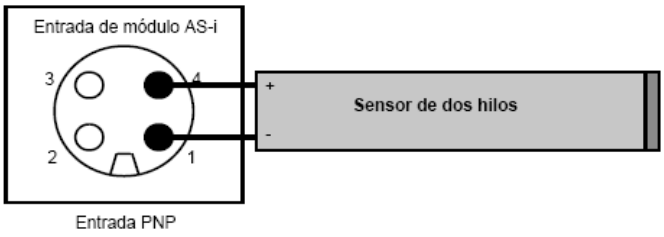
Entrada PNP estándar (hembra)		Salida PNP estándar (hembra)	
	1: "+" Alimentación. 2: Entrada de señal (= 4). 3: "-" Alimentación. 4: Entrada de señal (= 2).		1: n.e. 2: n.e. 3: "-" Alimentación. 4: Salida conmutada
Entrada 2 bit PNP (hembra)		**Distribuidor AS-i (hembra)**	
	1: "+" Alimentación. 2: Entrada de señal 2. 3: "-" Alimentación. 4: Entrada de final 1.		1: "+" Conexión AS-i. 2: n.e. 3: "-" Conexión AS-i. 4: n.e.
Alimentación auxiliar externa (macho)			
	1: "+" Alimentación. 2: n.e. 3: "-" Alimentación. 4: n.e.		n.e. : no existe

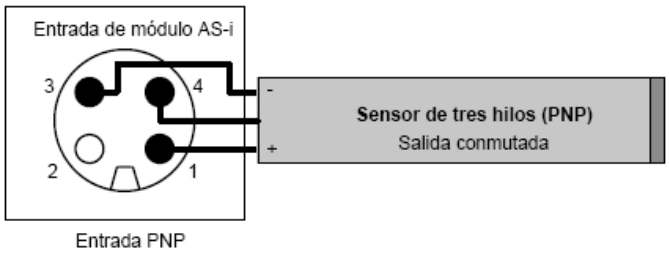
Fig. 5.34. Distribución de conectores hembra M12 en módulos de usuario.

5.6.1 CONEXIÓN DE SENSORES/ACTUADORES ESTÁNDAR MEDIANTE MÓDULOS AS-I.

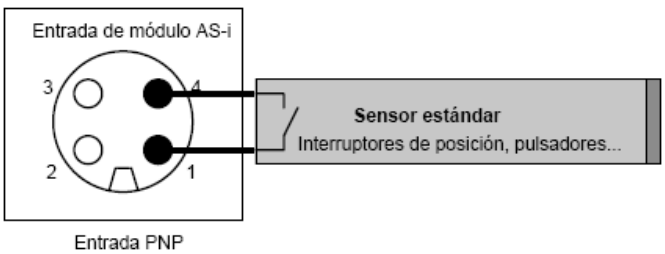
- Sensor de dos hilos PNP.



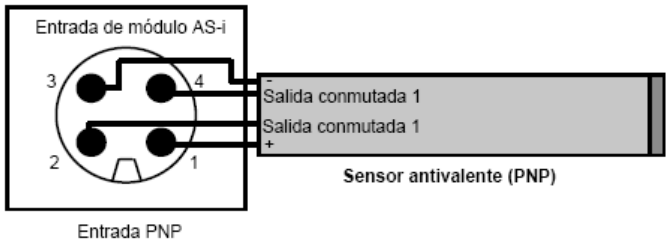
- Sensor de tres hilos PNP.



- Sensor estándar.

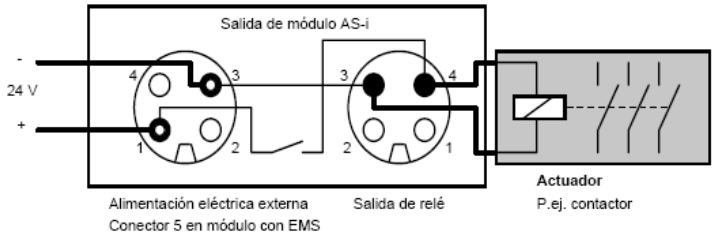


- Sensor antivalente (PNP que realiza la operación lógica O exclusiva).



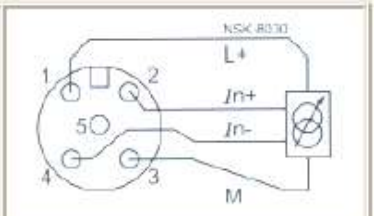
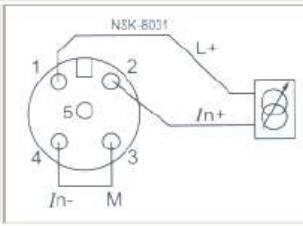
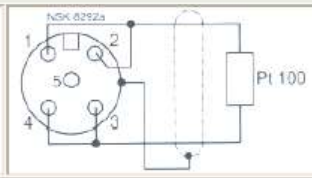
Este sensor tan sólo puede conectarse al módulo de entradas 2 x 2 entradas. No es posible la conexión a un módulo de 4E.

- Actuador estándar.



En módulos con EEMS (interface electromecánica ampliada) desaparece el conector 5 para la alimentación eléctrica externa. Esta se alimenta a través del módulo de acoplamiento.

Para los módulos esclavos analógicos las diferentes conexiones posibles pueden ser:

Módulo de entradas analógico de corriente para sensor a 4 hilos.	
Módulo de entradas analógico de corriente para sensor a 2 hilos. Puente de In- a M(3) también ajustable internamente con parametrización	
Módulo de entradas analógico de termorresistencia para sensor a 2 hilos.	

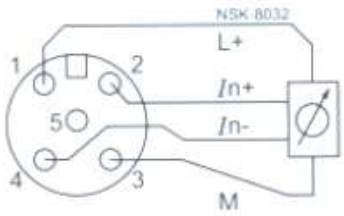
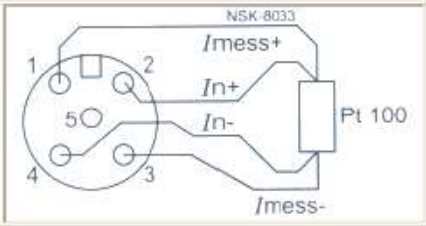
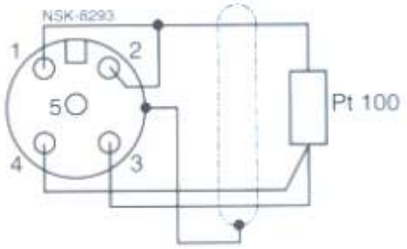
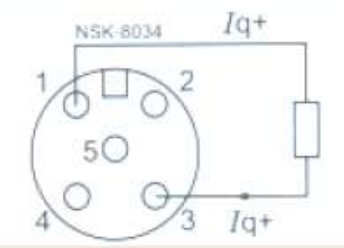
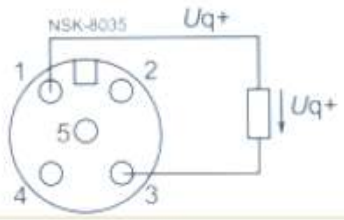
Módulo de entrada analógico de tensión, sensor a 4 hilos. Pin 1 es la alimentación de sensores DC 24 V. Pin3 es la conexión a masa de la alimentación (masa de la parte analógica) máx. 50 mA para ambos canales. Pin 5 es para la pantalla.	
Módulo de entradas analógico de termorresistencia para sensor a 4 hilos.	
Módulo de entrada analógico por termorresistencia, sensor a 3 hilos. Pin 1 es Imes+ Pin 2 es la alimentación de sensores Pin 3 es Imes- Pin 4 es la conexión a masa de la alimentación.	
Módulo de salida analógico de corriente.	
Módulo de salida analógico de tensión. Pin 5 es para la pantalla	

Fig. 5.35. Conexiones posibles para esclavos analógicos.

5.7 CREACIÓN DE UN PROYECTO EN STEP 7 MICRO/WIN.

5.7.1 AUTÓMATA PROGRAMABLE S7-200.

El PLC de la serie S7-200 necesita de una tarjeta conectada al propio bus de la CPU como un maestro AS-i, ej. el CP 243-2, que controla todos los esclavos AS-i conectados al mismo.

El intercambio de datos entre el programa de usuario y los datos de los esclavos AS-i a través del CP 243-2, depende del tipo de esclavo que se tenga en el bus As-i, éstos pueden ser:

- Esclavos estándar o esclavos A de tipo binario a través de la periferia de E/S de la CPU del PLC.
- Esclavos B de tipo binario por lectura o escritura mediante un bloque de función.
- Esclavos AS-i analógicos para la lectura o escritura mediante un bloque de función.

A través del programa de usuario, se accede a valores binarios de esclavos AS–i estándar o de esclavos a través de determinados comandos de periferia de STEP 7 Micro/Win.

El CP 243–2 asigna cuatro bits (llamado nibble) a cada esclavo estándar o A conectado al cable AS–i. El PLC puede acceder a este nibble con escritura (datos de salida de esclavo) y con lectura (datos de entrada de esclavo). De este modo se pueden activar también esclavos bidireccionales con E/S.

Los cuatro primeros bits de entrada (primer nibble) están reservados. Los cuatro primeros bits de salida (primer nibble) no tienen relevancia para el CP 243–2.

5.7.1.1 Configuración del hardware.

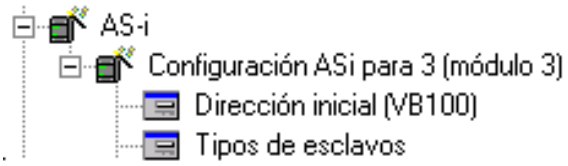
Para parametrizar el bus AS-i en un proyecto de automatización mediante el uso de la red AS-i se utiliza el asistente AS-i que se incluye en la versión 3.2 de STEP 7 Micro/Win, así como, en las versiones posteriores.

Para la utilización de este asistente se deben seguir una serie de pasos los cuales se brindan a continuación.

5.7.2 ASISTENTE AS-I DE STEP 7 MICRO/WIN.

Para utilizar el asistente AS-i, proceda de la manera siguiente:

1. Elija el comando de menú Herramientas > Asistente AS-i, o bien, en la barra de navegación, haga clic en el icono del asistente AS-i  , o bien, desplácese a la carpeta "Asistentes" en el árbol de operaciones y abra el asistente o una configuración existente.



2. Para acceder a la siguiente pantalla del asistente AS-i, haga clic en el botón "Siguiente>".

3. Para concluir la configuración en el asistente, haga clic en el botón "Finalizar".

En la pantalla inicial puede indicar si desea:

Cambiar las direcciones de esclavos AS-i

Si desea configurar la red de esclavos AS-i, programando las direcciones almacenadas en los esclavos AS-i. Para programar las direcciones de los esclavos se deberá haber establecido la comunicación entre STEP 7-Micro/Win, una CPU S7-200 conectada a un módulo maestro CP243-2 AS-i y la red de esclavos AS-i.

Configurar el proyecto para mapear los esclavos AS-i

El asistente AS-i ayuda a crear la lógica del proyecto STEP 7-Micro/Win necesaria para transferir datos entre el programa de usuario y los esclavos AS-i. Además, asigna símbolos mapeados a las direcciones AS-i de las entradas y salidas.

Esta sección del asistente se puede utilizar online u offline (es decir, sin estar conectada una CPU S7-200, un CP243-2 o una red de esclavos). Si el asistente se utiliza online, puede proporcionar información del módulo CP243-2 y efectuar comparaciones con la red AS-i.

Fig. 5.36. Asistente AS-i en STEP 7 Micro/Win.

5.7.2.1 Cambiar direcciones de esclavos AS-i.

Configure la red de esclavos AS-i programando las direcciones de red almacenadas en los esclavos AS-i. Para programar las direcciones de los esclavos se deberá haber establecido la comunicación entre STEP 7-Micro/Win, una CPU S7-200 conectada a un módulo maestro CP243-2 AS-i y la red de esclavos AS-i.

- *Seleccionar módulo:* Indique el módulo CP243-2 y la red AS-i a la que está conectado el esclavo. La CPU 222 S7-200 puede controlar dos módulos AS-i CP243-2, en tanto que las CPUs 224 y 226 pueden controlar hasta tres módulos AS-i CP243-2. Cada módulo CP243-2 puede controlar una red AS-i.

- *Seleccionar esclavo:* El esclavo debe existir en la red antes de poder cambiar su dirección. Seleccione el esclavo cuya dirección desea editar. En la lista desplegable figuran todos los esclavos conectados al módulo maestro.

- *Nueva dirección:* Puede cambiar :
  Una dirección de esclavo AS-i estándar (0 - 31) por otra dirección estándar (0 - 31).
  En este caso se desactivarán las casillas de verificación "Dirección A" y "Dirección B".

  Una dirección A ampliada (A0 - A31) por otra dirección A ampliada (A0 - A31)
  En este caso, no active la casilla de verificación "Dirección B".

  Una dirección A ampliada (A0 - A31) por una dirección B ampliada (B1 - B31)
  En este caso, active la casilla de verificación "Dirección B".

  Una dirección B ampliada (B1 - B31) por otra dirección B ampliada (B1 - B31)
  En este caso, no active la casilla de verificación "Dirección A".

  Una dirección B ampliada (B1 - B31) por una dirección A ampliada (A0 - A31)
  En este caso, active la casilla de verificación "Dirección A".

- *Botón "Cambiar":* Haga clic en el botón "Cambiar" para editar la dirección antigua y guardar la nueva dirección de red en el esclavo indicado.

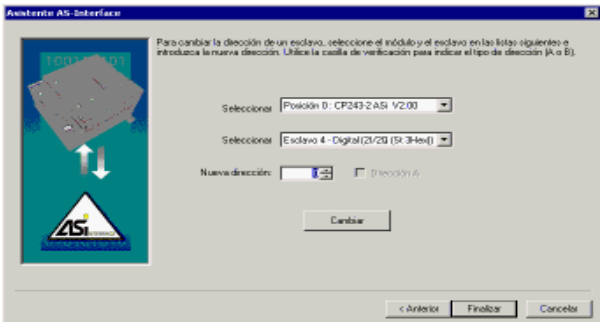


Fig. 5.37. Cambiar direcciones de esclavos AS-i.

5.7.2.2 Mapear esclavos AS-i.

- El asistente AS-i compila el programa y conmuta al modo de direccionamiento simbólico antes de comenzar con el proceso de configuración. Asimismo, comprueba el tipo de CPU seleccionado para el proyecto. El tipo y la versión de CPU actual debe soportar módulos AS-i.

- El asistente AS-i está disponible para la 2a generación de CPUs que soportan las operaciones BIR y BIW (MOV_BIR y MOV_BIW en KOP/FUP).

- Si el proyecto contiene configuraciones AS-i creadas con STEP 7 Micro/Win, deberá indicar que desea modificar una de ellas, o bien crear una nueva configuración antes de continuar con el PASO 4.

El asistente le solicita que introduzca las informaciones siguientes:

1 Definir la configuración AS-i a editar.
2 Transferir una configuración existente.
3 Comparar una configuración existente con la red AS-i actual.
4 Indicar el slot del módulo AS-i.
5 Definir las direcciones del módulo.
6 Indicar los tipos de esclavos.
7 Indicar los esclavos digitales.
8 Indicar los esclavos analógicos.
9 Asignar memoria a la configuración.
10 Generar los componentes del proyecto.

PASO 1 Definir la configuración AS-i a editar.

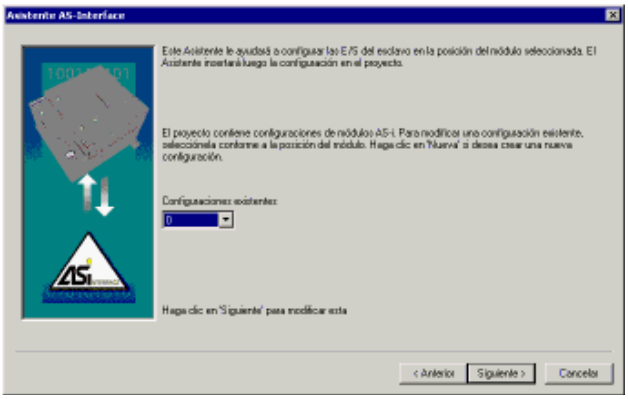


Fig. 5.38. Definir configuración a editar.

Indique la configuración que desea editar.

Si no existen configuraciones, este cuadro de diálogo no aparecerá, por lo que deberá continuar con el PASO 4. Si el proyecto contiene configuraciones AS-i, podrá modificar una de ellas seleccionando la respectiva posición del módulo, o bien crear una nueva configuración. Si desea crear una nueva configuración, continúe con el PASO 4.

PASO 2 Transferir una configuración existente.

Si desea editar una configuración existente, puede transferirla a un módulo diferente, o bien borrarla del proyecto.

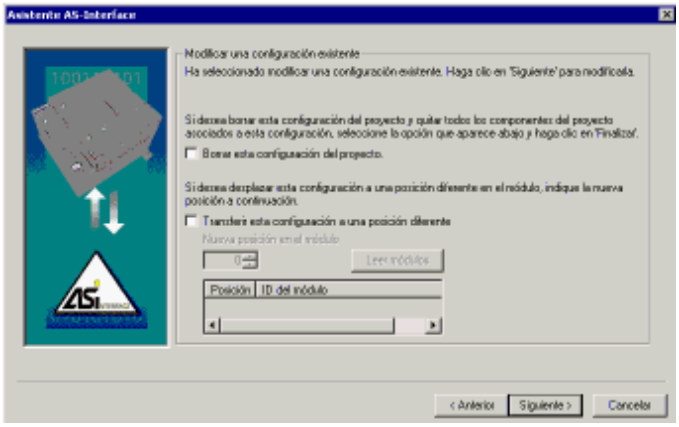


Fig. 5.39. Transferir una configuración existente.

PASO 3 Comparar una configuración existente con la red AS-i actual.

Si no desea desplazar una configuración existente a un módulo diferente, podrá comparar la configuración existente con la configuración online actual. A este efecto, haga clic en "Siguiente>" y después en el botón "Comparar online".

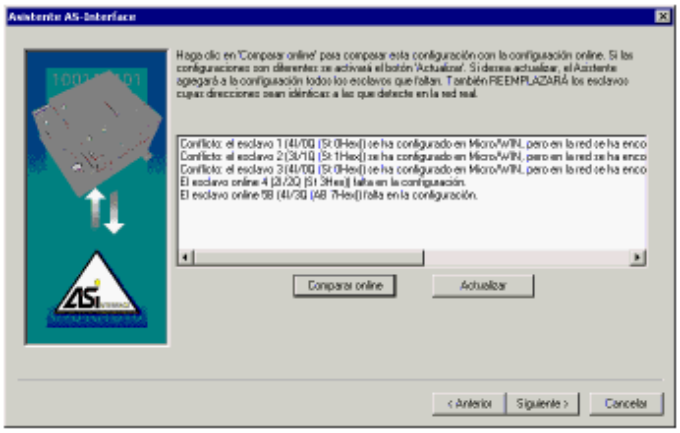


Fig. 5.40. Comparar una configuración existente con la red actual.

Comparar online:
Si hace clic en el botón "Comparar online", la configuración AS-i contenida en el proyecto se comparará con los esclavos físicos (es decir, los que forman parte de la red AS-i real).

La función "Comparar online" indica si hay:

- Esclavos configurados que concuerdan con los esclavos físicos.
- Esclavos configurados diferentes a los esclavos físicos.
- Esclavos configurados en el proyecto pero que faltan en la red.
- Esclavos físicos no contenidos en la configuración del proyecto.

Actualizar:

Si hace clic en el botón "Actualizar", el asistente AS-i actualizará la configuración del proyecto, incluyendo en ella los esclavos físicos.

Efectos de la función "Actualizar":

- Si un esclavo no está contenido en el proyecto, se agregará a la configuración.
- Si un esclavo físico difiere del esclavo configurado, se reemplazará el esclavo contenido en el proyecto.
- Si los esclavos concuerdan, no se efectuará ningún cambio en la configuración.
- Si los offsets de la dirección E/S del módulo son diferentes, se actualizará el proyecto.

PASO 4 Indicar el slot del módulo AS-i

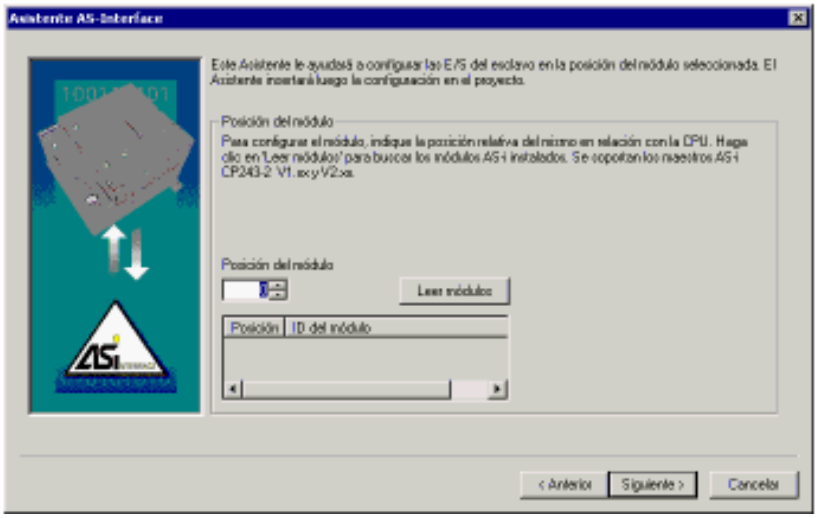


Fig. 5.41. indicar el slot del modulo AS-i.

Haga clic en el botón "Leer módulos" para leer automáticamente la posición de los módulos CP243-2 AS-i instalados.

Si la comunicación se establece correctamente, el asistente mostrará una lista de todos los módulos AS-i conectados a la CPU. Puede elegir uno de estos módulos "online", o bien seleccionar una posición de módulo que no figure en la lista (es decir, un módulo "offline").

Al seleccionar uno de los módulos online, las direcciones de E/S de ese módulo se cargarán en la siguiente pantalla del asistente. Éste intenta comunicarse entonces con el módulo online seleccionado para obtener información adicional de configuración.

PASO 5 Definir las direcciones del módulo.

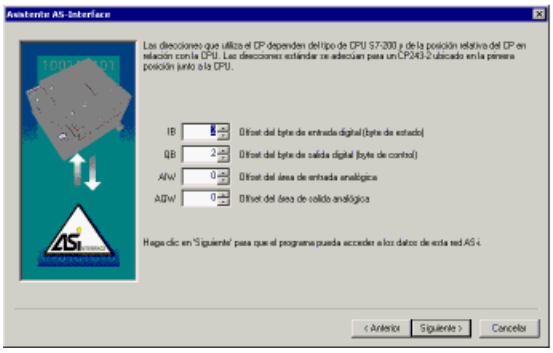


Fig. 5.42. Definir las direcciones del modulo.

Nota: Si selecciona un módulo online en el paso anterior, las direcciones ya se habrán cargado y los demás campos aparecerán atenuados.

Para definir la dirección inicial de las áreas de direccionamiento deberá indicar:

- El tipo de CPU S7-200 utilizado.
- El slot del CP243-2 en la CPU S7-200.
- Los demás tipos de módulos instalados en la CPU S7-200.

PASO 6 Indicar los tipos de esclavos.

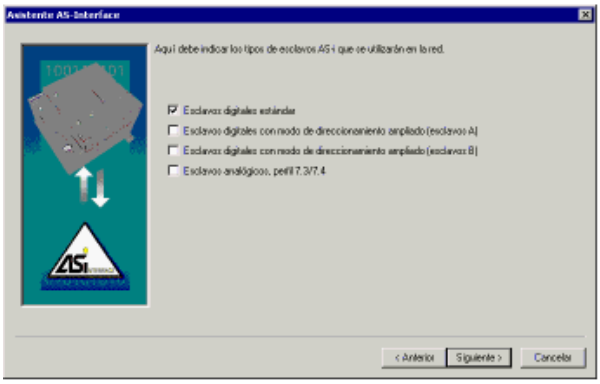


Fig. 5.43. Indicar los tipos de esclavos.

- Si desea configurar un módulo online, el módulo CP determina los tipos de esclavos integrados en la red. Por tanto, no es necesario que introduzca esta información manualmente.
- Si desea configurar un módulo offline deberá indicar los tipos de esclavos de la red.
- Si desea configurar esclavos analógicos, active la casilla de verificación "Esclavos analógicos, perfil 7.3/7.4" para que la pantalla de configuración de estos esclavos se visualice luego en el asistente.

Puede seleccionar un número cualquiera de tipos de esclavos que desee controlar en la red. Elija una de las opciones siguientes:

- Esclavos digitales estándar—antiguo modo de direccionamiento (0-31)

- Esclavos digitales con modo de direccionamiento ampliado (esclavos A)—nuevo modo de direccionamiento ampliado para esclavos A (0A-31A)
- Esclavos digitales con modo de direccionamiento ampliado (esclavos B)—nuevo modo de direccionamiento ampliado para esclavos B (1B-31B)
- Esclavos analógicos, perfil 7.3/7.4—soporte del nuevo perfil 7.3/7.4

PASO 7 Indicar los esclavos digitales.

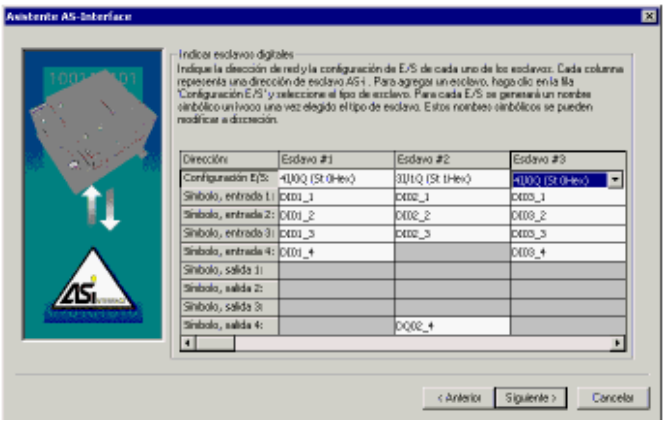


Fig. 5.44. Indicar los esclavos digitales.

Puede editar la tabla de los esclavos digitales conectados a la red si ha activado la casilla "Esclavos digitales" en la pantalla anterior del asistente.

La tabla contiene una columna para todas las direcciones posibles de los esclavos. La primera fila de la tabla muestra los diferentes tipos de esclavos. En las filas restantes se visualizan las configuraciones de E/S de los esclavos seleccionados, pudiendo introducir allí nombres simbólicos para esas E/S.

En modo online se visualizan en la tabla los esclavos digitales configurados actualmente en la red AS-i. No es necesario que edite la configuración de los esclavos en esta tabla. Sin embargo, puede agregar esclavos o modificar los nombres simbólicos predefinidos que se han asignado a todas las E/S digitales configuradas en la red.

En modo offline no se visualizan informaciones acerca de los esclavos digitales. En la tabla debe indicar los tipos de esclavos digitales conectados a la red y definir la configuración de E/S de cada uno de ellos. Cuando agregue esclavos a la tabla, se asignarán nombres simbólicos predefinidos a todas las E/S digitales configuradas en la red. Puede editar los nombres simbólicos que proponga el asistente.

Para agregar una configuración de esclavo:

1. Haga doble clic en la celda de configuración de E/S correspondiente a la dirección del esclavo.
2. Seleccione en la lista la configuración de esclavo deseada.

Notas:

- Los nombres simbólicos repetidos se indican con un subrayado verde ondulado. Los nombres que contengan símbolos no válidos (p. .ej. !@) se señalan en color rojo. No será posible avanzar a la pantalla siguiente si no se han eliminado los nombres repetidos, no válidos o vacíos.
- Los nombres simbólicos genéricos de las E/S digitales se asignan de la siguiente forma:

Esclavos digitales estándar *Sintaxis de las direcciones:*
Símbolos de las entradas: DI<Nº de slot #><Dirección del esclavo #>_<Entrada #>
Símbolos de las salidas: DQ<Nº de slot #><Dirección del esclavo #>_<Salida #>

Esclavos A digitales
Símbolos de las entradas: DI<Nº de slot #><Dirección del esclavo #>A_<Entrada #>
Símbolos de las salidas: DQ<Nº de slot #><Dirección del esclavo #>A_<Salida #>

Esclavos B digitales
Símbolos de las entradas: DI<Nº de slot #><Dirección del esclavo #>B_<Entrada #>
Símbolos de las salidas: DQ<Nº de slot #><Dirección del esclavo #>B_<Salida #>

Ejemplo:
El símbolo de la primera entrada del esclavo #1A (esclavo A) en el slot 0 es:
DI01A_1

El símbolo de la tercera entrada del esclavo #15 (esclavo estándar) en el slot 0 es:
DI015_3

El símbolo de la segunda salida del esclavo #1B (esclavo B) en el slot 0 es:
DQ01B_2

PASO 8 Indicar los esclavos analógicos

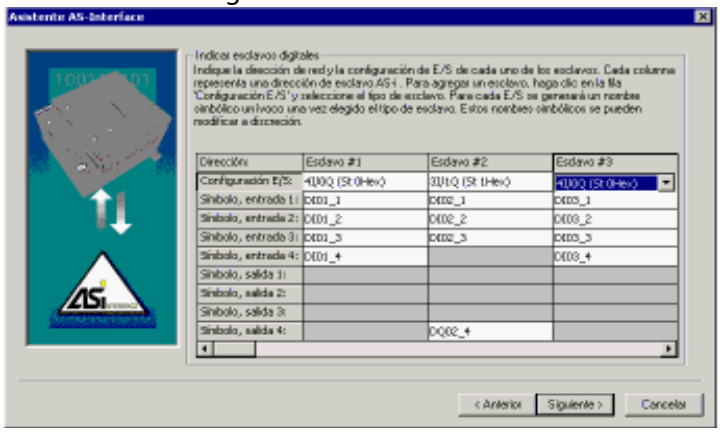


Fig. 5.45. Indicar los esclavos analógicos.

Puede editar la tabla de esclavos analógicos conectados a la red si:

- El asistente está en modo online y se han detectado esclavos analógicos en la red.
- Ha indicado que desea configurar tipos de esclavos analógicos en la pantalla anterior del asistente.

La tabla contiene una columna para todas las direcciones posibles de los esclavos. La primera fila de la tabla muestra los diferentes tipos de esclavos. En las filas restantes se visualizan las configuraciones de E/S de los esclavos seleccionados, pudiendo introducir allí nombres simbólicos para esas E/S.

En modo online se visualizan en la tabla los esclavos analógicos configurados actualmente en la red AS-i. No es necesario que edite la configuración de los esclavos en esta tabla. Sin embargo, puede agregar esclavos o modificar los nombres simbólicos predefinidos que se han asignado a todas las E/S digitales configuradas en la red.

En modo offline no se visualizan informaciones acerca de los esclavos analógicos en la tabla. Debe indicar los tipos de esclavos analógicos conectados a la red y definir la configuración de E/S de cada uno de ellos. Cuando agregue esclavos a la tabla, se asignarán nombres simbólicos predefinidos a todas las E/S analógicas configuradas en la red. Puede editar los nombres simbólicos que proponga el asistente.

Para agregar una configuración de esclavo:

1. Haga doble clic en la celda de configuración de E/S correspondiente a la dirección del esclavo.
2. Seleccione en la lista la configuración de esclavo deseada.

Nota:

Los nombres simbólicos genéricos de las E/S analógicas que propone el asistente se asignan de la siguiente forma:

Esclavos analógicos

Símbolos de las entradas: AI<Nº de slot #><Dirección del esclavo#>_<Entrada #>
Símbolos de las salidas: AQ<Nº de slot #><Dirección del esclavo#>_<Salida #>

Ejemplo:

El símbolo del primer canal de entrada del esclavo #1 en el slot 0 es: *AI01_1*
El símbolo del tercer canal de entrada del esclavo #14 en el slot 0 es: *AI014_3*
El símbolo del segundo canal de salida del esclavo #1 en el slot 0 es: *AQ01_2*

PASO 9 Asignar memoria a la configuración.

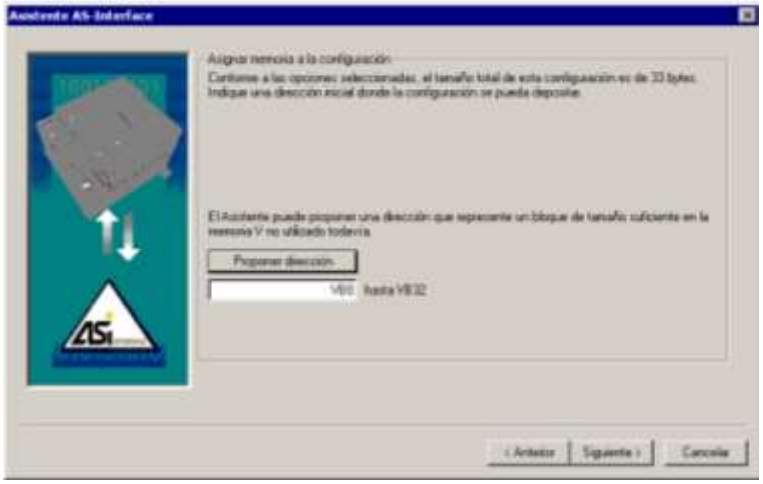


Fig. 5.46. Asignar memoria a la configuración.

El asistente AS-i crea bloques de función para el módulo AS-i que utilizan una determinada área de la memoria V de la CPU. Indique la dirección inicial a partir de la cual desea reservar un bloque de la memoria V.

El tamaño del bloque de memoria depende de lo que haya seleccionado en el asistente.

Puede determinar la dirección de la memoria V, o bien hacer clic en el botón "Proponer dirección" si desea que el asistente sugiera una dirección que represente un bloque de tamaño suficiente en la memoria V no utilizado todavía.

PASO 10 Generar los componentes del proyecto.

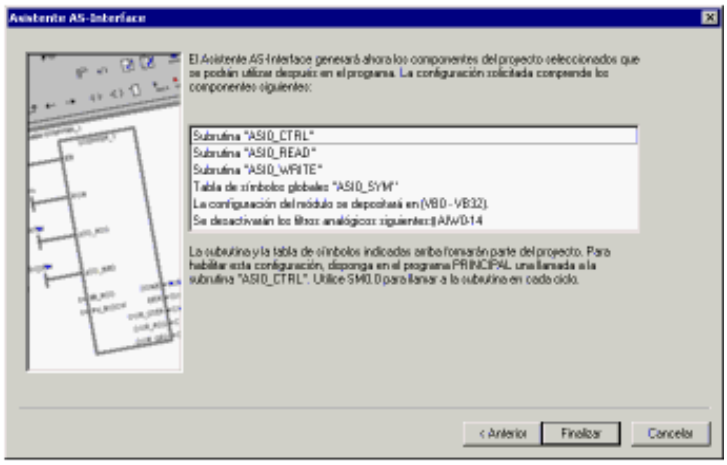


Fig. 5.47. Generar las componentes del proyecto.

El asistente AS-i genera los componentes del proyecto, poniéndolos a la disposición del programa de usuario. En la pantalla final del asistente se visualizan los componentes del proyecto solicitados. Antes de poder utilizar esos componentes es preciso cargar en la CPU la configuración del módulo AS-i.

5.7.3 COMPONENTES DE PROYECTOS AS-I.

En los nombres ASIx, el asistente reemplaza la x por el número de slot del módulo CP243-2 AS-i.

Al finalizar los pasos del asistente de configuración, este creara 3 subrutinas dentro del programa, así como una tabla de símbolos.

Subrutina ASIx_CTRL: Utilice SM0.0 para llamar a esa subrutina en cada ciclo. ASIx_CTRL se debe ejecutar en cada ciclo para poder procesar la comunicación normal en la red AS-i. Para cada módulo CP243-2 utilizado se requiere una operación ASIx_CTRL. La operación ASIx_CTRL copia los datos de E/S de los esclavos entre los módulos AS-i y la memoria V de la CPU, conforme a lo definido en la tabla de símbolos ASIx_SYM. "Error" contiene el estado del módulo.

Tabla de símbolos ASIx_SYM: La tabla de símbolos ASIx_SYM se agrega al proyecto. Utilice estos nombres simbólicos con las operaciones del programa para leer de y escribir en los esclavos AS-i mediante el control del programa.

Las subrutinas ASIx_READ y ASIx_WRITE están previstas para los usuarios con experiencia que deban poder acceder a los datos de diagnóstico y configuración en el módulo maestro CP243-2.

Subrutina ASIx_READ:

ASIx_READ (subrutina parametrizada) lee datos analógicos del banco especificado del CP y los deposita en la dirección de la memoria V indicada en DB_Ptr. Esta operación se genera conforme a la posición del módulo indicada en la configuración del asistente.

"Bank" indica el número de banco en el módulo AS-i del que se desea leer.

El parámetro "DB_Ptr" es un puntero a la dirección de la memoria V en la que se deben copiar los datos del banco. El formato de DB_Ptr es &VBx. Su longitud es 16 bytes.

Ejemplo:

&VB100 = DB_Ptr copia en VW100-VW114 las 8 palabras de datos del banco x del módulo AS-i.

Subrutina ASIx_WRITE:

La operación ASIx_WRITE (subrutina parametrizada) escribe datos analógicos de la dirección de la memoria V que indica DB_Ptr en el banco especificado del CP. Esta

operación se genera conforme a la posición del módulo indicada en la configuración del asistente.

"Bank" indica el número de banco en el módulo AS-i en el que desea escribir.

El parámetro "DB_Ptr" es un puntero a los datos de la memoria V que se copiarán en el banco indicado del CP243-2. El formato de DB_Ptr es &VBx. Su longitud es 16 bytes.

Ejemplo:

&VB100 = DB_Ptr copia las 8 palabras de datos de VW100-VW114 en el banco x del módulo AS-i.

5.7.4 CÓDIGOS DE ERROR DEL MÓDULO AS-I.

Código de error	*Descripción*
0000	Tarea finalizada sin errores.
0081	La dirección AS-i del esclavo no es correcta.
0082	El esclavo AS-i no está activado (no está en LAS).
0083	Error en el AS-interface.
0084	Comando no permitido en el estado actual del maestro AS-i.
0085	Hay un esclavo AS-i con la dirección 0.
0086	El esclavo AS-i tiene datos de configuración no válidos (E/S o ID).
00A1	El esclavo AS-i direccionado no se ha localizado en el AS-interface.
00A2	Hay un esclavo AS-i con la dirección 0.
00A3	Un esclavo AS-i con la nueva dirección ya existe en el AS-interface.
00A4	La dirección AS-i del esclavo no se puede borrar.
00A5	La dirección AS-i del esclavo no se puede ajustar.
00A6	La dirección AS-i del esclavo no se puede guardar permanentemente.
00A7	Error de lectura del código ID1 ampliado.
00A8	La dirección de destino no es plausible (p. ej. Una dirección de un esclavo B se ha utilizado para un esclavo estándar).
00B1	Error de longitud al transferir una cadena conforme al perfil 7.4.
00B2	Error de protocolo al transferir una cadena conforme al perfil 7.4.
00F8	Número o parámetro de tarea desconocido.
00F9	El maestro AS-i ha detectado un error de EEPROM.
0x01	Error del módulo maestro AS-i: – Módulo en posición incorrecta – Módulo no configurado – Otro error del módulo
0x02	La indicación del banco es demasiado larga.
0x03	El módulo maestro AS-i no está listo.
0x05	El módulo maestro AS-i está ocupado procesando un cambio de petición o un código de comando.

5.8 EJEMPLO DE APLICACIÓN.

La figura siguiente muestra el diseño propuesto para su futura realización por algún grupo de estudiantes que deseen conocer mas sobre el tema, este diseño cuenta con 4 esclavos, entre ellos un esclavo inteligente (LOGO! 24RC), una fuente de alimentación estándar, una fuente AS-i y un maestro AS-i.

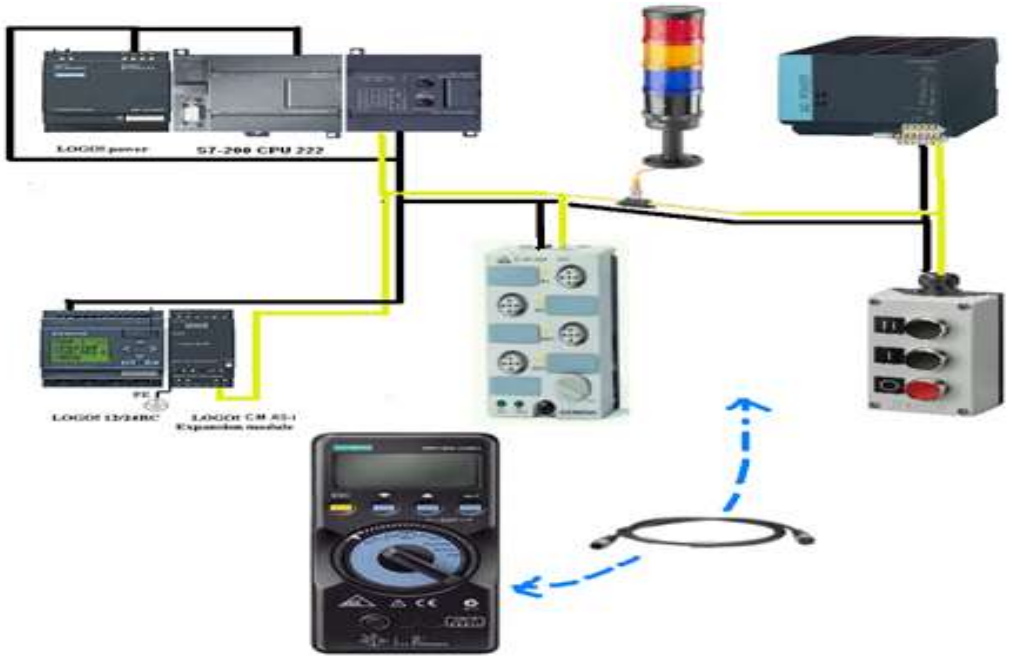


Fig. 5.48. Diseño brindado para futura implementación.

Equipo necesario:

Cantidad	Equipo
1	PLC S7-200, CPU 224XP
1	Modulo maestro AS-i CP 243-2.
1	LOGO! Power, fuente de alimentación estándar de LOGO de 24 VDC
1	Fuente de alimentación AS-i de 30 VDC.
1	LOGO! 24RC
1	Modulo LOGO! de expansión para comunicación CM AS-interface.
1	Botonera (modelo con 2 pulsadores y un piloto).
1	Esclavo AS-i serie K45 (Modelo con 2 entradas digitales y 2 salidas digitales).
1	Columna de señalización para redes AS-i (Modelo con 4 pilotos).
1	10 metros de cable AS-i amarillo y 10 metros de cable negro AS-i.
1	Consola de programación para esclavos AS-i.

Tabla 5.12. Equipo necesario para implementación de red AS-i básica.

La consola de configuración de esclavos AS-i no es necesaria ya que la configuración se puede realizar a través del software STEP 7 Micro/Win, aunque esta facilita el proceso de asignación del número de esclavo.

5.8.1 PROCEDIMIENTO A SEGUIR.

Realizar los pasos indicados:

- Conecte a la tensión de alimentación los productos como se muestra en el diagrama, con la línea de bus AS-i y la línea AS-i para la tensión auxiliar externa (esta se conectara únicamente a los equipos que necesiten tensión auxiliar externa).
- Direccionado de los esclavos, esto incluye el direccionamiento de LOGO! como esclavo inteligente, por medio de su modulo de comunicación CM AS-interface.
  Asigne las siguientes direcciones con la consola de direccionamiento:

Componente	Dirección
Modulo CM AS-interface	3
Esclavo AS-i serie K45	4
Columna de Señalización	5
Botonera	6

Tabla 5.13. Asignación de número de esclavo.

- Configure el maestro AS-i.
- Conecte la PC al PLC S7-200. Si utiliza el cable PC/PPI deberá seleccionar en STEP7 Micro/Win la correspondiente conexión local (COMx).
  Los interruptores DIP del cable PC/PPI deben ser ajustados de la siguiente manera:

1	2	3	4	5	6	7	8
0	0	0	0	1	0	0	0

Tabla 5.14. Posición de los interruptores del cable PC/PPI.

