UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN CONTROLADOR DIFUSO PARA MANTENER UNA PLATAFORMA DE CUATRO ROTORES SUSPENDIDA EN EL AIRE

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO ELÉCTRICO- ELECTRÓNICO

P R E S E N T A N:

ARÓSTEGUI BAUTISTA NOEL

PÉREZ MONTAÑO LUIS MIGUEL

TORRES HERNÁNDEZ JONATAN

DIRECTOR: ING. ROBERTO MACÍAS PÉREZ

MÉXICO. D.F. 2005

Agradecemos a la Universidad Nacional Autónoma de México por permitirnos pertenecer a esta máxima casa de estudios y por abrirnos las puertas del conocimiento para nuestra formación como profesionistas.

A la Facultad de Ingeniería por su compromiso con la formación no solo profesional sino también humana para con sus egresados.

Al ingeniero Roberto Macías por las enseñanzas, tiempo y apoyo brindados para la realización de este trabajo, el cual nos permite concluir una etapa más en nuestra vida.

También queremos agradecer a los ingenieros Eduardo Ramírez, Antonio Salva, Alejandro Sosa y Salvador Zamora por los conocimientos que nos compartieron a lo largo de nuestra carrera, además de la disposición y tiempo que dedicaron a la revisión de esta tesis.

Finalmente a los ingenieros Jesús Gutiérrez, Alejandro Balderas, Leonardo Rivero, Heriberto Meneses, Alfonso Bello y al físico Martín Soria de SATMEX por el apoyo que nos brindaron para la culminación de este proyecto.

Noel Aróstegui Bautista

Quiero dedicar este trabajo de tesis a las dos personas que me han dado todo en la vida y que han hecho de mí el ser humano y el profesionista en que me he convertido: a mi **Padre** y a mi **Madre**.

Por otro lado, existen muchas personas a las cuales quiero agradecer por la importancia que han tenido en mi vida personal y académica:

A mi **Padre**, por ser mi ejemplo de superación y aunque ya no este a mi lado, tendré siempre presente sus enseñanzas, confianza y apoyo incondicional.

A mi **Madre,** por su sacrificio, amor y comprensión en los momentos más importantes.

A mis **hermanos,** por sus consejos, cariño y ejemplo.

A mis **tíos y primos** por confiar y creer en mí.

A todos mis **amigos**, por su cariño y apoyo que de forma directa o indirecta me han ayudado a seguir adelante.

A **Rosalba** por todo su apoyo y amor incondicional, que me motiva a superarme constantemente para ser mejor en todos los aspectos de mi vida.

A la Dirección de Ingenieria de **SATMEX**, que me brindó la confianza para formar parte de su equipo de trabajo y me ha ayudado a crecer como profesionista.

A **Jonatan** y **Luis Miguel** por su amistad y esfuerzo para que juntos pudiéramos lograr una meta más en nuestras vidas.

Luis Miguel Pérez Montaño

Jon y Noel
Gracias por su tesón y creatividad, por esas ganas incansables de resolver los contratiempos que se fueron presentando a lo largo del proyecto, por su compromiso pero sobre todo, por su amistad. Muchas gracias equipo!

Neto, Elena, Gaby, Rick y Oscar
Sin su amor, compañía, ejemplo y esfuerzo no hubiese podido terminar mi carrera. Gracias por el cariño, comprensión y apoyo. Los quiero familia!

Nalle
El amor impulsa, cobija y alimenta a los hombres. Gracias por haberme impulsado, cobijado y alimentado con el tuyo a lo largo de estos maravillosos años. Te amo bonita.

Temo
Justo como lo planeamos hace ya tantos años, he aquí mi tesis terminada. Vaya entonces un agradecimiento a mi compañero de aventuras y metas, mi camarada de ideales, a ti mi hermano, por haberme acompañado en tantos momentos importantes y gracias también por seguir estando aquí.

Armando
El más animado de mis amigos. Siempre has compartido tus logros conmigo, ahora me toca a mi hacerte participe de uno de los míos. Gracias por tu amistad.

Alex, Beto y Leo
Muchas gracias por el apoyo, la paciencia y los conocimientos que han compartido conmigo.

Gracias **Gloria** por su cariñosa paciencia y ricas comidas, a **Alicia, Jess y Rebeca** por su ayuda y cariño, a **Coria** por su amistad y sinceridad, a **Martha** por hacerme sentir de la familia y a todos los que directa o indirectamente apoyaron la realización de esta tesis.

Jonatan Torres Hernández

- He llegado a este momento gracias a ti y al gran esfuerzo que has realizado para sacarnos adelante. Sin tu apoyo nunca lo hubiera logrado, quiero que sepas que eres el pilar más grande que me ha sostenido, siempre serás la fuente de apoyo, amor y comprensión más grande de mi vida. Para ti, **mamá Alicia**.

- Se dice que una madre no sólo es quien trae a la vida, sino quien cría a alguien. A tu lado he crecido, me has dado grandes lecciones y siempre has tenido una gran confianza en mí y en todo lo que hago. Gracias **mamá Rebeca**

- Siempre confiaste en que llegaría este momento, gracias por todo lo que me has enseñado. Para ti, **abuelo Juan**.

- Con ustedes he compartido momentos hermosos, saben que las tres estarán en mi corazón siempre. Para ti, **Jessica**, por que durante más de veinte años has sido más que mi hermana, has sido mi amiga, confidente y consejera, siempre estaré en deuda contigo. A mis pequeñas hermanas **Andrea y Adriana,** estoy muy orgulloso de ustedes, espero que éste logro les sirva de ejemplo, quiero que estén seguras de que las adoro y que siempre estaré a su lado para apoyarlas, como ustedes lo han hecho conmigo.

- Por la ausencia de mi padre, cada uno de ustedes me enseñó de manera diferente como afrontar la vida, me han aconsejado, me han guiado y siempre que los he necesitado han estado para mí. Hoy quiero agradecerles por todo. Gracias a mis tíos **Juan, David y Omar**.

- Siempre he sentido su cariño y confianza, sólo puedo corresponderles ofreciéndoles lo mismo de mi parte. Para **Verónica, Iván y María Fernanda**.

- Para mis primitos **Susana y Joaquín** a quienes les tengo mucho cariño y aprecio.

- Con cariño para mi tía **Rosario** y mis primas **Rocío**, **Charito** y **Alejandra**.

- Este trabajo se lo dedico en especial a mi bisabuelo **Rafael**, quien me ha brindado grandes enseñanzas y a la memoria de mi bisabuela **Alicia**. Ambos siempre confiaron en que lograría llegar a este momento.

- Porque siempre han estado atentos de mis logros y me han dado muchas cosas pero sobre todo algo muy importante: su cariño y amistad. Les dedico este trabajo a mis tíos **Angélica y Gabriel**, **Genoveva y Francisco** y **Ángeles y Rafael**.

- En la familia siempre se forjan lazos de unión y amistad, gracias a **Rafael**, **Malú** y **Cintya**, **Liliana** e **Ingrid** y a **Francisco** por todo lo que hemos convivido juntos en ese lugar tan maravilloso llamado Teocelo.

- A lo largo de mis estudios universitarios y durante el desarrollo de este trabajo conviví con dos personas que han llegado a ser más que mis amigos, son mis hermanos, les agradezco a **Noel** y a **Luis Miguel** por su amistad, comprensión y paciencia.

- Siempre es más fácil sobrellevar las adversidades cuando se tienen amigos como ustedes que estuvieron a mi lado en los momentos difíciles y en las alegrías, cada uno sabe el vínculo que hemos formado, trataré de no olvidar a nadie pero en todo caso disculpen mi memoria. Con cariño para **Rosalba, Roberto, Erika, Leticia, Manuel, Rubén, Izanami, Astrid, Oswaldo, Claudia, Emilio, Temo, Jaziel, Yadira, Yesica, Raúl, María, Mónica, Berenice, Jorge, Hugo y Héctor.**

- A todos mis amigos y amigas de MSD que aunque su amistad es de las más recientes hemos llegado a convivir y a conformar grandes lazos de amistad y convivencia. Para **Stalin, Hazel, Oscar V., Alejandra, Oscar O., Liliana, Adriana I., Adriana M., Patricia, Rosa, Angel, Marcela, Gabino, Margarita, Luis C., Reyna, Luis S. y Valeria.**

- Deseo agradecer a los ingenieros **Arturo Sánchez, Ulises Millán, Carlos Gómez, Rafael Huerta y al físico Ernesto Monroy** que confiaron en mí y me dieron todo su apoyo durante mi estancia en la GIT de PEMEX.

- En especial un agradecimiento a las familias **Aróstegui Bautista y Pérez Montaño** que me abrieron las puertas de sus casas haciéndome sentir parte de ustedes, el cariño que me brindaron no lo olvidaré nunca.

Introducción

La necesidad del ser humano de avanzar tecnológicamente con el fin de resolver los problemas que se presentan a diario, ha llevado a desarrollar nuevos sistemas o bien a mejorar los ya existentes.

Dentro del campo de la ingeniería, es necesario estudiar la teoría que explique el comportamiento de tales sistemas con el fin de proveer soluciones para cada uno de ellos. De tal forma que mientras más se aprende de un sistema, se disminuye su grado de complejidad y se incrementa el entendimiento sobre éste.

En particular, este trabajo de tesis está enfocado a automatizar el vuelo estático horizontal de un mini-helicóptero de radio control al cual llamaremos DIVAC (Dispositivo de Vuelo Auto Controlado) utilizando la teoría de lógica difusa aplicada a un controlador difuso. Este tipo de dispositivos son utilizados en diferentes aplicaciones tales como vigilancia y búsqueda ya que ofrecen fácil acceso a lugares riesgosos donde se expondría la vida de un ser humano.

Debido a la naturaleza no lineal del sistema, se propone realizar las acciones de control mediante un controlador difuso tipo Mamdani proporcional-derivativo (PD) evitando así la obtención de un modelo matemático riguroso que describa fielmente su comportamiento.

Por otro lado, ya que el tiempo en el cual se debe realizar el control del sistema es de suma importancia para mantener el vuelo, el código de programa se estructuró de tal forma que fuera el óptimo permitiendo efectuar las correcciones necesarias para operar en tiempo real. Para ello se utilizó un microcontrolador de última tecnología que por su gran velocidad permitió operar el sistema en tiempo real, calculando y ejecutando de forma oportuna las acciones de control requeridas.

Mediante el control difuso se puede describir el comportamiento deseado en el sistema a través del lenguaje cotidiano. Puesto que una proposición difusa es una declaración que involucra algún concepto sin límites claramente definidos es aceptable cierta cantidad de incertidumbre al realizar el control.

Por ello, se requiere definir a través de variables lingüísticas el grado de inclinación que presenta el DIVAC en un determinado instante, identificando la posición que ocupa la señal de entrada en cada uno de los conjuntos se puede evaluar el grado de pertenencia para cada uno de ellos y así aplicar las reglas convenientes.

La base de reglas está creada a partir del conocimiento y experiencia de un operador en el sistema, están desarrolladas de manera lingüística mediante proposiciones del tipo *Si X entonces Y*. Según las condiciones en las que se encuentra el DIVAC se seleccionan las reglas aplicables a ese caso y de ahí las acciones de control a ejecutarse.

Durante los últimos años, los controladores difusos han sido diseñados para ser utilizados en casi cualquier proceso ya que los podemos encontrar en aplicaciones que van desde productos comerciales tales como cámaras fotográficas y de video, lavadoras y hornos de microondas hasta procesos de control industrial, instrumentación médica y sistemas de decisión y soporte.

Capítulo I Antecedentes

1.1 Introducción a la lógica difusa

1.1.1 Antecedentes de la lógica difusa

La teoría de conjuntos difusos, originalmente introducida por Lotfi Zadeh en el año de 1965, se vale de información aproximada e incierta para generar decisiones certeras, de una manera semejante a como los seres humanos lo hacemos.

La lógica difusa fue diseñada específicamente para representar vaguedades, además de proveer herramientas formales para manejar las imprecisiones intrínsecas en los sistemas; se trata de un proceso de entendimiento que ofrece la posibilidad de que los razonamientos tengan grados de verdad, siendo viable que pertenezca a mas de un conjunto a la vez, a diferencia de la inclusión o exclusión absoluta en un conjunto regido por la lógica tradicional

Este acercamiento a la teoría de conjuntos no fue aplicado a los sistemas de control sino hasta los años 1970's. El profesor Zadeh observó, que para resolver problemas las personas no requieren de entradas de información numérica precisa, y aún así son capaces de un alto nivel de control adaptable.

La lógica difusa es un método de razonamiento en el cual las afirmaciones son parcialmente verdaderas, es decir, pueden tener diferentes grados de verdad. Brinda un acercamiento simple en la solución de problemas al utilizar reglas basadas en el conocimiento de un operador experto en el proceso que se desea controlar, en lugar de intentar modelar el sistema matemáticamente.

Una década después del escrito seminal en conjuntos difusos[1], se generaron en Japón, Europa y Estados Unidos múltiples desarrollos que contribuyeron a la formación de la teoría difusa.

A mediados de los 70's y hasta la fecha, los investigadores japoneses han sido la primera fuerza en el avance de la implementación práctica de la teoría; ellos han hecho un excelente trabajo en la comercialización de esta tecnología y ahora cuentan con más de 2000 patentes en el área.

[1] Dr.Lotfi Zadeh "Fuzzy sets" 1965

1.1.2 Características de la lógica difusa

1.1.2.1 Conjuntos Difusos

Uno de los primeros conceptos que se debe conocer para entender la lógica difusa es el de conjunto difuso. Básicamente, se trata de 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 a la no pertenencia de un elemento en un determinado conjunto, es gradual, y esta transición está caracterizada por funciones de membresía, las cuales les dan flexibilidad para modelar expresiones lingüísticas empleadas cotidianamente.

Un conjunto difuso *A* en *X*, donde *X* es una colección de objetos denotados genéricamente por *x*, se define como el conjunto de pares ordenados mostrado en la ecuación (1.1.2.1).

$$A = \left\{ \left(x, \mu_A(x) \right) \middle| x \in X \right\} \tag{1.1.2.1}$$

donde $\mu_A(x)$ se conoce como la función de membresía o pertenencia de *A*. La función de membresía asigna a cada *x* un grado de pertenencia entre 0 y 1, donde 1 representa la pertenencia total y 0 la ausencia absoluta.

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. En la práctica, cuando el universo de discurso *X* es un espacio continuo, generalmente se divide en varios conjuntos difusos cuyas funciones de membresía cubren a *X* de una manera uniforme. Estos conjuntos se denominan variables lingüísticas y normalmente, se les asignan nombres de adjetivos utilizados en el lenguaje común.

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* denotado por $C=A \cup B$, cuya función de membresía se relaciona con las de *A* y *B* mediante la ecuación (1.1.2.2).

$$\mu_C(x) = \max\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \cup \mu_B(x) \tag{1.1.2..2}$$

Intersección (conjunción): La intersección de dos conjuntos difusos *A* y *B* es un conjunto difuso *C* (escrito como $C=A \cap B$), cuya función de membresía se relaciona con las de *A* y *B* según la ecuación (1.1.2.3).

$$\mu_C(x) = \min\left(\mu_A(x), \mu_B(x)\right) = \mu_A(x) \cap \mu_B(x) \tag{1.1.2.3}$$

Donde el operador max(a,b) devuelve el valor máximo entre a y b y el operador min(a,b) el valor mínimo entre a y b.

Complemento: El complemento de un conjunto difuso *A*, denotado por $\overline{A}$ se define como lo indica la ecuación (1.1.2.4):

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \tag{1.1.2.4}$$

1.1.2.2 Variables Lingüísticas

En el espacio difuso se define un nuevo tipo de variable denominada variable lingüística. Esta se encuentra contenida dentro de los límites del universo del discurso y tiene un número infinito de posibles valores, siendo estos expresados como valores lingüísticos. Este concepto es determinante en la descripción de conjuntos difusos pues permite representar fácilmente conceptos vagos e imprecisos.

Por ejemplo, si se tiene una variable lingüística de entrada llamada "error", a partir de ella podemos crear varios conjuntos difusos particulares para cubrir el universo del discurso y asignarle a cada uno de ellos un valor lingüístico diferente, por lo tanto podríamos definir valores como “pequeño”, “medio”, “grande”, etc.

Una variable lingüística engloba las propiedades de aproximación o conceptos de imprecisión en un sistema y proporciona una forma de razonamiento lógico que facilita la creación y entendimiento de algoritmos computacionales.

1.1.2.3 Funciones de membresía

Las funciones de membresía son representadas mediante expresiones matemáticas. En la implementación de algoritmos difusos se suelen utilizar por simplicidad funciones parametrizadas poco complejas. Las más comunes son:

Función de membresía triangular (lambda).
Una función de membresía triangular (lambda) se especifica mediante tres parámetros $\lambda\{\alpha,\beta,\gamma\}$, según la función (1.1.2.5).

$$\mu_{\lambda}(x) = \begin{cases} 0 & x \le \alpha \\ \dfrac{x-\alpha}{b-\alpha} & \alpha \le x \le \beta \\ \dfrac{x-\alpha}{\beta-\gamma} & \beta \le x \le \gamma \\ 0 & x > \gamma \end{cases} \tag{1.1.2.5}$$

Los parámetros {α,β,γ} (con α<β<γ) determinan las coordenadas en x de los tres vértices de la función de membresía triangular. La representación gráfica se muestra en la figura 1.1.

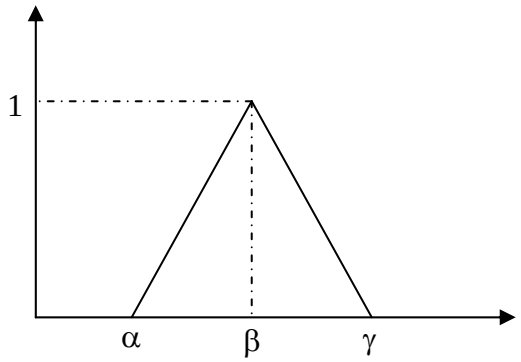


Figura 1.1 Función de membresía triangular

Función de membresía trapezoidal (pi).
Una función de membresía trapezoidal se determina con cuatro parámetros {α,β,γ,δ) como se muestra en la función (1.1.2.6).

$$\mu_{\pi}(x)=\begin{cases}0 & x\le\alpha\\ \dfrac{x-\alpha}{\beta-\alpha} & \alpha\le x\le\beta\\ 1 & \beta\le x\le\gamma\\ \dfrac{x-\delta}{\gamma-\delta} & \gamma\le x\le\delta\\ 0 & x>\delta\end{cases} \qquad (1.1.2.6)$$

Los parámetros {α,β,γ,δ} (con α<β<γ<δ) determinan las coordenadas en x de los cuatro puntos que definen a la función de membresía trapezoidal como se muestra en la figura 1.2.

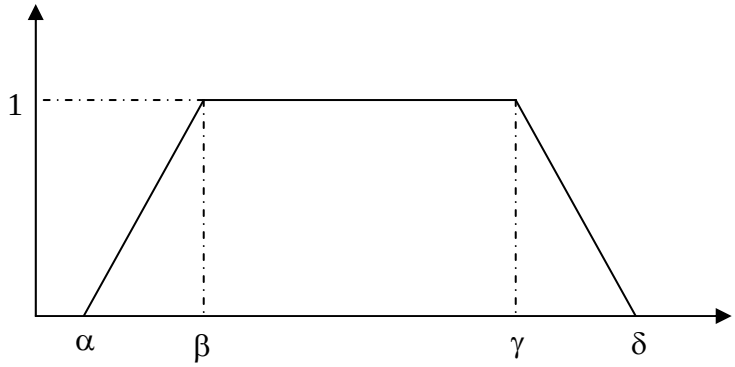


Figura 1.2 Función de membresía trapezoidal

Función singleton.
Este tipo de funciones se usan generalmente a la salida, sólo se define a través de un parámetro α, según la función (1.1.2.7).

$$\mu_{SG}(x) = \begin{cases} 1 & x = \alpha \\ 0 & otro\_caso \end{cases} \qquad (1.1.2.7)$$

Su representación gráfica se muestra en la figura 1.3.

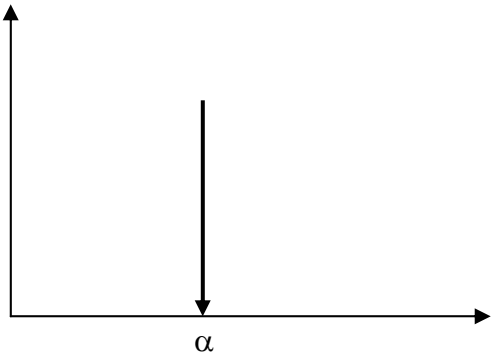


Figura 1.3 Función de membresía singleton

1.1.2.4 Reglas difusas

Para poder realizar las decisiones apropiadas y tomar las acciones necesarias se necesita establecer oraciones completas que definan los estados que se pueden presentar en un sistema. Por lo que se utilizan declaraciones condicionales del tipo *Si ... Entonces,* a las cuales se les conoce como reglas difusas.

Una regla difusa asume la siguiente forma:

Si x es A entonces y es B
o
$A \rightarrow B$

Donde A y B son variables lingüísticas definidas por conjuntos difusos en universos del discurso X e Y respectivamente. A la primera parte de la regla *"A"* se le conoce como antecedente o premisa, mientras que a la segunda parte *"B"* se le llama consecuente o conclusión.

El antecedente puede estar formado por mas de una variable, en tal caso estas se encontrarán relacionadas por los operadores de conjunción o disyunción. De esta forma se podrían generar reglas tales como:

Si x es A y *y es B entonces z es C*

Si x es A o *y es B entonces z es D*

1.1.3 Control difuso

El control difuso es uno de los tipos de control denominados sistemas de control inteligente, mediante el cual un sistema físico o un modelo matemático puede ser controlado por la combinación de una base de conocimiento y la aproximación al razonamiento humano.

En los sistemas de control difuso, un conjunto de reglas difusas representa el mecanismo a través del cual se generan las acciones de control. Uno de los principales objetivos de los sistemas de control difuso es sustituir la habilidad de un operador humano experto por un sistema basado en reglas.

Por tal motivo, las reglas se diseñan basadas en el conocimiento del comportamiento del sistema y no sólo en las predicciones que ofrezca de éste una aproximación matemática.

Una alternativa para implementar un sistema de control difuso de forma similar a las leyes del control convencional, consiste en utilizar un controlador clásico PID y luego utilizar las reglas difusas para sintonizar las ganancias.

1.1.3.1 Controlador Difuso

El proceso a través del cual se realiza el control difuso comienza con la obtención y procesamiento de señales de entrada registradas por transductores conectados al sistema a estas señales se les conoce como datos de entrada crisp. La primera acción realizada por el controlador consiste en transformar estos datos mediante la difusión obteniendo como resultado su equivalente difuso. Una vez que han sido transformados al dominio difuso, los valores pasan a través del proceso difuso, que contiene un conjunto de reglas del tipo *Si ... Entonces*. Como producto de la evaluación de las mismas se generan las salidas difusas que serán antitransformadas nuevamente a valores crisp mediante la desdifusión.

