Universidad Autónoma de Querétaro

Facultad de Ingeniería

Desarrollo de un Control Modular para maquinaria aplicado a una Máquina de Inyección de Plástico

TESIS

Que para obtener el grado de
Maestro en Ciencias línea terminal instrumentación y control automático

presenta
José Agustín Bravo Curiel

Asesor de tesis
Dr. Gilberto Herrera Ruiz

Querétaro, Qro. Abril, 2004

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Instrumentación y Control Automático

Diseño de un Control Modular para maquinaria aplicado a una Máquina de Inyección de Plástico

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Ciencias

Presenta:
José Agustín Bravo Curiel

Dirigido por:
Dr. Gilberto Herrera Ruiz

SINODALES

Dr. Gilberto Herrera Ruiz
Presidente

M. en C. Pedro Daniel Alaniz Lumbreras
Secretario

M. en C. Roberto Augusto Gómez Loenzo
Vocal

M. en C. Martín Larios Osorio
Suplente

M. en C. Roque Alfredo Osornio Ríos
Suplente

Firma

Firma

Firma

Firma

Firma

M. en I. Gerardo René Serrano Gutiérrez
Director de la Facultad de Ingeniería

Dr. Sergio Quesada Aldana
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Mayo de 2004
México

RESUMEN

En este trabajo se hizo el diseño y la implementación de un sistema de control aplicado a una máquina de inyección de plástico, que le permita a las empresas tener a costos competitivos las opciones que ofrecen los controles comerciales modernos. El sistema tiene características que le permitan adaptarse rápidamente a los cambios demandados por la cadena productiva. Una de ellas es una arquitectura abierta, que permite a futuros investigadores agregar funciones al sistema de manera sencilla y rápida. Otra de ellas es la posibilidad de ser utilizado para diferente tipo de maquinaria, adaptándose y configurándose mediante un programa de manera similar que un PLC. El control se puede dividir en dos grandes bloques. El primero de ellos es el equipo electrónico: la computadora personal (PC), y las tarjetas de señales de entradas y salidas. El segundo es el programa que tendrá la PC, y que procesará toda la información de entradas y salidas, y generará la interfaz con el usuario para permitir que éste modifique y adapte el proceso de acuerdo con sus necesidades. Se diseñaron tarjetas para manejar las señales más comunes dentro de la maquinaria. El programa de control se divide en tres secciones. La primera de ellas controla la ejecución del programa del usuario, la segunda es la interfaz del operador, y la tercera permite al usuario integrar datos tomados de variables del proceso para analizarlos y llevar a cabo un control estadístico del proceso. El proyecto se implementó en dos máquinas de inyección de plástico. Se instaló todo el equipo y se diseñó el programa de operación. Una vez terminada la instalación del equipo, éste se programó y se hicieron las pruebas de operación y se depuró el programa hasta lograr la operación cíclica sin interrupción. Los resultados del proyecto proveen un excelente punto de partida para la investigación de los procesos relacionados con la inyección de plástico, como la fabricación de moldes, el estudio de los parámetros del proceso, la modificación de la maquinaria para mejorar su eficiencia, y más desarrollo de equipo electrónico para control de maquinaria.

(**Palabras clave**: guía, referencia, tesis)

SUMMARY

This work explains the design and implementation of a control system applied to an injection molding machine. This system will allow factories to have all the advantages of modern commercial controls, but at competitive costs. The system has such characteristics that will allow it to adapt to the changes required by the assembly line. One of them is the open architecture concept, which allows researchers to add functions to the system in the future easily and quickly. Another one is the possibility to be used to control different types of machinery, adapting and configuring it through a program, as done with a PLC. The control can be divided into two big blocks. The first one is the electronic equipment: the personal computer (PC), and the input and output cards. The second one is the program that the PC will run, which will process all the input/output information. It will provide the user with an interface, to allow him to modify and adapt the process to his needs. Several electronic cards were designed to manage the most common types of signals used in machinery. Those signals control positions, security devices and sensors, valves, and actuators. The control program is divided into three sections. The first of them controls the execution of the user program, the second is the user interface, and the third one allows the user to record data taken from process variables for further analysis and to carry out statistical process control. This project was installed on two injection molding machines. Afterwards the equipment was connected and the operation program was designed. Once the installation was finished, the equipment was programmed and the operation test were run. The program was debugged until achieving continuous cyclic operation. This project's results provide an excellent departure point to do research on the processes related to injection molding, such as mold design and manufacture, studying process parameters, machine reengineering to improve efficiency, and more development of electronic equipment for machinery control.

(**Keywords**: control, inyection molding, open architechture)

A mis padres y mi hermana,
por sus grandes enseñanzas.

AGRADECIMIENTOS

Al Dr. Gilberto Herrera Ruiz, por su gran apoyo durante toda la maestría.

A mis amigos, porque demandan lo mejor de mí en el ámbito personal, profesional, y como estudiante.

A todos mis compañeros de la maestría, en quienes encontré grandes amigos, especialmente a Roque y Roberto, quienes además me apoyaron fuertemente durante los estudios.

A la UAQ, por haberme permitido llevar a cabo mis estudios de posgrado en sus instalaciones.

INDICE

		Página
Resumen		i
Summary		ii
Dedicatorias		iii
Agradecimientos		iv
Indice		v
Indice de tablas		vii
Indice de figuras		viii
I.	INTRODUCCION	1
II.	FUNDAMENTOS DEL SISTEMA PROPUESTO	4
	2.1 Estándares industriales	6
	2.2 Marcas comerciales e investigación	8
	2.3 Protocolos de comunicación	13
	2.4 Descripción del sistema propuesto	22
	2.5 Arquitectura abierta	27
III.	DESCRIPCIÓN DEL SISTEMA DESARROLLADO	30
	3.1 Comunicación entre tarjetas	31
	3.2 Descripcion del Hardware	40
	3.2.1 Tarjeta de entradas digitales	41
	3.2.2 Tarjeta de salidas digitales	43
	3.2.3 Tarjeta de entradas analógicas	45
	3.2.4 Tarjeta de salidas analógicas	48
	3.2.5 Tarjeta de entradas de termopar tipo J	49
	3.3 Descripción del Software	51
	3.3.1 Módulo de control secuencial	52
	3.3.2 Programación	55

3.3.3 Estructura interna y funcionamiento de los temporizadores 64
3.3.4 Estructura interna y funcionamiento de los contadores 67
3.3.5 Banderas internas 71
3.3.6 Registros internos 73
3.3.7 Módulos del programa 74

IV. IMPLEMENTACIÓN Y PRUEBAS 79

V. CONCLUSIONES 89

BIBLIOGRAFÍA 91

ANEXOS 92
A. Publicación 93
B. Pantallas de la aplicación implementada 101
C. Códigos de los microcontroladores 108
D. Códigos en C 130

INDICE DE TABLAS

TABLA		Página
3.1	Grupos de señales y su función	34
3.2	Rangos de direcciones asignados a cada ranura	36
3.3	Registros internos de cada tipo de tarjeta	37
3.4	Características de las tarjetas diseñadas	41
3.5	Herramientas para programar la secuencia de control	56
3.6	Códigos mnemónicos para los elementos de los diagramas	57
3.7	Elementos especiales	57
3.8	Interpretación del diagrama de la fig. 3.12	61
3.9	Extracción de ecuaciones de la fig. 3.13	62
3.10	Códigos de tipo de tarjetas	63
3.11	Archivos de definición del programa	74
3.12	Módulos principales	75
3.13	Nombres de clases y tipos de tarjetas	77
A.1	Archivos de la aplicación	98

INDICE DE FIGURAS

Figura		Página
2.1	Esquema del sistema propuesto	24
2.2	Conexión a la máquina	26
3.1	Bloques de la tarjeta de interfaz	34
3.2	Diagrama de tiempos de un ciclo de lectura	39
3.3	Diagrama de tiempos de un ciclo de escritura	40
3.4	Etapa de alto voltaje de entradas digitales	42
3.5	Etapa de control de entradas digitales	42
3.6	Etapa de potencia de salidas digitales de corriente alterna	44
3.7	Etapa de potencia de salidas digitales de corriente directa	44
3.8	Etapa de control de salidas digitales	45
3.9	Diagrama de la tarjeta de entradas analógicas	48
3.10	Diagrama de la tarjeta de salidas analógicas	49
3.11	Diagrama de la tarjeta de entradas de termopar	50
3.12	Ejemplo de diagrama de escalera	60
3.13	Ejemplo para extracción de ecuaciones	62
3.14	Estructura de temporizadores	64
3.15	Alojamiento en memoria de un temporizador	66
3.16	Ejemplo de uso de temporizadores	67
3.17	Estructura de un contador	68
3.18	Alojamiento en memoria de un contador	69
3.19	Ejemplos de uso de los contadores	71
3.20	Ejemplo de uso de banderas internas	72
3.21	Ejemplo de uso de registros internos	73
3.22	Procedimientos del módulo principal	75
3.23	Estructura jerárquica de las clases	78
4.1	Botonera original e interfaz de operación nueva	79
4.2	Disposición de equipo en el gabinete eléctrico	80
4.3	Válvula y monobloque de inyección	80
4.4	Vista general de la máquina	81
4.5	Terminal del operador	82
4.5	Estructura de árbol de las pantallas implementadas	84
A.1	Señales del bus de hardware	95
A.2	Botonera y terminal del operador	99
A.3	Monoblock para control de la velocidad de inyección	100

B.1 Pantalla 0 (Principal) 101
B.2 Pantalla 1 (Ajuste de mov. de molde) 101
B.3 Pantalla 2 (Ajuste de parámetros de inyección) 102
B.4 Pantalla 3 (Ajuste de expulsión) 102
B.5 Pantalla 4 (Ajuste de temperaturas) 103
B.6 Pantalla 5 (Ajuste de parámetros de carga 103
B.7 Pantalla 6 (Acceso a pantallas de mantenimiento) 104
B.8 Pantalla 61 (Tarjetas de salidas) 104
B.9 Pantalla 62 (Tarjetas de salidas) 105
B.10 Pantalla 63 (Banderas internas) 105
B.11 Pantalla 64 (Registros internos 16-bit) 106
B.12 Pantalla 65 (Timers) 106
B.13 Pantalla 66 (Contadores) 107
B.14 Pantalla 7 (Ajustes varios) 107

I. INTRODUCCIÓN

La industria manufacturera de nuestro país representa uno de los sectores económicos más importantes, especialmente en nuestra área geográfica. La grán mayoría de las empresas pertenece al sector de la micro y pequeña empresa. Sin embargo, a pesar de ser un sector de suma importancia, éstas empresas carecen de capital suficiente para ser competitivas comparadas con los grandes corporativos internacionales, que cuentan con equipo de alta tecnología. Su maquinaria es anticuada y carece de las herramientas de automatización que ayudan a aumentar la eficiencia dentro de las grandes empresas.

Existe una fuerte necesidad de contar con una planta productiva que tenga flexibilidad, y que permita efectuar los cambios requeridos para adaptar líneas de producción y manejar diferentes productos de una forma sencilla. Cuando una empresa cuenta con tal flexibilidad, puede producir con costos competitivos.

En nuestra zona geográfica, una parte muy importante de la industria se dedica a actividades relacionadas con maquinaria y equipo. Según los censos económicos del INEGI, en los años 1999 y 2000, este sector de la industria ocupó al 42% del total del personal laborando en la industria del estado, representando cerca del 26% del total de establecimientos de la industria de la manufactura.

La industria automotriz representa un sector muy importante dentro de nuestra área geográfica. Muchas empresas en este y otros sectores del área fabrican piezas de plástico por inyección. Sin embargo, la industria de la inyección del plástico, se encuentra también sufriendo de los problemas mencionados.

Una solución posible a esta situación es el cambio de controles en la maquinaria. Sin embargo, las opciones viables actualmente son de procedencia extranjera. Esto implica un alto costo para los empresarios, y para la micro y

pequeña empresa muchas veces es inaccesible. Si tomamos en cuenta que alrededor del 95.8% del total de establecimientos industriales del estado está conformado por micro y pequeña industria, podríamos decir que la situación tecnológica actual deja sin posibilidades de competir prácticamente a todas las empresas (INEGI, 2002).

Por otro lado, la gran mayoría de los sistemas de control incluidos en la maquinaria no permiten al usuario efectuar cambios en el mismo. El equipo tiene una cantidad determinada de opciones sobre las que el usuario tiene facultades limitadas para hacer modificaciones. Existe una fuerte tendencia dentro del mercado de máquinas herramientas para buscar controles que tengan la flexibilidad de agregar algoritmos de control que se adapten a los requerimientos específicos de un proceso (Pritschow, 2001). Se aborda el tema de la justificación de la modernización de maquinaria explicando que la mejora de la eficiencia del proceso a bajo costo es una justificación muy importante, y en algunos casos suficiente (Donald, 1998). La tendencia a nivel general en la industria es a desarrollar controles cada vez más flexibles, y que permitan generar corridas de producción cortas sin el alto costo inherente a las líneas de producción dedicadas.

La globalización y la eliminación de las barreras arancelarias dentro de la generación de tratados de libre comercio obligan a las industrias, entre otras cosas, a buscar la innovación tecnológica (Castellanos, 2001).

Esta situación ha provocado que se fortalezcan a nivel mundial los estándares que especifican sistemas de arquitectura abierta. Dichos sistemas se caracterizan por permitir que diferentes proveedores fabriquen equipo adaptándose a las características del físicas o al protocolo de comunicación de la arquitectura en cuestión.

Tomando en cuenta todos estos factores, el problema que se aborda aquí se puede dividir en dos partes fundamentales. La primera consiste en la falta de

controles para maquinaria flexibles y económicos que presenten a las empresas locales opciones viables para mejorar su competitividad y eficiencia en la producción. La segunda es la falta de plataformas (equipo y maquinaria) que permitan a los investigadores avanzar en el desarrollo de algoritmos y diseño de equipos a nivel local.

Parte del origen de este proyecto surge de observar que la mayoría de los proyectos de investigación generados dentro de nuestra facultad y que involucran el uso de hardware para adquisición y control de señales, utilizan una cantidad importante de tiempo en el desarrollo de éste hardware, y cada uno asume su propia arquitectura. Esto conlleva a que un fuerte porcentaje del tiempo sea invertido en la fabricación de hardware, desviando la atención del investigador, que debería estar centrada en la solución del problema principal de su proyecto. En otros casos, incluso limita el proyecto a simulaciones y cálculos

El objetivo del proyecto presentado aquí es proponer una arquitectura para hardware, que represente una plataforma de investigación que permita el desarrollo de nuevas investigaciones en el área de controladores industriales modulares y flexibles, y presentar una opción de estandarización de las características del hardware utilizado en los proyectos de nuestra Facultad; manteniendo un esquema de arquitectura abierta que permita en un futuro la mejora del equipo, y el desarrollo de equipos que cubran necesidades nuevas o específicas.

II. FUNDAMENTOS DEL SISTEMA PROPUESTO

Los procesos industriales en su mayoría se encuentran actualmente automatizados, y controlados por autómatas programables (PLC's por sus siglas en inglés)[1]. Otra porción menor se encuentra controlada por computadores de control de proceso y reguladores industriales. Los fabricantes de equipo han invertido grandes capitales en la investigación de áreas como Redes de Comunicación industrial y buses de campo (Piedrafita, 2001, 3).

La automatización es una parte fundamental de las empresas competitivas dentro del sector manufacturero. General Electric define la automatización como "la inteligencia, el sistema nervioso, y el músculo de inteligencia que integra y ejecuta actividades en un ambiente industrial." Y las agrupa en tres áreas principales:

- Actividades de Ingeniería
- Actividades de Manufactura
- Actividades de Administración de la fábrica

El área particular de interés en este caso, es la de Actividades de Manufactura. La automatización de esta área contribuye al incremento en productividad, flexibilidad, y calidad a través de áreas como Robótica, Controles Programables, Tecnología de Control Numérico, manejo de materiales, y almacenamiento (Gardner, 1985, 31).

Uno de los aspectos más importantes a tomar en cuenta es la estandarización. Ésta es importante porque permite que más gente tenga acceso al uso o diseño de un producto. El mayor beneficio lo recibe el usuario final. El uso de estándares permite que los costos de los productos disminuyan, y que más fabricantes ofrezcan productos similares. Sin embargo, se requiere de un gran esfuerzo para estandarizar algún producto o proceso (Gardner, 1985, 35).

[1] Programmable Logic Controller

En 1983, se realizó en California un Simposio sobre Manufactura Integrada Automatizada, auspiciada por la ASMT (American Society for Testing and Materials). Se especificaba entonces un estándar para la comunicación de sistemas de manufactura flexibles (FMS, por sus siglas en inglés), basado en los siguientes parámetros (Gardner, 1985, 61):

- Sincronización. Señales necesarias para comunicar equipos con diferentes velocidades de operación.
- Tamaño de bloque o línea. Se refiere a la cantidad de datos (o bits) transmitidos en un mensaje.
- Confirmación de recepción. Se refiere a una señal que se utiliza para confirmar que los datos llegaron a su destino.
- Comunicación en serie o paralelo. Aquí se determina si se aprovecha el costo reducido y la confiabilidad de la transmisión en serie, o la alta velocidad de la transmisión en paralelo.
- Interrupciones o revisión cíclica. Se define si la comunicación ocurre accesando cíclicamente los elementos conectados al sistema[2] o mediante interrupciones de un elemento a otro.
- Comunicación mediante mensajes o Entradas/Salidas digitales. Es posible enviar más información utilizando mensajes que utilizando entradas y salidas digitales, pero la comunicación por mensajes es más complicada de implementar.
- Envío de datos en códigos o en forma binaria. Especifica si se utiliza algún código especial (ASCII por ejemplo), o los datos son simplemente información procesada directamente.
- Estructura de protocolo. Define si existen encabezados, chequeos de seguridad, destino y fuente, etc. dentro de la información que se transmite.
- Conjunto de comandos. Se utilizan para dirigir los subsistemas del FMS.

[2] En inglés se utiliza el término *polling*.

- Otros. Otros elementos como correción de errores y definir si la comunicación es síncrona o asíncrona.

El sistema que se propone debe cumplir con ciertas características para resolver correctamente las necesidades expuestas y los objetivos del proyecto. Para ello, es importante tener un conocimiento básico del estado del arte, y los estándares que existen actualmente relacionados con el proyecto. A partir de ése conocimiento es posible tomar las características e ideas más interesantes, y plantear una solución pertinente al problema que se desea resolver.

Se incluye a continuación una breve descripción de los estándares industriales de arquitectura abierta, los equipos que ofrecen algunas marcas comerciales, y los protocolos de comunicación que se utilizan a nivel industrial para comunicar equipos.

2.1 Estándares industriales

Entre los estándares más importantes se encuentran el OSACA, OSEC, el JOP, y el OMAC, que promueven el uso de interfaces de programación para la comunicación entre el usuario y el controlador.

OSACA

OSACA (Open System Architecture for Controls within Automation systems) es una iniciativa Europea que tiene como objetivo reducir los costos de desarrollo, mantenimiento, entrenamiento y documentación de los sistemas de control del mercado mundial. Se basa en la idea de que los fabricantes de máquinas herramientas quieren asegurar sus inversiones y ser capaces de integrar su conocimiento específico a diferentes controles, y que los usuarios finales desean también independizarse de los proveedores únicos. Los diseñadores de software pueden proveer aplicaciones de automatización y monitoreo de propósito general, o algoritmos específicos para ciertos campos. La arquitectura de OSACA se basa en la definición de una interfaz de programación

independiente de cualquier plataforma de hardware o sistema operativo utilizando estándares existentes donde sea necesario.

OSACA especifica tres áreas principales:

- Sistema de comunicación. Define un hardware y una interfaz independientes del software para intercambiar información entre diferentes módulos de una aplicación. Permite la combinación de módulos distribuidos entre diferentes procesadores.
- Arquitectura de referencia. Determina las unidades funcionales de un controlador, como un Controlador de Movimiento, o un Control de Lógica, así como las interfaces externas de los mismos. Esto permite integrar la funcionalidad de unidades externas intercambiando datos de una forma bien definida.
- Sistema de configuración. Permite la configuración dinámica de un controlador cargando diferentes aplicaciones al inicializarlo. Esto permite la sincronización entre procesos distribuidos, y especificar la topología necesaria para una función especifica.

OSEC

La OSE (Open System Environment for Manufacturing) fue establecida en Diciciembre de 1994. La arquitectura que desarrollaron pretende proveer una API (Application Programming Interface) en forma de protocolo de interfaz que permita intercambiar información entre componentes de software. Cada uno de los bloques funcionales puede encapsularse en forma de objeto.

JOP

De forma paralela al OSE, se fundó en Japón el JOP (Japanese Open Promotion Group). La finalidad de éste comité era trabajar en la estandarización de tecnologías de controladores abiertos. Aproximadamente 50 participantes colaboraron, incluyendo vendedores de controles, fabricantes de máquinas herramientas, integradores, usuarios y académicos. Uno de los grupos se dedicó a

desarrollar una API estándar para hacer interfaces entre PC y máquinas de control numérico. El nombre de la interfaz de programación es PAPI, que fue aprobado y en Octubre del 2000 fue publicado por la JIS (Japanese Industrial Standard).

OMAC

Open Modular Architecture Controllers es un foro industrial cuya finalidad es promover el avance del estado del arte. Se definió una especificación de API que adoptó clases para dar una solución partiendo del modelo de objetos de componentes de Microsoft (COM) para formar componentes plug and play. El problema principal es la falta de calendarización de tareas de tiempo real apropiativas bajo Windows, pero existen extensiones del sistema operativo que se pueden utilizar para solucionar esto.

2.2 Marcas Comerciales e Investigación

Al hablar de automatización, una de las opciones más utilizada por la industria para transformar sus máquinas es la automatización a través de PLC o de controladores dedicados. En el caso de la inyección de plástico, las marcas más reconocidas para éstos propósitos son Allen Bradley (división de Rockwell), Siemens, y Barber Coleman. A continuación se detallan las características de algunos de estos equipos, y los avances que estos y otros fabricantes ofrecen a la industria.

Allen Bradley

Esta división de Rockwell Semiconductors se ha posicionado en el mercado mundial como uno de los líderes en la automatización industrial, no sólo en maquinaria para inyección de plástico, sino en todas las áreas de automatización.

Fabrica desde equipos muy pequeños como la serie MicroLogix, hasta equipos muy completos con múltiples microprocesadores y redes industriales para monitoreo por PC e Internet. Una de las series más utilizada en nuestro país es la

serie SLC500. En la maquinaria de inyección de plásticos, ésta serie ha ido reemplazando paulatinamente a la familia 5 de PLC's, que se sigue fabricando, pero implica costos mucho mayores; por ello la industria los ha ido desplazando.

Allen Bradley ha incluido dentro de sus productos equipos específicos para la automatización de maquinaria de inyección de plástico. Ejemplos de ellos son los controles PRO-SET 200 y PRO-SET 700 para la familia PLC5.

Estos productos cuentan con dos módulos llamados QDC y QI para controlar presiones y velocidades de carga e inyección. Estos realizan control adaptable en lazo cerrado y compensan las variaciones del proceso y de la máquina. Los módulos comparan el valor real de las variables del proceso contra su valor deseado y ajustan las señales de salida a los amplificadores de las válvulas hidráulicas. También están disponibles los módulos TCM, que permiten controlar las temperaturas del cañón de la máquina en lazo cerrado.

El sistema tiene una pantalla de cristal líquido que permite al usuario visualizar y monitorear el proceso. Aquí se tiene la posibilidad de cambiar y mover los campos de datos que son muestran ahí, o de quitar y agregar pantallas con los datos deseados.

Permite también graficar el desempeño de la inyección, y compararlo con gráficas anteriores para poder ajustar los parámetros del proceso hasta llegar a un desempeño predefinido. Y es posible llevar control estadístico del proceso en tiempo real.

Todas las gráficas y parámetros del proceso pueden ser grabadas en disco para respaldarlas, trasladarlas a otra máquina, o utilizarlas cuando se vuelva a instalar el mismo molde.

Siemens

Esta empresa se ha ubicado también dentro del grupo de líderes en automatización a nivel mundial, con una fuerte presencia en la comunidad europea – dada su procedencia alemana – y en Latino América. Algunos países que carecen de tratados de intercambio comercial con Estados Unidos importan modelos de equipos cuyo uso en Europa se descontinúa, beneficiándose así de precios más económicos.

Las familias de productos Siemens más destacadas en México para la automatización de maquinaria para inyección de plástico son la series Simatic S5 y S7. Aî igual que Allen Bradley, Siemens ha desarrollado productos especiales para agregar a estos equipos la capacidad de controlar las variables específicas de éste proceso.

Barber Colman

Dentro de la industria de la inyección de plástico, ésta empresa tiene un gran prestigio. Varios fabricantes de maquinaria tienen contratos con ella para que les fabrique sus controles – muchas otras lo hacen también con General Electric. Proveen desde un control de temperatura, hasta el control total de las máquinas más completas, especialmente con su serie de controles MACO.

Los controles MACO son equipos que el usuario puede adaptar a su máquina, y configurarlos según las señales de entradas, salidas, y opciones con las que cuenta la máquina.

Consisten en un controlador principal, una interfaz para el usuario, y un gabinete con ranuras para tarjetas. El usuario diseña las pantallas que tendrá el control con un software para PC. Después configura las entradas y salidas necesarias y las conecta a su gabinete. Las temperaturas se controlan en lazo cerrado auto ajustable, y permite guardar perfiles de moldeo. La serie IMPACT

mejora el desempeño de estos controles incluyendo algoritmos de control autoajustables para la inyección, carga, y movimientos de la prensa.

Ferromatik

Ferromatik Milacron (Cincinatti) ofrece un control basado en PC llamado XTREEM, que permite documentar los procesos y comunicar máquinas entre sí, o a una máquina central vía internet. El producto es una PC completa con pantalla de 10.4", unidad de disco flexible, puerto para teclado, puerto paralelo y puerto ethernet. Permite llevar control estadístico de procesos e integrar experimentos basados en el reconocimiento de piezas buenas y malas. Ofrece la posibilidad de identificar al operador y llevar su bitácora de asistencia.

Boy

Esta empresa ataca el problema de baja repetibilidad con comparadores de hardware fuera del microprocesador central. Ofrece comunicaciones via CAN y crea archivos individuales por día con 65 parámetros de la máquina. El sistema se llama PROCAN MD, y permite también guardar 30 perfiles en EEPROM y regular los accesos a través de cuatro niveles de seguridad.

Hunkar

Ofrece un sistema que trabaja con servoválvulas de alta velocidad con control de lazo cerrado para compensar cambios en el sistema hidráulico. Ofrece perfiles de inyección de 10 puntos con interpolación a 100. Incluye un control preciso de baja presión de inyección, y contrapresión para la carga. Tiene la capacidad de ajustar automáticamente la dosis, y puede llevar control estadístico de procesos para 32 parámetros. Su interfaz hombre-máquina es programable y puede generar reportes de producción y asistencia. Uno de sus atractivos es que incluye autodiagnóstico y ahorro de energía a través del uso de servoamplificadores hidráulicos.

Toyo

Este fabricante japonés de maquinaria, ha centrado su interés sobre el control de lazo cerrado en la velocidad, posición y presión de inyección. Su sistema permite que el cambio en la inyección sea vía posición, tiempo o presión. Tiene algunas funciones adicionales como la detección de falta de material, y control de lazo cerrado también para la velocidad y posición de prensa y expulsor. Al igual que otros sistemas presenta autodiagnóstico. Dentro de sus capacidades gráficas, puede mostrar el desarrollo de la presión y velocidad de inyección, de la posición del molde y del expulsor. También incluye control estadístico del proceso para los datos monitoreados, y un historial de alarmas.

Deicon

Un análisis de elementos finitos o experimentos para evaluar un perfil de presión deseado en el MOLDE es ofrecido por DEICON (Dynamics and control). El sistema toma acciones correctivas en tiempo real para ajustar el proceso a través del perfil de presión hidráulica para obtener el perfil de presión deseado en el molde.

Intellimold

Uno de los controles más avanzados e interesantes es el que presenta la empresa Intellimold. Se basa en un proceso de control en tiempo real que utiliza una cavidad pre-presurizada y control del flujo de material. En el sistema se calcula un nuevo parámetro llamado Presión Interna de Fusión (IMP por sus siglas en inglés). Dicho parámetro se calcula con base en el flujo, temperatura de fusión, factores de encogimiento, configuración de las partes, y niveles internos de esfuerzo. Se colocan transductores de presión en la boquilla de inyección y en la cavidad del molde en las últimas partes que se llenan al inyectar. Previo a la inyección del plástico se inyecta aire a presión en la cavidad, y durante la inyección los transductores monitorean presión y temperatura. El control ajusta la velocidad de inyección y mantiene constante el valor de la IMP. El resultado es que se compensan encogimientos. Es un sistema muy apto para procesos que

requieren alta precisión, apariencia estética o bajos niveles de esfuerzo en la pieza. El control preciso del flujo y de la presión de inyección dentro del molde permiten la mezcla de plásticos en el mismo cañón, forzando que uno de ellos forme el núcleo y otro de ellos forme la cubierta. Con esto se pueden formar acabados de primera calidad con un alto porcentaje de reciclaje, ya que éste se queda formando la parte interna de la pieza, mientras que el exterior es formado por el material virgen. El perfil de inyección logrado produce un eficiente contacto de la pieza con el molde, con lo que se mejora el proceso de enfriamiento de la pieza, reduciendo el tiempo de ciclo. Además se eliminan porosidades y rugosidades causadas por aire y gases dentro del molde. El procedimiento reduce la variación de densidad en la pieza y los esfuerzos internos. Dentro del proceso se compensan variaciones en variables como la viscosidad del aceite hidráulico, temperatura del molde y del ambiente, y diferencias entre lotes de materia prima. Se reducen los tiempo de ciclo, las presiones necesarias, y la fuerza de cierre necesaria.

Estudios sobre el proceso

Se tiene poca documentación sobre los estudios que actualmente se realizan sobre la inyección de plásticos. Uno de ellos es el del Dr. Danian Zheng, profesor del Departamento de Ingeniería Mecánica e Industrial de la Universidad de Illinois, en Chicago. Él realizó una investigación en la que intentó mejorar el desempeño del proceso centrando su atención en la transición entre la inyección en alta velocidad y la etapa de sostenimiento, para evitar cambios bruscos de presión y velocidad, utilizando controladores interactivos de aprendizaje.

2.3 Protocolos de Comunicación

Uno de los aspectos más importantes a considerar al diseñar un equipo para uso industrial es el protocolo de comunicación que utilizará. Existen varios estándares industriales aprobados por diferentes asociaciones internacionales alrededor del mundo. A continuación se explican las características de algunos de

los más utilizados en la industria, tanto en intercambio de información como en buses de campo (Piedrafita, 2001, 91).

PROFIBUS

El PROFIBUS está reglamentado bajo las normas DIN e IEC. Dado que es un bus muy complejo, requeriría de mucho hardware para utilizarlo directamente con sensores y actuadores. Sin embargo, es aplicable para entradas y salidas remotas en su versión DP.

Sus competidores más cercanos son el Interbus y el CAN para la versión DP, y el Fieldbus para la versión PA. Varios fabricantes proveen periféricos para PC que soportan este tipo de BUS, y proveen los drivers de software para manejar el protocolo.

El bus se comunica mediante dos cables con una longitud máxima de 1.2Kms. sin repetidores. El estándar especifica un máximo de 3 repetidores, pero en la práctica se pueden colocar hasta diez, para una longitud máxima alrededor de 10 kilómetros.

La comunicación puede ser del tipo punto a punto, multipunto, o broadcast, y la cantidad de maestros es ilimitada. Sin embargo, en la práctica, rara vez se encuentra más de uno en el mismo bus. En este tipo de bus se puede activar tanto la lectura de entradas como la escritura de salidas síncrona.

La seguridad de la información es alta, pues la transmisión es balanceada sobre los dos cables, y se puede configurar la retransmisión después de una falla hasta 8 veces.

AS-INTERFACE

El AS-Interface esta reglamentado bajo la norma IEC. Este protocolo fue diseñado especialmente para trabajar con actuadores y sensores. Actualmente no

tiene un competidor directo. Cada punto remoto está limitado a 4 entradas digitales y 4 salidas digitales, o 2 señales analógicas de entrada y 2 de salida. Las comunicaciones no están permitidas entre dispositivos inteligentes, por lo que solo permite un maestro conectado al bus. Existen periféricos comerciales para PC que soportan este tipo de comunicación, que se basa en dos alambres. Aunque no tiene una transmisión balanceada, tiene su seguridad basada en código Manchester por bit, y paridad para cada mensaje; y soporta dos retransmisiones en caso de falla.

El bus tiene su propio protocolo, y está diseñado para utilizar arquitecturas de estrella, bus, o árbol. Se comunica mediante 2 alambres cuya longitud máxima es de 100 metros, expandible a un máximo de 300 metros con dos repetidores. Su velocidad de comunicación es fija a 167 Kbps., y el tamaño de cada mensaje puede variar dependiendo de si se utilizan señales analógicas o digitales. Cada paquete tiene un mínimo de 24 bits, lo que significa una cantidad adicional de información considerable para los nodos con señales digitales. Por el tipo de comunicaciones que utiliza, no puede hacer lecturas y escrituras síncronas en las señales de entrada o salida.

INTERBUS

El INTERBUS está reglamentado bajo las normas DIN e IEC. Existen cuatro variantes, llamadas V1 a V4, y una más llamada interbus loop. Tiene capacidad para manejar directamente señales de E/S remotas. Permite la existencia de un solo maestro por bus. Cada variante es compatible con las anteriores. La más utilizada actualmente es la V4. Es el principal competidor del Profibus DP, y cuenta con una alta estabilidad.

Existen perifericos para PC, fabricados por un proveedor llamado Phoenix, que también provee los drivers de software.

Físicamente consta de 4 hilos que se pueden configurar en bus, estrella, o árbol. Los terminadores vienen incluidos en cada nodo. Se utiliza una transmisión balanceada del tipo RS485. Cada nodo se convierte automáticamente en un repetidor. La distancia máxima de conexión es de 13 kilómetros, y la velocidad de transmisión es de 500Kbps. La última versión permite manejar velocidades de 2Mbps.

La máxima cantidad de nodos que soporta es de 512. Está basado en mensajes de punto a punto, sin broadcast. El tamaño de cada mensaje tiene un mínimo de 4 bits, y un máximo de 8192. Cada paquete tiene un encabezado de 48 bits más 5 bits por cada byte. Contiene una CRC de 16 bits por mensaje, y permite hasta 3 retransmisiones después de una falla.

Este protocolo permite el manejo de E/S analógicas. Existe un máximo de 16 bytes para entradas, y 16 para salidas en cualquier combinación analógico-digital por nodo. La transferencia síncrona de datos también está soportada.

MODBUS

El MODBUS fue desarrollado por Modicon. No existe todavía un estándar internacional. Tiene dos variantes: la ASCII, y la RTU (remote terminal unit). Tiene una similitud del 99% con el bus francés JBUS. Por su diseño, no permite manejar sensores y actuadores directamente, pero sí de manera remota.

La variante RTU es la más utilizada. No requiere de interfaces para PC puesto que utiliza comunicación RS232 o RS485 con un convertidor convencional a RS232. Tanto Windows como Linux soportan el manejo de los puertos, por lo que no requiere drivers especiales.

Se utiliza el mismo cableado del RS232 o RS485, conectado en una topología de bus, requiriendo terminadores en el caso de utilizar RS485. La transmisión también se hace conforme al protocolo físico utilizado, y las distancias

máximas permitidas son de 60m para RS232, y de 1200m para RS485. Permite una velocidad variable, usualmente menor a 38.4Kbps. Lo más común es utilizar 19.2Kbps o 9.6Kbps. Se pueden colocar repetidores de acuerdo a la especificación del RS485. El número máximo de nodos es de dos cuando se utiliza RS232, y de 32 para RS485. Si se utilizan repetidores, se pueden utilizar hasta 250 nodos con el RS485.

La comunicación es de tipo maestro-esclavo punto a punto, con posibilidad de realizar broadcast, y con un solo maestro por red. La cantidad máxima de bytes por mensaje es de 250. No tiene encabezados, pero cuenta con control de error con paridad por byte y un checksum opcional de 8 bits en la versión ASCII, y 16 bits en la versión RTU. El número de reintentos de retransmisión está determinado por el software.

Permite el uso de E/S analógicas, en cualquier combinación analógica-digital de entradas y salidas con un máximo de 250 bytes para entradas y 250 bytes para salidas. Dependiendo del funcionamiento particular de los esclavos, es posible utilizar entradas y salidas síncronas.

ETHERNET

El protocolo ETHERNET es muy utilizado y conocido. Es el más utilizado en las redes locales para PC. Está reglamentado en la norma IEEE 802.3. Existen las versiones en cable coaxial: 10 base 2 y 10 base 5; y también las versiones en cable de par trenzado: 10 base T y 100 base TX. Debido a la complejidad del hardware necesario para manejar el protocolo, no es posible utilizar sensores y actuadores directamente. Sin embargo, sí es posible utilizar entradas y salidas remotas. Permite la comunicación entre controladores y equipos inteligentes. Las variantes más utilizadas son la 10 base T y la 100 base TX. Es muy utilizado en aplicaciones de automatización de oficinas y en los más altos niveles de automatización industrial.

No hay un estándar que permita determinar su estabilidad, puesto que todavía está en desarrollo. La mayoría de las PC's modernas lo tienen integrado y los drivers vienen incluidos con Windows y Linux.

Se utiliza un cable de 4 hilos de par trenzado, con la posibilidad de usar cableado redundante. Su topología es de estrella para 100 base TX y 10 base T. El protocolo eléctrico es específico para Ethernet. La longitud máxima sin repetidores es de 100 metros para 100 base T y 10 base T. La velocidad de transmisión es 100 Mbps. y 10 Mbps. respectivamente. Se pueden usar repetidores, sin embargo, lo más común es usar hubs y switches. La longitud máxima es teóricamente ilimitada con repetidores y prácticamente no hay límites en la cantidad de nodos que se pueden conectar. También se pueden usar multimaestros pero los niveles más altos del protocolo lo pueden limitar.

Los mensajes pueden ser del tipo punto a punto o broadcast. Dependiendo del software se pueden usar estrategias de tokenbus o maestro-esclavo. La máxima cantidad de datos a transmitir es de 1500 bytes y la mínima es de 46 bytes. Cada paquete tiene un encabezado de 38 bytes, y el mensaje mínimo es de 672 bits. La corrección de fallas se realiza mediante 32 bits de CRC, y la cantidad de reintentos es controlada por los niveles superiores del protocolo.

Con Ethernet es posible conectar E/S analógicas, y utilizar hasta 1500 bytes para datos de entradas, y 1500 bytes para datos de salidas por cada nodo en cualquier combinación analógico-digital. Los niveles superiores del protocolo pueden permitir el uso de E/S síncronas.

AT

El bus AT (Extended ISA, o EISA) tuvo mucha popularidad desde la década de los ochentas. Con la fuerte ubicación en el mercado de las computadoras personales compatibles con IBM, una gran cantidad de periféricos se fabricaron para este tipo de bus. Es una ampliación a 32 bits del bus ISA, que

era sólo de 16 bits, y que a su vez fue un desarrollo basado en el bus original de la IBM-PC.

Originalmente, el bus de la PC operaba a 4.77 Mbytes/s. Fue en 1982, cuando este bus se mejoró, incrementando su velocidad a 16 Mbytes/s y tomó el nombre de ISA (Estándar industrial de arquitectura, por sus siglas en inglés). Otro bus derivado de éste fue el VL-Bus (VESA Local Bus).

Este bus consiste en un grupo de señales que son conectadas punto a punto en paralelo en cada ranura disponible, y controladas por un maestro. Es posible implementar comunicación entre dispositivos inteligentes, mediante niveles superiores al protocolo físico. Un grupo de señales permiten realizar interrupciones al maestro, para realizar tareas de alta prioridad. Dentro del bus existe un grupo de señales para indicar la dirección de hardware que se desea acceder, otro grupo de señales bidireccionales para los datos (16 bits), y señales de sincronización. En cada periférico se decodifican las direcciones, de manera que todos pueden cambiarse de ranura sin que esto afecte su funcionamiento.

Con este tipo de bus es posible manejar directamente salidas y entradas, dada la sencillez del hardware necesario. La cantidad de señales está limitada por la integración de la tarjetas utilizadas, pudiendo éstas ser inteligentes y realizar tareas complejas como el procesamiento de señales digitales, y la adquisición de datos con filtros, entre otras.

Su velocidad está limitada a 8MBps (megabytes), puesto que fue desarrollado hace mucho tiempo, y siempre se pretendió que los dispositivos nuevos fueran compatibles con todos los anteriores.

PCI

A principios de la década de los 90's, Intel introdujo al mercado un nuevo estándar llamado PCI (Pheripheral Component Interconnect). Básicamente consiste en una mezcla de las características del VL-Bus y del ISA.

El bus PCI provee acceso directo a la memoria del sistema por parte de los dispositivos conectados, y utiliza hardware intermedio para su conexión al CPU. Puede alojar hasta cinco componentes externos. Dicho hardware regula también la velocidad del bus independientemente de la velocidad del procesador principal del sistema.

Originalmente, el bus PCI operaba a 33Mhz, y 32 bits por palabra. Posteriormente se hicieron revisiones al estándar para incrementar la velocidad a 66Mhz, y duplicar la cantidad de bits. Existe una variante, el PCI-X, que maneja transferencias de datos a una velocidad de 133Mhz, permitiendo transmisiones de 1Gbyte/s.

Utiliza 47 señales para conectarse con cada tarjeta de periféricos. Si se trata de una tarjeta maestra – aquella que puede manejar el bus PCI sin intervención del CPU- se utilizan 49 señales.

El bus PCI no permite el uso directo de entradas y salidas, puesto que requiere de hardware complejo para el manejo del protocolo.

