UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

CAMPUS ARAGÓN

"IMPLEMENTACIÓN DE UN SISTEMA INTELIGENTE DIFUSO PARA EL CONTROL DE PROCESOS BASADOS EN CORRIENTE CONTINUA Y CORRIENTE ALTERNA, UTILIZANDO UN PIC"

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO MECÁNICO-ELECTRICISTA
ÁREA: ELÉCTRICA-ELECTRÓNICA

P R E S E N T A:

FERMI VÁZQUEZ VILLANUEVA

ASESOR:

DR. NICOLÁS KEMPER VALVERDE

SAN JUAN DE ARAGÓN ESTADO DE MÉXICO, ENERO DEL 2005.

Dedicatoria.

A mis padres: Margarita Villanueva Carmona y Agustín Vázquez Hinojosa.

A ellos, por darme la vida y porque, sin su amor y apoyo incondicional; comprensión, paciencia y confianza en mí, este trabajo nunca hubiera sido posible.

A mis abuelos: Soledad Hinojosa y Rafael Vázquez.

A ellos por su amor incondicional y por brindarme su apoyo siempre que lo necesite.

A mis hermanos: Citlalli, Crhistian, Marcelino y Rafael, por brindarme su amor, apoyo y compañía incondicionales.

Agradecimientos.

A la Universidad Nacional Autónoma de México con la que siempre quedare en deuda..... gracias por todo.

Al mi tutor de tesis, Dr. Nicolás Kemper Valverde por el apoyo y la libertad brindados para el desarrollo del presente trabajo.

A Walter y Henry Kemper, por su valioso apoyo y aportación en la realización del presente trabajo.

A mis sinodales: Ing. Juan Gastaldi Pérez, Ing. Adrián Paredes Romero, por el tiempo dedicado a la revisión del presente trabajo y en especial al Ing. Eleazar Margarito Pineda Díaz e Ing. Francisco Raúl Ortiz Gonzáles, por sus valiosas observaciones, que contribuyeron a mejorar sustancialmente la calidad del presente trabajo.

A todos mis amigos y amigas en general, a ellos por su amistad incondicional y por los momentos inolvidables que pasamos juntos, los cuales me hicieron más grata la vida.

A todos aquellos que por olvido, no hubiera mencionado, y que también hayan contribuido de alguna manera a este logro.

A la vida misma.

Resumen.

Este trabajo consta de 5 capítulos, en los dos primeros se documenta la información teórica previa tomada en cuenta para el diseño y realización del prototipo y en los tres últimos se documenta el proceso de diseño e implementación a nivel hardware y resultados obtenidos del mismo. Al final se incluye un apéndice que contiene el código fuente del programa del microcontrolador

En el capítulo 1 se presenta la información referente a las motivaciones del presente trabajo como una herramienta para el diseño de controladores difusos, que pudiera en un futuro, ayudar en el aprendizaje y aplicación del control difuso en la solución y automatización de procesos productivos reales.

En el capítulo 2 se presenta los fundamentos teóricos tomados en cuenta para el diseño del prototipo. Se incluyen los conceptos fundamentales de la lógica difusa, un resumen de sus aplicaciones, y en especial su aplicación en el diseño de sistemas de control por medio de los controladores difusos. Finalizando el capítulo con una breve exposición del propósito y disponibilidad de las herramientas para el diseño de controladores difusos y su implementación en distintas tecnologías, ubicando la implementación del prototipo propuesto dentro de ese marco.

En el capítulo 3 se presenta la conceptualización y diseño del sistema prototipo. El sistema inteligente difuso para el control de procesos basados en corriente continua y corriente alterna, utilizando un PIC, es una herramienta de integración hardware y software para el diseño de sistemas de control difusos, la parte del software la constituye la interface gráfica de usuario para la edición, diseño, generación de código y monitorización del sistema de control difuso; la parte hardware la constituye la implementación física del controlador difuso donde se descargara el código asociado al mismo, para ser ejecutado directamente en la aplicación particular para la que fue diseñado.

En el capítulo 4 se presente la implementación del prototipo propuesto dentro del Laboratorio de Sistemas Inteligentes del Centro de Ciencias Aplicadas y Desarrollo Tecnológico (CCADET) de la UNAM, para las pruebas y validación del mismo a través de su aplicación al control de temperatura de un horno eléctrico acondicionado en el mismo laboratorio. También se exponen las especificaciones técnicas alcanzadas por el prototipo y finalmente se muestran lo resultados obtenidos de pruebas realizadas.

Finalmente en el capítulo 5 se presentan las conclusiones finales del presente trabajo basadas en los objetivos y resultados alcanzados.

Índice Temático.

CAPÍTULO 1. INTRODUCCIÓN.

1.1 Antecedentes.

1.1.1 Necesidad del control.

Es indudable que nuestra vida moderna incluye grandes y complejas organizaciones y sofisticados dispositivos técnicos que necesitan ser controlados. Las plantas industriales modernas poseen sofisticados sistemas de control que son cruciales para su operación correcta. De hecho, ninguno de los sistemas modernos (aviones, trenes de alta velocidad, reproductores de CD, etc.) podrían operar sin la ayuda de sofisticados sistemas de control. Un mejor control es la clave tecnológica para lograr productos de mayor calidad, minimización de desperdicios, protección del medio ambiente, mayor rendimiento de la capacidad instalada, mayores márgenes de seguridad y también mayor comodidad.

1.1.2 Control convencional.

Durante más de 50 años se han sentado las bases teóricas del control automático. Como productos de estos esfuerzos se han desarrollado las teorías de control clásica y la teoría de control moderna (métodos convencionales de control). Se dice que con el uso de estas metodologías se llegado a alcanzar un 90% de control satisfactorio.

1.1.3 Restricciones del control convencional.

Aun así, existen condiciones bajo las cuales, las técnicas de control clásicas (o convencionales) no pueden ser aplicadas y puede ser muy ventajoso basar la estrategia de diseño en un enfoque diferente. Algunas de tales condiciones pueden ser:

- No se cuenta con un modelo matemático del proceso a ser controlado, o éste, sólo puede ser obtenido con gran esfuerzo y costo,
- Aunque se tiene un modelo matemático parcial del proceso a ser controlado, la influencia de la dinámica no modelada en la calidad de actuación del controlador es significante y no puede ser despreciada,
- Cualesquiera de los parámetros del proceso o del punto de operación cambian de manera imprevisible,
- Sólo una parte de la información del proceso se encuentra disponible en forma cuantitativa, mientras que el resto de la información, es asequible únicamente en forma cualitativa,
- los datos que se obtienen del proceso son incompletos y/o imprecisos.

Los puntos mencionados no son en manera algunas mutuamente excluyentes. Enfatizan diferentes aspectos de un tema común: control en ausencia del conocimiento completo respecto de la planta a ser controlada, respecto de su entorno, o de ambos.

1.1.4 Herramientas da la Inteligencia Artificial como alternativa.

En situaciones tales como éstas, las aproximaciones matemáticas puramente cuantitativas suelen no trabajar bien y pueden ser reemplazadas con éxito por métodos alternativos de la Inteligencia Artificial (IA), es decir, por métodos que trabajan utilizando como base una descripción del comportamiento del proceso compuesta por la mezcla de información cualitativa y de la información cuantitativa medida con que se cuenta.

1.1.5 La lógica difusa como una herramienta alternativa.

La lógica difusa definitivamente es una de estas técnicas emergentes. Ha sido calificada como una de los primeros productos útiles de la IA en el dominio del control y una de las esencias de la computación cognoscitiva. El avance llego a mediados de los 1960 con la introducción de la lógica difusa por Zadeh. La aplicación de la teoría de Zadeh al control vino casi diez año mas tarde y tomaron incluso mas años antes de que el recibiera justamente el respeto y aceptación que merece.

Los sistemas basados en lógica difusa, imitan la forma de tomar decisiones de los humanos, con la ventaje de ser mucho más rápidos. Esos sistemas son generalmente robustos y tolerantes a imprecisiones y ruido en los datos de entrada. Por lo tanto no están sujetos a las restricciones de los sistemas convencionales. El control difuso aparece hoy en aplicaciones de tiempo real al extenderse desde aplicaciones domesticas a trenes de alta velocidad, automóviles y control de procesos.

1.1.6 Herramientas de diseño e implementación de sistemas de control difusos.

Como resultado de la creciente demanda de los controladores difusos en un gran número de aplicaciones, se han desarrollado paquetes de diseño o herramientas (software) que son exitosamente usados en diferentes aplicaciones para el diseño de controladores difusos. En cualquier caso las metas de los paquetes de diseño son usualmente.

- Acelerar un proceso de diseño.
- Hacer el diseño simple, y directo.
- Proporcionar la posibilidad de aplicar lógica difusa y métodos de control sin un entendimiento profundo de cómo trabajan.

La realización de los pasos de diseño de un controlador difusos puede ser lograda en una relativamente corta cantidad de tiempo utilizando uno de estos paquetes de software especializado. El paquete o herramientas de diseño difuso, pueden también ayudar en la implementación del diseño. Una vez, que es hecho, la herramienta puede generar un programa (en 'C' o en ensamblador) que represente el algoritmo de control lógico difuso. Este programa puede entonces ser descargado a un sistema basado en microcontrolador para correr la aplicación.

Como el control difuso ha llegado a ser popular y un gran número de aplicaciones exitosas han sido desarrolladas, los desarrolladores hardware han llegado a proponer soluciones complejas, integrando software y hardware en el diseño de sistemas de control difuso. El objetivo es proporcionar la oportunidad de completar el diseño e implementar un controlador difuso para una aplicación particular sin la necesidad de software y hardware extra.

1.1.7 Disponibilidad de las herramientas de diseño e implementación de controladores difusos.

El conjunto de herramientas software y hardware y combinación de ambas (sistemas integrados) para el diseño e implementación de controladores difusos en general no son disponibles para todo los sectores industriales y académicos del país ya que son productos costosos, entre otras cosas, por que se tienen que importar de países desarrollados.

Muchas universidades nacionales no cuentan con los recursos para disponer de tales herramientas para el proceso de enseñanza de la lógica difusa, aunado a esto existen sectores productivos del país en especial, pequeñas empresas industriales y agrícolas donde el uso de esta tecnológica podría utilizarse para mejorar la calida de sus productos, reducir su consumo de energía, mejorar la efectividad productiva, lo que les daría un mejor posición competitiva en el inminente mercado global, aunque por falta de recursos no pueden acceder a ella.

1.1.8 EL presente trabajo como una herramienta alternativa para el diseño de sistemas de control difusos.

EL presente trabajo persigue, la implementación del prototipo de una herramienta alternativa para el diseño e implementación sistemas de control difuso en tiempo real (Sistema Inteligente Difuso para el control de procesos basados en Corriente Continua y Corriente Alterna utilizando un PIC), que pueda aplicarse al control de plantas y procesos reales de la industria u otros sectores productivos que lo requieran, aprovechando las bondades del control difuso para tratar con este tipo de aplicaciones, y que derivado del éxito del trabajo se continué perfeccionándose para que pudiera ser en un futuro considerada como un medio disponible, confiable, y efectivo para el diseño de sistemas de control difusos. Demostrando con esto que también aquí en el país es posible desarrollar este tipo de herramientas, con la consecuente reducción de la dependencia tecnológica de países desarrollados.

1.2 Justificación.

En los tiempos actuales en nuestro país se ha observado una necesidad real de mejora de los procesos de producción tanto en la industria como en el sector agropecuario donde es necesario sustituir muchos operaciones manuales por sistemas automáticos con el fin, no de desplazar a la mano de obra humana si no mas bien librarle de tareas monótonas, repetitivas y muchas veces riesgosas, la cuales ponen en peligro su integridad física; pero no solo eso, si no, también, fomentar el ahorro de energía y mejorar la calidad de los productos, para darles una mejor posición estrategia en el mercado actual.

Por lo tanto es imprescindible que se dedique tiempo (investigación) y dinero (inversión) para desarrollar herramientas propias que pudieran ser aplicadas a la solución de los problemas atrás mencionados, y además que pudieran estar disponibles en las universidades para que los futuros técnicos y profesionistas en este campo del conocimiento las pudieran utilizar en la solución de los mismos, ayudando con esto a reducir la brecha entre la universidad y los sectores productivos.

El prototipo de implementación del sistema inteligente difusos para el control de procesos basados en Corriente Continua (CC) y Corriente Alterna (CA), utilizando un PIC (SID), busca ser un iniciativa en este rubro, de tal manera que el éxito de dicho prototipo pudiera incentivar a seguir a mas profesionales y autoridades gubernamentales y académicas a invertir tiempo y dinero en el desarrollo de dicho tipos de sistemas.

1.3 Planteamiento del problema y propuesta de solución

Basados en los antecedentes y en la justificación del trabajo, el control lógico difuso es una opción factible y efectiva para encarar las necesidades de control de los sistemas productivos actuales por lo tanto se hace indispensable que no solo se proponga la implementación de una herramienta para el control de los mismos si no que la estrategia de control este basado en técnicas que permitan encarar con éxito tales necesidades, la lógica difusa es una de ellas.

El sistema inteligente difuso para el control de procesos basados en CC y CA, utilizando un PIC se propone como una herramienta de integración software y hardware para el diseño de un sistema de control difuso de procesos basados en CC y CA, que proporcione la oportunidad de completar el diseño e implementar un controlador difuso para una aplicación particular sin la necesidad de software y hardware extra, siempre y cuando el controlador resultante cumpla con los requerimientos de la aplicación particular. El sistema estará integrado por dos componentes fundamentales (ver Figura 1.1):

a) Un parte hardware que se llamara "Tarjeta Electrónica de Procesamiento Difuso y de Entrada y Salida" (TEPDES), en donde se albergara: el microcontrolador que gestionara la operación global del sistema, y además se encargara de llevar acabo el procesamiento difuso, la adquisición y entrega de datos, y la comunicación RS – 232 con una PC; la fuente de alimentación de la tarjeta; los módulos hardware para el acondicionamiento de las entradas analógicas y la salida de control en CC o CA.

b) Una parte software que se llamara "Interface de Programación y Visualización del Sistema Inteligente Difuso" (IPVSID), que será una interface grafica donde se llevara acabo el diseño de un controlador difuso, que incluirá la edición de las entradas y la salida del controlador, edición de las funciones de membresía de las entradas y de las funciones singleton de la salida, la edición del conjunto de reglas difusas y finalmente la generación del código asociado a dicho diseño para que posteriormente sea descargado a la TEPDES par ser ejecutado. Además en dicha interface

durante la ejecución del controlador difuso diseñado se desplegara una ventana para la monitorización y visualización del estado de las entradas y la salida del controlador, así como de la salida del proceso controlado con la posibilidad poder cambiar desde dicha ventana la configuración de un conjunto de parámetros del sistema de control difuso.

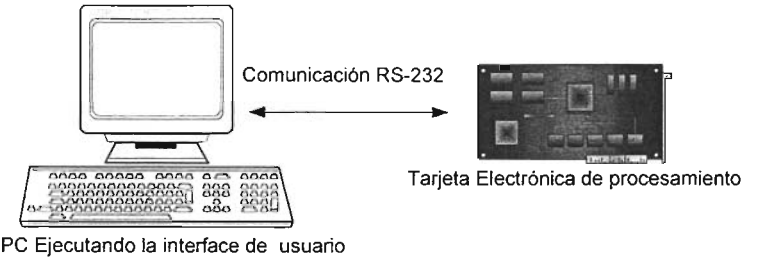


Figura 1. 1 Elementos fundamentales del SID.

1.4 Objetivos.

El objetivo de este trabajo se dividió en objetivo principal y objetivos particulares

Objetivo principal.

El objetivo principal es Implementar el prototipo del "Sistema Inteligente Difuso para el control de procesos basados en CC y CA utilizando un PIC" (SID).

Objetivo particulares.

El Sistema Inteligente Difuso estará compuesto de dos componentes por lo tanto dos objetivos particulares son.

a) Implementar el prototipo de la Tarjeta Electrónica de Procesamiento Difuso y de entrada y salida de datos (abreviada TEPDES), que corresponde a la parte hardware del SID

b) Implementar el prototipo de la Interface Grafica de Usuario para la Programación y Visualización del SID (IPVSID).

Y como último objetivo particular:

c) Probar el funcionamiento del prototipo completo en el control de un sistema o proceso, con el fin de comprobar su funcionalidad, desempeño y características generales, y en base a esto determinar su factibilidad de desarrollo y mejora futura, para aplicaciones de enseñanza y automatización.

1.5 Alcances, Delimitaciones y futuros desarrollos.

El alcance de este trabajo persigue la implementación y prueba de la herramienta prototipo en un sistema real, derivado del éxito de la aplicación del SID podremos decir si pudiera llegar a ser en un futuro una opción conveniente, confiable y de bajo costo para la soluciones de automatización de los sectores productivos

Las características más destacables de los controladores difusos que se podrán diseñar con el SID son:

- Definición de un máximo de tres entradas al controlador difuso, y una sola salida.
- Procesamiento difuso tipo Sugeno permitiendo únicamente conjuntos crisp o singleton escalares en la salida.
- El número máximo de conjuntos difusos por entrada será de nueve e igual el número de conjuntos singleton para la salida.
- Tendrá disponibles tres entradas analógicas.
- Contara con una salida capaz de manejar cargas de CC (12V, 400mA) y CA (127V, 15A y 60 Hz) que serán utilizadas como actuadores de la aplicación en particular.
- Solo podrá cubrir aplicaciones que no demanden mayores requerimientos de los que el ofrece, generalmente asociados con la velocidad de ejecución y numero de entradas y salidas y numero de reglas.
- Implementación de esquemas de control difuso como P (Proporcional), PP (Proporcional Prealimentado), y PI (Proporcional Integral).

CAPÍTULO2. INTELIGENCIA ARTIFICIAL Y LÓGICA DIFUSA.

2.1 Inteligencia Artificial y Sistemas Inteligentes.

2.1.1 Inteligencia Artificial.

Una definición de la inteligencia artificial es la siguiente: La Inteligencia Artificial (IA) es el comportamiento de una máquina que, si desempeñado por un humano, podría ser llamado inteligente. Otra definición es, "la Inteligencia Artificial es el estudio de cómo desarrollar computadores que hagan cosas que en el momento, las personas realizan mejor".

Varias habilidades son consideradas signos de inteligencia

- Aprender o entender de la experiencia.
- Tener sentido de mensajes ambiguos o contradictorios.
- Responder rápidamente y exitosamente a nuevas situaciones (respuestas diferentes, flexibilidad).
- Usar razonamiento en la solución de problemas y dirigir conductas eficientemente.
- Entender e inferir de maneras racionales ordinarias.
- Aplicar conocimiento para manipular el ambiente
- Pensar y razonar.
- Reconocer la importancia relativa de diferentes elementos en una situación.

Auque la última meta de la inteligencia artificial es construir máquinas que imiten la inteligencia humana, las capacidades de los productos comerciales actuales de la IA están lejos de exhibir algún éxito significativo en las habilidades justamente listadas. No obstante, los programas de la IA son continuamente mejorados e incrementan la productividad y calidad automatizando varias tareas que requieren alguna inteligencia humana.

2.1.2 El campo de la Inteligencia Artificial.

La inteligencia artificial no es en si misma un campo comercial; es una rama de la ciencias de la computación y una tecnología. Es una colección de objetos e ideas que son apropiadas para la investigación pero que no pueden ser vendidas. Sin embargo la IA proporciona los fundamentos científicos de varias tecnologías comerciales. Las cuales son mostradas en la Figura 2.1.

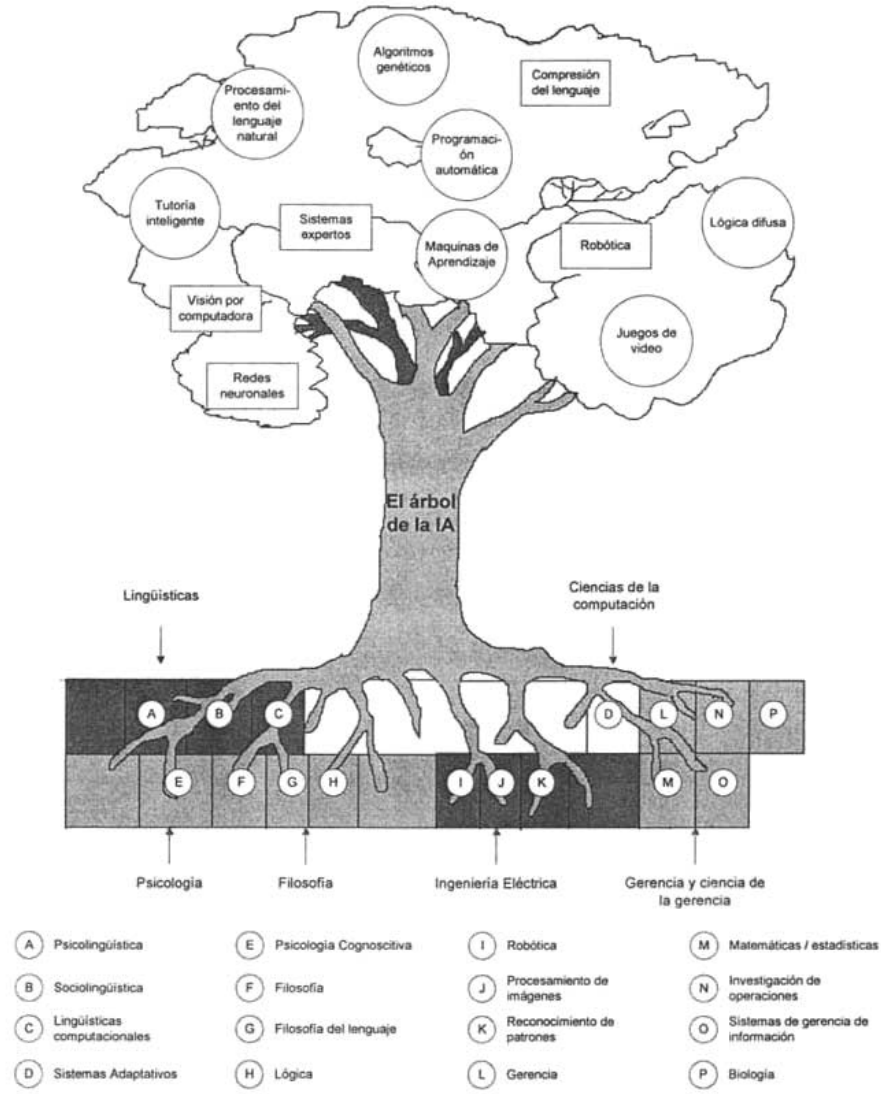


Figura 2.1 Las disciplinas de la IA (sus raíces) y sus aplicaciones.

De las aplicaciones observadas el árbol de la IA para el trabajo presente es de particular interés la lógica difusa, la cual extiende las nociones de la lógica más allá de una simple verdad / falsedad para permitir verdades parciales (o incluso continuas). El conocimiento inexacto y el razonamiento impreciso son aspectos importantes de la experticia en la aplicación del sentido común a las situaciones de toma de dediciones. En la lógica difusa, los grados de membresía son importantes. Por ejemplo en el marco de trabajo de la lógica boleana clásica, un carro puede patinarse o no patinarse cuando los frenos son aplicados. Sin

embargo, un conductor experto puede reconocer el grado en el que el carro se patina y puede aplicar el control de acuerdo a una cantidad variable de patinaje. Una de las primeras aplicaciones comerciales de la lógica difusa fue en la producción superior de frenos antibloqueo mejor conocidos como ABS (por sus siglas en ingles).

2. 2 Lógica Difusa.

Los expertos a menudo confían en el sentido común para resolver problemas. Vemos este tipo de conocimiento expuesto cuando un experto describe un problema usando términos vagos o ambiguos. Por ejemplo, el experto puede indicar, "Cuando el motor esta corriendo realmente caliente decremento un poco la velocidad." Estamos acostumbrados a oír un problema descrito de esta manera y usualmente tenemos poca dificultad con interpretar el uso de términos vagos. Sin embargo, proporcionar a una computadora con el mismo entendimiento es un desafío - ¿Cómo podemos representar y razonar términos vagos en una computadora?

Esta cuestión se responde explorando el tema de la lógica difusa.

Con la lógica convencional, las computadoras pueden manipular valores estrictamente duales, como verdadero/falso, si/no o ligado/desligado. En la lógica difusa, se usan modelos matemáticos para mapear nociones subjetivas, como *caliente / tibio / frío*, para valores concretos que puedan ser manipuladas por los ordenadores.

2.2.1 Conjuntos difusos.

Uno de los primeros conceptos que se debe conocer para entender la lógica difusa es el de *conjunto difuso*. Básicamente, un conjunto difuso es un conjunto que no tiene límites claramente definidos o precisos. A diferencia de los conjuntos clásicos, en los conjuntos difusos la transición de la pertenencia o no-pertenencia de un elemento a un cierto conjunto, es gradual, y esta transición está caracterizada por *funciones de membresía*, las cuales les dan a los conjuntos difusos flexibilidad para modelar expresiones lingüísticas empleadas cotidianamente.

Sea X una colección de objetos denotados genéricamente por x; entonces, un conjunto difuso A en X se define como el siguiente conjunto de pares ordenados:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \ldots\ldots\ldots (2.1)$$

donde $\mu_A(x)$ se conoce como la *función de membresía* (abreviado FM) de A. La

FM transforma cada elemento de X hacia un grado de membresía (o valor de membresía) entre 0 y 1.

Generalmente, X es llamado el *universo de discurso* o, simplemente, el universo, el cual puede consistir de objetos discretos (ordenados o no ordenados) o ser un espacio continuo.

Por ejemplo, el universo ordenado discreto definido por el número de hijos que una pareja puede desear tener, se expresa como: $X = \{0, 1, 2, 3, 4, 5, 6\}$. De esta forma, el conjunto A = *"número ideal de hijos en una familia"* se puede describir como:

$A = \{(0, 0.1), (1, 0.3), (2, 0.7), (3, 1), (4, 0.6), (5, 0.2), (6, 0.1)\}$

donde los grados de membresía asignados, son medidas subjetivas.

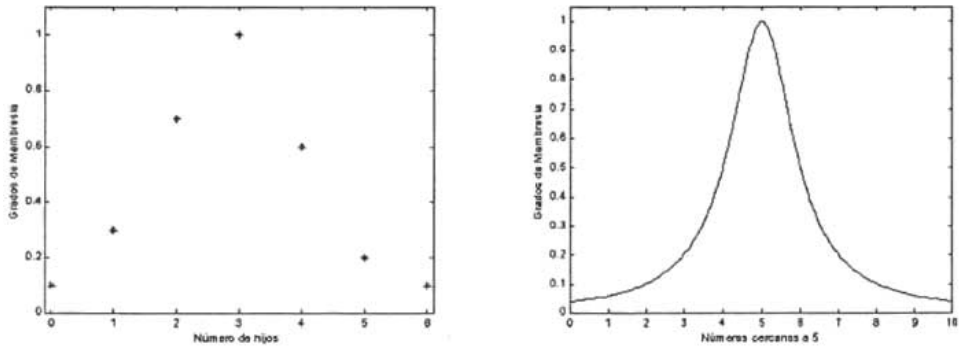
Un ejemplo de universo continuo es el de los números reales, el cual se expresa como: $X = \mathbf{R}$. El conjunto difuso B = *"números cercanos a 5"* se puede expresar de la siguiente manera:

$B = \{(x, \mu_B(x)) \mid x \in X\}$,

donde $\mu_B(x)$ se puede definir como:

$$\mu_B(x) = \frac{1}{1+(x-5)^2}$$

En la Figura 2.2 se muestran las FM correspondientes a los conjuntos difusos A y B arriba descritos.



(a) (b)

Figura 2.2 Funciones de membresía para un universo discreto (a), y para un universo continuo (b).

En la práctica, cuando el universo de discurso X es un espacio continuo, generalmente se divide en varios conjuntos difusos cuyas FM cubren a X de una manera más o menos uniforme. Estos conjuntos se denominan *valores lingüísticos* o *etiquetas lingüísticas* y, normalmente, se les asignan nombres de adjetivos utilizados en el lenguaje común, tales como "grande", "frío", "alto", etc.

Una *variable lingüística* se define mediante el quinteto de elementos siguiente: (x, $T(x)$, X, G, M), donde x es el nombre de la variable; $T(x)$, el conjunto de términos de x, esto es, el conjunto de sus valores lingüísticos o términos lingüísticos; X es el universo de discurso; G, una regla sintáctica que genera los términos en $T(x)$; y, M, una regla semántica, la cual le asocia a cada valor lingüístico A un valor correspondiente $M(A)$, que denota un conjunto difuso en X.

Por ejemplo, si *temperatura* se interpreta como una variable lingüística, entonces su conjunto de términos *T* (*temperatura*) podría ser:

T (*temperatura*) = {frío, *no frío, muy frío, no muy frío,...,*

tibio, medio tibio, algo tibio,...,

caliente, no caliente, muy caliente, no muy caliente, ... }

donde cada término en *T*(*temperatura*) se caracteriza por un conjunto difuso de un universo de discurso que puede ser: *X = [0, 100]*. Normalmente se dice "la temperatura es fría" para denotar la asignación del valor lingüístico "frío" a la variable lingüística *temperatura*. Por otro lado, cuando la variable *temperatura* se interpreta como una variable numérica, se emplea la expresión "temperatura = 30 °C" para asignarle el valor numérico "30" a la variable numérica *temperatura*. La regla sintáctica se refiere a la forma en que se generan los valores lingüísticos en el conjunto de términos *T*(*temperatura*). La regla semántica define la función de membresía de cada valor lingüístico del conjunto de términos.

La Figura 2.3 muestra una distribución de algunas de las funciones de membresía propuestas para la variable *temperatura*.

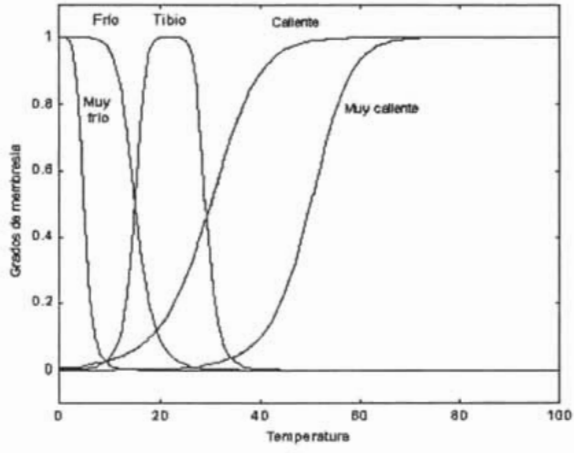


Figura 2.3 Funciones de membresía típicas para el conjunto de términos *T* (*temperatura*).

Del ejemplo previo se observa que el conjunto de términos se compone de varios *términos primarios* (frío, tibio, caliente), modificados por la *negación* ("no") y/o algunos *modificadores* (muy, algo, un poco, demasiado, etc.), que se pueden enlazar con *conectivas* tales como *y*, *o*, *ni... ni....* Las conectivas, los modificadores y la negación se consideran como operadores que cambian el significado de sus operandos en una forma específica, independiente del contexto.

2.2.1.2 Operaciones con conjuntos difusos.

Con los conjuntos difusos se pueden efectuar operaciones similares a las que se realizan con conjuntos clásicos. Las tres operaciones básicas son la *unión*, la *intersección* y el *complemento*.

Unión (disyunción): La unión de dos conjuntos difusos *A* y *B* es un conjunto difuso *C* (escrito como $C = A \cup B$ o como $C = A$ o *B*), cuya FM se relaciona con las de *A* y *B* por:

$$\mu_C(x) = max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x) \ldots\ldots\ldots (2.2)$$

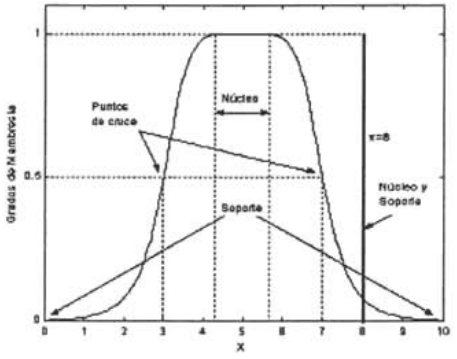
Intersección (conjunción): La intersección de dos conjuntos difusos *A* y *B* es un conjunto difuso *C* (escrito como $C = A \cap B$ o como $C = A$ y *B*), cuya FM se relaciona con las de *A* y *B* por:

$$\mu_C(x) = \min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x) \ldots\ldots\ldots (2.3)$$

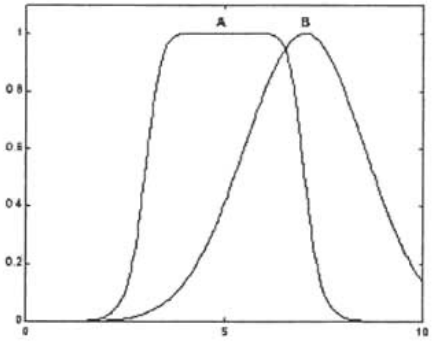
Complemento: El complemento de un conjunto difuso *A*, denotado por $\bar{A}$ ($\neg A$, No *A*), se define como sigue:

$$\mu_{\neg A}(x) = 1 - \mu_A(x) \quad (2.13)$$

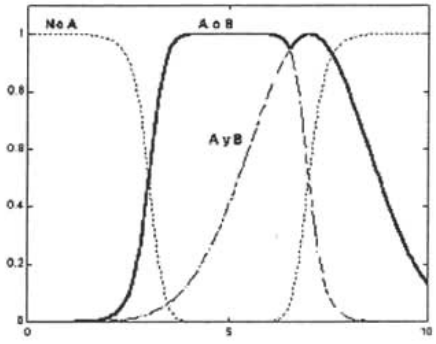
En la Figura 2.4, se muestran gráficamente algunas de las definiciones anteriores.



(a)



(b)



(c)

Figura 2.4 a) Núcleo, soporte y puntos de cruce de un conjunto difuso y de un impulso difuso en x = 8; b) Dos conjuntos difusos A y B; c) Los conjuntos difusos $\bar{A}$, $A \cup B$ *y* $A \cap B$.

2.2.1.2 Funciones de membresía.

Un conjunto difuso se caracteriza completamente por su Función de Membresía (FM). Una forma concisa y conveniente de definir una FM es expresándola como una fórmula matemática. A continuación se describirán algunas clases de funciones parametrizadas empleadas comúnmente para definir FM en una dimensión, esto es, que tienen un sola entrada. Para FM de un orden mayor se puede usar un razonamiento análogo.

a) Función de membresía triangular.

Una FM triangular se especifica mediante tres parámetros {a, b, c}, de la siguiente forma:

$$triangulo(x;a,b,c) = \begin{cases} 0 & \\ \dfrac{x-a}{b-a} & ,x \le a \\ \dfrac{c-x}{c-b} & ,a \le x \le b \\ 0, & ,b \le x \le c \\ & ,c \le x \end{cases} \quad \ldots\ldots\ldots (2.4)$$

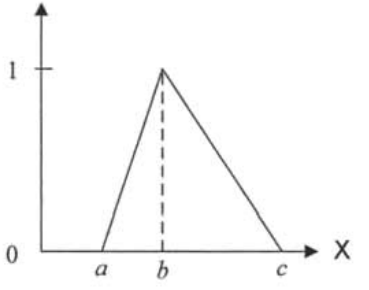


Figura 2.5. Función de membresía triangular

Utilizando operadores min y max, la expresión anterior se puede expresar como sigue

$$triangulo(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \ldots\ldots\ldots (2.5)$$

Los parámetros {a, b, c} (con a < b < c) determinan las coordenadas en *x* de los tres vértices de la FM triangular en cuestión.

b) Función de membresía trapezoidal.

Una FM trapezoidal se determina con cuatro parámetros {a, b, c, d}, como sigue:

$$trapecio(x;a,b,c) = \begin{cases} 0, & \\ \dfrac{x-a}{b-a} & ,x \le a \\ 1, & ,a \le x \le b \\ \dfrac{d-x}{d-b} & ,b \le x \le c \\ 0, & ,c \le x \end{cases} \quad \text{..... (2.6)}$$

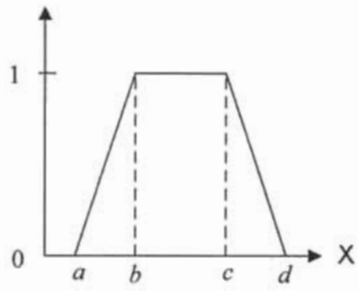


Figura 2.6.Función de membresía trapezoidal.

También se puede definir esta FM en una forma más concisa, empleando los operadores min y max, como sigue:

$$trapecio(x;a,b,c,d) = \max\left(\min\left(\frac{x-a}{b-a},1,\frac{d-x}{d-c}\right),0\right) \text{..... (2.7)}$$

Los parámetros {a, b, c, d} (con $a < b \le c < d$) determinan las coordenadas en *x* de los cuatro vértices de la FM trapezoidal definida.

Debido a la sencillez de sus fórmulas, así como a su eficiencia computacional, las FM triangulares y trapezoidales han sido ampliamente utilizadas, especialmente en aplicaciones de tiempo real. Sin embargo, puesto que ambas funciones se componen de segmentos de líneas rectas, no son suaves en los puntos de los vértices especificados por los parámetros. Por ello, también se emplean FM definidas por funciones suaves y no lineales.

c) Función de membresía gausiana.

Una FM gausiana se especifica con dos parámetros {c, σ}, como se muestra a continuación:

$$gauss(x,c,\sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2} \text{..... (2.18)}$$

Una FM gausiana se determina completamente con *c* y σ; *c* representa el centro de la FM y σ determina su anchura.

d) Función de membresía tipo campana generalizada.

Una FM del tipo campana generalizada (o FM tipo campana) se caracteriza mediante tres parámetros {a, b, c}, como sigue:

$$campana(x;a,b,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \quad \ldots\ldots\ldots \quad (2.9)$$

donde *c* y *a* definen el centro y el ancho de la FM, respectivamente, mientras que el parámetro *b* controla las pendientes en los puntos de cruce. Este parámetro *b* normalmente es positivo; *b* negativo generaría una FM con forma de campana invertida. Esta FM es una generalización directa de la función de distribución de Cauchy definida en la teoría de probabilidad, por ello también es referida como FM de Cauchy.

Debido a su suavidad y notación concisa, las FM gausiana y tipo campana, han llegado a ser cada vez más empleadas para especificar conjuntos difusos. Las funciones gausianas son bien conocidas en probabilidad y estadística, y tienen propiedades útiles como aquella en la que no cambia su forma al multiplicarse por otra (el producto de dos gausianas es una gausiana con un factor de escalamiento) ni con una transformada de Fourier (la transformada de Fourier de una gausiana es una gausiana). La FM tipo campana tiene un parámetro más que la gausiana, por lo cual tiene un grado más de libertad para ajustar sus pendientes en los puntos de cruce.

Aunque las FM gausiana y de tipo campana logran una mayor suavidad, no permiten especificar FM asimétricas, las cuales se emplean en ciertas aplicaciones.

Una FM sigmoidal puede ser abierta por la izquierda o por la derecha, y el producto o la diferencia absoluta de dos de ellas permite definir FM asimétricas. Una FM sigmoidal se define por:

$$sig(x;a,c) = \frac{1}{1+\exp[-a(x-c)]} \quad \ldots\ldots\ldots \quad (2.10)$$

donde *a* controla la pendiente en el punto de cruce *c*.

Otras FM se pueden crear para aplicaciones específicas, si ello se requiere. En general, cualquier tipo de función de distribución de probabilidad estadística puede emplearse como FM. Por ejemplo, si se si se requiere una FM con pendientes distintas en ambos de sus lados, se puede construir como sigue:

$$f(x;a,b,c) = \begin{cases} F_I\left(\frac{c-x}{a}\right), x \le c \\ F_D\left(\frac{x-c}{b}\right), x \ge c \end{cases} \quad \ldots\ldots\ldots \quad (2.11)$$

donde $F_I(x)$ y $F_D(x)$ son funciones monótonas decrecientes, definidas en [0, ∞), tal que $F_I(0)=F_D(0)=1$ y $\lim_{x\to\infty} F_I(x)=\lim_{x\to\infty} F_D(x)=0$.

e) Función de membresía singleton.

Un conjunto difuso que tiene un único elemento en, x_0, es denominado un a singularidad difusa o singleton difuso.

$$\mu_s(x) = 0 \text{, si } x \neq x_0 \text{ y } \mu_s(x) = 1 \text{, si } x = x_0$$

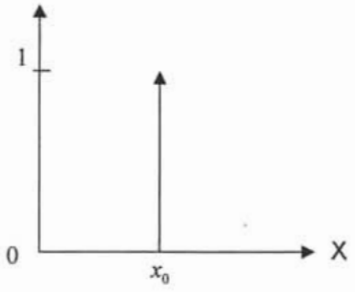


Figura 2.7. Función singleton difusa.

El uso de singleton simplifica considerablemente el proceso de inferencia difusa, y posibilita la implementación electrónica eficiente de los sistemas difusos.

En la Figura 2.8 se muestran las formas de las FM triangular, trapezoidal, gausiana y de Cauchy. Las FM que se muestran en la Figura corresponden a las definidas por los siguientes valores: triángulo(x; 6, 7, 9), trapecio(x; 5, 6, 8, 10), gauss(x; 0.7, 3) y campana (1.5, 5, 3).

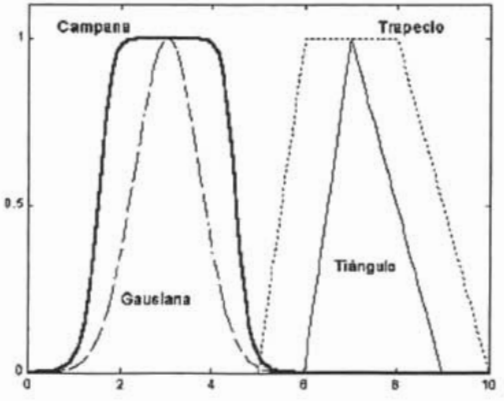


Fig. 2.8 Ejemplos de cuatro tipos de FM parametrizadas.

2.2.2 Sistemas de inferencias difusos.

Un sistema de inferencias difuso es una entidad de cómputo, basada en los conceptos de la teoría de conjuntos difusos, reglas difusas del tipo *si...entonces* y el razonamiento difuso. Para entender esta idea, se definirán inicialmente algunos conceptos.

Operador norma-T.

Un operador norma-T es una función que se aplica a una pareja de valores, de la forma T (*, *), y satisface las siguientes condiciones:

$T(0, 0) = 0, T(a, 1) = T(1, a) = a$ (límites)

$T(a, b) \leq T(c, d)$ *si* $a \leq c$ *y* $b \leq d$ (monotonicidad)

$T(a, b) = T(b, a)$ (conmutatividad) ..... (2.12)

$T(a, T(b, c)) = T(T(a, b), c)$ (asociatividad)

La primera condición establece la generalización para los conjuntos certeros; la segunda, implica que un decremento en los valores de membresía de *A* o *B*, no producirá un incremento en el valor de membresía de $A \cap B$. La tercera condición indica que el orden es indiferente al orden de los conjuntos difusos a ser combinados; la cuarta, permite tomar la intersección de cualquier cantidad de conjuntos, en cualquier orden, agrupándolos por parejas. Algunos de los operadores norma - T más comunes son:

Minimun (MIN): $T_{\min}(a,b) = \min(a,b) = a \wedge b$

Producto Algebraico: $T_{pa}(a,b) = ab$

Producto limitado: $T_{pl}(a,b) = 0 \vee (a+b-1)$ ..... (2.13)

Producto Drástico: $T_{pd}(a,b) = \begin{cases} a, & si\ b = 1 \\ b, & si\ a = 1 \\ 0 & si\ a,b < 1 \end{cases}$

Un operador conorma-T (o norma-S) es una función que se aplica a un par de valores, de la forma S (*, *), y satisface lo siguiente:

$S(1, 1) = 1, S(a, 0) = S(0, a) = a$ (límites)

$S(a, b) \leq S(c, d)$ *si* $a \leq c$ *y* $b \leq d$ (monotonicidad)

$S(a, b) = S(b, a)$ (conmutatividad) ..... (2.14)

$S(a, S(b, c)) = S(S(a, b), c)$ (asociatividad)

La justificación de estos requerimientos es similar a la de los correspondientes para los operadores norma-T. En forma similar, se presentan a continuación cuatro operadores norma-S comunes:

Minimun (MIN): $S_{\max}(a,b) = \max(a,b) = a \vee b$

Suma Algebraico: $S_{sa}(a,b) = a + b - ab$

Suma limitada: $S_{sl}(a,b) = 1 \wedge (a+b)$ ..... (2.15)

Suma Drástica: $$S_{sd}(a,b) = \begin{cases} a, & si\ b = 0 \\ b, & si\ a = 0 \\ 1 & si\ a,b > 0 \end{cases}$$

Reglas difusas si... entonces.

Una *regla difusa* del tipo *si... entonces* (también llamada simplemente regla difusa, implicación difusa o declaración condicional difusa) asume la siguiente forma:

si *x* es *A* **entonces** *y* es *B*..... (2.16)

donde *A* y *B* son valores lingüísticos, definidos por conjuntos difusos en los universos de discurso *X* y *Y*, respectivamente. Frecuentemente a "*x es A*" se le llama el *antecedente* o *premisa*, mientras que a "*y es B*" se le llama el *consecuente* o *conclusión*. En el lenguaje común se tienen muchos ejemplos de reglas difusas de este tipo: "**Si** la presión es alta, **entonces** el volumen es pequeño", "**Si** la velocidad es alta, **entonces** se aplican los frenos ligeramente", "**si** la manzana es roja, **entonces** está madura", etc.

La expresión 2.16 se puede abreviar como $A \rightarrow B$. Esencialmente, esta expresión describe una relación entre dos variables (x, y), lo cual sugiere que una regla difusa de este tipo se puede definir como una relación binaria difusa R en el espacio difuso $X \times Y$. La expresión $A \rightarrow B$ puede interpretarse básicamente de dos formas: *A* está acoplada con *B*, o *A* implica *B*.

En el primer caso (*A* está acoplada con *B*) la relación se define como:

$$R = A \rightarrow B = A \times B = \int_{X \times Y} \mu_A(x) * \mu_B(y)/(x,y) \text{ ..... (2.17)}$$

donde *~ es un operador norma-T y $A \rightarrow B$ representa la relación difusa *R*. En el segundo caso (*A* implica *B*), se tienen los siguientes casos:

Implicación material: $R = A \rightarrow B = \neg A \cup B$..... (2.18)

Cálculo propositivo: $R = A \rightarrow B = \neg A \cup (A \cap B)$..... (2.19)

Cálculo propositivo extendido: $R = A \rightarrow B = (\neg A \cap \neg B) \cup B$..... (2.20)

Modus ponens generalizado: $\mu_R(x, y) = sup\ \{c \mid \mu_A(x)\ *\sim\ c \leq \mu_B(y)\ y\ 0 \leq c \leq 1\}$ ..... (2.21)

donde $R = A \rightarrow B$ y *~ es un operador norma-T.

Aunque todas estas fórmulas son diferentes en apariencia, todas ellas se reducen a la identidad $R = A \rightarrow B \equiv \neg A \cup B$ cuando *A* y *B* son proposiciones en el sentido de lógica bipolar (de dos valores). La Figura 2.9 siguiente ilustra las dos interpretaciones de una regla difusa $A \rightarrow B$.

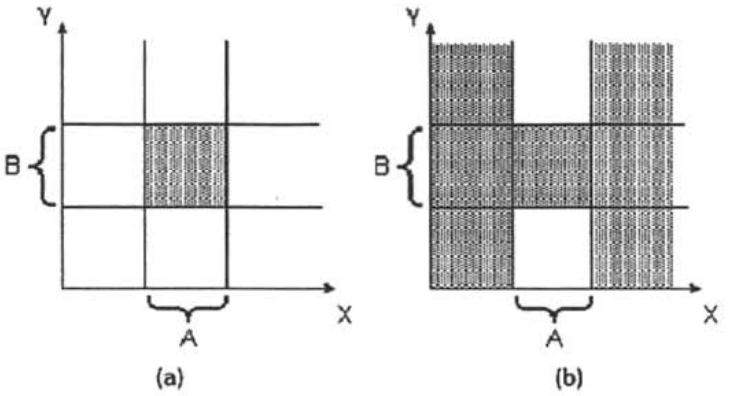


Figura 2.9 Dos interpretaciones de la regla difusa A → B: a) A acoplada con B; b) A implica B.

Se pueden formular diferentes métodos calificados para calcular la relación difusa $R = A \rightarrow B$, basándose en las dos interpretaciones y considerando diferentes operadores norma-T y conorma-T. La relación R puede verse como un conjunto difuso con una FM bidimensional: $\mu_R(x, y) = f(\mu A(x), \mu_B(y)) = f(a, b)$, con $a = \mu_A(x)$ y $b = \mu_B(y)$; la función f realiza la labor de transformar los grados de membresía de x en A y de y en B hacia los de (x, y) en $A \rightarrow B$.

Razonamiento difuso.

El razonamiento difuso es un procedimiento de inferencia que deriva conclusiones a partir de un conjunto de reglas difusas *si... entonces* y hechos conocidos [6]. La regla básica de inferencia en lógica tradicional es el *modus ponens*, de acuerdo con el cual es posible inferir la veracidad de una proposición B a partir de la veracidad de A y la implicación $A \rightarrow B$. Por ejemplo, si A se identifica con el hecho "la manzana es roja", y B con "la manzana está madura", entonces, si el hecho "la manzana es roja" es cierto, entonces el hecho "la manzana está madura" también será cierto.

Sean A, A' y B conjuntos difusos de X, X y Y, respectivamente. Si se asume que la implicación difusa $A \rightarrow B$ se expresa como la relación difusa R en $X \times Y$, entonces el conjunto difuso B inducido por "x es A" y la regla difusa "si x es A, entonces y es B" queda definido por:

$$\mu_{B'}(y) = \max_x \min[\mu_{A'}(x), \mu_R(x, y)] = \vee_x [\mu_{A'}(x) \wedge \mu_R(x, y)] \ldots\ldots\ldots (2.22)$$

O, equivalentemente: $B' = A' \circ R = A' \circ (A \rightarrow B)$

De esta forma, se puede utilizar el procedimiento de inferencia o razonamiento difuso para derivar conclusiones si la implicación difusa $A \rightarrow B$ se define como una relación binaria difusa apropiada.

Sistema de inferencias difuso.

En la literatura, estos sistemas han recibido diferentes nombres: Sistema basado en reglas difusas, sistema experto difuso, modelo difuso, memoria asociativa difusa, controlador lógico difuso o, simplemente, sistema difuso.

Su estructura básica consiste de tres componentes: Una ***base de reglas***, la cual contiene una selección de reglas difusas; una ***base de datos*** (o diccionario), el cual define las funciones de membresía empleadas con las reglas difusas; y, un mecanismo de razonamiento, el cual realiza el procedimiento de inferencia sobre las reglas y los hechos proporcionados, para producir una salida razonable o conclusión.

El sistema de inferencias difuso básico puede tener tanto entradas difusas como certeras (las cuales son vistas como impulsos difusos), pero las salidas que produce son casi siempre conjuntos difusos. En algunas situaciones se necesita una salida certera, por ejemplo, cuando el sistema se emplea como un controlador.

Entonces, surge la necesidad de un método de ***defuzzificación*** para extraer el valor certero que represente mejor a un conjunto difuso.

Un sistema difuso con entradas y salidas certeras efectúa una transformación no lineal de su espacio de entrada hacia su espacio de salida. Esta transformación se lleva a cabo mediante un número de reglas difusas *si... entonces*, cada una de las cuales describe el comportamiento local de la transformación. En particular, el antecedente de una regla define una región difusa en el espacio de entrada, mientras que el consecuente especifica la salida en la región difusa.

Existen diferentes tipos de sistemas de inferencia difusos, los cuales se han utilizado en diversas aplicaciones. Las diferencias básicas pueden estar en los consecuentes de sus reglas difusas, así como en los métodos de defuzzificación que emplean. Los nombres que se les han dado normalmente se toman de las personas que primero los propusieron; así, por ejemplo, se tiene el sistema de inferencias difuso tipo Mamdani, el de tipo Takagi-Sugeno - Kang (TSK) o el de tipo Tsukamoto.

El caso particular del que se ocupa el presente trabajo, se basa en un sistema de inferencias difuso tipo TSK (también llamado simplemente de tipo Sugeno). Por ello, a continuación se describe brevemente en qué consiste este sistema difuso.

Modelo difuso tipo Sugeno.

Este modelo fue propuesto inicialmente por Takagi, Sugeno y Kang en un esfuerzo por desarrollar un método sistemático para generar reglas difusas a partir de un conjunto dado de datos de entrada/salida. Una regla difusa típica en un modelo difuso tipo Sugeno tiene la forma: Si x es A y y es B, entonces $z = f(x, y)$, donde A y B son conjuntos difusos en la parte de antecedentes y $z = f(x, y)$ es una función certera en la parte de consecuentes. Normalmente $f(x, y)$ es un polinomio dependiente de las variables de entrada x y y, pero, en general, puede ser cualquier función que pueda describir apropiadamente la salida del modelo dentro de la región difusa especificada por el antecedente de la regla. Cuando $f(x, y)$ es un polinomio de primer orden, el sistema de inferencias difuso resultante es

llamado un *modelo difuso de Sugeno de primer orden*; si *f* es una constante, entonces se tiene un *modelo difuso de Sugeno de orden cero* el cual es el caso para el presente trabajo. Cabe mencionar que, aunque también se ha desarrollado un *modelo difuso de Sugeno de segundo orden*, esto es, $f(x, y)$ es un polinomio de segundo grado, el presente trabajo se basa en un sistema que emplea el modelo de 1er orden, aprovechando la facilidad de aplicar métodos de optimización a funciones lineales.

La salida de un modelo de Sugeno de orden cero es una función suave de sus variables de entrada, a condición de que las vecindades de sus FM en el antecedente tengan suficiente traslape. En otras palabras, el traslape de las FM en el antecedente determina la suavidad del comportamiento de entrada/salida resultante.

El traslape de las FM en el consecuente no tiene un efecto decisivo en la suavidad.

En la Figura 2.10 se muestra el procedimiento de razonamiento difuso para un modelo difuso de Sugeno de 1er orden.

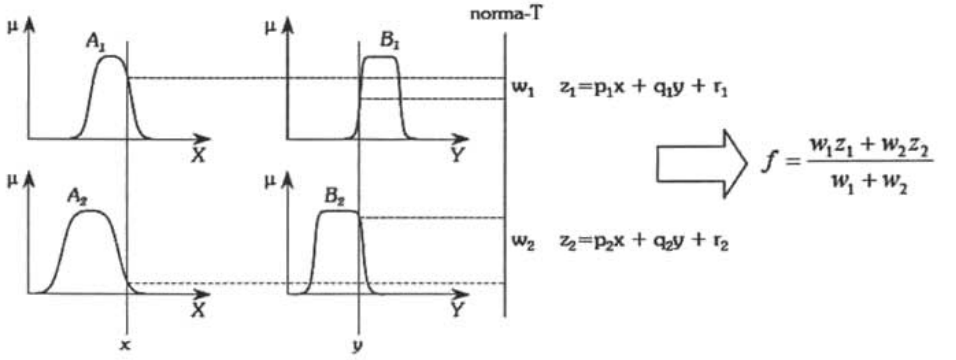


Figura 2.10 El modelo difuso de Sugeno.

Puesto que cada regla tiene una salida certera, la salida global se obtiene Mediante un *promedio ponderado*. En la práctica, la operación de promedio Ponderado se sustituye con una *suma ponderada* (esto es, $f = w_1z_1 + w_2z_2$ en la Figura anterior), a fin de reducir los cálculos, especialmente durante el entrenamiento de un sistema de inferencias difuso. Sin embargo, esta simplificación puede llevar a la pérdida de significado lingüístico de las FM, a menos que la suma de las intensidades de disparo (esto es, $\sum_i w_i$) sea cercana a la unidad.

Un ejemplo de un modelo difuso de Sugeno con dos entradas, una salida y cuatro reglas se expresa como:

Si *X* es pequeño y *Y* es pequeño, entonces $z = -x + y + 1$

Si *X* es pequeño y *Y* es grande, entonces $z = -y + 3$

Si *X* es grande y *Y* es pequeño, entonces $z = -x - 3$

Si X es grande y Y es grande, entonces $z = x + y + 2$

En un sistema de inferencias difuso, el antecedente de una regla define una región difusa local, mientras que el consecuente describe el comportamiento dentro de esa región mediante varios componentes. En el caso de un modelo de Sugeno, tales componentes pueden ser una constante (modelo de orden cero) o una ecuación lineal (modelo de 1er orden). El espacio difuso de entrada se puede particionar con diferentes métodos para formar los antecedentes de las reglas difusas. Los tres métodos básicos son: partición de rejilla, partición de árbol y partición dispersa.

La partición de rejilla sólo requiere un pequeño número de FM para cada entrada; sin embargo, tiene algunos problemas cuando se tiene un número moderadamente grande de entradas. Por ejemplo, un modelo difuso con 10 entradas y 2 FM por cada una de ellas, podría resultar en $2^{10} = 1024$ reglas difusas, lo cual es prohibitivamente grande.

En la partición de árbol cada región se puede especificar en forma única a través de un árbol de decisiones correspondiente. Esta partición resuelve el problema de un incremento exponencial del número de reglas, pero se requieren más FM en cada entrada para definir tales regiones difusas, y esas FM generalmente no tienen un significado lingüístico claro tal como "pequeño", "grande", etc.

Con la partición dispersa se puede limitar el número de reglas a una cantidad razonable; sin embargo, esta partición normalmente se define por los pares de datos de entrada/salida deseados, debido a lo cual no se mantiene ortogonalidad en X, Y o en $X \times Y$. Esto hace difícil estimar la función de transformación global directamente de los consecuentes de cada salida de las reglas.

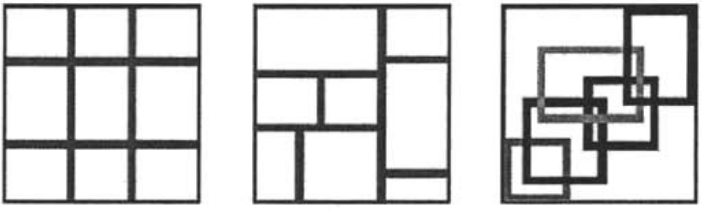


Figura. 2.11 Diferentes métodos de particionar el espacio de entrada: a) Partición de rejilla; b) partición de árbol; c) partición dispersa.

Con todos los elementos hasta aquí establecidos, se puede definir el proceso mediante el cual se construye un sistema de inferencias difuso para una aplicación específica. A este proceso normalmente se le llama *modelado difuso* y, conceptualmente, puede dividirse en dos etapas que no son totalmente independientes. La primera etapa es la identificación de la *estructura de la superficie*, la cual incluye las siguientes tareas:

- Seleccionar las variables de entrada y salida relevantes.
- Elegir un tipo específico de sistema de inferencias difuso.

- Determinar el número de términos lingüísticos asociados con cada variable de entrada y salida. (En un modelo de Sugeno, se debe determinar el orden de las ecuaciones del consecuente).
- Definir un conjunto de reglas difusas del tipo *si... entonces*.

Con la primera etapa del modelado difuso, se obtiene una base de reglas que, por medio de términos lingüísticos, describe en forma aproximada el comportamiento del sistema que se quiere reproducir. El significado de estos términos lingüísticos se determina en la segunda etapa, *identificación de estructura lingüística*, la cual determina las FM de cada término lingüístico (y los coeficientes de cada polinomio de salida de regla si el modelo difuso empleado es tipo Sugeno). Esta etapa incluye las siguientes tareas:

- Elegir una familia adecuada de FM parametrizadas.
- Consultar humanos expertos del sistema que se desea reproducir, para determinar los parámetros de las FM usadas en la base de reglas.
- Refinar los parámetros de las FM usando regresión y técnicas de optimización.

Esta última tarea asume la disponibilidad de un conjunto de datos de entrada/salida deseado, mientras que las dos primeras parten de la disponibilidad de un humano experto. Las técnicas de optimización y adaptación expanden las aplicaciones de los sistemas de inferencia difusos a campos tales como el control adaptable, procesamiento de señal adaptable, regresión no lineal y el reconocimiento de patrones.

2.2.3 Lógica Clásica contra Lógica Difusa.

La mayor parte del razonamiento humano involucra el uso de variables cuyos valores son conjuntos difusos. Esta observación es la base para el concepto de *variable lingüística*, esto es, una variable cuyos valores son definidos por palabras, en vez de números. El uso de estas variables representa un cambio significativo en el análisis de sistemas. Los sistemas difusos, rompen con la tradición histórica del pensamiento que establece que el mundo puede ser caracterizado en forma rigurosa y sin ambigüedades, dividirlo en categorías y, entonces, manipular estas descripciones de acuerdo a reglas formales y precisas.

La lógica clásica, o lógica bivaluada, no resulta adecuada cuando de describir el razonamiento humano se trata, ya que solo "conoce" dos valores, verdad (1) y falsedad (0), mientras que en la vida real existen hechos que no se pueden definir como totalmente verdaderos o totalmente falsos sino que tienen un grado de verdad, o falsedad, que puede variar de 0 a 1. Un ejemplo sencillo se puede apreciar cuando queremos saber si un vaso está lleno de agua, sin embargo al observarlo notamos que éste no está totalmente lleno. Nuestro sentido común no asigna inmediatamente el valor de 0 (falsedad) a nuestra inquietud sino que razona aceptando que está "algo lleno", es decir tiene una alta posibilidad de ser catalogado como lleno, lo cual se puede representar con un valor más cercano a 1 que a 0.

La Figura 2.12a muestra una posible representación de la variable lingüística estatura mediante la lógica clásica bivaluada y la Figura 2.12b muestra una posible representación de esa misma variable lingüística mediante la lógica difusa. Por cada variable lingüística se han establecido tres valores lingüísticos, estatura baja, estatura media y estatura alta.

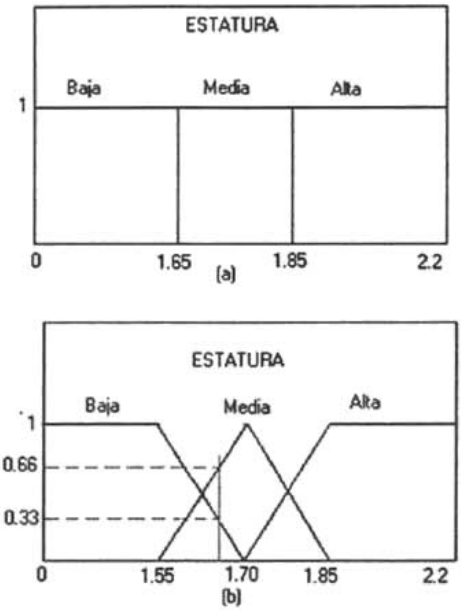


Figura 2.12. Variable lingüística estatura representada mediante: (a) lógica clásica y (b) lógica difusa.

Se puede apreciar que en la lógica clásica, una persona de 1.65m es considerada de estatura baja, mientras que otra con un (1) milímetro más de estatura, es decir, 1.651m es considerada de estatura media. En la lógica difusa, esa misma persona tiene una posibilidad de ser de estatura baja de 0.33 y de ser de estatura media de 0.66, aproximadamente, es decir tiene más posibilidad de ser de estatura media que de estatura baja, mas tiene pertenencia a dos conjuntos diferentes. De esta manera, la lógica difusa permite resolver problemas de la vida real con alto nivel de incertidumbre, que difícilmente pueden ser resueltos por la lógica clásica.

2.2.4 Aplicaciones de la Lógica Difusa.

La lógica difusa se utiliza cuando la complejidad del proceso en cuestión es muy alta y no existen modelos matemáticos precisos; para procesos altamente no lineales y cuando se envuelven definiciones y conocimiento no estrictamente definido (o subjetivo).

En el tiempo presente la lógica difusa esta explotando en Japón, donde un gran numero de compañías están usando tecnología difusa, en artículos y utensilios para el mantenimiento del hogar. Por instancia. Hitachi y Matsushita están

haciendo lavadoras 'inteligentes' que pueden seleccionar la cantidad apropiada de tiempo necesario y la cantidad correcta de detergente después de evaluar la carga de ropa y el tipo de suciedad.

La primera aplicación de la lógica difusa en electrodomésticos fue hecha por un proyectó que comenzó alrededor de 1989 por Toshinohu Haruki y Kenichi Kikuchi de Electrónicos Sanyo Co., Ltd. Ellos realizaron un sistema de control difuso unificado para las funciones básicas de una cámara de video. La lógica difusa fue aplicada para revelación automática, foco automático y balances de blanco automático.

En reconocimiento de voz y control, el mejor ejemplo que ilustra la potencia de la lógica difusa es el sorprendente helicóptero controlado por voz. El helicóptero es supuesto a seguir comandos de voz 'arriba', 'abajo', 'izquierda', 'derecha', y 'cernido'. El proyecto esta todavía en la fase de desarrollo en el Instituto de Tecnología de Tokio. Un modelo pequeño (rotor de 1 metro) esta trabajando aparentemente como fue reportado en 1992 en la primera conferencia internacional IEEE de sistemas difusos en San Diego.

La lógica difusa también ha encontrado su camino dentro y al lado de las computadoras. Un número de compañías están interesadas en usar controladores difusos para controladores de discos duros. El principal objetivo por lo que la lógica difusa ha sido usada es la reducción del tiempo de búsqueda.

La primera implementación comercial importante de la lógica difusa es el resultado del trabajo hecho por investigadores de Hitachi Ltd. Ellos desarrollaron un controlador lógico difuso para un tren. En 1987, el controlador difuso fue usado en un proyecto actual, "el metro de Sendai". El uso de la lógica difusa resultó en aceleraciones y desaceleraciones más suaves.

Todos los ejemplos anteriores están relacionados con hardware. Por esa razón son mas tangibles a la esencia humana y perecen dar una buena demostración para lo que la lógica difusa puede ser usada. Sin embargo, como mencionó Asia, Sugerro y Terano el uso de la lógica difusa no esta limitado a sistemas máquina. También puede ser usada en sistemas basados en humanos y sistemas máquina / humano. El número de aplicaciones en estos dominios es todavía relativamente bajo. Una de las aplicaciones más remarcables, en áreas relacionadas con humanos, es el sistema experto difuso que los seguros Yamachi de Tokio usan para manejar un gran portafolio de acciones. Si embargo, las reglas de pulgar usadas por negociantes son bastante más conocidas. En esta aplicación el sistema toma decisiones concernientes a las compras y ventas basado en un kernel de inferencia de 100 reglas.

Después de 1991 la lógica difusa salió de los laboratorios y llegó a ser una herramienta industrial. La tabla 2.1 incluye solo un pequeño número de proyectos exitosos e intenta demostrar un enorme número de posibles aplicaciones. Por otro lado, La tabla 2.3 presenta los tópicos de investigación actual y futura que están siendo considerados por ingenieros e investigadores japoneses. Uno puede ver que esta tabla representa una buena combinación de varias técnicas y sistemas sociales que prometen un interesante y fructífero futuro.

Tabla 2.1 Aplicaciones de la lógica difusa.

- Control automático de compuertas de represas para plantas de potencia hidroeléctricas. (Tokio Electrical Power.)
- Control simplificado de robots (Hirota, Fuji Electric, Toshiba, Omron)
- Cámara dirigible para la transmisión por televisión de eventos deportivos.(Omron)
- Control Eficiente y estable de motores de carro.(Nissan)
- Control de viaje para automóviles (Nissan, Subaru)
- Sustitución de un experto para la tarea de intercambio de valores.
- Planificación optimizada de horario de autobuses (Toshiba, Nipón-System, Keihan-Expres)
- Sistema archivador de documentos (Mitsubishi Elec.).
- Sistema de predicción para reconocimiento temprano de terremotos.(Oficina de sismología de metrología, Japón)
- Tecnología medica: diagnostico de cáncer.(Kawasaki Medical School)
- Control automático del motor de una aspiradora con reconocimiento del tipo de superficie y cantidad de polvo (Matsushita).
- Control de luz trasera para camcoders (Sanyo).

Durante la revolución difusa, la tecnología difusa fue introducida no solo para un mundo de complejos proyectos industriales, sino para simple aplicaciones caseras de la vida diaria.

Tabla 2.2 Aplicación de los controladores difusos.

Productos de consumo:
Lavadoras.
Hornos de microondas.
Cocedores de arroz.
Aspiradoras.
Cámaras fotográficas.

TVs and VCRs.
Tapetes térmicos.
Traductores de palabras

Sistemas:
elevadores
trenes
grúas
automóviles (motores, transmisión, frenos)
control de trafico

software:
Diagnóstico médico.
Seguridad.
Compresión de datos.

Tabla 2.3 Tópicos de investigación actual y futura.

Sistemas máquina	**Sistemas basados en humanos**	**Sistemas máquina humano.**
Reconocimiento de imágenes y voz.	Modelos de integridad humana.	Diagnostico medico
Reconocimiento de caracteres chinos.	Psicología cognoscitiva.	Procesamiento de datos de inspección
Entendimiento del lenguaje natural.	Modelos de comportamiento y pensamiento	Consulta de transfusión.
Robots inteligentes.	Investigación sensorial	Sistemas expertos

Reconocimiento de patrones.	Análisis público de conciencia.	CAD (diseño asistido por computadora)
Control de procesos.	Manejo de riesgos	Planificación de optimización.
Manejo de producción.	Tareas ambientales	Manejo de personal
Operación de carros y trenes.	Estructuras de relaciones humanas	Planificación de desarrollo
Sistemas de mantenimiento y seguridad.	Modelos de dirección de demanda.	Diagnóstico de equipo
Diagnóstico de averías.	Análisis de energía.	Evaluación de calidad
Operación de sistemas eléctricos de potencia.	Modelos de selección de mercado.	Sistemas de seguro.
Controladores difusos.	Análisis de categoría.	Interface humana.
Electrodomésticos.	Psicología social.	Toma de decisiones de gerencia.
Control aplicado.		Toma de decisiones multipropósito.
		Bases de conocimiento.
		Base de datos.

A pesar de los últimos sucesos que la lógica difusa ha logrado durante algunos de los últimos años, los críticos y los no creyentes están todavía altercando sobre su legitimidad. Esto es especialmente en el dominio de los sistemas de control. Eso es por que el diseño de controladores difusos no provee de un ambiente para

análisis matemático. Sin embargo nada de eso ha detenido ni detendrá a la lógica difusa de probarse en un gran número de aplicaciones.

2.2.5. Sistemas de control difusos.

El control difuso es la aplicación de la inferencia difusa a la automatización de procesos.

Una configuración básica de un sistema de control difuso se muestra en la Figura 2.13.

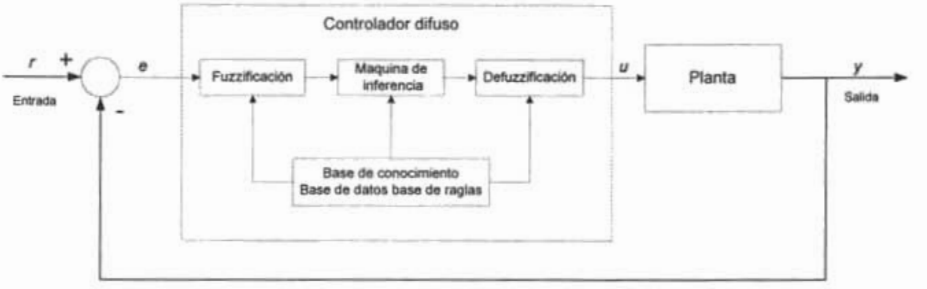


Figura 2.13. Sistema de Control Difuso

De la Figura se puede observar que el esquema de un sistema de control difuso retroalimentado es bastante similar al esquema de control de un sistema de control convencional retroalimentado a excepción de que el controlador convencional ha sido sustituido por un controlador lógico difuso.

El sistema de control cerrado generalizado es mostrado en la Figura 2.14. El bloque P representa la planta controlada, el bloque C representa el controlador, el bloque S especifica las especificaciones de desempeño deseadas del sistema cerrado. En control convencional el bloque P y C son asumidos lineales (o linealizados) y el bloque S define la función de costo, o criterio de desempeño, por ejemplo, margen de estabilidad, tiempo de levantamiento, tiempo de asentamiento, sobre paso, error en estado estacionario, error integral cuadrado, etc. Siguiendo algunos disturbios.

Las siguientes características se aplican a los procesos industriales:

- El proceso físico P es muy complejo, ya sea que no se conoce explícitamente o es muy difícil describir en términos analíticos, y
- Las especificaciones S idealmente demandan un alto grado de autonomía del sistema, así que el sistema cerrado puede operar satisfactoriamente y sin intervención a pesar de fallas en el sistema.

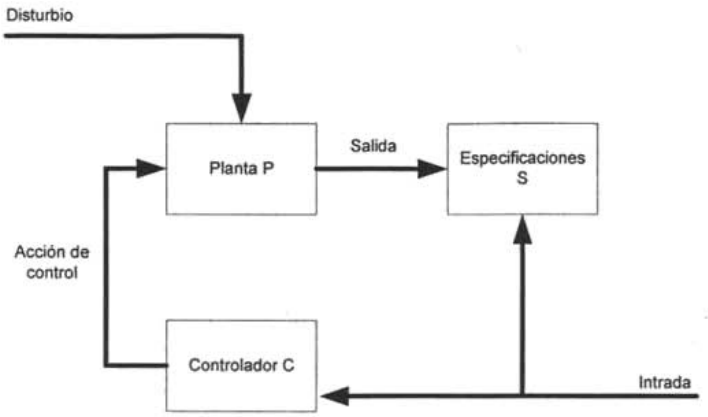


Figura 2.14 Variables de un sistema de control cerrado.

El problema del control difuso puede ser expresado precisamente de la misma manera: dada la planta P, encontrar un controlador C que satisfaga las especificaciones S que pueden ser cuantitativas o cualitativas. La diferencia básica aquí es que en el control difuso no es necesario tener un descripción explicita de la planta P. Además, puede no ser siempre posible separar la planta y el controlador. Ya que el control difuso es basado en reglas, las reglas de control son embebidas en el sistema y forman un elemento integral del sistema. Esta estructura presenta nuevas posibilidades para mejorar los sistemas de manufactura integrados.

2.2.5.1 Controlador Difuso.

La estructura de un controlador difuso, consiste de tres partes básicas: La unidad de fuzzificación en la terminal de entrada, la máquina inferencia construida en el la base de reglas de control lógico difuso en el centro, y la unidad de defuzzificación en la terminal de salida, como es mostrado en la Figura 2.13.

El modulo de fuzzificación transforma los valores físicos de la señal de proceso actual, la señal de error en la Figura 2.13 la cual es la entrada a el controlador difuso, en un subconjunto difuso normalizado consistente de un intervalo para el rango de los valores de entrada y una función de membresía asociada que describe los grados de confidencia en que la entrada pertenece a este rango. El propósito de este paso de fuzzificación es hacer la señal física de entrada compatible con la base de reglas de control difuso en el centro del controlador. El subconjunto difuso, el intervalo y la función de membresía, tiene que ser seleccionados por el diseñador de acuerdo a la aplicación particular en curso. En otras palabras, dependiendo de la naturaleza de la planta dada y la señal de referencia, el controlador difuso tiene que ser diseñado para ajustarse a la necesidad, así como hacer que el sistema de control de lazo cerrado de seguimiento de set point trabaje para esa aplicación particular. Esta situación es parecida a la del diseño de un controlador convencional para un sistema

específicamente dado, donde no existe controlador universal en el diseño practico.

El rol de la máquina de inferencia en el controlador difuso es importante para hacer que el controlador trabaje- y trabaje efectivamente. El trabajo de la máquina es crear las acciones de control, en términos difusos, de acuerdo a la información proporcionada por el modulo de fuzzificación y los requerimientos de seguimiento de set point. Específicamente, la base de reglas es un conjunto de reglas SI – ENTONCES de la forma.

R^1: SI la entrada del controlador e_1 es E_{11} Y....Y

la entrada del controlador e_n es E_{1n}

ENTONCES la salida del controlador u_1 es U_1.

.

.

R^m: SI la entrada del controlador e_1 es E_{m1} Y....Y

la entrada del controlador e_n es E_{mn}

ENTONCES la salida del controlador u_m es U_m.

Aquí, los subconjuntos difusos E_{11}, ..., E_{m1} comparten el mismo subconjunto E_1 y la misma función de membresía μ_{E_1} definida en E_1, y los subconjuntos E_{1n}, ..., E_{mn}, comparten el mismo subconjunto E_n y la misma función de membresía μ_{E_n} definida en E_n. En general, m reglas producen m salidas del controlador u_1,.....u_m, perteneciendo a m subconjunto difusos U_1,.....U_m, en los que, claro, alguno de ellos puede traslaparse. El establecimiento de estas reglas depende de la experiencia de trabajo del diseñador, el conocimiento sobre la planta física, la habilidad de análisis y de diseño, etc., y es, así, mas o menos subjetivo. Así, un buen diseño puede hacer que el controlador trabaje; un diseño óptimo puede hacer que trabaje efectivamente. Esta situación es justamente como en el diseño convencional: un diseño específico no es único en general. Aquí, básicamente, lo que se tiene que determinar son la selecciones de las variables de entrada y de salida del controlador y las reglas SI-ENTONCES.

El módulo de defuzzificación es la conexión entre la base de reglas de control y la planta física a ser controlada, el cual juega el rol de un transformador que mapea las salidas del controlador (generada por la base de reglas de control en términos difusos) otra vez a los valores crisp que la planta puede aceptar. Así, en un sentido, el modulo de defuzzificación es el inverso del modulo de fuzzificación. Las salidas del controlador u_1,.....u_m, por la base de reglas anterior, son señales difusas pertenecientes a los subconjuntos U_1,.....U_m, respectivamente. El trabajo del modulo de defuzzificación es convertir estas salidas del controlador en señales crisp reales, u, y entonces enviarlas a la planta física como una acción de control para seguimiento. Lo que tiene que ser determinado en esta fase es esencialmente una formula de defuzzificación. Existen disponibles varias formulas de defuzzificación comúnmente usadas, que son por naturaleza fórmulas de

promedios de peso en varias formas y que se mencionaran en el siguiente apartado, en el cual se describirá con mas detalle, el funcionamiento de cada una de las módulos constitutivos del controlador difuso.

2.2.5.2 Diseño y operación de un controlador difuso.

El esquema de diseño contiene los siguientes pasos:

1. Definir las entradas y variables de control – determinar que estados del proceso serán observados y que acciones de control serán consideradas.
2. Definir la interface de condición – fijar la manera en que las observaciones del proceso son expresadas como conjuntos difuso.
3. Diseñar la base de reglas – determinar que reglas deben aplicarse bajo que condiciones.
4. Diseñar la unidad computacional - suministrar los algoritmos para desempeñar la computación difusa. Aquellos que generalmente llevan a la salida difusa.
5. Determinar las reglas de acuerdo a que declaraciones de control difuso pueden ser transformadas en acciones de control crisp.

La estructura típica de un controlador es dada en la Figura 2.15

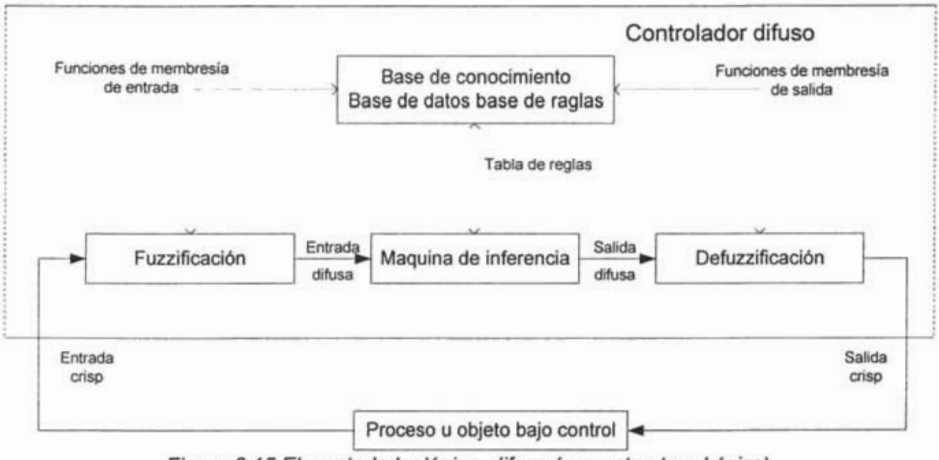


Figura *2.15 El controlador lógico difuso (una estructura básica)*

El primer paso es usual en el diseño de cualquier controlador. Se seleccionan las variables que pueden ser medidas. Generalmente llagan a ser las entradas del controlador. El paso 2 representa el proceso de fuzzificación, el paso 4 la inferencia difusa y el paso 5 el proceso de defuzzificación.

Para ilustrar el proceso de diseño de un sistema de control difuso mostraremos los pasos involucrados en el mismo a través de un ejemplo.

Un ejemplo de control difuso.

Suponga que tiene una solución química en un contenedor. La presión y la temperatura son dos entradas a nuestro sistema. Dependiendo de sus valores, se quiere variar el flujo de agua alrededor del contenedor. La Figura 2.16 muestra el diagrama básico del sistema.

Figura 2.16 Diagrama de bloques básico del sistema

Una vez definidas las entradas y variables de control el primer paso en el diseño un controlador difuso dado consiste en asociar términos lingüísticos a cada uno de los parámetros físicos involucrados. El objetivo es particionar el dominio de cada variable y dar un nombre a cada porción del dominio de esa variable.

La tabla 2.4 muestra las particiones para la variable temperatura. En este caso, los rangos fueron subjetivamente seleccionados. La selección generalmente depende del diseñador y/o experto. En cualquier caso, tiene que reflejar el conocimiento recolectado sobre el sistema.

Tabla 2.4 Rangos para la variable temperatura.

Conjunto Difuso	Termino lingüístico	Rango
1	Extremadamente frío	-50 a -10
2	Muy frío	-15 a 0
3	Frío	-5 a 20
4	Fresco	15 a 50
5	Caluroso	45 a 75
6	Caliente	70 a 110
7	Muy caliente	105 a 175

Cada uno de los rangos tiene una función asociada con él. Ella determinará los grados de membresía. Como mencionamos inicialmente, estas funciones son llamadas funciones de membresía. La asociación de una función de membresía a un rango lleva a la construcción de un conjunto difuso.