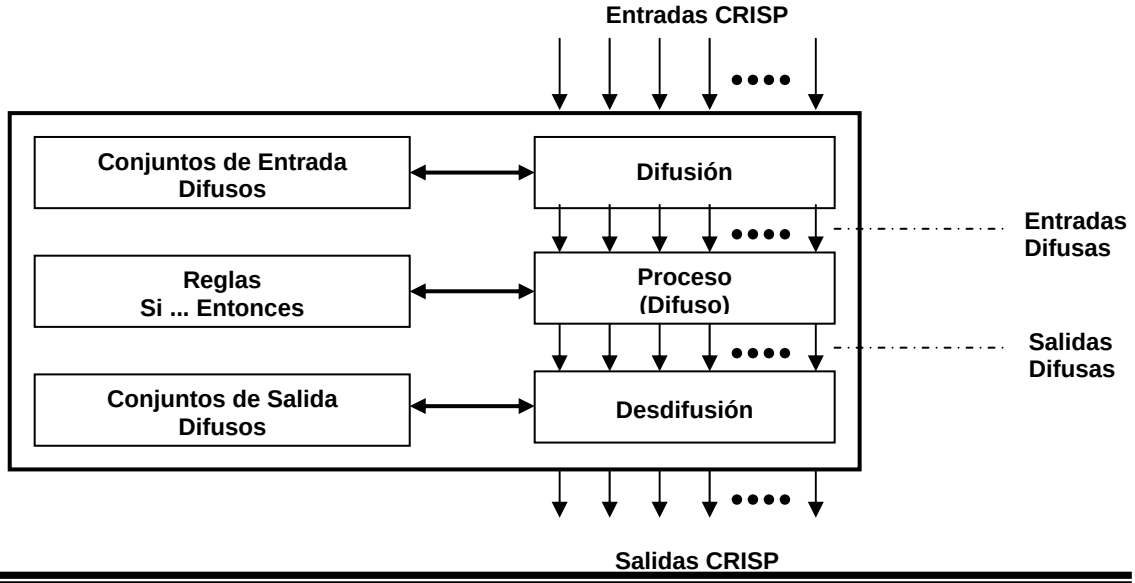


Figura 1. 4 Máquina de inferencia difusa

1.1.3.2 Difusión

Se le llama difusión a la acción de transformar una señal de entrada crisp a una forma lingüística, de esta manera, se le asigna un grado de pertenencia dentro de algún conjunto difuso definido.

En el proceso de difusión, las entradas siempre serán valores numéricos crisp contenidos en el universo del discurso y la salida serán los grados de membresía para los diferentes valores lingüísticos.

1.1.3.3 Evaluación de reglas

Las reglas difusas constituyen las relaciones entre las variables lingüísticas establecidas en un sistema, estas reglas determinan el curso de acción que el controlador debe seguir.

Cuando existen reglas que cuentan con más de un antecedente, se utilizan los operadores difusos MAX, MIN; para simplificarlos a uno solo. Del proceso de difusión, se conoce el grado para el cual cada parte del antecedente ha satisfecho cada regla. Este valor será aplicado posteriormente a la función de salida.

En general, una regla por sí misma no es muy útil dentro del proceso difuso, por tanto, es necesario contar con dos o más reglas relacionadas entre sí. El consecuente de una regla difusa es utilizado para asignar completamente un conjunto de salida; si el antecedente es parcialmente verdadero, entonces el conjunto de salida es truncado de acuerdo a un método de implicación.

El método de implicación es definido como la forma que tendrá el consecuente basado en un antecedente. Este proceso ocurre para cada regla. Dado que la salida de cada regla es un conjunto difuso, es necesario que se calcule un valor único de salida para una colección de reglas. A este proceso se le conoce como composición. Los procedimientos más utilizados para llevarlo a cabo son MAX-MIN y MAX-PROD.

El método MAX-MIN evalúa las magnitudes de los antecedentes para cada una de las reglas asignando el valor mínimo al consecuente de estas. Posteriormente simplifica los consecuentes de cada conjunto difuso de salida seleccionando el mayor como valor único.

El método MAX-PROD multiplica las magnitudes de los antecedentes para cada una de las reglas asignando el producto al consecuente de estas. Posteriormente simplifica los consecuentes de cada conjunto difuso de salida seleccionando el mayor como valor único.

1.1.3.4 Desdifusión

En la desdifusión el valor de la variable lingüística de salida inferida por las reglas difusas es trasladado a un valor crisp de salida. El objetivo es obtener un valor crisp único que represente de la mejor forma los valores difusos inferidos de las variables lingüísticas. La desdifusión es una transformación inversa que convierte la salida del sistema del dominio difuso al dominio crisp.

El método más utilizado para realizar el proceso de desdifusión, es el del Centroide (CoG). Es conocido también como centro de gravedad, debido a que realiza este cálculo en el área compuesta por los conjuntos de salida modificados por el consecuente. De forma discreta, el cálculo para este método se realiza como se muestra en la ecuación (1.1.3.1).

$$CoG = \frac{\sum_{j=1}^{p} y_j \, \mu(y_j)}{\sum_{j=1}^{p} \mu(y_j)} \qquad (1.1.3.1)$$

donde $\mu(y_j)$ es el grado de pertenencia de la función de membresía modificada por el resultado de la inferencia difusa y y_j son las posiciones de los centroides de las funciones de membresía individuales.

Este método presenta dos inconvenientes. El primero se debe a que no todas las funciones de membresía son iguales, por lo que, aquellas con una mayor área tendrán mayor impacto. De hecho, una de las razones al escoger una función de membresía con un área pequeña es para lograr un control más sensible cerca del punto de equilibrio. El segundo se debe a que requiere un esfuerzo computacional considerable para realizar los cálculos de integración requeridos. Cuando se utilizan singletones como conjuntos de salida se reduce significativamente el efecto de ambos inconvenientes.

1.1.4 Ventajas del control difuso

Existen muchas aplicaciones en las cuales se utiliza el control difuso. Algunas de estas son por ejemplo dispositivos de consumo común como videocámaras, lavadoras y artículos domésticos, así como en control de tráfico, trenes, etc. Los beneficios de esta tecnología regularmente pasan inadvertidos para los usuarios ya que los módulos o las funciones difusas se encuentran embebidas dentro de los productos.

Para los diseñadores de sistemas que requieren de control autónomo, la lógica difusa es de gran utilidad ya que puede manejar información aproximada de una manera sistemática, es ideal para controlar sistemas no lineales y sistemas complejos donde existe un modelo inexacto o sistemas donde la ambigüedad o vaguedades son comunes.

Los controladores difusos aprovechan la aproximación al razonamiento humano para utilizar un menor número de operaciones aritméticas, lo cual se traduce en controladores mas rápidos que pueden ser implementados en plataformas de instrucciones reducidas.

En la actualidad la investigación, el desarrollo y la incorporación de nuevos productos al mercado utilizando lógica difusa, continúa creciendo. Debido a la incorporación de microcontroladores más rápidos y con la reducción de precios de los módulos de memoria, la razón costo/beneficio ha disminuido significativamente, además existen ya algunos microcontroladores con módulos de lógica difusa, con lo cual el procesamiento y la programación son más rápidos y sencillos.

1.1.5 Aplicaciones del control difuso

La lógica difusa es empleada en muchas disciplinas. Canon en el año de 1990 lanzó al mercado la primera cámara con autoenfoque controlado por 13 reglas difusas. La primera videocámara difusa fue introducida por Sanyo-Fisher ajustando el enfoque mediante 8 reglas, mediante el análisis del contraste relativo en seis puntos diferentes de la imagen. Panasonic agregó más reglas para eliminar la vibración en la imagen producida por el temblor de la mano. Las reglas difusas discriminan entre el movimiento dentro del cuadro y el del cuadro completo, comparando puntos de movimiento entre el cuadro actual y el anterior

En sus aspiradoras difusas, Samsung utiliza un procesador de cuatro bits que analiza el flujo de polvo detectado por un sensor infrarrojo para determinar cuando se trata de un piso liso o alfombrado, ajustando la potencia de succión según las necesidades. De esta forma se logra un ahorro de energía en un rango del 40 al 80%.

Hitachi, Sanyo, Sharp y Toshiba han diseñado microondas difusos, los cuales miden la luz infrarroja, temperatura, humedad y forma de los alimentos. Las reglas difusas asocian estos parámetros a las posibles condiciones de los alimentos como congelado, precocido, crudo, etc y determinan el mejor ciclo de cocción.

Mitsubishi fabrica un sistema de aire acondicionado difuso que controla los cambios de temperatura de acuerdo a los índices de comodidad humana.

Matsushita construye una máquina lavadora difusa, que combina sensores inteligentes con lógica difusa. Los sensores detectan el color y tipo de ropa presente, así como la cantidad de suciedad, y un microprocesador difuso selecciona la combinación más apropiada de entre 600 combinaciones disponibles de temperatura del agua, cantidad de detergente, y el giro para ciclos de lavado y secado.

La ciudad japonesa de Sendai cuenta con un sistema subterráneo de transporte de 16 estaciones el cual es controlado por un sistema de control difuso. El viaje es tan suave que

los pasajeros no necesitan tener correas para sujetarse, y el controlador comete un 70% menos de errores de juicio en la aceleración y el frenado que un operador humano. Además, en comparación con un control tradicional PID se logró obtener un mejor aprovechamiento en el consumo de potencia eléctrica.

Nissan ha patentado un sistema de transmisión automática difusa, un sistema de frenado anti-derrape y un sistema de inyección. El sistema de inyección ajusta el flujo de combustible basado en la presión del cilindro, temperatura del agua, revoluciones por minuto y mediante la concentración de oxígeno en el ambiente.

El mercado de acciones de Tokio ha tenido por lo menos un portafolio de comercio de acciones basado en lógica difusa que ha superado el cambio promedio del Nikkei. En Japón existen sistemas de diagnóstico de golf, tostadores difusos, limpiadores de vacío difusos y muchos otros procesos de control industriales difusos.

Capítulo II Características del dispositivo de vuelo auto controlado

Debido a la evolución en la tecnología, los sensores, dispositivos de comunicación y microcontroladores cada vez se reducen más en tamaño y peso, incrementando el desarrollo potencial de vehículos no tripulados altamente manejables construidos a pequeña escala. Esta miniaturización ha impactado particularmente a los helicópteros, en los cuales la maniobrabilidad depende principalmente del peso intrínseco de su construcción. Mientras se reducen las dimensiones de estas aeronaves, también decrementan sus momentos de inercia volviéndolos más ágiles pero inestables.

La aplicación de estos vehículos ha aumentado en áreas donde se requieren imágenes aéreas, vigilancia, búsqueda y rescate. Además ofrecen la posibilidad de explorar ambientes riesgosos sin comprometer la vida de un operador a bordo.

Como los sistemas de control autónomo de vuelo para helicópteros representan retos significativos por su naturaleza no lineal e inestable hemos decidido utilizar uno de estos sistemas como plataforma experimental para aplicar nuestros conocimientos en materia de lógica difusa y de sistemas digitales, implementando un controlador difuso como sistema de control para estabilizar el vuelo del mismo.

2.1 Objetivo del proyecto

Diseñar e implementar un controlador embebido basado en la teoría de lógica difusa que sea capaz de mantener estable en posición horizontal, en tiempo real y de forma autónoma un mini-helicóptero de cuatro rotores al cual llamaremos Dispositivo de Vuelo Auto Controlado (DIVAC).

2.1.1 Características de funcionamiento y construcción del DIVAC

Elegimos como sistema a controlar la estructura mecánica del helicóptero de radio control llamado Draganflyer, mostrado en la figura 2.1.

La estructura mecánica de este, está construida a base de cuatro tubos de fibra de carbón de 19 cm de largo y 5 mm de diámetro que se intersecan en un mismo punto de forma perpendicular. En el extremo de cada tubo se encuentra un motor que cuenta con un engrane reductor 5.6:1 que transmite el movimiento a las hélices.

Figura 2.1 Dispositivo de Vuelo Auto Controlado DIVAC

La rotación en cada uno de los motores genera un momento en el cuerpo del helicóptero haciendo que la estructura completa gire en sentido opuesto a la dirección de la hélice.

Para disminuir el efecto de estos momentos, se dispone que el giro de dos hélices opuestas por el vértice se realice en sentido horario y el de las dos restantes en sentido antihorario como se muestra en la figura 2.2. En el caso ideal donde la velocidad de giro para las cuatro hélices es la misma, la suma de los momentos es cero.

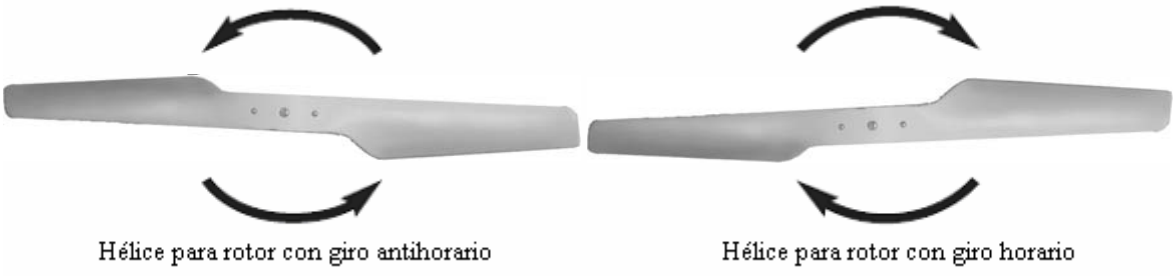


Figura 2.2 Hélices de giro horario y antihorario

Por otro lado, para mantener el DIVAC paralelo a un plano horizontal de referencia, es necesario que la suma de los momentos en dos ejes ortogonales contenidos en este plano sea igual a cero. Estos ejes son colineales a los tubos que sostienen a los motores, lo cual se puede observar en la figura 2.3.

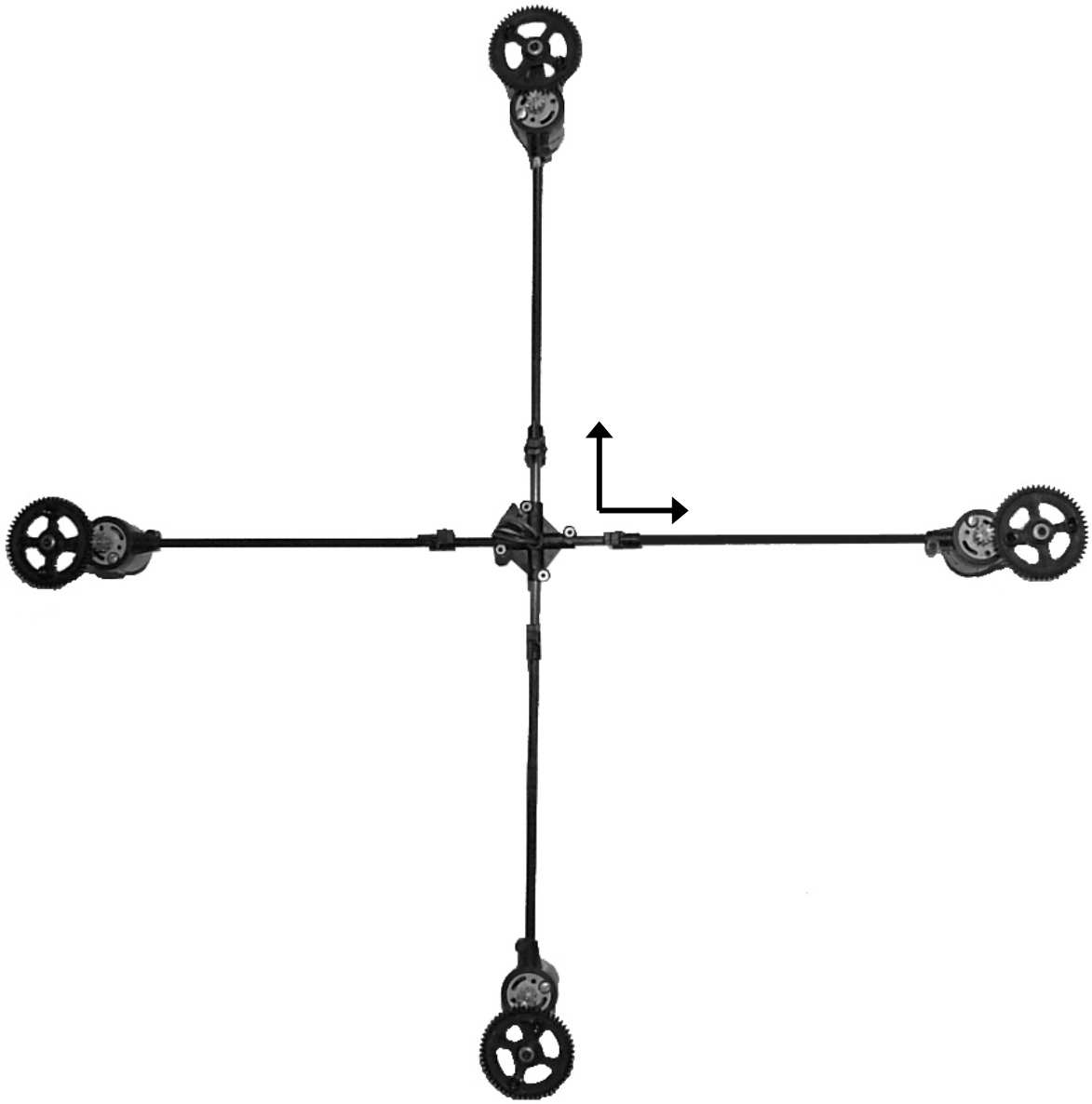


Figura 2.3 Vista superior del DIVAC

Ya que hemos mencionado el hecho de que el tiempo es un factor determinante para generar una respuesta y controlar al DIVAC, es necesario considerar que la construcción del sistema debe cumplir con las características necesarias para que el tiempo de procesamiento sea óptimo.

2.2 Sistemas en tiempo real

Debido a que los microcontroladores cada vez son más eficientes y veloces, su rango de aplicación se ha incrementado en tareas donde se requiere que el tiempo de respuesta para ejecutar una acción específica sea sumamente breve. A este tipo de aplicaciones se les conoce como embebidas o de tiempo real. Algunos sistemas en tiempo real incluyen tecnología de sistemas expertos e inteligencia artificial con lo que se incrementan los requerimientos y la complejidad.

Los sistemas de tiempo real pueden clasificarse en sistemas de tiempo real estrictos y sistemas de tiempo real suaves. Para los primeros, es absolutamente imperativo que se genere una respuesta dentro de un límite de tiempo específico. En los segundos, el tiempo en el que se obtiene una respuesta es importante pero el sistema seguirá funcionando correctamente aunque en ocasiones se rebase el límite establecido.

2.2.1 Características de los sistemas en tiempo real

Estos sistemas poseen las siguientes características:

- Tamaño y complejidad

El diseño de un sistema en tiempo real debe ser lo suficientemente sencillo y capaz de responder a todos los posibles eventos que podrían presentarse durante su operación. Debido a esto, los programas deben ser estructurados de tal manera que su entendimiento y mantenimiento no presenten problemas significativos.

El costo de rediseñar o rescribir el software para responder a los continuos cambios en los requerimientos del mundo real es prohibitivo.

- Confiabilidad y seguridad

Mientras se confíen funciones vitales a los sistemas de control, es necesario que no se produzcan fallas en el software y hardware, ya que esto podría propiciar consecuencias irreparables.

En ambientes hostiles como los encontrados en aplicaciones militares se debe diseñar e implementar sistemas que sólo fallen de manera controlada y en aquellos sistemas donde se requiere de la intervención de operadores humanos, la interfaz debe minimizar la posibilidad de que se presenten errores producidos por estos, así como sus efectos.

- Control concurrente de componentes separados del sistema.

En un sistema embebido existen varios elementos externos interactuando con el código de programa. En el mundo real, estos eventos ocurren de forma paralela o simultanea.
En la mayoría de los sistemas embebidos el programa debe interactuar con un sistema real, en el cual las actividades se realizan siguiendo una secuencia, pero debido a la velocidad con la que estas se ejecutan se logra aparentar que se efectúan de manera simultánea.

- Arquitectura en tiempo real

El tiempo de respuesta es crucial en cualquier sistema embebido. Desafortunadamente es difícil diseñar e implementar sistemas que garanticen que se generará la salida apropiada en el tiempo requerido bajo cualquier posible condición. Debido a esto, los sistemas en tiempo real son construidos utilizando microcontroladores con una mayor capacidad a la requerida, con lo cual se asegura que cuando se presente un comportamiento no deseado, no se produzcan retrasos durante períodos críticos en la operación del sistema.

- Interacción con interfaces de hardware

La naturaleza de los sistemas embebidos requiere que varios componentes interactúen con el mundo exterior, por lo que es necesario monitorear sensores o controlar actuadores. La interfaz se realiza a través de puertos de entrada y salida y sus requerimientos dependen del dispositivo.

- Eficiencia en la implementación

Debido a que los sistemas de tiempo real son críticos con respecto al tiempo, la eficiencia en la implementación es más importante que en otros sistemas. Cuando se utilizan lenguajes de alto nivel el programador se abstrae de los detalles de la implementación y se concentra en resolver el problema de forma global, pasando a segundo termino la eficiencia del código. Por lo que no resulta conveniente utilizar un lenguaje de alto nivel cuando se requiere ejecutar una tarea en el tiempo limite.

- Kernel de tiempo real

Los sistemas embebidos deben proveer un soporte básico para satisfacer de manera predecible restricciones de tiempo y tolerancia a fallas, administrando los recursos físicos necesarios tales como sensores, dispositivos de comunicación, memoria y puertos para utilizar de la manera mas eficiente el tiempo de operación.

Para el diseño de un kernel en tiempo real, se deben considerar los siguientes puntos:

- La variable tiempo debe ser establecida como el elemento principal para el diseño del sistema.

- El balance entre flexibilidad y predictibilidad, con lo cual el sistema debe permanecer siendo lo suficientemente flexible para permitir un ambiente adaptivo, pero al mismo tiempo debe ser capaz de predecir conflictos y evitar posibles problemas con los recursos para que los requisitos de tiempo se alcancen.

Se requiere un acercamiento altamente integrado para la asignación de recursos conforme a los requerimientos en tiempo y así cumplir con la adaptabilidad, exactitud, seguridad y la tolerancia a fallas necesarias en el sistema.

Capítulo III Diseño e implementación del sistema de control

En el presente capitulo se describe el diseño y la implementación de los diferentes sistemas necesarios para lograr el control del DIVAC. El primero de estos sistemas es el de adquisición y acondicionamiento de la señal de posición angular, en este se realiza la adquisición de la posición angular del DIVAC a través de un sensor inercial capacitivo. En el segundo, se detalla el desarrollo del controlador difuso Mamdani tipo proporcional derivativo (PD) que es el encargado de tomar las decisiones y ejecutar las acciones de control. El tercer sistema comprende el arreglo electrónico de semiconductores a través del cual el microcontrolador regula la potencia entregada a los motores. Por otro lado se describen las herramientas de hardware y software desarrolladas como complemento para la implementación del sistema de control; las primeras permitieron la programación y ejecución de código en el microcontrolador mientras que las segundas la simulación y la sintonización del controlador mediante el mapa de inferencia difuso y la superficie de control.
Finalmente se presenta la integración de los sistemas antes mencionados en un solo circuito.

3.1 Sistema de control

En principio, es necesario identificar cada uno de los motores con el fin de determinar la posición angular referida al horizonte de manera individual y así generar la corrección para cada uno. El criterio que se seleccionó está relacionado con la construcción de la estructura del DIVAC, en la cual los tubos de fibra de carbón forman dos ejes ortogonales los cuales se identificarán como X e Y. De esta forma a los motores que se encuentran sobre el eje X se les denomina X1 y X2 y a los ubicados sobre el otro Y1 y Y2 como se aprecia en la figura 3.1.

Las acciones de control necesarias para mantener el DIVAC sustentado en el aire y paralelo a un plano horizontal se logran a partir del controlador difuso tipo PD. Este recibe en un instante determinado la señal de inclinación del mini-helicóptero, la compara con la referencia horizontal y basado en la magnitud del error y la tendencia del mismo, realiza los ajustes necesarios para mantener el sistema en equilibrio. En la figura 3.2 se muestra el diagrama de bloques del sistema de control.