USB

El puerto USB nació con la finalidad de facilitar el uso de periféricos en equipos de cómputo. Provee una forma fácil y estandarizada para conectar hasta ciento veintisiete dispositivos a una computadora.

Si se trata de un nuevo dispositivo, el sistema operativo lo detecta y pide al usuario el controlador del dispositivo. Si el dispositivo ya ha sido instalado, la

computadora lo activa y comienza a comunicarse con el mismo. Este tipo de puerto permite la conexión y desconexión de los dispositivos en cualquier momento.

Se utilizan dos tipos de conectores (A y B). El conector tipo A se utiliza en la computadora, y el conector tipo B en los dispositivos periféricos.

El USB permite una longitud de cinco metros en sus cables, pero se puede extender hasta treinta metros con el uso de hubs. La velocidad máxima de transmisión es de 480 Mbps. con el estándar 2.0. La transmisión de los datos se hace a través de un par trenzado.

El puerto lleva también alimentación (5VDC) y tierra. Muchos dispositivos de bajo consumo de corriente toman corriente de ésta alimentación. Otros, que requieren de mayor capacidad de corriente, tienen su propia fuente de poder.

Cuando el maestro (la PC) enciende, hace un barrido de todos los dispositivos que se encuentran conectados al bus, y les asigna una dirección a cada uno. A este proceso se le llama enumeración, y también ocurre cuando se conecta un nuevo dispositivo al bus.

La transferencia de datos se puede dar en cuatro tipos de paquetes:

- Interrupción. Dispositivos como teclados o ratones, que envían poca información.
- Paquetes grandes. Dispositivos como impresoras, que transmiten o reciben mucha información. Estos paquetes se envían en bloques de sesenta y cuatro bytes.
- Isocrónica. Dispositivos que requieren de comunicación en tiempo real.
- Paquetes de control. Para enviar comandos o pedir información a los periféricos.

Mientras los dispositivos son enumerados, el maestro detecta el ancho de banda total que ocupan las transmisiones isocrónicas y de interrupción. A este tipo de transmisiones se les puede asignar hasta un noventa por ciento del ancho de banda total. Una vez que ha sido asignada esa cantidad, se niega el acceso a cualquier otro dispositivo isocrónico o de interrupción. Los paquetes grandes y de control utilizan el ancho de banda restante.

Para asignar el ancho de banda, el USB divide el tiempo en franjas de un milisegundo aproximadamente, llevando cada una mil quinientos bytes. Dentro de esas franjas, los dispositivos de interrupción e isocrónicos siempre reciben atención para garantizar el ancho de banda que requieren, y el resto de la franja es utilizado por los otros paquetes.

El estándar 2.0 para USB salió en Abril del 2000. Provee ancho de banda adicional para aplicaciones multimedia y de almacenamiento de datos, y tiene una velocidad de transmisión unas cuarenta veces más rápida que el USB 1.1. Soporta tres modos de transmisión: 1.5, 12 y 480 MBps. Con estas velocidades, permite usar varias aplicaciones y soportar varios dispositivos de alto desempeño al mismo tiempo.

2.4 Descripción del sistema propuesto

A continuación se describe el diseño de un sistema modular para control de maquinaria que se propone para ofrecer una solución a los problemas expuestos anteriormente. Existen varios aspectos que se deben considerar dentro del diseño de un control de este tipo, y también se detallan a continuación.

Para poder desarrollar un sistema que sea suficientemente flexible, y que se pueda utilizar en la gran mayoría de las máquinas existentes en la industria, se requiere diseñar tarjetas que manejen las señales más comunes. De esta forma

sería posible manejar diferentes tipos de maquinaria con el mismo sistema, haciendo cambios en el software en la interfase del usuario y el programa de la secuencia a seguir. Este control tendría en su sección de control secuencial un funcionamiento equivalente al de un Controlador Lógico Programable (PLC por sus siglas en inglés), pero con una arquitectura abierta, que permitirla modificar el sistema, y con una interfaz gráfica en comunicación con la aplicación de control. Se podrían agregar o quitar funciones según sea necesario, y diseñar nuevas tarjetas para cubrir necesidades que surjan en el futuro, ya sea para algún otro tipo de máquinas, o para mejorar el diseño implementado en este proyecto.

El sistema propuesto tiene por procesador central una computadora personal (PC), dadas las ventajas inherentes a este tipo de equipo y que se exponen a continuación.

Una de las ventajas más evidentes, es que se trata de un equipo fácil de expandir, con una capacidad de procesamiento que supera las necesidades de muchas aplicaciones industriales, y con una gran capacidad de almacenamiento de datos en memoria no volátil (disco). Aunado a esto se agrega el atractivo de poder editar en la misma máquina los programas de la secuencia de operación de la maquinaria y, si es necesario, las interfaces del usuario (siempre que se tengan los códigos fuente instalados en la misma).

El uso de una PC como el controlador de la máquina agrupa también una serie de ventajas interesantes: la facilidad de mantenimiento, el bajo costo de reposición y actualización, y la gran variedad de proveedores que existen en el mercado para este tipo de equipo. Además existe una gran cantidad de equipo en la industria que se está desechando, y que para este tipo de aplicaciones es más que suficiente. Cabe mencionar también, que por el tipo de fuente de alimentación utilizada (conmutada) son sistemas muy robustos, con una gran tolerancia a variaciones de voltaje y ruidos en la alimentación eléctrica. Este hecho ha sido probado extensivamente en el ambiente industrial, con proyectos realizados

previamente con este tipo de equipos y que tienen ya varios años trabajando de manera satisfactoria (nuestras observaciones no publicadas).

La implementación de una interfase hombre máquina (MMI por sus siglas en inglés) es muy sencilla con la ayuda de herramientas de programación como los lenguajes visuales, si se trabaja bajo Windows®, o mediante algún otro lenguaje como C, o Pascal, si se trabaja bajo algún sistema operativo de tiempo real del estilo de MS-DOS o en ambiente de texto. La existencia de un monitor y un teclado, así como la facilidad de manejo de archivos inherente a una PC, permite que el programa del usuario sea modificado en el mismo equipo. La figura 2.1 muèstra el esquema básico del sistema propuesto.

Figura 2.1 Esquema del sistema propuesto

Cuando se requiere de alta velocidad de procesamiento, se puede incluso trabajar con aplicaciones programadas en lenguaje ensamblador. Sin embargo, el rápido avance tecnológico por parte de los fabricantes de procesadores permite cada día más, preocuparse menos por estos aspectos y trabajar con lenguajes de más alto nivel sin que esto tenga repercusiones en la velocidad del sistema; superando en la gran mayoría de los casos los requerimientos de las aplicaciones, manteniendo costos conservadores.

Junto con la gran capacidad de procesamiento de datos, las PC's tienen el atractivo de proveer una muy alta capacidad de almacenamiento en memoria no

volátil (discos duros). Esto es especialmente útil si se desea hacer control estadístico de procesos. En los sistemas autónomos como los PLC's, esta es una capacidad que, si existe, se encuentra fuertemente limitada, obligando a los desarrolladores a invertir mucho tiempo en optimizar sus algoritmos y guardar solamente la información. Los discos duros existentes actualmente para las PC's están hechos a la medida de las necesidades de sistemas operativos cuyas aplicaciones gráficas y de diseño ocupan mucho espacio. Esto representa para las aplicaciones industriales - que comparativamente son mucho menos demandantes en este aspecto - un espacio más que suficiente para almacenar datos (en la mayoría de los casos). Aún así, si es insuficiente, es posible expandirla a un costo muy bajo comparado con lo que costaría tener la misma capacidad con un PLC equivalente.

Para conectar las tarjetas se utilizará un protocolo similar al de un bus tipo AT, en el cual las conexiones son comunes, de manera que todos los pines con el mismo número dentro de los conectores del bus están conectados uno con otro. Esto hace que las tarjetas puedan ser colocadas en cualquier ranura sin necesidad de ser reprogramadas o configuradas por el usuario. La forma de diferenciar el direccionamiento entre una y otra tarjeta es una línea única para cada ranura (la única excepción dentro del bus). Con ésta línea cada tarjeta será capaz de saber cuando la PC se está comunicando con ella. Cada tarjeta se conectará a las señales provenientes de la maquinaria, y a los actuadores de la misma. La figura 2.2 muestra éste esquema de conexión de manera general.

El método que la PC utilizará para comunicarse con las tarjetas será a través del puerto paralelo. El uso de este puerto representa varias ventajas. Entre ellas está el hecho de que se encuentra presente en cualquier computadora PC, por antigua que sea. Y su protocolo estándar de comunicaciones es soportado incluso por las primeras máquinas que se fabricaron, y muy sencillo de programar. Sin embargo, si se desean tener direccionamiento y datos simultáneamente con este puerto, se requiere el uso de circuitería adicional para poder demultiplexar las

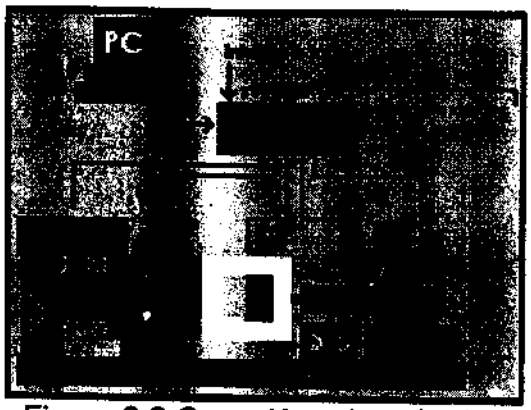

Figura 2.2 Conexión a la máquina

líneas de direccionamiento de las líneas de datos. Así mismo, su capacidad de entrada es de solamente 5 líneas, por lo que los datos de entrada también se tienen que dividir en dos lecturas de 4 bits cada una. Al requerir circuitería adicional para manejar todas estas señales, es necesario agregar software que controle dicha circuitería, y esto representa un tiempo adicional que llevará a la computadora el manejar la tarjeta (sobrecarga/overhead).

La tarjeta que contiene toda esta circuitería se inserta en la ranura principal de un rack que contiene espacios para más tarjetas. En dichos espacios o ranuras el usuario colocará las tarjetas del tipo que requiera la aplicación que desea implementar, según los tipos de señales de entrada y salida que tenga la máquina donde se va a instalar el equipo. Esto brindará al usuario una gran flexibilidad, y la capacidad de adaptarse a muchos tipos de maquinaria diferente. Entre las tarjetas que se diseñarán se encuentran las de entradas y salidas digitales para corriente alterna de 127/220V, y las de entradas y salidas analógicas en el rango de 0-10V, así como las de entradas para lectura de temperatura con termopar. Con éstos diseños se cubre una gran parte de los casos encontrados en la industria.

Para que la PC pueda comunicarse con cada una de las tarjetas y lea o escriba la información pertinente a ellas, es necesario que realice una serie de operaciones especiales para ese tipo de tarjeta. Por lo mismo, es necesario que el software sepa el tipo de tarjeta que se encuentra en cada ranura. Esto lo

configurará el usuario en un archivo de texto que contendrá un código para cada tipo de tarjeta que se encuentre instalada.

2.5 Arquitectura Abierta

Para poder sentar una base adecuada para futuras investigaciones, se diseñará el proyecto con base en una arquitectura de hardware "abierta". Se describirá el protocolo interno de comunicaciones del hardware con la PC, de manera que sea posible diseñar una nueva tarjeta e integrarla al sistema agregando al software únicamente el controlador específico para dicha tarjeta, tomando en cuenta ideas de trabajos anteriores en el área (Ruiz et al. 1993).

Este controlador consistirá de las siguientes partes:

- Rutinas de acceso a los datos internos de la tarjeta: entradas, salidas, y direccionamiento, así como el manejo de las líneas de sincronización (handshaking).
- Rutinas de manejo de los datos obtenidos de la tarjeta dentro del programa secuencial: cómo evaluará la computadora los datos dentro de un diagrama de escalera o un programa secuencial.
- Rutinas de codificación de los datos: definición de los códigos asignados para las operaciones válidas con los datos de la tarjeta (matemáticas, lógicas, o de asignación).

La modularidad del sistema permitirá que el equipo se integre a diferentes tipos de sistemas, y a maquinaria con requerimientos diferentes. Mediante el cambio del archivo de configuración, el usuario podrá describir el hardware conectado en el rack de tarjetas, con lo que el software ejecutará las rutinas de los controladores correspondientes a dichas tarjetas.

El diseño de la aplicación gráfica será independiente de la aplicación de control. Esto permite hacer cambios en una sin afectar la otra. La comunicación entre ambos módulos se hará mediante el acceso, desde la aplicación gráfica, de

un objeto público de intercambio de información con acceso a los datos internos de la aplicación de control.

Cualquier cambio en la aplicación de control será transparente para la aplicación gráfica siempre que se dé mantenimiento a los métodos de acceso del objeto de intercambio de información.

Se proveerá además la información para el manejo de la sección de interfaz con el usuario, para que sea eficiente la reutilización de los códigos objeto utilizados para construir campos de datos dentro de la misma, facilitando la construcción de pantallas nuevas que puedan intercambiar información con la aplicación de control.

La división de la aplicación en dos partes y la programación en objetos permiten también que en el futuro se diseñe una interfaz gráfica en diferente plataforma (por ejemplo, bajo Windows) sin cambiar la sección de control.

El diseño del hardware se hará basándose en la operación del bus AT. La implementación de hardware propio para éste tipo de bus representa la conveniencia de brindar una arquitectura abierta al diseño de nuevos equipos, mientras se conserva la ventaja de tener un bajo costo.

En ocasiones es necesario equipar la máquina con algún robot que extraiga las piezas, o incluir el control para inyectar gas al molde para fabricar piezas huecas. La flexibilidad del sistema permite adaptarlo a equipos periféricos como estos, o incluir a una máquina común este tipo de equipo sin que esto represente un alto costo como normalmente sucede. Los diferentes tipos de tarjetas permiten que su implementación sea también tan sencilla como incluir la programación para estos equipos y conectarlos a las tarjetas adecuadas.

En caso de que se desee instalar el equipo en otro tipo de maquinaria, sólo será necesario reconfigurar la interfaz gráfica para que muestre los datos y parámetros del proceso en la forma requerida. La división de la aplicación en dos módulos (interfaz y control) permite cambiar la operación de un módulo sin afectar el otro. El usuario simplemente organizará los campos de datos indicando al compilador la posición, la forma de desplegar el dato, y el registro interno de donde se toma el dato; facilitando la edición de las pantallas existentes y la generación de nuevas pantallas.

La electrónica utilizada en las tarjetas será sencilla para facilitar su reposición cuando sea necesario. Esto también permitirá que la mayor parte de las tarjetas sean reparables por una gran cantidad de técnicos.

III. DESCRIPCIÓN DEL SISTEMA DESARROLLADO

El sistema que se desarrollado combina una serie de tarjetas de hardware con el software que las controla. Las tarjetas de hardware junto con una sección del software, realizan un control secuencial similar al de un PLC, mientras que otra sección del software permite la comunicación entre el usuario y la máquina[3].

Un PLC puede dividirse en tres partes (Petruzella, 1989,2):

- Unidad Central de Procesamiento. Es donde se llevan a cabo las operaciones lógicas que permiten llevar a cabo la secuencia de control que se desea programar. Generalmente es un sistema basado en microprocesador, que contiene funciones especiales como temporizadores, contadores, y secuenciadores. La forma más común de programarlos es mediante lógica de escalera. El CPU también se encarga de manejar el hardware de entradas y salidas para obtener información de las entradas, y generar las señales de salida correspondientes, después de procesar el programa.
- Módulos de entrada y salida. Son módulos que permiten acondicionar señales de sensores y elementos de entrada como transductores, botónes, interruptores de límite, selectores, etc, y elementos de salida como arrancadores de motores, electroválvulas, y luces indicadoras.
- Dispositivo de programación. Se utiliza para introducir el programa y configurar el PLC para realizar las operaciones deseadas con la maquinaria.

A continuación se describe el funcionamiento de cada tarjeta, así como el funcionamiento del software.

[3] En inglés se utilizan los términos Man-Machine Interface (MMI), o Human-Machine Interface (HMI).

3.1 Comunicación con las tarjetas

La comunicación entre la PC y las tarjetas de entradas y salidas se lleva a cabo a través de la interfaz de comunicación. En éste proyecto se utiliza el puerto de impresión en modo estándar para llevar a cabo la comunicación.

La interfaz de comunicación con las tarjetas es una pieza clave dentro del proyecto. El protocolo de las señales utilizadas permite que el diseño de esta tarjeta se pueda modificar para comunicarse con la PC a través de algún otro puerto como una tarjeta interna en un puerto ISA o PCI, un puerto USB, o un puerto serie.

Las señales involucradas en el proceso de comunicación se pueden dividir en los siguientes grupos:

- Señales de datos
- Señales de direccionamiento
- Señales de sincronización

Las señales de datos consisten en ocho líneas bidireccionales de colector abierto. Para la transmisión de datos hacia las tarjetas, la interfaz de comunicación habilita sus salidas, y las tarjetas de entradas y salidas se mantienen en alta impedancia. Cuando los datos fluyen de las tarjetas hacia la interfaz, ésta mantiene sus salidas en alta impedancia y la tarjeta que manda la información activa sus salidas. Debido a que las salidas son del tipo de colector abierto, con resistencias de pull-up a 5V, la activación de dos o más tarjetas al mismo tiempo no produciría daños en ninguna de ellas.

Las señales de direccionamiento consisten en ocho líneas de dirección (un byte), y dieciséis líneas de selección de tarjetas, que son generadas por la interfaz de comunicación, y que tienen el objetivo de indicar la tarjeta con la que se establece la transferencia de información, y de seleccionar el registro interno que

se desea acceder en dicha tarjeta. Las dieciseis líneas de selección son decodificadas de los cuatro bits más significativos del byte de dirección. Cada una de éstas líneas llega a una ranura, permitiendo que la tarjeta detecte cuando está siendo direccionada mediante ésta línea. Los cuatro bits menos significativos permiten direccionar hasta dieciseis registros dentro de cada tarjeta. En los casos en que se requieran más de dieciseis registros dentro de una tarjeta, se pueden implementar mediante direccionamiento indirecto (ver descripción de tarjeta de entradas analógicas).

En caso de requerirse, se decodificar independientemente las ocho lineas de direccionamiento e ignorar las líneas de selección de las tarjetas, ya que éstas llegan a todas las ranuras del sistema. En las tarjetas diseñadas, sólo las cuatro líneas menos significativas se decodifican dentro de ellas para determinar el registro interno que se desea acceder.

Las señales de sincronización permiten el intercambio de datos en el sistema, o utilizar en el mismo tarjetas con velocidades inferiores a las que pueden manejarse con el puerto de impresión. Las señales de sincronización son la de lectura (RD), escritura (WR), acuse de recibo (ACK), y reset (RST),

La señal de lectura (RD) le indica a una tarjeta que la PC está esperando un dato. Cuando una tarjeta recibe esta señal en conjunto con la señal de selección de tarjeta, debe decodificar las líneas de direccionamiento y acceder el registro interno correspondiente. Esta señal es activa en un nivel lógico bajo (0) y es generada por la señal C2, pin 16 del puerto de control de la PC.

La señal de escritura (WR) le indica a una tarjeta que la PC está enviando un dato. Una vez que los datos ya se encuentran en las líneas correspondientes, la PC envía la señal de escritura. Las tarjetas responden a esta señal en conjunto con la señal de selección de tarjeta. Esta señal es activa en un nivel lógico bajo (0) y es generada por la señal C3, pin 17 del puerto de control de la PC.

La señal de acuse de recibo (ACK) le indica a la PC cuando una tarjeta ha tomado los datos que se le están enviando (tarjetas de salidas), o cuando ha puesto en las líneas correspondientes los datos que se le han solicitado. Esta señal es de colector abierto, y es común a todas las tarjetas. Cada tarjeta debe utilizar esta señal únicamente cuando sea seleccionada a través de la línea correspondiente, para evitar colisiones de datos en el Bus. Esta señal se incluye para poder conectar al bus tarjetas con periféricos de velocidades menores a las de la interfaz de comunicación. En éste caso, los microcontroladores tienen ciclos de acceso del orden de los cuatro microsegundos. Sin embargo, aunque esto represente una limitante en la velocidad con la que funciona el sistema, permite tener tarjetas con funciones más complejas como las de entradas y salidas analógicas – en las que hay que controlar los convertidores y multiplexar los diferentes canales. Y en el futuro permitirá implementar circuitos inteligentes en las tarjetas que desempeñen funciones de supervisión, detección automática de errores, o comunicación con otros sistemas. En caso de requerir una mayor velocidad de acceso, el uso de circuitos FPGA o CPLD es una opción adecuada. Esta señal se utiliza también para controlar el ciclo de software. En la lectura y escritura de las tarjetas, la computadora ejecuta un número determinado de intentos de comunicarse, y si después de varios intentos no se ha lograd, se ignora la tarjeta en ese ciclo y el programa continúa ejecutándose para evitar que el código se quede estancado en una sección en caso de falla de alguna tarjeta.

La señal de reset (RST) es generada por la tarjeta de interfaz durante un periodo pequeño cuando el voltaje de alimentación llega a la tarjeta. Es una señal activa en nivel lógico alto (1). Una señal complementaria (NRST=/RST) es también generada por la tarjeta de interfaz, y enviada a cada ranura. La figura 3.1 muestra el esquema de bloques de la tarjeta, y la tabla 3.1 resume los grupos de señales y la función de cada una dentro de la tarjeta de interfaz.

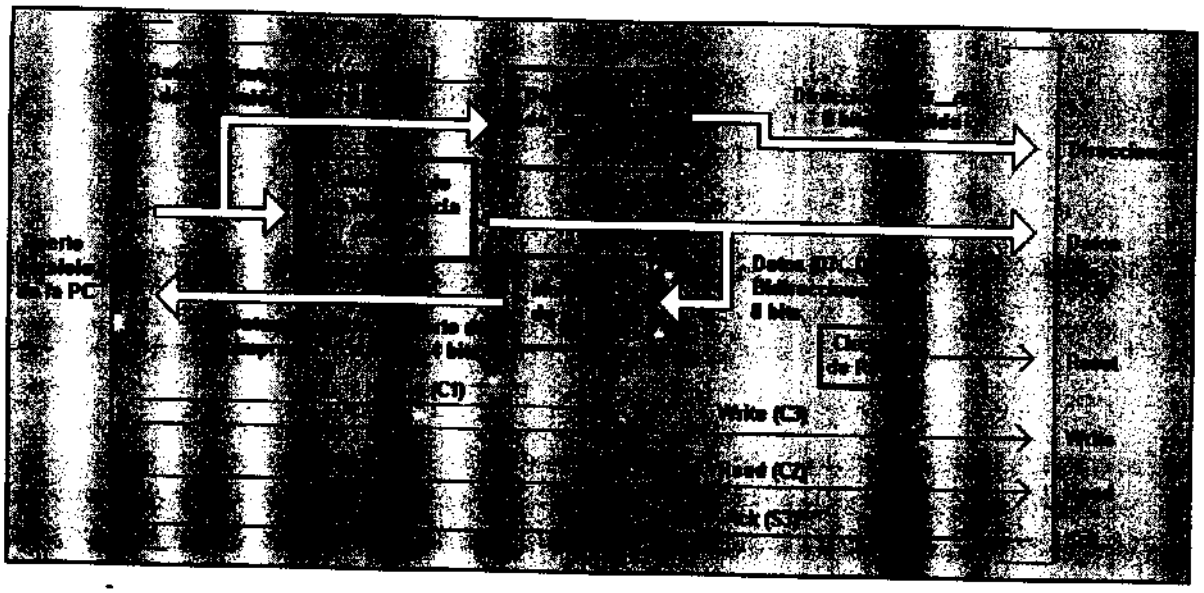

Figura 3.1 Bloques de la tarjeta de interfaz

Tabla 3.1 Grupos de señales y su función

Nomb señal	T al	
D0...D7	Datos	Bus de transferencia bidireccional de datos (colector abierto)
A0...A7	Direcciones	Bus de salida de direcciones demultiplexado del puerto paralelo (puerto de datos)
PCD0...PCD7	Datos de entrada del puerto paralelo de impresión	Bus de entrada de datos del puerto paralelo de la PC (utilizado en modo estándar)
ACK	Sincronización	Recibe en la PC la señal de recepción de las tarjetas (señal S3 del puerto paralelo). En el caso de una lectura, indica que la tarjeta ya colocó en el bus de datos la información solicitada. En el caso de una escritura, indica que la tarjeta ya leyó del bus la información enviada.
S4...S7	Datos de salida al puerto paralelo de impresión	Bus de salida de datos al puerto paralelo de la PC (8 bits multiplexados en dos grupos de 4 bits)
VCC	Alimentación	+5VDC provenientes de la fuente del rack de tarjetas
GND	Alimentación y referencia	Referencia de tierra de la PC, conectada a la referencia de la fuente de alimentación (0 VDC) del rack de tarjetas
RESET	Sincronización	Señal de inicio generada para inicializar todas las tarjetas del rack

BS0...BS15	Direccionamiento	Señales de selección de las tarjetas del rack, generadas a partir de los cuatro bits más significativos de las direcciones (A4...A7)
+12VDC	Alimentación	+12VDC provenientes de la fuente del rack de tarjetas
-12VDC	Alimentación	-12VDC provenientes de la fuente del rack de tarjetas
-5VDC	Alimentación	-5VDC provenientes de la fuente del rack de tarjetas
NIBBLE	Sincronización	Señal utilizada por la PC para seleccionar el grupo de 4 bits que se va a leer de la tarjeta de interfaz (A0...A3 o A4...A7)
READ	Sincronización	Solicita a la tarjeta direccionada que coloque los datos referentes a la lectura en el bus de datos del rack.
WRITE	Sincronización	Informa a la tarjeta direccionada que los datos referentes a la escritura están listos en el bus de datos del rack.

Dentro del módulo de control secuencial, a cada tipo de tarjeta se le asigna un espacio en memoria que permite almacenar los datos asociados con la misma. La cantidad asignada depende del ancho de la palabra de datos que maneje la tarjeta. Para cada palabra que se maneja en la tarjeta existen tres espacios alojados en memoria del mismo tamaño. La primera sección representa los datos de salida que el programa genera para la tarjeta, o los datos de entrada que la tarjeta leyó. La segunda y tercera sección se utilizan para forzar señales.

Las señales forzadas son útiles en la depuración y mantenimiento de un programa para un PLC. Cuando el usuario quiere simular el estado de una entrada para probar alguna parte del programa o de la maquinaria, o si desea encender o apagar una salida para probar el equipo, puede forzar la señal a un estado o un valor determinado. La segunda sección del alojamiento en memoria representa una serie de banderas (una por canal) que indican si la señal ha sido forzada, y la tercera sección indica – si es el caso - el valor al cual ha sido forzada dicha señal. Mientras una señal de entrada se encuentra forzada, el programa la interpreta con el valor al cual ha sido forzada independientemente de su valor real, y si una señal

de salida es forzada, la escritura a la tarjeta toma el valor predeterminado, independientemente del resultado del programa. Aunque aún no se han incluido en el software las rutinas para forzar señales, se reservó la memoria requerida por cada tarjeta para facilitar su implementación futura.

El proceso de comunicación se puede describir de la siguiente forma:

Ciclo de escritura y lectura de datos.

Para llevar a cabo un ciclo de escritura o lectura de datos con una tarjeta, el primer paso que realiza la tarjeta de interfaz consiste en escribir la dirección a la que se desean mandar los datos. Para ello se coloca en el puerto de datos de la PC la dirección que se desea acceder, y se da un ciclo completo (subir y bajar) a la señal ALE (generada a través del puerto de control de la PC). Con esto, las ocho señales correspondientes a la dirección quedan enclavadas en los registros de dirección, independientemente de los cambios en el puerto de datos de la PC. Los cuatro bits más significativos de la dirección representan un número del 0 al 15 que corresponde a una de las ranuras del sistema. Estos cuatro bits se conectan a dos decodificadores 74LS138, con lo que se obtienen dieciseis líneas. Cada una de éstas líneas va conectada a la señal BOARDSELECT (Selección de tarjeta) de una ranura. De esta forma, las tarjetas son accesadas de acuerdo a un rango de direcciones que se detalla en la tabla 3.1.

Tabla 3.2 Rangos de direcciones asignados a cada ranura

Número de ranura	Dirección inicial	Dirección final
0	00	0f
1	10	1f
2	20	2f
3	30	3f
4	40	4f
5	50	5f
6	60	6f
7	70	7f
8	80	8f
9	90	9f

10	a0	af
11	b0	bf
12	c0	cf
13	d0	df
14	e0	ef
15	f0	ff

Con esta combinación se pueden acceder hasta dieciséis registros internos en cada tarjeta, correspondientes a los cuatro bits menos significativos de las líneas de dirección. Los registros internos de cada tarjeta tienen la función de proporcionar o guardar los datos con los que la tarjeta va a trabajar. Para las tarjetas desarrolladas en éste proyecto los registros internos y sus funciones específicas son los que se muestran en la tabla 3.2.

Tabla 3.3 Registros internos de cada tipo de tarjeta

Tipo de tarjeta	Registros internos	Función
Entradas digitales (24 canales)	00 – 02	El registro 00 reporta el estado de las ocho entradas menos significativas (0-7), el registro 01 reporta el estado de las siguientes ocho entradas (8-15) y el registro 02 reporta el estado de las ocho entradas más significativas (16-23)
Salidas digitales (16 canales)	00 – 01	En el registro 00 se escribe el dato para los ocho canales menos significativos (0-7), y en el registro 01 se escribe el dato para los ocho canales más significativos.
Entradas analógicas (16 canales)	00	Se utiliza el registro 00 para escribir la dirección interna del registro que se desea leer. Al leer el registro 00, la tarjeta regresa el valor de la dirección interna escrita anteriormente. Cada par de registros indirectos consecutivos representa el valor de la entrada correspondiente. De éste par,

		la dirección con el valor más alto contiene el byte más significativo del valor de 16 bits. El primer canal ocupa las direcciones 01 y 00, el segundo las direcciones 03 y 02, y así sucesivamente.
Salidas analógicas (8 canales)	00 – 07	Cada registro guarda el valor de la salida correspondiente.
Entradas de termopar (8 canales)	00	La tarjeta se accede de la misma forma que la tarjeta de entradas analógicas, con la diferencia de que sólo existen 8 canales (hasta el registro interno 0f).

Una vez que la dirección ha sido colocada en las líneas correspondientes, es posible realizar un ciclo de lectura o escritura como se indica a continuación:

Ciclo de lectura.

La señal de lectura (RD) cambia a un nivel lógico bajo (0). Dentro de la tarjeta que ha sido seleccionada, las señales BOARDSELECT y RD se combinan para generar una señal interna que genera una interrupción en el controlador de la tarjeta. En seguida, dicho controlador accede los datos del registro indicado por las señales de dirección, y coloca las señales del dato deseado en las líneas correspondientes, y lleva a un nivel lógico bajo la señal de acuse de recibo (ACK). Con ésta señal, le indica al CPU que los datos están listos en el bus, y que los puede leer. El CPU toma los datos, los almacena, y termina el ciclo de lectura elevando la señal de lectura (RD) a un nivel lógico alto (1). En respuesta a este cambio, la tarjeta que se accedió libera la señal de acuse de recibo. En algunas tarjetas - como las de entradas analógicas – las señales de dirección son ignoradas, y la dirección de acceso se determina mediante un registro interno (direccionamiento indirecto, detallado en la descripción de la tarjeta de entradas analógicas).

Puesto que el puerto de impresión de una PC puede únicamente tener señales de salida en el puerto de datos cuando trabaja en modo estándar (el modo para el que está diseñada la tarjeta de interfaz), es necesario recibir los datos a través del puerto de estatus. Este puerto tiene la limitante de poder manejar solamente cinco bits al mismo tiempo. Uno de ellos es utilizado para recibir la señal de acuse de recibo (señal S3, pin 15 del puerto), lo que nos deja cuatro bits libres para recibir datos.

Con el fin de poder leer los ocho bits de datos del sistema, se utiliza un multiplexor que transmite cuatro bits a la vez, seleccionando los más significativos, o los menos significativos, dependiendo del estado de una señal interna de la tarjeta de interfaz llamada NIBBLE (generada con la señal C1, pin 15 del puerto de control). La figura 3.2 el diagrama de tiempos de las señales de un ciclo de lectura.

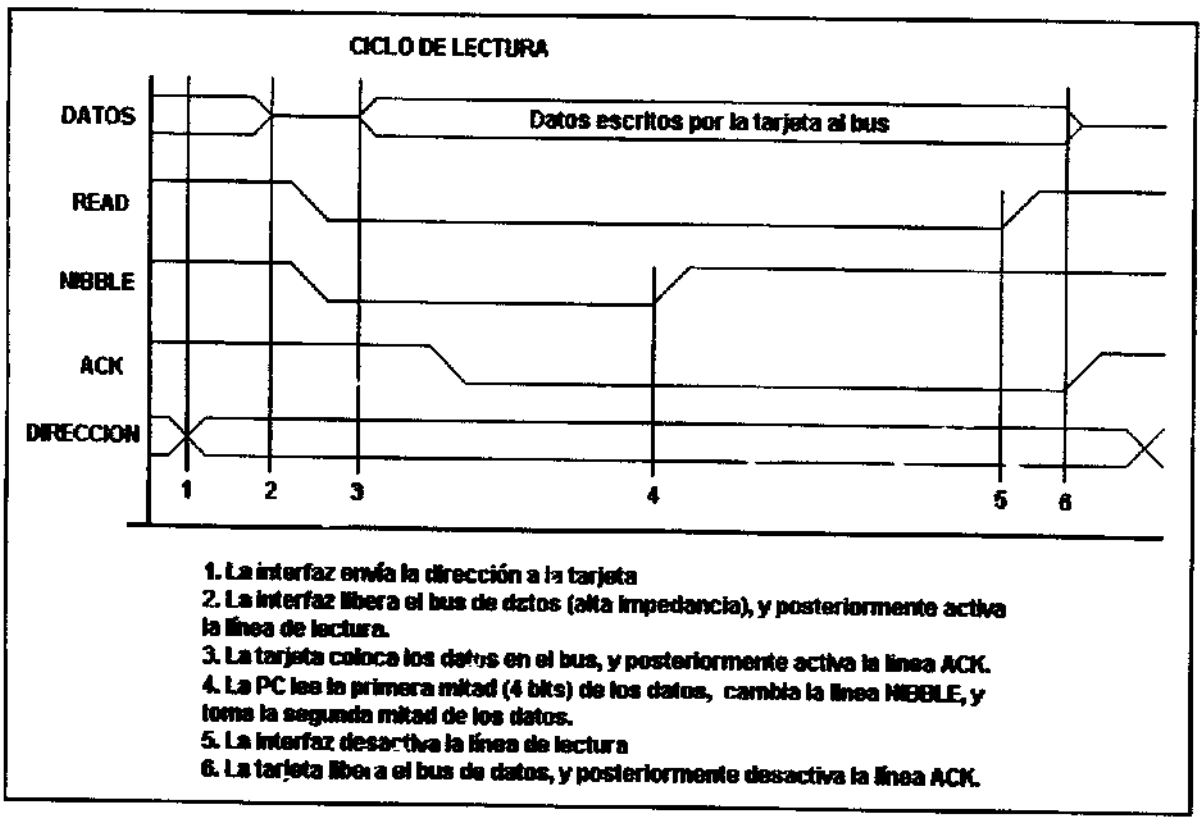

Figura 3.2 Diagrama de tiempos de un ciclo de lectura

Ciclo de escritura.

Una vez colocadas las señales de dirección, la PC envía a su puerto de datos el complemento del dato que desea transmitir a la tarjeta (existe una compuerta de negación entre el puerto de datos de la PC y el bus de datos del sistema). Posteriormente la señal de escritura (WR) cambia a un nivel lógico bajo (0). Esta señal se combina con la señal de selección de tarjeta (BOARDSEL) para generar una señal interna de interrupción en el controlador de la tarjeta que se desea acceder. Esta responde tomando los datos, y generando una señal de acuse de recibo (ACK). Con ésta señal, el CPU puede continuar con el ciclo y regresa la señal WR a un nivel lógico algo (1). Con éste cambio, la tarjeta libera la señal de acuse de recibo, terminando el ciclo de escritura. La figura 3.3 muestra el diagrama de tiempos de un ciclo de escritura.

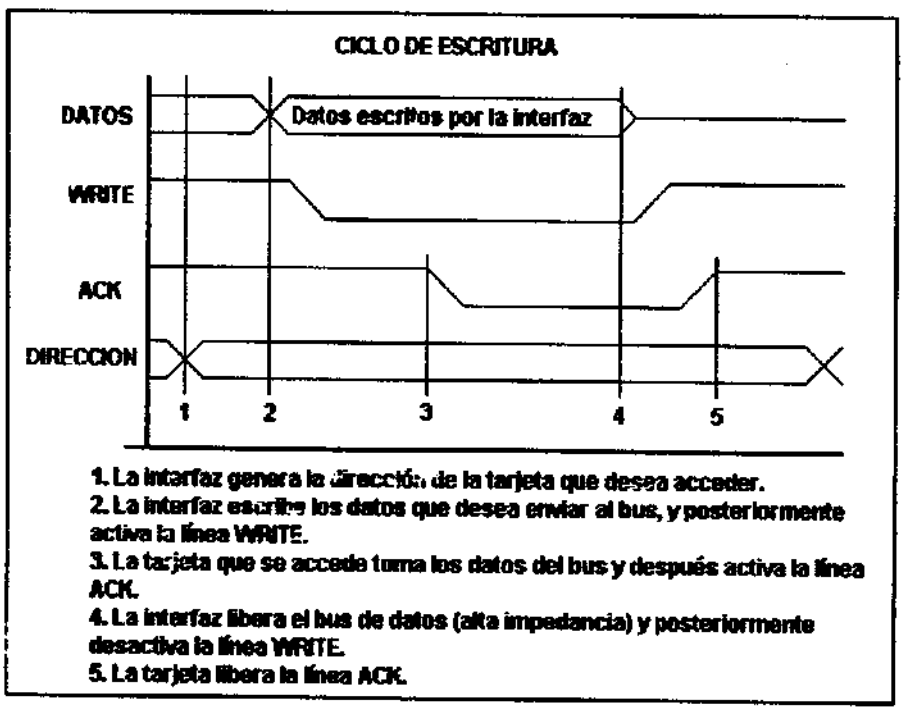

Figura 3.3 Diagrama de tiempos de un ciclo de escritura

3.2 Descripción del Hardware

Para éste proyecto se diseñaron tarjetas de entradas y salidas, tanto para señales analógicas como digitales. La tabla 3.3 resume las características de las tarjetas diseñadas.

Tabla 3.4 Características de las tarjetas diseñadas

Tipo de tarjeta	Número de canales	Características
Entradas digitales (primera versión)	24	Señales de corriente alterna de 120V o de corriente directa de 24V
Salidas digitales	16	Señales de corriente alterna de 120/220V con capacidad de 1.5Amps. por canal, conmutadas por tiristor (Triac) o señales de corriente directa de 24V conmutadas por transistor PNP
Entradas analógicas (primera versión)	16	Señales de voltaje en el rango 0-10V con resolución de 12 bits, multiplexadas a un solo ADC. Velocidad de muestreo aproximada de 0.01 seg./ canal
Salidas analógicas (primera versión)	8	Señales de voltaje en el rango 0-10V con resolución de 8 bits, multiplexadas con circuito de muestreo y retención a un solo DAC.
Entradas de termopar tipo J	8	Señales de termopar tipo J, con compensación de unión fría, multiplexadas a un solo convertidor. Rango de 0-500°C.

3.2.1 Tarjeta de entradas digitales

Para la detección de entradas digitales, se diseñó una tarjeta de 24 canales para recibir señales de 120VCA y 24VCD. Cada canal cuenta con un filtro rectificador y un divisor de voltaje que alimenta la entrada de un optoacoplador con salida de transistor que tiene la función de aislar la etapa de control de la etapa de potencia. La figura 3.4 muestra el diagrama electrónico de la etapa de potencia de cada canal.

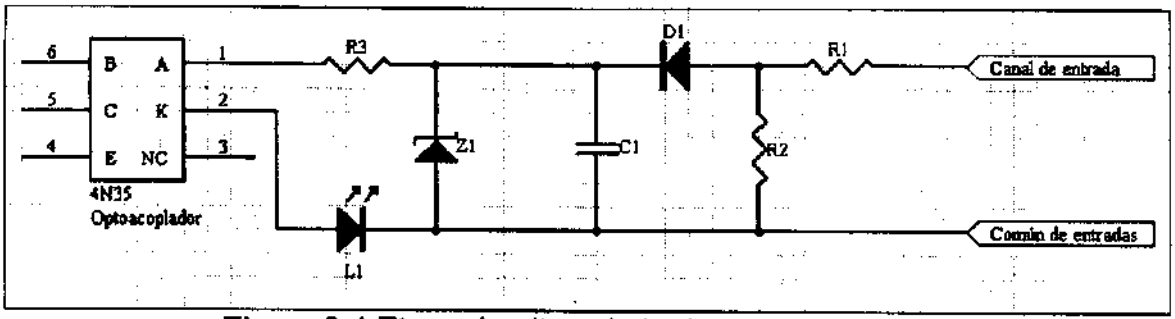

Figura 3.4 Etapa de alto voltaje de entradas digitales

La salida de los optoacopladores de un primer grupo de ocho canales se conecta directamente a un puerto de 8 bits de un microcontrolador de la familia Intel MCS-51, que se encarga de manejar el protocolo de comunicación con el CPU. Los dos grupos restantes de ocho canales son multiplexados con ayuda de transcievers con salida de alta impedancia a un solo puerto de 8 bits del microcontrolador. En la figura 3.5 se muestra el diagrama electrónico de la etapa de control de la tarjeta.