El mismo procedimiento es usado para definir los conjuntos difusos para las otras variables, presión y flujo de agua, las tablas no son dadas aquí, mas las particiones pueden ser vistas en las Figura 2.18 y 2.19.

Existe una variedad de funciones de membresía que pueden ser usadas dependiendo del proceso y el parámetro particular utilizado. Muchas funciones usadas en la practica son lineales o combinaciones de funciones lineales debido a que hacen los cálculos mas simples y no requieren de una gran cantidad de memoria de almacenamiento.

Las Figuras 2.17, 2.18 y 2.19 muestran las particiones y sus correspondientes funciones de membresía para la presión, la temperatura y el flujo de agua. El establecimiento de estas graficas constituye la primera parte del diseño.

La segunda parte del diseño consiste en definir el esquema de control y ponerlo en forma de base de reglas. La base de reglas o base de conocimiento debe contener la información recolectada sobre el sistema y debe implementar la estrategia de control necesaria.

En orden para proceder con este ejemplo, y por motivo de simplicidad, se asume que el sistema ha trabajado correctamente usando las cuatro reglas siguientes:

1. Si la temperatura es extra fría y la presión es media Entonces el flujo de agua es débil.
2. Si la temperatura es fría y la presión es media Entonces el flujo de agua es medio.
3. Si la temperatura es fresca y la presión es media Entones el flujo de agua es fuerte.
4. Si la temperatura es caliente y la presión es media Entonces el flujo de agua es muy fuerte.

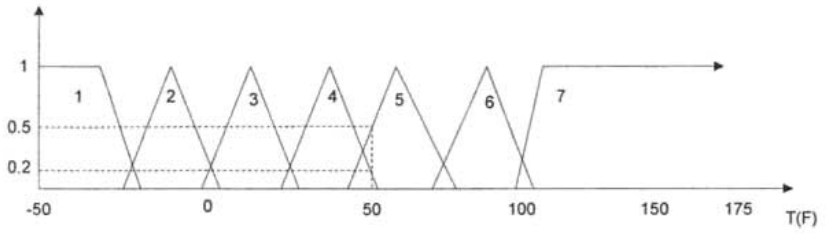


Figura 2.17. Particiones de temperatura y funciones de membresía. (1: Extremadamente-frió, 2: muy-frió, 3: frió, 4: fresco, 5: caluroso, 6: caliente, 7: muy-caliente).

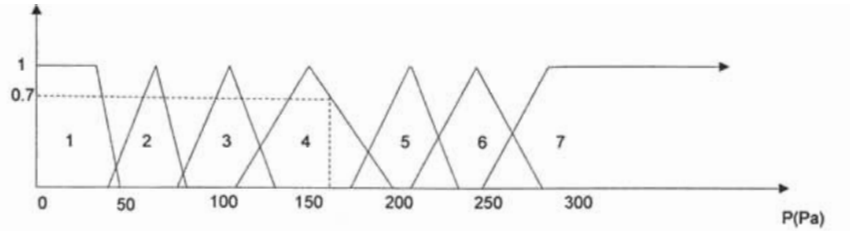
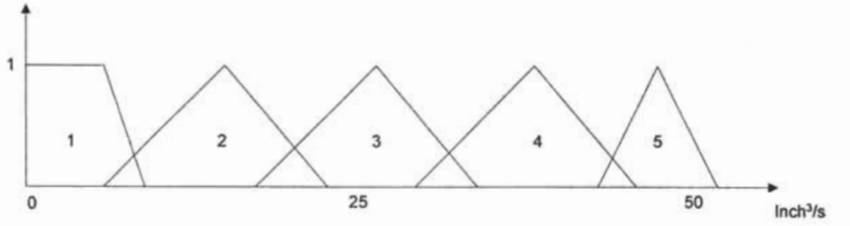


Figura 2.18 Particiones de presión y funciones de membresía. (1: muy-baja, 2: baja, 3: media-mínima, 4: media, 5: media-máxima, 6: alta, 7: muy-alta 15).



2.19 Particiones de flujo de agua y funciones de membresía. (1: muy-débil, 2: débil, 3: media, 4: fuerte, 5: muy fuerte).

En esta fase, el controlador difuso esta completamente diseñado. En otras palabras, los conjuntos difusos están definidos y las reglas están formuladas.

Una vez diseñado el controlador difuso el paso subsiguiente es implementar el controlador en el sistema de control.

El diagrama de bloques en la Figura 2.20 ilustra el esquema de implementación de un sistema de control basado en lógica difusa. Después que los datos han sido adquiridos, son alimentados al controlador difuso que consiste de tres pasos: fuzzificación, evaluación de reglas y defuzzificación.

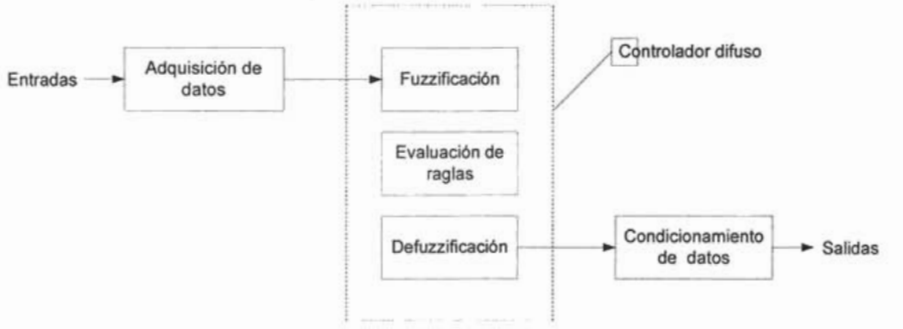


Figura 2.20. Esquemas de implementación de un sistema de control basado en lógica difusa.

La fuzzificación es el proceso que mapea los valores numéricos crisp del mundo real de las entradas a valores numéricos difusos. La parte de evaluación de reglas constituye la inteligencia del controlador, y es la responsable de la toma de

decisiones. La parte de defuzzificación corresponde a una operación de mapeo. Transforma valores numéricos del ambiente difuso al ambiente (mundo real) crisp.

Estos tres pasos son usualmente transparentes al usuario pues son ejecutados a gran velocidad por una unidad computacional. Sin embargo, una breve explicación de cada un de ellos es presentada en seguida en base al ejemplo previo, con el fin de clarificar la operación del controlador difuso.

Fuzzificación.

El proceso de fuzzificación es ejecutado durante el tiempo de ejecución y consiste de asignar un grado de membresía a la entrada crisp. Actualmente mapea valores del mundo real a un número entre 0 y 1. Continuando con el ejemplo, en la Figura 2.14 se muestra que el valor de 50°F es 20 % fresco y 50% caluroso. En otras palabras el pertenece al conjunto 'fresco' en un grado de 0.20 y pertenece al conjunto 'caluroso' en un grado de 0.50. Para la presión, la Figura 2.15 muestra que el valor de 150 Pascales, por ejemplo, es 70% medio, es decir, el pertenece al conjunto medio en un grado de 0.70.

Evaluación de reglas.

En general, todas las reglas son de la forma:

Si [condición x] **y** [condición y] **entonces** [resultado z]

El operador de conjunción '**y**' trabaja como una función mínimo que entregara el mínimo de la condición como el valor que el resultado debe tomar y también la regla se evalúa al mismo valor.

El proceso de evaluación de reglas consiste en usar los valores difusos (grados entre cero y uno) obtenidos después de la fuzzificación y evaluarles en orden vía la base de reglas para obtener un valor difuso para la salida.

Por ejemplo, cuando el valor crisp de la entrada de temperatura es 50ºF, los conjuntos difusos penitentes para la temperatura son los conjuntos "fresco" y "caluroso". Cuando el valor crisp de la entrada de presión es 150 Pascales (Pa) el conjunto difuso para presión que es aplicable es el conjunto "medio". Esto resulta en la activación de las reglas tres y cuatro. Para las entradas crisp (50°F, 150 Pa) solo las reglas tres y cuatro son relevantes y necesitan ser evaluadas, las reglas uno y dos no son relevantes y no hay necesidad de evaluarlas.

Evaluación de la regla 3.

Si la temperatura es "fresca" y la presión es "media" entonces el flujo de agua es "fuerte".

La temperatura es fresca en el grado de 0.20.

La presión es media en el grado de 0.70.

El mínimo de 0.70 y 0.20 es 0.20

La regla es evaluada a 0.20. De acuerdo a esta evaluación, la salida (flujo de agua) debe ser fuerte en el grado de 0.2.

Evaluación de la regla 4.

Si la temperatura es "calurosa" y la presión es "media" entonces el flujo de agua es "muy fuerte"

La temperatura es "calurosa" en el grado de 0.50.

La presión es media en el grado de 0.70.

El mínimo de 0.70 y 0.50 es 0.50

La regla es evaluada a 0.50. De acuerdo a esta evaluación, la salida (flujo de agua) debe ser "muy fuerte" en el grado de 0.2.

En esta fase, las dos reglas relevantes son evaluadas (3 y 4), pero ninguna decisión absoluta es hecha todavía hasta lo que debe ser el valor de la salida. Una inferencia final debe ser hecha a través del proceso de defuzzificación.

Defuzzificación.

El proceso de defuzzificación consiste de combinar los valores obtenidos del paso de evaluación de reglas. Y calcular en orden reciproco para obtener uno y solo un valor crisp a lo que la salida debe ser igual.

Algunas formulas disponibles de defuzzificación comúnmente usadas, lógicamente significativas, y prácticamente efectivas, son las tres siguientes:

El método de defuzzificación máximo.

En este método, solo el número más grande obtenido del paso de evaluación de reglas será tomado en cuenta, los otros son descartados. En el ejemplo que nosotros tenemos aquí, la evaluación de la regla tres y cuatro ha llevado a los grados 0.20 y 0.50. Con el método de defuzzificación máximo, solo el valor 0.5 es tomado en cuenta, y el término "muy fuerte" será usado. El valor exacto que el flujo de agua debe tomar es el valor que corresponde a el máximo de su función de membresía, es decir, el punto de membresía máxima, en este caso es 43 pulgadas cúbicas por segundo como se muestra en la Figura 2.21

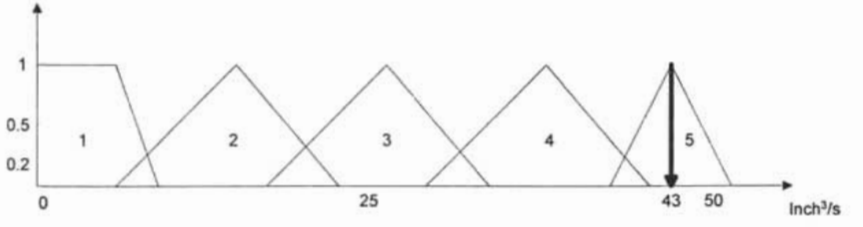


Figura 2.21 El método de defuzzificación máximo (1: muy-débil, 2: débil, 3: media, 4: fuerte, 5: muy fuerte).

El método del centro de los máximos (CM).

Este método toma todos los valores en cuenta y computa un peso medio basado en la formula del centro de gravedad.

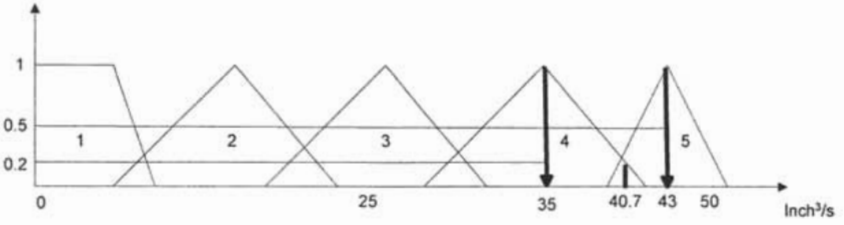
$$X = \sum_{i=1}^{n} \frac{f_i x_i}{fi} \quad \text{..... (2.23)}$$

Donde f_i es el termino difuso i – enésimo para la salida del sistema difuso y x_i es la posición singleton del mismo termino, n es el numero de términos asociados con la salida.

Aplicando la formula al ejemplo lleva a:

$$Salida\_Defuzzificada = \frac{(0.2)(35)+(0.5)(43)}{0.2+0.5} = 40.7$$

El valor de 40.7 inch^3/s es el valor actual que el flujo del agua debe tener como es mostrado en la Figura 2.22



2.22 El método de defuzzificación centro de los máximos (1: muy-débil, 2: débil, 3: media, 4: fuerte, 5: muy fuerte).

El método del centro de área (CA).

Este método CA es similar al método CM excepto que el usa una integración en lugar de una sumatoria. El Método CA no es frecuentemente usado como el método CM debido a que requiere más cálculos.

2.2.5.3 La necesidad de control difuso.

El ejemplo previo ilustra la simplicidad con que un controlador difuso puede ser diseñado. Muestra una alternativa para las ecuaciones matemáticas y el análisis. La simplicidad consiste en definir las entradas y la salida del sistema, definiendo sus correspondientes conjuntos difusos y escribiendo la base de reglas. Ninguna ecuación matemática es usada y ningún análisis clásico es requerido. Lo único que es necesario, es justamente un buen conocimiento del sistema para definir los conjuntos difusos adecuados para cada parámetro y una base de reglas completa para implementar la estrategia de control.

La demanda de este tipo de métodos de diseños y de implementación esta incrementándose a medida que los sistemas llegan a ser más y más complejos. Esto es especialmente cierto en el caso donde el sistema que se va a controlar no tiene un modelo matemático exacto o el sistema no es lineal. Para estos sistemas, otros métodos ascienden del dominio de la inteligencia artificial (IA) comenzando a presentar soluciones eficientes y razonables.

La lógica difusa definitivamente es una de estas técnicas emergentes. Ha sido calificada como una de los primeros productos útiles de la IA en el dominio del control y una de las esencias de la computación cognoscitiva.

Los japoneses han estado usando lógica difusa incluso para sistemas que pueden ser implementados usando métodos convencionales. Esto es debido al hecho de que con el control difuso, el ciclo de diseño parece ser mas corto por que los pasos analíticos se saltan y son remplazados por el paso de implementación del conocimiento. La otra razón es, con las funciones de membresía lineales que son usualmente usadas en control, la cantidad de memoria necesaria para almacenar el código relacionado al controlador es muy pequeña. En adición, solo cálculos mínimos son requeridos para implementar los controladores difusos. Esto hace a los simples microcontroladores de 8 y 16 bits muy adecuados para un gran número de aplicaciones.

En resumen las características principales de un sistema de control difuso son las siguientes: son más amigables en su uso, consumen menos energía, pueden tratar con información vaga e imprecisa, no usan un número excesivo de reglas, es decir, son más robustos y todo ello los hace mucho más económicos. Básicamente la lógica difusa explota el poder propio de la imprecisión humana.

2.2.5.4 Controladores difuso vs. Controladores convencionales.

Los sistemas convencionales han sido usados por más de 50 años. Estos sistemas están basados en modelos analíticos. Un modelo es usualmente representado matemáticamente por un conjunto de ecuaciones. Dadas las entradas las ecuaciones deben ser usadas para calcular las salidas. Hasta hora esta aproximación ha trabajado para un gran número de sistemas. Algunas veces sin embargo, las ecuaciones son demasiado complejas para trabajarlas en microcontroladores de 8 y 16 bits. En otros casos el sistema a controlar es demasiado complejo, y no existe un modelo matemático exacto que pueda representarlo. También, los sistemas no lineales son, en muchos casos, aproximados por representaciones lineales, y los controladores para estos sistemas son usados con muchas restricciones.

En orden para tratar con este tipo de sistemas, los ingenieros de control, comenzaron mirando hacia nuevas metodologías. Las soluciones potenciales se levantaban del dominio de la inteligencia artificial.

Una de estas técnicas prometedoras fue la lógica difusa. Esta técnica se probó a través de un número de aplicaciones. Uno de los primeros estudios en esta área mostró, desde el punto de vista teórico, que las acciones de control PID pueden ser realizadas (en algunos casos) usando métodos de control difuso. En otras palabras, los controladores PID se mostraban como un caso especial (un subconjunto) de los controladores difusos.

En el nivel práctico un estudio comparativo llego a la conclusión de que un controlador difuso (CD) presentaba mas robustez que un controlador PID convencional para un ejemplo lineal de servo – control. El CD mostró mejor desempeño cuando el sistema fue sujeto a disturbios internos (modificación de los parámetros del proceso) y también cuando el sistema fue sujeto a disturbios en el

control (disturbios externos). En adición, el CD mostró menor consumo de energía. Sin embargo, el PID, presenta mejores resultados cuando se introducen retardos descuidados. La principal conclusión de este estudio fue que, una combinación entre el control PID y el CD, llevara a sistemas que puedan compensar retardos descuidados, y al mismo tiempo ser robustos.

Esta opinión también ha sido compartida por otros investigadores. Ha sido comentado que si el 90 % de control satisfactorio ha sido logrado por el control convencional. El último 10 % para el control perfecto debería ser probablemente realizado a través del uso de control inteligente.

Por ejemplo, se han desarrollado controladores difusos para poner a punto controladores PID, cambiando las constantes *Kp*, *Ki* y *Kd*.

El uso de técnicas convencionales en conjunción con lógica difusa no esta limitado para poner a punto PIDs. Otros investigadores han usado las dos técnicas juntas pero independientemente. En este tipo de aplicaciones, los dos controladores son usados en diferentes etapas del proceso de control.

Las tablas 2.5 y 2.6 muestran respectivamente algunas de las desventajas de los controladores convencionales y las ventajas que tiene el uso de controladores difusos.

Tabla 2.5 Limitaciones de los controladores convencionales.

• Plantas no lineales. Los modelos lineales eficientes de los procesos u objetos bajo control son muy restrictivos. Los modelos no lineales son computacionalmente intensivos y tienen problemas complejos de estabilidad.
• Plantas con incertidumbre. Una planta no tiene modelos de calidad debido a la incertidumbre y carencia de un conocimiento perfecto.
• Multivariables, multiciclos y ambientes restrictivos. Los sistemas multivariable y multiciclo tienen complejas restricciones y dependencias.
• Incertidumbre en mediciones. Las mediciones inciertas no necesariamente tiene modelos de ruido estocástico.
• Comportamiento temporal. Las plantas, los controladores y los ambientes y su restricciones y varían con el tiempo, Además los retardos de tiempo son difíciles de modelar.

Tabla 2.6 Beneficios de los controladores difusos.

• Los controladores difusos son más robustos que los controladores PID debido a que ellos pueden cubrir un rango más amplio de condiciones de operación que la que los PID pueden, y pueden operar con ruido y disturbios de diferente naturaleza.

- Desarrollar un controlador difuso es mas barato que desarrollar uno basado en modelo u otro controlador que haga la misma cosa.

- Los controladores difusos son personalizables, ya que es más fácil de entender y modificar sus reglas, las cuales no solamente usa una estrategia de un operador humano sino que también son expresadas en lenguaje natural.

- Es fácil de comprender como opera un controlador difuso y como diseñarlo y aplicarlo a una aplicación concreta.

Después de observar las aplicaciones anteriores y las investigaciones hechas en orden para comparar el CD y los métodos de control convencional, llega a ser claro que lo principal para hacer control perfecto es usar las dos técnicas en conjunción una con otra siempre que sea lo apropiado. Aun así el control difuso puede ser aplicado independientemente con resultados muy satisfactorios.

Por lo tanto son dos los tipos de control difuso: en el primero el controlador difuso ejecuta directamente las acciones de control y así remplaza completamente al algoritmo de control convencional, en el segundo el controlador lógico difuso es implicado en un sistema de control convencional y así llega a ser parte del algoritmo híbrido de control, para realzar o mejorar el desempeño de el sistema de control global. El presente trabajo esta basado en el primer tipo.

2.2.6 Desarrollo e implementación de Sistemas de Control Difusos.

La realización de los pasos de diseño de un controlador difusos puede ser lograda relativamente en una corta cantidad de tiempo usando un paquete o software especializado. El paquete o herramientas de diseño difuso, pueden también ayudar en la implementación del diseño. Una vez, que es hecho, la herramienta puede generar un programa (en 'C' o en ensamblador) que represente el algoritmo de control difuso. Este programa puede entonces ser descargado a un sistema basado en microcontrolador para correr la aplicación. Durante el tiempo de ejecución este programa toma cuidado de las tres fases del proceso difuso, es decir, fuzzificación, evaluación de reglas y defuzzificación.

2.2.6.1 Herramientas software para el diseño de sistemas difusos.

Como resultado de la creciente demanda de los controladores difusos en un gran número de aplicaciones y gracias a la creciente industria tecnológica de la información se han desarrollado más de una docena de paquetes de diseño o herramientas software (herramientas CAD) que son exitosamente usados en diferentes aplicaciones para el diseño de controladores difusos. Algunos de ellos son específicos para alguna tecnología difusa, otros son universales e incluyen una caja de herramientas de diseño difuso. Estos productos comenzaron a aparecer en el mercado en los finales de 1980 y principios de 1990.

En cualquier caso las metas de los paquetes de diseño son usualmente:

- Acelerar un proceso de diseño.
- Hacer el diseño simple, de uso fácil y acertado
- Proporcionar la posibilidad de aplicar lógica difusa y métodos de control sin un entendimiento profundo de cómo trabajan.

Generalmente estos herramientas asisten al diseñador en la análisis, simulación y diseño de sistemas difusos y son universales, algunas se incluyen como cajas de herramientas en herramientas de diseño universales y constituyen partes difusas de los mismos, en general este tipo de herramientas proporcionan como salida un código C portable. También existen otras herramientas software conocidas como paquetes de diseño difusos y que además de las características antes mencionadas permiten la implementación de un sistema difuso ya que como salida producen un código ensamblador (directo) para un microprocesador o microcontrolador de propósito general específico. Sin embargo en general estas herramientas necesitan que las demás partes que constituyen la implementación del controlador difuso tales como la adquisición y salida de datos, y el manejo de periféricos para el control de la aplicación se programen por separado y luego se enlace todos los bloques correspondientes para tener el sistema completo y funcional.

2.2.6.2 implementaciones de sistemas difusos.

La forma más simple y la más usual para implementar controladores difusos es realizarle como un programa de computadora sobre un procesador de propósito general (PC). Sin embargo, un gran número de aplicaciones de control difuso requieren, una operación en tiempo real para interconexión de dispositivos externos de alta velocidad. Por ejemplo el control de velocidad de automóviles, control de motores eléctricos y control de robots, están caracterizados por restricciones de velocidad severas. La implementación software de la lógica difusa no puede ser considerada como una solución de diseño conveniente para este tipo de aplicaciones. En tales casos, las especificaciones de diseño pueden ser sincronizadas por procesadores difusos especializados o microprocesadores o microcontroladores de propósito general.

Los requerimientos para la implementación hardware son:

- Desempeño de alta velocidad;
- Baja complejidad;
- Alta flexibilidad;

Estas condiciones se contradicen unas con otras. Así no es fácil seleccionar el camino correcto, especialmente si uno toma en cuanta otros factores, tales como costo de manufactura (muy importante para controladores difusos de productos de consumo) o costo de diseño (importante en investigación y desarrollo).

Baja complejidad significa que los algoritmos para el procesamiento difuso, la fuzzificación y la defuzzificación tienen que ser muy simples y demandar una pequeña cantidad de memoria tanto como sea posible para cada realización. La

flexibilidad significa la habilidad del hardware para ser usado exitosamente en diferentes aplicaciones y configuraciones.

Generalmente hablando, tres maneras diferentes de implementar el hardware del controlador difuso pueden ser propuestas. Estas son resumidas, juntas con sus ventajas y desventajas en la tabla 2.7.

Tabla 2.7 Tipos de implementación de un controlador difuso.

Clase de implementación Hardware	**Ventajas**	**Desventajas**
Procesador digital de propósito general	Flexibilidad en selección de herramientas hardware y software	Bajo desempeño a menos que uno de muy gran alcance sea usado
Procesador digital especializado	Incremento de desempeño	Incremento de complejidad y costo, ya que debe ser unido con un procesador estándar anfitrión. Carencia de flexibilidad, ya que puede ser aplicado a una clase limitada de problemas Más alto costo.
Procesador analógico	Alto desempeño Bajo costo Consumo de potencia bajo	Principalmente el asunto de investigación Baja Calidad Carencia de flexibilidad

Se puede ver que cualquier tipo de hardware tiene sus lados positivos y negativos para aplicaciones de controladores difusos, en general una selección

adecuada se tiene que hacer tomando en cuenta los requerimientos de la aplicación.

Sin embargo en cualquier caso antes de hacer la implementación de un controlador difuso, este debe ser diseñado previamente en algún paquete de diseño software. Una vez, que es hecho, la herramienta puede generar un programa (en 'C', en ensamblador, o código máquina para un procesador especializado) que represente el algoritmo de control lógico difuso. Este programa puede entonces ser descargado a un sistema basado en algún de los tres tipos de implementación para ejecutar la aplicación, conjuntamente con el programa encargado de las tares restantes del sistema (Adquisición de datos, salida de datos) de control difuso.

2.2.7.3 Integración de hardware de control convencional y difuso.

Debido a que el proceso de integración entre los distintos elementos que forman parte del sistema de control difuso puede a veces incrementar el tiempo de desarrollo de los mismos, también en el mercado han aparecido herramientas que integran hardware y software que permiten llevar acabo todas las etapas de diseño e implementación de un sistema de control basado en lógica difusa sin la necesidad de hardware ni software extra, lo que reduce el tiempo de desarrollo y puesta a punto de los sistemas.

Los desarrolladores hardware han llegado a proponer soluciones complejas, integrando software y hardware difuso independiente o con hardware de PLC (Controladores lógico programables) y DCS (sistemas de control distribuidos). El objetivo es proporcionar la oportunidad de completar el diseño e implementar un controlador para una aplicación particular sin la necesidad de software y hardware extra.

El prototipo propuesto entra dentro de esta categoría de herramientas.

CAPITULO 3. DISEÑO DEL SISTEMA INTELIGENTE DIFUSO.

EL SID (Sistema Inteligente Difuso) se pensó, para ser una herramienta alternativa en el diseño e implementación de un sistema de control difuso, para el control de procesos donde los actuadores o elementos finales de control funcionen con Corriente Continua (CC) y Corriente Alterna (CA), de tal manera que se logre un ahorro energético. Este proporcionara la oportunidad de completar toda la implementación del sistema de control difuso sin la necesidad de software o hardware extra ya que estará integrado de una programa software y una tarjeta electrónica.

3.1. Conceptualización del sistema.

En la Figura 3.1 se muestran los elementos del SID y su interacción con el proceso CC y CA controlado. La parte software la constituye la interface grafica de usuario para la edición , diseño, generación de código y monitorización del sistema de control difuso; la tarjeta electrónica constituye la implementación física del controlador difuso donde se descargara el código asociado a dicho controlador para ser ejecutado directamente para la aplicación particular que fue diseñado.

La utilización del SID consistirá primero en diseñar el controlador difuso para una aplicación particular desde la IPVSID(Interface de Programación y visualización del Sistema Inteligente Difuso) en base a sus requerimientos (síntesis del controlador), que corresponde a definir las entradas y la salida de controlador (variables lingüísticas), definir las funciones de membresía de las entradas, definir las funciones singletons para la salida (controlador tipo Sugeno) y finalmente definir el conjunto de reglas que representara la estrategia de control, después de lo cual debe ser implementado y probado para monitorear su desempeño (análisis) en el sistema de control difuso completo. La única manera de hacer esto en el prototipo propuesto es poner en ejecución el diseño efectuado. Para lograr esto, antes que nada se debe generar desde la IPVSID, el código asociado a la base de conocimientos del controlador, así como, el correspondiente al algoritmo de control difuso para procesarla; y enseguida descargarlo a la TEPDES (Tarjeta Electrónica de Procesamiento Difuso y de Entrada y Salida), donde es implementado el controlador difuso y después ejecutado. Ya en ejecución el sistema de control difuso es monitoreado, para visualizar en tiempo real el estado de las entradas y la salida, del controlador difuso, así como el de la salida del proceso controlado, con el fin de observar el desempeño del sistema de control entero (ver Figura 3.1) y tomar si es necesario las medidas correctivas correspondientes para poner a punto el controlador difuso, medidas que se deben llevar acabo suspendiendo la ejecución en curso, haciendo los ajustes necesarios a la definición y diseño, para luego ejecutarlo y probarlo una vez más y las veces que sean necesarias hasta poner a punto el controlador para la aplicación particular que fue diseñado.

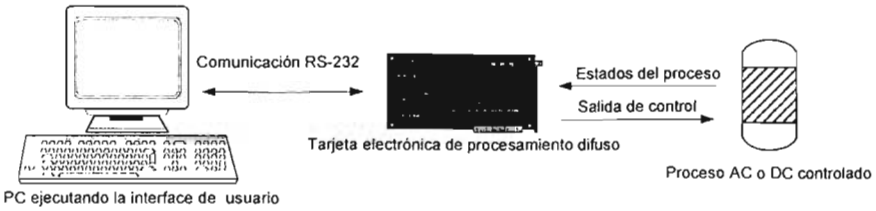


*Figura 3.1 Diagrama de bloques del **SID.***

La salida del controlador difuso del SID estará orientada al control de procesos basados en CC y CA, con esto específicamente se indica que dicha salida se usara para controlar o regular la potencia CC o CA que se entrega al actuador o elemento final de control de la máquina o proceso controlado, para llevarlo al estado deseado. Por lo tanto contara con una interface de potencia que permita manejar actuadores que basen su funcionamiento en CC y CA.

El diseño e implementación del prototipo de "Sistema inteligente difuso" radicara en diseñar las dos componentes del mismo de tal modo que permitan una interacción entre ellos dos y el proceso controlado.

3.2 Tarjeta Electrónica de Procesamiento Difuso.

La tarjeta electrónica de procesamiento constituye la parte hardware del SID y es la encargada de alojar a los distintos dispositivos electrónicos que lo constituyen, es decir la implementación física del mismo. En ella se lleva acabo el procesamiento difuso, además de ser la encargada de interactuar directamente a través de una interface de entrada y salida con el proceso controlado y de comunicarse vía RS232 con la PC donde se ejecuta la IPVSID, para su programación y configuración.

En la tarjeta electrónica se encuentran alojados:

a) El microcontrolador de procesamiento, que es el cerebro de SID, puesto que es el encargado de gestionar su operación general, llevar acabo el procesamiento difuso; el control de la adquisición y entrega de datos; el control de la comunicación serie vía RS232 con la PC donde corre el software de diseño y monitoreo del controlador difuso para un aplicación particular.

b) La fuente de alimentación que genera los voltajes de +12V, -12V, +5 V para suministro de los diversos dispositivos electrónicos que conforman la tarjeta.

c) El Módulo de comunicaciones que se encarga de acoplar la comunicación RS – 232 entre la tarjeta y la PC

d) Un complemento de la misma tarjeta donde se encuentra: el módulo de acondicionamiento de las entradas analógicas al SID que provienen de los emisores del proceso controlado y el módulo de potencia de salida para controlar los dispositivos o cargas de CC y CA que constituyen los

actuadores o elementos finales de control del la máquina o proceso controlado.

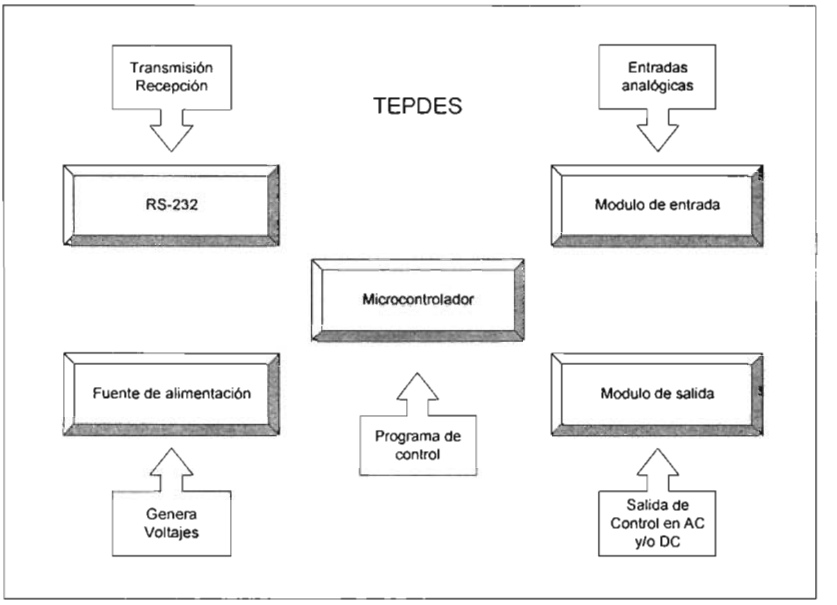


Figura 3.2 Elementos de la tarjeta electrónica de procesamiento.

Todos estos elementos en conjunción constituyen lo que se llamo "Tarjeta Electrónica de Procesamiento Difuso y de Entrada y Salida" (TEPDES).

La TEPDES interactúa con la IPVSID vía una comunicación serial RS232, para poder descargar el código generado de un diseño y luego poder adquirir los datos de información del estado de las entradas y salida del controlador difuso, así como, el de la salida del proceso controlado, además de permitir el envió de otros parámetros y comandos para cambiar el funcionamiento del sistema de control difuso en tiempo de ejecución.

La TEPDES interactuará con el proceso que se pretende controlar a través de los módulos de entrada y salida de la misma, de esta manera se cierra el lazo del sistema de control.

El diseño de la TEPDES comprende las siguientes tareas:

1. Selección del elemento microcontrolador.
2. Diseño del módulo de comunicaciones.
3. Diseño del módulo de entrada.
4. Diseño del módulo de salida para dispositivos de CC.
5. Diseño del módulo de salida para dispositivos de CA.
6. Diseño de la fuente de alimentación.

7. Diseño de los programas y rutinas de control del microcontrolador.

Debido que el diseño de los programas de control del microcontrolador es el tema mas extenso y complejo se tratara al final, además esto permitirá utilizar la información previa para facilitar la comprensión de los programas y rutinas diseñadas.

3.2.1 Especificaciones de diseño de la TEPDES.

En la Tabla 3.1 se resumen las especificaciones de diseño de cada uno de los módulos que componen la TEPDES.

Tabla 3.1 Especificaciones de diseño de la TEPDES.

• Control de la TEPDES a través de uso de un elemento microcontrolador para gestionar el funcionamiento general del sistema, procesar el algoritmo de control difuso y sincronizar la operación de cada uno de los demás elementos. En el residirán todos los programas, rutinas de control, memoria de funcionamiento y los periféricos adecuados de control.
• Módulo de comunicaciones para acoplar la comunicación serie RS232 entre la TEPDES y la PC donde se ejecutara la IPVSID.
• Módulo de tres entradas analógicas para permitir el acoplamiento entre los emisores de las variables de estado o salidas de proceso bajo control con el SID.
• Módulo de dos salida CA, para convertir los niveles de las señales de control producidas por el microcontrolador a los niveles de los dispositivos CA (127V, 15A y 60HZ) que funcionan como actuadores o elementos finales de control en general, de una máquina o proceso bajo control.
• Módulo de dos salida CC, para convertir los niveles de las señales de control producidas por el microcontrolador a los niveles de los dispositivos CC (12V, 400mA) que funcionan como actuadores o elementos finales de control en general, de una máquina o proceso bajo control.
• Fuente de alimentación con entrada de voltaje en CA de 127V, 60HZ y salida de voltaje regulado de CC de $\pm 12V$ y $+5V$ con una corriente de salida de 1 Ampere.

3.2.2 El módulo microcontrolador.

El microcontrolador integrado (embebido) en la TEPDES es el cerebro de SID, además de gestionar el funcionamiento global del mismo, en su momento debe de llevar a cabo el procesamiento difuso, controlar lo periféricos internos que

manejan los circuitos de potencia para cargas de CC y CA, realizar el procesamiento de las entrada analógicas (conversión analógica digital) y la gestión de las comunicaciones (comunicación serial vía RS232).

El microcontrolador elegido para soportar los requerimientos de la TEPDES fue el PIC16F877 debido a su rica y potente variedad de recursos (ver Tabla 3.2); además de que su sistema de desarrollo es mínimo (requiere poco hardware) y su disponibilidad (bajo costo) y soporte esta muy extendido.

Tabla 3.2 Principales recursos del PIC16F877.

• **Recursos fundamentales.**
• Procesador de arquitectura RISC avanzada.
• Juego de 35 instrucciones con 14 bits de longitud todas ellas se ejecutan en un ciclo de instrucción menos las de salto que tardan dos.
• Frecuencia de 20 MHz.
• Hasta 8 K palabras de 14 bits para la memoria de código, tipo FLASH.
• Hasta 368 bytes de memoria de datos RAM.
• Hasta 256 bytes de Memoria de datos EEPROM.
• Hasta 14 fuentes de interrupción internas y externas.
• Pila con 8 niveles.
• Modos de direccionamiento directo, indirecto y relativo.
• Perro Guardián (WDT).
• Código de protección programable.
• Modo SLEEP de bajo consumo.
• Programación serie en circuito con dos pines.
• Voltaje de alimentación comprendido entre 2 y 5.5 V.
• Bajo consumo (menos de 2mA a 5V y 5Mhz).

• **Dispositivos Periféricos.**
• Timer0: temporizador-contador de 8 bits con preescalador de 8 bits.
• Timer1: temporizador-contador de 16 bits con preescalador.
• Timer2: temporizador-contador de 8 bits con preescaldor y postdivisor.
• Dos módulos de captura /comparación/ PWM.
• Conversor A/D de 10 bits.
• Puerto Serie Síncrono con SPI e I2C.
• USART.
• Puerto Paralelo Esclavo (PSP).

De la tabla anterior puede verse claramente que el microcontrolador elegido cubre todas las necesidades de diseño de la TEPDES.

El sistema mínimo de desarrollo de cualquier aplicación basada en un microcontrolador PIC16F87X comprende los tres elementos siguientes.

I. Polarización del microcontrolador
II. Circuito de reset
III. Oscilador de trabajo.

Estos elementos son claramente identificados en el esquema eléctrico de la Figura 3.3. En dicho esquema también se observa un módulo LCD que ocupa como línea de datos los 8 pines el puerto D del PIC y como líneas de control los tres pines del puerto E, tiene también su respectiva polarización y circuito de control de contraste. Este módulo tiene la función de mostrar el estado actual del SID una vez que ha sido encendido y posteriormente cuando es utilizado.

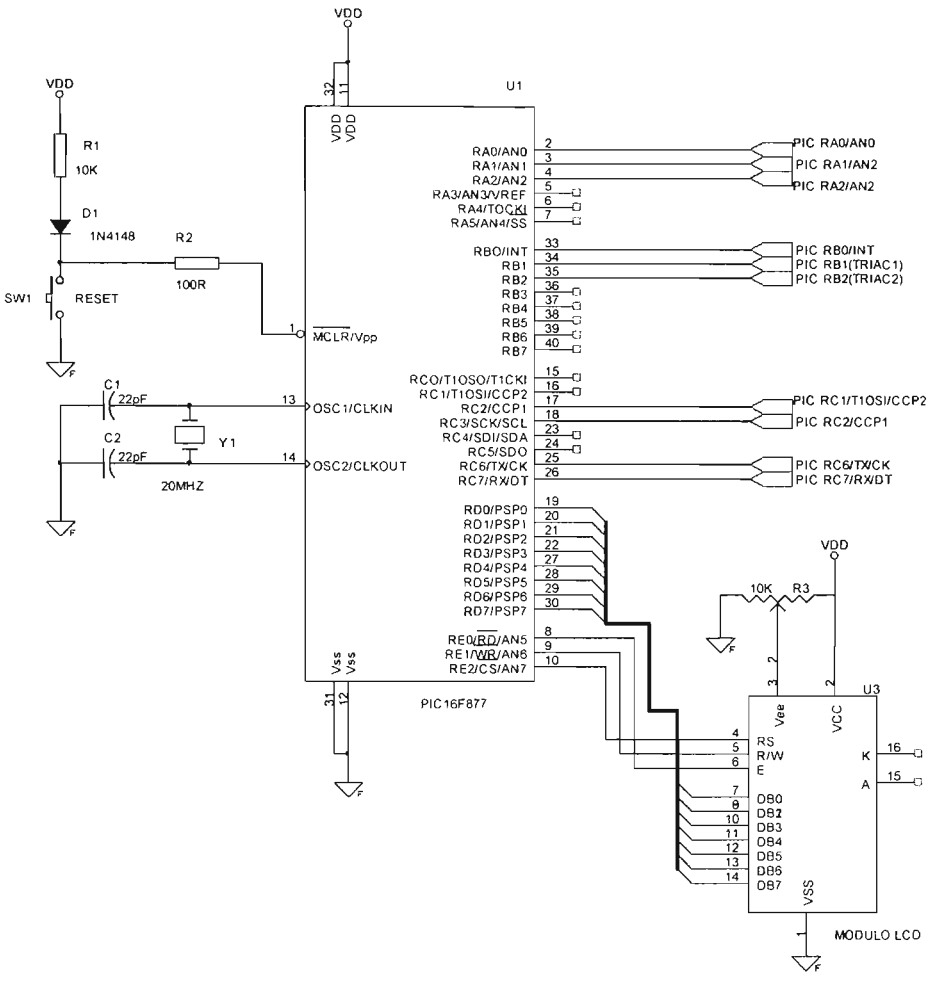


Figura 3.3. Diagrama eléctrico del PIC y el módulo LCD.

Finalmente se resaltan un conjunto de líneas de entrada y otras de salida que tiene funciones bien definidas dentro de la TEPDES, y que son:

I. RA0/AN0. Dentro de su comportamiento general puede funcionar como línea digital de E/S o como entrada analógica al Convertidor Analógico Digital (canal 0), en su utilización para el presente trabajo se configura para que funcione de la segunda manera, representando la primera entrada analógica del SID.

II. RA1/AN1. Igual que RA0/AN0, pero representando la segunda entrada.

III. RA2/AN2. puede ser línea digital de E/S, entrada analógica o entrada de voltaje negativo de referencia para el convertidor AD, sin embargo funciona Igual que RA0/AN0, representando la tercera entrada analógica.

IV. RB0/INT. Línea de E/S digital o entrada de petición de interrupción externa, se utiliza en su segundo modo de funcionamiento, en la detección de cruce por cero de la señal de CA, necesario para el control de potencia CA de salida del SID.

V. RB1. Línea de E/S digital. Se configura como salida para sacar el pulso PWM que controla el suministro de potencia de una carga CA (carga AC1).

VI. RB2. Igual que RB1, pero para una segunda carga (carga AC2)

VII. RC1/T1OSI/CCP2. linea digital de E/S o entrada al oscilador de entrada del Timer 1 o entrada al módulo de captura2/salida comparacion2/salida PWM2, en este caso se utiliza como salida PWM2 para controlar la potencia CC que se entrega a una carga (carga DC2).

VIII. RC2/CCP1. Se utiliza Igual que RC1, sin embargo, como la salida PWM1 (carga DC1).

IX. RC6/TX/CK. E/S digital o como pin del transmisor del USART asíncrono o como reloj del síncrono. Para el presente trabajo se utiliza en su primer modo de funcionamiento.

X. RC7/RX/DT. E/S digital o receptor del USART asíncrono o como datos en el síncrono. Para el presente trabajo se utiliza en su primer modo de funcionamiento.

El estado de estos pines es controlado por las rutinas y programas de control correspondientes en el microcontrolador, además de estar también interconectadas con los módulos propios, que hace uso de ellas. Conforme se avance en la explicación del presente trabajo se comprenderá más detalladamente la operación de cada una de ellas.

3.2.3 Diseño del módulo de comunicaciones.

La comunicación entre la TEPDES y mas específicamente entre el microcontrolador y la IPVSID que corre en una PC se lleva acabo por medio de una comunicación serie utilizando el bus RS-232 de la PC (COM1 o COM2).

Para poder llevar acabo la comunicación entre el PIC y la PC por medio del protocolo RS-232, fue necesario implementar un interface que permitiera convertir los niveles TTL del PIC a los niveles del bus RS-232, y viceversa. Esta interface se baso en un integrado comercial MAX232, que permite dichas conversiones. El diagrama eléctrico de esta interface y que constituye el módulo de comunicación de la TEPDES es mostrado en la Figura 3.4.

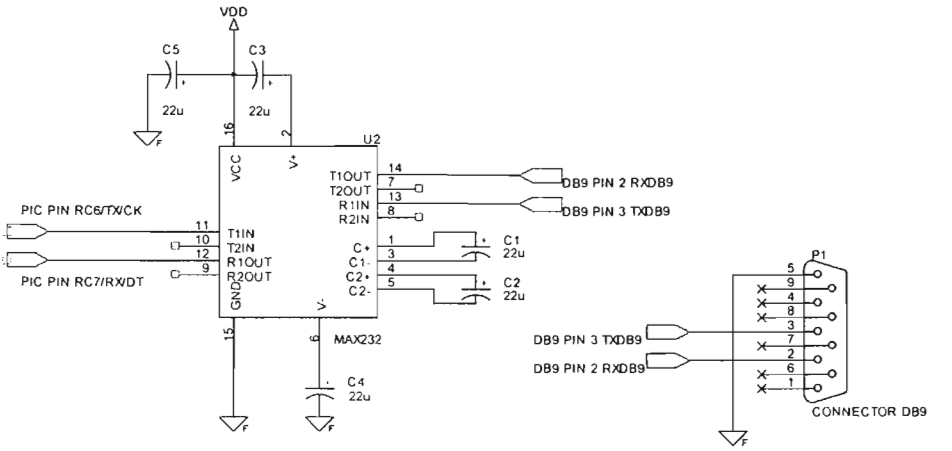


Figura 3.4 Diagrama eléctrico del módulo de comunicaciones de la TEPDES.

El circuito integrado lleva internamente dos conversores de nivel de TTL a RS – 232 y otros 2 de RS – 232 a TTL con lo que en total podremos manejar cuatro señales del puerto serie del PC, por lo general las mas usadas son; TX, RX, RTS, CTS, estas dos últimas son las usadas para el protocolo handshaking pero no es imprescindible su uso, de hecho en el presente trabajo no se utilizan la línea RTS ni la línea CTS. Para que el MAX232 funcione correctamente deben poner unos condensadores externos, como se puede ver en la Figura 3.4. También es observado que solo fueron utilizados dos conversores, el primero es un conversor de TTL (señal proveniente del pin RC6/TX/CK) a RS–232 para poder trasmitir del PIC a la PC y el segundo un conversor de RS–232 a TTL (señal introducida por el pin RC7/RX/DT del PIC) para poder transmitir de la PC al PIC.

Finalmente la interconexión entre la TEPDES y la PC se hace por medio de un cable serie con los conectores DB9 apropiados en cada uno de sus extremos.

Las rutinas de control del módulo de comunicaciones (control de transmisión y recepción asíncronas del USART del PIC) se explicadas mas adelante en este capitulo.

3.2.4 Diseño del módulo entrada.

Los módulos de estrada y salida constituyen la interface entre los emisores de señal y los actuadores de la máquina o proceso que se va a controlar.

El módulo de entradas analógicas (MEA) de la TEPDES constituye la interface entre las variables de estado o salidas de proceso bajo control con el SID, tiene como finalidad acondicionar las señales analógicas provenientes de los sensores de las variables estados o salidas del proceso controlado antes de ser introducidas en los respectivos canales del Conversor Analógico Digital (CA/D) , para su respectiva digitalización y posterior procesamiento, por el algoritmo de control difuso, que producirá las respectivas salidas de control para llevar al proceso controlado al estado deseado. Este acondicionamiento implica ajustar el

voltaje de salida de los sensores en el rango de 0 a 5 V (ya que estos límites corresponden a la referencia configurada para el módulo AD del PIC) y el filtrado correspondiente.

De acuerdo a las especificaciones de diseño el MEA se constituyó de tres conectores que permiten la conexión de tres entradas analógicas. En la Figura 3.5 se muestra solo el esquema eléctrico del acondicionador de la primera entrada analógica (canal 0 del conversor A/D), puesto que los acondicionadores restantes son exactamente iguales (por lo que solo se explica el diseño de uno).

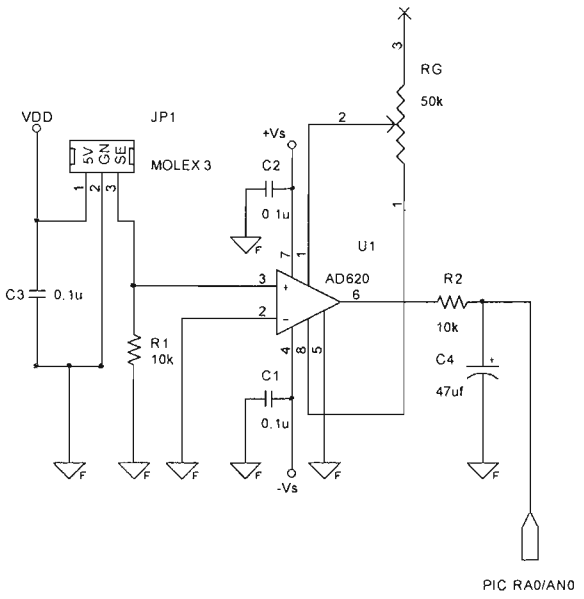


Figura 3.5 Diagrama eléctrico de una entrada del módulo de acondicionamiento de entradas analógicas de la TEPDES.

El circuito de acondicionamiento esta basado en el amplificador de instrumentación AD620, el cual fue seleccionado en base a las especificaciones de diseño de la Tabla 3.1 junto con su bajo costo. Sus características más destacables son ordenadas en la Tabla 3.3:

Tabla 3.3 Características del amplificador de instrumentación AD620.

• Una ganancia configurable por medio de un resistor externo,
• Amplio rango de suministro de alimentación (+/- 2.3 V a +/- 18V)
• Baja potencia, 1.3 mA de máxima corriente de alimentación.
• Excelente desempeño CC.

• Bajo ruido
• Excelentes especificaciones CA.

La entrada + del AD620 es la señal de salida del sensor la cual se conecta paralelamente con una resistencia de 10 k para asegurar que en caso de la ausencia de la señal del sensor esta entrada este a tierra. La entrada – se ha conectado a tierra, así como también la patita de referencia, para tener una salida referenciada a tierra. Los voltajes suministrados a +Vs y -Vs son +12 y -12 v respectivamente, los cuales se conectados paralelamente con un capacitor de 0.1 uF para mejorar la estabilización de la fuente de alimentación.

La salida del AD620 es alimentada a un canal analógico del conversor AD (para este caso especifico al pin RA0/AN0 del PIC) a través de un filtro paso bajas, con el propósito de reducir el ruido de la señal sensada.

Los pines 8 y 1 del AD620 son conectados a un potenciómetro de precisión que funciona como un reóstato para controlar la ganancia del amplificador, la cual esta dada por:

$$G = \frac{49.4k\Omega}{R_G + 1} \quad \text{..... (3.1) con lo que,} \quad R_G = \frac{49.4k\Omega}{G - 1} \quad \text{..... (3.2)}$$

El control de esta ganancia se hace de tal modo que la salida del sensor se ajuste en el rango de 0 a 5V.

Finalmente se puede observar un conector molex de tres pines para conectar el sensor de un estado del proceso controlado. Estos pines sirven respectivamente para proporcionar una polarizaron de 5 y 0 V para el sensor y suministrar la salida del sensor al módulo de acondicionamiento.

El proceso de conversión analógica digital es realizado por el módulo Conversor analógico digital de 10 bits incluido en el PIC, y cuyas rutinas de control son explicadas mas adelante en el apartado correspondiente de este mismo capitulo.

3.2.5 Diseño del módulo de salida para dispositivos de Corriente Continua.

El módulo de salida para dispositivos o cargas de CC (MSCC) constituye una interface que tiene como finalidad convertir los niveles de la señal de control producida por el CPU (microcontrolador) donde se ejecuta el algoritmo de control difuso, a los niveles de los dispositivos CC que funcionan como actuadores o elementos finales de control en general, de una máquina o proceso bajo control

El control del suministro de potencia a los dispositivos de CC se controla en base al tren de pulsos PWM (Pulse Width Modulation, Modulación de Ancho de Pulso) de nivel digital (5V) proveniente de un pin de E/S del CPU, con lo que se evita el uso convertidores digital analógicos (D/A).

El módulo esta compuesto por dos circuitos de control de suministro de potencia a cargas o dispositivos CC (debido a que ambos son iguales, solo basta explicar uno).

En la Figura 3.6 se muestra los circuitos de diseño considerado para las dos salidas este módulo. El control de potencia se consigue alimentado un tren de pulsos PWM de suficiente frecuencia a la base del transistor de potencia a través de una resistencia que le asegure un funcionamiento alternativo en corte y saturación. Variando el tiempo en que el porcentaje del pulso esta en alto (ciclo de trabajo), se consigue controlar el porcentaje de potencia que se entrega a la carga de CC.

El pulso PWM es alimentado de un PIN del PIC que ha sido configurado para funcionar como la salida PWM1 de módulo CCP1 (módulo de Captura 1, Comparación 1 y PWM 1), el cual es inicializado y configurado por programa para funcionar en la modalidad PWM, después de lo cual otra rutina se encargara de controlar el ciclo de trabajo de la salida PWM en base a la salida de control de la inferencia difusa. Estas rutinas de control serán explicadas en el apartado de diseño de los programas y rutinas de control del microcontrolador.

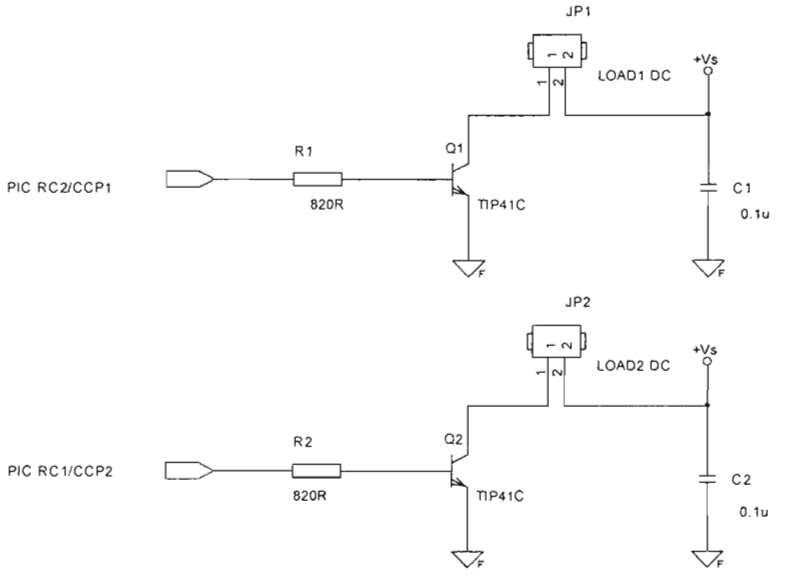


Figura 3.6 Diagrama eléctrico de los módulos de salida para cargas de CC

El transistor utilizado fue el TIP41, tomado en cuenta las especificaciones de diseño de la Tabla 3.1. Sus características eléctricas más relevantes son presentadas en la Tabla 3.4:

Tabla 3.4 Características eléctricas del TIP41.

Parámetros	Condiciones de Prueba	MIN	TÍPICO	MAX	UNIDAD
$V_{(BR)CEO}$ voltaje de ruptura colector emisor	I_C = 30 mA I_B = 0	100	120		V
h_{FE} tasa de transferencia de corriente directa	V_{CE} = 4V I_C = 300mA V_{CE} = 4V I_C = 3A	30 15	100 60		
$V_{CE(sat)}$ voltaje colector emisor de saturación	I_B = 600mA I_C = 6A		1.2	1.5	V
V_{be} voltaje base emisor	Vce = 4V I_C = 6A		1	2	V
h_{fe} tasa de transferencia de corriente directa en señal pequeña	V_{CE} = 10V I_C = 500mA	20			

La máxima corriente de saturación que puede ser entregada a la carga de CC esta dada por:

$$I_C = \beta \times I_B$$

De acuerdo a las especificaciones para una corriente máxima de 400mA, con una β promedio de 80

$I_B = \frac{400mA}{80} = 5mA$, con lo que $R1 = \frac{VpinPic - V_{BE}}{I_B} = \frac{5V - 0.7V}{5mA} = 860\Omega$,

seleccionando un valor comercial de 820Ω se tiene que,

$$I_B = \frac{VpinPic - V_{BE}}{R1} = \frac{5V - 0.7V}{820} = 5.2439mA$$

con una $I_C = \beta \times I_B = 80 \times 5.2439mA = 419.5122mA$.

El cual es un valor aceptable de corriente, ya que ofrece más de los requerimientos.

Ya que el MSCC cuenta con dos salidas para cargas de CC, podría pensarse que la salida del controlador difuso del SID tiene más de una salida de control, que en realidad solo es una, y se han diseñado dos salidas de CC debido a que esta única salida de control puede ser usada para controlar alternativamente dos cargas (actuadores) con la restricción de que ambas actúen alternadamente sobre el mismo proceso controlado, es decir controlando la misma variable, pero en sentidos opuestos (esta mismas observaciones son aplicadas al módulo de salida para dispositivos CA). Estas cuestiones serán aclaradas cuando se expliqué las rutinas de control (ver apartado 3.3.7.5.3) que producen las salidas del controlador difuso.

3.2.6 Diseño del módulo de salida para dispositivos de Corriente Alterna.

El módulo de salida para dispositivos o cargas de CA (MSAC) es una interface que tiene como finalidad convertir los niveles de la señal de control producida por el CPU (microcontrolador) donde se ejecuta el algoritmo de control difuso, a los niveles de los dispositivos CA que funcionan como actuadores o elementos finales de control en general, de una máquina o proceso bajo control

El módulo esta compuesto por dos circuitos que controlan el suministro de potencia de cargas o dispositivos de CA (debido a que ambos son iguales, solo se explica uno).

El circuito de control de suministro de potencia se baso en el uso de un opto acoplador y un triac. El opto acoplador tiene la finalidad de aislar eléctricamente el circuito de potencia CA del microcontrolador. Y variando el ángulo de disparo del triac por medio de un tren de pulsos PWM sincronizado con la señal de CA se consigue controlar el suministro de potencia de las cargas de CA. De acuerdo a las consideraciones anteriores y las especificaciones de diseño los circuitos finales del MSAC son los mostrados en la Figura 3.7.

Como se puede ver en la Figura 3.7 el pulso PWM para controlar la potencia de la primera carga es suministrado desde el pin RB1 del PIC al MOC3020 que es un dispositivo que funciona como interface entre el sistema electrónico digital de control (microcontrolador) y el triac de potencia con el objetivo de aislar eléctricamente ambos sistemas. La salida del MOC3020 manejada por el pulso PWM, controla directamente el ángulo de disparo del TRIAC MAC223, con lo que se consigue regular la potencia entregada a las cargas de CA que operan a 127 V y 60 Hz. El circuito puede soportar cargas resistivas como calefactores y cargas inductivas como motores.

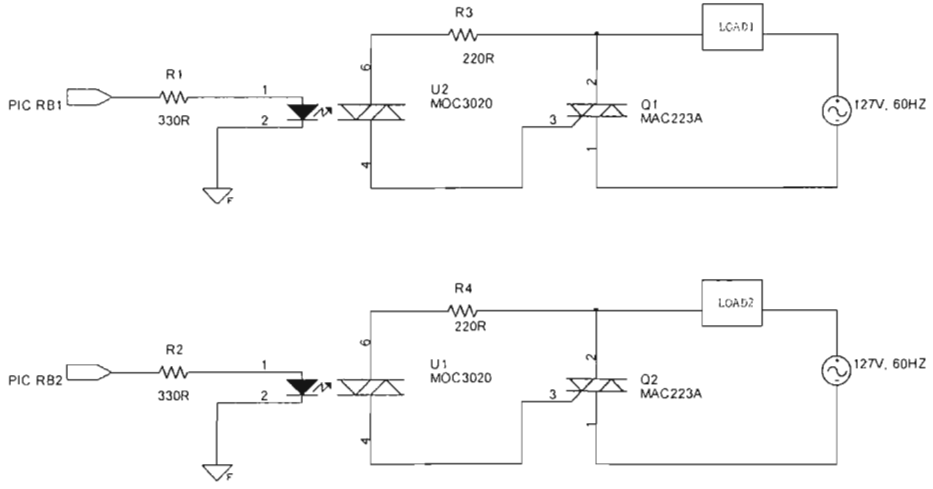


Figura 3.7 Diagrama eléctrico del módulo de potencia para cargas CA de la TEPDES.

Para poder controlar la potencia de las cargas es necesario que el pulso PWM este sincronizado con la señal de CA. Esto se consigue por medio de un circuito detector de cruce por cero de la señal de CA (cuando la onda cambia de valores positivos a negativos y viceversa), el cual produce un pulso de salida cada medio periodo de la señal de CA. La detección de este pulso se utiliza para sincronizar el periodo de la señal PWM (la frecuencia del pulso PWM resulta el doble de la frecuencia de la señal de CA) al medio periodo de la señal de CA, de tal manera que controle el ángulo de disparo para cada medio, es decir tanto para valores positivos como negativos (ver Figura 3.9).

El circuito detector de cruce por cero diseñado es el mostrado en la Figura 3.8 , donde se puede observar que esta basado en un rectificador de onda completa cuya salida es alimentada a la base de un transistor, el cual permanece en estado de saturación con un voltaje de colector a emisor muy cercano cero, mientras la señal de CA no cruce por cero, momento en el cual el transistor pasa a su estado de corte, con lo que el voltaje de colector a emisor pasa a tener un valor de VDD (pulso indicador), el cual cae una vez mas que la señal de alterna comienza a crecer. Esta operación se repite cada medio periodo de la señal de CA. La salida del circuito detector de cruce por cero es mostrada en la Figura 3.9.

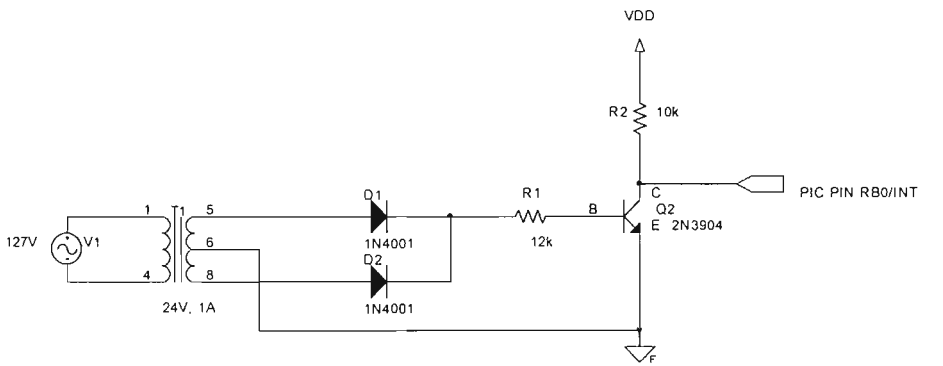


Figura 3.8 Diagrama eléctrico del circuito detector de cruce por cero de la señal de CA de 127V y 60HZ, para el control de potencia de las cargas CA del SID.

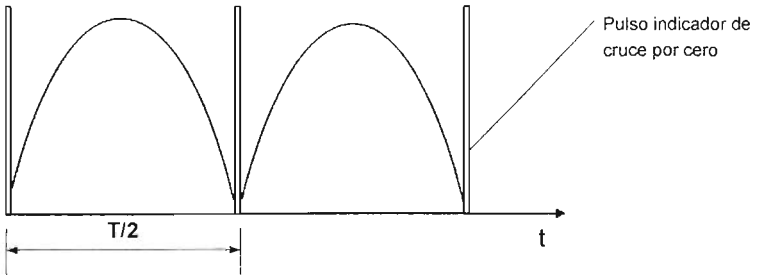


Figura 3.9 Salida del circuito detector de cruce por cero sobrepuesto sobre la salida del rectificador de onda completa.

El pulso indicador de cruce por cero es alimentado a la patita RB0/INT del PIC, para producir una interrupción, donde una rutina de atención a interrupción adecuada se encargara de realizar las acciones necesarias para sincronizar la señal de CA y el pulso PWM. Esta rutina y las de control del pulso PWM son presentadas mas adelante en el apartado correspondiente de este mismo capitulo.

3.2.7 Diseño de la fuente de alimentación.

La fuente de alimentación de la TEPDES es la encargada de producir los voltajes de alimentación de los diferentes dispositivos electrónicos que conforman los módulos de la misma.

De acuerdo a las especificaciones la fuente que se diseño es la mostrada en la Figura 3.10. Esta produce el voltaje regulado de +5V para alimentar a todos lo dispositivos digitales de la TEPDES y los voltajes regulados de +12V y -12V necesarios para alimentar los amplificadores de instrumentación utilizados en el módulo de entradas analógicas (MEA), el voltaje de +12V también se utiliza como voltaje de alimentación de las cargas de CC (este voltaje puede ser cualquier otro externo sin embargo para el presente trabajo así se definió, por lo que también quedo limitado el consumo de corriente de la cargas de CC).

Las características eléctricas de la fuente de alimentación son las de la Tabla 4.5:

Tabla 4.5 Características eléctricas de la fuente de alimentación.

• Corriente máxima de 1 A.
• Entrada de potencia de 127V y 60HZ.
• Salida de voltajes regulados de $+5V$ y $\pm 12V$.

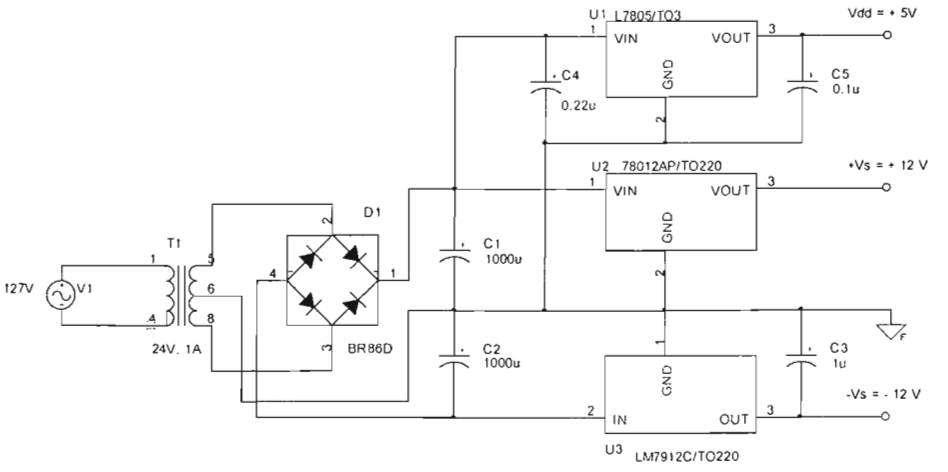


Figura 3.10 Diagrama eléctrico de la fuente de alimentación de la TEPDES del SID.

Con la explicación de la fuente de alimentación de la TEPDES se concluye con todo lo relativo a la parte hardware del SID, por lo tanto el siguiente apartado principal da comienzo a la explicación del diseño de los programas y rutinas de control del microcontrolador que se encargan de, controlar a los elementos hardware atrás mencionados, del control global del funcionamiento del SID y del procesamiento del algoritmo de control difuso.

3.3 Diseño de los Programas y rutinas de control del microcontrolador y del sistema de control difuso.

3.3.1 Programas del control.

El SID esta pensado para ser una herramienta para el diseño, implementación y depuración de un controlador difuso, en otras palabras una herramienta que auxilie a la gente interesada en el diseño e implementación de un sistema de control lógico difuso en un controlador embebido.

La tarea de automatización esta basada en el diseño del controlador difuso, que implica la definición del tipo de entradas y el tipo de salida del controlador difuso; y la definición de la base de conocimiento, que involucra la definición de las funciones de membresía de cada entrada, la definición de las funciones singleton de la salida y finalmente la definición del conjunto de reglas de control. Desde la IPVSD se llevan acabo los pasos anteriores de diseño, conjuntamente con la generación del código que representara la base de conocimientos (datos) y el código de operación (programa) que representara la máquina de inferencia que se encargara de procesar la base de conocimiento, que conjuntamente formaran la rutina de control difuso.

De lo anterior se desprende, que el programa del microcontrolador debe contar con una parte fija de programa para gestionar la operación general del SID, la de los periféricos auxiliares y de control; y otra parte programable para alojar los datos que representan la base de conocimiento y para escribir el código de operación del sistema de control difuso, de una aplicación particular.

En base a las consideraciones anteriores y otras que se consideran mas adelante, el programa y rutinas del microcontrolador se dividieron en tres grupos.

a) Programa de Sistema (PS) y rutinas de sistema. Comprende el conjunto de programas y rutinas que se encargan de las funciones operativas del SID. Entre sus tareas principales están controlar la escritura-lectura del programa de aplicación (controlador difuso diseñado) y la ejecución del mismo; encargase de la configuración general de las comunicaciones, demás inicializaciones y configuraciones necesarias; en otras palabras de la administración general.

b) Rutinas y Programa de Control Difuso (PCD). Comprende el conjunto de rutinas y programas que representan el sistema de control difuso, es decir aquellas que controlan y procesan su ejecución.

c) Programa de datos. Comprende el conjunto de localidades de memoria de datos y de programa reservadas para almacenar la definición de la base de

conocimiento del controlador difuso así como ciertas constantes necesarias para el correcto funcionamiento del sistema mismo.

Cada uno de estos tipos de programas y rutinas tiene localidades reservadas respectivamente, en la memoria de programa del PIC, las cuales son observadas en la Figura 3.11.

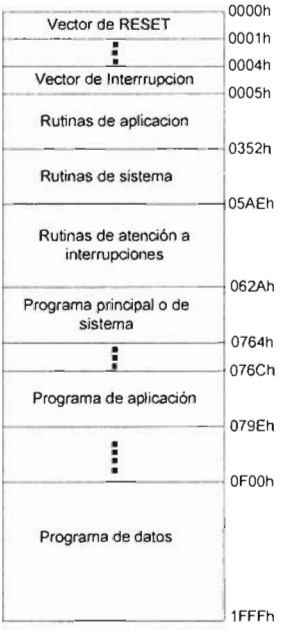


Figura 3.11 Mapa de memoria del microcontrolador.

3.3.2 Programa principal o de sistema.

EL programa principal o de sistema (PS) controla las funciones operativas del SID en base a los comandos que le son enviados desde la IPVSID, una de las primeras funciones necesarias del PS es configurar e inicializar la comunicación serie vía RS-232 con la PC donde se ejecuta la IPVSID (esto, para poder establecer el dialogo entre ambas partes) e inicializar el módulo LCD que funcionara como indicador del estado operativo del PS desde la propia TEPDES.

Los tres posibles estados operativos en los que puede estar el PS son tres a saber:

- El **Estado de Espera General de Modo de Operación**(EEGMO)
- El **Modo de Operación de Programación** (MOP).
- Y el **Modo de Operación de Ejecución** (MOE).

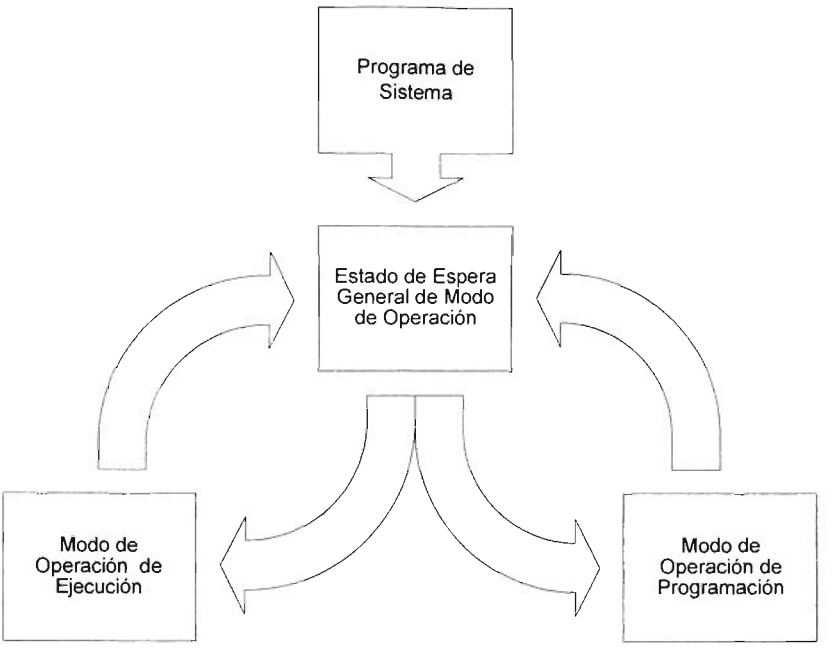


Figura 4.12 Diagrama de bloques los modos de operación del programa de sistema y su interacción.

En el diagrama de bloques de la Figura 4.12 se puede observar la estructura general del PS y la manera en como interactúan los tres estados operativos en los que puede encontrarse.

Estado de Espera General de Modo de Operación.

En este estado el PS inicialmente informa sobre la operatividad del sistema tanto a la IPVSID como en el monitor de estado (LCD) del la TEPDES, después de lo cual entra a un ciclo de espera del cual saldrá hasta que se reciba un comando que le indique el modo de operación al que debe pasar.

Modo de Operación de Programación.

El PS entra en este modo de operación una vez que recibe el correspondiente comando en el EEGMO. Este modo se caracteriza por permitir las operaciones de escritura o descarga del código generado a la TEPDES para un controlador difuso diseñado en la IPVSID y la lectura del mismo para verificar que ha sido correctamente escrito. Al igual que el EEGMO al inicio de este modo de operación se manda un mensaje para informar tanto a la IPVSID como en el monitor de estado (LCD) del la TEPDES que el PS ha entrado al MOP. Este modo se abortara si en lugar de recibir el correspondiente comando de escritura o lectura se recibe el comando de fin de MOP, con lo que el control de programa regresara una vez mas al EEGMO (ver Figura 3.13).

Modo de Operación de Ejecución.

Al igual que en el MOP el PS entra en este estado de operación cuando es recibido el correspondiente comando en el EEGMO. La función principal de este modo de operación es llamar al PCD, donde se ejecutara el algoritmo de control difuso, descargado de la IPVSID, donde antes de ser llamado se configuran e inicializan tanto periféricos como registros de control necesarios para asegurar su funcionamiento correcto. AL igual que en los dos modos de operación anteriores al inicio de este modo de operación se manda un mensaje tanto a la IPVSID como al monitor de estado (LCD) del la TEPDES con el objetivo de informar que el PS ha entrado al MOE. Este modo se abortara si durante la ejecución del PCD, se recibe el comando de fin de MOE, con lo que el control de programa regresara una vez mas al EEGMO (ver Figura 3.13)

El diagrama de flujo que especifica el comportamiento general del PS diseñado es mostrado en la Figura 3.13.

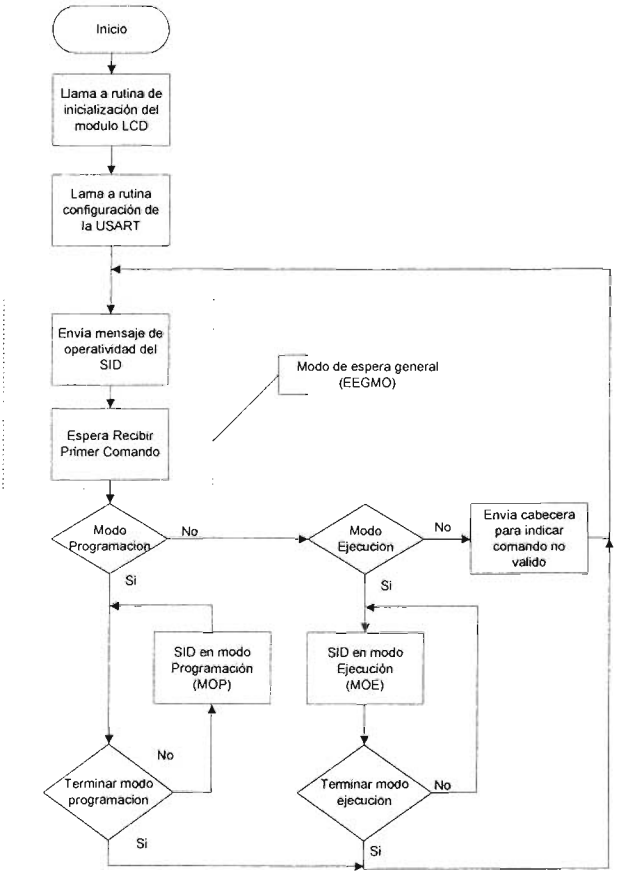


Figura 3.13 Diagrama de flujo general de los modos de operación de PS.

3.3.3 Diseño del Control de lo modos de operación del Programa de Sistema

De acuerdo al diagrama de flujo de la Figura 3.13, la primera acción que se realiza al comenzar la ejecución del programa de sistema es llamar a la rutina de inicialización de la LCD que funciona como monitor de estado del PS en la TEPDES.

Esta rutina se llamo INILCD y realiza las siguientes acciones:

- Configura un bus de datos de 8 bits, dos líneas en pantalla, y una matriz de carácter de 5 por 7 puntos;
- Enciende la pantalla, apaga el cursor junto con su intermitencia;
- Define la entrada de caracteres con incremento del cursor, sin desplazamiento de texto;
- Borra LCD y apunta al inicio del primer renglón:
- Además configura el puerto D como salida para que actué como línea de datos de la LCD y los tres bits de menos peso del puerto E como líneas de control de la LCD es decir R/S, R/W y Enable.

Ejecutada la última rutina el Moduló LCD esta listo para recibir instrucciones y datos para visualizar mensajes en su pantalla.

En seguida de haber ejecutado la última rutina se llama a la rutina de configuración del USART.

El PIC cuenta con un módulo USART el cual puede ser configurado y utilizado para establecer una comunicación serie asíncrona por medio del bus RS-232 entre una PC y el PIC.

La rutina de configuración del USART se llamó CONFIGURAR_USART y realiza las siguientes acciones.

- Inicializa el puerto C, pone el pin RC6/TX como salida (pin de transmisión de la USART) y el pin RC7/RX (pin de recepción de la USART) como entrada y todos los demás pines como salidas (ver apartado 3.2.3).
- Configura y habilita el USART para una comunicación serie asíncrona de alta velocidad a 57, 600 baudios;
- Finalmente habilita el módulo de transmisión y recepción de la misma.

La velocidad de 57, 600 baudios fue seleccionada porque además de ser estándar, hace que el tiempo de transmisión de datos sea menor, lo cual es muy importante, debido a que una vez que se han actualizado y almacenado en el buffer de transmisión todas las variables del sistema de control, al final de este son transmitidas, lo cual toma cierto tiempo dependiendo de esta velocidad, situación que repercute en el tiempo mínimo de ejecución del algoritmo de control difuso y por lo tanto en la elección de la frecuencia de muestreo máxima(ver apartado 3.9.4.6), que como consecuencia limita la aplicabilidad del SID. Además

durante la descarga de código a la TEPDES esta velocidad aun permite que la escritura en memoria EEPROM, que es lenta, sea efectuada con éxito antes de recibir un nuevo dato.

Una vez ejecutada esta última rutina el microcontrolador esta en condiciones de realizar una comunicación serie asíncrona FULL DUPLEX vía el bus RS-232 (COM1 o COM2) de con una PC, auxiliado por la interface de conversión de los niveles digitales de microcontrolador a los niveles del bus RS-223 y viceversa, que fue explicada en el apartado 3.2.3.

Nota: Para mayor información sobre el código de todas las rutinas y programas del microcontrolador, consultar el apéndice A y para mayor información acerca del funcionamiento de los módulos periféricos consultar las referencias indicadas en la mesografía.

3.3.4 Modo de Espera General de Modo de Operación.

En seguida de haber ejecutado las dos últimas rutinas el PS entra al que abreviaremos *Modo de Espera General (MEG)* (en lugar de Modo de Espera General de Modo de Operación), el diagrama de flujo detallado de este modo de operación es mostrado en la Figura 3.14.

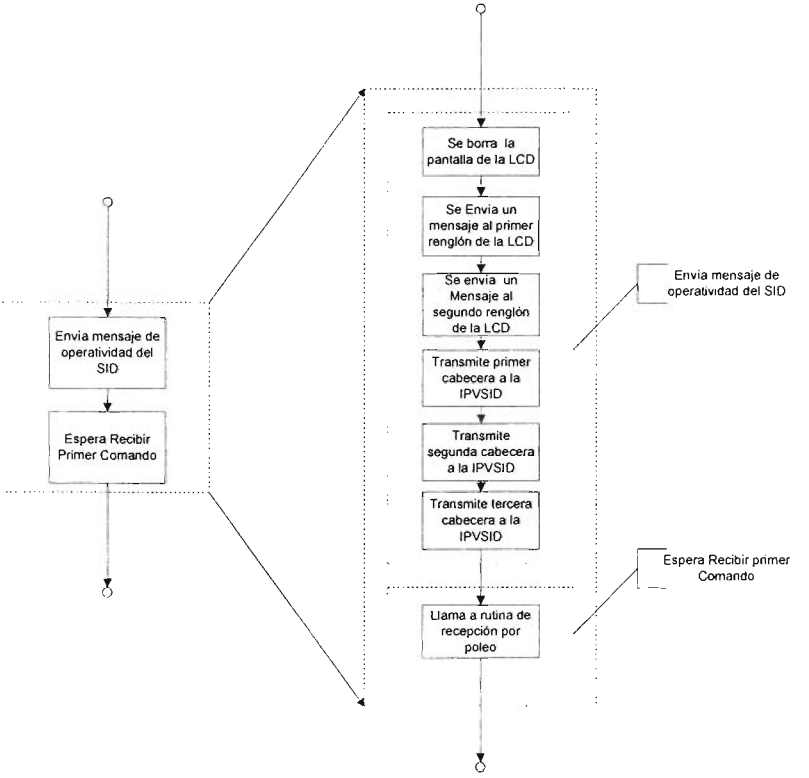


Figura 3.14 Diagrama de flujo detallado del Modo de Espera General de PS.

La primera tarea que se realiza al entrar al modo ejecución es enviar el mensaje de operatividad del SID tanto al monitor de estado de la TEPDES como a la IPVSID.

Para enviar mensajes al módulo LCD el PS se auxilia de una rutina y una macro previamente definidos.