- Verificar la asignación del numero de cada esclavos mediante la ayuda del asistente AS-i del software STEP 7 Micro/Win.
- Seguir todos los pasos de mapeo de esclavos establecidos por el asistente de STEP 7 Micro/Win.
- Verificar que se creen las tres subrutinas establecidas por el asistente de mapeo, así como, la tabla de símbolos.
- Diseñar un programa en la ventana principal de Micro/Win con las especificaciones que se desee, según la función que se quiere que realice la red AS-i.
- Conmute la S7-200 CPU a STOP y cargue el programa en la CPU.
- Programar el LOGO! 24RC con las funciones que se desea que este realice.
- Cambie el PLC a modo RUN.

5.8.2 COSTOS DE LA CONSTRUCCION DE UN MODULO DE PRACTICAS BASICO.

Cantidad	Equipo	Descripción.	Código de catalogo.
1	PLC SIMATIC S7-200.	CPU 224XP, ap. Compacto, alimentación DC, 14 ED dc/10 SD dc, 2EA/1SA, /16 kb prog./10 kb datos, 2 puertos PPI/Freeport.	6ES7214-2AD23-0XB0
1	Cable PC/PPI	MM Multimaestro, para conexión de S7-200 a interfaz serie del PC, soporta Freeport y modem GSM.	6ES7901-3CB30-0XA0
1	Software Micro/Win + Licencia.		
1	LOGO! Power.	24 VDC, potencia estable, entrada de alimentación: 100-240 VAC, salida: 24 VDC/4 A	6EP1332-1SH51
1	Paquete de inicio para redes AS-i	1 – CP 243-2 Modulo maestro AS-i para S7-200.	6GK7243-2AX01-0XA0
		1 - Fuente de alimentación AS-i IP20, 3 A, 115/230 VAC (conmutable).	3RX9 501-0BA00
		50 m, cable perfilado AS-i Goma amarillo.	3RX9 010-0AA00
		50 m, cable perfilado AS-i Goma negro.	3RX9 020-0AA00
		2 – Modulos E/S digitales K45, IP67, 2E/2S.	3RK1 400-1BQ20-0AA3
		2- Placa de montaje K45.	3RK1 901-2EA00
		1–Consola de direccionamiento y diagnostico AS-i	3RK1 904-2AB01
		8 – Cables de conexión M12 – M12.	3RX8 000-0GF32-1AB5
1	Pulsadores en caja AS-i.	Version de mat. Aislante c= pulsador negro, plaq. II b= pulsador negro, plaq. II a= pulsador rojo, plaq. 0 AS-i	3SF5813-0DB00
1	Columna de señalización para redes AS-i	Elemento luz permanente, rojo uc 12v - uc 230v	8WD4200-1AB
		Elemento luz permanente, verde 12v - 230v	8WD4200-1AC
		Elemento luz permanente, amarillo 12v - 230v	8WD4200-1AD
		Elemento adaptador estándar AS-i para columna de señalización.	8WD4228-0BB
		Elemento de conexión con tapa terminal.	8WD4208-0AA
		1 – Soporte para montaje en pared.	8WD4208-0CD
		1 – conector M12 hembra 1 m.	3RK1 901-1NR21

TABLA 5.15. Lista de materiales y equipos siemens necesarios para realización de un modulo de prácticas para redes as-i.

•

La siguiente es una cotización de los productos necesarios en SIEMENS El Salvador:

SIEMENS

Antiguo Cuscatlán, Viernes, 19 de Febrero de 2010

Señores:
UNIVERSIDAD DE E.S. FACULTAD DE ING. Y ARQ.

Atención:
Carlos Palacios

Nuestra Referencia
SALQ32032
Ing. Victor Callejas

Asunto: **PAQUETE DE INICIO REDES AS-I**

Tenemos el agrado de ofrecer el suministro de:

Item	Cant.	Descripción	Precio Unitario	Total
2	1	PAQUETE DE INICIO PARA REDES AS-INTERFACE Incluye suministro de: Módulo de interfaz AS-i para S7-200 Fuente de alimentación AS-i para línea de comunicación 50m de cable de goma perfilado AS-i amarillo 2 módulos esclavos 2 entradas + 2 salidas digitales K45 Conectores M12 con cable para alambrado de señales	$1,798.97	$1,798.97

Item	Cant.	Descripción	Precio Unitario	Total
3	2	SENSOR AS-I CON CABEZA DE RODILLO	$92.27	$184.54
4	2	CAJA AS-I 3 POSICIONES, PULSADOR VERDE + ROJO + PARO EMERGENCIA	$105.69	$211.38
			SubTotal	$2194.89
			13% IVA	$285.34
			Total	$2480.23

NOTAS:

a) Forma de pago:
50% con su pedido
50% con su entrega
b) Tiempo de entrega:
3 a 4 semanas a partir de su pedido.
c) Validez de la oferta:
30 días a partir de esta fecha.
d) Garantía:
Los materiales y equipos a suministrarse gozan de una garantía normal de un año a partir de la fecha de entrega de los mismos salvo casos fortuitos como: terremotos, incendios, descargas atmosféricas, reparaciones hechas por terceros, operaciones indebidas del equipo, enemigos públicos, etc.
e) Precios:
Los precios han sido calculados en base a las condiciones actuales del mercado y al cambio interbancario actual. Sin embargo, estos deberán ser revisados en caso de que presenten variaciones en el mismo.
f) Reserva de suministro:
El cumplimiento del contrato por parte de SIEMENS está sujeto a que no hayan impedimentos por disposiciones legales nacionales e internacionales, especialmente en lo que se refiere a disposiciones de control de exportación.
g) Contacto:
Cualquier consulta relacionada con la oferta, favor comunicarse con nuestro departamento con el Ing. Víctor Callejas.

En espera que nuestra oferta sea de su agrado y conveniencia, le saludamos

Ing. Victor Callejas
Ingeniero de Ventas A&D

Lic. Roberto Huezo
Sub-Gerente Comercial

SIEMENS S.A. Calle Siemens #43 Parque Industrial Santa Elena Apartado 1525 San Salvador El Salvador Tel.: (503)2248-7333 Fax: (503)2278-0233

CONCLUSIONES CAPITULO 5.

Las redes As-interface reducen y facilitan el cableado en los sistemas de automatización moderna, ya que permiten la interconexión de sensores y actuadores a una red a través de un cable de dos hilos, que también suministra la potencia. Al eliminar el número de conductores del cableado tradicional y su sustitución por el cable amarillo AS-Interface, se logra una reducción significativa en materiales, instalación y mantenimiento. A demás, se agiliza la detección de errores en una instalación industrial.

El AS-Interface, se puede decir que más que un bus de campo es una forma inteligente de cableado. AS-interface no puede ni pretende sustituir a las redes complejas. Pero en el nivel inferior de la comunicación industrial, en los sensores e interruptores de nivel y otros, este sistema destaca por sus soluciones sencillas y rentables. Los estudios modernos, incluso verifican la ventaja económica de la integración de AS-Interface con interruptores y botones en los paneles de control. El costo añadido del chip AS-Interface incorporado en los esclavos están compensado por los reducidos gastos de cableado y mantenimiento.

Algunos años atrás era tan complicado realizar una red AS-i que muchas veces se dejaba de lado las ventajas constructivas que esta tenia, debido a la dificultad de realizar programas para esta. Hoy en día con la ayuda de los asistentes es tan fácil desarrollar aplicaciones para este tipo de red que se están empezando a hacer cada vez más populares.

El sistema básico de red AS-i que se plantea en este capítulo, tiene todo lo necesario para que el alumno, aprenda a un nivel básico, la teoría y la práctica sobre la construcción y programación en redes AS-i. A pesar de ser una buena propuesta de diseño, no fue factible su realización debido a su elevado costo, el cual sobrepasaba las expectativas de esta tesis. Por lo cual se deja como referencia todo este capítulo para que posteriormente cualquier alumno interesado en el área en conjunto con las autoridades de la escuela de ingeniería eléctrica, tenga las bases para desarrollar algún proyecto en el futuro.

REFERENCIAS BIBLIOGRÁFICAS

Capitulo 1

1. Titulo: **“Manual del sistema de Automatización S7-200”**
   Serie: “SIMATIC”
   Autor: “SIEMENS”
   Edición: 08/2008

2. Pagina WEB: infoPLC
   Titulo: **“GRAFCET. Diseño e implantación en autómatas”**
   URL:“http://www.infoplc.net/Documentacion/Docu_GEMA_GRAFCET/infoPLC_net_Tema_4_GRAFCET_Diseno_e_implantacion_en_automatas.html”

3. Pagina WEB: infoPLC
   Titulo: **“Modos de marcha y parada. Tutorial de la guía GEMMA”**
   URL:“http://www.infoplc.net/Documentacion/Docu_GEMA_GRAFCET/infoPLC_net_Tutorialgemma.html”

4. Pagina WEB: infoPLC
   Titulo: **“Manuales y software del S7-200”**
   URL:“http://www.infoplc.net/Descargas/Descargas_Siemens/Descargas-Siemens.htm”

Capitulo 3

5. Categoría: Tesis
   Titulo: “**Using Rapid Prototyping Tools for Automatic Control System Laboratory**”
   Autores: “Robert S. Cochran”, “Todd D. Batzel” and “Peter J. Shull”
   Institución: “The Pennsylvania State University, PENNSTATE”
   Publicación: 10/2006, Altona, Pennsylvania.

6. Pagina WEB: CONNEXIONS
   Titulo: **“DC Motor Control”**
   URL:“ http://cnx.org/content/m22189/latest/”

7. Titulo: **“eZdspF2812 Technical Reference”**
   Serie: “C2000”
   Autor: “Digital Spectrum”
   Edición: 02/2003

Capitulo 4

8. Titulo: “**Template Builder Guide**”
   Autor: “SIELCO SISTEMI srl”
   Edición: 2009

9. Titulo: "**Run Time Guide**"
   Autor: "SIELCO SISTEMI srl"
   Edición: 2009

10. Titulo: "**Protocols Guide**"
    Autor: "SIELCO SISTEMI srl"
    Edición: 2009

11. Titulo: "**Project Manager Guide**"
    Autor: "SIELCO SISTEMI srl"
    Edición: 2009

12. Titulo: "**Gate Builder Guide**"
    Autor: "SIELCO SISTEMI srl"
    Edición: 2009

13. Titulo: "**Code Builder Guide**"
    Autor: "SIELCO SISTEMI srl"
    Edición: 2009

Capitulo 5

14. Titulo: "**AS-Interface – Introducción y Fundamentos**"
    Serie: "SIMATIC NET"
    Autor: "SIEMENS"
    Edición: 12/99

15. Titulo: **"Manual del sistema AS-interface"**
    Autor: "SIEMENS"
    Edición: 11/2008

16. Titulo: **"Guia de selección de dispositivos AS-interface"**
    Autor: "SIEMENS"
    Edición: 2007

17. Titulo: **"AS-Interface Maestro CP 243-2"**
    Serie: "SIMATIC NET"
    Autor: "SIEMENS"
    Edición: 2002

18. Titulo: **"AS-Interface LOGO! CM"**
    Serie: "SIMATIC NET"
    Autor: "SIEMENS"
    Edición: 2003

ANEXOS

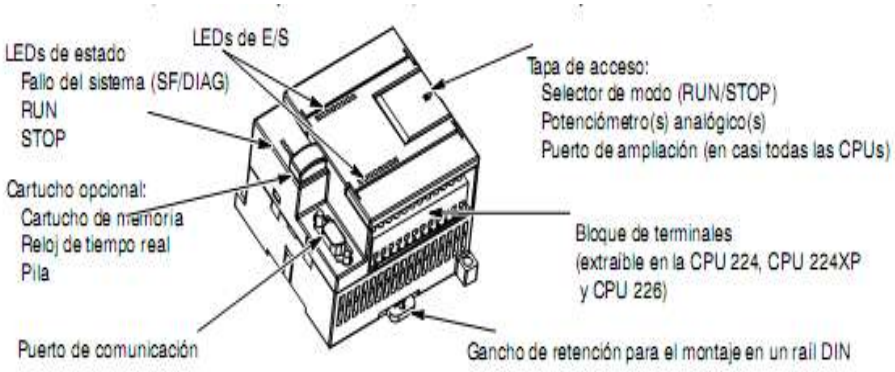
ANEXO A
GENERALIDADES SOBRE EL PLC S7-200

A.1 DESCRIPCION GENERAL CPUs S7-200

La familia SIMATIC S7 se divide de la siguiente forma:

- ✓ PLC de Gama Alta: S7-400
- ✓ PLC de Gama Media: S7-300
- ✓ PLC de Gama Baja (Micro PLC) : S7-200

La CPU S7-200 incorpora en una forma compacta un microprocesador, una fuente de alimentación integrada, así como circuitos de entrada y de salida que conforman un potente micro PLC (fig.A-1).



1. Salidas digitales integradas
2. LEDs de estado de las salidas digitales
3. Terminales de alimentación
4. Conmutador Stop/Run
5. Conector para el cable de ampliación
6. LEDs de estado de la CPU
7. Ranura para el cartucho de memoria
8. Puerto de comunicaciones (p. Ej. PPI)
9. Entradas digitales integradas
10. LEDs de estado de las entradas digitales
11. Fuente de alimentación integrada
12. Potenciómetros integrados
13. Módulo de ampliación
14. Fijadores para tornillo (DIN métrica M4, diámetro 5 mm)
15. Pestaña de fijación

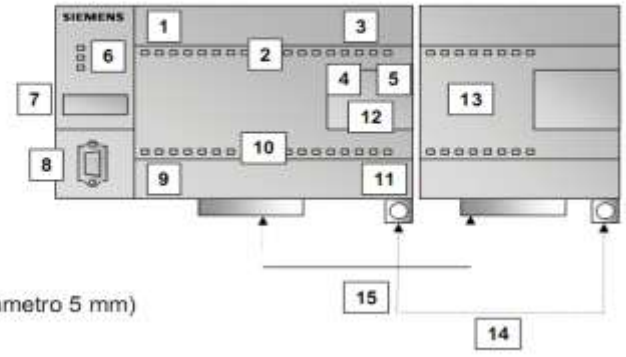


Fig. A.1. PLC s7-200

Siemens ofrece diferentes modelos de CPUs S7-200 que incorporan una gran variedad de funciones y prestaciones para crear soluciones efectivas de automatización destinadas a

numerosas aplicaciones. En la tabla A.1 se describen algunas características comunes de todas las CPU s7-200.

Características técnicas comunes de las CPUs 221, 222, 224, 224XP, 224XPsi y 226:	
Característica	CPU 221, 222, 224, 224XP, 224XPsi, 226
Aritmética en coma fija de 32 bits según norma IEEE	sí
Reguladores PID integrados plenamente parametrizables	sí, hasta 8 reguladores PID independientes
Velocidad de procesamiento al bit	0,22 µs
Interrupciones controladas por tiempo	2 (tiempo de ciclo entre 1 y 255 ms con 1 ms de resolución)
Interrupciones hardware (detección de flancos en entradas)	máx. 4 entradas
Marcas, temporizadores, contadores	256 de cada
Contadores rápidos	4–6 (según CPU), máx. 30 kHz, ó 200 kHz en la CPU 224 XP
Salidas de impulsos (modulables en ancho o frecuencia)	2 salidas, 20 kHz cada una (para variantes DC), 100 kHz en CPU 224 XP
Memoria de programas y datos	remanente (no volátil)
Memorización de datos dinámicos en caso de fallo de alimentación	remanencia: mediante condensador interno de alto rendimiento o módulo de pila adicional. No volátil: carga del bloque de datos con STEP 7-Micro/WIN, TD 200C o vía programa de usuario en la EEPROM integrada
Respaldo de los datos dinámicos mediante módulo de pila	típ. 200 días
Puerto integrado de comunicación	sí, puerto RS 485 que soporta los modos siguientes: maestro o esclavo PPI/esclavo MPI/ Freeport (protocolo ASCII programable)
Velocidad de transferencia máx.	187,5 kbaudios (PPI/MPI) ó 115,2 kbaudios (Freeport)
Software de programación	STEP 7-Micro/WIN que sirve para todos los lenguajes como AWL, FUP o KOP
Módulo de memoria de programa opcional	sí, programable en la CPU, para transferir programas, Data Logging, recetas, documentación
Variante DC/DC/DC	sí
Alimentación	24 V DC
Entradas digitales	24 V DC
Salidas digitales	24 V DC, máx. 0,75 A, pueden conectarse en paralelo para aumentar la potencia
Variante AC/DC/relés	sí
Alimentación	85–264 V AC
Entradas digitales	24 V DC
Salidas digitales	5–30 V DC ó 5-250 V AC, máx. 2 A (relés)

Tabla A.1. Características comunes CPU s7-200

En la tabla A.2 se comparan de forma resumida algunas de las funciones de las CPUs S7-200.

Datos específicos de cada CPU					
Característica	CPU 221[1]	CPU 222[1]	CPU 224[1]	CPU 224XP[1] CPU 224XPsi[2]	CPU 226[1]
					
Entradas/salidas digitales integradas	6 ED/4 SD	8 ED/6 SD	14 DE/10 DA	14 DE/10 DA	24 DE/16 DA
Entradas/salidas digitales Nº de canales vía módulos de ampliación	–	48/46/94	114/110/224	114/110/224	128/128/256
Entradas/salidas analógicas Nº de canales vía módulos de ampliación	–	16/8/16	32/28/44	2 EA/1 SA integradas 32/28/44	32/28/44
Memoria de programas	4 kbytes	4 kbytes	8/12 kbytes	12/16 kbytes	16/24 kbytes
Memoria de datos	2 kbytes	2 kbytes	8 kbytes	10 kbytes	10 kbytes
Memorización de datos dinámicos vía condensador de alto rendimiento	tip. 50 h	tip. 50 h	tip. 100 h	tip. 100 h	tip. 100 h
Contadores rápidos	4x30 kHz, de ellos, 2x20 kHz usables como contadores A/B	4x30 kHz, de ellos, 2x20 kHz usables como contadores A/B	6x30 kHz, de ellos, 4x20 kHz usables como contadores A/Br	4 x 30 kHz, 2 x 200 kHz, de ellos, 3 x 20 kHz y 1 x 100 kHz usables como contadores A/B	6x30 kHz, de ellos, 4x20 kHz usables como contadores A/Br
Puertos de comunicación RS 485	1	1	1	2	2
Protocolos soportados::				sí, en los dos puertos	sí, en los dos puertos
– PPI maestro / esclavo	sí	sí	sí	sí	sí
– MPI esclavo	sí	sí	sí	sí	sí
– Freeport (protocolo ASCII programable)	sí	sí	sí	sí	sí
Posibilidades de comunicación opcionales	no ampliable	sí, esclavo PROFIBUS DP y/o maestro AS-Interface/Ethernet/ Internet/módem	sí, esclavo PROFIBUS DP y/o maestro AS-Interface/Ethernet/ Internet/módem	sí, esclavo PROFIBUS DP y/o maestro AS-Interface/Ethernet/ Internet/módem	sí, esclavo PROFIBUS DP y/o maestro AS-Interface/Ethernet/ Internet/módem
Potenciómetro analóg. de 8 bits integrado	1	1	2	2	2
(para p. en marcha, cambio de valores)					
Reloj de tiempo real	opcional	opcional	sí	sí	sí
Alimentación p. sensores 24 V DC integrada	máx. 180 mA	máx. 180 mA	máx. 280 mA	máx. 280 mA	máx. 400 mA
Regleta de conexión desenchufable	–	–	sí	sí	sí
Dimensiones (A x A x P en mm)	90 x 80 x 62	90 x 80 x 62	120,5 x 80 x 62	140 x 80 x 62	196 x 80 x 62

Tabla A.2. Comparación CPU S7-200

La gama S7-200 incluye una gran variedad de módulos de ampliación para poder satisfacer aún mejor los requisitos de la aplicación. Estos módulos se pueden utilizar para agregar funciones a la CPU S7-200. La fig. A.2 muestra una lista de los módulos de ampliación disponibles en la actualidad.

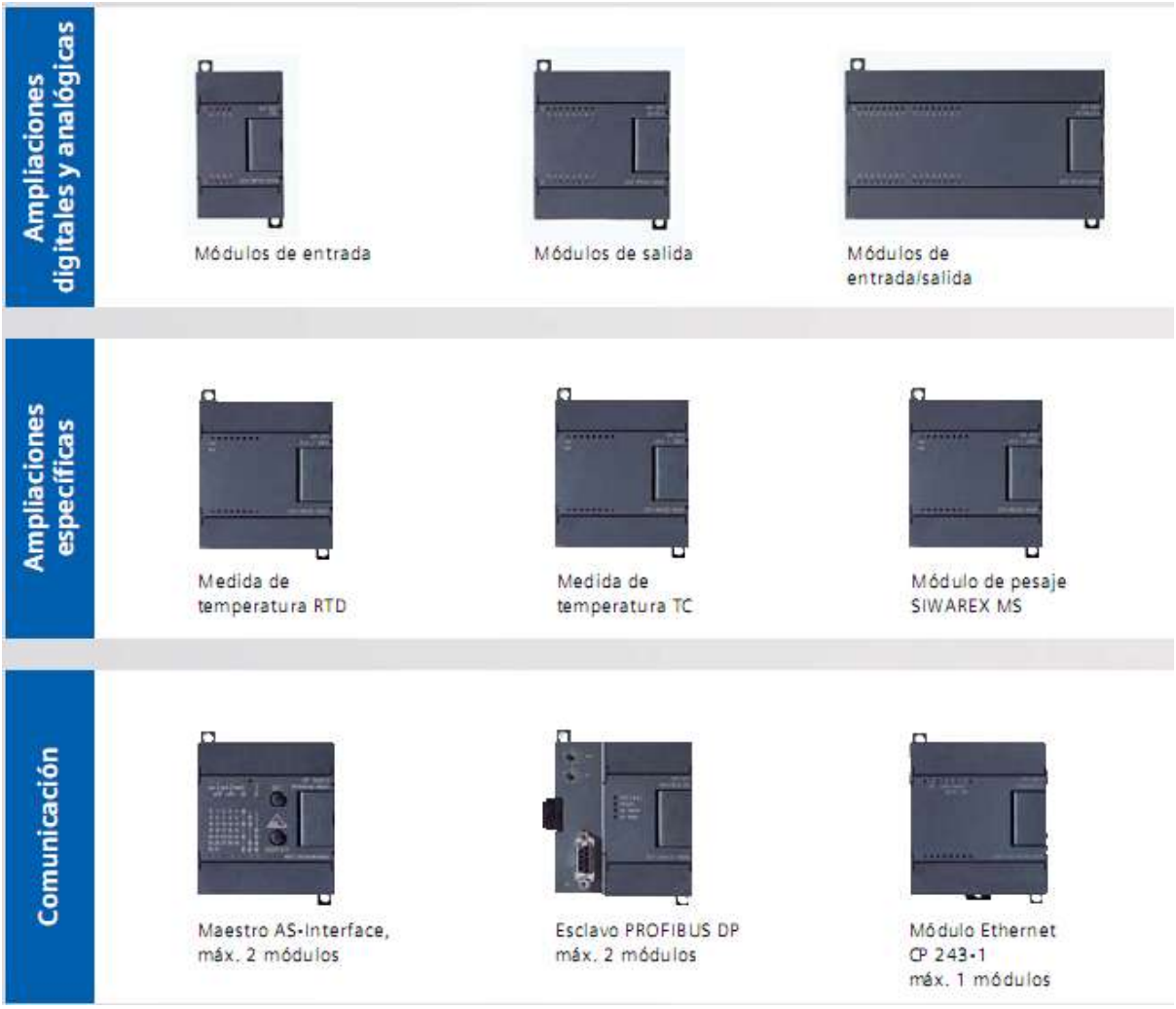


Fig. A.2. Módulos de ampliación S7-200

Fig. A.3. Diferencias entre algunas CPU's del S7-200.

Las CPUs S7-200 son también aptas para trabajar en condiciones extremas:
Integración en maquinas con vibraciones

- ✓ Hasta 1G (Unidad de medida de vibración mecánica) en montajes sobre perfil DIN.
- ✓ Hasta 2G (Unidad de medida de vibración mecánica) en montajes con sujeción mediante tornillos.
- ✓ Mejora de la estabilidad contra vibración a través de la conexión entre módulos mediante cable flexible.
- ✓ Mejora de la fijación contra deslizamiento sobre el carril DIN.

En la programación de una CPU S7-200 se pueden utilizar distintos lenguajes de programación. Entre los más significativos se encuentran:

- ✓ Lenguaje de contactos (KOP)

Este lenguaje también llamado lenguaje de escalera permite crear programas con componentes similares a los elementos de un esquema de circuitos.

Los programas se dividen en unidades lógicas pequeñas llamadas networks, y el programa se ejecuta segmento a segmento, secuencialmente, y también en un ciclo.

Las operaciones se representan mediante símbolos gráficos que incluyen 3 formas básicas:

- Contactos representan condiciones lógicas de “entrada” Ej.: interruptores, botones, condiciones internas, etc.
- Bobinas representan condiciones lógicas de “salida”, actuadores
- Cuadros, representan operaciones adicionales tales como temporizadores, contadores u operaciones aritméticas

Las ventajas de KOP o Ladder son:

- Facilita trabajo de programadores principiantes.
- La representación grafica de la aplicación “estado de programa” colabora a la fácil comprensión del desarrollo del código.
- Se puede editar con AWL.

- ✓ Lenguaje por lista de instrucciones (AWL)

Este incluye una lista de instrucciones que se ejecutan secuencialmente dentro de un ciclo. Una de las principales ventajas que presenta es que cualquier programa creado en FUP o KOP puede ser editado por AWL, no así a la inversa.

✓ Plano de funciones lógicas (FUP)

Consiste en un diagrama de funciones que permite visualizar las operaciones en forma de cuadros lógicos similares de los de de las compuertas lógicas.
El estilo de representación en forma de compuertas gráficas se adecua especialmente para observar el flujo del programa.

- Se puede editar con AWL o KOP

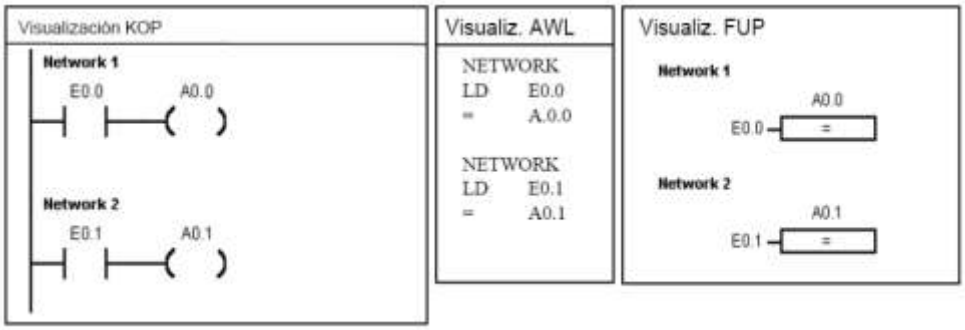


Fig. A.4. Lenguajes de Programación.

A.2 LÓGICA DE CONTROL EN EL S7-200

El S7-200 ejecuta cíclicamente la lógica de control del programa, leyendo y escribiendo datos. El funcionamiento básico del S7-200 es muy sencillo:

- ✓ El S7-200 lee el estado de las entradas.
- ✓ El programa almacenado en el S7-200 utiliza las entradas para evaluar la lógica. Durante la ejecución del programa, el S7-200 actualiza los datos.
- ✓ El S7-200 escribe los datos en las salidas.

La figura A.5 muestra cómo se procesa un esquema de circuitos simple en el S7-200. En este ejemplo, el estado del interruptor para arrancar el motor se combina con los estados de otras entradas. El resultado obtenido establece entonces el estado de la salida que corresponde al actuador que arranca el motor.

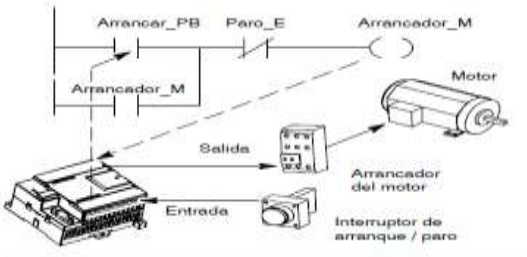


Fig. A.5. Controlar entradas y salidas.

El S7-200 ejecuta una serie de tareas de forma repetitiva. Esta ejecución se denomina ciclo o ciclo de trabajo. Como muestra la figura A.6, el S7-200 ejecuta la mayoría de las tareas siguientes (o todas ellas) durante un ciclo:

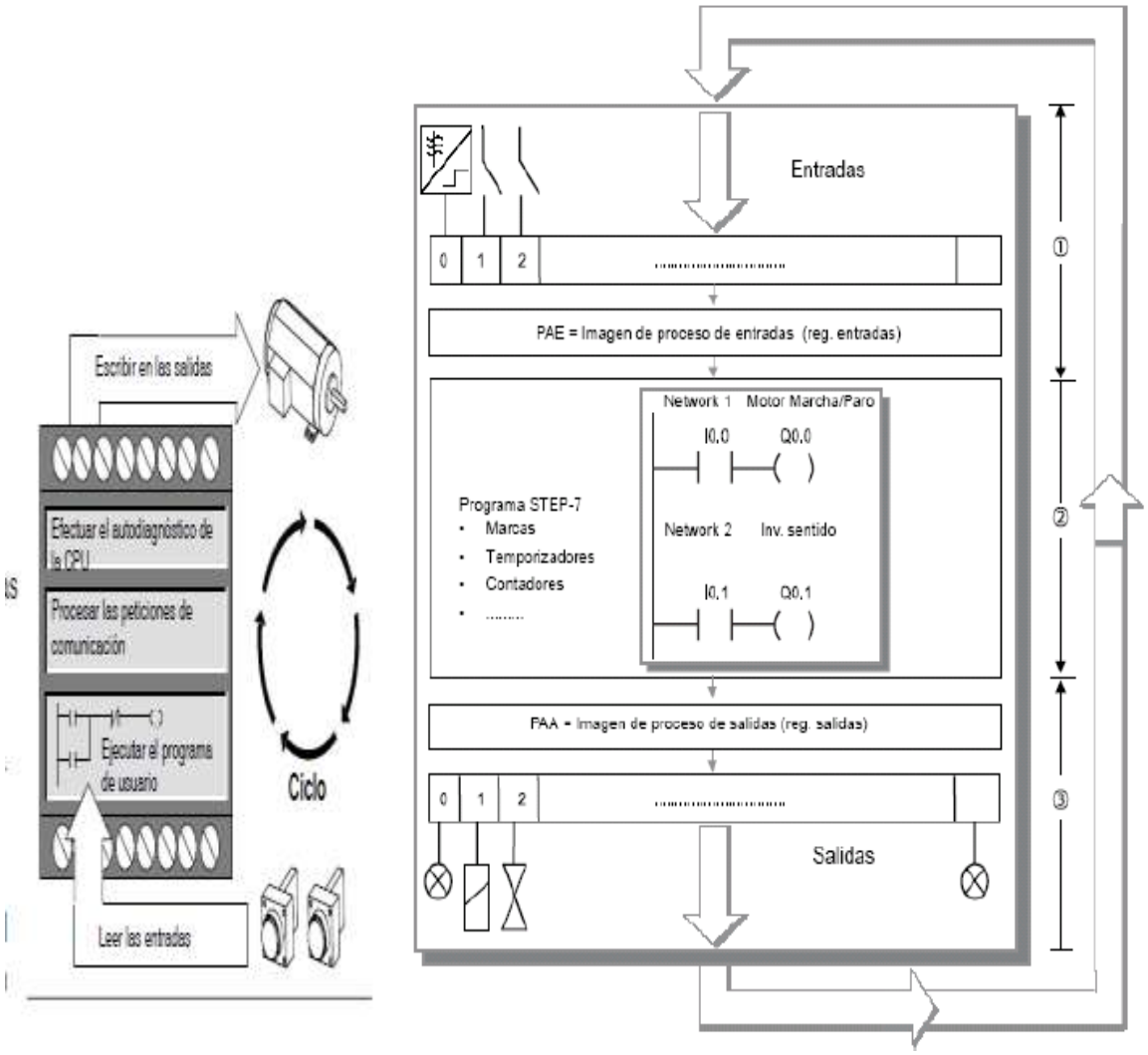


Fig. A.6. Ciclo de Trabajo S7-200.

- ✓ Leer las entradas: el S7-200 copia el estado de las entradas físicas en la imagen del proceso de las entradas.
- ✓ Ejecutar la lógica de control en el programa: el S7-200 ejecuta las operaciones del programa y guarda los valores en las diversas áreas de memoria.
- ✓ Procesar las peticiones de comunicación: el S7-200 ejecuta las tareas necesarias para la comunicación.
- ✓ Efectuar el auto diagnóstico de la CPU: el S7-200 verifica si el firmware, la memoria del programa y los módulos de ampliación están trabajando correctamente.
- ✓ Escribir en las salidas: los valores almacenados en la imagen del proceso de las salidas se escriben en las salidas físicas.

Un ciclo dura normalmente entre 3 y 10 ms. La duración depende del número y tipo de instrucciones (operaciones) utilizadas. El ciclo consta de dos partes principales:

- ✓ Tiempo del sistema operativo, normalmente 1 ms.
- ✓ Tiempo para ejecutar las instrucciones, normalmente 2 ms.

La ejecución del programa de usuario depende de si el S7-200 está en modo STOP o RUN. El programa se ejecutará si el S7-200 está en modo RUN. En cambio, no se ejecutará en modo STOP.

Leer las entradas

Entradas digitales: Al principio de cada ciclo se leen los valores actuales de las entradas digitales y se escriben luego en la imagen del proceso de las entradas.

Entradas analógicas: El S7-200 no actualiza las entradas analógicas de los módulos de ampliación como parte del ciclo normal, a menos que se haya habilitado la filtración de las mismas. Existe un filtro analógico que permite disponer de una señal más estable. Este filtro se puede habilitar para cada una de las entradas analógicas.

Si se habilita la filtración de una entrada analógica, el S7-200 actualizará esa entrada una vez por ciclo, efectuará la filtración y almacenará internamente el valor filtrado. El valor filtrado se suministrará cada vez que el programa accede a la entrada analógica.

Si no se habilita la filtración, el S7-200 leerá de los módulos de ampliación el valor de la entrada analógica cada vez que el programa de usuario acceda a esa entrada.

Las entradas analógicas AIW0 y AIW2 incorporadas en la CPU 224XP se actualizan en cada ciclo con el resultado más reciente del convertidor analógico/digital. Este convertidor es de tipo promedio (sigma-delta) y, por lo general, no es necesario filtrar las entradas en el software.

La filtración de las entradas analógicas permite disponer de un valor analógico más estable. Utilice el filtro de entradas analógicas en aplicaciones donde la señal de entrada cambia lentamente. Si la señal es rápida, no es recomendable habilitar el filtro analógico.

No utilice el filtro analógico en módulos que transfieran informaciones digitales o indicaciones de alarma en las palabras analógicas. Desactive siempre el filtro analógico si utiliza módulos RTD, termopar o AS-Interface Máster.

Ejecutar el programa

Durante esta fase del ciclo, el S7-200 ejecuta el programa desde la primera operación hasta la última. El control directo de las entradas y salidas permite acceder directamente a éstas mientras se ejecuta el programa o una rutina de interrupción.

Si se utilizan interrupciones, las rutinas asociadas a los eventos de interrupción se almacenarán como parte del programa. Las rutinas de interrupción no se ejecutan como parte del ciclo, sino sólo cuando ocurre el evento (en cualquier punto del ciclo).

Procesar las peticiones de comunicación.

Durante esta fase del ciclo, el S7-200 procesa los mensajes que haya recibido por el puerto de comunicación o de los módulos de ampliación inteligentes.

Efectuar el auto diagnóstico

Durante el auto diagnóstico, el S7-200 comprueba si la CPU está funcionando correctamente, así como el estado de los módulos de ampliación.

Escribir las salidas digitales

Al final de cada ciclo, el S7-200 escribe los valores de la imagen del proceso de las salidas en las salidas digitales. (Las salidas analógicas se actualizan de inmediato, independientemente del ciclo.)

A.3 ACCEDER A LOS DATOS DEL S7-200

Dentro de la CPU dispondremos de varias áreas de memoria, las cuales emplearemos para diversas funciones:

- ✓ **Memoria del programa de usuario**: aquí introduciremos el programa que el autómata va a ejecutar cíclicamente.

- ✓ **Memoria de la tabla de datos**: se suele subdividir en zonas según el tipo de datos (como marcas de memoria, temporizadores, contadores, etc.).

- ✓ **Memoria del sistema**: aquí se encuentra el programa en código maquina que monitoriza el sistema (programa del sistema o firmware). Este programa es ejecutado directamente por el microprocesador/microcontrolador que posea el autómata.

- ✓ **Memoria de almacenamiento**: se trata de memoria externa que empleamos para almacenar el programa de usuario, y en ciertos casos parte de la memoria de la tabla de datos. Suele ser de uno de los siguientes tipos: EPROM, EEPROM, o FLASH

En los PLC S7-200 se pueden acceder a los datos en las siguientes áreas de memoria:

Imagen del proceso de las entradas I

El S7-200 lee las entradas físicas al comienzo de cada ciclo y escribe los correspondientes valores en la imagen del proceso de las entradas. A ésta última se puede acceder en formato de bit, byte, palabra o palabra doble:

Bit:	I*[direcc. del byte].[direcc. del bit]*	I0.1
Byte, palabra o palabra doble:	I*[tamaño][direcc. del byte inicial]*	IB4

Imagen del proceso de las salidas Q

Al final de cada ciclo, el S7-200 copia en las salidas físicas el valor almacenado en la imagen del proceso de las salidas. A ésta última se puede acceder en formato de bit, byte, palabra o palabra doble:

Bit:	Q*[direcc. del byte].[direcc. del bit]*	Q0.1
Byte, palabra o palabra doble:	Q*[tamaño][direcc. del byte inicial]*	QB5

Memoria de variables V

La memoria de variables (memoria V) se puede utilizar para depositar los resultados intermedios calculados por las operaciones en el programa. La memoria V también permite almacenar otros datos que pertenezcan al proceso o a las tareas actuales. A la memoria V se puede acceder en formato de bit, byte, palabra o palabra doble:

Bit:	V*[direcc. del byte].[direcc. del bit]*	V10.2
Byte, palabra o palabra doble:	V*[tamaño][direcc. del byte inicial]*	VW100

Área de marcas M

El área de marcas (memoria M) se puede utilizar como relés de control para almacenar el estado inmediato de una operación u otra información de control. Al área de marcas se puede acceder en formato de bit, byte, palabra o palabra doble:

Bit:	M*[direcc. del byte].[direcc. del bit]*	M26.7
Byte, palabra o palabra doble:	M*[tamaño][direcc. del byte inicial]*	MD20

Área de temporizadores T

Los temporizadores del S7-200 tienen resoluciones (intervalos) de 1 ms, 10 ms y 100 ms. Existen dos variables asociadas a los temporizadores:

- ✓ Valor actual: en este número entero de 16 bits con signo se deposita el valor de tiempo contado por el temporizador.

- ✓ Bit del temporizador (bit T): este bit se activa o se desactiva como resultado de la comparación del valor actual con el valor de preselección. Éste último se introduce como parte de la operación del temporizador.

A estas dos variables se accede utilizando la dirección del temporizador (T + número del Temporizador). Dependiendo de la operación utilizada, se accede al bit del temporizador o al valor actual. Las operaciones con operandos en formato de bit acceden al bit del temporizador, en tanto que las operaciones con operandos en formato de palabra acceden al valor actual. Como muestra la figura A.7, la operación Contacto normalmente abierto accede al bit del temporizador, en tanto que la operación Transferir palabra accede al valor actual del temporizador.

Fig. A.7. Acceder al bit del temporizador o al valor actual de un temporizador.

Área de contadores C

Los contadores del S7-200 son elementos que cuentan los cambios de negativo a positivo en la(s) entrada(s) de contaje. Hay contadores que cuentan sólo adelante, otros que cuentan sólo atrás y otros cuentan tanto adelante como atrás. Existen dos variables asociadas a los contadores:

- ✓ Valor actual: en este número entero de 16 bits con signo se deposita el valor de contaje acumulado.

- ✓ Bit del contador (bit C): este bit se activa o se desactiva como resultado de la comparación del valor actual con el valor de preselección. El valor de preselección se introduce como parte de la operación del contador.

A estas dos variables se accede utilizando la dirección del contador (C + número del contador).

Dependiendo de la operación utilizada, se accede al bit del contador o al valor actual. Las operaciones con operandos en formato de bit acceden al bit del contador, en tanto que las operaciones con operandos en formato de palabra acceden al valor actual. Como muestra la figura A.8, la operación Contacto normalmente abierto accede al bit del contador, en tanto que la operación Transferir palabra accede al valor actual del contador.

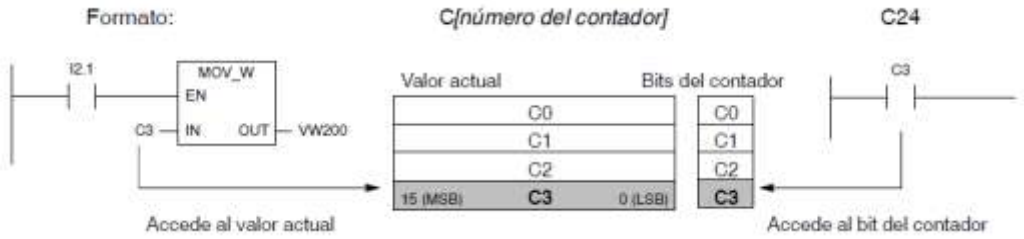


Fig. A.8. Acceder al bit del contador o al valor actual de un contador.

Contadores rápidos (HC)

Los contadores rápidos cuentan eventos rápidos, independientemente del ciclo de la CPU. Tienen un valor de contaje de entero de 32 bits con signo (denominado también valor actual).
Para acceder al valor de contaje del contador rápido, se indica la dirección del mismo (utilizando el identificador HC) y el número del contador (por ejemplo, HC0). El valor actual del contador rápido es de sólo lectura, pudiéndose acceder al mismo sólo en formato de palabra doble (32 bits).

Formato:	HC*[número del contador rápido]*	HC1

Acumuladores AC

Los acumuladores son elementos de lectura/escritura que se utilizan igual que una memoria. Por ejemplo, se pueden usar para transferir parámetros de y a subrutinas, así como para almacenar valores intermedios utilizados en cálculos. El S7-200 dispone de cuatro acumuladores de 32 bits (AC0, AC1, AC2 y AC3). A los acumuladores se puede acceder en formato de byte, palabra o palabra doble.

La operación utilizada para el acceso al acumulador determina el tamaño de los datos a los que se accede. Como muestra la figura A.9, cuando se accede a un acumulador en formato de byte o de palabra se utilizan los 8 ó 16 bits menos significativos del valor almacenado en el acumulador.

Cuando se accede a un acumulador en formato de palabra doble, se usan todos los 32 bits.

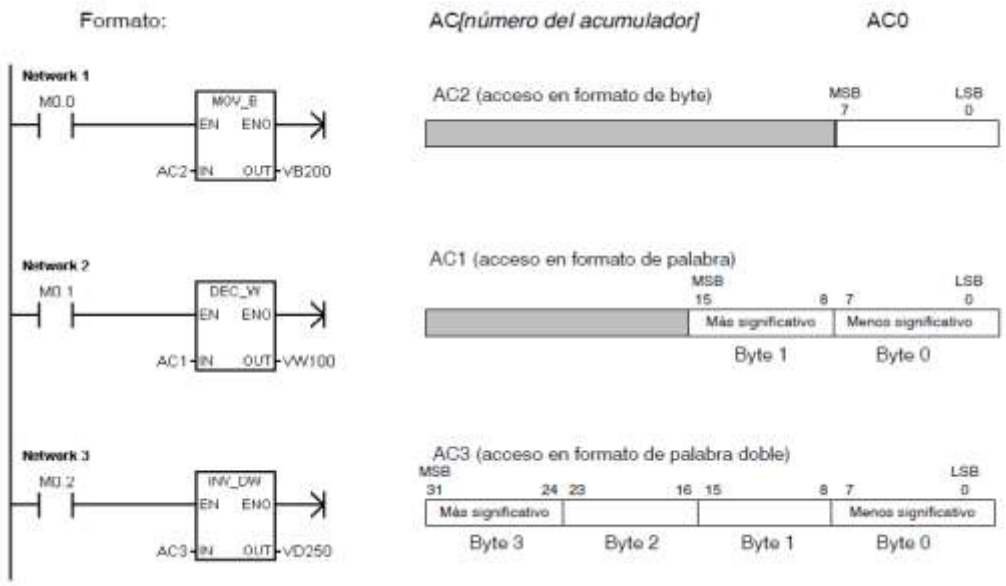


Fig. A.9. Acceder a los acumuladores.