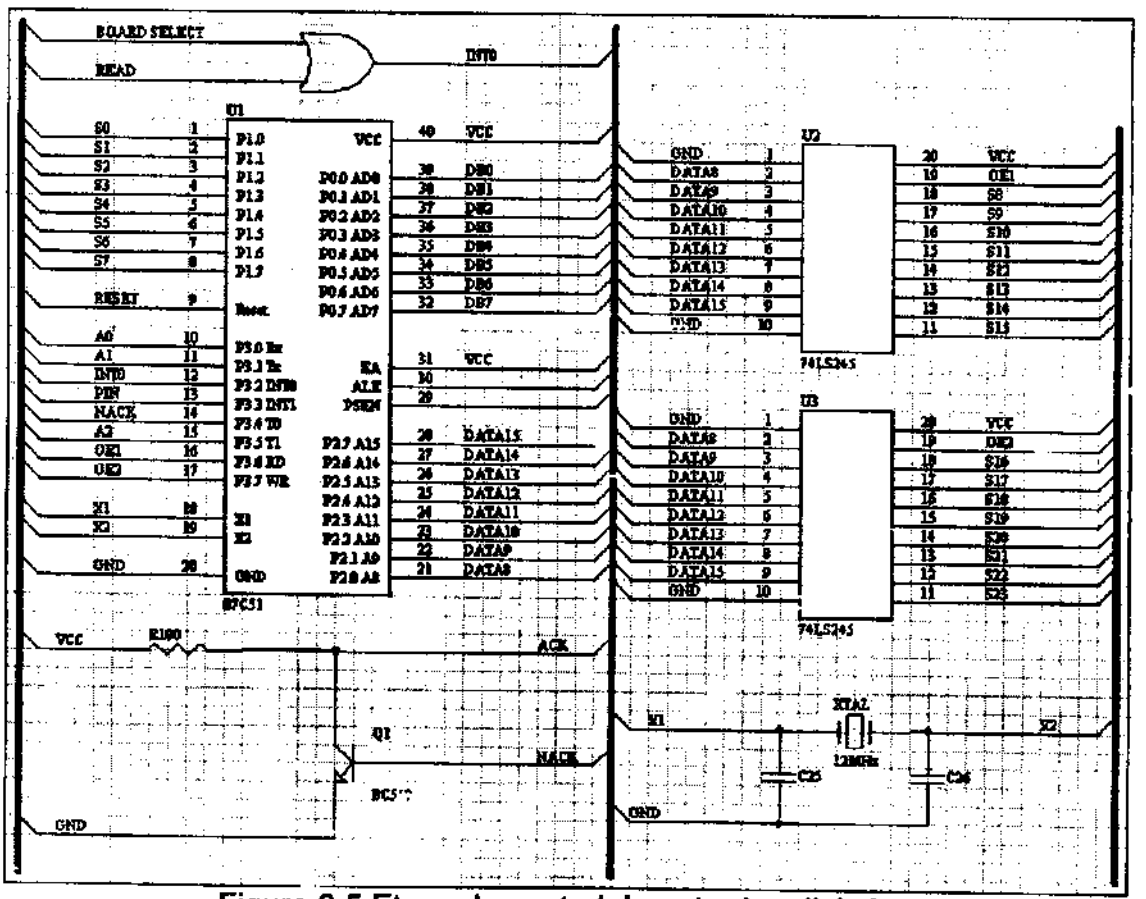

Figura 3.5 Etapa de control de entradas digitales

Para filtrar el rebote característico de las señales que son generadas por interruptores mecánicos, elementos muy comunes en la maquinaria industrial, se realiza un muestreo de los veinticuatro canales aproximadamente cada 20 mseg.

Este tipo de tarjeta ocupa un total de nueve bytes dentro de la memoria del módulo de control. Los primeros tres son utilizados para almacenar el estado de las veinticuatro entradas. Los siguientes seis bytes están reservados para uso en un futuro de las funciones de forzado de las señales.

Las entradas digitales tienen el mismo diseño en 127VAC y corriente directa de 24V. La única diferencia es que para utilizarla en corriente directa se eliminan algunos componentes.

El anexo D contiene el código en ensamblador generado para el microcontrolador de esta tarjeta.

3.2.2 Tarjeta de salidas digitales

Las salidas digitales que se manejan con éste tipo de tarjeta son de corriente alterna de 120/220V conmutadas por un tiristor (Triac), o de 24V de corriente directa conmutadas por transistor PNP, Debido a la gran cantidad de espacio que ocupan los tiristores o transistores, éste tipo de tarjeta maneja sólo dieciséis canales. Cada uno de los canales tiene, al igual que la tarjeta de entradas digitales, un optoacoplador que separa la sección de la lógica de control de las etapas de potencia. La tarjeta se diseñó para una capacidad nominal de aproximadamente 1.5Amps. por canal, aunque se utilizan tiristores y transistores con capacidad mínima de 4 Amps. La figura 3.6 muestra el diagrama electrónico de la etapa de potencia de cada canal en el caso de la tarjeta de corriente alterna, y la figura 3.7 muestra la etapa de potencia en el caso de la tarjeta de corriente

directa. Debido a la separación óptica entre las etapas de la tarjeta, se requiere una fuente de voltaje adicional.

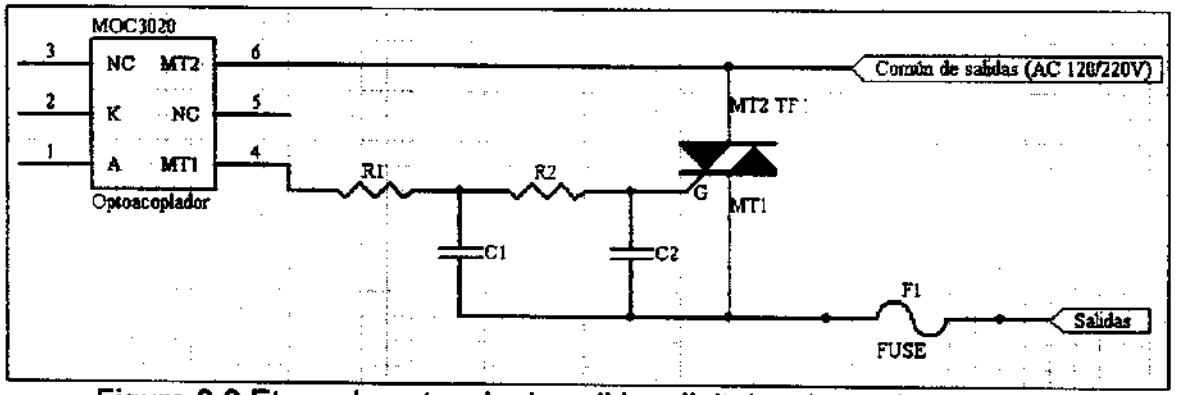

Figura 3.6 Etapa de potencia de salidas digitales de corriente alterna

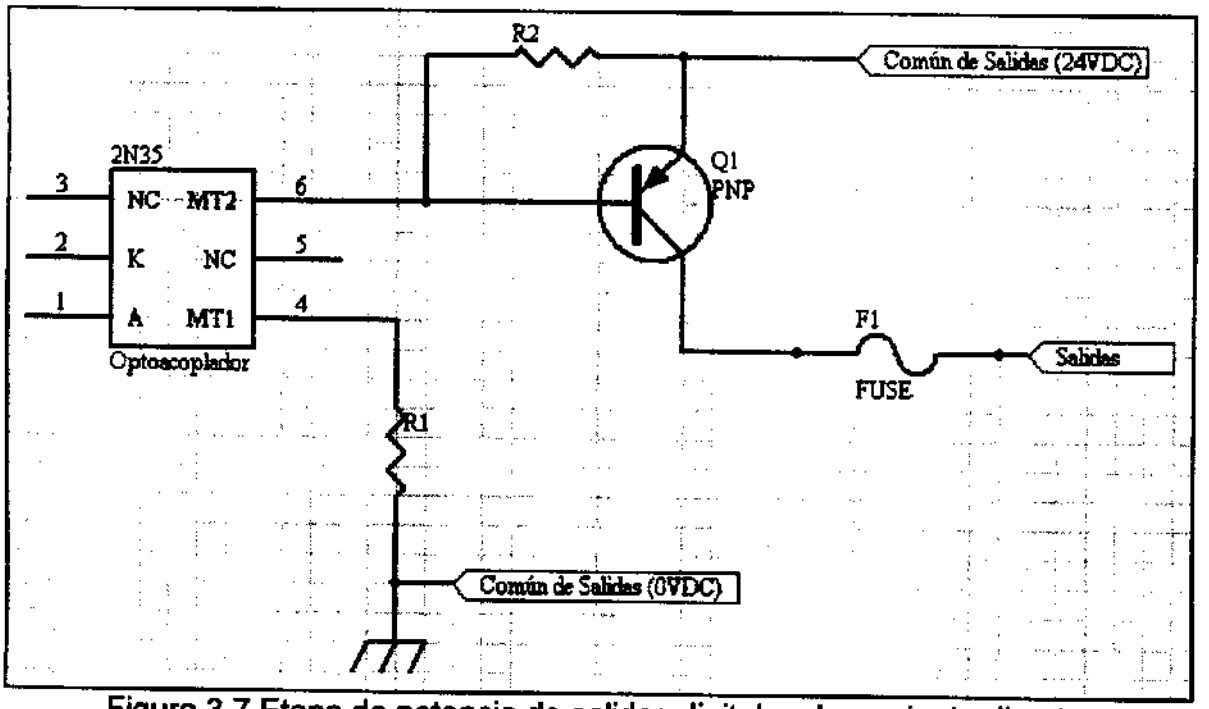

Figura 3.7 Etapa de potencia de salidas digitales de corriente directa

Las entradas de los optoacopladores son controladas directamente por dos puertos de un microcontrolador cuya función es también controlar la comunicación con el CPU. La escritura de los datos a los canales de salida es continua, de tal forma que el tiempo de retardo entre la comunicación con el CPU del sistema y el cambio de las señales de salida sea el menor posible.

El diagrama electrónico de la etapa de control de la tarjeta se muestra en la figura 3.8.

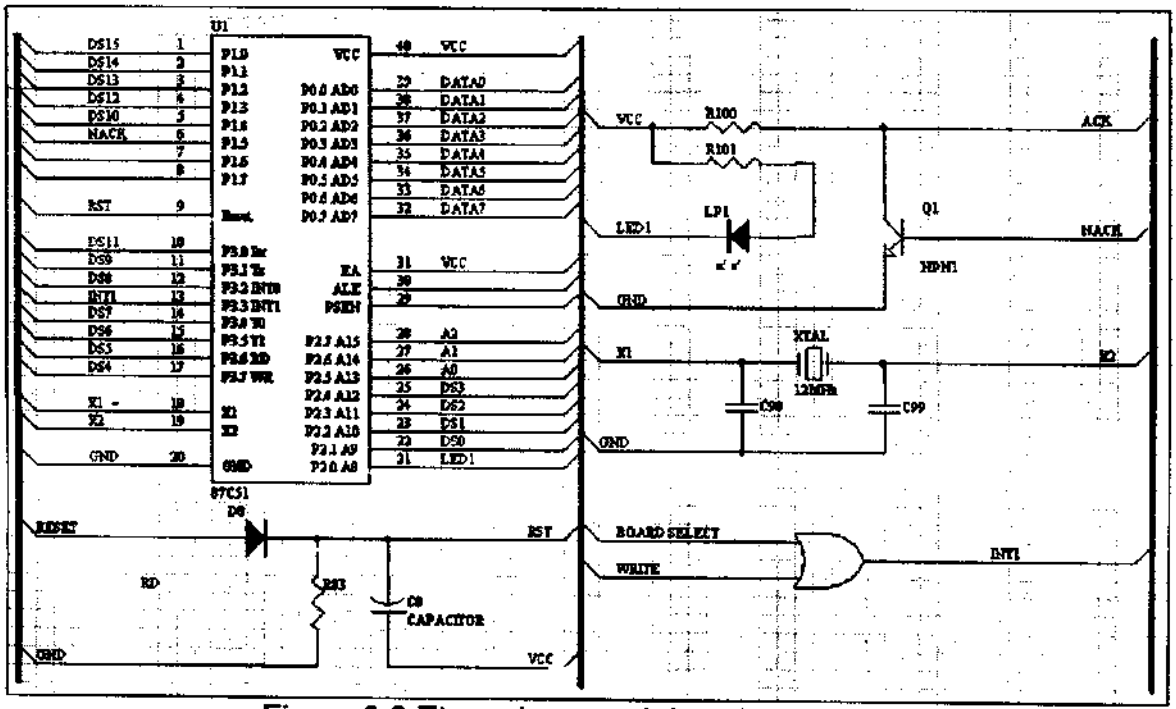

Figura 3.8 Etapa de control de salidas digitales

El anexo D contiene el código en ensamblador generado para el microcontrolador de esta tarjeta.

3.2.3 Tarjeta de entradas analógicas

Las señales de entradas analógicas que maneja este tipo de tarjeta se encuentran en el rango de 0-10V. La tarjeta tiene dieciséis canales de entrada, conectados a un solo convertidor analógico digital (ADC) de 12 bits de resolución. El ADC que se seleccionó es el ADC1240 de Nacional Semiconductors. Este ADC cuenta con funciones de autocalibración y autocero, permitiendo una alta repetibilidad en las mediciones y evitando requerir elementos externos al ADC para calibrar la tarjeta. El criterio principal utilizado para seleccionar este ADC fue la facilidad para conseguirlo con proveedores en México y su bajo costo comparado con circuitos con resoluciones similares (circuitos entre 10 y 14 bits). Con esto se logra una resolución de 2.44mV en el bit menos significativo.

Cada canal tiene un circuito seguidor para presentar una alta impedancia de entrada y evitar alteraciones en la medición. Para evitar la entrada de ruido eléctrico y mediciones aleatorias en los canales no utilizados, cada canal tiene una resistencia de carga del orden de 100K hacia tierra. El OP-AMP utilizado es el TL084, que presenta un bajo offset, una frecuencia de corte de alrededor de 2Mhz y un desempeño satisfactorio a bajo costo además de una alta impedancia a la entrada debido a que contiene transistores tipo JFET en dicha etapa. Para aplicaciones con requerimientos más estrictos se puede utilizar algún otro OP-AMP que cubra dichas especificaciones.

El microcontrolador que incluye el diseño se encarga de manejar la comunicación con el CPU del sistema, y de controlar el multiplexor de los canales y el ADC. La frecuencia de muestreo de cada canal es de 100 ciclos por segundo, y para cada medición se hacen tres lecturas del ADC, que son introducidas a un filtro mediana para reducir el ruido eléctrico. El resultado de la medición filtrada es almacenado en la memoria interna del microcontrolador.

Cada dato se almacena en dos registros contiguos de ocho bits. De ellos, la localidad de memoria con la dirección menor almacena el byte menos significativo de la lectura. Los dieciseis canales generan entonces un total de treinta y dos bytes de información.

Debido a la construcción de la tarjeta, no es posible acceder directamente más de ocho registros internos dentro de una tarjeta (sólo se disñone de las líneas A2...A0), por lo que se requiere hacer un direccionamiento indirecto en la misma. Para llevarlo a cabo, la PC realiza un ciclo de escritura en el registro cero (0) de la tarjeta con la dirección del registro interno que desea leer, y posteriormente realiza un ciclo de lectura. Para leer de la tarjeta el valor de un canal, es necesario leer una localidad par para obtener los ocho bits menos significativos, y unirlos con el valor de la siguiente lectura, que representa los ocho bits más significativos. Dado

que la resolución del convertidor analógico digital es de doce bits, los cuatro bits más significativos siempre son cero.

Por ejemplo, si se desea leer el canal cero, la parte menos significativa de la conversión se encuentra en el registro cero, y la parte más significativa en el registro uno. Habría que escribir un cero en la dirección cero de la tarjeta, y efectuar una lectura de la tarjeta. Posteriormente habría que escribir un uno en la dirección cero de la tarjeta, y después de leer esa misma dirección, efectuar un corrimiento y unir los datos con los de la primera lectura para formar el dato completo. Si se deseara leer el valor del último canal de la tarjeta, habría que leer las direcciones treinta y treinta y uno de la tarjeta.

Por la forma en que se hace este procedimiento, es necesaria una doble lectura y una doble escritura para cada canal. En un futuro se podría implementar una rutina que permita que se incremente automáticamente la dirección interna de la tarjeta con cada operación de lectura, para requerir únicamente de la primera escritura, y después efectuar lecturas consecutivas de la memoria interna de la tarjeta.

La tarjeta proporciona una señal de 10V como referencia para utilizarse con sensores de posición resistivos (potenciómetros).

La figura 3.9 muestra el diagrama electrónico de la tarjeta, y el anexo D contiene el código en ensamblador generado para el microcontrolador de esta tarjeta.

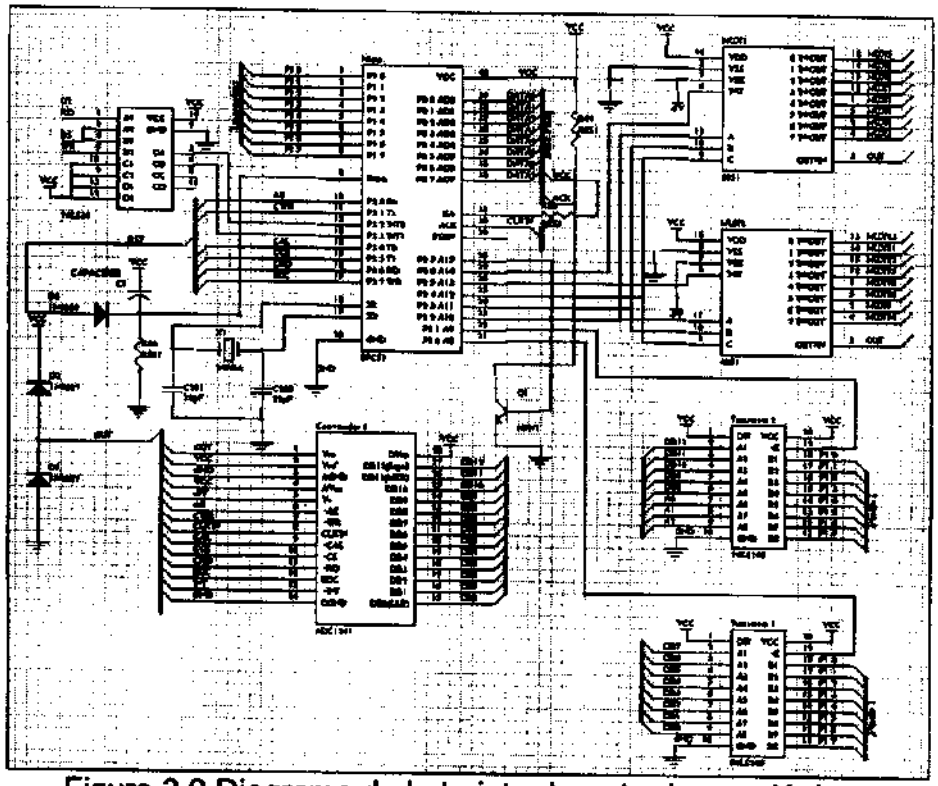

Figura 3.9 Diagrama de la tarjeta de entradas analógicas

3.2.4 Tarjeta de salidas analógicas

Las señales de salidas analógicas se generan dentro de un rango de 0-10V. La tarjeta maneja ocho canales con un solo convertidor digital analógico (DAC) de ocho bits, multiplexando las salidas con circuitos de muestreo y retención individual para cada canal. La multiplexión de las salidas es continua, para poder mantener los voltajes a la salida de cada canal con el menor desajuste posible.

El circuito de muestreo y retención está diseñado a partir de capacitores cerámicos y amplificadores operacionales con entrada JFET. La multiplexión se lleva a cabo con un circuito analógico CMOS CD4051, controlada por el microcontrolador. La conversión digital analógica se realiza con un circuito DAC0800 de ocho bits de resolución, y los circuitos de salida proveen un rango de 0-10V, lo que equivale a un voltaje de 39mV para el bit menos significativo.

Dado que el DAC maneja una resolución de ocho bits, se utilizan ocho registros internos en la tarjeta de un byte cada uno.

La figura 3.10 muestra el diagrama de ésta tarjeta.

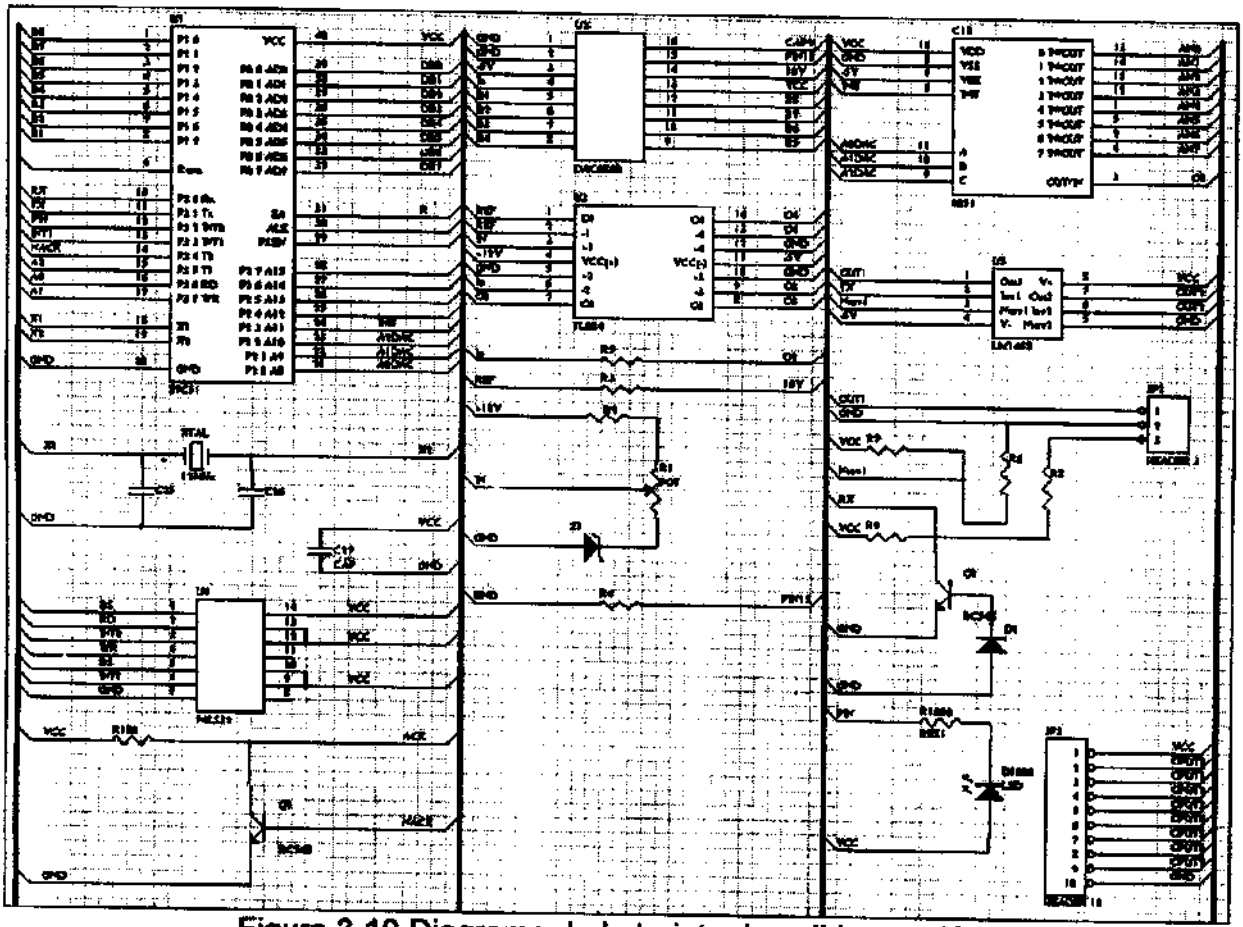

Figura 3.10 Diagrama de la tarjeta de salidas analógicas

3.2.5 Tarjeta de entradas de termopar tipo J

Para la medición y control de temperaturas se diseñó una tarjeta para recibir señales de termopar tipo J con ocho canales. La circuitería de la lógica de la tarjeta es similar a la de la tarjeta de entradas analógicas, utilizando el mismo convertidor analógico digital, pero con diferencias menores que se explican adelante. Esto permite reutilizar una buena parte del diseño de la tarjeta, así como del código para controlar el ADC.

Los termopares de los ocho canales se multiplexan a través de dos circuitos independientes (uno para cada terminal del termopar). Cada uno de estos circuitos tiene su salida conectada a la entrada correspondiente de un amplificador AD596 (terminal positiva y negativa del termopar). Este amplificador se seleccionó

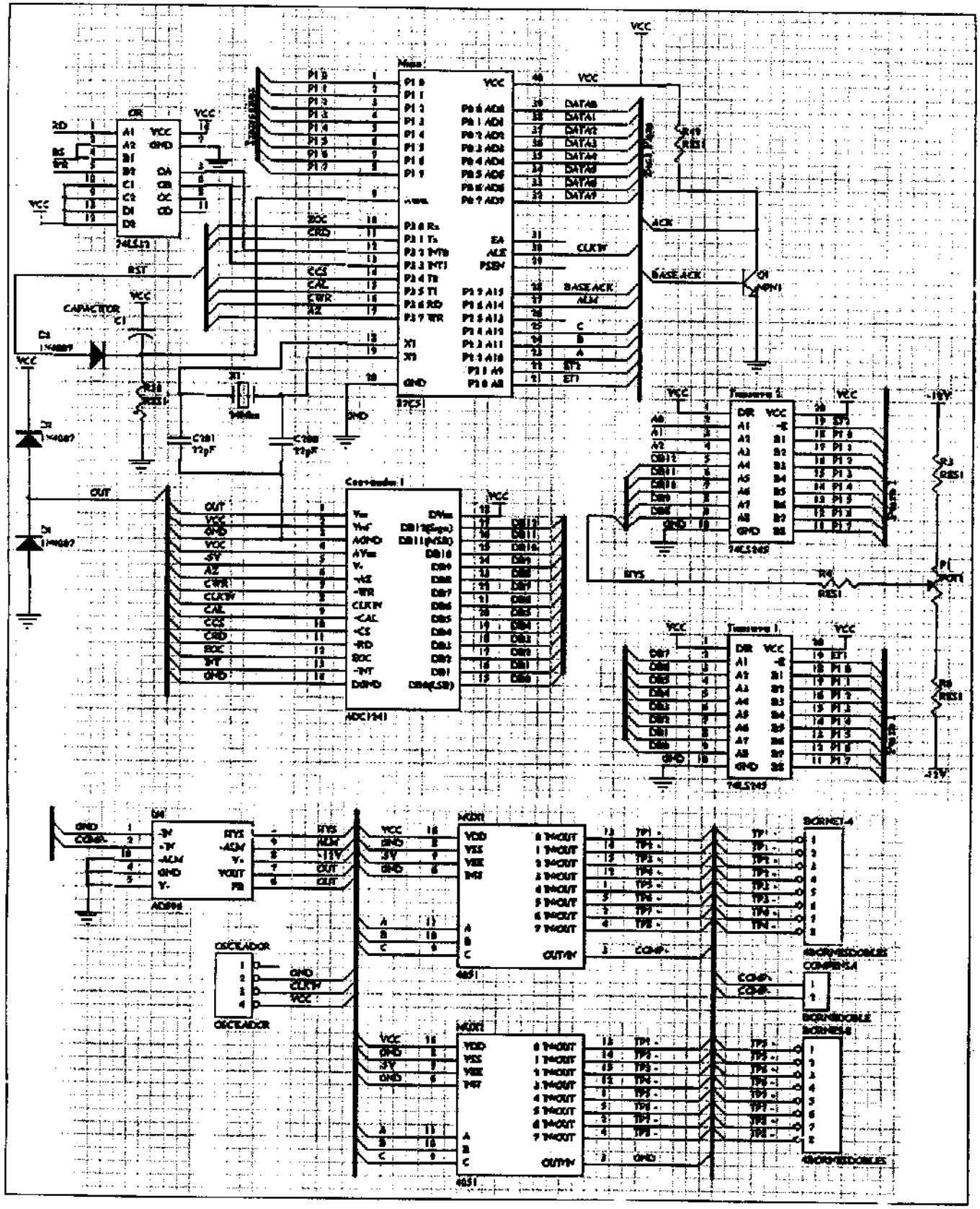

Figura 3.11 Diagrama de la tarjeta de entradas de termopar

debido a que no necesita componentes externos, e incluye ya los circuitos requeridos para compensar la unión fría que se forma al conectar los termopares a la tarjeta, y que se debe considerar con el fin de evitar que la temperatura ambiente afecte la medición. Además, éste amplificador presenta en su salida un voltaje linealizado para este tipo de termopar, y proporcional a la temperatura a razón de 10mV/°C. La salida de este único amplificador para termopar va directamente conectada al convertidor analógico digital. La operación de la tarjeta está contemplada dentro del rango de 0-500°C, lo que representa voltajes de entrada al ADC en el rango de 0-5V. Otra diferencia comparada con la tarjeta de entradas analógicas, es que en ésta tarjeta sólo se implementaron ocho señales (contra dieciseis en la otra tarjeta). Esto implica que cada lectura ocupe dos registros internos de la tarjeta, generando un total de dieciseis registros, direccionados de la misma forma que la tarjeta de entradas analógicas. La figura 3.11 muestra el circuito de multiplexión y el amplificador para termopar de ésta tarjeta.

3.3 Descripción del software

El software de control de la PC se compone de dos módulos. Cada uno de esos módulos tiene una función específica que es independiente del otro, y corre sin adueñarse de los recursos de la máquina. De este modo, cada uno de los módulos trabaja como una tarea no apropiativa. La prioridad de cada módulo es asignada a través de la cantidad de ciclos que ejecuta en un período de tiempo determinado con respecto al otro módulo (porcentaje de ocupación del tiempo).

Los módulos de la aplicación son:

- Módulo de control secuencial
- Módulo de interfaz con el usuario

El módulo con la mayor prioridad es el módulo de control secuencial. Se ejecuta continuamente y se debe procurar que, como mínimo, iguale la velocidad de los temporizadores de la mayoría de los PLC's industriales y se ejecute unas cien veces por segundo (velocidad de escaneo del programa de unos 10 mseg.). Esto se logra con la combinación del equipo utilizado (procesador 486, 586, pentium, etc.) y ajustando la prioridad de los módulos. En las pruebas efectuadas hasta el momento, se han logrado desempeños satisfactorios en máquinas con procesador 486 a 120Mhz y superiores (Pentium II, Pentium III, y Celeron).

3.3.1 Módulo de control secuencial

El módulo de control secuencial trabaja de forma similar a la de un controlador lógico programable (PLC). Existe un programa que determina la forma en que éste genera las salidas dependiendo de variables internas, entradas, y salidas anteriores. El programa se ejecuta continuamente y el usuario lo puede modificar. Lo más común en un PLC es que se programe mediante diagramas de escalera, y se genere a partir del mismo un código objeto que es el que se almacena en la memoria del PLC. En este caso, se utiliza un programa de ecuaciones booleanas que se derivan de un diagrama de escalera, y que se almacena en un archivo en disco. La extracción de las ecuaciones del diagrama es muy sencilla y se detalla más adelante. Aunque la elaboración de un software que permita realizar los dibujos de los diagramas de escalera y extraer automáticamente las ecuaciones de los mismos (código objeto) podría desarrollarse con relativa sencillez, no está contemplada dentro del alcance de este proyecto.

El ciclo de ejecución del PLC se compone de las siguientes cinco partes:

- Escaneo de teclado
- Escaneo de entradas
- Procesamiento del programa
- Escritura de salidas
- Actividades adicionales

En el escaneo de entradas, un ciclo recorre todas las ranuras del sistema, verificando en cada caso el tipo de tarjeta que se configuró en esa posición. Cuando se trata de una tarjeta de entradas, el programa manda llamar la rutina correspondiente a la lectura de hardware para esa tarjeta.

En la escritura de salidas existe un ciclo similar, que manda llamar la rutina de escritura correspondiente cuando se trata de una tarjeta de salidas.

Las actividades adicionales incluyen el manejo de la pantalla, guardar en disco el contenido de la memoria interna del PLC para volver retentivos los datos del mismo, y el control de la interacción del usuario para habilitar el protector de pantalla.

Los registros, banderas, contadores y temporizadores se guardan en un archivo de respaldo cada determinado número de ciclos. Este archivo se carga al inicio del programa para tener en memoria los últimos datos con que trabajó la aplicación, y de esta forma hacer retentivos todos los parámetros de trabajo del PLC.

El software emulador de PLC opera contínuamente su ciclo y sólo puede terminarse con el uso del teclado de la PC (No se puede finalizar el programa mediante el teclado del operador). Esto previene la interrupción accidental del programa por parte del operador, y restringe la operación de la máquina a las funciones permitidas por el programa mediante el teclado de operador.

Previo al ciclo de operación del PLC se realizan actividades para inicializar todos los valores del sistema. Los archivos de definición de señales se cargan a memoria, la última copia de la memoria interna del PLC también se lee de un archivo para reestablecer los datos retentivos. Por último, los

temporizadores y contadores son inicializados, y comienza el ciclo con la lectura a bajo nivel del teclado.

Dado que la aplicación trabajará con maquinaria, es necesario que las rutinas de procesamiento, lectura y escritura, se ejecuten continuamente. Por lo mismo, todo el código esta conformado por rutinas que no detienen la aplicación.

Cada pantalla maneja los códigos tecleados por el usuario de forma inmediata, y sin detener el avance del programa, y cada una de ellas tiene variables no volátiles que le permiten regresar a la sección de código adecuada cuando el usuario está introduciendo datos o editando campos. De manera que cada rutina procesa la tecla, escribe a la pantalla los datos necesarios, y termina, dando lugar a que continúe la ejecución del programa.

Debido a su diseño, es necesario que el manejo del teclado se haga a bajo nivel. El teclado del operador consiste en una serie de botones conectados en una matriz a un controlador de teclado común. Éste teclado es multiplexado junto con el teclado de la PC y conectado al puerto PS2 de la tarjeta madre.

Para facilitar la construcción del teclado del operador, el diseño de la tarjeta consiste en una matriz de once columnas y cuatro renglones, para un total de cuarenta y cinco botones (una de las columnas contiene cinco renglones). Las columnas y renglones se conectan a un controlador de un teclado convencional. Esto permite construir un teclado muy económico, y que se puede manejar con software desde la PC, sin necesidad de tarjetas adicionales. Con éste diseño, los códigos correspondientes a cada tecla pueden variar dependiendo del controlador que se esté utilizando. Para corregir esto, se programaron tablas de códigos, que convierten el código escaneado de un botón a un código fijo asignado para cada tecla.

Una rutina lee los códigos de escaneo y ASCII de la tecla presionada, y copia el estado de cada uno de los botones a la memoria interna del PLC, para que puedan utilizarse dentro de la secuencia de operación de la maquinaria.

Para permitir tanto el uso del teclado del usuario como el teclado de la PC sin necesidad de realizar cambios al código, se incluyeron funciones que permiten habilitar temporalmente el juego de códigos correspondientes al teclado de la PC. Cuando ha transcurrido un tiempo determinado sin interacción del usuario con el teclado de la PC, el juego de códigos del teclado de usuario se reactiva automáticamente.

3.3.2 Programación

Para programar la secuencia de operación se utilizan diferentes elementos dentro del diagrama de escalera, y en consecuencia, dentro del archivo de ecuaciones booleanas. La tabla 3.5 explica éstas herramientas.

Tabla 3.5 Herramientas para programar la secuencia de control

Elementos	Descripción
Entradas y salidas	Señales externas obtenidas o generadas a través de alguna tarjeta. Pueden ser señales digitales (ocupan un solo bit), o analógicas (ocupan uno o dos bytes de información). La cantidad de entradas y salidas depende de las tarjetas que se coloquen en el sistema.
Contadores	Permiten repetir secuencias un cierto número de veces, o llevar la cuenta de veces que ocurre algún evento. El número de contadores es definido por el usuario en un archivo de configuración.
Temporizadores	Controlan eventos que deben regirse por un tiempo determinado. El número de temporizadores es definido por el usuario.
Banderas o relevadores internas	El usuario puede utilizarlas para generar señales internas al igual que un relevador en un tablero. La

	cantidad es definida por el usuario.
Registros internos	Los registros internos son localidades de memoria de dieciseis o de ocho bits que se utilizan para guardar parámetros del proceso. El número de registros es determinado por el usuario. Existen algunos registros con funciones específicas predeterminadas (ver sección sobre registros internos)

El programa secuencial consiste en un archivo de texto que contiene elementos y operadores en forma de ecuaciones booleanas. Estas ecuaciones son convertidas por un programa auxiliar a formato de notación polaca para obtener la inversión del orden de evaluación de los operandos de la ecuación. La razón es que la solución de ecuaciones booleanas con esta notación se puede implementar fácilmente con el uso de una pila. Una vez que las ecuaciones han sido convertidas, se calcula la dirección efectiva en memoria de cada elemento (bit o byte) y se genera un archivo binario con la misma información codificada (código objeto para el PLC).

El archivo se carga al inicio de la aplicación, y su contenido es recorrido durante cada ejecución del ciclo de control secuencial. El PLC interpreta el código objeto, e inserta en la pila la información necesaria para cada operación. Posteriormente llama a las funciones adecuadas para realizar operaciones con dicha información, y finalmente coloca en algún lugar el resultado de las operaciones.

Para calcular la dirección de algún dato dentro de la memoria es necesario conocer la cantidad de temporizadores, contadores, banderas, registros, y el tipo de entradas y salidas utilizadas en el proyecto. Cualquier cambio en estos datos requiere recompilar el programa para calcular de nuevo las direcciones efectivas de todos los elementos de la secuencia.

Los elementos del archivo de programación se describen mediante tres letras que representan un código mnemónico seguidas del número de elemento. Existen varios tipos de elementos, enlistados en la tabla 3.5 (cuando se muestran dos nemónicos dentro de la tabla, su uso es equivalente – e indistinto - en el programa).

Tabla 3.6 Códigos mnemónicos para los elementos de los diagramas

Tipo de elemento	Mnemónico
Señal de entrada (INPUT)	INP
Señal de salida (OUTPUT)	OUT
Alimentación de Temporizador (TIMER ON)	TIM/TON
Relevador temporizado al cierre de un temporizador (TIMER RELAY)	TIM
Incremento de un Contador (COUNTER INCREMENT)	CNT/CNU
Relevador temporizado al final del conteo (COUNTER RELAY)	CNT
Decremento de un Contador (COUNTER DECREMENT)	CND
Reinicio de un Contador (COUNTER RESET)	CNR
Relevador interno o bandera (BANNER)	BAN
Registro de valor de un Temporizador (TIMER VALUE)	TIM
Registro de valor de precarga de un Temporizador (TIMER LOAD)	TLD
Registro de valor de un Contador (COUNTER VALUE)	CNT
Registro de valor de precarga de un Contador (COUNTER LOAD)	CLD
Registro interno de 16 bits	REG
Registro interno de 8 bits	RE8

Además de los elementos anteriores, existen ciertos elementos especiales que representan operaciones con varios elementos, enlistados en la tabla 3.7 (OP1 y OP2 son elementos descritos según la tabla anterior 3.6). Los movimientos de registros son las únicas operaciones de ésta tabla que se utilizan del lado izquierdo de una ecuación. Los demás elementos (comparaciones) regresan un valor booleano como su resultado.

Tabla 3.7 Elementos especiales

Tipo de operación	Mnemónico
Movimiento de registros de 16 bits	RM.OP1.OP2
Movimiento de registros de 8 bits	RM8OP1.OP2
Comparación menor o igual de registros de 16 bits	CLEOP1.OP2

Comparación menor o igual de registros de 8 bits	CL8OP1.OP2
Comparación mayor o igual de registros de 16 bits	CGEOP1.OP2
Comparación mayor o igual de registros de 8 bits	CG8OP1.OP2
Comparación igual de registros de 16 bits	CEQOP1.OP2
Comparación igual de registros de 8 bits	CE8OP1.OP2
Operación de conexión en serie (AND lógico)	*
Operación de conexión en paralelo (OR lógico)	+
Operador de negación	/
Operador de comentario	//

Los operadores de conexión en serie y en paralelo tienen la misma precedencia, y son evaluados conforme son encontrados en las ecuaciones. Para especificar el orden en que deben ser efectuadas las operaciones con igual precedencia, se puede utilizar paréntesis (más adelante se ejemplifica un caso).

A continuación se muestra un ejemplo y se explica la forma en que se implementó la operación de la secuencia del programa.

Tomemos como ejemplo las siguientes líneas:

(1) BAN0=TIM0*INP1+INP2 (La bandera cero enciende cuando la entrada 2 está encendida, o cuando la entrada uno está encendida y el temporizador cero terminó de contar su tiempo preestablecido.)

(2) TIM0=INP3+INP4 (El temporizador cero cuenta su tiempo si la entrada tres, o la entrada cuatro se encuentra encendida.)

La conversión a notación polaca nos da el siguiente resultado:

(1) TIM0 INP1 * INP2 + = BAN0

(2) INP3 INP4 + = TIM0

Para la primera ecuación los pasos serían los siguientes:

- Se inserta el valor del bit (contacto del temporizador) TIM0 en la pila

- Se inserta el valor del bit (contacto de la entrada) INP1 en la pila
- Se ejecuta la operación de multiplicación en la pila con los dos elementos insertados (TIM0 e INP1), el resultado queda como elemento superior en la pila
- Se inserta el valor del bit INP2 en la pila
- Se ejecuta la operación de suma en la pila con los dos elementos superiores
- Se extrae el elemento superior de la pila como resultado
- Se copia dicho valor en la dirección en memoria que almacena la bandera (relevador interno) BAN0.

- Para la segunda ecuación los pasos serían los siguientes:
- Se inserta el valor del bit INP3 en la pila
- Se inserta el valor del bit INP4 en la pila
- Se ejecuta la operación OR en la pila
- Se extrae el resultado de la pila
- Se copia el valor a la localidad de memoria que almacena la variable TIM0

El alojamiento de la memoria dentro del PLC se explica a continuación. Los primeros dieciseis bytes son reservados para funciones especiales en un futuro, y en seguida se enlistan las fórmulas que determinan la localidad de memoria del primer elemento del bloque.