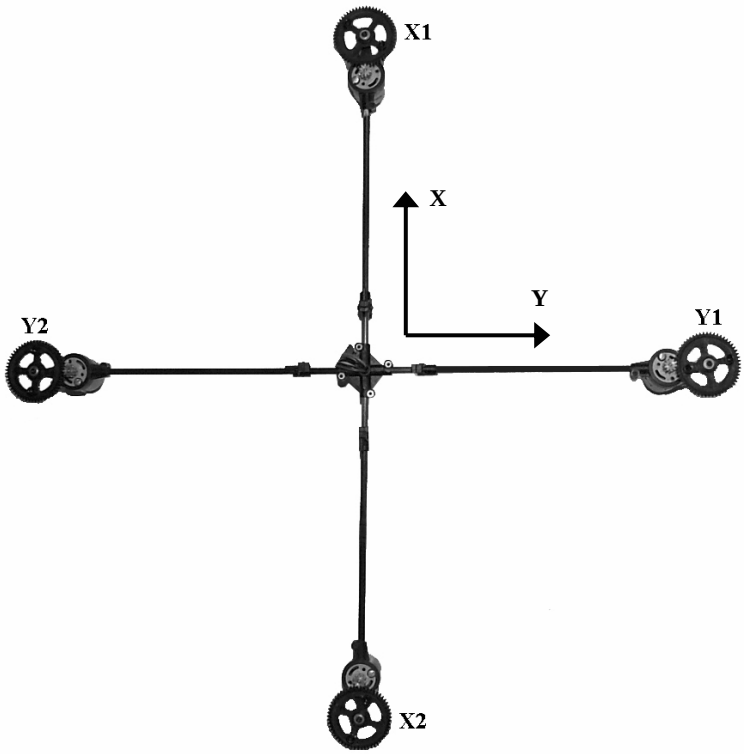


Figura 3.1 Identificación de los motores con respecto a los ejes de referencia

Se eligió el PIC18F452 como plataforma de desarrollo ya que tiene las características necesarias para efectuar el procesamiento de las acciones de control. Cuenta con un conjunto de instrucciones tipo RISC, opera con la frecuencia de reloj máxima de 40 Mhz que se traduce en la capacidad para ejecutar 10,000 instrucciones por segundo. Está basado en arquitectura Harvard con buses independientes para memoria de datos y de programa, así como la estructura pipeline que asegura la ejecución de una instrucción cada cuatro ciclos de reloj. Incluye un módulo de 32k de memoria flash en el cual se pueden almacenar hasta 16384 instrucciones y 1536 bytes de memoria RAM. Es capaz de adquirir señales analógicas a través del convertidor analógico-digital de 10 bits cuya frecuencia de muestreo máxima ideal es de 40.5kHz.

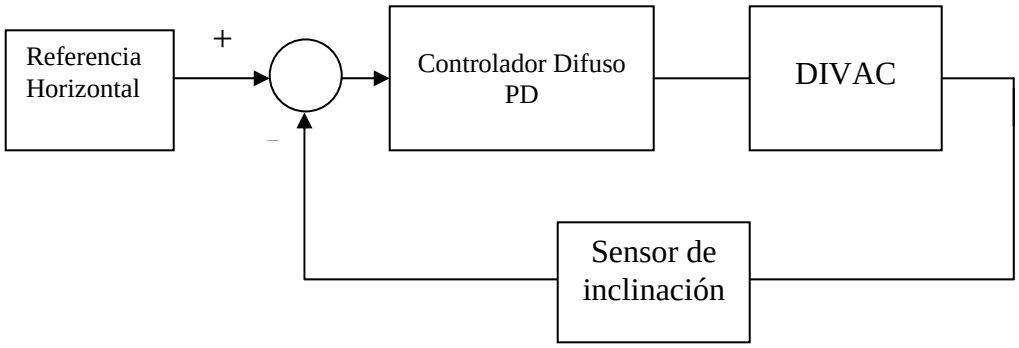


Figura 3.2 Sistema de control

Contiene cuatro temporizadores; a partir de uno de ellos es posible generar 2 señales moduladas por ancho de pulso (PWM). La programación del microcontrolador se realiza por medio de la interfaz serial ICSP entre la PC y el microcontrolador.

El controlador difuso, implementado en el PIC18F452 tendrá como entradas las señales del sensor de inclinación para los dos ejes y como salidas cuatro señales PWM para el control de los motores a través del driver con tecnología MOSFET. En la figura 3.3 se muestra el diagrama funcional de la acción de control.

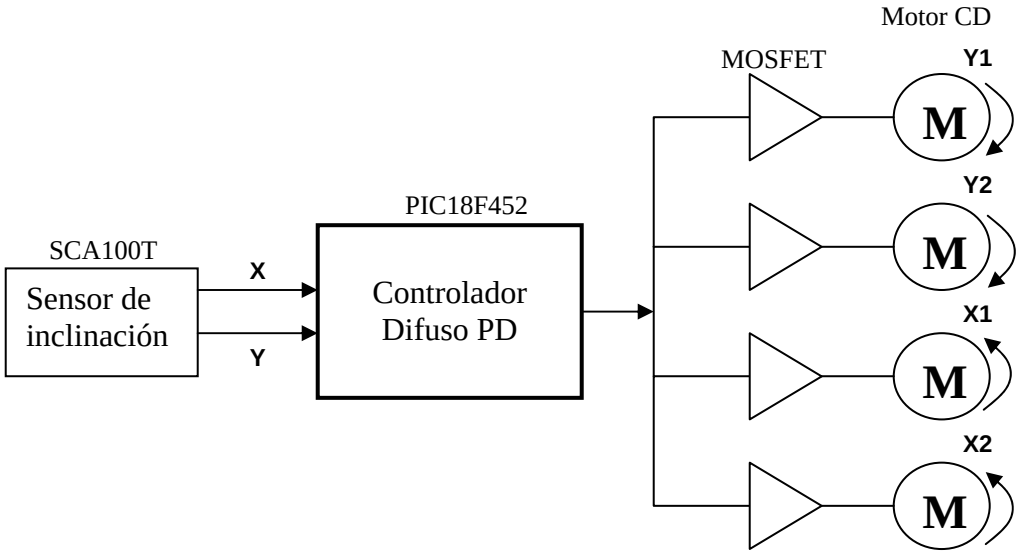


Figura 3.3 Diagrama funcional de la acción de control

Para hacer más eficiente la organización y el diseño del sistema, el proyecto se dividió en cinco módulos complementarios. Estos módulos son:

- Adquisición y acondicionamiento de la señal de posición angular
- Diseño del controlador difuso PD
- Diseño de la interfaz de potencia para los motores
- Diseño de herramientas de hardware y software necesarias para el proyecto
- Implementación Final

3.2 Adquisición y acondicionamiento de la señal de posición angular

Para que el sistema pueda ser controlado es fundamental conocer la posición angular del DIVAC referida a un sistema de coordenadas fijo, esto se logra utilizando el transductor de posición angular a voltaje. Durante el proceso de selección de este fue necesario evaluar parámetros como la disponibilidad, susceptibilidad a interferencia electromagnética, construcción, respuesta en frecuencia, peso, tamaño y costo. Basados en estos criterios, se eligió el sensor de inclinación SCA100T-D01 para adquirir la señal de entrada.

Se trata de un sensor inercial capacitivo, que toma como referencia para la medición del ángulo de inclinación la resultante de la aceleración (g). Es capaz de sensar simultáneamente la inclinación presente en dos ejes ortogonales X e Y. Para cada uno de estos, el rango de medición es de ±30º. La salida es voltaje en el rango continuo de 0 a 5 volts. La relación entre ángulo/voltaje esta dada por la ecuación 3.2.1.

$$\alpha = \arcsin\left(\frac{V_{salida} - 2.5}{4}\right) \tag{3.2.1}$$

El comportamiento del sensor es prácticamente lineal en el intervalo de –30° a 30°, que es el rango de interés para la acción de control.

A partir de la respuesta en frecuencia proporcionada por el fabricante, se puede deducir que el sensor es un sistema de primer orden cuya ganancia se calcula con la ecuación 3.2.2.

$$Ganancia\left(f, f_{3dB}\right) = 20\log\left[\frac{1}{\sqrt{1+\left(\frac{f}{f_{3dB}}\right)^2}}\right] \quad [dB] \tag{3.2.2}$$

Para la frecuencia de corte de 18 Hz, se obtiene la gráfica mostrada en la figura 3.4.

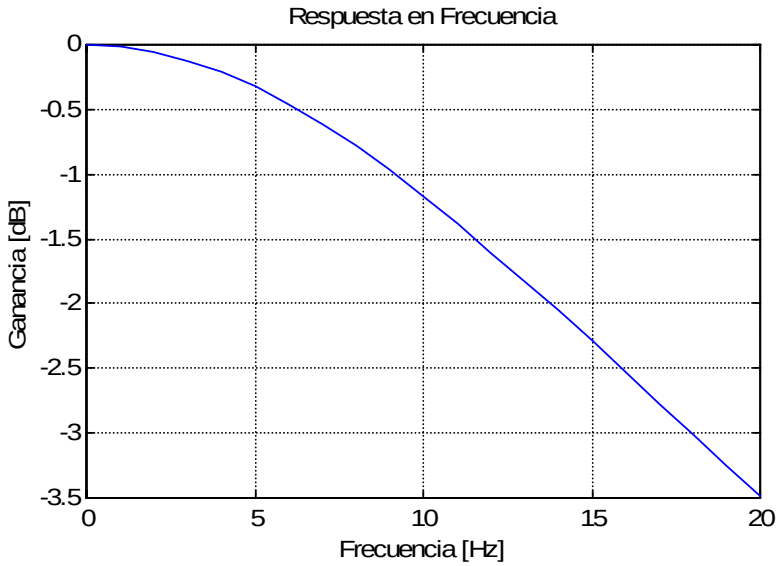


Figura 3.4 Respuesta en frecuencia del sensor

Partiendo de los datos anteriores, la función de transferencia resulta en la ecuación 3.2.3.

$$G(s) = \frac{113.097}{s + 113.097} \tag{3.2.3}$$

Después de aplicar fracciones parciales y antitransformar la función de transferencia, se obtiene la respuesta a escalón quedando la ecuación 3.2.4.

$$G(t) = \left[1 - e^{(-113.097\,t)}\right] u(t) \qquad (3.2.4)$$

Resultando la gráfica de la figura 3.5.en la cual se puede apreciar que el tiempo de asentamiento del sensor es de aproximadamente 20ms,

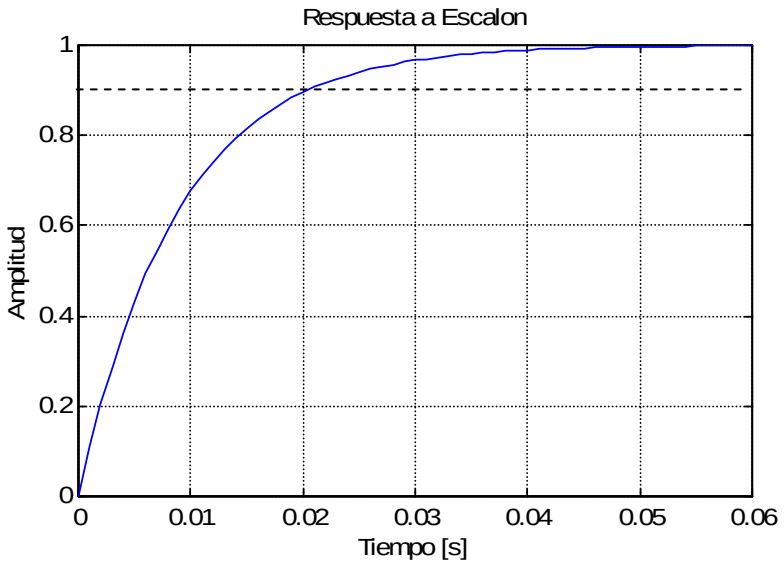


Figura 3.5 Respuesta al escalón del sensor

La información sobre la inclinación del DIVAC en los ejes X e Y se obtiene de las respectivas salidas analógicas del sensor SCA100T-D01 y se adquieren en los canales AN1 y AN2 del convertidor A/D del microcontrolador durante la ejecución del programa. La figura 3.6 muestra a detalle los pines empleados del sensor.

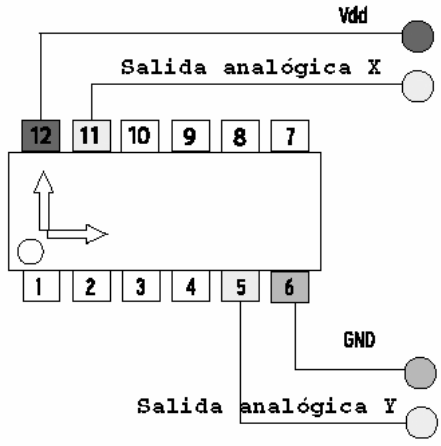


Figura 3.6 Diagrama de pines del sensor de inclinación

Para eliminar la componente de ruido generada por la operación del sensor mismo, el fabricante recomienda filtrar la señal mediante el filtro paso bajas con frecuencia de corte aproximada a 3 kHz.

Basados en el modelo del filtro paso bajas de primer orden y tomando en cuenta que la máxima carga capacitiva permitida es de 20 nF, se empleó la ecuación 3.2.5 para calcular la frecuencia de corte, resultando ser de 3.38 kHz a partir de los valores C=10nF y R=4.7 kΩ.

$$f_{-3dB} = \frac{1}{2\pi RC} \tag{3.2.5}$$

En la figura 3.7 se muestra el diagrama de acondicionamiento de la señal de posición angular para los dos canales analógicos AN1 y AN2.

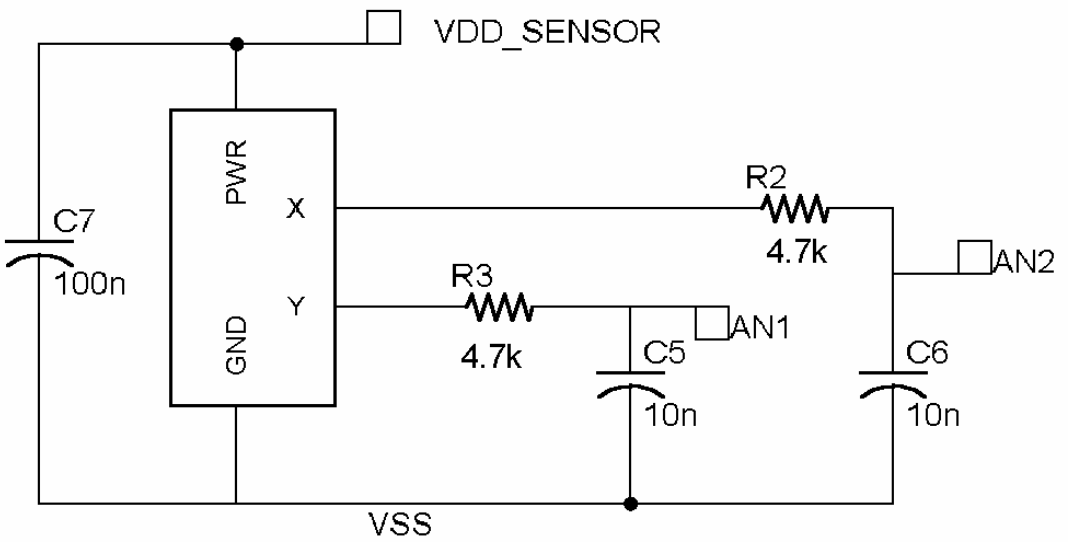


Figura 3.7 Filtro de acondicionamiento de la señal de entrada

El transductor de posición angular es sensible a los cambios de voltaje, por lo que se requiere de voltaje regulado en el rango de 4.75 a 5.25 V.

Como se puede observar en la figura 3.8, cuando el sensor se encuentra en posición horizontal, es decir a 0º, la salida se halla en el punto medio de la escala 2.5 [V], que corresponde al valor digital 80h, representando ángulos negativos los valores inferiores y positivos los superiores

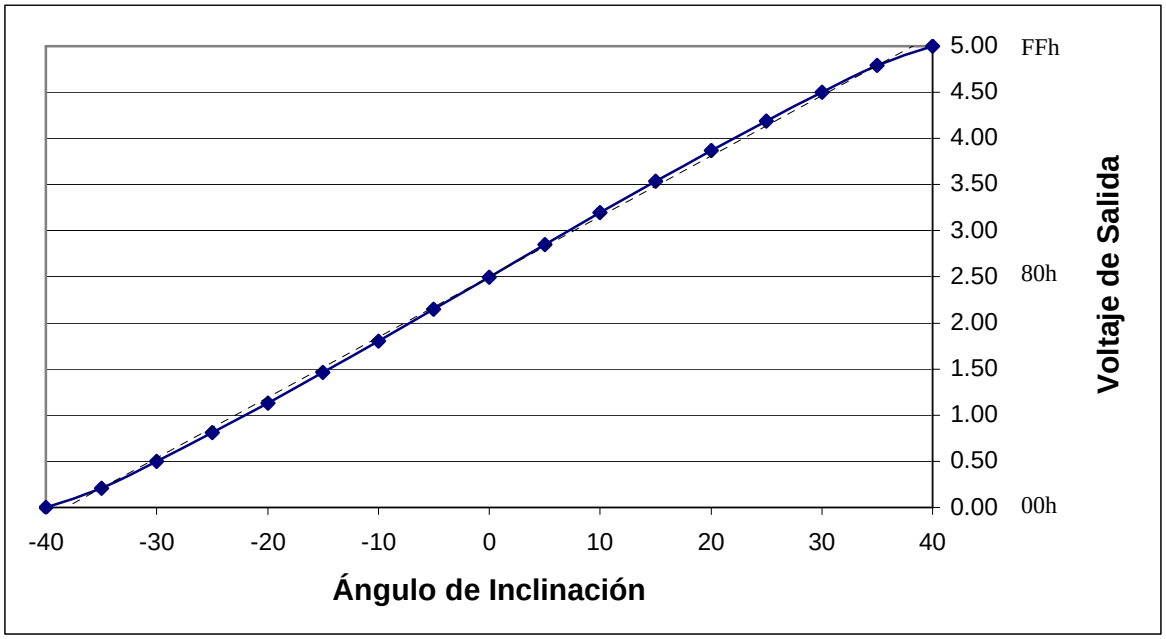


Figura 3.8 Representación de la escala entre ángulo de inclinación y su valor digital

3.2.1 Convertidor Analógico-Digital

El módulo de conversión analógico-digital, es un componente del microcontrolador a través del cual es posible transformar un valor dentro de un rango continuo de voltaje a un valor digital. Dicho módulo cuenta con ocho canales de captura y se configuró para obtener la señal del sensor de inclinación a través de los canales analógicos AN2 para el eje X y AN1 para el eje Y. El tiempo de adquisición por bit se define como T_{AD} y operando a 40 MHz el valor mínimo para éste es de 1.6 μs.

Cuando se ha seleccionado el canal de adquisición, se requiere esperar la carga del capacitor de retención. El tiempo de carga del capacitor se obtiene a partir del modelo mostrado en la figura 3.9 resultando la ecuación 3.2.6.

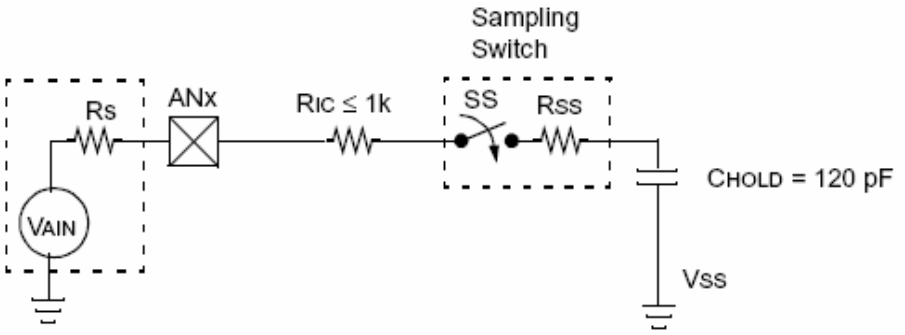


Figura 3.9 Modelo simplificado de la entrada analógica

$$Tc = -(120\,pF)(R_{IC} + Rss + Rs)\ln(1/2048) \qquad (3.2.6)$$

Donde Rss= 5kΩ y R_{IC}= 1 kΩ son proporcionados por el fabricante, mientras que Rs= 4.7kΩ corresponde a la resistencia mostrada en la figura 3.7. Por lo que a partir de estos valores, el tiempo de carga mínimo del capacitor resulta de Tc = 9.8 μs

Además del tiempo de carga del capacitor, es necesario esperar 12 T_{AD} después de iniciada la conversión antes de obtener el resultado. Por lo cual el periodo mínimo de muestreo del sistema es de 29 μs.

Para cumplir estos requisitos en tiempo, la adquisición de las señales analógicas provenientes del sensor ha sido distribuida a lo largo del código de programa, según se muestra en la tabla 3.1.

Operación	Ubicación
Selecciona el canal de captura AN1 y se enciende el convertidor	Inicio de la rutina difusa
Inicia la conversión en el eje Y	Final de la rutina difusa
Se obtiene el valor de la conversión	Inicio de la desdifusión
Selecciona el canal de captura AN2 y se enciende el convertidor	Inicio de la desdifusión
Inicia la conversión en el eje X	Inicio de la división en la desdifusión
Se obtiene el valor de la conversión	Final de la división en la desdifusión

Tabla 3.1 Adquisición de las señales analógicas durante la ejecución del programa

Ya que ha sido obtenido el resultado de la conversión de 10 bits, es justificado a la izquierda, almacenando ocho bits en la localidad ADRESH y los dos bits restantes en la parte alta del registro ADRESL. Por simplicidad y debido a la arquitectura del microcontrolador sólo se opera con los ocho bits contenidos en ADRESH y los otros dos bits son despreciados. Aunque la resolución se ve disminuida en una cuarta parte al utilizar sólo ocho bits, se consideró que la información obtenida es suficiente para esta aplicación.

El proceso de adquisición de las señales provenientes del sensor, se describe en el diagrama de bloques de la figura 3.10.

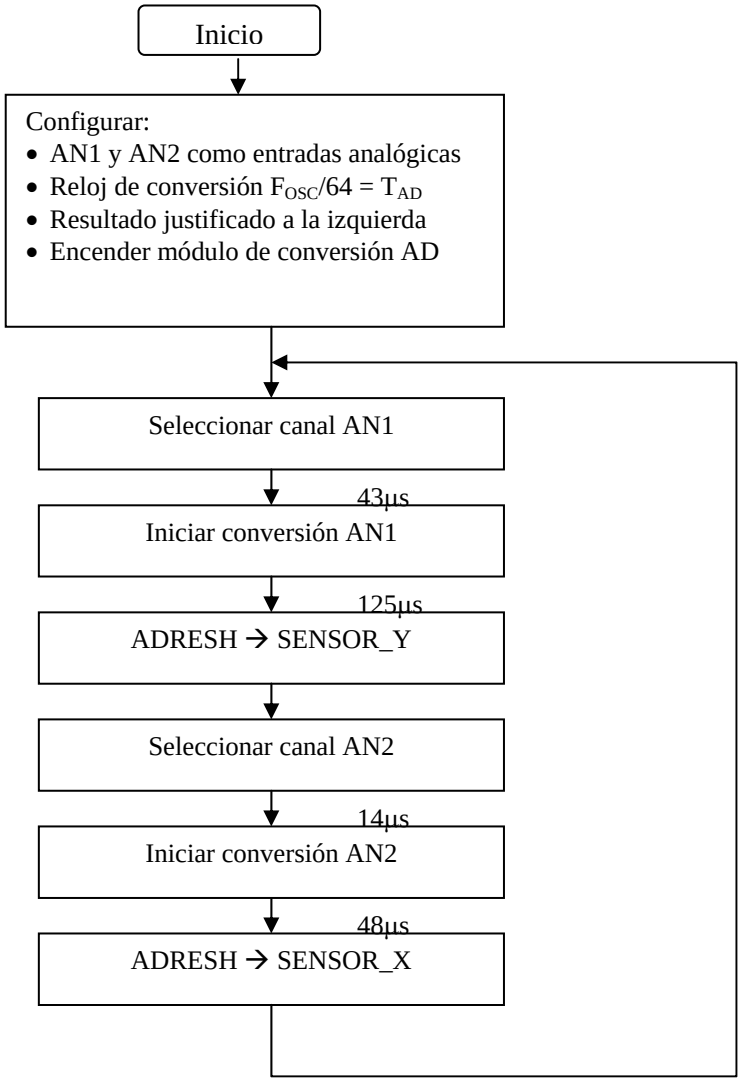


Figura 3.10 Proceso de adquisición

3.3 Diseño del controlador difuso PD

La lógica difusa permite emplear el razonamiento lógico incorporando en los sistemas de automatización esquemas típicamente humanos. Una de las principales características es su capacidad para operar con conceptos vagos e imprecisos, los cuales resultan propios del razonamiento cualitativo. Está fundamentada sobre bases matemáticas que permiten extraer conclusiones cuantitativas a partir de conjuntos de observaciones (antecedentes) y reglas (base de conocimiento).