La rutina se llamo BORRAR_LCD, que lo que hace, es mandar un comando a la LCD que borra todo el contenido anterior de la pantalla de la LCD y direcciona el cursor en la primera posición del primer renglón.

La macro se llamo MENSAJE_LCD3 NUMTABLA, NUMRENGLON; que como se puede ver a esta macro es necesario pasarle dos argumentos, el primero de ellos NUMTLABA, define la tabla que representa el mensaje que se va a escribir en la LCD y el segundo NUMRENGLON, define el renglón de la LCD donde se va a desplegar el mensaje.

Esta macro es utilizada dos veces la primera para desplegar en el primer renglón el mensaje "SID OPERATIVO" y la segunda para desplegar en el segundo renglón el mensaje "ENVIAR COMANDO", que indica que el SID esta en el MEG y listo para recibir un comando que le haga pasar al MOP o al MOE, dependiendo de la elección del usuario.

Después de enviar los mensajes al monitor de estado de la TEPDES, se transmiten en orden tres cabeceras a la IPVSID cuyos valores son 20h, 21h y 22h; esto para notificar que se ha establecido la comunicación entre ambas partes, que el SID esta operativo y que el usuario ya puede según lo que desee, enviar una instrucción que haga pasar al SID al MOP o al MOE.

Para transmitir las cabeceras el PS primero carga el dato que se va a transmitir en una localidad de memoria RAM que se llamo DATOX y después se auxilia de la llamada a una rutina que se nombro TRANSMITE, la cual transmite el dato (a estas dos operaciones conjuntas para transmitir un dato se les llamo método de transmisión por poleo).

El diagrama de flujo de la rutina TRANSMITE es mostrado en la Figura 3.15

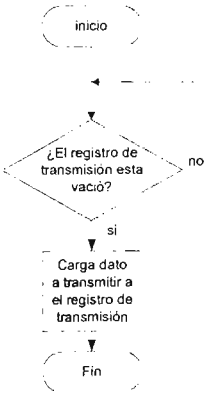


Figura 3.15 Diagrama de flujo de la rutina de transmisión por poleo "TRANSMITE".

La operación general de esta rutina consiste en verificar continuamente (poleo), si el registró de datos del módulo de transmisión (TXREG) del USART del PIC esta vació, de no ser así continua verificando hasta que se vacía, después de lo cual copia el dato cargado en DATOTX al registro de datos del módulo de transmisión, el cual se copia automáticamente al registro de corrimiento (TSR) del mismo módulo para ser transmitido.

La ejecución de esta rutina toma cierto tiempo, ya el poleo continua indefinidamente mientras no se haya terminado de enviar el ultimo dato colocado en TXREG, la duración de esta espera depende de la velocidad a la que se halla configurado el USART. Con esta acción se evita la pérdida de datos que pudiera suceder cuando se transmiten varios datos sucesivamente, es decir no se procede a enviar un nuevo dato mientras no se completa la transmisión del dato previo.

La última tarea que es realizada en el MEG es la espera (recepción) del comando respectivo que le indique al PS el nuevo estado al que debe pasar (MOP o MOE). El PS permanece en este estado de espera hasta que se recibe el comando. Para realizar la recepción de un dato el PS se auxilio de la llamada a la rutina RECIBE.

EL diagrama de flujo de la rutina RECIBE es mostrado en la Figura 3.16

Figura 3.16 Diagrama de flujo de la rutina de recepción por poleo "RECIBE".

Esta rutina checa continuamente (poleo) si se ha completado la recepción del ultimo dato enviado al módulo de recepción de la USART del PIC, de no ser así continua con esta verificación hasta que se completa la recepción, después de lo cual se copia el contenido del buffer de recepción RCREG a una localidad de memoria RAM que se llamo DATORX, de donde el dato recibido puede ser tomado para ser procesado.

Esta rutina permanece en el ciclo de verificación mientras no llegue y se complete la recepción de un dato, este hecho es aprovechado en el MEG para mantener al PS detenido en este punto hasta que no se envié un comando desde la IPVSID.

Mirando otra vez al diagrama de flujo de la Figura 3.14, una vez que es recibido un comando, se termina la ejecución del modo de espera general, y enseguida el comando recibido es comparado con dos constantes que definen respectivamente

el comando de Modo Programación (abreviatura de Modo de Operación de Programación) con un valor de 01h y el comando de Modo Ejecución (abreviatura de Modo de Operación de Programación) con un valor de 02h. Si el comando coincide con alguna de ellos el PS pasa inmediatamente el respectivo modo de operación, en caso contrario el PS regresa de vuelta al inicio del MEG.

Nota: De aquí en adelante la transmisión y recepción de cualquier dato considerara implícitamente la utilización de la rutinas TRANSMITE y RECIBE, respectivamente, a menos que se indique lo contrario.

3.3.5 Modo de Operación de Programación.

Si el comando enviado ha sido el de MP (Modo Programación), el PS procede a ejecutar dicho modo de operación en un ciclo While, cuya condición de terminación es que se reciba el comando de terminación de MP.

El diagrama de bloques detallado del modo programación es mostrado en la Figura 3.17.

Como se mencionó, la primera acción que es tomada en cualquiera de los tres estados del PS es informar acerca de los mismos, tanto al monitor de estado de la TEPDES como a la IPVSID.

En el modo programación primero se envía el mensaje "SID CORRIENDO EN MODO PROGRAMACIÓN" al monitor de estado de la TEPDES, utilizando dos veces la macro MENSAJE_LCD3, ya explicada (apartado 3.3.4).

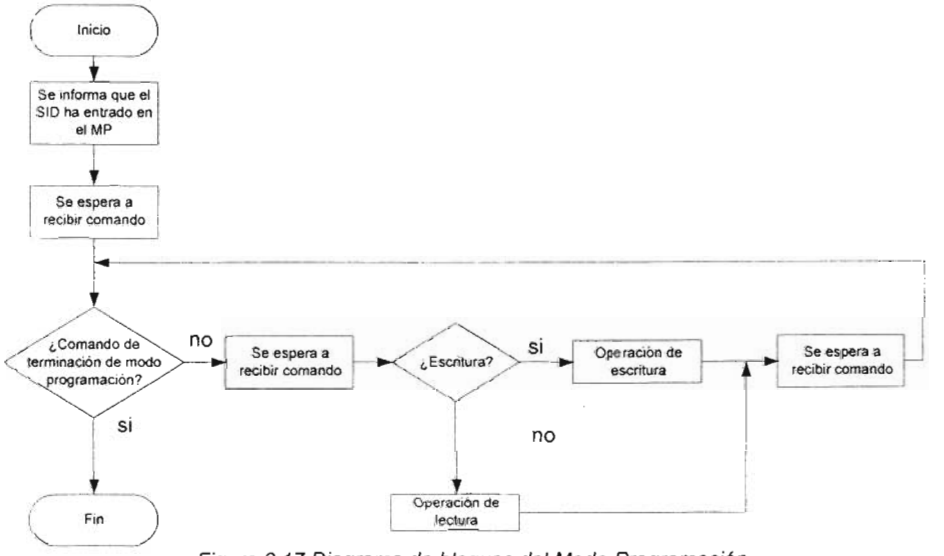


Figura 3.17 Diagrama de bloques del Modo Programación.

Enseguida se transmite a la IPVSID la constante CABECERA_MP previamente definida con un valor de 04h, que indica que el PS ha entrado al modo programación con lo que el usuario ya puede según desee, escribir (descargar) o leer el código generado para un controlador difuso diseñado.

Una vez notificado el nuevo estado del PS, en el MP se ejecuta el bloque de espera de recepción de un comando de terminación o no terminación del MP, en este punto al igual que en el MEG, el programa se queda detenido hasta que se completa la recepción de un comando.

Después de que se completa la recepción del comando, este se compara con la constante de iniciación del MP que fue definida a un valor de 06h y se llamó INICIA_PROGRAMACION, de ser así se procede a ejecutar la siguiente etapa del MP, en caso contrario se asume que el comando ha sido de terminación del MP definido a el valor de 10h y llamado FIN_PROGRAMACION, acción que provoca que el control de programa vuelva otra vez al MEG para ejecutar una vez mas sus operaciones respectivas.

De haber enviado el comando de inicio de MP, enseguida el PS se detiene a la espera de la recepción de un comando que le indique cual de las dos operaciones del modo programación ha sido seleccionada por el usuario para proceder a ejecutarla.

Las dos operaciones posibles del modo programación son:

1. Escritura o descarga a la TEPDES, del código generado para un controlador difuso diseñado.
2. Lectura de verificación del código descargado para compararlo con el código generado (esta operación no es necesaria, por que es raro que ocurra un error durante la descarga, por lo tanto se puede prescindir de ella).

De cualquier modo una vez ejecutada la operación de lectura o la operación de escritura, el PS dentro de modo programación se detiene una vez mas a la espera de la recepción de un comando de terminación o no terminación del MP, para determinar otra vez mas la permanencia en el modo programación.

3.3.5.1 Operación de Escritura del Modo Programación.

Si el comando recibido es el de escritura, que fue definido en un valor de 07h y llamado COMANDO_ESCRITURA, el PS procede a ejecutar la operación de escritura; en caso contrario el comando es interpretado como el de lectura, que fue definido con un valor 17h y llamado COMANDO_LECTURA.

El objetivo principal de la operación de escritura es descargar a la TEPDES el código generado para un controlador difuso diseñado, el cual corresponde al código de la base de conocimiento y el código de operación del sistema de control difuso.

La descripción del código generado, no se da en esta sección y se reserva para cuando se explique el diseño del PCD, esto, por que la operación de escritura es independiente del tipo de código o datos que se vayan a escribir.

EL diagrama de flujo general de la operación de escritura es mostrado en la Figura 3.18.

El funcionamiento general de la rutina de escritura consiste en recibir inicialmente dos bytes que concatenados indican el número de datos que se escribirán, los cuales son almacenados en memoria RAM con el nombre de NUMERO. Enseguida se recibe el comando (1 byte) de selección de memoria de escritura, que puede ser la EEPROM de datos o la FLASH de programa del PIC (la memoria de programa también puede ser utilizada como memoria de datos). Después de que se completa la recepción del comando, es comparado con la constante de selección de memoria FLASH que fue definida en un valor de 13h y se llamo MEMORIA_FLASH, de coincidir con ella se procede a ejecutar la instrucción de llamada a la rutina de escritura de datos en memoria FLASH, la cual se encargara de recibir y escribir en memoria FLASH todos los datos indicados por la variable NUMERO; en caso contrario el comando es asumido como el de selección de memoria EEPROM el cual es definido por una constante cuyo valor fue fijado a 08h y que se llamo MEMORIA_EEPROM, en este caso se procede a llamar a la rutina de escritura de datos en memoria EEPROM, que es análoga a la de la memoria FLASH.

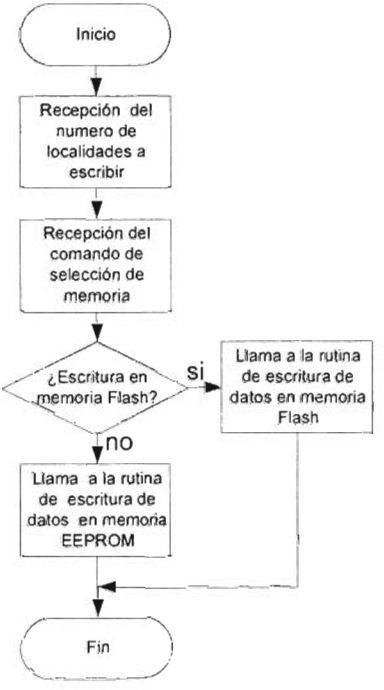


Figura 4.18 Diagrama de flujo general de la operación de escritura.

El acceso a la memoria de datos EEPROM y a la memoria FLASH de programa se lleva acabo indirectamente por un conjunto de registros especiales llamados:

- EEDATA

- EEDATH
- EEADR
- EEADRH
- EECON1
- EECON2

La memoria FLASH de programa del PIC tiene 8 k palabras de 14 bits que además de ser direccionadas por el contador de programa (PC), puede ser direccionadas indirectamente por el par de registros EEADRH y EEADR (solo son tomados en cuenta los 13 bits de menos peso) y accedida para la lectura y escritura a través del par de registros EEDATH y EEDATA (solo son validos los 14 bits de menos peso), operaciones controladas por el par de registros EECON1 y EECON2.

La memoria EEPROM de programa del PIC tiene 256 bytes, que son direccionados por el registro EEADR y accedidos para la lectura y escritura a través del registro EEDATA, operaciones controladas por el par de registros EECON1 y EECON2. Para la memoria EEPROM el valor cargado en los registros EEADRH y EEDATH no tiene importancia.

Rutina de Escritura de datos en memoria FLASH.

De acuerdo a las características de la memoria [b], el diagrama de flujo que ilustra la operación general de la rutina de escritura de datos en memoria FLASH es mostrado en la Figura 3.19.

Se puede ver que la operación general de esta rutina esta controlada por la ejecución de un ciclo Do While cuya condición de terminación es que se reciban y escriban todos los datos indicados inicialmente por la variable NUMERO inicializada al principio de la operación de escritura (ver Figura 3.18).

La primera acción que se realiza al comenzar el cuerpo del ciclo es llamar a un rutina nombrada "RECEPCIÓN_DIR_DATO", cuyo fin principal es recibir la dirección y el dato de escritura, esto es, esperar la recepción de la parte alta de la dirección de escritura y copiarla en EEADRH, la recepción de la parte baja de la dirección de escritura y copiarla en EEADR, la recepción de la parte alta del dato de escritura y copiarla en EEDATH y finalmente la recepción de la parte baja del dato de escritura y copiarla a EEDATA.

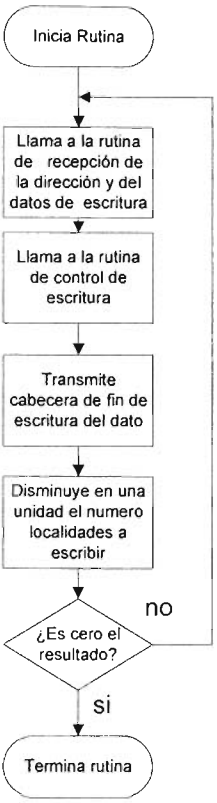


Figura 4.19 Rutina de escritura de datos en la memoria FLASH de programa.

Ejecutada la rutina anterior, enseguida se llama a una rutina nombrada ESCRIBIR_FLASH, cuyo objetivo principal es controlar la escritura de la memoria FLASH. Esta rutina manipula adecuadamente los bits de los registros EECON1 y EECON2 para hacer una escritura exitosa, y cuyas acciones generales son:

- Seleccionar la memoria FLASH.
- Habilitar de la operación de escritura.
- Aplicar la acción de seguridad de escritura.
- Comenzar la escritura.
- Esperar la finalización de la escritura
- Deshabilitar la escritura, para evitar la posible escritura de datos erróneos.

Completada la ejecución de la ultima rutina, los datos previamente copiados a el par de registros EEDATH:EEDATA, habrán sido escritos en la memoria FLASH, en la dirección indicada por EEADRH: EEADR.

Para informar de esta situación se transmite una cabecera cuyo valor fue definido en 41h. Esto se hace para informar al programa de gestión de escritura de la IPVISID, que la escritura del último dato transmitido ha sido completada, por lo que de haber más, se puede proceder a enviar el siguiente dato y su respectiva dirección.

Enseguida de haber ejecutado la última acción se decrementa en una unidad el par de registros llamados NUMERO, cuyo valor ha sido inicialmente cargado con el numero de datos a escribir para una operación de escritura dada. EL nuevo valor de NUMERO es comparado con 0, para determinar si hay mas datos por escribir, continuando con la ejecución del ciclo o si se ha completado la escritura del bloque de datos con lo que ejecución del ciclo de escritura actual se termina.

Rutina de Escritura de datos en memoria EEPROM

La rutina de escritura de datos en memoria EEPROM funciona igual a la de la memoria FLASH con la diferencia de que en lugar de llamar a la rutina de control de escritura ESCRIBIR_FLASH se llama a la rutina de control de escritura de memoria EEPROM, la cual en vez de seleccionar la memoria FLASH, selecciona la memoria EEPROM. Aunque la rutina de escritura de memoria EEPROM también recibe el valor del los registros EEDATH y EEADRH, su valor es ignorado, ya que para ella solo son validos EEDATA y EEADR, a un así, estos valores son puestos a cero desde el programa de gestión de escritura de datos en la IPVSID.

La operación de Escritura concluye con la terminación de la ejecución de la rutina de escritura de datos en memoria EEPROM o en memoria FLASH según se haya seleccionado (ver Figura 4.18).

3.3.5.2 Operación de Lectura del Modo Programación.

Si el comando recibido es el de lectura se procede a ejecutar la operación de lectura, cuyo diagrama de flujo es mostrado en la Figura 3.20.

Como se puede ver de este diagrama la operación de lectura es muy similar a la operación de escritura.

La diferencia principal es que para la operación de lectura solo basta especificar el número de datos que se leerán a partir de una dirección inicial, y determinar el tipo de memoria para llamar a la correspondiente rutina de lectura de datos.

Rutina de lectura de datos en memoria FLASH.

Si la selección ha sido una lectura de la memoria FLASH, enseguida se llama a la rutina de lectura de datos de la memoria FLASH, la cual fue llamada LEER_DATOS_FLASH, cuyo diagrama de flujo es mostrado en la Figura 3.21.

El cuerpo de esta rutina se ejecuta en un ciclo Do while cuya condición de terminación es que se hayan leído todos los datos de la operación actual de lectura (el contador de datos <variable NUMERO> es igual a cero).

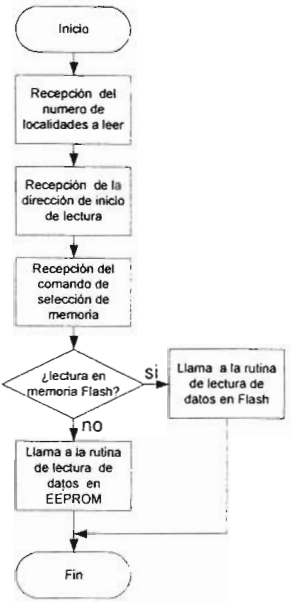


Figura 4.20 Diagrama de flujo de la operación de lectura del Modo Programación.

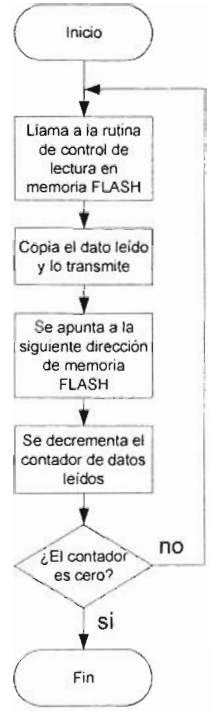


Figura 4.21 Rutina de lectura de datos de la memoria FLASH.

La primera acción que es ejecutada, es la llamada de la rutina de control de lectura de la memoria FLASH que fue llamada LEER_FLASH y que realiza las siguientes acciones:

- Selecciona la memoria FLASH de programa.
- Da el comando de lectura.
- Espera tres ciclos de Instrucción para que el dato localizado en la dirección apuntada por EEADRH y EEADR sea copiado al par de registros EEDATH y EEDATA, para su posterior transmisión.

Una vez ejecutada la última rutina, los datos contenidos en el par de registros EEDATH y EEDATA son transmitidos.

Terminada la transmisión la dirección especificada por el par de registros EEADRH y EEADR es incrementada para apuntara a la siguiente dirección de lectura.

Enseguida la variable NUMERO inicialmente cargada con el número de localidades a leer se decrementa en una unidad (situación que indica que se ha completado la lectura de una palabra de la memoria FLASH), después de lo cual es comparada con la constante cero, que de coincidir con ella, significa que se han leído todos los datos especificados, en caso contrario aun no se habrá completado la lectura de todos los datos, por lo que el ciclo continua ejecutando.

Rutina de lectura de datos en memoria EEPROM.

La rutina de lectura de datos en memoria EEPROM que fue llamada LEER_DATOS_EEPROM, funciona igual que la rutina de lectura de datos en memoria FLASH.

Para la memoria EEPROM solo son validos los registros EEDATA y EEADR, ya que solo tiene implementados 256 bytes, por lo tanto en lugar de transmitir el valor de EEDATH se transmite un valor constante de 255 para que la rutina de gestión de lectura de datos de IPVSID se entere de este hecho y haga el procesamiento adecuando de los datos recibidos.

La operación de lectura concluye con la terminación de la ejecución de la rutina de lectura de datos en memoria EEPROM o en memoria FLASH según se halla seleccionado.

3.3.6 Modo de Operación de Ejecución.

Si el comando enviado ha sido el de ME (Modo Ejecución), el PS procederá a ejecutar dicho modo de operación si así es confirmado, de lo contrario no se ejecutara y el control de programa pasará una vez mas al MEG.

De ser confirmado, se ejecutarán las acciones correspondientes al modo de operación, el cual será abortado hasta que se reciba el comando de terminación del ME.

El diagrama de flujo general del modo ejecución es mostrado en la Figura 3.22

La primera acción que se ejecuta en este modo de operación es esperar la recepción de un comando.

Enseguida, el comando recibido es comparado con la constante que define el comando de inicio del ME, que fue fijada en el valor 0Eh (nombrada INICIA_EJECUCION), de coincidir se procede a ejecutar el cuerpo del ME, en caso contrario no se ejecuta y el control de programa pasa directamente al MEG.

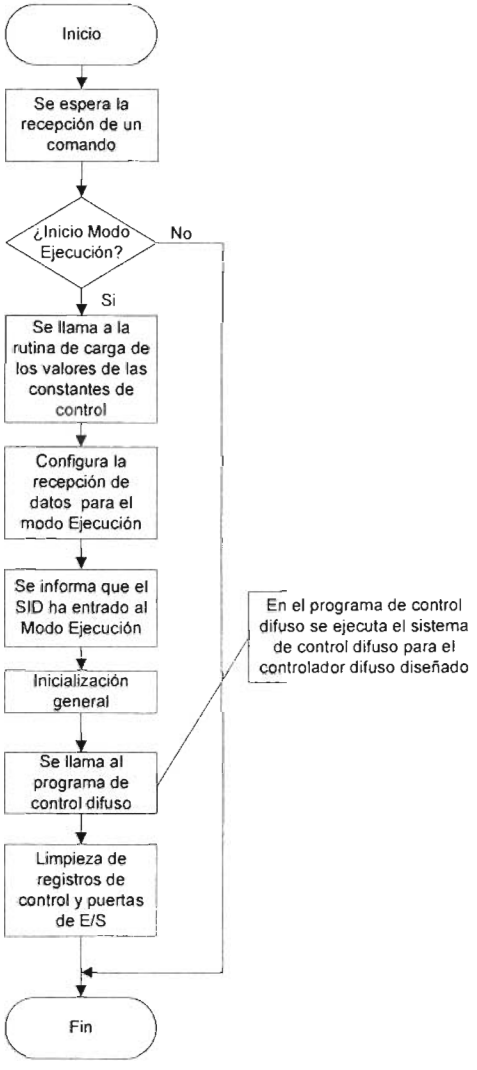


Figura 3.22 Diagrama de flujo general del Modo de Operación de Ejecución.

Dentro del ME la primera acción realizada es la ejecución de la instrucción de llamada a la rutina de carga de valores de las constantes de control nombrada

CARGAR_CONSTANTES_DE_CONTROL, la cual es encargada de copiar un conjunto de valores escritos como datos en la memoria FLASH durante la descarga del código generado para un controlador difuso diseñado, a un conjunto de localidades de memoria RAM reservadas, las cuales son necesarias para el correcto funcionamiento de muchas de las rutinas de aplicación que son utilizadas en la ejecución del sistema de control difuso.

Una vez cargadas todas las constantes de control se ejecuta un bloque para la configuración e inicialización de la recepción de datos para el modo ejecución. Esto se hace por que la rutina de recepción que se ha utilizado hasta ahora esta basada en el método del poleo y como fue explicado, esta rutina detiene temporalmente la ejecución secuencial del programa mientras no se complete la recepción, además el dato recibido es procesado inmediatamente según el conjunto de instrucciones particulares de programa que siguen a continuación, por lo tanto esta rutina se debe ejecutar siguiendo un orden definido previamente. En conclusión, las características de esta rutina no permiten un procesamiento en tiempo real necesario para la ejecución del sistema de control difuso, de lo cual surgió la necesidad de definir un nuevo método de recepción para el modo ejecución, el cual es explicado en detalle, mas adelante en este mismo capítulo.

A continuación de haber configurado la recepción de datos para el modo ejecución se procede a informar tanto al monitor de estado de la TEPDES como a la IPVSID que el SID ha entrado al modo de operación de ejecución. Al monitor de estado de la TEPDES es enviado el mensaje "SID CORRIENDO EN MODO EJECUCION" y a la IPVSID se transmite la cabecera de modo ejecución definida en un valor de 09h y llamada CABECERA_ME.

Después de ser enviado el mensaje del estado del PS se ejecuta el bloque de inicialización general, que comprende la configuración de algunos periféricos y la inicialización de algunos registros de control que son utilizados durante la ejecución de las rutinas del sistema de control difuso. La explicación detallada de este bloque es dada mas adelante.

Ejecutado el último bloque se procede a llamar al programa de de control difuso (PCD) dentro del cual se ejecuta el proceso de control difuso del controlador difuso diseñado. El control de programa retorna de esta rutina solamente cuando en la misma es recibido el comando de terminación del modo ejecución.

Cuando el control de programa retorna del PCD, enseguida se ejecuta el bloque de limpieza para restablecer los periféricos y las salidas de control que pudieran permanecer encendidas.

Finalizada la ejecución del último bloque el control de programa retorna una vez más al MEG.

3.3.6.1 Carga de las constantes de control para el sistema de control difuso.

Como se mencionó la carga de los valores de las constates de control consiste en llamar a la rutina CARGAR_CONSTANTES_DE_CONTROL, la cual es encargada

de leer y copiar el valor de unas localidades de la memoria FLASH a un conjunto consecutivo de localidades de memoria RAM reservadas para guardar el valor de un conjunto de constantes de control que son utilizadas por las rutinas de aplicación, que en combinación forman el algoritmo de control difuso. El valor de las localidades de memoria FLASH es previamente cargado con dichos valores, los cuales se generan y descargan (escriben) en conjunción con el código de la base de conocimiento y el algoritmo de control difuso. El valor de estas constantes de control se genera para cada diseño ya que su valor depende del mismo.

Estas constantes de control en orden en la memoria RAM son:

DIVME: constante que indicará la dirección inicial (apuntador) a partir de donde se almacenaran en RAM las entradas difusas generadas por el proceso de fuzzificaciòn de las entradas fisicas o crisp para un controlador difuso diseñado (ver apartado 3.3.7.4.1.2). En realidad el valor de esta constante siempre será el mismo para cada diseño, ya que dentro de la memoria RAM se tienen reservadas de antemano ciertas localidades para tal propósito, pero aun así se deja con la posibilidad de poder cambiar su valor, aunque esta no este disponible para el usuario, sin embargo, si para propósitos de soporte del autor.

DIVMS: constante que indicará la dirección inicial a partir de donde se localizaran en memoria RAM, las salidas difusa producto de la evaluación de reglas, que son interpretadas como los pesos de cada uno de las funciones singleton de la variable lingüística de salida del controlador difuso diseñado (ver apartado 3.3.7.4.3). El valor de esta constante si es variable de un diseño a otro ya que su valor depende del numero de localidades reservadas para las entradas difusas (que depende del numero de entradas y las etiquetas lingüísticas de cada una), ya que a continuación de estas se reservan las localidades para las salidas difusas. El cálculo del valor de esta variable es transparente al usuario, ya que es realizado automáticamente por la IPVSID.

TXDAT: esta constante es utilizada para indicar el número de datos (bytes) que serán transmitidos desde la TEPDES durante la ejecución del sistema de control difuso, a la IPVSID, para visualizar el estado de cada una de variables involucradas en el proceso de control difuso de un diseñado dado. En otras palabras esta constante define el tamaño del buffer de transmisión en modo ejecución y su valor (calculado automáticamente por la IPVSID) depende del número y tipo de entradas definidas en un diseño.

RXDAT: esta constante indicará el número de datos (bytes) máximo del buffer de recepción en modo ejecución. Su valor esta fijo a un valor de 2 (ver apartado 3.3.6.2) pero existe la posibilidad de ser cambiando solo que para propósitos de diseño por parte del autor.

NFCS: constante que indicará el número de conjuntos singleton que tiene la salida del controlador difuso diseñado (su valor es asignado automáticamente por la IPVSID).

NFPE1: esta constante indicará el número de conjuntos difusos para la primera entrada del controlador difuso diseñado (su valor es asignado automáticamente por la IPVSID).

NFPE2: Esta constante indicará el número de conjuntos difusos para una segunda entrada del controlador difuso diseñado si es que hubiera (su valor es asignado automáticamente por la IPVSID).

NFPE3: Esta constante indicará el número de conjuntos difusos para una tercera entrada del controlador difuso diseñado si es que hubiera (su valor es asignado automáticamente por la IPVSID).

LOADF: constante de control, de la cual algunos de sus bits indicarán el tipo de cargas que manejara la salida del controlador difuso diseñado cuando se le especifica una salida tipo SNZP o una salida tipo CCONTROL, que serán explicadas detalladamente cuando corresponda (ver apartados 3.3.7.5.3 y 3.3.7.5.4)

AFPEE: se utiliza para indicar la dirección inicial en memoria FLASH a partir de donde se encontrará la definición de las funciones de membresía de cada una de las entradas que pudiera tener el controlador difuso diseñado. Esta compuesta de dos bytes AFPEEH y AFPEEL de los cuales solo son validos los 13 bits de menos peso para poder apuntar a las 8k palabras de 14 bits de la memoria FLASH, de las cuales solo son utilizadas para almacenamiento de datos las localidades arriba de los 4k bytes

AFCEE: constante que indicará la dirección inicial en memoria FLASH a partir de donde se encontrara la definición de los conjuntos singleton de la salida del controlador difuso diseñado. Al igual que AFPEE esta compuesta de dos bytes AFCEEH y AFCEEL, los mismos comentarios son validos para estas constantes.

ABREE: constante que indicará la dirección inicial en memoria FLASH o EEPROM a partir de donde se encontrara la definición de la base de reglas del controlador difuso diseñado. Esta compuesta de dos bytes ABREEH y ABREEL, auque para la memoria EEPROM solo se utiliza ABREEL.

Cada un de esta constantes de control se utilizan por distintas rutinas de aplicación que son utilizadas en el proceso de control difuso (PCD) y su utilidad se detallara mas cuando se expliquen tales rutinas donde corresponde

3.3.6.2 Recepción de datos en el Modo Ejecución.

En seguida de que la rutina de inicialización de las constantes de control termina su tarea se procede a configurar e inicializar la recepción para el modo ejecución.

Configurar la recepción de datos para el modo ejecución simplemente incluye la ejecución de tres instrucciones que son:

- Llamar a la rutina INTCONFRX
- Seleccionar el banco 2 y 3 para direccionamiento indirecto
- Llamar a la rutina INTGLOB.

La rutina INCONFRX habilita la interrupción para el receptor del USART cuando su buffer se llena y realiza la inicialización de dos registros de control necesarios en la rutina de atención a interrupción respectiva.

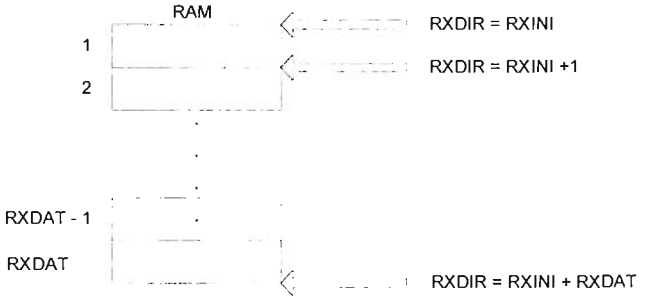
Los bancos 2 y 3 para direccionamiento indirecto son seleccionados debido a que en ellos están definidos los buffer de recepción y transmisión y todas las demás variables que usan el direccionamiento indirecto.

La rutina INTGLOB habilita el permiso global de interrupciones y de periféricos.

Para comprender el sentido de la ejecución de estas instrucciones se explicara la estructura y el diseño de la recepción durante el modo ejecución.

Las características de diseño de la recepción durante el modo ejecución fueron las de proporcionar un método de recepción independiente del programa principal el cual permitiera la recepción de parámetros de manera general.

En base a estas características se opto por utilizar un buffer de recepción compuesto de una serie de localidades RAM para almacenar los datos recibidos y de donde posteriormente deben ser copiados para ser procesados. La estructura general de este buffer es mostrada en la Figura 3.23A, la cual funciona como una cola, es decir que el primer dato en llegar es el primer dato en salir y cuando se sobrepasa su tamaño sin haber leído ningún dato de la misma los nuevos datos se sobrescriben cíclicamente. Para apuntar a cualquier posición dentro del buffer de recepción se definió un apuntador que se llamo RXDIR el cual es cargado inicialmente con la dirección inicial RXINI del buffer y conforme se va llenando se va incrementando hasta llegar a el valor de RXDAT + RXINI, donde RXDAT es la constante de control (ver apartado 3.3.6.1) que especifica el tamaño del buffer de recepción.



A) Estructura General

B) Estructura particular con RXDAT = 2

Figura 3.23 Estructura del buffer de recepción.

Para que la recepción sea independiente del programa principal, se debe utilizar el método de interrupción y por lo tanto diseñar la rutina atención a interrupción adecuada.

Figura 3.24 Diagrama de flujo de la operación de la rutina de atención a la interrupción por recepción de un dato en el módulo receptor del USART.

Si lo que se desea es un método de recepción general, en el cual no se haga ningún procesamiento especifico dependiendo del parámetro recibido se deben tomar en cuanta el propósito de la recepción en modo ejecución, el cual es permitir la posibilidad de cambiar el valor de algunos parámetros del sistema de control difuso en cualquier momento mientras este se esta ejecutando. El valor de cada de uno de estos parámetros es almacenado en localidades de memoria RAM reservadas para tal, y son rutinas (en alguna parte del programa de de control difuso) especificas las encargadas de tomara dicho valor y hacer su procesamiento correspondiente. Por los tanto dicho valor puede ser cambiado

desde cualquier otra parte del programa de tal manera que este disponible cuando tenga que ser utilizado. Para poder cambiarlo basta simplemente con conocer su dirección y el nuevo valor con el que se va cargar, por lo tanto solo son necesarios dos bytes para cambiar el valor de un parámetro, que son su dirección en RAM y el valor con el que se va a cargar, después de lo cual con un direccionamiento indirecto el valor es copiado en la dirección correspondiente para su posterior procesamiento.

En base a la información anterior la estructura de buffer de recepción resultante es la observada en la Figura 3.23B, con una RXDAT fijada en un valor de 2, suficiente para recibir la dirección y el valor de un parámetro

El diagrama de flujo de la rutina de atención a la interrupción por recepción del USART que se llamo INTRX, es mostrado en la Figura 3.24.

Con lo que queda claro el propósito de cada una de la tres instrucciones para configurar el la recepción del modo ejecución.

3.3.3.3 Inicialización general del modo ejecución.

Una vez configuradas las comunicaciones para el modo ejecución el siguiente bloque que es ejecutado es el de inicialización general el cual consiste de manera global en configurar e inicializar un conjunto de periféricos y registros necesarios para el correcto funcionamiento del PCD.

Estas acciones incluyen la ejecución de una serie de instrucciones que son:

1. llamar a la rutina INIGEN_MOD_EJECUCION.
2. llamar a la rutina CFCAD.
3. llamar a la rutina INTCONFTX.
4. llamar a la rutina FIJAR_T_MUESTREO.

La rutina INIGEN_MOD_EJECUCION ejecuta una serie de instrucciones que clarean los bits de todos los puertos de E/S del PIC y llama a una rutina nombrada HABILITAR_EJECUCION que carga la variable de control CONTINUAE con el valor de permanencia del Modo Ejecución para que se ejecute continuamente el ciclo del sistema de control difuso, y pone también a cero el registro de banderas (llamado BANDERAS) que son utilizadas por diversas rutinas del PCD.

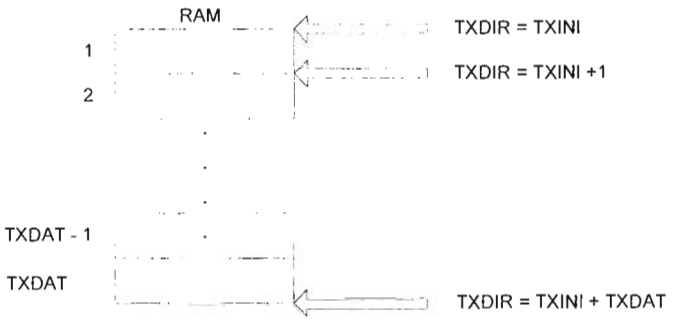
La rutina CFCAD es la encargada de configurar e inicializar el módulo A/D del PIC para prepáralo para la conversión (ver apartado 3.3.7.3.1), entre sus tareas se incluyen:

- Configurar el convertidor A/D con una justificación del resultado a la derecha, los cinco primeros pines de E/S del puerto A como canales analógicos, con Vdd y Vss como las referencias Vref+ y Vref- respectivamente.

- Configurar los tres primeros pines del puerto A como entradas, ya que es el máximo número de entradas analógicas permitido en el SID, los dos canales restantes son ignorados y se dejan como salidas.
- Configurar un frecuencia de conversión de $F_{osc}/32$ con un reloj de 20 MHZ, y encender el convertidor.

El objetivo principal INTCONFTX es inicializar los registros de control del buffer de transmisión, que es utilizado por la rutina de transmisión del estado de las variables del sistema de control difuso a la IPVSID, donde son visualizadas.

La estructura de dicho buffer de transmisión es similar a la del buffer de recepción y es mostrado en la Figura 3.25.



A) Estructura General

Figura 3.25 Estructura del buffer de transmisión.

Este buffer es llenado con el valor de las variables del sistema de control difuso que incluyen las entradas al controlador difuso (variables de estado del proceso bajo control), la salida del controlador difuso (variable de control) y la salida del proceso bajo control (variable controlada). Este es cargado en orden en las diferentes rutinas que componen la adquisición y salida de datos en el algoritmo de control difuso. Una vez que se completa una ejecución del algoritmo de control difuso (un muestreo) el buffer habrá sido llenado con los valores correspondientes de la variables y estará listo para ser transmitido.

La rutina encargada de transmitir el buffer de transmisión se llamó TRANSMITIR_BUFFER y su diagrama de flujo es mostrado en la Figura 3.26. Esta rutina direcciona el buffer de transmisión y transmite cada uno de sus datos uno tras otro llamando a la rutina de transmisión por poleo TRANSMITE (explicada en el apartado 3.3.4) mientras falten mas datos por transmitir, y cuyo numero es indicado por la constante de control TXDAT (definida en el apartado 3.3.6.1).

Esta es una rutina de aplicación, que es llamada al final de una ejecución(un muestreo) del algoritmo de control difuso (dentro del PCD), es decir cuando se tiene ya disponibles todas la variables que deben ser visualizadas para un periodo de muestreo. Su ejecución toma cierto tiempo que depende del tamaño del buffer y la velocidad de transmisión que como se ha mencionado, está fija a un valor de 56700 bits por segundo. Puesto que su ejecución se lleva acabo dentro del cuerpo

del proceso de control difuso limita el tiempo de procesamiento difuso y por lo tanto el periodo de muestreo mínimo. Más detalles acerca de esta cuestión son dados cuando se explique el diseño de las rutinas de control de muestreo del PCD (apartado 3.3.7.6).

Por lo tanto, lo único que realiza la rutina INTCONFTX es cargar el apuntador TXDIR de buffer de transmisión con la posición inicial TXINI del mismo.

Figura 3.26 Diagrama de flujo de la rutina de transmisión de buffer (TRANSMITIR_BUFFER).

La última instrucción que es ejecutada dentro de este bloque de inicialización es la llamada de la rutina FIJAR_T_MUESTREO, cuyo objetivo principal es configurar el Timer 0 del PIC e inicializar los registros de control para el control de la frecuencia de muestreo del sistema de control difuso. Las acciones que realiza esta rutina son:

- Configurar el Timer 0 como temporizador con incremento en los flancos ascendentes, y asignarle un predivisor de frecuencia de 1/256.
- Habilitar las interrupciones por desbordamiento del Timer 0 y limpiar inicialmente su bandera de interrupción.
- Cargar la variable de control de desbordes del timer 0, CONTADOR3 con su valor inicial, cargar el registro de cuentas del Timer 0 (TMR0) con su valor inicial, esto para configurar la frecuencia inicial de muestreo del sistema de control difuso.

Más detalles acerca da la configuración e inicialización a los valores anteriores en el control de la frecuencia de muestreo se dan en la sección de diseño correspondiente (ver apartado 3.3.7.6).

Una vez que es hecha la inicialización general del modo ejecución se procede a llamar al PCD, donde se ejecutara el algoritmo de control difuso, este retornará el control de programa otra vez mas al ME del PS desde donde ha sido llamado hasta que se reciba el comando de terminación de modo ejecución, como se vera cuando se describa la estructura, diseño y funcionalidad del PCD.

El último bloque de proceso que se ejecuta en el modo ejecución es el correspondiente al de limpieza de registros de control y de limpieza de puertas de entrada salida, donde se llaman a un conjunto de rutinas de utilidad, las cuales no se detallaran, pero cuyo objetivo es restablecer el estado de los periféricos utilizados en el modo ejecución y PCD, a su configuración inicial que tenían, antes de haber entrado en dicho modo. La limpieza de las puertas de E/S se hace por motivos de seguridad para apagar todas aquellas salidas de control de potencia que pudieran haber quedado activadas al abortar la ejecución del proceso de control difuso.

3.3.7 Programa de control difuso.

En el programa de control difuso (PCD) es donde se implementa y ejecuta el sistema de control difuso generado para un controlador difuso diseñado en la IPVSID. El PCD comprende un conjunto de instrucciones fijas que se encargan de controlar la frecuencia de muestreo y la ejecución del sistema de control difuso, además esta compuesto por una serie de posiciones de memoria reservadas para colocar o escribir las instrucciones (rutinas de aplicación) que representaran el algoritmo de control difuso para un controlador difuso diseñado.

La estructura del PCD es mostrada en la Figura 3.27. En el, las instrucciones son un conjunto de llamadas a las rutinas de aplicación que representan la implementación del algoritmo de control difuso para un diseño particular o para una tarea de automatización particular. El código de operación (código máquina) de estas instrucciones de llamada es generado y descargado en conjunción con la base de conocimiento para un controlador difuso diseñado en la IPVSID en las localidades de memoria reservadas para tal. La filosofía de este método se basa en una programación estructurada de tal manera que cada rutina realiza tareas bien específicas dentro de algoritmo de control difuso (ver Figura 3.28).

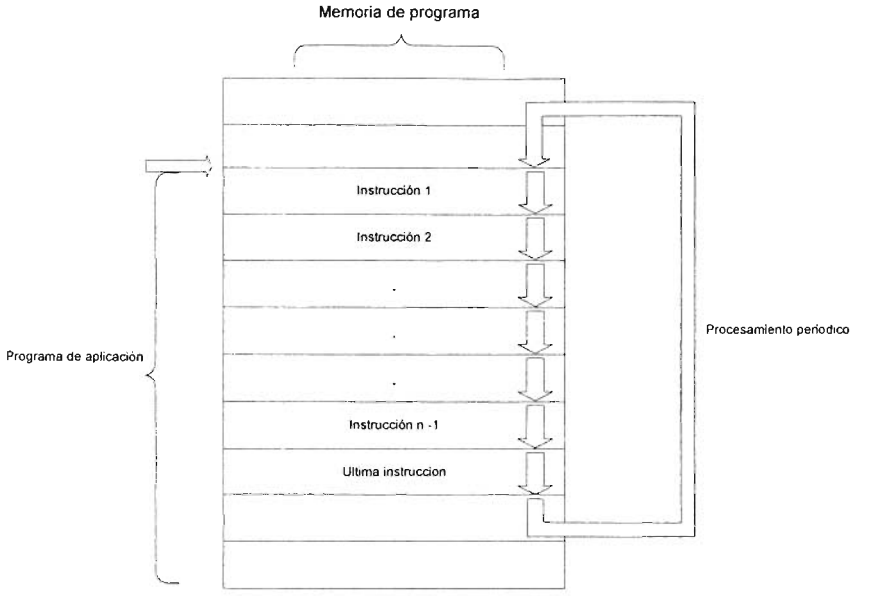


Figura 3.27 Estructura en memoria del programa de control difuso

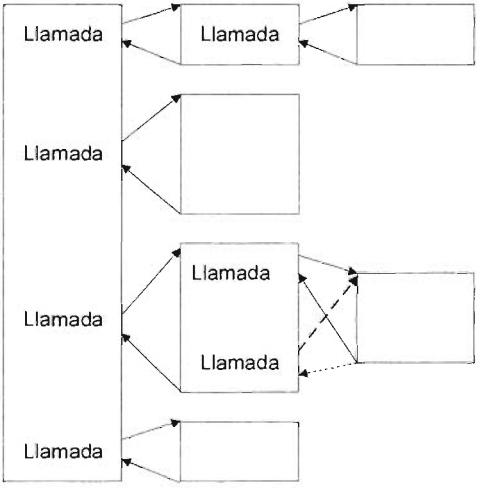


Figura 3.28 Filosofía del programa de control difuso.

EL diagrama de flujo del funcionamiento general del PCD es mostrado en la Figura 3.29.

Se distinguen dos partes del PCD, la primera constituye la ejecución de un bloque que lleva acabo la configuración de los periféricos de control de potencia de la salida del proceso de control difuso y la otra la ejecución de un ciclo Do While, el

cual se ejecutara mientras no se envié el comando de terminación del modo ejecución, dentro del cuerpo de este ciclo se ejecuta primero un ciclo While el cual se ejecuta sin realizar acción alguna mientras no se complete el periodo de muestreo, momento en el cual procede a ejecutar el bloque del proceso de control difuso.

El segmento de programa encargado del control de permanecía en el PCD (ciclo Do While) y el segmento de programa encargado del control de la frecuencia de muestreo (ciclo While) son partes del PCD fijas, es decir su código permanece inalterable para cualquier diseño; en cambio el segmento de programa correspondiente al bloque de configuración del los periféricos de control de potencia de salida y el bloque de ejecución del proceso de control difuso son partes del PCD cuyo código de operación depende de diseño, o en otras palabras representa la parte de memoria del PCD reservada para escribir el código del algoritmo de control difuso generado para un controlador difuso diseñado para una aplicación particular (tarea de automatización).

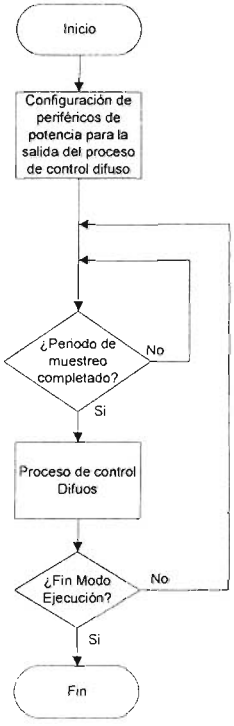


Figura 3.29 Diagrama de flujo del funcionamiento general del programa de control difuso.

En primer lugar se describe el diseño e implementación del conjunto de rutinas que definen el sistema de control difuso.

3.3.7.1Sistema de control difuso.

EL sistema de control difuso esta representado por un conjunto de rutinas (rutinas de aplicación) que tienen funciones bien definidas dentro del esquema de implementación de un sistema de control difuso. Este esquema de implementación es mostrado en la Figura 3.30. Después que los datos son adquiridos, son alimentados a la parte difusa (controlador difuso) que consiste de los tres pasos de fuzzificaciòn, evaluación de reglas y la defuzzificaciòn, la salida de este proceso de inferencia difusa es alimentada a la parte de condicionamiento de datos para producir la salida de control real.

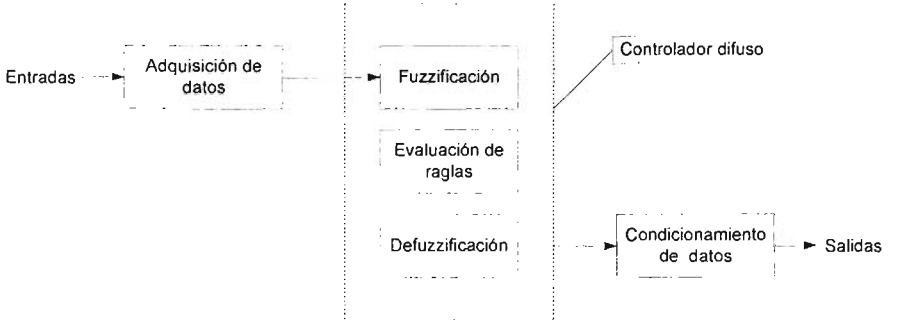


Figura 3.30 Esquema básico de implementación de un sistema de control difuso.

La adquisición de datos comprende el proceso de digitalización de las señales analógicas (variables de estado) provenientes del proceso bajo control y su procesamiento y acondicionamiento para generar las entradas crisp del controlador difuso.

La fuzzificaciòn es el proceso que mapea los valores numéricos crisp del mundo real de las entradas a valores numéricos difusos. La parte de evaluación de reglas constituye la inteligencia del controlador, y es la responsable de la toma de decisiones. La parte de defuzzificaciòn corresponde a una operación de mapeo. Transforma valores numéricos del ambiente difuso al ambiente (mundo real) crisp.

El condicionamiento de los datos consiste del procesamiento que se le hace a la salida crisp del controlador difuso para adaptarla como entrada de control de los periféricos que controlan el suministro de potencia (salida de control real) de los actuadotes del sistema de control para llevar a la salida del proceso controlado a su valor deseado.

El conjunto de rutinas de aplicación deben cubrir las tres operaciones anteriores del esquema de implementación, por lo tanto se dividieron de manera general en tres grupos, es decir el conjunto de rutinas encargadas de la adquisición de datos, el conjunto encargado del los pasos del proceso de inferencia difusa, y el conjunto de rutinas encargado del condicionamiento de la salida.

Las rutinas de adquisición diseñadas fueron:

- **RCAD**
- **CANAL0**
- **CANAL1**
- **CANAL2**
- **SERROR**
- **SCERROR**

Las rutinas de inferencia difusa diseñadas fueron:

- **INIFUZZY**
- **RNUMCDE**
- **RFUZZYFI**
- **RINIEVAREG_FLASH**
- **REVA_REG_FLASH**
- **RINIEVAREG_EEPROM**
- **REVA_REG_EEPROM**
- **INIDEFUZ**
- **RDEFUZZY**

Las rutinas de condicionamiento diseñadas fueron:

- **INIPWM**
- **CONFPWM1**
- **INITRIAC1**
- **CONFPWM2**
- **INITRIAC2**
- **ACTPWM1**
- **ACTTMR1**
- **ACTPWM2**
- **ACTTMR2**
- **SNZP**
- **CCONTROL**

Adicionalmente al conjunto de rutinas anteriores se diseñaron otras rutinas cuya funcionalidad dentro del sistema de control es diferente a la de los tres grupos anteriores pero que también fueron necesarias:

- **TXHEAD**

- **TRANSMITIR_BUFFER**
- **VERUNO**
- **VERCERO**

Cada una de estas rutinas para cumplir con su cometido hacen uso de otras rutinas de utilidad que no se consideran rutinas de aplicación, las cuales se mencionaran y explicaran conforme se exponga el diseño y funcionalidad de cada una de las rutinas anteriores, solo que antes de pasar a eso, se definen los esquemas de control considerados para los controladores difusos que se podrán diseñar con el SID, ya que en base a ellos se diseñaron las rutinas de adquisición y condicionamiento de datos.

3.3.7.2 Esquemas de control considerados por el SID.

El primer esquema de control considerado fue el control proporcional (P), cuyo esquema de implementación es mostrado en la Figura 3.31.

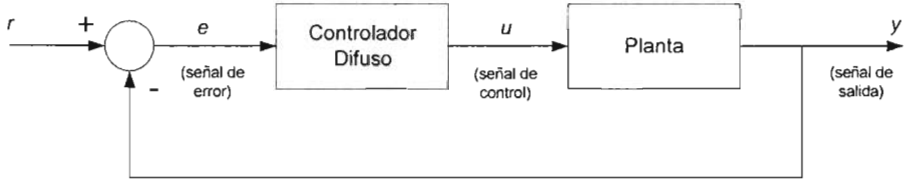


Figura 3.31 Un sistema de seguimiento de set point de lazo cerrado típico (controlador difuso como controlador proporcional (P)).

Donde la ecuación del controlador proporcional es simplemente

$$u(k) = K_p \times e(t) \ldots\ldots\ldots (3.3)$$

y K_p es el factor de ganancia proporcional.

El segundo esquema de control difuso considerado fue un control prealimentado combinado con retroalimentación, el cual es ilustrado en la Figura 3.32, en el cual se observa que una de la entrada al controlador difuso es la señal de error (retroalimentación) y la otra entrada, es la referencia o salida deseada (prealimentación), esquema de control que también se conoce como Proporcional Prealimentado (PP).

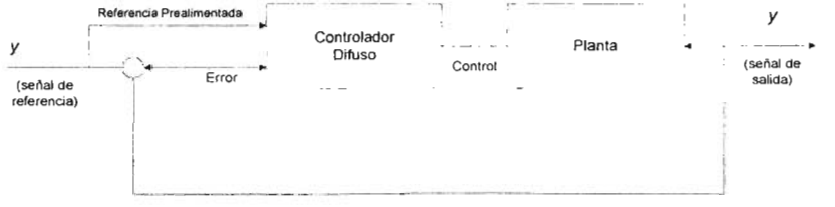


Figura 3.32 Esquema de control con prealimentación de la referencia o proporcional prealimentado (PP)

Otro dos esquemas también considerados fueron el control PI y el control PD, auque fueron implementados en el SID, no fueron utilizados para pruebas, como se vera en el siguiente capitulo. Los diagramas de bloques y esquemas de implementación que fueron considerados para este tipo de controladores son los mostrados en las Figura 3.33, 3.34.

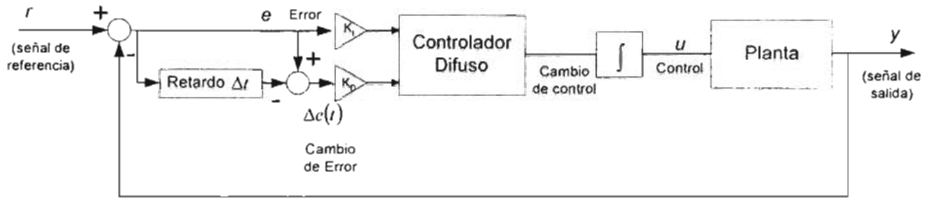


Figura 3.33 Diagrama de bloques de un sistema de control con un controlador difuso como PI.

Donde la ecuación para el controlador PI de la Figura 3.33 esta dada por:

$$\Delta u(t) = K_p \times \Delta e(t) + K_I e(t) \ldots\ldots\ldots (3.4)$$

En este caso se puede considerar la salida del controlador no como una señal de control si no como el cambio en la señal de control, lo que implicaría la integración de esta ultima salida para obtener la correspondiente señal de control.

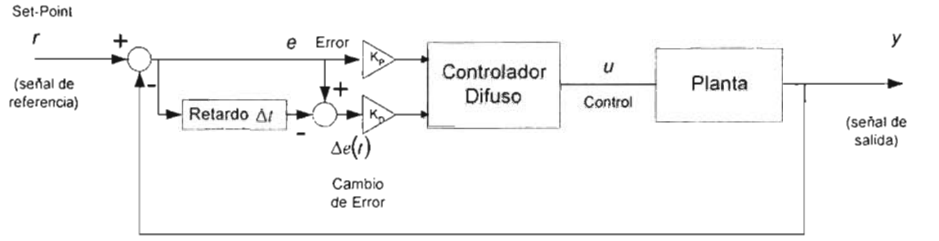


Figura 3.34 Diagrama de bloques de un sistema de control con un controlador difuso como PD.

Donde la ecuación del controlador PD es:

$$u(k) = K_p \times e(t) + K_D \times \Delta e(t) \ldots\ldots\ldots (3.5)$$

Donde K_p y K_D son los factores de ganancia proporcional y diferencial.

A veces cuando la información sobre el objeto o proceso bajo control y su estructura están disponibles, uno puede no querer estar confinado a usar el error, cambio de error, si no mas bien usar las variables de estado del proceso actual (directas). El diagrama de bloques de este tipo de sistemas de control es mostrado en la Figura 3.35.

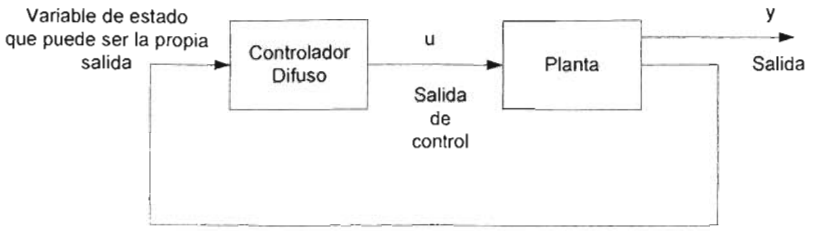


Figura 3.35 Diagrama de bloques de un sistema de control difuso con retroalimentación de estados.

También, auque este esquema fue implementado, no se utilizó en la pruebas, como se vera en el siguiente capitulo.

En resumen el tipo de entradas de acuerdo a los esquemas de control considerados, que podrá tener el controlador difuso serán cuatro:

- Señal de proceso o señal directa (variable de estado directa del proceso a controlar),
- Señal de error definida como $e(t) = r - y(t)$ ..... (3.6)
- Señal de cambio de error definida como, $\Delta e(t) = e(t) - e(t-1)$ ..... (3.7)
- Señal de referencia (prealimentada)

Y el tipo de salidas serán dos:

- Señal de control definida por, $u(t)$ ..... (3.8)
- Señal de cambio de control definida por, $\Delta u(t) = u(t) - u(t-1)$ ..... (3.9)

El máximo número de entradas que podrá tener el controlador difuso será de tres, las cuales deberán ser seleccionadas entre las cuatro anteriores. Tomando en cuenta que no se pueden definir dos señales de error para un controlador difuso y que no se puede definir una señal de entrada de cambio de error sin haber definido antes una señal de error, la cual tampoco se puede repetir.

El número de entradas se limitó a tres en primer lugar por que ese es el promedio de entradas que se maneja para cualquier aplicación de control difuso y el otro motivo es debido a la velocidad de procesamiento, ya que con el numero de entradas generalmente se eleva en forma exponencial el número de reglas y es el tiempo de procesamiento de estas el que más influye en el tiempo total de procesamiento difuso, lo que reduce las frecuencias de muestreo máximas que se pueden manejar y por lo tanto la aplicabilidad del sistema.

El controlador difuso estará limitado a una sola salida cuyo tipo será seleccionado entre las dos mencionadas, con algunas variantes que son introducidas por las rutinas de condicionamiento como se verán mas adelante en los apartados correspondientes.

3.3.7.3 Entradas del controlador difuso.

Las entradas al controlador difuso son generadas por el conjunto de rutinas que representan la implementación de la adquisición de datos.

De acuerdo al apartado 3.3.7.2 estas entradas se pueden escoger entre una o mas variable de estado directas del proceso u objeto bajo control, o una variable de estado de error y/o una variable de estado de cambio de error y finalmente de la referencia prealimentada.

Estas variables de estado deben ser medibles por que de otro modo no se podrían utilizar como entradas físicas, para posteriormente ser digitalizadas y procesadas.

Cabe mencionar que tanto la señal de error y cambio de error en un sistema de control no son variables de estado directas ya que ellas se obtienen a partir del la diferencia de la salida deseada (Set Point o referencia) y la salida real de proceso u objeto bajo control, por lo tanto la única señal que necesita ser digitalizada en este caso es la salida real del sistema bajo control, y las señales de error se calculan partir del resultado digital dentro del microcontrolador.

Para realizar la digitalización de las señales se decidió utilizar el módulo convertidor analógico digital que forma parte de los periféricos incluidos en el PIC , con esto se evito la integración de mas componentes electrónicos en la TEPDES, lo que hubiera complicado mas el diseño electrónico. Otra razones, son la alta velocidad conversión y el ancho de palabra de 10 bits del CA/D incluido, con lo que resulta adecuado para una amplia variedad de aplicaciones de control.

Para comprender mejor el diseño de la rutinas de adquisición enseguida se de una breve descripción del módulo CA/D del PIC. En la Figura 3.36 puede ser observado el diagrama de bloques de módulo CA/D, del cual se destaca:

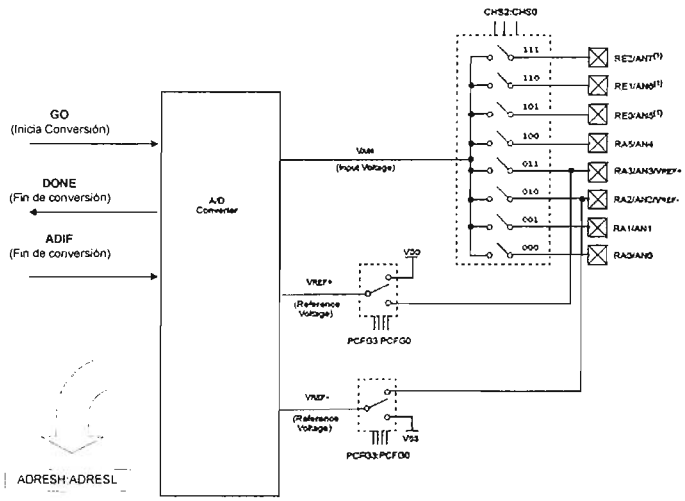


Figura 3.36 Diagrama de bloques del módulo CA/D.

El multiplexor, ya que el CA/D es un convertidor de aproximaciones sucesivas de 10 bits, el cual puede realizar la conversión de una de las 8 entradas (o canales) analógicas $AN0,...,AN7$ multiplexadas por la lógica interna que utiliza como líneas de selección del canal los bits CHS2:CHS0, en donde se coloca el número binario del canal a convertir.

Los voltajes de referencia.- Todo CA/D requiere valores de referencia que determina el valor de mínima escala (Vref-) y el de plena escala (Vref+), de manera que la conversión de un voltaje analógico Vin en el rango de Vref- a Vref+ producirá un valor equivalente binario *D* en el rango de 0 a 2^n, donde *n* es la resolución del convertidor (*n* = 10).

Como la relación de escalas es lineal, una regla de tres nos da la relación entre el voltaje analógico de entrada (Vin) y el valor digital (*D*) obtenido por el CA/D.

$$\frac{D}{2^n - 1} = \frac{Vin - Vref_-}{Vref_+ - Vref_-} \quad \text{..... (3.10)}$$

Con la elección más común: $Vref_+ = V_{DD} = 5V$, $Vref_- = V_{SS} = 0V$, y con *n* = 10, obtenemos:

$$D = \frac{1023}{5} Vin = (204.6)Vin \quad \text{..... (3.11)}$$

De donde se ve que cuando *Vin* varia en todo su rango, desde 0 hasta 5V, el valor el valor obtenido *D* varia en todo su rango, de 0 a 1023.

Si a la inversa, obtenemos un valor D y deseamos saber que voltaje representa, basta con despejar:

$$Vin = \frac{5}{1023} D = (0.04887585533)D \quad \text{..... (3.12)}$$

La salida de los sensores de las variables de estado son antes acondicionadas por el módulo de acondicionamiento de entradas analógicas de la TEPDES (ver apartado 3.2.4) para que el voltaje que se suministre a un determinado canal del convertidor analógico digital este en el rango de Vref- a Vref+.

Registros de trabajo.

El funcionamiento del CA/D requiere de la manipulación de 4 registros de trabajo:

1. ADRESH: parte alta del resultado de la conversión
2. ADRESL: parte baja del resultado de la conversión.
3. ADCON0: registro de control 0.
4. ADCON1: registro de control 1.

En la pareja de registros ADRESH:ADRESL se deposita el resultado de la conversión, que al estar compuesta por 10 bits, solo son significativos los 10 bits de dicha pareja.

El registro ADCON0 controla la operación del CA/D, mientras ADCON1 sirve para configurar las patitas de la puerta A como entradas analógicas o E/S digitales.

Registro ADCON0 (1F)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON
bit 7							bit 0

Registro ADCON1 (9F)

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Figura 3.37 Asignación de los bits de los registros de control del CAD ADCON0 y ADCON1.

Los bits ADCON<7:6> sirven para seleccionar la frecuencia del reloj que se emplea en la conversión, con la siguiente asignación:

Tabla 3.6. Selección de la frecuencia de reloj que se emplea en la conversión.

ADCS1:0	**FRECUENCIA**	T_{AD}
0 0	F_{OSC} / 2	2 T_{OSC}
0 1	F_{OSC} / 8	8 T_{OSC}
1 0	F_{OSC} / 32	32 T_{OSC}
1 1	F_{RC}(Procede del oscilador interno)	Oscilador interno en el C A/D

Se designa como T_{AD} el tiempo que dura la conversión analógica digital y en el caso de trabajar con valores digitales de 10 bits, se requiere un tiempo mínimo de $12T_{AD}$. El valor de T_{AD} se selecciona por software mediante estos bits (ADCS1:ADCS0) y en los PIC16F877 nunca debe ser menor de 1.6 microsegundos.

En el SID, donde el microcontrolador estará trabajando a una frecuencia de oscilación de 20 MHz, la única selección correcta es ADCS1:ADCS0 = $1\ 0_b$, con lo se tiene un tiempo de conversión de:

$$T_{AD} = 32(T_{OSC}) = \frac{32}{F_{OSC}} = \frac{32}{20\times10^6\,Hz} = 1.6\times10^{-6}\,s = 1.6\,\mu s$$

que esta en el limite de tiempo de conversión mínimo.

Los bits CHS<2:0> seleccionan el canal por donde se introduce la señal analógica a convertir, de acuerdo con el siguiente código:

Tabla 3.7. Selección del canal de conversión.

CHS2- 0	CANAL
000	Canal 0 (RA0/AN0)
001	Canal 1 (RA1/AN1)
010	Canal 2 (RA2/AN2)
011	Canal 3 (RA3/AN3)
100	Canal 4 (RA4/AN4)
101	Canal 5 (RE0/AN5)
110	Canal 6 (RE1/AN6)
111	Canal 7 (RE2/AN7)

Como el SID esta restringido a un máximo de tres entradas analógicas, por lo tanto solo se usaran los tres primeros canales del CA/D. Para hacer la selección de cada uno de estos tres canales en cualquier momento se diseñaron tres rutinas respectivamente, que lo que hacen es poner adecuadamente a 1 o 0 los bits CHS<2:0> según el canal que deba ser seleccionado, las cuales detallamos mas adelante en esta mismo apartado.

El bit GO/DONE# es el bit de estado de la conversión. Poniéndolo a 1 se inicia la conversión y mientras este a 1 se esta realizando dicha operación. Cuando GO/DONE# pasa a 0 confirma el final de la conversión y la puesta del resultado en la pareja de registros ADRESH:ADRESL.

El bit ADON sirve para activar el CAD poniéndolo a 1 y para inhibir su funcionamiento poniéndolo a 0.

El bit de más peso (ADFM) del registro ADCON1 selecciona el formato de la conversión. Si vale 1, el resultado esta justificado en el registro ADRESH, que tiene sus 6 bits de más peso a cero; mientras que si vale 0 la justificación se realiza sobre el registro ADRESL, que tiene sus 6 bits de menos peso a 0. Esto significa que los 16 bits que forman la concatenación de los registros ADRESH:ADRESL una veces tiene a cero los seis bits de mas peso y otras los 6 bits de menos peso (alineación a la derecha o a la izquierda). Esta situación es mostrada en la Figura 3.38.

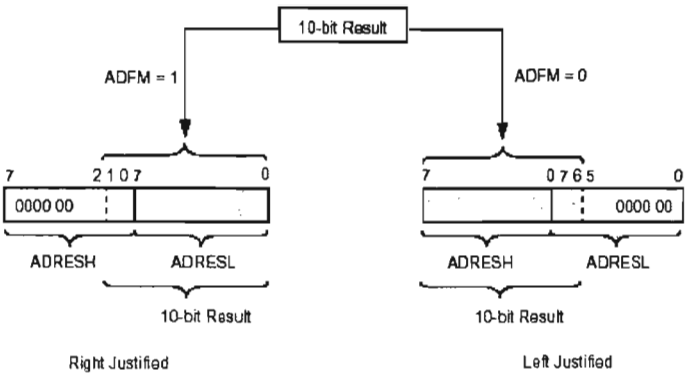


Figura 3.38. Justificación del resultado del CA/D.

Los restantes cuatro bits (PCFG3:0) de ADCON1 se utilizan para configurar las patitas de los canales de entrada al CA/D como entradas analógicas o como E/S digitales, de acuerdo con la siguiente tabla:

Tabla 3.8. Selección de canales analógicos con lo bits de control PCFG3:0.

PCFG3: PCFG0	AN7(1) RE2	AN6(1) RE1	AN5(1) RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs(2)
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A = Entrada analogica D = Entrada/Salida Digital

En el SID solo se usan los tres primeros canales analógicos y como referencias se usan respectivamente V_{DD} y V_{SS}, por lo tanto la opción que mas convino a este propósito, fue el valor de 0010_b para los bits PCFG3:0 con lo que se seleccionan cinco canales como entradas analógicos de las cuales solo se utilizan las tres primeras.

En el diagrama de tiempo de la Figura 3.39 se muestran los eventos que tienen lugar durante el proceso de conversión analógica digital.

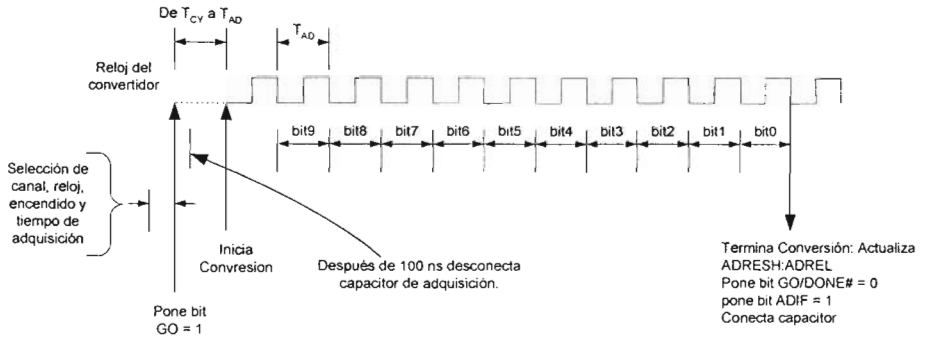


Figura 3.39 Ciclos T_{DA} de la conversión A/D.

De acuerdo a la Figura, para echar a andar el convertidor se deberán seguir los siguientes pasos:

1) Configurar el módulo A/D:

   a) Configurar los pines analógicos y los voltajes de referencia V_{REF-} y V_{REF+}, mediante el registro ADCON1 (9Fh) y los correspondientes bits de la puerta A como entradas.

   b) Seleccionar el canal de entrada a convertir mediante los bits CHS2:CHS0 del registro ADCON0 (1Fh)

   c) Seleccionar el reloj de conversión mediante los bits ADCS1:ADCS2 (ADCON0<7:6>).

2) Configurar interrupciones pare el convertidor si así se desea, para ellos: limpiar ADIF y poner los bits ADIE, PEIE y GIE.

3) Esperar mientras transcurre el tiempo de adquisición (unos $20\mu s$).

4) Iniciar la conversión poniendo el bit GO/DONE# (ADCON0<2>).

5) Esperar a que termine la conversión.

   a) Por "poleo" (Polling): Consultando continuamente el bit GO/DONE# (el cual es limpiado por el convertidor cuando esta completada la conversión).

   b) Por interrupciones: Cuando la conversión termina, la bandera ADIF se activa y esto genera una solicitud de interrupción, al cual deberá ser atendida por una rutina de atención a la interrupción diseñada para ello.

6) leer el dato convertido *D* de los registros (ADRESH:ADREL).

7) Para la siguiente conversión, esperar al menos $2T_{AD}$ (Donde T_{AD} es el tiempo de conversión por bit).

Características Eléctricas del convertidor. Algunas de las especificaciones eléctricas más importantes son mostradas en la Tabla 4.9.

Tabla 3.9. Características eléctricas del convertidor.

Características		mínimo	típico	máximo
Vref+ - Vref-		2V	-	Vdd + 0.3V
Vref+		Vdd – 2.5V	-	Vdd + 0.3V
Vref-		Vss – 0.3V	-	Vref+ - 2V
Voltaje analógico V_{AIN}		Vss – 0.3V	-	Vref+ + 0.3V
Impedancia de la fuente de señal externa Z_{AIN}		-	-	$10K\Omega$
Corriente promedio consumida Por el convertidor	Estándar	-	$220\mu A$	-
	Extendido	-	$90\mu A$	-

Toda esta es la información relevante al convertidor analógico digital del PIC y con ella en mente, se detalla el diseño de las rutinas para generar los tipos de entradas que se pueden seleccionar para un controlador difuso en el SID.

Fueron dos rutinas las que se diseñaron para el control del CA/D. La primera de ellas ya ha sido mencionada (ver apartado 4.3.6.3), CFCAD que abrevia "configurar el CA/D" y como su propio nombre lo dice tiene como objetivo configura e inicializar el CAD para prepararlo para realizar las digitalización de la variables de estado provenientes del proceso u objeto bajo control. Esta rutina forma parte de las rutinas de sistema. Para complementar la explicación de esta rutina, el código ensamblador de la misma es mostrado en la Figura 3.40 el cual habla por si solo y no requiere comentarios adicionales.

Rutina de inicialización y configuración del modulo CAD

```
CFCAD.

        BANKSEL  ADCON1   ; BANCO 1
        MOVLW    0X82
        MOVWF    ADCON1   ; EL RESULTADO SE JUSTIFICA A LA DERECHA, ADFM = 1, 5 ENTRADAS ANALÓGICAS, PCFG3:0 = b'0010'
        MOVLW    NUMEA
        MOVWF    TRISA    ; LOS TRES PRIMEROS BITS DEL PUERTO A COMO ENTRADAS, PARA LAS TRES ENTRADAS ANALÓGICAS RESPECTIVAMENTE
        BANKSEL  ADCON0   ; BANCO 0
        MOVLW    0X81     ; FRECUENCIA DE CONVERSIÓN IGUAL A Fosc/32 CON RELOJ DE 20MHZ, TAD = 1.6US, SE ACTIVA EL CONVERTIDOR AD, ADON = 1
        MOVWF    ADCON0   ; SE SELECCIONA EL CANAL 0 POR OMISIÓN
        RETURN
```

Figura 3.40 Segmento de código ensamblador de la rutina de CFCAD.

La segunda rutina que se diseño es la que propiamente se encarga de realizar el proceso de conversión analógica digital. Dicha rutina convierte a formato digital el voltaje de entrada analógico de uno de los tres canales definidos, según se halla seleccionado previamente (siempre se encuentra seleccionado en primer lugar el canal 0), el resultado de esa conversión es almacenado en memoria RAM, además es copiado al buffer de transmisión para ser enviado a la IPVSID, con lo que es posible conocer su valor en todo momento. La variable RAM que se definió para almacenar el resultado de la conversión se llamo PX, la cual esta dividida en dos bytes que se llamaron PXH y PXL, en donde se guarda respectivamente la parte alta y la parte baja del resultado de la conversión.

En base a los requerimientos de esta rutina y lo requerimientos de módulo CA/D, el diagrama flujo resultante de esta rutina que se llamo RCAD, la cual forma parte de la rutinas de adquisición, es mostrada en la Figura 3.41

Se destacan algunas cosas de este rutina, primero que nada, no se hace una sección del canal de conversión, esto es debido a que por omisión el primer canal seleccionado es el canal 0 y para seleccionar los otros canales se diseñaron tres rutinas especiales que manipulan los bits CHS2:CHS0 para seleccionar cada uno de los canales necesarios, las cuales deben ser llamadas antes de la rutina RCAD para seleccionar el canal de conversión. Las tres rutinas se llamaron respectivamente: CANAL0, CANAL1, CANAL2, las cuales forman parte de las rutinas de adquisición. Para comprender mejor su cometido el código correspondiente a la rutina CANAL 1 es mostrado en la Figura 3.42.

Rutina para seleccionar el canal 1 del modulo CA/D

```
CANAL1:

        BANKSEL  ADCON0   ; SELECCIONA EL BANCO DEL REGISTRO ADCON0
        BSF      ADCON0,3 ; PONE CHS2:CHS0<5:3> = B'001'
        BCF      ADCON0,4
        RETURN
```

Figura 3.42 Código de la rutina para seleccionar el canal 0 del CA/D.

Las otras dos rutinas son similares, pero ponen a los bits CHS2:CHS0<5:3> a los valores correspondientes para seleccionar los canales correspondientes.

La segunda característica de la rutina RCAD, es que esta, utiliza el método de poleo para verificar el fin de conversión, método seleccionado debido a su sencillez, cuando se realizan conversiones A/D sucesivas.

Finalmente se debe mencionar que en la rutina RCAD no hay un segmento de programa que implemente el tiempo de espera de adquisición, esto es a causa de que entre la conversión de un canal y otro y entre conversiones del mismo canal se ejecutan otras rutinas que son propias del algoritmo de control difuso, la cuales toman tiempo suficiente o incluso mas del necesario, para cubrir el tiempo de adquisición.

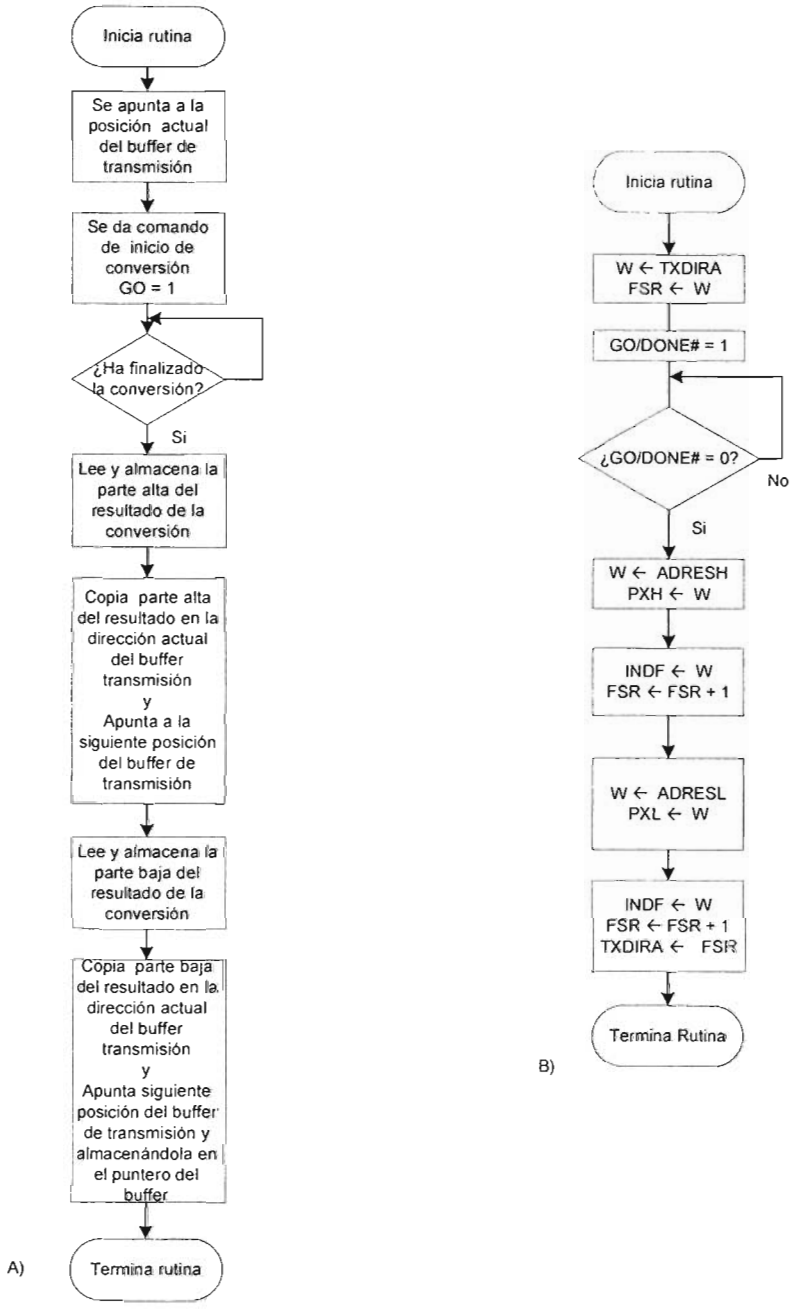


Figura 3.41 Diagramas de flujo de la rutina de conversión analógica digital "RCAD", A) Diagrama de flujo explicativo, B) Diagrama de flujo a nivel ensamblador.

3.3.7.3.1 Variable de estado del proceso controlado como entrada directa al controlador difuso.

Como se dijo en el apartado 3.3.7.1, a veces cuando la información sobre el objeto o proceso bajo control y su estructura están disponibles, uno puede no querer estar confinado a usar el error, cambio de error, suma de error como variables de estado del proceso, si no mas bien usar las variables de estado del proceso actual (ver Figura 3.35). En este caso el resultado directo de la conversión analógica digital de dicha variable puede ser utilizado como entrada crisp al controlador difuso, y no es necesario ningún otro tipo de manipulación. Por lo tanto para generar este tipo de entrada para un controlador solo basta usar la rutina RCAD, que almacena el resultado de la conversión en el par de registros PXH y PXL (PX), de donde la rutina de fuzzificaciòn tomo el valor crisp de entrada para producir las entradas difusas.

3.3.7.3.2 Señal de error como entrada al controlador difuso.

La señal de error se define como $e(t) = r - y(t)$, donde r es la referencia (set point) o señal de salida deseada y $y(t)$ es la señal de salida actual del proceso u objeto bajo control. La señal de error es generada por software en el microcontrolador y se obtiene de la siguiente manera:

PX = ***SETP*** – ***PX***..... (3.13)

Donde PX es la variable donde se deposita el resultado de la conversión A/D y SETP (formada por la concatenación de los dos bytes SEPH: SETPL) es una variable definida en RAM para almacenar la referencia o set point del sistema. Como puede verse el resultado de la diferencia se vuelve a depositar en PX, ya que de esta, la rutina de fuzzificaciòn correspondiente toma el valor crisp de entrada para producir las entradas difusas.