Variable dentro del código	Descripción
TIMINI (16)	Inicio de los temporizadores, valor fijo, después del bloque reservado de memoria
X=TIMINI+5*nTemporizadores	Inicio de los contadores
Y=X+5*nContadores	Inicio de las banderas que mapean el teclado
Ap=nBanderas/8+1	Número de bytes de banderas
A=Y+Ap+8	Inicio de los registros de 8 bits
B=A+nReg8Bit	Inicio de los registros de 16 bits
Bw=nReg16*2	Cantidad de bytes utilizados por los registros de 16 bits

C=B+Bw	Fin del bloque de datos reservados, primera localidad libre dentro de la memoria para los datos de las tarjetas de entradas y salidas

La memoria que se utiliza para los datos de las tarjetas de entradas y salidas depende del tipo de tarjeta que se coloque en cada ranura. Es por ello que se genera un archivo de configuración con extensión CFG, en el que se enlistan estas tarjetas. Su formato se explica más adelante.

Cuando se desea incluir un comentario, se inicia un renglón con el operador de comentario (doble diagonal), y el resto del renglón es ignorado:

//Línea de comentario

Las ecuaciones que utilizan estos elementos se obtienen de un diagrama de escalera. En seguida se ejemplifica su extracción en la figura 3.12 y en la tabla 3.8.

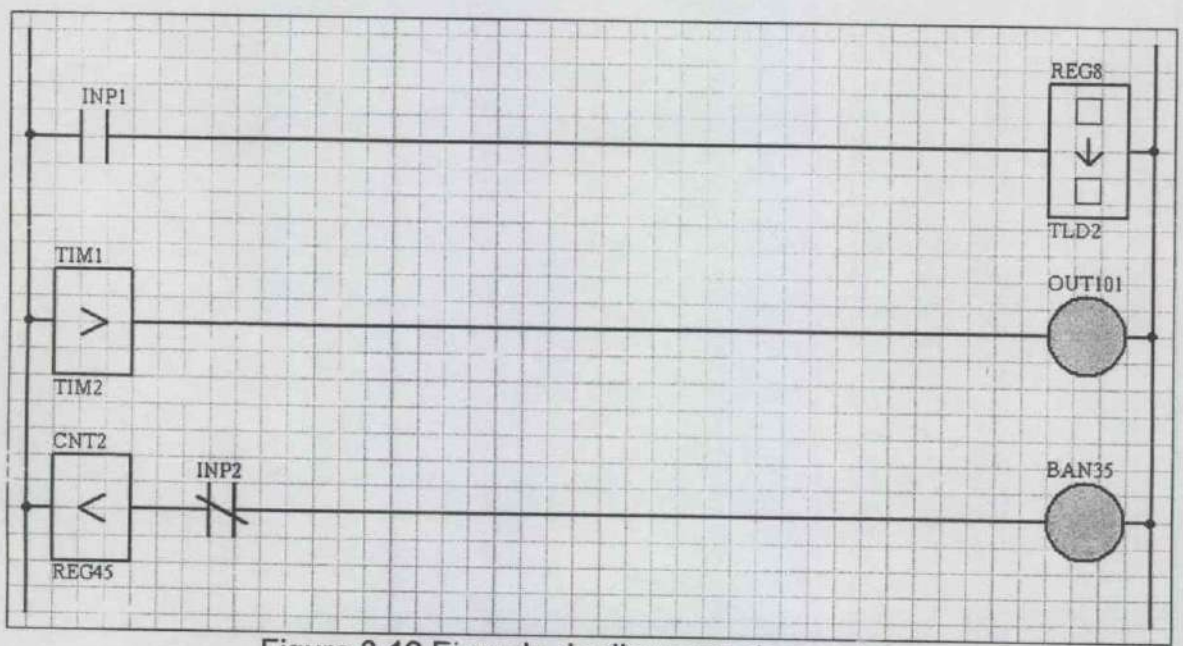

Figura 3.12 Ejemplo de diagrama de escalera

Tabla 3.8 Interpretación del diagrama de la fig. 3.12

Operación	Explicación
RM.REG8TLD2=INP1	Si la entrada 1 se encuentra encendida, mueve el contenido del registro de dieciseis bits no. 8 al registro de precarga del temporizador 2.
OUT101=CGETIM1TIM2	La salida 101 se enciende si el valor actual del temporizador 1 es mayor o igual al valor actual del temporizador 2.
BAN35=CLECNT2REG45*/INP2	La bandera 35 se enciende si suceden LAS DOS siguientes condiciones: El valor del contador número 2 es menor que el contenido del registro 45 y, La entrada 2 está apagada.

Las conexiones en serie se convierten en operaciones de multiplicación lógica (AND), y las conexiones en paralelo se convierten en operaciones de suma lógica (OR). Con éstas dos operaciones y el operador de negación (/) es posible realizar cualquier operación lógica (esto es comprobable a través de las leyes de Morgan).

Un ejemplo concreto de extracción de ecuaciones a partir de un diagrama se muestra en la figura 3.13. Dicha figura muestra una secuencia en la cual el operador inicia una secuencia presionando un botón conectado a la entrada 0. Una vez presionado este botón, la bandera interna 0 se enclava, apagándose de nuevo cuando el temporizador 1 llegue a su fin. Cuando la bandera interna 0 se enclava, el temporizador 0 inicia, y la salida 0 se activa. Una vez que el temporizador 0 llega a su valor predeterminado, se activa el temporizador 1, se apaga la salida 0, y se enciende la salida 1, que dura encendida hasta que el temporizador 1 llega a su valor predeterminado. La extracción de las ecuaciones correspondientes se detalla en la tabla 3.9.

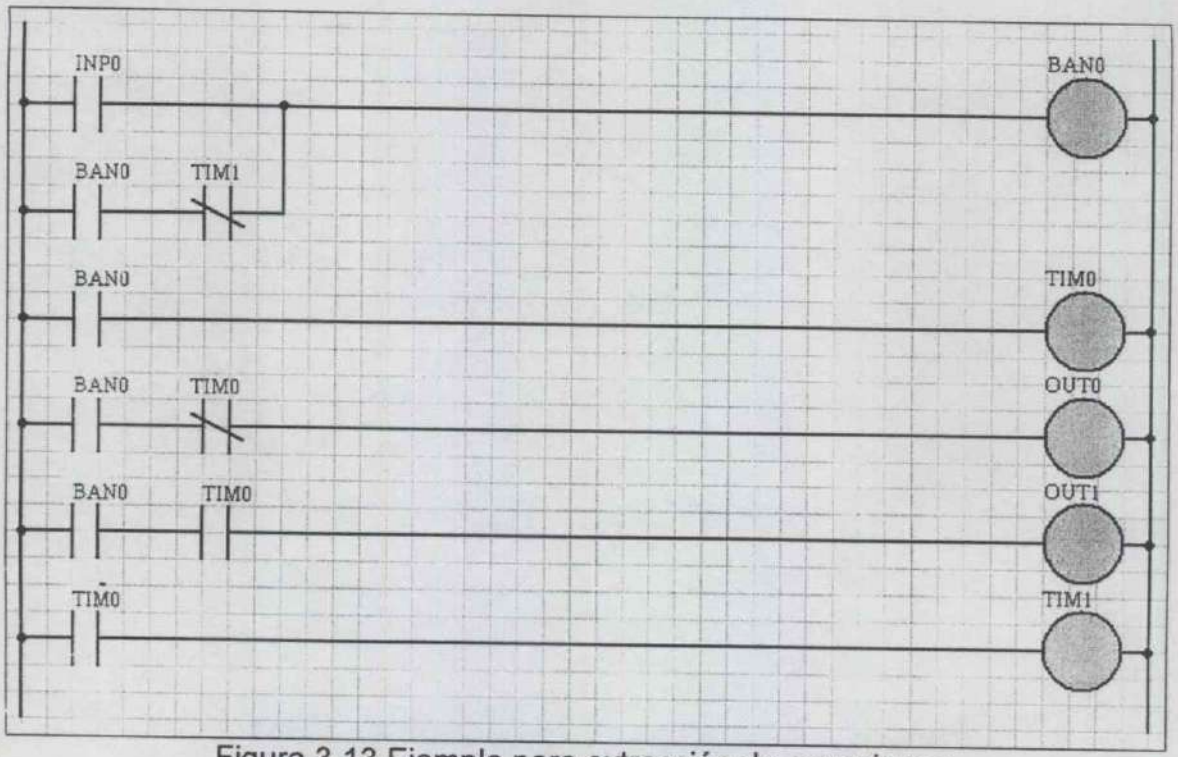

Figura 3.13 Ejemplo para extracción de ecuaciones

Tabla 3.9 Extracción de ecuaciones de la figura 3.13

Operación	Explicación
BAN0=INP0+(BAN0*/TIM1)	La primera parte – entrada 0 – se encuentra en paralelo con la segunda parte – bandera 0 en serie con la negación del Temporizador 1 – y por ello se encuentran sumadas. Dado que el operador de suma y de multiplicación (paralelo y serie, respectivamente) tienen aquí la misma precedencia, la segunda parte se coloca entre paréntesis. Si esto no se hiciera, se evaluaría primero el paralelo de la entrada 0 con la bandera 0, y posteriormente se realizaría a operación en serie con la negación del temporizador 1. El enclavamiento de la bandera cero se realiza utilizándola dentro de su propia rama, en serie con el elemento que la apaga (temporizador 1). Una vez que la entrada 0 es accionada, la bandera cero enciende, y se mantiene así hasta que el temporizador 1 llegue a su tiempo preestablecido.

TIM0=BAN0	La bandera 0 enciende el temporizador 0.
OUT0=BAN0*/TIM0	La salida 0 se mantiene encendida mientras la bandera 0 se encuentra encendida y el temporizador 0 no haya llegado a su tiempo preestablecido.
OUT1=BAN0*TIM0	Una vez que el temporizador 0 llega a su tiempo preestablecido, la salida 0 se apaga, y la salida 1 se enciende.
TIM1=TIM0	Cuando el temporizador 0 llega a su tiempo preestablecido, provoca que el temporizador 1 comience a avanzar, y cuando llegue a su tiempo preestablecido, apagará la bandera 0 (primera rama).

Todos los elementos que se proveen ocupan un espacio en memoria interna. Esta memoria consiste en un arreglo lineal de registros de 1 byte. Estos registros se combinan y seccionan según la cantidad de temporizadores, contadores, entradas, salidas, etc. que defina el usuario. Para definir la cantidad de cada uno de ellos, el usuario utiliza un programa auxiliar de configuración, que genera un archivo de texto con las especificaciones de hardware de la aplicación llamado PLC.CFG, como se mencionó anteriormente. Dicho archivo contiene la siguiente información:

- Cantidad de temporizadores
- Cantidad de contadores
- Cantidad de banderas
- Cantidad de registros de 8 bits
- Cantidad de registros de 16 bits
- Dieciseis números que representan los códigos del tipo de tarjeta utilizado en cada ranura (ver tabla 3.10)

Tabla 3.10 Códigos de tipo de tarjetas

Tipo de tarjeta	Código asignado
Entradas digitales (24 canales)	1
Salidas digitales (16 canales)	101
Entradas analógicas (16 canales)	2

Salidas analógicas (8 canales)	102
Entradas de termopar (8 canales)	103
Entradas digitales (8 canales) marca Allen Bradley Fam. 5	11
Salidas digitales (8 canales) Allen Bradley Fam. 5	111

El espacio que requiere cada tarjeta depende del tamaño de la palabra que maneja la tarjeta, y la cantidad de canales que tiene la misma.

3.3.3 Estructura interna y funcionamiento de los temporizadores

Los temporizadores funcionan en base a un contador que se incrementa con la interrupción del reloj de la PC. Cada ciclo del módulo de control secuencial, los temporizadores son evaluados y actualizados para simular una estructura de hardware similar a la de la figura 3.14.

Figura 3.14 Estructura de temporizadores

Cada temporizador cuenta con una señal de encendido, equivalente al solenoide de un temporizador real. En un diagrama de escalera se representa con un solenoide al que se le llama TONXXX o TIMXXX (ambas opciones son válidas y evaluadas por el software de control de la misma forma), donde XXX es el número de temporizador con el que se está trabajando.

La señal de salida corresponde a un contacto normalmente abierto en un temporizador real. Para evaluar el estado del contacto se utiliza un contacto normalmente abierto común, y se le llama TIMXXX. Mientras la señal de entrada (solenoide) se encuentra apagada, el valor del temporizador (registro TIMXXX) se mantiene en cero.

A pesar de que existen diferentes elementos con el mismo nombre (TIMXXX), el software evalúa el elemento correcto dependiendo de su uso dentro del diagrama de escalera y el archivo de ecuaciones. Una vez que enciende, el valor del temporizador incrementa hasta llegar al valor de un registro llamado precarga (TLDXXX). Cuando el valor del temporizador es igual o mayor al de precarga, la salida se activa, y se mantiene en ese estado hasta que la señal de entrada se apague nuevamente.

La operación interna de los temporizadores consiste en un contador de centésimas de segundo transcurridas desde la última actualización de temporizadores. Este contador es un dato de doble precisión (***double***), y es incrementado por la interrupción del reloj de la PC cada vez que ésta ocurre en una cantidad igual a 1/18*100 seg. Cuando se ejecuta un nuevo ciclo en el módulo de control, una rutina de actualización es llamada.

La rutina de actualización aumenta los valores de los temporizadores que se encuentran encendidos en la cantidad de centésimas de segundo completas – el valor entero del contador – transcurridas desde la última actualización. Con éste método se consigue una resolución máxima en los temporizadores de 55 mseg. El valor de tiempo que se desea tener se especifica en el registro de 16 bits TLDXXX, y el valor del tiempo transcurrido se genera dentro del registro de 16 bits TIMXXX. Ambos registros se almacenan en la memoria interna del módulo de control, cada uno mediante dos registros contiguos de 1 byte. Para el control del temporizador se utilizan las banderas de entrada (TON/TIMXXX), y salida (TIMXXX), que son almacenadas en un byte adicional. De esta manera, cada temporizador utiliza 5 bytes contiguos dentro de la memoria, organizados como se muestra en la figura 3.15.

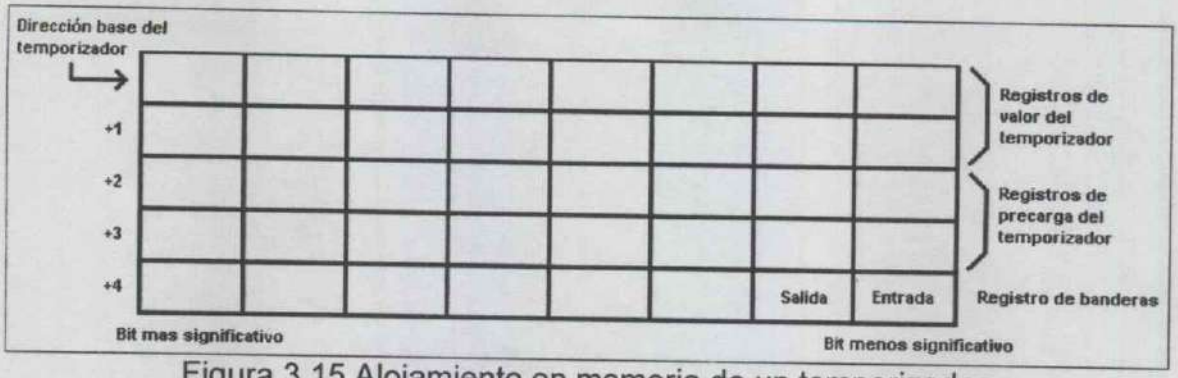

Figura 3.15 Alojamiento en memoria de un temporizador

Dentro de la memoria interna, el software reserva una cantidad determinada de registros para control interno y variables especiales como la fecha y la hora actual. Los temporizadores son colocados después de los registros reservados. La dirección exacta dentro de la memoria para cada temporizador se calcula en base a la siguiente fórmula:

Dbase = TIMINI + 5 * XXX

donde TIMINI es la dirección base del primer temporizador (determinada según la cantidad de registros reservados), y XXX representa el número de temporizador a evaluar.

Una vez calculada la dirección base para el temporizador, existe uno de los siguientes casos:

Se accede el valor actual del temporizador (TIMXXX) utilizándolo como un registro fuente de 16 bits (el usuario no puede cambiar su valor, es controlado internamente). La dirección del registro a acceder es igual a la dirección base.

Se accede el valor del precarga (TLDXXX) utilizándolo como registro fuente o destino de 16 bits. La dirección del registro a acceder es igual a la dirección base mas dos (Dbase+2).

Se accede el bit de encendido del temporizador como destino (no se puede utilizar como fuente). La dirección del registro a acceder es igual a la dirección base mas tres (Dbase+3), y se utiliza el bit 0 (menos significativo).

Se accede el bit de salida del temporizador como fuente (no se puede utilizar como destino). La dirección del registro a acceder es igual a la dirección base mas tres (Dbase+3), y se utiliza el bit 1.

El caso que el software utiliza depende del uso que se le da al temporizador dentro del diagrama de escalera, y consecuentemente, dentro del archivo de ecuaciones. La figura 3.16 muestra los diferentes casos y su representación en el diagrama de escalera.

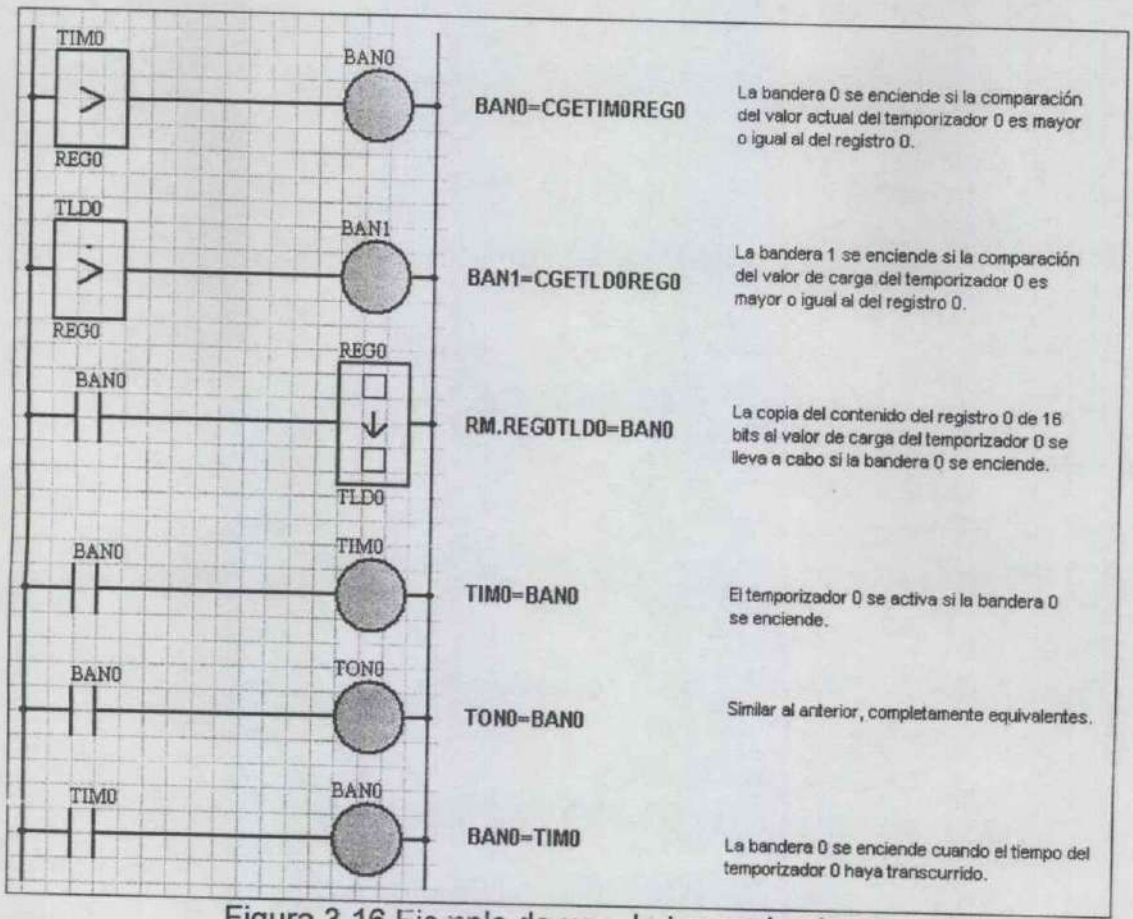

Figura 3.16 Ejemplo de usc de temporizadores

3.3.4 Estructura interna y funcionamiento de los contadores

Los contadores se procesan al inicio de cada ciclo del módulo de control. El diagrama de bloques de los contadores se muestra en la figura 3.17.

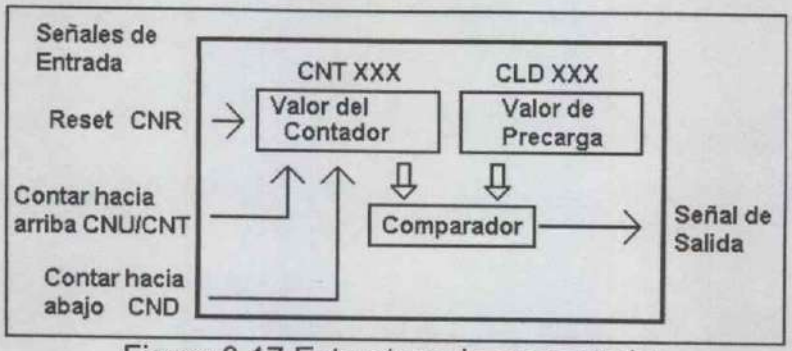

Figura 3.17 Estructura de un contador

En cada contador existen tres señales de entrada. Las tres señales se simbolizan en un diagrama de escalera como solenoides.

Cuando la señal llamada Reset (CNRXXX) se activa, los registros del valor actual del contador se inicializan a cero. Mientras esta señal se mantenga en ese estado, las demás entradas son ignoradas. La entrada de conteo hacia arriba puede utilizarse en un diagrama con el nombre CNUXXX o CNTXXX de forma indistinta (al igual que el caso de TON/TIM). La entrada de conteo hacia abajo se identifica con el código CNDXXX. Estas dos entradas producen el cambio correspondiente en los registros del valor del contador cuando tienen una transición de un valor lógico bajo (0) a un valor lógico alto (1). El estado de las entradas ascendente y descendente es monitoreado y almacenado en cada ciclo para detectar los cambios, de forma que sólo se incrementa o decrementa cuando la transición es positiva.

La señal de salida se evalúa en forma de contacto. Su valor lógico es alto cuando se encuentra activada (normalmente abierto) y se identifica con el código CNTXXX dentro del diagrama y el programa del usuario.

De la misma forma que en los temporizadores, cada uno de los diferentes usos del mismo código (CNT) es evaluado según su posición dentro del diagrama para determinar el registro o bandera correctos que se deben acceder.

El valor del registro de precarga (CLDXXX, 16 bits) es comparado en cada ciclo contra el registro del valor actual (CNTXXX, 16 bits) después del incremento o decremento necesario y genera la salida del contador. El registro de precarga se debe llenar con el valor del número de eventos que se desean contar. Estos registros se almacenan en la memoria interna de la PC en dos registros de 1 byte contiguos cada uno.

Además de los registros de precarga y valor actual, se requiere de un registro adicional para almacenar el estado actual y anterior de las entradas de conteo ascendente y descendente en cada ciclo. Sin el estado anterior, sería imposible detectar la transición.

En total, cada contador ocupa 5 bytes contiguos que se organizan como muestra la figura 3.18.

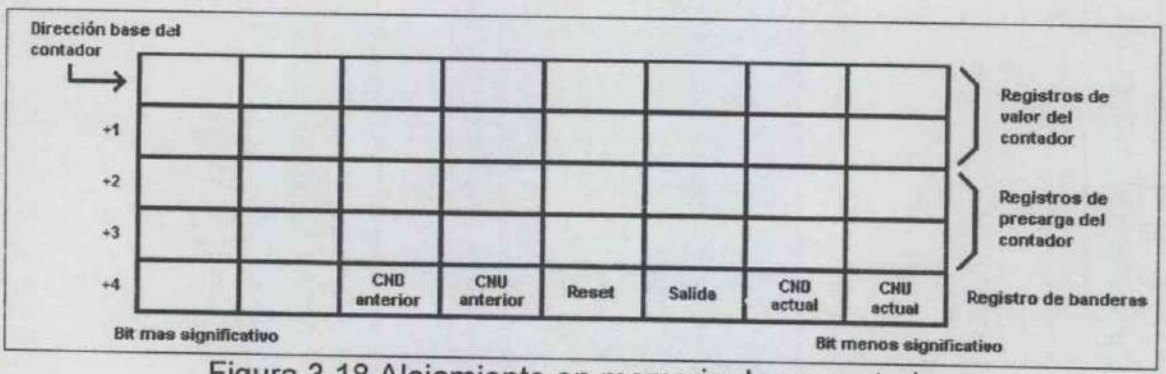

Figura 3.18 Alojamiento en memoria de un contador

La dirección de la memoria base para cada contador es calculada en la siguiente forma:

Dbase = CNTINI + 5 * XXX

donde CNTINI es la dirección base del primer contador, y XXX representa el número de contador a evaluar. La dirección del primer contador se determina con la fórmula:

CNTINI = TIMINI + 5 * No. de temporizadores

Al igual que con los temporizadores, existen varios casos:

Se accede el valor actual del contador (CNTXXX) como registro fuente de 16 bits. La dirección del registro a acceder es igual a la dirección base.

Se accede el valor del precarga (CLDXXX) como registro fuente o destino de 16 bits. La dirección del registro a acceder es igual a la dirección base mas dos (Dbase+2).

Se accede el bit de reset del contador como destino. La dirección del registro a acceder es igual a la dirección base mas tres (Dbase+3), y se utiliza el bit 3.

Se accede el bit de salida del contador como fuente (no se puede utilizar como destino). La dirección del registro a acceder es igual a la dirección base mas tres (Dbase+3), y se utiliza el bit 2.

Se accede el bit de entrada de conteo ascendente como destino (no se puede utilizar como fuente). La dirección del registro a acceder es igual a la dirección base mas tres (Dbase+3), y se utiliza el bit 0.

Se accede el bit de entrada de conteo descendente como destino (no se puede utilizar como fuente). La dirección del registro a acceder es igual a la dirección base mas tres (Dbase+3), y se utiliza el bit 1.

La figura 3.19 muestra los diferentes casos y su representación en el diagrama de escalera.

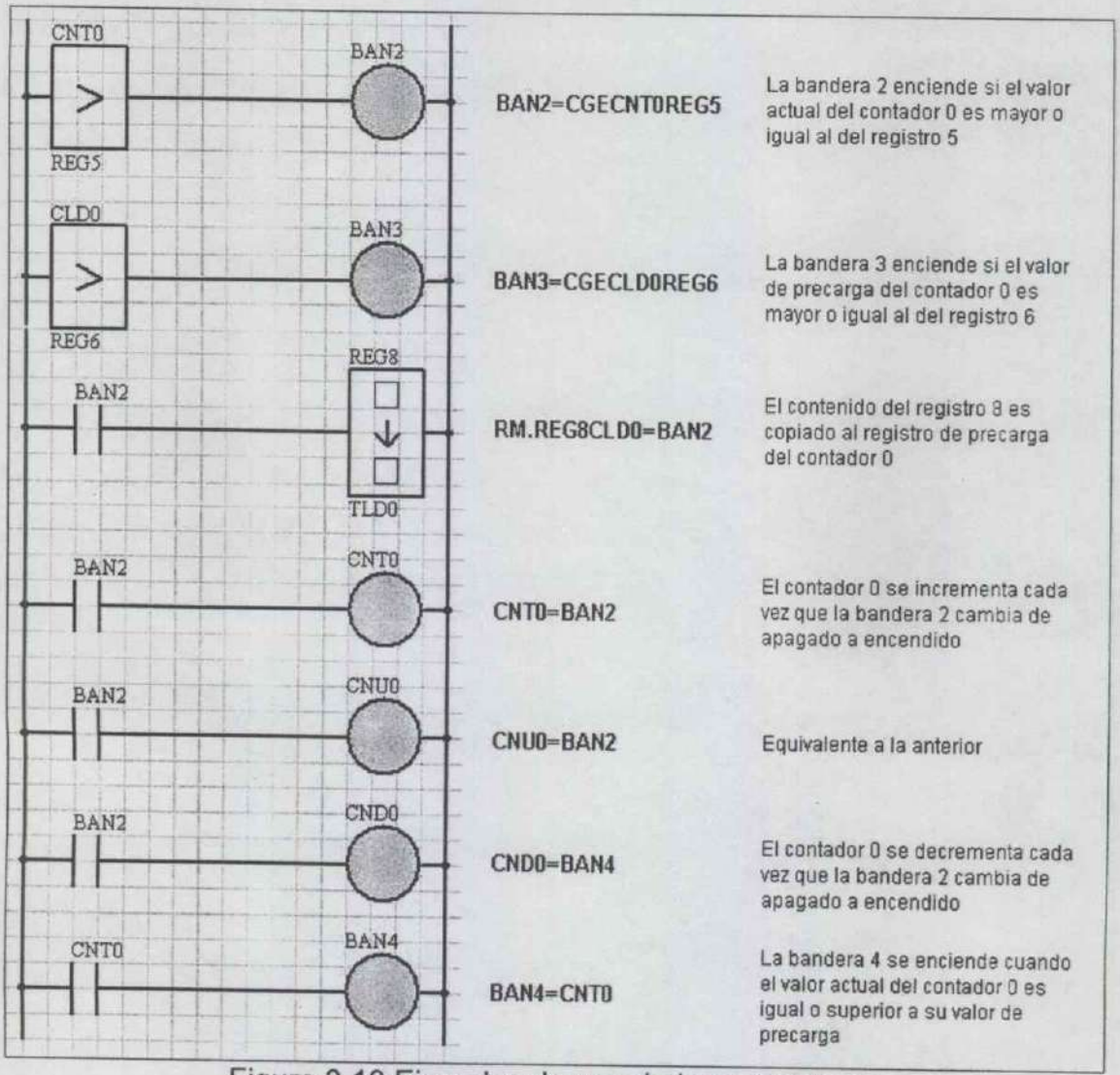

Figura 3.19 Ejemplos de uso de los contadores

3.3.5 Banderas Internas

Es posible que en un diagrama de escalera sea necesario incluir un relevador auxiliar que permita tener varios contactos dependientes de una señal. Esta función es realizada por las banderas internas. Como las banderas solamente existen internamente en el PLC, la cantidad de contactos que se pueden utilizar es ilimitada, y la cantidad total de banderas es limitada solamente por la cantidad de memoria disponible.

Las banderas se alojan en la memoria interna en grupos de ocho bits, ocupando un byte. La cantidad de bytes ocupados por las banderas es:

Bytes = Nb / 8 + 1

donde Nb es el número de banderas asignadas por el usuario. En la operación se toma sólo la parte entera de la división. De ésta forma la cantidad de bytes necesarios es redondeada a la siguiente unidad.

Las primeras 64 banderas (0-63) son reservadas para representar el estado de cada botón del teclado, permitiendo al usuario utilizarlas dentro de su programa. El software se encarga de copiar el estado de los botones a estas banderas. De esta forma, la primera bandera que el usuario puede utilizar es la bandera número 64.

Las banderas se representan en un diagrama de escalera como contactos normalmente abiertos o cerrados, correspondientes a un solenoide común. Un ejemplo del uso de las banderas internas en un diagrama de escalera, así como su explicación y la extracción de las ecuaciones correspondientes se muestra en la figura 3.20.

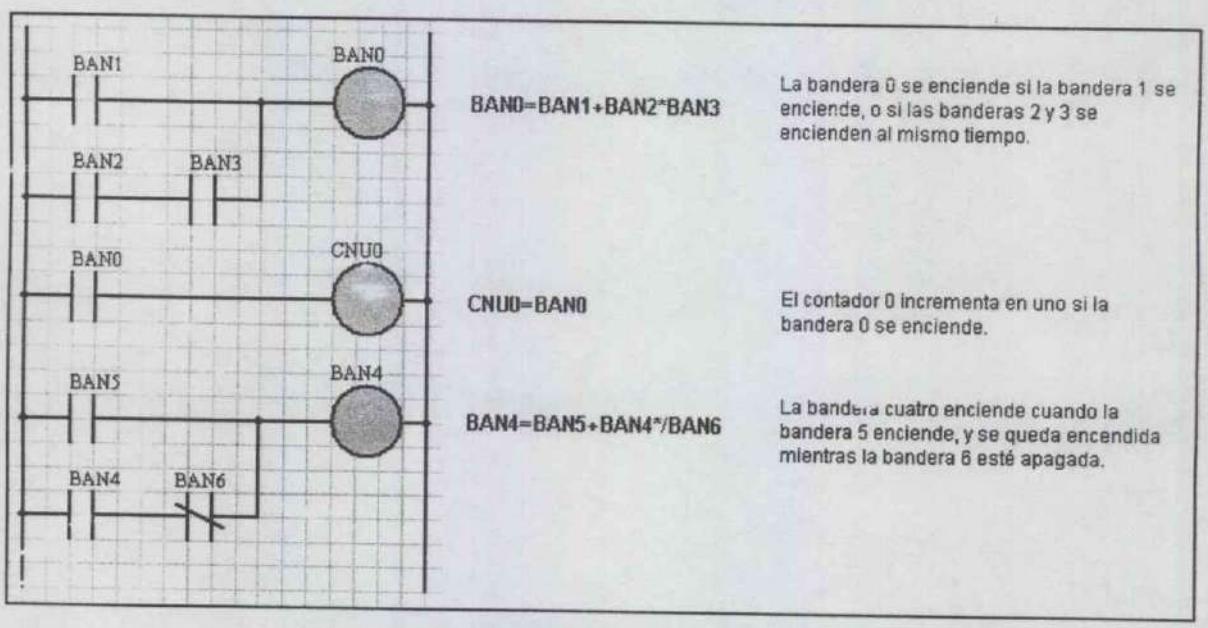

Figura 3.20 Ejemplo de uso de banderas internas

3.3.6 Registros Internos

Los registros internos son localidades de memoria de uno o dos bytes, que puede utilizar el usuario para almacenar información. Para un optimizar el uso de la memoria, se definieron campos de ocho y campos de dieciseis bits. El usuario especifica la cantidad de registros de cada tipo, y la memoria correspondiente es alojada.

Al programar el uso de registros dentro de una aplicación, el usuario determina si utiliza un registro de dieciseis o de ocho bits mediante el elemento correspondiente: REG para registros de dieciseis bits, y RE8 para registros de ocho bits.

Los registros se utilizan para guardar datos de la aplicación, opciones y parámetros seleccionados por el usuario, variables que cambian durante el proceso, etc. En el diagrama de escalera se utilizan con las instrucciones de movimiento y comparación de datos. La figura 3.21 muestra un ejemplo del uso de los registros de dieciseis bits, junto con la extracción correspondiente de ecuaciones, y una breve explicación de cada rama. El uso de los registros de ocho bits es equivalente, excepto por el uso de RE8 en lugar de REG.

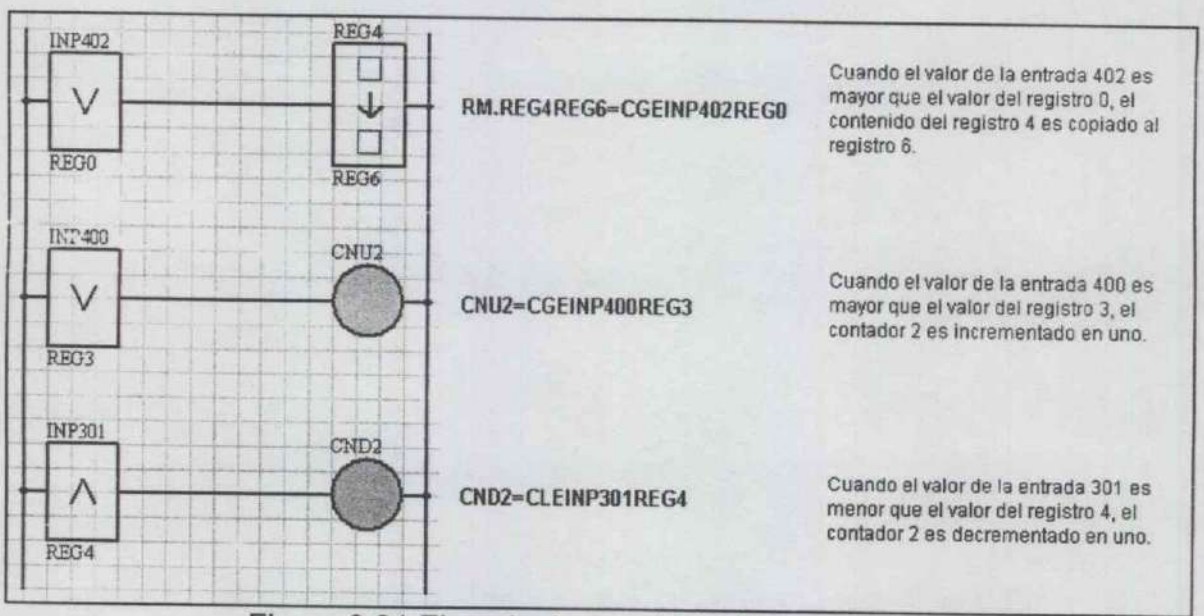

Figura 3.21 Ejemplo de uso de registros internos

3.3.7 Módulos del programa

El software está organizado en varios módulos, que se encargan de realizar las diferentes tareas de la aplicación. En seguida se explica brevemente cada uno de esos módulos.

Módulo principal (main.cpp)

Contiene la rutina de manejo del teclado y la rutina principal del programa (main).

La rutina principal inicia con la ejecución de una serie de rutinas que cargan en memoria los archivos de configuración del programa. Éstos archivos contienèn la información sobre el programa, las tarjetas, y los nombre de las señales, temporizadores, contadores, etc. La tabla 3.11 resume los archivos que se cargan al iniciar el programa y la función de cada uno de ellos.

Tabla 3.11 Archivos de definición del programa

Nombre del archivo	Descripción
PLC.XPP	Contiene el código objeto de la secuencia que ejecuta el PLC (ecuaciones booleanas)
PLC.CFG	Define el tipo de tarjeta insertado en cada ranura del rack
PLC.TDF	Define el nombre de cada temporizador utilizado
PLC.RDF	Define el nombre de cada registro utilizado
PLC.CDF	Define el nombre de cada contador utilizado
PLC.BDF	Define el nombre de cada bandera utilizada
PLC.DEF	Define el nombre de cada señal de entrada o salida de las tarjetas insertadas en el rack

Una vez cargados los archivos de configuración, el ciclo de operación comienza, y sólo termina cuando el usuario activa el teclado de la PC y presiona la tecla '<'. La figura 3.22 muestra los procedimientos del módulo principal.

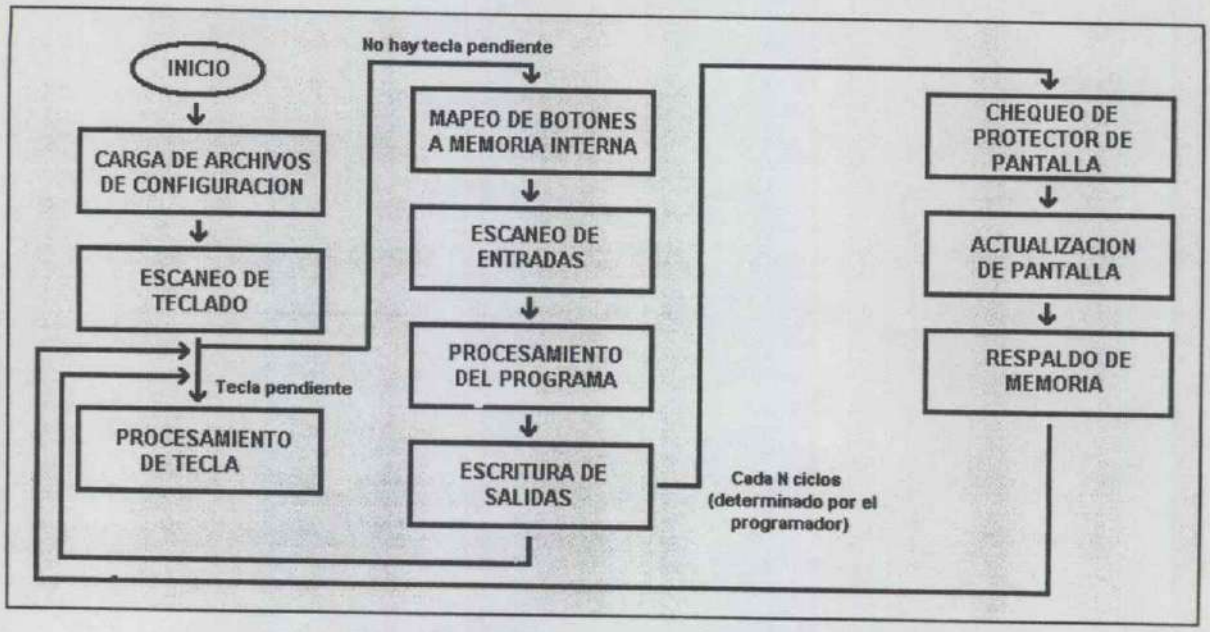

Figura 3.22 Procedimientos del módulo principal

Además del módulo principal, los módulos más importantes son los que se resumen en la tabla 3.12. El anexo D contiene secciones de código de algunos de éstos módulos.