El control difuso, se ofrece como nueva alternativa en las estrategias de control, de esta forma se establece como decisión o acción final de control la intuición, conocimiento y experiencia humana, permitiendo obtener controles semi-inteligentes.

La aplicación de la lógica difusa en el control de procesos no representa un obstáculo en el empleo de otras técnicas de control convencionales. Por el contrario, la lógica difusa resulta especialmente adecuada para la formulación de controladores híbridos, permitiendo convertir estructuras de control muy diversas.

Se decidió implementar un controlador difuso, con acciones del tipo proporcional-derivativo (PD), procesando como entradas del sistema el error de posición angular y la derivada del error, generando como salida una señal de control PWM.

El controlador del DIVAC es la unión de cuatro controladores difusos idénticos e independientes, cada uno está encargado de corregir la inclinación presente en sólo uno de los cuatro motores. Cada controlador difuso opera sin conocer las acciones de control de los 3 restantes, por tal motivo se requiere ejecutar cuatro de estos subciclos de control difuso para generar el ciclo completo de control como se muestra en la figura 3.11.

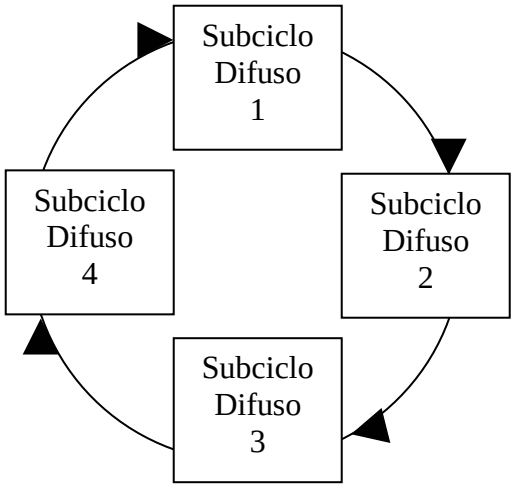


Figura 3.11 Ciclo completo de control

Cada vez que se termina la ejecución de un subciclo difuso, se actualiza la salida generada.

Los subciclos de control están constituidos por el controlador difuso proporcional-derivativo (PD) tipo Mamdani. Este tipo de controladores se caracteriza por utilizar el esquema de inferencia basado en reglas del tipo *Si x es A entonces y es B* donde A y B representan conjuntos difusos en los dominios de entrada y salida respectivamente. La característica principal de este modelo de inferencia reside en que tanto los antecedentes como los consecuentes están expresados de manera lingüística, de tal forma que la información sobre el control puede ser obtenida a partir de la experiencia de operadores humanos.

La máquina de inferencia difusa para cada subciclo difuso del controlador es mostrada en la figura 3.12, donde a las entradas crisp se les ha asignado la variable lingüística de ERROR y DERIVADA DEL ERROR para el error angular y derivada del error angular respectivamente. Después de completar los procesos de difusión, evaluación de reglas y desdifusión, se obtiene como salida crisp el valor de ciclo de trabajo para el control de los motores.

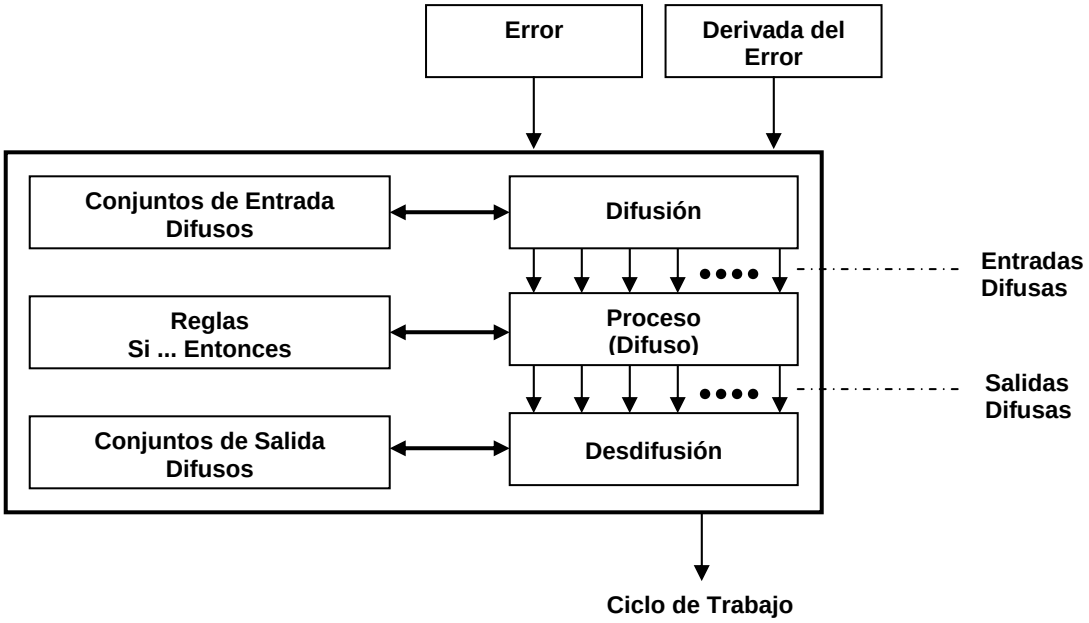


Figura 3.12 Subciclo difuso

Para hacer más eficiente la programación y depuración del código, se dividió el desarrollo del controlador en siete módulos como se muestra en el diagrama de flujo de la figura 3.13.

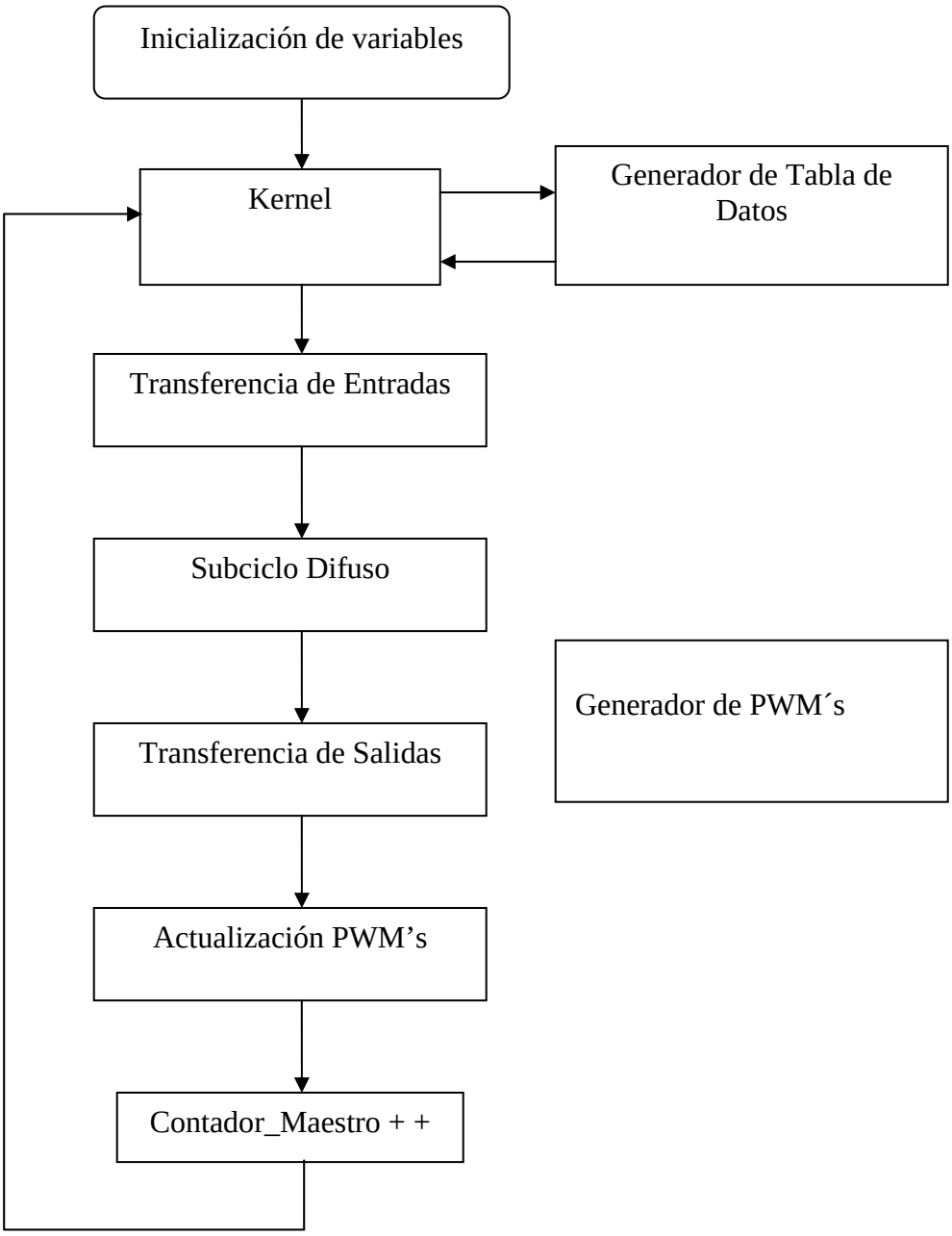


Figura 3.13 Diagrama de flujo del controlador

3.3.1 Kernel

La función de este módulo consiste en administrar los recursos y controlar la ejecución del código para generar el ciclo completo de control.

La secuencia en la que se realizan los cuatro subciclos es modificada y readaptada cada ciclo completo por el kernel, el cual determina cuál de los ejes tiene la mayor aberración y por tanto que motor necesita ser atendido con mayor rapidez.
En el momento en el que el kernel detecta el final del ciclo completo de control ordena que se genere la tabla de datos que contiene la información de errores y derivadas del error

actualizados, con estos datos asigna las prioridades para cada motor. Esta tabla será utilizada durante los cuatro subciclos para generar las salidas de cada uno de los motores.

Esta rutina se describe mediante el diagrama de flujo de la figura 3.14.

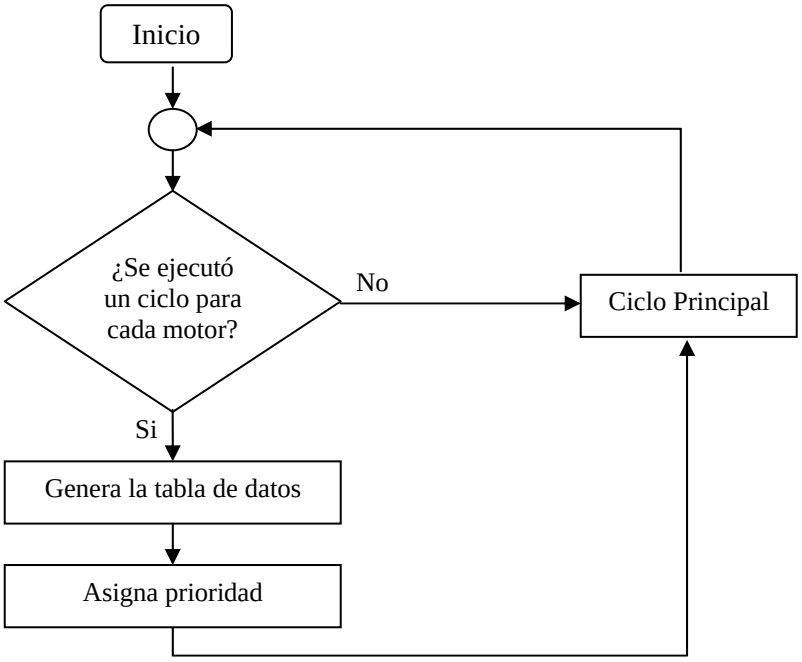


Figura 3.14 Proceso del Kernel

3.3.1.1 Asignación de prioridad

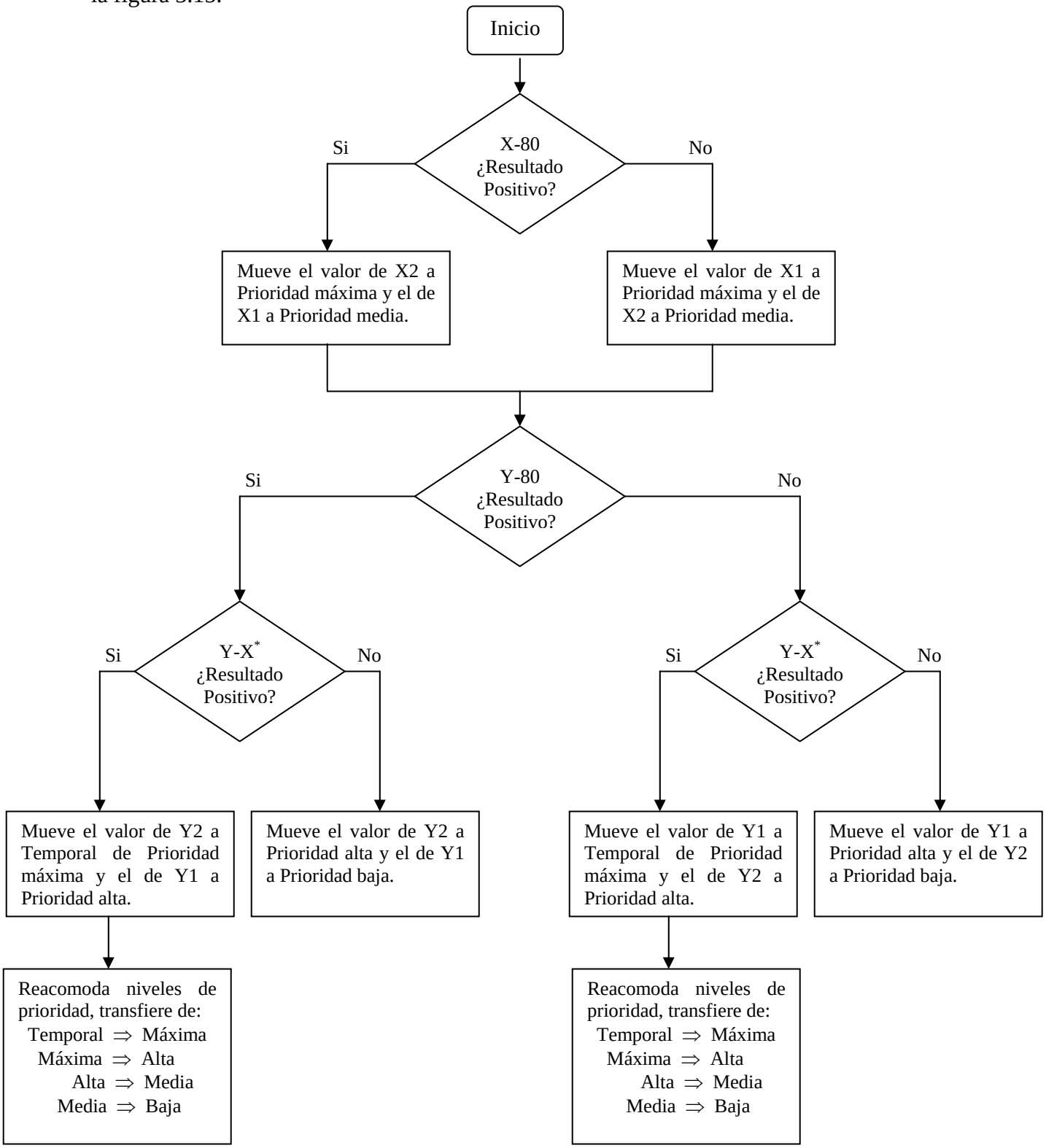
Se desarrolló el algoritmo encargado de establecer el orden en que los motores deben ser corregidos. Este proceso consiste en comparar la magnitud de los errores de la señal de posición angular con respecto a la horizontal para cada motor con el fin de determinar cual de ellos se encuentra más bajo. Basándose en estos valores, se asigna el nivel de prioridad a cada uno de ellos que permitirá establecer el orden en que deberán ser corregidos.

En la tabla 3.2 de apuntadores de ciclo, se almacenan los vectores de dirección donde se encuentran los valores de error para cada motor, con el fin de direccionarlos posteriormente según la prioridad asignada por el kernel.

032h	APUNTADOR_CICLO_0	Temporal de máxima prioridad
033h	APUNTADOR_CICLO_1	Prioridad máxima
034h	APUNTADOR_CICLO_2	Prioridad alta
035h	APUNTADOR_CICLO_3	Prioridad media
036h	APUNTADOR_CICLO_4	Prioridad baja

Tabla 3.2 Dirección en memoria y prioridades para la tabla de apuntadores de ciclo

La asignación de prioridad se describe con mayor detalle mediante el diagrama de flujo de la figura 3.15.



X^* se refiere al valor de mayor magnitud de X seleccionado anteriormente, según el caso que se haya presentado.

Figura 3.15 Funcionamiento de la rutina Asignación de Prioridad

3.3.2 Generador de la tabla de datos

En esta etapa se realizan dos diferentes acciones:

- Actualización de las señales de entrada.
- Cálculo de la Derivada del Error.

1.- Actualización de las señales de entrada.

Para conocer y predecir el comportamiento del DIVAC es necesario contar con los valores actuales y anteriores del error de cada uno de los motores, por lo cual deben ser almacenados en memoria RAM y reactualizados antes de cada ciclo completo de control.

En la tabla 3.3 se muestra la estructura empleada para almacenar la información de los valores anteriores y actuales reportados por el sensor de inclinación sobre la posición del DIVAC.

020h	ERROR_ACTUAL_X1
021h	DERIVADA_DE_ERROR_X1
022h	INDICE_X1
023h	ERROR_ANTERIOR_X1
024h	ERROR_ACTUAL_X2
025h	DERIVADA_DE_ERROR_X2
026h	INDICE_X2
027h	ERROR_ANTERIOR_X2
028h	ERROR_ACTUAL_Y1
029h	DERIVADA_DE_ERROR_Y1
02Ah	INDICE_Y1
02Bh	ERROR_ANTERIOR_Y1
02Ch	ERROR_ACTUAL_Y2
02Dh	DERIVADA_DE_ERROR_Y2
02Eh	INDICE_Y2
02Fh	ERROR_ANTERIOR_Y2

Tabla 3.3 Dirección en memoria de los valores de error

Para realizar la actualización de las señales de entrada, primero se transfieren los valores contenidos en las localidades ERROR_ACTUAL_X1 y ERROR_ACTUAL_X2 a ERROR_ANTERIOR_X1 y ERROR_ANTERIOR_X2, para después actualizar ERROR_ACTUAL_X1 con la señal proveniente del sensor y ERROR_ACTUAL_X2 con el complemento. Este procedimiento se realiza de igual forma para el eje Y.

2.- Cálculo de la Derivada del Error.

Dado que el sistema es discreto, la derivada del error es la diferencial del error entre la unidad de tiempo como se muestra en la relación 3.3.2.1.

$$\frac{\Delta E}{T} \tag{3.3.2.1}$$

Donde:
ΔE = Error Actual - Error Anterior
T = Periodo de muestreo = 1u

Las variables Error Actual y Error Anterior están contenidas en el intervalo [0 , 255] , ΔE puede tomar valores en el intervalo [-255 , 255], pero la variable Derivada del error esta restringida al intervalo [0 , 255] del cual los valores superiores a 127 representan derivadas positivas y los menores a la mitad del intervalo derivadas negativas cuya magnitud aumenta mientras mas se alejen del punto medio. Por tal motivo se debe agregar el offset positivo de magnitud 127 y limitar la magnitud absoluta de |ΔE| = 127 aplicando a esta la siguiente función:

$$\dot{E}(\Delta E) = \begin{cases} 0 & \Delta E <= -127 \\ \Delta E + 127 & -127 < \Delta E < 127 \\ 255 & \Delta E >= 127 \end{cases} \tag{3.3.2.2}$$

Las operaciones realizadas en la rutina generador de la tabla de datos se muestran mediante el diagrama de bloques de la figura 3.16.

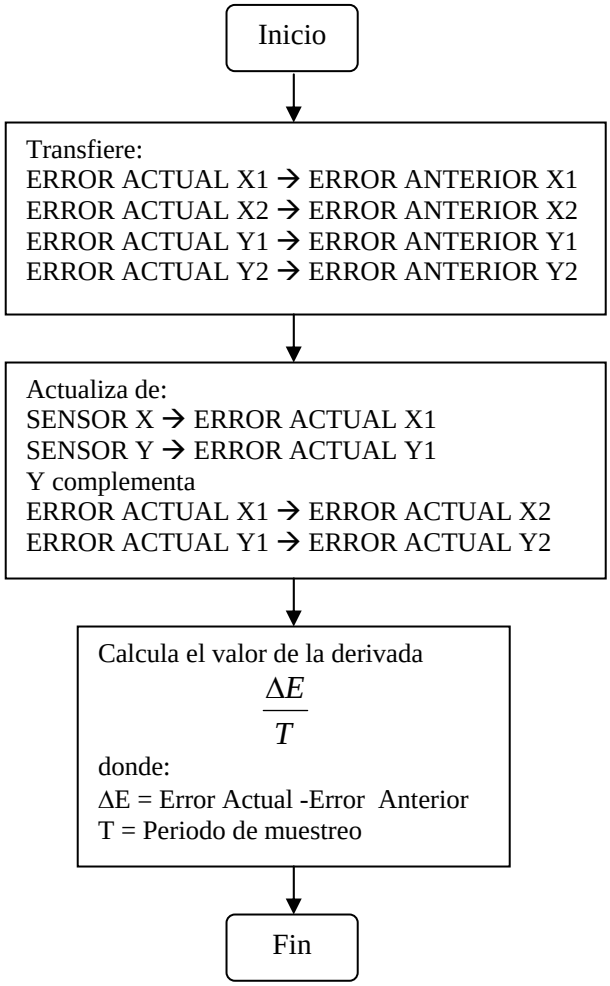


Figura 3.16 Diagrama de bloques para generar la tabla de datos

3.3.3 Transferencia de entradas

Este módulo se encarga de proporcionar los valores actualizados de las dos entradas que serán procesadas en cada subciclo difuso, según la prioridad asignada en la tabla apuntadores de ciclo.

Para ejemplificar esta rutina se muestra la tabla 3.4 de apuntadores de ciclo donde se encuentran contenidas las direcciones de memoria correspondientes a los errores para cada motor. Se observa a partir de esta tabla de datos, que la prioridad máxima está asignada al motor X2 que será el primero en ser atendido, por lo que se transfieren los valores de ERROR_ACTUAL_X2 y DERIVADA_DE_ERROR_X2 como entradas del subciclo difuso. Según el orden de la tabla, se realiza el mismo procedimiento para Y2, X1 e Y1 en los subciclos difusos posteriores.