Marcas especiales (SM)

Las marcas especiales permiten intercambiar datos entre la CPU y el programa. Estas marcas se pueden utilizar para seleccionar y controlar algunas funciones especiales de la CPU S7-200. Por ejemplo, hay una marca que se activa sólo en el primer ciclo, marcas que se activan y se desactivan en determinados intervalos, o bien marcas que muestran el estado de las operaciones matemáticas y de otras operaciones. A las marcas especiales se puede acceder en formato de bit, byte, palabra o palabra doble:

Bit:	SM*[direcc. del byte].[direcc. del bit]*	SM0.1
Byte, palabra o palabra doble:	SM*[tamaño][direcc. del byte inicial]*	SMB86

Marcas especiales			
SM0.0	Siempre ON	SM1.0	Resultado de la operación = 0
SM0.1	Primer ciclo	SM1.1	Desbordamiento o valor no válido
SM0.2	Datos remanentes perdidos	SM1.2	Resultado negativo
SM0.3	Alimentación	SM1.3	División por 0
SM0.4	30 s OFF / 30 s ON	SM1.4	Tabla llena
SM0.5	0,5 s OFF / 0,5 s ON	SM1.5	Tabla vacía
SM0.6	OFF 1 ciclo / ON 1 ciclo	SM1.6	Error de conversión de BCD a binario
SM0.7	Selector en posición RUN	SM1.7	Error de conversión ASCII a hexadecimal

Tabla A-3. Marcas Especiales

Memoria local L

El S7-200 dispone de 64 bytes de memoria local (L), de los cuales 60 se pueden utilizar como memoria "borrador" para transferir parámetros formales a las subrutinas.

La memoria local es similar a la memoria V (memoria de variables), con una excepción: la memoria V tiene un alcance global, en tanto que la memoria L tiene un alcance local. El término "alcance global" significa que a una misma dirección de la memoria se puede acceder desde cualquier parte del programa (programa principal, subrutinas o rutinas de interrupción). El término "alcance local" significa que la dirección de la memoria está asociada a una determinada parte del programa. El S7-200 asigna 64 bytes de la memoria L al programa principal, 64 bytes a cada nivel de anidado de las subrutinas y 64 bytes a las rutinas de interrupción.

A los bytes de la memoria L asignados al programa principal no se puede acceder ni desde las subrutinas ni desde las rutinas de interrupción. Una subrutina no puede acceder a la asignación de la memoria L del programa principal, ni a una rutina de interrupción, ni tampoco a una subrutina diferente. Una subrutina tampoco puede acceder a la asignación de la memoria L del programa principal, ni a la de otra subrutina.
El S7-200 asigna la memoria L según sea necesario en ese momento. Por consiguiente, mientras se está ejecutando la parte principal del programa, no existen las asignaciones de la memoria L para las subrutinas y las rutinas de interrupción. Cuando ocurre una interrupción o cuando se llama a una subrutina, la memoria local se asigna según sea necesario. La nueva asignación de la memoria L puede reutilizar las mismas direcciones de la memoria L de una subrutina o de una rutina de interrupción diferentes.

El S7-200 no inicializa la memoria L durante la asignación de direcciones, pudiendo contener cualquier valor. Al transferir parámetros formales a una llamada de subrutina, el S7-200 deposita los valores de los parámetros transferidos en las direcciones de la memoria L que se hayan asignado a esa subrutina. Las direcciones de la memoria L que no reciban un valor como resultado de la transferencia de parámetros formales no se inicializarán, pudiendo contener cualquier valor en el momento de la asignación.

Bit:	L*[direcc. del byte].[direcc. del bit]*	L0.0
Byte, palabra o palabra doble:	L*[tamaño][direcc. del byte inicial]*	LB33

Entradas analógicas AI

El S7-200 convierte valores reales analógicos (por ejemplo, temperatura, tensión, etc.) en valores digitales en formato de palabra (de 16 bits). A estos valores se accede con un identificador de área (AI), seguido del tamaño de los datos (W) y de la dirección del byte inicial.

Puesto que las entradas analógicas son palabras que comienzan siempre en bytes pares (por ejemplo, 0, 2, 4, etc.), es preciso utilizar direcciones con bytes pares (por ejemplo,

AIW0, AIW2, AIW4, etc.) para acceder a las mismas. Las entradas analógicas son valores de sólo lectura.

Formato:	AIW*[dirección del byte inicial]*	AIW4

Salidas analógicas (AQ)

El S7-200 convierte valores digitales en formato de palabra (de 16 bits) en valores reales analógicos (por ejemplo, intensidad o tensión). Estos valores analógicos son proporcionales a los digitales.

A los valores analógicos se accede con un identificador de área (AQ), seguido del tamaño de los datos (W) y de la dirección del byte inicial. Puesto que las salidas analógicas son palabras que comienzan siempre en bytes pares (por ejemplo, 0, 2, 4, etc.), es preciso utilizar direcciones con bytes pares (por ejemplo, AQW0, AQW2, AQW4, etc.) para acceder a las mismas. Las salidas analógicas son valores de sólo escritura.

Formato:	AQW*[dirección del byte inicial]*	AQW4

Relés de control secuencial SCR S

Los relés de control secuencial (SCR o bits S) permiten organizar los pasos del funcionamiento de una máquina en segmentos equivalentes en el programa. Los SCRs sirven para segmentar lógicamente el programa de usuario. A los relés de control secuencial (SCR) se puede acceder en formato de bit, byte, palabra o palabra doble.

Bit:	S*[direcc. del byte].[direcc. del bit]*	S3.1
Byte, palabra o palabra doble:	S*[tamaño][direcc. del byte inicial]*	SB4

Las distintas áreas de memoria y sus rangos se presentan a continuación en forma de resumen en la tabla A.4.

Descripción	CPU 221	CPU 222	CPU 224	CPU 224XP	CPU 226
Tamaño del programa de usuario con edición en runtime sin edición en runtime	 4096 bytes 4096 bytes	 4096 bytes 4096 bytes	 8192 bytes 12288 bytes	 12288 bytes 16384 bytes	 16384 bytes 24576 bytes
Tamaño de los datos de usuario	2048 bytes	2048 bytes	8192 bytes	10240 bytes	10240 bytes
Imagen de proceso de las entradas	I0.0 a I15.7	I0.0 a I15.7	I0.0 a I15.7	I0.0 a I15.7	I0.0 a I15.7
Imagen de proceso de las salidas	Q0.0 a Q15.7	Q0.0 a Q15.7	Q0.0 a Q15.7	Q0.0 a Q15.7	Q0.0 a Q15.7
Entradas analógicas (sólo lectura)	AIW0 a AIW30	AIW0 a AIW30	AIW0 a AIW62	AIW0 a AIW62	AIW0 a AIW62
Salidas analógicas (sólo escritura)	AQW0 a AQW30	AQW0 a AQW30	AQW0 a AQW62	AQW0 a AQW62	AQW0 a AQW62
Memoria de variables (V)	VB0 a VB2047	VB0 a VB2047	VB0 a VB8191	VB0 a VB10239	VB0 a VB10239
Memoria local (L)[1]	LB0 a LB63	LB0 a LB63	LB0 a LB63	LB0 a LB63	LB0 a LB63
Área de marcas (M)	M0.0 a M31.7	M0.0 a M31.7	M0.0 a M31.7	M0.0 a M31.7	M0.0 a M31.7
Marcas especiales (SM) Sólo lectura	SM0.0 a SM179.7 SM0.0 a SM29.7	SM0.0 a SM299.7 SM0.0 a SM29.7	SM0.0 a SM549.7 SM0.0 a SM29.7	SM0.0 a SM549.7 SM0.0 a SM29.7	SM0.0 a SM549.7 SM0.0 a SM29.7
Temporizadores	256 (T0 a T255)	256 (T0 a T255)	256 (T0 a T255)	256 (T0 a T255)	256 (T0 a T255)
Retardo a la conexión con memoria 1 ms	T0, T64	T0, T64	T0, T64	T0, T64	T0, T64
10 ms	T1 a T4 y T65 a T68	T1 a T4 y T65 a T68	T1 a T4 y T65 a T68	T1 a T4 y T65 a T68	T1 a T4 y T65 a T68
100 ms	T5 a T31 y T69 a T95	T5 a T31 y T69 a T95	T5 a T31 y T69 a T95	T5 a T31 y T69 a T95	T5 a T31 y T69 a T95
Retardo a la conexión/desconexión 1 ms	T32, T96	T32, T96	T32, T96	T32, T96	T32, T96
10 ms	T33 a T36 y T97 a T100	T33 a T36 y T97 a T100	T33 a T36 y T97 a T100	T33 a T36 y T97 a T100	T33 a T36 y T97 a T100
100 ms	T37 a T63 y T101 a T255	T37 a T63 y T101 a T255	T37 a T63 y T101 a T255	T37 a T63 y T101 a T255	T37 a T63 y T101 a T255
Contadores	C0 a C255	C0 a C255	C0 a C255	C0 a C255	C0 a C255
Contadores rápidos	HC0 a HC5	HC0 a HC5	HC0 a HC5	HC0 a HC5	HC0 a HC5
Relés de control secuencial (S)	S0.0 a S31.7	S0.0 a S31.7	S0.0 a S31.7	S0.0 a S31.7	S0.0 a S31.7
Acumuladores	AC0 a AC3	AC0 a AC3	AC0 a AC3	AC0 a AC3	AC0 a AC3
Saltos a metas	0 a 255	0 a 255	0 a 255	0 a 255	0 a 255
Llamadas a subrutinas	0 a 63	0 a 63	0 a 63	0 a 63	0 a 127
Rutinas de interrupción	0 a 127	0 a 127	0 a 127	0 a 127	0 a 127
Detectar flanco positivo/negativo	256	256	256	256	256
Lazos PID	0 a 7	0 a 7	0 a 7	0 a 7	0 a 7
Puertos	Puerto 0	Puerto 0	Puerto 0	Puerto 0, puerto 1	Puerto 0, puerto 1

[1] STEP 7Micro/WIN (versión 3.0 o posterior) reserva LB60 a LB63.

Tabla A.4. Áreas de Memoria PLC S7-200

A.3.1 DIRECCIONAMIENTO DIRECTO

La S7-200 almacena información en diferentes áreas de la memoria que tienen direcciones unívocas. Es posible indicar explícitamente la dirección a la que se desea acceder.

El programa puede acceder entonces directamente a la información. La tabla A.5 muestra el rango de números enteros representables en diversos tamaños de datos.

Representación	Byte (B)	Palabra (W)	Palabra doble (D)
Entero sin signo	0 a 255 0 a FF	0 a 65.535 0 a FFFF	0 a 4.294.967.295 0 a FFFF FFFF
Entero con signo	-128 a +127 80 a 7F	-32.768 a +32.767 8000 a 7FFF	-2.147.483.648 a +2.147.483.647 8000 0000 a 7FFF FFFF
Real IEEE de 32 bits en coma flotante	*No aplicable*	*No aplicable*	+1,175495E-38 a +3,402823E+38 (positivo) -1,175495E-38 a -3,402823E+38 (negativo)

Tabla. A.5. Rangos decimales y hexadecimales para los diferentes tamaños de datos.

Para acceder a un bit en un área de memoria es preciso indicar la dirección del mismo, compuesta por un identificador de área, la dirección del byte y el número del bit. La figura A.10 muestra un ejemplo de direccionamiento de un bit (denominado también direccionamiento "byte.bit"). En el ejemplo, el área de memoria y la dirección del byte (I = entrada y 3 = byte 3) van seguidas de un punto decimal (".") que separa la dirección del bit (bit 4).

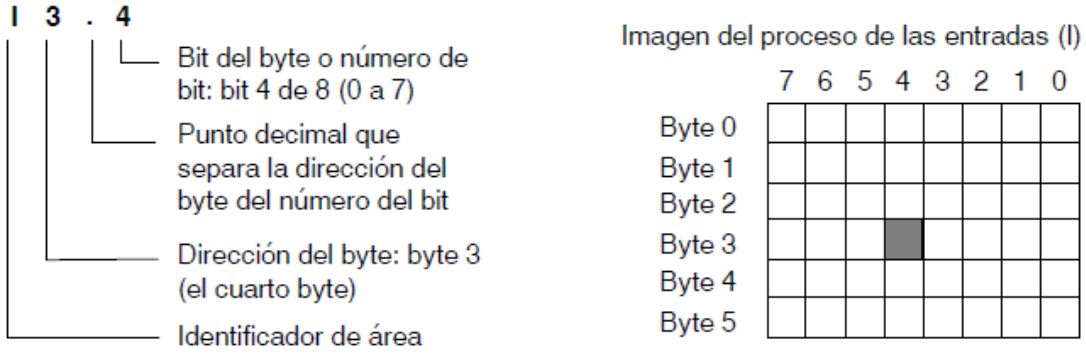


Fig. A.10. Direccionamiento byte.bit.

Utilizando el formato de dirección de byte se puede acceder a los datos de la mayoría de las áreas de memoria (V, I, Q, M, S y SM) en formato de bytes, palabras o palabras dobles. La dirección de un byte, de una palabra o de una palabra doble de datos en la memoria se indica de forma similar a la dirección de un bit. Esta última está compuesta por un identificador de área, el tamaño de los datos y la dirección inicial del valor del byte, de la palabra o de la palabra doble, como muestra la figura A.11.

Para acceder a los datos comprendidos en otras áreas de la memoria (por ejemplo, T, C, HC y acumuladores) es preciso utilizar una dirección compuesta por un identificador de área y un número de elemento.

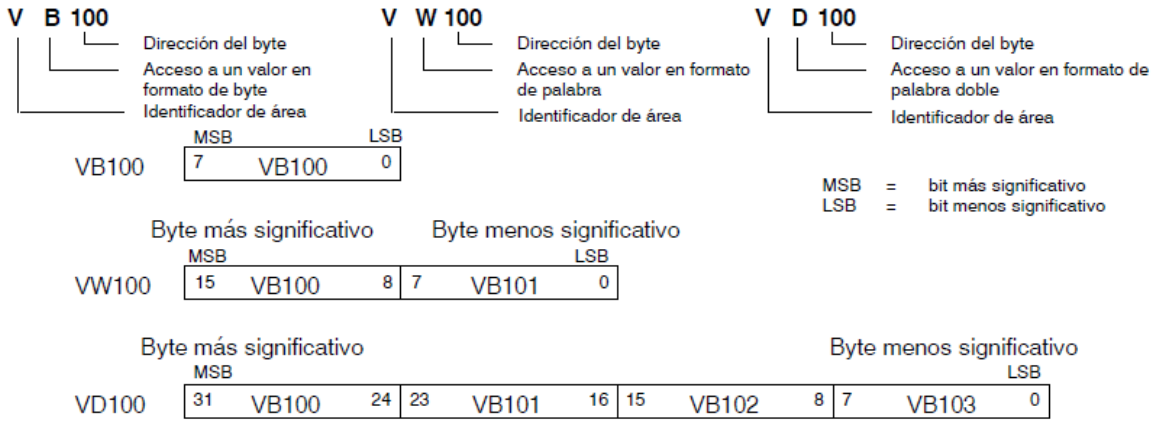


Fig. A.11. Acceso a una misma dirección en formato de byte, palabra y palabra doble.

A.3.2 DIRECCIONAMIENTO INDIRECTO

El direccionamiento indirecto utiliza un puntero para acceder a los datos de la memoria. Los punteros son valores de palabra doble que señalan a una dirección diferente en la memoria.

Como punteros sólo se pueden utilizar direcciones de la memorias V y L, o bien los acumuladores (AC1, AC2 y AC3). Para crear un puntero se debe utilizar la operación Transferir palabra doble, con objeto de transferir la dirección indirecta a la del puntero. Los punteros también se pueden transferir a una subrutina en calidad de parámetros.

El S7-200 permite utilizar punteros para acceder a las siguientes áreas de memoria: I, Q, V, M, S, AI, AQ, SM, T (sólo el valor actual) y C (sólo el valor actual). El direccionamiento indirecto no se puede utilizar para acceder a un bit individual ni para acceder a las áreas de memoria HC o L.

Para acceder indirectamente a los datos de una dirección de la memoria es preciso crear un puntero a esa dirección, introduciendo para ello un carácter “&” y la dirección a la que se desea acceder. El operando de entrada de la operación debe ir precedido de un carácter “&” para determinar que a la dirección indicada por el operando de salida (es decir, el puntero) se debe transferir la dirección y no su contenido.

Introduciendo un asterisco (*) delante de un operando de una operación, se indica que el operando es un puntero. En el ejemplo que muestra la figura A.12, *AC1 significa que AC1 es el puntero del valor de palabra indicado por la operación Transferir palabra (MOVW). En este ejemplo, los valores almacenados en VB200 y VB201 se transfieren al acumulador AC0.

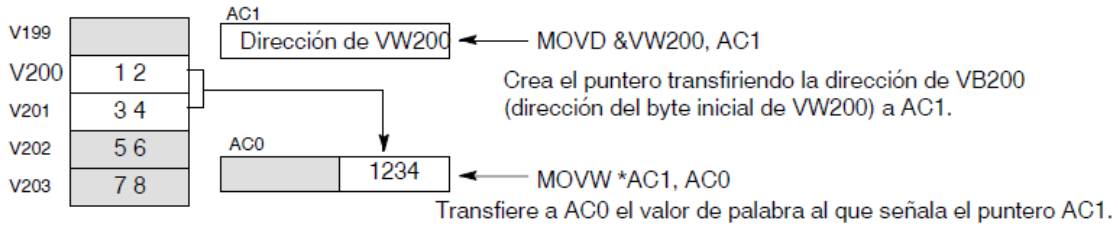


Fig. A.12. Crear y utilizar un puntero.

Como muestra la figura A.13, es posible modificar el valor de los punteros. Puesto que los punteros son valores de 32 bits, para cambiarlos es preciso utilizar operaciones de palabra doble.

Las operaciones aritméticas simples, tales como sumar o incrementar, se pueden utilizar para modificar los valores de los punteros.

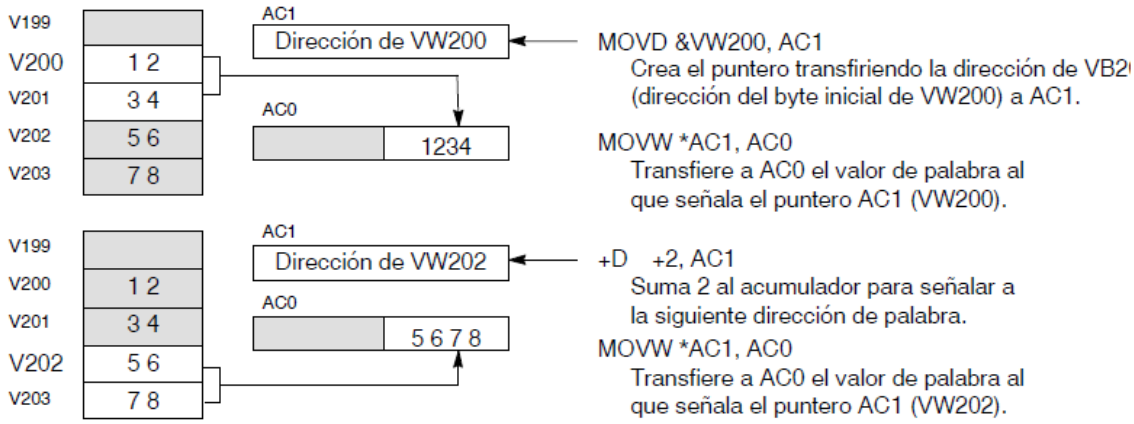
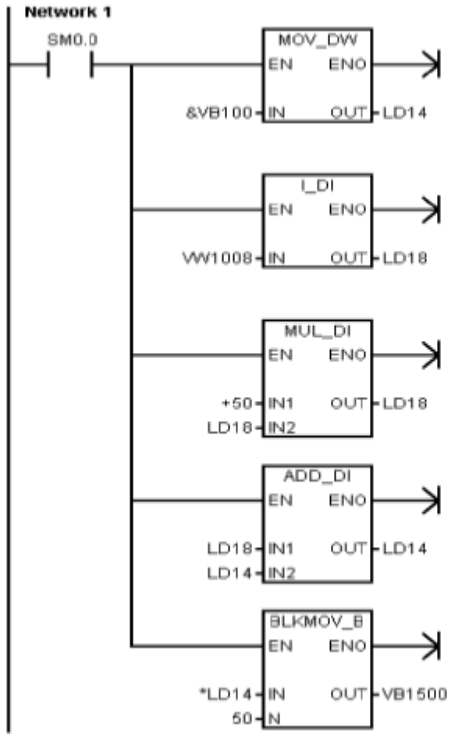


Fig. A.13. Modificar un puntero.

Recuerde que debe indicar el tamaño de los datos a los que desea acceder. Para acceder a un byte, incremente el valor del puntero en 1. Para acceder a una palabra, o bien al valor actual de un temporizador o de un contador, incremente el valor del puntero en 2. Para acceder a una palabra doble, incremente el valor del puntero en 4.

Programa de ejemplo de un puntero para acceder a datos de una tabla

El presente ejemplo utiliza LD14 como puntero a una receta almacenada en una tabla de recetas que comienza en VB100. En el presente ejemplo, VW1008 almacena el índice en una determinada receta de la tabla. Si cada una de las recetas tiene una longitud de 50 bytes, el índice se deberá multiplicar por 50 para obtener el offset de la dirección inicial de una determinada receta. Sumando el offset al puntero es posible acceder a la receta en cuestión. En el presente ejemplo, la receta se copia a los 50 bytes que comienzan en VB1500.




```
Network 1   //Transferir una receta de una tabla de recetas:
            // -  Las recetas tienen una longitud de 50 bytes c/u.
            // -  El parámetro del índice (VW1008) identifica
            //    la receta a cargar.
            //1º Crear un puntero a la dirección inicial
            //    de la tabla de recetas.
            //2º Convertir el índice de la receta a
            //    un valor de palabra doble.
            //3º Multiplicar el offset para ajustar
            //    el tamaño de cada receta.
            //4º Sumar al puntero el offset ajustado.
            //5º Transferir la receta seleccionada a
            //    VB1500 hasta VB1549.
LD      SM0.0
MOVD    &VB100, LD14
ITD     VW1008, LD18
*D      +50, LD18
+D      LD18, LD14
BMB     *LD14, VB1500, 50
```

Fig. A.14 Ejemplo de un puntero para acceder a los datos de una tabla

A.4 PAQUETE DE PROGRAMACIÓN STEP 7-MICRO/WIN

El paquete de programación STEP7-Micro/WIN constituye un entorno de fácil manejo para desarrollar, editar y observar el programa necesario con objeto de controlar la aplicación. STEP7-Micro/WIN comprende tres editores que permiten desarrollar de forma cómoda y eficiente el programa de control. Para encontrar fácilmente las informaciones necesarias, STEP7-Micro/WIN incorpora una completa Ayuda en pantalla, así como asistentes de configuración para las distintas formas de comunicación del micro PLC.

El software de programación STEP 7-Micro/WIN ofrece potentes herramientas que permiten ahorrar mucho tiempo, lo que redunda en un enorme ahorro de costos durante el trabajo cotidiano. El software de programación se maneja de forma análoga a las aplicaciones estándar de Windows. Micro/WIN está dotado de todas las herramientas necesarias para programar la serie completa de PLCs S7-200. Para ello, tiene a su disposición tanto un repertorio de instrucciones de gran rendimiento como la programación conforme a la norma IEC 1131.

Al programar es posible conmutar a voluntad entre los editores estándar KOP/FUP y AWL. Numerosas funciones nuevas y nuevos asistentes perfeccionados simplifican más la programación. Y STEP 7-Micro/WIN 4.0 ofrece otras cosas más, entre ellas una memoria de datos segmentada, un manejo más eficaz de las estructuras de programas e instrucciones, o funciones de diagnóstico como un LED personalizado, historial de fallos o edición en runtime y descarga online.

Requisitos del sistema

STEP7-Micro/WIN se puede ejecutar en un ordenador (PC), o bien en una unidad de programación de Siemens (por ejemplo, en una PG-760). El PC o la PG deberán cumplir los siguientes requisitos mínimos:

- ✓ Sistema operativo: Windows2000, Windows XP (Professional o Home)
- ✓ Espacio de 100MB libres en el disco duro (como mínimo)
- ✓ Dispositivo Mouse (recomendado)

Instalar STEP 7 - Micro /WIN

Inserte el CD de STEP7-Micro/WIN en la unidad de CD-ROM. El asistente de instalación arrancará automáticamente y le conducirá por el proceso de instalación.

Para instalar STEP7-Micro/WIN en el sistema operativo Windows NT, Windows2000 o Windows XP (Professional o Home), deberá iniciar la sesión con derechos de administrador.

A.4.1 DESCRIPCIÓN DE LA VENTANA PRINCIPAL

Algunos de los elementos más importantes de la ventana principal de STEP 7 - Micro /WIN se muestran a continuación:

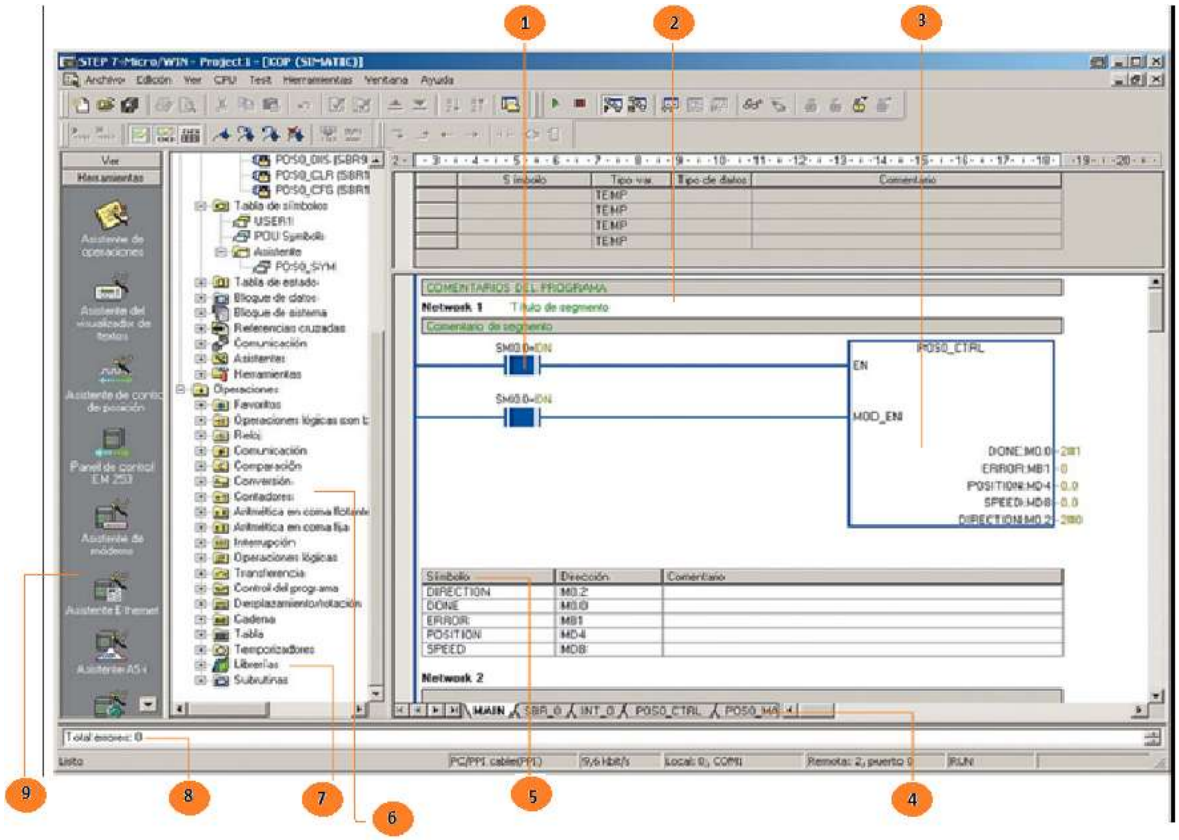


Fig. A.15. Ventana principal STEP 7.

1. Funciones online integradas
   - ✓ Edición en runtime
   - ✓ Estado online

2. Editor de programas.

3. Posibilidad de ayuda contextual online para todas las funciones

4. Programación estructurada con subprogramas
   - ✓ Subprogramas parametrizables
   - ✓ Subprogramas protegidos por contraseña
   - ✓ Activación repetida en el programa de usuario
   - ✓ Posibilidad de importar/exportar subprogramas

5. Notación simbólica y tablas de símbolos claras e informativas
   - ✓ Tablas de símbolos estándar
   - ✓ Tablas autodefinidas.
6. Árbol de operaciones.

7. Programación estructurada con librerías
   - ✓ Protocolo USS para el control de accionamientos
   - ✓ Librería Modbus
   - ✓ Librerías autodefinidas

8. Búsqueda y eliminación de errores
   - ✓ Búsqueda rápida de errores online
   - ✓ Localización de errores mediante clic con el ratón.

9. Barra de navegación.

A.4.2 UTILIZAR STEP 7-MICRO/WIN PARA CREAR PROGRAMAS

Para iniciar STEP 7-Micro/WIN, haga doble clic en el icono de STEP 7-Micro/WIN o elija los comandos **Inicio > SIMATIC > STEP 7 Micro/WIN 32 V4.0**.

Las barras de herramientas incorporan botones de método abreviado para los comandos de menú de uso frecuente. Estas barras se pueden mostrar u ocultar.

La barra de navegación comprende iconos que permiten acceder a las diversas funciones de programación de STEP 7-Micro/WIN.

En el árbol de operaciones se visualizan todos los objetos del proyecto y las operaciones para crear el programa de control. Para insertar operaciones en el programa, puede utilizar el método de “arrastrar y soltar” desde el árbol de operaciones, o bien hacer doble

clic en una operación con objeto de insertarla en la posición actual del cursor en el editor de programas.

El editor de programas contiene el programa y una tabla de variables locales donde se pueden asignar nombres simbólicos a las variables locales temporales. Las subrutinas y las rutinas de interrupción se visualizan en forma de fichas en el borde inferior del editor de programas. Para acceder a las subrutinas, a las rutinas de interrupción o al programa principal, haga clic en la ficha en cuestión.

STEP 7-Micro/WIN incorpora los tres editores de programas siguientes: Esquema de contactos (KOP), Lista de instrucciones (AWL) y Diagrama de funciones (FUP). Con algunas restricciones, los programas creados con uno de estos editores se pueden visualizar y editar con los demás.

Funciones del editor AWL

El editor AWL visualiza el programa textualmente. Permite crear programas de control introduciendo la nemotécnica de las operaciones. El editor AWL sirve para crear ciertos programas que, de otra forma, no se podrían programar con los editores KOP ni FUP. Ello se debe a que AWL es el lenguaje nativo del S7-200, a diferencia de los editores gráficos, sujetos a ciertas restricciones para poder dibujar los diagramas correctamente. Como muestra la figura A.15, esta forma textual es muy similar a la programación en lenguaje ensamblador.

```
LD    I0.0    //Leer una entrada
A     I0.1    //AND con otra entrada
=     Q0.0    //Escribir en el valor en
              //la salida 1
```

Fig. A.16. Programa ejemplo AWL.

Considere los siguientes aspectos importantes cuando desee utilizar el editor AWL:

- ✓ El lenguaje AWL es más apropiado para los programadores expertos.
- ✓ En algunos casos, AWL permite solucionar problemas que no se podrían resolver fácilmente con los editores KOP o FUP.
- ✓ El editor AWL soporta sólo el juego de operaciones SIMATIC.
- ✓ En tanto que el editor AWL se puede utilizar siempre para ver o editar programas creados con los editores KOP o FUP, lo contrario no es posible en todos los casos. Los editores KOP o FUP no siempre se pueden utilizar para visualizar un programa que se haya creado en AWL.

Funciones del editor KOP

El editor KOP visualiza el programa gráficamente, de forma similar a un esquema de circuitos.

Los programas KOP hacen que el programa emule la circulación de corriente eléctrica desde una fuente de alimentación, a través de una serie de condiciones lógicas de entrada que, a su vez, habilitan condiciones lógicas de salida. Los programas KOP incluyen una barra de alimentación izquierda que está energizada. Los contactos cerrados permiten que la corriente circule por ellos hasta el siguiente elemento, en tanto que los contactos abiertos bloquean el flujo de energía.

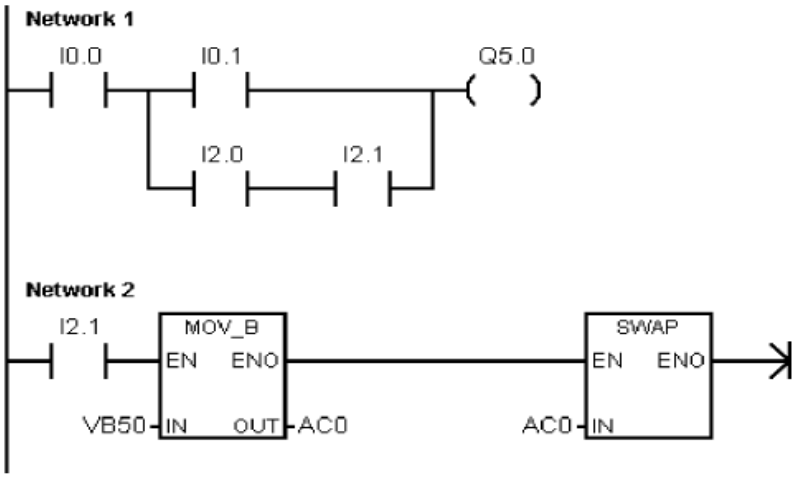


Fig. A.17. Programa ejemplo KOP.

La lógica se divide en segmentos (”networks”). El programa se ejecuta un segmento tras otro, de izquierda a derecha y luego de arriba a abajo. La figura A.17 muestra un ejemplo de un programa KOP. Las operaciones se representan mediante símbolos gráficos que incluyen tres formas básicas.

Los contactos representan condiciones lógicas de entrada, tales como interruptores, botones o condiciones internas.

Las bobinas representan condiciones lógicas de salida, tales como lámparas, arrancadores de motor, relés interpuestos o condiciones internas de salida.

Los cuadros representan operaciones adicionales, tales como temporizadores, contadores u operaciones aritméticas.

Considere los siguientes aspectos importantes cuando desee utilizar el editor KOP:

- ✓ El lenguaje KOP les facilita el trabajo a los programadores principiantes.

- ✓ La representación gráfica es fácil de comprender, siendo popular en el mundo entero.

- ✓ El editor KOP se puede utilizar con los juegos de operaciones SIMATIC e IEC 1131-3.

- ✓ El editor AWL se puede utilizar siempre para visualizar un programa creado en KOP SIMATIC.

Funciones del editor FUP

El editor FUP visualiza el programa gráficamente, de forma similar a los circuitos de puertas lógicas. En FUP no existen contactos ni bobinas como en el editor KOP, pero sí hay operaciones equivalentes que se representan en forma de cuadros.

La figura A.18 muestra un ejemplo de un programa FUP.

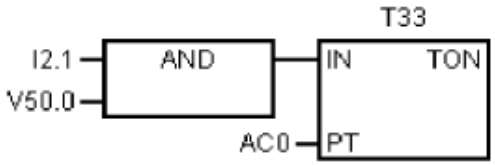


Fig. A.18. Programa ejemplo FUP.

El lenguaje de programación FUP no utiliza las barras de alimentación izquierda ni derecha. Sin embargo, el término "circulación de corriente" se utiliza para expresar el concepto análogo del flujo de señales por los bloques lógicos FUP.

El recorrido "1" lógico por los elementos FUP se denomina circulación de corriente. El origen de una entrada de circulación de corriente y el destino de una salida de circulación de corriente se pueden asignar directamente a un operando.

La lógica del programa se deriva de las conexiones entre las operaciones de cuadro. Ello significa que la salida de una operación (por ejemplo, un cuadro AND) se puede utilizar para habilitar otra operación (por ejemplo, un temporizador), con objeto de crear la lógica de control necesaria. Estas conexiones permiten solucionar numerosos problemas lógicos.

Considere los siguientes aspectos importantes cuando desee utilizar el editor FUP:

- ✓ El estilo de representación en forma de puertas gráficas se adecúa especialmente para observar el flujo del programa.
- ✓ El editor FUP soporta los juegos de operaciones SIMATIC e IEC 1131-3.
- ✓ El editor AWL se puede utilizar siempre para visualizar un programa creado en SIMATIC FUP.

Juegos de operaciones SIMATIC e IEC 1131-3

La mayoría de los sistemas de automatización ofrecen los mismos tipos básicos de operaciones.

No obstante, existen pequeñas diferencias en cuanto al aspecto, al funcionamiento, etc. de los productos de los distintos fabricantes. Durante los últimos años, la Comisión Electrotécnica Internacional (CEI) o International Electrotechnical Commission (IEC) ha desarrollado una norma global dedicada a numerosos aspectos de la programación de autómatas programables (denominados "sistemas de automatización" en la terminología SIMATIC). El objetivo de esta norma es que los diferentes fabricantes de autómatas programables ofrezcan operaciones similares tanto en su aspecto como en su funcionamiento.

El S7-200 ofrece dos juegos de operaciones que permiten solucionar una gran variedad de tareas de automatización. El juego de operaciones IEC cumple con la norma IEC 1131-3 para la programación de autómatas programables (PLCs), en tanto que el juego de operaciones SIMATIC se ha diseñados especialmente para el S7-200.

Si en STEP 7-Micro/WIN está ajustado el modo IEC, junto a las operaciones no definidas en la norma IEC 1131-3 se visualizará un diamante rojo en el árbol de operaciones.

Existen algunas diferencias básicas entre los juegos de operaciones SIMATIC e IEC:

- ✓ El juego de operaciones IEC se limita a las operaciones estándar comunes entre los fabricantes de autómatas programables. Algunas operaciones incluidas en el juego SIMATIC no están normalizadas en la norma IEC 1131-3. Éstas se pueden utilizar en calidad de operaciones no normalizadas. No obstante, en este caso, el programa ya no será absolutamente compatible con la norma IEC 1131-3.

- ✓ Algunos cuadros IEC soportan varios formatos de datos. A menudo, esto se denomina sobrecarga. Por ejemplo, en lugar de tener cuadros aritméticos por separado, tales como ADD_I (Sumar enteros), ADD_R (Sumar reales) etc., la operación ADD definida en la norma IEC examina el formato de los datos a sumar y selecciona automáticamente la operación correcta en el S7-200. Así se puede ahorrar tiempo al diseñar los programas.

- ✓ Si se utilizan las operaciones IEC, se comprueba automáticamente si los parámetros de la operación corresponden al formato de datos correcto (por ejemplo, entero con signo o entero sin signo). Por ejemplo, si ha intentado introducir un valor de entero en una operación para la que se deba utilizar un valor binario (on/off), se indicará un error. Esta función permite reducir los errores de sintaxis de programación.

Considere los siguientes aspectos a la hora de seleccionar el juego de operaciones (SIMATIC o IEC):

- ✓ Por lo general, el tiempo de ejecución de las operaciones SIMATIC es más breve. Es posible que el tiempo de ejecución de algunas operaciones IEC sea más prolongado.

- ✓ El funcionamiento de algunas operaciones IEC (por ejemplo, temporizadores, contadores, multiplicación y división) es diferente al de sus equivalentes en SIMATIC.

- ✓ Las operaciones SIMATIC se pueden utilizar en los tres editores de programas disponibles (KOP, AWL y FUP). Las operaciones IEC sólo se pueden utilizar en los editores KOP y FUP.

- ✓ El funcionamiento de las operaciones IEC es igual en las diferentes marcas de autómatas programables (PLCs). Los conocimientos acerca de cómo crear un programa compatible con la norma IEC se pueden nivelar a lo largo de las plataformas de PLCs.

- ✓ Aunque la norma IEC define una menor cantidad de operaciones de las disponibles en el juego de operaciones SIMATIC, en los programas IEC se pueden incluir siempre también operaciones SIMATIC.

- ✓ La norma IEC 1131-3 especifica que las variables se deben declarar tipificadas, soportando que el sistema verifique el tipo de datos.

Convenciones utilizadas en los editores de programas

En todos los editores de programas de STEP 7-Micro/WIN rigen las convenciones siguientes:

- ✓ Si un nombre simbólico (por ejemplo, #var1) va antecedido de un signo de número (#), significa que se trata de un símbolo local.

- ✓ En las operaciones IEC, el símbolo % identifica una dirección directa.

- ✓ El símbolo de operando “?.?” ó “????” indica que el operando se debe configurar.

  Los programas KOP se dividen en segmentos denominados “networks”. Un segmento es una red organizada, compuesta por contactos, bobinas y cuadros que se interconectan para formar un circuito completo. No se permiten los cortocircuitos, ni los circuitos abiertos, ni la circulación de corriente inversa. STEP 7-Micro/WIN ofrece la posibilidad de crear comentarios para cada uno de los

segmentos del programa KOP. El lenguaje FUP utiliza el concepto de segmentos para subdividir y comentar el programa.

Los programas AWL no utilizan segmentos. Sin embargo, la palabra clave NETWORK se puede utilizar para estructurar el programa.

Convenciones específicas del editor KOP

En el editor KOP, las teclas de función F4, F6 y F9 sirven para acceder a los contactos, los cuadros y las bobinas. El editor KOP utiliza las convenciones siguientes:

- ✓ El símbolo "--->>" representa un circuito abierto o una conexión necesaria para la circulación de corriente.

- ✓ El símbolo " " indica que la salida es una conexión opcional para la circulación de corriente en una operación que se puede disponer en cascada o conectar en serie.

- ✓ El símbolo ">>" indica que se puede utilizar la circulación de corriente.

Convenciones específicas del editor FUP

En el editor FUP, las teclas de función F4, F6 y F9 sirven para acceder a las operaciones AND y OR, así como a las operaciones con cuadros. El editor FUP utiliza las convenciones siguientes:

- ✓ El símbolo "--->>" en un operando EN es un indicador de circulación de corriente o de operando. También puede representar un circuito abierto o una conexión necesaria para la circulación de corriente.

- ✓ El símbolo " " indica que la salida es una conexión opcional para la circulación de corriente en una operación que se puede disponer en cascada o conectar en serie.

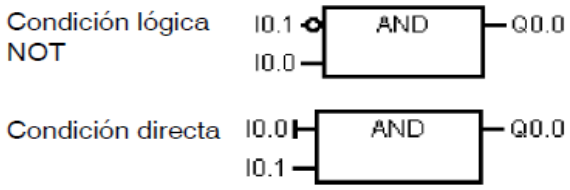


Fig. A.19. Convenciones FUP.

- ✓ Los símbolos "<<" y ">>" indican que se puede utilizar bien sea un valor, o bien la circulación de corriente.

Símbolo de negación: La condición lógica NOT (la condición invertida) del operando o la corriente se representa mediante un círculo pequeño en la entrada. En la figura A.19 Q0.0 es igual al NOT de I0.0 AND I0.1.

Los símbolos de negación sólo son aplicables a las señales booleanas, que se pueden indicar en forma de parámetros o de circulación de corriente.

- ✓ Indicadores directos: Como muestra la figura A.19, el editor FUP visualiza una condición directa de un operando booleano mediante una línea vertical en la entrada de una operación FUP. El indicador directo causa una lectura directa de la entrada física indicada.
  Los indicadores directos sólo son aplicables a las entradas físicas.

- ✓ Cuadro sin entradas ni salidas: Un cuadro sin entradas ni salidas indica que la operación no depende de la circulación de corriente.

La cantidad de operandos se puede incrementar hasta 32 entradas en el caso de las operaciones AND y OR. Para agregar o quitar operandos, utilice las teclas "+" y "-" del teclado, respectivamente.

Definición de EN/ENO

EN (entrada de habilitación) es una entrada booleana para los cuadros KOP y FUP. Para que la operación se pueda ejecutar, el estado de señal de la entrada EN deberá ser "1" (ON). En AWL, las operaciones no tienen una entrada EN, pero el valor del nivel superior de la pila deberá ser un "1" lógico para poder ejecutar la operación AWL correspondiente.
ENO (salida de habilitación) es una salida booleana para los cuadros KOP y FUP. Si el estado de señal de la entrada EN es "1" y el cuadro ejecuta la función sin errores, la salida ENO conducirá corriente al siguiente elemento. Si se detecta un error en la ejecución del cuadro, la circulación de corriente se detendrá en el cuadro que ha generado el error.