Tabla 3.12 Módulos principales

Nombre del módulo	Descripción
ABCPRINT.CPP	Contiene las rutinas para impresión de pantallas de mantenimiento. Las pantallas de éste módulo permiten ver la información sobre el estado de las entradas y salidas, así como temporizadores internos, contadores, banderas y registros.
ABCPRIN3.CPP	Contiene las rutinas para impresión en pantalla de los parámetros del proceso.
CTARJETA.CPP	Contiene las clases para las tarjetas
INTER.CPP	Contiene el objeto de intercambio de información entre la aplicación gráfica y la aplicación de control secuencial
REGISTER.CPP	Contiene la clase de registros
TIMER.CPP	Contiene la clase de temporizadores
COUNTER.CPP	Contiene la clase de contadores
SELTEXT.CPP TEXTEDIT.CPP	Contienen las rutinas para manejo de campos de edición en pantalla

El módulo ABCPRIN3 define las pantallas de usuario. Éstas pantallas son accedidas mediante las teclas de función (F1...F8) y se organizan en una estructura jerárquica de arbol. La definición de cada pantalla se realiza especificando los textos fijos que contiene, y la posición en que se colocan. Posteriormente se definen las características de cada campo de datos.

Existe un objeto llamado **CFieldSet** (seltext.cpp), que se encarga de manejar todos los campos de edición. Cada campo de edición es una instancia de la clase **CIntText,** que contiene la información sobre la posición factor de conversión, forma en que se despliega el dato en pantalla (porcentaje, entero, tiempo, binario, etc) . La definición de estos objetos es la siguiente (seltext.h):

```
// Archivo de encabezado para campo seleccionable

#ifndef SELTEXT_H
#define SELTEXT_H

#define FIELDMAX 40

enum datatype {NONE=0, TVAL, TLOAD, CVAL, CLOAD, RVAL};
enum percent {PNONE=0, PINT, PCHAR, PCOMP, PBIN, PTIME/*, PPERINT, PPERCHAR*/
                                                    };

class CIntText
{
private:
      unsigned char x,y;
      datatype tipo;
      unsigned numdato;
      unsigned char left, right, up, down;
      percent pcent;
public:
      float KConv;
      CIntText();
      void SetAll(unsigned char px, unsigned char py, datatype tip, unsigned num,
percent porcentaje, float Kconv=1.0);
      void Print(int selected);
      void SetLeft(char num);
      void SetDown(char num);
      void SetRight(char num);
      void SetUp(char num);
      void SetPos(unsigned char px, unsigned char py);
      void SetType(datatype tip);
      void SetNumber(unsigned num);
      datatype GetType();
      percent GetPercent();
      unsigned char GetLeft();
      unsigned char GetRight();
      unsigned char GetUp();
      unsigned char GetDown();
      unsigned GetNum();
};

class CFieldSet
{
private:
      unsigned selected;
      CIntText Field[FIELDMAX];
```

```
public:
      CFieldSet();
      void Process(char key);
      void Reset();
      void SetUD(unsigned u, unsigned d);
      void SetLR(unsigned l, unsigned r);
};

#endif
```

El método **Process()** se encarga de controlar la edición de los campos de forma que el programa no se detenga. Mediante variables estáticas se guarda la información pertinente al estado actual de la edición de un campo, y solo cuando una tecla es presionada por el usuario, el código efectúa alguna acción, y regresa el control a la rutina que lo llamó.

El módulo INTER contiene la definición y el código para la clase que permite el intercambio de información entre el PLC y la interfaz del usuario. La información se obtiene a través de métodos públicos de acceso.

El anexo D contiene secciones del código de los archivos ABCPRINT, ABCPRIN3, y SELTEXT.CPP.

El módulo CTARJETA contiene la definición de clases para cada tipo de tarjeta que se diseño. Cada clase derivada de CTARJETA contiene las rutinas y el espacio para almacenar los datos específicos que maneja el tipo de tarjeta al que corresponde. Las rutinas especifican la forma en que se lee o escribe a la tarjeta. Una clase llamada CINTERFACE contiene las rutinas de manejo de la tarjeta de interfaz, y está incluida en la clase CTARJETA. La tabla 3.13 indica la correspondencia entre el nombre de la clase, y el tipo de tarjeta que controla. La figura 3.23 muestra la estructura jerárquica de las clases.

Tabla 3.13 Nombres de clases y tipos de tarjetas

Nombre de la clase	Tipo de tarjeta
CTIn24	Entradas digitales, 24 canales
CTOut16	Salidas digitales, 16 canales
CTAninput16	Entradas analógicas, 16 canales
CTAnoutput8	Salidas analógicas, 8 canales
CTTpjinput8	Termopar tipo J, 8 canales
CABIn8	Marca Allen-Bradley, entradas digitales, 8 canales
CABOut8	Marca Allen-Bradley, salidas digitales, 8 canales

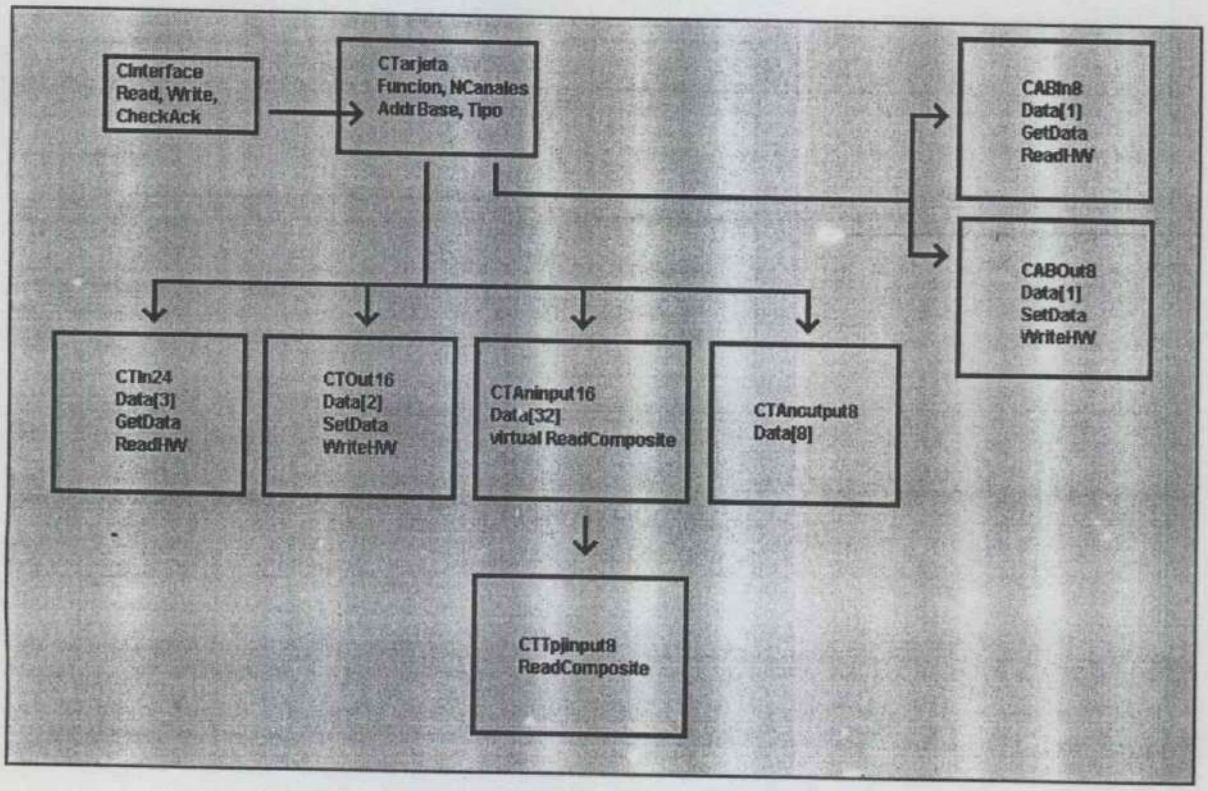

Figura 3.23 Estructura jerárquica de las clases

IV. IMPLEMENTACIÓN Y PRUEBAS

La primera máquina en que se implementó el control propuesto en esta tesis es una Reed Prentice de 300 ton. de cierre ubicada en una planta industrial de fabricación de electrodomésticos. La máquina contaba originalmente con un control de tarjetas electrónicas con lógica digital. Se instaló el equipo, de modo que se sustituyó la botonera del operador por la terminal del usuario consistente en una pantalla de cristal líquido y un teclado de propósito especial. La figura 4.1 muestra la botonera del operador original, y la terminal del usuario por la que fue sustituida; y la figura 4.2 muestra el control con su rack de tarjetas ya instalado en el gabiñete de la máquina.

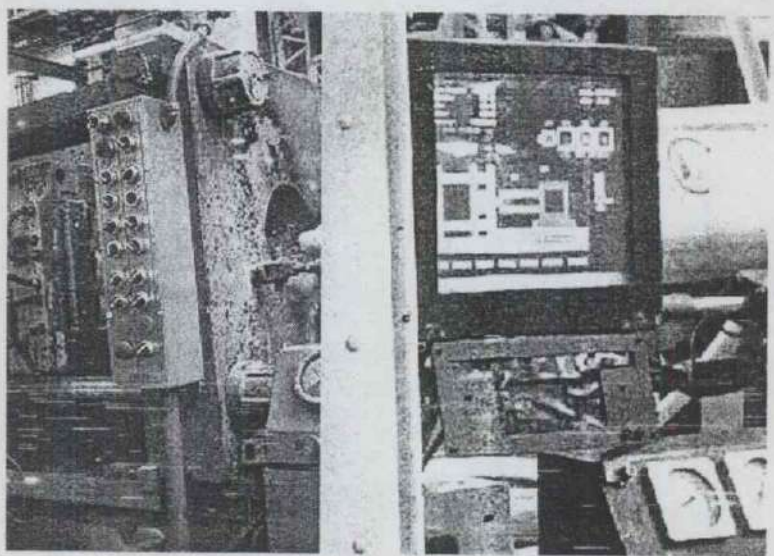

Figura 4.1 Botonera original e interfaz de operación nueva

Debido al deterioro de los conectores de plástico de las tarjetas y las señales de control para las válvulas, se decidió reemplazarlo para corregir falsos contactos y fallas en las tarjetas que implicaban un mantenimiento complicado y largo.

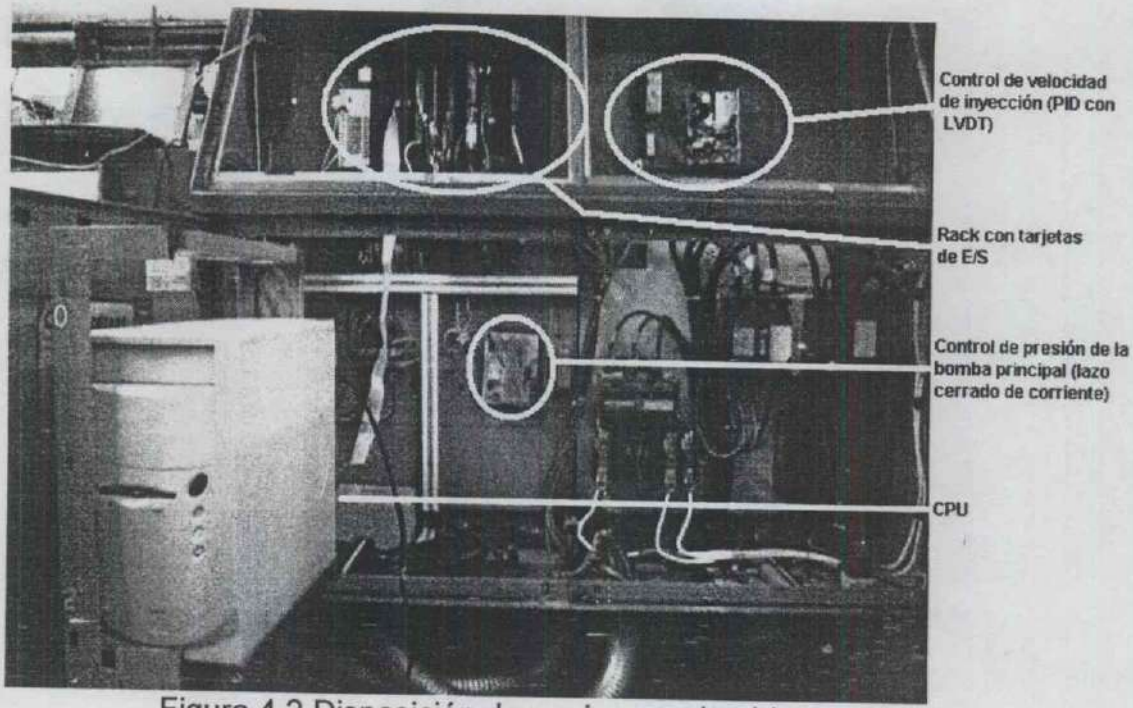

Figura 4.2 Disposición de equipo en el gabinete eléctrico

El sistema hidráulico cuenta originalmente con válvula proporcional de control de presión en la bomba, lo que permitió controlar la mayoría de las presiones de operación de la misma desde la PC. Se efectuó una modificación hidráulica para adaptar una válvula de control de velocidad para la inyección, lo que requirió del maquinado de un bloque adaptador para incluir la válvula proporcional de velocidad, y una válvula check que se muestran en la figura 4.3.

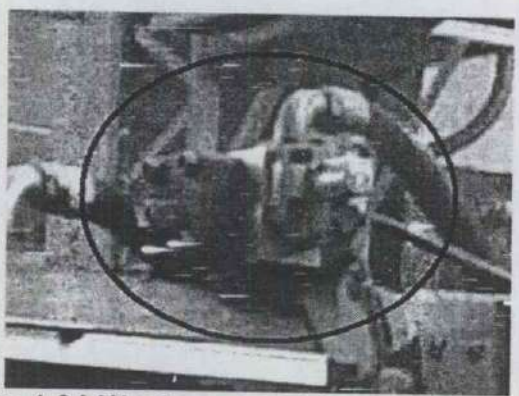
Figura 4.3 Válvula y monobloque de inyección

Para implementar el control de velocidad de inyección, se realizó una tarjeta amplificadora con un controlador PID de lazo cerrado retroalimentado por

LVDT. La sintonización se llevó a cabo de forma empírica con base en los conocimientos empíricos de los operadores de la máquina.

Una particularidad que tenía esta máquina es el hecho de que utiliza RTD para la medición de temperaturas en el cañón. Para poder utilizarlos, se diseñó una tarjeta para realizar un ajuste de ganancia-offset en las señales de los RTD's. La salida de éstas tarjetas se conectó a la tarjeta de entradas analógicas.

Esta máquina trabaja con el equipo en una planta de electrodomésticos desde Octubre de 2002. La figura 4.4 muestra la vista de la máquina desde el lado del operador.

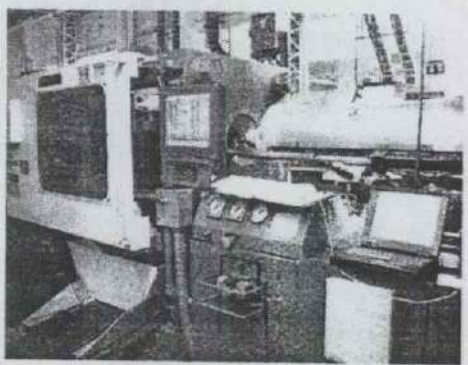

Figura 4.4 Vista general de la máquina

La segunda máquina en que se implementó es una Cincinnati Milacron que se convirtió recientemente (Febrero 2004). Se cotizaron varias opciones y el uso de un equipo marca Allen Bradley involucraba una inversión de alrededor de 9000 dólares (sin incluir ingeniería, instalación, etc.), y el desembolso requerido era inmediato. A este costo hubiera habido necesidad de agregar los costos de ingeniería e instalación. Una opción sencilla, con una capacidad de visualización mucho menor (pantalla de cristal líquido), en equipo Siemens, representaba una inversión total de alrededor de 8000 dólares. El costo total de este proyecto fue aproximadamente de 5000 dólares. La figura 4.5 muestra la terminal del operador, que se adaptó en el mueble de la terminal original.

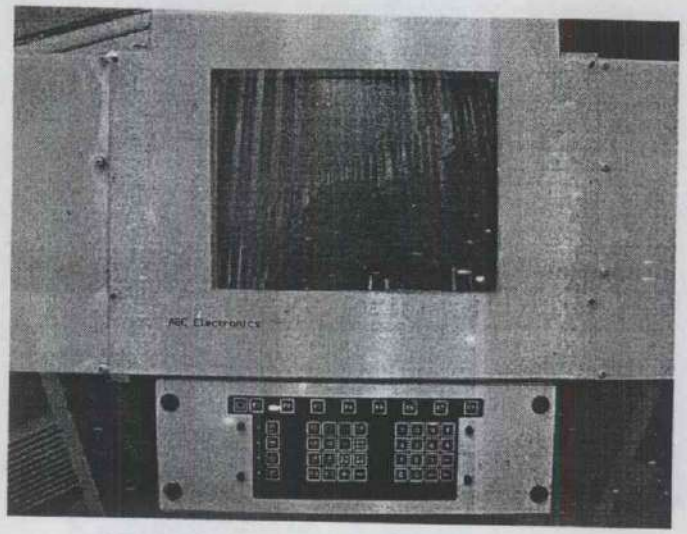

Figura 4.5 Terminal del operador

En la conversión de una máquina se deben que tomar en cuenta las capacidades de expansión para agregar diferentes opciones posteriormente. El hecho de introducir una PC da la ventaja de la economía de las refacciones y una gran capacidad de procesamiento. La electrónica utilizada dentro de las tarjetas es muy sencilla, siendo así reparable por un técnico calificado. La única parte no reparable por un técnico es el microcontrolador, que sencillamente se sustituye.

En esta máquina existían válvulas proporcionales en la inyección, carga, y los movimientos del molde, permitiendo controlar sus velocidades desde la terminal del usuario, a través de la tarjeta de salidas analógicas, y una tarjeta amplificadora para cada señal. Las tarjetas amplificadoras consistieron en una etapa de ganancia variable y offset, con una etapa de fuente de corriente controlada por voltaje a la salida.

La existencia de potenciómetros en la inyección, la expulsión y la platina, permitió el monitoreo y control de estos movimientos desde la computadora - mediante comparaciones simples de las diferentes posiciones establecidas por el usuario para cambiar de una etapa a otra - contra la posición actual. Por ejemplo: El cierre tiene varias etapas, comenzando por un cierre lento, posteriormente el avance rápido, seguido por el cierre de baja presión, y por último la acumulación

de presión. En la inyección se tienen 6 etapas de alta presión, y 3 de baja cuya conmutación depende del tiempo y la posición especificados por el usuario.

Existen varios archivos de texto que definen la programación de la máquina. El primer archivo contiene los códigos de las tarjetas de cada ranura. Una vez establecido el tipo de cada tarjeta, la aplicación muestra en las pantallas de mantenimiento la información sobre el estado de las entradas o las salidas de cada tarjeta. También se definieron los archivos de señales (PLC.CDF), y los de temporizadores, banderas, registros y contadores. El archivo TDF, para timers, CDF, contadores, RDF, registros, y BDF, banderas.

Se definieron los campos que se pueden configurar para cada pantalla. Cada campo puede mostrar un registro, timer, o contador como entero, un porcentaje, o en el caso de timers como segundos y centésimas. Al mostrarse en pantalla se puede desplegar multiplicado por un factor. Se especificó su posición dentro de la pantalla y la relación que guardan respecto a los demás campos, para determinar la forma en que las flechas del teclado seleccionan cada campo. Esta relación se especifica determinando pares de campos con una relación izquierda-derecha, y arriba-abajo. Todos los registros, contadores, banderas y temporizadores son visualizables también desde la pantalla de registros, organizados en orden numérico, a diferencia de las pantallas de campos, que los muestran de forma amigable y lógica para el usuario.

Existe una pantalla principal que despliega la máquina con sus cuatro movimientos: unidad de inyección, pistón de inyección. expulsores y platina. Muestra también las temperaturas de cada zona del cañón de la máquina, así como el estado de encendido o apagado de las resistencias de dichas zonas. Algunos datos adicionales se muestran en esta pantalla como son el total de ciclos efectuados, el tiempo de ciclo, tiempo de curado de la pieza, tiempo de inyección, hora y fecha. Después de no existir interacción por parte del usuario con la

aplicación, entra en funcionamiento un protector de pantalla que muestra los tiempos mencionados, la hora y la fecha, cambiando de posición y color.

En el borde inferior se enlistan las pantallas que se accesan con las teclas de función (F1-F8). En el código se especifica la organización de las pantallas en forma de árbol como se muestra en la figura 4.6. Para fines de explicar los parámetros de cada una, y sus funciones, nos referiremos a ellas por el número especificado en la figura. En el apéndice A se muestran las pantallas (se omiten aquí por cuestión de espacio y legibilidad).

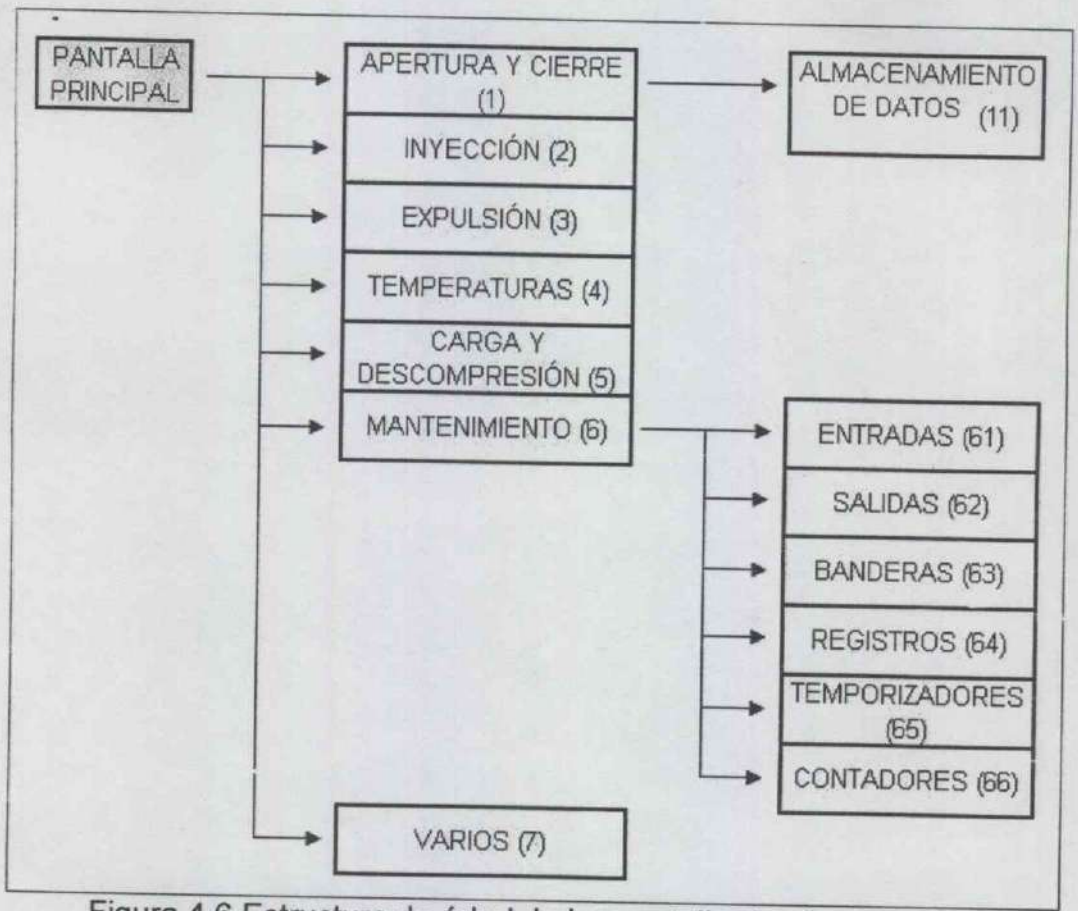

Figura 4.6 Estructura de árbol de las pantallas implementadas

La primera pantalla (1) permite especificar los parámetros de velocidades, presiones, y posiciones del cierre y la apertura del molde. Tanto el cierre como la apertura se conforman de varias etapas.

El cierre comienza con una etapa lenta, seguida por una etapa rápida, una etapa de baja presión, y una etapa de acumulación de presión. El usuario puede especificar los parámetros para todas ellas excepto la etapa lenta inicial, que toma los mismos parámetros de la etapa de baja presión. Para cada una, se puede ajustar la velocidad, la presión, y la posición en la cual el cierre cambia a la siguiente etapa. El cambio de la etapa lenta inicial a la etapa rápida está determinado por un temporizador (No. 37), se ha encontrado en la práctica que un tiempo aproximado de medio segundo produce un arranque suave de la platina móvil al cerrar. Si éste tiempo se reduce a cero, la etapa lenta inicial es suprimida de la secuencia de la máquina. Este tiempo sólo puede ser cambiado dentro de las pantallas de mantenimiento de temporizadores.

La apertura inicia con la descompresión o despresurización del molde. Esta etapa sólo se presenta cuando el molde se encuentra cerrado a alta presión. Los parámetros de presión y velocidad de despresurización son los mismos que los de la etapa de acumulación de presión del cierre. Generalmente, las máquinas incluyen un presostato o un sensor de posición en la rodillera (dependiendo del tipo de mecanismo de cierre) que permite saber cuando el molde se encuentra cerrado a presión, y consecuentemente detectar cuando se ha despresurizado. Esta señal permite el cambio de la etapa de despresurización a la de apertura lenta inicial. Cuando se llega a la posición de cambio especificada se inicia la apertura rápida, y de la misma forma se cambia a la etapa de frenado de apertura, que termina en la posición de apertura total del molde.

Esta pantalla (1) despliega también la posición actual que se mide en el transductor de la platina, y el tiempo de curado del molde.

La pantalla 2 permite visualizar y editar los parámetros de inyección. Se incluyeron en el programa de la máquina nueve etapas para formar un perfil de inyección. Las primeras seis etapas tienen la finalidad de ser utilizadas para la inyección de alta presión, y las ultimas tres para las etapas de sostenimiento (baja

presión). A pesar de estar desplegadas de forma separada (las etapas de alta en la parte superior de la pantalla, y las etapas de baja en la parte inferior), las nueve etapas funcionan de la misma forma, como se explica a continuación.

Se especifica tiempo, velocidad, presión, y posición para cada etapa. Durante cada etapa su presión y velocidad son transferidas a las salidas analógicas que controlan las válvulas proporcionales. El cambio de una etapa a otra se produce por tiempo, o por posición, lo que ocurra primero (Típicamente, se desea que el cambio sea por posición, para mantener el proceso con una alta repetibilidad; entonces el tiempo de cada etapa se especifica con un valor mayor al que realmente toma la ejecución de esa etapa).

La única diferencia entre las etapas de alta y baja presión, es que en las etapas de baja presión no se especifica un cambio por posición, sino que todas cambian por tiempo.

En caso de que en el proceso no se desee utilizar todas las etapas, se especifica un tiempo de cero en las mismas.

Esta pantalla (2) también muestra la posición actual del pistón de inyección, e indica durante el ciclo de operación, la etapa que se encuentra activa durante la inyección.

La pantalla 3 nos permite especificar las condiciones deseadas para la expulsión. Este movimiento está definido por dos etapas para el avance, y dos etapas para el retroceso. Para cada etapa se define una presión y una posición de cambio a la siguiente etapa. Si la aplicación no requiere de dos etapas en alguno de los movimientos, se pueden especificar los mismos parámetros para ambas. Aunque la mayoría de los procesos requieren únicamente de una etapa para cada movimiento, se incluyeron dos etapas por solicitud de un cliente. También se

puede especificar la cantidad de veces que accionan los expulsores, así como un tiempo de pausa entre el avance y el retroceso.

Desde la pantalla 4, el usuario puede encender o apagar todo el control de temperatura, y establecer las temperaturas deseadas para cada zona (setpoint). Actualmente, aunque no se encuentran implementadas, la pantalla muestra campos que permiten especificar las temperaturas máximas y mínimas de operación para cada zona. En un futuro se utilizarán para que la máquina no opere fuera de los rangos establecidos, y envíe alarmas al usuario en caso de salirse de los mismos.

En pantalla 5 se pueden modificar los parámetros de carga. Los datos especificados para este movimiento son presión, velocidad, y dos posiciones: una para la cantidad de carga, y otra para el fin de la descompresión del husillo.

La finalidad de la pantalla 6 es permitir a un técnico dar mantenimiento a la máquina. A través de ella es posible acceder pantallas en un nivel inferior (dentro de la estructura de arbol), en las que se puede ver la información interna del PLC del software.

Las pantallas 61 y 62 muestran la información sobre todas las tarjetas de entradas y salidas respectivamente, existentes en el sistema, según se especificó en el archivo con extensión CFG. Con las flechas del teclado del usuario es posible cambiar la tarjeta o el rango que se visualiza en pantalla (las tarjetas de entradas digitales muestran sólo la mitad de sus señales en una sola pantalla). Para cada tarjeta se especifica su dirección, el número de ranura en que se encuentra, y el estado actual de sus señales con un texto que describe la función de la señal (especificado en el archivo con extensión DEF). En caso de no existir tarjetas de entradas o de salidas, la pantalla correspondiente mostrará un mensaje especificando que no existen tarjetas de ese tipo definidas en el archivo de configuración.

La pantalla 7 muestra información adicional sobre la máquina. Aquí es posible establecer la presión mínima del sistema para especificar la señal de salida deseada en la válvula proporcional de la bomba principal. Este parámetro es utilizado para establecer la presión que se desea en movimientos como expulsión (en el caso de máquinas sin válvula proporcional en la expulsión) y movimiento de la unidad de inyección. Otro parámetro que se muestra aquí es el contador general de piezas, y se puede inicializar a cero (F1). Por último, se incluyen también los parámetros de lubricación automática (en caso de que la máquina cuente con el sistema), permitiendo establecer la cantidad de ciclos entre lubricaciones, y la duración de la lubricación.

Dentro de la pantalla de ajuste de parámetros de cierre y apertura (11) se tiene la opción de guardar o cargar la memoria interna. Se pueden almacenar hasta 30 archivos diferentes, con parámetros de trabajo para diferentes moldes, y cargarlos cuando sea necesario. Un archivo de texto permite identificar en pantalla el nombre de cada uno de los moldes correspondientes a estos archivos.

Es conveniente también incluir el control de temperaturas de la máquina. Lo más común es encontrar controles en lazo cerrado con PID, el cual controla el ciclo activo de un PWM de período muy largo. Esto es conveniente porque en ocasiones se utilizan todavía contactores electromecánicos que podrían tener un fuerte desgaste mecánico si conmutaran muy seguido; y es factible debido a que se trata de un sistema con una respuesta muy lenta (un cambio en el control puede tardar varios minutos en reflejarse). Algunas máquinas utilizan relevadores de estado sólido, o de mercurio, en cuyo caso el desgaste mecánico de los contactos no existe. Inicialmente se implementó un control de tipo encendido-apagado.

V. CONCLUSIONES

El equipo desarrollado permitió llevar a cabo instalaciones exitosas en las cuales se tiene la flexibilidad para reprogramar las funciones de la maquinaria. Agregar opciones, o sincronizar la maquinaria con equipo periférico (robots, y alimentadores entre otros) es algo que se puede realizar con sencillez. El uso de equipo de cómputo tradicional no solo permitió que el costo de estas intalaciones fuera mucho menor al que hubiera tenido utilizando equipo convencional, sino que da al usuario la capacidad de monitorear su proceso y dar mantenimiento más eficientemente.

El uso de componentes comúnes garantiza el abastecimiento durante un tiempo prolongado para la fabricación de éste equipo. La forma en que se definió la arquitectura deja abierta la posibilidad de diseñar nuevos tipos de tarjetas con sencillez.

La estructura del código nos permitió aislar los módulos de control y de interfaz con el usuario, para poder hacer cambios en cualquiera de los dos sin interferir con el funcionamiento del otro módulo. La adición de tarjetas fue paulatina y conforme se fueron requiriendo se diseñaron las necesarias, comprobando así la versatilidad del sistema.

El proyecto deja abiertas las puertas para futuros desarrollos tanto en el área de electrónica digital, como en el desarrollo de software y en la investigación sobre el proceso de inyección de plástico, hidráulica, y fabricación de moldes entre otros. También se puede utilizar en otro tipo de maquinaria, para realizar el control de la misma.

La implementación del equipo en tres máquinas - dos de las cuales se encuentran en líneas de producción - permitió probar la calidad de los diseños en condiciones de trabajo industrial, y abordar apropiadamente los problemas que se presentaron.

Se espera que los diseños presentados sean un sólido inicio en la estandarización del equipo que se utiliza en los proyectos de investigación y desarrollo de la Universidad.

BIBLIOGRAFIA

Anuario Económico 2002, Secretaría de Desarrollo Sustentable del Estado de Querétaro.

Castellanos G. J. J. 2001. Diseño y Construcción de una Tarjeta de Control de Arquitectura Abierta, UAQ, Qro. México. p. 1-12.

Donald W. 1998. How do you Justify a Retrofit? Society of Manufacturing Engineering.

Grierson D. K. 1985. The Role of Standards in the Factory with a Future. Automated Manufacturing. Symposium sponsored by ASTM Committee ·E-31 on Computerized Systems. California, USA. p. 31-37

Herrera G. 1992, Design of CNC Modular System. IFAC Workshop on Automatic Control for Quality and Productivity ACQP'92, Istambul Turkey.

Herrera G. et al. 1993. Design of a New Generation of CNC Architecture for Intelligent Machinig Systems. 4 International DAAAM Symposium. Brno, Czechia.

McEwen S. M. 1985. A Standard for the Intercommunication of Computers in a Flexible Manufacturing System. Automated Manufacturing. Symposium sponsored by ASTM Committee ·E-31 on Computerized Systems. California, USA. p. 31-37

Petruzella F. D. 1989. Programmable Logic Controllers. McGraw-Hill, Ohio, USA.. p. 1-128

Piedrafita M. R.. 2001. Ingeniería de la Automatización Industrial. Alfaomega Grupo Editor, México, D.F. p. 1-198.

Pritschow G. 2001. Open Controller Architecture – Past, Present and Future. Society of Manufacturing Engineering

ANEXOS

ANEXO A. PUBLICACIÒN

DESARROLLO DE UN SISTEMA DE CONTROL MODULAR PARA MAQUINARIA

Bravo C. José Agustín, Herrera R. Gilberto
Universidad Autónoma de Querétaro
Cerro de las Campanas S/N
76010, Querétaro, Qro.
virtualcab@yahoo.com.mx

RESUMEN.
Actualmente la industria tiene la necesidad de contar con una planta productiva flexible que le permita cambiar su línea de productos de forma rápida y a bajo costo. Los procesos de producción deben permitir producir diversos productos a costos competitivos con la misma infraestructura. Desgraciadamente los sistemas de control actuales no son abiertos y no permiten en su mayoría adaptar más ejes o cambios para controlar otra máquina diferente, esto se observa principalmente en los controles para máquinas-herramienta o máquinas inyectoras. El presente artículo describe el desarrollo de un control modular de bajo costo para maquinaria de este tipo; toma ideas de propuestas de arquitectura abierta desarrolladas en Estados Unidos, Europa, y Japón y presenta un prototipo propio. El objetivo es desarrollar un sistema de control que permita a los usuarios e investigadores ampliarlo o modificarlo, y agregar nuevos algoritmos de control si se encuentra necesario.

1. INTRODUCCIÓN.

La industria manufacturera representa en nuestro país uno de los sectores económicos más importantes. En particular, el estado de Querétaro se caracteriza por la existencia de una fuerte presencia de industria dedicada a actividades relacionadas con maquinaria y equipo. Para valorar este hecho podemos mencionar algunas cifras extraídas de los censos económicos del INEGI. En el año 2001, el subsector de maquinaria y equipo fue responsable de producir el 36.6% del valor del total de las ventas netas en la industria manufacturera a nivel nacional, y el 37.7% en el estado de Querétaro. Éste sector trabajó durante los años 1999 y 2000 aproximadamente al 74% de su capacidad instalada, ocupando al 42% del total del personal laborando en este subsector, y representó cerca del 26% del total de establecimientos de la industria manufacturera. Ahora bien, el total de los establecimientos industriales esta conformado en un 95.8% por micro y pequeñas empresas; es decir, que ocupan cada una a menos de cien personas [5]. Es por ello que se debe reforzar el apoyo brindado a este sector, para facilitar su crecimiento y desarrollo.

En particular, es necesario ofrecerle opciones que lo vueivan un sector más productivo y con costos

operativos más bajos. Para este sector, la existencia de maquinaria con controles flexibles y que requieran de poco capital, comparado con las opciones existentes actualmente sería una fuerte ayuda.
Los controles flexibles han sido muy populares desde el principio de la década de los 90's. El usuario de maquinaria busca equipos que se adapten a sus necesidades y le permitan extender sus capacidades posteriormente, o modificarlas para mejorar sus procesos productivos. Günther Pritschow menciona en [1] una tendencia de los mercados de máquinas-herramienta a buscar controles cuyo costo sea reducido y permitan agregar algoritmos de control que cubran necesidades específicas de su proceso. Y al igual que en muchas otras áreas de la industria, la mayoría de los controles existentes dan al usuario pocas posibilidades de modificar o adaptar el control cuando su proceso cambia.
Existen varios grupos internacionales que definen arquitecturas para sistemas abiertos como el OSEC y el JOP en Japón, el OMAC en E.U., o el OSACA en Europa. Todos ellos promueven el uso de interfaces de programación para que el usuario se pueda comunicar con el controlador[4] sin embargo no presentan propuestas concretas en cuanto al hardware a utilizar. Otros estándares ampliamente utilizados en la industria definen sus protocolos de comunicación, y en ocasiones el hardware necesario, pero presentan otro tipo de problemática como la obsolescencia – como el caso del bus ISA- y en consecuencia la falta de soporte y refacciones, o requieren de circuitos complejos para implementar el protocolo (como el PCI, y otros estándares industriales como Profibus, Fieldbus, AS-Interface y Ethernet, entre otros).

[4] API (Application Programming Interface) por sus siglas en inglés.

2. DESCRIPCIÓN DEL SISTEMA

El controlador descrito aquí pretende sentar una base, tanto de hardware como de software, para poder realizar en el futuro investigaciones sobre diferentes procesos. El equipo pretende dar a los investigadores una herramienta que les permita agregar o modificar algoritmos para observar sus implicaciones sobre el proceso. Esto se logra mediante el uso de tecnología modular, que permite al usuario implementar funciones no incluidas dentro del núcleo de la aplicación. Trabajos anteriores [3] y [4] muestran desarrollos de controladores para máquinas-herramienta que siguen esta tendencia y permiten al usuario integrar sus algoritmos de control al equipo para cubrir sus necesidades específicas.
Por otro lado, el sistema de control utiliza equipo de desarrollo propio, lo que soluciona una necesidad existente en una gran cantidad de micro y pequeñas empresas que no cuentan con el capital suficiente como para automatizar sus procesos, proveyéndoles de un equipo que represente una opción más económica, pero con capacidades equivalentes a los equipos comerciales disponibles en el mercado. En [2], Donald explica que la modernización de maquinaria, al

mejorar la eficiencia del proceso a bajo costo es una justificación importante. Y la posibilidad de abarcar una gran cantidad de marcas y modelos de maquinaria es el resultado de presentar un diseño modular y expansible.
El sistema desarrollado no toma ninguno de los estándares mencionados como base, pero toma de ellos varios aspectos que le dan ventajas importantes.

2.1. Descripción del hardware

Se implementó un bus con señales similares a las del bus ISA, que permite el diseño de periféricos con componentes electrónicos sencillos, pero manejado a través del puerto de impresión. Mediante una tarjeta de interfaz, las señales de dicho puerto se multiplexan para proveer un bus de datos bidireccional de 8 bits, un bus de direcciones de salida de 8 bits,, 16 líneas de selección de tarjetas de periféricos, y líneas de sincronización (lectura, escritura, y aviso de respuesta de las tarjetas). La figura 1 muestra las señales obtenidas con dicha tarjeta.

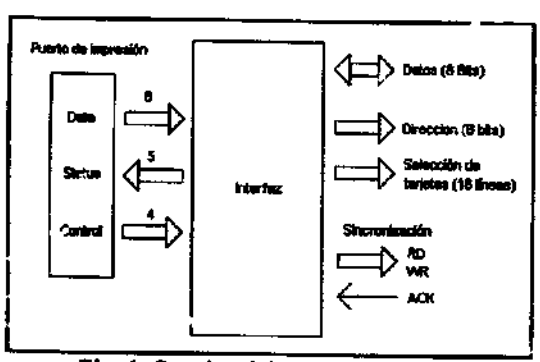

Fig. 1. Señales del bus de hardware

A pesar de que todo esto representa un intercambio de datos con velocidades relativamente lentas[5] e implica integrar software para el control de las señales de enlace entre la PC y los periféricos, es suficiente para una gran cantidad de procesos, a un costo mucho menor que con los equipos convencionales. En todo caso, un diseño basado en PC permite al usuario integrar tarjetas en el bus PCI de la PC cuando sus requerimientos superen los del equipo desarrollado. El uso del puerto paralelo evita que el usuario tenga que accesar la parte interna de la PC para instalar el sistema. Así mismo, permite que el equipo desarrollado sea utilizado con diferentes arquitecturas con modificaciones mínimas a la tarjeta interconexión del bus.
En una primera etapa, se diseñaron tarjetas de periféricos para ser insertadas en conectores tipo peine. Para proveer un equipo suficientemente robusto en el aspecto mecánico, las tarjetas se rediseñaron posteriormente con los tamaños y conectores estándares del VME-BUS. Esto le da al equipo un mejor desempeño que los estándares que manejan peines como el ISA en ambientes industriales con vibración y polvo.
Se diseñaron tarjetas de entradas y salidas digitales para señales de corriente alterna, utilizando en ellas componentes programables para reducir el número de componentes necesarios y manejar la comunicación con la PC. Las tarjetas

[5] Aproximadamente 250KBPS., tomando en cuenta la velocidad del puerto y los ciclos requeridos para completar una transferencia. (KBPS = Kilobytes por segundo).

de entrada manejan 24 señales, y las de salida manejan 16 señales. Los primeros diseños están hechos para manejar señales de corriente alterna hasta 220V, utilizando rectificación y filtrado en las entradas, y tiristores en las salidas. Todas las señales cuentan con aislamiento óptico. La programación de los dispositivos programables se llevó a cabo en ensamblador.