APUNTADOR_CICLO_1	024h	Prioridad máxima
APUNTADOR_CICLO_2	02Ch	Prioridad alta
APUNTADOR_CICLO_3	020h	Prioridad media
APUNTADOR_CICLO_4	028h	Prioridad baja

020h	ERROR_ACTUAL_X1
021h	DERIVADA_DE_ERROR_X1
022h	INDICE_X1
023h	ERROR_ANTERIOR_X1
024h	ERROR_ACTUAL_X2
025h	DERIVADA_DE_ERROR_X2
026h	INDICE_X2
027h	ERROR_ANTERIOR_X2
028h	ERROR_ACTUAL_Y1
029h	DERIVADA_DE_ERROR_Y1
02Ah	INDICE_Y1
02Bh	ERROR_ANTERIOR_Y1
02Ch	ERROR_ACTUAL_Y2
02Dh	DERIVADA_DE_ERROR_Y2
02Eh	INDICE_Y2
02Fh	ERROR_ANTERIOR_Y2

Tabla 3.4 Ejemplo de transferencia de entradas

3.3.4 Subciclo difuso

Esta rutina utiliza los valores actualizados por el módulo de transferencia de entradas operándolos de forma independiente mediante los procesos de difusión, evaluación de reglas y desdifusión. Está diseñada para aceptar hasta tres entradas crisp que pueden representarse en el dominio difuso con hasta ocho conjuntos de entrada y sólo una salida de ocho conjuntos.

Los conjuntos difusos de entrada, salida y las reglas están almacenados en la memoria Flash y son transferidos a la memoria RAM al inicio de la ejecución del código, residiendo a partir de la localidad 0100h y hasta la 01FBh.

3.3.4.1 Difusión

Para poder llevar a cabo este proceso, primero se definieron siete conjuntos de entrada difusos para cada una de las dos entradas crisp del sistema. Dichos conjuntos son funciones de membresía triangulares y conjuntos de forma trapezoidal en los extremos del universo del discurso, dispuestos de tal forma que cualquier valor crisp pertenezca como máximo a dos conjuntos adyacentes o a un único conjunto con pertenencia unitaria.

En las figuras 3.17 y 3.18 se muestran los conjuntos difusos para las variables ERROR y DERIVADA DEL ERROR, así como sus valores lingüísticos.

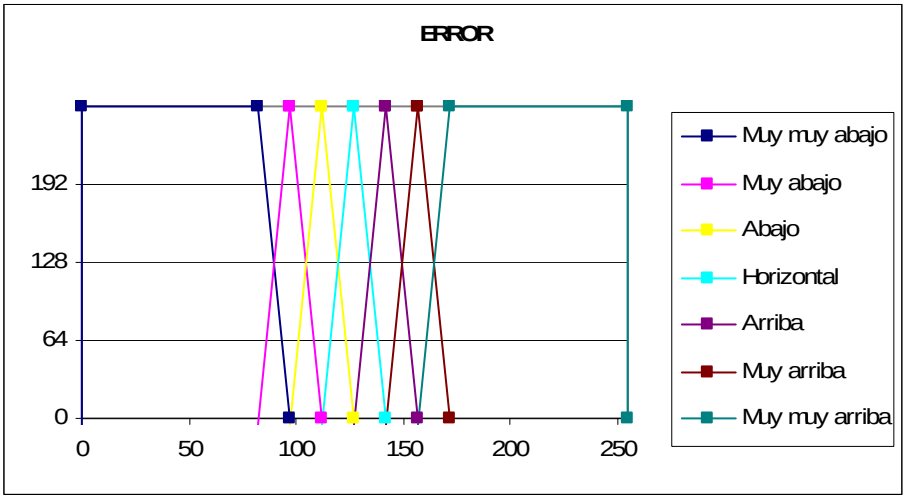


Figura 3.17 Representación gráfica de los conjuntos difusos del error

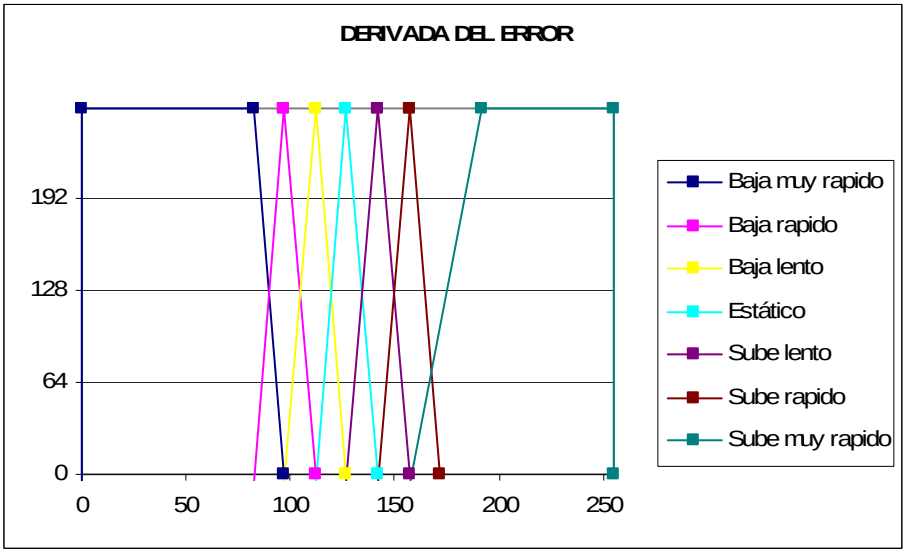


Figura 3.18 Representación gráfica de los conjuntos difusos de la derivada del error

Cada conjunto se define mediante dos puntos y dos pendientes.

Para transformar los valores crisp al dominio difuso, es necesario segmentar los conjuntos como se muestra en la figura 3.19 y utilizar el método de punto pendiente.

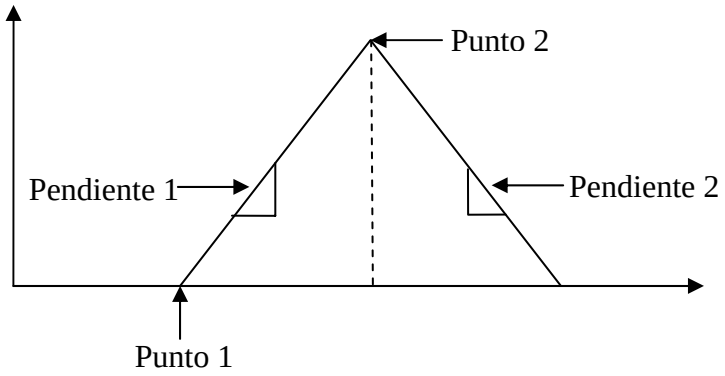


Figura 3.19 Segmentación punto pendiente de los conjuntos difusos de entrada

Después de que se ha seleccionado la entrada crisp que va a ser procesada, esta debe evaluarse para cada uno de los siete conjuntos. La pertenencia para cada conjunto se realiza comparando el valor de entrada crisp con el punto 2 de cada conjunto efectuando la resta. Si el resultado de esta operación es positivo entonces lo multiplica por el valor de la pendiente M2, si el producto resultante es mayor a FFh entonces se asigna el grado de pertenencia de 00h; en el caso contrario se calcula el grado de pertenencia según la ecuación 3.3.4.1.

$$\mu_x = MAX\left[FF - (X_i - PT2)\right] \qquad (3.3.4.1)$$

Cuando el resultado de la resta es negativo, se efectúa nuevamente esta operación tomando como sustraendo el punto 1. En este caso asigna el valor de membresía de FFh si la pendiente es igual a 00h ó realiza el cálculo del valor de membresía según la expresión 3.3.4.2.

$$\mu_x = MIN\left[X_i - PT1\right] \tag{3.3.4.2}$$

Este proceso de comparación es realizado de la misma forma para todos los conjuntos difusos definidos y una vez que se han obtenido los grados de membresía para cada entrada crisp, estos valores son colocados en la tabla de entradas difusas.

La tabla 3.5 puede representarse como la matriz, donde la cantidad de columnas está relacionada con el número de entradas crisp y los renglones con los conjuntos difusos de entrada.

Localidad	Índice relativo	Entrada Crisp 1
50h	0000b	Grado de membresía para el Conjunto 1
51h	0001b	Grado de membresía para el Conjunto 2
52h	0010b	Grado de membresía para el Conjunto 3
53h	0011b	Grado de membresía para el Conjunto 4
54h	0100b	Grado de membresía para el Conjunto 5
55h	0101b	Grado de membresía para el Conjunto 6
56h	0110b	Grado de membresía para el Conjunto 7
57h	0111b	Grado de membresía para el Conjunto 8

Localidad	Índice relativo	Entrada Crisp 2
58h	1000b	Grado de membresía para el Conjunto 1
59h	1001b	Grado de membresía para el Conjunto 2
5Ah	1010b	Grado de membresía para el Conjunto 3
5Bh	1011b	Grado de membresía para el Conjunto 4
5Ch	1100b	Grado de membresía para el Conjunto 5
5Dh	1101b	Grado de membresía para el Conjunto 6
5Eh	1110b	Grado de membresía para el Conjunto 7
5Fh	1111b	Grado de membresía para el Conjunto 8

Tabla 3.5 Representación de la tabla de ENTRADAS_DIFUSAS

3.3.4.2 Evaluación de Reglas

La base de conocimiento para controlar el funcionamiento del DIVAC, esta conformada por el conjunto de 49 reglas creadas mediante dos antecedentes y un consecuente.

El antecedente y el consecuente se representan como un byte de información de la siguiente forma:

7	*6*	*5*	*4*	*3*	*2*	*1*	*0*
C	0	0	X	X	A	A	A

Bits 2-0 **AAA**: representa el índice relativo del grado de membresía para los conjuntos de entrada cuando se trata del antecedente o para los conjuntos de salida cuando es el consecuente.

AAA	
000	Grado de membresía para el conjunto de entrada/salida 1
001	Grado de membresía para el conjunto de entrada/salida 2
:	:
111	Grado de membresía para el conjunto de entrada/salida 8

Bits 4-3 **XX**: representa el número de entrada crisp para la que se está realizando la evaluación.

XX	
00	Entrada 1
01	Entrada 2
10	Entrada 3

Bits 5-6: sin uso.

Bit 7 **C**: indica que es el consecuente cuando es 1 ó el antecedente cuando es 0.

Para realizar la evaluación de las reglas son considerados principalmente los índices relativos asignados a los valores de entrada difusos. Mediante estos, el controlador carga el valor de las entradas difusas y los compara entre si, seleccionando el menor. Este valor es direccionado mediante el índice del consecuente para colocarlo en la tabla 3.6 denominada SALIDAS_DIFUSAS, la cual contiene los valores de las salidas difusas.

Cuando alguna regla implica el mismo consecuente, se actualiza el valor de la salida difusa estableciendo el mayor entre estos. Esto se debe a que el proceso de evaluación de reglas se realiza mediante el método MAX-MIN.

Localidad	Índice relativo	Salidas Difusas
60h	1000 0000b	Salida difusa 1
61h	1000 0001b	Salida difusa 2
62h	1000 0010b	Salida difusa 3
63h	1000 0011b	Salida difusa 4
64h	1000 0100b	Salida difusa 5
65h	1000 0101b	Salida difusa 6
66h	1000 0110b	Salida difusa 7
67h	10000111b	Salida difusa 8

Tabla 3.6 Representación de la tabla SALIDAS_DIFUSAS

Para determinar cuando finaliza el proceso de evaluación de reglas, se carga el valor del apuntador de tabla después de cada consecuente. Si este valor es igual a FFh, se da por concluida la evaluación.

3.3.4.3 Desdifusión

En este proceso se definen siete conjuntos de salida para cubrir el universo del discurso de la variable lingüística Potencia según la figura 3.20.

A diferencia de los conjuntos de entrada, estos se definen mediante la función de membresía tipo singleton y se les han asignado los siguientes valores lingüísticos:

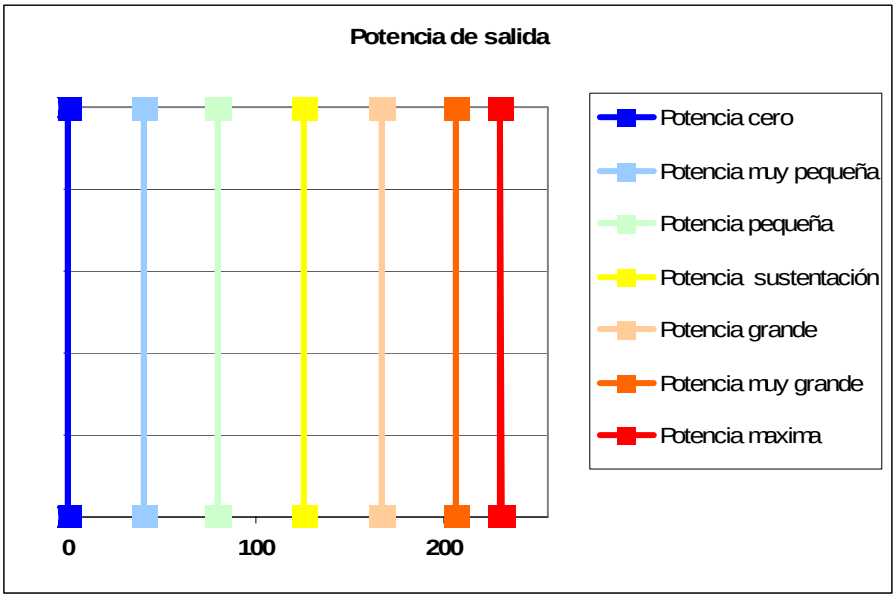


Figura 3.20 Representación gráfica de los conjuntos difusos de salida

Para realizar el proceso de desdifusión se utilizará el método de centro de gravedad en el cual se realizan operaciones de suma, multiplicación y división de forma sistemática.

De acuerdo al método de centro de gravedad, el valor discreto de la salida crisp se obtiene de la ecuación 3.3.4.3.

$$CoG = \frac{\sum_{j=1}^{p} y_j \ \mu(y_j)}{\sum_{j=1}^{p} \mu(y_j)} \qquad (3.3.4.3)$$

donde $\mu(y_j)$ representa los grados de membresía para las salidas difusas los cuales se encuentran contenidos en la tabla de salida difusa y y_j representa los valores de la variable lingüística Potencia almacenados en la tabla de singleton. El resultado de la desdifusión se almacena en la variable de salida crisp.

Debido a que la arquitectura del microcontrolador es de 8 bits, se requiere emplear más de una localidad de memoria de datos para almacenar los operandos que van a ser utilizados.

Ya que el método de desdifusión seleccionado requiere ejecutar la operación de división, y dado que en el PIC18F452 no se encuentra implementada esta operación, fue necesario diseñarla a partir de las instrucciones disponibles, considerando que el valor máximo del numerador que se puede generar es de 24 bits mientras que el del denominador es de 16 bits, lo cual resulta en el cociente de 8 bits.

23 — 16	15 — 8	7 — 0
NUMERADOR_SUPERIOR	NUMERADOR_MEDIO	NUMERADOR_INFERIOR

15 — 8	7 — 0
DENOMINADOR_ALTO	DENOMINADOR_BAJO

Tabla 3.7 Representación del numerador y denominador en localidades de memoria

3.3.5 Transferencia de salida

Se encarga de almacenar de forma organizada el valor del ciclo de trabajo que va a ser entregado al módulo que genera los PWM's. Los resultados son obtenidos de la desdifusión y se almacenan en la tabla de PWM.

La organización depende directamente del valor de prioridad asignado por el kernel.

TABLA_PWM	Ciclo de trabajo para el motor X1	PWM0
	Ciclo de trabajo para el motor X2	PWM1
	Ciclo de trabajo para el motor Y1	PWM2
	Ciclo de trabajo para el motor Y2	PWM3

Tabla 3.8 Transferencia de salida

3.3.6 Actualización de PWM's

En esta última etapa se transfieren los valores del ciclo de trabajo almacenados en la tabla de PWM a los registros encargados de generar los PWM's.

PWM0	→	PWM_INTERRUP_3
PWM1	→	CCPR1L
PWM2	→	PWM_INTERRUP_1
PWM3	→	CCPR2L

Tabla 3.9 Actualización de PWM's

3.3.7 Generador de PWM´s

Debido a que la acción de control requiere de cuatro señales de PWM y que el microcontrolador sólo es capaz de generar dos a través del módulo de PWM embebido es necesario implementar una rutina cuasi paralela que genere las dos restantes con las mismas características de las dos obtenidas por hardware y sin afectar el tiempo de ejecución del código principal.

El módulo interno de PWM utiliza el temporizador 2 y tiene la opción de configurarse para una frecuencia específica según se requiera. En nuestro caso, se configuró para operar a la frecuencia de 2.44 kHz con periodo de FFh y prescalador de 1:16. El ciclo de trabajo se actualiza en los registros CCPR1L y CCPR2L a partir de los valores contenidos en TABLA_PWM.
Las señales moduladas por este hardware se generan en los bits 1 y 2 del puerto C.

Para las señales generadas por software se utilizan tres temporizadores y mediante interrupciones se obtiene la señal modulada, de tal forma que la unidad aritmética lógica del microcontrolador quede disponible para continuar con la ejecución del resto del programa.

Se configuró el temporizador 0 para ser utilizado como el periodo base de la señal modulada; los temporizadores 1 y 3 fungieron como comparadores a los cuales se les

asignó el valor del ciclo de trabajo. Cuando se produce la interrupción por desbordamiento del temporizador 0, la rutina de servicio asociada activa la señal de PWM a nivel alto manteniéndola hasta que se produce la interrupción en cualquiera de los comparadores, con lo cual se cambia el estado de la señal a nivel bajo. La señal generada es colocada en los bits 0 y 3 del puerto C.

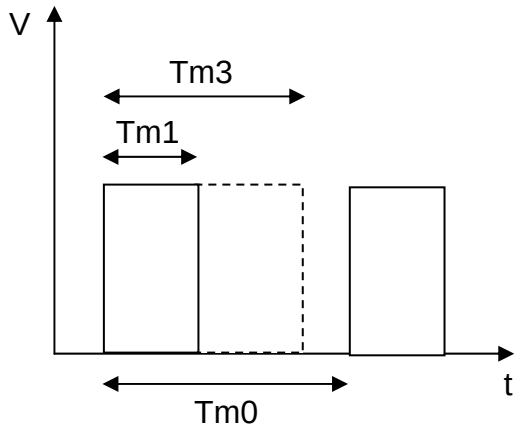


Figura 3.21 Funcionamiento de la señal de PWM generada por software

donde:

Tm0 es el periodo de la señal de PWM y es generado por el temporizador 0
Tm1 es el ciclo de trabajo generado por el temporizador1
Tm3 es el ciclo de trabajo generado por el temporizador3

3.4 Diseño de la interfaz de potencia para los motores

Esta interfaz le permite al microcontrolador controlar la potencia promedio entregada a cada uno de los motores. Para llevar a cabo el control fino de los motores, se escogió la frecuencia base para los PWM´s de 2.44 kHz, por lo cual se seleccionaron transistores tipo MOSFET de conmutación rápida.

La señal de PWM entregada por el controlador difuso para cada motor se conecta a la compuerta del MOSFET IRLZ44N aislando eléctricamente el puerto del microcontrolador. Para proteger el transistor de las corrientes inversas se colocó el diodo volante en antiparalelo al motor.

El diseño del circuito utilizado para cada motor se muestra en la figura 3.22.

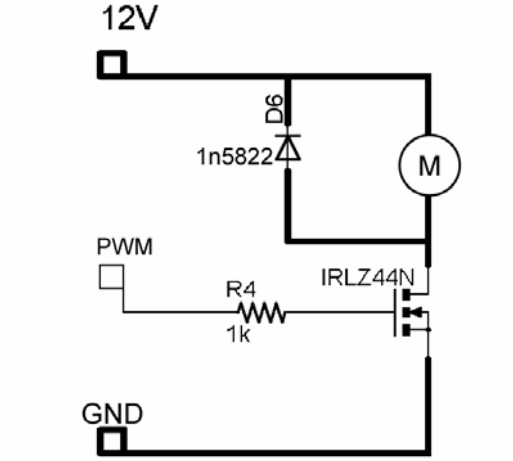


Figura 3.22 Esquema del driver para los motores

3.5 Diseño de herramientas de hardware y software necesarias para el proyecto

Para facilitar la comprensión y visualización de las operaciones que se están realizando en el microcontrolador fue indispensable desarrollar herramientas complementarias de hardware y software, para emular los procesos ejecutados por el mismo.

Se requirió de una interfaz gráfica de desarrollo para PC que incluyera un módulo simulador que generara y procesara los datos de la misma forma en que lo haría el microcontrolador, con el fin de verificar la correcta operación de este, además de lograr la depuración del código en lenguaje ensamblador de una forma más sencilla.

Por otro lado, debía contar con herramientas gráficas de diseño de sistemas difusos como lo son el mapa asociativo difuso y la superficie de control, mismas que permitieron evaluar el desempeño del controlador difuso.

Además, fue necesario contar con la tarjeta de desarrollo que permitió interactuar con los recursos del microcontrolador de manera confiable, así como también programarlo a través de la conexión serial con la PC.

La interfaz gráfica de desarrollo fue programada en Visual Basic y está basada en el entorno gráfico de Microsoft Excel. Su propósito fundamental es hacer más sencilla la creación, modificación y depuración del controlador.

A través de esta interfaz se permite al usuario modificar de manera confiable los parámetros del controlador difuso, mientras que al mismo tiempo se genera la información codificada en el sistema numérico hexadecimal de los conjuntos de entrada, salida y las reglas difusas. Esta información se agrega al programa en lenguaje ensamblador y es utilizada posteriormente por el microcontrolador durante la ejecución de las acciones de control.

La interfaz gráfica incluye el simulador del controlador embebido, el cual opera de manera similar y genera las mismas tablas de valores y vectores que el sistema físico. Esta función es muy útil en la detección de errores de programación pues permite comparar los resultados parciales y totales en las diferentes subrutinas y operaciones del microcontrolador ante cualquier entrada posible.

Los 5 módulos que componen la Interfaz Gráfica son:

- Creación de variables lingüísticas
- Creación de reglas
- Simulador
- Salida de datos en lenguaje ensamblador
- Herramientas gráficas de sintonización

3.5.1 Módulo de creación de Variables lingüísticas

En este módulo se diseñan los conjuntos de entrada y salida del controlador asignando los valores lingüísticos para cada uno de ellos. Está constituido por las dos primeras hojas del libro de Excel "Simulador Difuso.xls" y el macro llamado "Crear Gráficas". La interfaz gráfica así como el simulador soportan hasta 6 variables de entrada con 8 conjuntos y sólo una variable de salida de hasta 8 conjuntos.

Después de determinar el número de variables de entrada, se ingresan los valores lingüísticos de estas en las celdas correspondientes, así como también las dimensiones de los conjuntos que los definen. La notación empleada supone que los conjuntos tienen forma trapezoidal con altura igual a uno.

DERIVADA DEL ERROR				
Punto 1	Punto 2	Punto 3	Punto 4	Valores lingüísticos
0	0	82	97	baja muy rápido
82	97	97	112	baja rápido
97	112	112	127	baja lento
112	127	127	142	Estático
127	142	142	157	sube lento
142	157	157	172	sube rápido
157	172	255	255	sube muy rápido
0	0	0	0	

Tabla 3.10 Definición de variables lingüísticas

Con los datos contenidos en la tabla se crea la gráfica que representa los conjuntos de entrada o valores lingüísticos para la variable lingüística en cuestión.