En AWL no existe la salida ENO, pero las operaciones AWL correspondientes a las funciones KOP y FUP con salidas ENO activarán un bit ENO especial. A este bit se accede mediante la operación AND ENO (AENO), pudiendo utilizarse para generar el mismo efecto que el bit ENO de un cuadro.

Los operandos y los tipos de datos EN/ENO no figuran en la tabla de operandos válidos de las operaciones, puesto que son idénticos para todas las operaciones KOP y FUP. La tabla A.6 muestra los operandos y tipos de datos EN/ENO para KOP y FUP, siendo aplicables a todas las operaciones KOP y FUP descritas en el presente manual.

Editor de programas	Entradas/salidas	Operandos	Tipos de datos
KOP	EN, ENO	Circulación de corriente	BOOL
FUP	EN, ENO	I, Q, V, M, SM, S, T, C, L	BOOL

Tabla. A.6. Operandos y tipos de datos EN/ENO para KOP y FUP.

Entradas condicionadas e incondicionadas

En KOP y FUP, un cuadro o una bobina que dependa de la circulación de corriente aparecerá conectado a algún elemento a la izquierda. Una bobina o un cuadro que no dependa de la circulación de corriente se mostrará con una conexión directa a la barra de alimentación izquierda. La tabla A.7 muestra dos entradas: una condicionada y otra incondicionada.

Circulación de corriente	KOP	FUP
Operación dependiente de la circulación de corriente (condicionada)	1 JMP	JMP
Operación independiente de la circulación de corriente (incondicionada)	NEXT	NEXT

Tabla. A.7. Representación de entradas condicionadas.

Operaciones sin salidas

Los cuadros que no se puedan conectar en cascada se representan sin salidas booleanas. Estos cuadros incluyen las llamadas a subrutinas, JMP y CRET. También hay bobinas KOP que sólo se pueden disponer en la barra de alimentación izquierda, incluyendo las operaciones Definir meta, NEXT, Cargar relé de control secuencial, Fin condicionado del relé de control secuencial y Fin del relé de control secuencial. Estas operaciones se representan en FUP en forma de cuadros con entradas sin meta y sin salidas.

Operaciones de comparación

Las operaciones de comparación se ejecutan sin tener en cuenta el estado de señal. Si el estado es “0” (FALSO), el estado de señal de la salida también será “0” (FALSO). Si el estado de señal es “1” (VERDADERO), la salida se activará dependiendo del resultado de la comparación. Las operaciones de comparación FUP (SIMATIC), LD (IEC) y FBD (IEC) se representan con cuadros, aunque la operación se ejecute en forma de contacto.

A.4.3 EDITOR DE BLOQUE DE DATOS

El editor de bloques de datos permite asignar datos iníciales sólo a la memoria V (memoria de variables). Se pueden efectuar asignaciones a bytes, palabras o palabras dobles de la memoria V. Los comentarios son opcionales.

El editor de bloques de datos es un editor de texto de libre formato. Por tanto, no hay campos específicos definidos para un tipo determinado de información. Tras introducir una línea, pulse la tecla INTRO. El editor formatea la línea (alinea las columnas de direcciones, los datos y los comentarios; pone las direcciones de la memoria V en mayúsculas) y la visualiza de nuevo. Si pulsa CTRL-INTRO, tras completar una línea de

asignación, la dirección se incrementará automáticamente a la siguiente dirección disponible.

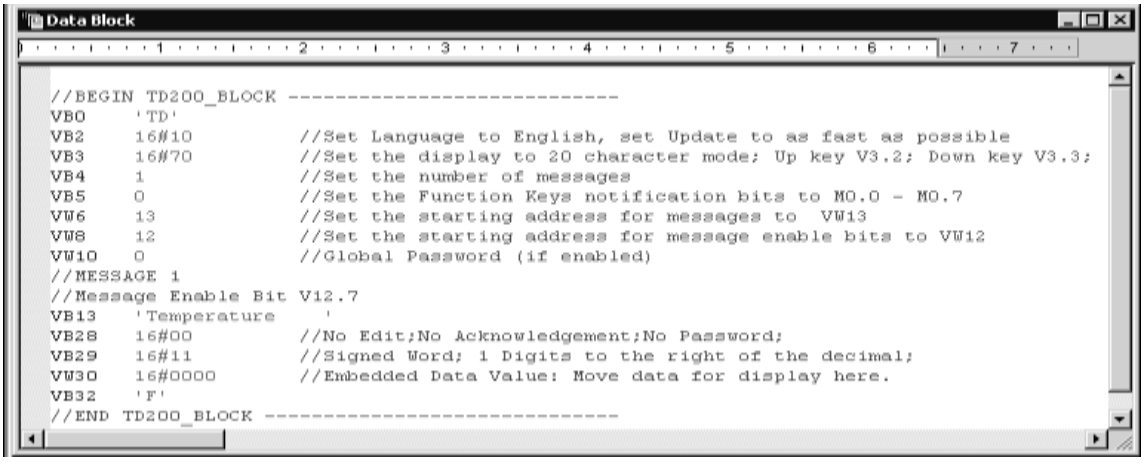


Fig. A.20. Representación de entradas condicionadas

El editor asigna una cantidad suficiente de la memoria V, en función de las direcciones que se hayan asignado previamente, así como del tamaño (byte, palabra o palabra doble) del (de los) valor(es) de datos.

La primera línea del bloque de datos debe contener una asignación de dirección explícita. Las líneas siguientes pueden contener asignaciones de direcciones explícitas o implícitas. El editor asignará una dirección implícita si se introducen varios valores de datos tras asignarse una sola dirección o si se introduce una línea que contenga únicamente valores de datos.

En el editor de bloques de datos se pueden utilizar mayúsculas y minúsculas. Además, es posible introducir comas, tabuladores y espacios que sirven de separadores entre las direcciones y los valores de datos.

A.4.4 UTILIZAR LA TABLA DE SÍMBOLOS PARA EL DIRECCIONAMIENTO SIMBÓLICO DE VARIABLES

En la tabla de símbolos es posible definir y editar los símbolos a los que pueden acceder los nombres simbólicos en cualquier parte del programa. Es posible crear varias tablas de símbolos.

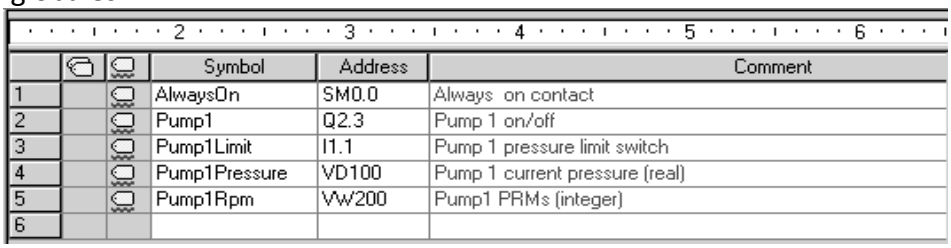
La tabla de símbolos incorpora también una ficha que contiene los símbolos definidos por el sistema utilizables en el programa de usuario. La tabla de símbolos se denomina también tabla de variables globales.

A los operandos de las operaciones se les pueden asignar direcciones absolutas o simbólicas.

Una dirección absoluta utiliza el área de memoria y un bit o un byte para identificar la dirección.

Una dirección simbólica utiliza una combinación de caracteres alfanuméricos para identificar la dirección.

En los programas SIMATIC, los símbolos globales se asignan utilizando la tabla de símbolos. En los programas IEC, los símbolos globales se asignan utilizando la tabla de variables globales.



	Symbol	Address	Comment
1	AlwaysOn	SM0.0	Always on contact
2	Pump1	Q2.3	Pump 1 on/off
3	Pump1Limit	I1.1	Pump 1 pressure limit switch
4	Pump1Pressure	VD100	Pump 1 current pressure (real)
5	Pump1Rpm	VW200	Pump1 PRMs (integer)
6			

Fig. A.21. Tabla de símbolos.

Para asignar un símbolo a una dirección:

1. Haga clic en el icono “Tabla de símbolos” en la barra de navegación para abrir la tabla de símbolos.
2. En la columna “Nombre simbólico”, teclee el nombre del símbolo (por ejemplo, “Entrada1”). Un nombre simbólico puede comprender 23 caracteres como máximo.
3. En la columna “Dirección”, teclee la dirección (por ejemplo, I0.0).
4. Si está utilizando la tabla de variables globales (IEC), introduzca un valor en la columna “Tipo de datos” o seleccione uno del cuadro de lista.

Es posible crear varias tablas de símbolos. No obstante, una misma cadena no se puede utilizar más de una vez como símbolo global, ni en una misma tabla ni en tablas diferentes.

Utilizar variables locales

La tabla de variables locales del editor de programas se puede utilizar para asignar variables que existan únicamente en una subrutina o en una rutina de interrupción individual (v. fig. A.22).

Las variables locales se pueden usar como parámetros que se transfieren a una subrutina, lo que permite incrementar la portabilidad y la reutilización de la subrutina.

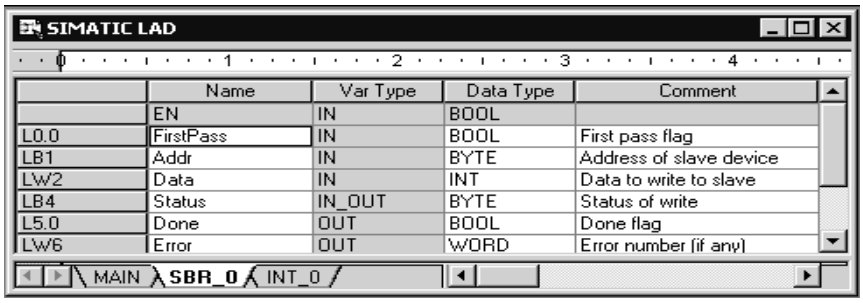


Fig. A.22. Tabla de variables locales.

A.5 CONFIGURACIÓN DE LA COMUNICACIÓN (CABLE PC/PPI)

Conectar el cable multimaestro RS-232/PPI

La figura A.23 muestra un cable multimaestro RS-232/PPI que conecta el S7-200 con la unidad de programación.
Para conectar el cable:

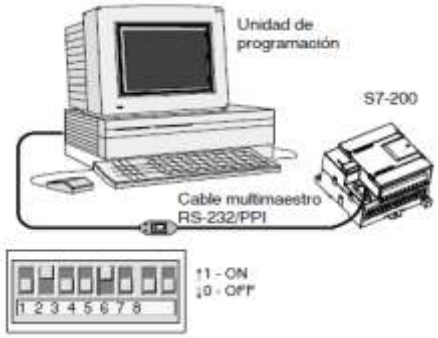


Fig. A.23. Conectar el cable multimaestro RS-232/PPI.

1. Una el conector RS-232 (identificado con “PC”) del cable multimaestro RS-232/PPI al puerto de comunicación de la unidad de programación. (En el presente ejemplo, conectar a COM 1.)
2. Una el conector RS-485 (identificado con “PPI”) del cable multimaestro RS-232/PPI al puerto 0 ó 1 del S7-200.
3. Vigile que los interruptores DIP del cable multimaestro RS-232/PPI estén configurados como muestra la figura A.23.

En los ejemplos del presente manual se utiliza el cable multimaestro RS-232/PPI. El cable multimaestro RS-232/PPI sustituye el cable PC/PPI que se empleaba anteriormente. El cable multimaestro USB/PPI también está disponible.

Iniciar STEP 7-Micro/WIN

Haga clic en el icono de STEP 7-Micro/WIN para abrir un nuevo proyecto La figura A.24 muestra un nuevo proyecto.

Aprecie la barra de navegación. Puede utilizar los iconos de la barra de navegación para abrir los elementos del proyecto de STEP 7-Micro/WIN.

En la barra de navegación, haga clic en el icono “Comunicación” para abrir el cuadro de diálogo correspondiente.

Utilice este cuadro de diálogo para configurar la comunicación de STEP 7-Micro/WIN.

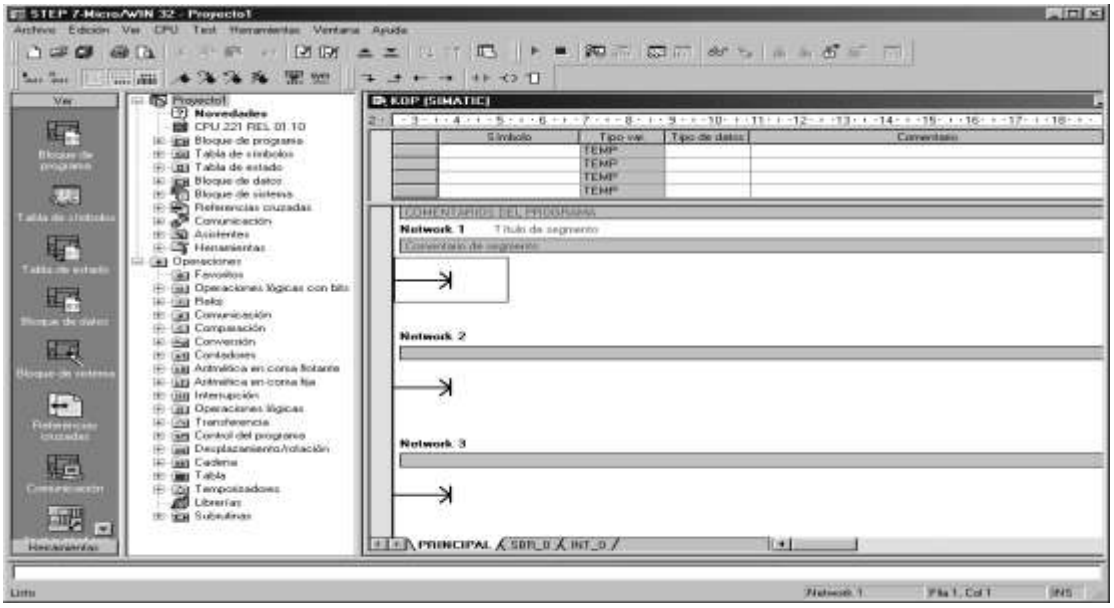


Fig. A.24. Nuevo proyecto de STEP 7-Micro/WIN.

Verificar los parámetros de comunicación de STEP 7-Micro/WIN

En el proyecto se utilizan los ajustes estándar de STEP 7-Micro/WIN y del cable multimaestro RS-232/PPI. Para verificar los ajustes:

Fig. A.25. Verificar los parámetros de comunicación.

1. Vigile que la dirección del cable PC/PPI esté ajustada a 0 en el cuadro de diálogo “Comunicación”.

2. Vigile que la interfaz del parámetro de red esté configurada para el cable PC/PPI (COM1).

3. Vigile que la velocidad de transferencia esté ajustada a 9,6 kbit/s.

Establecer la comunicación con el S7-200

Utilice el cuadro de diálogo “Comunicación” para establecer la comunicación con el S7-200:

1. En el cuadro de diálogo “Comunicación”, haga doble clic en el icono “Actualizar”. STEP 7-Micro/WIN buscará el S7-200 y visualizará un icono “CPU” correspondiente a la CPU S7-200 conectada.

2. Seleccione el S7-200 y haga clic en “Aceptar”. Si STEP 7-Micro/WIN no encuentra el S7-200, verifique los parámetros de comunicación y repita los pasos descritos arriba.

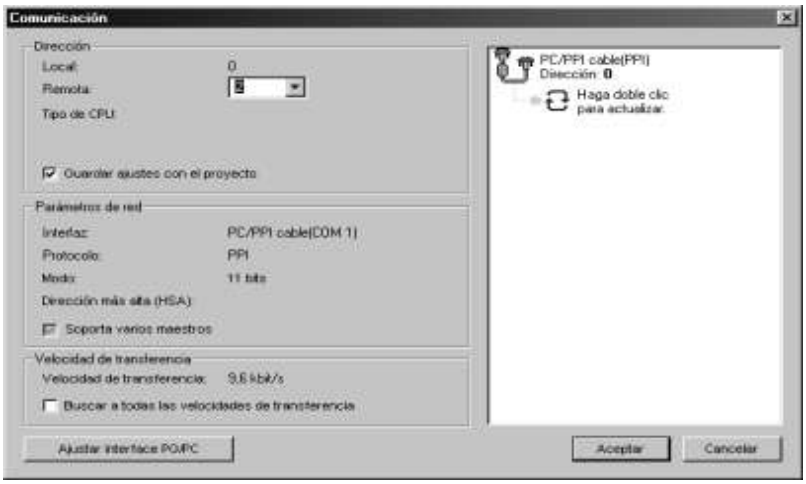


Fig. A.26. Establecer la comunicación con el S7-200.

Tras haber establecido la comunicación con el S7-200 podrá crear el programa de ejemplo y cargarlo.

A.6 CARGAR Y DEPURAR PROGRAMAS EN LA CPU

Todos los proyectos de STEP 7-Micro/WIN están asociados a un determinado tipo de CPU (CPU 221, CPU 222, CPU 224, CPU 224XP ó CPU 226XM). Si el tipo de proyecto no concuerda con la CPU conectada, STEP 7-Micro/WIN visualizará un mensaje de error, indicándole que debe tomar una determinada medida.

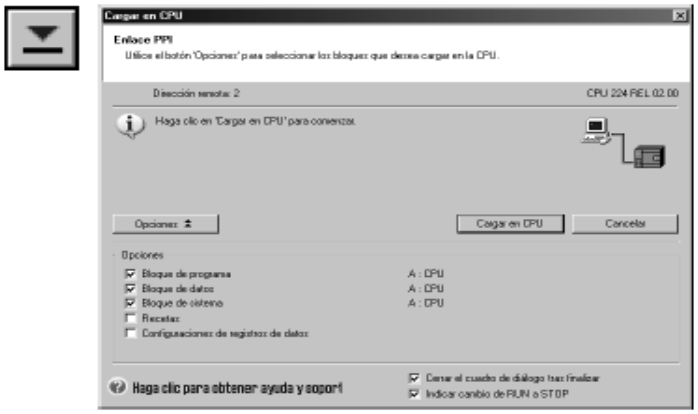


Fig. A.27. Cargar el programa de ejemplo.

2 En la barra de herramientas, haga clic en el botón "Cargar" o elija el comando **Archivo > Cargar** para cargar el programa en la CPU (v. fig. A.27).

3 Haga clic en "Aceptar" para cargar los elementos de programa en el S7-200.

Si el S7-200 está en modo RUN, aparecerá un mensaje indicando que debe cambiar el S7-200 a modo STOP. Haga clic en "Sí" para poner el S7-200 en modo STOP.

A.6.1 SELECCIONAR EL MODO DE OPERACIÓN DEL S7-200

El S7-200 tiene dos modos de operación, los cuales son: STOP y RUN. Los diodos luminosos (LEDs) ubicados en la parte frontal de la CPU indican el modo de operación actual. En modo STOP, el S7-200 no ejecuta el programa. Entonces es posible cargar un programa o configurar la CPU. En modo RUN, el S7-200 ejecuta el programa.

- ✓ El S7-200 incorpora un selector de modos que permite cambiar el modo de operación. El modo de operación se puede cambiar manualmente accionando el selector (ubicado debajo de la tapa de acceso frontal del S7-200). Si el selector se pone en STOP, se detendrá la ejecución del programa. Si se pone en RUN, se iniciará la ejecución del programa. Si se pone en TERM, no cambiará el modo de operación. Si se interrumpe la alimentación estando el selector en posición STOP o TERM, el S7-200 pasará a modo STOP cuando se le aplique tensión. Si se interrumpe la alimentación estando el selector en posición RUN, el S7-200 pasará a modo RUN cuando se le aplique tensión.

- ✓ STEP 7-Micro/WIN permite cambiar el modo de operación del S7-200 conectado. Para que el modo de operación se pueda cambiar mediante el software, el selector del S7-200 deberá estar en posición TERM o RUN. Elija para ello el comando de menú **CPU > STOP** o **CPU > RUN**, respectivamente (o haga clic en los botones correspondientes de la barra de herramientas).

- ✓ Para cambiar el S7-200 a modo STOP es posible introducir la correspondiente operación (STOP) en el programa. Ello permite detener la ejecución del programa en función de la lógica. Para más información sobre la operación STOP,

Poner el S7-200 en modo RUN

Para que STEP 7-Micro/WIN pueda poner el S7-200 en modo RUN, el selector de modo de la CPU deberá estar en posición TERM o RUN. El programa se ejecuta cuando el S7-200 cambia a modo RUN:

1. En la barra de herramientas, haga clic en el botón "RUN" o elija el comando de menú **CPU > RUN**.

2. Haga clic en "Aceptar" para cambiar el modo de operación del S7-200.

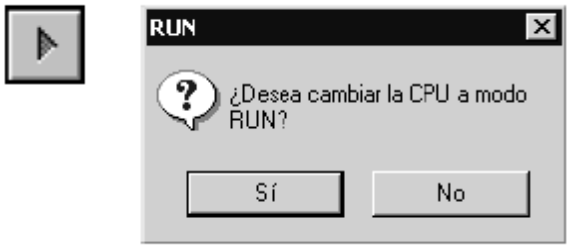


Fig. A.28. Poner el S7-200 en modo RUN.

Cuando el S7-200 cambia a modo RUN, el LED correspondiente al modo RUN se enciende. Para observar el programa puede seleccionar el comando de menú **Test > Estado del programa**. STEP 7-Micro/WIN visualizará los valores de las operaciones. Para detener la ejecución del programa, cambie el S7-200 a modo STOP haciendo clic en el botón "STOP" de la barra de herramientas, o bien eligiendo el comando de menú **CPU > STOP**.

A.6.2 UTILIZAR LA TABLA DE ESTADO PARA OBSERVAR EL PROGRAMA

La tabla de estado sirve para observar o modificar los valores de las variables del proceso a medida que el S7-200 ejecuta el programa. Es posible observar el estado de las entradas, salidas o variables del programa, visualizando para ello los valores actuales. La tabla de estado también permite forzar o modificar los valores de las variables del proceso.

Es posible crear varias tablas de estado para visualizar elementos de diferentes partes del programa. Para acceder a la tabla de estado, elija el comando de menú **Ver > Componente > Tabla de estado** o haga clic en el icono "Tabla de estado" en la barra de navegación.

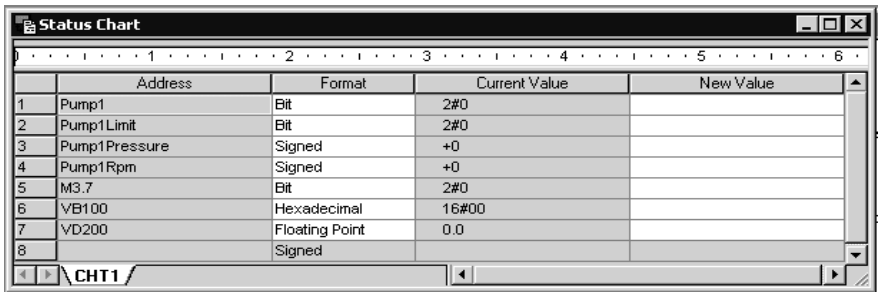


Fig. A.29. Tabla de estado.

Al crear una tabla de estado se deben introducir las direcciones de las variables del proceso que se desean observar. No es posible visualizar el estado de las constantes, ni de los acumuladores, ni tampoco de las variables locales. Los valores de los temporizadores y contadores se pueden visualizar en formato de bit o de palabra.

En formato de bit, se visualizará el estado del bit del temporizador o del contador. En formato de palabra, se visualizará el valor del temporizador o del contador.

Para crear una tabla de estado y observar las variables:

1. En el campo “Dirección”, introduzca la dirección de cada valor deseado.
2. En la columna “Formato”, seleccione el tipo de datos.
3. Para visualizar el estado de las variables del proceso en el S7-200, elija el comando de menú **Test > Tabla de estado**.
4. Para observar continuamente los valores o para efectuar una sola lectura del estado, haga clic en el botón correspondiente en la barra de herramientas. La tabla de estado también permite modificar o forzar los valores de las variables del proceso.

Para insertar filas adicionales en la tabla de estado, elija los comandos de menú **Edición > Insertar > Fila**.

Es posible crear varias tablas de estado para estructurar las variables en grupos lógicos, de manera que cada grupo se pueda visualizar por separado en una tabla de estado más pequeña.

A.7 JUEGO DE OPERACIONES DEL S7-200

- ✓ **OPERACIONES LÓGICAS CON BITS**

Operaciones lógicas con contactos:

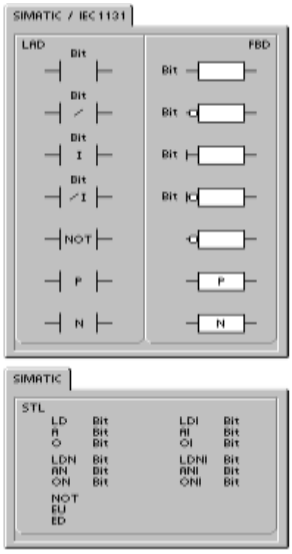


Fig. A.30. Operaciones lógicas con bits.

Contactos estándar

Las operaciones Contacto normalmente abierto (LD, A y O) y Contacto normalmente cerrado (LDN, AN y ON) leen el valor direccionado de la memoria (o bien de la imagen del proceso, si el tipo de datos es I o Q).

El Contacto normalmente abierto se cierra (ON) si el bit es igual a 1, en tanto que el Contacto normalmente cerrado se cierra (ON) si el bit es igual a 0. En FUP, la cantidad de entradas de los cuadros AND y OR se puede incrementar a 32 como máximo. En AWL, el Contacto normalmente abierto carga, o bien combina con Y u O el valor binario del bit de dirección en el nivel superior de la pila. El Contacto normalmente cerrado carga, o bien combina con Y u O el valor negado del bit de dirección en el nivel superior de la pila.

Contactos directos

Los contactos directos no dependen del ciclo del S7-200 para actualizarse, sino que se actualizan inmediatamente. Las operaciones del Contacto abierto directo (LDI, AI y OI) y del Contacto cerrado directo (LDNI, ANI y ONI) leen el valor de la entrada física cuando se ejecuta la operación, pero la imagen del proceso no se actualiza. El Contacto abierto directo se cierra (ON) si la entrada física (bit) es 1, en tanto que el Contacto cerrado directo se cierra (ON) si la entrada física (bit) es 0. El Contacto abierto directo carga, o bien combina con Y u O directamente el valor de la entrada física en el nivel superior de la pila. El Contacto cerrado directo carga, o bien combina con Y u O directamente el valor binario negado de la entrada física en el nivel superior de la pila.

NOT

La operación NOT cambia el estado de la entrada de circulación de corriente (es decir, modifica el valor del nivel superior de la pila de “0” a “1”, o bien de “1” a “0”).

Detectar flanco positivo y negativo

El contacto Detectar flanco positivo (EU) permite que la corriente circule durante un ciclo cada vez que se produce un cambio de “0” a “1” (de “off” a “on”). El contacto Detectar flanco negativo (ED) permite que la corriente circule durante un ciclo cada vez que se produce un cambio de “1” a “0” (de “on” a “off”). Cuando se detecta un cambio de señal de “0” a “1” en el primer valor de la pila, éste se pone a 1. En caso contrario, se pone a 0. Cuando se detecta un cambio de señal de “1” a “0” en el primer valor de la pila, éste se pone a 1. En caso contrario, se pone a 0.

Entradas/salidas	Tipos de datos	Operandos
Bit	BOOL	I, Q, V, M, SM, S, T, C, L, circulación de corriente
Bit (directo)	BOOL	I

Tabla A.8. Operandos válidos para las operaciones lógicas con bits de entrada

Operaciones lógicas con salidas:

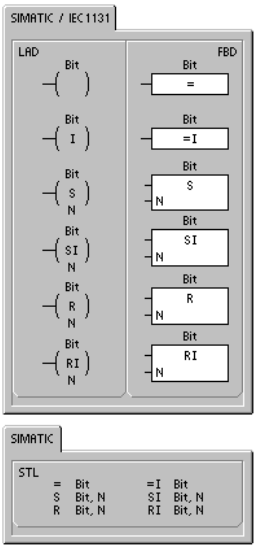


Fig. A.31. Operaciones lógicas con bits.

Asignar

La operación Asignar (=) escribe el nuevo valor del bit de salida en la imagen del proceso. Cuando se ejecuta la operación Asignar, el S7-200 activa o desactiva el bit de salida en la imagen del proceso. En KOP y FUP, el bit indicado se ajusta de forma equivalente a la circulación de la corriente. En AWL, el primer valor de la pila se copia en el bit indicado.

Asignar directamente

La operación Asignar directamente (=I) escribe el nuevo valor tanto en la salida física como en la correspondiente dirección de la imagen del proceso.

Cuando se ejecuta la operación Asignar directamente, la salida física (bit) se ajusta directamente de forma equivalente a la circulación de la corriente. En AWL, la operación

copia el primer valor de la pila directamente en la salida física indicada (bit). La "I" indica que la operación se ejecuta directamente. El nuevo valor se escribe entonces tanto en la salida física como en la correspondiente dirección de la imagen del proceso. En cambio, en las operaciones no directas, el nuevo valor se escribe sólo en la imagen del proceso.

Poner a 1 y Poner a 0

Las operaciones Poner a 1 (S) y Poner a 0 (R) activan (ponen a 1) o desactivan (ponen a 0) el número indicado de E/S (N) a partir de la dirección indicada (bit). Es posible activar o desactivar un número de entradas y salidas (E/S) comprendido entre 1 y 255.

Si la operación Poner a 0 indica un bit de temporización (T) o un bit de contaje (C), se desactivará el bit de temporización o de contaje y se borrará el valor actual del temporizador o del contador, respectivamente.
Condiciones de error que ponen ENO a 0:

- 0006 (direccionamiento indirecto)
- 0091 (operando fuera de rango)

Poner a 1 directamente y Poner a 0 directamente

Las operaciones Poner a 1 directamente (SI) y Poner a 0 directamente (RI) activan (ponen a 1) o desactivan (ponen a 0) directamente el número indicado de E/S (N) a partir de la dirección indicada (bit). Es posible activar o desactivar directamente un número de entradas y salidas (E/S) comprendido entre 1 y 128.

La "I" indica que la operación se ejecuta directamente. El nuevo valor se escribe tanto en la salida física como en la correspondiente dirección de la imagen del proceso. En cambio, en las operaciones no directas, el nuevo valor se escribe sólo en la imagen del proceso.
Condiciones de error que ponen ENO a 0:

- 0006 (direccionamiento indirecto)
- 0091 (operando fuera de rango)

Entradas/salidas	Tipos de datos	Operandos
Bit	BOOL	I, Q, V, M, SM, S, T, C, L
Bit (directo)	BOOL	Q
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, constante

Tabla A.8. Operandos válidos para las operaciones lógicas con bits de salida

- ✓ **OPERACIONES DE COMPARACIÓN**

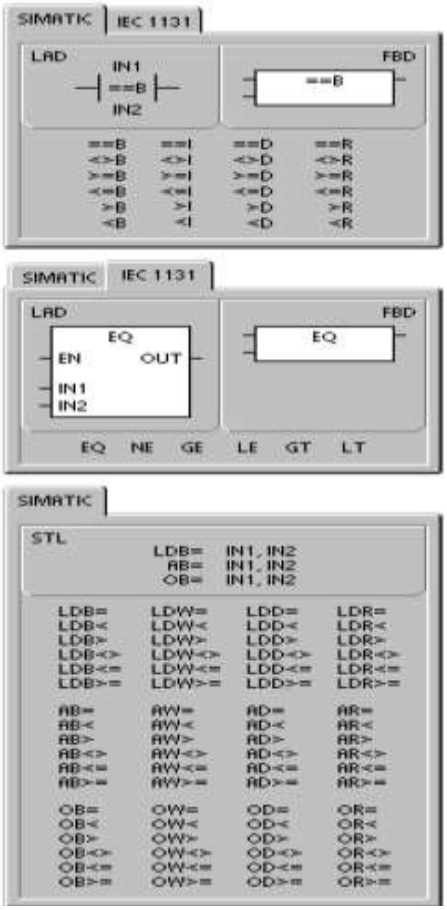


Fig. A.32. Operaciones de comparación

Las operaciones de comparación se utilizan para comparar dos valores:

IN1 = IN2 IN1 >= IN2 IN1 <= IN2
IN1 > IN2 IN1 < IN2 IN1 <> IN2

- Las comparaciones de bytes no llevan signo.
- Las comparaciones de enteros llevan signo.
- Las comparaciones de palabras dobles llevan signo.
- Las comparaciones de números reales llevan signo.

En KOP y FUP: Si la comparación es verdadera, la operación de comparación activa el contacto (KOP) o la salida (FUP).

En AWL: Si la comparación es verdadera, la operación de comparación carga un 1 en el nivel superior de la pila, o bien lo combina con Y u O.

Si se utilizan las operaciones de comparación IEC, es posible utilizar diversos tipos de datos para las entradas. No obstante, el tipo de datos de los dos valores de entrada deberá ser idéntico.

Nota
Las siguientes condiciones son errores fatales que detendrán inmediatamente la ejecución del programa en el S7-200:

- Detección de una dirección indirecta no válida (en todas las operaciones de comparación)
- Detección de un número real no válido (por ejemplo, NAN) (en la operación Comparar reales)

Para evitar estas condiciones de error, inicialice correctamente los punteros y los valores que contengan números reales antes de ejecutar las operaciones de comparación que utilicen estos valores. Las operaciones de comparación se ejecutan sin tener en cuenta el estado de señal.

- ✓ **OPERACIONES DE CONVERSIÓN**

Operaciones de conversión normalizadas

Conversiones numéricas
Las operaciones Convertir byte en entero (BTI), Convertir entero en byte (ITB), Convertir entero en entero doble (ITD), Convertir entero doble en entero (DTI), Convertir entero doble en real (DTR), Convertir BCD en entero (BCDI) y Convertir entero en BCD (IBCD) convierten un valor de entrada IN en el formato indicado y almacenan el valor de salida en la dirección especificada por OUT. Por ejemplo, es posible convertir un valor de entero doble en un número real. También es posible convertir un entero en un número BCD y viceversa.

Redondear a entero doble y Truncar
La operación Redondear (ROUND) convierte un valor real (IN) en un valor de entero doble y deposita el resultado redondeado en la variable indicada por OUT.
La operación Truncar (TRUNC) convierte un número real (IN) en un entero doble y carga la parte del número entero del resultado en la variable indicada por OUT.

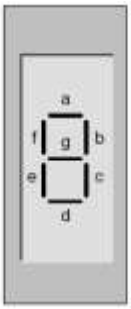
Segmento
La operación Segmento (SEG) sirve para generar una configuración binaria (OUT) que ilumina los segmentos de un indicador de siete segmentos.

Con objeto de iluminar los segmentos de un indicador de siete segmentos, la operación Segmento (SEG) convierte el carácter (byte) indicado por IN para generar una configuración binaria (byte) en la dirección indicada por OUT.

Los segmentos iluminados representan el carácter depositado en el dígito menos significativo del byte de entrada. La figura A.33 muestra la codificación del indicador de siete segmentos utilizado por la operación Segmento.
Condiciones de error que ponen ENO a 0:

- 0006 (direccionamiento indirecto)

(IN) LSD	Indicador	(OUT) -gfe dcba
0	0	0011 1111
1	1	0000 0110
2	2	0101 1011
3	3	0100 1111
4	4	0110 0110
5	5	0110 1101
6	6	0111 1101
7	7	0000 0111



(IN) LSD	Indicador	(OUT) -gfe dcba
8	8	0111 1111
9	9	0110 0111
A	A	0111 0111
B	b	0111 1100
C	C	0011 1001
D	d	0101 1110
E	E	0111 1001
F	F	0111 0001

Fig.A.33. Codificación de un indicador de siete segmentos

Funcionamiento de las operaciones Convertir BCD en entero y Convertir entero en BCD
La operación Convertir BCD en entero (BCDI) convierte el valor decimal codificado en binario IN en un valor de entero y carga el resultado en la variable indicada por OUT. El rango válido de IN está comprendido entre 0 y 9999 BCD. La operación Convertir entero en BCD (IBCD) convierte el valor entero de entrada IN en un valor BCD y carga el resultado en la variable indicada por OUT. El rango válido de IN está comprendido entre 0 y 9999 enteros.

Condiciones de error que ponen ENO a 0:

- SM1.6 (BCD no válido)
- 0006 (direccionamiento indirecto)

Marcas especiales afectadas:

- SM1.6 (BCD no válido)

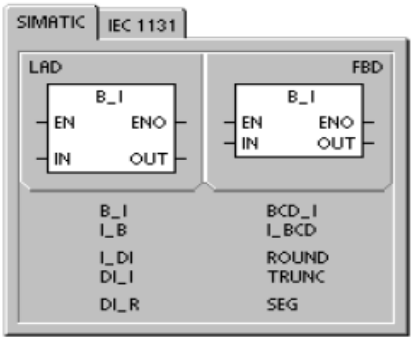


Fig. A.34. Función convertir BCD en entero y entero en BCD

- ✓ **OPERACIONES DE CONTAJE**

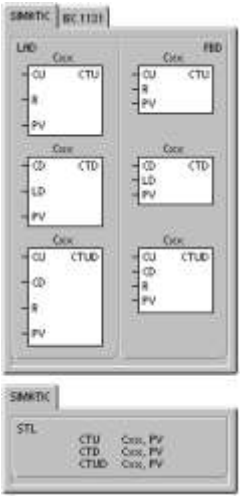


Fig.A.35. Operaciones de contaje

Incrementar contador

La operación Incrementar contador (CTU) empieza a contar adelante a partir del valor actual cuando se produce un flanco positivo en la entrada de contaje adelante (CU). Si el valor actual (Cxx) es mayor o igual al valor de preselección PV, se activa el bit de contaje Cxx. El contador se inicializa cuando se activa la entrada de desactivación (R) o al ejecutarse la operación Poner a 0. El contador se detiene cuando el valor de contaje alcance el valor límite superior (32.767).

Funcionamiento en AWL:

- Entrada de desactivación: valor superior de la pila
- Entrada de contaje adelante: valor cargado en el segundo nivel de la pila

Decremento contador

La operación Decremento contador (CTD) empieza a contar atrás a partir del valor actual cuando se produce un flanco negativo en la entrada de contaje atrás (CD). Si el valor actual Cxx es igual a 0, se activa el bit de contaje Cxx. El contador desactiva el bit de contaje Cxx y carga el valor actual con el valor de preselección (PV) cuando se activa la entrada de carga LD. El contador se detiene al alcanzar el valor cero y el bit de contaje Cxx se activa.

Funcionamiento en AWL:

- Entrada de carga: valor superior de la pila
- Entrada de contaje atrás: valor cargado en el segundo nivel de la pila

Incrementar/decrementar contador.

La operación Incrementar/decrementar contador (CTUD) empieza a contar adelante cuando se produce un flanco positivo en la entrada de contaje adelante (CU), y empieza a contar atrás cuando se produce un flanco positivo en la entrada de contaje atrás (CD). El valor actual Cxx del contador conserva el contaje actual. El valor de preselección PV se compara con el valor actual cada vez que se ejecuta la operación de contaje.

Cuando se alcanza el valor máximo (32.767), el siguiente flanco positivo en la entrada de contaje adelante invertirá el contaje hasta alcanzar el valor mínimo (-32.768). Igualmente, cuando se alcanza el valor mínimo (-32.768), el siguiente flanco positivo en la entrada de contaje atrás invertirá el contaje hasta alcanzar el valor máximo (32.767). Si el valor actual (Cxx) es mayor o igual al valor de preselección PV, se activa el bit de contaje Cxx. En caso contrario, se desactiva el bit. El contador se inicializa cuando se activa la entrada de desactivación (R) o al ejecutarse la operación Poner a 0. El contador adelante/atrás se detiene al alcanzar el valor de preselección (PV).

Funcionamiento en AWL:

- Entrada de desactivación: valor superior de la pila
- Entrada de contaje atrás: valor cargado en el segundo nivel de la pila

- Entrada de contaje adelante: valor cargado en el tercer nivel de la pila

Tipos de datos	Funcionamiento	Bit de contaje	Alimentación/primer ciclo
CTU	CU incrementa el valor actual. El valor actual se sigue incrementando hasta alcanzar 32.767.	El bit de contaje se activa si: valor actual >= valor de preselección	El bit de contaje está desactivado. El valor actual se puede conservar.[1]
CTUD	CU incrementa el valor actual. CD decrementa el valor actual. El valor actual se sigue incrementando o decrementando hasta que se inicialice el contador.	El bit de contaje se activa si: valor actual >= valor de preselección	El bit de contaje está desactivado. El valor actual se puede conservar.[1]
CTD	CD decrementa el valor actual hasta que éste alcance 0.	El bit de contaje se activa si: valor actual >= 0	El bit de contaje está desactivado. El valor actual se puede conservar.[1]

[1] Es posible ajustar que se memorice el valor actual del contador. Para más información sobre el respaldo de la memoria de la CPU S7-200,

Tabla A.9. Funcionamiento de las operaciones de contaje

- ✓ **OPERACIONES ARITMETICAS**

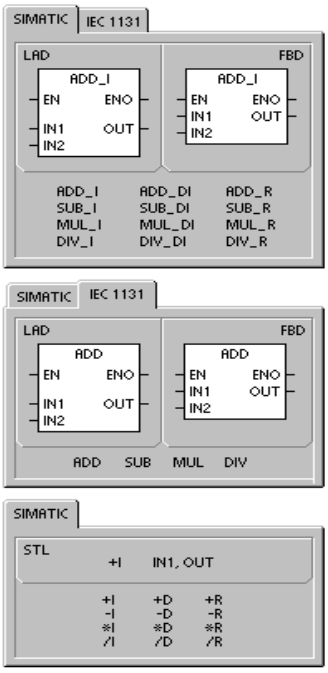


Fig.A.36. Operaciones aritméticas

Operaciones de sumar, restar, multiplicar y dividir

Sumar Restar

IN1 + IN2 = OUT *IN1 - IN2 = OUT* KOP y FUP
IN1 + OUT = OUT *OUT - IN1 = OUT* AWL

Las operaciones Sumar enteros (+I) y Restar enteros (-I) suman/restan dos enteros de 16 bits, arrojando un resultado de 16 bits. Las operaciones Sumar enteros dobles (+D) y Restar enteros dobles (-D) suman/restan dos enteros de 32 bits, arrojando un resultado de 32 bits. Las operaciones Sumar reales (+R) y Restar reales (-R) suman/restan dos números reales de 32 bits, dando como resultado un número real de 32 bits.

Multiplicar Dividir

*IN1 * IN2 = OUT*	*IN1 / IN2 = OUT*	KOP y FUP
*IN1 * OUT = OUT*	*OUT / IN1 = OUT*	AWL

Las operaciones Multiplicar enteros (*I) y Dividir enteros (/I) multiplican o dividen dos enteros de 16 bits, respectivamente, arrojando un resultado de 16 bits. (En la división no se conserva un resto.) Las operaciones Multiplicar enteros dobles (*D) y Dividir enteros dobles (/D) multiplican o dividen dos enteros de 32 bits, respectivamente, arrojando un resultado de 32 bits. (En la división no se conserva un resto.) Las operaciones Multiplicar reales (*R) y Dividir reales (/R) multiplican o dividen dos números reales de 32 bits, respectivamente, dando como resultado un número real de 32 bits.

Marcas especiales y ENO

SM1.1 indica errores de desbordamiento y valores no válidos. Si se activa SM1.1, el estado de SM1.0 y de SM1.2 no será válido y no se alterarán los operandos de entrada originales. Si SM1.1 y SM1.3 no se activan, la operación aritmética habrá finalizado con un resultado válido, y tanto SM1.0 como SM1.2 contendrán un estado válido. Si se activa SM1.3 durante una operación de división, permanecerán inalterados los demás bits aritméticos de estado.