También se elaboraron tarjetas para el manejo de señales analógicas tanto de entrada como de salida. Ambos diseños manejan señales en el rango de 0-10V y utilizan microcontroladores para multiplexar los canales de entrada o salida con los convertidores analógico-digitales.

Para proveer sincronización entre la PC y el hardware, independientemente de la velocidad de las tarjetas de los periféricos, se agregó una señal común para todas las tarjetas con la que éstas responden a las señales de lectura y escritura para indicar cuando han efectuado la operación solicitada.

Los cuatro bits más significativos de direcciones son decodificados a dieciséis líneas de selección de tarjetas. Con éstas líneas cada tarjeta detecta cuando la PC se intenta comunicar con ella. Los cuatro bits inferiores se utilizan para seleccionar registros dentro de cada tarjeta. Esta combinación permite tener hasta 16 registros de 8 bits cada uno dentro de cada una de las 16 tarjetas. Esto representa una limitante en tarjetas con más de 16 bytes de información. Tal es el caso de las tarjetas de entrada analógicas.

La tarjeta de entradas analógicas maneja 16 canales con una resolución de 12 bits, cuya información se guarda en registros internos de la tarjeta alineada en palabras de 16 bits, con los cuatro bits más significativos en cero. Esto representa un total de 32 bytes necesarios para transferir la información de todos los canales de entrada a la PC. Fue necesario implementar una secuencia de lectura indirecta a través de un registro interno de la tarjeta para solucionar este problema. Se realiza un muestreo múltiple (6 muestras) de cada canal para aplicarle un filtro mediana-promedio. La velocidad de la tarjeta está limitada por software a 100 muestras por segundo para los 16 canales. Se utilizó un solo convertidor con capacidad de auto calibración y auto cero, multiplexado para todos los canales.

La tarjeta de salidas analógicas se implementó con una resolución baja de 8 bits. Se utilizó un solo convertidor digital-analógico, con un circuito de muestreo y retención implementado con capacitor y amplificadores operacionales con entrada tipo FET.

2.2. Descripción del software

La aplicación se dividió en dos módulos independientes: el primero se encarga del control, y el segundo de la interfaz gráfica del usuario. La primera razón de la división responde a la necesidad de que cualquiera de estas dos partes pueda ser modificada sin afectar el funcionamiento de la otra. Otro atributo que tiene ésta separación es que se puede asignar prioridad al módulo de control sobre el módulo de interfaz. La interfaz gráfica ejecuta

sus rutinas con una limitación de un número predeterminado de veces por segundo, excepto cuando el usuario ha utilizado el teclado (caso en el que recibe atención al terminar el procesamiento del módulo de control). También abre la posibilidad a futuro de implementar una interfaz gráfica bajo diferentes sistemas operativos, manteniendo sin cambios el módulo de control.

La aplicación se hizo en un compilador de C++ utilizando objetos. El encapsulamiento permitió un aislamiento adecuado entre ambos módulos, que intercambian información a través de un objeto con privilegios de acceso a la información del módulo de control. Con este esquema, la capacidad del módulo de interfaz de usuario para modificar información del otro módulo queda regulada a través de los métodos del objeto de intercambio.

El módulo de control contiene una estructura de datos que incluye temporizadores, contadores, registros, banderas internas, y rutinas de acceso a las tarjetas de periféricos, así como un intérprete del programa que define la operación secuencial de la máquina (de forma similar a un PLC). Todos los elementos de este módulo son configurables por el usuario a través de archivos de texto que definen el tipo de tarjeta utilizado en cada posición del bus de hardware.

El módulo de interfaz gráfica del usuario maneja un esquema de pantallas organizadas en estructura de árbol, cuya información es accesible y modificable mediante el teclado del operador. El software permite al usuario configurar cada pantalla para colocar en ella la información que requiera mediante un archivo. Así mismo, es posible describir la función de cada una de las señales y registros del programa para visualizarlas fácilmente dentro de seis pantallas de mantenimiento (temporizadores, contadores, registros, banderas, entradas, y salidas).

El teclado de usuario se fabricó a partir de interruptores normalmente abiertos, y se conectó al puerto de teclado de la PC. Para leerlo se utilizaron rutinas de bajo nivel que permiten identificar, para cada tecla y de forma independiente, cuando el usuario la presiona, y cuando la suelta. Con ello es posible eliminar una cantidad importante de botones en la mayoría de las aplicaciones industriales. Todos los botones, excepto los destinados a cambios de pantallas, son mapeados a una sección de las banderas internas.

Como se mencionó anteriormente, el usuario puede programar el control secuencial de forma similar a la de un PLC, mediante un archivo que contiene ecuaciones booleanas extraíbles de un diagrama de escalera común. Con esto es posible modificar la secuencia de operación del control, haciendo del sistema un sustituto funcional de un PLC, con diversos valores agregados.

El software permite al usuario tener las herramientas comunes en los sistemas modernos de automatización. Entre ellas se encuentran el monitoreo de todas las señales internas y externas del control, elaborar gráficas para visualizar el progreso de algunas variables de la máquina, aplicar

control estadístico del proceso y almacenar datos en disco para su análisis posterior. También permite programar alarmas y lleva una bitácora de las mismas.

En una segunda versión, en la que se trabaja actualmente, se optó por utilizar un segmento de memoria lineal, donde se almacenan todos los datos del módulo de control, y sin la separación de los mismos en objetos. Esto implica un mayor riesgo en el intercambio de datos entre los dos módulos de la aplicación, debido a que se pierde la protección de encapsulamiento provista por los objetos. Sin embargo, facilita y optimiza las funciones internas del módulo de control, mejorando su velocidad de ejecución de forma considerable. Además, el código de dicho módulo se redujo aproximadamente en un 70%. En la búsqueda de controles de bajo costo, esto abre las puertas para el uso de equipos con requerimientos de hardware muy accesibles (corre satisfactoriamente en máquinas con procesador 486).

En esta segunda versión también se modificaron las rutinas de lectura y escritura de hardware, para permitir el diseño de nuevas tarjetas y poderlas manejar sin necesidad de programar código. El diseñador del hardware sólo necesita incluir un archivo que defina las acciones necesarias para el acceso de su tarjeta tales como escitura de dirección, ciclo de lectura, ciclo de escritura, y espera de señal de reconocimiento entre otras.

La Tabla 1 muestra los archivos que definen las opciones de la aplicación.

Módulo	Categoría	Elementos descritos	Configurable por	Requerido por la aplicación
Interfaz gráfica	Definición de señales	Temporizadores	Usuario	Opcional
		Contadores	Usuario	Opcional
		Banderas	Usuario	Opcional
		Registros	Usuario	Opcional
		Entradas	Usuario	Opcional
		Salidas	Usuario	Opcional
	Definición de información en pantallas	Datos a desplegar	Usuario	Opcional, si no se define, solo existen las pantallas de mantenimiento.
Control	Definición de hardware utilizado	Tipo de tarjeta de cada ranura	Usuario	Obligatorio
	Descripción de tarjetas	Rutinas de lectura y escritura	Diseñador de Hardware	Obligatorio

Tabla 1. Archivos de la aplicación

3. IMPLEMENTACIÓN Y PRUEBA DEL PROYECTO

Para probar el sistema desarrollado, se instaló en una maquina de inyección de plástico marca Reed Prentice modelo 300TE fabricada en el año 1981, en una fábrica de electrodomésticos.. La máquina contaba originalmente con un control basado en tarjetas electrónicas. La instalación se hizo en Octubre del 2002 utilizando los primeros diseños (conectores de peine). Con el fin de integrar todo el control de la máquina en la PC, se agregó al software la capacidad de controlar temperaturas con el uso de termopares tipo J, mediante un control PID. Los interruptores que controlan los cambios de velocidades y presiones del ciclo se eliminaron, y la posición de los cuatro movimientos básicos (prensa, unidad de inyección, pistón de inyección, y barra de expulsión) ahora se mide continuamente mediante transductores resistivos de posición. Se programó una pantalla principal en la interfaz gráfica, que muestra una animación

representando la posición actual de la máquina, según las señales de posición medidas a través de los transductores. También muestra los tiempos más importantes del ciclo de operación, y las temperaturas y estado (encendido/apagado) de las cuatro zonas que conforman el cañón donde se funde el plástico.
La terminal del operador se hizo montando un monitor de cristal líquido con el teclado del operador bajo el mismo, dentro de un gabinete de aluminio que se pintó con un color similar al de la máquina. La figura 2 muestra la botonera que tenía la máquina antes del cambio, y la terminal de operador ya instalada. Con ésta disposición los botones dedicados a los cambios de pantallas - colocados en la parte superior del teclado – quedan justo debajo del borde inferior del monitor, permitiendo indicar en éste último la función asignada para cada tecla.

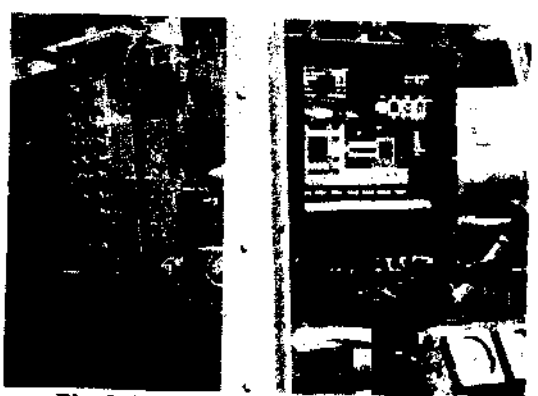
Fig. 2. Botonera y Terminal del operador

Se hicieron modificaciones al sistema hidráulico para agregar una válvula proporcional de control de flujo en el circuito hidráulico de inyección y así poder controlar su velocidad desde la aplicación[6]. Se maquinó un monoblock que se agregó a la máquina en la entrada del pistón de inyección, y donde se colocó la válvula proporcional. Este tipo de válvulas consisten en un solenoide que permite el paso del aceite, y cuentan con un LVDT para retroalimentar la posición de apertura de la válvula. Para controlarla fue necesario diseñar e implementar un amplificador PI con salida de PWM de 24V, conectada a una de las señales de salida analógicas, a través de la cuál la PC enviaba la señal de referencia de la velocidad deseada. En la figura 3 se puede apreciar el monoblock, junto con sus válvulas.
Con el fin de minimizar el tiempo de instalación del equipo se diseñó un simulador de la máquina, consistente en software de emulación de movimientos con tarjetas externas para generar señales iguales a las que produciría la máquina real. Este procedimiento permitió probar el equipo y la secuencia de control antes de instalarlo. El mantenimiento correctivo del control de la máquina se redujo considerablemente, ya que el equipo original de la misma se encontraba muy deteriorado, y presentaba fallas constantemente. Además, la interfaz gráfica permite visualizar todas las señales internas y externas del control, para rastrear e identificar los problemas que se puedan presentar

[6] La velocidad de inyección es uno de los parámetros más importantes del proceso de inyección de plástico.

Fig. 3. Monoblock para control de la velocidad de inyección.

El costo del equipo fue mucho menor al de un equipo comercial. Como ejemplo podemos citar los costos del equipo básico (sin tarjetas de entradas y salidas) para ésta máquina: la PC con monitor de pantalla plana de 15" a color, teclado de operador de 45 botones, chasis y fuente para 16 tarjetas tiene un costo aproximado de $1200 dlls. Un equipo comercial con capacidades similares consistente en un Panel de operador monocromático con 27 botones, procesador central y chasis con fuente para 13 tarjetas, tendría un costo aproximado de $2750 dlls. (229%).

Actualmente, la segunda versión del software se encuentra en pruebas en una máquina de inyección Krauss-Maffei mod. STM235, y proximamente se instalará otro sistema en una máquina marca Husky.

4. REFERENCIAS.

[1] Günter Pritschow, Yusuf Altintas et.al. "Open Controller Architecture – Past, Persent and Future." Society of Manufacturing Engineering 2001.

[2] Donald W. "How do you Justify a Retrofit?". Society of Manufacturing Engineering. May 1998.

[3] Gilberto Herrera Ruiz. "Design of CNC Modular System". IFAC Workshop on Automatic Control for Quality and Productivity ACQP'92, Istambul Turkey, June 1992.

[4] Gilberto Herrera Ruiz et al. "Design of a New Generation of CNC Architechture for Intelligent Machining Systems". 4th International DAAAM Symposium, Technical University of Brno-Czechia, September 1993.

[5] Anuario Económico 2002, Secretaría de Desarrollo Sustentable del Estado de Querétaro.

ANEXO B. PANTALLAS DE LA APLICACIÓN IMPLEMENTADA

Pantallas de la aplicación.

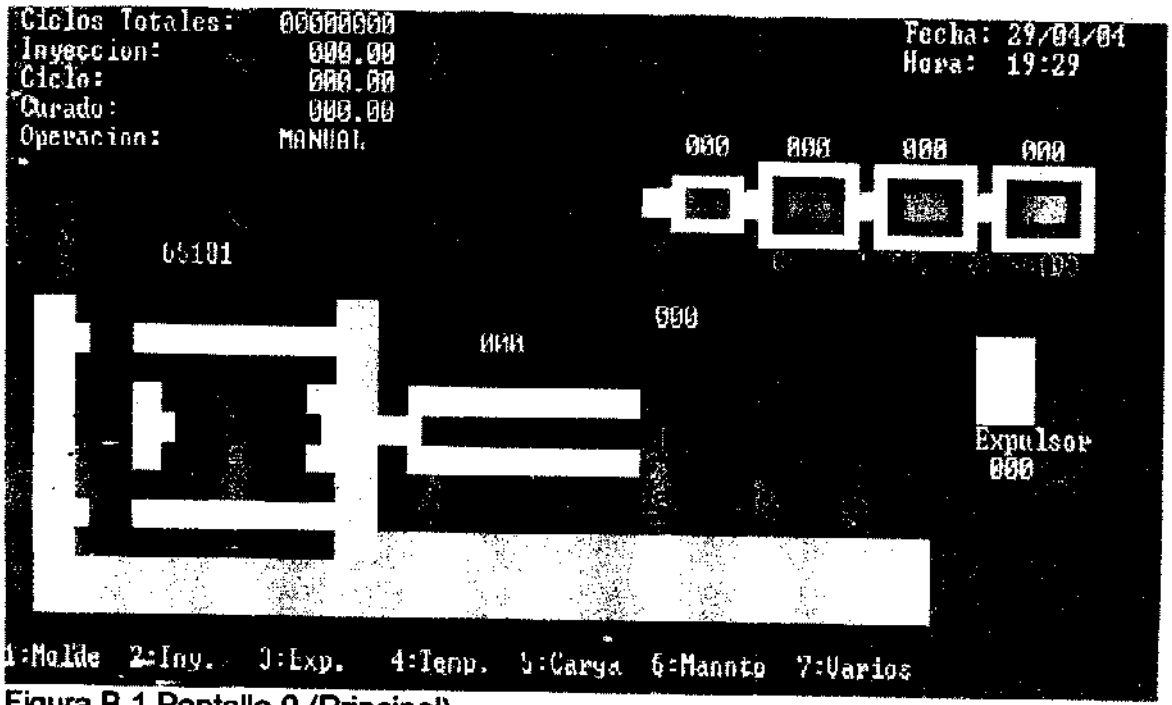

Figura B.1 Pantalla 0 (Principal)

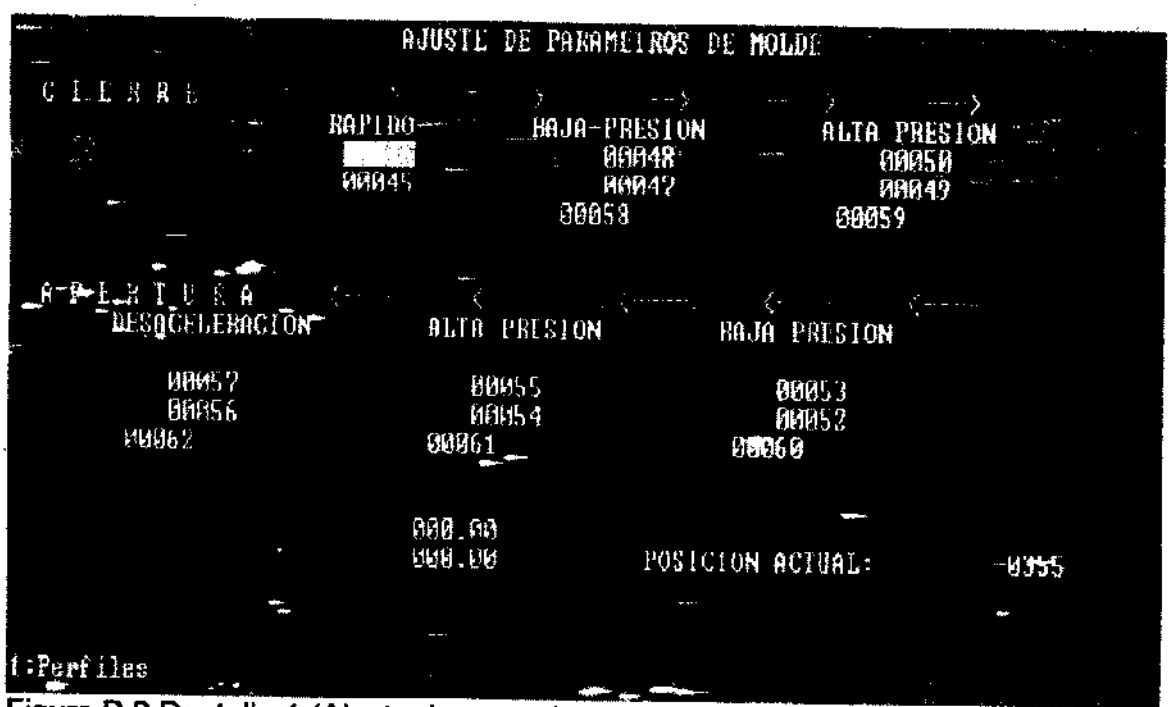

Figura B.2 Pantalla 1 (Ajuste de mov. de molde)

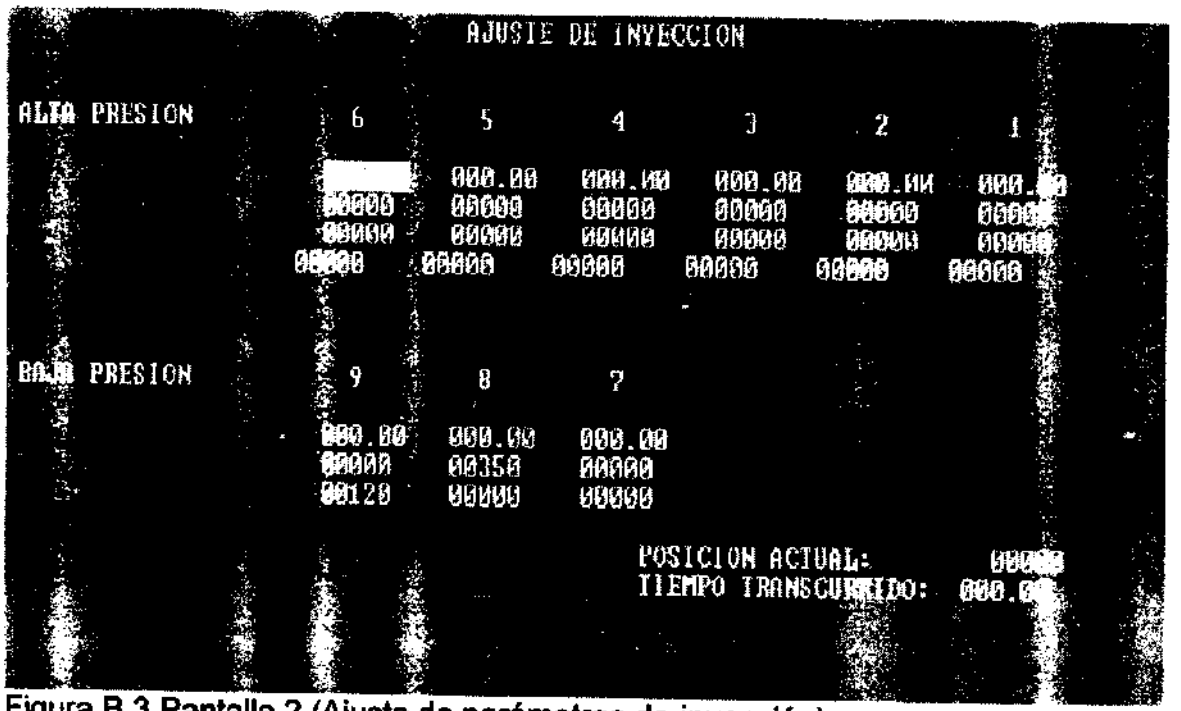

Figura B.3 Pantalla 2 (Ajuste de parámetros de inyección)

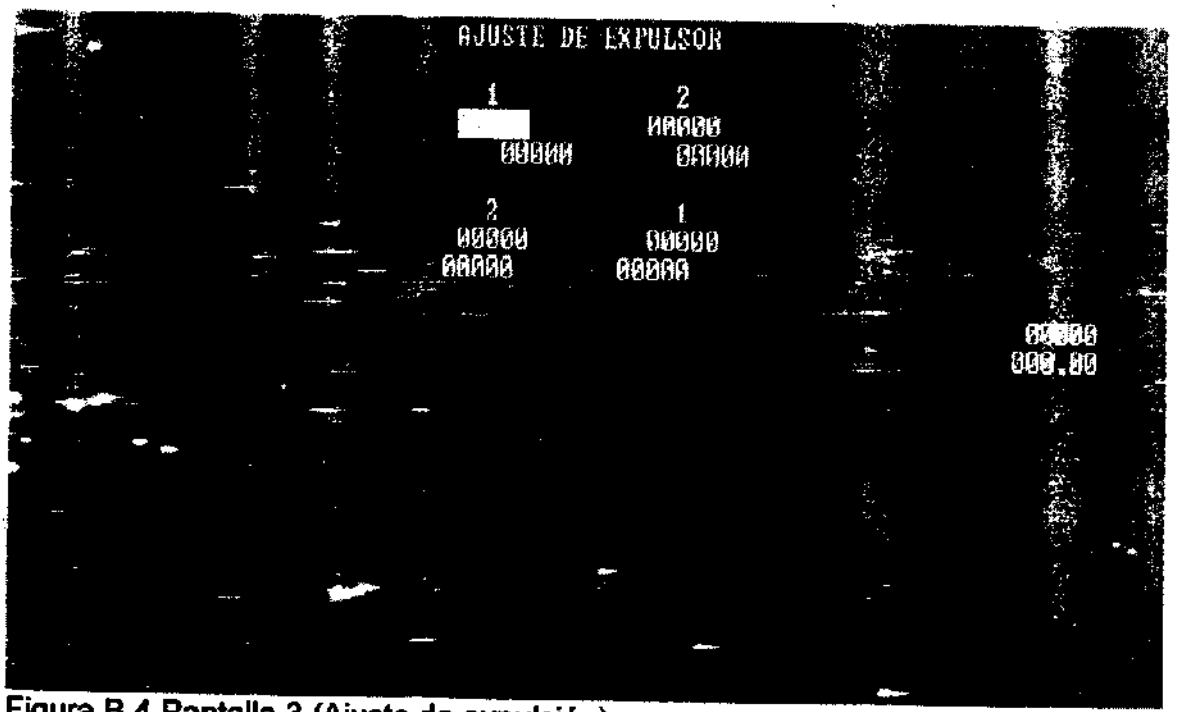

Figura B.4 Pantalla 3 (Ajuste de expulsión)

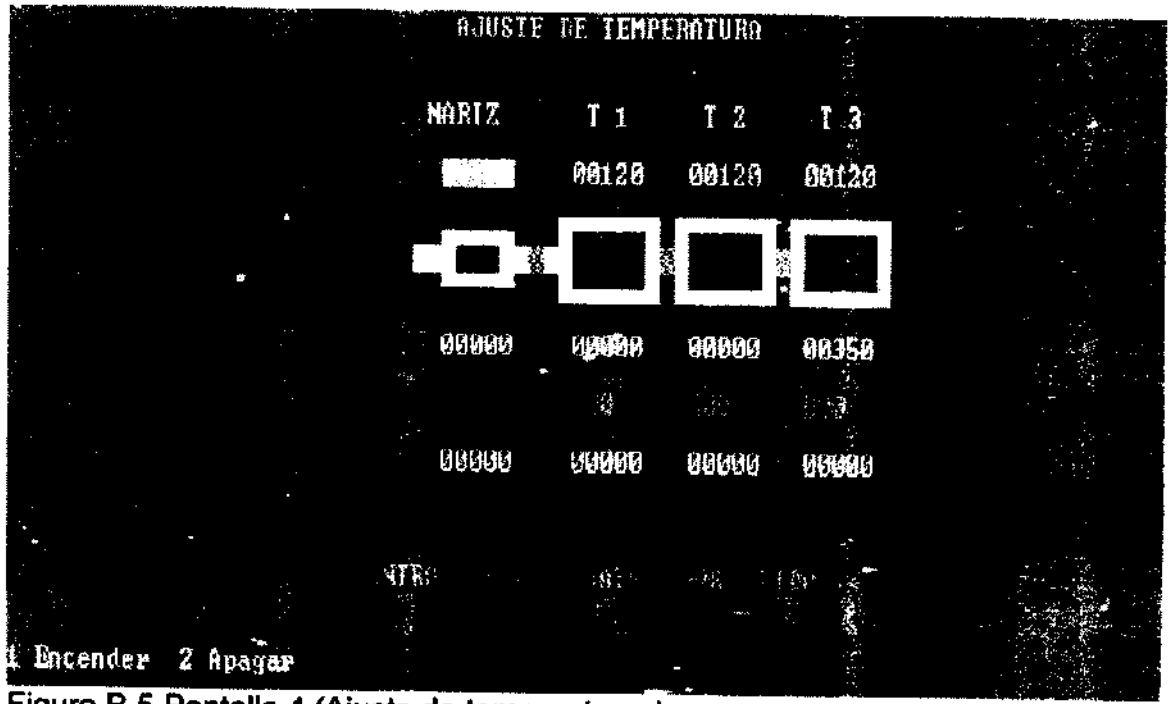

Figura B.5 Pantalla 4 (Ajuste de temperaturas)

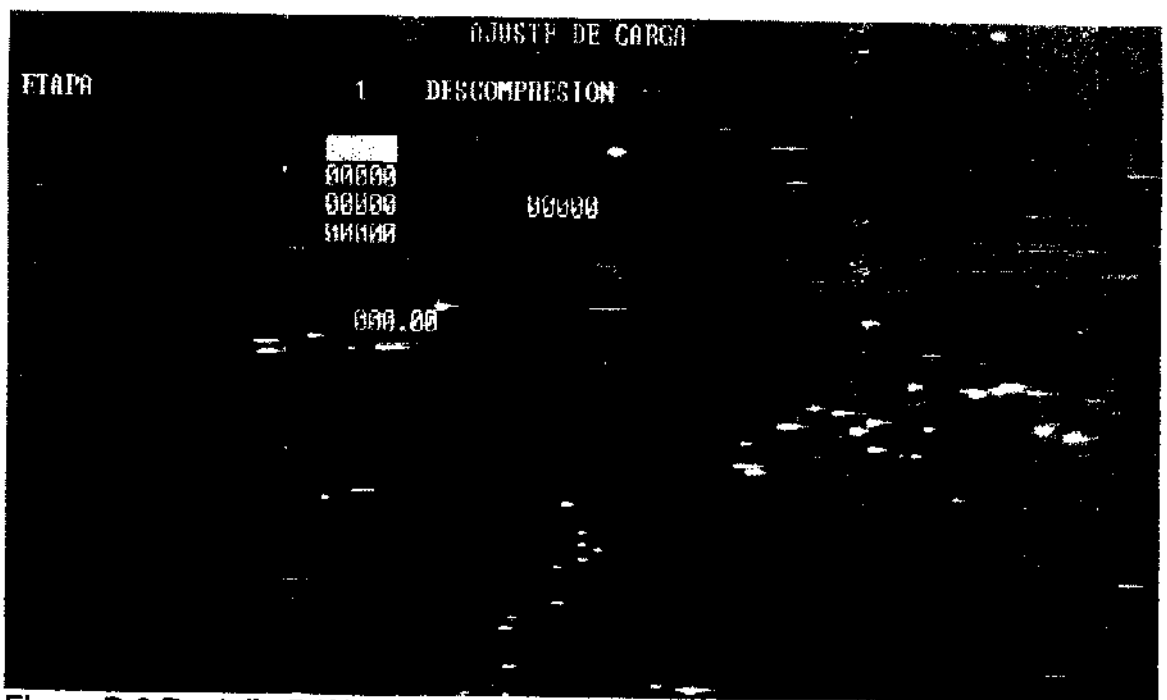

Figura B.6 Pantalla 5 (Ajuste de parámetros de carga)

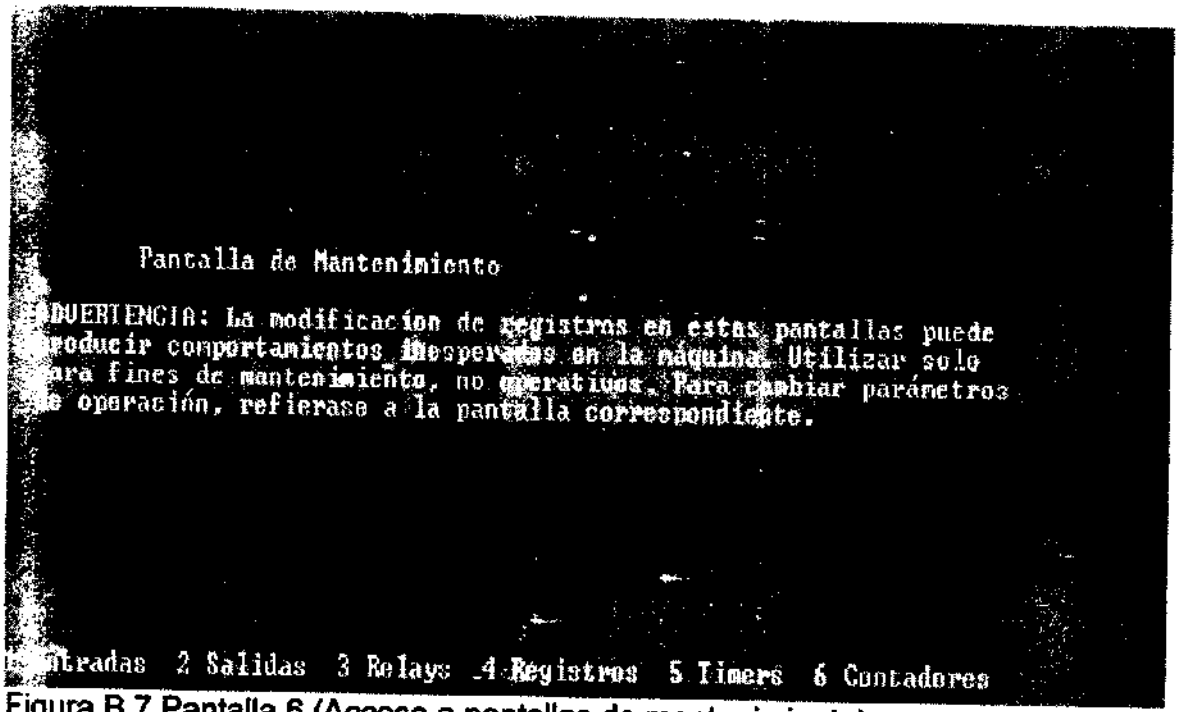

Figura B.7 Pantalla 6 (Acceso a pantallas de mantenimiento)

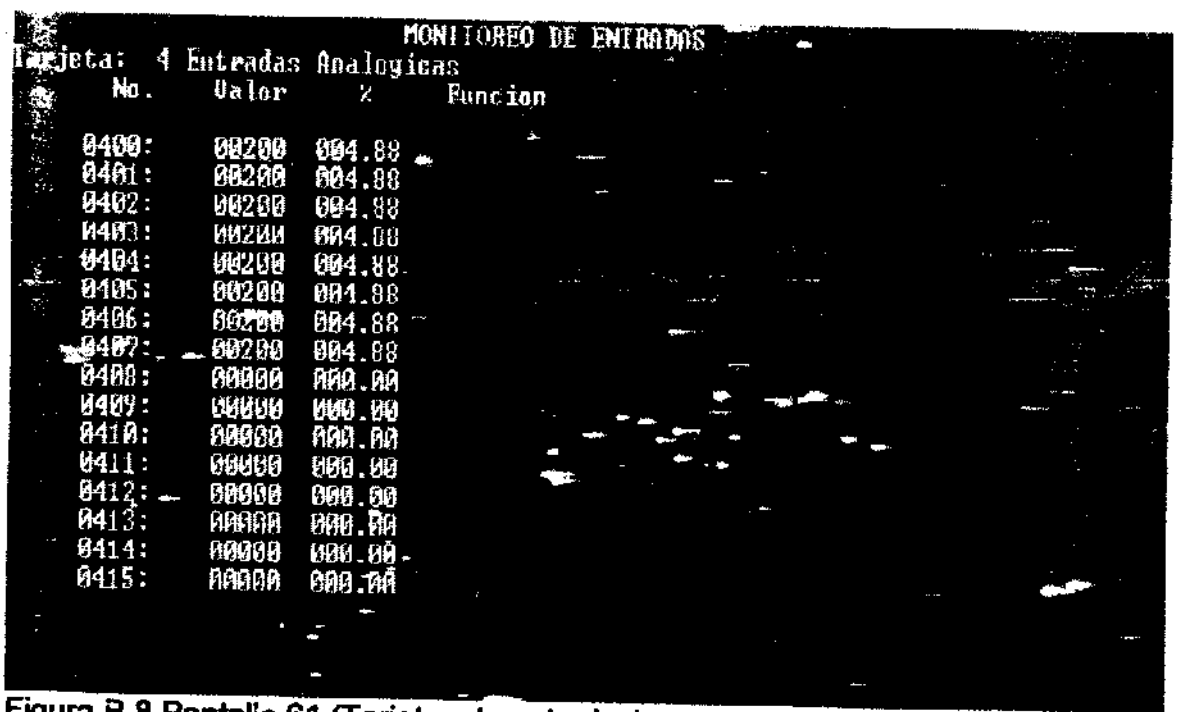

Figura B.8 Pantalla 61 (Tarjetas de entradas)

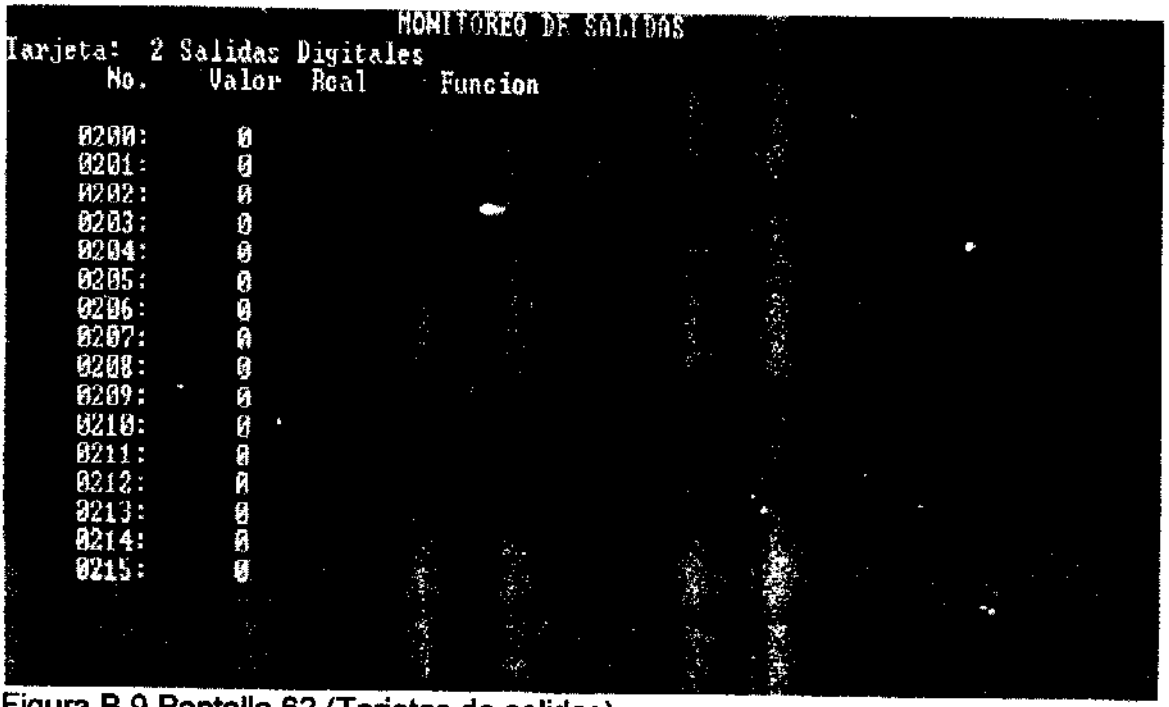

Figura B.9 Pantalla 62 (Tarjetas de salidas)

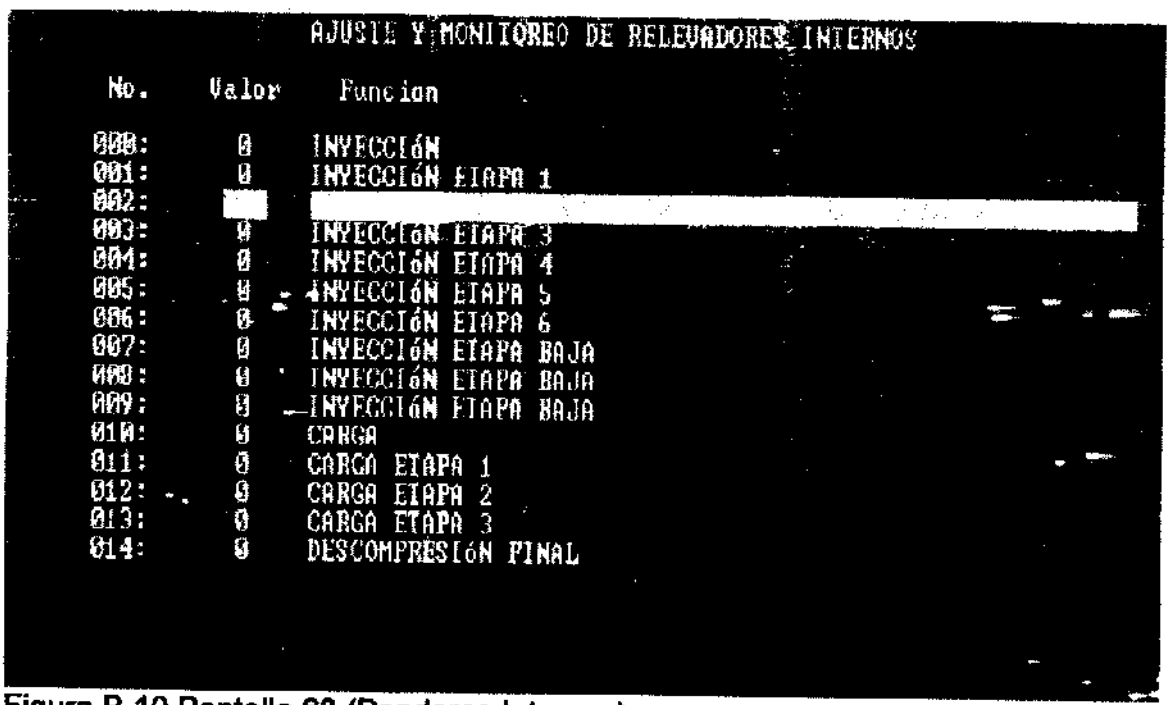

Figura B.10 Pantalla 63 (Banderas Internas)

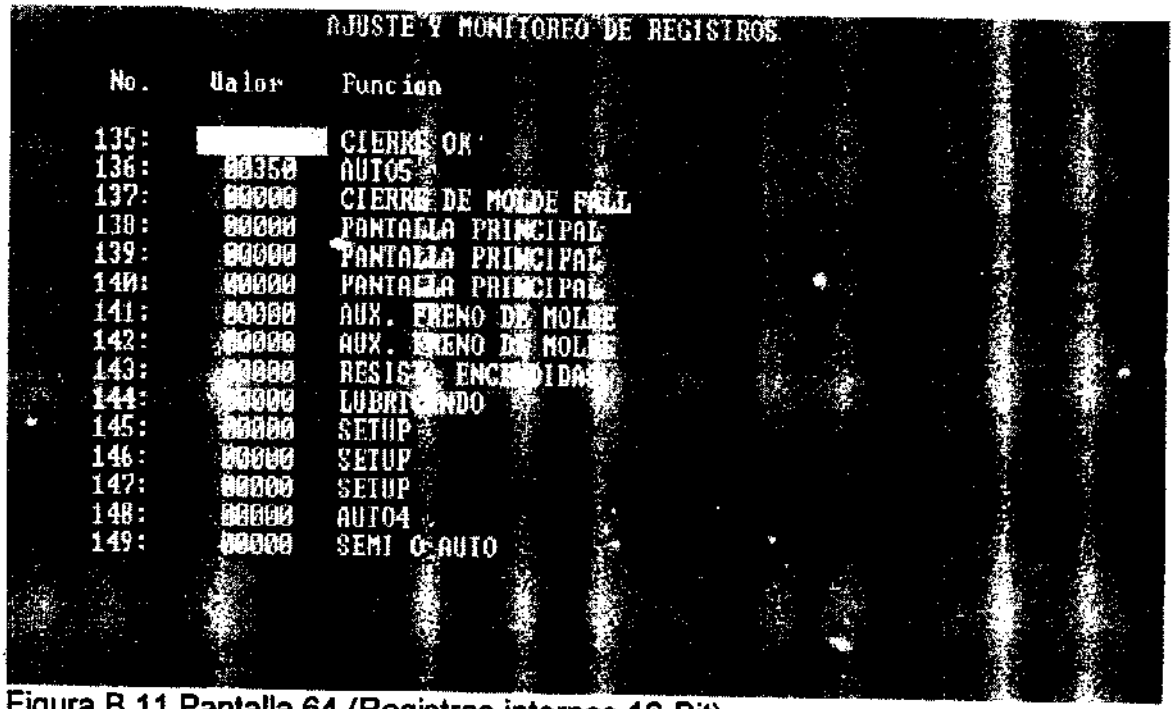

Figura B.11 Pantalla 64 (Registros internos 16-Bit)