Entonces los pasos que se siguen para generar la señal de error son:

a) Hacer la conversión A/D de la salida del proceso u objeto bajo control, esto se hace llamado a la rutina RCAD, solo que previamente llamando a la rutina de selección de canal adecuada. El resultado es depositado en la variable de dos bytes PX.

b) Restar el resultado de la conversión A/D o PX del valor del set point (SETP), para obtener la señal de error, dejando el resultado en PX.

La rutina encargada de hacer la diferencia entre el set point y el resultado de la conversión A/D de la salida del proceso u objeto bajo control se llamo SERROR. Esta rutina también debe copiar la *señal de error* resultante al buffer de transmisión de donde posteriormente es enviada a la IPVSID para su monitorización continua. El diagrama de flujo de dicha rutina es mostrado en la Figura 3.43.

Cabe mencionar una última consideración acerca de la generación de la señal de *error*, de la ecuación 3.13, si el set point es menor que el resultado de la conversión, el cual tiene una rango de 0 a 2^{10} = 1023, el resultado de la diferencia será negativo, para evitar el manejo de números negativos en las operaciones

aritméticas, al valor de SETP correspondiente a un set point, se le suman 511_D. Debido a que SETP es un parámetro configurable que podrá ser cambiado a voluntad, el valor adicional es incluido desde que es enviado de la IPVSID, reduciendo el numero de operaciones aritméticas necesarias en la rutina, y por lo tanto el tiempo de procesamiento.

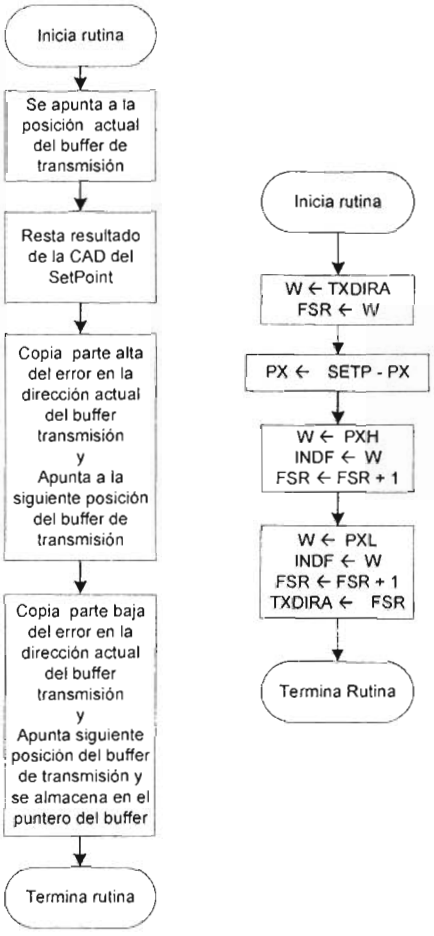


Figura 3.43 Diagrama de flujo de la rutina ***SERROR.***

3.3.7.3.3 Señal de cambio de error como entrada al controlador difuso.

La señal de cambio de error se define como $\Delta e(t) = e(t) - e(t-1)$, donde $e(t)$ es el error para el tiempo de muestreo actual y $e(t-1)$ es el error para un tiempo de muestreo previo. Para generar la señal de cambio de error no se necesita realizar una conversión A/D, pero si se necesita previamente haber generado una señal de error. Para generar la señal de cambio de error se necesita un medio para almacenar el error para un tiempo de muestreo anterior, para llevar acabo dicho

procedimiento se definieron dos variables de dos bytes en memoria RAM, que se llamaron ER1 y ER0, donde la primera se usa para almacenar el valor del error actual y que en el siguiente muestreo es copiado a la segunda que ahora será el error para tiempo de muestreo previo.

La señal de cambio de error se obtiene de esta manera

$$PX = PX - ER0 + 511 \quad \ldots\ldots\ldots (3.14)$$

Donde PX contiene el error para el muestreo actual y ER0 el error para un tiempo de muestreo previo, el resultado se vuelve a dejar en PX (por las mismas razones que la señal de error (ver apartado anterior) se deja en PX). El numero 511 se añade por la mismas razones mencionadas en el apartado anterior.

La rutina que genera la señal de *cambio de error* se llamó SCERROR, su diagrama de flujo es mostrado en la Figura 3.44, del mismo se puede ver que la rutina también debe copiar el valor actual de la señal *cambio de error* al buffer de transmisión, de donde es enviado a la IPVSID para su monitorización continua.

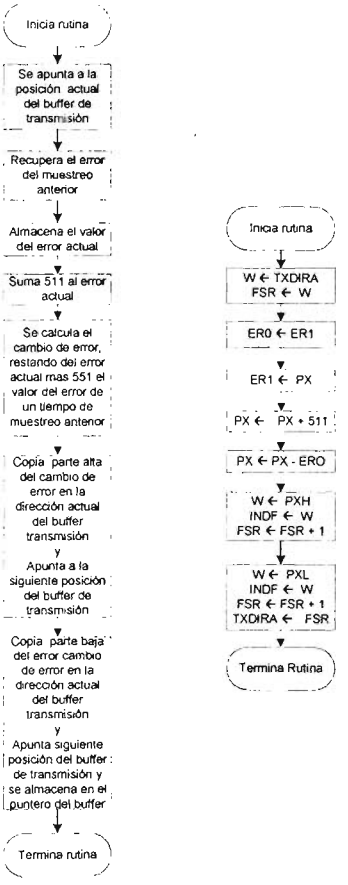


Figura 3.44 Diagrama de flujo de la rutina ***SCERROR***.

3.3.7.3.4 Señal de referencia como entrada al controlador difuso.

Este tipo de entrada es muy sencilla de generarse, ya que, simplemente es el valor del Set Point actual, por los tanto, si como se había mencionado, el valor del (ver apartado 3.3.7.3.2) Set Point se recibe con un incremente de 511, para obtener el valor de la referencia solo es necesario restarle estos 511, es decir.

PX = **SETP** – 511..... (3.15)

La rutina que genera la *señal de referencia* se llamo REFERENCIAP (referencia prealimentada). El valor se deposita en PX por la mimas razones previamente explicadas para la señal de error y cambio de error. Esta ultima rutina también copiar el valor actual de la *señal de referencia* al buffer de transmisión, de donde es enviado a la IPVSID para su monitorización continua.

Con la explicación de esta última rutina finaliza lo referente al diseño de las rutinas de adquisición de datos, de las cuales, cuatro son las rutinas encargadas de generar las entradas crisp al controlador difuso:

- RCAD
- SERROR
- SCERROR
- REFERENCIAP.

Y tres las rutinas auxiliares:

- CANAL0
- CANAL1
- CANAL2

En los siguientes apartados se explica el proceso de diseño de las tres fases de operación del controlador difuso.

3.3.7.4 Controlador Difuso.

En la Figura se observa la estructura general de un controlador difuso, la cual esta dividida en tres operaciones fundamentales: La fuzzificaciòn, la evaluación de reglas y la defuzzificaciòn, que hacen uso de la base de conocimiento del proceso u objeto a controlar, para que, a partir del las entradas del controlador difuso (variables de estado del proceso) se produzcan las correspondientes acciones o salidas de control (señales de control) para llevar a la salida del proceso al estado deseado. En la sección previa se ha mostrado como se generan la entradas crisp para un controlador difuso, ahora en esta sección y las respectivas subsecciones se explicaran el proceso de diseño de los algoritmos de las tres fases de operación del controlador difuso.

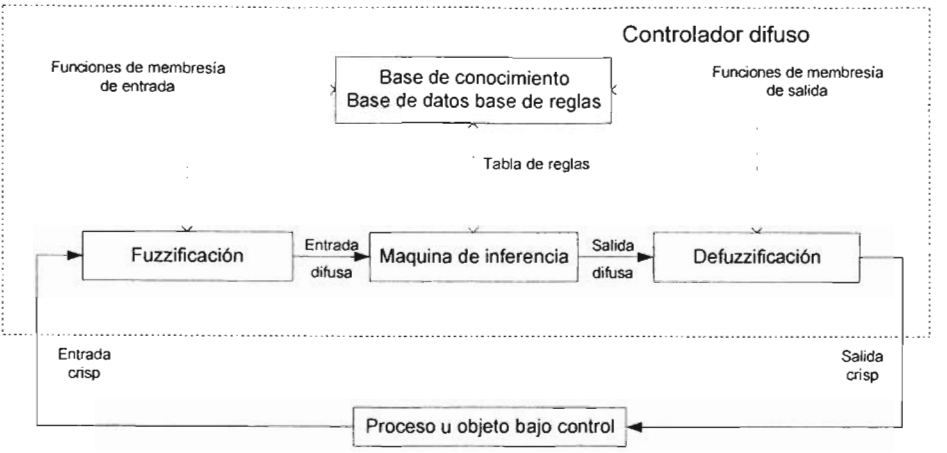


Figura 3.45 Estructura básica de un controlador difuso.

Un controlador difuso tiene dos partes, la primera parte es el núcleo de inferencia difuso que comprende las operaciones de fuzzificaciòn, evaluación de reglas y defuzzificaciòn, el cual es ejecutado periódicamente para determinar las salidas del controlador basados en las entradas actuales del controlador. La segunda parte del controlador es una base de conocimiento que contiene la definición de funciones de membresía y la base de reglas, que en conjunto representan el conocimiento ganado acerca de un proceso u objeto a controlar. Un controlador difuso basado en microcontrolador, debe de tener estas dos partes. La Figura 3.46 es un diagrama de bloques de esta clase de sistema de lógica difusa.

La base de conocimiento puede ser desarrollada por un experto en la aplicación sin ninguna experiencia en programación de microcontroladores. Las funciones de pertenencia o membresía son simplemente expresiones del entendimiento del experto de los términos lingüísticos que describen al sistema a ser controlado.

Las reglas son postulados en lenguaje ordinario ya que describen las acciones que un humano experto tomaría para resolver el problema de la aplicación. Las reglas y funciones de membresía pueden ser reducidas a estructuras de datos relativamente simples (la base de conocimiento) almacenados en memoria no volátil (EEPROM o FLASH). Un núcleo de inferencia difusa puede ser escrito por un programador quien no conoce como trabaja el sistema de la aplicación. La única cosa que el programador necesita hacer con la información de la base de conocimiento es almacenarla en localidades de memoria usadas por el núcleo.

Una ejecución del núcleo de inferencia difusa genera señales de salida del sistema en respuesta a las condiciones actuales de entrada. El núcleo es ejecutado periódicamente, tanto como sea necesario para mantener el control. Si el núcleo es ejecutado mas rápido de lo necesario, el ancho de banda y la potencia del procesador se desperdician; en caso contrario puede causar que el sistema se salga de control. La elección de la tasa periódica para un sistema de control difuso es igual a la de un sistema de control convencional (ver sección de diseño del control de la frecuencia de muestreo).

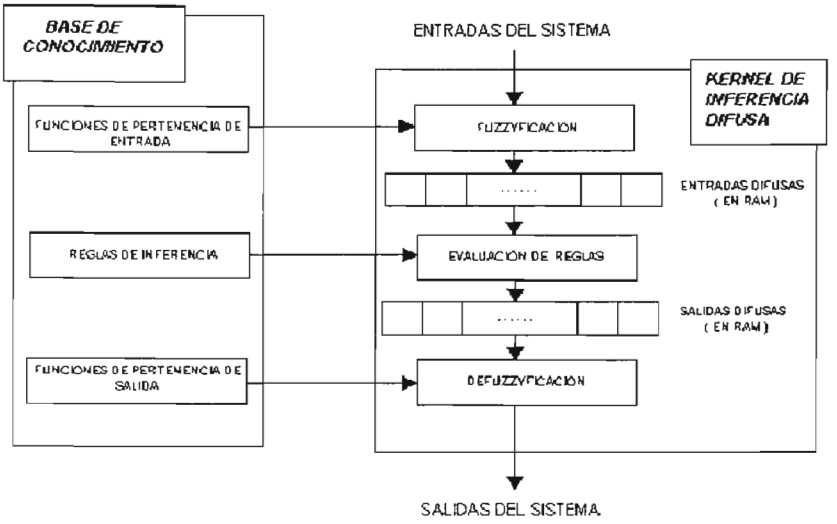


Figura 3.46. Diagrama de bloques de un sistema de lógica difusa basado en microcontrolador.

3.3.7.4.1 Fuzzificación.

Durante el paso de fuzzificaciòn, los valores de entrada actuales del sistema son comparados contra las funciones de membresía de entrada almacenadas para determinar el grado en que cada etiqueta de entrada del sistema es verdad. Esto se logra hallando el valor ***y*** (grado de membresía) para el valor actual de entrada en una función de membresía trapezoidal (única función permitida en el SID) para cada etiqueta de cada entrada del sistema.

Para realizar este cálculo para una etiqueta de una entrada del sistema se diseño una rutina llamada GDM (Grado de Membresía). Y para realizar todo el proceso de fuzzificación completo se diseño una rutina llamada RFUZZIFI, que llama a la rutina GDM para cuantas etiquetas tenga una entrada del sistema.

En la Figura 3.47 se exhibe un sistema de tres funciones de membresía de entrada, una para cada etiqueta o valor lingüístico para una variable lingüística de temperatura (entrada al sistema). El eje ***x*** de las tres funciones de membresía representa el rango de valores posibles de la entrada del sistema. La línea vertical a través de la tres funciones de membresía representa un valor especifico (puntual o crisp) de entrada del sistema. El eje ***y*** representa el grado de membresía o de verdad y varia desde completamente falso (0 ó 0x000) hasta completamente verdadero (1 ó 0x3FF).

El rango de 0x000 hasta 0x3FF se debe a que el rango del CAD tiene un ancho de palabra de 10 bits.

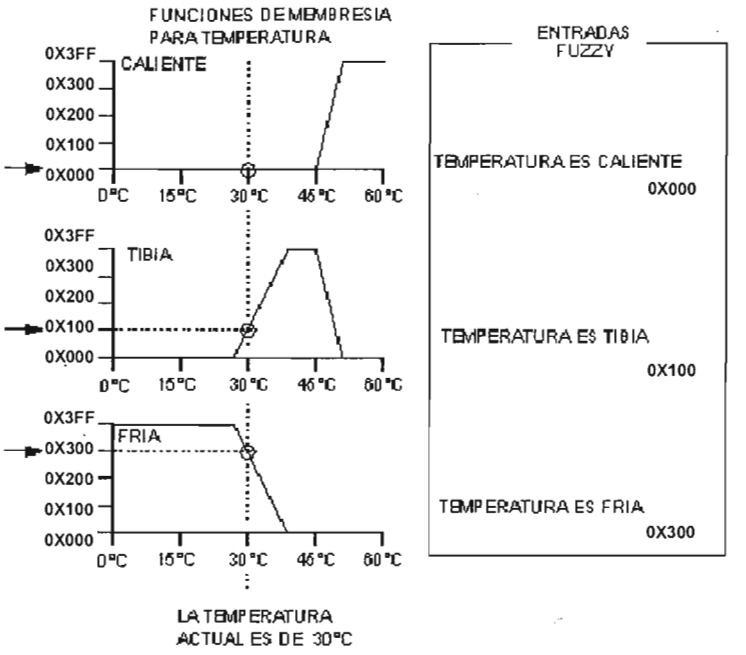


Figura 3.47 fuzzificación mediante funciones de membresía.

El valor ***y*** donde la línea vertical intercepta a cada una de las funciones de membresía, es el grado al cual el valor actual de entrada concuerda con la etiqueta asociada para esta entrada del sistema. Por ejemplo, la expresión "la temperatura es tibia" es 25% verdad (0x100). El valor 0x100 es almacenado en 2 localidades RAM (10 bits), y es llamado una entrada difusa (en este caso, la entrada difusa para "la temperatura es tibia"). Hay 2 localidades RAM para cada entrada difusa (por cada etiqueta de cada entrada del sistema).

Cuando el paso de fuzzificación empieza (llamando a la rutina RFUZZIFI), el valor actual de la entrada al sistema esta en una variable RAM llamada PX (PX es la localidad de memoria RAM donde las rutinas que generan los tipos de entradas dejan su resultado, ver apartados 3.3.7.3.1, 2, 3, 4), los registros de direccionamiento (apuntadores) de la memoria de la memoria FLASH de programa apuntan a la primera definición de función de membresía en la base de conocimiento, y una apuntador a la primera entrada difusa en RAM. En cuanto, cada entrada difusa es calculada el ejecutar la rutina GDM, el resultado es almacenado en la entradas difusas y durante este proceso ambos apuntadores son actualizados para apuntar a las localidades asociadas con la siguiente entrada difusa si es que hay. La rutina RFUZZIFI se encarga de todo excepto de cargar el numero de etiquetas por entrada del sistema y de cargar el valor actual de cualquiera entrada subsiguiente del sistema.

Antes de realizar cualquier fuzzificación de alguna entrada del sistema por medio de la rutina RFUZZIFI, se debe llamar a una rutina llamada INIFUZZIFI, que no hace otra cosa que cargar el valor inicial del apuntador a la definición de las funciones de membresía de cada entrada y el del apuntador a las entradas difusas.

El resultado final del paso de fuzzificación es una tabla de entradas difusas representando las condiciones actuales del sistema.

3.3.7.4.1.1 Diseño de la rutina RFUZZIFI

La rutina RFUZZIFI debe llamara la rutina GDM un numero de veces igual el numero de etiquetas lingüísticas definidas para una entrada del sistema. El diagrama de flujo de esta rutina es mostrado en la Figura 3.48.

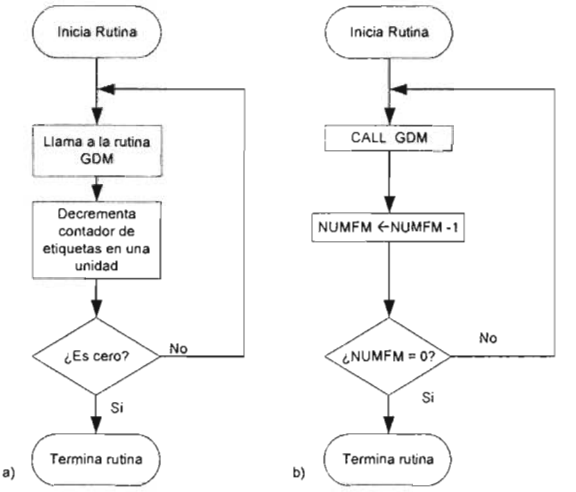


Figura 3.48 Diagrama de flujo de la rutina RFUZIFI.

Se observa que esta rutina simplemente repite la llamada a la rutina GDM para el número inicial especificado en NUMFM (variable para contar el número de etiquetas lingüísticas para una entrada). El valor de NUMFM debe ser previamente cargado con el número de funciones de membresía para una entrada, esta operación es realizado por una rutina llamada RNUMCDE.

La rutina RNUMCDE maneja un apuntador (APFME) que apunta a una serie de localidades sucesivas de memoria RAM donde se almacenan el número de conjuntos difuso para cada entrada del sistema, como el SID esta restringido para tener un máximo de tres entradas, las localidades de memoria se reducen al mismo numero, estas localidades permanecen al mismo valor para un controlador difuso diseñado, y forman parte de las constantes de control que se cargan con la rutina CARGAR_CONSTANTES_DE_CONTROL (ver apartado 3.3.6.1), y se llaman respectivamente NFPE1 NFPE2, NFPE3. Entonces, lo que hace la rutina RNUMCDE es cargar el valor contenido en cada uno de estos registros a la

variable NUMFM, para la entrada respectiva. El la Figura 3.49 se muestra la estructura de estas localidades en RAM.

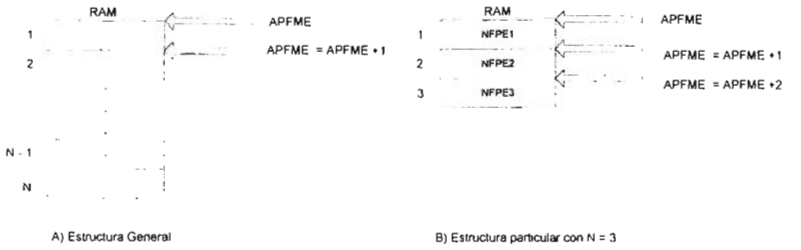


Figura 3.49 Estructura de la tabla de número de etiquetas por entrada.

La estructura de estas rutinas permite extender el proceso de fuzzificación a cuantas entradas se deseen siempre y cuando la memoria y la potencia del procesador lo permitan, para el SID, este número se restringió a un máximo de tres.

Entonces lo que se debe hacer en orden para fuzzificar una o mas entradas es:

1. Llamar a la rutina respectiva de generación de entradas crisp del sistema, las cuales son seleccionadas entre: RCAD, SERROR, SCERROR y REFERNCIAP, dependiendo del esquema de control que se considere, en cualquier caso cada una de ellas coloca su resultado en PX, que representa el valor actual para esa entrada.
2. Llamar a la rutina INIFUZZIFI, que inicializa lo punteros a la definición de funciones de membresía y las entradas difusas en RAM.
3. Llamar a la rutina RNUMCDE, para cargar el número de etiquetas para la entrada a fuzzificar.
4. Llamar a la rutina RFUZZIFI, que completa la operación de fuzzificación para la entrada, y produce una tabla de entradas difusas en RAM, para ser usadas por el proceso de evaluación de reglas.

Como se había dicho previamente, la rutina que propiamente calcula el valor de verdad (grado de membresía) del valor actual de una entrada a una función de membresía trapezoidal de una etiqueta de esa entrada del sistema se llamo GDM (grado de membresía), la cual es descrita en el siguiente apartado.

3.3.7.4.1.2 Diseño de la rutina GDM.

Los tipos de funciones de membresía que pueden tener la etiquetas lingüísticas de una variable lingüística pueden tener varias formas las mas comunes son formas rectilíneas como funciones de membresía trapezoidales y triangulares; o formas no lineales como funciones gaussianas, cuadráticas o exponenciales. En general no hay un criterio para determinar que funciones de membresía son mejores.

Tomando en cuenta la potencia del procesador del sistema la selecciones mas adecuadas son formas rectilíneas ya suponen un calculo del grado de membresía mas sencillo.

La función de membresía rectilínea más general es la trapezoidal, ya que con una adecuada asignación de los valores de sus parámetros se pueden generar otros tipos de funciones de membresía. En la Figura 3.50 se muestre este tipo de función de membresía y las funciones de membresía que se pueden derivar de ella.

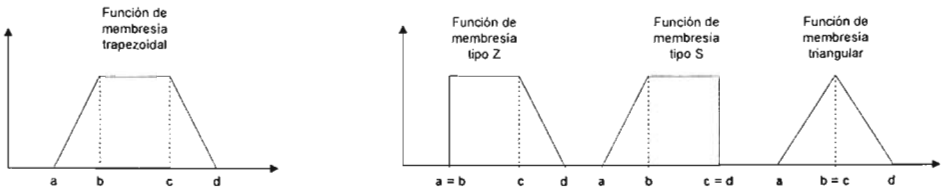


Figura 3.50 Función de membresía trapezoidal y funciones de membresía derivadas.

Una función de membresía trapezoidal queda perfectamente definida asignado un valor a cada uno de sus cuatro parámetros.

Pero para especificar una función de membresía trapezoidal en el microcontrolador se opto por una definición diferente que es ilustrada en la Figura 3.51, esto se debió a que permite la implementación de un algoritmo de fuzzificación más sencillo (reduce el número de operaciones aritméticas, de multiplicación y división las cuales no están definidas en el PIC y deben ser implementadas por programa). Esta operación será transparente al usuario, puesto que la definición de las funciones de membresía se lleva acabo desde un editor de funciones de membresía en la IPVSID, donde esta se hace de manera normal, definiendo los parámetros a, b, c y d de la función de membresía, generándose por programa la definición utilizada por el microcontrolador.

La rutina GDM calcula el valor ***y*** donde la entrada actual intercepta a la función de membresía.

Las funciones de membresía deben tener las siguientes características:

- $0x000 \leq punto\_1(PA) \leq 0x3FF$
- $0x000 \leq punto\_2(PB) \leq 0x3FF$
- $punto\_1 < punto\_2$.
- Los lados verticales del trapezoide se ponen a 0 (pendiente indefinida).

Puesto que el valor de pendiente 0 no se usa de otra manera, el es asignado para indicar una pendiente infinita.

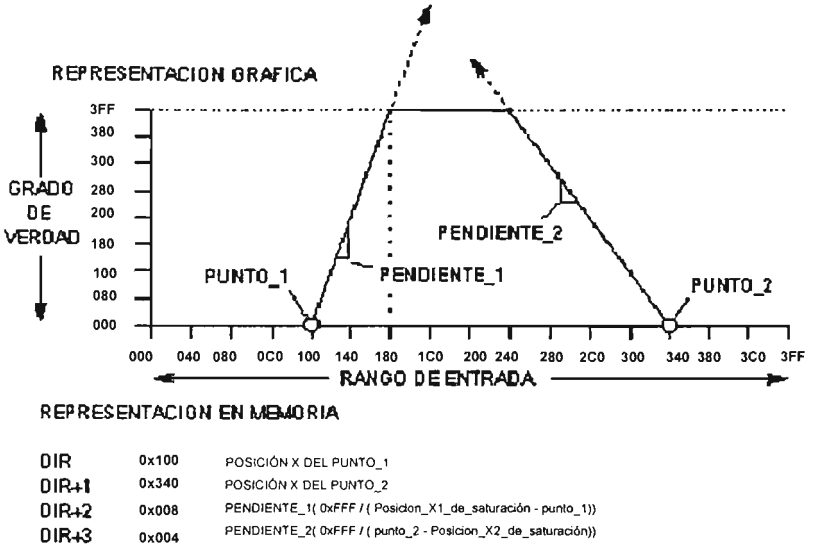


Figura 4.51 Definición de una función de membresía en el microcontrolador.

El diagrama de flujo de la operación general de la rutina GDM es mostrado en la Figura 3.52, donde VM hace referencia al valor de membresía, PA y PB son el primero y segundo punto del soporte del la función de membresía, M1 y M2 son la primera y segunda pendientes de la función de membresía.

Los dos primeros rombos de selección determinan si el valor actual de entrada PX se encuentra dentro de la base o soporte de la función de membresía en cuestión, de no ser así el grado de membresía que tendría dicha entrada a la etiqueta asociada a dicha función de membresía seria cero (VM = 0).

Enseguida se determina si la primera pendiente de la función de membresía trapezoidal es infinita, de ser así, se asume que el valor de membresía es uno (VM = 1), en caso contrario, se calcula VM para la primera pendiente de la FM (VM1). Si VM1 resulta mayor de uno, quiere decir que el valor de PX se encuentra mas allá del intervalo de valores cubiertos por la primera pendiente (M1), por lo que asume que como máximo puede tener un valor de uno (VM = 1), en caso contrario el valor resultante se considera un valor valido de membresía (VM = VM1). En cualquiera de los tras casos anteriores, para determinar si el valor de membresía actual hasta este punto es definitivo, se verifica si la segunda pendiente es infinita, de ser así, el valor de membresía actual es el definitivo y el algoritmo finaliza, en caso contrario, el valor de PX, puede estar dentro del intervalo de valores cubierto por la segunda pendiente (M2), por lo que calcula el valor de membresía correspondiente (VM2), si este ultimo valor es mayor que uno, se toma como valor de membresía definitivo al valor previo, en caso contrario, el ultimo valor es el definitivo (VM = VM2) y el algoritmo finaliza.

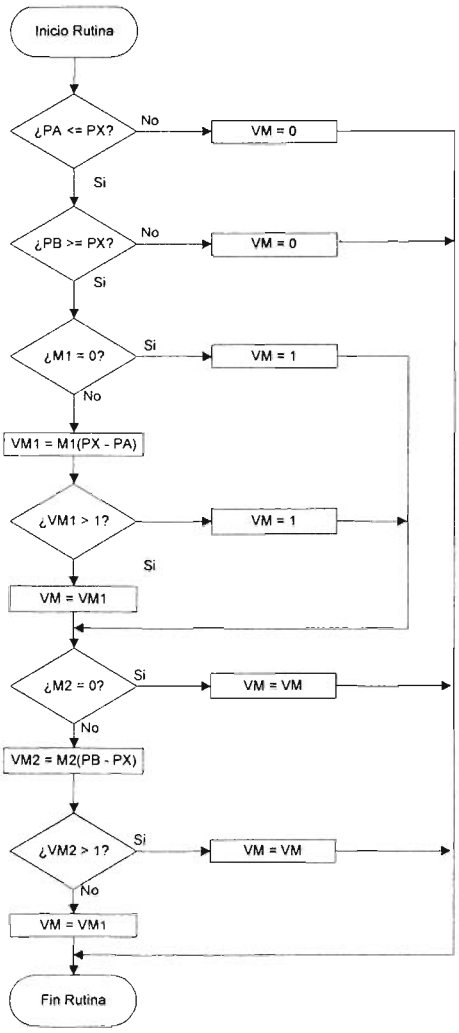


Figura 3.52 Diagrama de flujo de la operación de la rutina GDM que calcula el valor de membresía de una entrada puntual a una etiqueta lingüística de una variable lingüística.

3.3.7.4.2 Inferencia Difusa.

La evaluación de reglas es el elemento central de un programa de inferencia de lógica difusa. Este paso procesa una lista de reglas de la base de conocimiento usando los valores actuales de entradas difusas en RAM para producir una lista de salidas difusas en RAM. Dichas salidas difusas pueden considerarse como sugestiones algo burdas de lo que la salida del sistema debería ser en respuesta a las actuales condiciones de entrada. Antes que los resultados puedan aplicarse, las salidas difusas deben de procesarse posteriormente, o ser defuzzificadas, para

producir un solo valor de salida que representa el efecto combinado de todas las salidas difusas.

Para la evaluación de reglas se diseñaron dos rutinas una que se llamó REVA_REG_FLASH y otra REVA_REG_EEPROM, la primera evalúa un conjunto de reglas definidas en la memoria FLASH y la segunda en la memoria EEPROM. La principal razón de haber decidido tener dos rutinas para la evaluación de reglas es debido a que la memoria EEPROM tiene un número reducido de 256 bytes disponibles para codificar las reglas, en contrate con la memoria FLASH del la cual quedan disponibles para datos mas de 4K palabras de 14 bits. Considerando el peor caso, si la definición de una regla en memoria EEPROM utiliza 5 bytes (como se vera mas adelante) para codificar una reglas difusa de un controlador de tres entradas y una salida, y tomando en cuenta que se pueden definir un máximo de 9 conjuntos difuso por entrada, entonces necesitaríamos definir como máximo $9 \times 9 \times 9 = 729$ reglas, con lo que se requerirían un máximo de $729 \times 5 = 3645$ bytes que seria imposibles de cubrir con la memoria EEPROM. De esta manera el máximo número de reglas que se pueden cubrir por la memoria EEPROM es de 63, si cada una de ellas es codificada con 4 bytes, que correspondería con un controlador con dos entradas y una salida, con un máximo de 8 conjuntos difusos por cada entrada. De esta manera la propia aplicación se diseñó de tal manera que esta situación fuera transparente al usuario y ella misma decide cual de los dos rutinas utilizar de acuerdo a la cantidad de bytes de memoria requeridos para codificar la reglas en memoria EEPROM.

Auque el acceso (lectura) a la memoria FLASH y memoria EEPROM es muy similar, el primero es mas lento que el segundo, y debido a la diferencia en el ancho de palabra de la memoria, la codificación de las reglas en cada tipo de memoria fue un poco diferente, sin embargo la operación general de la evaluación de reglas funcionan exactamente igual.

La comprensión de la estructura y la sintaxis de las reglas es necesaria para entender cómo un microcontrolador realiza la tarea de evaluación de reglas. La siguiente es un ejemplo de una regla típica.

Si la temperatura es tibia y la presión es alta entonces el calor es (debería ser) apagado.

A primera vista, parece que codificar esta regla en una forma compacta comprensible para el microcontrolador sería difícil, mas es realmente simple reducir la regla a una lista breve de apuntadores de memoria. La porción izquierda de la regla es un postulado de condiciones de la entrada y la porción derecha de la regla es un postulado de acciones de salida. La porción izquierda de una regla está formada de uno o más (en este caso dos) antecedentes conectados por un operador difuso ***Y***. Cada expresión antecedente consiste del nombre de la entrada del sistema, seguido por **es**, seguido por un nombre de etiqueta. La etiqueta debe estar definida mediante una función de membresía en la base de conocimiento. Cada expresión antecedente corresponde a una de las entradas difusas en RAM. Puesto que ***Y*** es el único operador permitido para conectar expresiones antecedentes, no hay necesidad de incluir a éstos en la codificación de la regla.

Los antecedentes pueden codificarse como una simple lista de apuntadores hacia (o direcciones de) las entradas difusas a las cuales ellos hacen referencia.

La porción derecha de una regla se forma de una o más (en este caso una) consecuencias. Cada expresión de consecuencia consiste del nombre de una salida del sistema, seguido por ***es***, seguido por un nombre de etiqueta. Cada expresión de consecuencia corresponde a una salida difusa específica en RAM. Las consecuencias para una regla pueden codificarse como una simple lista de apuntadores a (o direcciones de) las salidas difusas a las cuales hacen referencia.

Las reglas completas son almacenadas en la base de conocimiento como una lista de apuntadores o direcciones de entradas difusas y salidas difusas. Para que trabaje la lógica de evaluación de reglas, debe haber algunos medios de saber cuales apuntadores se refieren a entradas difusas, y cuáles se refieren a salidas difusas.

También debe haber una manera de saber cuando la última regla del sistema ha sido alcanzada. Un método de organización es tener un número fijo de reglas con un número específico de antecedentes y consecuencias (un sistema de arquitectura cerrada). Un segundo método, que es el empleado en la rutina REVA_REG_FLASH, es marcar el fin de la lista de reglas con un valor reservado, y usar un bit en los apuntadores para distinguir antecedentes de consecuencias. Un tercer método de organización, usado en la rutina REVA_REG_EEPROM, es marcar el fin de la lista de reglas con un valor reservado, y separar antecedentes y consecuencias con otro valor reservado. Ello permite cualquier número de reglas, y permite a cada regla tener cualquier número de antecedentes y consecuencias, sujeto a los límites impuestos por la disponibilidad de memoria en el sistema y velocidad de procesamiento.

Cada regla es evaluada secuencialmente, pero las reglas como grupo son tratadas como si ellas fueran todas evaluadas simultáneamente. Dos operaciones matemáticas toman lugar durante la evaluación de reglas. El operador difuso ***Y*** que corresponde a la operación matemática de hallar el mínimo y la operación difusa ***O*** que corresponde a la operación matemática de hallar el máximo. La operación difusa ***Y*** es usada para conectar antecedentes dentro de una regla. La operación difusa ***O*** está implícita entre reglas sucesivas. Antes de evaluar cualquier Regla, todas las salidas difusas se ponen a cero (lo que significa nada de verdad). Cuando cada regla es evaluada, el más pequeño (el mínimo) antecedente es tomado para ser la verdad global de la regla. Este valor de verdad de la regla es aplicado a cada consecuencia de la regla (al almacenar este valor a la correspondiente salida difusa) a menos que la salida difusa sea ya más grande (máxima). Si dos o más reglas afectan a la misma salida difusa, la regla que es más verdadera gobierna el valor en la salida difusa porque las reglas se conectan por un difuso **O** implícito.

El resultado final del paso de evaluación reglas es una tabla de salidas difusas sugeridas o "crudas" en RAM. Estos valores fueron obtenidos al alimentar las condiciones actuales (valores de entradas difusas) en las reglas del sistema en la base de conocimiento. Estos resultados crudos no pueden ser suministrados

directamente a las salidas del sistema porque ellos pueden ser ambiguos. Por ejemplo, una salida cruda puede indicar que la salida del sistema debería ser media con a grado de verdad de 50% mientras, al mismo tiempo, otra indica que la salida del sistema debería ser baja con a grado de verdad de 25%. El paso de defuzzificación (el cual se describe en apartado 3.3.7.4.3) resuelve tales ambigüedades.

Estas rutinas implementan la evaluación básica min-max de reglas. Los registros en RAM son usados como apuntadores y para resultados intermedios de los cálculos. Puesto que las rutinas REVA_REG_FLASH y REVA_REG_EEPROM esencialmente procesan listas, el tiempo de ejecución es dependiente del número de elementos en la lista de reglas. Para realizar las operaciones ***Y*** y ***O*** difusas se diseñaron dos rutinas respectivamente. La rutina que implementa la operación ***Y*** difusa, obtiene el mínimo de dos operados de 10 bits que deben ser previamente cargados con sus respectivos valores y coloca el mínimo de ellos en dos localidades RAM que se llamaron RMINH y RMINL (RMIN para abreviar RMINH:RMINL). Esta rutina solo se ejecuta para reglas de más de dos antecedentes pues para reglas de un solo antecedente solo se tiene un operando el cual es colocado directamente en RMIN. La operación **O** difusa obtiene el máximo de dos operandos uno de los cuales es RMIN (el resultado del procesamiento de la parte antecedente de la regla) y el otro es el valor previo de la salida difusa a la cual hace referencia la regla en cuestión, el resultado de esta operación es colocado en el par de registros RMAXH:RMAXL (abreviado RMAX), y al final de la evaluación de la regla el resultado (RMAX) es colocado en la misma salida difusa.

Para apuntar a las entradas y salidas difusas se usa un direccionamiento indirecto, cuyas direcciones están dadas por las mismas reglas. Las salidas difusas (localidades de trabajo en RAM) necesitan ser puestas a 0x000. Si estos valores no son iniciados antes de la ejecución de rutinas de evaluación de reglas, los resultados serán erróneos.

Por lo tanto antes de ejecutar las rutinas de evaluación de reglas se debe llamar respectivamente a un par de rutinas que fueron llamadas RINIEVAREG_FLASH y RINIEVAREG_EEPROM, las cuales forman parte de las rutinas de inferencia difusa (ver apartado 3.3.7.1). Las acciones que realizan estas rutinas son:

- Seleccionar la memoria EEPROM o memoria FLASH respectivamente.
- Inicializar respectivamente los registros de direccionamiento EEADRH y EEADR de la memoria EEPROM y memoria FLASH con el valor del par de constantes de control ABREEH y ABREEL (ver apartado 3.3.6.1) que apuntan a la definición de la primera regla de la base de reglas.
- Borrar las salidas difusas, es decir ponerlas a cero antes de que se evalué regla alguna.

EL par de registros EEADRH y EEADR son actualizados por la rutina de evaluación de reglas para apuntar a cada parte de una regla, que es accedida a

través del registro EEDATA y en base a cuyo valor se hace el procesamiento respectivo.

En el registro FSR (apuntador en el direccionamiento indirecto) se pone la dirección para apuntar a las entradas y salidas difusas (en RAM de trabajo) que es copiada de EEDATA, la cual puede apuntar a un antecedente o un consecuente según la parte de la regla que se este procesando. Cada antecedente de regla es una dirección RAM de de 8 bits desde la dirección base a la entrada difusa referida. Cada consecuencia de regla es una dirección RAM de 8 bits desde la dirección base a la salida difusa referida.

La operación de la evaluación de reglas, así como la codificación de una regla en memoria EEPROM es mostrada en la Figura 3.53.

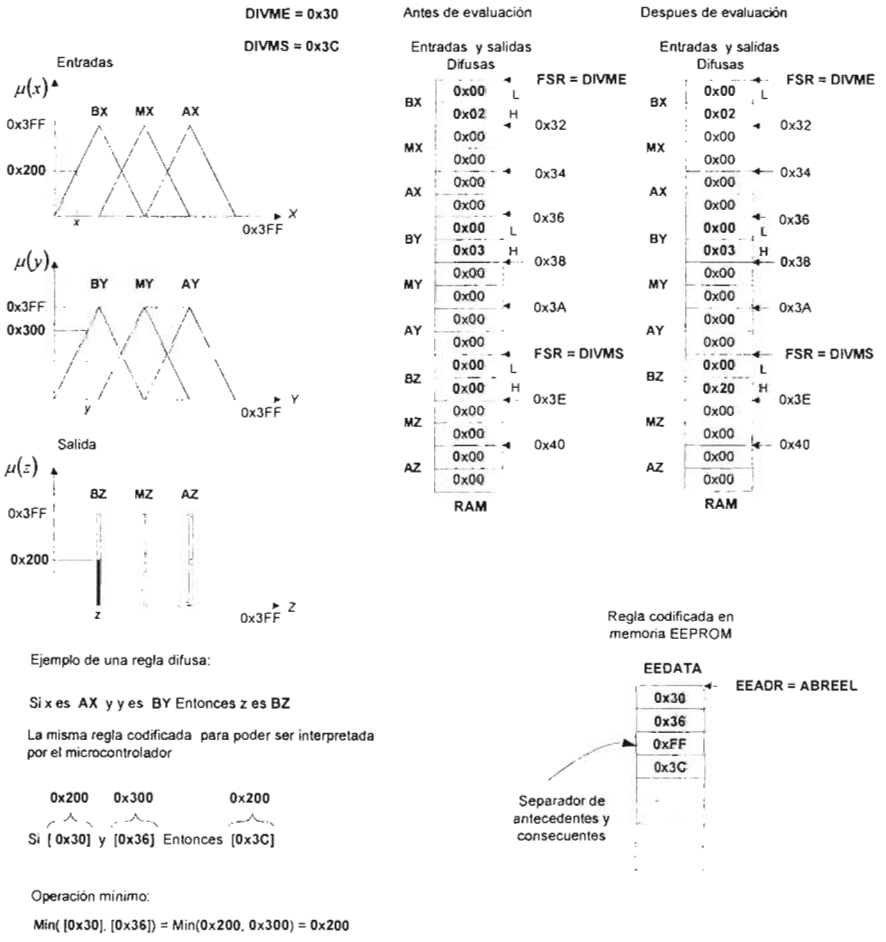


Figura 3.53 Estructura de la evaluación de reglas y la codificación de una regla en memoria EEPROM para un controlador de 2 entradas y una salida, con tres funciones de membresía cada una.

3.3.7.4.3 Defuzzificación.

El paso final en el programa de lógica difusa combina las salidas difusas crudas en una salida compuesta del sistema. A diferencia de las formas trapezoides usadas para las entradas, la salida típicamente usa barras (singletons) para funciones de membresía de salida (inferencia tipo Sugeno). Al igual que con las entradas, el eje *x* representa el rango de posibles valores para la salida del sistema. Las funciones de pertenencia tipo barra (Singleton) consisten de las posiciones en el eje *x* para cada etiqueta de la salida del sistema. Las salidas difusas corresponden a la altura o peso en el eje *y* de la correspondiente función de membresía de salida.

El algoritmo de defuzzificación se basó en la fórmula de promedio de pesos para un controlador tipo Sugeno, es decir con conjuntos singleton escalares definidos en la variable lingüística de salida y se expresa como:

$$U = \frac{\sum_{i=1}^{n} w_i * u_i}{\sum_{i=1}^{n} w_i} \quad \text{..... (3.16)}$$

Donde *n* es el número de etiquetas de la salida del sistema, u_i son las posiciones de los singleton de la base de conocimiento, y w_i son las salidas difusas (pesos) en RAM, y U la salida puntual o crisp de la inferencia difusa.

Para un controlador difuso diseñado en el SID, *n* puede tomar hasta un valor máximo de 9, u_i y w_i son valores de 10 bits. El valor U también es un valor de 10 bits.

Para procesar esta ecuación en el microcontrolador se definieron tres variables, la primera para acumular la sumatoria del numerador y la segunda para acumular la sumatoria del denominador y la tercera para almacenar el resultado de la división de estas dos sumatorias. Las variables definidas para guardar el resultado de dichas operaciones fueron ADD, SUM y SALIDA respectivamente. En base a estas variables el diagrama de flujo de la rutina de defuzzificación diseñada y llamada RDEFUZZY es mostrado en la Figura 3.54.

El ciclo se ejecuta tantas veces como conjuntos o etiquetas lingüísticas tenga la salida, este número es indicado por n, pero que en la rutina RDEFUZZY fue sustituida por una variable que se llamo NUMSINGLETON.

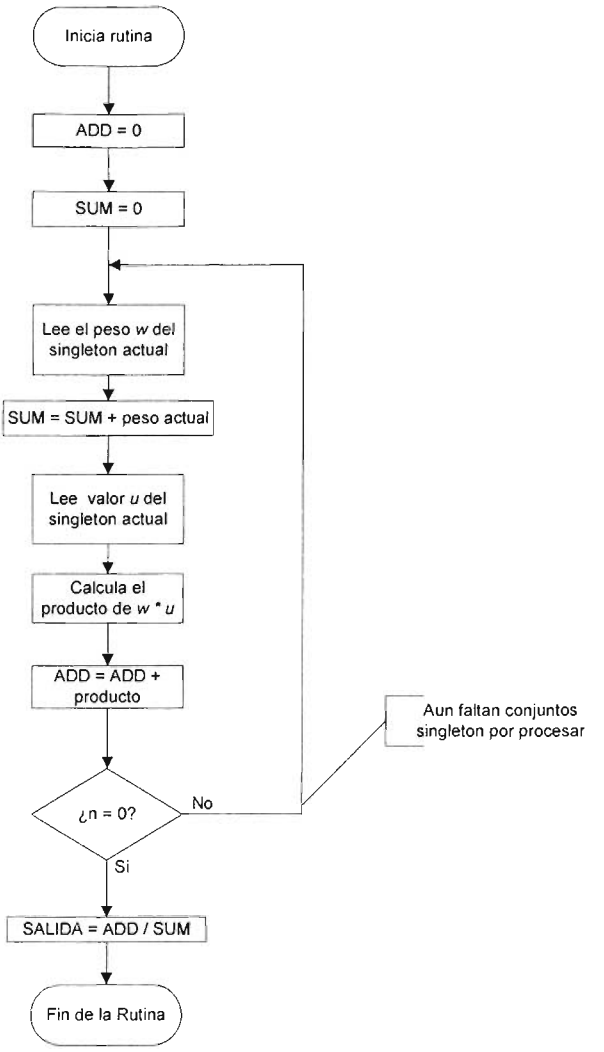


Figura 3.54 Diagrama de flujo de la rutina RDEFUZZY.

Antes de ejecutara la rutina RDEFUZZY se deben inicializar algunos apuntadores y variables que son utilizados por la misma, para tal propósito se diseño una rutina que se llamo INIDEFUZ, que hace las siguientes tareas:

1. Inicializa NUMSINGLETON con el valor de la constante de control NFCS previamente cargada en la rutina CARGAR_CONSTANTES_DE_CONTROL (ver apartado 3.3.6.1), la cual contiene el número de conjuntos o etiquetas lingüísticas definidas en la salida.

2. Cargar el apuntador DIRVMSAL con el valor de la constante de control DIVMS (ver apartado 4.3.6.1), que apunta a las salidas difusas en RAM producto de la evaluación de reglas.
3. Cargar el par de registros EEADRH:EEADR con el valor de las constantes de control AFCEEH: AFCEEL, que apuntan a la definición en memoria FLASH de las posiciones de los conjuntos singleton de salida.

Ejecutadas las tareas anteriores se esta en condiciones de ejecutar el proceso de defuzzificación llamando a la rutina RDEFUZZY que entrega su resultado de 10 bits en el par de registros SALIDAH:SALIDAL. A continuación este valor debe ser utilizado por el periférico apropiado para manejar el dispositivo externo de potencia adecuado que aplique la señal de actuación correspondiente para llevar a la salida del proceso u objeto bajo control al estado deseado. Para llevar acabo este propósito se diseñaron rutinas que tienen como cometido acondicionar esta salida, producto de la inferencia difusa para controlar los dispositivos de potencia del SID.

3.3.7.5 Salidas del sistema de control difuso.

El SID esta orientado al control de procesos basados en CC y CA, esto quiere decir que, con su salida se controlara o regulara la potencia entregada a cargas de CA o CC que funcionaran como actuadores o elementos finales de control de la máquina o proceso controlado, es decir un dispositivo eléctrico que base su funcionamiento en este tipo de corrientes y que las transforme o no en otra forma de energía para controlar la variable manipulada, llevando a la variable controlada del proceso al estado deseado.

De acuerdo a las especificaciones de diseño estos actuadores o elementos finales de control deben funcionar con una CA de 127V, 60Hz y un máximo de 15 A o con una CC de 12V y un máximo de corriente de 400mA.

Como los módulos de salida en CA y CC ambos tienen dos salidas, en primera instancia hace pensar, que el SID tiene más de una salida de control, pero en realidad este no es el caso, puesto que la salida del controlador difuso esta restringida a una sola. Esto mas bien, se debe a que la única salida control, con algunas manipulaciones adecuadas puede ser usada para controlar dos actuadores con la restricción de que actúen sobre la misma variable, solo que en sentidos opuesto, por ejemplo en un sistema de control de temperatura uno podría ser el encargado de manejar el sistema de enfriamiento (reducción de energía en el sistema) y otro el sistema de calentamiento (incremento de la energía del sistema), con lo que los dos estarían actuando sobre la misma variable de temperatura, en sentidos opuestos(aunque este tipo de salida no es utilizado en la pruebas). Esto también se puede extender a hecho de que un actuador pudiera estar basado en CA y otro en CC según el caso y conveniencia. Por eso dentro de las rutinas de salida o de acondicionamiento de salida se define una que considera estas posibles combinaciones, como se vera mas adelante en el apartado correspondiente (ver apartado 3.3.7.5.3).

En el apartado 3.3.7.1 se definieron los esquemas de control que se podrían diseñar con el SID, y de acuerdo a ellos las variables de salida entra las que se podían escoger eran:

- Salida de control $u(t)$, y
- Salida cambio de control $\Delta u(t)$.

En el primer caso, una vez disponible $u(t)$, esta puede ser, aplicada directamente como entrada de control de algún periférico, mas para el caso de la salida tipo cambio de control $\Delta u(t)$, esta no se puede ser aplicada directamente a un periférico, por que representa el cambio en la señal de la salida de control y no la señal de control en si, por tanto debe ser integrada para producir la señal de salida de control $u(t)$, después, de lo cual, puede ser utilizada para el control de algún periférico. En cualquier caso se debe tener disponible $u(t)$ para ser aplicada como entrada para el control de un periférico que maneje el suministró de potencia ya sea en CA y CC de algún dispositivo eléctrico que desempeñe la función de actuador o elemento final de control de la planta o sistema bajo control donde sea utilizando el SID.

En base a lo anterior se definieron tres tipos de salidas para el controlador difuso del SID que fueron:

1. **Salida Estándar.**
2. **Salida NZP.**
3. **Salida CControl.**

Antes de pasar a describir el diseño y conceptualización de este tipo de salidas se describe el diseño de un conjunto de rutinas que se utilizan por los tres tipos de salidas, para controlar la potencia que se entrega a las cargas de CA y CC.

3.3.7.5.1 Rutinas de control de potencia.

El conjunto de estas rutinas tienen como fines, inicializar el conjunto de periféricos y recursos del PIC que se utilizan para controlar los dispositivos de suministro de potencia de la cargas de CA y CC; además de encargarse propiamente de actualizar las entradas de estos periféricos con la señal de salida de la inferencia difusa o señal de control $u(t)$ (la cual es producida o generada por alguno de los tres tipos de salida mencionados en el apartado anterior) para controlar el suministro de potencia por parte de los dispositivos correspondientes. Estas rutinas están dentro del conjunto de rutinas de condicionamiento (ver apartado 3.3.7.1).

El método PWM (Modulación de Ancho de Pulso) fue seleccionado para controlar el suministro de potencia tanto para dispositivos de CA y CC. Esta selección se hizo con base en las bondades del método, en conjunción a la disponibilidad de periféricos adecuados en el PIC para la aplicación del mismo, situación que redujo el uso de hardware extra, y también evitó el uso de convertidores D/A.

3.3.7.5.1.1 Rutinas de control de potencia corriente continua.

EL PIC cuenta con dos módulos llamados CCP1 y CCP2 (módulos de Captura, Comparación y PWM), los cuales fueron utilizados para controlar el suministró de potencia a dispositivos que funcionen con CC. Estos módulos pueden operara como comparadores, como capturadores o como PWMs según se configuren inicialmente, sin embargo, solo interesa la operación PWM en este caso.

En seguida se da una breve descripción de cómo funcionan estos módulos en su operación PWM (solo se hará referencia a uno de estos, pues el otro funciona exactamente igual en modo PWM) y las rutinas que se diseñaron para su utilización en el control de potencia CC.

El módulo CCP1: El registro principal de este módulo (CCPR1) se compone de dos registros de 8 bits, denominados CCPR1H (16h) (parte más significativa) y CCPR1L (15h) (parte menos significativa). La operación del módulo se controla mediante el registro CCP1CON.

Selección del modo de operación.

La selección del modo en que trabajara el módulo CCPx se realiza mediante los cuatro bits menos significativos del registro CCPxCON, es decir, mediante los bits CCPx3:CCPx0 (CCP<CON<3:0>) de acuerdo a la Tabla 4.10.

Tabla 3.10. Bits de selección del modo de operación de los módulos CCPX.

CCPxM3:CCPxM0	Modo seleccionado
0 0 0 0	Captura/Comparación/PWM deshabilitados
0 1 0 0	Captura cada transición de bajada
0 1 0 1	Captura cada transición de subida
0 1 1 0	Captura cada cuarta transición de subida
0 1 1 1	Captura cada 16 transiciones de subida
1 0 0 0	Comparación, pone salida cada coinidencia
1 0 0 1	Comparación, limpia salida cada coincidencia
1 0 1 0	Comparación, genera interrupción cada coincidencia (salida inalterada)
1 0 1 1	Comparación, dispara evento espacial (CCP1 resetea TMR1; CCP2 resetea TMR1 y arranca una conversión A/D).
1 1 x x	Modo PWM

De aquí solo interesa la operación PWM.

Modo PWM (Modulación de Ancho de Pulso).

En este modo se puede producir una salida de frecuencia fija seleccionable modulada en ancho de pulso (o ciclo de trabajo) con una resolución de 10 bits, a través de la patita RC2/CCP1, como se muestra en la Figura 3.55.

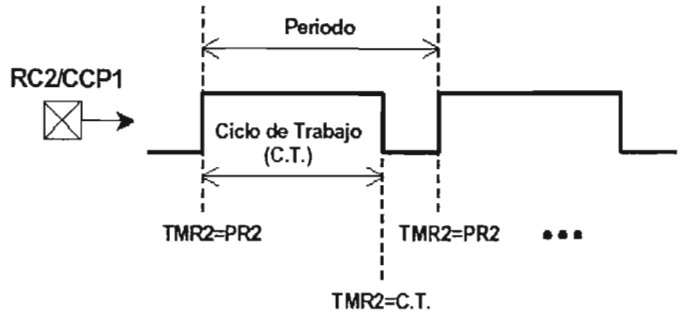


Figura 3.55 Salida PWM.

Cada modo de operación requiere como recurso uno de los Timers del PIC. La operación PWM requiere al timer 2. En la siguiente Figura se muestra un diagrama de bloques simplificado que resume la operación básica del PWM.

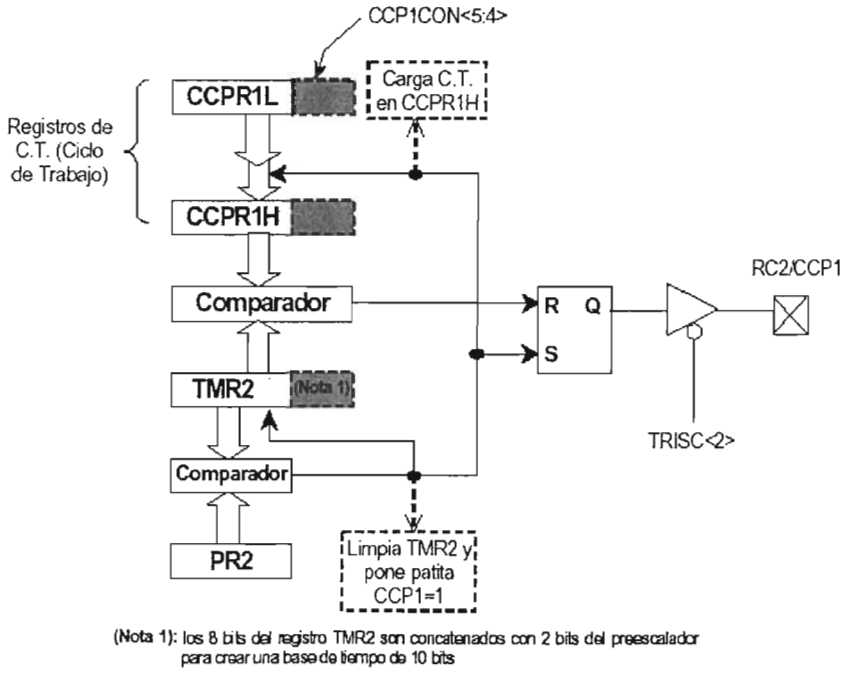


Figura 3.56 Diagrama de bloques PWM simplificado.

Control del Periodo del PWM

Para especificar el periodo del PWM se usa el registro PR2, de manera que el valor del periodo será:

$$PeridoPWM = (PR2 + 1) * 4 * T_{OSC} * M \quad \ldots\ldots\ldots \quad (3.17)$$

Donde 1/M es el valor del preescalador del Timer 2.

Cuando el valor en TMR2 alcanza el valor PR2 los siguientes tres eventos ocurren en el siguiente ciclo (Ver Figura anterior):

- El registro TMR2 es limpiado
- La patita CCP1 es puesta en alto (Excepto si el ciclo de Trabajo del PWM vale cero).
- El Ciclo de Trabajo es cargado de CCPR1L (15h) a CCPR1H (16h).

De esta manera, de acuerdo a la Figura anterior, el siguiente valor de comparación para TMR2 en el comparador de 10 bits es el Ciclo de Trabajo, el cual al alcanzarse limpiará la patita CCP1.

Control del Ciclo de Trabajo del PWM.

El ciclo de Trabajo se especifica escribiendo un valor de 10 bits al registro CCPR1L (los 8 bits más significativos (msb)) y los dos bits menos significativos (lsb) a CCP1CON<5:4> este valor de 10 bits lo representaremos como

CT = CCPR1L:CCP1CON<5:4>. El valor de tiempo que dura el ciclo de trabajo para un valor del preescalador de 1/M, se calcula de acuerdo a la siguiente ecuación:

$$T_{PWM} = CT * T_{OSC} * M \quad \ldots\ldots\ldots \quad (3.18)$$

Como se puede ver en la Figura anterior, el valor que determina la duración de C.T. del PWM no es el cargado en CT (CCPR1L), sino en CCPR1H, el cual sólo se actualiza copiando el valor de CT en el momento en que TMR2 alcanza el valor de PR2 (es decir, cada vez que se completa un periodo). Por ello, aunque CCPR1L puede ser escrito en cualquier momento, el Ciclo de Trabajo solo se actualiza hasta que termina el periodo que está en transcurso.

No hay otra manera de escribir al registro CCPR1H, ya que este es un registro de sólo lectura.

Cuando el valor de TMR2 (concatenado con dos bits internos) alcanza el valor de CCPR1H (concatenado con dos bits internos también) la patita CCP1 es limpiada (ver Figura anterior 3.56).

Como puede verse, el número de divisiones que se pueden tener en un Ciclo de Trabajo será 2^r, donde **r** es el número de bits usados, por lo tanto su duración máxima será este número de divisiones multiplicada por la duración del ciclo más pequeño del sistema T_{OSC}. Por lo tanto:

$$T_{PWM} = (2^r) * T_{OSC} * M \quad \ldots\ldots\ldots \quad (3.19)$$

Sin embargo, dependiendo del valor de ciclo de trabajo máximo (T_{PWM}) deseado, no será posible realizar las 2^r divisiones y por lo tanto no se podrán usar los r bits de resolución. O al revés, si se elige una resolución deseada **r** no será posible tener cualquier ciclo de trabajo máximo (T_{PWM}) deseado.

Además, si se elige T_{PWM} mayor que el periodo del PWM ($PeridoPWM$) la patita CCP1 no será limpiada (y por lo tanto no funcionará el PWM).

Ejemplo: Así, por ejemplo, suponiendo un cristal de 20 MHz, si deseamos usar la máxima resolución **r** = 10 bits, el ciclo de trabajo máximo posible será (para *M* = 1):

$$T_{PWM\,max} = 1024 * 50 \times 10^{-6} = 51.2\mu s.$$

O bien, la "Frecuencia del PWM" definida como $F_{PWM} = 1/PeriodoPWM = 1/T_{PWM\,max}$, tendrá un valor de:

$$F_{PWM} = 1/51.2 \times 10^{-6} = 19.53125kHz$$

O de lo contrario la patita CCP1 no podrá ser limpiada. Para tener este valor de F_{PWM} se requiere un valor en PR2 que está dado por la ecuación (3.21), de la cual despejando PR2, se tiene:

$$PR2 = ((PeriodoPWM / 4 * T_{OSC} * M)) - 1 \ldots\ldots\ldots \text{(3.20)}$$

Sustituyendo valores, se obtiene un PR2 = 255 = FF h.

En la siguiente tabla se resumen diversas elecciones de resolución **r** y la correspondiente frecuencia FPWM máxima, así como el valor de PR2 con el que se logra (para una frecuencia del cristal de 20 MHz).

F_{PWM} máxima	1.22 Khz	4.88 Khz	19.53 Khz	78.12 Khz	156.3 Khz	208.3 Khz
Preescalador	16	4	1	1	1	1
PR2	FFh	FFh	FFh	3Fh	1Fh	17h
Resolución máxima	10	10	10	8	7	5.5

Secuencia de configuración del PWM

A continuación se resumen los pasos para realizar la configuración inicial del PWM:

1. Establecer el periodo del PWM escribiendo al registro PR2.
2. Establecer el Ciclo de Trabajo del PWM escribiendo al registro CCPR1L y a los bits CCP1CON<5:4>.
3. Configurar como salida la patita CCP1, limpiando el bit TRISC<2>.
4. Configurar el preescalador del Timer 2 y habilitar el Timer 2, escribiendo al registro T2CON.
5. Configurar el módulo CCP1 para operación PWM. Poniendo en alto los bits CCP1CON <2:3>.

Con base en la información antes mencionada se diseñaron cinco rutinas asociados con los módulos PWM para control de la potencia en CC.

Estas rutinas fueron respectivamente:

1. **INIPWM**. Que realiza las tareas de cargar el registro PR2 con el valor predefinido de 255 o FF h, seleccionar un preescalador de 1/1, con lo que se define un valor de frecuencia PWM de 19.53125 Khz y finalmente encender el timer 2 para la operación PWM. Esta frecuencia se definió en base a que los tipos de cargas CC que pueden presentarse son de tipo inductivo o tipo resistivo. Para cargas de tipo inductivo como motores se deben usar frecuencias de mas de 1 KHZ (así no se descarga la bobina y los picos de corriente son menores, es por tanto mas efectivo), por lo que la frecuencia de 19.53125 Khz. resulta adecuada. Para las cargas resistivas no hay limite en cuanto a la frecuencia superior de operación salvo las limitaciones de baja frecuencia que le impone la respuesta térmica para que el parpadeo no sea visible, es decir, se puede usar desde una frecuencia lenta (del orden de unos 20 Hz) hasta frecuencias tan altas como el PWM pueda soportar (del orden de los Khz.), y la frecuencia de 19.53125 Khz. también resulta adecuada.
2. **CONFPWM1**. Esta rutina configura al módulo CCP1 para que opere en modo PWM y además inicializa en cero los registros de control del ciclo de trabajo.
3. **CONDPWM2**. Esta rutina es idéntica a rutina CONFPWM1 pero para el módulo CCP2.
4. **ACTPWM1**. Esta rutina copia el valor 10 bits de la señal de control $u(t)$ almacenado en el par de registros definidos en RAM SALIDAH:SALIDAL a el registro CCP1H donde se copian los 8 bit mas significativos del CT y a los bits CCP1CON<5:4> donde se copian los dos bit menos significativos, los cuales en conjunto actualizan el ciclo de trabajo del módulo PWM en base al resultado de la inferencia difusa. La salida PWM sacada por la patita RC2/CCP1 es alimentada al primero de los circuitos del *módulo de salida para cargas de CC* (ver el apartado 3.3.5), para controlar el suministro de potencia de una carga DC1.
5. **ACTPWM2**. Esta rutina es idéntica a rutina ACTPWM1 pero para el módulo CCP2. La salida PWM sacada por la patita RC1/CCP2 es alimentada al segundo de los circuitos del *módulo de salida para cargas de CC* (ver el apartado 4.6), para controlar el suministro de una carga DC2.

3.3.7.5.1.2 Rutinas de control de potencia de corriente alterna.

El método PWM también se utilizó para controlar el suministro de potencia a cargas de CA, solo que su implementación fue diferente. A continuación se describe el diseño y estructura de las rutinas que se utilizan para controlar el suministro de potencia a cargas CA con el método PWM.

El pulso PWM maneja el ángulo de disparo de un triac que controlará la potencia suministrada a las cargas CA. Debido a las características del triac el pulso PWM

es inverso al pulso PWM que se utilizó para el control de cargas CC, como se puede ver en la Figura 3.57.

La técnica consiste en detectar el cruce por cero de la señal de CA, el cual sucede cada vez que la señal CA cambia de valores positivos a negativo y viceversa (ver Figura 3.57), y el tiempo transcurrido entre dos cruces por cero determina el periodo del tren de pulsos PWM, con lo que se logra sincronizar la señal de CA al tren de pulsos PWM. Una vez detectado un cruce por cero se inicia un retardo de tiempo, que es inversamente proporcional, al producto de la inferencia difusa. Cuando se completa el retardo, el pulso PWM se activa en un nivel alto para dispara el triac y permanece así hasta un nuevo cruce por cero, momento en el cual es desactivado en un nivel bajo, repitiéndose la operación para cada nuevo cruce por cero. Por lo tanto el pulso activo PWM o ciclo de trabajo (CT) es proporcional a la inferencia difusa, lo que implica que el retardo de tiempo sea proporcional al ángulo de disparo del triac y ambos inversamente proporcionales al producto de la inferencia difusa.

Pera detectar el cruce por cero de la señal de CA se diseñó un circuito detector de cruce por cero que esta basado en un rectificador de onda completa, y cuyo diseño fue explicado en apartado 3.3.6 (Diseño del Módulo de Salida para Cargas de Corriente Alterna). La salida que produce este circuito es un pulso como el que es observado en la Figura 3.58, cada vez que la onda CA tiene un cruce por cero, el flanco positivo de este pulso se utiliza para producir un interrupción por el pin RB0/INT (ver apartado 3.3.6) del puerto B del PIC, y la rutina de atención a interrupción apropiada, se encargara de actualizar el valor de carga del los registros del timer 1(carga AC1) y timer 2(carga AC2) del PIC para producir el retardo correspondiente, antes de activar el pulso de disparo de los triac correspondientes.

EL valor de carga del timer 1 es calculado por una rutina que se llamo ACTTMR1, y que toma el valor resultante de la inferencia difusa, es decir el valor de 10 bits almacenado en los registros SALIDAH:SALIDAL, para calcular la nueva carga del timer 1, para producir el retardo adecuado, antes de activar el pulso de disparo del triac 1, a través de la patita RB1 del PIC(ver apartado 3.3.6) .

Para comprender el diseño de la rutina ACTTMR1 antes se describe rápidamente el funcionamiento del timer 1.

El timer 1 es un contador/temporizador de 16 bits. El conteo es realizado por el par de registros de 8 bits TMR1H:TMR1L (abreviados como TMR1) y su operación es controlado por el registro T1CON.

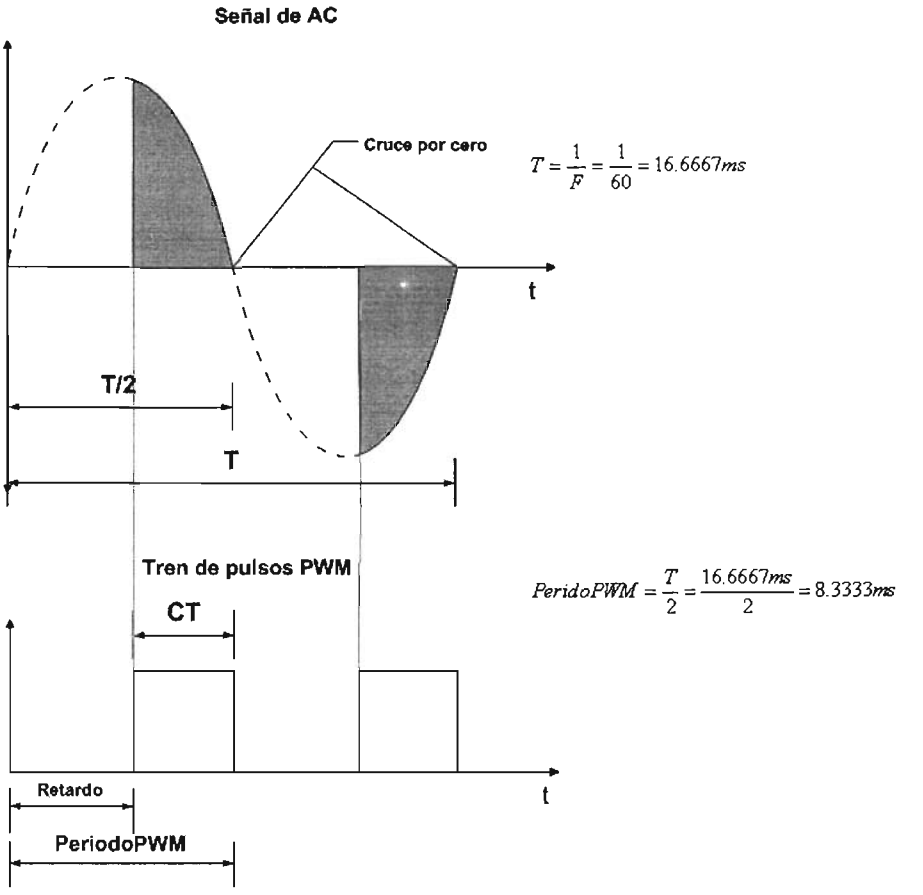


Figura 3.57 Pulso PWM para el control de cargas CA.

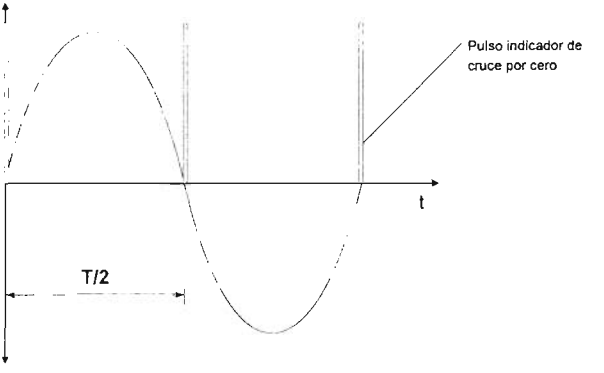


Figura 3.58 Tren de pulsos indicadores de cruce por cero.

Así, el registro TMR1 se incrementa de 0000 h hasta FFFF h y en la siguiente cuanta se reinicia a 0000h y así sucesivamente, el reciclarse se activa (en alto) la bandera TMR1IF (PIR1<0>), la cual puede ser utilizada para generar una interrupción, o bien, para ser consultado por poleo. Para nuestro caso se usa la interrupción, por permitir una ejecución en tiempo real.

EL timer 1 puede funcionar como temporizador o contador, mas en este caso solo intereso su funcionamiento como temporizador, en el cual el timer se incrementa (si no se considera el preescalador) en cada ciclo de instrucción (a la frecuencia $F_{OSC}/4$). El preescalador que se puede intercalar entre el reloj $F_{OSC}/4$ y el registro TMR1 puede tener solo 4 valores: 1/1, 1/2, 1/4 y 1/8.

Cualquier temporización del timer 1 esta dada por:

$T_{TIMER1} = N * 4 * T_{OSC} * M$ ..... (3.21)

Donde,

N: es el número de cuentas del timer 1, y se define como $N = TMR1 + 1$, es decir, el valor del timer más 1(TMR1 hace referencia al para de registros TMR1H:TMR1L (16 bits)),

T_{OSC}: es el periodo de oscilación del reloj de trabajo del PIC y M es el inverso del preescalador es decir $M = 1/p$ preescalador seleccionado para el timer 1.

Con base en la información anterior, antes de usar el timer 1 se debe configurar e inicializar adecuadamente. La rutina encargada de dicha configuración e inicialización se llamo INICTRIAC1, la cual además se encarga de la habilitación de las interrupciones correspondientes para la generación del pulso PWM. Las operaciones que se ejecutan en esta rutina son:

1. Configuración del bit RB0/INT del puerto B como entrada para la detección del flanco de subida del pulso indicador de cruce por cero, con los bits restantes como salidas.
2. Configuración de la interrupción externa (INTEDG = 1) por detección de un flaco descendente.
3. Habilitación de la interrupción externa, INTIE = 1; y habilitación de la interrupción por desbordamiento del timer 1, TMR1IE = 1.
4. Configuración del timer 1 como temporizador y un preescalador de 1/8 de su frecuencia de operación ($F_{OSC}/4$).
5. Desactivación del pulso de disparo e inicialización del TMR1 con el valor de carga para producir el máximo retardo (no se dispara el triac), es decir el necesario en un suministro de potencia de cero.

Para diseñar la rutina ACTTMR1 se parte del hecho de que, a frecuencia PWM debe ser dos veces la frecuencia de la señal de CA, es decir, $PeridoPWM = T_{AC}/2$ (ver Figura 3.57), con lo que el máximo valor que puede tener el ciclo de trabajo PWM es $T_{PWM} = PeridoPWM$, que corresponde con la

máxima entrega de potencia a la carga controlada y por lo tanto a un retardo de tiempo mínimo.

EL planteamiento inicial de diseño para generar el retardo correspondiente antes de producir el pulso de disparo fue el siguiente: encontrar un valor de N (numero de cuentas) para obtener una temporización máxima de $T_{PWM} = T_{AC}/2 = 8.3333ms$, la cual corresponde a el máximo retardo de tiempo, o en otras palabras a la entrega de potencia mínima. De acuerdo a lo anterior, con un preescalador de 1/8 y reloj de operación de 20 MHz, el número cuentas utilizando la ecuación 3.21 es:

$$N = \frac{T_{TIMER1}}{4 * T_{OSC} * M} = \frac{8.3333 \times 10^{-3}}{4 * 50 \times 10^{-9} * 8} = 5208.3125 \approx 5208$$

Con el que se tiene una temporización de:

$$T_{TIMER} = (5208.3125) * 4 * 50 \times 10^{-9} * 8 = 8.3328ms$$

El siguiente paso de diseño consintió en encontrar una relación tal que, para el valor máximo de salida de la inferencia difusa, de 3FFh o 1023_d (por lo diez bits utilizados por el CA/D) se tenga un retardo mínimo (por ser inversamente proporcionales) de tal manera que se entregue la potencia máxima, y por el otro lado para el valor mínimo de la inferencia difusa de 000h o 0 se tenga el retardo máximo (temporización máxima) y se entregue la potencia mínima a la carga de CA. Esto operación es ilustrada por la recta de la Figura 3.59

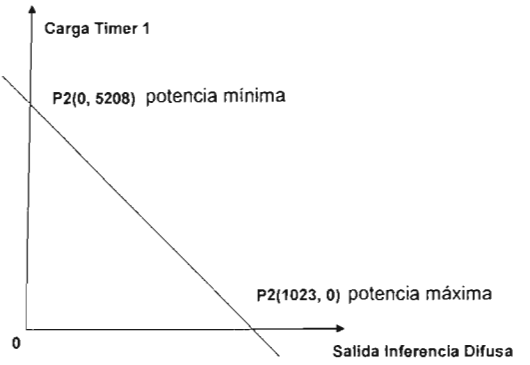


Figura 3.59 Carga del timer 1 en función del producto de la inferencia difusa.

La ecuación de esta recta esta dada por:

$$m = \frac{-5208}{1023} = -5.0909 \approx -5$$

$$y = -5(x - 1023)$$

$$y = 5115 - 5x \ldots\ldots\ldots\ldots (3.22)$$

Donde *y* representa la carga de timer 1 y *x* la salida de inferencia difusa.

Aplicando la ecuación para una *x* máxima de 1023.

$y = 5115 - 5 * 1023 = 5115 - 5115 = 0$, valor de cuentas del timer 1 que corresponde a un retardo de tiempo mínimo (no se realiza cuenta alguna) y con el que se tiene un error absoluto en este extremo de cero.

Ahora para un valor de x = 0

$y = 5115$, valor de cuentas del timer 1 con el que se obtiene un retardo máximo de:

$T_{TIMER1} = (5115) * 4 * 50 \times 10^{-9} * 8 = 8.184ms$

Con un error absoluto en este extremo de $EA = 8.3333ms - 8.184ms = 149.3\mu s$.

Si se quiere, que el error sea en el extremo máximo y no en el mínimo, se debe cambiar el valor de 5115 por el de 5208 es decir $y = 5208 - 5x$ ..... (3.23) con la que se obtiene:

Para x = 1023:

$y = 5208 - 5 * 1023 = 5208 - 5115 = 93$, que corresponde con un retardo mínimo de: $T_{TIMER1} = (93) * 4 * 50 \times 10^{-9} * 8 = 148.8\mu s$ y por lo tanto un error absoluto del mismo valor

Y para x = 0:

$y = 5208$, valor con el que se obtiene un retardo máximo de: $T_{TIMER1} = (5208) * 4 * 50 \times 10^{-9} * 8 = 8.3328ms$ y un $EA = 8.3333 - 8.3328 = 0.5\mu s$

Ahora bien, lo mas adecuado, es que en los dos extremos se tenga un margen de seguridad de igual duración, el valor para lograr lo anterior se obtiene como:

$(5115 + 5208)/2 = 5161.5 \approx 5162$, con lo que se obtiene la ecuación $y = 5162 - 5x$ ..... (3.24), con la que se tiene respectivamente:

Para x = 1023

$y = 5162 - 5 * 1023 = 5162 - 5115 = 47$, que corresponde con un retardo mínimo de: $T_{TIMER1} = (47) * 4 * 50 \times 10^{-9} * 8 = 75.2\mu s$ y un $EA = 75.2\mu s - 0 = 75.2\mu s$

Para x = 0

$y = 5162$, valor con el que se obtiene un retardo máximo de: $T_{TIMER1} = (5162) * 4 * 50 \times 10^{-9} * 8 = 8.2595ms$ y un $EA = 8.3333 - 8.2592 = 74.1\mu s$

De acuerdo a la ecuación 3.24, la señal de salida CA para un valor de salida máximo de la inferencia difusa y un valor mínimo, es como la observada en la Figura 3.60.

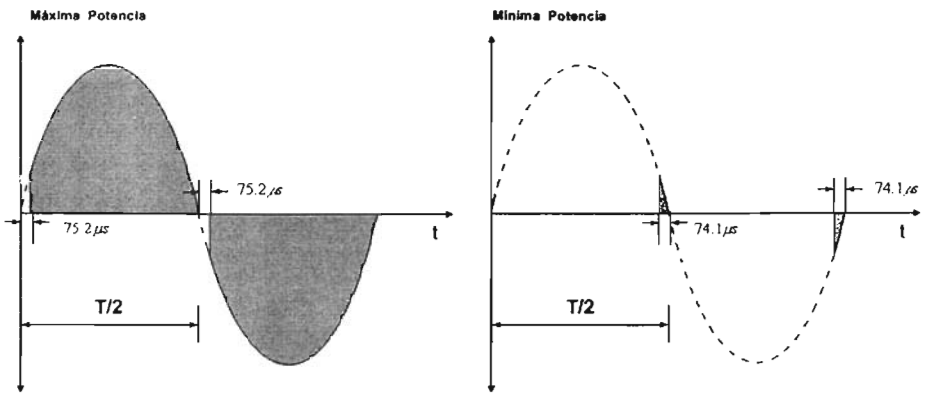


Figura 3.60 Salida de potencia máxima y mínima.

Entonces lo que hace la rutina ACTTMR1 es tomar el valor resultante de la inferencia difusa o salida de control $u(t)$ y por medio de la ecuación 3.24 calcula el valor de cuentas de timer 1 para producir el retardo de tiempo correspondiente antes de activar el pulso de disparo del triac 1, aun así esta rutina no es la encargada de actualizar los registro de trabajo del timer 1 con dicho valor.

La rutina de encargada actualizar los registros de trabajo del timer 1 es la de atención a la interrupción externa por RB0/INT, a través de la cual se detecta el cruce por cero de la señal CA. El diagrama de flujo de esta rutina que se llamo ACTUALIZAR_RETARDOS es mostrado en la Figura 4.59.

Dentro de esta rutina se actualiza la carga del timer 1 y timer 2 (el cual es utilizado para controlar al segundo triac) para producir los retardos respectivos siempre y cuando estén habilitadas las interrupciones por desbordamiento de los mismos. Si la interrupción por desbordamiento del timer 1 esta habilitada el valor de cuentas calculado por la rutina ACTTMR1 es copiado a los registros de trabajo del timer 1, después de lo cual es encendido para comenzar su cuenta hasta completar el retardo correspondiente, inicialmente el pulso de disparo es desactivado.

El valor de cuentas copiado es el complementado a 1 de este valor, de tal modo que cuando se completen el número de cuentas, correspondientes al retardo actual se produzca una interrupción por desbordamiento, y la rutina de atención a interrupción por desborde del timer 1 correspondiente, ejecute las acciones necesarias para activar el pulso PWM.

La rutina de atención a la interrupción por desbordamiento del timer 1 que fue diseñada se llamo DISPARAR_TRIAC1. La Figura 3.62 muestra el diagrama de flujo de la operación de esta rutina.