Condiciones de error que ponen ENO a 0:

- SM1.1 (desbordamiento)
- SM1.3 (división por cero)
- 0006 (direccionamiento indirecto)

Marcas especiales afectadas

- SM1.0 (cero)
- SM1.1 (desbordamiento, valor no válido generado durante la operación o parámetro de entrada no válido)
- SM1.2 (negativo)
- SM1.3 (división por cero)

Entradas/salidas	Tipos de datos	Operandos
IN1, IN2	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, constante
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, constante
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, constante
OUT	INT	IW, QW, VW, MW, SMW, SW, LW, T, C, AC, *VD, *AC, *LD
	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Tabla A.10. Operandos válidos para las operaciones de sumar, restar, multiplicar y dividir

Incrementar y decrementar

Incrementar
IN + 1 = OUT KOP y FUP
OUT + 1 = OUT AWL
Decrementar
IN - 1 = OUT KOP y FUP
OUT - 1 = OUT AWL

Las operaciones Incrementar y Decrementar suman/restan 1 al valor de la entrada IN y depositan el resultado en OUT. Las operaciones Incrementar byte (INCB) y Decrementar byte (DECB) no llevan signo. Las operaciones Incrementar palabra (INCW) y Decrementar palabra (DECW) llevan signo. Las operaciones Incrementar palabra doble (INCD) y Decrementar palabra doble (DECD) llevan signo.

Condiciones de error que ponen ENO a 0:
- SM1.1 (desbordamiento)
- 0006 (direccionamiento indirecto)

Marcas especiales afectadas:
- SM1.0 (cero)
- SM1.1 (desbordamiento)
- SM1.2 (negativo) para operaciones con palabras y palabras dobles.

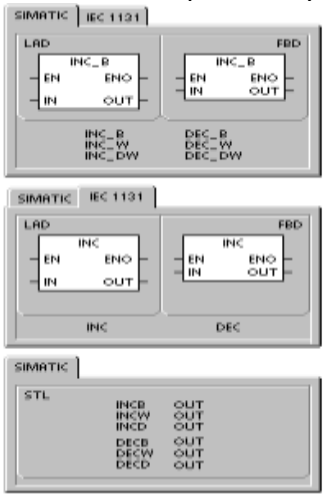


Fig.A.37. Incrementar y Decrementar

- ✓ **OPERACIONES LOGICAS**

  Operaciones de combinación con Y, O y O-exclusiva Combinación Y con bytes, con palabras y con palabras dobles

Las operaciones Combinación Y con bytes (ANDB), Combinación Y con palabras (ANDW) y Combinación Y con palabras dobles (ANDD) combina los bits correspondientes de dos valores de entrada IN1 e IN2 mediante Y, y cargan el resultado en una dirección de la memoria OUT.

Combinación O con bytes, con palabras y con palabras dobles

Las operaciones Combinación O con bytes (ORB), Combinación O con palabras (ORW) y Combinación O con palabras dobles (ORD) combinan los bits correspondientes de dos valores de entrada IN1 e IN2 mediante O y cargan el resultado en una dirección de la memoria OUT.

Combinación O-exclusiva con bytes, con palabras o con palabras dobles

Las operaciones Combinación O-exclusiva con bytes (XROB), Combinación O-exclusiva con palabras (XORW) y Combinación O-exclusiva con palabras dobles (XORD) combinan los bits correspondientes de dos valores de entrada (IN1 e IN2) mediante O-exclusiva y cargan el resultado en una dirección de la memoria OUT.

Marcas especiales y ENO

En todas las operaciones descritas en esta página, las condiciones siguientes afectan a las marcas especiales y a ENO:

Condiciones de error que ponen ENO a 0:

- 0006 (direccionamiento indirecto)

Marcas especiales afectadas:

- SM1.0 (cero)

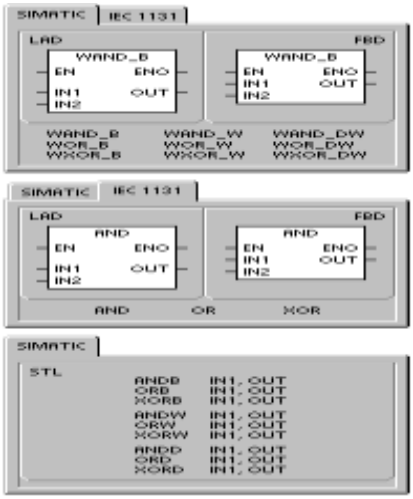


Fig.A.38. Operaciones Lógicas

- ✓ **OPERACIONES DE TRANSFERENCIA**

Transferir bytes, palabras, palabras dobles y números reales

Las operaciones Transferir byte (MOVB), Transferir palabra (MOVW), Transferir palabra doble (MOVD) y Transferir real (MOVR) transfieren un valor de una dirección (IN) a una nueva dirección (OUT) sin modificar el valor original. Si desea crear un puntero, utilice la operación Transferir palabra doble.

En el caso de la operación IEC Transferir (MOVE), los tipos de los datos de entrada y salida pueden ser diferentes, pero su tamaño debe ser igual.

Condiciones de error que ponen ENO a 0:

- 0006 (direccionamiento indirecto)

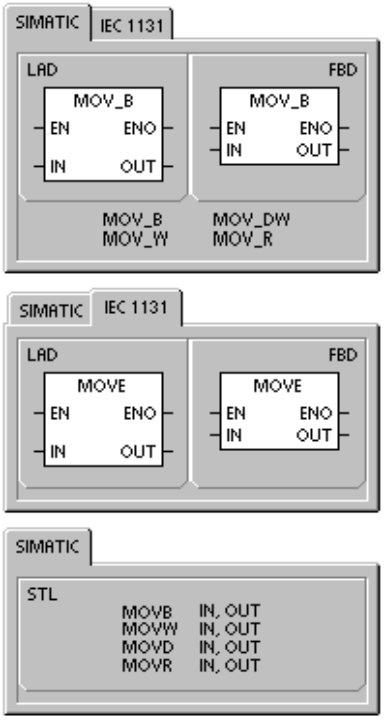


Fig.A.39. Operaciones de Transferencia

Entradas/salidas	Tipos de datos	Operandos
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, constante
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, constante
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, HC, &VB, &IB, &QB, &MB, &SB, &T, &C, &SMB, &AIW, &AQW, AC, *VD, *LD, *AC, constante,
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, constante
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC
	DWORD, DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Tabla A.11. Operandos válidos para las operaciones de transferencia

- ✓ **OPERACIONES DE DESPLAZAMIENTO Y ROTACION**

Desplazar a la derecha y Desplazar a la izquierda

Las operaciones de desplazamiento desplazan el valor de entrada IN a la derecha o a la izquierda tantas posiciones como indique el valor de desplazamiento N y cargan el resultado en la salida OUT.

Las operaciones de desplazamiento se rellenan con ceros cada vez que se desplaza un bit. Si el valor de desplazamiento (N) es mayor o igual al valor máximo permitido (8 en las operaciones con bytes, 16 en las operaciones con palabras y 32 en las operaciones con palabras dobles), se desplazará el valor máximo permitido para la operación en cuestión. Si el valor de desplazamiento es mayor que 0, la marca de desbordamiento (SM1.1) adoptará el valor del último bit desplazado hacia afuera. La marca cero (SM1.0) se activará si el resultado de la operación de desplazamiento es cero.

Las operaciones de desplazamiento de bytes no llevan signo. En el caso de las operaciones con palabras y con palabras dobles, el bit de signo se desplaza cuando se utilizan tipos de datos con signo.

Condiciones de error que ponen ENO a 0:

- 0006 (direccionamiento indirecto)

Marcas especiales afectadas:

- SM1.0 (cero)
- SM1.1 (desbordamiento)

Rotar a la derecha y Rotar a la izquierda

Las operaciones de rotación rotan el valor de entrada (IN) a la derecha o a la izquierda tantas posiciones como indique el valor de desplazamiento (N) y cargan el resultado en la dirección de la memoria (OUT). La rotación es circular. Si el valor de desplazamiento es mayor o igual al valor máximo permitido (8 en las operaciones con bytes, 16 en las operaciones con palabras y 32 en las operaciones con palabras dobles), el S7-200 ejecutará una operación módulo en el valor de desplazamiento para obtener un valor válido antes de ejecutarse la rotación. De ello resulta un valor de desplazamiento de 0 a 7 en las operaciones con bytes, de 0 a 15 en las operaciones con palabras y de 0 a 31 en las operaciones con palabras dobles.

Si el valor de desplazamiento es igual a 0, no se rotará el valor. Si se ejecuta la rotación, el valor del último bit rotado se copiará en la marca de desbordamiento (SM1.1).

Si el valor de desplazamiento no es un entero múltiplo de 8 (en las operaciones con bytes), de 16 (en las operaciones con palabras) o de 32 (en las operaciones con palabras dobles), el último bit rotado se copiará en la marca de desbordamiento (SM1.1). La marca cero (SM1.0) se activará si el valor a rotar es igual a cero.

Las operaciones de desplazamiento de bytes no llevan signo. En el caso de las operaciones con palabras y con palabras dobles, el bit de signo se desplaza cuando se utilizan tipos de datos con signo.

Condiciones de error que ponen ENO a 0:

- 0006 (direccionamiento indirecto)

Marcas especiales afectadas:

- SM1.0 (cero)
- SM1.1 (desbordamiento)

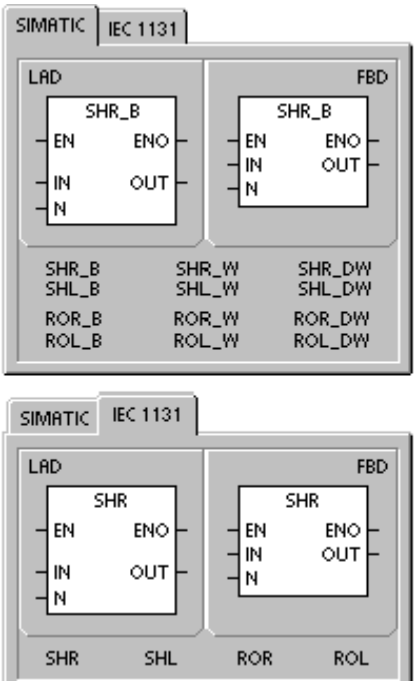


Fig.A.40. Operaciones de desplazamiento y rotación

Entradas/salidas	Tipos de datos	Operandos
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, constante
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, constante
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, constante
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, constante

Tabla A.12. Operandos válidos para las operaciones de desplazamiento y rotación

- ✓ **OPERACIONES DE TEMPORIZACIÓN**

Temporizador como retardo a la conexión con memoria

Las operaciones Temporizador como retardo a la conexión (TON) y Temporizador como retardo a la conexión con memoria (TONR) cuentan el tiempo al estar activada (ON) la entrada de habilitación. El número de temporizador (Txx) determina la resolución del mismo. Ésta se visualiza entonces en el cuadro de la operación.

Temporizador como retardo a la desconexión

El Temporizador como retardo a la desconexión (TOF) se utiliza para retardar la puesta a "0" (OFF) de una salida durante un período determinado tras haberse desactivado (OFF) una entrada. El número del temporizador (Txx) determina la resolución del mismo.

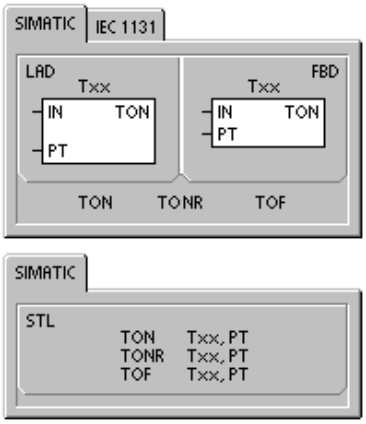


Fig.A.41. Operaciones de temporización

Entradas/salidas	Tipos de datos	Operandos
Txx	WORD	Constante (T0 a T255)
IN	BOOL	I, Q, V, M, SM, S, T, C, L, circulación de corriente
PT	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, constante

Tabla A.13. Operandos válidos para las operaciones de temporización.

No se pueden utilizar números iguales (Txx) para un temporizador como retardo a la conexión (TON) y un temporizador como retardo a la desconexión (TOF). Por ejemplo, no puede haber tanto un TON T32 como un TOF T32.

Como muestra la tabla A.14, los tres tipos de temporizadores ejecutan diferentes tareas de temporización:

- Los temporizadores como retardo a la conexión se utilizan para temporizar un solo intervalo.
- Los temporizadores como retardo a la conexión con memoria se utilizan para acumular varios intervalos temporizados.
- Los temporizadores con retardo a la desconexión se utilizan para ampliar el tiempo después de un cambio a OFF, por ejemplo, para enfriar un motor tras haber sido desconectado.

Tipos de datos	Actual >= Preselección	Estado de la entrada de habilitación (IN)	Alimentación/primer ciclo
TON	Bit de temporización ON El valor actual continúa contando hasta 32.767.	ON: El valor actual cuenta el tiempo. OFF: Bit de temporización OFF. Valor actual = 0	Bit de temporización OFF Valor actual = 0
TONR	Bit de temporización ON El valor actual continúa contando hasta 32.767.	ON: El valor actual cuenta el tiempo. OFF: El bit de temporización y el valor actual conservan el último estado.	Bit de temporización OFF El valor actual se puede conservar[1]
TOF	Bit de temporización OFF. Valor actual = valor de preselección, se detiene el contaje.	ON: Bit de temporización ON. Valor actual = 0 OFF: El temporizador cuenta tras un cambio de ON a OFF.	Bit de temporización OFF Valor actual = 0

[1] El valor actual del temporizador como retardo a la conexión con memoria se selecciona para que quede memorizado cuando se interrumpa la alimentación. Para más información sobre el respaldo de la memoria de la CPU S7-200

Tabla A.14. Funcionamiento de las operaciones de temporización.

A.7.1 TIEMPO DE EJECUCION DE LAS OPERACIONES AWL.

Los tiempos de ejecución de las operaciones son muy importantes en las aplicaciones con tiempos críticos. Estos tiempos figuran en la tabla A.16.

Impacto de la circulación de corriente

La tabla muestra el tiempo necesario para ejecutar la lógica o función de la operación cuando se aplica corriente en la operación (es decir, cuando el nivel superior de la pila es = 1 u ON). Si no hay circulación de corriente, el tiempo de ejecución de la operación será 1 µs.

Impacto del direccionamiento indirecto

La tabla además muestra el tiempo necesario para ejecutar la lógica o función de la operación si los operandos y constantes se direccionan de forma indirecta.

Cuando las operaciones utilizan operandos direccionados de forma indirecta, el tiempo de ejecución de la operación se incrementará en 14 µs por cada uno de esos operandos.

Impacto del acceso a ciertas áreas de memoria

El acceso a ciertas áreas de memoria, tales como AI (entradas analógicas), AQ (salidas analógicas), L (memoria local) y AC (acumuladores), prolonga también el tiempo de ejecución.

La tabla A.15 muestra el tiempo adicional que se debe sumar al tiempo de ejecución de la operación cuando esas áreas de memoria se indiquen en un operando.

Área de memoria	Tiempo de ejecución adicional
Entrada analógica integrada (AI) Filtración inhibida Filtración habilitada	 9,4 µs 8,4 µs
Entrada analógica de ampliación (AI) Filtración inhibida Filtración habilitada	 134 µs 8,4µs
Salida analógica integrada (AQ)	µs
Salida analógica de ampliación (AQ)	48 µs
Memoria local (L)	2,8 µs
Acumuladores (AC)	2,8 µs

Tabla A.15. Tiempo adicional para accesar a ciertas áreas de memoria.

Los tiempos de ejecución se muestra en la tabla A.16, mostrada a continuación:

Operación			µs
=	Utilizando:	I SM, T, C, V, S, Q, M L	0,24 1,3 10,5
+D			29
-D			29
* D			47
/ D			250
+I			25
I			25
* I			37
/ I			64
=I	Utilizando:	Salidas integradas Salidas en un módulo de ampliación	16 24
+R			Tip. 71 Máx. 99
R			Tip. 72 Máx. 100
*R			Tip. 56 Máx. 166
/R			Tip. 177 Máx. 230
A	Utilizando:	I SM, T, C, V, S, Q, M L	0,22 0,72 6,1
AB <=, =, >=, >, <, <>			18
AD <=, =, >=, >, <, <>			27
AENO			0,4
=I	Utilizando:	Entradas integradas Entradas en un módulo de ampliación	15 21
ALD			0,22
AN	Utilizando:	I SM, T, C, V, S, Q, M L	0,22 0,72 6,1
ANDB			19
ANDD			30
ANDW			25
ANI	Utilizando:	Entradas integradas Entradas en un módulo de ampliación	15 21
AR <=, =, >=, >, <, <>			29
AS=, <>Total = tiempo básico + (LM + N) Tiempo básico Multiplicador de longitud (ML) N es el número de caracteres comparados			33 6,3
ATCH			12
ATH	Total = tiempo básico + (longitud + ML) Tiempo básico (longitud constante) Tiempo básico (longitud variable) Multiplicador de longitud (ML)		23 31 10,2
ATT			36
AW <=, =, >=, >, <, <>			23
BCDI			35

Operación			µs
BITIM			16
BIR	Utilizando:	Entradas integradas Entradas en un módulo de ampliación	23 30
BIW	Utilizando:	Salidas integradas Salidas en un módulo de ampliación	24 32
BMB	Total = tiempo básico + (long. + ML) Tiempo básico (longitud constante) Tiempo básico (longitud variable) Multiplicador de longitud (ML)		 10 28 5,7
BMD	Total = tiempo básico + (long. + ML) Tiempo básico (longitud constante) Tiempo básico (longitud variable) Multiplicador de longitud (ML)		 11 29 10,6
BMW	Total = tiempo básico + (longitud + ML) Tiempo básico (longitud constante) Tiempo básico (longitud variable) Multiplicador de longitud (ML)		 10 28 8,6
BTI			16
CALL	Sin utilizar parámetros: Utilizando parámetros: Total = tiempo básico + Σ (tiempo de operandos) Tiempo básico Tiempo de operandos bit (entrada, salida) byte (entrada, salida) word (entrada, salida) Dword (entrada, salida) Nota: Los operandos de salida se procesan durante el retorno desde la subrutina.		9 14 10, 11 6, 7 10, 9 12, 10
CEVNT			24
CFND	Tiempo máx. = Tiempo básico + N1 + ((ML1 + N2) + ML2) Tiempo básico Multiplicador de longitud 1 (ML1) Multiplicador de longitud 2 (ML2) N1 es longitud de la cadena origen N2 es longitud del juego caracteres		 35 8,6 9,5
CITIM			23
COS			Tip. 900 Máx.1070
CRET	Con circulación de corriente Sin circulación de corriente		16 0,8
CRETI	Sin circulación de corriente		0,2
CSCRE			3,1
CTD	En un flanco de la entrada de contaje De lo contrario		27 19
CTU	En un flanco de la entrada de contaje De lo contrario		31 19
CTUD	En un flanco de la entrada de contaje De lo contrario		37 24
DECB			16
DECD			22
DECO			19
DECW			20
DISI			9
DIV			67

Operación		µs
DLED		14
DTA		302
DTI		21
DTCH		12
DTR		Típ. 35 Máx. 40
DTS		305
ED		8
ENCO		Máx. 24
END	Sin circulación de corriente	0,2
ENI		11
EU		8
EXP		Típ. 720 Máx. 860
FIFO	Total = tiempo básico + (long. * ML) Tiempo básico Multiplicador de longitud (ML)	 30 7
FILL	Total = tiempo básico + (longi. * ML) Tiempo básico (longitud constante) Tiempo básico (longitud variable) Multiplicador de longitud (ML)	 15 29 3,2
FND <, =, >, <>	Total = tiempo básico+(longitud * ML) Tiempo básico Multiplicador de longitud (ML)	 39 6,5
FOR	tiempo básico + (número de bucles * ML) Tiempo básico Multiplicador de bucles (ML)	 35 28
GPA		16
HDEF		18
HSC		30
HTA	Total = tiempo básico + (long. * ML) Tiempo básico (longitud constante) Tiempo básico (longitud variable) Multiplicador de longitud (ML)	 20 28 5,2
IBCD		52
INCB		15
INCD		22
INCW		20
INT	Tiempo típico con 1 interrupción	24
INVB		16
INVD		22
INVW		20
ITA		136
ITB		17
ITD		20
ITS		139
JMP		1,8
LBL		0,22

Operación			µs
LD	Utilizando:	I SM, T, C, V, S, Q, M L	0,22 0,8 6
LDB <=, =, >=, >, <, <>			18
LDD <=, =, >=, >, <, <>			27
LD	Utilizando:	Entradas integradas Entradas en un módulo de ampliación	15 21
LDN	Utilizando:	I SM, T, C, V, S, Q, M L	0,3 0,9 6,1
LDNI	Utilizando:	Entradas integradas Entradas en un módulo de ampliación	15 21
LDR<=, =, >=, >, <, <>			29
LDS			0,22
LDS=, <>Total = tiempo básico + (LM * N) Tiempo básico Multiplicador de longitud (ML) N es el número de caracteres comparados			 33 6,3
LDW <=, =, >=, >, <, <>			24
LIFO			37
LN			Típ. 680 Máx. 820
LPP			0,22
LPS			0,24
LRD			0,22
LSCR			7,3
MOVB			15
MOVD			20
MOVR			20
MOVW			18
MUL			37
NETR			99
NETW	Total = tiempo básico + (ML * N) Tiempo básico Multiplicador de longitud (ML) N es el número de bytes a enviar		 95 4
NEXT			0
NOP			0,22
NOT			0,22
O	Utilizando:	I SM, T, C, V, S, Q, M L	0,22 0,72 6,4
OB <=, =, >=, >, <, <>			18
OD <=, =, >=, >, <, <>			26
OI	Utilizando:	Entradas integradas Entradas en un módulo de ampliación	15 21
OLD			0,22
ON	Utilizando:	I SM, T, C, V, S, Q, M L	0,22 0,72 6,4

Operación			µs
ONI	Utilizando:	Entradas integradas	15
		Entradas en un módulo de ampliación	21
OR<=, =, >=, >, <, <>			29
ORB			19
ORD			29
ORW			25
OS=, < >	Total = tiempo básico + (LM * N)		
	Tiempo básico		33
	Multiplicador de longitud (ML)		6,3
	N es el número de caracteres comparados		
OW <=, =, >=, >, <, <>			24
PID	Típico		400
	Cambio de manual a automático		Máx. 800
	Recálculo coeficiente		Máx. 770
	Autosintonización		Máx. 650
PLS:	Utilizando:	PWM	31
		PTO monosegmento	36
		PTO multisegmento	50
R	Longitud=1 indicada como constante		
	Tiempo básico para contadores (C)		9,3
	Tiempo básico para temporizad. (T)		16
	Tiempo básico para todos los demás		2,9
	De lo contrario: Total = tiempo básico + (longitud * ML)		
	Tiempo básico para contadores (C)		8,6
	Tiempo básico para temporizad. (T)		8,3
	Tiempo básico para todos los demás		14
	Multiplicador de longitud (ML) para el operando C		5,1
	Multiplicador de longitud (ML) para el operando T		9,9
	Multiplicador de longitud (ML) para todos los demás		0,5
	Si la longitud se ha guardado como variable, sumar al tiempo básico		17
RCV			51
RET			16
RI	Total = tiempo básico + (long. * ML)		
	Tiempo básico		8,9
	Multiplicador de longitud (ML) utilizando salidas integradas		13
			21
	Multiplicador de longitud (ML) utilizando salidas de ampliación		
	Si la longitud se ha guardado como variable, sumar al tiempo básico		17
RLB	Total = tiempo básico + (ML * N)		
	Tiempo básico		23
	Multiplicador de longitud (ML)		0,2
	N es el contaje de desplazamiento		
RLD	Total = tiempo básico + (ML * N)		
	Tiempo básico		28
	Multiplicador de longitud (ML)		1,4
	N es el contaje de desplazamiento		
RLW	Total = tiempo básico + (ML * N)		
	Tiempo básico		27
	Multiplicador de longitud (ML)		0,9
	N es el contaje de desplazamiento		
ROUND			Típ. 56
			Máx. 110
RRB	Total = tiempo básico + (ML * N)		
	Tiempo básico		22
	Multiplicador de longitud (ML)		0,5
	N es el contaje de desplazamiento		
RRD	Total = tiempo básico + (ML * N)		
	Tiempo básico		28
	Multiplicador de longitud (ML)		1,7
	N es el contaje de desplazamiento		
RRW	Total = tiempo básico + (ML * N)		
	Tiempo básico		26
	Multiplicador de longitud (ML)		1,2
	N es el contaje de desplazamiento		
RTA	Total = tiempo básico + (ML * N)		
	Tiempo básico (para el primer dígito del resultado)		149
	Multiplicador de longitud (ML)		96
	N es el número de dígitos adicionales en el resultado		
RTS	Total = tiempo básico + (ML * N)		
	Tiempo básico (para el primer dígito del resultado)		154
	Multiplicador de longitud (ML)		96
	N es el número de dígitos adicionales en el resultado		
S	Para longitud = 1, indicada como constante		2,9
	De lo contrario: Total = tiempo básico + (long. * LM)		
	Tiempo básico		14
	Multiplicador de longitud (ML)		0,5
	Si la longitud se almacena como variable, sumar al tiempo básico		17
SCAT	Total = tiempo básico + (ML * N)		
	Tiempo básico		30
	Multiplicador de longitud (ML)		5,3
	N es el número de caracteres añadidos		
SCPY	Total = tiempo básico + (ML * N)		27
	Tiempo básico		4,6
	Multiplicador de longitud (ML)		
	N es el número de caracteres copiados		
SCRE			0,24
SCRT			10
SEG			15
SFND	Tiempo máx. = Tiempo básico +((N1-N2) * ML2) +(N2 * ML1)		
	Tiempo básico		39
	Multiplicador de longitud 1 (ML1)		7,6
	Multiplicador de longitud 2 (ML2)		6,8
	N1 es la longitud de la cadena de origen		
	N2 es la longitud de la cadena de búsqueda		
SHRB	Total = tiempo básico + (longitud * ML1) + ((long. /8) * ML2)		
	Tiempo básico (longitud constante)		48
	Tiempo básico (longitud variable)		52
	Multiplicador de longitud 1 (ML1)		1,0
	Multiplicador de longitud 2 (ML2)		1,5

Operación		µs
SI	Total = Tiempo básico + (long. * ML) Tiempo básico ML utilizando salidas integradas ML utilizando salidas de ampliación Si la longitud se almacena como variable, sumar al tiempo básico	 8,9 13 21 17
SIN		Típ. 900 Máx. 1070
SLB	Total = tiempo básico + (ML * N) Tiempo básico Multiplicador de longitud (ML) N es el contaje de desplazamiento	 23 0,2
SLD	Total = tiempo básico + (ML * N) Tiempo básico Multiplicador de longitud (ML) N es el contaje de desplazamiento	 29 1,1
SLEN		21
SLW	Total = tiempo básico + (ML + N) Tiempo básico Multiplicador de longitud (ML)) N es el contaje de desplazamiento	 27 0,6
SPA		371
SQRT		Típ. 460 Máx. 550
SRB	Total = tiempo básico + (ML + N) Tiempo básico Multiplicador de longitud (ML)) N es el contaje de desplazamiento	 22 0,6
SRD	Total = tiempo básico + (ML + N) Tiempo básico Multiplicador de longitud (ML)) N es el contaje de desplazamiento	 28 1,5
SRW	Total = tiempo básico + (ML + N) Tiempo básico Multiplicador de longitud (ML)) N es el contaje de desplazamiento	 27 1
SSCPY	Total = tiempo básico + (ML * N) Tiempo básico Multiplicador de longitud (ML) N es el número de caracteres copiados	 42 5,3
STD	Total = tiempo básico + (ML * N) Tiempo básico (para el 1er. carácter de orige) Multiplicador de longitud (ML) N es el número de caracteres de origen adicionales	 69 27

Operación		µs
STI	Total = tiempo básico + (ML * N) Tiempo básico (para el 1er. carácter de orige) Multiplicador de longitud (ML) N es el número de caracteres de origen adicionales	 58 27
STOPI	Sin circulación de corriente	4
STR	Total = tiempo básico + (ML * N) Tiempo básico (para el 1er. carácter de orige) Multiplicador de longitud (ML) N es el número de caracteres de origen adicionales	 51 81
SWAP		17
TAN		Típ. 1080 Máx.1300
TODR		331
TODRX Corrección del horario de verano		Típ. 391 Típ. 783
TODW		436
TODWX		554
TOF		36
TON		33
TONR		32
TRUNC		Típ. 53 Máx. 106
WDR		7
XMT		42
XORB		19
XORD		29
XORW		25

Tabla A.16. Tiempo de ejecución

A.8 FUNCIONES PARA ESTRUCTURAR PROGRAMAS EN EL S7-200.

A.8.1 ESTRUCTURAR UN PROGRAMA UTILIZANDO FUNCIONES SCR (SECUENTIAL CONTROL RELAY)

La operación CARGAR RELÉ DE CONTROL SECUENCIAL (LSCR) indica el comienzo de un segmento SCR. Si n = 1, se habilita la circulación de la corriente hacia el segmento SCR. La operación LSCR se debe finalizar con una operación SCRE.

La operación TRANSICIÓN DEL RELÉ DE CONTROL SECUENCIAL (SCRT) identifica el bit SCR que se debe habilitar (el siguiente bit S a activar). Cuando la corriente fluye hasta la bobina, el bit S indicado se activa y el bit S de la operación LSCR (que habilitó este segmento SCR) se desactiva.

La operación FIN DEL RELÉ DE CONTROL SECUENCIAL (SCRE) indica el fin de un segmento SCR.
Uso restringido:

Al utilizar los relés de control secuencial, deberá tener en cuenta los siguientes puntos:

- Los relés de control secuencial (SCRs) se pueden utilizar en el programa principal, mas no en las subrutinas o en las rutinas de interrupción.
- En un segmento SCR no se pueden usar las operaciones Saltar a meta (JMP) ni Definir meta LBL). Por tanto, no se pueden utilizar para saltar hacia adentro, ni hacia afuera el segmento SCR, ni tampoco dentro del mismo. No obstante, las operaciones de salto y de meta se pueden emplear para saltar por encima de segmentos SCR.
- En un segmento SCR no se pueden utilizar las operaciones FOR, NEXT ni END.

Dividir cadenas secuenciales:
En numerosas aplicaciones es necesario dividir una cadena secuencial en dos o más cadenas. Si una cadena secuencial se divide en varias cadenas, es preciso activar simultáneamente todas las nuevas cadenas secuenciales como muestra en la figura A.42.

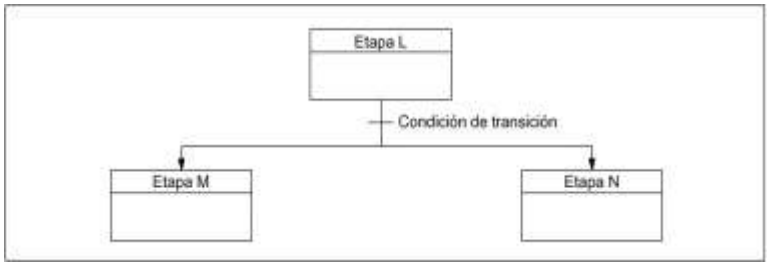


Fig. A.42. Dividir Cadenas Secuenciales.

La división de cadenas secuenciales se puede implementar en un programa SCR, activando varias operaciones SCRT con una misma condición de transición como muestra en la siguiente figura:

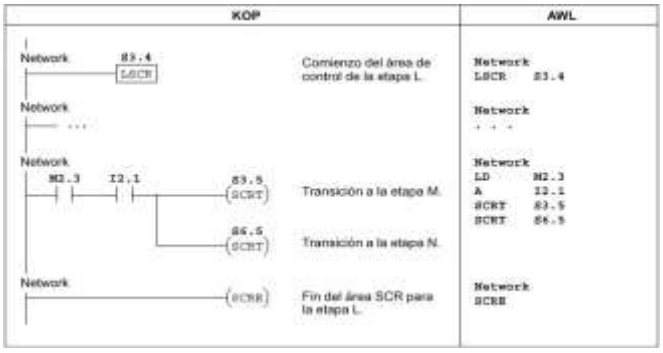


Fig. A.43. Dividir Cadenas Secuenciales SCR.

Converger cadenas secuenciales

Al ser preciso converger dos o más cadenas secuenciales para crear una cadena, se presenta una situación similar. Todas las cadenas secuenciales se deben terminar antes de poder ejecutar la siguiente etapa. La siguiente figura muestra la convergencia de dos cadenas secuenciales.

Fig. A.44. Converger Cadenas Secuenciales.

La convergencia de cadenas secuenciales se puede implementar en un programa SCR creando una transición de la etapa L a la etapa L', y de la etapa M a la etapa M'. Si los bits SCR que representan L' y M' son verdaderos, se podrá habilitar la etapa N como se muestra a continuación.

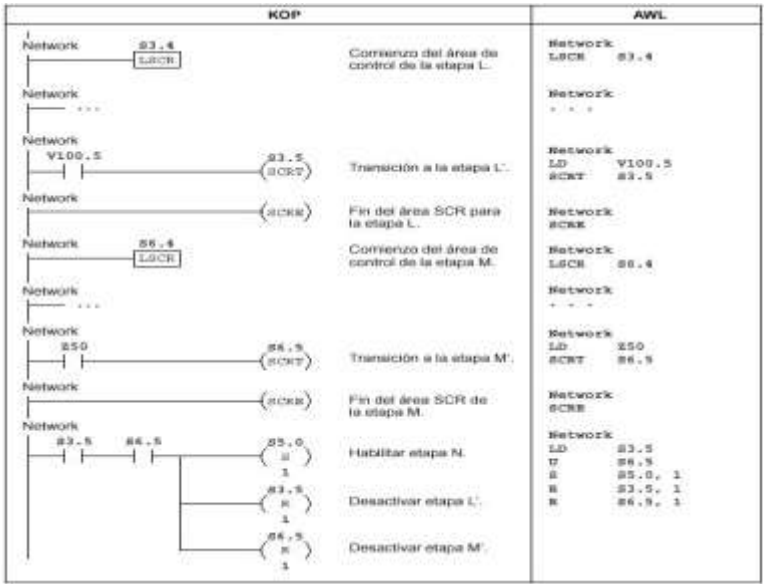


Fig. A.45. Converger Cadenas Secuenciales SCR.

En otras situaciones, una cadena secuencial se puede dirigir a una de varias cadenas secuenciales posibles, dependiendo de la primera condición de transición que sea verdadera. La siguiente figura muestra dicha situación.

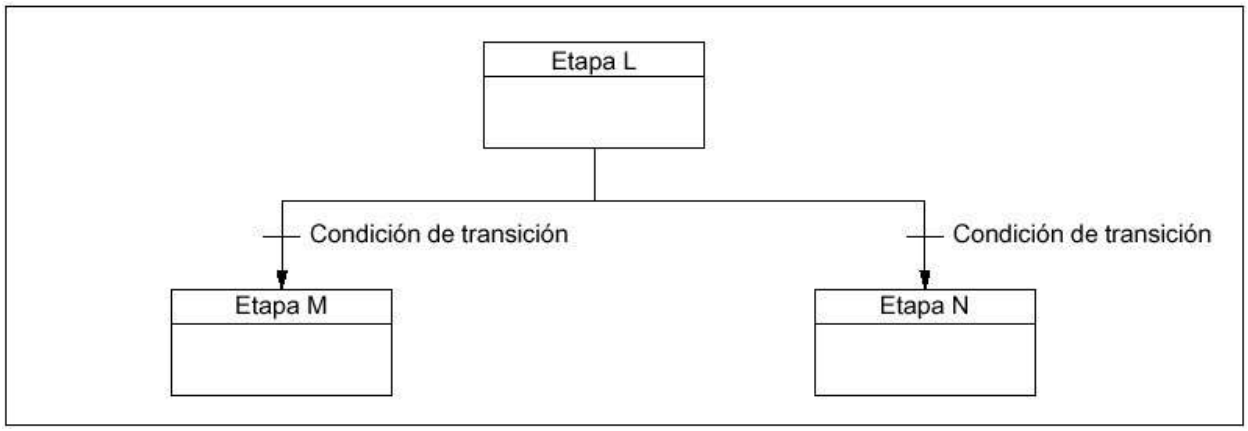


Fig. A.46. Dividir Cadenas Secuenciales.

La siguiente figura muestra el correspondiente programa SCR.

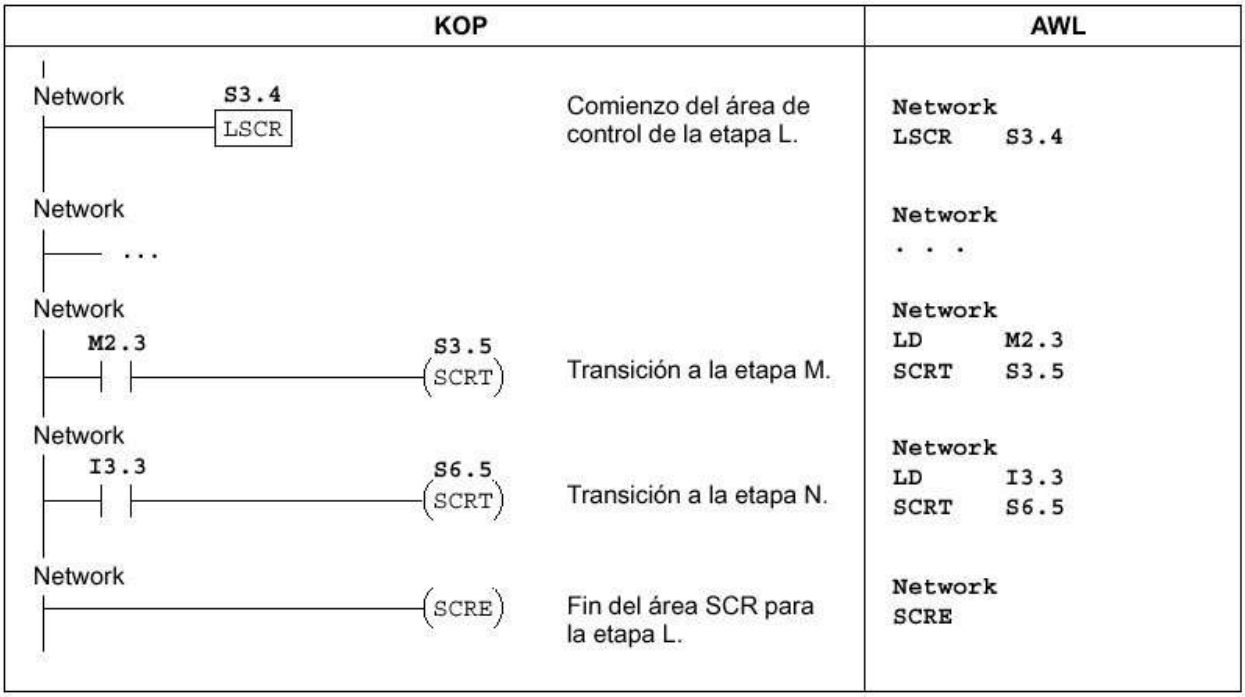


Fig. A.47. Dividir Cadenas Secuenciales SCR.

A.8.2 SUBRUTINAS

Es un grupo de instrucciones escritas por separado del programa principal para realizar una función que puede ser usada repetidamente por el programa principal.

Las subrutinas facilitan la estructuración del programa, una subrutina se implementa asociada con dos instrucciones:

- ✓ CALL llamado a la subrutina.
- ✓ RETURN retorno de la subrutina.

Cuando el programa principal llama a una subrutina para que esta se ejecute, la subrutina ejecuta su programa hasta el final. El sistema retorna de nuevo el control al segmento del programa principal desde donde se llamo la subrutina.

Las subrutinas sirven para estructurar o dividir el programa en bloques más pequeños y, por tanto más fáciles de gestionar. Esta ventaja se puede aprovechar a la hora de realizar tareas de comprobación y mantenimiento del programa.

Los bloques más pequeños facilitan la comprobación y eliminación de errores tanto en las subrutinas como en el programa entero.

La CPU también se puede usar más eficientemente, llamando al bloque solo cuando se necesite, en vez de ejecutar todos los bloques en cada ciclo.

Para realizar una subrutina en un programa es preciso realizar tres tareas:

- ✓ Crear la subrutina.
- ✓ Definir los parámetros (en caso necesario) en la tabla de variables de la subrutina.
- ✓ Llamar a la subrutina desde la unidad de organización del programa en cuestión(es decir, desde el programa principal o desde una subrutina diferente).

Los tipos de subrutinas pueden ser:

- ✓ Llamadas múltiples.
- ✓ Anidadas.
- ✓ De final múltiple.

La llamada a varias subrutinas desde el programa principal se muestra en la siguiente figura:

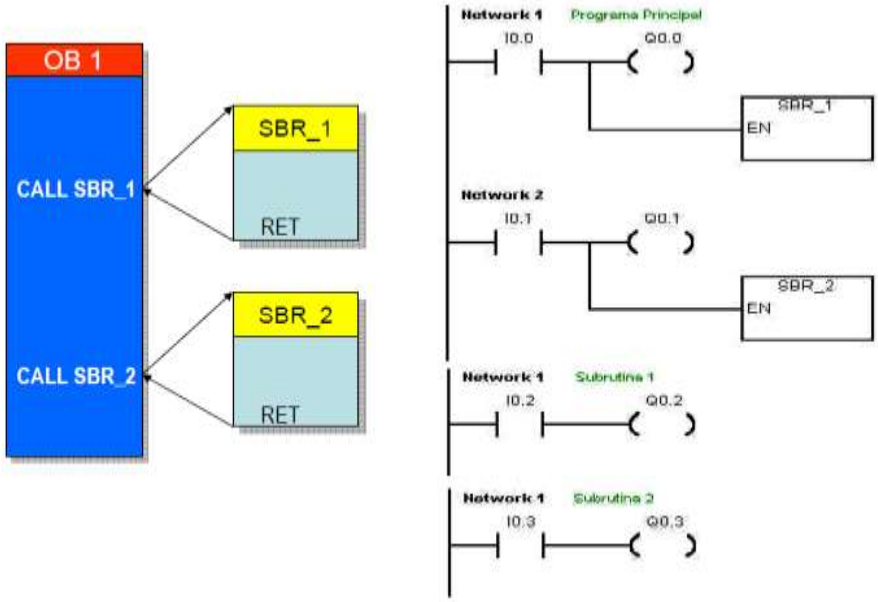


Fig. A.48. Llamada a una subrutina.

Una vez ejecutada la subrutina, el control vuelve a la operación que sigue a la llamada de la subrutina (CALL).

La llamada a una subrutina anidada se puede observar en la siguiente figura:

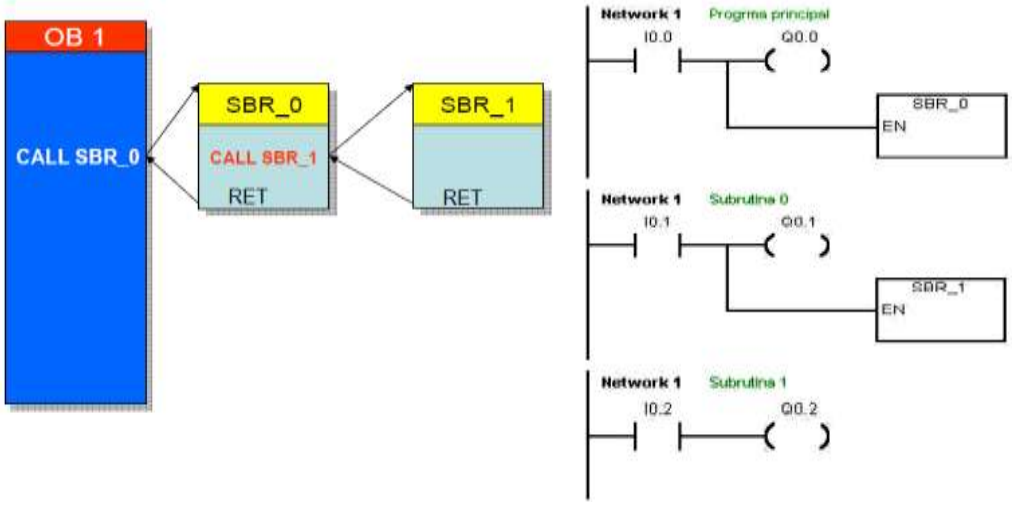


Fig. A.49. Subrutina anidadas.

La llamada a una subrutina con llamado múltiple se muestra en la fig. A.50.