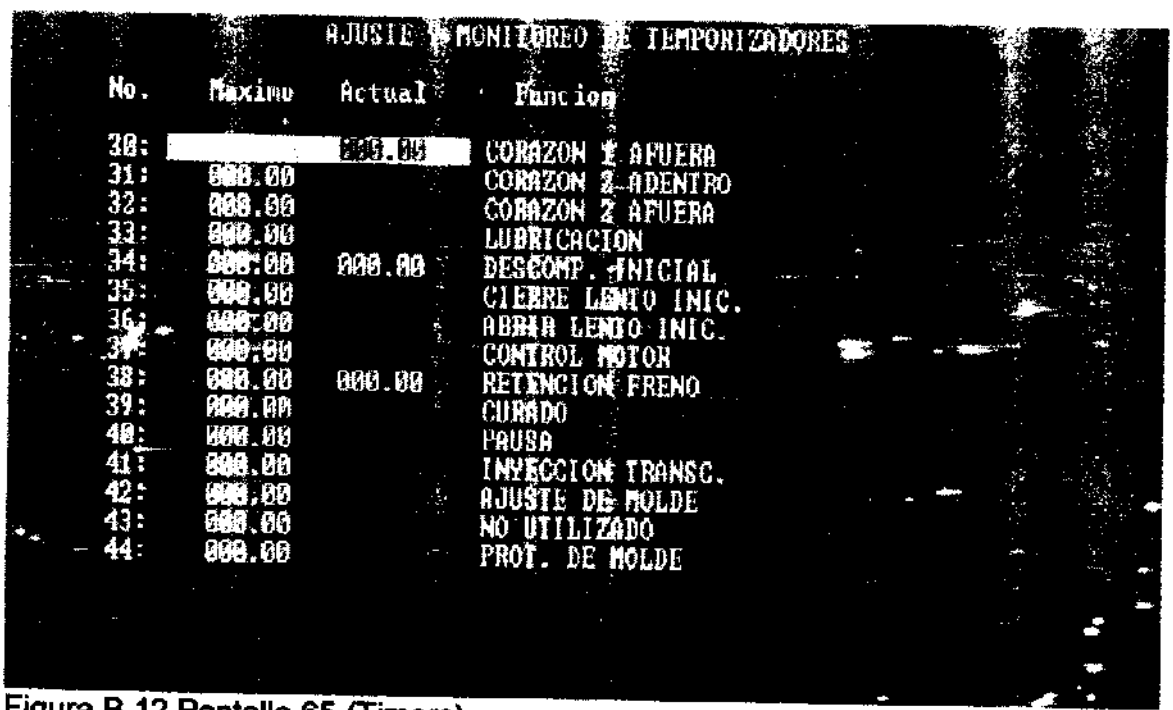

Figura B.12 Pantalla 65 (Timers)

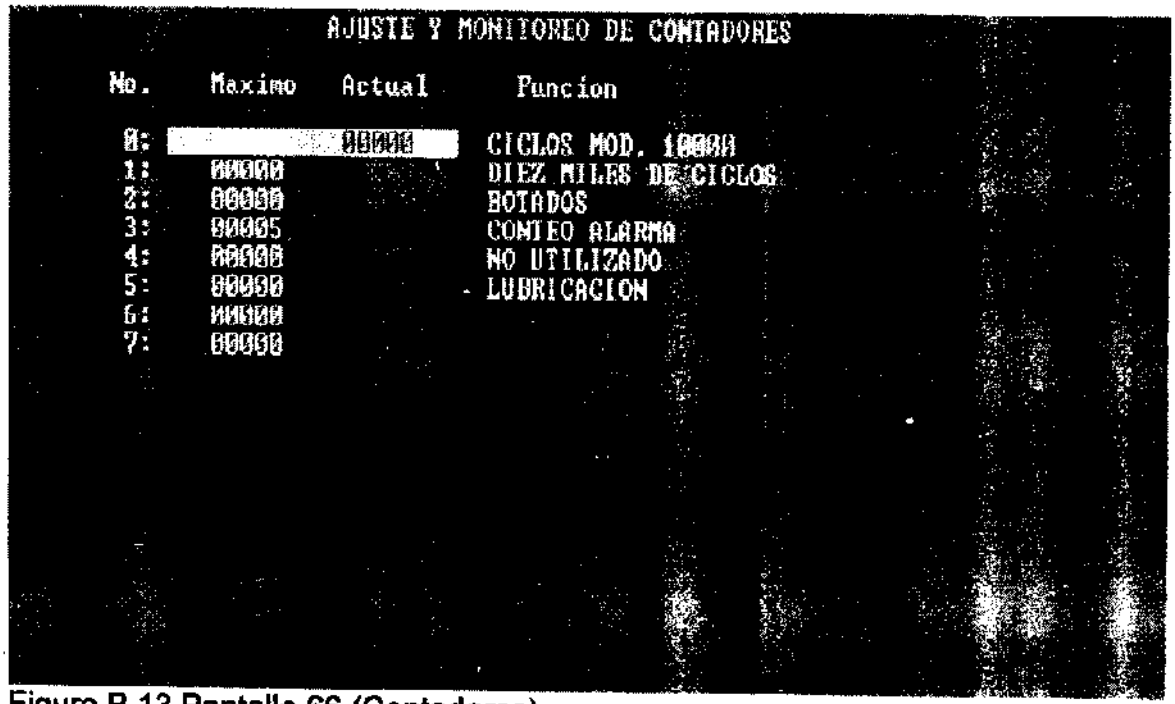

Figura B.13 Pantalla 66 (Contadores)

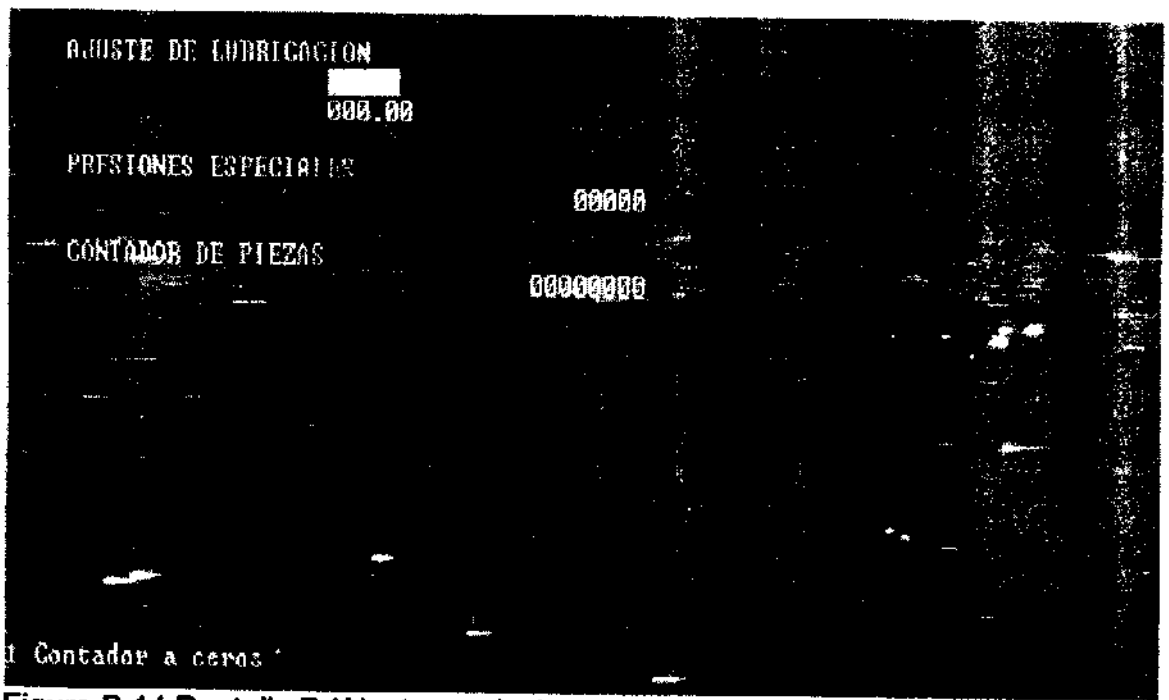

Figura B.14 Pantalla 7 (Ajustes varios)

ANEXO C. CÓDIGOS EN ENSABLADOR PARA LOS MICROCONTROLADORES DE LAS TARJETAS

C.1 Tarjeta de entradas digitales

```
$NOMOD51
$NOLIST
$INCLUDE(RG51FB.PDF)
$LIST

; PROGRAMA PARA TARJETA DE ENTRADAS DIGITALES
; Agustin Bravo Curiel

SP_INIT         EQU 47h         ;STACK POINTER INICIA EN LA DIRECCION 48h
CURR_ADDR       DATA 01h        ;DIRECCION QUE SE ACCESA
DATA_BASE       EQU 30h                 ;INICIO DE ALMACENAMIENTO DE DATOS

DATA_WR         BIT 0
IO_WR           BIT P3.3        ;INTERRUPCION DE ESCRITURA, DESHABILITADA
IO_RD           BIT P3.2        ;INTERRUPCION DE LECTURA
ACK             BIT P3.4
A0              BIT P3.0        ;DIRECCIONES DE ACCESO A LOS
A1              BIT P3.1        ;REGISTROS INTERNOS DE LA TARJETA
A2              BIT P3.5
OE1             BIT P3.6        ;LEER CANALES 8-15
OE2             BIT P3.7        ;LEER CANALES 16-23
DPORT           EQU P0

CICLOS          EQU 0B0h        ;INTERVALO PARA TIEMPO DE FILTRADO

START   EQU 0000h

        ORG START
        AJMP INIT               ;BRINCAR VECTORES DE INTERRUPCION

        ORG START+03h
        AJMP READ

        ORG START+0Bh
;         AJMP TIM_INT

        ORG START+13h
;         AJMP WRITE

        ORG START+1Bh
        AJMP TIM1_INT

        ORG START+23h
;         AJMP S_INT

        ORG START+34h
INIT:
        MOV T2CON,#0

        MOV SP,#SP_INIT         ;INICIALIZAR STACK POINTER
        MOV TMOD,#20h           ;TIMER 1 EN AUTORELOAD
        MOV TH1,#-250           ;INTERVALO PARA TIMER 250 MICROSEGUNDOS

        CLR ES                  ;HABILITAR INTERRUPCIONES Y TIMERS
        SETB TR1
        CLR TR0
        SETB ET1
        SETB IT0
        SETB EX0
```

```
        SETB EA
        SETB PX0
        MOV R7,#CICLOS
        MOV R5,#80

        CLR ACK                         ;ASEGURAR QUE SE LIBERA LA LINEA ACK
LOOP:
        SJMP LOOP

TIM1_INT:                               ;INTERRUPCION DE TIMER 1
        DJNZ R7,OUTINT                  ;250*80=20 MILISEG.
        MOV R7,#CICLOS
        ACALL PROCESA_LECTURA
OUTINT:
        DJNZ R6,OUT1INT
        CPL IO_WR
OUT1INT:
        RETI

PROCESA_LECTURA:                        ;LECTURA DE LOS 3 PUERTOS DE 8 BITS
        MOV R0,#DATA_BASE
        MOV A,P1
        MOV @R0,A
        INC R0
        CLR OE1
        MOV A,P2
        SETB OE1
        MOV @R0,A
        INC R0
        CLR OE2
        MOV A,P2
        SETB OE2
        MOV @R0,A
        RET

READ:                                   ;INTERRUPCION DE LECTURA DESDE LA PC
        CLR EX0
        PUSH ACC
        PUSH 01
        MOV A,P3                        ;GENERAR DIRECCION DE REGISTRO INTERNO
        MOV C,ACC.5
        MOV ACC.2,C
        ANL A,#07h
        ADD A,#DATA_BASE
        MOV R1,A
        MOV A,@R1                       ;TRAER DATO AL PUERTO DE SALIDA
        MOV DPORT,A
        SETB ACK
WAIT_RD:
        JNB IO_RD,WAIT_RD               ;ESPERAR A QUE SUBA LA LINEA DE READ
        MOV DPORT,#0FFh                 ;LIBERAR EL PUERTO
        CLR ACK                         ;LIBERAR ACK
OUT_READ:
        POP 01
        POP ACC
        SETB EX0
        RETI

ZFINAL:
END
```

C.2 Tarjeta de salidas digitales

```
$NOMOD51
$NOLIST
$INCLUDE(RG51FB.PDF)
$LIST

; PROTOCOLO DE COMUNICACIONES CON RACK DE BUS
; Agustin Bravo Curiel

SP_INIT         EQU 47h          ;STACK POINTER INICIA EN LA DIRECCION 48h
CURR_ADDR       DATA 01h
DATA_BASE       EQU 30h
DATA_BASEB      EQU DATA_BASE+8

CYCLEBIT        BIT 1
IO_WR           BIT P3.3
ACK             BIT P1.5
A0              BIT P2.5
A1              BIT P2.6
A2              BIT P2.7
LED             BIT P2.0
DPORT           EQU P0
CH0             BIT P2.1         ;BITS DE SALIDA A LOS CANALES DE POTENCIA
CH1             BIT P2.2
CH2             BIT P2.3
CH3             BIT P2.4
CH4             BIT P3.7
CH5             BIT P3.6
CH6             BIT P3.5
CH7             BIT P3.4
CH8             BIT P3.2
CH9             BIT P3.1
CH10            BIT P1.4
CH11            BIT P3.0
CH12            BIT P1.3
CH13            BIT P1.2
CH14            BIT P1.1
CH15            BIT P1.0
TIMTIME         DATA 2Dh         ;TIEMPOS DEL LED DE DIAGNOSTICO
TIMUP           DATA 2Eh
CYCLETIM        DATA 2Fh
CICLOS          EQU 010h         ;INTERVALO PARA TIEMPO DE FILTRADO

START   EQU 0000h

        ORG START
        AJMP INIT          ;BRINCAR VECTORES DE INTERRUPCION

        ORG START+03h
;         AJMP READ

        ORG START+0Bh
;         AJMP TIM_INT

        ORG START+13h
        AJMP WRITE

        ORG START+1Bh
        AJMP TIM1_INT

        ORG START+23h
;         AJMP S_INT

        ORG START+34h
INIT:
```

```
        MOV SP,#SP_INIT          ;INICIALIZAR STACK POINTER
        MOV TMOD,#20h            ;TIMER 1 EN AUTORELOAD
        MOV TH1,#-250            ;ACTUALIZAR CADA 250 MICROSEG.

        CLR ES                   ;HABILITAR INTERRUPCIONES Y TIMERS
        SETB TR1
        CLR TR0
        SETB EX1
        SETB EA
        SETB PX1
        SETB ET1
        MOV R7,#CICLOS           ;INTERVALO PARA TIEMPO DE FILTRADO
        MOV R6,#2
        MOV TIMUP,#100
        MOV CYCLETIM,#2[illegible]
        MOV TIMTIME,#0
        MOV DPORT,#0FFh
        MOV R0,#8
        MOV A,#0
        MOV R1,#DATA_BASE
RESETVAL:                        ;BORRAR MEMORIA INTERNA
        MOV @R1,A
        INC R1
        DJNZ R0,RESETVAL
        CLR ACK

LOOP:
        ACALL GEN_OUT            ;ACTUALIZAR SALIDAS
        SJMP LOOP

WRITE:                           ;INTERRUPCION DE ESCRITURA DESDE LA PC
        PUSH PSW
        CLR EX1
        PUSH ACC
        PUSH 01
        MOV A,P2
        SWAP A
        RR A
        ANL A,#07h               ;GENERAR DIRECCION INTERNA DE LA TARJETA
        ADD A,#DATA_BASE
        MOV R1,A                 ;GUARDAR DIRECCION EN R1
        MOV A,P0
        MOV @R1,A
        SETB ACK
WAIT_WR:
        JNB IO_WR,WAIT_WR        ;ESPERAR A QUE SUBA LA LINEA DE WRITE
        CLR ACK                  ;LIBERAR ACK
OUT_READ:
        MOV R6,#2
        POP 01
        POP ACC
        POP PSW
        SETB EX1
        RETI

TIM1_INT:                        ;MANEJO DEL LED DE DIAGNOSTICO
        PUSH PSW
        PUSH ACC
        DJNZ R7,OUT1INT
GOTIM1:
        MOV R7,#CICLOS
        MOV A,TIMTIME            ;CONTROL DE TIEMPO DE ENCENDIDO
        INC A
        CJNE A,CYCLETIM,$+3
        JC TIM_MENOR
        DJNZ R6,GOTIM2
        MOV R6,#1
        MOV TIMUP,#20
```

```
        MOV CYCLETIM,#40
        SJMP GOTIM3
GOTIM2:
        MOV TIMUP,#100
        MOV CYCLETIM,#200
GOTIM3:
        CLR A
TIM_MENOR:
        MOV TIMTIME,A
        CJNE A,TIMUP,$+3        ;CONTROL DE TIEMPO DE APAGADO
        JC CYCLEUP
        CLR CYCLEBIT
        SJMP OUT1INT
CYCLEUP:
        SETB CYCLEBIT
OUT1INT:
        MOV C,CYCLEBIT
        CPL C
        MOV LED,C
        POP ACC
        POP PSW
        RETI

GEN_OUT:
        MOV A,DATA_BASE
        CPL A
        CLR C
        RRC A
        MOV CH0,C
        RRC A
        MOV CH1,C
        RRC A
        MOV CH2,C
        RRC A
        MOV CH3,C
        RRC A
        MOV CH4,C
        RRC A
        MOV CH5,C
        RRC A
        MOV CH6,C
        RRC A
        MOV CH7,C

        MOV A,DATA_BASE+1
        CPL A
        CLR C
SEGUNDO:
        RRC A
        MOV CH8,C
        RRC A
        MOV CH9,C
        RRC A
        MOV CH10,C
        RRC A
        MOV CH11,C
        RRC A
        MOV CH12,C
        RRC A
        MOV CH13,C
        RRC A
        MOV CH14,C
        RRC A
        MOV CH15,C
        RET

ZFINAL:
END
```

C.3 Tarjeta de entradas analógicas

```
$NOMOD51
$NOLIST
$INCLUDE(RG51FB.PDF)
$LIST

; PROGRAMA DE CONTROL DE TARJETA DE 16 ENTRADAS ANALOGICAS
; Agustin Bravo Curiel

SP_INIT         EQU 57h          ;STACK POINTER INICIA EN LA DIRECCION 58h
CURR_ADDR       DATA 01h
DATA_BASE       EQU 30h

DATA_WR         BIT 0
IO_WR           BIT P3.3        ;PIN DE INTERRUPCION DE ESCRITURA DESDE LA PC
IO_RD           BIT P3.2        ;PIN DE INTERRUPCION DE LECTURA DESDE LA PC
ACK             BIT P2.7        ;PIN DE ACKNOWLEDGE
A0              BIT P1.5        ;DIRECCIONAMIENTO DE LA TARJETA
A1              BIT P1.6
A2              BIT P1.7
OE0             BIT P2.0        ;LEER LSB DE ADC
OE1             BIT P2.1        ;LEER MSB DE ADC
DPORT           EQU P0          ;PUERTO DE DATOS DE E/S AL RACK
ADC             EQU P1          ;PUERTO DE MULTIPLEXION DEL ADC
CAZ             BIT P3.0        ;ADC AUTOZERO
LED             bit 07Fh        ;BIT AUXILIAR
SEMAPHORE       BIT 07Eh        ;BIT AUXILIAR
CCS             BIT P3.5        ;ADC CHIP SELECT
CCAL            BIT P3.4        ;ADC CALIBRATE
CWRITE          BIT P3.1        ;ADC CONVERSION START
CREAD           BIT P3.6        ;ADC READ
CEOC            BIT P3.7        ;ADC END OF CONVERSION
CVTH            DATA 10h        ;MSB DE LA CONVERSION
CVTL            DATA 11h        ;LSB DE LA CONVERSION
CHAN            DATA 12h        ;CANAL DE CONVERSION
PREVCHAN        DATA 13h        ;CANAL ANTERIOR DE CONVERSION
CVTH1           DATA 14h        ;MSB LECTURA 1
CVTL1           DATA 15h        ;LSB LECTURA 1
CVTH2           DATA 16h        ;MSB LECTURA 2
CVTL2           DATA 17h        ;LSB LECTURA 2
CVTH3           DATA 18h        ;MSB LECTURA 3
CVTL3           DATA 19h        ;LSB LECTURA 3
CMPH1           DATA 23h        ;MSB COMPARACION 1
CMPL1           DATA 24h        ;LSB COMPARACION 1
CMPH2           DATA 25h        ;MSB COMPARACION 2
CMPL2           DATA 26h        ;LSB COMPARACION 2
SEQREAD         DATA 29h        ;REGISTRO DE LECTURA EN SECUENCIA
OFFSET          DATA 21h        ;REGISTRO PARA CALCULO DE DIRECCIONAMIENTO
READREG         DATA 22h        ;REGISTRO A LEER
INHIBIT0        BIT P2.6        ;HABILITAR MULTIPLEXOR 1
INHIBIT1        BIT P2.5        ;HABILITAR MULTIPLEXOR 2
DIRC            BIT P2.4        ;DIRECCION DEL CANAL A MULTIPLEXAR
DIRB            BIT P2.3
DIRA            BIT P2.2

CICLOS          EQU 04

START   EQU 0000h

        ORG START
        AJMP INIT       ;BRINCAR VECTORES DE INTERRUPCION

        ORG START+03h
        AJMP READ
```

```
        ORG START+0Bh
;         AJMP TIM_INT

        ORG START+13h
        AJMP WRITE

        ORG START+1Bh
        AJMP TIM1_INT

        ORG START+23h
;         AJMP S_INT

        ORG START+34h
INIT:
         ACALL CALIBRAR
        MOV T2CON,#0

        MOV SP,#SP_INIT          ;INICIALIZAR STACK POINTER
        MOV PCON,#80h            ;CONFIGURACION DEL PUERTO SERIAL
        MOV TMOD,#20h            ;TIMER 1 EN AUTORELOAD
        MOV TH1,#-250           ;250 MICROSEG.

NOSERIAL:
        CLR ES                  ;HABILITAR INTERRUPCIONES Y TIMERS
         SETB TR1
        CLR TR0
        SETB ET1
        SETB IT0
         SETB EX0
         SETB EX1
        SETB EA
        ACALL CALIBRAR          ;CALIBRACION INICIAL DEL ADC
        MOV R7,#CICLOS
        MOV READREG,#0
        CLR ACK
        mov cvtl,#0
        mov cvth,#0
protocol:                       ;INICIAR CILO DE MUESTREOS EN EL CANAL 15
         mov r2,#16
protloop:
        mov a,r2
        dec a
         mov chan,a
        JNB SEMAPHORE,$
        CLR SEMAPHORE
        acall ADCCONVERT        ;LLAMAR SUBRUTINA DE CONVERSION DEL ADC
        mov a,chan
        ACALL CVT_CHANNEL       ;ORDENAR CANAL SEGUN SU CONEXION AL MULTIPLEXOR
        rl a
        add a,#data_base        ;GENERAR DIRECCION INTERNA PARA GUARDAR EL DATO
         CLR EA                 ;ZONA CRITICA DE CODIGO REENTRANTE
        mov r1,a
        mov a,cvtl
        mov @r1,a
        inc r1
        mov a,cvth
        mov @r1,a
         SETB EA
         djnz r2,protloop       ;DECREMENTAR CONTADOR Y MUESTREAR SIGUIENTE CANAL
         SJMP PROTOCOL

TIM1_INT:                       ;CONTROL DE BANDERAS AUXILIARES DE TIEMPO DE MUESTREO
        PUSH ACC
        DJNZ R7,OUT1INT
        MOV R7,#CICLOS
        SETB SEMAPHORE
OUTINT:
        DJNZ R6,OUT1INT
```

```
        mov r6,#25
        CPL led
        jnb led,outlint
        CLR CAZ
OUTIINT:
        POP ACC
        RETI


WRITE:                          ;ESCRITURA DE LA PC DE LA DIRECCION A LEER
        CLR EX1
        PUSH ACC
        SETB OE0
        CLR OE1
        acall waiter
        MOV A,P0
        PUSH ACC
        MOV A,ADC               ;GUARDAR LA DIRECCION
        SETB ACK
WAIT_WR:
        JNB IO_WR,WAIT_WR        ;ESPERAR A QUE SUBA LA LINEA DE WRITE
        CLR ACK                  ;LIBERAR ACK
        SWAP A
        RR A
        ANL A,#07h
        JZ NORMALACCESS
SEQACCESS:
        POP SEQREAD
        SJMP OUT_WRITE
NORMALACCESS:
        POP READREG
OUT_WRITE:
        POP ACC
        SETB EX1
        RETI

READ:
        CLR EX0
        PUSH ACC
        PUSH 01
        SETB OE0
        CLR OE1
        acall waiter
        MOV A,ADC               ;GENERAR DIRECCION INTERNA DE LECTURA
        SWAP A
        RR A
        ANL A,#07h
        JZ NORMALREAD
        MOV A,SEQREAD
        PUSH ACC
        INC A
        MOV SEQREAD,A
        POP ACC
        SJMP BRINGREG
NORMALREAD:
        MOV A,READREG
BRINGREG:                       ;TRAER DATO DE LA MEMORIA INTERNA
        ADD A,#DATA_BASE
        MOV R1,A
        MOV A,@R1
TXBYTE:
        MOV DPORT,A             ;COLOCAR DATO EN EL BUS
HSHAKE:
        SETB ACK
WAIT_RD:
        JNB IO_RD,WAIT_RD        ;ESPERAR A QUE SUBA LA LINEA DE READ
        MOV DPORT,#0FFh          ;LIBERAR EL PUERTO
        CLR ACK                  ;LIBERAR ACK
```

```
OUT_READ:
        POP 01
        POP ACC
;         SETB EX0
        RETI

ADCCONVERT:                     ;RUTINA DE MUESTREO TRIPLE Y FILTRO MEDIANA
        MOV A,CHAN
        ACALL ADCCONV
        MOV A,CVTL
        MOV CVTL1,A
        MOV A,CVTH
        MOV CVTH1,A

        MOV A,CHAN
        ACALL ADCCONV
        MOV A,CVTL
        MOV CVTL2,A
        MOV A,CVTH
        MOV CVTH2,A

        MOV A,CHAN
        ACALL ADCCONV
        MOV A,CVTL
        MOV CVTL3,A
        MOV A,CVTH
        MOV CVTH3,A

                                ;COMPARAR 1 CONTRA 2 Y ORDENAR
        MOV CMPH1,CVTH1
        MOV CMPL1,CVTL1
        MOV CMPH2,CVTH2
        MOV CMPL2,CVTL2
        ACALL ORDER
        MOV CVTH1,CMPH1
        MOV CVTL1,CMPL1
        MOV CVTH2,CMPH2
        MOV CVTL2,CMPL2

                                ;COMPARAR 2 CONTRA 3 Y ORDENAR
        MOV CMPH1,CVTH2
        MOV CMPL1,CVTL2
        MOV CMPH2,CVTH3
        MOV CMPL2,CVTL3
        ACALL ORDER
        MOV CVTH2,CMPH1
        MOV CVTL2,CMPL1
        MOV CVTH3,CMPH2
        MOV CVTL3,CMPL2

                                ;COMPARAR 1 CONTRA 2 Y ORDENAR
        MOV CMPH1,CVTH1
        MOV CMPL1,CVTL1
        MOV CMPH2,CVTH2
        MOV CMPL2,CVTL2
        ACALL ORDER
        MOV CVTH1,CMPH1
        MOV CVTL1,CMPL1
        MOV CVTH2,CMPH2
        MOV CVTL2,CMPL2

        MOV CVTL,CVTL2
        MOV CVTH,CVTH2
        RET

ADCCONV:                        ;CONECTAR EL CANAL SELECCIONADO AL MUX
        MOV C,ACC.0
        MOV DIRA,C
        MOV C,ACC.1
        MOV DIRB,C
        MOV C,ACC.2
```

```
        MOV DIRC,C
        MOV C,ACC.3
        MOV INHIBIT1,C
        CPL C
        MOV INHIBIT0,C
        ACALL WAITER
        CLR CCS                 ;INICIAR CONVERSION
        CLR CWRITE
        CLR EA
        SETB CWRITE
        JB CEOC,$               ;ESPERAR FIN DE CONVERSION
        SETB EA
        JNB CEOC,$
        CLR CWRITE
        CLR CCS
        CLR CREAD               ;LEER DATOS DEL ADC
        SETB OE1
        CLR OE0
        acall waiter
        MOV A,ADC
        ACALL INVERT
        MOV CVTL,A
        SETB OE0
        CLR OE1
        acall waiter
        MOV A,ADC
        SETB CREAD
        SETB CCS
        ACALL INVERT            ;ORDENAR BITS
        RR A
        RR A
        RR A
        ANL A,#01FH
        JNB ACC.4,CONV_OK       ;ELIMINAR LECTURAS NEGATIVAS
        MOV CVTL,#0
        CLR A
CONV_OK:
        MOV CVTH,A
        SETB CAZ
        RET

INVERT:                         ;INVERTIR ORDEN DE BITS EN EL ACUMULADOR
        PUSH 03
        PUSH 04
        PUSH 05
        MOV R3,#8
CAMBIOS:
        RLC A
        MOV R4,A
        MOV A,R5
        RRC A
        MOV R5,A
        MOV A,R4
        DJNZ R3,CAMBIOS
        MOV A,R5
        POP 05
        POP 04
        POP 03
        RET

CALIBRAR:                       ;RUTINA DE CALIBRACION DEL ADC
        SETB CCS
        SETB CREAD
        SETB CWRITE
        CLR EA
        CLR CCAL
        JB CEOC,$
        SETB EA
```

```
        SETB CCAL
        JNB CEOC,$
        RET

CVT_CHANNEL:                    ;AJUSTA EL NUMERO DE CANAL AL LAYOUT DE LA TARJETA
        MOV DPTR,#BASE_TABLA
        MOVC A,@A+DPTR
        RET
BASE_TABLA:
        DB 12d
        DB 10d
        DB 11d
        DB 14d

        DB 09d
        DB 15d
        DB 08d
        DB 13d

        DB 04d
        DB 05d
        DB 07d
        DB 06d

        DB 01d
        DB 03d
        DB 00d
        DB 02d

ORDER:                          ;ORDENAR DOS DATOS DE 16 BITS
        MOV A,CMPH1
        CJNE A,CMPH2,COMPARE1
        SJMP NOCAMBIES1
COMPARE1:
        JC NOCAMBIES2
        ACALL SWAPCOMP
        RET
NOCAMBIES1:
        MOV A,CMPL1
        CJNE A,CMPL2,COMPARE2
        SJMP NOCAMBIES2
COMPARE2:
        JC NOCAMBIES2
        ACALL SWAPCOMP
NOCAMBIES2:
        RET

SWAPCOMP:
        PUSH CMPH1
        PUSH CMPL1
        PUSH CMPH2
        PUSH CMPL2
        POP CMPL1
        POP CMPH1
        POP CMPL2
        POP CMPH2
        RET

WAITER:
        NOP
        NOP
        NOP
        NOP
        NOP
        RET
ZFINAL:
END
```

C.4 Tarjeta de salidas analógicas

```
$NOMOD51
$NOLIST
$INCLUDE(RG51FB.PDF)
$LIST

; PROGRAMA DE CONTROL PARA TARJETA DE 8 SALIDAS ANALOGICAS
; Agustin Bravo Curiel

SP_INIT         EQU 47h          ;STACK POINTER INICIA EN LA DIRECCION 48h
CURR_ADDR       DATA 01h
DATA_BASE       EQU 30h

DATA_WR         BIT 0
IO_WR           BIT P3.3        ;INTERRUPCION DE ESCRITURA DE LA PC
IO_RD           BIT P3.2        ;INTERRUPCIN DE LECTURA DE LA PC, DESHABILITADA
ACK             BIT P3.4        ;ACKNOWLEDGE
A0              BIT P3.6        ;DIRECCIONAMIENTO DE LA TARJETA
A1              BIT P3.7
A2              BIT P3.5
DPORT           EQU P0          ;PUERTO DE DATOS DEL RACK
DAC             EQU P1          ;SALIDA AL DAC
AD0             BIT P2.0        ;DIRECCION DEL CANAL A MUESTREAR
AD1             BIT P2.1
AD2             BIT P2.2
INHIBIT         BIT P2.3        ;HABILITAR MUESTREO Y RETENCION

CICLOS          EQU 10

START   EQU 0000h

        ORG START
        AJMP INIT       ;BRINCAR VECTORES DE INTERRUPCION

        ORG START+03h
;         AJMP READ

        ORG START+0Bh
;         AJMP TIM_INT

        ORG START+13h
        AJMP WRITE

        ORG START+1Bh
        AJMP TIM1_INT

        ORG START+23h
;         AJMP S_INT

        ORG START+34h
INIT:
        MOV T2CON,#0

        MOV SP,#SP_INIT          ;INICIALIZAR STACK POINTER
        MOV PCON,#80h            ;CONFIGURACION DEL PUERTO SERIAL
        MOV TMOD,#20h            ;TIMER 1 EN AUTORELOAD
        MOV TH1,#-250            ;250 MICROSEG.

        CLR ES                  ;HABILITAR TIMERS E INTERRUPCIONES
        SETB TR1
        CLR TR0
        SETB EX1
        SETB EA
        SETB PX1
        SETB ET1
```

```
        MOV R7,#CICLOS
        MOV DPORT,#0FFh
        MOV R0,#8
        MOV A,#0
        MOV R1,#DATA_BASE
RESETVAL:                       ;INICIALIZAR MEMORIA INTERNA
        MOV @R1,A
        INC R1
        DJNZ R0,RESETVAL
        CLR ACK

LOOPBASE:                       ;CICLO DE MUESTREO Y RETENCION, INICIA EN CANAL 7
        MOV R2,#8
        MOV R1,#DATA_BASE+7
LOOP:
        MOV A,@R1
        SETB INHIBIT
        MOV R3,#8
nocambios:
        MOV DAC,A               ;MANDAR DIRECCION DEL CANAL AL DEMUX

      - MOV A,R2
        DEC A
        RRC A
        MOV AD0,C
        RRC A
        MOV AD1,C
        RRC A
        MOV AD2,C

        CLR INHIBIT             ;HABILITAR MUESTREO
        ACALL WAIT_SETTLE       ;ESPERAR CARGA DEL CAPACITOR
        SETB INHIBIT            ;DESHABILITAR MUESTREO (RETENCION)
        DEC R1
        DJNZ R2,LOOP
        SJMP LOOPBASE           ;SIGUIENTE CANAL

WAIT_SETTLE:                    ;TIEMPO DE MUESTREO
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        RET

WRITE:                          ;ESCRITURA DE LA PC A LA TARJETA
        CLR EX1
        PUSH ACC
        PUSH 01
        MOV A,P3
        MOV C,ACC.5
        MOV ACC.2,C
        MOV C,ACC.6
        MOV ACC.0,C
        MOV C,ACC.7
        MOV ACC.1,C
        ANL A,#07h
        ADD A,#DATA_BASE
        MOV R1,A                ;GUARDAR DIRECCION EN R1
        MOV A,PC
        MOV @R1,A
        SETB ACK
WAIT_WR:
```

```
        JNB IO_WR,WAIT_WR       ;ESPERAR A QUE SUBA LA LINEA DE WRITE
        CLR ACK                 ;LIBERAR ACK
OUT_READ:
        POP 01
        POP ACC
        SETB EX1
        RETI

TIM1_INT:
        DJNZ R7,OUT1INT
        MOV R7,#CICLOS
        DJNZ R6,OUT1INT
        CPL IO_RD               ;CONTROL DE LED DE DIAGNOSTICO
OUT1INT:
        RETI
ZFINAL:
END
```