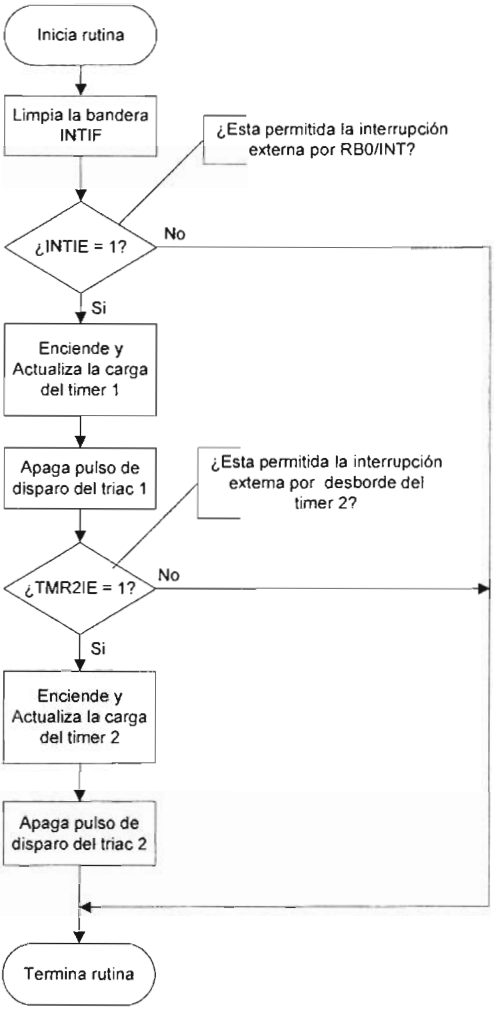


Figura 3.61 Diagrama de flujo de la rutina a la interrupción externa por RB0/INT ACTUALIZAR_RETARDOS

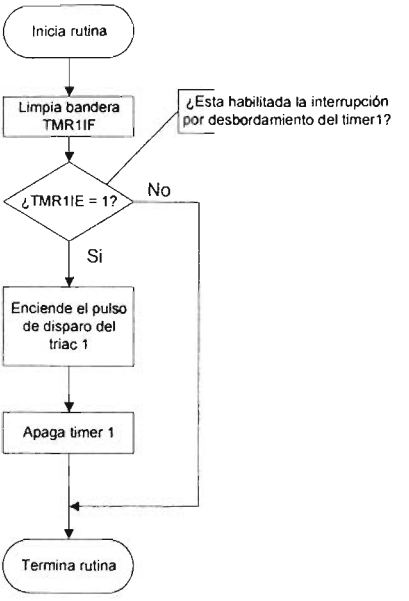


Figura 3.62 Diagrama de flujo de la rutina de atención a la interrupción por desbordamiento de timer 1, DISPARAR_TRIAC1.

Como se puede ver de la Figura anterior la acción fundamental de esta rutina es activar el pulso de disparo del triac 1 una vez que se ha completado el retardo de tiempo correspondiente, el cual permanecerá activo hasta que una nueva interrupción externa por RB0/INT se produzca, y se cargue un nuevo valor en el timer 1, para el siguiente retardo. Dentro de esta misma rutina el timer 1 es apagado.

Se diseñaron tres rutinas mas, para generar la señal PWM que controla el disparo del triac de la segunda salida de potencia CA (ver apartado 3.3.6), en las que el recurso ocupado para producir el retardo no fue el timer 1 si no el timer 2. Debido a que la estructura y funcionamiento de estas rutinas se diseño igual, que las rutinas de control del triac 1, solo se especifica su operación.

1. INITRIAC2. Esta rutina configura e inicializa el timer 2, además de habilitar la interrupción por desborde del timer 2.
2. ACTTMR2. AL igual que ACTTMR1 toma el valor resultante de la inferencia difusa o salida de control $u(t)$ y por medio de la ecuación $y = 255 - x$ (3.25) calcula el valor de cuentas de timer 2 para producir el retardo de tiempo correspondiente, antes de activar el pulso de disparo del triac 2.
3. DISPARAR_TRIAC2. Es la rutina de atención a la interrupción por desbordamiento del timer 2, y tiene un diagrama de flujo igual que el de la Figura 3.62, solo que su función es activar el pulso de disparo del triac 2, a través de la patita RB2 del PIC (ver apartado 3.3.6).

Como se menciono, la rutina ACTUALIZAR_RETARDOS (ver Figura 3.61) también se encarga de actualizar el valor de carga del timer 2 con el valor calculado de cuentas en la rutina ACTTMR2, aplicando las mismas operaciones que para el timer 1.

Esto es lo referente al conjunto de rutinas que se encargan del control de potencia de los dispositivos o cargas CA y CC. En los siguientes apartados se describen los tipos de salida del SID y como es que hacen uso de estas rutinas de control de potencia.

3.3.7.5.2 Salida Estándar.

La **Salida Estándar** se nombró así por que es el tipo de salida más común que puede tener un controlador difuso. Para este tipo, la salida del controlador difuso es $u(t)$, la cual puede ser usada directamente como entrada a las rutinas de control de potencia en CA o CC explicadas previamente, razón por la cual no se diseño una rutina especial para procesar y generar este tipo de salida.

La **Salida Estándar,** tiene dos opciones de carga de salida, esto es que se controle el suministro de potencia a un actuador de CC o que se controle el suministro de potencia a un actuador CA. Esta selección se hace cuando se diseña el controlador difuso en la IPVSID.

Dependiendo del tipo de de carga seleccionada se utilizan determinadas rutinas de control de potencia en el algoritmo de control difuso como sigue:

- Si la selección de carga de salida es CA, las rutinas que se utilizaran son INITRIAC1, ACTTMR1, ACTUALIZAR_RETARDOS y DIPARAR_TIRAC1, ya que con el uso conjunto y en orden de estas rutinas se consigue el manejo del pulso de disparo del triac 1, que controlara el suministro de potencia de una carga CA en una aplicación particular. Cada una de estas rutinas es llamada en orden dentro del PCD para que el pulso PWM se genere correctamente, INITRIAC1 es las primera, y solo se ejecutan una sola vez antes de comenzar el ciclo de control difuso, a diferencia de ACTTMR1, que es ejecutada una vez por cada periodo de muestreo, y ACTUALIZAR_RETARDOS y DIPARAR_TRIAC1 que se ejecutan cada medio periodo de la señal de CA ($PeriodoPWM$) para estar actualizando constantemente el ciclo de trabajo en base al resultado de la inferencia difusa.
- Si la selección de carga de salida es CC, las rutinas que se utilizan son INIPWM, CONFPWM1 y ACTPWM1, ya que con el uso conjunto y en orden de estas rutinas se consigue producir el tren de pulsos PWM para controlar el suministro de un dispositivo CC para una aplicación particular. Cada una de estas rutinas es llamada en orden dentro del PCD para que el módulo PWM funcione correctamente, INIPWM y CONFPWM1 son las primeras, y solo se ejecutan una sola vez antes de comenzar el ciclo de control difuso, a diferencia de ACTPWM1, que se ejecuta una vez por cada periodo de muestreo, para estar actualizando constantemente el ciclo de trabajo del pulso PWM en base al resultado de la inferencia difusa.

3.3.7.5.3 Salida SNZP.

La salida **SNZP** fue pensada para que pudiera controlar dos actuadores, uno para valores positivos y otro para valores negativos de la salida del controlador difuso, es decir el rango o universo de discurso de la variable lingüística de salida debe ser simétrico con valores negativos y positivo, pasando por cero (a lo que se le debe el nombre de este tipo de salida).

Para valores positivos de la señal de salida se controlaran actuadores que suministren energía positiva al sistema u objeto bajo control (por ejemplo suministro de calor en el caso del control de temperatura) y para valores negativos se controlaran actuadores que suministren energía negativa al sistema u objeto bajo control (por ejemplo enfriamiento o extracción de calor en el caso del control de temperatura), en otras palabra ambas actuadores deben actuar sobre la misma variable controlada, pero en sentidos apuestos (ver apartado 3.3.7.5).

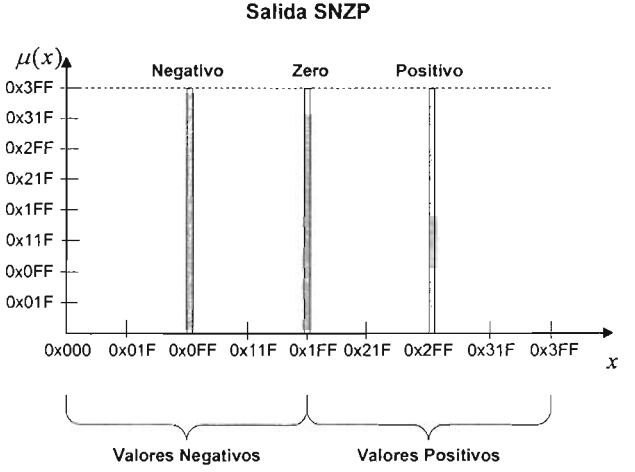


*Figura 4.63 Ejemplo de la definición de una variable lingüística para una Salida **SNZP**.*

En esta Figura se observa una salida SNZP con tres etiquetas **Negativo**, **Zero**, **Positivo**, que son asignadas cada una a un uno conjunto singleton constante. El conjunto singleton para la etiqueta **Zero** debe estar siempre situado en el centro del rango de la salida.

En la Tabla 3.11 se resumen las distintas combinaciones de tipos de carga (actuadores) de salida para valores positivos y negativos de la salida SNZP, de acuerdo a las salidas disponibles en CA y CC del módulo de salida (ver apartado 3.3.5 y 3.3.6).

Tabla 3.11. Combinaciones de tipo de carga para la salida SNZP.

Caso	Negativos	Positivos
A	DC2	AC1
B	AC1	DC1
C	DC2	DC1
D	AC2	AC1

Entonces la rutina encargada de generar este tipo de señal debe determinar que valores se toman como positivos y cuales como negativos y de acuerdo a eso actualizar la salida de control de potencia de las cargas o actuadores correspondientes.

De acuerdo con la información anterior se diseñó una rutina llamada igual que el nombre de la salida SNZP, y cuyo diagrama de flujo es mostrado en la Figura 3.64

En este tipo de salida, la salida del controlador difuso es $u(t)$, sin embargo antes de poder ser usada para actualizar la entrada de los periféricos de control de potencia debe recibir un procesamiento por medio de la rutina SNZP.

De la Figura 3.64 se observa que lo primero que se le hace a la variable de salida de la inferencia difusa SALIDA es determinar si se encuentra del lado positivo o negativo de la variable lingüística de salida, lo que se consigue comparando su valor con 511_D o 1FFh, si es mayor es positiva y si es menor es negativa. Una vez determinado el signo de la salida a la diferencia se le multiplica por dos para volver a ajustar el valor a 10 bits pues después de la resta el valor queda reducido al rango de valores posibles para 9 bits. Este valor resultante representa ya la salida de control $u(t)$ que se aplicara a los actuadores correspondientes. En cada caso, tanto para valores positivos como negativos, se determina el tipo de carga que se va a controlar, y las combinaciones posibles son mostradas en la Tabla 3.11. El tipo de combinaciones de cargas se define cuando se diseña el controlador difuso en la IPVSID. Para tal motivo se definió una constante de control que es cargada con un valor apropiado según el tipo de combinación de carga que se haya seleccionado. Un bit de esta variable indicará el tipo de carga para los valores positivos y en otro bit el tipo de carga para lo valores negativos. Esta variable fue llamada LOADF (ver apartado 3.3.6.1) y las banderas LOADP (LOADF<0>) y LOADN (LOADF<1>) respectivamente. Si las banderas están a 1 se trata de una carga CA, en caso contrario están a 0, y por lo tanto se trata de una carga CC. Una vez que se determina el tipo de carga se llama a la rutina de control de potencia respectiva. Por ejemplo, si para valores positivos el tipo de carga ha sido CA entonces se procede a llamar a la rutina ACTTMR1 la cual toma el valor calculado para SALIDA (ver Figura 3.64) para valores positivos y calcula el valor de cuentas del timer 1, para producir el retardo correspondiente (ver apartado 3.3.7.5.1.2).

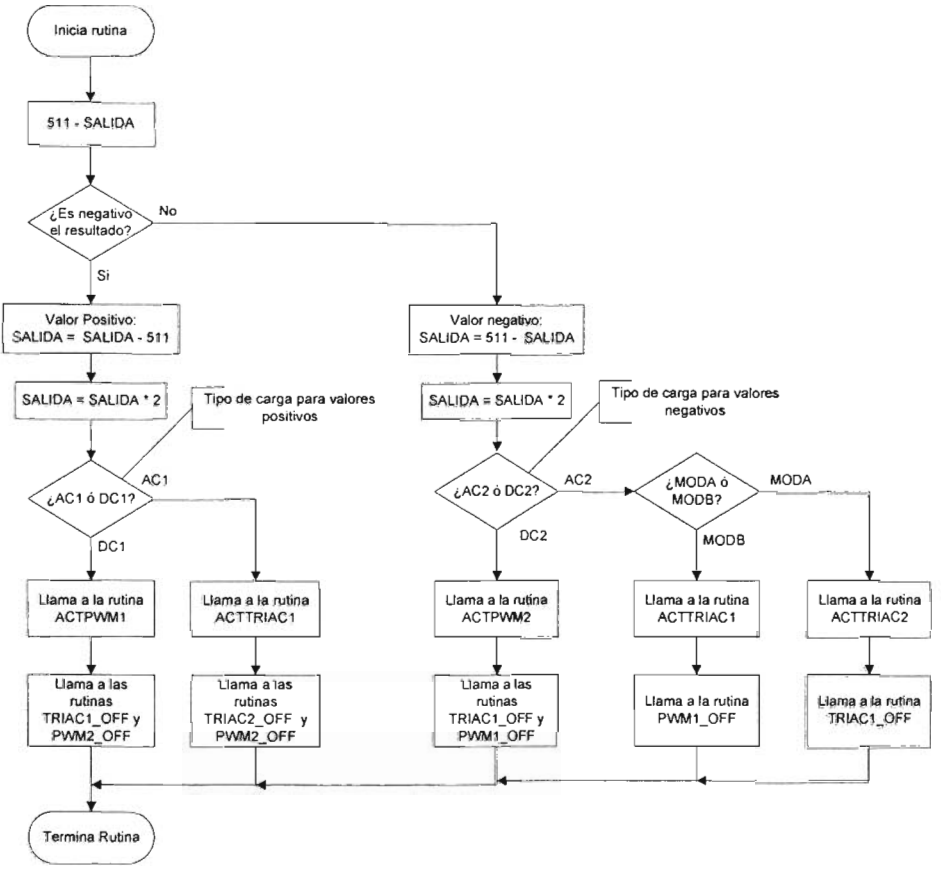


Figura 3.64 Diagrama de flujo de la rutina SNZP.

Un aspecto muy importante del diagrama de flujo de la rutina SNZP es que a parte de llamar a la rutina de actualización correspondiente de la salida de potencia ya se en CA y CC, enseguida se llama a otras rutinas. Por ejemplo para una carga CA para valores positivos aparte de llamar a la rutina ACTTMR1 se llama a las rutinas TRIAC2_OFF, PWM2_OFF. Para comprender la razón de la llamada de estas rutinas primero se debe echar un vistazo a la tabla 3.12, en la cual se muestran las rutinas de control de dispositivos de potencia que son llamadas para cada una de las combinaciones de cargas de la Tabla 3.11. Considerando el mismo ejemplo, cuando la carga es CA para valores positivos se llama a la rutina ACTTMR1, esta situación se presenta en el caso **A** y en el caso **D.** Para valores negativos en el primer caso, la rutina que se debe ser llamada es ACTPWM2(carga DC2); para el segundo, la que tiene que ser llamada es ACTTMR2(carga AC2), ahora bien si la salida toma valores positivos, la potencia que se entrega a las cargas para valores negativos de la salida debe ser cero (carga DC2 o AC2), y son las rutinas TRIAC2_OFF, PWM2_OFF las que se

encargan de asegurar esta situación, este razonamiento se aplica en modo inverso, si la salida toma valores negativos, entonces la potencia que se entrega a las cargas para valores positivos de la salida debe ser cero (se llama a la rutina TRIAC1_OFF). En otras palabras estas rutinas son llamadas para evitar que accidentalmente pudieran estar funcionando los dos actuadores de una salida SNZP, uno correctamente y el otro incorrectamente, que es una situación peligrosa que debe evitarse. La misma explicación es valida para los demás casos.

Tabla 3.12. Rutinas que se llaman de acuerdo a las combinaciones de cargas de la Tabla 3.11.

Caso	Negativos	Positivos
A	ACTPWM2	ACTTMR1
B	ACTTMR1	ACTPWM1
C	ACTPWM2	ACTPWM1
D	ACTTMR2	ACTTMR1

La función de cada una de las rutinas TRIAC1_OFF, PWM1_OFF, TRIAC2_OFF y **PWM2_OFF** es resumida a continuación:

TRIAC1_OFF: la finalidad de esta rutina es poner a cero o desactivar el pulso de disparo del triac 1, además de cargar un valor en el timer 1 de tal manera que su temporización resulte en un retardo máximo y no se suministre nada de potencia a la carga CA que controla, es decir un ciclo de trabajo de 0 %.

PWM1_OFF: la finalidad esta rutina es poner a 0% el ciclo de trabajo del módulo PWM1, además de asegurarse el pulso PWM esta desactivado, de tal manera que no se suministre nada de energía a la carga CC que controla.

TRIAC2_OFF: La finalidad de esta rutina es la misma que la rutina TRIAC1_OFF, pero para el triac 2.

PWM2_OFF: La finalidad de esta rutina es la misma que la rutina PWM1_OFF, sin embargo para el módulo PWM2.

Mirando por última vez a la Figura 3.64 es observado que para valores negativos si el tipo de carga seleccionado ha sido de CA, enseguida de eso a diferencia de los valores positivos, se hace una selección más para determinar que rutina de control de potencia CA va a ser llamada. Esta situación surge del caso **B**, como se puede observar en la Tabla 3.11, en este caso para valores negativos se tiene una carga CA y la rutina que se definió para ser llama es ACTTMR1; y para valores positivos se tiene una carga CC y la rutina que se definió para ser llama es ACTPWM1; cuando la combinación que se debería hacer es ACTTMR2 para valores negativos; y manteniéndose ACTPWM1 para valores positivos, esto si se sigue la lógica de que para los valores positivos se deban usar las rutinas ACTTMR1, ACTPWM1 y para valores negativos las rutinas ACTTMR2, ACTPWM2. Solo que esto no sucede así a causa de que, cualquiera de los

módulos CCP en su operación PWM, necesitan del timer 2, al igual que la rutina ACTTMR2 para producir la salida PWM para controlar el segundo triac. De combinar ACTTMR2 con cualquiera de las rutinas ACTPWM1 y ACTPWM2 se produciría una interferencia de funcionamiento entre las dos rutinas, lo que resultaría en una situación muy peligrosa. Para evitar este inconveniente se definieron don modos, dentro de los cuales se clasificaban cada uno de los casos. Esta clasificación fue definida así.

Modo **A**: Incluye a los caso **A**, **C** y **D**.

Modo **B**: incluye solo el caso **B.**

Para indicar el modo de cada caso se definió una bandera que es bit 2 de la constante de control LOADP y se llamo MODAB, y si vale 1 se trata del modo B y si vale 0 del modo **A.** Entonces cuado se selecciona una carga CA para valores negativos, enseguida se checa el valor de esta bandera MODAB par decidir cual de las rutinas ACTMR1 y ACTMR2 es llamada, siendo llamada la primer para el caso **B** y la segunda para el caso **A**, con lo que se evita que interfieran en su funcionamiento las rutinas ACTMR2 y ACTPM1.

3.3.7.5.4 Salida CCONTROL.

La salida **CCONTROL** se pensó para que la salida del controlador difuso no fuera $u(t)$ si no $\Delta u(t)$, que es interpretada como el cambio en la señal de control, es decir lo que se debe aumentar o disminuir a $u(t)$. Por los tanto para obtener $u(t)$ es necesario integrar $\Delta u(t)$. Uno se podría preguntar ¿que caso tiene definir este tipo de salida si al final también se obtiene $u(t)$? La respuesta es que con este tipo de salida consigue la implementación del esquema de control PI de la Figura 3.33. El tipo de cargas que maneja este tipo de salida solo se tiene dos opciones, o se maneja una carga CA o se maneja una carga CC.

La rutina que se diseño para generar este tipo de salida se llamo CCONTROL, su función principal es integrar la salida $\Delta u(t)$ del controlador difuso para obtener $u(t)$ y usarla como entrada para la rutinas de control de potencia ya sea en CA o en CC según se haya seleccionado el tipo de carga en la IPVSID. EL diagrama de flujo de la rutina es mostrado en la Figura 3.65

Las mismas consideraciones que se aplicaron para la definición de la variable lingüística para la salida SNZP son validas para la salida CCONTROL, es decir se debe definir el rango simétricamente para cambios negativos y positivos pasando por cero (ver Figura 3.63).

En la rutina CCONTROL, lo primero que se le hace a la variable de Salida de la inferencia difusa SALIDA (que hemos llamado por conveniencia CControl) es determinar si se encuentra del lado positivo o negativo de la variable lingüística de salida, lo que se consigue comparando su valor con 511_D o 1FFh, si es mayor es positiva y si es menor es negativa. Una vez determinado el signo de la salida a la diferencia se le multiplica por dos para volver a ajustar el valor a 10 bits pues después de la resta el valor queda reducido al rango de valores posibles para 9 bits. Este valor resultante representa el cambio de la señal $u(t)$, si es negativo se

resta del valor actual de $u(t)$ y si es positivo se suma, con lo que se logra integrar $\Delta u(t)$. Puesto que el rango de valores de $u(t)$ tiene una resolución de 10 bits que va de 0 a 1023, se debe evitar que tome valores menores a 0 y mayores a 1023, pues al tener una salida integral el rango de valores puede extenderse mas allá de esto limites. Hecho esto ultimo tenemos un valor valido para $u(t)$ el cual es almacenado una vez mas en la variable SALIDA, la cual, en estas condiciones puede ser utilizada por la rutina de control de potencia CA o la rutina de control de potencia CC, según el tipo de carga que se haya seleccionado para este tipo de salida.

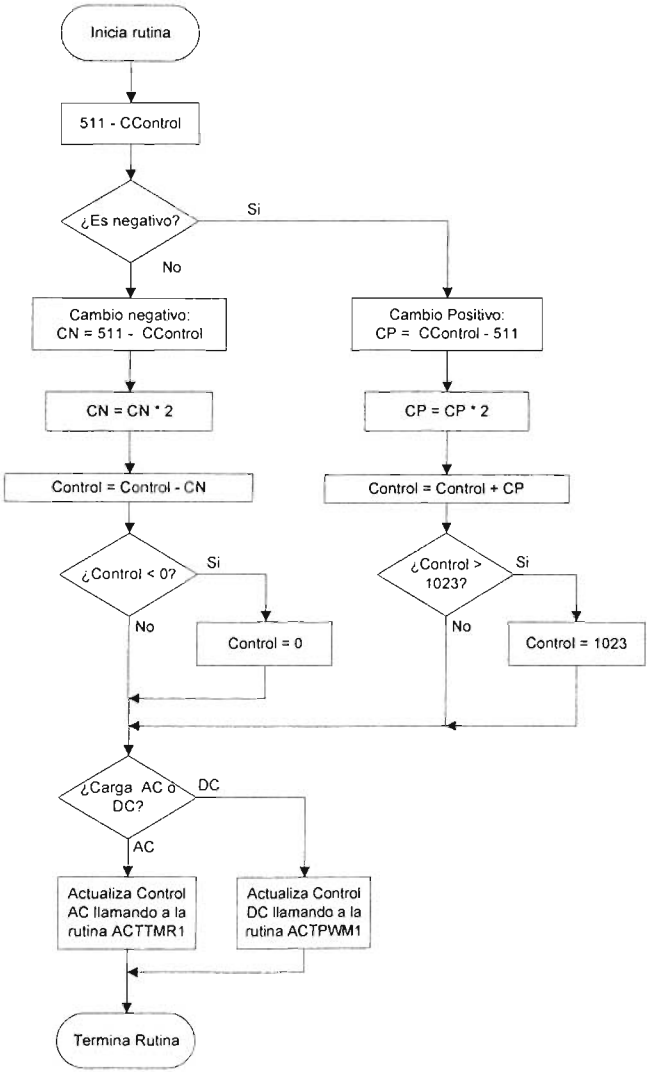


Figura 3.65 Diagrama de flujo de la rutina CCONTROL.

Con esto se finaliza la explicación de la rutinas que se encargan de acondicionar la salida de la inferencia difusa para controlar las cargas de CA y CC que funcionan como actuadores o elementos finales de control en el proceso u objeto bajo control. Y también se finaliza la explicación de las rutinas de aplicación que forman parte de la implementación del sistema de control difuso del SID.

3.3.7.6 Control de la frecuencia de muestreo.

El control y selección de la frecuencia de muestreo es un parámetro muy importante en el diseño de sistemas de control digital tanto convencionales como los basados en lógica difusa, en cualquier caso los criterios de control y selección se basan teóricamente en el teorema de muestreo. Esto viene a limitar la aplicabilidad del SID ya que la frecuencia de muestreo para un sistema particular limitara el tiempo máximo de ejecución de ciclo del algoritmo de control difuso (adquisición, inferencia, condicionamiento) el cual debe ser menor que el periodo de muestreo, pues en caso contrario no podría cumplir los requerimientos de la aplicación. La situación anterior es ilustrada en la Figura 3.66.

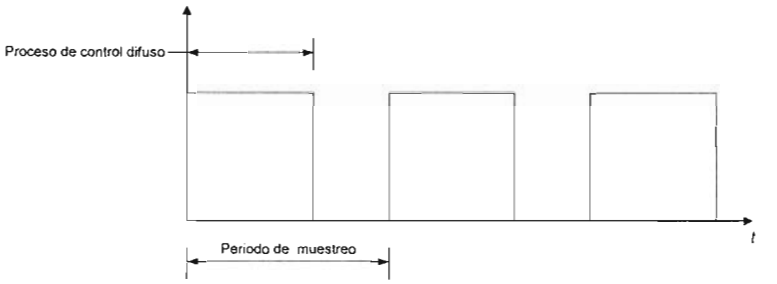


Figura 3.66 Diagrama de tiempos para el control de la frecuencia de muestreo.

La ejecución de cada uno de los pasos de la inferencia difusa toma tiempo, el proceso de fuzzificaciòn depende del numero de entradas y del numero de etiquetas lingüísticas de cada entrada, así también la defuzzificaciòn depende del numero de etiquetas lingüísticas de salida, sin embargo esta dependencia es mínima, generando un tiempo a veces casi constante para estas operaciones, pero no así para el proceso de evaluación de reglas ya que el microcontrolador ejecuta las reglas de un manera serial cada vez que nuevas entradas son adquiridas o una nueva salida deba ser generada. Como una consecuencia incrementar el numero de reglas incrementa el tiempo de evaluación de reglas. Esta evaluación es limitada por el tiempo de ciclo del sistema. El tiempo de ciclo es el tiempo necesario para ejecutar el código de captura de una entrada para generar una salida. Esto en turno, esta limitado por el teorema de muestreo. La frecuencia mas alta de la señales de entrada y de salida definen el tiempo de ejecución. El teorema de muestreo requiere que la frecuencia de muestreo deba ser al menos dos veces la frecuencia más alta de la señal muestreada. Por ejemplo, si la entrada es 1 Khz., la frecuencia de muestreo debe ser mínimo de 2 Khz. Esto requiere que el microcontrolador cuide un lazo de 1/2KHz o 500 microsegundos. No todos los 500 microsegundos serán asignados al proceso de inferencia. Durante este tiempo el microcontrolador debe capturar los datos de

entrada, ejecutar la inferencia y generar la salida. Los requerimientos de funcionamiento están limitados por el número de reglas a ser evaluadas y el periodo de muestreo en la entrada y salida, y la velocidad con que los periféricos de adquisición y control realizan sus operaciones.

Otro factor importante que afecta la frecuencia de muestreo es el tiempo requerido para transmitir el valor de las variables del sistema de control difuso a la IPVSID. Como se había mencionado (ver apartado 3.3.6.3), la rutina encargada de transmitir estos datos (TRANSMITIR_BUFFER), se ejecuta al final del proceso de control difuso (adquisición, inferencia difusa, condicionamiento de salida) y su ejecución toma cierto tiempo el cual es agregado al proceso de control difuso y por tanto también limita la frecuencia de muestreo y los requerimientos del sistema.

Todos estos factores deben ser tomados en cuenta, en cuanto a la aplicabilidad del SID ya que solo podrá cubrir aplicaciones que no exijan más de los requerimientos que el ofrece.

Tomando en cuenta los factores mencionados, las frecuencias de muestreo seleccionables se limitaron a un rango de valores discretos que van desde 5 MPS (Muestras Por Segundo) hasta 200 MPS. La lista de los posibles valores en el rango anterior, fueron definidos según la Tabla 3.13:

Tabla 3.13 Lista de los valores validos de frecuencia de muestreo en el SID.

• 5 MPS
• 10 MPS
• 20 MPS
• 40 MPS
• 50 MPS
• 80 MPS
• 100 MPS
• 120 MPS
• 140 MPS
• 160 MPS
• 180 MPS
• 200 MPS

Por lo tanto el SID solo podrá cubrir aplicaciones que estén dentro de este rango de requerimientos en cuanta velocidad de procesamiento, es decir aquellas aplicaciones donde se cumpla el diagrama de tiempos ilustrado en la Figura 3.66.

En la siguiente sección se define el proceso de diseño, del control de la frecuencia de muestreo.

3.3.7.6.1 Diseño del programa y las rutinas del control de la frecuencia de muestreo.

De acuerdo con la Figura 3.66 cada vez que transcurre un periodo de muestreo se inicia una nueva ejecución del proceso de control difuso cuya duración debe ser menor que dicho periodo (dentro de un límite de seguridad), es decir debe terminar antes de iniciarse un nuevo periodo de muestreo y que en caso de no cumplirse, se pierde el control sobre la frecuencia de muestreo con lo que el sistema pude volverse inestable (situación muy riesgosa). Este proceso se repite mientras nos se aborte la ejecución del sistema de control difuso.

De acuerdo a las características del PCD, el control de la frecuencia de muestreo se hizo por medio de la consulta por poleo de una bandera que se definió con el nombre de BANDERA_MUESTREO (ver Figura 3.29, apartado 3.3.7), la cual permanece a 0 mientras no se complete un periodo de muestreo, momento en el cual se pone a 1, lo que indica que se ha completado un periodo de muestreo y que ha comenzado uno nuevo, con lo que se debe ejecuta una vez mas el proceso de control difuso. La operación del segmento del PCD que se encarga de este control es mostrado en la Figura 3.67

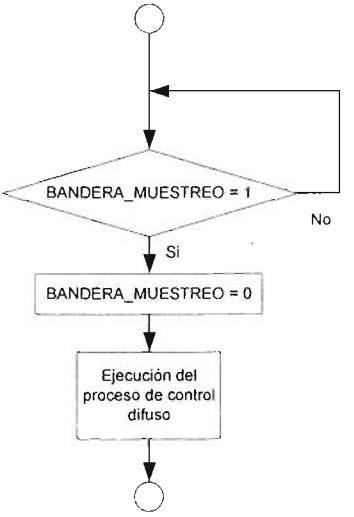


Figura 3.67 Operación del control de la frecuencia de muestreo dentro del programa de control difuso.

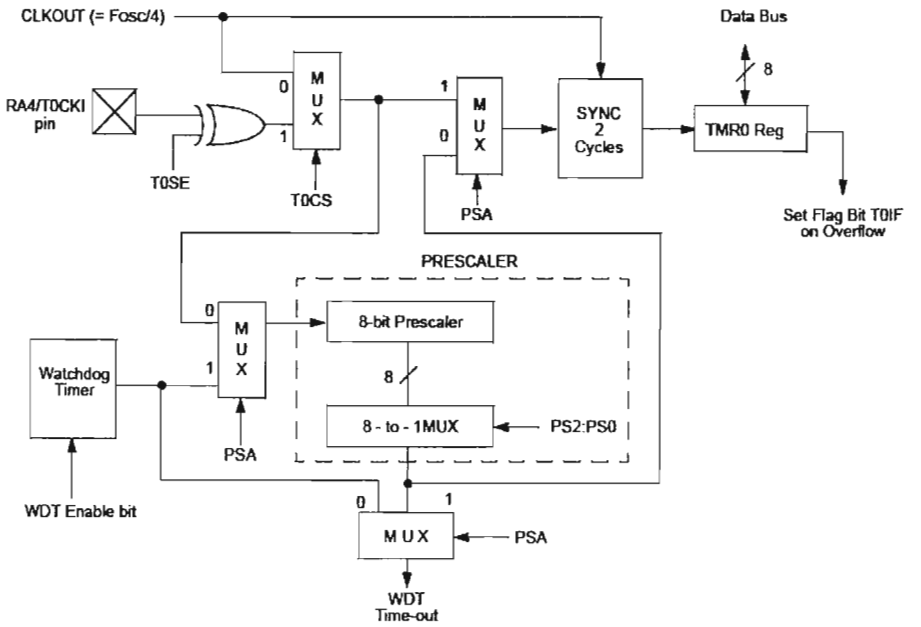
El la Figura 3.67 se ve que una vez que se cumple un periodo de muestreo, es decir la bandera se pone a 1, antes de proceder a ejecutar el proceso de control difuso se pone una vez mas a 0, con el propósito de detectar la terminación del

periodo de muestreo actual y el comienzo del nuevo (con la nueva puesta a 1 de dicha bandera).

El control de tiempos para poner a 1 esta bandera debe ser independiente del programa principal, lo cual se consiguió haciendo uso del timer 0 del PIC. Para comprender el diseño de la rutina de control de tiempos se describe resumidamente el funcionamiento de timer 0.

El timer 0 es un contador / temporizador de 8 bits. El registro principal de este módulo es TMR0 [b] (01h y 101h en RAM). Este registro se incrementa continuamente a una frecuencia seleccionable manejada por un preescalador y el reloj interno Fosc/4 (modo temporizador) o bien, por un preescalador y una señal externa (modo contador).

En la Figura 3.68 se muestra un diagrama de bloques de este módulo, en donde se indican los bits que afectan su operación y la manera en que lo hacen.



Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION_REG<5:0>).

Figura 3.68 Diagrama de bloques del preescalador del TMR0/WDT

El modo temporizador. En el modo temporizador la señal de reloj que controla el incremento del registro TMRO es la frecuencia *Fcy = Fosc/4*, la cual puede ser dividida opcionalmente por el preescalador si así se desea. Como se puede ver en la Figura anterior, este modo es seleccionado al limpiar el bit T0CS (bit 5 del OPTION_REG). En este modo el contenido del registro TMR0 se incrementara a la frecuencia Fcy dividida de acuerdo al preescalador

El modo contador no es descrito, ya que no es utilizado en el trabajo presente.

El registro TMR0 se incrementa continuamente en cualquiera de sus dos modos, desde 00h hasta FFh y en la siguiente cuenta se reinicia con 00h y así sucesivamente.

Al momento del reinicio se activa la bandera T0IF (bit 2 de INTCON) poniéndose a 1. Esta activación puede usarse de dos maneras:

a) para solicitar una interrupción.

b) para ser consultado por poleo.

En ambos casos debe tenerse en cuanta que para detectar un activación (un 1) en esta bandera, previamente habrá que limpiarla por software. Esto debe realizarse en la inicialización del timer y después de que un reciclo lo ha activado. Lo último puede hacerse en la rutina de atención a la interrupción, que es el caso que aquí concierne.

El preescalador es un divisor de frecuencias seleccionable. Como se puede ver en la figura 3.68, el preescalador esta compartido entre el Timer 0 y el módulo watchdog, sin embargo solo puede conectarse a uno de lo dos y esto se establece mediante el bit PSA (OPTION_REG<3>), así, con ese bit en alto el preescalador es asignado al reloj del watchdog, mientras con un nivel bajo en PSA el preescalador dividirá la frecuencia que maneja el Timer 0.

La selección del módulo (valor de división de frecuencia) del preescalador se puede realizar mediante los bits PS2, PS1, PS0 (OPTION_REG<2:0>) de acuerdo a la siguiente tabla:

Tabla 3.14 Bits de Selección del preescalador del timer 1.

PS2 PS1 PS0	Divisor (Timer 0)	Divisor(WDT)
0 0 0	1/2	1/1
0 0 1	1/4	1/2
0 1 0	1/8	1/4
0 1 1	1/16	1/8
1 0 0	1/32	1/16
1 0 1	1/64	1/32
1 1 0	1/128	1/64
1 1 1	1/256	1/128

En base a lo antes expuesto las temporizaciones que se pueden hacer con el Timer 0 esta dadas por:

$T = N * Tcy * M$ ..... (3.26)

Donde:

N es el numero de incrementos o cuentas del timer 0 es decir $N = valor\_cargado\_en\_TMRO + 1$ ..... (3.27)

Tcy es el periodo de instrucción que esta dado por $4/Fosc$ donde $Fosc$ es la frecuencia de oscilación del reloj externo al cual trabaja el PIC.

M es el inverso del preescalador es decir $M = 1/p$

De la ecuación anterior tenemos que el valor que se debe cargar en el TMR0 para un temporización esta dada por:

$$TMR0 = \frac{T}{Tcy * M} - 1 \text{ ..... (3.28)}$$

Cabe destacar que el valor real que se cargara en TMR0 no es el resultante de esta ecuación si no mas bien el complemento a 1 del valor resultante, de esta manera al llegar el numero de impulsos deseados el timer 0 se desborda y al pasar a 00h la bandera T0IF se activa para hacer una petición de interrupción.

La frecuencia de muestreo simplemente estará dada por, $F\_muestreo = 1/T$ (3.29)

De acuerdo a las ecuaciones 3.26 y 3.29, la frecuencia de muestreo mínima que se puede obtener es el inverso de la temporización máxima que se puede lograr con el timer 0, esta es alcanzada, con la selección de un preescalador de 1/256 y 256 cuentas para el Timer 0, por lo tanto con un reloj de 20MHz se tiene que:

$N = 256$

$Tcy = 4/Fosc = 4/20x10^{6} = 200x10^{-9}$

$$M = 1/p = \frac{1}{1/256} = 256$$

$T = 256 * 200x10^{-9} * 256 = 13.1072ms$

$$F\_muestreo = \frac{1}{13.1072ms} = 76.2939Hz$$

Con lo que el valor que debe cargarse en TMR0 para producir la ultima frecuencia debe ser:

$TMR0 = N - 1 = 256 - 1 = 255$

Como el timer 0 es un contador ascendente el último valor no es el que debe ser cargado realmente si no mas bien su complemento a 1, de esta manera el complemento a 1 es:

$\sim TMR0 = 255 - 255 = 0$

Que es el valor se debe cargar al TMR0 para obtener la frecuencia de muestreo mínima con un reloj de 20Mhz y un preescalador de 1/256.

Como se vio la frecuencia mínima que se puede obtener con el timer 0, con un reloj de 20MHz es de 76.2939 Hz, con lo que no se podría definir las frecuencias de muestreo de 50 Hz, 40 Hz, 20 Hz, 10 Hz, 5 Hz consideradas en el apartado 3.3.7.6.

Para poder definir estas frecuencias basta observar que estos valores pueden ser submúltiplos de otras frecuencias superiores que si se pudieran obtener con el timer 0. Lo mas adecuado es definir un valor de frecuencia que se múltiplo de las 5 frecuencias anteriores. Esto valor se calcula obteniendo el mínimo común múltiplo de estas frecuencias, que es 200 Hz.

Con una frecuencia de 200 Hz tenemos un periodo o temporización de 5ms, la cual se obtiene cargando el valor adecuado al TMR0, el cual se calcula aplicando la ecuación (3.28) con un preescalador de 1/256 y con el reloj de 20 MHz, como sigue:

$$TMR0 = \frac{5x10^{-3}}{200x10^{-9} * 256} - 1 = 96.6563 \approx 97$$

Complementando a 1

$$\sim TMR0 = 255 - 97 = 158$$

que es el valor que se debe cargar al registro TMR0 del timer 0 para lograr una frecuencia de 200 Hz.

Ahora bien si lo que se quiere es obtener una frecuencia de 50 Hz a partir una de 200 Hz se debe obtener el factor de escala de estos dos valores, simplemente dividiendo la frecuencia de 200 Hz entre la de 50 Hz, que da como resultado un valor de 4, que indica que el valor de temporización de 5ms para una frecuencia de 200 Hz se debe multiplicar por este valor o lo que es lo mismo sumarse el mismo cuatro veces para obtener una temporización de $4 \times 5 = 20ms$, que es el periodo para un frecuencia de 50 Hz. La misma operación se aplica para las obtener las frecuencias restantes.

Por lo tanto la técnica consistirá en sumar tantas veces como indique el factor de escala, la temporización de 5ms, entonces una vez completada esta cuenta, significara que el tiempo de muestreo ha transcurrido y la bandera BANDERA_MUESTREO será activada, para indicar en el segmento designado del programa de control difuso que el periodo de muestreo se ha completado, por lo que una vez mas, el proceso de control difuso debe ser ejecutado. Esto se puede lograr fácilmente definiendo un contador que se inicialice con el factor de escala y que a cada desborde o fin de cuenta del timer 0 se decremente en una unidad, mientras no sea cero, momento en el cual se habrá completado el periodo de muestreo para un frecuencia dada. Cabe mencionar que para una frecuencia de 200 Hz el valor del contador debe ser iniciado con un valor de 1, es decir solo se debe contar una vez la temporización de 5ms, esta operación también se aplicara a aquellas frecuencias de la lista que se puede obtener directamente con un valor de carga en timer 0, es decir frecuencias mayores de 77 Hz.

Con base en la información anterior el diagrama de flujo resultante para la rutina de atención a la interrupción (rutina de control de tiempos) provocada por el desborde del timer 0 y consecuentemente involucrada en el control de la frecuencia de muestreo, es el indicado en la Figura 3.69.

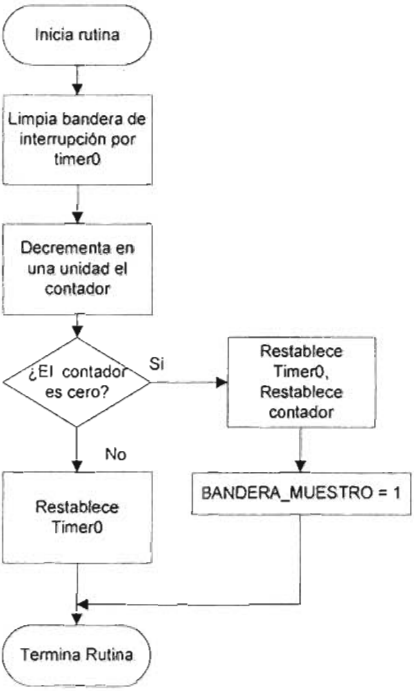


Figura 3.69 Diagrama de flujo de la rutina de atención a la interrupción por desborde del timer 0 (EJECUTAR_PROCESO_DIFUSO)

Del diagrama de flujo se observa que el valor de carga del timer 0 para una temporización predefinida se debe restablecer cada vez que se desborda, así como el contador de desbordes se debe restablecer cada vez que llega a su valor de cero (se completa la temporización), por lo tanto se necesita tener almacenado en otras localidades de memoria este valor, para cargarlo nuevamente en ambos registros de control, cada vez que se necesite. Para tal motivo se reservaron dos localidades de memoria RAM que se llamaron T_MUESTREO y T_CUENTA donde la primera almacena el valor de carga del timer 0 y el segundo el numero de veces que se cuentan los desbordes del timer 0.

Estas variables se pueden cargar a un valor constante o fijo para una frecuencia de muestreo predefinida, y el sistema de control difuso se mantendría a esta frecuencia indefinidamente. Esto ofrece poco flexibilidad ya que para cambiar la frecuencia de muestreo seria necesario suspender la ejecución actual del proceso de control difuso (abortar el modo ejecución), definir la nueva frecuencia de muestreo y volver a ejecutar el proceso de control difuso. Para evitar este constante encendido y apagado se opto por que estas variables se pudieran

modificar en línea o en otras palabras que se pudiera modificar la frecuencia de muestreo mientras el proceso de control difuso es ejecutado. La única manera de modificara el valor de estos parámetros en tiempo de ejecución es enviándoles un nuevo valor desde la IPVSID, es decir su dirección en la memoria RAM y su nuevo valor, y el método de recepción del modo ejecución (ver artado 3.3.6.2), será el encargado de actualizar su valor. De esta manera en la parte correspondiente de la rutina de atención a interrupción por desborde del timer 0, este y el contador de desbordes se reestablecerían con lo nuevos valores de T_MUESTREO y T_CUENTA, lo cuales definen la nueva frecuencia de muestreo.

En resumen las partes de programa y rutinas que se ejecutan para lograr el control de la frecuencia de muestreo son:

a) La rutina FIJAR_T_MUESTREO de configuración e inicialización del Timer0 y demás variables utilizadas en el control de la frecuencia de muestreo (ver apartado 3.3.6.3)

b) El segmento del programa de control difuso encargado del poleo de la BANDERA_MUESTREO para el detectar el momento en que se ha completado el periodo de muestreo y comenzado otro y ejecutar una vez más el proceso de control difuso.

c) La rutina de atención a la interrupción por desborde del Timer 0, que es la encargada de controlar el tiempo de muestreo independientemente del programa principal.

3.4 Interface de Programación y Visualización del Sistema Inteligente Difuso.

La Interface de Programación y Visualización del Sistema Inteligente Difuso (IPVSID) es una herramienta de interface grafica de usuario (GUI, por sus siglas en ingles) para llevar acabo el diseño de un controlador difuso, que incluye la edición de entradas y salida del controlador, edición de las funciones de membresía de las entradas y de las funciones crisp de la salida, la edición del conjunto de reglas difusas y finalmente la generación del código asociado a dicho diseño para que posteriormente sea descargado a la TEPDES par ser ejecutado. Además en dicha interface, durante la ejecución del controlador difuso diseñado se despliega una ventana para la monitorización y visualización del estado de las entradas y la salida del controlador, así como el de la salida del proceso controlado con la posibilidad de cambiar desde dicha ventana los parámetros de frecuencia de muestreo y Set Point del sistema de control, todo los anterior con el fin de poder observar el comportamiento del sistema de control entero y en base a esto realizar las ajustes necesarios para poner a punto el sistema.

La IPVSID se compuso de dos partes:

1. El Editor de Sistemas Difuso (ESD).
2. La Interface de Monitorización y Configuración (IMC)

3.4.1 Editor de Sistema Difusos.

El Editor de Sistemas Difusos fue desarrollado en el lenguaje de programación Visual Basic 6 bajo el sistema operativo Windows XP. Este editor es un paquete software que proporciona varias herramientas GUI para el desarrollo de aplicaciones difusas en el SID, las cuales son:

- El editor de las entradas y la salida del controlador difuso, basado en la aplicación.
- El editor de funciones de membresía tanto para las entradas como para la salida, basado en el conocimiento de la aplicación.
- El editor para crear reglas difusas.
- El generador de código de la base de conocimiento y el algoritmo de control difuso para ser descargada a la TEPDES.

El diagrama de bloques del ESD es mostrado en la Figura 3.70. Como puede verse, esta compuesto de cuatros formularios o herramientas de interface grafica de usuario para diseñar, editar y generar el código de un controlador difuso en el SID para un aplicación de control difuso particular. Estos formularios están enlazados automáticamente, ya que los cambios que se hagan a un diseño dado en uno de ellos, pueden afectar lo que se ve en cualquiera de los otros formularios abiertos. Se puede tener alguno o todos los formularios abiertos para una diseño dado.

El editor de las entradas y la salida del controlador difuso manipulan las ediciones de alto nivel del sistema. Se fijan el número, tipo, nombre y rango de las entradas del controlador difuso. Para la salida del controlador difuso se define su tipo, nombre, rango y el tipo de carga que manejara. Que en conjunción definen el esquema de control difuso seleccionado (ver apartado 3.3.7.2)

El editor de funciones de membresía es usado para definir el número, la forma, y el valor de los parámetros de las funciones de membresía (funciones trapezoidales) de las entradas y la salida (funciones singletons constantes).

El editor de reglas difusas es para editar el conjunto o lista de reglas de control difuso que definan la estrategia de control para una aplicación particular.

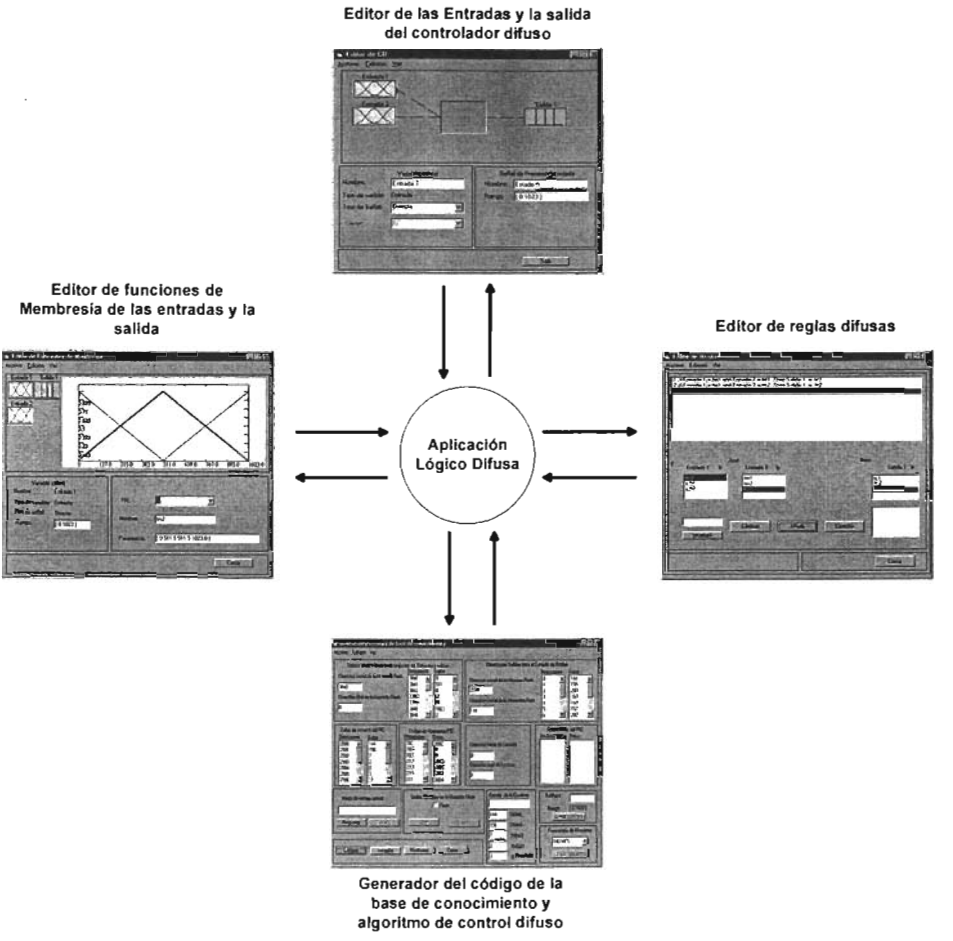


Figura 3.70 Diagrama de bloques del Editor de Sistemas Difusos.

El generador de código se utiliza para generar la base de conocimientos que incluye la definición de las funciones de membresía de las entradas y la salida en una estructura de datos definida para el microcontrolador (ver apartado 3.3.7.4.1); la definición de la lista de reglas difusas en un formato de datos para el microcontrolador (ver apartado 3.3.7.4.2) y la generación del código de operación de las instrucciones del sistema de control difuso para un diseño dado que incluyen las instrucciones de inicialización de los periféricos de control de potencia, la instrucciones de adquisición de datos (ver apartado 3.3.7.3), la instrucciones de inferencia difusa (ver apartado 3.3.7.4) y finalmente las instrucciones de condicionamiento de salida (ver apartado 3.3.7.5). Una vez generado el código anterior, desde este mismo formulario es descargado a la TEPDES para ser ejecutado.

La versión actual del ESD permite aplicaciones de control difusas con:

- Hasta 3 entradas.
- Hasta 1 Salida.
- Hasta 9 funciones de membresía por entrada.
- Hasta 9 funciones de membresía por salida.
- Hasta 729 reglas.
- Hasta tres antecedentes y 1 consecuente por regla.
- 11 caracteres por nombre.
- Funciones de membresía trapezoidales o triangulares.
- Funciones de membresía de salida singleton.
- Esquemas de control P, PD, PI y PP.

3.4.2 Interface de Monitorización y Configuración.

La Interface de Monitorización y Configuración fue desarrollada en el programa de instrumentación virtual LabWindows 5.0.1 para desarrolladores en C, bajo el sistema operativo Windows XP. La IMC es un programa que adquiere y visualiza continuamente el estado de las entradas y la salida de un controlador difuso diseñado para una aplicación particular, así como el de la salida o variable controlada del proceso u objeto bajo control en la misma aplicación. Paralelamente desde esta IMC el diseñador tiene la posibilidad de enviar algunos comandos o parámetros (frecuencia de muestreo y Set Point) que modifican el funcionamiento del sistema de control particular.

Este programa es invocado desde la ventana de generación de código del editor de sistemas difusos, y se enlaza a el, a través de la lectura y escritura de un conjunto de ficheros de texto. Desde este programa se activa o se detiene la ejecución de la aplicación de control difuso diseñada en el editor de sistemas difusos.

El diseño e implementación de la IMC esta compuesta por un conjunto de objetos llamados instrumentos virtuales, donde cada uno realiza una función específica dentro del programa principal, estos instrumentos tienen asociada una función de servicio repetido codificada en C, la cual es ejecutada cada vez que se generan ciertos eventos sobre ellos. El diagrama de bloques general del programa de la IMC es mostrado en la Figura 3.71.

La descripción de cada uno de los componentes más importantes de la interface grafica de usuario es dada en seguida.

Panel Principal. Es la parte de la Interface Grafica de Usuario donde se alojan los instrumentos virtuales que componen la aplicación. Una función de servicio repetido es ejecutada cuando dicho panel es cargado, la cual tiene la primera finalidad de establecer un enlazado con el ESD, de tal modo que se le comunique el número, nombre, rangos y tipo de entradas del controlador difuso diseñado así

como los de la salida; todo con el objetivo de que la IMC se prepare para adquirir y visualizar de manera correcta las variables del proceso de control difuso que se va a poner en ejecución, además también crea un fichero de texto en con el mismo nombre del diseño en cuestión, en la misma carpeta donde esta guardado, para almacenar los datos adquiridos.

La segunda tarea que realiza es la inicialización del puerto RS-232 de la PC para establecer las comunicaciones entre la IPVSID y la TEPDES a fin de poder hacer la adquisición de los datos del sistema de control difuso y enviar los comandos de configuración correspondientes. Los parámetros de funcionamiento fueron definidos en base las características de configuración de las comunicaciones de la TEPDES y en función de los datos que se deben recibir. Estos parámetros de configuración son mostrados en la Tabla 3.15:

Tabla 3.15. Parámetros del puerto de comunicaciones

Numero puerto de comunicaciones	COM 1
Velocidad de comunicación	57, 600 baudios
Tamaño buffer de transmisión	512 bytes
Tamaño Buffer de recepción	Variable *
Dato	8 bits
Bits de parada	1 bit
Paridad	Ninguna

** Es variable ya que depende del número de entradas que tenga el controlador difuso.*

Control de Ejecución. Es un interruptor binario que Controla la activación y desactivación del proceso de control difuso. Cada vez que cambia de estado (pasa del estado de apagado al de encendido y viceversa) genera un evento el cual es atendido por la respectiva función de servicio repetido. Si el estado al que ha pasado es el de encendido la función de servicio repetido envía a la TEPSDES el comando de inicio de Modo Ejecución, el parámetro de Frecuencia de muestreo inicial, y el Set Point inicial, para poner en ejecución el sistema de control difuso correspondiente a un diseño dado y además abrir el fichero de texto previamente creado en el panel principal, para almacenar los datos adquiridos, en caso contrario simplemente manda el comando de fin de modo ejecución para suspender la ejecución del proceso de control difuso actual y cierra el fichero abierto.

Frecuencia de Muestreo. Es un botón de comando para cambiar el parámetro de funcionamiento de frecuencia de muestreo del sistema de control difuso desde

la IPVSID mientras esta en ejecución. Los datos de frecuencia de muestreo son transmitidos en el momento en que se pulsa dicho botón (se ejecuta la función de servicio repetido correspondiente), los cuales dependen del valor de frecuencia previamente seleccionada de la lista de frecuencias de muestreo validas para el SID (ver apartado 3.3.7.6).

Set Point. Es un botón de comando para cambiar el parámetro de funcionamiento del Set Point del sistema de control difuso desde la IPVSD mientras este se ejecuta. Los datos para un Set Point valido seleccionado son transmitidos en el momento que se pulsa dicho botón. Los valores de Set Point son seleccionados de un control numérico para dicho propósito y el rango de valores depende de la aplicación particular.

Adquisición de datos. La función de servicio repetido del puerto de comunicación es invocada cada vez que se reciben un número de bytes especificado que depende del número de variables del sistema de control difuso que van a ser adquiridas, esta recepción se repite periódicamente de acuerdo con la frecuencia de muestreo actual. En la función de servicio repetido se realiza la lectura de los datos digitales de las variables de proceso de control difuso, cada variable es representada por 2 bytes concatenados, de los cuales solo son validos los diez bits de menos peso, que son procesados y acondicionados, para ser desplegados en los monitores de registro, para visualizar su estado en tiempo real. Estos datos también son escritos en el fichero de texto creado en el panel principal, con el objetivo de que estén disponibles para un análisis posterior.

Monitores de registro. Los monitores de registros son un conjunto de controles gráficos donde puede ser observado el estado y el comportamiento de cada una de las variables del sistema de control difuso en curso, estos solo funcionan como indicadores y no tienen ninguna función de servicio repetido asociada, y su valor es actualizado en la función de servicio repetido de la adquisición de datos. Por omisión la IMC solo tiene tres monitores, considerando un controlador difuso de una entrada y una salida, sin embargo en caso de que el diseño tenga más de una entrada, hasta un máximo de tres, se agrega automáticamente un monitor más por cada entrada adicional.

Lista de frecuencias de muestreo. Es un control numérico de lista, el cual despliega la lista de valores de frecuencia de muestreo validos en el SID (ver apartado 3.3.7.6) para que el usuario seleccione la deseada para el sistema de control difuso actual. Este valor seleccionado es tomado para enviar los datos correspondientes de frecuencia de muestreo con el botón de comando correspondiente. Es un control indicador, por lo tanto no tiene ninguna función de servicio repetido asociada.

Selector de Set Point. Se utiliza para seleccionar un nuevo Set Point para un proceso controlado dado. Los valores seleccionables se restringen a un conjunto de valores enteros que están dentro del rango de la variable controlada (salida del proceso controlado). Es un control indicador, por lo tanto no tiene ninguna función de servicio repetido asociada.

El manejo, y funcionamiento del ESD y la IMC es explicado en el siguiente capitulo.

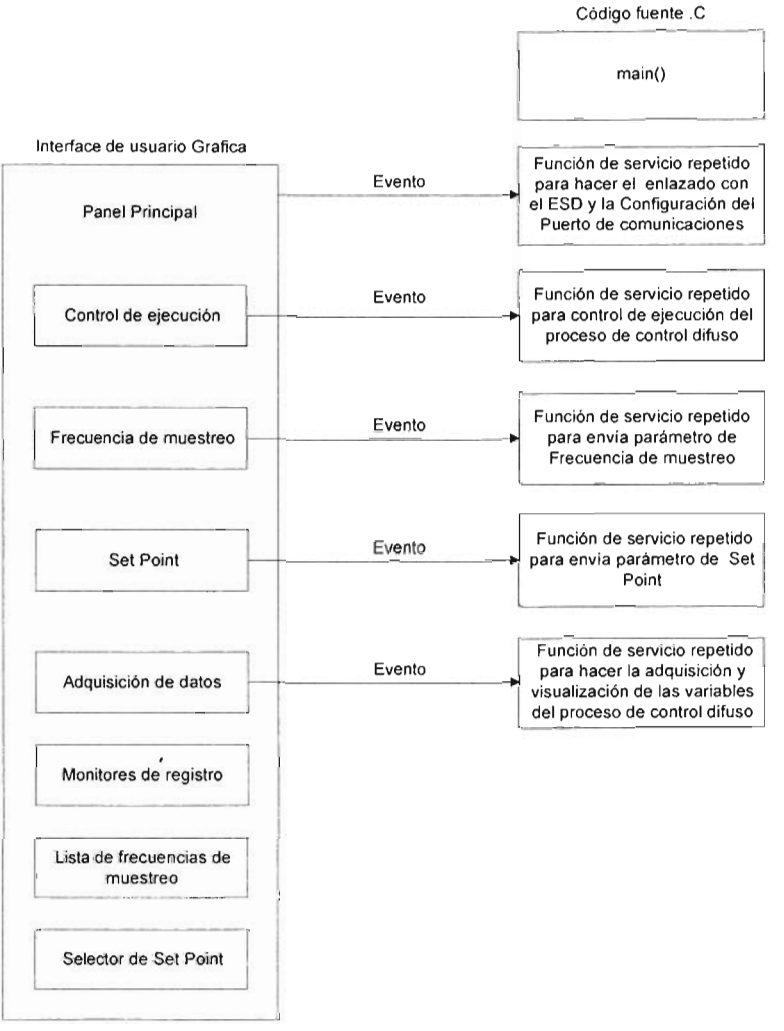


Figura 6.71 Diagrama de bloques de los principales componentes de la Interface de Monitorización y Configuración.

Nota: Los detalles de diseño y programación del programa software de la IPVSID, no son incluidos, mas pueden ser solicitados al autor en la dirección de correo electrónico, ***fermi14@hotmail.com.***

CAPÍTULO 4. IMPLEMENTACIÓN, PRUEBAS Y RESULTADOS.

4.1 Implementación del prototipo diseñado.

La implementación del prototipo diseñado se llevo acabo en el Laboratorio de manufactura Inteligente en el Centro de Ciencias Aplicadas y Desarrollo tecnológico de la UNAM. Esta básicamente constituida por:

a) La construcción de la tarjeta de circuito impreso de la TEPDES y el montaje de cada uno de sus elementos.

b) La instalación de la IPVSID para ejecutarse en una PC y la interconexión de la misma con la TEPDES.

4.1.1 Implementación de la TEPDES.

En la Figura 4.1 se observan la distribución de los distintos elementos y dispositivos electrónicos que constituyeron la TEPDES sobre la tarjeta de circuito impreso diseñada. Esta está constituida de dos tarjetas, una donde se alojo la fuente de alimentación, el modulo de comunicaciones y el microcontrolador de procesamiento y su monitor de estado, y otra donde se alojaron el modulo de entrada y el modulo de salida de potencia en CC y CA, ambas se encuentran interconectadas por medio de un cable de 11 líneas.

Los principales conectores y elementos de entrada y de salida de la tarjeta son:

Modulo de entrada.

El modulo de entrada esta compuesta por tres conectores molex (tres entradas) de tres pines de 3.9 mm, para recibir las señales analógicas del proceso bajo control y adaptarlas a los niveles y formatos de señal internos del CPU (microcontrolador). Este ejecuta el programa de control difuso y en base a el se producen la señales de comando o de control. Uno de los pines corresponde a la entrada y los otros dos se utilizan para suministrar una polarización de +5V a los sensores o emisores de las señales del proceso bajo control (ver apartado 3.2.4, capítulo 3).

Modulo de salida para dispositivos de CA.

El modulo de salida para cargas de CA esta compuesta por dos clavijas bipolares hembras de 15 A – 250 V que sirven para interconectar los dispositivos de CA (elementos finales de control) a los que se les controlara el suministro de potencia. En otras palabras se toman las señales de respuesta elaboradas por el CPU (microcontrolador) y se realiza una conversión en sentido contrario (Al modulo de entrada), es decir a partir de los niveles internos de la salidas de control se producen lo niveles adecuado para alimentar a los elementos finales de control en CA (ver apartado 3.2.6).

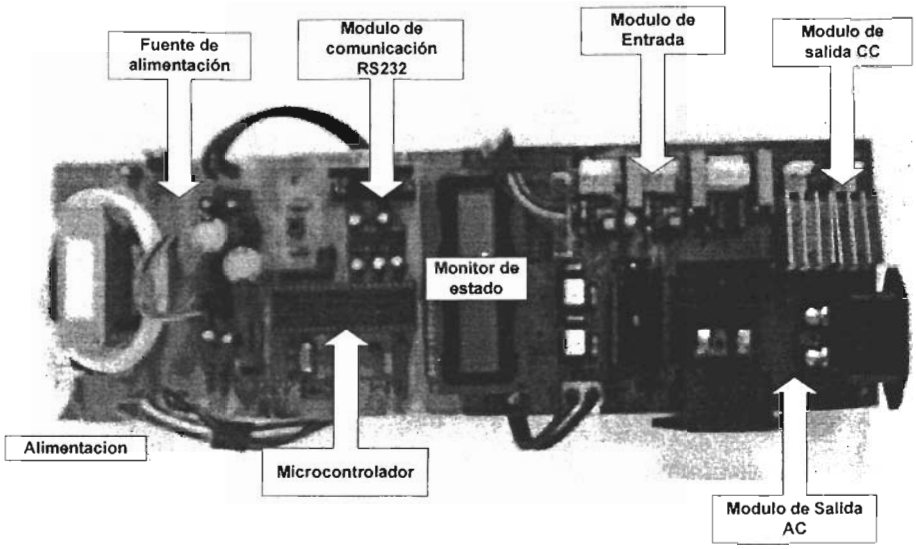


Figura 4.1 TEPDES

Modulo de salida para dispositivos de CC.

La salida para cargas de CC esta compuesta por dos conectores molex de dos pines de 3.9 mm, para interconectar las cargas o dispositivos de CC a los que se les controla el suministró de potencia (ver apartado 3.2.5 del capítulo 3).

Alimentación.

Es un cable con una clavija bipolar de 15 A - 127V en uno de sus extremos para conectar la TEPDES con la toma de corriente de 127 V. (ver apartado 3.2.7 del capítulo 3).

RS232.

Es un conector metálico DB9 para interconectar la TEPDES y la PC donde se ejecuta la IPVSID, a fin de poder establecer la comunicación vía RS232 entre ambos elementos, comunicación que incluye la recepción de comandos y parámetros del funcionamiento de la TEPDES y la transmisión de información del proceso difuso cuando este, está en ejecución (ver apartado 3.2.3 del capítulo 3).

Monitor de estado.

Le monitor de estado es un módulo LCD de dos líneas de 16 caracteres, que tiene la función de comunicar al usuario desde la TEPDES, el estado o modo de operación en el que se encuentra el SID.

4.1.2 Instalación de la IPVSID.

La IPVSID fue instalada en una PC bajo el sistema operativo Windows XP. Esta es un programa de Interface Grafica de Usuario (GUI, por sus siglas en ingles) que consta de dos componentes:

1. El Editor de Sistemas Difuso (ESD).
2. La Interface de Monitorización y Configuración (IMC)

El ESD es un programa de GUI diseñado y desarrollado en Visual Basic 6 para la definición y edición de un controlador difuso para el control de procesos basados en CC y CA. Este programa fue corrido bajo el sistema operativo Windows XP, mostrando bajo esta plataforma un buen desempeño para la aplicación. Las herramientas que incluye son:

- El editor de las entradas y la salida del controlador difuso, basado en la aplicación.
- El editor de funciones de membresía tanto para las entradas como para la salida, basado en el conocimiento de la aplicación
- El editor para crear reglas difusas.
- El generador de código de la base de conocimiento y el algoritmo de control difuso para ser descargados a la TEPDES.

La GUI de cada una de estas herramientas es mostrada en las Figuras 4.2, 4.3, 4.4 y 4.5 respectivamente. Como se pede ver en general están compuestas de botones de comando para ejecutar diversas acciones de control y edición propias de cada GUI, etiquetas de mensaje para informar el cometido de cada control u objeto del GUI y otras para informar sobre el estado del diseño en curso, entradas de texto para cambiar los parámetros del diseño en curso, menús para seleccionar las diversas opciones de edición de cada GUI, y finalmente controles gráficos para observar el estado y definición del diseño en curso y seleccionar los diferentes componentes del mismo para su edición.

En cada una de la figuras se muestra el cometido de cada uno de lo elementos que conforman cada GUI, así como la acción que se lleva acabo cuando se realiza un evento con el mouse o teclado sobre ese elemento, en los apartados 4.3.2, 4.3.3, 4.3.4 y 4.4.5 se dan mas detallas acerca de la edición de un sistema de control difuso utilizando cada una de estas herramientas.

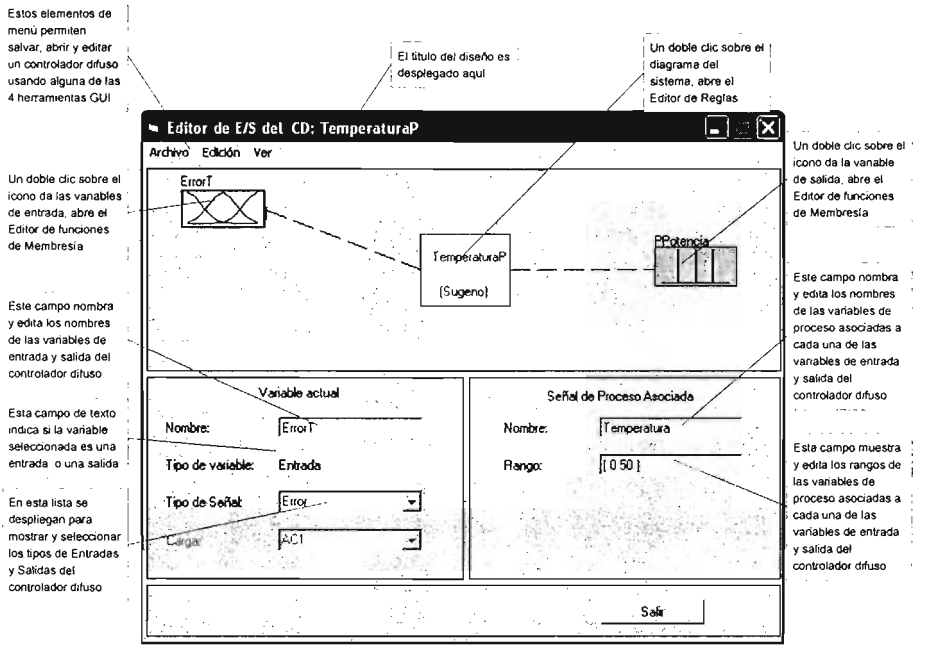


Figura 4.2 Editor de las Entradas y la Salida del Controlador Difuso.

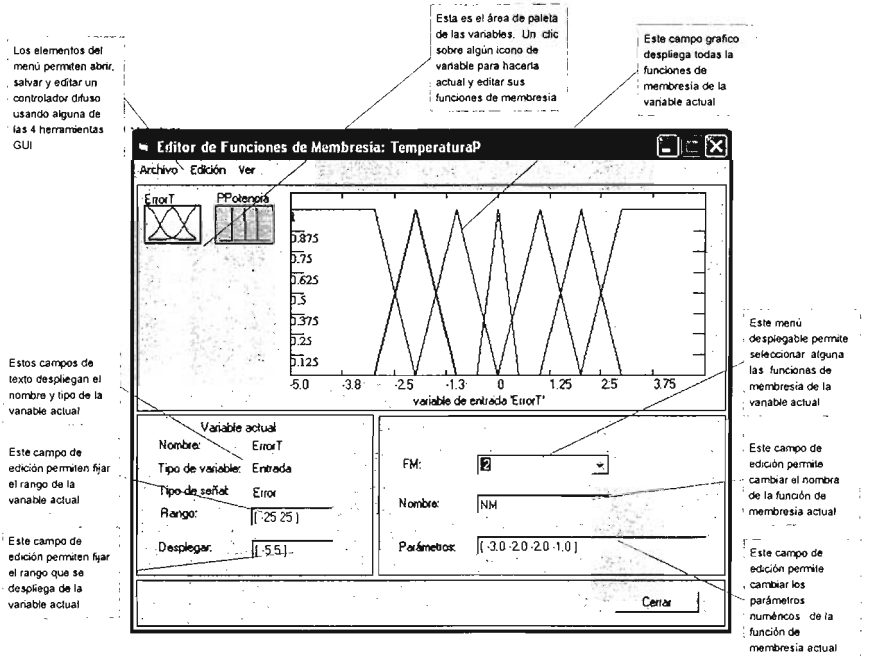


Figura 4.3 Editor de Funciones de Membresía.

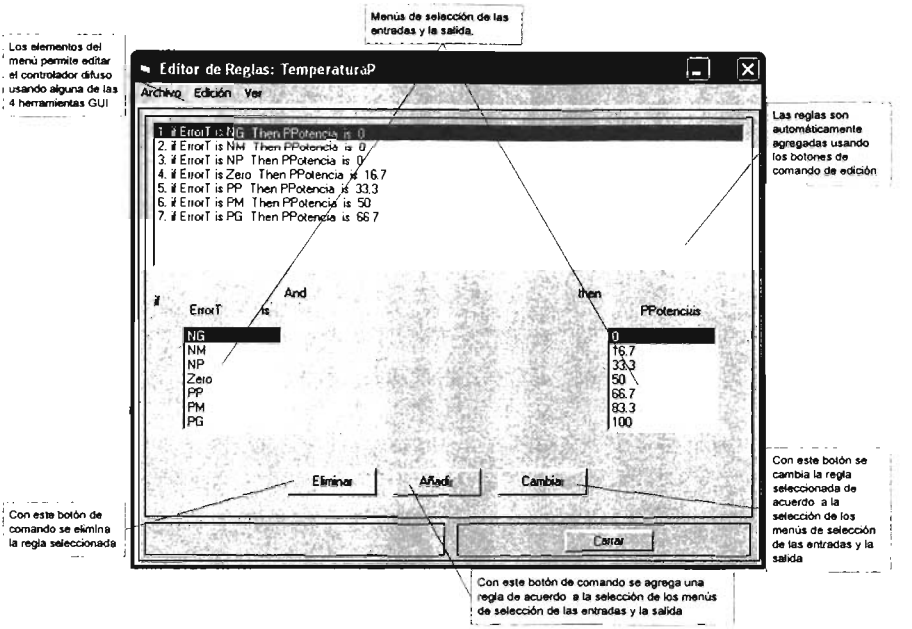


Figura 4.4 Editor de Reglas.

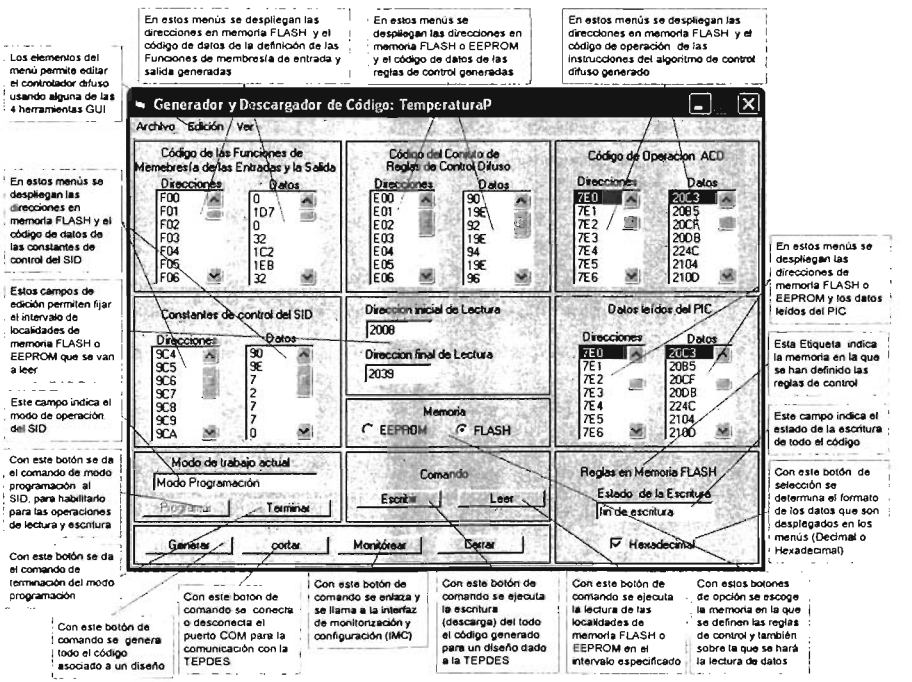


Figura 4.5 Generador y Descargador de Código.

La IMC es un programa GUI desarrollado en el programa de instrumentación virtual LabWindows 5.01, para configuración y monitorización de la ejecución del controlador difuso diseñado, directamente con el proceso u objeto a controlar. El programa fue corrido en el sistema operativo Windows XP, obteniendo bajo esta plataforma un desempeño aceptable y adecuado para la aplicación.

En la Figura 4.6 se muestra la IMC, como se puede observar esta compuesta de botones de comando para controlar y configura la ejecución del controlador difuso, mensajes de texto para informar sobre la utilidad y estado de cado uno de los otros componentes de la GUI, controles gráficos para mostrar el estado del sistema de control difuso y configurar la apariencia de la IMC.

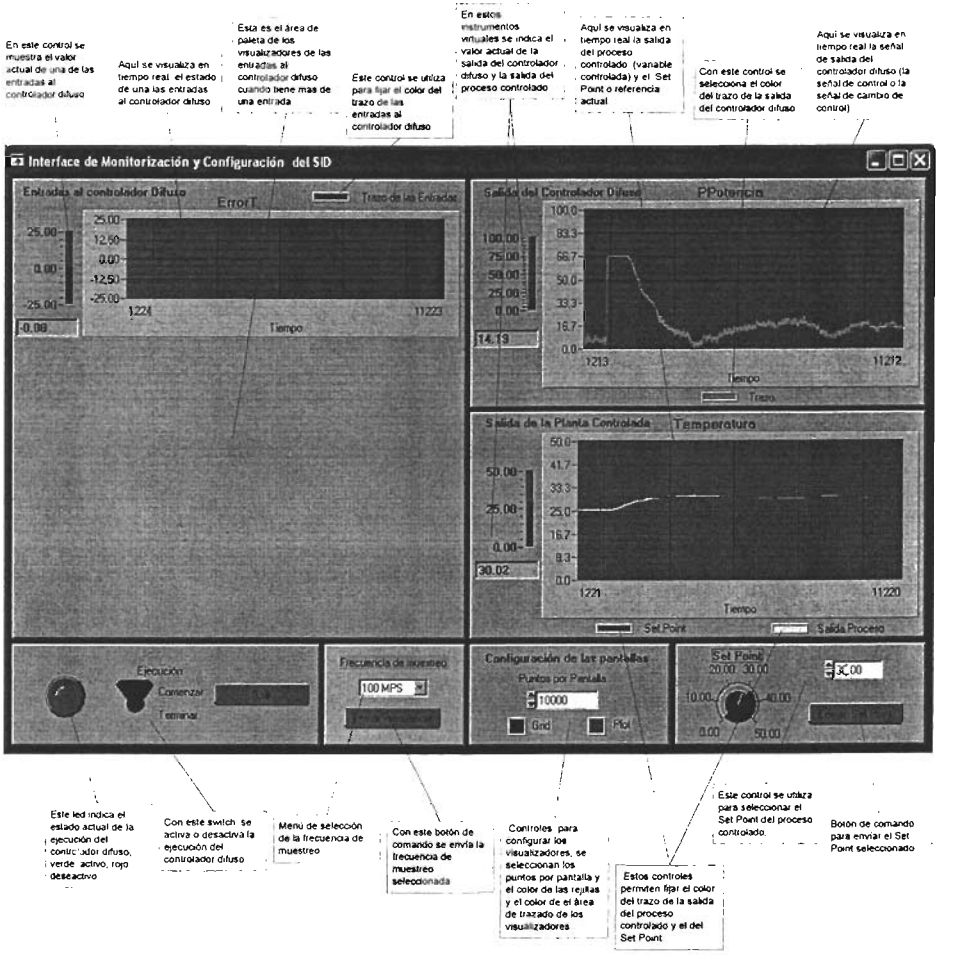


Figura 4.6 Interface de Monitorización y Configuración (IMC)

En la Figura 4.7 se muestran una fotografía del la instalación de la TEPDES junto con la PC en la cual se ejecuto la IPVSID.

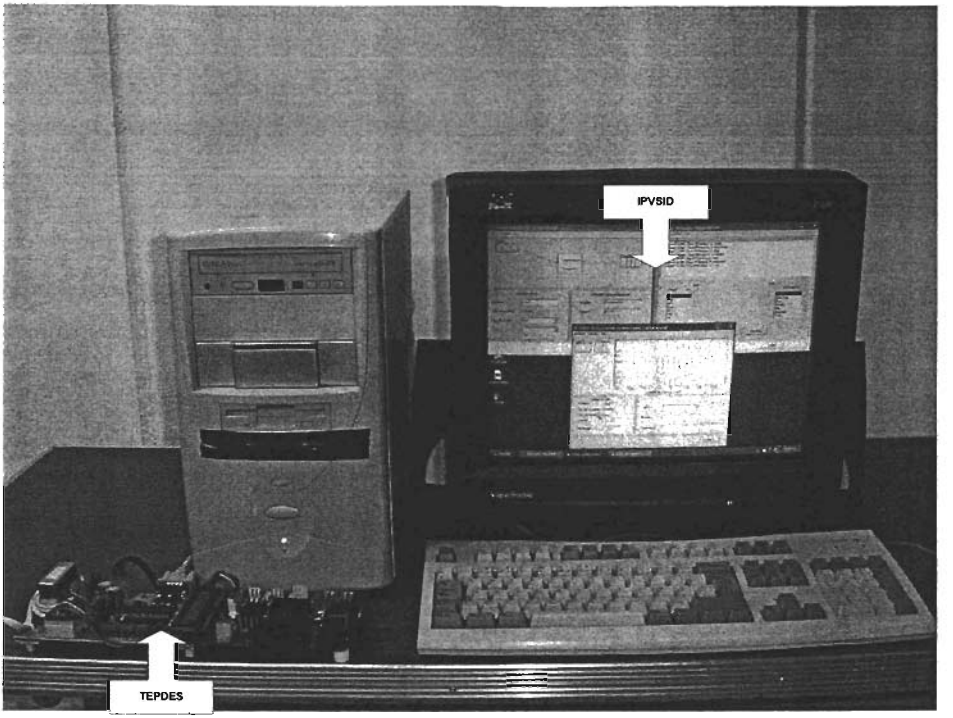


Figura 4.7 Instalación de la TEPDES y la IPVSID.