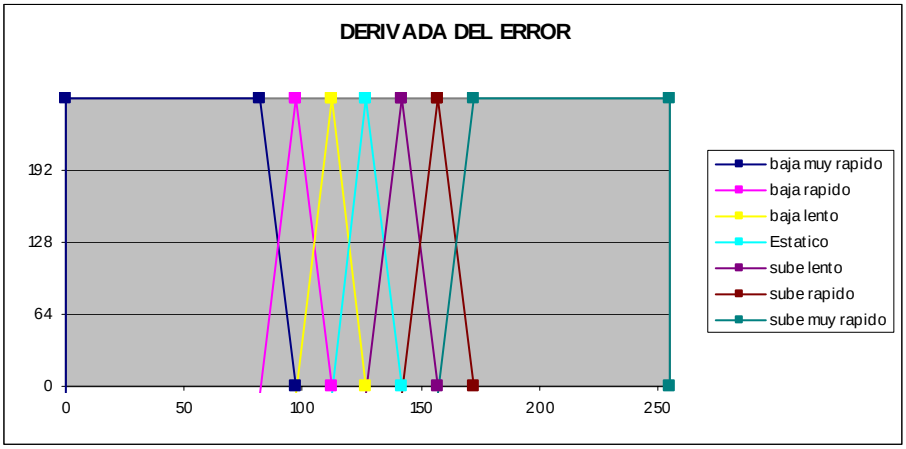


Figura 3.23 Representación Gráfica de una Variable de entrada

Las modificaciones posteriores a los conjuntos de entrada pueden realizarse cambiando los valores directamente en la tabla o directamente de la gráfica arrastrando los puntos de los conjuntos.

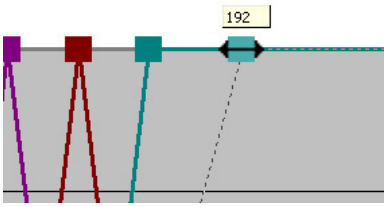


Figura 3.24 Método gráfico de modificación de los valores lingüísticos

El procedimiento anterior se aplica también para la creación de la variable de salida, pero en este caso se asume que los conjuntos de salida tienen estructura tipo singleton, por lo cual en la tabla únicamente se precisa la posición y los valores lingüísticos de los mismos.

Potencia de Salida	
Potencia cero	0
Potencia muy pequeña	40
Potencia pequeña	80
Potencia sustentación	125
Potencia grande	170
Potencia muy grande	210
Potencia máxima	255
	0

Tabla 3.11 Definición de valores lingüísticos de salida

Con los datos contenidos en la tabla se crea la gráfica que representa los conjuntos de salida.

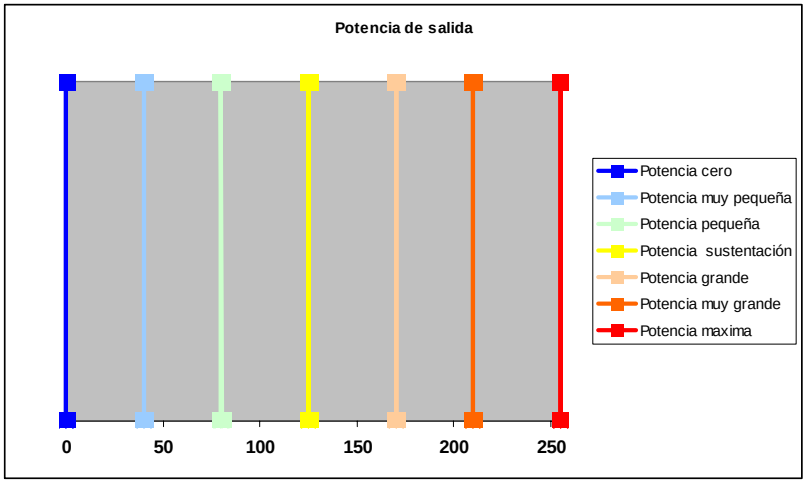


Figura 3.25 Representación Gráfica la Variable de Salida

3.5.2 Módulo de creación de reglas

Las reglas se crean mediante la ventana de captura, la cual presenta de forma cíclica los diferentes casos de control y permite al operador experto determinar cual de los conjuntos de salida disponibles es el idóneo para la condición dada.

Para facilitar la corrección y depuración de las reglas, la ventana también muestra el valor anterior asignado a la condición presente si es que el usuario ya ha asignado alguno en un evento anterior. La ventana permite que se ingresen los valores mediante el uso del mouse o bien con el teclado numérico. Después de presentar todas las posibles combinaciones del sistema, la ventana se cierra y crea la tabla con los índices relativos que requerirá el microcontrolador para interpretar las reglas, además de la leyenda con los valores lingüísticos asociados a estas.

Cuando se ejecuta el macro “Crear Reglas” el código busca los valores lingüísticos validos de la variable de salida y los despliega en la ventana mostrada en la figura 3.26.

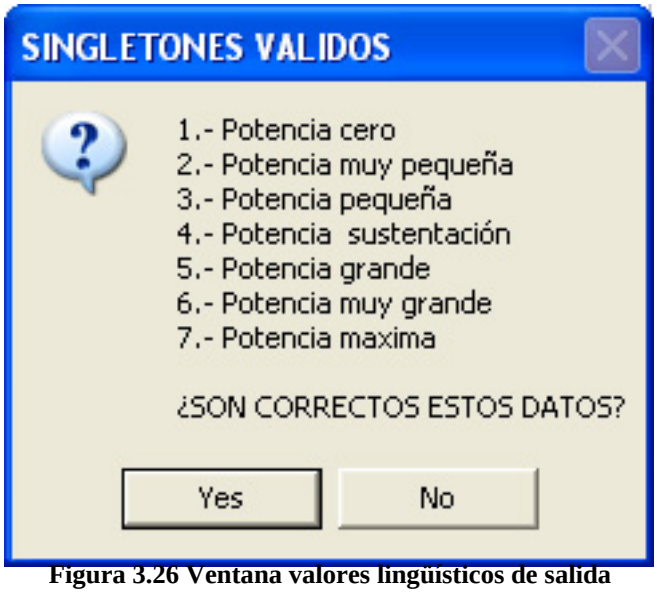


Figura 3.26 Ventana valores lingüísticos de salida

Posteriormente, determina el número de variables lingüísticas de entrada y el número de valores lingüísticos de cada una de estas. Con estos datos construye de forma dinámica la ventana de captura.

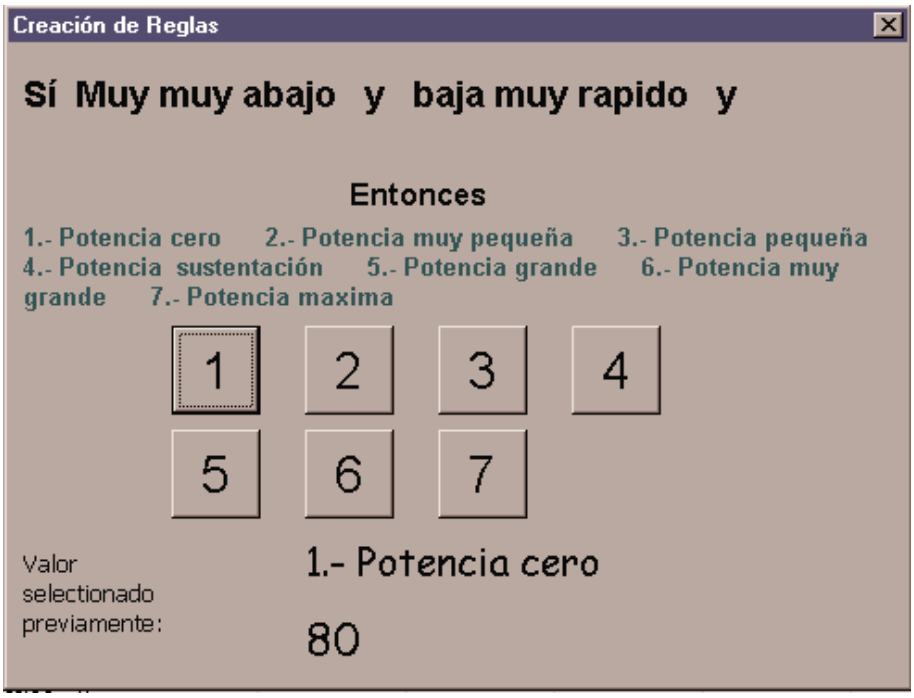


Figura 3.27 Ventana de Captura de reglas

Por último se genera la tabla de índices relativos codificada en hexadecimal con las reglas.

00	Nivel -3 y
08	baja muy rapido y
86	7.- Potencia maxima
00	Nivel -3 y
09	baja rapido y
86	7.- Potencia maxima
00	Nivel -3 y
0A	baja lento y

Tabla 3.12 Ejemplo de codificación hexadecimal

3.5.3 Módulo de Simulador

En este módulo se pueden simular las acciones de control para los cuatro motores dadas dos entradas llamadas sensorX y sensorY. Consta de la hoja "simulador" donde el usuario ingresa ambas entradas y el macro llamado "Simulador General"el cual calcula y reporta en esta misma hoja los resultados. En el proceso de simulación el algoritmo crea las tablas de grados de membresía por conjunto, errores, vectores de prioridad y pwm's. Los datos son presentados tanto en sistema decimal como en hexadecimal (negritas).

Entradas generales		
140	SensorX	**8C**
50	SensorY	**32**

Tabla 3.13 Ejemplo entradas del simulador

GRADOS DE MEMBRESIAS POR CONJUNTO b_{10}								
Conjuntos	1	2	3	4	5	6	7	8
Error	0	0	0	0	85	170	0	0
Derivada del error	0	0	51	204	0	0	0	0
GRADOS DE MEMBRESIAS POR CONJUNTO b_{16}								
Conjuntos	1	2	3	4	5	6	7	8
Error	**0**	**0**	**0**	**0**	**55**	**AA**	**0**	**0**
Derivada del error	**0**	**0**	**33**	**CC**	**0**	**0**	**0**	**0**

Tabla 3.14 Ejemplo Grados de membresía

Tabla de vectores		
40	APUNTADOR_CICLO_1	28
36	APUNTADOR_CICLO_2	24
44	APUNTADOR_CICLO_3	2C
32	APUNTADOR_CICLO_4	20

Tabla 3.15 Ejemplo Apuntadores de ciclo

Tabla de errores		
140	X1 ACTUAL	8C
127	X1 DERIVADA	7F
1	INDICE X1	01
140	X1 ANTERIOR	8C
115	X2 ACTUAL	73
127	X2 DERIVADA	7F
2	INDICE X2	02
115	X2 ANTERIOR	73
50	Y1 ACTUAL	32
127	Y1 DERIVADA	7F
3	INDICE Y1	03
50	Y1 ANTERIOR	32
205	Y2 ACTUAL	CD
127	Y2 DERIVADA	7F
4	INDICE Y2	04
205	Y2 ANTERIOR	CD

Tabla 3.16 Ejemplo Datos

Tabla de PWM'S		
74	X1	4A
140	X2	8C
255	Y1	FF
40	Y2	28

Tabla 3.17 Ejemplo PWM's

3.5.4 Módulo de salida de datos en lenguaje ensamblador

En la hoja “Salida” se generan en leguaje de ensamblador y de forma automática los conjuntos de entrada, salida y las reglas traducidas a índices relativos que serán utilizadas por el microcontrolador en la ejecución del código embebido.

```
;CONJUNTOS DIFUSOS DE ENTRADA

DA      0x0000,0x5211           ;Muy muy abajo
DA      0x5211,0x6111           ;Muy abajo
DA      0x6111,0x7011           ;Abajo
DA      0x7011,0x7F11           ;Horizontal
DA      0x7F11,0x8E11           ;Arriba
DA      0x8E11,0x9D11           ;Muy arriba
DA      0x9D11,0xFF00           ;Muy muy arriba
DA      0x0000,0x0000           ;0

DA      0x0000,0x5211           ;baja muy rapido
DA      0x5211,0x6111           ;baja rapido
DA      0x6111,0x7011           ;baja lento
DA      0x7011,0x7F11           ;Estatico
DA      0x7F11,0x8E11           ;sube lento
DA      0x8E11,0x9D11           ;sube rapido
DA      0x9D11,0xFF00           ;sube muy rapido
DA      0x0000,0x0000           ;0


; CONJUNTOS DIFUSOS DE SALIDA

DA      0X0046                  ;Potencia cero, Potencia muy pequeña
DA      0X6482                  ;Potencia pequeña ,Potencia  sustentación
DA      0XA0BE                  ;Potencia grande ,Potencia muy grande
DA      0XD200                  ;Potencia maxima ,
```

Tabla 3.18 Ejemplo de conjuntos difusos en lenguaje ensamblador

3.5.5 Módulo de herramientas gráficas de sintonización

Para sintonizar el controlador difuso se utiliza el mapa asociativo difuso y la superficie de control, ambas herramientas nos ayudan a visualizar el comportamiento de las acciones de control permitiendo realizar los ajustes necesarios de manera sencilla.

3.5.5.1 Mapa asociativo difuso

El mapa asociativo difuso representa las diferentes combinaciones de los valores lingüísticos de ambas variables. A las diferentes intersecciones se les asigna el color que representa el nivel de energía entregado por la condición de salida establecida como se muestra en la figura 3.28.

.

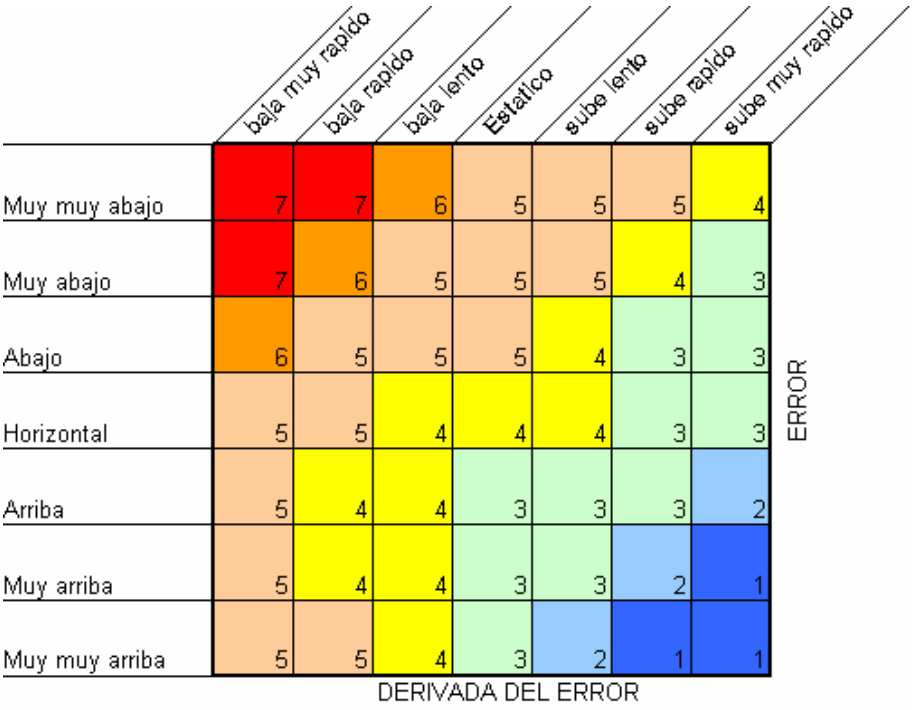


Figura 3.28 Mapa asociativo difuso

El conjunto de reglas también puede ser modificado a través del mapa asociativo difuso, escribiendo el valor deseado en la casilla correspondiente y ejecutando el macro “Mapa asociativo”

3.5.5.2 Superficie de control

La superficie de control representa de forma gráfica el resultado de la simulación de las acciones de control para el número discreto de valores dentro del universo del discurso de ambas variables como se muestra en la figura 3.29. Cuando se ejecuta el macro “superficie de control” se inicia el proceso de simulación y se genera la matriz con los resultados. Esta matriz proporciona la información de altura para la superficie. La estructura de los datos es compatible también con Matlab.

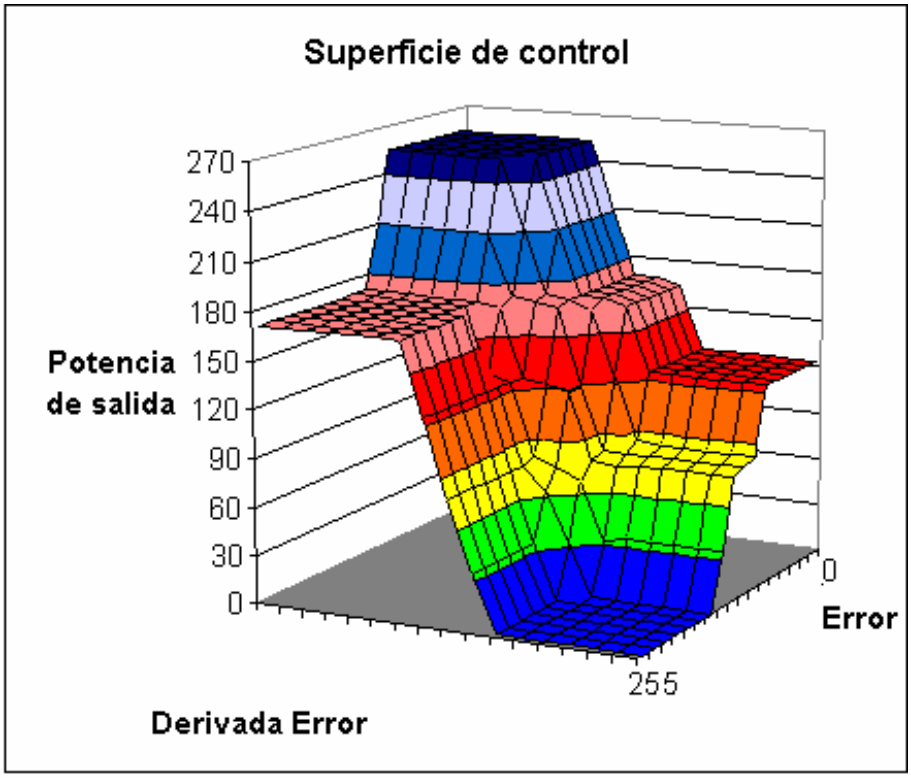


Figura 3.29 Superficie de control

3.5.6 Herramientas de hardware

3.5.6.1 Tarjeta de desarrollo

Esta tarjeta está conformada por dos módulos construidos en el circuito impreso con superficie de 10 cm de ancho por 20 de largo. El primer módulo contiene el circuito para realizar la programación de la familia de microcontroladores PIC18FXXX, el segundo cuenta con conexiones para cada uno de los cinco puertos digitales y en cada uno de ellos presenta dos pines extras para Vcc y tierra. Existe un puerto adicional para los ocho canales de entrada del convertidor analógico digital y la conexión ICSP[1] para programar el microcontrolador en la misma tarjeta. Incluye circuito de reinicio y de regulación de voltaje.

[1] Programación en el circuito mismo (In-Circuit Serial Programming)

El módulo de programación está basado en el PROPIC 2, utiliza el puerto paralelo de la PC como interfaz serial a través de la cual manda los datos que serán escritos en memoria flash y los comandos que ejecuta la ALU para programar el microcontrolador. Los voltajes de programación y lectura 13 y 5 volts respectivamente son proporcionados por el arreglo de reguladores de voltaje que aseguran su estabilidad. Es esta característica la que le permite ser independiente de la potencia del puerto pero requiere de voltaje externo mayor o igual a 15 volts.

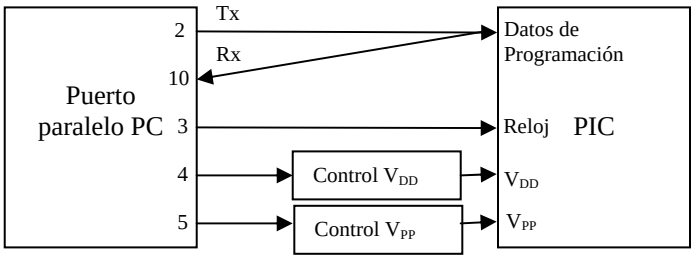


Figura 3.30 Esquema funcional de comunicación

Para programar el microcontrolador se utiliza el software EPICWIN a través de la función de programación en el circuito mismo. La interfaz del programa posibilita la visualización de los bits de configuración permitiendo su modificación como se muestra en la figura 3.31.

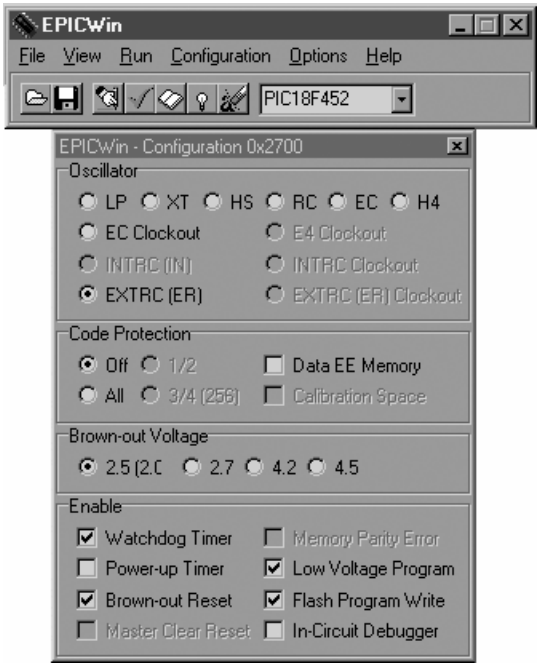


Figura 3.31 Ventana del software de programación

El circuito diseñado para la programación del PIC18FXXX es el mostrado en la figura 3.32.

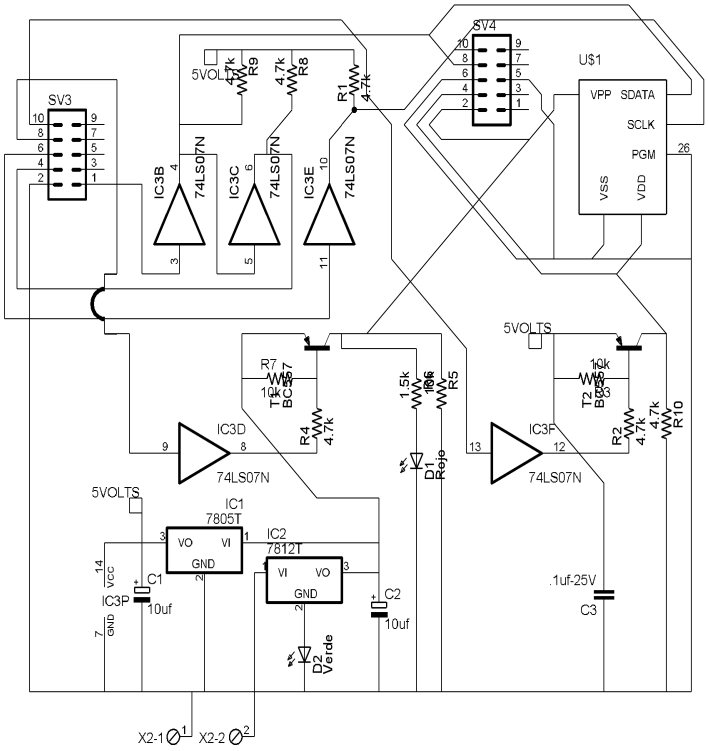


Figura 3.32 Circuito del programador

En la figura 3.33 se muestra el diagrama de conexión de los puertos y la etapa de regulación de la tarjeta de desarrollo.