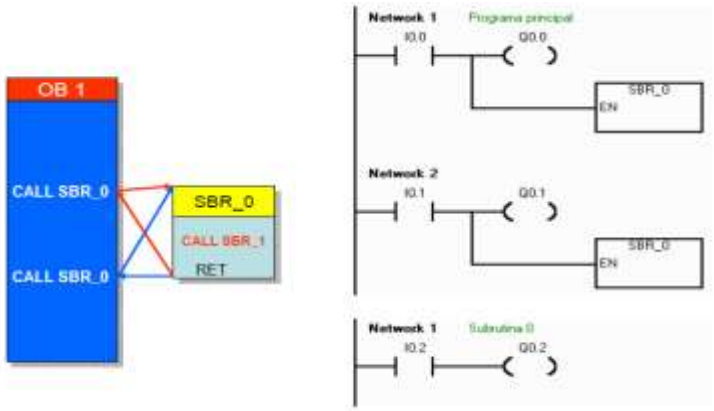


Fig. A.50. Subrutina llamado múltiple.

Si bien la recursión (la subrutina se llama a si misma) está permitida, hay que utilizarla cuidadosamente.

Cuando se llama a una subrutina, se almacena toda la pila lógica, poniendo a “1” el nivel superior de la pila. Sus demás niveles se ponen a “0” y la ejecución se transfiere a la subrutina que se ha llamado. Cuando esta se termina de ejecutar, se restablece la pila con los valores almacenados al llamar a la subrutina y se retorna a la rutina que ha efectuado la llamada.

Así mismo cuando se llama a una subrutina el primer valor de la pila es siempre “1” lógico. Por lo tanto es posible conectar salidas o cuadros directamente a la barra izquierda del segmento que sigue a la operación Comenzar subrutina (SBR).

Al utilizar subrutinas deberá tomar en cuenta los siguientes puntos:

- ✓ Ubique todas las subrutinas después del final del programa principal KOP.
- ✓ No utilizar las operaciones RET para finalizar las subrutinas.
- ✓ En una subrutina no se puede utilizar la operación END.
- ✓ Los editores insertan automáticamente las operaciones absolutas que finalizan las unidades de organización del programa (END para el programa principal, RET para SBR y RETI para INT)

La operación llamar a subrutina CALL transfiere el control a la subrutina (SBR_n), tal como se muestra en la fig.A.51.

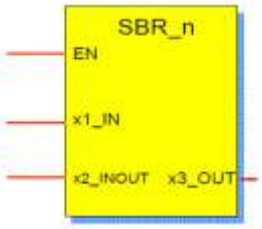


Fig. A.51. Función CALL

Esta operación se puede usar con o sin parámetros. Una vez ejecutada la subrutina el control vuelve a la operación que sigue a CALL.

El límite máximo de parámetros de entrada/salida en cada llamada a subrutina es de 16. Si intenta cargar un programa que exceda este límite ocurrirá un error.

Las flechas indican las operaciones que STEP 7 Micro-Win procesa automáticamente.

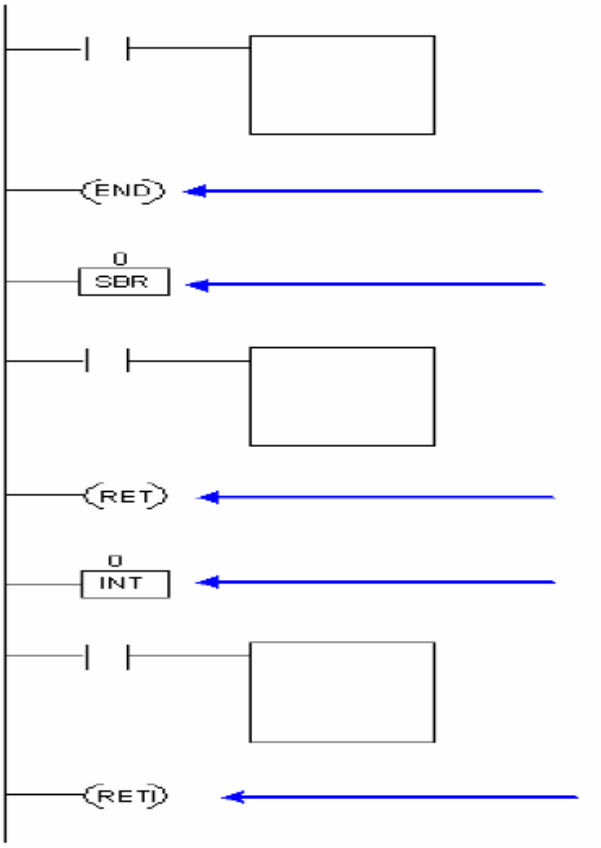


Fig. A.52. Operaciones que automáticamente procesa Micro-Win para una interrupción y subrutina.

A.8.3 INTERRUPCIONES

El procesamiento de interrupciones permite reaccionar rápidamente ante determinados eventos internos o externos.

Las rutinas de interrupción se deben estructurar de forma que una vez ejecutadas determinadas tareas devuelvan el control al programa principal. Para ello es conveniente crear rutinas de interrupción cortas con indicaciones precisas, de manera que se puedan ejecutar rápidamente sin interrumpir otros procesos durante períodos demasiado largos.

Fig. A.53. Rutina de interrupción

Si no se observan estas medidas, es posible que se produzcan estados imprevistos que pueden afectar a la instalación controlada por el programa principal.

Fig. A.54. Interrupciones en un PLC

Interrupciones del puerto de comunicación:

- ✓ El programa puede controlar el puerto serie de comunicación del S7-200.
- ✓ La comunicación a través de este puerto se denomina modo Freeport (comunicación programable por el usuario).
- ✓ En modo Freeport, el programa define la velocidad de transferencia, los bits por carácter, la paridad y el protocolo.
- ✓ Las interrupciones de transmisión y recepción permiten controlar la comunicación mediante el programa.

Interrupciones de E/S

Las interrupciones de E/S abarcan:

- ✓ interrupciones al producirse flancos positivos y negativos.
- ✓ interrupciones de los contadores rápidos, así como
- ✓ Interrupciones de salidas de impulsos.

El S7-200 puede generar una interrupción en los flancos positivos y/o negativos de una entrada (bien sea I0.0, I0.1, I0.2, o bien I0.3).

Los eventos Flanco positivo y Flanco negativo se pueden capturar para cada una de esas entradas. Estos eventos también sirven para indicar una condición que requiera atención inmediata en cuanto se produzca el evento.

Las interrupciones de los contadores rápidos permiten responder rápidamente a condiciones tales como:

a) El valor actual ha alcanzado el valor predeterminado.
b) El sentido de contaje ha cambiado de forma inversa al sentido de giro del árbol de accionamiento.
c) El contador se ha puesto a “0" externamente.

Todos estos eventos de los contadores rápidos permiten reaccionar ante eventos que no se puedan controlar durante el tiempo de ciclo del sistema de automatización.

Las interrupciones de salida de impulsos avisan inmediatamente cuándo ha finalizado la salida del número indicado de impulsos. Por lo general, las salidas de impulsos se utilizan para controlar motores paso a paso.

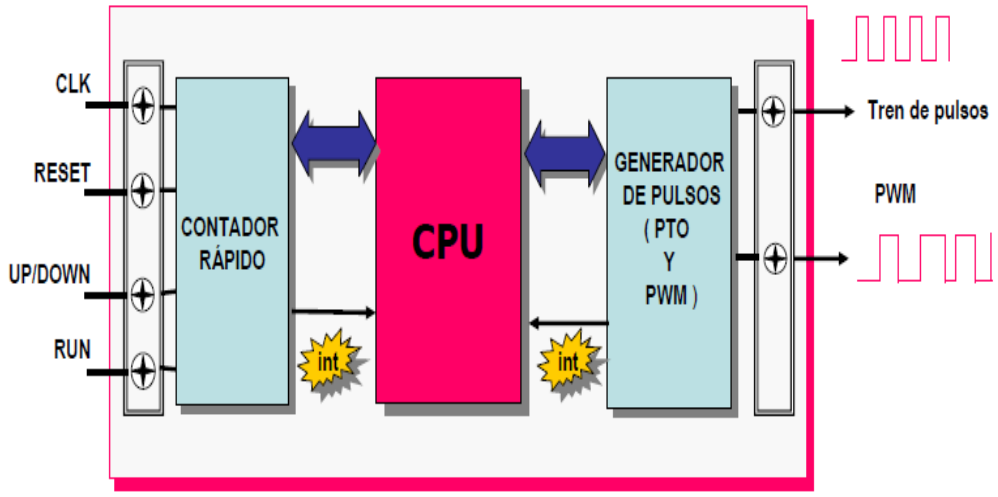


Fig. A.55. Interrupciones generadas por hardware interno

Interrupciones temporizadas

Una vez habilitada, la interrupción temporizada funciona de forma continua, ejecutando la rutina asociada cada vez que transcurre el intervalo de tiempo indicado. La interrupción temporizada se inhibe saliendo del modo RUN o desasociándola de la rutina correspondiente.

Si se ejecuta la operación Inhibir todos los eventos de interrupción, se siguen generando interrupciones temporizadas, pero se ponen en cola de espera (hasta que se habiliten nuevamente o hasta llenarse la cola).

Las interrupciones de los temporizadores T32 y T96 permiten reaccionar una vez transcurrido un determinado intervalo de tiempo.

Estas interrupciones se soportan únicamente en T32 y T96, siendo éstos temporizadores como retardo a la conexión (TON) y a la desconexión (TOF) con resolución de 1 ms. Por lo demás, el comportamiento de T32 y T96 es normal.

Una vez habilitada la interrupción, la rutina asociada se ejecutará cuando el valor actual del temporizador activo sea igual a su valor de preselección al actualizar el S7-200 el temporizador de 1 ms.

Estas interrupciones se habilitan asociando las correspondientes rutinas de interrupción a los eventos de temporización T32/T96.

Las interrupciones temporizadas comprenden también las de los temporizadores T32/T96. Estas interrupciones se utilizan para indicar tareas que deban ejecutarse cíclicamente. El tiempo de ciclo se incrementa en intervalos de 1 ms, abarcando desde 1 ms hasta 255 ms. El tiempo de ciclo de la interrupción temporizada 0 se debe escribir en SMB34, y el de la interrupción temporizada 1, en SMB35.

Cada vez que termina la temporización, el evento de interrupción temporizado transfiere el control a la rutina de interrupción correspondiente

Por lo general, las interrupciones temporizadas se utilizan para controlar el muestreo de las entradas analógicas o para ejecutar un bucle PID en intervalos regulares.

Asociando un evento de interrupción temporizado a una rutina de interrupción, se habilita el evento e inmediatamente se empieza a temporizar. Durante ese proceso, el sistema captura el valor del tiempo de ciclo, de forma que los cambios siguientes en SMB34 y SMB35 no lo pueden alterar. Para poder modificar el tiempo de ciclo se deberá cambiar el valor del mismo y re-asociar luego la rutina de interrupción al evento de la interrupción temporizada. Al re-asociar la rutina de interrupción, la función borra los tiempos acumulados de la asociación anterior, con lo cual se vuelve a temporizar a partir del nuevo valor.

OPERACIONES DE INTERRUPCIÓN

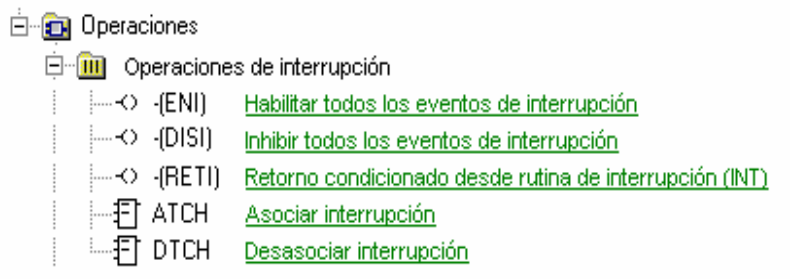


Fig. A.56. Operaciones de Interrupción.

- ✓ (ENI) habilita la ejecución de todos los eventos asociados.
- ✓ La operación Inhibir todos los eventos de interrupción (DISI) inhibe la ejecución de todos los eventos asociados.
- ✓ Cuando la CPU pasa a modo RUN, las interrupciones se inhiben.
- ✓ En modo RUN es posible habilitar el procesamiento de las interrupciones con la operación Habilitar todos los eventos de interrupción.
- ✓ Ejecutando la operación Inhibir todos los eventos de interrupción se inhibe el procesamiento de las interrupciones. No obstante, los eventos de interrupción activos se siguen poniendo en la cola de espera.

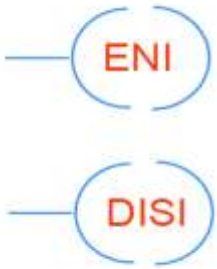


Fig. A.57. Habilitar / Inhibir eventos de interrupción

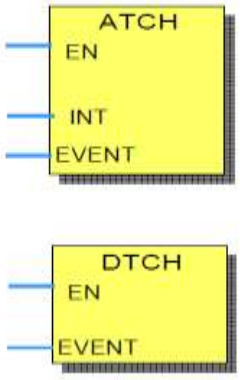


Fig. A.58. Operación Asociar interrupción

Asocia el número de una rutina de interrupción (INT) a un evento de interrupción (EVNT), habilitando así éste último.

Antes de poder llamar a una rutina de interrupción es preciso establecer un enlace entre el evento de interrupción y la parte del programa que se desee ejecutar cuando se presente el evento.

La operación Asociar interrupción sirve para asignar el evento de interrupción (indicado por el número de evento) a una parte del programa (indicada por el número de la rutina de interrupción).

También es posible asociar varios eventos de interrupción a una única rutina de interrupción. Por el contrario, no se puede asociar un sólo evento a distintas rutinas. Si se inhiben todos los eventos de interrupción, cada vez que se presente la interrupción se pondrá en cola de espera hasta que las interrupciones se habiliten de nuevo (utilizando para ello la operación Habilitar todos los eventos de interrupción), o bien hasta que se desborde la cola de espera de interrupciones.

También es posible inhibir ciertos eventos de interrupción, eliminando la asociación entre el evento y la correspondiente rutina mediante la operación Desasociar interrupción (DTCH).

Esta operación retorna la interrupción a un estado inactivo o ignorado.

Eventos de interrupción

La tabla A.17 muestra los distintos eventos de interrupción disponibles en el S7-200.

Evento	Descripción		CPU 221 CPU 222	CPU 224	CPU 224XP CPU 226
0	I0.0	Flanco positivo	Si	Si	Si
1	I0.0	Flanco negativo	Si	Si	Si
2	I0.1	Flanco positivo	Si	Si	Si
3	I0.1	Flanco negativo	Si	Si	Si
4	I0.2	Flanco positivo	Si	Si	Si
5	I0.2	Flanco negativo	Si	Si	Si
6	I0.3	Flanco positivo	Si	Si	Si
7	I0.3	Flanco negativo	Si	Si	Si
8	Puerto 0	Recibir carácter	Si	Si	Si
9	Puerto 0	Transmisión finalizada	Si	Si	Si
10	Interrupción temporizada 0	SMB34	Si	Si	Si
11	Interrupción temporizada 1	SMB35	Si	Si	Si
12	HSC0	CV=PV (valor actual = valor predeterminado)	Si	Si	Si
13	HSC1	CV=PV (valor actual = valor predeterminado)		Si	Si
14	HSC1	Cambio de sentido		Si	Si
15	HSC1	Puesto a 0 externamente		Si	Si
16	HSC2	CV=PV (valor actual = valor predeterminado)		Si	Si
17	HSC2	Cambio de sentido		Si	Si
18	HSC2	Puesto a 0 externamente		Si	Si
19	PLS0	Interrupción Valor de contaje de impulsos PTO	Si	Si	Si
20	PLS1	Interrupción Valor de contaje de impulsos PTO	Si	Si	Si
21	Interrupción temporizador T32	CT=PT	Si	Si	Si

Evento	Descripción		CPU 221 CPU 222	CPU 224	CPU 224XP CPU 226
22	Interrupción temporizador T96	CT=PT	Sí	Sí	Sí
23	Puerto 0	Recepción de mensajes finalizada	Sí	Sí	Sí
24	Puerto 1	Recepción de mensajes finalizada			Sí
25	Puerto 1	Recibir carácter			Sí
26	Puerto 1	Transmisión finalizada			Sí
27	HSC0	Cambio de sentido	Sí	Sí	Sí
28	HSC0	Puesto a 0 externamente	Sí	Sí	Sí
29	HSC4	CV=PV (valor actual = valor predeterminado)	Sí	Sí	Sí
30	HSC4	Cambio de sentido	Sí	Sí	Sí
31	HSC4	Puesto a 0 externamente	Sí	Sí	Sí
32	HSC3	CV=PV (valor actual = valor predeterminado)	Sí	Sí	Sí
33	HSC5	CV=PV (valor actual = valor predeterminado)	Sí	Sí	Sí

Tabla A.17. Eventos de Interrupción.

A continuación se muestra un ejemplo de una rutina de interrupción utilizando la entrada I0.0 disparo por flanco:

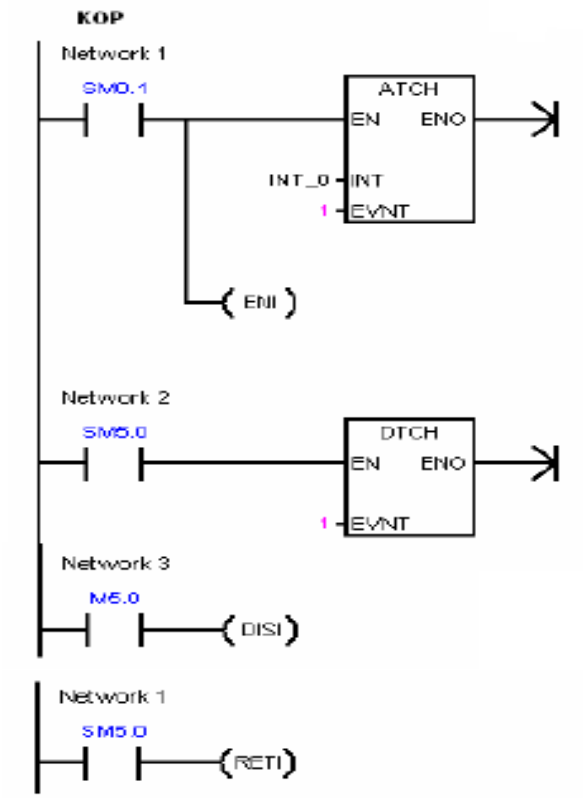


Fig. A.59. Ejemplo 1 interrupción.

- ✓ En el primer ciclo, definir que la rutina de interrupción INT_0 sea una interrupción de flanco negativo en I0.0 y habilitar todas las interrupciones.

- ✓ Si se detecta un error de E/S, inhibir la interrupción de flanco negativo en I0.0. (Este segmento es opcional)

- ✓ Si M5.0 está activada, inhibir todas las interrupciones.

- ✓ Rutina de interrupción 0

- ✓ Rutina de interrupción de flanco negativo en I0.0.

- ✓ Retorno condicionado debido a un error de E/S

Un segundo ejemplo de rutina de interrupción temporizada de 100 ms se muestra a continuación:

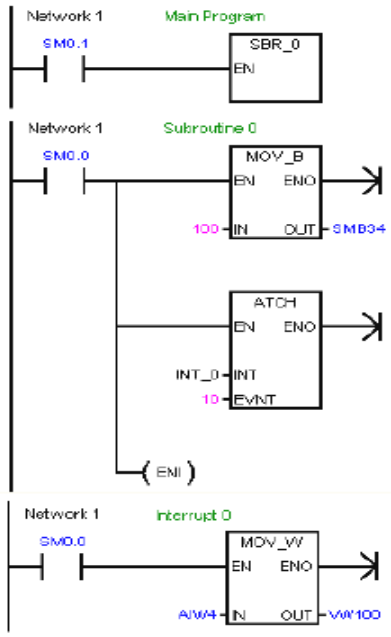


Fig. A.60. Ejemplo 2 interrupciones

Programa principal

- ✓ En el primer ciclo, llamar a la subrutina 0

Subrutina 0

- ✓ Ajustar de 0 a 100 ms el intervalo de tiempo de la interrupción temporizada.

- ✓ Asociar la interrupción temporizada 0 (evento 10) a INT_0.

- ✓ Habilitar todos los eventos de interrupción

Rutina de interrupción 0

- ✓ Leer el valor de AIW4 cada 100 ms

ANEXO B
MÉTODO PARA LA FABRICACIÓN DE CIRCUITOS IMPRESOS

En este tema, se mostrará una forma fácil y practica de hacer placas de circuito impreso, además es un método rápido y con buenos resultados a muy bajo costo.

Recopilación

Lo primero que se debe hacer, es recopilar el material necesario para la placa, y estos son:

- Agua oxigenada 110 Vol.
- Agua fuerte (clorhídrico).
- 1 plancha.
- 1 placa virgen para circuito impreso.
- 1 dremell o taladro que acepte brocas pequeñas.
- 1 par de brocas de 1mm.
- 1 permanente antiácido.
- Los componentes necesarios para nuestro proyecto.
- Acetona.
- Lana de acero.
- 1 martillo.
- 1 puntilla o punzón.
- 1 Hoja de papel Cuche.

Diseño

Diseñaremos nuestra placa con algún programa de diseño de circuitos por ordenador para obtener un resultado profesional, en nuestro caso se uso Ares; el editor de PCB`s del paquete Proteus con muy buenos resultados.

Impresión

Se imprime el diseño con una impresora láser, o se fotocopia el mismo en papel cuche, se han usado formatos de dibujo, o papel de colores. Se imprimirá con tóner negro y en buena calidad.

Fig. B.1 Impresión PCB.

Recorte

Se recorta la fotocopia como se indica en la imagen, de esta forma, se puede pegar los bordes a la placa.

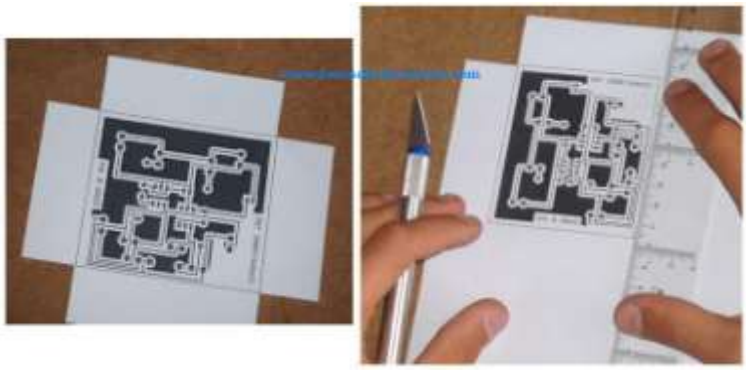


Fig. B.2 Recorte Impresión.

Recorte de la placa

Este es un proceso pesado y laborioso, ya que el corte de la placa con discos produce mucho polvo que no es conveniente respirar, así que se debe proteger de este. O se puede utilizar una sierra de mano para obtener un resultado mas fino.

Limpiado de la placa.

Para este proceso se deben tomar el tiempo necesario, se debe usar una lana de acero y la acetona, este proceso debe ser llevado lo mejor posible, ya que si la placa no queda bien limpia nunca fijara el tóner a la misma. Al terminar de limpiar secaremos la placa con un paño limpio y volveremos a limpiarla sin poner más los dedos sobre el cobre, ya que estos dejan grasa.

La limpieza de la placa solo será efectiva cuando esta quede brillante y con rayones en circulo para que agarre mejor el tóner. Esto se ve en la siguiente imagen.

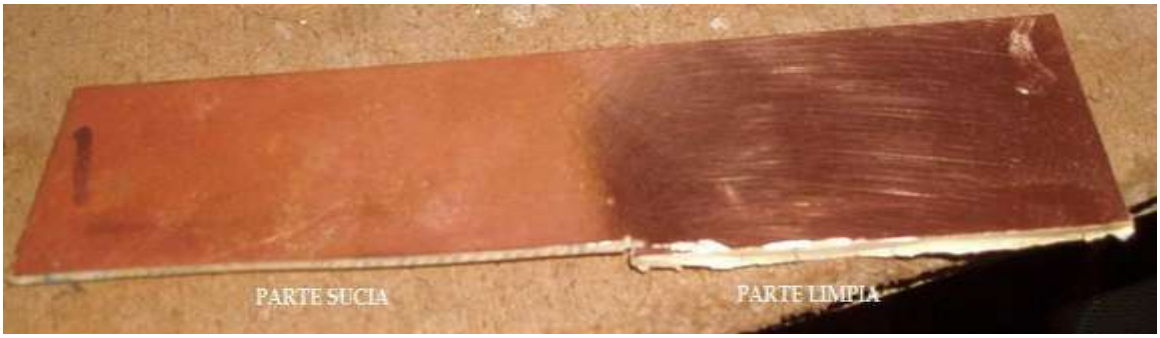


Fig. B.3 Limpiado Placa de cobre.

Planchado.

Con la plancha a tope de calor, se le aplica a la placa por la cara donde estaba el cobre, NUNCA por la trasera pues no serviría. Es importante insistir con el calor por toda la placa

y con vapor humedeciendo el papel para que no se queme pero sin empaparlo. Si se llegase a empapar, cortar la llave de vapor y dar calor seco unos instantes.

Fig. B.4 Planchado.

Enfriamiento.

En el instante que se retira la plancha de la placa, después de 10 minutos de calor intenso, a veces mas, se coloca la placa en un recipiente con agua para que el papel no tire (suelte) el tóner hacia arriba al enfriarse y se fije a la placa, esta debe mantenerse en el agua durante unos 5 minutos.

Fig. B.5 Enfriamiento.

Eliminar el papel.

Después de haber esperado 5 o 10 minutos en el agua, sacamos la placa y vamos frotando con los dedos para quitarle el papel que no nos sirve, intentando quitarlo todo, hasta que quede una capa muy fina de papel que se retira con un cepillo de dientes que ya no tengan en uso, con cuidado de no partir el tóner que define las pistas. Si pasa eso, se recomienda volver a la fase de limpiado.

Fig. B.6 Eliminar Residuos de Papel.

Repasar la Placa

Este es un paso que no se suele llevar a cabo, aunque de ser necesario, debe realizarse. Se recomienda repasar todas las pistas que lleve la placa para que al atacarla con el acido no queden poros y tengan luego que estañar o hacer puentes. Usen edding 3000 o superior (marcador permanente). Este simple paso, puede ahorrarnos luego mucho trabajo.

Fig. B.7 Placa lista para Atacar con acido.

La primera placa que se hizo con este método y no se repaso con el permanente, observen los poros que quedan en el tóner, lo que llevo a desecharla y empezar de nuevo.

Secado.

Una vez repasadas todas las pistas de la placa con el marcador permanente, se espera un par de minutos para que este fije y seque. Mientras tanto, podemos ir preparando el acido para atacar la placa.

Preparando el acido

Este es un proceso fácil; para preparar el acido mezclamos 2 partes de agua fuerte con 4 de agua oxigenada 110 vol. y 1 de agua. Si la mezcla resulta poco corrosiva, añadir agua fuerte y agua oxigenada en mismas proporciones.

Fig. B.8 Preparación Acido.

Atacando

Esta es la fase en la que debemos estar más atentos, pues si el acido resultara fuerte podría diluir el tóner. Lo ideal es que cuando coloques la placa en disolución, el cobre coja un color rojizo y empiece a burbujear. Miren la imagen.

Fig. B.9 Atacando PCB.

Enjuague y Limpieza

Una vez se saque la placa del acido hay que enjuagarla con abundante agua para que el acido no la sigua comiendo, luego conviene secarla con un trapo limpio. Una vez seca, se empapara el tóner con acetona y se rascara con un cepillo de dientes o con la lana de acero, eliminando así todo el tóner de la placa.

Taladrado de la Placa

Una vez listas las marcas, procederemos a taladrar la placa, para lo cual usaremos un taladro que acepte brocas de 1mm. Si la broca quedase pequeña y no fuera agarrada por el taladro, pueden colocar un trozo de cinta aislante, pero una mejor solución que se me ocurrió fue, con un trozo de cable rígido fino (del usado en telefonía), ir liando en vueltas muy juntas toda la parte trasera de la broca, una vez liada, la cojo con el tronillo o gato y la lleno de estaño, intentando que quede toda una pieza y solucionado, todavía y después de al menos 10 placas mas, la broca no me da ningún problema.

Fig. B.10 Taladrado Placa.

Soldadura de componentes

Luego de haber taladrado la tarjeta, se procedió a soldar los distintos componentes de la placa de circuito impreso, es importante hacer notar que si se iba a usar IC's se deben de montar en zócalos o bases, que mantengáis la punta del soldador limpia, y que vayáis soldando los componentes de los más pequeños a grandes (resistencias, zócalos, etc.).

Probado de la placa

Ya solo queda probar que todo funciona correctamente, y que el proyecto, cumpla bien su cometido. Ahora solo espero que todo les haya dado buen resultado y que como yo, hayan disfrutado haciendo sus trabajos.

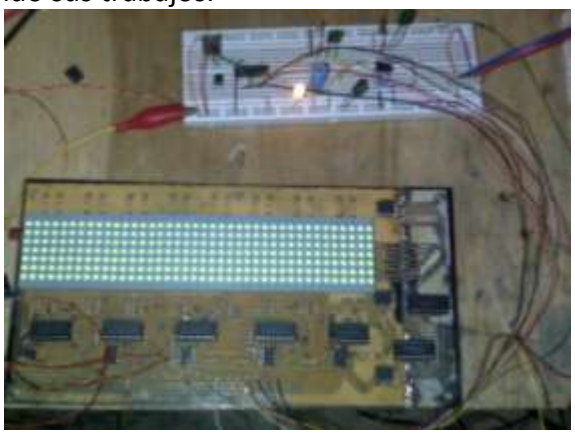


Fig. B.11 Prueba de Placa.

ANEXO C
FABRICACIÓN DEL CIRCUITO IMPRESO Y CHASIS DEL MODULO DIGITAL CON eZdsp F2812 Y TUTORIAL DE SIMULINK.

C.1 DISEÑO DEL PCB.

El desarrollo del esquema de circuito impreso al igual que las simulaciones se realizo en la herramienta de software Proteus 7.6. El PCB está realizado en una tableta de una sola cara, las dimensiones del circuito impreso son: 3'' de ancho por 5.4'' de largo, las mismas que la DSP. Se procuró en la medida de lo posible reducir al máximo la cantidad de puentes, pero por la densidad de pistas y el espacio reducido se logro minimizar hasta dejar 5 puentes en toda la tarjeta, algunos de los cuales están algo complicados como los de los integrados U5 y U6 que van de pin a pin en el mismo integrado, estos se marcan con una línea delgada verde en la figura.

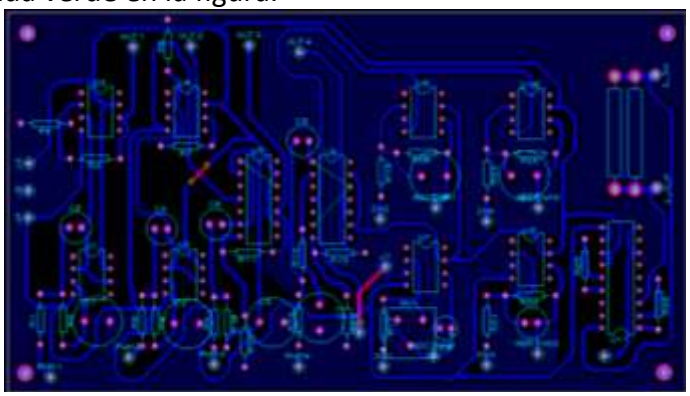


Fig. C.1. PCB del circuito de interfaz.

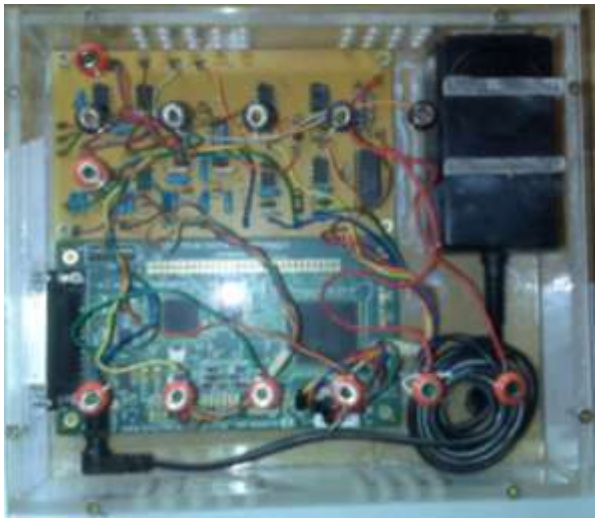
C.2 CONSTRUCCIÓN DE CHASIS.

Fig. C.2. Proceso de construcción del chasis.

El chasis fue construido en acrílico transparente de 5mm de espesor, sus dimensiones son 18cm de ancho, 21cm de largo y 5.5cm de alto. En la figura se muestra el proceso de construcción el cual consistió básicamente en el marcado de las piezas, el cortado de estas, el pegado de todas las piezas, la perforación de los agujeros de ventilación tornillos bananas y conectores, el alisado de los bordes y el ensamble total.

El ensamble de todos los elementos se puede ver en la figura C.3, es importante mencionar que a todo el modulo es necesario suministrarle +/- 15Vdc para el circuito de interfaz y una alimentación de 120Vac para la tarjeta ezDSPf2812 que cuenta con su propia fuente de alimentación.

Nota: debido a que el conector paralelo del modulo está montado directamente en la tarjeta DSP al momento de conectar el cable no forzar demasiado el conector, utilizar los tornillos de sujeción para hacer que el cable entre completamente.



a)

b)

*Fig. C.3. **a)** vista superior del modulo controlador **b)** vista del controlador con la conexión del cable paralelo y la extensión de alimentación.*

C.3 TUTORIAL DE SIMULINK.

El programa de MathWorks para simulación (modelización y análisis) de sistemas dinámicos no lineales fue presentado en 1990, con el nombre de SIMULAB para computadoras personales y con el nombre de SIMULINK para estaciones de trabajo. Su aparición estuvo unida a la primera versión de MATLAB para Windows. Desde mayo de 1994, que está disponible la versión 1.3, SIMULINK tiene un tratamiento similar a los otros Toolboxes de MATLAB, en el sentido que se instala de forma separada, pero sigue siendo la mejor herramienta para aprovechar toda la potencia de MATLAB y de los otros "Toolboxes".

Simulink es una herramienta interactiva para modelar, simular y analizar sistemas dinámicos. Permite construir diagramas de bloque gráficos, evaluar el rendimiento del sistema y refinar sus diseños. Simulink esta firmemente integrado con Stateflow para modelar comportamiento even-driven. Simulink es la herramienta a escoger para el diseño de sistemas de control, diseños DSP, diseños de sistemas de comunicaciones y otras aplicaciones de simulación. Como una extensión de Matlab, Simulink adiciona muchas características específicas a los sistemas dinámicos, mientras conserva toda la funcionalidad de propósito general de Matlab. Así Simulink no es completamente un programa separado de Matlab, sino un anexo a él. El ambiente de Matlab está siempre disponible mientras se ejecuta una simulación en Simulink.

Simulink tiene dos fases de uso: la definición del modelo y el análisis del modelo. La definición del modelo significa construir el modelo a partir de elementos básicos construidos previamente, tal como, integradores, bloques de ganancia o servomotores. El análisis del modelo significa realizar la simulación, linealización y determinar el punto de equilibrio de un modelo previamente definido.

Para simplificar la definición del modelo Simulink usa diferentes clases de ventanas llamadas ventanas de diagramas de bloques. En estas ventanas se puede crear y editar un modelo gráficamente usando el mouse. Simulink usa un ambiente gráfico lo que hace sencillo la creación de los modelos de sistemas. Después de definir un modelo este puede ser analizado seleccionando una opción desde los menús de Simulink o entrando comandos desde la línea de comandos de Matlab.

Simulink puede simular cualquier sistema que pueda ser definido por ecuaciones diferenciales continuas y ecuaciones diferenciales discretas. Esto significa que se puede modelar sistemas continuos y discretos en el tiempo o sistemas híbridos.

Simulink usa diagramas de bloques para representar sistemas dinámicos. Mediante una interfaz gráfica con el usuario se pueden arrastrar los componentes desde una librería de bloques existentes y luego interconectarlos mediante conectores y alambre.

Algunas Características Principales:

Facilidad de Uso:

- Una biblioteca extensa de bloques predefinidos para construir modelos gráficos de su sistema.
- El Debugger gráfico de Simulink para localizar y diagnosticar fallos en el modelo.
- El Visualizador de modelos para navegar por las jerarquías del modelo.
- Utilidad gráfica para buscar modelos y bibliotecas.
- Bloques personalizables que pueden incorporar código ya existente del usuario en C, Ada, MATLAB, y Fortran.

Ayuda para aplicaciones más grandes:

- Sistemas lineales, no lineales, tiempo continuo, tiempo discreto, multirate, ejecutados condicionalmente, e híbridos.
- Los modelos pueden agruparse en jerarquías para crear una vista simplificada de los componentes o los subsistemas.
- Los objetos de datos de Simulink le permiten crear tipos de datos de MATLAB específicos de la aplicación para los modelos de Simulink.
- Simulink Explorer GUI para visualizar y editar objetos de datos de Simulink.
- Visualizador de la biblioteca para seleccionar los bloques en una manera conveniente.
- Protección de propiedad intelectual utilizando funciones S (requiere Real-Time Workshop 4.0).
- Puede ejecutar las simulaciones desde la línea de comando de MATLAB, de forma interactiva o en modo batch.

Ayuda Computacional:

- Ayuda para las señales y las operaciones de matrices.
- El bloque Bitwise Logical Operator enmascara, invierte, o desplaza los bits de una señal entero y sin signo en una manera lógica.

Generador de código-C en Simulink.

Una vez se ha creado un modelo dinámico en Simulink, se puede invocar el generador de código-C que permite convertir el diagrama de bloques implementado en un código C. Este puede ser útil para varios propósitos: puede ser usado para control en tiempo real, simulación en tiempo real o simulación acelerada en tiempo no real. Sus aplicaciones pueden ser control de movimiento, control de procesos, sistemas automotores, equipos médicos, robótica, etc.

El código-C es diseñado tal que puede ser ejecutado en tiempo real. No requiere ser escrito manualmente por un programador pues es creado a nivel de diagramas de bloques en Simulink.

El código generado puede correr sobre un amplio rango de hardware ubicado en estaciones de trabajo, PC o microprocesadores. Este código es la forma en la que puede usare el Simulink para adquisición de datos.

Construcción de modelos

Simulink le permite crear modelos y simulaciones de prácticamente cualquier tipo de sistema dinámico del mundo real. Usando las potentes funciones de simulación, puede crear modelos, evaluar diseños y corregir imperfecciones en éstos antes de construir los prototipos.

Algunas recomendaciones para construir modelos:

- **Use sistemas jerárquicos:** Los modelos complejos se benefician de la incorporación de jerarquías usando subsistemas. La agrupación de bloques simplifica el nivel superior del modelo haciéndolo fácil de entender.
- **Cree modelos claros:** Modelos bien organizados y documentados son fáciles de entender e interpretar. Para lo anterior, utilice etiquetas para las señales y anotaciones sobre el modelo.
- **Diseñe el modelo en un papel y después use la PC**, coloque primero los bloques que se van a utilizar y luego realice las conexiones, esto último reduce el tiempo de diseño a través de sucesivas aperturas de las bibliotecas de bloques.

Los elementos fundamentales para construir modelos con Simulink son: Líneas, Bloques y Subsistemas, estos serán explicados en secciones posteriores.

Qué hace el Simulink

Permite definir de una forma lógica las diferentes etapas del proceso de diseño en el algoritmo de DSP y el flujo de datos entre ellas. Cada bloque puede representar un solo elemento del proceso o bien un subsistema, además de ser fácilmente modificable para reflejar un cambio en el algoritmo o el enfoque del diseño. Estos diagramas de bloques permiten una descripción en alto nivel del sistema además de ser fácilmente modificables con la finalidad de conseguir el comportamiento deseado, es decir, proporcionan una estructura jerárquica para evaluar el comportamiento de algoritmos alternativos bajo diferentes condiciones de funcionamiento.

En los grupos de investigación y desarrollo, SIMULINK permite que cada persona integrante del mismo pueda entender y modificar de una forma sencilla los detalles del algoritmo y el progreso hacia la fase de implementación.

Simulink incluye muchas características para el análisis detallado de sistemas. Las capacidades principales incluyen: linealización, determinación de puntos de equilibrio, animación, optimización de los parámetros, y análisis paramétrico.

- **Extracción de Modelos Lineales**
  Las dinámicas de diagramas de bloques no lineales pueden ser aproximadas por linealización, permitiendo aplicar técnicas de diseño que requieran representaciones de modelos lineales. Puede usar la función linmod de Simulink para conseguir modelos lineales del espacio de estados de su diagrama de bloques.

- **Animación**
  Simulink proporciona acceso inmediato a los potentes gráficos 2-D y 3-D y capacidades de animación de MATLAB. Puede usar MATLAB para mejorar las visualizaciones y entender mejor el comportamiento de su sistema durante el progreso de la simulación. Para acceder a las capacidades de animación de MATLAB, usted puede usar las funciones S de Simulink para incorporar comandos gráficos estándar de MATLAB.

- **Análisis Paramétrico**
  Con Simulink, el proceso de diseño puede ser automatizado con el uso de ficheros con comandos de MATLAB (M-files) para ejecutar simulaciones múltiples con parámetros variables. Los gráficos resultantes muestran familias de curvas que ilustran la respuesta temporal en función de los parámetros variados. Puede también mejorar el rendimiento del sistema con el uso de funciones contenidas en el Toolbox de Optimización o el Blockset de Diseño de Control no Lineal para sintonizar los parámetros del modelo.

- **Integración con MATLAB**
  Dado que Simulink está construido a partir de MATLAB, proporciona un entorno único de desarrollo. Este sistema permite ejecutar las simulaciones de manera interactiva , con el uso de las interfaces gráficas de Simulink, o de manera sistemática, con la ejecución de conjuntos de experimentos en el modo batch desde la línea de comandos de MATLAB. Entonces puede generar vectores de prueba y analizar los resultados colectivamente.

C.3.1 CREACIÓN DE UN MODELO.

C.3.1.1 Ventanas y Menús en Simulink.

Para comenzar Simulink, se debe arrancar primero Matlab. En el indicador de Matlab, se introduce la orden Simulink. El computador incluirá la ventana de órdenes de Matlab y la ventana Simulink Block Library.

La ventana Simulink Block Library visualiza los íconos de sus bibliotecas de bloque. Construye modelos copiando bloques de la biblioteca de bloques en una ventana de modelo.

Cuando se ejecuta una simulación y se analizan sus resultados, se puede utilizar órdenes de Matlab que se introducen desde la ventana de órdenes de Matlab.

Simulink utiliza ventanas separadas para visualizar una biblioteca de bloques, un modelo o la salida de una simulación gráfica. Estas ventanas no son ventanas de figura de Matlab y no se pueden manipular utilizando las órdenes del entorno gráfico de Matlab.

Los menús de Simulink aparecen cerca de la parte superior de cada ventana Simulink. Las órdenes del menú se aplican a los contenidos de esa ventana.

Un error frecuente de los nuevos usuarios de Simulink es comenzar una simulación mientras el Simulink Block Library es la ventana activa. Hay que asegurarse de que la ventana del modelo es la ventana activa antes de comenzar una simulación.

Para finalizar una sesión de Simulink se escoge Exit Matlab en el menú File.

C.3.1.2 Construcción de un modelo.

Esta sección analiza las tareas que se llevan a cabo durante la construcción de un modelo. Para crear un nuevo modelo, se escoge la orden New en el menú File. Simulink crea una nueva ventana. Se puede mover la ventana de la misma forma que se hace con otras ventanas.

Para editar el diagrama de un modelo existente, se debe hacer una de las dos acciones siguientes:

- Escoger la orden Open en el menú File y especificar el archivo-mdl que describe el modelo que se desea editar.
- Introducir el nombre del modelo en la ventana de orden de Matlab.

Simulink crea una nueva ventana y visualiza ese modelo en la ventana.

Seleccionar objetos.

Muchas acciones de edición y construcción de modelos requieren que primero se seleccione uno o más bloques y líneas (objetos).

Selección de un objeto.

Para seleccionar un objeto, se sitúa el cursor encima del objeto y se pulsa el botón del ratón. Aparecen pequeñas asas en las esquinas del objeto. Por ejemplo, la figura que sigue muestra la selección de un bloque Sine Wave y de una línea:

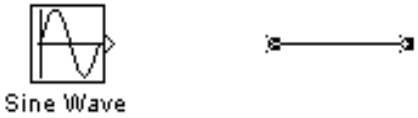


Fig. C.4. Bloque de onda senoidal.