C.5 Tarjeta de entradas de termopar

```
$NOMOD51
$NOLIST
$INCLUDE(RG51FB.PDF)
$LIST

; PROTOCOLO DE COMUNICACIONES CON RACK DE BUS
; Agustin Bravo Curiel

SP_INIT         EQU 67h          ;STACK POINTER INICIA EN LA DIRECCION 58h
CURR_ADDR       DATA 01h
DATA_BASE       EQU 30h
DATA_STORE      EQU 40h
DATA_WR         BIT 0
IO_WR           BIT P3.3        ;PIN DE INTERRUPCION DE ESCRITURA
IO_RD           BIT P3.2        ;PIN DE INTERRUPCION DE LECTURA
ACK             BIT P2.7        ;PIN DE ACKNOWLEDGE
A0              BIT P1.5        ;DIRECCIONAMIENTO DE LA TARJETA
A1              BIT P1.6
A2              BIT P1.7
OE0             BIT P2.0        ;LEER LSB DE ADC
OE1             BIT P2.1        ;LEER MSB DE ADC
DPORT           EQU P0          ;PUERTO DE DATOS DE E/S AL RACK
ADC             EQU P1          ;PUERTO DE MULTIPLEXION DEL ADC
CAZ             BIT P3.0        ;ADC AUTOZERO
LED             bit 07Fh        ;BIT AUXILIAR
SEMAPHORE       BIT 07Eh        ;BIT AUXILIAR
CCS             BIT P3.5        ;ADC CHIP SELECT
CCAL            BIT P3.4        ;ADC CALIBRATE
CWRITE          BIT P3.1        ;ADC CONVERSION START
CREAD           BIT P3.6        ;ADC READ
CEOC            BIT P3.7        ;ADC END OF CONVERSION
CVTH            DATA 10h        ;MSB DE LA CONVERSION
CVTL            DATA 11h        ;LSB DE LA CONVERSION
CHAN            DATA 12h        ;CANAL DE CONVERSION
PREVCHAN        DATA 13h        ;CANAL ANTERIOR DE CONVERSION
CVTH1           DATA 14h        ;MSB LECTURA 1
CVTL1           DATA 15h        ;LSB LECTURA 1
CVTH2           DATA 16h        ;MSB LECTURA 2
CVTL2           DATA 17h        ;LSB LECTURA 2
CVTH3           DATA 18h        ;MSB LECTURA 3
CVTL3           DATA 19h        ;LSB LECTURA 3
CMPH1           DATA 23h        ;MSB COMPARACION 1
CMPL1           DATA 24h        ;LSB COMPARACION 1
CMPH2           DATA 25h        ;MSB COMPARACION 2
CMPL2           DATA 26h        ;LSB COMPARACION 2
RDREGH          DATA 27h        ;REGISTRO DE LECTURA MSB
RDREGL          DATA 28h        ;REGISTRO DE LECTURA LSB
SEQREAD         DATA 29h        ;REGISTRO DE LECTURA EN SECUENCIA
SUMH            DATA 2Ah
SUML            DATA 2Bh
OFFSET          DATA 21h        ;REGISTRO PARA CALCULO DE DIRECCIONAMIENTO
READREG         DATA 22h        ;REGISTRO A LEER
ALARM           BIT P2.6        ;ALARMA DE TERMOPAR
LED             BIT P2.5        ;AUXILIAR
DIRC            BIT P2.4        ;DIRECCION DEL CANAL A MULTIPLEXAR
DIRB            BIT P2.3
DIRA            BIT P2.2

CICLOS          EQU 030h

START   EQU 0000h

        ORG START
        AJMP INIT       ;BRINCAR VECTORES DE INTERRUPCION
```

```
        ORG START+03h
        AJMP READ

        ORG START+0Bh
;         AJMP TIM_INT

        ORG START+13h
        AJMP WRITE

        ORG START+1Bh
        AJMP TIM1_INT

        ORG START+23h
;         AJMP S_INT

        ORG START+34h
INIT:
        ACALL CALIBRAR
        MOV T2CON,#0

      . MOV SP,#SP_INIT          ;INICIALIZAR STACK POINTER
        MOV TMOD,#20h            ;TIMER 1 EN AUTORELOAD
        MOV TH1,#088h           ;120 MICROSEG

NOSERIAL:                       ;HABILITAR INTERRUPCIONES Y TIMERS
        CLR ES
        SETB TR1
        CLR TR0
        SETB ET1
        SETB IT0
        SETB EX0
        SETB EX1
        SETB EA
        ACALL CALIBRAR
        MOV R7,#CICLOS
        MOV READREG,#0
        CLR ACK
        mov cvtl,#0
        mov cvth,#0
protocol:
        mov r2,#8               ;MUESTREAR CANALES, INICIO EN EL 7
protloop:
        mov a,r2
        dec a
        mov chan,a
        JNB SEMAPHORE,$
        CLR SEMAPHORE
        acall ADCCONVERT
        ACALL INSERT            ;INSERTAR EN EL FILTRO MEDIANA PROMEDIO

        mov a,chan
        ACALL CVT_CHANNEL
        rl a
        add a,#data_base
        CLR EA                  ;ZONA CRITICA
        mov r1,a
        mov a,cvtl
        mov @r1,a
        inc r1
        mov a,cvth
        mov @r1,a
        SETB EA
        djnz r2,protloop        ;MUESTREAR SIGUIENTE CANAL
        JNB SEMAPHORE,$
        CLR SEMAPHORE
        SJMP PROTOCOL
```

```
TIM1_INT:                       ;ESPERAR TIEMPO DE MUESTREO
        PUSH ACC
        DJNZ R7,OUT1INT
        MOV R7,#CICLOS
         SETB SEMAPHORE
OUTINT:
        DJNZ R6,OUT1INT
         SETB SEMAPHORE
         mov r6,#200
        CPL led
        jnb led,outlint
         CLR CAZ
OUT1INT:
        POP ACC
        RETI


WRITE:                          ;ESCRITURA DE LA PC A LA TARJETA
        CLR EX1
         PUSH PSW
        PUSH ACC
      .  SETB OE0
         CLR OE1
         acall waiter
         MOV A,P0
         PUSH ACC
         MOV A,ADC
        SETB ACK
WAIT_WR:
        JNB IO_WR,WAIT_WR        ;ESPERAR A QUE SUBA LA LINEA DE WRITE
        CLR ACK                  ;LIBERAR ACK
         SWAP A
         RR A
         ANL A,#07h
         JZ NORMALACCESS
SEQACCESS:
         POP SEQREAD
         SJMP OUT_WRITE
NORMALACCESS:
         POP READREG
OUT_WRITE:
        POP ACC
         POP PSW
        SETB EX1
        RETI

READ:                           ;LECTURA DE LA PC A LA TARJET
         CLR EX0
        PUSH ACC
         PUSH PSW
        PUSH 01
        SETB OE0
        CLR OE1
         acall waiter
         MOV A,ADC
         SWAP A
         RR A
         ANL A,#07h
         JZ NORMALREAD
         MOV A,SEQREAD
         PUSH ACC
         INC A
         MOV SEQREAD,A
         POP ACC
         SJMP BRINGREG
NORMALREAD:
       MOV A,READREG
BRINGREG:
```

```
        ADD A,#DATA_BASE
        MOV R1,A
        MOV A,@R1
TXBYTE:
        MOV DPORT,A
HSHAKE:
        SETB ACK
WAIT_RD:
        JNB IO_RD,WAIT_RD       ;ESPERAR A QUE SUBA LA LINEA DE READ
        MOV DPORT,#0FFh         ;LIBERAR EL PUERTO
        CLR ACK                 ;LIBERAR ACK
OUT_READ:
        POP 01
        POP PSW
        POP ACC
        SETB EX0
        RETI

ADCCONVERT:                     ;MUESTREO Y FILTRO MEDIANA
        MOV A,CHAN
        ACALL ADCCONV
      . MOV A,CVTL
        MOV CVTL1,A
        MOV A,CVTH
        MOV CVTH1,A

        MOV A,CHAN
        ACALL ADCCONV
        MOV A,CVTL
        MOV CVTL2,A
        MOV A,CVTH
        MOV CVTH2,A

        MOV A,CHAN
        ACALL ADCCONV
        MOV A,CVTL
        MOV CVTL3,A
        MOV A,CVTH
        MOV CVTH3,A
                                ;COMPARAR 1 CONTRA 2 Y ORDENAR
        MOV CMPH1,CVTH1
        MOV CMPL1,CVTL1
        MOV CMPH2,CVTH2
        MOV CMPL2,CVTL2
        ACALL ORDER
        MOV CVTH1,CMPH1
        MOV CVTL1,CMPL1
        MOV CVTH2,CMPH2
        MOV CVTL2,CMPL2
                                ;COMPARAR 2 CONTRA 3 Y ORDENAR
        MOV CMPH1,CVTH2
        MOV CMPL1,CVTL2
        MOV CMPH2,CVTH3
        MOV CMPL2,CVTL3
        ACALL ORDER
        MOV CVTH2,CMPH1
        MOV CVTL2,CMPL1
        MOV CVTH3,CMPH2
        MOV CVTL3,CMPL2
                                ;COMPARAR 1 CONTRA 2 Y ORDENAR
        MOV CMPH1,CVTH1
        MOV CMPL1,CVTL1
        MOV CMPH2,CVTH2
        MOV CMPL2,CVTL2
        ACALL ORDER
        MOV CVTH1,CMPH1
        MOV CVTL1,CMPL1
        MOV CVTH2,CMPH2
```

```
        MOV CVTL2,CMPL2

        MOV CVTL,CVTL2
        MOV CVTH,CVTH2

        MOV R3,#4
ROTRIGHT:                       ;ELIMINAR BITS MENOS SIGNIFICATIVOS
        CLR C
        MOV A,CVTH
        RRC A
        MOV CVTH,A
        MOV A,CVTL
        RRC A
        MOV CVTL,A
        DJNZ R3,ROTRIGHT

        RET

ADCCONV:                        ;RUTINA DE CONVERSION DEL ADC
       MOV C,ACC.0
       MOV DIRA,C               ;CONECTAR CANAL AL ADC
     . MOV C,ACC.1
       MOV DIRB,C
       MOV C,ACC.2
       MOV DIRC,C
        ACALL WAIT2
       CLR CCS                  ;INICIAR CONVERSION
       CLR CWRITE
       CLR EA
       SETB CWRITE
        JB CEOC,$
       SETB EA
        JNB CEOC,$              ;ESPERAR FIN DE CONVERSION
       CLR CWRITE
       CLR CCS
       CLR CREAD                ;LEER DATOS DEL ADC
       SETB OE1
       CLR OE0
       acall waiter
       MOV A,ADC
        ACALL INVERT
       MOV CVTL,A
       SETB OE0
       CLR OE1
       acall waiter
       MOV A,ADC
       SETB CREAD
       SETB CCS
        ACALL INVERT
        RR A
        RR A
        RR A
        ANL A,#01Fh
        SJMP CONV_OK            ;ELIMINAR LECTURAS NEGATIVAS
        JNB ACC.4,CONV_OK
        MOV CVTL,#0
        CLR A
CONV_Ok:
       MOV CVTH,A
       SETB CAZ
       RET

INVERT:                         ;INVERTIR BITS DEL ACUMULADOR
       PUSH 03
       PUSH 04
       PUSH 05
       MOV R3,#8
CAMBIOS:
```

```
        RLC A
        MOV R4,A
        MOV A,R5
        RRC A
        MOV R5,A
        MOV A,R4
        DJNZ R3,CAMBIOS
        MOV A,R5
        POP 05
        POP 04
        POP 03
        RET

CALIBRAR:                       ;CALIBRAR ADC
        SETB CCS
        SETB CREAD
        SETB CWRITE
        CLR EA
        CLR CCAL
         JB CEOC,$
        SETB EA
      . SETB CCAL
         JNB CEOC,$
        RET

CVT_CHANNEL:            ;AJUSTA EL NUMERO DE CANAL AL LAYOUT DE LA TARJETA
        MOV DPTR,#BASE_TABLA
        MOVC A,@A+DPTR
        RET
BASE_TABLA:
         DB 6
         DB 5
         DB 4
         DB 7

         DB 0
         DB 3
         DB 1
         DB 2

ORDER:                  ;ORDENA DOS DATOS DE 16 BITS
        MOV A,CMPH1
        CJNE A,CMPH2,COMPARE1
        SJMP NOCAMBIES1
COMPARE1:
        JC NOCAMBIES2
        ACALL SWAPCOMP
        RET
NOCAMBIES1:
        MOV A,CMPL1
        CJNE A,CMPL2,COMPARE2
        SJMP NOCAMBIES2
COMPARE2:
        JC NOCAMBIES2
        ACALL SWAPCOMP
NOCAMBIES2:
        RET

SWAPCOMP:
        PUSH CMPH1
        PUSH CMPL1
        PUSH CMPH2
        PUSH CMPL2
        POP CMPL1
        POP CMPH1
        POP CMPL2
        POP CMPH2
        RET
```

```
WAITER:
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        RET

WAIT2:
        PUSH 07
        MOV R7,#-200
WAIT3:
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        DJNZ R7,WAIT3
        POP 07
        RET

INSERT:                         ;INSERTAR EN ARREGLO UN CANAL PARA
        PUSH 00                         ;SACAR PROMEDIO DE CUATRO MUESTRAS
        PUSH 01
        PUSH 02
        MOV A,CHAN
        RL A
        RL A
        ADD A,#DATA_STORE+3
        MOV R1,A
        MOV A,CVTL
        MOV R2,#4
CYCINSERT:                      ;ROTAR MUESTRAS ANTERIORES
        XCH A,@R1
        DEC R1
        DJNZ R2,CYCINSERT
        INC R1
        MOV R2,#4
        CLR A
        MOV SUML,A
        MOV SUMH,A
CYCINSERT1:
        MOV A,SUML
        ADD A,@R1
        MOV SUML,A
        MOV A,SUMH
        ADDC A,#0
        MOV SUMH,A
        INC R1
        DJNZ R2,CYCINSERT1

        CLR C                   ;EFECTUAR SUMA Y PROMEDIO
        MOV A,SUMH
        RRC A
        MOV SUMH,A
        MOV A,SUML
        RRC A
        MOV SUML,A
        CLR C
        MOV A,SUMH
        RRC A
```

```
        MOV SUMH,A
        MOV A,SUML
        RRC A
        MOV SUML,A
        MOV CVTL,SUML
        MOV CVTH,SUMH
        POP 02
        POP 01
        POP 00
        RET

ZFINAL:
END
```

ANEXO D. CÓDIGOS EN C

D.1 Módulo principal (main.cpp)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

#define NDEBUG
#define EDITING
// Programa principal para control de maquinaria
// Basado en nueva clase ABCPLC manejada por puerto paralelo unidireccional

#include "headers.h"
#include "abcprint.h"

#define ASCII

#define INPUT 1
#define OUTPUT 0

#ifndef false
        #define false 0
        #define true 1
#endif

#define MAXOPCION 7
#define COMMANDS 5

//*********************
int opcion=2;
struct time hora;
char update;
unsigned char commands[COMMANDS]={0,0,0,0,0};
unsigned char molde[21];
//*********************

int posx=11;
int posy=16;
int run=1;
int desplegar=1;

#include "register.h"
#include "timers2.h"
#include "counter.h"
#include "banner.h"
#include "abcprint.h"
#include "abcplc.h"
#include "inits.h"
#include "keyboard.h"
#include "timdecl.h"
#include "diskmgr.h"

//***********************************************
// Control de teclado
//***********************************************

#include "keydefs.h"

int keyhit(Keystroke &ky, int deftable=2)
{
static unsigned int prevscan;
static unsigned long keyrepeats=0;
```

```
static int table=2;
unsigned int ascii=0, scan=0, i;
for(i=0;i<COMMANDS;i++)
        commands[i]=0;
scan=inportb(0x60);             // Escaneo a bajo nivel del teclado
gotoxy(1,1);
printf("Scan: %02x",scan);
if(scan==0x2a)                  // Activar teclado de usuario
        table=deftable;
if(scan==0x36 && table==deftable)     // Activar teclado de PC
        {
        table=0;
        keyrepeats=0;
        }
if(table!=deftable)             // Verificar tiempo transcurrido de teclado de PC
        {
        keyrepeats++;
        if(keyrepeats==1000)
                table=deftable;
        }
gotoxy(60,1);
printf("R:%06ld",keyrepeats);
gotoxy(70,1);
printf("Teclado:%03d",table);
gotoxy(12,1);
scan=translatekey(table,scan);        // Mapeo de codigos de teclado
printf("  %02x",scan);
if(scan!=0)
        keyrepeats=0;
while(kbhit())
        {
        ScreenSaveReset();
        getch();
        }
if(!(scan&0x80))                // Caso 1. Tecla presionada
        {
        switch(scan&0x7f)
                {
                case KeyMClose:                 // Identificacion de tecla presionada
                        commands[0]=0x1;
                        break;
                case KeyMOpen:
                        commands[0]=0x2;
                        break;
                case KeyKOIn:
        .
        .
        .
                case KeySemi:
                        commands[4]=0x2;
                        break;
                case KeyAuto:
                        commands[4]=0x4;
                        break;
                case KeyMotor:
                        commands[4]=0x8;
                        break;
                }
        if(prevscan!=scan)              // Caso 2. El codigo de escaneo cambia
                {
                prevscan=scan;
                scan=scan&0x7f;
                switch(scan)
                        {
                        case KeyF0:     // Teclas de movimiento y cambio de pantalla
                        case KeyF1:
                        case KeyF2:
                        case KeyF3:
```

```
                        case KeyF4:
                        case KeyF5:
                        case KeyF6:
                        case KeyF7:
                        case KeyF8:
                                ascii=scan;
                                break;
                        case KeyEnter:
                                ascii='\n';
                                break;
                        case KeyCancel: //0x1:
                                ascii=27;
                                break;
                        case KeyPlus:
                                ascii='+';
                                break;
                        case KeyMinus:
        ...
        ...
        ...
                        case Key8:
                                ascii='8';
                                break;
                        case Key9:
                                ascii='9';
                                break;
#ifdef EDITING
                        case 0x56:
//                      case 0x58:
                                ascii='x';      // Teclas de salida del programa
                                break;
#endif
                        }
                ky=Keystroke(scan,ascii);       // Generar codigo ASCII y de escaneo
                return 1;
                }
        else
                {
                return 0;
                }
        }

if(ascii==0)
        prevscan=scan;
return 0;
}

void CopyButtonInputs(unsigned char b1, unsigned char b2, unsigned char b3, unsigned char
b4, unsigned char b5)
{
int i;
for(i=0;i<8;i++)
        Banner[i+150].SetState((b1&(0x1<<i%8))?1:0);
for(i=8;i<16;i++)
        Banner[i+150].SetState((b2&(0x1<<i%8))?1:0);
for(i=16;i<24;i++)
        Banner[i+150].SetState((b3&(0x1<<i%8))?1:0);
for(i=24;i<32;i++)
        Banner[i+150].SetState((b4&(0x1<<i%8))?1:0);
for(i=32;i<40;i++)
        Banner[i+150].SetState((b5&(0x1<<i%8))?1:0);
}

main(int argc, char *argv[])
{
Keystroke ky;           // Define una instancia de la clase Keystroke para almacenar cuando
el usuario
                        // presiona alguna tecla
```

```
int i;
char ch;
randomize();
clrscr();
_setcursortype(_NOCURSOR);
unsigned char contador;
char filestring[9]="PLC";                    // Definicion de nombres de timers, contadores y
señales
char exefilestring[20], cfgfilestring[20];
char deffilestring[20], timerdefstring[20];
char bannerdefstring[20], counterdefstring[20];
char registerdefstring[20];
if(argc==2)
        {
        strcpy(filestring,argv[1]);
        }

strcpy(exefilestring,filestring);      // Generar nombres de archivos con extension
strcpy(cfgfilestring,filestring);
PLC.LoadProgram(exefilestring);
strcat(cfgfilestring,".cfg");
PLC.Configure(cfgfilestring);
strcpy(timerdefstring,filestring);
strcpy(registerdefstring,filestring);
strcpy(bannerdefstring,filestring);
strcpy(counterdefstring,filestring);
strcpy(deffilestring,filestring);
strcat(timerdefstring,".tdf");
strcat(registerdefstring,".rdf");
strcat(bannerdefstring,".bdf");
strcat(counterdefstring,".cdf");
strcat(deffilestring,".def");
LoadTimerFunc(timerdefstring);                  // Cargar las funciones de los archivos de
configuración
LoadRegisterFunc(registerdefstring);
LoadBannerFunc(bannerdefstring);
LoadCounterFunc(counterdefstring);
PLC.LoadSignalFunc(deffilestring);

int key;     //Guarda el valor de la tecla presionada
update=1;    //Determina si es necesario actualizar los datos retentivos a disco
Update();    //Actualiza los valores retentivos de timers y opciones (carga PLC.ROM)

ClearTim();
ClearBan();

randomize();
screensave=0;
while(keyhit(ky))
        ch=ky.GetChar();        // Llamar rutina de escaneo de teclado
do
        {
        while(!keyhit(ky))      // Procesar mientras no se presionen teclas
                {

                CopyButtonInputs(commands[0],    commands[1],    commands[2],    commands[3],
commands[4]);
                                // Mapeo de teclado a la memoria interna del PLC
                if(run)
                        PLC.Scan();     // LECTURA DE ENTRADAS
                PLC.Process();          // PROCESAMIENTO DEL PROGRAMA
                if(run)
                        PLC.Write();    // ESCRITURA DE SALIDAS
                if(alloworint>=ALLOWPRINT)     // Codigo que se ejecuta cada N veces
                        {
                        static int rewrite=0;
                        allowprint=0;
                        rewrite++;
```

```
                        if(rewrite>80)          // Despues de 80 ciclos
                                {
                                ScreenSaveProcess();   // Activar protector de pantalla
                                if(update || PLC.record)
                                        {
                                        PLC.record=0;
                                        update=0;
                                        Backup();       // Respaldar memoria del PLC
                                        }
                                rewrite=0;
                                }
                        PrintDiags();           // Actualizar pantalla
                        }
                }
        ch=ky.GetChar();                // Interpretar tecla presionada
        ScreenSaveReset();              // Desactivar protector de pantalla
        PrintDiags(ch);                         // Actualizar pantalla y procesar tecla
        }
while(ch!='x');
_setcursortype(_NORMALCURSOR);
return(1);
}
```

D.2 Sección de código del módulo ABCPRINT

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>

#include "abcprint.h"
#include "abcplc.h"
#include "tipdef.h"
#include "banner.h"
#include "register.h"
#include "timers2.h"
#include "timdecl.h"
#include "counter.h"
#include "alarm.h"
#include "textedit.h"
#include "inter.h"
#include "seltext.h"
#include "diskmgr.h"

#include "string.h"

CFieldSet Campos;              // definicion de instancia de controlador de campos en
pantalla
float temperatura[4][TEMPPROM];

unsigned char screensave=0, screensave_on=0;
unsigned char screensavecount=0;

#include "abcprin3.h"
#include "keydefs.h"

void ScreenSaveProcess()       // Controlador de protector de pantalla
{
screensavecount++;
if(screensavecount>=MAXSCREENSAVE)
        screensave_on=1;
else
        screensave_on=0;
screensave=1;
}

void ScreenSaveReset()         // Desactivar protector de pantalla
{
screensavecount=0;
screensave_on=0;
}

void ScreenClear()
{
editstop();
textbackground(BLACK);
clrscr();
}

void PrintDiags(char key)      // Controlador de arbol de pantallas
{
static int pantalla=0, p_ant=0xff;
pantalla*=10;
switch(key)
        {

//*****************************************************************
        case 'j':
                pantalla/=10;
                break;
```

```
        case 'g':
                pantalla/=10;
                break;
 //*****************************************************************
        case KeyF1:             // La pantalla actual se multiplica por diez, y se le
                                // suma el numero de la tecla de funcion presionada
                pantalla+=1;
                break;
        case KeyF2:
                pantalla+=2;
                break;
        case KeyF3:
                pantalla+=3;
                break;
        case KeyF4:
                pantalla+=4;
                break;
        case KeyF5:
                pantalla+=5;
                break;
        case KeyF6:
                pantalla+=6;
                break;
        case KeyF7:
                pantalla+=7;
                break;
        case KeyF0:
                pantalla/=100;
                break;
        default:
                pantalla/=10;  // Si no se presiono tecla alguna, se divide entre diez
                               // y el numero resultante es la misma pantalla que esta
                               // desplegada
                break;
        }
if(screensave_on)
       pantalla=-1;
switch(pantalla)
        {
        case -1:                        // Imprimir protector de pantalla
                if(screensave)
                        {
                        screensave=0;
                        PrintScreenSaver(KeyNull);
                        }
                else
                        PrintScreenSaver(0);
                break;
        case 0:                         // Pantalla Principal
                if(pantalla!=p_ant)
                        {
                        ScreenClear();
                        PrintMain(1);
                        }
                PrintMain(key);
                break;
        case 1:
                if(pantalla!=p_ant)
                        {
                        ScreenClear();
                        PrintAbrirmolde(1);
                        }
                PrintAbrirmolde(key);
                break;
       case 11:
       case 115:
       case 116:
                if(pantalla!=p_ant)
```

```
                {
                ScreenClear();
                MoldSelect(1);
                }
        MoldSelect(key);
        pantalla=11;
        break;
  case 2:
        if(pantalla!=p_ant)
                {
                ScreenClear();
                PrintInyeccion(1);
                }
        PrintInyeccion(key);
        break;
  case 3:
        if(pantalla!=p_ant)
                {
                ScreenClear();
                PrintExpulsor(1);
                }
        PrintExpulsor(key);
        break;
  case 4:
        if(pantalla!=p_ant)
                {
                ScreenClear();
                PrintTemp(1);
                }
        PrintTemp(key);
        break;
  case 41:
        Inter.RlySet(143,1);
        pantalla=p_ant;
        break;
  case 42:
        Inter.RlySet(143,0);
        pantalla=p_ant;
        break;
  case 5:
        if(pantalla!=p_ant)
                {
                ScreenClear();
                PrintCarga(1);
                }
        PrintCarga(key);
        break;

  case 6:
        if(pantalla!=p_ant)
                {
                ScreenClear();
                PrintMantenimiento(1);
                }
        PrintMantenimiento(key);
        break;
 case 61:
        if(pantalla!=p_ant)
                {
                ScreenClear();
                PrintInputs(1);
                }
        PrintInputs(key);
        break;
 case 62:
        if(pantalla!=p_ant)
                {
                ScreenClear();
```

```
                        PrintOutputs(1);
                        }
                PrintOutputs(key);
                break;
        case 63:
                if(pantalla!=p_ant)
                        {
                        ScreenClear();
                        PrintBanners(1);
                        }
                PrintBanners(key);
                break;
        case 64:
                if(pantalla!=p_ant)
                        {
                        ScreenClear();
                        PrintRegs(1);
                        }
                PrintRegs(key);
                break;
        case 65:
                if(pantalla!=p_ant)
                        {
                        ScreenClear();
                        PrintTimers(1);
                        }
                PrintTimers(key);
                break;
        case 66:
                if(pantalla!=p_ant)
                        {
                        ScreenClear();
                        PrintCounters(1);
                        }
                PrintCounters(key);
                break;
        case 67:
                if(pantalla!=p_ant)
                        {
                        ScreenClear();
                        PrintSummary(1);
                        }
                PrintSummary(key);
                break;
        case 7:
                if(pantalla!=p_ant)
                        {
                        ScreenClear();
                        PrintVarios(1);
                        }
                PrintVarios(key);
                break;
        case 71:
                Inter.CntReset(0);
                Inter.CntReset(1);
                pantalla=p_ant;
                break;
        default:
                pantalla=p_ant;
                break;
        }
p_ant=pantalla;
}

#pragma warn -par

void PrintMantenimiento(char key)
{
```

```
gotoxy(10,10);
cprintf("Pantalla de Mantenimiento");
gotoxy(3,12);
cprintf("ADVERTENCIA: La modificacion de registros en estas pantallas puede");
gotoxy(3,13);
cprintf("producir comportamientos inesperados en la m quina. Utilizar solo");
gotoxy(3,14);
cprintf("para fines de mantenimiento, no operativos. Para cambiar par metros");
gotoxy(3,15);
cprintf("de operacion, refierase a la pantalla correspondiente.");
textcolor(YELLOW);
textbackground(BLUE);
gotoxy(CF1,24);
cprintf("1 Entradas");
gotoxy(CF2+3,24);
cprintf("2 Salidas");
gotoxy(CF3+5,24);
cprintf("3 Relays");
gotoxy(CF4+6,24);
cprintf("4 Registros");
gotoxy(CF6+1,24);
cprintf("5 Timers");
gotoxy(CF6+11,24);
cprintf("6 Contadores");
}

...
...
...

void PrintInputs(char key)              // Rutina de monitoreo de entradas en pantalla
{
int i,ncanal,t_busqueda;
unsigned char tip;
static t_actual=0, subpantalla=0;
unsigned int l,m;
unsigned char j,searchok;
volatile unsigned char y=6;
switch(key)                             // Control de tarjeta a desplegar
        {
        case 'j':
                subpantalla++;
                delayline+=10;
                break;
        case 'g':
                subpantalla--;
                delayline-=10;
                break;
        case 'y':
                t_actual++;
                subpantalla=0;
                if(t_actual==MAXCARDS-1)
                        t_actual=0;
                break;
        }
gotoxy(1,2);
textcolor(CYAN);
textbackground(BLACK);
cprintf("                               MONITOREO DE ENTRADAS                   ");
searchok=0;
tip=PLC.Card[t_actual]->GetType();      // Obtener el tipo de tarjeta que se esta desplegando
if(tip>0 && tip<INPUTLIMIT)             // Buscar la siguiente tarjeta de entradas en el rack
        searchok=1;
else
        t_busqueda=(t_actual==MAXCARDS-1)?0:t_actual+1;
while((tip>INPUTLIMIT || tip==0) && (!searchok))
        {
        tip=PLC.Card[t_busqueda]->GetType();
```

```
        if(tip>0 && tip<INPUTLIMIT)
                {
                searchok=1;
                t_actual=t_busqueda;
                }
        else
                {
                t_busqueda=(t_busqueda==MAXCARDS-1)?0:t_busqueda+1;
                if(t_busqueda==t_actual)
                        {
                        searchok=1;
                        tip=0;
                        }
                }
        }
switch(tip)                             // Imprimir la informacion correspondiente al tipo de
tarjeta seleccionada
        {
        case DIGINPUT24:                // Entradas digitales
                gotoxy(1,3);
                textcolor(WHITE);
                cprintf("Tarjeta: %2u Entradas Digitales ",t_actual);
                gotoxy(1,4);
                cprintf("                       No.             Valor       Real                Funcion
");
                if(subpantalla>1)
                        subpantalla=1;
                if(subpantalla<0)
                        subpantalla=0;
                ncanal=subpantalla*12;
                for(j=0;j<12;j++)
                        {
                        m=((CTIn24 *)PLC.Card[t_actual])->GetData(ncanal/8);
                        l=m&(0x1<<ncanal%8);
                        m=l?1:0;
                        gotoxy(6,y);
                        cprintf("%04u:    ",t_actual*100+ncanal);
                        cprintf("       %lu                                                    %s
",m,PLC.Card[t_actual]->GetFuncion(ncanal));
                        ncanal++;
                        y++;
                        }
                for(j=12;j<16;j++)
                        {
                        gotoxy(1,y++);
                        cprintf("
");
                        }
                break;
#ifdef SAVESPACE
        case ALLENIN8:                  // Entradas digitales ALLEN BRADLEY
                gotoxy(1,3);
                textcolor(WHITE);
                cprintf("Tarjeta: %2u Entradas Digitales AB",t_actual);
                gotoxy(1,4);
                cprintf("                       No.             Valor       Real                Funcion
");
                m=((CABIn8 *)PLC.Card[t_actual])->GetData();
                for(j=0;j<8;j++)
                        {
                        l=m&(0x1<<j);
                        l=l?1:0;
                        gotoxy(6,y);
                        cprintf("%04u:    ",t_actual*100+j);
                        cprintf("       %lu                                                    %s
",l,PLC.Card[t_actual]->GetFuncion(j));
                        y++;
                        }
```

```
                for(j=8;j<16;j++)
                        {
                        gotoxy(1,y++);
                        cprintf("
");
                        }
                break;
#endif
        case ANINPUT16:                         // Entradas analogicas
                gotoxy(1,65);
                cprintf("%0101u",delayline);
                gotoxy(1,3);
                textcolor(WHITE);
                cprintf("Tarjeta: %2u Entradas Analogicas",t_actual);
                gotoxy(1,4);
                cprintf("                       No.          Valor            %%              Funcion
");
                for(j=0;j<16;j++)
                        {
                        m=((CTAninput16 *)PLC.Card[t_actual])->GetData(j);
                        gotoxy(6,y);
                        cprintf("%04u:   ",t_actual*100+j);
                        cprintf(" %05d ",m);
                        m=PorcentajeADC(m);
//                      cprintf(" %04X      %s                          ",m,PLC.Card[t_actual]-
>GetFuncion(j));
                        cprintf("          %03u.%02u                                                %s
",m/100,m%100,PLC.Card[t_actual]->GetFuncion(j));
                        y++;
                        }
                break;
        case TPJINPUT8:                         // Entradas de termopar
                gotoxy(1,3);
                textcolor(WHITE);
                cprintf("Tarjeta: %2u Termopares tipo J  ",t_actual);
                gotoxy(1,4);
                cprintf("                       No.          Valor          Temp        Descripcion
");
                for(j=0;j<8;j++)
                        {
                        m=((CTTpjinput8 *)PLC.Card[t_actual])->GetData(j);
                        gotoxy(6,y);
                        cprintf("%04u:   ",t_actual*100+j);
                        cprintf(" %05u  ",m);
                        m=TempTpj(m);
                        cprintf(" %03u.%02u",m/100,m%100);
                        y++;
                        }
                for(j=8;j<16;j++)
                        {
                        gotoxy(1,y++);
                        cprintf("
");
                        }
                break;
        case 0:                         // No se encontraron tarjetas de entradas
                gotoxy(1,3);
                textcolor(WHITE);
                cprintf("No hay tarjetas de este tipo!   ",t_actual);
                gotoxy(1,4);
                cprintf("                       No.          Valor      Real              Funcion
");
                for(j=0;j<16;j++)
                        {
                        gotoxy(6,y);
                        cprintf("
");
                        }
```

```
                break;
        }
}


void PrintTimers(char key)              // Monitoreo de timers
{
const porpantalla=15, maxval=MAXTIMERS-1;
static char pantalla=0, refresh=1;
static int selected=0;
unsigned i,j,actual,y=6;
unsigned int l, newval;
static unsigned oldval=0;
editmode modo;
if(key==1)
        refresh=1;
if(refresh)
        {
        gotoxy(1,2);
        textcolor(CYAN);
        cprintf("                       AJUSTE Y MONITOREO DE TEMPORIZADORES      ");
        gotoxy(1,4);
        textcolor(WHITE);
        cprintf("                    No.          Maximo        Actual                Funcion
");
        }
for(i=0;i<porpantalla;i++)              // Impresion de una pantalla de timers
        {
        gotoxy(8,y+i);
        textcolor(WHITE);
        actual=pantalla*porpantalla+i;
        if(actual<=maxval)
                {
                cprintf("%2u: ",actual);
                if(selected==actual)
                        textbackground(LIGHTCYAN);
                else
                        textbackground(BLACK);
                textcolor(WHITE);
                j=Timer[actual].GetLoad();
                cprintf("   %03u.%02u   ",j/100,j%100);
                if(Timer[actual].On())
                        textcolor(WHITE);
                else
                        {
                        if(Timer[actual].IsSet())
                                textcolor(RED);
                        else
                                textcolor(BLUE);
                        }
                j=Timer[actual].GetValue();
                if(selected==actual)
                        textbackground(LIGHTCYAN);
                else
                        textbackground(BLACK);
                cprintf("%03u.%02u   ",j/100,j%100);
                textbackground(BLACK);
                textcolor(WHITE);
                if(refresh)
                        {
                        gotoxy(34,y+i);
                        cprintf("                                                    ");
                        gotoxy(34,y+i);
                        cprintf("%s",Inter.GetTimFunc(actual));
                        }
                }
        else
```

```
                cprintf("
");
        }
refresh=1;
switch(editfield(key, newval, oldval, 64, 1, DIVIDIDO, 65535))          //   Control   de
edicion de campos
        {
        case SUCCESS:
                Timer[selected].Load(newval);
                PLC.record=1;
                break;
        case EDITING:
                break;
        default:
                switch(key)             // Control de pantalla a desplegar
                        {
                        case 'j':
                                pantalla++;
                                if pantalla>=maxval/porpantalla+1)
                                        pantalla=0;
                                selected=pantalla*porpantalla;
                                break;
                        case 'g':
                                pantalla--;
                                if pantalla<0)
                                        pantalla=maxval/porpantalla;
                                selected=pantalla*porpantalla;
                                break;
                        case 'n':
                                selected++;
                                if selected>maxval)
                                        {
                                        selected=0;
                                        pantalla=0;
                                        }
                                if selected>=(pantalla+1)*porpantalla)
                                        pantalla++;
                                break;
                        case 'y':
                                selected--;
                                if selected<0)
                                        {
                                        selected=maxval;
                                        pantalla=maxval/porpantalla;
                                        }
                                if selected<(pantalla*porpantalla))
                                        pantalla--;
                                break;
                        default:
                                refresh=0;
                                break;
                        }
                oldval=Timer[selected].GetLoad();
                break;
        }
}


void PrintScreenSaver(char key)                 // Impresion del protector de pantalla
{
unsigned long i;
static COLORS j;
static x=1, y=1;
        {
        if(key==KeyNull)
                {
                clrscr();
                x=random(59)+1;
```

```
            y=random(17)+1;
            j=(COLORS)(random(4)+1);
            gotoxy(x,y);
            cprintf("ÛÛ ABC Electronics ÛÛ");
            gotoxy(x,y+1);
            cprintf("Û                    Û");
            gotoxy(x,y+2);
            cprintf("Û                    Û");
            gotoxy(x,y+3);
            cprintf("Û                    Û");
            gotoxy(x,y+4);
            cprintf("Û                    Û");
            gotoxy(x,y+5);
            cprintf("ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ");
            }
        textcolor(j);
        textbackground(BLACK);
        gotoxy(x+2,y+1);
        i=Inter.GetCntVal(0)+Inter.GetCntVal(1)*10000;
        cprintf("Ciclos:  %08lu",i);
        gotoxy(x+2,y+2);
        i=Inter.GetReg(102);
        cprintf("Inyeccion: %03i",i/100);
        cprintf(".%02i",i%100);
        gotoxy(x+2,y+3);
        i=Inter.GetReg(103);
        cprintf("Ciclo:     %03i",i/100);
        cprintf(".%02i",i%100);
        gotoxy(x+2,y+4);
        i=Inter.GetReg(104);
        cprintf("Curado:    %03i",i/100);
        cprintf(".%02i",i%100);
        }
}
```

D.3 Seccion de codigo del módulo ABCPRIN3

```
#include <conio.h>
#include <stdio.h>
#include "seltext.h"
#include "inter.h"
#include "abcprint.h"
#include "abcprin3.h"
#include "keydefs.h"
#include "timerr.h"


void PrintLogo(int x, int y, unsigned valor, COLORS color);

static CFieldSet Campos;

void PrintVarios(char key)
{
int x=3;
unsigned long i;
#ifndef SAVESPACE
if(key==0x1)
        {
        textcolor(CYAN);
        gotoxy(x+2,3);
        cprintf("AJUSTE DE LUBRICACION");
        gotoxy(x+2,7);
        cprintf("PRESIONES ESPECIALES");
        gotoxy(x+2,10);
        cprintf("CONTADOR DE PIEZAS");

        textcolor(BLUE);
        gotoxy(x,4);
        cprintf("CICLOS      :");
        gotoxy(x,5);
        cprintf("DURACION    :");
        gotoxy(x,8);
        cprintf("PRESION MINIMA DEL SISTEMA :");
        gotoxy(x,11);
        cprintf("CICLOS DE INYECCION:        ");

        //textcolor(YELLOW);
        textcolor(YELLOW);
        textbackground(BLUE);
        gotoxy(CF1,24);
        cprintf("1 Contador a ceros");

        Campos.Reset();
        Campos.Field[0].SetAll(23,4,CLOAD, 5, PNONE);
        Campos.Field[1].SetAll(23,5,TLOAD, 45, PNONE);
        Campos.Field[2].SetAll(40,8,RVAL, 101, PNONE);
        Campos.SetUD(0,1);
        Campos.SetUD(1,2);
        Campos.SetUD(2,0);
        }
else
#endif
        {
        Campos.Process(key);
        textcolor(WHITE);
        textbackground(BLACK);
        gotoxy(37,11);
        i=Inter.GetCntVal(0)+Inter.GetCntVal(1)*10000;
        cprintf("%08lu",i);
        }
}
```

```
void PrintCarga(char key)
{
#ifndef SAVESPACE
int x=3;
if(key==0x1)
        {
        gotoxy(33,2);
        textcolor(CYAN);
        cprintf("AJUSTE DE CARGA");
        gotoxy(2,4);
        textcolor(WHITE);
        cprintf("ETAPA                    1    DESCOMPRESION");

        textcolor(BLUE);
//      gotoxy(3,6);
//      printf("TIEMPO      :");
        gotoxy(x,6);
        cprintf("VELOCIDAD    :");
        gotoxy(x,7);
        cprintf("PRESION      :");
        gotoxy(x,8);
        cprintf("POSICION     :");
        gotoxy(x,9);
        cprintf("CONTRAPRESION:");
        gotoxy(x,12);
        cprintf("TIEMPO DE CURADO:");

        //textcolor(YELLOW);

        Campos.Reset();
        Campos.Field[0].SetAll(23,6,RVAL, 137, PNONE);
        Campos.Field[2].SetAll(23,7,RVAL, 99, PNONE);
        Campos.Field[3].SetAll(23,8,RVAL, 94, PNONE);
        Campos.Field[4].SetAll(37,8,RVAL, 95, PNONE);
        Campos.Field[5].SetAll(23,9,RVAL, 138, PNONE);
        Campos.Field[6].SetAll(25,12,TLOAD, 39, PNONE);
        Campos.SetUD(0,2);
        Campos.SetUD(2,3);
        Campos.SetUD(3,5);
        Campos.SetUD(5,6);
        Campos.SetLR(3,4);
        Campos.SetLR(2,4);
        Campos.SetLR(0,4);
        }
else
#endif
        Campos.Process(key);
}

...
...
...
```

D.4 Sección de código del módulo SELTEXT

```
// Archivo de campo seleccionable

#include "seltext.h"
#include "inter.h"
#include "conio.h"
#include "textedit.h"

...
...
...

void CIntText::Print(int selected)
{
unsigned data;
float dat;
gotoxy(x,y);
textcolor(YELLOW);
if(selected)
        textbackground(LIGHTCYAN);
else
        textbackground(BLACK);
switch(tipo)
        {
        case TVAL:
                data=Inter.GetTimVal(numdato);
                break;
        case TLOAD:
                data=Inter.GetTimLoad(numdato);
                break;
        case CVAL:
                data=Inter.GetCntVal(numdato);
                break;
        case CLOAD:
                data=Inter.GetCntLoad(numdato);
                break;
        case RVAL:
                data=Inter.GetReg(numdato);
                break;
        }
switch(tipo)
        {
        case TVAL:
        case TLOAD:
                cprintf("%03u.%02u",data/100,data%100);
                break;
        case CVAL:
        case CLOAD:
                cprintf("%05u",data);
                break;
        case RVAL:
                switch(prcent)
                        {
                        case PINT:
                                dat=((float)data)/65535.0*100.0;
                                cprintf(" %3u ",(unsigned)dat);
                                break;
                        case PCHAR:
                                dat=((float)data)/255.0*100.0;
                                cprintf(" %3u ",(unsigned)dat);
                                break;
                        case PNONE:
                                cprintf("%05u",data);
                                break;
                        case PCOMP:
                                dat=((float)data)*KConv;
                                cprintf("%05u",(unsigned)dat);
```

```
                        break;
/*                  case PPERCNT:
                        dat=((float)data)/655.35*KConv;
                        cprintf("%3.2f%%",dat);
                        break;
                    case PPERCHAR:
                        dat=((float)data)/2.55*KConv;
                        cprintf("%3.2f%%",dat);
                        break;
*/
                    case PBIN:
                        cprintf("%5u",data?1:0);
                        break;
                    case PTIME:
                        cprintf("%02u:",data/100);
                        cprintf("%02u",data%100);
                        break;
                    }
            break;
        case NONE:
            break;
      }
textbackground(BLACK);
}

...
...
...


void CFieldSet::Process(char key)
{
int i;
static unsigned newval, oldval, maxval=65535;
static editline edmode=NORMAL;
float value;
switch(editfield(key, newval, oldval, 64, 1, edmode, maxval))
        {
        case SUCCESS:
            if(Field[selected].KConv!=0)
                value=newval/Field[selected].KConv;
            else
                value=0;
            value=round(value);
            switch(Field[selected].GetType())
                {
                case TLOAD:

        Inter.SetTimLoad(Field[selected].GetNum(),value);//(double)newval/Field[selected].KC
onv);
                    break;
                case CLOAD:

        Inter.SetCntLoad(Field[selected].GetNum(),value);//(double)newval/Field[selected].KC
onv);
                    break;
                case RVAL:

        Inter.SetReg(Field[selected].GetNum(),value);//((double)newval)/Field[selected].KCon
v);
                    break;
                }
            break;
        case EDITING:
            break;
        default:
            switch(key)
                {
```

```
                        case 'y':
                                if(Field[selected].GetType()!=NONE)
                                        selected=Field[selected].GetUp();
                                break;
                        case 'n':
                                if(Field[selected].GetType()!=NONE)
                                        selected=Field[selected].GetDown();
                                break;
                        case 'g':
                                if(Field[selected].GetType()!=NONE)
                                        selected=Field[selected].GetLeft();
                                break;
                        case 'j':
                                if(Field[selected].GetType()!=NONE)
                                        selected=Field[selected].GetRight();
                                break;
                        }
                switch(Field[selected].GetType())
                        {
                        case TLOAD:

        oldval=Inter.GetTimLoad(Field[selected].GetNum())*Field[selected].KConv;
                                edmode=DIVIDIDO;
                                break;
                        case CLOAD:

        oldval=Inter.GetCntLoad(Field[selected].GetNum())*Field[selected].KConv;
                                edmode=NORMAL;
                                break;
                        case RVAL:

        oldval=Inter.GetReg(Field[selected].GetNum())*Field[selected].KConv;
                                switch(Field[selected].GetPercent())
                                        {
                                        case PBIN:
                                                maxval=1;
                                                edmode=NORMAL;
                                                break;
                                        case PTIME:
                                                edmode=HORA;
/*                                              maxval=2359;
                                        case PPERINT:
                                                edmode=DIVIDIDO;
                                                maxval=10000;

        oldval=Inter.GetReg(Field[selected].GetNum())*Field[selected].KConv/65535;
                                        case PPERCHAR:
                                                edmode=DIVIDIDO;
                                                maxval=10000;

        oldval=Inter.GetReg(Field[selected].GetNum())*Field[selected].KConv/255;
*/
                                        default:
                                                maxval=65535;
                                                edmode=NORMAL;
                                                break;
                                        }
                        break;
                        }
                break;
        }
for(i=0;i<FIELDMAX;i++)
        Field[i].Print(i==selected?1:0);
}
```