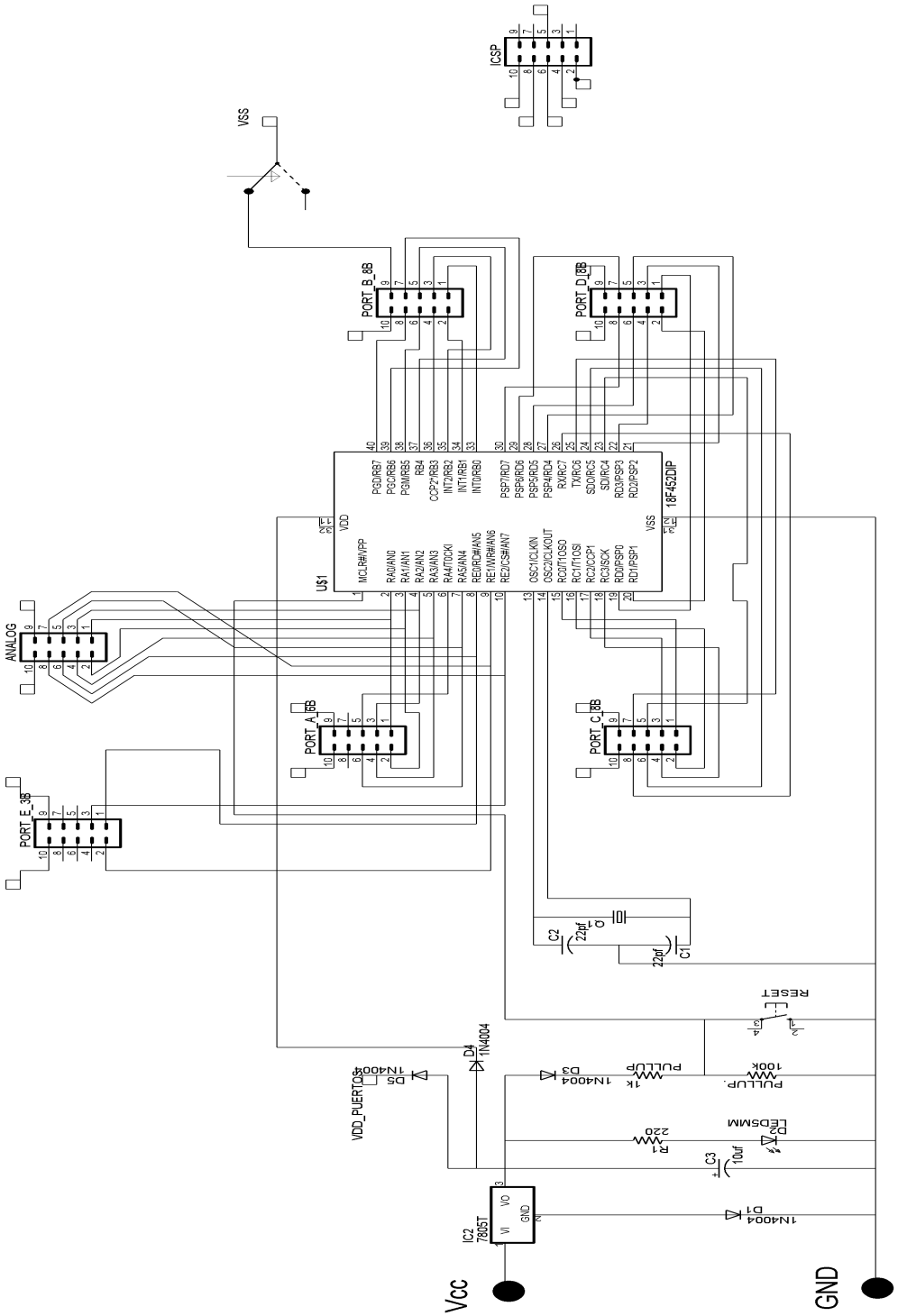


Figura 3.33 Circuito de conexión de los puertos y la etapa de regulación de la tarjeta de desarrollo

3.6 Implementación Final

Para probar el funcionamiento del controlador difuso sobre el sistema real, se diseño una tarjeta que contiene los elementos necesarios para efectuar el control sobre el DIVAC.

Debido a esto, se construyó la tarjeta sobre una placa rectangular con doble superficie de cobre cuyas dimesiones son 7.5 x 9.5 cm y que contiene los recursos justos para satisfacer las necesidades diseñadas para la etapa de adquisición y acondicionamiento de la señal de posición angular y de la etapa de potencia, además de contener al microcontrolador PIC18F452 responsable de ejecutar el código del controlador difuso, el respectivo circuito de reloj, cuatro salidas por el puerto C para controlar los drivers de los motores, la etapa de regulación de voltaje y el puerto para la programación ICSP.

A diferencia de la tarjeta de desarrollo, el diseño de esta tarjeta incluyó una disminución de tamaño y peso, lo cual redujo la potencia necesaria en los motores para lograr la sustentación del DIVAC.

La distribución de los componentes se realizó considerando posibles efectos de interferencia electromagnética por lo que se dejó espacio libre para colocar una jaula de Faraday como se muestra en la figura 3.34.

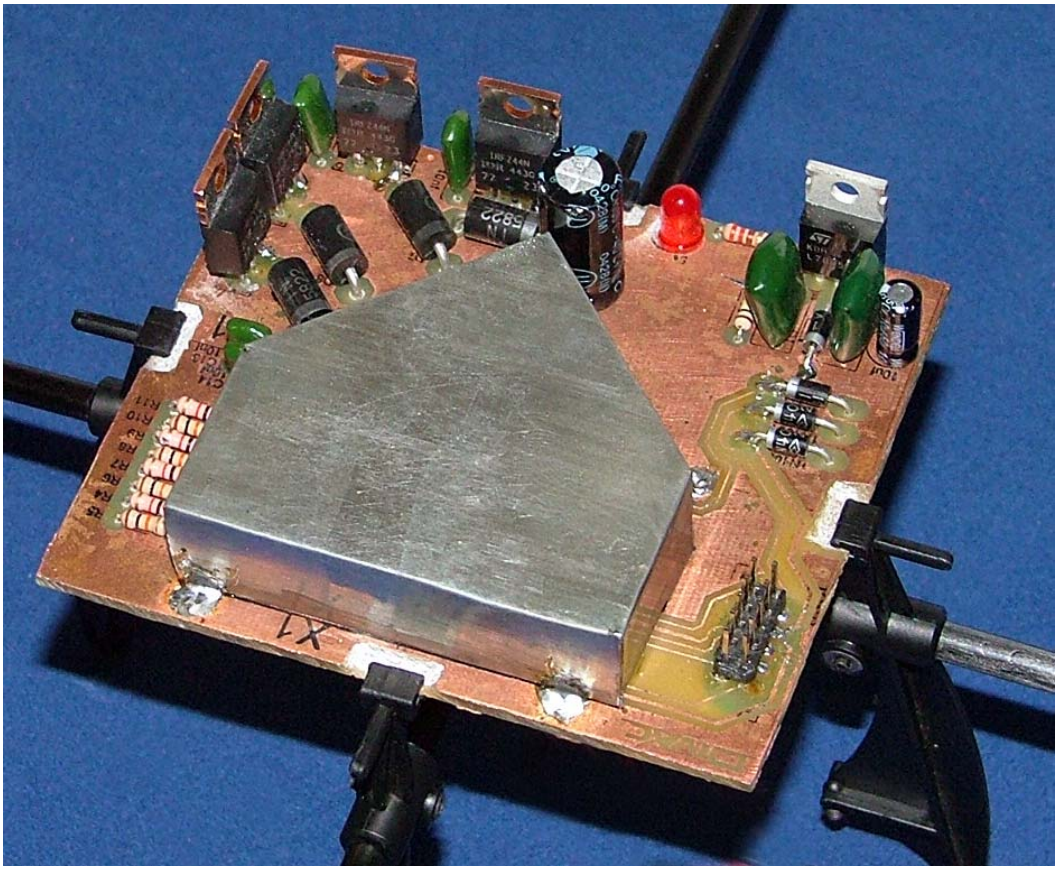


Figura 3.34 Implementación final con jaula de Faraday

Por otro lado el sensor de posición angular se colocó justo en el centro geométrico de la tarjeta para asegurar una correcta medición del ángulo de inclinación con respecto al centro de gravedad del DIVAC.

Para proteger las señales analógicas del sensor contra inducciones de voltaje generadas por la interferencia electromagnética de los motores, las rutas de conexión se encuentran rodeadas por el plano de tierra. Las figuras 3.35 y 3.36 muestran la vista superior e inferior del circuito impreso.

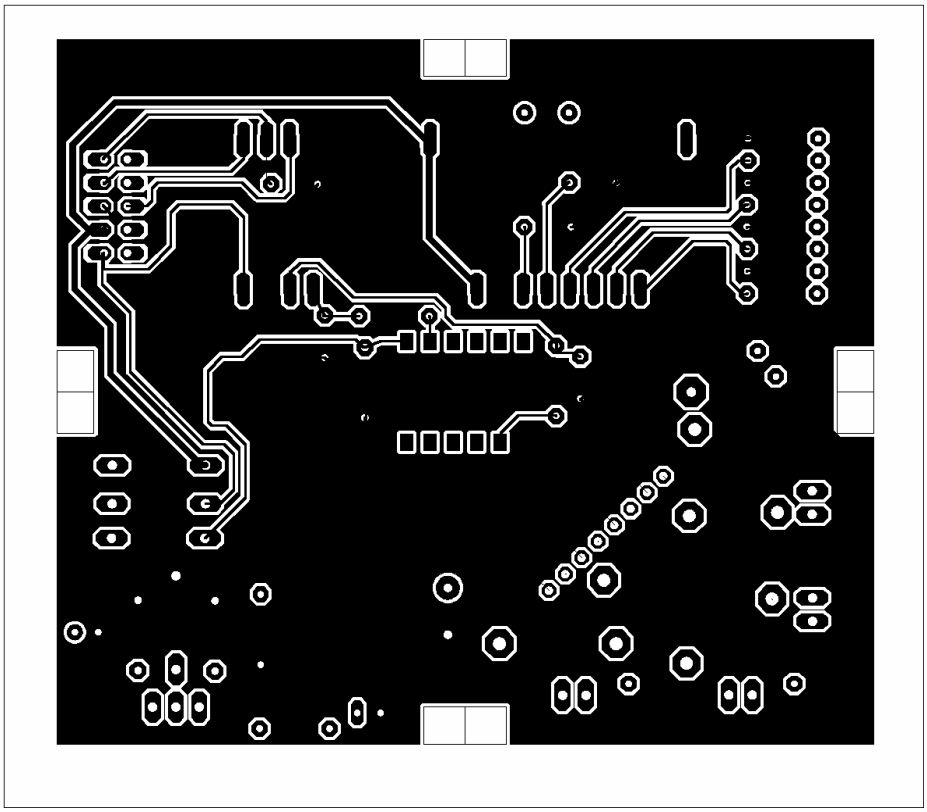


Figura 3.35 Vista superior del circuito impreso

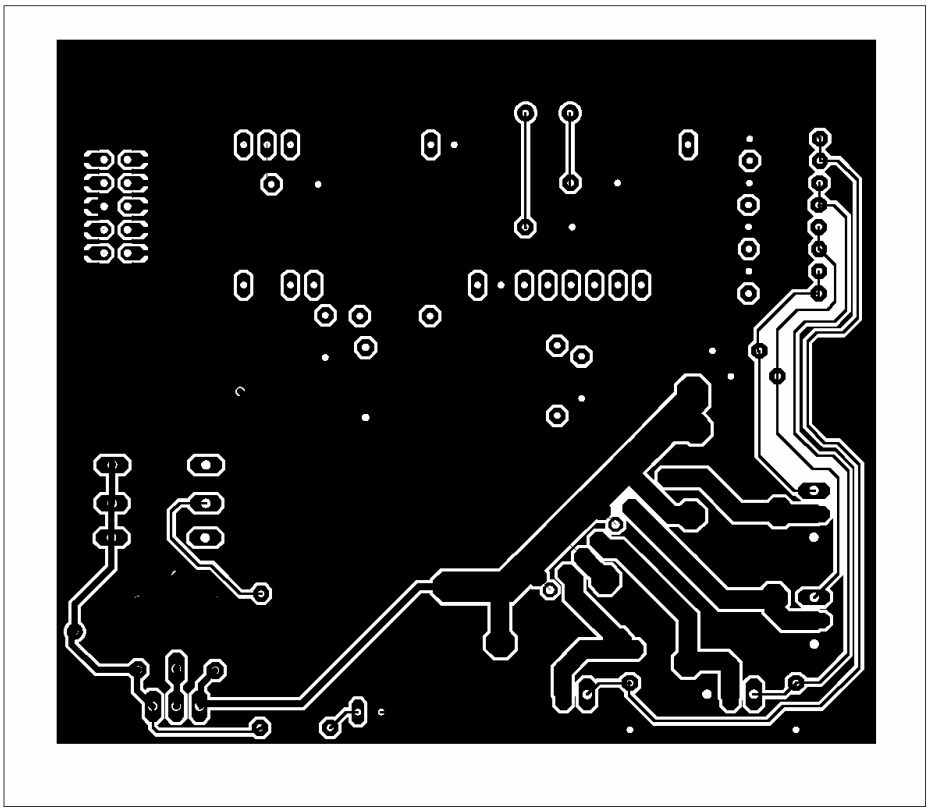


Figura 3.36 Vista inferior del circuito impreso

Capítulo IV Pruebas y resultados

Una vez que se contó con todos los elementos tanto de hardware como de software para el funcionamiento del DIVAC, se procedió a realizar las pruebas de funcionamiento y la sintonización. Este último proceso tiene como objetivo optimizar los conjuntos difusos para cada una de las entradas crisp y las salidas difusas, así como las reglas que permitan el funcionamiento del dispositivo.

Utilizando el software desarrollado en Excel como herramienta complementaria fue posible modificar de manera rápida y sencilla los conjuntos de entrada utilizando el módulo de creación de variables lingüísticas.

4.1 Diseño inicial propuesto para el control del DIVAC

Para realizar las primeras pruebas y basados en la experiencia adquirida sobre el funcionamiento del DIVAC, se propuso el siguiente diseño inicial para el control del DIVAC. Los conjuntos de entrada para el error y para la derivada del error son mostrados en la figura 4.1 y 4.2 respectivamente.

ERROR				
Punto 1	Punto 2	Punto 3	Punto 4	V. lingüísticos
0	0	67	87	Muy muy abajo
67	87	87	107	Muy abajo
87	107	107	127	Abajo
107	127	127	147	Horizontal
127	147	147	167	Arriba
147	167	167	187	Muy arriba
167	187	255	255	Muy muy arriba
0	0	0	0	

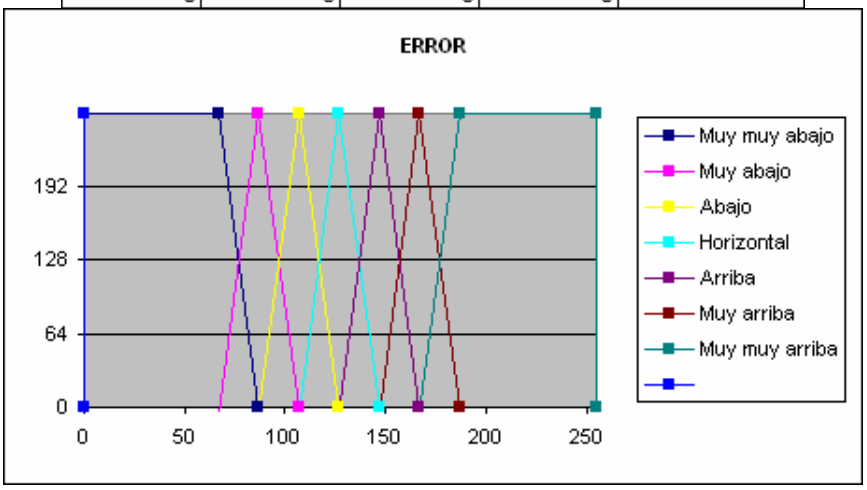


Figura 4.1 Diseño propuesto para los conjuntos de la entrada error

DERIVADA DEL ERROR				
Punto 1	Punto 2	Punto 3	Punto 4	V. lingüísticos
0	0	22	57	baja muy rapido
22	57	57	92	baja rapido
57	92	92	127	baja lento
92	127	127	162	Estatico
127	162	162	197	sube lento
162	197	197	232	sube rapido
197	232	255	255	sube muy rapido
0	0	0	0	

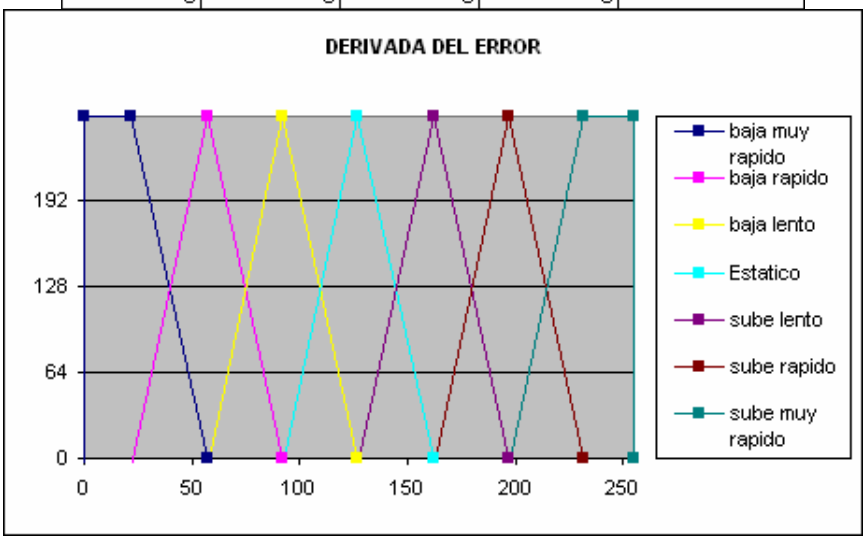


Figura 4.2 Diseño propuesto para los conjuntos de la entrada derivada del error

Los conjuntos de salida se muestran en la figura 4.3.

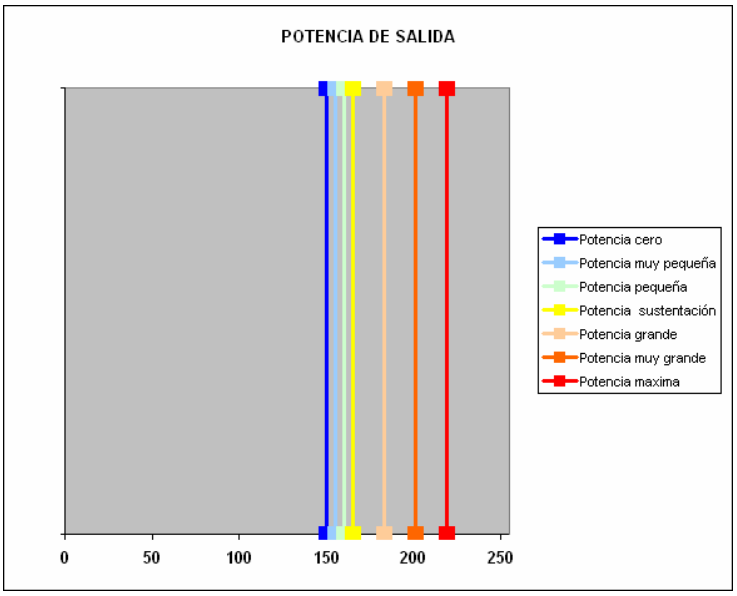


Figura 4.3 Diseño propuesto para los conjuntos de salida

El conjunto de reglas se muestra en la figura 4.4 dentro del mapa asociativo.

ERROR \ DERIVADA DEL ERROR	baja muy rapido	baja rapido	baja lento	Estatico	sube lento	sube rapido	sube muy rapido
Muy muy abajo	7	7	6	5	5	5	4
Muy abajo	7	6	5	5	5	4	3
Abajo	6	5	5	5	4	3	3
Horizontal	5	5	4	4	4	3	3
Arriba	5	4	4	3	3	3	2
Muy arriba	5	4	4	3	3	2	1
Muy muy arriba	5	5	4	3	2	1	1

Figura 4.4 Diseño propuesto para las reglas

Finalmente, se realizó una simulación para obtener la superficie de control, la cual se muestra en la figura 4.5.

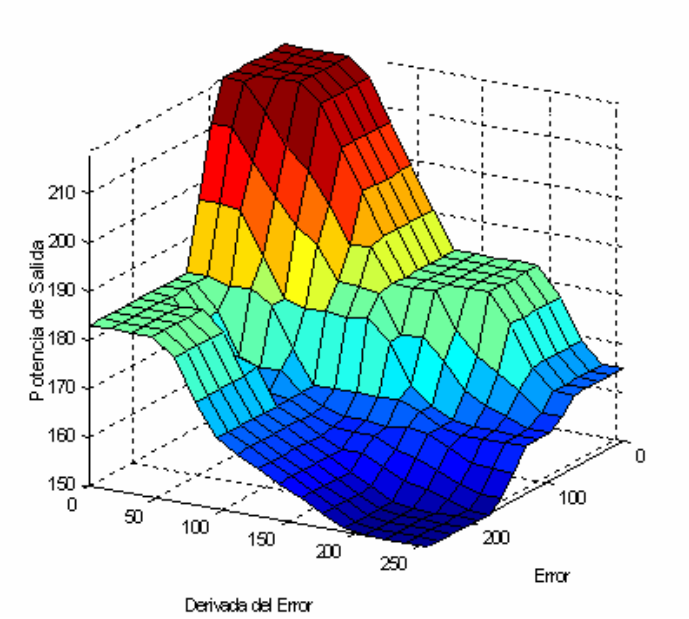


Figura 4.5 Superficie de control para el diseño propuesto

4.2 Realización de pruebas

Las primeras pruebas realizadas, se enfocaron en comprobar que el controlador difuso estuviera interpretando correctamente la información proveniente del sensor angular y calculando correctamente el ERROR y la DERIVADA DEL ERROR para todo el universo del discurso correspondiente.

Para tales fines, el procedimiento consistió en aislar cada caso posible a la vez empezando con los conjuntos de la entrada ERROR y posteriormente con los de la DERIVADA DEL ERROR.

Para aislar cada conjunto, se les asignó a todas las reglas relacionadas con éste la POTENCIA MAXIMA del sistema y al resto de ellas la POTENCIA CERO. El mapa asociativo difuso y la superficie de control para uno de estos casos pueden apreciarse en la figura 4.6. Como resultado de estas pruebas concluimos que el controlador era capaz de asignar correctamente la pertenencia en los diferentes conjuntos de la variable de entrada ERROR para las diferentes posiciones angulares del DIVAC.

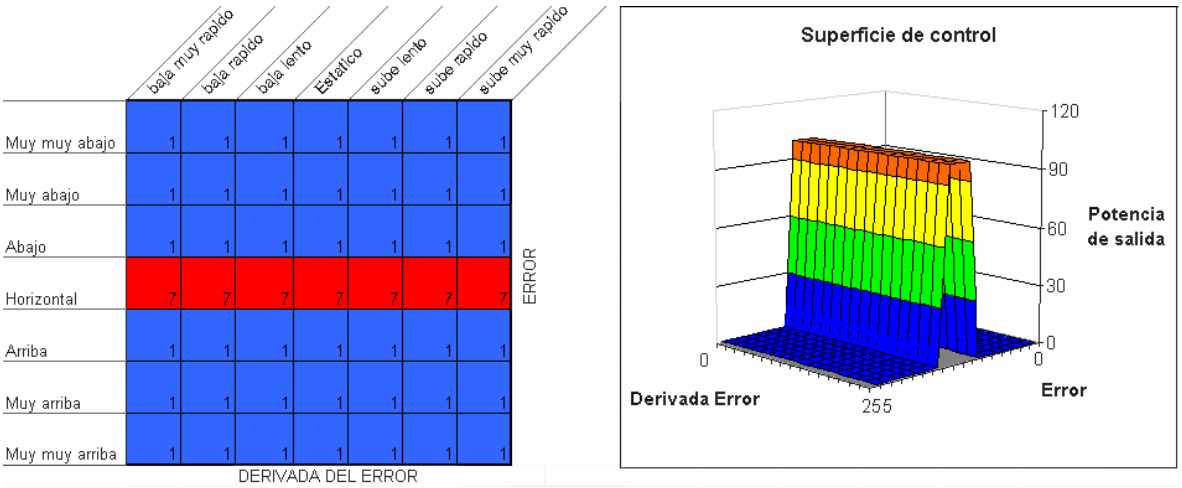


Figura 4.6 Reglas modificadas para probar los conjuntos del error

Un procedimiento similar fue aplicado para evaluar la entrada DERIVADA DE ERROR, un ejemplo de este proceso se puede apreciar en la figura 4.7.