Cuando selecciona un objeto pulsando encima de él, cualquier otro que lo estuviera deja de estarlo.

Selección de más de un objeto.

Puede seleccionar más de un objeto seleccionándolos uno a uno o todos a la vez si están próximos utilizando un recuadro que los englobe.

Selección de objetos de uno en uno.

Para seleccionar más de un objeto, haciéndolo de uno en uno, se debe mantener pulsada la tecla Shift y pulsar sobre cada objeto que se desea seleccionar. Para desactivar la selección de un objeto que está seleccionado, se pulsa otra vez sobre el mismo mientras se mantiene pulsada la tecla Shift.

Selección de objetos utilizando un cuadro de delimitación.

Una manera fácil de seleccionar más de un objeto en la misma área de la ventana es dibujar un cuadro de delimitación alrededor de los objetos. Para definir el cuadro de delimitación se debe hacer lo siguiente:

- Se define la esquina de comienzo de un cuadro de delimitación posicionando el puntero en un ángulo del cuadro, a continuación se pulsa el botón del ratón.

Fig. C.5. Posición del puntero para seleccionar más de un bloque a la vez.

- Luego se arrastra el puntero al ángulo opuesto del cuadro.

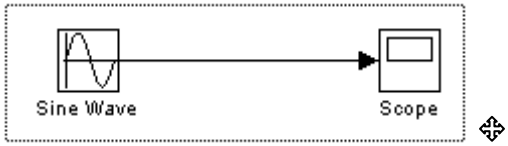


Fig. C.6. Posición del puntero luego de seleccionar los bloques.

- Seguido se debe liberar el botón del ratón. Quedan seleccionados todos los bloques y líneas que están parcialmente encerrados por el cuadro de delimitación.

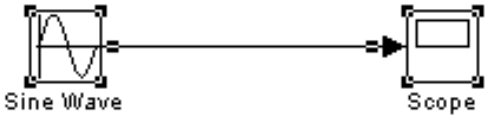


Fig. C.7. Bloques seleccionados.

C.3.1.2.2 Manipulación de bloques.

Se presentará cómo efectuar acciones útiles para construir modelos en los que intervienen bloques.

C.3.1.2.2.1 Copiar y mover bloques de una ventana a otra

Para copiar y mover bloques de una ventana a otra se deben hacer los siguientes pasos:

1. Abrir la biblioteca de bloques apropiada o la ventana del modelo fuente.
2. Arrastrar el bloque que se desea copiar en la ventana del modelo.

También se puede copiar bloques utilizando **Copy** y **Paste** en el menú **Edit** mediante los siguientes pasos:

1. Seleccionar el bloque que se desea copiar.
2. Escoger Copy en el menú Edit.
3. Seleccionar la ventana en el modelo de forma que sea la ventana activa.
4. Escoger Paste en el menú Edit.

Otra forma de copiar o mover bloques entre aplicaciones que sean compatibles, es empleando las órdenes **Copy, Cut** y **Paste.**

C.3.1.2.2.2 Mover bloques en un modelo.

Para mover un solo bloque de un lugar a otro en una ventana de modelo, se de seleccionar y arrastrar a una nueva posición.

Para mover más de un bloque, incluyendo las líneas de conexión, se debe hacer lo siguiente:

1. Seleccionar los bloques y las líneas.
2. Arrastrar los bloques y las líneas seleccionadas a sus nuevas posiciones y liberar el botón del ratón.

C.3.1.2.2.3 Duplicar bloques en un modelo.

La forma de duplicar bloques en un modelo depende del tipo de computador que se esté utilizando:

- ✓ **Windows:** mantener pulsada la tecla **Ctrl,** con el botón izquierdo del ratón seleccionar el bloque y arrastrarlo a una nueva localización. También se puede seleccionar el bloque pulsando el botón derecho del ratón mientras el puntero se encuentra sobre el bloque y lo arrastra a una nueva posición.

- ✓ **Macintosh:** mantener pulsada la tecla **Crtl,** seleccionar el bloque y arrastrarlo a una nueva localización.

C.3.1.2.2.4 Especificación de los parámetros del bloque.

Ciertos aspectos de una función de bloque se definen a través de sus parámetros. Se puede asignar valores a los parámetros de un bloque accediendo a su cuadro de diálogo. Haciendo doble clic, al bloque, se visualiza el cuadro de diálogo del bloque, que lista los parámetros y sus valores actuales. Se puede cambiar estos valores o aceptar los valores visualizados.

C.3.1.2.2.5 Suprimir bloques.

Para suprimir uno o más bloques, se debe seleccionar y pulsar la tecla **Delete** o escoger del menú **Edit** la opción **Clear o Cut.** La orden **Cut** escribe el bloque o los bloques en el portapapeles dejándolos disponibles para que se puedan pegar en un modelo. La utilización de la tecla **Delete** o la orden **Clear** no afecta a los contenidos del portapapeles.

C.3.1.2.2.6 Desconectar bloques.

Para desconectar un bloque del modelo sin suprimirlo, se debe mantener pulsada la tecla **Shift** y seleccionar y arrastrar el bloque desde su posición original en el modelo.

C.3.1.2.2.7 Cambiar la orientación de los bloques.

Hay varias órdenes que permiten cambiar la orientación de un bloque, éstas son:

- La orden **Rotate** en el menú **Options** gira un bloque 90° en el sentido de las agujas de un reloj.
- La orden **Flip Horizontal** (sistema Windows) o la orden **Flip** (sistema Macintosh) en el menú **Options** gira el bloque 180°.
- La orden **Orientation** en el menú **Style** le permite seleccionar la orientación del bloque **Left to Right, Right to Left, Up o Down.**

En la siguiente figura Simulink ordena los puertos después de cambiar la orientación de un bloque utilizando las órdenes **Rotate y Flip**. Los textos en los bloques indican su orientación.

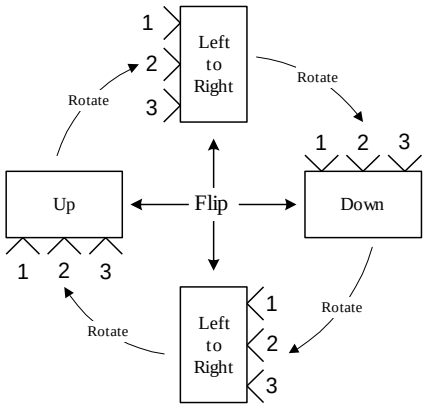


Fig. C.8.Efecto de la función Rotar en los bloques.

C.3.1.2.2.8 Redimensionar los bloques.

Para cambiar el tamaño de un bloque, se debe seleccionar y arrastrar cualquiera de sus asaderas de selección. El tamaño mínimo de un bloque es de cinco por cinco pixeles. El tamaño máximo está limitado por el tamaño de la ventana. La forma del cursor refleja la esquina y la dirección en la que se está siendo arrastrada. Mientras el bloque está siendo redimensionado, un rectángulo punteado muestra el tamaño propuesto.

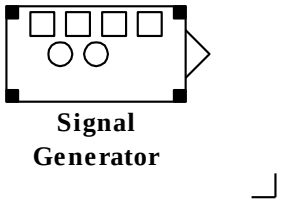


Fig. C.9. Cambio de tamaño de un bloque.

C.3.1.2.2.9 Manipulación de los nombres de los bloques.

Los nombres de bloques en un modelo deben ser únicos y deben contener al menos un carácter. Estos nombres aparecen debajo de los bloques si los puertos están en los laterales y a la derecha de los bloques si están en la parte superior o inferior. Puede cambiar los nombres de los bloques y sus localizaciones.

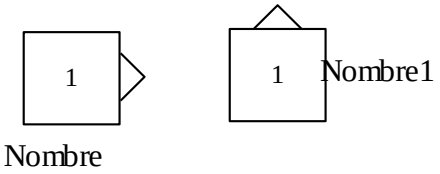


Fig. C.10. Cambio de nombre a los bloques.

C.3.1.2.2.9.1 Cambiar los nombres de los bloques.

Se puede editar los nombres de los bloques de una de las tres formas:

1. Seleccionando el recuadro en el que se visualiza el nombre e introduciendo el nuevo.
2. Colocando el punto de inserción en el nombre e introduciendo el nuevo texto.
3. Arrastrando el ratón para seleccionar el trozo de texto a reemplazar e introduciendo el nuevo texto.

Cuando se pulsa el puntero sobre otro bloque o se realiza cualquier otra acción, el nombre se acepta o se rechaza. Si intenta modificar el nombre de un bloque a un nombre que ya existe o a uno que no tiene caracteres, se visualiza un mensaje de error.

También se puede modificar las fuentes utilizadas en los nombres de los bloques seleccionando el bloque o los bloques y escogiendo un tipo fuente del submenú **Fonts** que está en el menú **Style.**

C.3.1.2.2.9.2 Cambiar la localización de un nombre de bloque.

Se puede cambiar la localización del nombre de un bloque eligiendo la opción **Title** que está en el menú **Style:**

- ✓ **Displayed,** visualiza el nombre.
- ✓ **Hidden,** no visualiza el nombre.
- ✓ **Top/Left** coloca el nombre encima del bloque cuando su orientación en ***Left to Right*** o ***Right to Left*** o a la izquierda del bloque cuando su orientación es ***Up o Down*.**
- ✓ **Bottom/Right,** coloca el nombre debajo del bloque cuando su orientación es ***Left to Right*** *o* ***Right to Left*** o a la derecha del bloque cuando su orientación es ***Up o Down*.**

Por ejemplo en la figura se muestra la posición de los nombres de bloque Top/Left.

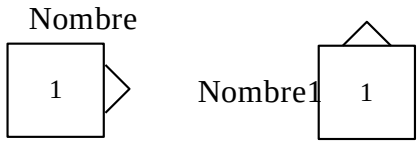


Fig. C.11. Cambio de posición del nombre en los bloques.

C.3.1.2.2.10 Vectorización de los bloques.

Casi todos los bloques incorporados aceptan entradas escalares o vectoriales y le permiten especificar parámetros de uno u otro tipo.

Para determinar qué líneas en un modelo llevan señales vectoriales, se elige la opción **Wide Vector Lines** en el menú **Style,** luego se dibuja las líneas que llevan vectores más gruesas que las que contienen escalares.

Después de elegir esta opción y si ha cambiado el modelo, se debe actualizar explícitamente la visualización seleccionando la opción **Update Diagram** en el menú **Style**. Para comenzar la simulación también actualiza la visualización.

C.3.1.2.2.11 Expansión escalar de entradas y parámetros.

La expansión escalar es la conversión de un valor escalar en un vector de elementos idénticos. Se puede aplicar la expansión escalar a las entradas y parámetros de bloques.

- ✓ **Entradas:** cuando se utilizan bloques que poseen más de un puerto de entrada, puede mezclar entradas vectoriales y escalares. Las entradas escalares se expanden en vectores que tienen la misma longitud que las entradas vectoriales. Por ejemplo, el bloque Sum muestra el resultado de expandir una entrada escalar para que concuerde el tamaño de una entrada vectorial al bloque.

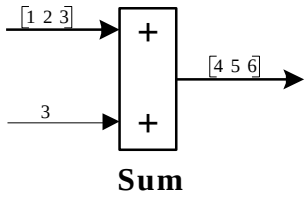


Fig. C.12. Diferentes entradas.

- ✓ **Parámetros:** se puede especificar los parámetros para bloques vectorizados como vectores o como escalares. Cuando se especifica parámetros, cada elemento se asocia con el vector de entrada correspondiente. Cuando se especifica parámetros escalares, se aplica la expansión escalar para convertirlos automáticamente en vectores del tamaño adecuado. Por ejemplo, la figura muestra el parámetro escalar (Gain) se expande para coincidir con el tamaño de la entrada al bloque, un vector de tres elementos.

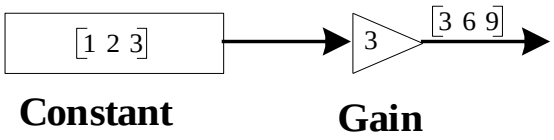


Fig. C.13. Cambio de parámetros.

C.3.1.2.3 Manipulación de líneas.

Las líneas conectan la salida de un bloque a la entrada de otro bloque. Las líneas también conectan otras líneas a la entrada de un bloque. Pueden conectarse cualquier número de líneas a un puerto de salida, pero solamente se puede conectar una línea a cada puerto de entrada. (El bloque MUX es útil para combinar algunas líneas en una única línea vectorial).

C.3.1.2.3.1 Dibujar líneas entre bloques.

1. Posicionar el cursor sobre el puerto de salida del primer bloque. No es necesario posicionar el cursor de forma precisa sobre el puerto.

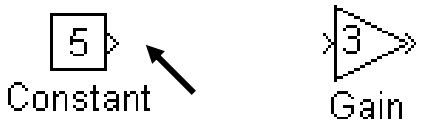


Fig. C.14. Unión de bloques.

2. Presionar y mantener pulsado el botón del ratón. El cursor cambia a una forma de cruz.

Fig. C.15. Unión de bloques.

3. Arrastrar el puntero al puerto de entrada del segundo bloque. Se puede posicionar el cursor sobre o cerca del puerto o dentro del bloque. Si se coloca el cursor en el bloque, la línea se conecta al primer puerto de entrada disponible. Para conectar la línea a un puerto especifico, se debe posicionar el cursor sobre ese puerto antes de soltar el botón del ratón.

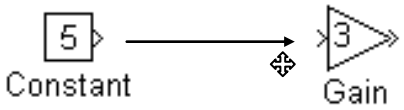


Fig. C.16. Unión de bloques.

4. Soltar el botón del ratón. Simulink sustituye los símbolos de los puertos por una línea de conexión con una flecha que muestra la dirección del flujo de señal. Se puede crear líneas de conexión o desde la salida a la entrada o desde la entrada a la salida. En cualquier de los dos casos, la flecha se dibuja en el puerto de entrada apropiado y la señal es la misma.

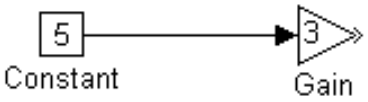


Fig. C.17. Unión de bloques.

Encaminamiento de líneas alrededor de bloques

SIMULINK encamina automáticamente las líneas alrededor de los bloques en lugar de pasar a través de ellos. Sin embargo, se le puede indicar a SIMULINK que dibuje una línea exactamente como se desee, manteniendo pulsada la tecla SHIFT mientras se dibuja la línea, dibujando la línea del puerto de entrada al puerto de salida o dibujando una secuencia de segmentos de líneas.

La opción **Reroute Lines** es útil para limpiar zonas muy pobladas del diagrama de bloques. Utilizando el cuadro de delimitación, se selecciona un área del modelo y se escoge **Reroute Lines** en el menú **Options**.

Dibujar líneas desde otras líneas

Se puede añadir una línea que comience en cualquier punto de una línea existente. Ambas líneas transportan la misma señal a sus salidas.

Por ejemplo, en la siguiente figura el diagrama de bloque del lado izquierdo muestra una única línea que va desde el bloque **Product** al bloque **Scope**. El diagrama detallado derecho muestra una línea adicional que va desde el bloque **Product** al bloque **To Workspace**. La misma señal va a cada bloque.

Para añadir una línea desde otra línea, se deben seguir estos pasos.

1. Colocar el puntero en la línea donde se desea comenzar la nueva línea.
2. Mientras se mantiene pulsada la tecla Ctrl., pulsar y mantener presionado el botón del ratón.
3. Arrastrar el puntero al puerto destino y soltar el botón del ratón y la tecla Ctrl. SIMULINK crea una nueva línea entre los puntos de comienzo y finalización.

En el sistema Windows, se puede también utilizar el botón derecho del ratón en lugar de mantener pulsada la tecla **Ctrl** mientras se utiliza el botón izquierdo del ratón.

Dibujar un segmento de línea

Para dibujar un segmento de línea, se dibuja una línea que finaliza en una zona no ocupada del diagrama. Aparece una flecha en el final no conectado de la línea. Para añadir otro segmento de línea, se pulsa el botón del ratón mientras el puntero este sobre la flecha y luego repetir el procedimiento.

Se puede utilizar esta técnica para dibujar una línea con segmentos exactamente donde se desee o para dibujar líneas antes de copiar bloques a los cuales se conectan las líneas. Cuando se comienza una simulación, Simulink proporciona un mensaje de aviso si el modelo tiene algunas líneas que no están conectadas.

La figura que se muestra a continuación tiene un segmento de línea sin conectar.

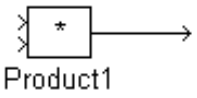


Fig. C.18. Segmento de línea no conectado.

Ángulos de líneas.

SIMULINK dibuja líneas de conexión en múltiplos de 45 grados con estas excepciones:

- Si el puntero se mueve cerca de un puerto que está disponible, la línea se conecta a ese puerto.
- Si se crea una línea mientras se mantiene pulsada la tecla **Shift**, SIMULINK dibuja la línea tal como la crea.
- Si se crea una línea moviendo el puntero sobre el bloque y se suelta, la línea va al puerto no usado en el bloque que está más arriba o más a la izquierda.

C.3.1.2.3.2 Suprimir líneas.

Para suprimir una o más líneas, se selecciona la línea o líneas que se van a eliminar y se pulsa la tecla **Delete** o se escoge **Clear** o **Cut** en el menú **Edit.**

C.3.1.2.3.3 Mover segmentos de línea.

Para mover un segmento de línea se deben seguir estos pasos:

1. Posicionar el puntero sobre el segmento

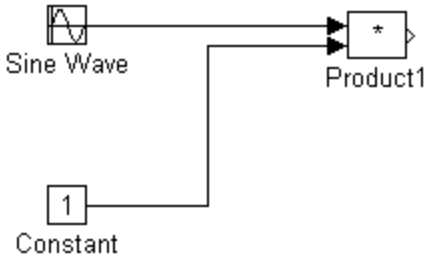


Fig. C.19. Seleccionar segmentos de línea.

2. Pulsar el botón del ratón y mantenerlo así

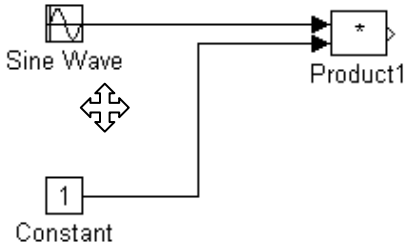


Fig. C.20. Seleccionar segmentos de línea.

3. Arrastrar el puntero a la posición deseada

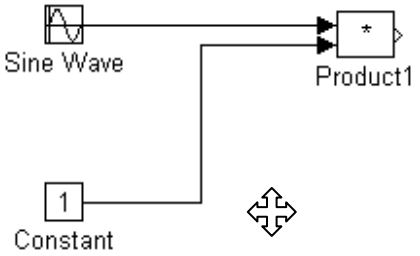


Fig. C.21. Seleccionar segmentos de línea.

4. Soltar el botón del ratón

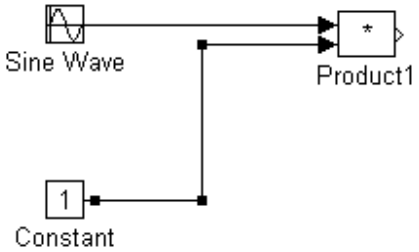


Fig. C.22. Seleccionar segmentos de línea.

No se pueden mover los segmentos conectados directamente a los puertos de los bloques.

C.3.1.2.3.4 Mover vértices.

Para mover un vértice de una línea, se debe posicionar el puntero sobre el vértice, pulsar y manteniendo así el botón del ratón, se arrastra el puntero a la posición deseada y se suelta el botón del ratón. No se puede mover los vértices que están en los extremos de la línea.

La figura que sigue muestra la forma del cursor y el movimiento del vértice cuando se arrastra. Se puede arrastrar el vértice en cualquier dirección

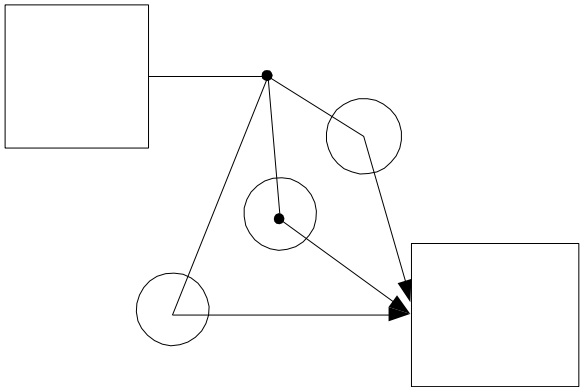


Fig. C.23. Movimiento de bloques.

C.3.1.2.3.5 Dividir una línea en segmentos.

Se puede dividir una línea en dos segmentos (o un segmento de línea en dos segmentos), dejando los extremos de la línea en sus posiciones originales. Simulink crea segmentos de línea y un vértice que los une. Para dividir una línea en segmentos, se deben seguir estos pasos:

1. Colocar el puntero sobre la línea donde se desea el vértice.

Fig. C.24. Dividir líneas en segmentos.

2. Mientras se mantiene pulsada la tecla **Shift**, pulsar y mantener presionado el botón del ratón.

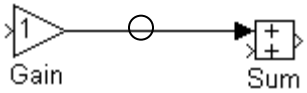


Fig. C.25. Dividir líneas en segmentos.

3. Arrastrar el puntero a la posición deseada.

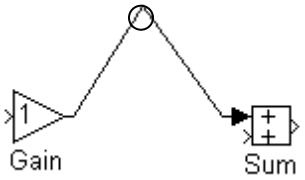


Fig. C.26. Dividir líneas en segmentos.

4. Soltar el botón del ratón y la tecla **Shift**.

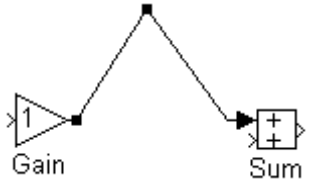


Fig. C.27. Dividir líneas en segmentos.

C.3.1.2.4 Resumen de las acciones del ratón y el teclado.

Acciones de Construcción del modelo	**Windows**
Seleccionar objetos	Botón izquierdo del ratón
Seleccionar más de un objeto	**Shift** + botón izquierdo del ratón
Copiar objetos de otras ventanas	Seleccionar el objeto y arrastrar
Duplicar objeto	**Option** + Botón izquierdo del ratón y arrastrar; o Botón derecho del ratón y arrastrar
Mover objeto	Seleccionar el objeto y arrastrar
Conectar bloques	Botón izquierdo del ratón
Desconectar bloques	**Shift** + arrastrar bloque
Encaminar líneas desde otra línea	**Shift** + dibujar líneas
Dibujar líneas desde otra línea	**Ctrl** + arrastrar línea
Mover segmento de línea	Seleccionar segmento y arrastrar
Mover vértice	Seleccionar vértice y arrastrar
Dividir líneas en segmentos	**Shift** + arrastrar líneas

Tabla C.1. Acciones del teclado.

C.3.1.2.5 Añadir anotaciones de texto al diagrama del modelo.

- ✓ Posicionando el puntero donde se desea colocar el texto, luego pulsar el botón del ratón y por último escribir el texto.
- ✓ El texto debe ser único en el modelo; y puede ser utilizada para fechar el modelo y etiquetar las líneas.

✓ Para modificar las fuentes de texto, se debe seleccionar mediante un cuadro de delimitación y posteriormente, se escoge la fuente del submenú **Fonts**, que está accesible en el menú **Style**.

C.3.1.2.6 Crear subsistemas.

Los subsistemas son usados cuando nuestro modelo se hace complicado, ya que aumenta de tamaño y complejidad. El agrupamiento es útil por una serie de razones:

- Ayuda a reducir el número de bloques visualizados.
- Permite mantener juntos los bloques que están funcionalmente relacionados.
- Hace posible establecer un diagrama de bloques jerárquico.

Se pueden crear los subsistemas de dos maneras:

1. Añadiendo un bloque **Subsystem** al modelo, y luego añadiendo los bloques que contiene.
2. Añadiendo los bloques que constituyen el subsistema y posteriormente agrupándolos.

Para crear un subsistema antes de añadir los bloques que contiene, se debe insertar un bloque Subsystem y crear los bloques que constituyen el subsistema de la forma siguiente:

a. Copiar el bloque **Subsystem** de la biblioteca **Connnections** del modelo.
b. Abrir el bloque **subsystem** (doble clic).
c. En la ventana nueva del bloque **Subsystem**, crear el correspondiente subsistema. Utilizar bloques **Inport** para representar las entradas que vienen desde fuera del subsistema y bloques **Output** para las salidas externas. Por ejemplo, el bloque Sum que se muestra a continuación es el único bloque de un subsistema. El diagrama representa el bloque y sus bloques Inport y Outport asociados:

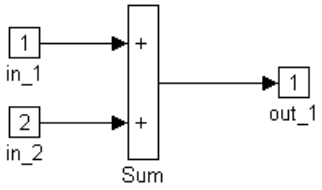


Fig. C.28. Entradas y salidas en los subsistemas.

Una vez creados los bloques que se desea convertir en un subsistema, se debe hacer lo siguiente:

a. Encierre los bloques y líneas de conexión que desea incluir dentro del subsistema mediante un cuadro de delimitación. Por ejemplo, la figura que viene a continuación muestra un modelo que representa un contador. Los bloques Sum y Delay se seleccionan dentro de un cuadro de de delimitación:

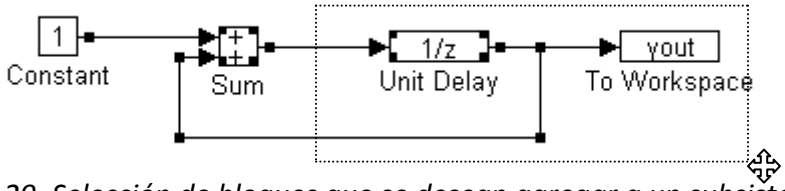


Fig. C.29. Selección de bloques que se desean agregar a un subsistema.

Cuando se suelta el botón del ratón, se seleccionan los bloques y todas las líneas de conexión.

b. Escoger **Options**, luego **Group**. Simulink reemplaza los bloques que están en el grupo por un único bloque **subsystem**. La figura muestra el modelo después de escoger la orden **Group**:

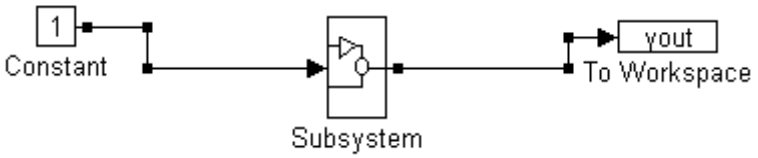


Fig. C.30. Creación de subsitema.

C.3.1.2.7 Modelar ecuaciones.

Uno de los temas más confusos para los nuevos usuarios de Simulink es cómo modelar ecuaciones. Algunos ejemplos ayudarán a comprender cómo modelarlas.

C.3.1.2.7.1 Conversión de grados centígrados en Fahrenheit.

Modele la ecuación que convierte grados centígrados a grados Fahrenheit:

En primer lugar, se consideran los bloques que se necesitan para construir el modelo:

- Un bloque Gain de la biblioteca Linear, para multiplicar la señal de entrada por 9/5
- Un bloque Constant de la biblioteca Sources, para definir una constante de 32
- Un bloque Sum de la biblioteca Linear, para sumar las dos cantidades
- Un bloque Sine Wave de la biblioteca Sources, para introducir la señal
- Un bloque Scope de la biblioteca Sinks, para visualizar la salida

A continuación, reúna los bloques en la ventana de su modelo:

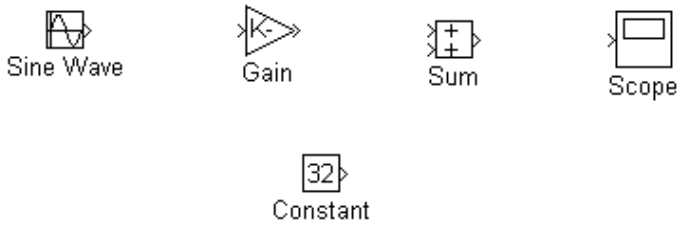


Fig. C.31. Bloques necesarios para convertir grados centígrados a Fahrenheit.

Se asignan valores a los bloques Gain y Constant abriendo cada uno de ellos (mediante una doble pulsación) e introduciendo los valores apropiados. Luego, pulsar el botón OK. Ajustar la amplitud del bloque Sine Wave a 10 para conseguir una mayor variación de temperatura. Ahora, se debe conectar los bloques.

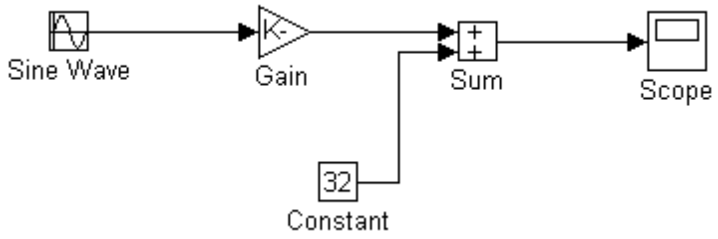


Fig. C.32. Conexión de bloques para convertir grados centígrados a Fahrenheit

El bloque Sine Wave representa la temperatura en grados centígrados. El bloque Gain genera 915(Tc). Ese valor se suma a la constante 32 mediante el bloque Sum. La salida de ese bloque es la temperatura en grados Fahrenheit. Se debe abrir el bloque Scope para visualizar la salida. Se debe fijar la escala horizontal en algún pequeño valor de tiempo, por ejemplo 10 segundos. Luego, fijar la escala vertical de forma que se pueda visualizar todas las salidas - al menos 50. Manténgase abierto el bloque Scope.

Se debe especificar los parámetros de simulación seleccionando **Parameters** en el menú **Simulation**. Especificar un tiempo de parada de 10 segundos y un tamaño de paso máximo de 0.1. Estos valores deberían ejecutar la simulación rápidamente. Ahora, se debe escoger **Start** en el menú **Simulation** para ejecutar la simulación.

C.3.1.2.7.2 Modelar un sistema continuo sencillo

Modelemos la siguiente ecuación diferencial:
x = -2x_ + u

El bloque **Integrator** integra la entrada, dx/dt y produce x. Otros bloques necesitados en este modelo incluyen un bloque Gain y un bloque Sum. Para generar una onda cuadrada, se utiliza un bloque Signal Generator. Otra vez, se visualizará la salida empleando un bloque Scope. Se debe reunir los bloques y definir el valor de la ganancia del bloque Gain.

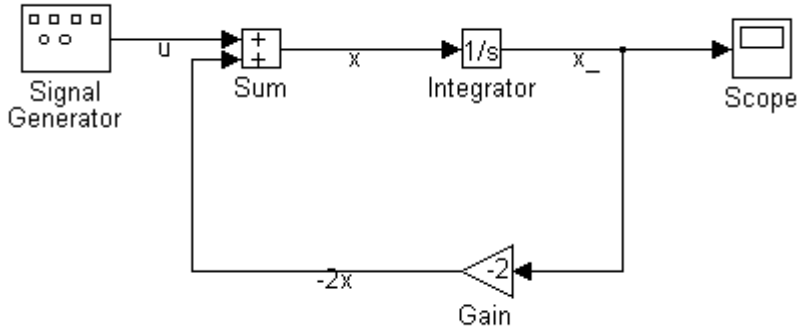


Fig. C.33. Modelo de un sistema continuo.

En este modelo, para invertir la dirección del bloque Gain, se escoge **Flip Horizontal** en el menú **Options**. También, para crear la línea de la salida del bloque Integrator al bloque Gain se debe mantener pulsada la tecla **Ctrl** mientras se dibuja la línea. Posteriormente, conectar todos los bloques.

Un concepto importante en este modelo es el bucle que incluye al bloque Sum, al bloque Integrator y al bloque Gain. En esta ecuación, x_ es la salida del bloque Integrator así como la entrada a los bloques que calculan x, sobre el cual se basa. Esta relación se implementa utilizando un bucle.

El bloque Scope visualiza x_ en cada paso de tiempo. Para una simulación que dura 10 segundos y que el rango vertical del bloque Scope es de 1, la salida que resulta se muestra en la figura.

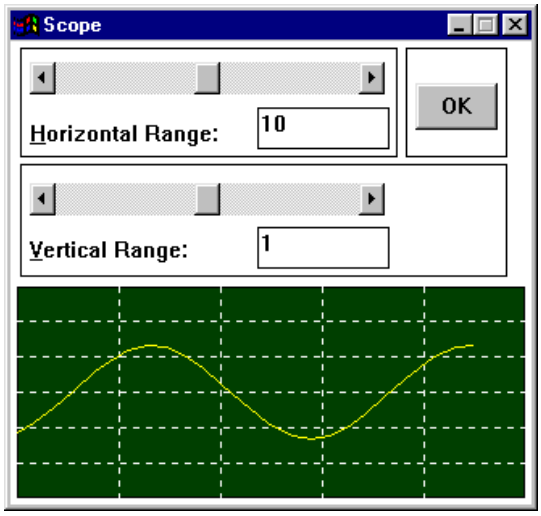


Fig. C.34. Respuesta del bloque Scope.

C.3.1.3 Un ejercicio de construcción de un modelo.

Este ejemplo muestra cómo construir un modelo utilizando muchas de las órdenes y acciones que se utilizarán para desarrollar nuestros propios modelos.

El modelo genera una onda sinusoidal utilizando un bloque Signal Generator. Modifica una copia de la forma de onda pasándola a través de un bloque Gain. Ambas señales, la original y la modificada, se combinan en una señal vectorial empleando un bloque Mux. Esa señal se visualiza utilizando un bloque Scope y enviándola a una variable del espacio de trabajo. El diagrama de bloques del modelo tiene el siguiente aspecto:

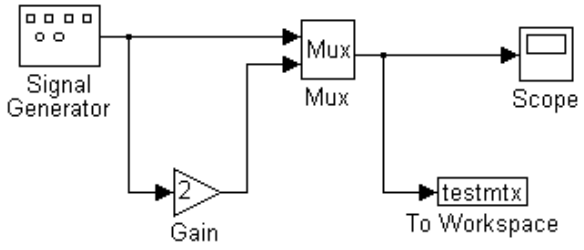


Fig. C.35. Ejercicio.

Los bloques utilizados en este modelo, se obtienen de las bibliotecas siguientes:

- Biblioteca Sources (para el bloque Signal Generator)
- Biblioteca Linear (para el bloque Gain)
- Biblioteca Connections (para el bloque Mux)
- Biblioteca Sinks (para los bloques Scope y To Workspace)

Primero, generar una nueva ventana de modelo seleccionando **New** en el menú **File**.

Segundo, Abrir la biblioteca Sources para copiar el bloque Signal Generator.

Para copiar un bloque desde la biblioteca de bloques, se arrastra dentro de la ventana del modelo. Para hacer esto, se posiciona el cursor sobre el bloque Signal Generator, a continuación se presiona el botón del ratón, manteniéndolo en esa posición. El cursor cambiará.

Después, se arrastra el bloque dentro de la ventana del modelo. Cuando se mueve el bloque, se puede observar que el recuadro del bloque y su nombre se mueven con el puntero.

Cuando el puntero se encuentre dentro de la ventana del modelo, se debe soltar el botón del ratón. Una copia del bloque Signal Generator se encontrará en la ventana del modelo.

De la misma forma, se deben copiar el resto de los bloques dentro de la ventana del modelo. Se puede mover un bloque, desde un lugar a otro dentro de la ventana del modelo, utilizando la misma técnica de arrastre que se empleó para copiar el bloque.

Cabe destacar que el bloque Mux tiene tres puertos de entradas pero sólo dos señales de entrada. Para ajustar el número de puertos de entrada, se debe abrir el bloque Mux dando un doble clic encima de él. SIMULINK visualiza su cuadro de diálogo. Cambia el valor del parámetro **Number of inputs** a 2, a continuación se pulsa el botón OK. SIMULINK ajusta el número de puertos de entrada.

Con todos los bloques ya copiados en la ventana del modelo, el modelo debería tener un aspecto parecido al siguiente:

Ahora se deben conectar los bloques. Si se examinan los íconos de los bloques, se verá un signo en forma de ángulo a la derecha del bloque Signal Generator y dos a la izquierda del bloque Mux. El símbolo > apuntando hacia fuera de un bloque es un puerto de salida; si el símbolo apunta hacia el bloque es un puerto de entrada. Cuando los bloques se conectan, los símbolos del puerto desparecen.

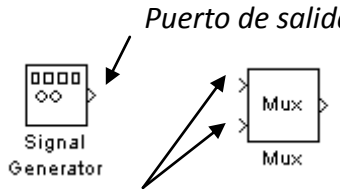


Fig. C.36. Puertos de entrada y salida de los bloques.

A continuación se conecta el bloque Signal Generator al bloque Mux. Posicionando el puntero sobre el puerto de salida en el lado derecho del bloque Signal Generator.

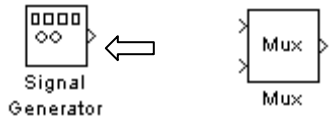


Fig. C.37. Conexión de una salida con una entrada.

Luego se presiona y se mantiene pulsado el botón del ratón. Se observa que el botón cambia a una forma de una cruz.

Fig. C.38. Conexión de una salida con una entrada.

Manteniendo pulsado el botón del ratón, se debe mover el cursor al puerto de entrada superior del bloque Mux o sobre el propio bloque Mux. Obsérvese que el cursor mantiene su forma de cruz y que una línea conecta el bloque Signal Generator con el puerto de entrada superior del bloque Mux.

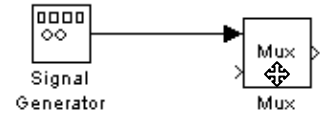


Fig. C.39. Conexión de una salida con una entrada.

Luego, se libera el botón del ratón. Los bloques se conectan.

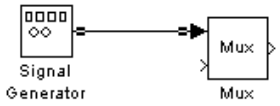


Fig. C.40. Conexión de una salida con una entrada.

En el modelo que hay al comienzo se observa que la mayoría de las líneas conectan puertos de salida de bloques a puertos de entrada de otros bloques. Sin embargo, dos líneas conectan líneas a puertos de entrada de otros bloques. Estas líneas conectan la salida de Signal Generator al bloque Gain, y la salida Mux al bloque To Workspace y llevan la misma señal que las líneas desde las cuales se originan.
Dibujar esta clase de línea es ligeramente diferente a dibujar la línea que se acaba de dibujar. Para unir una conexión a una línea ya existente, se deben seguir estos pasos. En primer lugar, posicionar el cursor sobre la línea que hay entre los bloques Signal Generator y Mux.

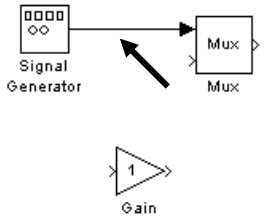


Fig. C.41. Conexión de bloques en paralelo.

Mientras se mantiene pulsado el botón del ratón, se debe presionar y mantener también pulsada la tecla ctrl. Luego, se arrastra el cursor al puerto de entrada del bloque Gain o sobre el propio bloque Gain.

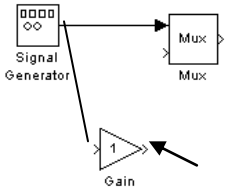


Fig. C.42. Conexión de bloques en paralelo.

Ahora, se libera el botón del ratón. SIMULINK dibuja una línea entre el punto de comienzo y el puerto de entrada del bloque gain.

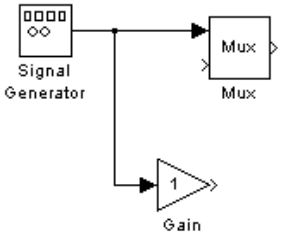


Fig. C.43. Conexión de bloques en paralelo.

De forma análoga se debe dibujar una línea similar entre la línea de salida del bloque Mux y el bloque To Workspace. Se debe acabar de realizar las conexiones. Cuando se a ha

finalizado de conectar los bloques, es necesario ajustar algunos de sus parámetros. En primer lugar, se debe abrir el bloque Gain y cambie el parámetro **Gain** a 2.

A continuación, se debe abrir el bloque To Workspace y modificar el parámetro **Variable name** a testmtx. Éste es el nombre de la variable del espacio de trabajo que mantendrá la salida de la simulación.

La salida por defecto del bloque signal generator es una onda sinusoidal con una amplitud de 1. Como esto es aceptable para este ejercicio, no hay necesidad de modificar ningún parámetro para este bloque.
Se debe ejecute la simulación durante 10 segundos. Primero, se debe ajustar los parámetros de simulación seleccionando **Parameters** del menú **Simulation**. En el cuadro de diálogo que aparece, ajuste el **Stop Time** a 10 y cambie el **Maximum Step Size** a 0.1.

A continuación se debe abrir el bloque Scope para visualizar la salida de la simulación. Antes de que comience la simulación, se debe ajustar los parámetros de forma que se pueda visualizar la simulación completa. Se modifican los parámetros **Horizontal Range** (tiempo) a 10 (segundos) y **Vertical Range** (para esta simulación, corresponde a la amplitud de la onda sinusoidal) a 3.

Manteniendo abierta la ventana del bloque Scope se debe ejecutar la simulación. Se escoja **Start** en el menú Simulation y se observan las trazas del vector de entrada al bloque Scope.

Para guardar este modelo, se selecciona **Save** en el menú **File** y se especifica el nombre y la localización del archivo-M que describe el modelo. Para finalizar SIMULINK y MATHLAB se debe escoja **Exit MATHLAB** en el menú **File**.

C.3.1.4 Guardar el modelo

- ✓ Se escoge la orden **save** o **save as** en el menú **File**. Simulink guarda el modelo generando un ***archivo.M*** que contiene las ordenes de MATLAB necesarias para recrear el modelo.
- ✓ Si se está guardando por primera vez, se debe asignar un nombre y una localización al archivo.M, y luego pulsar OK para guardar.

C.3.2 SIMULACIÓN Y ANÁLISIS

C.3.2.1 Cómo trabaja Simulink.

Cada bloque mediante un modelo de **Simulik** tiene estas características generales: Un conjunto de entradas *u*, un conjunto de salidas *y* y un conjunto de estados *x.*

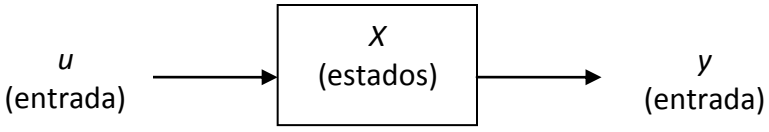


Fig. C.44. Características de los bloques.

El vector de estado puede constar de estados continuos, estados discretos o una combinación de ambos.
La simulación consta de dos fases: inicialización y simulación. Algunas acciones tienen lugar durante la fase de inicialización.

En primer lugar, los parámetros del bloque se pasan a MATLAB para su evolución. Los valores numéricos resultantes se utilizan como los parámetros actuales de bloque.

En segundo lugar, la jerarquía del modelo se reduce a su nivel inferior. Es decir, cada subsistema se sustituye por los bloques que contiene.

En tercer lugar, los bloques se disponen en el orden en que se necesita que se actualicen. El algoritmo de ordenación construye una lista tal que cualquier bloque con alimentación directa no se actualiza hasta que se calculan los bloques que excitan sus entradas. Es durante este paso cuando se detectan los lazos algebraicos.

Finalmente, se comprueban las conexiones entre bloques para asegurar que la longitud del vector de salida de cada bloque coincide con la entrada que esperan los bloques a los que se conecta.

Ahora ya la simulación está ya preparada para poderse ejecutar.

C.3.2.1.1 Lazos algebraicos.

Lazos algebraicos o implícitos ocurren cuando dos o mas bloques con alimentación directa de sus entradas forman un lazo de realimentación. Cuando esto ocurre, SIMULINK debe efectuar iteraciones en cada paso para determinar si existe una solución a este problema.

Ejemplos de bloques con alimentación directa son:
- Bloques Gain
- La mayoría de los boques no lineales
- Bloques Transfer Fen, cuando el numerador y el denominador tienen el mismo orden
- Bloques Zero-Pole, cuando hay tantos ceros como polos
- Bloque State-Space, cuando hay una matriz D distinta de cero

SIMULINK informa de un error cuando no puede resolver un lazo algebraico en 200 iteraciones de una rutina de Newton-Raphson.

Para romper lazos algebraicos en lugar de permitir que SIMULINK que los resuelva de forma iterativa, inserte un bloque Memory entre dos bloque cualesquiera incluidos en el lazo.

C.3.2.2 Simulación

Se puede ejecutar una simulación seleccionando órdenes desde los menús de SIMULINK o introduciéndolas desde la ventana de órdenes de MATLAB.