4.2 Especificaciones técnicas del prototipo desarrollado.

4.2.1 Especificaciones técnicas de la TEPDES.

Tabla 4.1 Especificaciones técnicas de la TEPDES.

Características	**Parámetros**
Entradas analógicas	3
Salidas de control	1
Salidas de CA	2
Salidas de CC	2
Resolución	Dato de 10 bits
Frecuencia de muestreo (*)	5, 20, 50, 80, 100, 120, 140, 160, 180, 200 muestras por segundo
Interfaz RS232 Full Duplex	57, 600 bits por segundo
Set Point (*)	Depende del rango de salida de la aplicación
Programa de control y configuración	GUI IPVSID

(*) Estos parámetros pueden ser cambiados a solicitud del usuario.

Tabla 4.2 Especificaciones Eléctricas máximas de la etapas de salidas en CC y CA.

Características	**Parámetros**
Corriente Máxima de cada salida CA	*15 A*
Voltaje Máximo de cada salida CA	*127 V* (monofásicos)
Corriente Máxima de cada salida CC	*400 mA*
Voltaje Máximo de cada salida CC	*12 V*

El modulo de salida para cargas de CC y CA tienen disponibles dos salidas, que son identificadas como DC1, DC2, AC1 y AC2, respectivamente. Estas salidas son identificadas en la Figura 4.8.

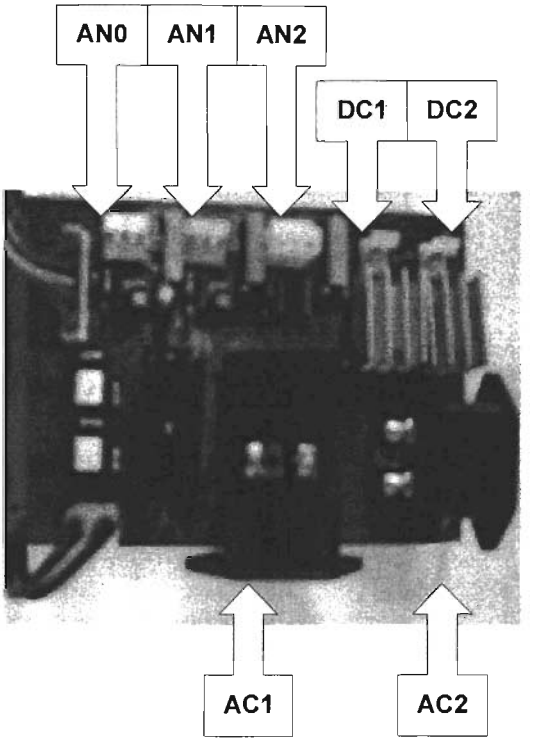


Figura 4.8 Localización de las salidas de potencia en CA y CC, y de las entradas analógicas en la TEPDES.

En la Figura 4.8 además de la localización de las salidas se muestra la localización y orden de las entradas analógicas.

De acuerdo a la definición del tipo de salida y tipo de carga para un diseño dado, se deben conectar las cargas o elementos finales de control (actuadores) en los conectores de salidas de los módulos correspondientes. A continuación se muestran los conectores disponibles de acuerdo al tipo de salida y carga.

Salida Estándar.

En la Tabla 4.3 se resumen las opciones de tipo de carga para la salida estándar (ver apartado 3.3.7.5.2)

Tabla 4.3 Opciones de carga para la salida Estándar.

• DC1
• AC1

Salida SNZP.

En la Tabla 4.4 se resumen las opciones de combinaciones de tipo de carga para la salida SNZP (ver apartado 3.3.7.5.3).

Tabla 4.4 Combinaciones de tipo de carga para la salida SNZP.

Caso	Valores Negativos	Valores Positivos
A	DC2	AC1
B	AC1	DC1
C	DC2	DC1
D	AC2	AC1

Salida CControl.

En la Tabla 4.5 se resumen las opciones de tipo de carga para la salida CControl (ver apartado 3.3.7.5.4).

Tabla 4.5 Opciones de carga para la salida CControl.

• AC1
• DC1

4.2.2 Especificaciones técnicas da la IPVSID.

4.2.2.1 Editor de Sistemas Difusos.

Tabla 4.6 Especificaciones técnicas del ESD.

• Sistema operativo XP
• Editor de entradas y salida del controlador difuso.
• Editor de las funciones de membresía de entrada y salida
• Editor del conjunto de reglas de control difuso
• Generador y descargador del código del controlador difuso diseñado
• Hasta 3 entradas.

• Hasta 1 salida.
• Hasta 9 funciones de membresía por entrada.
• Hasta 9 funciones de membresía por salida.
• Hasta 729 reglas.
• Hasta tres antecedentes y 1 consecuente por regla.
• 11 caracteres por nombre.
• Funciones de membresía trapezoidales o triangulares.
• Funciones de membresía de salida singleton.
• Inferencia difusa tipo Sugeno
• Esquemas de control P, PI, PD y PP

4.2.2.2 Interface de Monitorización y Configuración.

Tabla 4.7 Especificaciones técnicas de la IMC.

• Sistema operativo XP
• Adquisición del estado de las variables del proceso de control difuso por el puerto de comunicaciones RS232.
• Visualización en tiempo real de cada un de las variables del proceso de control difuso.
• Configuración de los parámetros de funcionamiento de frecuencia de muestreo y Set Point de sistema de control difuso.
• Habilitación y deshabilitación de la ejecución del controlador difuso diseñado a solicitud del usuario.
• Configuración de la apariencia de los visualizadores

4.3 Pruebas y resultados de la operación del prototipo desarrollado.

El prototipo de Implementación de un sistema inteligente difuso para el control de procesos basados en CC y CA, utilizando un PIC, instalado para pruebas en el Laboratorio de Sistemas Inteligentes del Centro de Ciencias Aplicadas y Desarrollo Tecnológico de la UNAM, fue probado para controlar la temperatura interior de un horno eléctrico resistivo, prototipo que también fue implementado en el mismo laboratorio.

Primero se planteó el problema de control que se quería resolver, en base a eso se seleccionaron los esquemas de control que se seguirían, luego se llevaron acabo en el ESD cada uno de los pasos involucrados en el proceso de diseño del controlador difuso y finalmente se realizaron las pruebas pertinentes y se analizaron los resultados.

4.3.1 Planteamiento del problema de la aplicación de prueba de control de temperatura de un horno eléctrico resistivo.

El planteamiento del problema de la aplicación de control de temperatura fue:

Diseñar un controlador difuso que mantenga a una temperatura constante predefinida el interior de un horno eléctrico resistivo en un rango de 0 a 50 °C con un error en estado estacionario dentro del intervalo +/- 1 °C, tomando en cuenta que el Set Point o temperatura deseada debe estar arriba de la temperatura ambiente.

EL diagrama del sistema de control propuesto es mostrado en la Figura 4.9

De acuerdo al proceso y esquema de implementación de cualquier sistema de control, lo primero que se tiene que hacer es determinar la entradas y salidas de control.

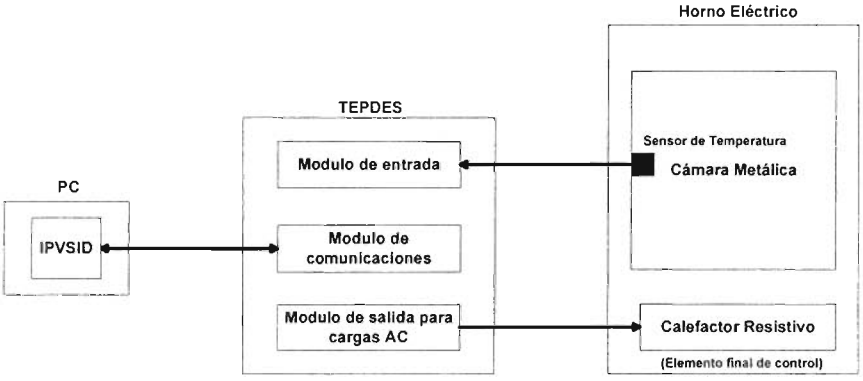


Figura 4.9. Diagrama del sistema de control de temperatura del horno eléctrico resistivo utilizando el SID.

Tomando en cuenta el tipo de controladores que se pueden diseñar con el SID (ver apartado 3.3.7.2) y por motivos de comparación y sencillez se opto por hacer uso del control proporcional (P) y un control proporcional prealimentado (PP)

Para un controlador P en un sistema de control de lazo cerrado de seguimiento de Set Point típico la entrada del controlador difuso es la señal de error $e(t)$ y la salida $u(t)$. Para la aplicación particular de control de temperatura la señal de error se definió como $e(t) = Temperatura\_Deseada - Temperatura\_Actual$ y la salida de control $u(t)$ se definió como el porcentaje de potencia entregado al calefactor del horno

Figura 4.10 Controlador difuso como Controlador Proporcional.

Para un controlador PP se tiene una entrada adicional al controlador difuso llamada referencia y que representa el set point actual del sistema.

Figura 4.11 Controlador difuso como Controlador Proporcional Prealimentado.

Cuando de antemano se define el tipo de controlador que se va utilizar las entradas y salida de control quedan definidas.

Nótese que la salida de control $u(t)$ también puede ser definida como el ángulo de disparo del triac en el rango de 0° a 180° o mas adecuadamente ponerse en función de la potencia total (en Watts), pero por motivos de sencillez se definió como el porcentaje de potencia entregado, debido a que resultó igual de significativo puesto que precisamente es difuso.

Debido a que el rango de la temperatura del horno se fijo de 0 a 50 °C, el rango del *error de temperaturas* se definió de -25 a 25 ° C e igualmente el rango para la referencia de temperatura. Como la salida del controlador se definió como el *porcentaje de potencia* entregado al calefactor su rango se definido del 0% al 100 % de potencia.

Antes de pasar a la explicación del proceso de diseño del controlador difuso para el control de temperatura del horno eléctrico en el SID se mencionaran los detalles acerca del la instrumentación de las entradas y salidas para probar el sistema.

De acuerdo el esquema de la Figura 4.9 se debe contar con un sensor de temperatura, para tal motivo se selecciono el sensor de temperatura comercial LM35A cuyas características más destacables son:

Tabla 4.8 Características del sensor LM35A.

• Calibrado directamente en °C.
• Factor de escala lineal de +10m V / °C.
• Rango de -55 °C a +150 °C
• Exactitud de 0.5°C (a +25 °C).
• Adecuado para aplicaciones remotas.

• Bajo costo debido al recorte de bajo nivel de agua.
• Opera de 4 V a 30 V.
• Menos de $60\mu A$ de corriente de drenaje.
• Bajo auto calentamiento, 0.08 °C en aire sin gas.
• No linealidad de solo $\pm 1/4°C$.
• Baja impedancia de salida, 0.1 W por un 1mA de carga.

Las cuales resultaron adecuadas para nuestra aplicación. Debido a que nuestra aplicación de prueba se definió en un rango de 0° a 50 °C la configuración utilizada del sensor fue la mostrada en la Figura 4.12, de tal manera que se proporcionara un voltaje de 0mV para 0°C y de 500mV para 50°C.

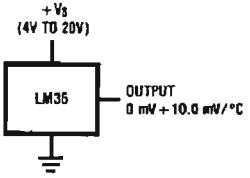


Figura 4.12. Sensor de temperatura centígrada básico (de 0°C a 150 °C).

El sensor se coloco en la cámara metálica de tal forma que registrara la temperatura global del horno eléctrico y se conecto a la entrada analógica 0 (AN0, ver Figura 4.7) de la TEPDES. Esto se hizo a través de un cable unido a las patitas del LM35A en un extremo y con un conector MOLEX hembra de tres pines unido al otro extremo, para conectarse con la TEPDES (ver apartado 3.2.4).

Una vez conectado el sensor en la TEPDES se ajusto la ganancia del amplificador de instrumentación de la entrada analógica 0 del modulo de entrada (ver apartado 3.5), a través de RG, de tal manera que se ajustara la salida del sensor del rango de 0mV a 500mV al rango de 0V a 5V (referencias del CAD); necesitándose una ganancia de:

$$G = \frac{5-0}{500\times 10^{-3}} = 10$$

Y de acuerdo a la ecuación (3.2) se calculo una

$$R_G = \frac{49.4k\Omega}{10-1} = 5.489K\Omega$$

Hecho lo anterior se tuvo listo el lazo de retroalimentación del sistema de control de temperatura y solo falta enlazar el elemento final de control o actuador para cerrar el lazo de control. De acuerdo a la Figura 4.9, Esto ultimo se lleva acabo

conectando le resistencia eléctrica del horno (elemento final de control) a la salidas AC1 de la TEPDES (ver Figura 4.7 y el apartado 3.2.6), esto se hizo, a través de una cable de 15A y 300V conectado del horno a la TEPDES con una clavija bipolar macho para poder hacer el enchufe adecuado.

La potencia máxima que se entrega al horno esta dada por $P_{MAX} = \frac{V^2}{R}$, si la resistencia eléctrica del horno tiene un valor de 12.8Ω y $V_{RMS} = 127V$ se tiene una potencia máxima de $P_{MAX} = \frac{(127V)^2}{12.8\Omega} = 1260.0781W$ y una corriente máxima de $I = \frac{P}{V} = \frac{1260.0781W}{127V} = 9.9219A$

Entonces el porcentaje de potencia que se entrega a la carga se lleva acabo tomando como referencia la potencia máxima de 1260.0781 Watts para un 100 %.

Por lo tanto los requerimiento de voltaje y corriente que la aplicación requiere son satisfechos por la TEPDES y por los elementos utilizados para la instrumentación del sistema de control de temperatura. En la Figura 4.13 se muestra una fotografía de la instalación de todos los elementos del sistema de control de temperatura del horno eléctrico utilizando el SID.

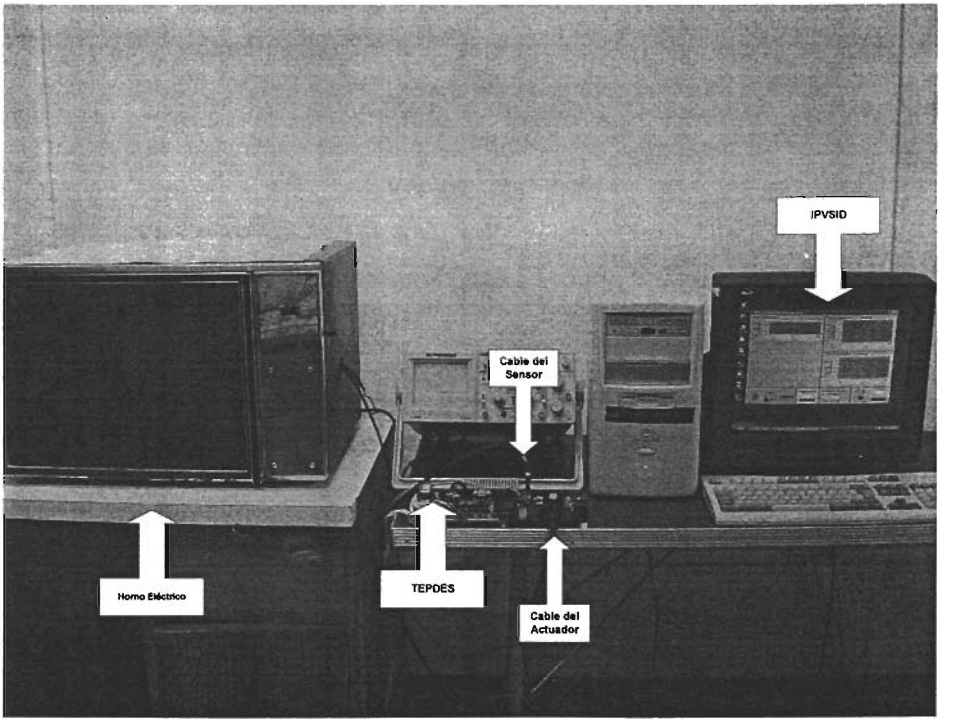


Figura 4.13 Instalación del sistema de control de Temperatura utilizando el SID.

Con esto se está en condiciones de pasar a la descripción del proceso de diseño del controlador difuso para el control de temperatura del horno eléctrico en el ESD.

EL primer paso de diseño es llevado a cabo en el Editor de las entradas y salidas del controlador difuso en una corta cantidad de tiempo.

A continuación se describe cada uno de los pasos que se deben seguir para llevar acabo esta tarea para el controlador difuso como PP (Proporcional Prealimentado), y para el P (Proporcional) solo se muestra el resultado final.

4.3.2 Edición de las entradas y salidas del controlador para la aplicación de control de temperatura.

Una vez puesta en ejecución la IPVSID la primera ventana que se muestra es la del Editor de las Entradas y de la Salida del controlador difuso (ver Figura 4.2 para referencia)

El editor de E/S abre con un diseño sin titulo, con una entrada, llamada **Entrada1**, y la salida llamada **Salida1.** Para nuestra aplicación con el controlador difuso como PP, se construye un sistema de dos entradas y una salida, si se va al menú **Edición** y se selecciona **Añadir entrada.** Una segunda caja amarilla nombrada **Entrada2** aparecerá. Las dos entradas para nuestra aplicación son *el error de temperatura* y *la referencia de Temperatura* y *la salida es el porcentaje de potencia*. Por lo tanto se tiene que cambiar el nombre de las variables, mas como el editor solo acepta nombres de 11 caracteres sus nombres tienen que ser abreviados, quedando como **ErrorT**, **CErrorT**, y **CPPotencia**, pero pueden ser otros que sean igualmente representativos. Puesto que por defecto las entradas al controlador se definen como entradas directas, la primera debe ser cambiada a tipo Error y la segunda a Tipo Referencia. Por defecto la salida es de tipo Estándar y en el caso del controlador PP se utiliza este tipo de salida.

Existen dos campos de edición muy importantes dentro del Editor de E/S el primero hace referencia al nombre de la señal de proceso asociada a la entrada del controlador difuso, o en otras palabras la variable física medible asociada a la entrada del controlador difuso. Esta Conceptualización es particularmente importante para la entrada tipo señal de error y cambio de error ya que para generar estas señales se debe conocer el estado de la salida actual o variable controlada del proceso, pues de otro modo no podrían ser generadas, en este caso la señal de proceso asociada a la entrada del controlador difuso tipo Error es la variable de temperatura. Lo mismo sucede con una entrada tipo cambio de Error (CError), no así con la entrada directa ya que esta es propiamente una variable de estado actual del proceso controlado, es decir ella misma seria su señal de proceso asociada, lo mismo sucede para la entrada tipo Referencia (ReferenciaP).

El segundo campo de edición, es el utilizado para definir el rango de la señal de proceso asociada a la entrada del controlador difuso, definiendo este automáticamente queda definido el rango de la entrada del controlador difuso asociada. Por ejemplo para nuestra aplicación particular, si el rango de la variable controlada o temperatura se define en el intervalo de 0 a 50 °C,

automáticamente la entrada del controlador difuso *Error de temperatura* se fija en el rango de -25 a 25 °C y lo mismo sucede con la entrada *cambio del error de temperatura*. Para una entrada *Directa* o de *Referencia* no es muy relevante ya que el mismo rango definido en la señal de proceso asociada es el rango de la entrada al controlador difuso, pues son la misma variable.

Finalmente se define el tipo de actuador de salida, que para el caso de una salida tipo de *Estándar* se puede escoger entre uno de tipo CA y otro de tipo CC, para nuestra aplicación se selecciono de tipo CA (calefactor resistivo en CA).

Todo el proceso anterior se puede logra haciendo los siguientes pasos (ver Figura 4.2 para referencia):

1. Hacer un clic sobre la caja amarrilla etiquetada Entrada1 (la caja será remarcada en rojo).
2. En el campo de edición derecho, Nombre, cambiar Entrada1 por **"ErrorT"** y presionar entrar.
3. En el menú desplegable Tipo de señal, cambiar la selección "**Directa**" por la selección de "**Error**".
4. En el campo de edición Izquierdo Nombre de la señal de proceso asociada, cambiar "**Estado n**" por "**Temperatura**" y presionar entrar.
5. En el campo de edición Izquierdo Rango, se cambia el rango de [0, 1023] a el rango de [0, 50].
6. Hacer un clic sobre la caja amarilla etiquetada Entrada2 (la caja será remarcada en rojo).
7. En el campo de edición derecho Nombre, cambiar Entrada2 por "**Referencia**" y presionar entrar.
8. En el menú desplegable Tipo de señal, cambiar la selección "**Directa**" por la selección de "**Referencia**".
9. En el campo de edición Izquierdo Nombre de la señal de proceso asociada, cambiar "**Estado n**" por "**Temperatura**" y presionar entrar.
10. En el campo de edición Izquierdo Rango, se cambia el rango de [0, 1023] a el rango de [0, 50].
11. Hacer un clic sobre la caja azul etiquetada Salida1 (la caja será remarcada en rojo).
12. En el campo de edición derecho Nombre, cambiar Salida1 por **PPotencia** y presionar entrar.
13. En el menú desplegable Tipo de señal, la selección "**Estándar**" se mantiene.
14. En el menú desplegable Carga, seleccionar **AC1**.
15. Del menú **Archivo** seleccionar **Guardar.**

16. En la caja de dialogo Guardar como, teclear el nombre del diseño, seleccionar la ruta de almacenamiento y hacer un clic sobre el botón **aceptar** (el nombre del diseño fue "***Temperatura*RP**").

17. Con esto se vera el diagrama actualizado el cual reflejara los nuevos parámetros de las entradas y salida del controlador difuso. La ventana resultante se vera como la mostrada en la Figura 4.14.

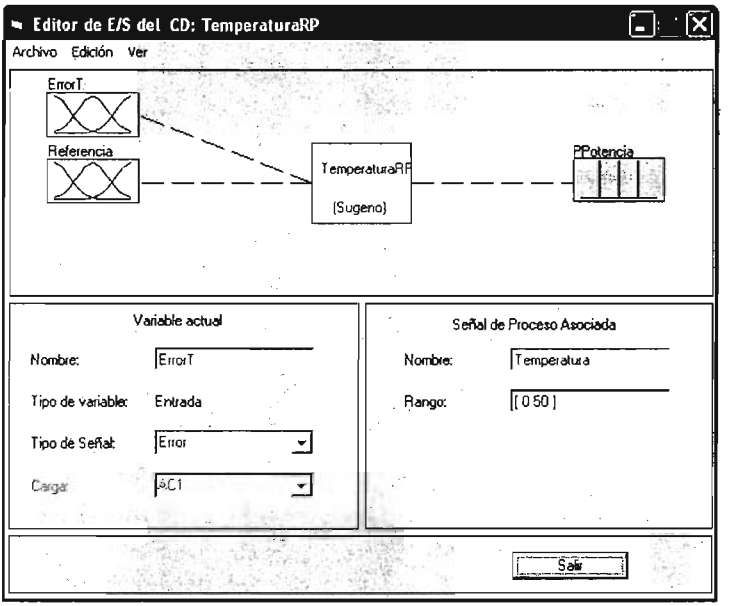


Figura 4.14 Edición de Entradas y Salida de controlador como PP

En la Figura 4.15 se muestra la ventana resultante para el controlador difuso como proporcional.

El segundo paso en el diseño de un controlador difuso dado consiste en asociar términos lingüísticos a cada uno de los parámetros físicos involucrados. El objetivo es particionar el dominio de cada variable de entrada y salida, y dar un nombre a cada porción del dominio de esa variable.

En este caso, los rangos son generalmente subjetivamente seleccionados. La selección generalmente depende del diseñador y/o experto. En cualquier caso, tiene que reflejar el conocimiento o experiencia recolectada sobre el sistema.

Las Tablas 4.9, 4.10, 4.11, 4.12 y 4.13 muestran las particiones (rango y parámetros de las funciones de membresía), para *el error de temperatura*, *la referencia de temperatura* y *el porcentaje de potencia.*

Debido a que la inferencia utilizada por el SID es tipo sugeno para la salida solo se permiten definir funciones singleton constantes, entonces mas que particionar la variable de salida, se definen los valores puntuales o crisp que la salida puede tener y que también pueden tener un termino lingüístico asociado.

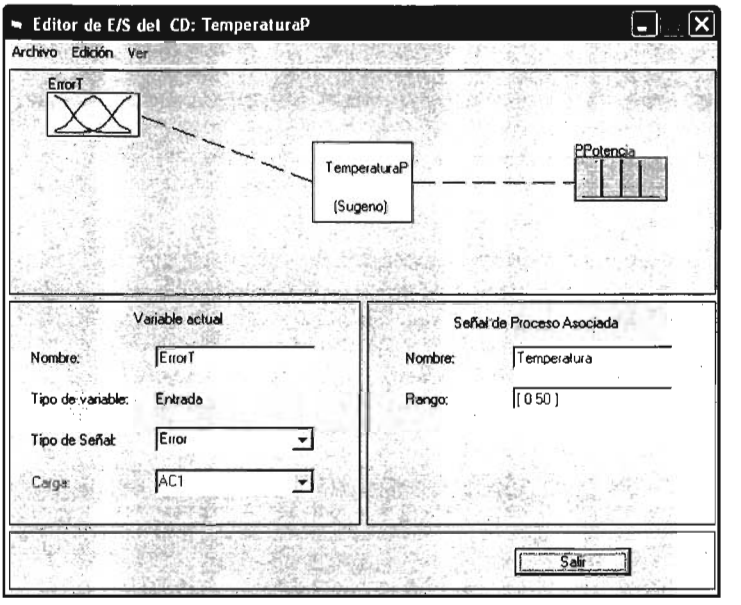


Figura 4.15 Edición de Entradas y Salida del controlador difuso como P.

Tabla 4.9 Rangos para la variable de entrada "error de temperatura" para el controlador Proporcional.

Conjunto Difuso	Termino lingüístico	Rango (°C)
1	Negativo Grande	[-25 -25 -1.5 --1]
2	Negativo Medio	[-1.5 -1 -1 -0.5]
3	Negativo Pequeño	[-1 -0.5 -0.5 0]
4	Cero	[-0.5 0 0 0.5]
5	Positivo Pequeño	[0 0.5 0 0.5 1]
6	Positivo Medio	[0.5 1 1 1.5]
7	Positivo Grande	[1 1.5 25]

Tabla 4.10 Rangos para la variable de entrada "error de temperatura" para el controlador Proporcional Prealimentado.

Conjunto Difuso	Termino lingüístico	Rango (°C)
1	Negativo	[-25 -25 -0.7 -0.2]

2	Cero	[-0.7 -0.2 -0.1 0.7]
3	Positivo	[0.1 0.7 25 25]

Tabla 4.11 Rangos para la variable de entrada de "Referencia de Temperatura."

Conjunto Difuso	Termino lingüístico	Rango(°C)
1	Muy Baja (MB)	[0 0 5 15]
2	Baja (B)	[5 15 15 25]
3	Media (M)	[15 25 25 35]
4	Alta (A)	[25 35 35 45]
5	Muy Alta (MA)	[35 45 50 50]

Tabla 4.12 Rangos para la variable de salida "porcentaje de potencia" para el controlador Proporcional Prealimentado.

Conjunto Difuso	Termino lingüístico	Rango (%)
1	Nada	0
2	Muy bajo (MB)	1
3	Baja (B)	2
4	Media (M)	8
5	Alta (A)	16.5
6	Muy alta (MA)	22.5
7	Toda	100

Tabla 4.13 Rangos para la variable de salida "porcentaje de potencia" para el controlador Proporcional.

Conjunto Difuso	Termino lingüistico	Rango (%)
1	Nada	0
2	Muy bajo (MB)	2
3	Baja (B)	5
4	Media (M)	15

5	Alta (A)	20
6	Muy alta (MA)	50
7	Toda	100

Cada uno de los rangos tiene una función de membresía asociada con el. Ella determinara los grados de membresía. La asociación de una función de membresía a un rango lleva a la construcción del conjunto difuso.

Existe una variedad de funciones de membresía que pueden ser usadas dependiendo del proceso y el parámetro particular utilizado. El editor de Funciones de membresía solo permite editar funciones de membresía trapezoidales, sin embargo con una selección adecuada de los valores de sus parámetros se poden definir funciones de membresía triangulares, tipo Z y tipo S (ver apartado 3.3.7.4.1.2).

Una vez definidas las particiones de cada una de las variables de entrada y de la salida, la edición de las funciones de membresía asociada a cada una de ellas, se lleva acabo en el Editor de Funciones de Membresía del ESD.

A continuación se describe cada uno de los pasos que se deben seguir para llevar acabo esta tarea para el controlador difuso como PP, y para el controlador difuso como P solo se muestra el resultado final.

4.3.3 Edición de las funciones de membresía de entrada y salida del controlador difuso para la aplicación de control de temperatura.

Lo primero que se tiene que hacer es abrir el Editor de Funciones de Membresía (ver Figura 4.3 para referencia). Este se puede abrir de dos maneras desde el Editor de E/S:

1. Desplegando los elementos del menú **Ver** y seleccionar **Editor de FM.**
2. Haciendo un doble clic sobre un icono de entrada (cajas amarillas) o de salida (caja azul)

Cuando se abre el Editor de Funciones de Membresía para trabajar sobre un diseño dado, no hay todavía función de membresía alguna asociada con las variables que justamente se han definido en el Editor de E/S.

En el lado superior izquierdo del área grafica en el Editor de Funciones de Membresía esta la paleta de variables que permite definir las funciones de membresía para una variable dada. Para fijar las funciones de membresía asociadas a una variable de entrada o de salida del controlador difuso, se selecciona una variable en esta región haciendo clic sobre ella.

En seguida se selecciona el menú desplegable **Edición,** y se selecciona **Añadir FM.** Una nueva ventana aparecerá, la cual permitirá seleccionar el número de funciones de membresía asociadas con la variable seleccionada.

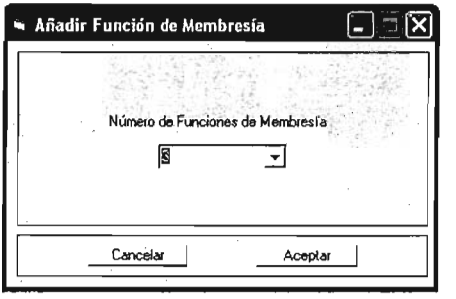
En el lado derecho inferior del EFM están los controles que permiten seleccionar una de las funciones de membresía agregadas a una variable dada y cambiar su nombre y sus parámetros.

Las funciones de membresía de la variable actual son desplegadas en la grafica principal. Para editar las función de membresía de la variable actual, primero debe ser seleccionada de un menú desplegable que lista los números de la funciones de membresía de la variable actual. Su nombre y sus parámetros pueden ser cambiados en los campos de edición respectivos.

En el lado inferior izquierdo debajo de la paleta de variables se muestra información acerca del nombre, tipo de variable y tipo de entrada, tipo de salida y rango de la variable actual.

El proceso para especificar las funciones de membresía de las dos entradas del controlador difuso como PP es como sigue.

1. Seleccionar la variable de entrada, **ErrorT**, haciendo un clic sobre ella.
2. Seleccionar **Añadir FM** del menú **Edición**. La ventana de abajo se abrirá.



3. Usar el tabulador desplegable para seleccionar 3, para el número de funciones de membresía. Esto agregara 3 funciones de membresía trapezoidales en forma triangular a la variable de entrada **ErrorT**.
4. Seleccionar la primera de las funciones de membresía en la lista desplegable FM. Cambie su nombre y parámetros correspondientes de acuerdo a la Tabla 4.10.
5. Siguiendo el mismo orden de las funciones de membresía en el editor y el de la Tabla 4.10, y como en el paso 4, cambie el nombre y parámetros de cada una de las funciones de membresía de la variable **ErrorT.**
6. Seleccionar la variable de entrada, **Referencia**, haciendo un clic sobre ella.
7. Seleccionar **Añadir FM** del menú **Edición**. Y agregue 5 funciones de membresía.
8. Siguiendo el mismo orden de las funciones de membresía en el editor y el de la Tabla 4.11, y como en el paso 4, cambie el nombre y parámetros de cada una de las funciones de membresía de la variable **Referencia.**

En seguida es necesario crear las funciones de membresía para la variable de salida. Como el tipo de inferencia difuso utilizado por SID es tipo sugeno solo se permiten usar funciones Singleton constantes para definir los conjuntos difuso de salida.

Para crear las funciones crisp de la variable de salida, se da un clic sobre el icono de la variable de salida, **PPotencia**, localizada en la paleta de variables. Se selecciona **Añadir FC** del menú Edición, y se agregan 7 funciones crisp. Después se fijan los nombres y parámetros de cada función crisp siguiendo el orden de la Tabla 4.12.

EL sistema resultante debe ser como el mostrado en la Figura 4.16

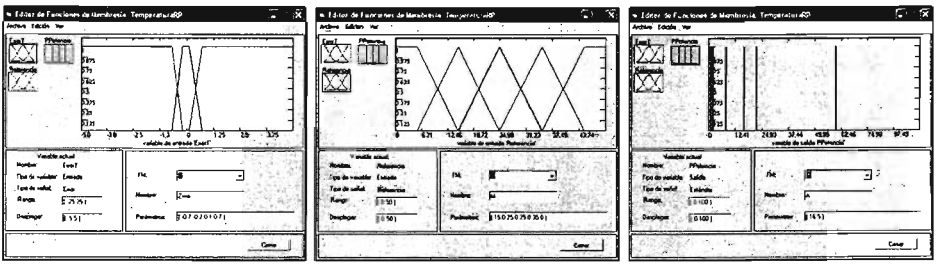


Figura 4.16 Funciones de membresía de: A) Entrada de "error de temperatura", B) Entrada "Referencia de Temperatura" y B) salida "Porcentaje de Potencia"; para el controlador difuso como proporcional prealimentado.

Es importante destacar que la selección de la forma de las funciones de membresía se hizo de manera subjetiva e intuitivamente usando el sentido común, después de haber hecho varias pruebas de tal manera que se cumplieran los requerimientos de la aplicación de control de temperatura.

En las Figuras 4.17 se muestran el sistema resultante para el controlador difuso como Proporcional.

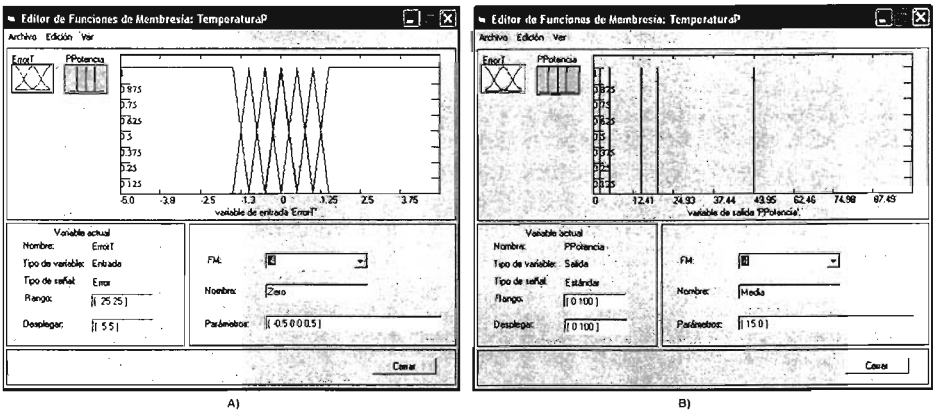


Figura 4.17 Funciones de membresía de: A) Entrada de "error de temperatura" y B) salida "Porcentaje de Potencia"; para el controlador difuso como proporcional.

El siguiente y último paso en el diseño del controlador difuso es definir la base de reglas de control difuso, las cuales representan la estrategia de control y sustituyen los algoritmos y ecuaciones utilizadas en el diseño de controladores convencionales.

Estas reglas pueden ser definidas de manera intuitiva y utilizando el sentido común por parte del diseñador o consultando a un experto en el área de dominio, para nuestra aplicación particular de control de temperatura la definición de dicho conjunto de reglas se hizo utilizando como referencia la grafica de Figura 4.18 y aplicando el sentido común, con lo que se llego a la Tabla de control 4.14, para el controlador difuso proporcional prealimentado de dos entradas.

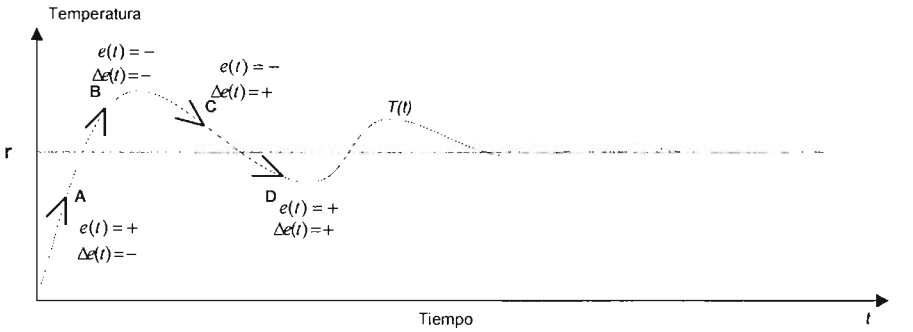


Figura 4.18 Ejemplo de seguimiento de set point de temperatura.

En la Figura 4.18, $e(t) = r - T(t)$.

Tabla 4.14 Reglas difusas del controlador difuso como PP para el control del sistema de seguimiento de set point de Temperatura.

Referencia / e	MB	B	M	A	MA
N	Nada	Nada	Nada	Nada	Nada
Zero	MB	B	M	A	MA
P	MA	MA	MA	Toda	Toda

En esta Tabla cada columna representa una etiqueta lingüística de la entrada *referencia de temperatura* y cada renglón representa una etiqueta lingüística para la entrada de *error de temperatura*. La intersección de un renglón y una columna representa la salida del controlador difuso, que en el caso del controlador difuso como PP es *el porcentaje de la potencia de salida (PPontencia)*, cuando las entradas toman los respectivos valores, es decir constituye una regla de control de la forma:

Si *el error de temperatura es cero* y *referencia es Media* Entonces *el porcentaje de potencia es Medio.*

Regla que expresa que si el error de temperatura es cero, es decir, la temperatura actual es igual a la referencia, y si la referencia de temperatura es Media, en otras palabras, la temperatura tiene el valor de la referencia, la cual es considerada Media, entonces el porcentaje de potencia es Medio, de tal manera que se mantenga la temperatura en el valor deseado, es decir el porcentaje de potencia aplicado actualmente es el adecuado para mantener la temperatura igual a la referencia y por lo tanto no necesita ser incrementado o disminuido.

Todas las demás reglas pueden ser similarmente interpretadas.

La Tabla 4.15 muestra la lista de reglas resultante para el controlador difuso proporcional.

Tabla 4.15 Reglas difusas del controlador difuso P para el control del sistema de seguimiento de set point de Temperatura.

e	% de potencia
NG	Nada
NM	Muy Baja
NP	Baja
Zero	Media
PP	Alta
PM	Muy Alta
PG	Toda

Una vez formuladas las reglas de control difuso, la edición de las reglas en el ESD, se lleva acabo en el Editor de Reglas difusa en una corta cantidad de tiempo.

A continuación se describe cada uno de los pasos que se deben seguir para llevar acabo esta tarea para el controlador difuso como PP, y para el controlador P solo se muestra el resultado final.

4.3.4 Edición de las reglas difusa de control del controlador difuso para la aplicación de control de temperatura.

Para comenzar con la edición del conjunto de reglas de control primero que nada se debe abrir el Editor de reglas difusas (para referencia ver la Figura 4.4). Este se puede abrir de dos maneras:

1. Desplegando los elementos del menú **Ver** en el **Editor de E/S** o en el **EFM** y seleccionar **Editor de Reglas.**
2. Haciendo un doble clic sobre el icono del controlador difuso (caja blanca) del **Editor de E/S**.

La construcción de reglas en el Editor de Reglas es bastante evidente en si. Basado en la descripción de las entradas y salidas, permite construir las declaraciones de las reglas automáticamente, seleccionado las descripciones lingüísticas de cada entrada y de la salida en los menús correspondientes.

La primera regla de la Tabla 4.14 es construida como sigue:

1. Seleccionando **N** en la variable **ErrorT**.
2. Seleccionando **MB** en la variable **Referencia**.
3. Seleccionando **MB** en la variable **PPontencia**.
4. Haciendo un clic sobre el botón de comando **Añadir** o seleccionado **Añadir Regla** en el menú **Edición** de Editor de reglas.

La regla resultante es:

1. If ErrorT is N and Referencia is MB then PPotencia is MB

Con esto la regla se agregara automáticamente en la lista de reglas. El resto de reglas de la Tabla 4.14 se agregan de manera similar simplemente seleccionando la etiqueta lingüística de cada entrada y la salida y Haciendo un clic sobre el botón de comando **Añadir** o seleccionado **Añadir Regla** en el menú **Edición** del Editor de reglas.

Para cambiar una regla, primero se hace un clic sobre la regla que va a ser cambiada. En seguida se hacen los cambios deseados a esa regla, y luego se da un clic en el botón de comando **Cambiar** o en **Cambiar Regla** del menú **Edición.**

Para eliminar una regla simplemente se selecciona la regla que se desea eliminar y se hace un clic en el botón de comando **Eliminar** o en **Eliminar Regla** del menú **Edición**.

Las 10 primeras reglas del controlador difuso como PP, aparecen como se muestra en la Figura 4.19.

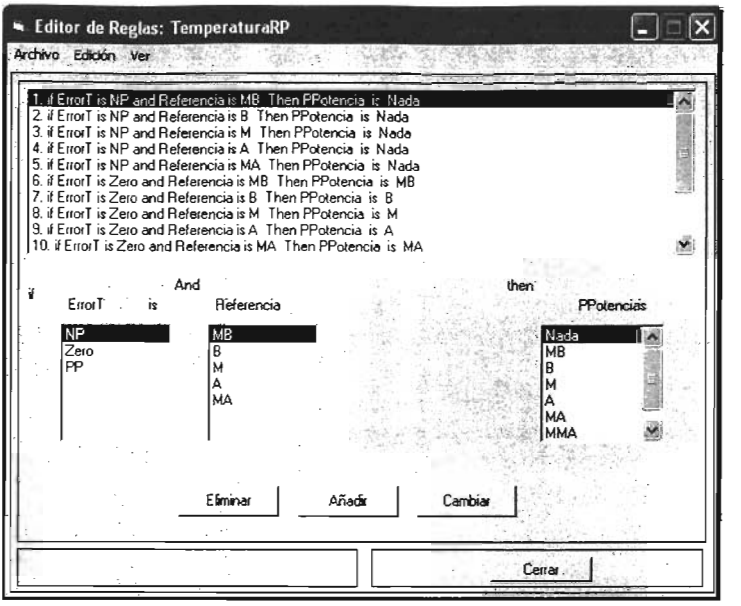


Figura 4.19 Primeras 10 reglas de controlador difuso PP.

Las reglas del controlador difuso proporcional son mostradas en la Figura 4.20.

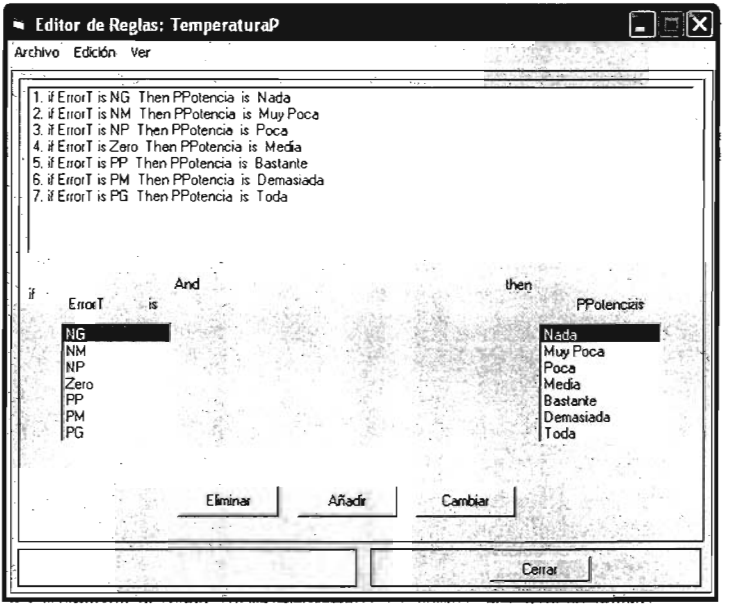


Figura 4.20 Conjunto de reglas del controlador difuso Proporcional.

En este punto, el sistema difuso ha sido completamente definido, eso es, las variables, las funciones de membresía, y las reglas necesarias para controlar la temperatura están definidas.

El siguiente paso es generar y descargar el código asociado a los controladores diseñados para que sean implementados y puestos en ejecución para verificar su funcionamiento y desempeño; y posteriormente si es necesario hacer las correcciones adecuadas para poner a punto los controladores.

El pasó de generación y descarga del código se llevan acabo en el Generador y descargador de Código y a continuación se describen los pasos necesarios para llevar acabo esta operación.

4.3.5 Generación y descarga del código asociado al controlador difuso para la aplicación de control de temperatura.

EL Generador y Descargador de código (ver Figura 4.5 para referencia) se puede llamar desde cualquiera de las otras tres herramientas GUI del ESD, seleccionando **Generación de Código** en el menú **Ver** de cada una.

Cuando se abre por primera vez el Generador de código, todas las listas de direcciones y datos que despliegan el código asociado a la definición de las funciones de membresía, a la definición del conjunto de reglas de control difuso y el código sistema de control difuso, aparecen vacías. Para generar el código del diseño actual basta con hacer un clic en el botón de comando llamado **Generar** o seleccionando la opción **Generar Código** del menú **Edición**. Antes de hacerlo se puede opcionalmente seleccionar el tipo de memoria del microcontrolador en la que se escribirá el código asociado al conjunto de reglas difusas, que son la memoria EEPROM o la memoria FLASH, por omisión, siempre se encuentra seleccionada la primera. Debido a que la memoria EEPROM esta reducida a un máximo de 256 bytes disponibles, cuando el código que define el conjunto de reglas requiere mas de esta cantidad, automáticamente el generador de código selecciona la memoria FLASH para escribirlo, por lo tanto se puede pasar por alto esta selección, ya que la propia aplicación puede decidir cual es la adecuada (ver apartado 3.3.7.4.2).

Una vez generado el código, este se despliega en las listas correspondientes, ya sea en formato hexadecimal o decimal según se haya afirmado o no el control de selección **Hexadecimal** (que se encuentra en la esquina inferior derecha). El Siguiente paso es descargarlo a la TEPDES, para llevar acabo esta tarea se siguen los siguientes pasos:

1. Verificar que el cable de comunicación serie entre la TEPDES y la PC, se encuentre debidamente conectado.
2. En seguida hacer un clic en el botón de comando **Conectar**, para establecer la comunicación entres la TEPDES y el IPVSID. Después de esto la leyenda del botón de comando pasara a decir **Cortar.**
3. Después se hace un clic sobre el botón de comando **Programar**, que envía el comando de modo programación al microcontrolador de la TEPDES. Si el

comando ha sido correctamente recibido, en la caja de texto con la etiqueta **Modo de trabajo actual**, se desplegara el mensaje **Modo Programación**, lo que indica que el programa de sistema ha entrado al modo programación a la espera de un comando de escritura o lectura. Esto también se observara en el monitor de estado de la TEPDES donde se desplegara el mensaje "SID CORRIENDO EN MODO PROGRAMACION" (ver apartado 3.3.6.2).

4. Luego se hace un clic sobre el botón de comando **Escribir**, con lo que se procederá a descargar todo el código asociado al diseño actual, lo cual tomara una pequeña cantidad de tiempo, y que una vez concluida la descarga se indica por medio de una caja de mensaje que dice "Fin de escritura".
5. Si ya no se desea hacer otra operación en el modo programación (Por ejemplo una operación de lectura) se procede a abortar dicho modo de operación, haciendo un clic en el botón de comando **Terminar**, con lo que en la caja de texto con la etiqueta **Modo de trabajo actual**, se despliega el mensaje "Dispositivo listo" y paralelamente la caja de mensaje con la leyenda "Ahora puede enviar comando", lo que indica que el PS de microcontrolador esta a la espera de una nuevo comando de operación (ver apartado 3.3.4).

Con estos cinco pasos se consigue descargar el código asociado al diseño en curso, y en orden se procede a procesar y ejecutar dicho código. Para llevar acabo dichas acciones se hace un clic en el botón de comando **Monitorear**. Con esto se cargara la Interface de Monitorización y Configuración (IMC) desde donde se controlara la ejecución del controlador difuso para la aplicación particular para la que fue diseñado.

4.3.6 Ejecución del controlador difuso para la aplicación de control de Temperatura.

En la IMC se, se muestra un visor grafico que despliega en tiempo real el estado y comportamiento de cada una de las entradas controlador difuso así como el de la salida de control y la variable controlada del proceso bajo control (temperatura en esta caso).

Para proceder a poner en ejecución el sistema de control difuso se deben seguir y tomar en cuenta el siguiente conjunto de pasos:

1. Seleccionar la frecuencia de muestreo adecuada para la aplicación, por omisión, inicialmente se encuentra seleccionada la frecuencia de 200 muestras por segundo.
2. Seleccionar el Set Point inicial del Proceso bajo control, por omisión se fija un set point a la mitad del rango completo de la variable controlada del proceso bajo control.

3. Si se desea, cambiar la presentación de los visores gráficos esto es cambiar el color del grid y el área de trazado; cambiar el color del trazo de cada variable; y fijar el número de puntos que se visualizan por pantalla.
4. Proceder a ejecutar el diseño cambiando el estado del interruptor **Ejecución** de **Terminar** a **Ejecutar** por medio de un clic.

Una vez llevados acabo estos pasos, el sistema de control difuso se debe estar ejecutando, lo que es observado en los visualizadores gráficos en los cuales se desplegara el estado actual y comportamiento en el tiempo real de cada una de las variables involucradas.

De esta manera mientras el sistema de control difuso esta en ejecución, se podrán cambiar el parámetro de frecuencia de muestreo y el Set Point, y observar el impacto de la mismos en el desempeño y la estabilidad del sistema.

4.3.7 Resultados de la pruebas.

Las condiciones de las pruebas tanto para el controlador difuso proporcional como para el proporcional prealimentado se fijaron en los siguientes términos:

1. Los Set Points (referencias o temperaturas deseadas) que se fijaron para el rango de 0 °C a 50 °C fueron de 30 °C, 35 °C 40 °C y 45 °C. Cada Set Point se fija partiendo de una temperatura ambiental en el rango de 21 °C a 26 °C.
2. Se fijo una frecuencia de muestreo de 200 Hz considerando que el componente en frecuencia mas alto del sistema es el de la salida de potencia en CA, que es de 60 Hz, con lo que $200 \geq 2 \times 60$, satisfaciendo el teorema de muestreo.
3. El nivel de ruido máximo medido experimentalmente y expresado en porcentaje para el rango de 25 °C a 50 °C fue de +/- 0.68 %.
4. El tiempo de cada prueba fue de 400 segundos (mas de 6.5 minutos).

Las ilustraciones del desarrollo de las pruebas solo son mostradas para el controlador difuso proporcional (P) cuando se hace un cambio de set point de 25 °C a 35 °C, ya que el procedimiento es el mismo para los demás cambios y lo mismo es valido para el controlador difuso Proporcional Prealimentado (PP).

De acuerdo a lo explicado en el apartado 4.3.6 lo único que se hizo fue:

1. Seleccionar una frecuencia de muestreo de 200 MPS, aunque, por omisión, inicialmente esta ya se encuentra seleccionada.
2. En el control adecuado de la IMC se selecciono el Set Point inicial de temperatura, que en este caso se fijo en 35 °C.
3. La apariencia de los visualizadores se mantuvo en sus valores predeterminados (se pudo haber cambiado).
4. Finalmente para poner en ejecución el controlador difuso, en la IMC se cambio el estado del interruptor **Ejecución** de **Terminar** a **Ejecutar** por medio de un clic.

Concluyendo esto paso el controlador difuso proporcional comenzó e ejecutarse

En la Figura 4.21 es mostrada la IMC para el controlador difuso como P para el cambio de set point de 25 °C a 35 °C. En ella se observa el visualizador en tiempo real de la única entrada del *controlador difuso proporcional* que en este caso es el *error de temperatura*, el de la salida que es *el porcentaje de potencia* aplicado al elemento final de control o resistencia eléctrica y el de la variable controlada o temperatura interior del horno.

De la misma puede ver como la señal de error disminuye tendiendo a cero conforme la temperatura del horno alcanza la referencia. Del mismo modo se ve como el porcentaje de potencia aplicado a la resistencia eléctrica es máximo cuando se hace el cambio de set Point, manteniéndose así hasta la señal de error entra en la zona de transición de los funciones de membresía (la zona de transición de las funciones de membresía es el intervalo de valores de la variable lingüística donde se traslapan todas ellas) de esa señal de error (ver Tabla 4.7 y Figura 4.17) , momento en el cual el porcentaje de potencia empieza a disminuir hasta un valor mínimo y después comienza a subir lentamente (periodo transitorio de la señal de salida) a un valor estable de tal manera que la temperatura interior del horno eléctrico se mantenga en el valor constante 35 °C .

En la Figura 4.22 se muestra la IMC cuando la temperatura interior del horno alcanzo la referencia de 35 °C y se mantuvo dentro de la banda de asentamiento.

Como puede observares de la misma, la temperatura del horno eléctrico se ha estabilizado alrededor de los 35 °C, con un pequeño error en estado estacionario, y las oscilaciones prácticamente han desaparecido. Además se observa como el porcentaje de potencia también prácticamente se ha estabilizado alrededor de un valor de 17 %, ya que las oscilaciones de la misma han desaparecido y solo varía ligeramente cuando se presentan pequeñas perturbaciones.

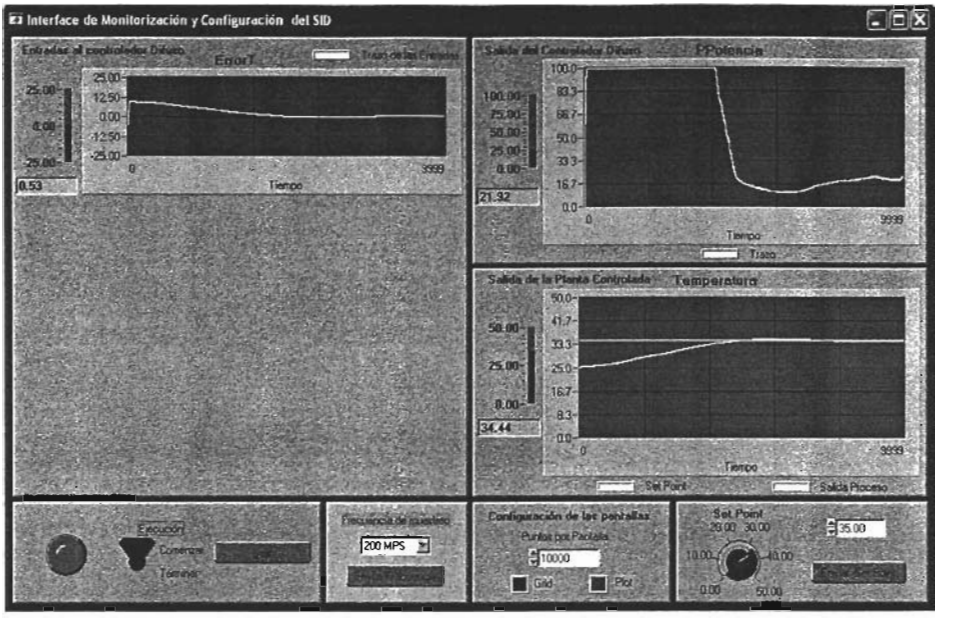


Figura 4.21 IMC para el controlador difuso proporcional cuando se hace un cambio de Set Point de 25 ºC a 35 ºC.

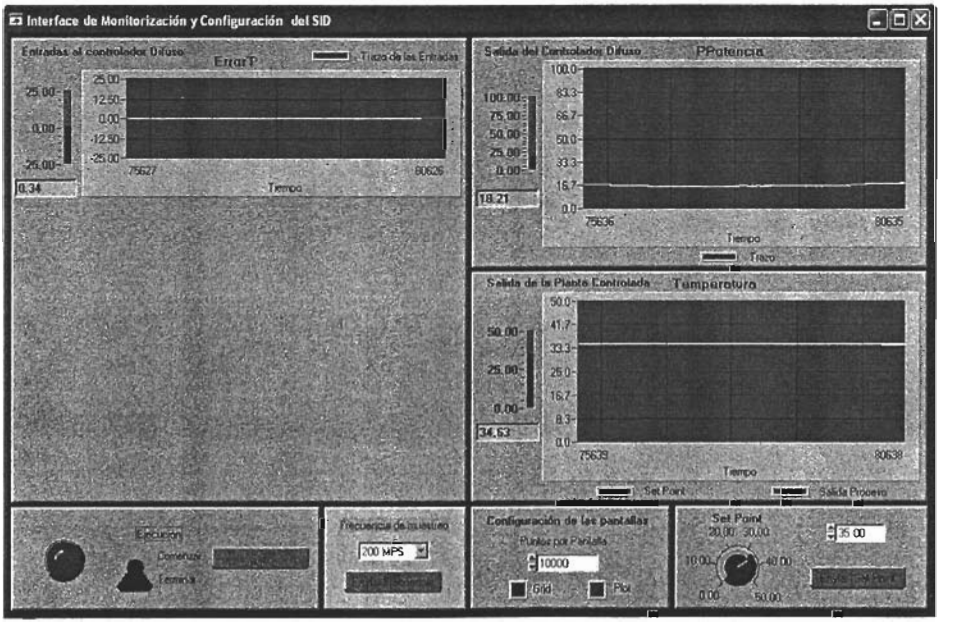


Figura 4.22 IMC, cuando le temperatura interior del horno eléctrico se mantiene estable en la temperatura de 35 ºC.

En las Figuras 4.23 se muestra la foto del horno eléctrico cuando se hace el cambio de set point de 25 °C a 35 °C para el controlador difuso como P.

En la foto se puede apreciar que la intensidad de potencia aplicada a la resistencia es muy alta ya que en ese momento se entrega el 100 % de potencia a la misma.

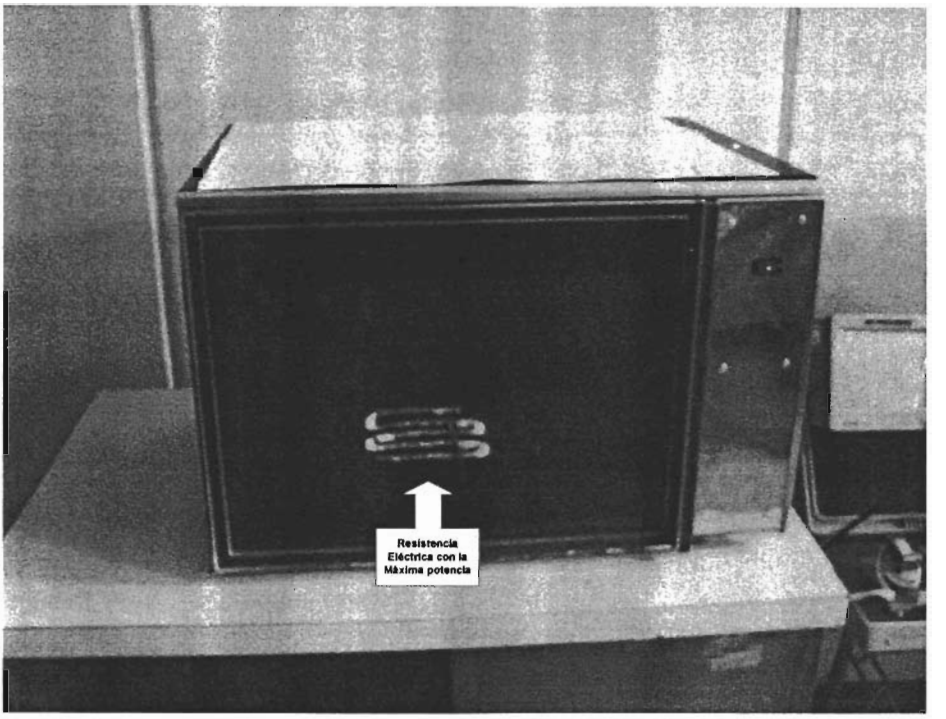


Figura 4.23 horno eléctrico cuando se hace un cambio de set point de temperatura de 25 °C a 35 °C para el controlador difuso como P.

En la Figura 4.24 se muestra la foto de la pantalla de osciloscopio en cual se despliega la señal de CA aplicada al actuador resistivo del horno eléctrico cuando el set point se hubo estabilizado en 35 °C; en ella se puede observar un ángulo de disparo del triac de salida de aproximadamente 145°, de tal manera que solo se entrega un pequeño porcentaje de la potencia total cuando la temperatura se ha estabilizado alrededor de 35° C.

Las graficas de las respuestas de los dos controladores difuso considerados para cada uno de los Set Point definidos son mostradas, para poder apreciar mejor, el desempeño del SID. Cabe mencionar que las graficas se hicieron por medio de un programa específico, también, desarrollado en LabWindows, y que toma como entrada los ficheros de texto generados por la IMC donde los datos adquiridos que representan las variables del sistema de control en ejecución son almacenados (ver apartado 3.4.2). Las graficas de las respuesta del controlador difuso como proporcional (P) para cada uno de los Set Point definidos son mostradas en las Figuras 4.25, 4.26, 4.27, 4.28.

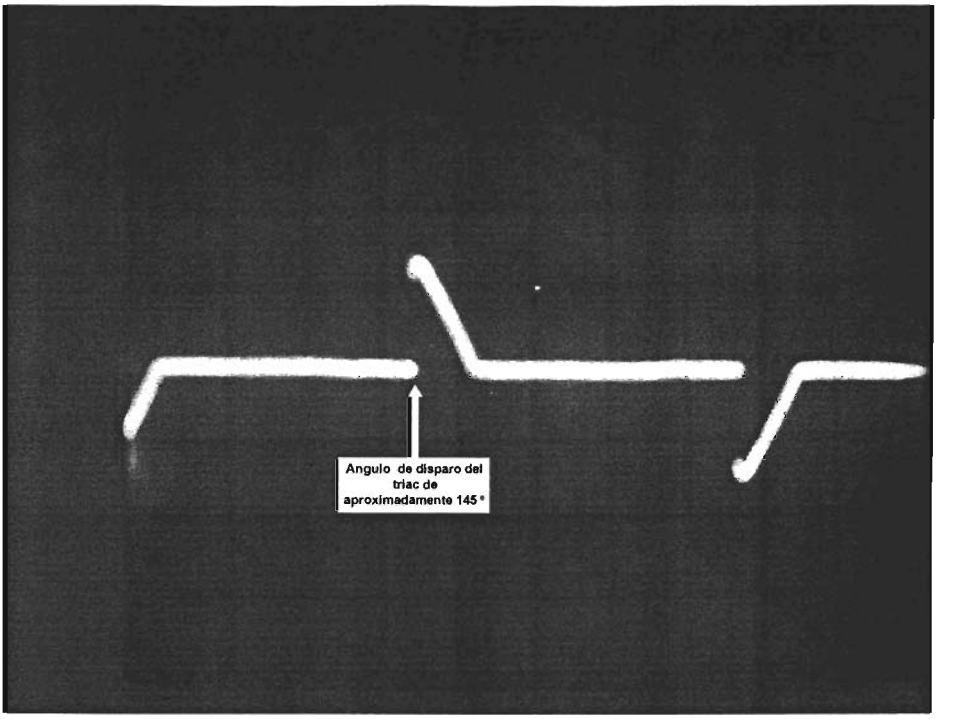


Figura 4.24 Señal se salida CA aplicada a la resistencia, cuando la temperatura se ha estabilizado en 35 °C.

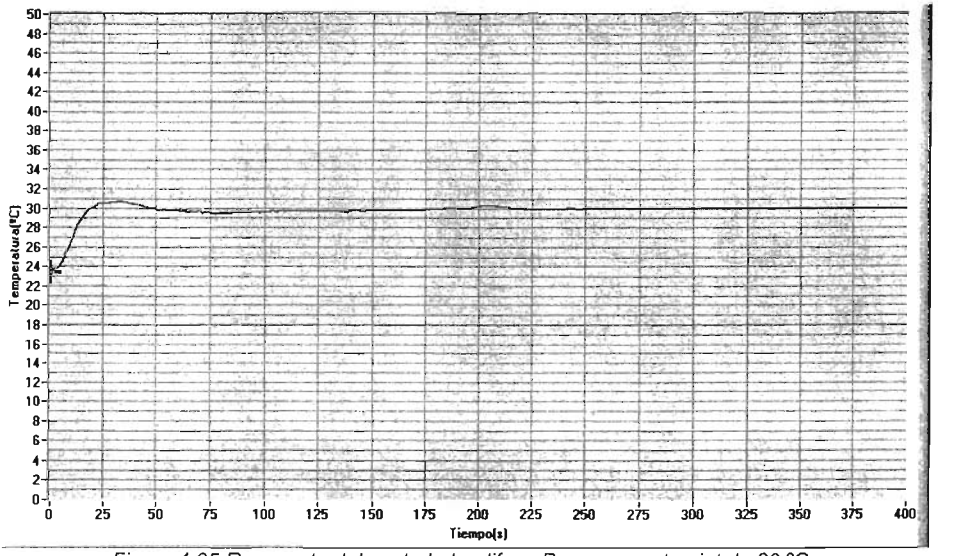


Figura 4.25 Respuesta del controlador difuso P para un set point de 30 °C.

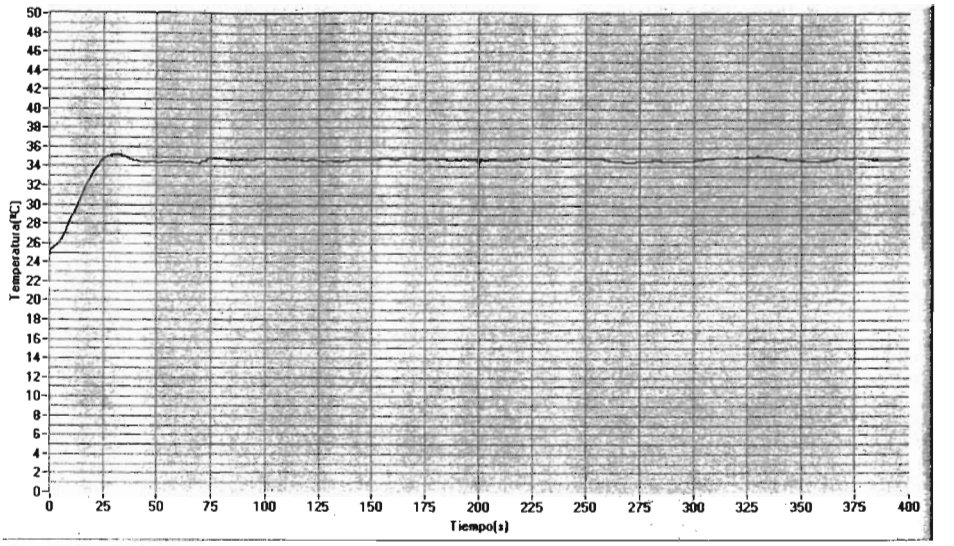


Figura 4.26 Respuesta del controlador difuso P para un set point de 35 ºC.

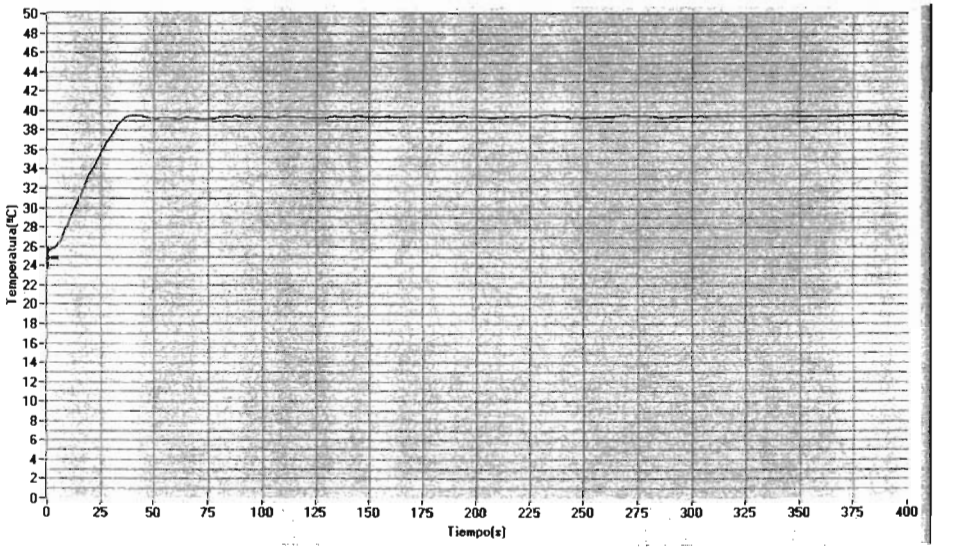


Figura 4.27 Respuesta del controlador difuso P para un set point de 40 ºC.

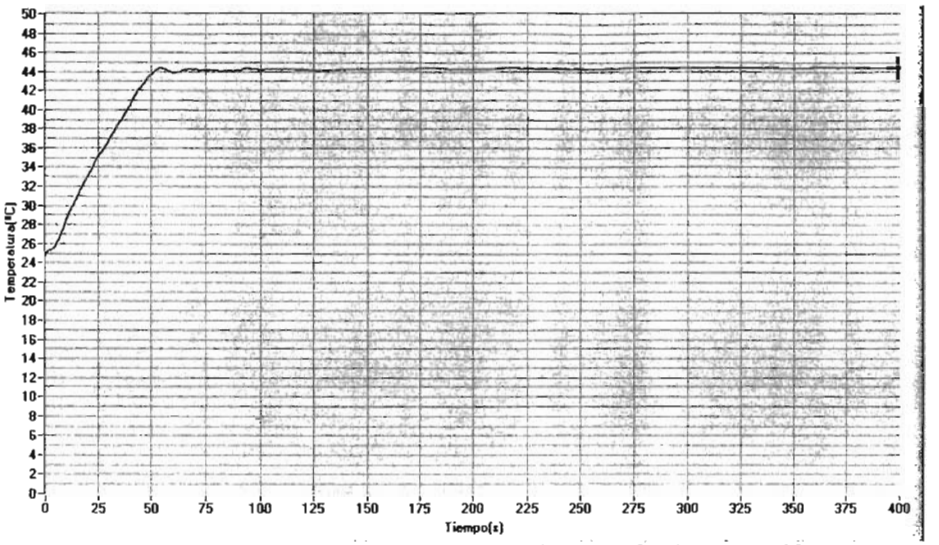


Figura 4.28 Respuesta del controlador difuso P para un Set Point de 45 °C.

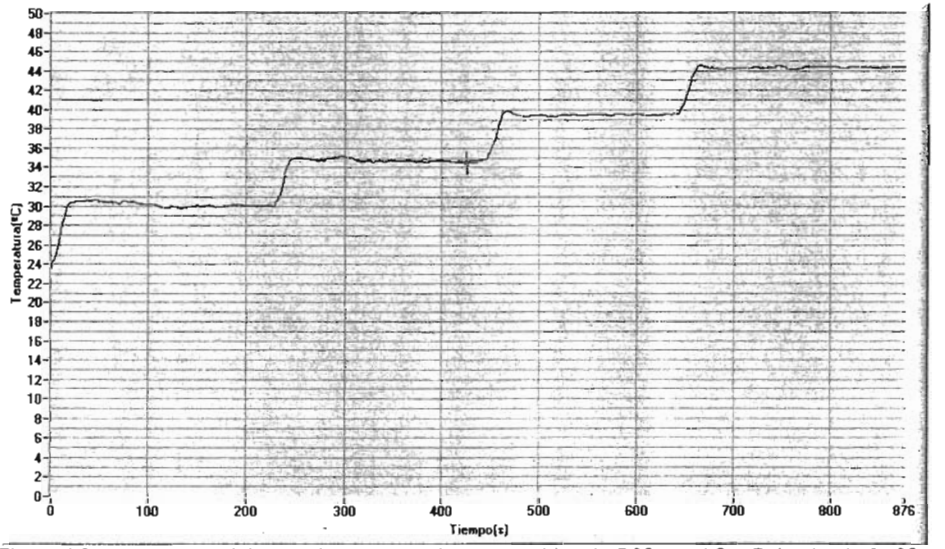


Figura 4.29 Respuesta del controlador difuso P para cambios de 5 °C en el Set Point desde 25 °C hasta 50 °C.

Las gráficas de la respuestas del controlador difuso como proporcional prealimentado (PP) para cada uno de los Set Point definidos son mostradas en las Figuras 4.30, 4.31, 4.32, 4.33.

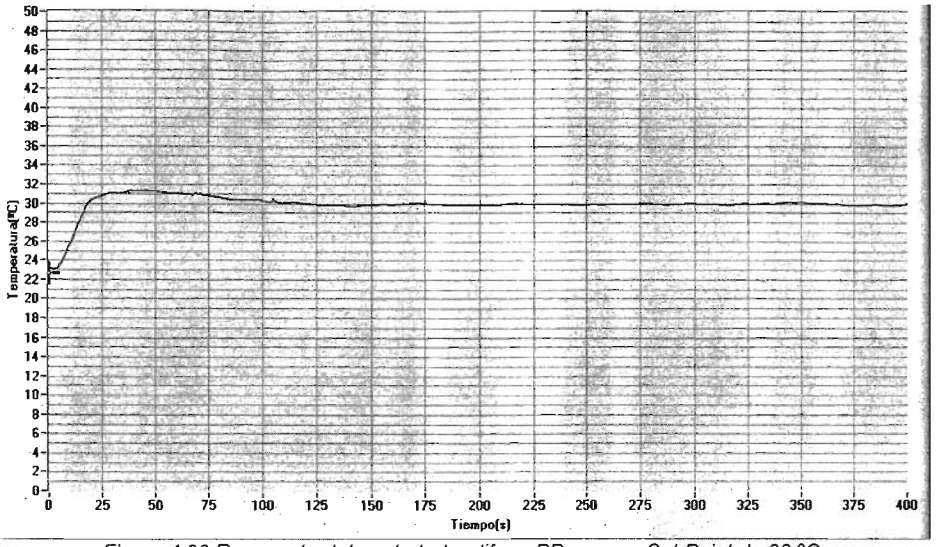


Figura 4.30 Respuesta del controlador difuso PP para un Set Point de 30 ºC.

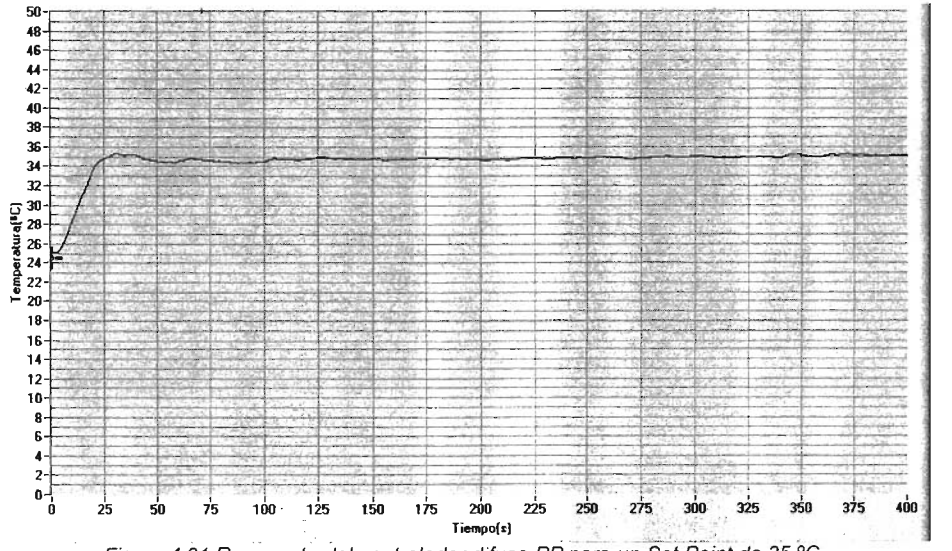


Figura 4.31 Respuesta del controlador difuso PP para un Set Point de 35 ºC.

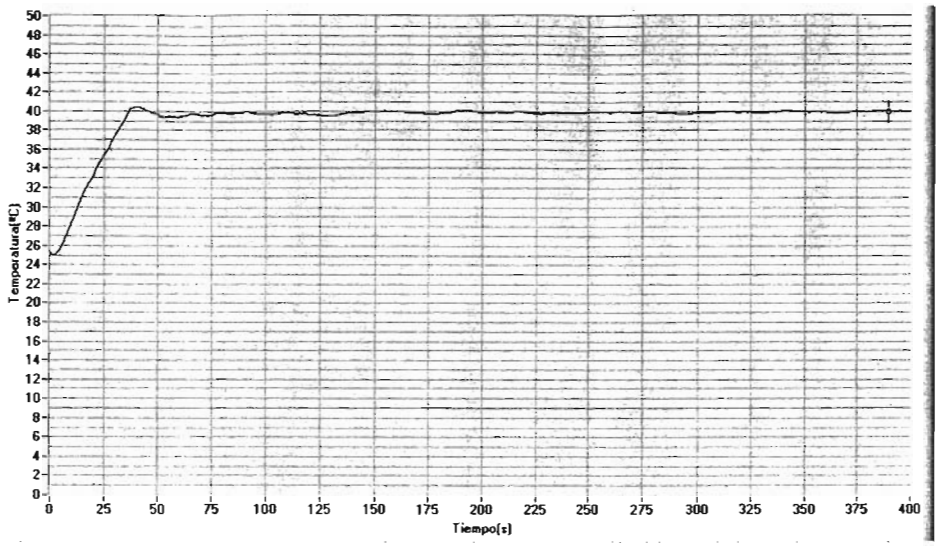


Figura 4.32 Respuesta del controlador difuso PP para un Set Point de 40 °C.

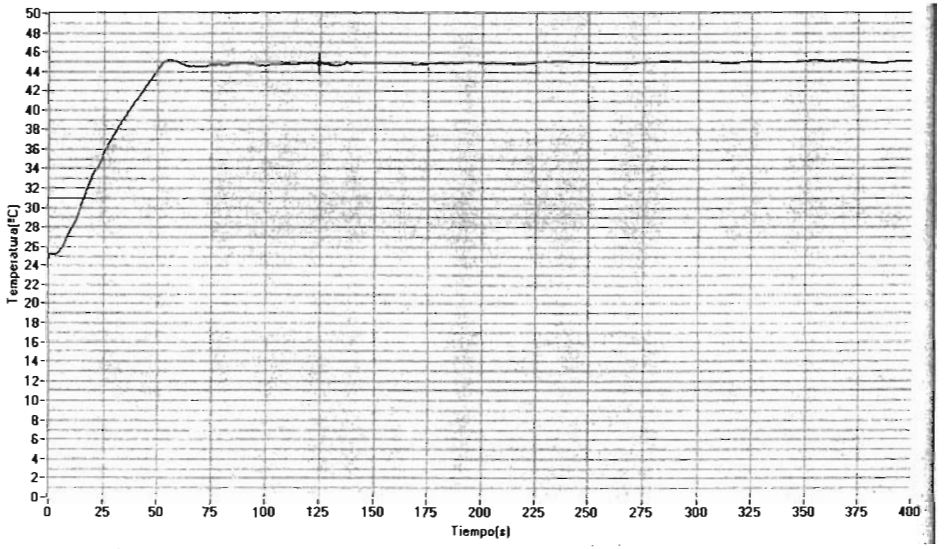


Figura 4.33 Respuesta del controlador difuso PP para un Set Point de 45 °C.

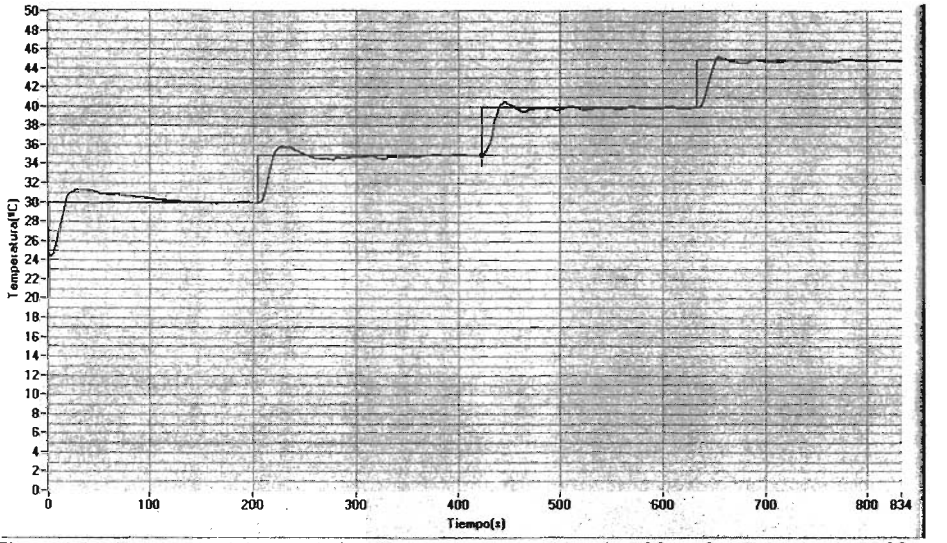


Figura 4.34 Respuesta del controlador difuso PP para cambios de 5 °C en Set Point desde 25 °C hasta 50 °C.

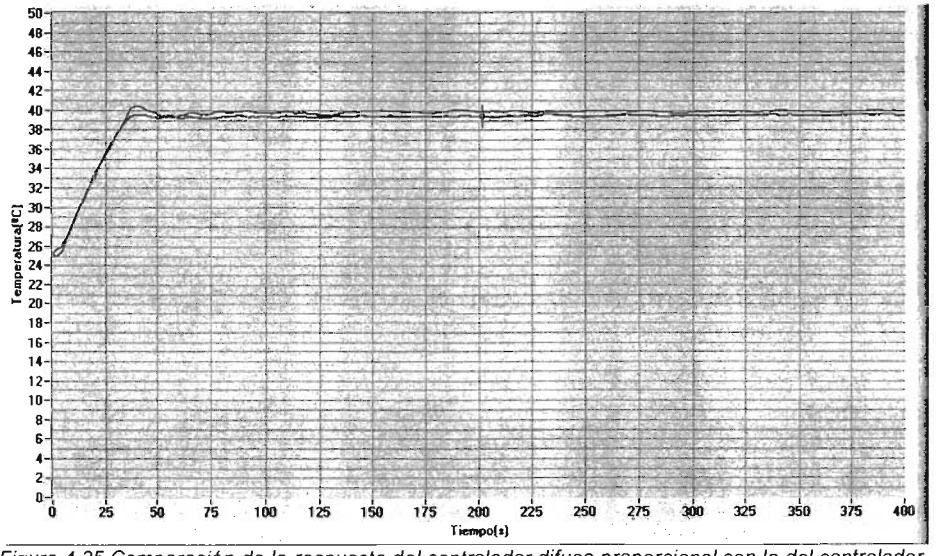


Figura 4.35 Comparación de la respuesta del controlador difuso proporcional con la del controlador proporcional prealimentado para un set point de 40 °C. (Roja: P, azul: PP)

En la Figura 4.35 se aprecia claramente el error en estado estacionario que tiene la respuesta del controlador difuso proporcional en comparación con el controlador difuso prealimentado.

En la Tabla 4.16 se hace la comparación de ambos controladores, utilizando criterios de desempeño de la respuesta transitoria en el dominio del tiempo tales como el sobre paso máximo, tiempo de levantamiento, tiempo de asentamiento,

tiempo de pico, tiempo de retardo y en el estado estacionario se utiliza el error en estado estacionario en °C.

Tabla 4.16 Resumen y Comparación de la respuesta de los controladores difusos P y PP.

Referencia	Sobre paso Máximo (%)		Tiempo de Levantamiento (s)		Tiempo de Asentamiento (s)		Tiempo de pico(s)		Tiempo de Retardo (s)		Error en estado Estacionario (°C)	
	P	PP	P	PP	P	PP	P	PP	P	PP	P	PP
30 °C	2.4	4.63	18.84	18.62	36.6	80.84	33.84	45.17	10.22	11.29	± 0.05	± 0.1
35 °C	0.57	0.83	30.45	27.32	70.3	21.86	30.45	31.14	12.84	12.58	± 0.1	± 0.1
40 °C	0	1.125	41.16	36.45	74.57	34.77	41.16	40.23	17.17	17.99	± 0.5	± 0.1
45 °C	0	0.4	54.59	52.93	83.3	49.61	54.59	58.86	24.22	23.87	± 0.6	± 0.1

De la Tabla 4.16 se puede ver que ambos controladores han satisfecho el criterio de desempeño de diseño, debido a que ambos mantuvieron el rango de la señal de error de temperatura dentro del intervalo de $\pm 1°C$ para el intervalo considerado de temperatura (0°C a 50°C), incluso mejor de lo necesario.

Comparando los las graficas de las curvar de respuesta de ambos controladores y tomando en cuenta la Tabla 4.16 se hacen las siguientes observaciones.

- EL controlador PP mantiene un comportamiento similar para un rango más amplio de temperaturas, como se pudo observar el error en estado estacionario permanece casi constante para todas las referencias consideradas, no así para el controlador P. Esto se debe a que el controlador PP utiliza el valor de la referencia como entrada y en base a esta se determina el nivel de potencia aplicado para un set point dado.
- En general el tiempo de respuesta de ambos controladores es similar, sin embargo es ligeramente más rápido el control PP (tiempo de levantamiento), aunque tiene un mayor sobre paso máximo.
- Para temperaturas arriba de 35 grados en el controlador P no logra alcanzar la referencia, no así el controlador PP (ver Figura 4.29).

De lo anterior se concluye que en general el controlador PP reporta un mejor desempeño con respecto al controlador P, ya que muestra un comportamiento más autónomo, es decir se adapta a un rango más amplio de control, no así el controlador proporcional el cual solo muestra una buena respuesta para un rango muy limitado de operación.

Con esto, finalizan las pruebas del prototipo de la Implementación de un sistema Inteligente difuso para el control de procesos basados en CC y CA, utilizando un PIC.

Como conclusiones de las pruebas realizadas y de los resultados obtenidos se destaca:

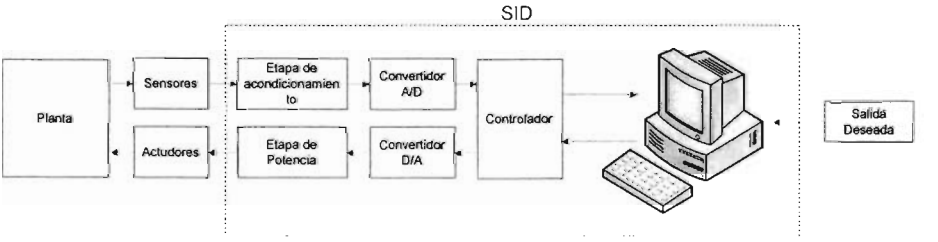
- En el diseño del controlador difuso de la temperatura interior del horno eléctrico no se necesitó conocer un modelo matemático explicito de este (necesario en el diseño de un controlador convencional), lo único necesario fue el conocimiento subjetivo (sentido común) acerca de cómo la temperatura del horno se debía controlar de una manera manual, lo cual es una característica distintiva de la utilización del control difuso.
- Una vez definidas las funciones de membresía de las entradas y salida del controlador; y definidas las reglas de control, el proceso de edición del controlador difuso en el ESD se llevo acabo de una forma rápida y directa
- EL único hardware extra al prototipo, que fue necesario para implementar el sistema de control entero de la temperatura interior del horno eléctrico, fue la instrumentación del sensor de temperatura para conectarse como una entrada analógica y la instrumentación para proporcionar la salida de potencia en CA del actuador o elemento final de control (resistencia eléctrica)
- Aunque el proceso de puesta a punto del controlador no tiene la característica de ser en línea (una característica deseable para futuros desarrollos), aun así se lleva acabo en una corta cantidad de tiempo, ya que un cambio en el diseño inicial implica solo reedicionar algunas funciones de membresía y reglas de control en el EDS, y descargar y ejecutar una vez mas el controlador diseñado, en el peor de los casos redefinir el esquema de control y las variables de control, solo que sin la necesidad de hacer un solo cambio hardware.

En resumen el prototipo propuesto mostró alta flexibilidad y buen desempeño global en su primera versión, aun así le faltan por mejorar muchos aspectos hardware y software, para que pueda ser considerado como una opción real en la automatización de procesos productivos reales.

CAPÍTULO 5. CONCLUSIONES.

De acuerdo a los objetivos del presente trabajo de tesis.

(1) EL Sistema Inteligente Difuso (SID) es el prototipo de una herramienta alternativa que fue diseñado con el objetivo de auxiliar en la enseñanza y aplicación de la lógica difusa al control de procesos basados en Corriente Continua (CC) y Corriente Alterna (CA), ya que es una sistema de integración software y hardware, que proporciona la oportunidad de completar el diseño e implementación de un controlador difuso para una aplicación particular sin la necesidad de software y hardware extra, siempre y cuando el sistema resultante cumpla con los requerimientos de la aplicación. En la figura abajo mostrada, el SID sustituye a la etapa de acondicionamiento, la etapa de conversión analógica digital (A/D), al controlador y a la etapa de conversión digital analógica (D/A), la etapa de potencia, así como el programa de control y configuración, dentro de un esquema de control digital, y solo es necesario proporcionarle las entradas analógicas respectivas y conectarle los actuadores necesarios para completar el sistema de control entero.



(2) La Tarjeta Electrónica de Procesamiento Difuso y de Entrada y Salida (TEPDES) es un sistema electrónico programable basado en microcontrolador diseñado para el control difuso de maquinas y procesos basados en CC y CA.

(3) La Interface de Programación y Visualización del Sistema Inteligente Difuso (IPVSID) es un interface gráfica de usuario diseñada como complemento de la TEPDES, que permite el diseño, la edición y monitorización del sistema de control difuso ejecutado en esta ultima.

(4) El prototipo SID fue diseñado completamente en el laboratorio de Sistemas Inteligentes perteneciente al Centro de Ciencias Aplicadas y Desarrollo Tecnológico (CCADET) de la UNAM. El prototipo al final de su diseño e implementación software y hardware alcanzó las siguientes características resumidas en las siguientes tres tablas.

Características técnicas de la Tarjeta Electrónica de Procesamiento Difuso y de Entrada y Salida (TEPDES).

Características	**Parámetros**
Entradas analógicas	3
Salidas de control	1
Salidas de CA	2
Salidas de CC	2
Resolución	Dato de 10 bits
Frecuencia de muestreo (*)	5, 20, 50, 80, 100, 120, 140, 160, 180, 200, 220 muestras por segundo
Interfaz RS232 Full Duplex	57, 600 bits por segundo
Set Point (*)	Depende del rango de salida de la aplicación
Programa de control y configuración	GUI IPVSID

(*) Estos parámetros pueden ser cambiados a solicitud del usuario.

Especificaciones Eléctricas máximas de la etapas de salidas en CC y CA

Características	**Parámetros**
Corriente Máxima de cada salida AC	*15 A*
Voltaje Máximo de cada salida AC	*127 V* (monofásicos)
Corriente Máxima de cada salida CC	*400 mA*
Voltaje Máximo de cada salida CC	*12 V*

Características del Editor de Sistemas Difusos (ESD).

• Sistema operativo XP
• Editor de Entrada y salida del controlador difuso.
• Editor de las funciones de membresía de entrada y salida
• Editor del conjunto de reglas de control difuso
• Generador y Descargador del código del controlador difuso diseñado

• Hasta 3 entradas.
• Hasta 1 Salida.
• Hasta 9 funciones de membresía por entrada.
• Hasta 9 funciones de membresía por salida.
• Hasta 729 reglas.
• Hasta tres antecedentes y 1 consecuente por regla.
• 11 caracteres por nombre.
• Funciones de membresía trapezoidales o triangulares.
• Funciones de membresía de salida singleton.
• Inferencia difusa tipo Sugeno
• Esquemas de control P, PI, PD y PP

Características de la Interface de Monitorización y Configuración (IMC).

• Sistema operativo XP
• Adquisición del estado de las variables del proceso de control difuso por el puerto de comunicaciones RS232.
• Visualización en tiempo real de cada un de las variables del proceso de control difuso.
• Configuración de los parámetros de funcionamiento de frecuencia de muestreo y Set Point de sistema de control difuso.
• Habilitación y deshabilitación de la ejecución del controlador difuso diseñado a solicitud del usuario.
• Configuración de la apariencia de los visualizadores a selección del usuario

(5) Como resultado de las pruebas realizadas, se puede decir que a través de un conjunto de mejoras e inversión de tiempo y dinero el SID podría llegar a ser una herramienta confiable, flexible y de bajo costo para el diseño de sistemas de control difusos, en la automatización de sistemas y procesos productivos reales.

BIBLIOGRAFÍA.

[1] Angulo Usategui José M., Romero Y. Susana y Angulo M. Ignacio. 2000. Microcontroladores PIC, diseño práctico de aplicaciones. Mc Graw Hill, España.

[2] Bart Kosko. 1992. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Prentice Hall, NJ.

[3] Chen Phan. 2001. *Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control System*. CRR Press.

[4] Efraim Turban, Jay E. Aronson. 2001. *Decision Support Systems Intelligent Systems*. Prentice Hall.

[5] E. H. Mamdani and S. Assilian. 1975. *An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller*. International Journal of Man-Machine Studies Vol. 7, No. 1, pp. 1-13.

[6] Leonid Reznik. 1997. *Fuzzy Controllers*. Newnes. Victoria University of technology, Melbourne, Australia.

[7] M. Sugeno and G. T. Kang. 1988. *Structure Identification of Fuzzy Model*. Fuzzy Sets and Systems, vol. 28, pp. 15-33,

[8] Manual Intel.1994. *Fuzzy Logic Applications Handbook*.

[9] Robert E. King. 1999. *Computational Intelligence in Control Engineering*. Marcel Dekker, Inc. New York.

[10] T.Takagi, and M. Sugeno. 1985. *Fuzzy Identification of Systems and its applications to Modeling and Control*. IEEE Tran. on Sys. Man, and Cybernetics, vol. 15, pp. 116-132.

[11] Y. Tsukamoto. 1979. *An Approach to Fuzzy Reasoning Method*, in Madan M. Gupta, Rammohan. K. Ragade and Ronald R Yager, editors, *Advances in fuzzy sets theory and applications*, pp.137-149. North Holland, Amsterdam.

MESOGRAFÍA.

[a] Data Sheet LCD TM162AAA6-2

Tianma

http://www.tianma.com/spec_sheets/TM162Aaa.PDF

[b] Data Sheet PIC16F87X

Microchip

http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

[c] Data Sheet MAX232

Texas Instrument

http://focus.ti.com/lit/ds/symlink/max232.pdf

[d] Data Sheet AD620

Analog Devices

http://www.analog.com/UploadedFiles/Data_Sheets/37793330023930AD620_epdf

[e] Data Sheet TIP41

Fairchild Semiconductor

http://www.fairchildsemi.com/ds/TI%2FTIP41C.pdf

[f] Data Sheet MOC3020

Fairchild Semiconductor

http://www.fairchildsemi.com/ds/MO/MOC3020-M.pdf

[g] Data Sheet MAC223

Philips Semiconductor

http://www.semiconductors.philips.com/acrobat/datasheets/MAC223_SERIES_HG_2.pdf

[h] Data Sheet LM35

National Semiconductor

http://cache.national.com/ds/LM/LM35.pdf

GLOSARIO DE ABREVIATURAS.

A/D: Analógico Digital.

CA: Corriente Alterna.

CAD: Diseño Asistido Computadora.

CA/D: Convertidor Analógico digital.

CC: Corriente Continua.

CD: Control Difuso.

D/A: Digital Analógico.

GUI: Interface Gráfica de Usuario.

EEGMO: Estado de Espera General de Modo de Operación (Abreviado MEG).

ESD: Editor de Sistemas Difusos.

FM: Función de Membresía.

GDM: Grado de Membresía

IA: Inteligencia Artificial.

IMC: Interface de Monitorización y Configuración

IPVSID: Interface de Programación y Visualización del Sistema Inteligente Difuso.

LCD: Display de cristal liquido.

ME: Modo Ejecución.

MEA: Módulo de Entradas Analógicas.

MEG: Modo de Espera General.

MOE: Modo de Operación de Ejecución (Abreviado ME).

MOP: Modo de Operación de Programación (Abreviado MP).

MP: Modo Programación.

MPS: Muestras por Segundo.

MSAC: Módulo de Salida para cargas de Corriente Alterna.

MSCC: Módulo de Salida para cargas de Corriente Continua.

P: Proporcional.

PC: Computadora Personal.

PCD: Programa de Control Difuso.

PD: Proporcional Derivativo.

PI: Proporcional Integral.

PIC: (Peripheral Interface Controller) Controlador de Interface Periférica

PID: Proporcional Integral Derivativo.

PP: Proporcional Prealimentado.

PS: Programa de Sistema.

PWM: (Pulse Width Modulation) Modulación por Ancho de Pulso.

SID: Sistema Inteligente Difuso.

TEPDES: Tarjeta Electrónica de Procesamiento Difuso y de Entrada y Salida.

APÉNDICE A. CÓDIGO DEL PROGRAMA DEL MICROCONTROLADOR.

A.1 Programa de sistema del SID.

```
;*********************************************************************
;*** PROGRAMA DE SISTEMA FINAL DEL PIC16F877                      ****
;*********************************************************************
;*
;* NOMBRE DEL ARCHIVO:        PS.ASM
;*              FECHA:        31-07-2004
;*            VERSIÓN:        1
;              AUTOR:          FERMI VÁZQUEZ VILLANUEVA
;*********************************************************************
;                                                                    *
;  ARCHIVOS REQUERIDOS:       DEFRUT.ASM                             *
;                             DEFVM.ASM                              *
;                             PA.ASM                                 *
;                                                                    *
;*********************************************************************
;                                                                    *
;  NOTAS:  CONTROLA LA OPERACION DE LA TEPDES E INTERACTUA            *
;  CON  EL PROGRAMA SOFTWARE IPVSID                                  *
;                                                                    *
;*********************************************************************

;*********************************************************************
        LIST P=16F877               ;******* TIPO DE PROCESADOR
        RADIX   HEX                 ;******* FORMATO HEXADECIMAL
        INCLUDE <P16F877.INC>       ;******* LIBRERIA DEL PROCESADOR
;*********************************************************************

;****************************************
;*** PALABRA DE CONFIGURACION  ****
;****************************************

        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _HS_OSC & _WRT_ENABLE_ON &
_LVP_OFF & _DEBUG_OFF & _CPD_OFF

;*****************************************************
;** AQUI SE INCLUYEN LAS LIBRERIAS RESPECTIVAS   **
;*****************************************************

        #INCLUDE        <defvm.asm>     ; INCLUYE EL FICHERO QUE CONTIEN ELA DEFINICION DE LAS
                                        ; VARIABLES Y MACROS UTILIZADAS EN TODO EL PROGRAMA
                                        ; DEFVM.ASM

;****************************************
;*** INICIO DEL SISTEMA OPERATIVO ***
;****************************************

        ORG     0X00                ; VECTOR DE RESET
        GOTO    INICIO
        ORG     0X04                ; VECTOR DE INTERRUPCION
        GOTO    INTERRUPCION

;************************************
;**** PROGRAMA PRINCIPAL    ******
;************************************

        #INCLUDE        <defrut.asm>    ; INCLUYE EL FICHERO QUE CONTIEN LA DEFINICION DE TODAS
                                        ; LAS RUTINAS UTILIZADAS EN TODO EL PROGRAMA
                                        ; DEFRUT.ASM
```

```
INICIO:                                 ; INICIA PROGRAMA DE SISTEMA

        CALL    INILCD                  ; INICIALIZA LCD
        CALL    CONFIGURA_USART         ; INICIALIZA USART PARA ESTABLECER LA COMUNICACIÓN
                                        ; ENTRE LA TEPDES Y LA IPVSID

;*****************************************************************************
;** ENVIA CABECERA PARA INDICAR QUE EL PIC ESTA LISTO PARA RECIBIR INSTRUCCIONES ***
;*****************************************************************************

COMIENZO:

        CALL    BORRAR_LCD              ; BORRA CONTENIDO ANTERIOR DE LA LCD, PARA ENVIAR
                                        ; EL MENSAJE DE OPERATIVIDAD DEL SID A LA LCD
        MENSAJE_LCD3    TABLA1, RENGLON1        ; ENVIA MENSAJE A RENGLON1
        MENSAJE_LCD3    TABLA2, RENGLON2        ; ENVIA MENSAJE A RENGLON2

        BANK0
        MOVLW 0X20                      ; ENVIA MENSAJE DE OPERATIVIDAD A LA IPVSID
        MOVWF DATOTX
        CALL    TRANSMITE
        MOVLW 0X21
        MOVWF DATOTX
        CALL    TRANSMITE
        MOVLW 0X22
        MOVWF DATOTX
        CALL    TRANSMITE

;*************************************************
;** EL PIC LISTO PARA RECIBIR INSTRUCCIONES    **
;*************************************************

RCCOMANDO:

        CALL    RECIBE
        MOVLW 0X01              ; CABECERA DE PROGRAMACION
        SUBWF   DATORX,W
        BTFSC   STATUS, Z
        GOTO    PROGRAMAR       ; VETE A MODO PROGRAMACION
        MOVLW 0X02              ; CABECERA DE EJECUCION
        SUBWF   DATORX, W
        BTFSC   STATUS, Z
        GOTO    EJECUTAR        ; VETE A MODO EJECUCION
        NOP
        NOP
        NOP
        GOTO    COMIENZO

;*********************************
;** MODO PROGRAMACIÓN      ***
;*********************************

PROGRAMAR:

;*************************************************************************
;** ENVIA MENSAJE DE MODO PROGRAMACION A LA LCD Y A LA IPVSID ***
;*************************************************************************

        CALL    BORRAR_LCD              ; BORRA CONTENIDO ANTERIOR DE LA LCD, PARA PODER
                                        ; ENVIAR EL MENSAJE DE MODO PROGRAMACION A LA LCD
        MENSAJE_LCD3    TABLA3, RENGLON1        ; ENVIA MENSAJE A PRIMER RENGLON
        MENSAJE_LCD3    TABLA4, RENGLON2        ; ENVIA MENSAJE A SEGUNDO RENGLON

        BANK0
        MOVLW CABECERA_MP               ; SE ENVIA LA CABECERA PARA INDICAR MODO PROGRAMACION
                                        ; EN LA IPVSID CABECERA_MP = 0X04
        MOVWF DATOTX
        CALL    TRANSMITE               ; TRANSMITE CABECERA
        BANK0
```

```
;*****************************************************************************
;*** ESPERA A RECIBIR CABECERA PARA VALIDAR SI SE CONTINUA EN MODO  PROGRAMACION ***
;*****************************************************************************

CHECACAB:

        CALL    RECIBE
        MOVLW  INICIA_PROGRAMACION   ; COMANDO  DE INICIO DE MODO PROGRAMACION
        SUBWF  DATORX, W
        BTFSS  STATUS, Z             ; ES EL COMANDO DE INICIO DE MODO PROGRAMACION
        GOTO   COMIENZO              ; NO: ERA COMADO DE TERMINACION, REGRESA A ESPERAR
                                     ; NUEVA INSTRUCCION
                                     ; SI: INICIA EL MODO PROGRAMACION
INIPROG:

;*************************************************************
;*** RECIBE CABECERA PARA DETERMINAR LA OPERACION    ***
;*************************************************************

  CALL  RECIBE                  ; RECIBE COMANDO  DE ESCRITURA
        MOVLW 0X07
        SUBWF   DATORX, W
        BTFSC   STATUS, Z       ; ES CABECERA DE ESCRITURA
        GOTO    ESCRITURA       ; SI, VE A LA RUTINA DE ESCRITURA
        GOTO    LECTURA         ; NO, ENTONCES ES EL COMANDO ERA DE LECTURA Y
                                ; SE VA A LA RUTINA LECTURA

;*********************************
;*** OPERACION DE ESCRITURA ***
;*********************************

ESCRITURA:

        CALL    RECEPCION_NUMERO_DATOS      ; SE RECIBE EL NUMERO DE DATOS A ESCRIBIR EN LA
                                            ; MEMORIA FLASH O EN LA MEMORIA EEPROM
        CALL    RECIBE                      ; RECIBE COMANDO DE SELECCION DE MEMORIA DE
                                    ; ESCRITURA (FLASH O EEPROM)
        MOVLW 0X08
        SUBWF   DATORX, W
        BTFSS   STATUS, Z
        GOTO    ESCRITURA_EEPROM    ; DE LO CONTRARIO SE RECIBIO UN COMANDO DE ESCRITURA EN LA
                                    ; MEMORIA EEPROM, Y SE VA LA RUTINA DE ESCRITURADE
                                    ; MEMORIA EEPROM

        CALL    ESCRIBIR_DATOS_FLASH        ; SE ESCRIBEN  "NUMERO" DATOS EN MEMORIA FLASH

        GOTO    CHECACAB

ESCRITURA_EEPROM:

        CALL    ESCRIBIR_DATOS_EEPROM       ; SE ESCRIBEN  "NUMERO" DATOS EN MEMORIA EEPROM

        GOTO    CHECACAB

;*********************************
;*** OPERACION DE LEECTURA ***
;*********************************

LECTURA:

        CALL    RECEPCION_NUMERO_DATOS      ; SE RECIBE EL NUMERO DE DATOS A LEER DE LA MEMORIA
                                            ; FLASH O DE LA MEM0RIA EEPROM
        CALL    RECIBE
        MOVF    DATORX, W
        BANK2
        MOVWF EEADRH                ; RECIBE DIRECCION  DE INICIO DE LECTURA  PARTE ALTA


        CALL    RECIBE
        MOVF    DATORX, W
```

```
        BANK2
        MOVWF EEADR                     ; RECIBE DIRECCION  DE INICIO DE LECTURA PARTE BAJA


        CALL    RECIBE
        MOVLW 0X08
        SUBWF   DATORX,W
        BTFSS   STATUS,Z                ; ES CABECERA DE LECTURA DE LA MEMORIA FLASH
        GOTO    LECTURA_EEPROM          ; NO: ES CABECERA DE LECTURA DE MEMORIA EEPROM,
                                        ; VA A LEER LA MEMORIA EEPROM
        CALL    LEER_DATOS_FLASH        ; SI: LECTURA DE LA MEMORIA FLASH

        GOTO    CHECACAB                ; REGRESA A RECIBIR NUEVA INSTRUCCION

LECTURA_EEPROM:

        CALL    LEER_DATOS_EEPROM       ; LECTURA DE LA MEMORIA EEPROM
        GOTO    CHECACAB                ; REGRESA A RECIBIR NUEVA INSTRUCCION

;*************************
;*** MODO EJECUCION ***
;*************************

EJECUTAR:

;*************************************************************
;*** RECIBE CABECERA  DE INICIO DEL MODO EJECUCION  ***
;*************************************************************

        CALL    RECIBE          ; RECIBIR COMANDO
        MOVLW 0X0E
        SUBWF   DATORX, W
        BTFSS   STATUS, Z       ; ¿ES COMANDO DE INICIO DE MODO EJECUCION?
        GOTO    COMIENZO        ; NO: REGRESAR A LA ESPERA DE NUEVO COMANDO DE OPERCION
                                ; SI: PROCEDER A INICIAR EL MODO EJECUCION

;*****************************************************************************************************
;** AQUI VA LA PARTE DEL PROGRAMA QUE SE ENCARGA DE MOVER DE LA MEMORIA FLASH A LA          ***
;** MEMORIA RAM, LOS VALORES DE LAS CONSTANTES DE CONTROL NECESARIAS PARA EJECUTAR          ***
;** CORRECTAMENTE EL ALGORITMO DE CONTROL DIFUSO                                           ***
;*****************************************************************************************************

        CALL    CARGAR_CONSTANTES_DE_CONTROL

;*************************************************************************
;***     SE CONFIGURA LA RECEPCION POR INTERRUPCIONES             *****
;*************************************************************************

        BSF     STATUS, IRP     ; SE CAMBIA A BANCO 2 Y 3 PARA DIRECCIONAMIENTO INDIRECTO
        CALL    INTCONFRX       ; CONFIGURA RECEPCION USANDO INTERRUPCIONES
        CALL  INTGLOB           ; HABILITACION GLOBAL DE INTERRUPCIONES
                                ; EN ESTE PUNTO YA SE ESTA EN CONDICIONES DE RECIBIR PARAMETROS

        MOVLW 0X09              ; ENVIA CABECERA DE MODO EJECUCION PARA INDICAR QUE
                                ; YA SE PUEDEN  RECIBIR PARAMETROS
        MOVWF DATOTX
        CALL    TRANSMITE

        CALL    INIGEN_MOD_EJECUCION        ; CLAREA TODAS LAS PUERTAS DE ENTRADA Y SALIDA
        CALL    BORRAR_LCD                  ; BORRA CONTENIDO ANTERIOR DE LA LCD, PARA PODER
                                            ; ENVIAR EL MENSEJE DE MODO EJECUCION A LA LCD
        MENSAJE_LCD3    TABLA3, RENGLON1    ; ENVIA MENSAJE A PRIMER RENGLON DE LA LCD
        MENSAJE_LCD3    TABLA5, RENGLON2    ; ENVIA MENSAJE A SEGUNDO RENGLON DE LA LCD

        CALL    CFCAD           ; CONFIGURA EL MUDULO AD PARA LA CONVERSION ANALOGICA DE
                                ; SEÑALES PROVENIENTES DEL PROCESO U OBJETO BAJO CONTROL
        CALL  INTCONFTX         ; PREPARA LOS REGISTROS NECESARIOS PARA LA TRANSMISION DE DATOS
                                ; DURANTE LA EJECUCION DEL SISTEMA DE CONTROL DIFUSO
        CALL  FIJAR_T_MUESTREO  ; CONFIGURA E INICIALIZA EL TIMER0 ASI COMO LOS REGISTROS DE
                                ; CONTROL ASOCIADOS AL CONTROL DE LA FRECUENCIA DE
```

```
                                        ; MUESTREO DEL SISTEMA DE CONTROL DIFUSO


;*****************************************************************************************
; SE LLAMA AL PROGRAMA DE APLICACION PARA EJECUTAR EL CONTROLADOR DIFUSO DISEÑADO  ***
; DEL CUAL SE RETORNA HASTA QUE SE ENVIA EL COMANDO DE FIN DE MODO EJECUCION      ***
;*****************************************************************************************

        CALL    PRODIF

;*************************************************************************
;***   SE RESTABLECE LA CONFIGURACION DEL SISTEMA Y DE LOS PERIFERICOS   ****
;*************************************************************************

RESTABLECER:

        CALL    INTGLOBDES              ; DESCONFIGURA INTERRUPCION GLOBAL
        CALL    INTDESCTX               ; DESCONFIGURA TRANSMISION POR INTERRUPCION
        CALL    INTDESCRX               ; DESCONFIGURA RECEPCION POR INTERRUPCION
        CALL    DCFCAD                  ; DESCONFIGURA CAD
        CALL    DCINITRIAC1             ; DESCONFIGURA TIMER1 (TRIAC1)
        CALL    DCINITRIAC2             ; DESCONFIGURA TIMER2 (TRIAC2)
        CALL    DESPWM                  ; DESCONFIGURA MODULO TIMER2 PARA PWM
        CALL    DESCONFPWM1             ; DESCONFIGURA CCP1, DE MODO PWM
        CALL    DESCONFPWM2             ; DESCONFIGURA CCP2, DE MODO PWM
        CALL    LIMPIA_SALIDAS          ; SE LIMPIAN TODAS LAS SALIDAS
        BCF     INTCON, T0IE            ; DESHABILITA INTERRUPCION POR TIMER0
        GOTO    INICIO

;*************************************************************************
;*** INCLUYE EL FICHERO FUENTE QUE CONTIENE EL PROGRAMA DE CONTROL DIFUSO ***
;*************************************************************************

        #INCLUDE        <PA.ASM>        ; INCLUYE EL FICHERO DEL PROGRMA DE APLICACION

;************************************
;***** MENSAJES PARA LA LCD    ****
;************************************

        ORG     D'2050'

                ;"1234567891234567"
TABLA1:

        MOVF    PCLATH_TEMP, W
        MOVWF   PCLATH
        MOVF    APTABLA, W
        ADDWF   PCL, F

        DT      "SID OPERATIVO", 0XFF

TABLA2:

        MOVF    PCLATH_TEMP, W
        MOVWF   PCLATH
        MOVF    APTABLA, W
        ADDWF   PCL, F
        DT      "ENVIAR COMANDO", 0XFF

TABLA3:

        MOVF    PCLATH_TEMP, W
        MOVWF   PCLATH
        MOVF    APTABLA, W
        ADDWF   PCL, F
        DT      "SID CORRIENDO EN", 0XFF
```

```
TABLA4:

        MOVF    PCLATH_TEMP, W
        MOVWF   PCLATH
        MOVF    APTABLA, W
        ADDWF   PCL, F
        DT      "MOD0 PROGRAMACION", 0XFF

TABLA5:

        MOVF    PCLATH_TEMP, W
        MOVWF   PCLATH
        MOVF    APTABLA, W
        ADDWF   PCL, F
        DT      "MODO EJECUCION", 0XFF

        END

;*****************************************
;*** FIN DE PROGRAMA DE SISTEMA  ***
;*****************************************
```

A.2 Programa de control difuso del SID.

```
;*****************************************************************************
;*** PROGRAMA DE APLICACION FINAL DEL PIC16F877                          ***
;*****************************************************************************
;*
;* NOMBRE DEL ARCHIVO:        PA.ASM
;*              FECHA:        31-07-2004
;*           VERSIÓN:         1                    ;
;*            AUTOR:          FERMI VÁZQUEZ VILLANUEVA
;
;*****************************************************************************
;                                                                            *
;  ARCHIVOS REQUERIDOS: NIN                                                  *
;                                                                            *
;*****************************************************************************
;                                                                            *
;  NOTAS:                                                                    *
;                                                                            *
; FICHERO FUENTE QUE CONTIENE EL PROGRAMA DE CONTROL DIFUSO DONDE SE         *
; ESCRIBEN LAS INSTRUCCIONES DE LLAMADA A  LAS RUTINAS DEL ALGORITMO         *
; DE CONTROL DIFUSO PARA UN CONTROLADOR DIFUSO DISENADO Y QUE SON            *
; EJECUTADAS CICLICAMENTE A LA FRECUNCIA DE MUESTREO ACTUAL DEL              *
; SISTEMA MIENTRAS NO SE ENVIE EL COMANDO DE FIN DE EJECUCION DEL            *
; PROGRAMA DE CONTROL DIFUSO                                                 *
;                                                                            *
;*****************************************************************************


ORG     D'2008'   ; APARTIR DE LA POSICION 1950 EN MEMORIA FLASH SE RESERVA
                  ; ESPACIO DE MEMORIA DE PROGRAMA PARA COLOCAR EL CODIGO DEL
                  ; PROGRAMA DE APLICACION.

PRODIF:

;*****************************************************************************
;*** AQUI SE RESERVAN TRES LOCALIDADES DE MEMORIA DE PROGRAMA PARA ESCRIBIR ***
;*** EL CODIGO DE OPERACION DE LAS INTRUCCIONES DE CONFIGURACION DE LOS     ***
;*** PERIFERICOS ENCARGADOS DEL CONTROL DE POTENCIA, ESTE CODIGO DEPENDE    ***
;*** DEL DISEÑO PARTICULAR                                                  ***
;*****************************************************************************

        NOP     ; 1
        NOP     ; 2
        NOP     ; 3

;*****************************************************************************
```

```
;** SEGMENTO DEL PROGRAMA DE APLICACION ENCARGADO DEL CONTROL DE LA FRECUENCIA DE MUESTREO **
;*****************************************************************************************

CICLO_DE_CONTROL:

        BANK2
        BTFSS   BANDERA_MUESTREO        ; ¿HA PASADO EL PERIODO DE MUESTREO?
        GOTO    CICLO_DE_CONTROL        ; NO: ESPERA A QUE SE COMPLETE
        BCF     BANDERA_MUESTREO        ; SI: EJECUTA EL ALGORITMO DE CONTROL DIFUSO

;*****************************************************************************
;** AQUI SE RESERVAN 24 LOCALIDADES DE MEMORIA DE PROGRAMA PARA ESCRIBIR  ***
;** EL CODIGO DE OPERACION DE LAS INTRUCCIONES DEL ALGORITMO DE CONTROL   ***
;** DIFUSO  PARA UN CONTROLADOR DIFUSO DISEÑADO                          ***
;*****************************************************************************

        NOP     ; 4
        NOP     ; 5
        NOP     ; 6
        NOP     ; 7
        NOP     ; 8
        NOP     ; 9
        NOP     ; 10
        NOP     ; 11
        NOP     ; 12
        NOP     ; 13
        NOP     ; 14
        NOP     ; 15
        NOP     ; 16
        NOP     ; 17
        NOP     ; 18
        NOP     ; 19
        NOP     ; 20
        NOP     ; 21
        NOP     ; 22
        NOP     ; 23
        NOP     ; 24
        NOP     ; 25
        NOP     ; 26
        NOP     ; 27

;*****************************************************************************
;** EL SIGUIENTE SEGMENTO DEL PROGRAMA DE APLICACION  SE ENCARGA DE       ***
;** CONTROL DE LA PERMANENCIA EN EL MISMO                                 ***
;*****************************************************************************

PREFIN:

        BANK2
        MOVLW   FIN_EJECUCION           ; COMANDO DE FIN DE EJECUCION
        SUBWF   CONTINUAE, W            ; COMPARAR CONTINUE CON FIN_EJECUCION
        BTFSS   STATUS, Z               ; ¿SE HA RECIBIDO EL COMANDO DE FIN DE MODO EJECUCION?
        GOTO    CICLO_DE_CONTROL        ; NO: CONTINUA CON EL PROCESO DE CONTROL DIFUSO
                                        ; (CONTINUA EN EL PROGRAMA DE APLICACION)
        RETURN                          ; SI: RETORNA DEL PROGRAMA DE APLICACION
                                        ; (SE SUSPENDE LA EJECUCION DEL CONTROLADOR DIFUSO)

;*********************************************
;****    FIN DEL PROGRMA DE APLICACIÓN  ***
;*********************************************
```

A.1 Definición de las variables y macros utilizadas en el programa del microcontrolador.

```
;*************************************************************************
;***  DEFINICION DE VARIABLES Y MACROS DEL PROGRAMA  DEL PIC16F877  ***
;*************************************************************************
```

```
;*
;*   NOMBRE DEL ARCHIVO:        DEFVM.ASM
;*                FECHA:        31-07-2004
;*              VERSIÓN:        1
;
;*                AUTOR:        FERMI VÁZQUEZ VILLANUEVA
;*****************************************************************
;                                                                *
;  ARCHIVOS REQUERIDOS: NINGUNO                                  *
;                                                                *
;*****************************************************************
;                                                                *
;  NOTAS: EN ESTE FICHERO SE DECLARAN Y DEFINEN TODAS LAS        *
;  VARIABLES, CONSTANTES Y MACROS UTILIZADAS EN EL PROGRMA       *
;  DEL PIC16F877                                                 *
;                                                                *
;*****************************************************************

;*************************************************************************
;** VARIABLES PARA OPERACIONES ARITMETICAS Y REGISTROS TEMPORALES ********
;*************************************************************************

        CBLOCK 0X70     ; TODOS LOS BANCOS

                        OPAL1
                        OPAL2
                        OPAL3
                        OPAL4
                        OPAL5
                        OPAL6
                        OPAH1
                        OPAH2
                        OPAH3
                        OPAH4
                        OPAH5
                        CONTADOR
                        FSR_TEMP
                        STATUS_TEMP
                        W_TEMP
        ENDC

;*****************************************************
;*** VARIABLES PARA MANIPULACION DE RESULTADOS ***
;*****************************************************

        CBLOCK 0X110    ; BANCO 2, ESTE BLOQUE ABARCA
                        ; DESDE LA POSICION 0X110 HASTA LA 0X12F EN RAM (32 BYTES)

                        RESAL           ; 0X110
                        RESAH           ; 0X111
                        RESBL           ; 0X112
                        RESBH           ; 0X113
                        DIRVM           ; 0X114
                        DIRSIGFUN       ; 0X115
                        NUMFM           ; 0X116
                        PXL             ; 0X117
                        PXH
                        RMINL
                        RMINH
                        RMAXL
                        RMAXH
                        NUMSINGLETON    ; 0X11D
                        DIRVMSAL
                        SALIDAL         ; 0X11F
                        SALIDAH         ; 0X120
                        SUML
                        SUMH
                        ADDL1
                        ADDH1           ; 0X124
                        ADDL2
```

```
                NUMEROH
                NUMEROL
                AUXL
                APFPE
                BANDERAS        ; 0X12A
                ER0H            ; 0X12B BYTE ALTO DE ERROR ANTERIOR
                ER0L            ; 0X12C BYTE BAJO DE ERROR ANTERIOR
                ER1H            ; 0X12D BYTE ALTO DE ERROR ACTUAL
                ER1L            ; 0X12E BYTE BAJO DE ERROR ACTUAL
                CONTADOR3       ; 0X12F

ENDC

CBLOCK 0X145    ; D'325', BANCO 2, ESTE BLOQUE ABARCA DESDE
                ; LA POSICION 0X145 HASTA LA 0X14F EN RAM (11 BYTES)
                CUENTA  ; 0x145, D'322' REGISTRO DOS DE PERIODO DE MUESTREO (0X45 = D'69')
                SETPH           ; 0X146, D'326'BYTE ALTO DEL SET POINT (0X48 = D'70')
                SETPL           ; 0X147, D'327'BYTE BAJO DEL SET POINT (0X49 = D'71')
                CONTINUAE       ; 0X148, D'328'(0X4A = D'72')
                T_MUESTREO      ; 0X149, D'329'REGISTRO DE PERIODO DE MUESTREO (0X4B = D'73')
                CCONTROL_PH     ; OX14A
                CCONTROL_PL     ; OX14B
                CCONTROL_NH     ; 0X14C
                CCONTROL_NL     ; 0X14D
                CONTROLH        ; 0X14E
                CONTROLL        ; 0X14F

ENDC

CBLOCK 0X150    ; BANCO 2, CONSTANTES DE CONTROL, ESTE BLOQUE
                ; ABARCA DESEDE LA POSICION 0X150 HASTA LA 0X15F EN RAM (16 BYTES)

                DIVME           ; 0X150
                DIVMS
                TXDAT
                RXDAT
                NFCS
                NFPE1           ; 0X155, DIRECCION
                NFPE2
                NFPE3
                LOADF
                AFPEEH
                AFPEEL
                AFCEEH
                AFCEEL
                ABREEH          ; 0X15D
                ABREEL          ; 0X15E
                SSERRORH
                SSERRORL
ENDC

CBLOCK 0X20     ; BANCO 1, ESTE BLOQUE ABARCA DESDE LA POSICION 0X20
                ; HASTA LA 0X2F EN RAM (16 BYTES)

                CONTADOR1       ; CONTADORES DE POPOSITO GENERAL
                CONTADOR2
                DATORX
                DATOTX
                DELAYL
                DELAYH
                TXDIR
                TXDIRA
                TXDIRA_TEMP
                TXDIRACOPY
                RXDIR
                RXDIRA
                DELAY_TMR2      ; VARIABLE QUE ALMACENA TEMPORALMENTE EL VALOR DE
                                ; CARGA DEL TIMER 2 PARA EL CONTRO DEL TRIAC 2
                DIRECCION_RAM   ; VARIABLE QUE LAMACENA LA DIRECCION DEL DATO
                                ; QUE SE VA ESCRIBIR EN RAM (DIRECCIONAMIENTO INDIRECTO)
```

```
                    DATO_RAM        ; VARIABLE QUE ALMACENA EL DATO QUE SE VA
                                    ; ESCRIBIR EN RAM (DIRECCIONAMIENTO INDIRECTO)
                    DATO            ; REGISTRO DE ALMACENAMIENTO TEMPORAL DE LO
                                    ; DATOS QUE SE ESCRIBEN EN LA LCD
                    APTABLA         ; APUNTADOR DE TABLAS
                    DATL            ; DATO LEIDO DE LA LCD
                    MSBNIB
                    LSBNIB
                    PCLATH_TEMP
        ENDC

;*************************************************************************
; BITS DE CONTROL PARA OPERCIONES ARITMETICAS  DE DOS O TRES BYTES  ***
;*************************************************************************

        #DEFINE  ACARREO  BANDERAS, 0  ; BADERA DE CONTROL PARA OPERACIONES
                                       ; ARITMETICAS DE MAS DE DOS BYTES

;**********************************
;** BITS DE CONTROL DE LA LCD ***
;**********************************

        #DEFINE         R_S     PORTE, 2        ; BIT DE COMANDO DATO DE LA LCD
        #DEFINE         RW      PORTE, 1        ; BIT DE LECTURA ESCRITURA DE LA LCD
        #DEFINE         ENABLE  PORTE, 0        ; BIT DE HABILITACION DE LA LCD

;*********************************************************
;** BITS DE CONTROL DEL PROCESO DE CONTROL DIFUSO **
;*********************************************************

        #DEFINE BANDERA_MUESTREO        BANDERAS, 1     ; BANDERA DE CONTROL DEL
                                                        ;  PERIODO DE MUESTREO
        #DEFINE BANDERA_RECEPCION       BANDERAS, 2     ; BANDERA DE CONTROL DE LA RECEPCIÓN
                                                        ; POR INTERRUPCIONES
        #DEFINE FINAL_DE_TRANSMISION    BANDERAS, 3     ; BANDERA DE CONTROL DE TRANSMISION

        #DEFINE TIEMPO_CONTROL          PORTC, 3        ; PIN QUE MUESTRA EL TIEMPO DE DURACION
                                                        ; DEL PROCESO DE CONTROL DIFUSO (BIT EN 1)

;*********************************************************
;** BITS DE PULSO DE DISPARO DEL LOS TRIACS *********
;*********************************************************

        #DEFINE PULSO_TRIAC1    PORTB, 1        ; HACE REFERENCIA AL PULSO DE DISPARO DEL TRIAC1
        #DEFINE PULSO_TRIAC2    PORTB, 2        ; HACE REFERENCIA AL PULSO DE DISPARO DEL TRIAC2

;*****************************************************************************************************
;** BITS DE CONTROL DE LOS TIPO DE CARGA PARA UNA SALIDA TIPO SNZP Y UNA SALIDA TIPO  CCONTROL ***
;*****************************************************************************************************

        #DEFINE LOADP   LOADF, 0        ; PARA DETERMINAR EL TIPO DE CARGA QUE TIENE
                                        ; LA SALIDA POSITIVA
        #DEFINE LOADN   LOADF, 1        ; PARA DETERMINAR EL TIPO DE CARGA QUE TIENE
                                        ; LA SALIDA NEGATIVA
        #DEFINE MODAB   LOADF,2         ; PARA DETERMINAR EL MODO DE OPERACION

;****************************************
;** CONSTANTES DE PROGRAMA      *****
;****************************************

NUMEA = 0X07            ; B'0000 0111' LOS TRES BITS  MENOS SIGNIFICATIVOS DEL PUERTO A COMO
                        ; ENTRADAS Y LOS RESTANTES COMO SALIDAS
RXINI = 0X40            ; BANCO  2 CON DIRECCIONAMIENTO INDIRECTO ES DECIR 0X140, D'320'
TXINI = 0X30            ; BANCO  2 CON DIRECCIONAMIENTO INDIRECTO ES DECIR 0X130, D'304'
TXINICOPY = 0XB0        ; BANCO 3 CON DIRECCIONAMIENTO INDIRECTO ES DECIR 0X1B0, D'432'
TXCABECERA = 0XF7               ; D'247'
DC_FIN_EJECUCION = 0X4A         ; BANCO  2 CON DIRECCIONAMIENTO INDIRECTO ES DECIR 0X148, D'328'
DIRCCL = 0XC4                   ; DIRECCION BAJA EN MEMORIA FLASH DE LAS CONSTANTES DE CONTROL
DIRCCH = 0X09                   ; DIRECCION ALTA EN MEMORIA FLASH DE LAS CONSTANTES DE CONTROL
CABECERA_MP = D'4'              ; CABECERA MODO PROGRMACION
```

```
MEMORIA_EEPROM = D'19'          ; COMANDO DE SELECCIÓN DE MEMORIA EEPROM
MEMORIA_FLASH = D'8'            ; COMANDO DE SELECCIÓN DE MEMORIA FLASH
CONTINUA_PROGRAMACION = D'5'    ; COMANDO DE  PERMANENCIA EN EL MODO PROGRAMACION
CONTINUA_EJECUCION = D'11'      ; COMANDO DE  PERMANENCIA EN EL MODO EJECUCION
INICIA_PROGRAMACION = D'6'      ; COMANDO DE INICIO DEL MODO PROGRAMACION
INICIA_EJECUCION = D'14'        ; COMANDO DE INICIO DEL MODO PEJECUCION
FIN_PROGRAMACION = D'16'        ; COMANDO DE FIN DE MODO PROGRMACION
FIN_EJECUCION = D'10'           ; COMANDO DE FIN DE MODO EJECUCION
MODO_PROGRAMACION = D'1'        ; COMANDO DE SELECCIÓN DE MODO PROGRMACION
MODO_EJECUCION = D'2'           ; COMANDO DE SELECCIÓN DE MODO EJECUCION
COMANDO_ESCRITURA = D'7'        ; COMANDO DE ESCRITURA
COMANDO_LECTURA = D'17'         ; COMANDO DE LECTURA

;*************************************************************
;*** DEFINICION DE CONSTANTES DE CONTROL  DE LA LCD  ***
;*************************************************************

RENGLON1 = 0X80          ; COMANDO PARA POSICIONAR A LA LCD EN EL PRIMER RENGLON VISIBLE DE LA LCD
RENGLON2 = 0XC0          ; COMANDO PARA POSICIONAR A LA LCD EN EL SEGUNDO RENGLON
                         ; VISIBLE DE LA LCD
CLRFLCD  = 0X01          ; BORRA LA PANTALLA Y REGRESA EL CURSOR A LA DIRECCION
                         ; CERO (HOME) RHOMELCD = 0X02; RETORNA EL CURSOR A LA POSICION CERO (HOME)
SETLCD1  = 0X38          ; CONFIGURA LCD, CON BUS DE DATOS 8 BITS (DL = 1), 2 LINEAS (N = 1),
                         ; MATRIZ DE CARACTER DE 5*7 (F = 0)
SETLCD2 = 0X3C           ; CONFIGURA LCD, CON BUS DE DATOS 8 BITS (DL = 1), 2 LINEAS(N = 1),
                         ; MATRIZ DE CARACTER DE 5*10 (F = 1)
ONLCD1   = 0X0C          ; ENCIENDE PANTALLA (D = 1), CURSOR APAGADO(C = 0), INTERMITENCIA DEL
                         ; CURSOR APAGADA (B = 0)
ONLCD2   = 0X0E          ; ENCIENDE PANTALLA (D = 1), CURSOR ENCENDIDO(C = 1),
                         ; INTERMITENCIA DEL CURSOR APAGADA (B = 0)
ONLCD3   = 0X0F          ; ENCIENDE PANTALLA (D = 1), CURSOR ENCENDIDO(C = 1),
                         ; INTERMITENCIA DEL CURSOR ENCENDIDA (B = 1)
MINCHAR1 = 0X06          ; INCREMENTO DEL CURSOR EN UNA POSICION (I/D = 1), SIN DEZPLAZAMIENTO
                         ; DEL TEXTO (S = 0)

;*****************************
;**DEFINICION DE MACROS ***
;*****************************

BANK0  MACRO                    ; MACRO DE SELECCION DEL BANCO 0
       BCF     STATUS, RP0
       BCF     STATUS, RP1
       ENDM

BANK1  MACRO                    ; MACRO DE SELECCION DEL BANCO 1
       BSF     STATUS, RP0
       BCF     STATUS, RP1
       ENDM

BANK2  MACRO                    ; MACRO DE SELECCION DEL BANCO 2
       BCF     STATUS, RP0
       BSF     STATUS, RP1
       ENDM

BANK3  MACRO                    ; MACRO DE SELECCION DEL BANCO 3
       BSF     STATUS, RP0
       BSF     STATUS, RP1
       ENDM

;*****************************************************************
;**  MACRO DE CONTROL DE  LECTURA DE LA MEMORIA FLASH  ***
;*****************************************************************

FLASHRD MACRO
       BSF     STATUS, RP0      ; BANCO 3
       BSF     EECON1, RD
       NOP
       NOP
       NOP
       BCF     STATUS, RP0      ; BANCO 2
```

```
        ENDM

;*****************************************************************
;** MACRO DE CONTROL DE LECTRURA DE LA MEMEORIA EEPROM ***
;*****************************************************************

EEPRD   MACRO
        BSF     STATUS, RP0
        BSF     EECON1, RD
        BCF     STATUS, RP0
        ENDM

;*****************************************************
;*** MACROS PARA ESCRIBIR UN MESAJE EN LA LCD ***
;*****************************************************

; CON ESTA MACRO LAS TABLAS  PUEDEN SER ESCRITAS EN CUALQUIER PARTE DEL MEMORIA DE PROGRAMA
; SOLO ESCRIBE LO CARACTERES DEL MENSAJE, ESTE MACRO SOLO PUEDE SER LLAMADO DESDE EL BANCO CERO
; DE LA MEMORIA DE PROGRAMA

MENSAJE_LCD3    MACRO  NUMTABLA, NUMRENGLON

        LOCAL   FIN_MENSAJE
        LOCAL   REP
        LOCAL   INCREMENTA_PCH
        LOCAL   NO_IMCREMENTA_PCH
        BANK0
        MOVLW NUMRENGLON                ; RENGLON DE INICIO EN LA LCD
        CALL    CONTLW                  ; ENVIA COMANDO A LA LCD
        CALL    RET40US                 ; RETARDO DE ESPERA INSTRUCCION
        CLRF    APTABLA                 ; APUNTA A LA PRIMERA LETRA DEL MENSAJE
REP:

        MOVLW HIGH NUMTABLA
        MOVWF PCLATH_TEMP
        MOVLW LOW     NUMTABLA          ; PARTE BAJA DE LA DIRECCION DE PC DE LA TABLA QUE SE VA A
ESCRIBIR
        ADDLW 0X04
        BTFSC   STATUS, C
        GOTO    INCREMENTA_PCH
        ADDWF   APTABLA,W
        BTFSC   STATUS,C        ; VERIFICA SI EL PCL NO SE DESBORDA CON LA
                                ; SUMA DEL APUNTADOR A LOS CARACTERES DE LA TABLA A ESCRIBIR

INCREMENTA_PCH

        INCF    PCLATH_TEMP,F           ; SI: SUMA 1 A LA PARTE ALTA DEL PC
        MOVLW HIGH NUMTABLA
        MOVWF PCLATH
        CALL    NUMTABLA                ; AQUIERE EL CARACTER ACTUAL DE LA TABLA
        MOVWF DATO
        MOVLW HIGH FIN_MENSAJE
        MOVWF PCLATH
        MOVLW 0XFF                      ; CABECERA DE FIN DE MENSAJE
        SUBWF   DATO,W
        BTFSC   STATUS, Z                       ; ¿SE HA RECIBIDO LA CABECERA DE FIN DE MENSAJE?
        GOTO    FIN_MENSAJE             ; SI: FINALIZO LA ESCRITURA DEL MENSAJE
        MOVF    DATO,W                  ; NO:
        CALL   DATALW                   ; ESCRIBE LETRA EN LA LCD
        INCF   APTABLA, F                       ; APUNTA A LA SIGUIENTE LETRA DEL MENSAJE
        GOTO   REP                      ; NO: ESCRIBE EL SIGUENTE CARACTER

FIN_MENSAJE:

        CLRF   PCLATH
        ENDM
;*********************************************
;** MACRO DE EVALUACION DE REGLAS        **
;** (REGLAS DEFINIDAS EN MEMORIA FLASH) **
;*********************************************
```

```
EVA_REG_FLASH  MACRO

        LOCAL   REPETIR, MAS2ANTE, MAS_ANTE, CONSE, FIN_REGLAS

REPETIR:

        FLASHRD
        MOVF    EEDATA, W
        BTFSC   STATUS, Z
        GOTO    FIN_REGLAS
        MOVWF   FSR

;*****************************************************************************************
;*** CARGA EL OPERANDO 1 DEL LA OPARACION MIN CON EL VALOR DE MEBRESIA ACTUAL ***
;*****************************************************************************************

        MOVF    INDF, W
        MOVWF OPAL1
        INCF    FSR, F
        MOVF    INDF, W
        MOVWF OPAH1

        INCF    EEADR, F            ; APUNTA A LA SIGUIENTE DIRECCION DE LA EEPROM PARA
                                    ; VERIFICAR LA REGLA
        FLASHRD                     ; SE EJECUTA EL PROCESO DE LECTURA DE LA MEMORIA FLASH

;*************************************************************************
; REVISAR SI NO SE TRATA DE UNA REGLA CON UN SOLO ANTECEDENTE  ***
;*************************************************************************

        BTFSS   EEDATH, 0

        GOTO    MAS2ANTE            ; NO, SE TRATA DE UNA REGLA CON 2 O MAS ANTECEDENTES

;*************************************************************************
;SI EFECTIVAMENTE SE TRATA DE UNA REGLA CON UN ANTECEDENTE  ***
;*************************************************************************

        MOVF    OPAL1, W
        MOVWF RMINL
        MOVF    OPAH1, W
        MOVWF RMINH
        GOTO    CONSE

;*********************************************
; CONTINUA CON EL PROCESO NORMAL        ***
; DE REGLAS DE DOS O MAS ANTECEDENTES  ***
;*********************************************

MAS2ANTE:

        MOVF    EEDATA, W
        MOVWF   FSR

;*****************************************************************************************
;*** CARGA EL OPERANDO 2 DEL LA OPARACION MIN CON EL VALOR DE MEBRESIA ACTUAL  ***
;*****************************************************************************************

        MOVF    INDF, W
        MOVWF OPAL2
        INCF    FSR, F
        MOVF    INDF, W
        MOVWF OPAH2

MAS_ANTE:

        CALL    MIN_BAS
        INCF    EEADR, F
        FLASHRD
```

```
        BTFSC   EEDATH, 0
        GOTO    CONSE
        MOVF    EEDATA, W
        MOVWF   FSR
        CALL    STOPMIN
        GOTO    MAS_ANTE

CONSE:

        MOVF    EEDATA, W
        MOVWF   FSR
        CALL    LDOPMAX
        CALL    MAX_BAS
        DECF    FSR, F
        MOVF    RMAXL, W
        MOVWF   INDF
        INCF    FSR, F
        MOVF    RMAXH, W
        MOVWF   INDF
        INCF    EEADR, F
        GOTO    REPETIR

FIN_REGLAS:

  ENDM

;*************************************************
;** MACRO DEL PROCESO DE DEFUZZYFICACION **
;*************************************************

DEFUZZY  MACRO

        LOCAL   REGRESA         ; SERIE QUE CORRESPONDEN AL VALOR DIGITAL
        CLRF    SUML            ; DE LA SALIDA DEL SISTEMA
        CLRF    SUMH
        CLRF    ADDL1
        CLRF    ADDH1
        CLRF    ADDL2

REGRESA:

        MOVF    DIRVMSAL, W
        MOVWF   FSR
        MOVF    INDF, W
        MOVWF   OPAL1
        INCF    FSR, F
        MOVF    INDF, W
        MOVWF   OPAH1
        INCF    FSR, W
        MOVWF   DIRVMSAL
        CALL    SUM_DEN
        FLASHRD
        MOVF    EEDATA, W
        MOVWF   OPAL2
        MOVF    EEDATH, W
        MOVWF   OPAH2
        INCF    EEADR, F
        CALL    MULT1
        CALL    SUMATORIA

        DECFSZ  NUMSINGLETON, F
        GOTO    REGRESA

        CALL    LDOPA
        CALL    DIVISION
        BANK0
        MOVF    TXDIRA, W               ; TRAE DIRECCION ACTUAL DEL BUFER DE TRANSMISION
        MOVWF   FSR
        BANK2
        MOVF    OPAH2, W
```

```
        MOVWF  SALIDAH
        MOVWF  INDF                     ; ENVIA PRIMER DATO POR EL PUERTO SERIE
        INCF   FSR, F
        MOVF   OPAL2, W
        MOVWF  SALIDAL
        MOVWF  INDF                     ; ENVIA SEGUNDO DATO POR EL PUERTO SERIE
        INCF   FSR, W
        BANK0
        MOVWF  TXDIRA                   ; SALVA DIRECCION ACTUAL DEL BUFFER DE TRANSMISION
        BANK2
        ENDM

;*****************************************
;** MACRO DE EVALUACION DE REGLAS **
;** (DEFINIDAS EN MEMORIA EEPROM)    **
;*****************************************

EVA_REG_EEPROM  MACRO

    LOCAL               REPETIR, MAS2ANTE, MAS_ANTE, EVA_MAX, FIN_REGLAS

REPETIR:

        EEPRD                   ; SE LE LA MEMORIA EEPROM
        MOVF   EEDATA,W         ; SE MUEVE EL DATO LEIDO (EEDATA) AL WORKING REG(W)
        BTFSC  STATUS,Z         ; VERIFICA QUE NO SEA CERO
        GOTO   FIN_REGLAS       ; SI ES CERO SE HA TERMINADO DE EVALUAR TODAS LAS REGLAS
        MOVWF  FSR              ; SI NO ES CERO, ENTOCES  A UN FALTEN REGLAS POR EVALUAR
                                ; POR LO TANTO MUEVE EL DATO LEIDO DE "EEDATA" AL REGISTRO "FSR",
                                ; ESTO QUIERE DECIR QUE SE TRATA DEL
                                ; PRIMER ANTECEDENTE DE UNA REGLA (DIRECCION DE RAM DONDE ESTA
                                ; ALMACENDO EL VALOR DE MEMBRESIA DEL VALOR ACTUAL DE VARIABLE
                                ; LINGUISTICA DE ENTRADA
                                ; A UN CONJUTO DIFUSO DEL LA MISMA)

;*************************************************************************************
;*** CARGA EL OPERANDO 1 DEL LA OPARACION MIN CON EL VALOR DE MEBRESIA ACTUAL ****
;*************************************************************************************

        MOVF   INDF, W
        MOVWF  OPAL1
        INCF   FSR, F
        MOVF   INDF, W
        MOVWF  OPAH1

        INCF   EEADR, F; APUNTA A LA SIGUIENTE DIRECCION DE LA EEPROM PARA VERIFICAR LA REGLA
        EEPRD                   ; SE EJECUTA EL PROCESO DE LECTURA DE LA MEMORIA EEPROM
        MOVLW  0XFF
        SUBWF  EEDATA, W
        BTFSS  STATUS, Z
        GOTO   MAS2ANTE         ; SE TRATA DE UNA REGLA DE DOS O MAS ANTECEDENTES

;*************************************************************************
; SI EFECTIVAMENTE SE TRATA DE UNA REGLA CON UN ANTECEDENTE ***
;*************************************************************************

        MOVF   OPAL1, W
        MOVWF  RMINL
        MOVF   OPAH1, W
        MOVWF  RMINH
        GOTO   EVA_MAX                  ; VE A LA EVALUACION DEL CONSECUENTE

;************************************************
;  CONTINUA CON EL PROCESO NORMAL        ****
; DE REGLAS DE DOS O MAS ANTECEDENTES  ****
;************************************************
MAS2ANTE:

        MOVF   EEDATA, W
        MOVWF   FSR
```

```
;*****************************************************************************
;*** CARGA EL OPERANDO 2 DEL LA OPARACION MIN CON EL VALOR DE MEBRESIA ACTUAL *****
;*****************************************************************************

        MOVF    INDF, W
        MOVWF   OPAL2
        INCF    FSR, F
        MOVF    INDF, W
        MOVWF   OPAH2

MAS_ANTE:

        CALL    MIN_BAS         ; OBTIENE EL MINIMO
        INCF    EEADR, F
        EEPRD
        MOVF    EEDATA, W
        SUBLW   0XFF
        BTFSC   STATUS, Z
        GOTO    EVA_MAX
        MOVF    EEDATA, W
        MOVWF   FSR
        CALL    STOPMIN
        GOTO    MAS_ANTE

EVA_MAX:

        INCF    EEADR, F
        EEPRD
        MOVF    EEDATA, W
        MOVWF   FSR
        CALL    LDOPMAX
        CALL    MAX_BAS
        DECF    FSR, F
        MOVF    RMAXL, W
        MOVWF   INDF
        INCF    FSR, F
        MOVF    RMAXH, W
        MOVWF   INDF
        INCF    EEADR, F
        GOTO    REPETIR

FIN_REGLAS:

        ENDM

;*************************************
;*** FIN DE FICHERO DEFVM.ASM  ***
;*************************************
```