- Seleccionar órdenes desde los menús es rápido de aprender. Se puede visualizar gráficamente la conducta del sistema con bloque tipo scope.
- La introducción de órdenes de simulación y análisis desde la ventana de órdenes de MATLAB o desde los propios programas le permite visualizar los efectos de cambiar los bloques o parámetros de integración.

C.3.2.2.1 Parámetros de Simulación.

Antes de que se ejecute una simulación, se debe especificar los parámetros de simulación y elegir el método de integración. Los parámetros de simulación incluyen:

- Tiempo de comienzo y finalización
- Tamaño del paso mínimo
- Tamaño del paso máximo
- Tolerancia o error relativo
- Variables de retorno

Cuando se ejecuta la simulación utilizando órdenes del menú, se debe asignar los parámetros de simulación seleccionando **Parameters** en el menú **Simulation,** después selecciona un método de integración y rellenar los parámetros en el cuadro de dialogo de **Control Panel** (Windows).

Tiempos de comienzo y terminación.

Los parámetros **Star Time** y **Stop Time** especifican los valores de *t* en los que la simulación comienza y termina. El tiempo de simulación y el tiempo del reloj de pared no son iguales.

La cantidad de tiempo que toma ejecutar una simulación depende de muchos factores, entre los que se incluyen la complejidad del modelo, los tamaños de paso mínimo y máximo y la velocidad del reloj del computador.

Tamaño de paso mínimo

El parámetro Minimun Step Size es el tamaño de paso utilizado al comienzo de una simulación. Los integradores no emplean un tamaño de paso por debajo de este valor cuando generan un punto de salida a menos que el sistema contenga bloques discretos con períodos de muestreo más pequeños que el tamaño de paso mínimo. Un punto de salida es un punto generado en un bloque tipo sumidero (sink), tales como los bloques Scope o To Workspace o devuelto en una trayectoria de estado o de salida. Un punto de salida se genera después que se han completado las iteraciones del método de integración.

Cuando hay discontinuidades en el sistema, si se asigna un valor muy pequeño al tamaño de paso mínimo puede originar que se genere un número enorme de puntos, lo que podría malgastar memoria y recursos.

Para los métodos de integración de adams y gear, el tamaño de paso mínimo no afecta la precisión de la solución, pero si el número de puntos de salida generados.

Tamaño de paso máximo

Se debe asignar el tamaño de paso máximo bastante pequeño para que la simulación no deje de tomar en cuenta detalles importantes. Un tamaño de paso relativamente grande puede originar que algunos modelos se hagan inestables.

Algunas veces una simulación produce resultados que son precisos pero que no son buenos para generar gráficas suaves. En ese caso, puede ser necesario limitar el tamaño del paso máximo de forma que los resultados gráficos tengan una apariencia regular.

Tolerancia o error relativo

El parámetro **Tolerance** (Windows) controla el error relativo aceptable de la integración en cada paso. En general, este parámetro se debería fijar en el rango que va desde 0.1 a 1e-6. Cuanto más pequeño sea el valor, más pasos de integración hay que utilizar, lo que origina una simulación más precisa. No obstante, cuando la tolerancia se fija en un valor muy pequeño (1e-10) puede producir un paso tan pequeño que el error de redondeo aumenta muy significativamente.

Variables de retorno

Se pueden especificar nombres de variables en este campo para conseguir que SIMULINK escriba valores para el tiempo y las trayectorias de estado y de salida en el espacio de trabajos. La primera variable almacena el tiempo, la segunda el estado y la tercera la salida.

C.3.2.2.2 Simulación desde el menú.

La ejecución de una simulación desde el menú permite efectuar ciertas operaciones de forma iterativa durante una simulación:

- Cambiar los parámetros de un bloque, a condición de que no origine un cambio en el número de estados, entradas o salidas para ese bloque.
- Cambiar cualquier parámetro de simulación excepto las variables de retorno y el comienzo.
- Cambiar el método de simulación.
- Simular al mismo tiempo otro sistema.
- Pulsar sobre una línea para ver la señal que transporta esa línea sobre un bloque Scope de tipo flotante (no conectado).

Cambios en la estructura del modelo durante una simulación, tales como añadir o suprimir líneas o bloques, originan que se pare la simulación. Se debe seleccionar otra vez **Start** para ver el resultado del cambio.

Se asignan los parámetros de simulación escogiendo **Parameters** en el menú **Simulation**.

Simulink visualiza el cuadro de diálogo **Control Panel** (Windows) que se muestra a continuación.

Estos cuadros de diálogo permiten escoger el método de integración y definir los parámetros de simulación.

Cuando se han definido los parámetros de simulación, está ya todo preparado para iniciar la simulación. Se ejecuta una simulación seleccionando **Star** del menú de **Simulation**.

Se puede suspender una simulación que está en ejecución escogiendo **Pause** en el menú de **Simulation** y proseguir con una simulación que ha quedado en suspenso seleccionando **Continue**.

C.3.2.2.3 Visualización de las trayectorias de salida

Las trayectorias de salida de SIMULINK se pueden representar gráficamente utilizando uno de los tres métodos siguientes:

- Bloques Scope
- Variables de retorno y las órdenes de representación gráfica de MATLAB
- Bloques To Workspace y las órdenes de representación gráfica de MATLAB

Utilización del bloque Scope

El bloque Scope se puede utilizar para visualizar trayectorias de salida mientras la simulación está en ejecución. El modelo sencillo que se muestra a continuación es un ejemplo del uso del bloque Scope.

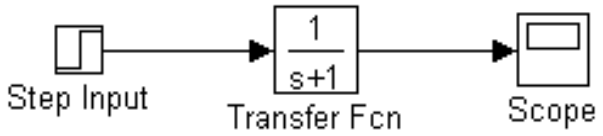


Fig. C.45. Ejemplo de uso del bloque Scope.

La visualización en el bloque Scope es bastante básica; muestra la trayectoria de salida sin ninguna anotación. Los Bloques Graph proporcionan tipos de líneas y colores pero se ejecutan más lentamente que el bloque Scope.

Utilización de las variables de retorno

Devolviendo el tiempo y las historias de las salidas a MATLAB, puede utilizar las órdenes de representación gráfica de MATLAB para visualizar y poner anotaciones a dichas trayectorias.

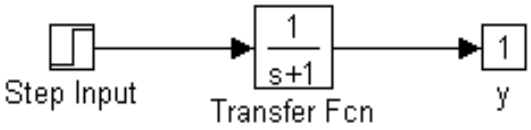


Fig. C.46. Integracion con Matlab.

El bloque etiquetado '*y'* es un bloque Outport de la biblioteca Connections. La trayectoria de salida 'y' se devuelve por la función de integración. Por ejemplo, llamado al sistema tfout e invocando a la simulación desde la línea de orden:

[t, x, y] = linsim ('tfout', 2);

Produce historias temporales. Podría también ejecutarse esta simulación desde el menú **Simulation** especificando [t, x, y] como el parámetro **Return Variables**. Puede entonces representar gráficamente estos resultados utilizando:

plot (t, y)

Utilización del bloque Workspace

El bloque To Workspace se puede utilizar para devolver trayectorias de salida al espacio de trabajo de MATLAB. El modelo que se muestra a continuación ilustra este método:

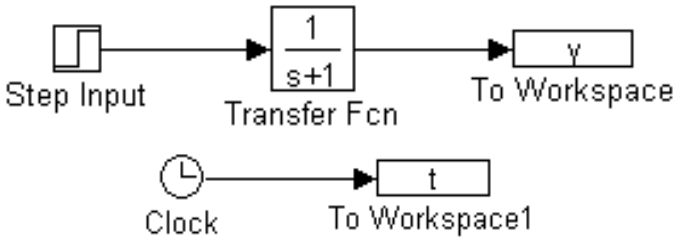


Fig. C.47. Uso del bloque To Workspace.

Las variables **y** y **t** apareen en el espacio de trabajo cuando se completa la simulación.

El vector de tiempos se almacena alimentando un bloque Clock en el bloque To Workspace. El vector de tiempos se puede obtener también introduciendo t en el campo **Return Variables** para simulaciones conducidas por menú o retornándola desde la función de integración:

t = linsim ('tfout', 2);

El bloque To Workspace puede aceptar una entrada vectorial donde cada trayectoria de los elementos de entrada se almacena como un vector columna en la variable del espacio de trabajo resultante.

C.3.2.2.4 Métodos de integración

La simulación de modelos Simulink lleva consigo la integración numérica de conjuntos de ecuaciones diferenciales ordinarias. Simulink proporciona una serie de métodos de integración para la simulación de tales ecuaciones:

linsim	Método que extrae la dinámica lineal
rk23	Método de Runge-Kutta de tercer orden
rk45	Método de Runge-Kutta de quinto orden
gear	Método predictor-corrector de Gear para sistemas stiff
adams	Método predictor-corrector de Adams
euler	Método de Euler

Tabla C.2. Metodos de integración en Simulink.

Un sistema stiff es aquel en el que coexisten dinámicas lentas y rápidas y estas últimas alcanzan su estado estacionario.

Elección de un método

La elección del método apropiado y la selección cuidadosa de los parámetros de simulación son consideraciones importantes para obtener resultados rápidos y precisos.
El comportamiento de la simulación en términos de velocidad y precisión varía para los

diferentes modelos y condiciones.

- **Linsim**

  Linsim utiliza un método que extrae la dinámica lineal de un sistema, dejando solamente la dinámica no lineal del sistema para ser simulado. Este método funciona muy bien cuando el sistema que se desea simular es relativamente lineal. Los modelos lineales se componen de bloques Transfer Fcn, State-Space, Zero-Pole, Sum y Gain. El método puede tomar tamaños de pasos muy grandes para tales sistemas. Por lo tanto, para obtener puntos de salida razonablemente espaciados, es necesario limitar el tamaño de paso máximo.

  Linsim es particularmente bueno comparado con otros métodos cuando los bloques lineales tienen ambas dinámicas, rápidas y lentas.

- **rk45, rk23**

  Los métodos de Rungr-Kutta, rk23 y rk45 son buenos algoritmos de propósito general que trabajan bien para un abanico amplio de problemas. Normalmente funcionan mejor que los otros métodos cuando el sistema es fuertemente no lineal y/o presenta discontinuidades. Sin embargo, no operan bien cuando el sistema tiene ambas dinámicas, rápidas y lentas. Los métodos rk23 y rk45 dan buen resultado para sistemas híbridos que contienen elementos de tiempo continuo y tiempo discreto.
  Aunque rk45 es generalmente más rápido y preciso que rk23, produce menos puntos de salida; por lo tanto, rk23 puede ser la elección preferida para gráficas más regulares.

- **adams, gear**

  Adams y gear son métodos del tipo predictor-corrector que funcionan bien en problemas donde las trayectorias de estados son regulares. El método de gear está diseñado para sistemas stiff. Es menos eficiente que otros métodos para sistemas no stiff y no da buenos resultados cuando el sistema tiene discontinuidades.

  Se utiliza gear para sistemas que sean regulares y no lineales.

  Se utiliza adams para sistemas que son regulares y no lineales pero que no tienen constantes de tiempo que varían en un amplio rango.

- **euler**

  euler es una implementación del método de Euler explícito, un método de primer orden que, en general, utilizará más puntos de tiempo que los métodos de orden superior. Este método no produce resultados tan precisos como otros métodos.

Debería evitar utilizarlo a menos que esté investigando la ejecución de su modelo en tiempo real.

Comparación de los métodos: Un ejemplo

El comportamiento de la simulación depende de la elección de los métodos y de los valores de los parámetros. Este ejemplo compara los métodos de simulación utilizando la ecuación de Van der Pol, un modelo no lineal simple de segundo orden:

$\dot{x} + (x-1)\dot{x} + x = 0$

Esta ecuación se puede describir como un conjunto de ecuaciones diferenciales de primer orden:

$$\dot{x}_1 = x_1(1-x_2^2) - x_2$$

$$\dot{x}_2 = x_1$$

Estas ecuaciones se representan como el sistema vdp, que se proporciona como una demo con Simulink.

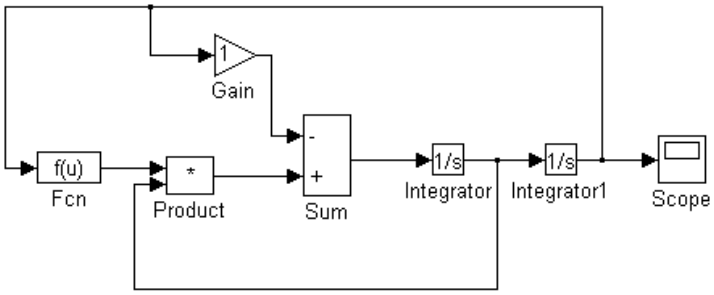


Fig. C.48. Modelado de una ecuación.

En este ejemplo, comparamos los métodos de integración utilizando un valor de 1e-3 para el parámetro de tolerancia. Los parámetros de integración se definen como:

```
tol = 1e-3;
minstep =1e-5;
maxstep = 10;
options = [tol, minstep, maxstep];
x0 = [1;1];
```

Comparando los métodos de integración linsim, rk23 y rk45:

```
[tls, xls] = linsim ('vdp', 10, x0, options);
[tr23, xr23] = rk23 ('vdp', 10, x0, options);
[tr45, xr45] = rk45 ('vdp', 10, x0, options);
plot (tls, xls, tr23, xr23, 'o', tr45, xr45, '+')
```

C.3.3 LA BIBLIOTECA DE BLOQUES DE SIMULINK

Simulink organiza sus bloques en bibliotecas de bloques de acuerdo con su conducta. La ventana Simulink visualiza los nombres de las bibliotecas y de los iconos.

La biblioteca Sources (Fuentes) incluye bloques que originan señales. La tabla que sigue a continuación describe los bloques de la biblioteca Sources.

Nombre del bloque	Objetivo
Band-Limited White Noise	Introduce ruido blanco en un sistema continuo
Chirp Signal	Genera una onda sinusoidal de frecuencia creciente
Clock	Visualiza y proporciona el tiempo de simulación
Constant	Genera un valor constante
Digital Clock	Genera tiempo de simulación en el intervalo de muestreo especificado
From File	Lee datos de un archivo
From Workspace	Lee datos de una matriz definida en el espacio de trabajo
Pulse Generator	Genera pulsos a intervalos regulares
Random Number	Genera números aleatorios distribuidos normalmente
Repeating Sequence	Genera una señal arbitraria repetible regularmente
Signal Generator	Genera diferentes formas de ondas
Sine Wave	Genera una onda sinusoidal
Step Input	Genera una función en salto

Tabla C.3. Bloques de la biblioteca Sources.

La biblioteca Sinks (Sumidero) incluye bloques que visualizan o escriben su salida. La tabla que se muestra a continuación describe los bloques de la biblioteca Sinks.

Nombre del bloque	Objetivo
Auto-Scale Graph Scope	Visualiza señales en ventanas de figuras de MATLAB autoescaladas
Graph Scope	Visualiza señales utilizando la ventana de figuras de MATLAB
Hit Crossing	Aumenta el número de pasos de simulación en tomo a un valor especificado.
Scope	Visualiza señales durante la simulación
Stop Simulation	Para la simulación cuando la entrada es distinta de cero
To File	Escribe datos en un archivo
To Workspace	Escribe datos en una matriz en el espacio de trabajo
XY Graph Scope	Visualiza gráficas X-Y de señales en la ventana de figuras de MATLAB

Tabla C.4. Bloques de la biblioteca Sinks.

La biblioteca Discrete (Discretos) contiene bloques que describen componentes de tiempo-discreto. La tabla que se muestra a continuación describe los bloques de la biblioteca Discrete.

Nombre del bloque	Objetivo
Discrete-Time Integrator	Realiza la integración en tiempo discreto de una señal
Discrete-Time Limited Integrator	Realiza la integración en tiempo discreto de una señal con límites.
Discrete State-Space	Implementa un sistema discreto en el espacio de estados
Discrete Transfer Fcn	Implementa una función de transferencia discreta
Discrete Zero-Pole	Función de transferencia discreta en términos de polos y ceros
Filter	Implementa filtros IIR y FIR
First-Order Hold	Implementa un muestreador-retenedor de orden uno
Unit Delay	Retarda una señal en un período de muestreo
Zero-Order Hold	Retenedor de orden cero de un período de muestreo

Tabla C.5. Bloques de la biblioteca Discrete.

La biblioteca Linear (Lineal) contiene bloques que describen funciones lineales estándar. La tabla que se muestra a continuación describe los bloques que contiene la biblioteca Linear.

Nombre del bloque	Objetivo
Derivative	Genera la derivada respecto al tiempo de la entrada
Gain	Multiplica la entrada al bloque
Inner Product	Genera el producto escalar
Integrator	Integra una señal
Matrix Gain	Multiplica la entrada por una matriz
Slider Gain	Varía una ganancia escalar utilizando una corredera
State-Space	Implementa un sistema lineal en el espacio de estados
Sum	Genera la suma de las entradas
Transfer Fcn	Implementa una función de transferencia lineal
Zero-Pole	Función de transferencia especificada en términos de polos y ceros

Tabla C.6. Bloques de la biblioteca Linear.

La biblioteca Nonlinear (No-lineal) contiene bloques que describen funciones no lineales estándar. La tabla que se muestra a continuación describe los bloques de la biblioteca Nonlinear.

Nombre del bloque	Objetivo
Abs	Genera el valor absoluto de la entrada
Backlash	Modela la conducta de un sistema con huelgo
Combinatorial Logic	Implementa una tabla de verdad
Coulombic Friction	Discontinuidad en cero con cualquier valor de ganacia lineal
Dead Zone	Proporciona una región de salida cero
Fcn	Aplica una expresión especificada a la entrada
Limited Integrator	Integra dentro de límites especificados
Logical Operator	Realiza operaciones lógicas especificadas sobre las entradas
Look-Up Table	Realiza una transformación lineal a tramos de la entrada
MATLAB Fcn	Aplica una función de MATLAB a la entrada
Memory	Saca la entrada al bloque en el paso de integración previo
Product	Multiplica las entradas
Quantizer	Discretiza la entrada en un intervalo especificado
Rate Limiter	Limita la velocidad de cambio de una señal
Relational Operation	Realiza las operaciones relacionales especificadas sobre la entrada
Relay	Conmuta la salida entre dos valores
Reset Integrator	Reinicializa los estados del integrador durante la simulación
Saturation	Limita el valor de una señal
Sign	Devuelve el signo de la entrada
Switch	Conmuta entre dos entradas
Transpon Delay	Retarda la entrada en una cantidad dada de tiempo
2-D Look-Up Table	Realiza una transformación lineal a tramos de dos entradas
Variable Transpon Delay	Retarda la entrada una cantidad variable de tiempo

Tabla C.7. Bloques de la biblioteca Nonlinear.

La biblioteca Connections (Conexiones) contiene bloques que permiten multiplexación y de multiplexación, implementa Entradas/Salidas externas y crea subsistemas. La tabla que se muestra a continuación describe los bloques de la biblioteca Connections.

Nombre del bloque	Objetivo
Demux	Separa una señal vectorial en sus señales de salida
Inport	Proporciona un enlace a una entrada externa para linealización
Mux	Combina algunas líneas de entrada en una línea vectorial
Outport	Proporciona un enlace a una salida externa para linealización
Subsystem	Representa un sistema dentro de otro sistema

Tabla C.8. Bloques de la biblioteca Connections.

La biblioteca Extras contiene programas de demostración y algunas bibliotecas adicionales: la biblioteca Conversions, la biblioteca de Flip-Flops, la biblioteca PID Controllers, la biblioteca Analyzers, la bilioteca Filter y una biblioteca de bloques que se utilizan con la linealización.

ANEXO D. CÓDIGO EJECUTABLE EN TIEMPO REAL DEL EJEMPLO SCADA.

Código Principal

```
//=============================================================================//
//Titulo: SCADA par mini central hidroelectrica
//Proyecto: Diseño de una estacion de laboratorio de automatizacion y control
//          Automatico para la escuela de ingenieria electrica de la UES
//Tema: Ejemplo de diseño y desarrollo de un sistema SCADA en el programa
//          Winlog.
//Hecho por: Carlos Palacios, Sixto Edwin, Mario Alejandro
//
//=============================================================================//

//Funcion principal, esta se inicia desde el momento de correr
//el SCADA
Function void main()
#Startup

//*****************************************************************************
//Declaracion de las variables internas del programa
//*****************************************************************************
    real Caudal=0;  //Se almacena el valor calculado del caudal
    real Potencia=0; //Se almacena el valor calculado de la potencia
    int  retraso=UnsignedGetTickCount()/1000; //se almacena el numero
            //de milisegundos desde que inicio el dia
    real  VoltajeTrafo=0;   //Se almacena el valor calculado del
            //voltaje del primario del transformador
    real  CorrienteTrafo=0; //se almacena el valor calculado de la
            //corriente del primario del transformador
    real  PotenciaEntregada=0; //se almacena el valor de la potencia
            //entregada a la red

   //*****************************
   // Monitoreo de sensores
   //*****************************
   while (WindowIsOpen())

//Calculo del voltaje y corriente en el primario del trafo y la potencia
//entregada a la RED y se muestran en el SCADA
        VoltajeTrafo=GetNumGateValue("Voltaje",1)*48;
        CorrienteTrafo=GetNumGateValue("Corriente",1)/48;
        PotenciaEntregada=(VoltajeTrafo*CorrienteTrafo*Cos(0.26))/1000;
        SetNumGateValue("VoltajeTrafo",1,VoltajeTrafo);
        SetNumGateValue("CorrienteTrafo",1,CorrienteTrafo);
        SetNumGateValue("PotenciaEntregada",1,PotenciaEntregada);

//Cada 5 segundos se resetean las variables que activan las alarmas ya que
//la gran mayoria son de querer hacer acciones no permitidas.
        if (retraso == (UnsignedGetTickCount()/1000)-5) then
            retraso=(UnsignedGetTickCount()/1000);
            if (GetDigGateValue("Alarma1",1)) then
            //Se resetea la alarma de tubo lleno no abrir valvula
                SetDigGateValue("Alarma1",1,0);
            end
            if (GetDigGateValue("Alarma2",1)) then
            //Se resetea la alarma de compuerta ya esta abierta
                SetDigGateValue("Alarma2",1,0);
            end
            if (GetDigGateValue("Alarma3",1)) then
            //se resetea la alrma de compuerta ya esta cerrada
                SetDigGateValue("Alarma3",1,0);
            end
            if (GetDigGateValue("Alarma4",1)) then
            //se resetea la alarma de presion en los cojinetes ya
            //es la adecuada
                SetDigGateValue("Alarma4",1,0);
            end
            if (GetDigGateValue("Alarma5",1)) then
            //se resetea la alarma de no sicronizado no conexion
                SetDigGateValue("Alarma5",1,0);
            end
```

```
69              if (GetDigGateValue("Alarma7",1)) then
70              //se resetea la alarma tuberia vacia no abrir compuerta
71                  SetDigGateValue("Alarma7",1,0);
72              end
73          end
74
75  //Poner la imagen de la tubeia llenandose si la valvula esta activa
76          if (GetDigGateValue("ValPreAct",1) && GetNumGateValue("ImgTubo",1)!=1) then
77              SetNumGateValue("ImgTubo",1,1);
78          end
79  //Monitoreo de tuberia llena o basia
80          if (GetDigGateValue("TuboLleno",1)) then
81              TuberiaLlena(); //si tuberia esta llena
82          else
83              TuberiaVacia(); //si tuberia esta basia
84          end
85  //si la compuerta se esta abriendo se debe mostrar la imagen de compuerta abriendose
86          if (GetDigGateValue("AbrCompServ",1) && GetNumGateValue("ImgCompSer",1)==0) then
87              SetNumGateValue("ImgcompSer",1,1);
88          end
89  //si la compuerta se esta cerrando se debe mostrar la imagen de compuerta cerrandose
90          if (GetDigGateValue("CrrCompServ",1) && GetNumGateValue("ImgCompSer",1)==3) then
91              SetNumGateValue("ImgcompSer",1,4);
92          end
93  //si se detecta que el sensor de compuerta de servicio cerrada esta activo se debe
94  //mostrar la imagen de la compuerta cerrada.
95          if ((GetDigGateValue("CompServCrd",1) && GetDigGateValue("CrrCompServ",1)) ||
96              (GetDigGateValue("AbrCompServ",1) && GetDigGateValue("CompServAbt",1))) then
97              Detener_compuerta_servicio();
98          end
99  //Si la compuerta de servicio esta abierta y los alabes estan abiertos entoces mostrar la
100 //imagen de el agua corriendo en la tuberia forzada
101         if (GetNumGateValue("PulsosAlabes",1)>0 && GetDigGateValue("CompServCrd",1)==0) then
102             SetNumGateValue("ImgTubo",1,4);
103         else if (GetNumGateValue("ImgTubo",1)==4) then //sino mostrar la imagen de tuberia
104                 SetNumGateValue("ImgTubo",1,3); //llena
105             end
106         end
107
108 //Cuando se detecta que el sensor de presion esta activo y la bomba de levantamiento esta activa
109 //se debe apagar la bonba
110         if (GetDigGateValue("BombaLevantamiento",1) && GetDigGateValue("Sensorpresion",1)) then
111             BombaLevantamiento();
112         end
113
114 //Mientras los alabes esten en movimiento se debe de monitorear el esta de las imagenes en la
115 //turbina
116         if (GetDigGateValue("AbrAlabes",1) || GetDigGateValue("CrrAlabes",1) ||
117             GetNumGateValue("EstadoTurbina",1)==1 || GetNumGateValue("EstadoTurbina",1)==3 ||
118             GetNumGateValue("EstadoTurbina",1)==5 || GetNumGateValue("EstadoTurbina",1)==7)then
119             Alabes();
120         end
123 //que se ha simplificado a Q=5.0*[% de apertura de los alabes] y P=839*[% de apertura de los
124 //alabes] respectivamente
125         Caudal=5.0*(GetNumGateValue("PulsosAlabes",1)/100);
126         Potencia=839*(GetNumGateValue("PulsosAlabes",1)/100);
127         SetNumGateValue("Caudal",1,Caudal);
128
129 //si la apertura de los alabes es menor que 10, la potencia es 0 porque la turbina no se mueve
130         if(GetNumGateValue("PulsosAlabes",1)>=10)then
131             SetNumGateValue("Potencia_Mecanica",1,Potencia);
132         else
133             SetNumGateValue("Potencia_Mecanica",1,0.0);
134         end
135
136 //conectar el equipo a la red si la exitacion esta activa y la turbina esta en movimiento
137         if (GetDigGateValue("ConectarRED",1) && (GetDigGateValue("Exitacion",1)==0 ||
138             GetNumGateValue("EstadoTurbina",1)<3)) then
139             Conectar();
140         end
141     end
142 end
```

Código Funciones

```
//============================================================================//
//Todas las funciones que realizan cambios en la interfaz grafica, y en las
//entradas salidas del PLC, haci como tambien las reglas de validacion para
//la operacion correcta del sistema se declaran en este achivo.
//Si se preciona algun boton o el programa principal llama alguna funcion
//especifica, esta estará declarada aqui; ademas, dentro del codigo se
//declarán la activacion de algunas alarmas.
//============================================================================//

//Abre La valvula prellenado si la tuberia forzada no esta llena, de lo
//contrario se activa una alarma.
function void Abrir_valvula_prellenado()
    if (GetDigGateValue("TuboLleno",1)) then
        SetDigGateValue("Alarma1",1,1); //Se activa la alarma
    else
        SetDigGateValue("ValPreAct",1,1);   //Se activa la valvula
        SetNumGateValue("ImgTubo",1,1); //Se cambia de imagen en la plantilla
    end
end

//Se cierra la valvula prellenado solo si previamente esta activa y se muestra
//en pantalla la imagen de tuberia medio llena.
function void Cerrar_Valvula_Prellenado()
    if (GetDigGateValue("ValPreAct",1)) then
        SetDigGateValue("ValPreAct",1,0);   //Se desactiva la valvula
        SetNumGateValue("ImgTubo",1,2); //Se cambia de imagen en la plantilla
    end
end

//Esta funcion desactiva la Valvula de prellenado (si esta previamente activa)
//y muestra en la plantilla la imagen de tuberia forzada llena.
function void TuberiaLlena()
    if (GetDigGateValue("ValPreAct",1)) then
        SetDigGateValue("ValPreAct",1,0);   //desactiva la valvula de prellenado
    end
    SetNumGateValue("ImgTubo",1,3); //Muuestra la imagen de tubo lleno
end

//Si no esta activo el sensor de TuboLleno entonces esta funcion es Llamada y
//se verifica que imagen mostrar puede ser tubo totalmente vasio o tubo medioLleno
function void TuberiaVacia()
    if (GetNumGateValue("ImgTubo",1)==3) then
        if (GetNumGateValue("PulsosAlabes",1)==0) then
            SetNumGateValue("ImgTubo",1,2); //Imagen tubo Medio Lleno
        else
            SetNumGateValue("ImgTubo",1,0); //Imagen tubo Vacio
        end
    end
end
//Con esta funcion se abre la compuerta de servicio, si en algun caso no esta
//abierta y si prebiamente la tuberia forzada esta llena, si ya estaba abierta
//entonces se activa una alarma.
function void Abrir_compuerta_servicio()
    if (GetDigGateValue("CompServAbt",1)) then
        SetDigGateValue("Alarma2",1,1); //activar alarma de compuerta abierta
    else
        if (GetDigGateValue("TuboLleno",1)) then
            SetDigGateValue("AbrCompServ",1,1); //abre la compuerta
            SetDigGateValue("CompServAbt",1,0); //sensor compuerta abierta a 0
            SetNumGateValue("ImgCompSer",1,1);  //Imagen de compuerta abriendo
            if (GetNumGateValue("ImgTubo",1)==2) then //Imagen no actualizada
                SetNumGateValue("ImgTubo",1,3); //se pone la imagen tubo lleno
            end
        else
            SetDigGateValue("Alarma7",1,1);//se activa la alarma de tu vacio
        end                                 //no puede abrir la compuerta
    end
end
```

```
71 //Esta funcion cierra la compuerta de servicio si previamente esta abierta
72 //si estaba cerrada e intentaba cerrarse se activa una alarma
73 function void Cerrar_compuerta_servicio()
74     if (GetDigGateValue("CompServCrd",1) || GetNumGateValue("ImgCompSer",1)==0) then
75         SetDigGateValue("Alarma3",1,1); //Se activa la alarma
76     else
77         SetDigGateValue("CrrCompServ",1,1); //Se cierra la compuerta
78         SetDigGateValue("CompServCrd",1,0); //Se pone a 0 el sensor de cerrada
79         SetNumGateValue("ImgCompSer",1,4);  //se pone la imagen de cerrando
80     end
81 end
82
83 //esta funcion detiene el movimiento de la compuerta ya sea hacia arriba o hacia
84 //abajo mostrando la compuerta a medio camino si fue detenida por el operario
85 //, arriba si fue detenida por el sensor puerta abierta o abajo si fue detenida
86 //por el sensor puerta cerrada, ademas apaga las salidas ya sea para cerrar o abrir.
87 function void Detener_compuerta_servicio()
88     SetDigGateValue("AbrCompServ",1,0); //Apaga las salidas
89     SetDigGateValue("CrrCompServ",1,0);
90     if (GetNumGateValue("ImgCompSer",1)==1) then //si fue detenida por sensor
91         if (GetDigGateValue("CompServAbt",1)==1) then// de compuerta abierta
92             SetNumGateValue("ImgCompSer",1,3);  //se muestra la compueta arriba
93         else                                    //si no se muestra la compuerta
94             SetNumGateValue("ImgCompSer",1,2);  //a medio camino
95         end
96     else    //si la compuerta esta cerrando y se detubo por sensor de compuerta
97         if (GetDigGateValue("CompServCrd",1)==1) then // cerrada se muestra la
98             SetNumGateValue("ImgCompSer",1,0);  //imagen de la compuerta abajo
99             SetNumGateValue("ImgTubo",1,0);     //y tubo lleno
100            if (GetNumGateValue("PulsosAlabes",1)>=10) then // si fue detenida
101                SetNumGateValue("EstadoTurbina",1,2); //por el operario se
102            end //muetra la compuerta a medio camino
103        else
104            SetNumGateValue("ImgCompSer",1,2); //para cualquier otro caso se
105        end //muestra la imagen de la compuerta a medio camino.
106    end
107 end
108
109 //activa o desactiva la bonba de levantamiento, cambia el texto que se presenta en
110 //el boton a activar si esta desactiva y desactivar si esta activa.
111 function void BombaLevantamiento()
112     if (GetDigGateValue("BombaLevantamiento",1)) then//si esta activa
113         SetDigGateValue("BombaLevantamiento",1,0);// Desactiva la Bomba
114         TObjBeginUpdate(40);    //actualiza el label del boton a activar
115         TObjSetPropertyString(40,"Label","Activar");
116         TObjEndUpdate(40);
117     else if (GetDigGateValue("SensorPresion",1)) then//si el sensor de presion
118             SetDigGateValue("Alarma4",1,1); //esta activo y se quiere activar
119         else    //la bomba se lanza una alarma de presion adecuada. si no
120             SetDigGateValue("BombaLevantamiento",1,1);//se activa la bonba
121             TObjBeginUpdate(40);    //se actualiza el label del boton.
122             TObjSetPropertyString(40,"Label","Desactivar");
123             TObjEndUpdate(40);
124         end
125     end
126 end
127
128 //Esta funcion deside si abrir o cerrar los alabes depende del valor de las variables Aprtalabes
129 // y PulsosAlabes, tambien se modifican las imagenes que se presentan en el SCADA.
130 function void Alabes()
131 //Para cuando el valor de refercia de apertura de los alabes es mayor que el valor de apertura fisico
132     if (GetNumGateValue("ApertAlabes",1)>GetNumGateValue("PulsosAlabes",1))then
133         //Para valores de PulsosAlabes entre 0 y 10
134         if(GetNumGateValue("EstadoTurbina",1)==0 && ((GetNumGateValue("PulsosAlabes",1)>=0 ||
135            GetNumGateValue("EstadoTurbina",1)==2) && GetNumGateValue("PulsosAlabes",1)<10)) then
136             SetNumGateValue("EstadoTurbina",1,1); //Imagen abrir sin movimiento de la turbina
137             SetDigGateValue("AbrAlabes",1,1);   //Abrir Alabes
138              //Para valores de PulsosAlabes entre 10 y 40
139         else if((GetNumGateValue("EstadoTurbina",1)==1 || GetNumGateValue("EstadoTurbina",1)==2 ||
140                 GetNumGateValue("EstadoTurbina",1)==4) && GetDigGateValue("CompServCrd",1)==0 &&
141                 (GetNumGateValue("PulsosAlabes",1)>=10 && GetNumGateValue("PulsosAlabes",1)<40)) then
142                 SetNumGateValue("EstadoTurbina",1,3);//Imagen abrir con movimiento lento
143                 SetDigGateValue("AbrAlabes",1,1);   //Abrir Alabes
```

```
144                 //Para valores de PulsosAlabes entre 40 y 70
145             else if((GetNumGateValue("EstadoTurbina",1)==3 || GetNumGateValue("EstadoTurbina",1)==4
146                     || GetNumGateValue("EstadoTurbina",1)==6) && GetDigGateValue("CompServCrd",1)==0 &&
147                     (GetNumGateValue("PulsosAlabes",1)>=40 && GetNumGateValue("PulsosAlabes",1)<70)) then
148                     SetNumGateValue("EstadoTurbina",1,5);//Imagen Abrir con movimiento medio
149                     SetDigGateValue("AbrAlabes",1,1);   //Abrir Alabes
150                      //para valores de PulsosAlabes entre 70 y 100
151                 else if((GetNumGateValue("EstadoTurbina",1)==5 || GetNumGateValue("EstadoTurbina",1)==6 ||
152                         GetNumGateValue("EstadoTurbina",1)==8) && GetDigGateValue("CompServCrd",1)==0 &&
153                         (GetNumGateValue("PulsosAlabes",1)>=70 && GetNumGateValue("PulsosAlabes",1)<=100)) then
154                         SetNumGateValue("EstadoTurbina",1,7); //Imagen Abrir con movimiento medio
155                         SetDigGateValue("AbrAlabes",1,1);   //Abrir Alabes
156                          //Para cualquier otro valor de PulsosAlabes
157                     else if (GetNumGateValue("EstadoTurbina",1)==2 && GetDigGateValue("CompServCrd",1)) then
158                             SetNumGateValue("EStadoTurbina",1,1);//Imagen abrir sin movimiento de la turbina
159                             SetDigGateValue("AbrAlabes",1,1);   //Abrir Alabes
160                         end
161                     end
162                 end
163             end
164         end
165 //Para cuando el valor de refercia de apertura de los alabes es menor que el valor de apertura fisico
166     else if (GetNumGateValue("ApertAlabes",1)<GetNumGateValue("PulsosAlabes",1))then
167             //Para valores de PulsosAlabes entre 70 y 100
168             if(GetNumGateValue("EstadoTurbina",1)==8 && GetDigGateValue("CompServCrd",1)==0 &&
169               (GetNumGateValue("PulsosAlabes",1)>=70 && GetNumGateValue("PulsosAlabes",1)<=100)) then
170                 SetNumGateValue("EstadoTurbina",1,7);//Imagen cerrar con movimiento rapido de la turbina
171                 SetDigGateValue("CrrAlabes",1,1); //Cerrar alabes
172                  //Para valores de PulsosAlabes entre 40 y 70
173             else if((GetNumGateValue("EstadoTurbina",1)==6 || GetNumGateValue("EstadoTurbina",1)==7) &&
174                     GetDigGateValue("CompServCrd",1)==0 && (GetNumGateValue("PulsosAlabes",1)>=40 &&
175                     GetNumGateValue("PulsosAlabes",1)<70)) then
176                     SetNumGateValue("EstadoTurbina",1,5);//Imagen cerrar con movimiento medio
177                     SetDigGateValue("CrrAlabes",1,1);   //Cerrar alabes
178                      //Para valores de PulsosAlabes entre 10 y 40
179                 else if((GetNumGateValue("EstadoTurbina",1)==4 || GetNumGateValue("EstadoTurbina",1)==5) &&
180                         GetDigGateValue("CompServCrd",1)==0 && (GetNumGateValue("PulsosAlabes",1)>=10 &&
181                         GetNumGateValue("PulsosAlabes",1)<40)) then
182                         SetNumGateValue("EStadoTurbina",1,3);//Imagen cerrar con movimiento lento
183                         SetDigGateValue("CrrAlabes",1,1);   //Cerrar alabes
184                          //Para valores de PulsosAlabes entre 0 y 10
185                     else if((GetNumGateValue("EstadoTurbina",1)==2 || GetNumGateValue("EstadoTurbina",1)==3) &&
186                             GetDigGateValue("CompServCrd",1)==0 && (GetNumGateValue("PulsosAlabes",1)>0 &&
187                             GetNumGateValue("PulsosAlabes",1)<10)) then
188                             SetNumGateValue("EStadoTurbina",1,1);//Imagen cerrar sin movimiento de la turbina
189                             SetDigGateValue("CrrAlabes",1,1);   //Cerrar alabes
190                              //Para cualquier otro valor de PulsosAlabes
191                         else if (GetNumGateValue("EstadoTurbina",1)==2 && GetDigGateValue("CompServCrd",1)) then
192                                 SetNumGateValue("EStadoTurbina",1,1);//Imagen cerrar sin movimiento de la turbin
193                                 SetDigGateValue("CrrAlabes",1,1);   //Cerrar alabes
194                             end
195                         end
196                     end
197                 end
198             end
199 //Para cuando el valor de refercia de apertura de los alabes es igual que el valor de apertura fisico
200         else if(GetNumGateValue("ApertAlabes",1)==GetNumGateValue("PulsosAlabes",1))then
201                 //Para valores de PulsosAlabes entre 0 y 10
202                 if(GetNumGateValue("EstadoTurbina",1)==1 && (GetNumGateValue("PulsosAlabes",1)>0 &&
203                    GetNumGateValue("PulsosAlabes",1)<10)) then
204                     SetNumGateValue("EstadoTurbina",1,2);
205                     SetDigGateValue("AbrAlabes",1,0);
206                     SetDigGateValue("CrrAlabes",1,0);
207                      //Para valores de PulsosAlabes entre 10 y 40
208                 else if(GetNumGateValue("EstadoTurbina",1)==3 && (GetNumGateValue("PulsosAlabes",1)>=10 &&
209                         GetNumGateValue("PulsosAlabes",1)<40)) then
210                         SetNumGateValue("EstadoTurbina",1,4);
211                         SetDigGateValue("AbrAlabes",1,0);
212                         SetDigGateValue("CrrAlabes",1,0);
213                          //Para valores de PulsosAlabes entre 40 y 70
214                     else if(GetNumGateValue("EstadoTurbina",1)==5 && (GetNumGateValue("PulsosAlabes",1)>=40 &&
215                             GetNumGateValue("PulsosAlabes",1)<70)) then
216                             SetNumGateValue("EStadoTurbina",1,6);
217                             SetDigGateValue("AbrAlabes",1,0);
218                             SetDigGateValue("CrrAlabes",1,0);
```

```
                            //Para valores de PulsosAlabes entre 70 y 100
                        else if(GetNumGateValue("EstadoTurbina",1)==7 && (GetNumGateValue("PulsosAlabes",1)>=70 &&
                                GetNumGateValue("PulsosAlabes",1)<=100)) then
                                SetNumGateValue("EStadoTurbina",1,8);
                                SetDigGateValue("AbrAlabes",1,0);
                                SetDigGateValue("CrrAlabes",1,0);
                                 //Para valores de PulsosAlabes y ApertAlabes igual a cero
                            else if(GetNumGateValue("EstadoTurbina",1)==1 && (GetNumGateValue("PulsosAlabes",1)==0 &&
                                    GetNumGateValue("ApertAlabes",1)==0)) then
                                    SetNumGateValue("EStadoTurbina",1,0);
                                    SetDigGateValue("AbrAlabes",1,0);
                                    SetDigGateValue("CrrAlabes",1,0);
                                     //Para valores de PulsosAlabes mayores que 0 y la compuerta de cervicio cerrada
                                else if (GetNumGateValue("PulsosAlabes",1)>0 && GetDigGateValue("CompServCrd",1)) then
                                        SetNumGateValue("EStadoTurbina",1,2);
                                        SetDigGateValue("AbrAlabes",1,0);
                                        SetDigGateValue("CrrAlabes",1,0);
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end

//Se activa en el PLC el proceso de inicio automatico del sistema de generacion en la central.
function void InicioAutomatico()
    SetDigGateValue("InicioPC",1,1);
end

//Se activa en el PLC el proceso de paro automatico del sistema de generacion en la central.
function void Terminar_Generacion()
    SetDigGateValue("ParoPC",1,1);
end

//Aumenta la varible sincronoscopio, la cual es el parametro de sincronia con la red
function void Sincronizar_arriba()
    int  Sincronoscopio;
    Sincronoscopio=GetNumGateValue("Sincronoscopio",1)+5;
    SetNumGateValue("Sincronoscopio",1,Sincronoscopio);
end

//disminuye la varible sincronoscopio, la cual es el parametro de sincronia con la red
function void Sincronizar_abajo()
    int  Sincronoscopio;
    Sincronoscopio=GetNumGateValue("Sincronoscopio",1)-5;
    SetNumGateValue("Sincronoscopio",1,Sincronoscopio);
end
//se conecta o desconecta la central a la red.
function void Conectar()
    if (GetDigGateValue("ConectarRED",1)) then //si esta conectada
        TObjBeginUpdate(60);     //se actualiza el label del boton
        TObjSetPropertyString(60,"Label","Conectar");
        TObjEndUpdate(60);
        SetDigGateValue("ConectarRED",1,0); //y se desconecta de la red
    else // si esta desconectada
        if (GetNumGateValue("Sincronoscopio",1)==0 && GetDigGateValue("Exitacion",1) &&
            GetNumGateValue("EstadoTurbina",1)>3) then //Si estamos en sincronia y esta activa
            //la exitacion
            TObjBeginUpdate(60);//Se actualiza el label del boton
            TObjSetPropertyString(60,"Label","Desconectar");
            TObjEndUpdate(60);
            SetDigGateValue("ConectarRED",1,1);//Se conecta a la red
        else
            SetDigGateValue("Alarma5",1,1); //de lo contrario se lanza una emergencia
        end
    end
end
```