A.4 Librería de rutinas del microcontrolador

```
;*****************************************************************************
;*** LIBRERIA DE RUTINAS  DEL PIC16F877                                  ***
;*****************************************************************************
;*
;*  NOMBRE DEL ARCHIVO:        DEFRUT.ASM
;*               FECHA:        31-07-2004
;*            VERSIÓN:         1
;*              AUTOR:         FERMI VÁZQUEZ VILLANUEVA
;*
;*****************************************************************************
;                                                                            *
;  ARCHIVOS REQUERIDOS: NINGUNO                                              *
;                                                                            *
;*****************************************************************************
;                                                                            *
;  NOTAS: FICHERO FUENTE QUE CONTIENE LA DEFINICION DE TODAS                 *
```

```
;  LAS RUTINAS UTILIZADAS POR EL PROGRAMA DE SISTEMA, POR      *
;  EL PROGRAMA DE CONTROL DIFUSO Y LAS RUTINAS                 *
;  DE INTERRUPCION                                             *
;                                                              *
;**************************************************************

;**************************************************************
;*** RUTINA DE INICIALIZACION DE PUERTOS GENERAL (INIGEN) ***
;**************************************************************

INIGEN_MOD_EJECUCION:

        BANK0
        CLRF    PORTA                   ; LIMPIA LA PUERTA A
        CLRF    PORTB                   ; LIMPIA LA PUERTA B
        CLRF    PORTC                   ; LIMPIA LA PUERTA C
        CLRF    PORTE                   ; LIMPIA LA PUERTA E
        CALL    HABILITAR_EJECUCION     ; PONE CONTINUAE = 11
        RETURN

;*******************************************************************************
;***  RUTINAS DE CONFIGURACION E INICIALIZACION DEL USART PARA          ***
;***  LA COMUNICACIÓN SERIE CON LA PC VIA EL BUS RS232                  ***
;*******************************************************************************

CONFIGURA_USART:

        BANK1
        MOVLW   B'11000000'     ; PONER EL BIT 7 DEL PUERTO B COMO ENTRADA
                                (RX PARA LA USART) Y TODOS LOS DEMAS COMO SALIDAS
                                (BIT 6, TX DE LA USART)

        MOVWF   TRISC
        BSF     TXSTA, BRGH     ; COMUNICACION DE ALTA VELOCIDAD
        MOVLW   D'21'
        MOVWF   SPBRG           ; VELOCIDAD DE TRANSMISION DE 57600 BAUDIOS
        BCF     TXSTA, SYNC     ; COMUNICACION ASINCRONA
        BSF     TXSTA, TXEN     ; HABILITA MODULO DE TRANSMISION TX
        BANK0
        BSF     RCSTA, SPEN     ; HABILITA LA EL PUERTO SERIE
        BSF     RCSTA, CREN     ; HABILITA EL MOSULO DE RECEPCION RX
        RETURN

;*****************************************************
;*** CONFIGURACION DE TRANSMISION Y RECEPCION  ***
;*** DEL USART POR INTERRUPCIONES                ***
;*****************************************************

INTCONFTX:

        BANK0
        MOVLW   TXINI           ; SE INICIALIZAN LOS REGISTROS DE CONTROL
                                ; DE TRANSMISION POR INTERRUPCION
        MOVWF   TXDIRA
        MOVLW   TXINICOPY
        MOVWF   TXDIR
        BANK1
        BCF     PIE1, TXIE      ; SE ACTIVA INTERRUPCION POR TRANSMISION
        BCF     PIE1, PSPIE     ; SE HABILITAN LAS INTERRUPCIONES DEL MODULO PSP, PERO PARA UN USO
                                ; ALTERNATIVO (CONTROL DE TRANSMISION DE DATOS POR LA USART)
        RETURN

INTDESCTX:

        BANK1
        BCF     PIE1, TXIE
        BCF     PIE1, PSPIE
        BANK0
        RETURN
```

```
INTCONFRX:

        BANK1
        BSF     PIE1, RCIE          ; SE ACTIVA INTERRUPCION POR RECEPCION DE LA USART
        BANK0
        MOVLW RXINI                 ; SE INICIALIZAN LOS REGISTROS DE CONTROL DE LA  RECEPCION POR
                                    ; INTERRUPCION
        MOVWF  RXDIR
        MOVWF  RXDIRA
        RETURN

INTDESCRX:

        BANK1
        BCF     PIE1, RCIE          ; SE DESACTIVA INTERRUPCION POR RECEPCION DE LA USART
        BANK0
        RETURN


;************************************************
;*** RUTINA DE RECEPCION POR EL USART  ***
;*** UTILIZANDO EL METODO DE POLEO     ***
;************************************************

RECIBE:

        BANK0

CHECA:

        BTFSS   PIR1, RCIF          ; HA FINALIZADO LA RECEPCION DE UN DATO
        GOTO    CHECA               ; NO: CONTINUA CHECANDO
        MOVF    RCREG, W            ; SI: LEE EL DATO RECIBIDO
        MOVWF  DATORX               ; ALMACENA EL DATO RECIBIDO
        RETURN

;******************************************************
;*** RUTINA DE TRANSMISION POR EL PUERTO SERIE  ***
;*** UTILIZANDO EL METODO DE POLEO              ***
;******************************************************

TRANSMITE:

        BANK0

LICAT:

        BTFSS   PIR1, TXIF          ; HA FINALIZADO LA TRANSMISIÓN DE UN DATO
        GOTO    LICAT               ; NO: ESPERA A QUE TERMINE
        MOVF    DATOTX, W           ; SI: PREPARATE PARA ENVIAR NUEVO DATO
        MOVWF  TXREG                ; COPIA NUEVO DATO PARA TRANSMISIÓN
        RETURN


;************************************************
;*** RUTINA DE CONFIGURACION DE CAD (CFCAD) ***
;************************************************

CFCAD:

        BANKSEL  ADCON1             ; BANCO 1
        MOVLW   0X82
        MOVWF   ADCON1              ; EL RESULTADO SE JUSTIFICA A LA DERECHA, ADFM = 1; 5 ENTRADAS
                                    ; ANALOGICAS, PCFG3:0 = b'0010'
        MOVLW   NUMEA
        MOVWF   TRISA               ; LOS TRES PRIMEROS BITS DEL PUERTO A COMO ENTRADAS, PARA LAS TRES
                                    ; ENTRADAS ANALOGICAS RESPECTIVAMENTE.
        BANKSEL  ADCON0             ; BANCO 0
        MOVLW   0X81                ; FRECUENCIA DE CONVERSION IGUAL A Fosc/32 CON RELOJ DE 20MHZ, TAD =
```

```
                                ; 1.6US, SE ACTIVA EL CONVERTIDOR AD, ADON = 1
        MOVWF   ADCON0          ; SE SELECIONA EL CANAL0 POR DEFAULT
        RETURN

DCFCAD:

        BANKSEL  ADCON1
        CLRF    ADCON1  ;
        BANKSEL  ADCON0
        CLRF    ADCON0
        RETURN


;*********************************************************************************
;*** RUTINA DE CONFIGURACION DEL TIMER1  Y DE LA INTERRUPCION              ***
;*** EXTERNA POR RB0 PARA DETECTAR EL CRUCE POR CERO DE LA SEÑAL DE AC ***
;*** Y GENERAR EL PUSLSO DE CONTROL DEL          TRIAC1                   ***
;*********************************************************************************

INITRIAC1:

        BANK1
        MOVLW B'00000001'       ; BIT 1 DEL PUERTO B COMO ENTRADA, LOS DEMAS COMO SALIDAS
        MOVWF TRISB
        BSF     OPTION_REG, 7   ; RESIETENCIAS DE PULL-UP DE LA PUERTA B DESACTIVADAS
        BSF     OPTION_REG, INTEDG      ; FLANCO ASCENDENTE PARA INTERRUPCION EXTERNA
        BSF     INTCON,INTE             ; HABILITA LA INTERRUPCION EXTERNA
        BSF     PIE1, TMR1IE    ; HABILITA LAS INTERRUPCIONES POR DESBORDAMIENTO DEL TIMER1
        BANK0
        MOVLW B'00110000'       ; PRESCALADOR DE 1/8, RELOJ INTERNO
        MOVWF T1CON
        BCF     PULSO_TRIAC1    ; SE LIMPIA EL PULSO DE DISPARO DEL TRIAC 1
        MOVLW 0X2A
        MOVWF DELAYL            ; CICLO DE TRABAJO DEL TIMER 1 EN CERO INICIALMENTE
        MOVLW 0X14
        MOVWF DELAYH
        RETURN

DCINITRIAC1:

        BANK1
        BCF     INTCON, INTE    ; DESHABILITA INTERRUPCION EXTERNA
        BCF     PIE1, TMR1IE    ; DESHABILITA INTERRUPCION POR DESBORDAMIENTO DEL TIMER 1
        CLRF    TRISB           ; TODOS LOS BITS DEL PUERTO B COMO SALIDAS
        BANK0                   ; BANCO 0
        CLRF   T1CON            ; SE CLAREA EL REGISTRO DE CONTROL DEL TIMER 1
        RETURN

;*************************************************************************
;*** RUTINA DE CONFIGURACION DEL TIMER2, PARA GENERAR EL PULSO  ***
;*** DE CONTROL DEL TRIAC2                                              ***
;*************************************************************************

INITRIAC2:

        BANK0
        MOVLW B'01001011'       ; PREDIVISOR 1/16 (BIT 0 Y 1 A UNO), POSTDIVISOR 1/10, NO SE ENCIENDE
                                ; TIMER2 AQUI (OFF)
        MOVWF T2CON
        CLRF    DELAY_TMR2      ; INICILIZA REGISTRO DE RETARDO DEL TIMER 2 EN CERO
        BCF     PULSO_TRIAC2    ; SE LIMPIA PULSO DE DISPARA DEL TRIAC2
        BANK1
        BSF     PIE1, TMR2IE    ; SE PERMITE LAS INTERRUPCIONES DEL TIMER2
        RETURN

DCINITRIAC2:

        BANK0
        CLRF    T2CON           ; SE CLAREA EL REGISTRO DE CONTROL DEL TIMER2
        BANK1
```

```
        BCF     PIE1, TMR2IE        ; SE PROHIBEN LAS INTERRUPCIONES DEL TIMER2
        RETURN


;*****************************************************************************
;*** RUTINAS DE CONFIGURACION  Y DESCONFIGURACION DE LOS MODULOS PWM    ***
;*****************************************************************************

INIPWM:

        BANK1
        MOVLW  0XFF
        MOVWF  PR2                  ; SE CARGA (255) EL  REGISTRO DE PERIODO DEL TIMER 2
        BANK0
        MOVLW  0X04                 ; SE ENCIENDE EL TIMER 2 (ON), PRESCALER 1/1 Y POTSCALER 1/1
        MOVWF  T2CON
        RETURN

DESPWM:

        BANK0
        CLRF    T2CON   ; SE CLAREA EL REGISTRO DE CONTROL DEL TIMER2 (OFF)
        RETURN


CONFPWM1:                           ;SE CONFIGURA EL MODULO PWM1

        BANK0
        CLRF    CCPR1L              ; SE INICIALIZA LA PARTE ALTA DEL CICLO DE TRABAJO
        CLRF    CCPR1H              ; REGISTRO DE CONTROL CICLO DE TREABAJO INICIALMENTE A CERO
        BCF    CCP1CON, CCP1X       ; SE INICILIZAN LOS DOT BITS MENOS SIGNIFICATIVOS DE CICLO DE TRABAJO
        BCF    CCP1CON, CCP1Y
        MOVLW   0X0F                ; SE CONFIGURA EL MODULO CCP1, EN MODO PWM
        MOVWF   CCP1CON
        RETURN

DESCONFPWM1:

        BANK0
        CLRF    CCP1CON             ; SE CLAREA  EL REGISTRO DE CONTROL DEL MODULO CCP1
        RETURN

CONFPWM2:                           ; SE CONFIGURA EL MODULO PWM2

        BANK0
        CLRF    CCPR2L              ; SE INICIALIZA LA PARTE ALTA DEL CICLO DE TRABAJO
        CLRF    CCPR2H              ; REGISTRO DE CONTROL CICLO DE TREABAJO INICIALMENTE A CERO
        BCF     CCP2CON, CCP2X      ; SE INICILIZAN LOS DOT BITS MENOS SIGNIFICATIVOS DE CICLO DE TRABAJO
        BCF    CCP2CON, CCP2Y
        MOVLW   0X0F                ; SE CONFIGURA EL MODULO CCP2, EN MODO PWM
        MOVWF   CCP2CON
        RETURN

DESCONFPWM2:

        BANK0
        CLRF    CCP2CON
        RETURN

;*****************************************************************************
;*** RUTINA DE INICIALIZACION GLOBAL DE INTERRUPCIONES (INTGLOB) ***
;*****************************************************************************


INTGLOB:

        BSF     INTCON, PEIE        ; ACTIVACION DE LA INTERRUPCION GLOBAL Y LA INTERRUPCION DE LOS
        BSF     INTCON, GIE         ; PERIFERICOS DEL REGISTRO PIE1
        RETURN
```

```
;*******************************************************************************
;*** RUTINA DE DESCONFIFGURACION GLOBAL DE INTERRUPCIONES (INTGLOBDES) ***
;*******************************************************************************

INTGLOBDES:

        BCF     INTCON, GIE     ; SE DESACTIVAN LA INTERRUPCION GLOBAL Y  LA INTERRUPCION DE
        BCF     INTCON, PEIE    ; LOS PERIFERICOS DEL REGISTRO PIE1
        RETURN

;*********************************************************************
;*** RUTINA DE TRANSMISION DE LA CABECERA PARA EL PROCESO DE ***
;*** MONITOREO DEL ESTADO DE LAS ENTRADAS Y LAS SALIDAS      ***
;*********************************************************************

TXHEAD:

        BANK0           ; COMIENZA EN BANCO 0
        MOVLW TXINI
        MOVWF TXDIRA    ; SE ACTUALIZA EL APUNTADOR DEL BUFFER DE TRANSMISION CON  LA
                        ; DIRECCION INICIAL DEL BUFFER DE TRANSMISION
        MOVWF FSR       ; DIRECCIONAMIENTO INDIRECTO PARA ACCEDER AL BUFFER DE TRANSMISION
        MOVLW TXCABECERA        ; LA CABECERA SE DE LA TRAMA SE ALMACENA LA DIRECCION INICIAL DEL
                                ; BUFFER DE TRANSMISION
        MOVWF INDF      ; PRIMER DATO
        INCF    FSR, W
        MOVWF TXDIRA    ; SE APUNTA A LA SIGUIENTE REGISTRO DEL BUFFER DE TRANSMISION
        BANK2
        MOVLW NFPE1     ; 0X155,  DIRECCION INICIAL DE LOS REGISTROS DE CONTROL QUE INDICAN EL
        MOVWF APFPE     ; NUMERO DE CONJUNTOS DIFUSOS, POR VARIABLE LINGUISTICA DE ENTRADA O
                        ; SALIDA DEL DISEÑO ACTUAL
        RETURN

;*********************************************************************
;*** RUTINAS DE VERIFICACION DEL TIEMPO DE EJECUCION DEL          ***
;*** ALGORITMO DE CONTROL DIFUSO Y LA FRECUANCIA DE MUESTREO      ***
;*** "VERUNO" PONE A UNO EL BIT 3 DEL PUERTO C                    ***
;*** "VERCERO" PONE A CERO EL BIT 3 DEL PUERTO C                  ***
;*********************************************************************

VERUNO:

        BANK0
        BSF     TIEMPO_CONTROL
        BANK2
        RETURN


VERCERO:

        BANK0
        BCF     TIEMPO_CONTROL
        BANK2
        RETURN

;*******************************************************************************
;*** RUTINA DE INICIALIZACION DEL PROCESO DE  FUZZIFICACION (INIFUZZY) ***
;*******************************************************************************

INIFUZZY:

        BANK2
        MOVF    DIVME, W        ; APUNTA AL VALOR  DE MEMBRESIA DE LA 1° ENTARADA
        MOVWF DIRVM             ; A L A  PRIMERA FUNCION DE MEMBRESIA
        MOVF    AFPEEL, W       ; APUNTA A LA DEFINICION  PRIMERA FUNCION DE MEMBRESIA
        MOVWF EEADR
        MOVF    AFPEEH, W
        MOVWF EEADRH
```

```
        BSF     STATUS, RP0     ; BANCO 3
        BSF     EECON1, EEPGD   ; SELECCION DE LA MEMORIA FLASH
        BCF     STATUS, RP0     ; BANCO 2
        RETURN

;************************************************
;*** RUTINA DEL PROCESO DE LA CAD (RCAD) ***
;************************************************

RCAD:

        BANK0                   ; BANCO 0
        MOVF    TXDIRA, W       ; APUNTAR A DIRECCION ACTUAL DEL BUFFER DE TRANSMISION
        MOVWF   FSR
        BSF     ADCON0, GO      ; INICIO DE CAD

LICA:

        BTFSC   ADCON0, 2       ; ESPERA A QUE FINALICE LA CONVERSION
        GOTO    LICA
        MOVF    ADRESH, W       ; DATO ALTO DE LA CONVERSION
        BSF     STATUS, RP1     ; BANCO 2
        MOVWF   PXH
        MOVWF   INDF            ; ENVIA EL PRIMER DATO POR EL PUERTO SERIE
        INCF    FSR, F
        BANK1                   ; BANCO 1
        MOVF    ADRESL, W       ; DATO BAJO DE LA CONVERSION
        BANK2                   ; BANCO 2
        MOVWF   PXL
        MOVWF   INDF            ; ENVIA EL SEGUNDO DATO POR EL PUERTO SERIE
        INCF    FSR, W
        BANK0                   ; BANCO 0
        MOVWF   TXDIRA
        BANK2                   ; REGRESA EN BANCO 2
        RETURN

;*****************************************************************************
;*** RUTINAS DE SELECCION DEL CANAL DE CONVERTIDOR ANALOGICO DIGITAL ***
;*****************************************************************************

CANAL1:

        BANKSEL  ADCON0         ; SELECCIONA EL BANCO DEL REGISTRO ADCON0
        BSF     ADCON0, 3       ; PONE CHS2:CHS0 (ADCON0<5:3>) = B'001'
        BCF     ADCON0, 4       ; SELECCIONA EL BANCO 1
        RETURN

CANAL2:

        BANKSEL  ADCON0         ; SELECCIONA EL BANCO DEL REGISTRO ADCON0
        BCF     ADCON0, 3       ; PONE CHS2:CHS0 (ADCON0<5:3>) = B'010'
        BSF     ADCON0, 4       ; SELECCIONA EL BANCO 2
        RETURN

CANAL0:

        BANKSEL  ADCON0         ; SELECCIONA EL BANCO DEL REGISTRO ADCON0
        BCF     ADCON0, 3 ; PONE CHS2:CHS0 (ADCON0<5:3>) = B'000'
        BCF     ADCON0, 4 ; SELECCIONA EL BANCO 0
        RETURN

;*****************************************************************************
;*** RUTINA PARA INDICAR EL NUMERO DE CONJUNTOS DIFUSOS POR ENTRADA ***
;*****************************************************************************

RNUMCDE:

        BANK2
        MOVF    APFPE, W
        MOVWF   FSR             ; APUNTA A LOS REGISTROS QUE ALMACENAN EL NUEMRO DE
```

```
        MOVF    INDF, W         ; FUNCIONES DE MEMBRESIA POR ENTRADA
        MOVWF   NUMFM           ; CONTADOR DEL NUMERO DE FUNCIONES DE MEMBRESIA POR ENTRADA
        INCF    FSR, W
        MOVWF   APFPE
        RETURN

;***********************************************
;*** RUTINA DE FUZZYFICACION (RFUZZYFI) ***
;***********************************************

RFUZZYFI:

        BANK2

BUCLE:

        CALL    GDM
        DECFSZ  NUMFM, F        ; PARA TODOS LAS FUNCIONHES DE MEMBRESIA POR  ENTRADA
        GOTO    BUCLE
        RETURN

;*************************************************************
;*** RUTINA DE INICIALIZACION DEL PROCESO                ***
;*** DE EVALUACION DE REGLAS EN MEMORIA EEPROM     ***
;*************************************************************

RINIEVAREG_EEPROM:

        BANK2
        BSF     STATUS, RP0     ; BANCO 3
        BCF     EECON1, EEPGD   ; SELECCION EEPROM
        BCF     STATUS, RP0     ; BANCO 2
        CLRF    EEADR           ; APUNTA A DIRECCION DE INICIO DE LA MEMORIA EEPROM (0X00)
        MOVF    DIVMS, W        ; APUNTA A VALORES DE MEMBRESIA DE LA SALIDA
        MOVWF   FSR

BORRAE:                         ; BORRA PESO DE LOS CONJUNTOS CRISP DE LA SALIDA

        CLRF    INDF
        INCF    FSR, F
        RLF     NFCS, W         ; NFCS * 2
        ADDWF   DIVMS, W        ; NFCS * 2 + DIVMS
        SUBWF   FSR, W
        BTFSS   STATUS, Z
        GOTO    BORRAE
        RETURN

;*************************************************************************
;*** RUTINA DE EVALUACION DE REGLAS EN MEMORIA EEPROM (REVA_REG) ***
;*************************************************************************

REVA_REG_EEPROM:

        BANK2
        EVA_REG_EEPROM
        RETURN

;*****************************************************
;** RUTINA DE INICIALIZACION DEL PROCESO          ***
;** DE EVALUACION DE REGLAS EN MEMORIA FLASH  ***
;*****************************************************

RINIEVAREG_FLASH:

        BANK2
        BSF     STATUS, RP0     ; BANCO 3
        BSF     EECON1, EEPGD   ; SELECCION FLASH
        BCF     STATUS, RP0     ; BANCO 2
        MOVF    ABREEH, W       ; APUNTA A INICIO DE REGLAS
        MOVWF   EEADRH
```

```
        MOVF    ABREEL, W
        MOVWF   EEADR
        MOVF    DIVMS, W        ; APUNTA A VALORES DE MEMBRESI DE LA SALIDA
        MOVWF   FSR

BORRA:                          ; BORRA PESO DE LOS CONJUNTOS CRISP DE LA SALIDA

        CLRF    INDF
        INCF    FSR, F
        RLF     NFCS, W         ; NFCS * 2
        ADDWF   DIVMS, W        ; DIVMS + NFCS * 2
        SUBWF   FSR, W
        BTFSS   STATUS, Z
        GOTO    BORRA
        RETURN

;*************************************************************************
;*** RUTINA DE EVALUACION DE REGLAS EN MEMORIA FLASH (REVA_REG) ***
;*************************************************************************

REVA_REG_FLASH:

        BANK2
        EVA_REG_FLASH
        RETURN

;*************************************************************************
;*** RUTINA DE INICIALIZACION DEL PROCESO DE  DEFUZZIFICACION (INIDEFUZ) ***
;*************************************************************************

INIDEFUZ:

        BANK2
        MOVF    NFCS, W         ; 5 SINGLETONS DE SALIDA
        MOVWF   NUMSINGLETON
        MOVF    DIVMS, W        ; APUNTA A VALORES DE MEMBRESIA DE LAS 5 SINGLETONS
        MOVWF   DIRVMSAL
        MOVF    AFCEEH, W       ; APUNTA A DEFINICION DE SINGLETONS DE SALIDA
        MOVWF   EEADRH
        MOVF    AFCEEL, W
        MOVWF   EEADR
        BSF     STATUS, RP0     ; BANCO 3
        BSF     EECON1, EEPGD   ; SELECCION FLASH
        BCF     STATUS, RP0     ; BANCO 2
        RETURN

;*************************************************************
;*** RUTINA DEL PROCESO DE DEFUZZYFICACION (RDEFUZZY) ***
;*************************************************************

RDEFUZZY:

        BANK2
        DEFUZZY
        RETURN

;*************************************************************************
;*** RUTINA DE  ACTUALIZACION DEL ANGULO DE DISPARO DEL TRIAC 1 (ACTTMR1) ***
;*************************************************************************

ACTTMR1:

        BANK2                   ; INICIA EN EL BANCO 2
        MOVF    SALIDAL, W
        MOVWF   OPAL1
        MOVF    SALIDAH, W
        MOVWF   OPAH1
        MOVLW   0X05
        MOVWF   OPAL2
        CALL    MULFACT
```

```
        BANK0                           ; BANCO 0
        MOVF    OPAH3, W
        SUBLW   0X2A
        MOVWF   DELAYL ; ACTUALIZA RETARDO PARTE BAJA
        BTFSS   STATUS, C
        INCF    OPAL4, F
        MOVF    OPAL4, W
        SUBLW   0X14
        MOVWF   DELAYH ; ACTUALIZA RETARDO PARTE ALTA
        BANK2                           ; FINALIZA EN EL BANCO 2
        RETURN

;*******************************************************************************
;*** RUTINA DE  ACTUALIZACION DEL ANGULO DE DISPARO DEL TRIAC 2 (ACTTMR2) ***
;*******************************************************************************

ACTTMR2:

        BANK2
        BCF     STATUS, C ; DIVIDE LA SLAIDA ENTRE CUATRO PARA AJUSTARLA A LOS 8 BITS DEL TIMER 2
        RRF     SALIDAH, F
        RRF     SALIDAL, F
        RRF     SALIDAH, F
        RRF     SALIDAL, W
        BANK0
        MOVWF   DELAY_TMR2      ; ACTUALIZA CICLO DE TRABAJO DEL TRIAC 2
        RETURN


;*******************************************************************************
;** RUTINAS DE ACTUALIZACION DEL CICLO DE TRABAJO DE LOS  MODULOS PWM  ***
;*******************************************************************************

;*************************************************************************
;*** RUTINA PARA ACTUALIZAR EL CICLO DE TRABAJO  DEL MODULO PWM1   ***
;*** CON RESOLOCION DE 10 BITS                                     ***
;*************************************************************************

ACTPWM1:

        BANK2
        RLF     SALIDAL, F
        RLF     SALIDAH, F
        RLF     SALIDAL, F
        RLF     SALIDAH, F
        RLF     SALIDAL, F
        RLF     SALIDAH, F
        RLF     SALIDAL, F
        RLF     SALIDAH, F
        MOVLW   0X30
        ANDWF   SALIDAL, W
        BCF     STATUS, RP1         ; BANCO 0
        BCF     CCP1CON, 4          ; SE CLAREA EL BIT MENOS SIGNIFICATIVO  DEL CICLO DE TRABAJO
        BCF     CCP1CON, 5          ; SE CLAREA EL BIT 1 DEL CICLO DE TRABAJO
        IORWF   CCP1CON, F          ; SE ACTUALIZAN LOS DOS BITS DE MENOS PESO DEL CICLO DE TRABAJO
        BSF     STATUS, RP1         ; BANCO 2
        RLF     SALIDAL, F
        RLF     SALIDAH, F
        RLF     SALIDAL, F
        RLF     SALIDAH, W
        BCF     STATUS, RP1         ; BANCO 0
        MOVWF   CCPR1L
        RETURN

;*************************************************************************
;** RUTINA PARA ACTUALIZAR EL CICLO TRABAJ DEL MODULO PWM2  ***
;** CON RESOLUCION DE 10 BITS                                ***
;*************************************************************************

ACTPWM2:
```

```
        BANK2
        RLF     SALIDAL, F
        RLF     SALIDAH, F
        RLF     SALIDAL, F
        RLF     SALIDAH, F
        RLF     SALIDAL, F
        RLF     SALIDAH, F
        RLF     SALIDAL, F
        RLF     SALIDAH, F
        MOVLW   0X30
        ANDWF   SALIDAL, W
        BCF     STATUS, RP1         ; BANCO 0
        BCF     CCP2CON, 4          ; SE CLAREA EL BIT MENOS SIGNIFICATIVO  DEL CICLO DE TRABAJO
        BCF     CCP2CON, 5          ; SE CLAREA EL BIT 1 DEL CICLO DE TRABAJO
        IORWF   CCP2CON, F          ; SE ACTUALIZAN LOS DOS BITS DE MENOS PESO DEL CICLO DE TRABAJO
        BSF     STATUS, RP1         ; BANCO 2
        RLF     SALIDAL, F
        RLF     SALIDAH, F
        RLF     SALIDAL, F
        RLF     SALIDAH, W
        BCF     STATUS, RP1         ; BANCO 0
        MOVWF   CCPR2L              ; SE ACTUALIZAN LOS OCHO BITS DE MAS PESO SEL CICLO DE TRABAJO
        RETURN

;*****************************************************************************************
;*** RUTINAS PARA GENERAR LA SEÑEL DE ERROR Y PARA GENERAR LA SEÑAL DE          **
;*** CAMBIO DE ERROR      (SERROR: Y SCERROR)                                  **
;*****************************************************************************************


;************************************************
;*** RUTINA PARA GENERAR LA SEÑAL DE ERROR ***
;************************************************

SERROR:

;*****************************************************************************************
;*** RESTA EL VALOR CATUAL DE LA CONVERSION ANALOGICA PX, DIGITAL DE EL SETPOINT Y ***
;*** Y LO DEJA EN PX OTRA VEZ                                                    ***
;*****************************************************************************************

        BANK0
        MOVF    TXDIRA, W           ; TRAE DIRECCION ACTUAL DEL FUFFER DE TRANSMISION
        MOVWF   FSR
        BANK2
        MOVF    PXL, W              ; RESTA DEL SET POINT Y EL VALOR ACTUAL DE LA VARIBLE DE ENTRADA

        SUBWF   SETPL, W            ; SE RESTAN LOS BYTES BAJOS
        MOVWF   PXL                 ; DEJA EL RESULTADO EN PXL
        BTFSS   STATUS, C
        INCF    PXH, F
        MOVF    PXH, W
        SUBWF   SETPH, W            ; SE RESTAN LAS PARTES ALTAS
        MOVWF   PXH                 ; DEJA EL RESULTADO EN PXH

;************************************************
;*** SE ENVIAN LOS DATOS AL BUFFER DE TRANSMISION ***
;************************************************

        MOVWF   INDF                ; ENVIA EL PRIMER DATO POR  AL BUFFER DE TRANSMISION
        INCF    FSR, F              ; SE APUNTA A LA SIGUEINTE DIRECCION DEL BUFFER DE TRANSMISION
        MOVF    PXL, W
        MOVWF   INDF                ; ENVIA EL SEGUNDO DATO AL BUFFER DE TRANSMISION
        INCF    FSR, W              ; SE APUNTA A LA SUGIENTE DIRECCION DE BUFFER DE TRANSMISON
        BANK0
        MOVWF   TXDIRA              ; SE GUARDA LA DIRRECCION DEL BUFFER DE TRANSMISION
        RETURN                      ; REGERSA DE LA SUBRRUTINA
```

```
;*************************************************************
;*** RUTINA PARA GENERAR LA SEÑAL DE CAMBIO DE ERROR   ***
;*************************************************************

SCERROR:

        ;*** MUEVE EL LA VARIABLE ERROR ACTUAL A LA VARIABLE DE ERROR ANTERIOR  ***

        BANK0
        MOVF    TXDIRA, W       ; TRAE    DIRECCION ACTUAL DEL BUFFER DE TRANSMISION
        MOVWF FSR
        BANK2
        MOVF    ER1L, W
        MOVWF ER0L
        MOVF    ER1H, W
        MOVWF ER0H

        ;*** SE ALMACENA EL VALOR ACTUAL DE PX EN LA VARIABLE ERROR ACTUAL  ***

        MOVF    PXH, W
        MOVWF ER1H
        MOVF    PXL, W
        MOVWF ER1L

        ;** * SUMA 0X1FF AL VALOR DE PX ***

        MOVLW 0XFF
        ADDWF   PXL, F
        BTFSC   STATUS, C
        INCF    PXH, F
        MOVLW 0X01
        ADDWF   PXH, F

        ;*************************************************************************
        ;*** RESTA EL VALOR DE LA VARIABLE ERROR ANTERIOR DE PX + 0X1FF QUE     ***
        ;*** CONTIENE EL VALOR ACTUAL DEL ERROR  MAS 511, EL VALOR RESULTANTE  ***
        ;*** SE ALMACENA EN PX Y ES LA SEÑAL DE CAMBIO DE ERROR                ***
        ;*************************************************************************

        MOVF    ER0L, W
        SUBWF   PXL, F
        BTFSS   STATUS, C
        GOTO    SISUMCEA
        MOVF    ER0H, W

SISUMCEB:

        SUBWF   PXH, F

        ;** SE ENVIAN LOS DATOS AL BUFFER DE TRANSMISION  ***

        MOVF    PXH, W
        MOVWF INDF              ; ENVIA PRIMER DATO AL BUFFER DE TRANSMISION
        INCF    FSR, F          ; APUNTA A LA SIGUIENTE DIRECCION DEL BUFFER DE TRANSMISION
        MOVF    PXL, W
        MOVWF INDF              ; ENVIA EL SIGIENTE DATO AL BUFFER DE TRANSMISION
        INCF    FSR, W          ; APUNTA A LA SIGUIENTE DIRECCION DEL BUFFER DE TRASMISION
        BANK0
        MOVWF TXDIRA            ; SE ALMACENA LA DIRECCION DEL BUFFER DE TRANSMISION
        RETURN                  ; REGERSA DE LA SUBRRUTINA

SISUMCEA:

        INCF    ER0H, W
        GOTO    SISUMCEB

;*****************************************************************************************
;***  RUTINA DE GENERACION DE LA SALIDA CON VALORES NEGATIVOS, CERO Y POSITIVOS (SNZP) ****
;*****************************************************************************************
```

```
SNZP:
                                    ; SE RESTA SALIDA DE 0X1FF, SI EL RESULTADO ES NEGATIVO SE VA
SPOSITIVA
        BANK2                       ; SI EL RESULTADO ES POSITIVO SE VA A SNEGATIVA
        MOVF    SALIDAL, W          ; MUEVE SALIDAL A WORKING REG
        SUBLW   0XFF                ; RESTA WORKING DE 0XFF
        BTFSC   STATUS, C           ; ¿EL RESULTADO ES NEGATIVO?
        GOTO    NOADD               ; NO, VETE A NO SUMA
        INCF    SALIDAH, W          ; SI INCREMENTALE 1 SALIDAH Y DEJALO EN WORKING

NOADD1:

        SUBLW   0X01                ; RESTA WORKING DE 0X01
        BTFSS   STATUS, C           ; ES NEGATIVO
        GOTO    SPOSITIVA           ; SI VETE A CONTROLAR SALIDA POSITIVA
        GOTO    SNEGATIVA           ; SI VETE A CONTROLAR SALIDA NEGATIVA

NOADD:

        MOVF    SALIDAH,W           ;MUEVE SALIDAH A WORKING REG
        GOTO    NOADD1


                                    ; DE ANTEMANO SE SABE QUE SALIDA ES MAYOR DE 0X1FF
SPOSITIVA:                          ; RESTA 0X1FF DE SALIDA

        MOVLW   0XFF                ; MUEVE EL DATO INMEDIATO 0XFF A WORKING REG
        SUBWF   SALIDAL, F          ; RESTA WORKING DE SALIDAL Y DEJA EL RESULTADO EN SALIDAL
        BTFSS   STATUS, C           ; ¿ES NEGATIVO?
        DECF    SALIDAH, F          ; SI DECREMENTALE 1 A SALIDAH
        MOVLW   0X01                ; MEVE EL DATO  INMEDIATO A WORKING REG
        SUBWF   SALIDAH, F          ; RESTA WORKING DE SALIDAH, Y DEJA EL RESULTADO EN SALIDAH
        BCF     STATUS, C           ; CLAREA LA BANDERA ACARREO
        RLF     SALIDAL, F          ; MULTIPLICA POR DOS LA SALIDA
        RLF     SALIDAH, F
        BTFSS   SALIDAH, 2
        GOTO    NORMAL
        MOVLW   0XFF
        MOVWF   SALIDAL
        MOVLW   0X03
        MOVWF   SALIDAH

NORMAL:

        BTFSS   LOADP               ;*** DETERMINAR EL TIPO DE CARGA (CD O CA) ****
        GOTO    LOADDC1
        GOTO    LOADAC1

LOADDC1:

        CALL    ACTPWM1             ; LLAMA A RUTINA DE ACTUALIZACION DEL PWM1
        CALL    TRIAC1_OFF          ; EL TRIAC 1 NO SE DEBE DISPARAR DURANTE EN ESTA SITUACION
        CALL    PWM2_OFF            ; EL PWM2 DEBE TENER UN CICLO DE TRABAJO DE CERO DURANTE
        RETURN                      ; ESTA SITUACION

LOADAC1:

        CALL    ACTTMR1             ; LLAMA A RUTINA DE ACTUALIZACION DEL TIMER1
        CALL    TRIAC2_OFF          ; EL TRIAC 2 NO SE DEBE DISPARAR DURANTE EN ESTA SITUACION
        CALL    PWM2_OFF            ; EL PWM2 DEBE TENER UN CICLO DE TRABAJO DE CERO
        RETURN                      ; DURANTE  ESTA SITUACION

SNEGATIVA:                          ; DE ANTEMANO SE SABE QUE 0X1FF ES MAYOR QUE SALIDA
                                    ; RESTA SALIDA DE 0X1FF
        MOVF    SALIDAL, W          ; MUEVE SALIDAL A WORKING REG
        SUBLW   0XFF                ; RESTA WORKING  DE 0XFF
        MOVWF   SALIDAL             ; MUEVE WORKING A SALIDAL
        BTFSS   STATUS,C            ; ¿ES NEGATIVO?
        INCF    SALIDAH,F           ; SI INCREMETALE 1 A SALIDAH
```

```
        MOVF    SALIDAH,W       ; NO MUEVE SALIDAH A WORKING
        SUBLW   0X01            ; RESTA WORKING DE 0X01
        MOVWF   SALIDAH         ; MUEVE WORKING REG A SALIDAH
        BCF     STATUS, C       ; CLAREA LA BADERA DE ACARREO
        RLF     SALIDAL, F      ; MULTIPLICA POR 2 SALIDA (SALIDA*2)
        RLF     SALIDAH, F

        BTFSS   LOADN           ; SE DETERMINA EL TIPO DE CARGA (DC O AC)
        GOTO    LOADDC2
        GOTO    LOADAC2

LOADDC2:

        CALL    ACTPWM2         ; LLAMA A RUTINA DE ACTUALIZACION DEL PWM2
        CALL    TRIAC1_OFF      ; EL TRIAC1 NO SE DEBE DISPARAR DURANTE EN ESTA SITUACION
        CALL    PWM1_OFF        ; EL PWM1 DEBE TENER UN CICLO DE TRABAJO DE CERO
        RETURN                  ; DURANTE ESTA SITUACION

LOADAC2:

        BTFSS   MODAB
        GOTO    TRIAC2          ; SI ES MODOA UTILIZA EL TIMER2
        GOTO    TRIAC1          ; SI ES MODOB UTILIZA EL TIMER1

TRIAC2:

        CALL    ACTTMR2
        CALL    TRIAC1_OFF      ; EL TRIAC1 NO SE DEBE DISPARAR DURANTE EN ESTA SITUACION
        RETURN

TRIAC1:

        CALL    ACTTMR1         ; RUTINA DE ACTUALIZACION DEL ANGULO DE DISPARO DEL TRIAC
        CALL    PWM1_OFF        ; EL PWM1 DEBE TENER UN CICLO DE TRABAJO DE CERO
        RETURN                  ; DURANTE ESTA SITUACION

;*************************************************
;*** RUTINAS DE CONTROL DE LA SALIDA SNZP ****
;*************************************************

TRIAC2_OFF:                     ; TRIAC 2 APAGADO

        BANK0
        CLRF    DELAY_TMR2      ; CICLO DE TRABAJO DEL TRIAC 2 EN CERO
        BCF     PULSO_TRIAC2    ; TRIAC2 APAGADO
        RETURN

TRIAC1_OFF:                     ; TRIAC 1 APAGADO

        BANK0
        MOVLW   0X2A
        MOVWF   DELAYL          ; CICLO DE TRABAJO DEL TRIAC 1 EN CERO
        MOVLW   0X14
        MOVWF   DELAYH
        BCF     PULSO_TRIAC1    ; TRIAC1 APAGADO
        RETURN

PWM1_OFF:                       ; SALIDA PWM1 APAGADA

        BANK0
        CLRF    CCPR1L          ; SE LIMPIAN LOS DOS REGISTROS DE CONTROL DEL MODULO PWM1
        CLRF            CCPR1H
        BCF     CCP1CON, CCP1X  ; SE LIMPIAN LOS DOT BITS MENOS SIGNIFICATIVOS DE CICLO DE TRABAJO
        BCF     CCP1CON, CCP1Y
        BCF     PORTC, 2        ; EL PULSO PERMANECE APAGADO
        RETURN

PWM2_OFF:                       ; SALIDA PWM2 APAGADA
```

```
        BANK0
        CLRF    CCPR2L          ; SE LIMPIAN LOS DOS REGISTROS DE CONTROL DEL MODULO PWM1
        CLRF    CCPR2H
        BCF     CCP2CON, CCP2X  ; SE INICILIZAN LOS DOT BITS MENOS SIGNIFICATIVOS DE CICLO DE TRABAJO
        BCF     CCP2CON, CCP2Y
        BCF     PORTC, 1        ; EL PULSO PERMANECE APAGADO
        RETURN


;*****************************************************************************
;*** RUTINA PARA LA TRANSMISION DEL BUFFER QUE CONTIENE EL ESTADO    ***
;*** ACTUAL DE LAS VARIABLES DEL PROCESO DE CONTROL DIFUSO           ***
;*****************************************************************************

TRANSMITIR_BUFFER:

        BANK0
        MOVLW   TXINI
        MOVWF   FSR

TXOTRO:

        MOVF    INDF, W                 ; ENVIA DATO
        MOVWF   DATOTX
        CALL    TRANSMITE
        INCF    FSR, F                  ; SALVA DIRECCION DEL PROXIMI DATO A ENVIAR
        BSF             STATUS, RP1     ; BANCO 2
        MOVF    TXDAT, W
        BCF     STATUS, RP1             ; BANCO 0
        ADDLW   TXINI                   ; TXINI + TXDAT
        SUBWF   FSR, W
        BTFSS   STATUS, Z               ; SE  HAN ENVIADO TODOS LOS DATOS
        GOTO    TXOTRO                  ; NO: ENVIA OTRO DATO
        RETURN                          ; SI: RETORNA DE RUTINA

;*****************************************************************************************************
;***     RUTINA PARA GENERAR LA SEÑAL DE CONTROL APARTIR DE LA SEÑAL DE "CAMBIO DE CONTROL"  ***
;*****************************************************************************************************

CCONTROL:
                        ; SE RESTA SALIDA DE 0X1FF, SI EL RESULTADO ES NEGATIVO SE VA SPOSITIVA
        BANK2                           ; SI EL RESULTADO ES POSITIVO SE VA A SNEGATIVA
        MOVF    SALIDAL, W              ; MUEVE SALIDAL A WORKING REG
        SUBLW   0XFF                    ; RESTA WORKING DE 0XFF
        BTFSC   STATUS, C               ; ¿EL RESULTADO ES NEGATIVO     ?
        GOTO    NONEGATIVO              ; NO, VETE A NO SUMA
        INCF    SALIDAH, W              ; SI INCREMENTALE 1 SALIDAH Y DEJALO EN WORKING

NONEGATIVO1:

        SUBLW   0X01                    ; RESTA WORKING DE 0X01
        BTFSS   STATUS, C               ; ¿ES NEGATIVO?
        GOTO    CCONTROLP               ; SI: RETORNA CERO PARA INDICAR RESULTADO NEGATIVO
        GOTO    CCONTROLN               ; NO: RETORNA UNO PARA INDICRA RESULTADO POSITIVO

NONEGATIVO:

        MOVF    SALIDAH, W              ; MUEVE SALIDAH A WORKING REG
        GOTO    NONEGATIVO1

CCONTROLP:
                                        ; DE ANTEMANO SE SABE QUE SALIDA ES MAYOR DE 0X1FF
        MOVLW   0XFF                    ; RESTA 0X1FF DE SALIDA
        SUBWF   SALIDAL, W
        MOVWF   CCONTROL_PL
        BTFSC   STATUS, C
        GOTO    NODECRE
        DECF    SALIDAH, F

NODECRE
```

```
        MOVLW 0X01
        SUBWF SALIDAH, W
        MOVWF CCONTROL_PH

        BCF     STATUS, C               ; CLAREA LA BANDERA ACARREO
        RLF     CCONTROL_PL, F          ; MULTIPLICA POR DOS LA SALIDA
        RLF     CCONTROL_PH, F
        BTFSS   CCONTROL_PH, 2
        GOTO    VALIDO
        MOVLW 0XFF
        MOVWF CCONTROL_PL
        MOVLW 0X03
        MOVWF CCONTROL_PH

VALIDO:

        MOVF    CCONTROL_PL, W ; ACTUALIZA LA SEÑAL DE CONTROL, CONTRO = CONTROL + CCONTROLP
        ADDWF CONTROLL, F
        BTFSC   STATUS, C
        INCF    CONTROLH, F
        MOVF    CCONTROL_PH, W
        ADDWF CONTROLH, F
        BTFSS   CONTROLH, 2             ; SI LA SALIDA DE CONTROL ES MAYOR DE 0X3FF
        GOTO    ACTUALIZAR              ; NO: EL VALOR ES VALIDO
        MOVLW 0X03                      ; SI: EL VALOR NO ES VALIDO PONERLO A 0X3FF
        MOVWF CONTROLH
        MOVLW 0XFF
        MOVWF CONTROLL
        GOTO    ACTUALIZAR

CCONTROLN:                              ; DE ANTEMANO SE SABE QUE 0X1FF ES MAYOR QUE SALIDA
                                        ; RESTA SALIDA DE 0X1FF
        MOVF    SALIDAL, W              ; MUEVE SALIDAL A WORKING REG
        SUBLW 0XFF                      ; RESTA WORKING  DE 0XFF
        MOVWF CCONTROL_NL               ; MUEVE WORKING A SALIDAL
        MOVF    SALIDAH, W
        SUBLW 0X01

        MOVWF CCONTROL_NH               ; MUEVE WORKING REG A SALIDAH
        BCF     STATUS, C       ; CLAREA LA BADERA DE ACARREO
        RLF     CCONTROL_NL, F  ; MULTIPLICA POR 2 SALIDA (SALIDA*2)
        RLF     CCONTROL_NH, F

        MOVF    CCONTROL_NL, W ; ACTUALIZA LA SEÑAL DE CONTROL, CONTROL = CONTROL - CCONTROLN
        SUBWF CONTROLL, F
        BTFSC   STATUS, C
        GOTO    NONEG
        MOVLW 0X01
        SUBWF CONTROLH, F
        BTFSC   STATUS, C               ; ¿VALOR DE CONTROL NEGATIVO?
        GOTO    NONEG                   ; NO: VERIFICA LA PARTE ALTA
        CLRF    CONTROLL                ; SI: VALOR NO VALIDO
        CLRF    CONTROLH
        GOTO    ACTUALIZAR

NONEG:

        MOVF    CCONTROL_NH, W
        SUBWF CONTROLH, F
        BTFSC   STATUS, C               ; ¿VALOR DE CONTROL NEGATIVO?
        GOTO    ACTUALIZAR              ; NO: VALOR VALIDO
        CLRF    CONTROLL                ; SI: VALOR NO VALIDO
        CLRF    CONTROLH
        GOTO    ACTUALIZAR

ACTUALIZAR:

        MOVF    CONTROLL, W
        MOVWF SALIDAL
```

```
        MOVF    CONTROLH, W
        MOVWF   SALIDAH

        BTFSS   LOADP                   ; DETERMINAR EL TIPO DE CARGA (CD O CA)( 1:AC, 0: DC )
        GOTO    LOAD_DC
        GOTO    LOAD_AC

LOAD_DC:                                ; CARGA DE DC

        CALL    ACTPWM1                 ; LLAMA A RUTINA DE ACTUALIZACION DEL PWM1
        RETURN

LOAD_AC:                                ; CARGA DE AC

        CALL    ACTTMR1                 ; LLAMA A RUTINA DE ACTUALIZACION DEL TIMER1
        RETURN

;*****************************************************************************************
;*** ESTA RUTINA GENERA LA SEÑAL DE ENTRADA AL CONTROLADOR DIFUSOS TIPO SUMA DE ERROR  ***
;*****************************************************************************************

SSERROR:

        BANK0
        MOVF    TXDIRA, W       ; TRAE    DIRECCION ACTUAL DEL BUFFER DE TRANSMISION
  MOVWF         FSR

                                ; SE RESTA PX DE 0X1FF, SI EL RESULTADO ES NEGATIVO SE VA SPOSITIVA
        BANK2                   ; SI EL RESULTADO ES POSITIVO SE VA A SNEGATIVA
        MOVF    PXL, W          ; MUEVE SALIDAL A WORKING REG
        SUBLW   0XFF            ; RESTA WORKING DE 0XFF
        BTFSC   STATUS, C       ; ¿EL RESULTADO ES NEGATIVO      ?
        GOTO    NOSUMA2         ; NO, VETE A NO SUMA
        INCF    PXH, W          ; SI INCREMENTALE 1 SALIDAH Y DEJALO EN WORKING

NOSUMA12:

        SUBLW   0X01            ; RESTA WORKING DE 0X01
        BTFSS   STATUS, C       ; ¿ES NEGATIVO?
        GOTO    ERRORP          ;  SI: RETORNA CERO PARA INDICAR RESULTADO NEGATIVO
        GOTO    ERRORN          ;  NO: RETORNA UNO PARA INDICRA RESULTADO POSITIVO

NOSUMA2:

        MOVF    PXH, W          ; MUEVE SALIDAH A WORKING REG
        GOTO    NOSUMA12

ERRORP:
                                ; DE ANTEMANO SE SABE QUE PX ES MAYOR DE 0X1FF
        MOVLW   0XFF            ; RESTA 0X1FF DE SALIDA
        SUBWF   PXL, F
        BTFSC   STATUS,C
        GOTO    NODECRE2
        DECF    PXH, F

NODECRE2

        MOVLW   0X01
        SUBWF   PXH, F


        MOVF    PXL, W          ; ACTUALIZA LA SEÑAL DE CONTROL, CONTRO = CONTROL + CCONTROLP
        ADDWF   SSERRORL, F
        BTFSC   STATUS, C
        INCF    SSERRORH, F
        MOVF    PXH, W
        ADDWF   SSERRORH, F
        BTFSS   SSERRORH, 2     ; SI LA SALIDA DE CONTROL ES MAYOR DE OX3FF
        GOTO    ACTUALIZAR2     ; NO: EL VALOR ES VALIDO
        MOVLW   0X03            ; SI: EL VALOR NO ES VALIDO PONERLO A 0X3FF
```

```
        MOVWF   SSERRORH
        MOVLW   0XFF
        MOVWF   SSERRORL
        GOTO    ACTUALIZAR2

ERRORN:                         ; DE ANTEMANO SE SABE QUE 0X1FF ES MAYOR QUE SALIDA
                                ; RESTA SALIDA DE 0X1FF
        MOVF    PXL, W          ; MUEVE SALIDAL A WORKING REG
        SUBLW   0XFF            ; RESTA WORKING  DE 0XFF
        MOVWF   PXL             ; MUEVE WORKING A SALIDAL
        MOVF    PXH, W
        SUBLW   0X01

        MOVWF   PXH             ; MUEVE WORKING REG A SALIDAH
        BCF     STATUS, C       ; CLAREA LA BADERA DE ACARREO

        MOVF    PXL, W          ; ACTUALIZA LA SEÑAL DE CONTROL, CONTROL = CONTROL - CCONTROLN
        SUBWF   SSERRORL, F
        BTFSC   STATUS, C
        GOTO    NONEG2
        MOVLW   0X01
        SUBWF   SSERRORH, F
        BTFSC   STATUS, C       ; ¿VALOR DE CONTROL NEGATIVO?
        GOTO    NONEG2          ; NO: VERIFICA LA PARTE ALTA
        CLRF    SSERRORL        ; SI: VALOR NO VALIDO
        CLRF    SSERRORH
        GOTO    ACTUALIZAR2

NONEG2:

        MOVF    PXH, W
        SUBWF   SSERRORH, F
        BTFSC   STATUS, C       ; ¿VALOR DE CONTROL NEGATIVO?
        GOTO    ACTUALIZAR      ; NO: VALOR VALIDO
        CLRF    SSERRORL        ; SI: VALOR NO VALIDO
        CLRF    SSERRORH
        GOTO    ACTUALIZAR


ACTUALIZAR2:

        MOVF    SSERRORH, W
        MOVWF   PXH
        MOVWF   INDF
        INCF    FSR, F          ; SE COPIA LA PARTE ALTA AL BUFFER DE TRANSMISION
        MOVF    SSERRORL, W
        MOVWF   PXL
        MOVWF   INDF            ; SE COPIA LA PARTE  BAJA AL BUFFER DE TRANSMISION
        INCF    FSR, W
        BANK0
        MOVWF   TXDIRA
        RETURN

;*****************************************************************************************
;*** ESTA RUTINA GENERA LA SEÑAL DE ENTRADA AL CONTROLADOR DIFUSO COMO            ****
;*** PROPORCIONAL  PREALIMENTADO                                                  ****
;*****************************************************************************************

REFERENCIAP:

        BANK0
        MOVF    TXDIRA, W       ; TRAE DIRECCION ACTUAL DEL FUFFER DE TRANSMISION
        MOVWF   FSR
        BANK2
        CLRF    PXL
        CLRF    PXH
        MOVLW   0XFF            ; RESTA 0X1FF DEL  VALOR ACTUAL DE SETP
        SUBWF   SETPL, W        ; SE RESTAN LOS BYTES BAJOS
        MOVWF   PXL             ; DEJA EL RESULTADO EN PXL
```

```
        BTFSS   STATUS, C
        GOTO    NEGATIVOP
        MOVLW   0X01
        SUBWF   SETPH, W        ; SE RESTAN LAS PARTES ALTAS

GPXH

        MOVWF   PXH             ; DEJA EL RESULTADO EN PXH

        ;*********************************************************
        ;*** SE ENVIAN LOS DATOS AL BUFFER DE TRANSMISION  ***
        ;*********************************************************

        MOVWF   INDF            ; ENVIA PARTE ALTA A EL BUFFER DE TRANSMISION
        INCF    FSR, F          ; SE APUNTA A LA SIGUEINTE DIRECCION DEL BUFFER DE TRANSMISION
        MOVF    PXL, W
        MOVWF   INDF            ; ENVIA PARTE BAJA AL BUFFER DE TRANSMISION
        INCF    FSR, W          ; SE APUNTA A LA SUGIENTE DIRECCION DE BUFFER DE TRANSMISON
        BANK0
        MOVWF   TXDIRA          ; SE GUARDA LA DIRRECCION DEL BUFFER DE TRANSMISION
        RETURN                  ; REGERSA DE LA SUBRRUTINA

NEGATIVOP

        MOVLW   0X02
        SUBWF   SETPH, W        ; SE RESTAN LAS PARTES ALTAS
        GOTO    GPXH


;*************************************************************************
;***** RUTINA DE LIMPIEZA DE LA SALIDAS CUANDO SE TERMINA DE EJECUTAR    ***
;***** EL ALGORITMO DE CONTROL DIFUSO                                    ***
;*************************************************************************

LIMPIA_SALIDAS:

        BANK0
        CLRF    PORTA   ; LIMPIA PUERTA A
        CLRF    PORTB   ; LIMPIA PUERTA B
        CLRF    PORTC   ; LIMPIA PUERTA C
        CLRF    PORTD   ; LIMPIA PUERTA D
        CLRF    PORTE   ; LIMPIA PUERTA E
        RETURN

;*************************************************************
;*** RUTINA DE CONTROL LECTURA DE LA MEMORIA FLASH ***
;*************************************************************

LEER_FLASH:

        BANK3
        BSF     EECON1, EEPGD   ; SELECCIONA MEMORIA FLASH
        BSF     EECON1, RD      ; COMADO DE LECTURA
        NOP                     ; TRES CICLOS PARA ACCEDER AL DATO LEIDO
        NOP
        NOP                     ; EL DATO LEIDO SE ENCUANTRA EN EEDATH Y EEDATA
        RETURN

;*************************************************************
;*** RUTINA DE CONTROL DE ESCRITURA DE LA MEMORIA FLASH  **
;*************************************************************

ESCRIBIR_FLASH:

        BANK3
        BSF     EECON1, EEPGD   ; SELECCIONA MEMORIA FLASH
        BSF     EECON1, WREN    ; SE  HABILITA LA ESCRITURA
        MOVLW   0X55            ; ACCION DE SEGURIDAD
        MOVWF   EECON2
        MOVLW   0XAA
```

```
        MOVWF   EECON2
        BSF     EECON1, WR          ; SE DA COMANDO DE ESXCRITURA

LICAF:

        BTFSC   EECON1, WR          ; PERA A QUE FINALICE LA ESCRITURA
        GOTO    LICAF
        BCF     EECON1, WREN        ; DESHABILITA LA ESCRITURA
        RETURN

;*****************************************************************
;*** RUTINA DE CONTROL DE LECTURA DE  LA MEMORIA EEPROM ***
;*****************************************************************

LEER_EEPROM:

        BANK3
        BCF     EECON1, EEPGD       ; SELECCIONAR  MEMORIA EEPROM
        BSF     EECON1, RD          ; DAR COMANDO DE LECTURA
        RETURN

;*****************************************************************
;*** RUTINA DE CONTROL  DE ESCRITURA  EN LA MEMORIA EEPROM ***
;*****************************************************************

ESCRIBIR_EEPROM:

        BANK3
        BCF     EECON1, EEPGD       ; SELECCIONAR MEMORIA EEPROM
        BSF     EECON1, WREN        ; PERMITIR ESCITURA
        MOVLW   0X55                ; ENVIAR CABECERAS DE ESCRITURA
        MOVWF   EECON2              ; RECOMENDADO POR EL FABRICANTE
        MOVLW   0XAA
        MOVWF   EECON2
        BSF     EECON1, WR          ; DAR LA ORDEN DE ESCRITURA

LICAEE:

        BTFSC   EECON1, WR          ; ESPERAR  A QUE TERMINE DE ESCRIBIR
        GOTO    LICAEE
        BCF     EECON1, WREN        ; DESACTIVAR LA ESCRITURA
        RETURN

;*************************************************************************
;*** RUTINA DE RECEPCION DEL NUMERO DE DATOS QUE SE VAN A ESCRIBIR   ***
;*** O LEER DE LA MEMORIA EEPROM O DE LA MEMORIA FLASH               ***
;*************************************************************************

RECEPCION_NUMERO_DATOS:

        CALL    RECIBE              ; RECIBE LA PARTE ALTA DEL NUMERO DE LOCALIDADES A ESCRIBIR
        MOVF    DATORX, W
        BANK2
        MOVWF   NUMEROH
        CALL    RECIBE              ; RECIBE LA PARATE BAJA DEL NUMERO DE LOCALIDADES A ESCRIBIR
        MOVF    DATORX, W
        BANK2
        MOVWF   NUMEROL
        RETURN

;*********************************************************************************************************
;** RUTINA DE RECEPCION DE LA DIRECCION Y EL DATO DE ESCRITURA PARA LA MEMORIA FLASH O EEEPROM ***
;*********************************************************************************************************

RECEPCION_DIR_DATO:

        CALL    RECIBE              ; RECIBE LA PARTE ALTA DE LA DIRECCION DE ESCRITURA
        MOVF    DATORX, W
        BANK2
        MOVWF   EEADRH
```

```
        CALL    RECIBE          ; RECIBE LA PARTE BAJA DE LA DIRECCION DE ESCRITURA
        MOVF    DATORX, W
        BANK2
        MOVWF   EEADR
        CALL    RECIBE          ; RECIBE LA PARTE ALTA DEL DATO QUE SE VA A ESCRIBIR
        MOVF    DATORX, W

        BANK2
        MOVWF   EEDATH
        CALL    RECIBE          ; RECIBE LA PARTE BAJA DATO QUE SE VA A ESCRIBIR
        MOVF    DATORX, W
        BANK2
        MOVWF   EEDATA
        RETURN

;*********************************************************
;*** RUTINA DE ESCRITURA DE DATOS EN MEMORIA FLASH  ***
;*********************************************************

ESCRIBIR_DATOS_FLASH:

WROTROF:

        CALL   RECEPCION_DIR_DATO       ; ADQUIERE LA DIRECCION Y DATO DE ESCRITUA
                                        ; DE LA MEMORIA FLASH
        CALL    ESCRIBIR_FLASH  ; ESCIBE EL EEDATH Y EEDATL EN LA DIRECCION
                                ; ACTUAL DE EEADR Y EEADRH
        BANK0
        MOVLW   0X41            ; TRANSMITE CABECERA DE FIN DE ESCRITURA
        MOVWF   DATOTX
        CALL    TRANSMITE
        BANK2
        MOVLW   0X01
        SUBWF   NUMEROL, F      ; DECREMENTAR UNA UNIDAD LA PARTE BAJA DEL NUMERO DE DATOS
        BTFSS   STATUS, C       ; ¿ES NEGATIVO?
        DECF    NUMEROH, F      ; SI: DECREMENTA UNA UNIDAD A LA PARTE ALTA DEL NUEMRO DE DATOS
        MOVF    NUMEROH, F
        BTFSS   STATUS, Z       ; ¿ES CERO LA PARTE ALTA DEL NUEMRO DE DATOS?
        GOTO    WROTROF         ; NO: FALTAN DATOS POR RECIBIR Y ESCRIBIR
        MOVF    NUMEROL, F
        BTFSS   STATUS, Z       ; SI: ¿TAMBIEN ES CERO LA PARATE BAJA DEL NUEMRO DE DATOS
                                ;  HA COMPLETADO LA ESCRITURA
        GOTO    WROTROF         ; NO: FALTAN DATOS POR RECIBIR  Y ESCRIBIR
        RETURN                  ; SI: SE HA RECIBIDO Y ESCRITO TODO EL BLOQUE DE
                                ; DATOS, RETORNARA A LA ESPERA DE NUEVA INRTUCCION

;*****************************************************************
;*** RUTINA DE ESCRITURA DE DATOS EN MEMORIA EEPROM      ***
;*****************************************************************

ESCRIBIR_DATOS_EEPROM:

WROTROE:

        CALL   RECEPCION_DIR_DATO       ; ADQUIERE LA DIRECCION Y DATO DE ESCRITUA
                                        ;  DA LA MEMORIA EEPROM
        CALL    ESCRIBIR_EEPROM         ; ESCIBE EL DATO EN LA DIRECCION ACTUAL
                                        ; DE EEADR Y EEADRH
        BANK0
        MOVLW   0X41            ; TRANSMITE CABECERA DE FIN DE ESCRITURA DEL DATO RECIBIDO
        MOVWF   DATOTX
        CALL    TRANSMITE

        BANK2
        MOVLW   0X01
        SUBWF   NUMEROL, F      ; DECREMENTAR UNA UNIDAD LA PARTE BJA DEL NUEMRO DE DATOS

        BTFSS   STATUS, C       ; ¿ES NEGATIVO?
        DECF    NUMEROH,F       ; SI: DECREMENTA UNA UNIDAD A LA PARTE ALTA DEL NUEMRO DE DATOS
        MOVF    NUMEROH, F
```

```
        BTFSS   STATUS, Z       ; ¿ES CERO LA PARTE ALTA DEL NUEMRO DE DATOS?
        GOTO    WROTROE         ; NO: FALTAN DATOS POR RECIBIR Y ESCRIBIR
        MOVF    NUMEROL, F
        BTFSS   STATUS, Z       ; SI: ¿TAMBIEN ES CERO LA PARATE BAJA DEL NUMERO
                                ; DE DATOS  HA COMPLETADO LA ESCRITURA
        GOTO    WROTROE         ; NO: FALTAN DATOS POR RECIBIR  Y ESCRIBIR
        RETURN                  ; SI: SE HA RECIBIDO Y ESCRITO TODO EL BLOQUE
                                ; DE DATOS, RETORNARA A LA ESPERA DE NUEVA INRTUCCION

;*****************************************************************
;*** RUTINA DE LEECTURA DE DATOS DE LA MEMORIA EEPROM  ***
;*****************************************************************

LEER_DATOS_EEPROM:

RDOTROE:

        CALL    LEER_EEPROM
        MOVLW   0XFF
        BANK0
        MOVWF   DATOTX          ; TRANSMITE EL DATO ALTO LEIDO DE LA DIRRECCION ACTUAL
        CALL    TRANSMITE
        BANK2
        MOVF    EEDATA, W       ; TRANSMITE EL DATO BAJO LEIDO DE LA DIRRECCION ACTUAL
        BCF     STATUS, RP1     ; BANCO 0
        MOVWF   DATOTX
        CALL    TRANSMITE
        BANK2
        INCF    EEADR, F        ; APUNTA A SIGUIENTE DIRECCION DE MEMORIA EEPROM
        BTFSC   STATUS, Z
        INCF    EEADRH, F

        MOVLW   0X01
        SUBWF   NUMEROL, F      ; DECREMENTAR UNA UNIDAD LA PARTE BJA DEL NUEMRO DE DATOS
        BTFSS   STATUS, C       ; ¿ES NEGATIVO?
        DECF    NUMEROH, F      ; SI: DECREMENTA UNA UNIDAD A LA PARTE ALTA DEL NUEMRO DE DATOS
        MOVF    NUMEROH, F
        BTFSS   STATUS, Z       ; ¿ES CERO LA PARTE ALTA DEL NUEMRO DE DATOS?
        GOTO    RDOTROE         ; NO: FALTAN DATOS POR LEER
        MOVF    NUMEROL, F
        BTFSS   STATUS, Z       ; SI: ¿TAMBIEN ES CERO LA PARATE BAJA DEL NUEMRO
                                ; DE DATOS  HA COMPLETADO LA ESCRITURA
        GOTO    RDOTROE         ; NO: FALTAN DATOS POR LEER
        RETURN                  ; SI: SE HA LEIDO TODOS LOS DATOS, RETORNARA DE INTERRUPCION

;*****************************************************************
;*** RUTINA DE LEECTURA DE DATOS DE LA MEMORIA FLASH  ***
;*****************************************************************

LEER_DATOS_FLASH:

RDOTRO:

        CALL    LEER_FLASH
        BANK2
        MOVF    EEDATH, W
        BCF     STATUS, RP1     ; BANCO 0
        MOVWF   DATOTX          ; TRANSMITE EL DATO ALTO LEIDO DE LA DIRRECCION ACTUAL
        CALL    TRANSMITE
        BANK2
        MOVF    EEDATA, W       ; TRANSMITE EL DATO ALTO LEIDO DE LA DIRRECCION ACTUAL
        BCF     STATUS, RP1     ; BANCO 0
        MOVWF   DATOTX
        CALL    TRANSMITE
        BANK2
        INCF    EEADR, F        ; APUNTA A SIGUIENTE DIRECCION DE MEMORIA FLASH
        BTFSC   STATUS, Z
        INCF    EEADRH, F
        MOVLW   0X01
        SUBWF   NUMEROL, F      ; DECREMENTAR UNA UNIDAD LA PARTE BJA DEL NUEMRO DE DATOS
```

```
        BTFSS   STATUS, C           ; ¿ES NEGATIVO?
        DECF    NUMEROH, F          ; SI: DECREMENTA UNA UNIDAD A LA PARTE ALTA DEL NUEMRO DE DATOS
        MOVF    NUMEROH, F
        BTFSS   STATUS, Z           ; ¿ES CERO LA PARTE ALTA DEL NUEMRO DE DATOS?
        GOTO    RDOTRO              ; NO: FALTAN DATOS POR LEER
        MOVF    NUMEROL, F
        BTFSS   STATUS, Z           ; SI: ¿TAMBIEN ES CERO LA PARATE BAJA DEL NUEMRO
                                    ; DE DATOS  HA COMPLETADO LA ESCRITURA
        GOTO    RDOTRO              ; NO: FALTAN DATOS POR LEER
        RETURN                      ; SI: SE HA LEIDO TODOS LOS DATOS, RETORNARA DE INTERRUPCION

;*******************************************************************************
;** RUTINA DE CARGA DE VALORES DE LAS CONSTANTES DE CONTROL PARA EL ***
;** CORRECTO FUNCIONAMIENTO DEL ALGORITMO DE CONTROL DIFUSP         ***
;*******************************************************************************

CARGAR_CONSTANTES_DE_CONTROL:

        BANK2
        MOVLW DIRCCL                ; DIRECCION DE LA MEMORIA FLASH0X07D0 0 2000 decimal
        MOVWF EEADR                 ; DIRECCION BAJA DE LA MEMORIA FLASH
        MOVLW DIRCCH
        MOVWF EEADRH                ; DIRECCION ALTA DE LA MEMORIA FLASH
        BSF   STATUS, IRP           ; SELECCION EL BANCO 2 Y 3 DEL DIRECCIONAMIENTO INDIRECTO
        MOVLW 0X50
        MOVWF FSR       ; FSR DIRECCIONA INICIALMENTE LA POSICION 0X150 DE LA MEMORIA DE RAM

OTRO:

        CALL    LEER_FLASH
        BANK2
        MOVF    EEDATA, W
        MOVWF   INDF
        INCF    EEADR, F
        INCF    FSR, F
        MOVLW   DIRCCL + 0X09
        SUBWF   EEADR, W
        BTFSS   STATUS, Z
        GOTO    OTRO

OTRO1:

        CALL    LEER_FLASH
        BANK2
        MOVF    EEDATH, W
        MOVWF   INDF
        INCF    FSR, F
        MOVF    EEDATA, W
        MOVWF   INDF
        INCF    EEADR, F
        INCF    FSR, F
        MOVLW   DIRCCL + 0X0C
        SUBWF   EEADR, W
        BTFSS   STATUS, Z
        GOTO    OTRO1
        ;BCF    STATUS, IRP         ; SELECCION DE BANCO 0 Y 1 DEL DIRECCIONAMIENTO INDIRECTO
        RETURN

;*****************************************************************************************
;*** RUTINA DE INICIALIZACION Y CONFIGURACION PARA EL CONRTOL DE LA FRECUENCIA DE    ***
;*** MUESTREO DEL SCD                                                               ***
;*****************************************************************************************

FIJAR_T_MUESTREO:

        BANK1
        MOVLW B'11000111'           ; CONFIGURAR TIMER0,  PREDIVISOR 1/256
        MOVWF OPTION_REG
        BSF   INTCON, T0IE          ; HABILITA INTERRUPCION POR TIMER0
        BCF   INTCON, T0IF          ; LIMPIAR BANDERA DE INTERRUPCION
```

```
        BANK2
        MOVF    CUENTA, W
        MOVWF   CONTADOR3
        MOVF    T_MUESTREO, W
        BANK0
        MOVWF   TMR0
        RETURN

;*****************************************************************************
;*** RUTINA DE ASEGURAMIENTO INICIAL DEL COMANDO DE PERMANENCIA  ***
;*** EN EL MODO EJECUCION EJECUTANDO EL PROGRMA DE APLICACION      ***
;*****************************************************************************

HABILITAR_EJECUCION:

        BANK2
        CLRF    BANDERAS        ; PONE A CERO TODAS LAS BADERAS DEL PROCESO DE CONTROL DIFUSO
        MOVLW   CONTINUA_EJECUCION      ; CARGA EL COMANDO CONTINUAR EJECUCION
                                        ; AL REGISTRODE CONTROL DE PERMAMNECIA
        MOVWF   CONTINUAE
        RETURN

;*************************************
;*** RUTINAS DE CONTROL DE LCD  ***
;*************************************

;*********************************************
;*** RUTINA DE INICIALIZACION DE LA LCD   ***
;*********************************************

INILCD:

        BANKSEL  ADCON1         ; BANCO 1
        MOVLW   0X82
        MOVWF   ADCON1          ; EL RESULTADO SE JUSTIFICA A LA DERECHA, ADFM = 1;
                                ; 5 ENTRADAS ANALOGICAS, PCFG3:0 = b'0010',
                                ; LOS TRES PIENES DEL PUERTO E COMO E/S DIGITALES.

        BANKSEL  TRISD          ; SELECCIONA BANCO 0
        CLRF    TRISD           ; PONE PUERTO D COMO SALIDA
        CLRF    TRISE           ; PONE PUERTO E COMO SALIDA

        MOVLW   SETLCD1         ; CONFIGURA LCD
        CALL    CONTLW
        CALL    RETA5MS
        MOVLW   SETLCD1         ; CONFIGURA LCD
        CALL    CONTLW
        CALL    RETA5MS
        MOVLW  SETLCD1          ; CONFIGURA LCD
        CALL    CONTLW
        CALL    RETA5MS
        MOVLW  ONLCD1           ; ENCIENDE LCD
        CALL    CONTLW
        CALL    RET40US
        MOVLW  MINCHAR1         ; CONFIGURA MODO DE ENTRADA DE CARACTERES
        CALL    CONTLW
        CALL    RET40US
        MOVLW  CLRFLCD          ; BORRA LCD Y REGRESA CURSOR A LA POSICION CERO
        CALL    CONTLW
        CALL    RET1.64MS
        RETURN

;*********************************************
;** SUBRUTINAS PARA EL ENVIO DE DATOS Y ***
;** INSTRUCCIONES A LA LCD                ***
;*********************************************

DATALW:                         ; RUTINA DE ESCRITURA DE DATOS EN LA LCD
```

```
        BANK0
        MOVWF   PORTD           ; MOVER DATO AL BUS DE DATOS DE LA LCD
        BCF     RW              ; R/W = 0, ESCRITURA LCD
        BSF     R_S             ; R/S = 1, ENVIO DE DATO
        BSF     ENABLE          ;  HABILITACION DE LA LCD
        NOP
        NOP
        NOP
        BCF     ENABLE          ; DESHABILITA LCD
        CALL    RET40US
        RETURN

;DATALR:                        ; RUTINA DE LECTURA DE DATOS DE LA LCD

        ;   BANK0
        ;   BSF     RW
        ;   BSF     R_S
        ;   BSF     ENABLE
        ;   NOP
        ;   MOVF    PORTD, W
        ;   NOP
        ;   BCF     ENABLE
        ;   CALL    RET40US
        ;   MOVWF   DATL
        ;   RETURN

CONTLW:                         ; RUTINA DE ENVIO DE COMANDOS A LA LCD

        BANK0
        MOVWF   PORTD           ; MOVER DATO AL BUS DE DATOS DE LA LCD
        BCF     RW              ; R/W = 0, ESCRITURA LCD
        BCF     R_S             ; R/S = 0, ENVIO DE COMANDO
        BSF     ENABLE          ; HABILITACION DE LA LCD
        NOP
        NOP
        NOP
        BCF     ENABLE          ; DESHABILITA LCD
        RETURN

;*****************************************
;** RUTINA DE BORRADO DE LA LCD ***
;*****************************************

BORRAR_LCD:

        MOVLW   CLRFLCD         ; MUEVE COMANDO DE BORRADO
        CALL    CONTLW          ; ENVIA COMANDO A LA LCD
        CALL    RET1.64MS       ; RETARDO DE DURACION DE LA EJECUCION DEL COMANDO DE BORRAD
        RETURN

;*****************************************
;** RUTINAS DE RETARDO PARA LCD  **
;*****************************************

;RETA15MS:                      ; RETARDO APROXIMADO DE 15mS

;       MOVLW   D'98'
;       MOVWF   CONTADOR2
;       CLRF    CONTADOR1
;RETA
;       DECFSZ  CONTADOR1,F
;       GOTO    RETA
;       DECFSZ  CONTADOR2,F
;       GOTO    RETA
;       RETURN

RETA5MS:                        ; RETARDO APROXIMADO DE 5mS

        MOVLW   D'33'
        MOVWF   CONTADOR2
```

```
        CLRF    CONTADOR1
        RETAX:
        DECFSZ  CONTADOR1, F
        GOTO    RETAX
        DECFSZ  CONTADOR2, F
        GOTO    RETAX
        RETURN

RET40US:                        ; RETARDO APROXNIMADO DE 40uS

        MOVLW D'65'
        MOVWF CONTADOR1
        RETA2:
        DECFSZ CONTADOR1, F
        GOTO    RETA2
        RETURN

        ;RET20MS:               ; RETARDO APROXIMADO DE 20ms

        ;  MOVLW D'130'
        ;  MOVWF CONTADOR2
        ;  CLRF   CONTADOR1
        ;  RETA3:
        ;  DECFSZ CONTADOR1, F
        ;  GOTO   RETA3
        ;  DECFSZ CONTADOR2, F
        ;  GOTO   RETA3
        ;  RETURN

RET1.64MS:                      ; RETARDO APROXIMADO DE 1.64ms

        MOVLW  D'11'
        MOVWF  CONTADOR2
        CLRF   CONTADOR1
        RETA4:
        DECFSZ CONTADOR1, F
        GOTO   RETA4
        DECFSZ CONTADOR2, F
        GOTO   RETA4
        RETURN

;*****************************************************************************
;*** RUTINAS DE UTILIDAD DEL PROCESO DE INNFERENCIA DIFUSA (FUZZIFICACION, ***
;*** EVALUACION DE REGLAS Y DEFUZZIFICACION)                              ***
;*****************************************************************************


;*****************************************************************************
;** RUTINA PARA CALCULAR EL VALOR DE MEMBRESIA DE UNA ENTRADA CRISP A  UNA   ***
;** FUNCION DE MEMBRESIA TRAPEZOIDAL                                        ***
;*****************************************************************************

GDM:                            ; ESTA RUTINA NO MODIFICA EL VALOR DE PXL Y PXH

        CALL    RESA            ; PA<X
        BTFSC   STATUS, C
        GOTO    MEIGB           ; SI

        MOVLW   0X04            ; NO
        ADDWF   EEADR, F        ; APUNTA A LA SIGUIENTE FM
        VM01:
        CALL    VAMEM0
        RETURN

MEIGB:
        INCF    EEADR, F
        CALL    RESB            ; PB>X
        BTFSC   STATUS, C
        GOTO    M1IG0
```

```
        MOVLW 0X03
        ADDWF  EEADR, F             ; APUNTA A LA SIGUIENTE FM
        GOTO    VM01
        M1IG0:
        INCF    EEADR, F
        FLASHRD
        MOVF    EEDATA, W
        BTFSS   STATUS, Z       ; M1=0
        GOTO    AVANZA

        IDEM:
        CALL    VAMEM1
        GOTO    M2IG0
AVANZA:
        CALL    LDOPMA
        CALL    MULFACT
        CALL    VAMMA1
        BTFSS   STATUS, C
        GOTO    IDEM
VMESRM:
        CALL    VMIGRM
M2IG0:
        INCF    EEADR, F
        FLASHRD
        MOVF    EEDATA, W
        BTFSS   STATUS, Z       ; M2=0
        GOTO    AVANZA1
        GOTO    TERMINO1
AVANZA1:
        CALL    LDOPMB
        CALL    MULFACT
        CALL    VAMMA1
        BTFSS   STATUS, C
        GOTO    TERMINO1

        MOVLW   0X02
        SUBWF   DIRVM, F
        CALL    VMIGRM
TERMINO1:
        INCF    EEADR, F        ; APUNTA A LA SIGUIENTE FM
        RETURN

        ;*********************************
        ;*** RUTINAS INTERMEDIAS  ***
        ;*********************************

LDOPMB:
        MOVF    EEDATA, W
        MOVWF OPAL2
        CLRF    OPAH2
        MOVF    RESBL, W
        MOVWF OPAL1
        MOVF    RESBH, W
        MOVWF  OPAH1
        RETURN

LDOPMA:
        MOVF    EEDATA, W
        MOVWF  OPAL2
         MOVF   RESAL, W
         MOVWF OPAL1
        MOVF    RESAH, W
        MOVWF OPAH1
        RETURN

VAMMA1:
        MOVF    OPAH3, W
        SUBLW   0XFF
        BTFSC   STATUS, C
        GOTO    NOSUM5A
```

```
        INCF    OPAL4, W
NOSUM5B:
        SUBLW   0X03
        RETURN

NOSUM5A:
        MOVF    OPAL4, W
        GOTO    NOSUM5B

        ;*************************************
        ;*** VALOR DE MEMBRESIA CERO ***
        ;*************************************

VAMEM0:

        MOVF    DIRVM, W
        MOVWF   FSR
        CLRF    INDF
        INCF    FSR, F
        CLRF    INDF
        INCF    FSR, W
        MOVWF   DIRVM
        RETURN

        ;*************************************
        ;*** VALOR DE MEMBRESIA  UNO ***
        ;*************************************

VAMEM1:

        MOVF    DIRVM, W
        MOVWF FSR
        MOVLW  0XFF
        MOVWF  INDF
        INCF    FSR, F
        MOVLW  0X03
        MOVWF  INDF
        INCF    FSR, W
        MOVWF   DIRVM
        RETURN

        ;*********************************************
        ;*** VALOR DE MEMBRESIA IGUAL A RM ****
        ;*********************************************

VMIGRM:

        MOVF    DIRVM, W
        MOVWF  FSR
        MOVF    OPAH3, W
        MOVWF  INDF
        INCF    FSR, F
        MOVF    OPAL4, W
        MOVWF  INDF
        INCF    FSR, W
        MOVWF  DIRVM
        RETURN

                ;*************************************
                ;*** SUBRRUTINAS DE CONDICION ***
                ;*************************************

RESA:
        FLASHRD
        MOVF    EEDATA, W       ; PX-PA
        SUBWF   PXL, W          ; PXL SIGUE SIN SER TOCADO
        MOVWF  RESAL
        BTFSC   STATUS, C
        GOTO    NOSUMA
```

```
        INCF    EEDATH, W
        NOSUMB:
        SUBWF   PXH, W
        MOVWF RESAH
        RETURN

RESB:
        FLASHRD
        MOVF    PXL, W          ; PB-PX
        SUBWF   EEDATA, W
        MOVWF   RESBL
        BTFSC   STATUS, C       ; PX SIGUE INTACTO
        GOTO    NOSUM1A
        INCF    PXH, W
NOSUM1B:
        SUBWF   EEDATH, W
        MOVWF   RESBH
        RETURN

NOSUMA:
        MOVF    EEDATH, W
        GOTO    NOSUMB
NOSUM1A:
        MOVF    PXH, W
        GOTO    NOSUM1B

;*****************************************************************
;** RUTINAS DE UTILIDAD PARA LA EVALUACION DE REGLAS  ***
;*****************************************************************

;*****************************************************************
;** RUTINA QUE CARGA DE OPERANDOS PARA LA OPERACION MIN ***
;*****************************************************************

                                ; ESTA RUTINA CONSIDERA COMO MINIMO DOS ANTECEDENTES EN UNA
                                ; REGLA NO SE APLICA A REGLAS DE UN ANTECEDENTE

LDOPMIN:

        MOVF    INDF, W         ; MUEVE LA PARATE BAJA DEL VALOR DE MEMBRESIA OPAL1
        MOVWF   OPAL1
        INCF    FSR, F  ; INCREMENTA FSR PARA QUE APUNTE A LA PARTE ALTA DEL VALOR DE MEMBRESIA
        MOVF    INDF, W         ; MUEVE LA PARTE ALTA DEL VALOR DE MEMBRESIA OPAH1
        MOVWF   OPAH1
        INCF    EEADR, F        ; APUNTA A LA SIGUINTE DIRECCION DE EEPROM,
                                ; QUE CONTENDRA LA DIRECCION DEL
                                ; SIGUIENTE ANTECEDENTE SI HAY MAS DE UNO
        EEPRD
        MOVF    EEDATA, W
        MOVWF   FSR
        MOVF    INDF, W
        MOVWF   OPAL2
        INCF    FSR, F
        MOVF    INDF, W
        MOVWF   OPAH2
        RETURN

STOPMIN:

        MOVF    INDF, W
        MOVWF   OPAL1
        INCF    FSR, F
        MOVF    INDF, W
        MOVWF   OPAH1
        MOVF    RMINL, W
        MOVWF   OPAL2
        MOVF    RMINH, W
        MOVWF   OPAH2
        RETURN
```

```
;*********************************************************************
;*** RUTINA QUE CARGA DE OPERANDOS PARA LA OPERACION MAX ****
;*********************************************************************

LDOPMAX:

        MOVF    INDF, W
        MOVWF   OPAL1
        INCF    FSR, F
        MOVF    INDF, W
        MOVWF   OPAH1
        MOVF    RMINL, W
        MOVWF   OPAL2
        MOVF    RMINH, W
        MOVWF   OPAH2
        RETURN

;**********************
;*** OPERACION MIN ***
;**********************

MIN_BAS:
        CALL    RESO2O1
        BTFSC   STATUS, C
        GOTO    RMINOP2

        ;*********************************
        ;** EL MIN ES EL OPERADOR1 ***
        ;*********************************
RMINOP1:

        MOVF    OPAL1, W
        MOVWF   RMINL
        MOVF    OPAH1, W
        MOVWF   RMINH
        RETURN
        ;*********************************
        ;** EL MIN ES EL OPERADOR2 ***
        ;*********************************
RMINOP2:

        MOVF    OPAL2, W
        MOVWF   RMINL
        MOVF    OPAH2, W
        MOVWF   RMINH
        RETURN

;*********************************************************************
;** RUTINA DE CARGA DE OPERADORES PARA DETERMINAR MIN O MAX ***
;*********************************************************************

RESO2O1:
        MOVF    OPAL2, W
        SUBWF   OPAL1, W
        BTFSC   STATUS, C
        GOTO    CAM1A
        INCF    OPAH2, W
CAM1B:
        SUBWF   OPAH1, W
        RETURN
CAM1A:
        MOVF    OPAH2, W
        GOTO    CAM1B

;**********************
;** OPERCION MAX ***
;**********************
MAX_BAS:
```

```
        CALL    RESO2O1
        BTFSC   STATUS, C
        GOTO    RMAXOP1
        ;*********************************
        ;** EL MAX ES EL OPERADOR2 ***
        ;*********************************
RMAXOP2:

        MOVF    OPAL2, W
        MOVWF   RMAXL
        MOVF    OPAH2, W
        MOVWF   RMAXH
        RETURN
        ;*********************************
        ;** EL MAX ES EL OPERADOR1 ***
        ;*********************************

RMAXOP1:

        MOVF    OPAL1, W
        MOVWF   RMAXL
        MOVF    OPAH1, W
        MOVWF   RMAXH
        RETURN

;*****************************************
;*** SUBRRUTINA DE DEFUZZYFICACION  ***
;*****************************************

;*************************************************************************************************
;*** RUTINA PARA SUMAR LOS ELEMENTOS DEL DENOMINADOR DE LA FORMULA DE PROMEDIO  ***
;*** DE PESOS DE LA DEFUZZIFICACION                                             ***
;*************************************************************************************************

SUM_DEN:

        MOVF    OPAL1, W
        ADDWF   SUML, F
        BTFSC   STATUS, C
        INCF    SUMH, F
        MOVF    OPAH1, W
        ADDWF   SUMH, F
        RETURN


;*****************************************************
;*** SUBRUTINA DE MULTIPLICACION 16 POR 16 BITS ***
;*****************************************************

MULTI:

        MOVLW   D'16'
        MOVWF   CONTADOR

        CLRF    OPAL4
        CLRF    OPAH4
        CLRF    OPAL3
        CLRF    OPAH3

        BCF     STATUS, C
CICLOM:
        RRF     OPAH2, F
        RRF     OPAL2, F

        BTFSS   STATUS, C
        GOTO    NO_SUMA
        CALL    SUMA
NO_SUMA:
```

```
        RRF     OPAH4, F
        RRF     OPAL4, F
        RRF     OPAH3, F
        RRF     OPAL3, F

        DECFSZ  CONTADOR, F
        GOTO    CICLOM

        RETURN

;*****************************************************
;*** OPERACION DE SUMA PARA LA MULTIPLICACION ***
;*****************************************************

SUMA:

        MOVF    OPAL1, W
        ADDWF   OPAL4, F
        BTFSS   STATUS, C
        GOTO    NO_INC
        INCF    OPAH4, F
        BTFSS   STATUS, Z
        GOTO    NO_INC
        MOVF    OPAH1, W
        ADDWF   OPAH4, F
        BSF     STATUS, C
        RETURN
NO_INC:
        MOVF    OPAH1, W
        ADDWF   OPAH4, F
        RETURN

;*****************************************************************************************
;*** RUTINA PARA CALCULAR Y SUMAR LOS ELEMENTOS DEL NUMERADOR  DE LA FORMULA ***
;*** DE PROMEDIO DE PESOS DE LA DEFUZZIFICACION                              ***
;*****************************************************************************************

SUMATORIA:

        BCF     ACARREO
        MOVF    OPAL3, W
        ADDWF   ADDL1, F
        BTFSS   STATUS, C
        GOTO    NO_SUM1
        INCF    ADDH1, F
        BTFSC   STATUS, Z
        BSF     ACARREO
NO_SUM1:
        MOVF    OPAH3, W
        ADDWF   ADDH1, F
        BTFSC   ACARREO
        BSF     STATUS, C
        BTFSC   STATUS, C
        INCF    ADDL2, F
        MOVF    OPAL4, W
        ADDWF   ADDL2, F
        RETURN

;*********************************************
;*** SUBRRUTINA DE DIVISION DE 32/16 BITS ***
;*********************************************

DIVISION:

        CLRF    OPAL5
        CLRF    OPAH5
        CLRF    OPAL2
        CLRF    OPAH2

        MOVLW   D'16'
```

```
        MOVWF  CONTADOR

        BCF    STATUS, C
CICLOD:
        RLF    OPAL3, F
        RLF    OPAH3, F
        RLF    OPAL4, F
        RLF    OPAH4, F
        RLF    OPAL5, F
        RLF    OPAH5, F

        MOVF   OPAL6, W
        SUBWF  OPAL5, W

        BTFSS  STATUS, Z
        GOTO   NO_RES

        MOVF   OPAH1, W
        SUBWF  OPAH4, W

        BTFSS  STATUS, Z
        GOTO   NO_RES

        MOVF   OPAL1, W
        SUBWF  OPAL4, W
NO_RES:
        BTFSS  STATUS, C
        GOTO   NO_STRES

        MOVF   OPAL1, W
        SUBWF  OPAL4, F

        BTFSS  STATUS, C
        DECF   OPAH4, F

        MOVF   OPAH1, W
        SUBWF  OPAH4, F

        BTFSS  STATUS, C
        DECF   OPAL5, F

        MOVF   OPAL6, W
        SUBWF  OPAL5, F

        BSF    STATUS, C
NO_STRES:
        RLF    OPAL2, F
        RLF    OPAH2, F

        DECFSZ  CONTADOR, F
        GOTO   CICLOD
        RETURN

;*********************************************************************
;*** SUBRRUTINA PARA LA  CARGA DE OPERADORES PARA LA DIVISION ***
;*********************************************************************

LDOPA:

        BANK2
        MOVF   ADDL1, W
        MOVWF  OPAL3
        MOVF   ADDH1, W
        MOVWF  OPAH3
        MOVF   ADDL2, W
        MOVWF  OPAL4
        CLRF   OPAH4

        MOVF   SUML, W
        MOVWF  OPAL1
```

```
        MOVF    SUMH, W
        MOVWF   OPAH1
        RETURN

;*************************************
;*** RUTINA DE RETARDO DEL CDA ***
;*************************************

RET30US:

        MOVLW   D'20'
        MOVWF   CONTADOR1
RETA5
        DECFSZ  CONTADOR1, F
        GOTO    RETA5

;*********************************************************************
;** RUTINA PARA MULTIPLICAR (MULTIPLICACION DE 8 BITS) FACTOR 5 **
;*********************************************************************

MULFACT:

        MOVLW   D'8'
        MOVWF   CONTADOR

        CLRF    OPAL4
        CLRF    OPAH4
        CLRF    OPAH3


        BCF     STATUS, C
CICLOMF:
        RRF     OPAH2, F
        RRF     OPAL2, F

        BTFSS   STATUS, C
        GOTO    NO_SUMAF
        CALL    SUMA
NO_SUMAF:
        RRF     OPAH4, F
        RRF     OPAL4, F
        RRF     OPAH3, F

        DECFSZ  CONTADOR, F
        GOTO    CICLOMF
        RETURN

;*************************************************
;*** RUTINAS DE ATENCION A INTERRUPCIONES  ***
;*************************************************

INTERRUPCION:

;*****************************************************************************************************************
;***AQUI SE SALVA W, STATUS Y FSR ANTES DE ENTRAR A UNA DE LAS RUTINA DE ATENCION  INTERRUPCION ***
;*****************************************************************************************************************

        MOVWF   W_TEMP          ; SALVA EL VALOR DEL W
        SWAPF   STATUS, W
        MOVWF   STATUS_TEMP     ; SALVA EL VALOR DEL STATUS
        SWAPF   FSR, W
        MOVWF   FSR_TEMP        ; SALVA EL VALOR DEL FSR

;*********************************************************************
;*** SE VERIFICA QUE DISPOSITIVO HA PRODUCIDO LA INTERRUPCION ***
;*********************************************************************

        BANK0
        BTFSC   INTCON, T0IF                    ; INTERRUPCION POR TIMER0
        GOTO    EJECUTAR_PROCESO_DIFUSO
```

```
        BTFSC   INTCON, INTF                    ; INTERRUPCION EXTERNA POR RBO
        GOTO    ACTUALIZAR_RETARDOS
        BTFSC   PIR1, TMR1IF                    ; INTERRUPCION POR DESBORDE DEL TIMER 1
        GOTO    DISPARAR_TRIAC1
        BTFSC   PIR1, TMR2IF            ; INTERRUPCION DEL TIMER2
        GOTO    DISPARAR_TRIAC2
        BTFSC   PIR1, RCIF              ; INTERRUPCION POR RECEPCION
        GOTO    INTRX

;*****************************************************************************************
;** ANTES DE RETORNAR DE LAS RUTINASA DE ATANCION A  INTERRUPCIONES SE RECUPERA      ***
;** W, STATUS Y FSR                                                                  ***
;*****************************************************************************************

RETORNO_DE_INTERRUPCION:

        SWAPF   STATUS_TEMP, W
        MOVWF   STATUS                  ; RECUPERA EL VALOR DEL STATUS
        SWAPF   FSR_TEMP, W
        MOVWF   FSR                     ; RECUPERA EL VALOR DEL FSR
        SWAPF   W_TEMP, F
        SWAPF   W_TEMP, W               ; RECUPERA EL VALOR DE W

;*************************************
;** SE REGRESA DE INTERRUPCION  ***
;*************************************

        RETFIE

;*****************************************************************************
;** APARTIR DE AQUI SE ENCUNTRAN LAS RUTINAS DE ATENCION A INTERRUPCIONES  **
;*****************************************************************************

;*************************************************************************
;** RUTINA DE ATENCION A INTERRUPCION POR DESBORDAMIENTO DEL TIMER0 ***
;*************************************************************************

EJECUTAR_PROCESO_DIFUSO:

        BANK2
        BCF     INTCON, T0IF            ; LIMPIA BANDERA
        DECFSZ  CONTADOR3, F            ; ¿ES CERO EL CONTADOR?
        GOTO    NO_CERO

        BSF     BANDERA_MUESTREO        ; INICIAR MUESTREO
        MOVF    CUENTA, W
        MOVWF   CONTADOR3
        MOVF    T_MUESTREO, W
        BANK0
        MOVWF   TMR0
        GOTO    RETORNO_DE_INTERRUPCION         ; RETORNAR DE INTERRUPCION

NO_CERO:

        MOVF    T_MUESTREO, W
        BANK0
        MOVWF   TMR0
        GOTO    RETORNO_DE_INTERRUPCION         ; RETORNAR DE INTERRUPCION

;*****************************************************************
;*** RUTINA DE ATENCION A INTERRUPCION EXETERNA POR RB0     ***
;*****************************************************************

ACTUALIZAR_RETARDOS:
                                ; ACTUALIZA RETARDO DEL TRIAC1 Y EL TRIAC2,(SOLO SI ES NECESARIO)
                                ; ESTO ES: SE HA DETECTADO EL  CRUCE POR CERO DE LA SEÑAL
                                ; DE ALTERNA QUE MANJAN LOS TRIACS
        BCF     INTCON, INTF    ; SE CLAREA  LA BADERA DE INTERRUPCION EXTERNA
        BANK1
        BTFSS   INTCON, INTE    ; VERIFICAR SI ESTABA PERMITIDA LA INTERRUPCION
```

```
                        ; EXTERNA POR RB0 (INTE)
        GOTO    RETORNO_DE_INTERRUPCION     ; NO: RETORNA  DE INTERRUPCION
                                            ; SI: ATIENDE LA INTERRUPCION
        BANK0
        BCF     PIR1, TMR1IF                ; CLAREA LA BADERA DE INTERRUPCION DEL TIMER1
        BCF     PULSO_TRIAC1                ; SE PONE A CERO EL PULSO DE DISPARO DEL TRAIC1
        COMF    DELAYL, W                   ; ACTULIZA RETARDO DEL TIMER1
        MOVWF   TMR1L
        COMF    DELAYH, W
        MOVWF   TMR1H
        BSF     T1CON, TMR1ON               ; ENCIENDE EL TIMER1
        BANK1
        BTFSS   PIE1, TMR2IE                ; VERIFICA SI ESTA PERMITIDA LA INTERRUPCION DEL TIMER2
        GOTO    RETORNO_DE_INTERRUPCION     ; NO: RETORNA DE INTERRUPCION
                                            ; SI: ACTUALIZA EL REGISTRO DE PERIODO DEL TIMER 2
        BANK0
        BCF     PIR1, TMR2IF                ; CLAREA LA BANDERA DE INTERRUPCION DEL TIMER2
        BCF     PULSO_TRIAC2                ; SE PONE A CERO EL PULSO DE DISPARO DEL TRIAC2
        MOVF    DELAY_TMR2, W
        SUBLW   0XFF
        BANK1
        MOVWF   PR2
        BANK0                               ; ACTUALIZA EL VALOR DEL REGISTRO DE PERIODO DEL TIMER2
        BSF     T2CON, TMR2ON               ; ENCIENDE EL TIMER2
        GOTO    RETORNO_DE_INTERRUPCION

;*********************************************************************
;*** RUTINA DE ATENCION A INTERRUPCION POR DESBORDE DEL TIMER 1 ***
;*********************************************************************

DISPARAR_TRIAC1:

        BCF   PIR1, TMR1IF          ; CLAREA BANDERA DE INTERRUPCION DEL TIMER1
        BANK1
        BTFSS   PIE1, TMR1IE        ; VERIFICA SI ESTAN PERMITIDAS LAS INTERRUPCIONES DEL TIMER1
        GOTO    RETORNO_DE_INTERRUPCION     ; NO: REGRESA DE INTERRUPCION
                                            ; SI: ATIENDE INTERRUPCION
        BANK0
        BSF     PULSO_TRIAC1                ; DISPARA EL TRIAC1 PONIENDO A 1 EL PULSO DE DISPARO
        BCF     T1CON, TMR1ON               ; APAGA EL TIMER1
        GOTO    RETORNO_DE_INTERRUPCION     ; REGERSA DE INTERUPCION

DISPARAR_TRIAC2:

        BCF     PIR1, TMR2IF                ; CLAREA BANDERA DE INTERRUPCION DEL TIMER2
        BANK1
        BTFSS   PIE1, TMR2IE        ; VERIFICA SI ESTAN PERMITIDAS LAS INTERRUPCIONES DE ESTE DISPOSITIVO
        GOTO    RETORNO_DE_INTERRUPCION     ; NO: RETORNA DE INTERRUPCION
                                            ; SI: ATIENDE INTERRUPCION
        BANK0
        BSF     PULSO_TRIAC2                ; DISPARA EL TRIAC PONIENDO A 1 EL PULSO DE DISPARO
        BCF     T2CON, TMR2ON               ; APAGA EL TIMER2
        GOTO    RETORNO_DE_INTERRUPCION     ; REGRESA DE INTERRUPCION

;*********************************************************************
;*** RUTINA DE ATENCION A INTERRPCION POR RECEPCION  DE LA USART  ***
;*********************************************************************

INTRX:

        BANK0
        MOVF    RXDIR, W            ; DIRECCIÓN ACTUAL DE BUFFER DE RECEPCIÓN
        MOVWF   FSR
        MOVF    RCREG, W            ; LEE EL DATO RECIBIDO
        MOVWF   INDF                ; COLOCA DATOS RECIBIDO EN BUFFER DE RECEPCIÓN
        INCF    FSR, W              ; APUNTA A SIGUIENTE DIRECCIÓN DEL BUFFER DE RECEPCIÓN
        MOVWF   RXDIR
        BSF     STATUS, RP1         ; BANCO 2
        MOVF    RXDAT, W
```

```
        BCF     STATUS, RP1         ; BANCO 0
        ADDLW   RXINI               ; RXINI+RXDAT
        SUBWF   RXDIR, W
        BTFSS   STATUS, Z           ; ¿EL BUFFER DE RECEPCIÓN SE HA LLENADO?
        GOTO    RETORNO_DE_INTERRUPCION         ; NO: RETORNA DE INTERRUPCCIÓN

        MOVLW   RXINI               ; SI: RESTABLECE BUFFER Y ESCRIBE DATOS
        MOVWF   RXDIR
        MOVWF   FSR
        MOVF    INDF,W              ; LEER DIRECCIÓN DEL DATO
        MOVWF   DIRECCION_RAM       ; ALMACENAR DIRECCION TEMPORALMENTE
        INCF    FSR, F
        MOVF    INDF, W             ; LEE DATO
        MOVWF   DATO_RAM                    ; ALMACANAR DATO TEMPORALMENTE
        MOVF    DIRECCION_RAM, W            ; APUNTA A DIRECCIÓN DE ESCRITURA
        MOVWF   FSR
        MOVF    DATO_RAM, W
        MOVWF   INDF                ; ESCRIBE DATO EN RAM
        BANK2
        BSF     BANDERA_RECEPCION
        GOTO    RETORNO_DE_INTERRUPCION         ; RETORNA DE INTERRUPCIÓN

;*****************************************
;*** FIN DEL FICHERO  DEFRUT.ASM  ***
;*****************************************
```