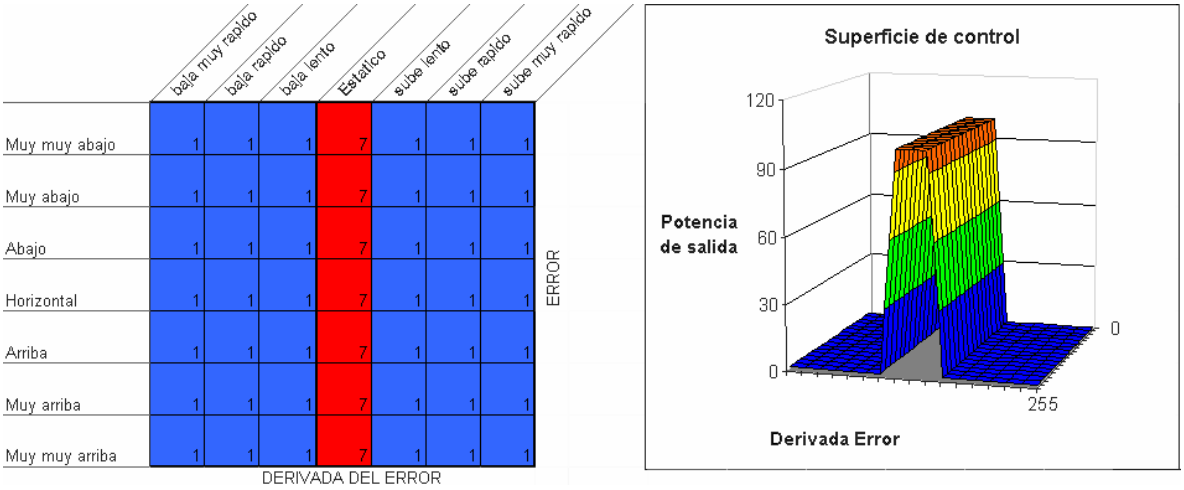


Figura 4.7 Reglas modificadas para probar los conjuntos de la derivada del error

Una vez finalizadas las pruebas sobre la DERIVADA DEL ERROR concluimos que la diferencial de tiempo con la cual se realizaba el cálculo de la misma era muy pequeña, teniendo como consecuencia que el controlador interpretara como muy pequeñas o nulas las tasas de variación en la posición angular del DIVAC.

Para solucionar este problema se modificó el código de programa con el fin de incrementar la duración de la diferencial de tiempo. Para encontrar el valor idóneo de dicho intervalo, esta modificación fue programada para permitir seleccionar el número de derivadas que se realizarían por segundo, teniendo como limite inferior 5 y 100 como superior.

Una vez incluidas las modificaciones propuestas, las pruebas realizadas sobre la entrada DERIVADA DEL ERROR reportaron que el controlador era capaz de discernir entre las diferentes tasas de cambio en la posición angular del DIVAC. Para tales efectos se configuró el controlador para que fuesen realizadas únicamente 4 derivadas por segundo.

En las primeras pruebas de sintonización se observó un comportamiento errático en las acciones de control, pues pese a que el DIVAC se encontraba fijo en una posición horizontal constante el controlador generaba correcciones como si éste estuviera presentando variación en la posición angular permanentemente.

Para localizar el origen de la falla, se analizó la señal voltaje generada por el sensor. En ella se descubrió la presencia de una señal de ruido de gran magnitud que sólo se presenta cuando los motores funcionan cerca o por encima de la velocidad nominal de sustentación. La señal no deseada se genera debido a la vibración producida por los motores en funcionamiento, misma que al ser trasmitida al sensor a través de la estructura de fibra de carbono modifican de manera instantánea la resultante de la aceleración que es el marco de referencia del sensor y por consiguiente el sensor interpreta las vibraciones como cambios súbitos en la posición angular.

Como se puede apreciar en el oscilograma de la señal en la figura 4.8, cuando el sensor se encuentra fijo en posición horizontal, la señal de ruido presente esta conformada fundamentalmente por la señal de frecuencia igual 33Hz con amplitud aproximada de 2Vpp la cual es aproximadamente la mitad del intervalo de la señal, es decir [0,5]V.

Habiendo sido localizado el origen de la falla, fueron propuestas dos soluciones complementarias con el fin de reducir los efectos del ruido sobre las acciones de control, dichas soluciones se implementaron en el siguiente orden:

1.- Por software
2.- Por hardware

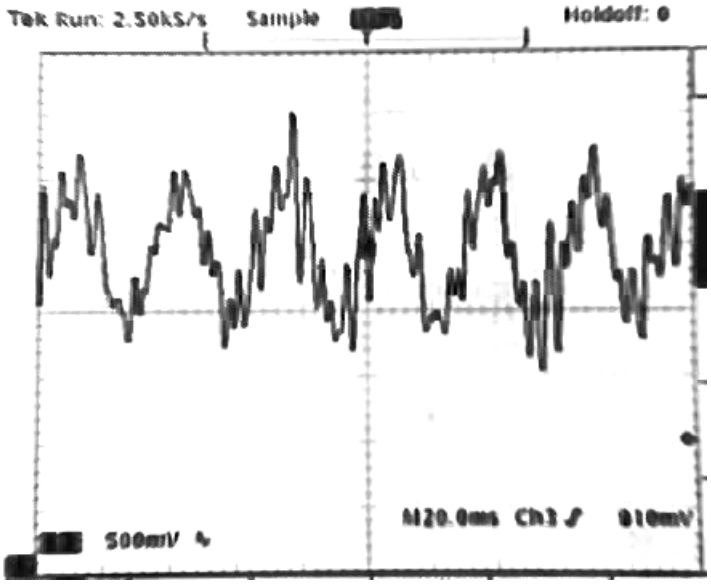


Figura 4.8 Oscilograma de la señal de ruido

Solución por software.

La solución por software, consiste en obtener un subpromedio y un promedio general de los valores obtenidos por el sensor.

El SUBPROMEDIO es realizado con 4 valores adquiridos durante un ciclo completo de control, con lo cual se obtiene un nuevo valor promediado cada 1ms. Con esto se incrementa la confiabilidad en la medición de la señal de voltaje proveniente del sensor de inclinación.

Como se puede apreciar en la figura 4.8 la señal no deseada se asemeja a una perturbación senoidal con periodo de 33ms y se encuentra montada sobre la señal de posición angular que para este análisis se considera constante. En la ecuación 4.2.1 se puede apreciar la función de la señal con ruido

$$sn(t) = s + sen\, w(t) \qquad (4.2.1)$$

Donde sn(t) es la señal con ruido
s es la señal de posición angular y
sen w(t) es la componente de ruido

Al integrar la ecuación 4.2.1 en el tiempo para un periodo completo de la señal de ruido y dividiendo entre el periodo de integración, se obtiene la ecuación 4.2.2 correspondiente a la señal de posición angular sin ruido.

$$\frac{\int_0^T sn(t)}{T} = \frac{\int_0^T s}{T} + \frac{\int_0^T sen\,\varpi(t)}{T} = s \qquad (4.2.2)$$

Extendiendo este concepto en el espacio discreto, la rutina GENERAL calcula la media de cuatro valores igualmente distribuidos en el periodo de la señal de ruido que es de 33ms.

Para incluir estas correcciones se anexó a la rutina KERNEL, las subrutinas de subpromedio y promedio, además se reubicó la subrutina derivada para que pudiese ser ejecutada desde la subrutina de promedio.

La figura 4.9 muestra la nueva estructura de la rutina KERNEL.

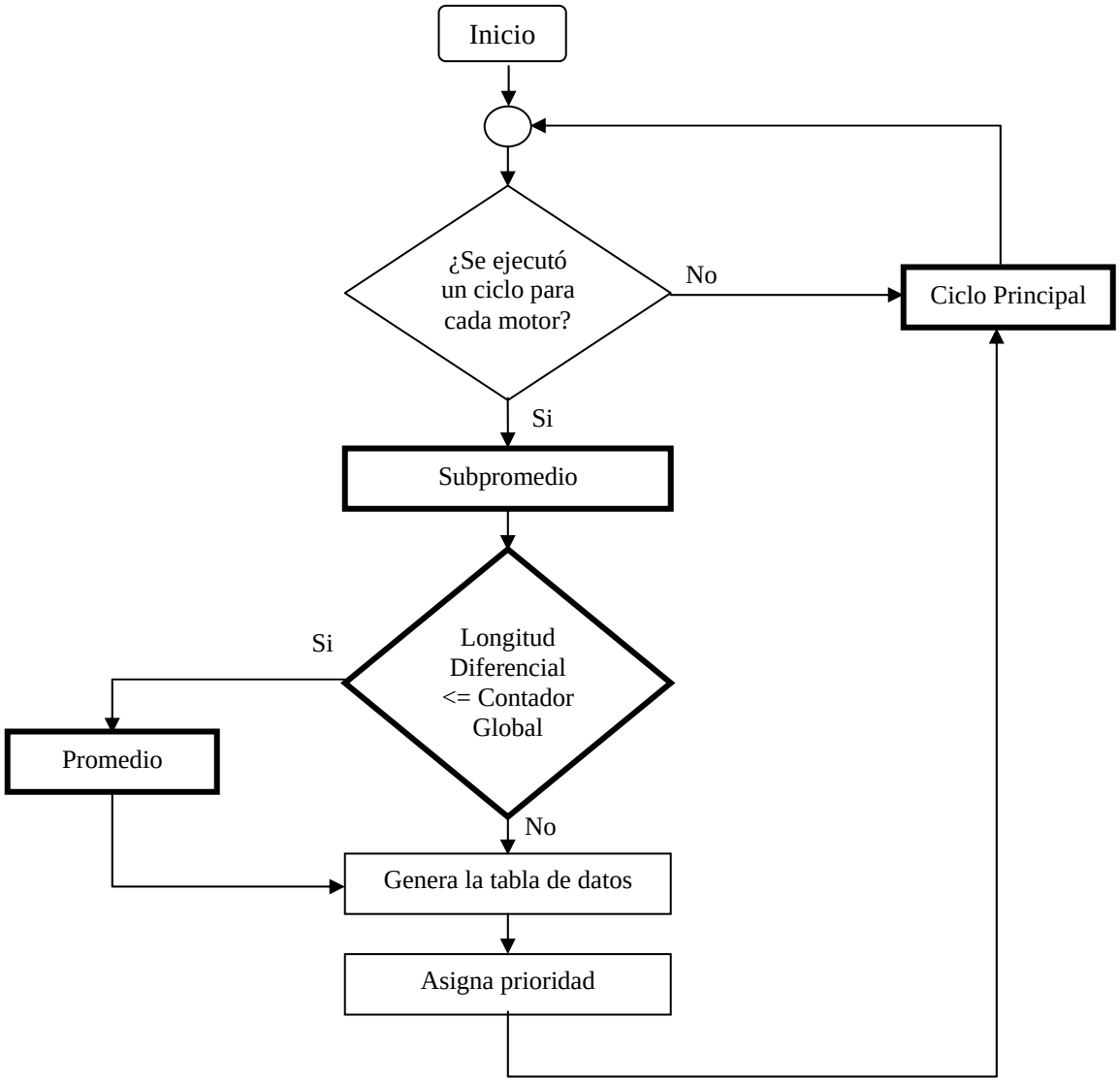


Figura 4.9 Diagrama de flujo de la rutina Kernel modificada

Al ciclo principal se agregó la subrutina temporales de sensores, la cual se encuentra después de la subrutina de actualización de PWM's. Esta rutina se encarga de almacenar en memoria RAM de forma indizada los valores de cada eje proporcionados por el sensor en cada subciclo difuso y prepararlos para ser operados por la rutina subpromedio.

La tabla 4.1 muestra las localidades en RAM donde se almacenan los valores temporales del sensor.

068h	SENSOR_X_TEMP_1
069h	SENSOR_X_TEMP_2
06Ah	SENSOR_X_TEMP_3
06Bh	SENSOR_X_TEMP_4
06Ch	SENSOR_Y_TEMP_1
06Dh	SENSOR_Y_TEMP_2
06Eh	SENSOR_Y_TEMP_3
06Fh	SENSOR_Y_TEMP_4

Tabla 4.1 Localidades de memoria donde se almacenan los valores temporales del sensor.

La subrutina Subpromedio utiliza cuatro registros temporales para cada eje. Cuando termina el ciclo completo de control realiza el promedio de los valores contenidos en estas localidades. La figura 4.10 muestra su estructura.

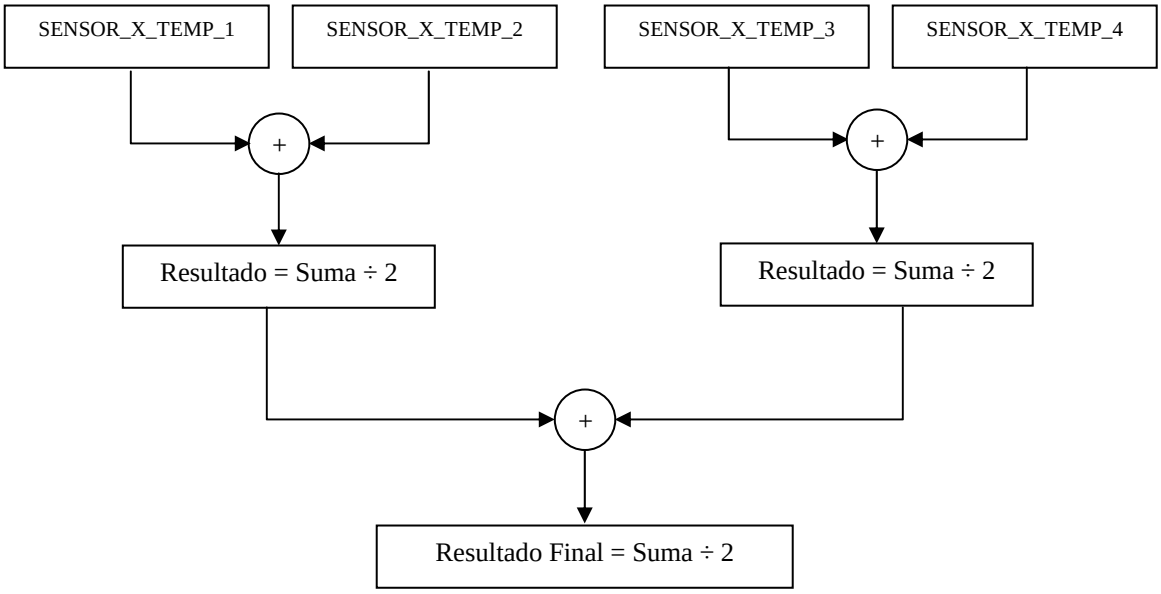


Figura 4.10 Estructura de la subrutina Subpromedio para el eje X

La subrutina Promedio utiliza dos contadores. Uno se encarga de guardar un valor proveniente del sensor para los dos ejes, cada 11 ms. El otro contador se usa para fijar el tiempo en que debe realizar el promedio general y la derivada del error. Tiene la misma estructura que la subrutina subpromedio según se muestra en la figura 4.11 y utiliza también ocho valores temporales dado que son necesarios cuatro por eje.

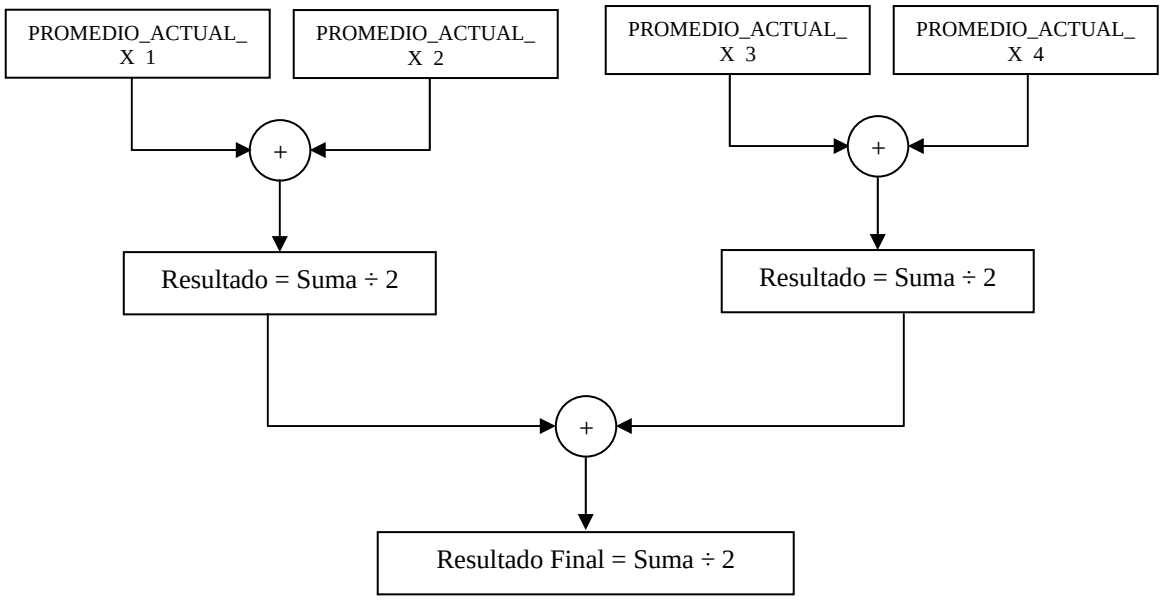


FIGURA 4.11 Estructura de la subrutina Promedio para el eje X

La tabla 4.2 muestra las localidades en RAM donde se almacenan los valores temporales del sensor para la subrutina promedio.

070h	PROMEDIO_ACTUAL_X_1
071h	PROMEDIO_ACTUAL_X_2
072h	PROMEDIO_ACTUAL_X_3
073h	PROMEDIO_ACTUAL_X_4
074h	PROMEDIO_ACTUAL_Y_1
075h	PROMEDIO_ACTUAL_Y_2
076h	PROMEDIO_ACTUAL_Y_3
077h	PROMEDIO_ACTUAL_Y_4

Tabla 4.2 Localidades de memoria donde se almacenan los valores temporales del sensor.

La nueva estructura del controlador se muestra en la figura 4.12.

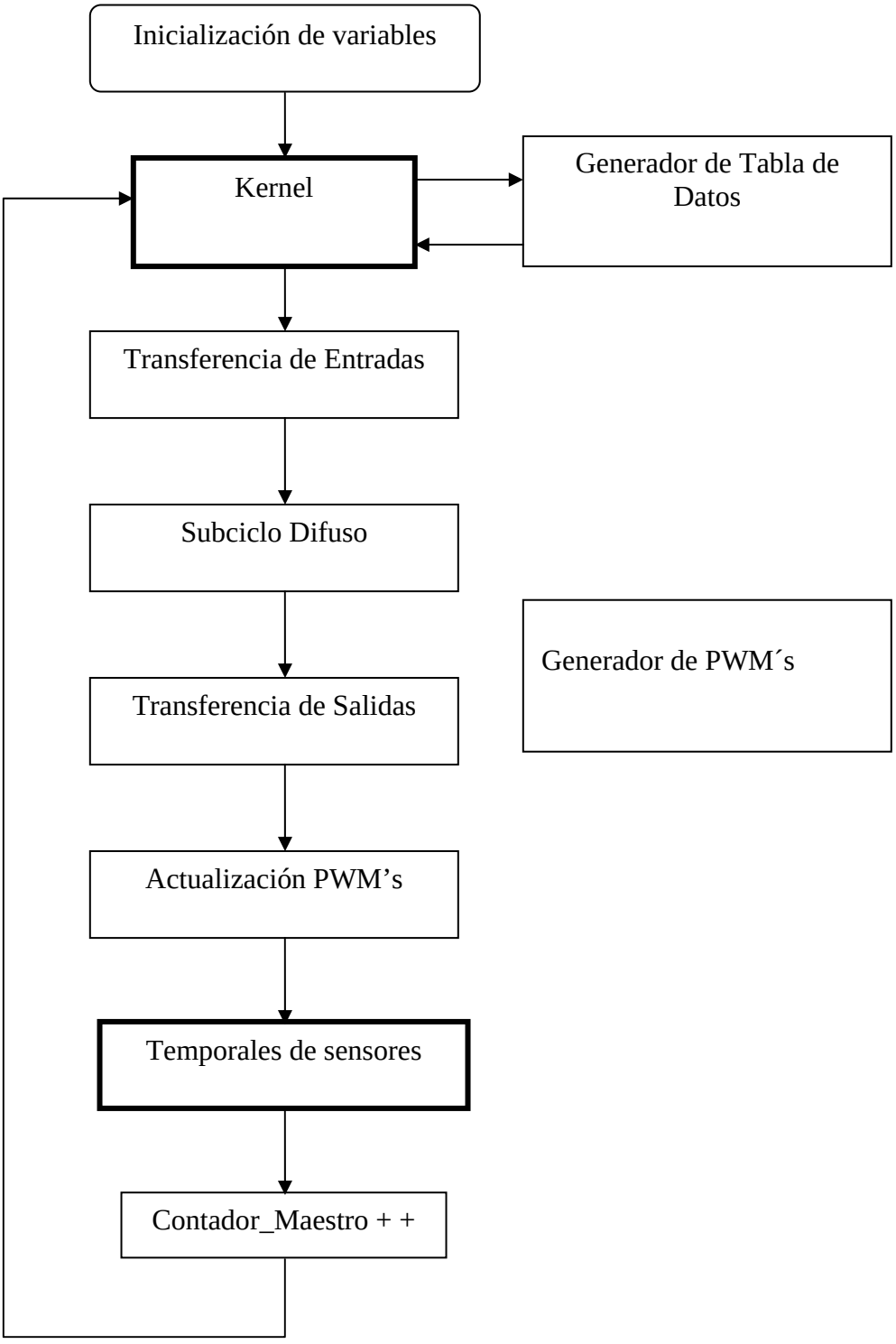


Figura 4.12 Diagrama de flujo modificado del controlador difuso

Solución por hardware

Esta consistió en implementar un filtro Butterworth de sexto orden para mantener la banda de paso plana y al mismo tiempo atenuar 120 dB/ década la banda de rechazo. Se fijo la frecuencia de corte en 20 Hz para minimizar la pérdida de la señal proporcionada por el filtro y atenuar 26dB la señal de ruido.

El proceso de diseño de este filtro se describe a continuación:

Partiendo del polinomio de sexto orden de Butterworth mostrado en la ecuación (4.2.3).

$$P(s) = s^6 + 3.864s^5 + 7.464s^4 + 9.142s^3 + 7.464s^2 + 3.864s + 1 \qquad (4.2.3)$$

Se puede obtener la función de transferencia para un filtro Butterworth paso bajas como se muestra en la ecuación (4.2.4).

$$H(s) = \frac{1}{(s^2 + 0.5176s + 1)(s^2 + 1.4142s + 1)(s^2 + 1.9319s + 1)} \qquad (4.2.4)$$

Esta función de transferencia puede ser implementada con tres etapas de segundo orden conectadas en cascada.

El modelo general de cada una de estas etapas es el mostrado en la ecuación (4.2.5).

$$A_i(s) = \frac{A_0}{(1 + a_i s + b_i s^2)} \qquad (4.2.5)$$

Considerando ganancia unitaria (A_0=1), y una topología Sallen-Key la estructura general de cada etapa se muestra en la figura 4.13.

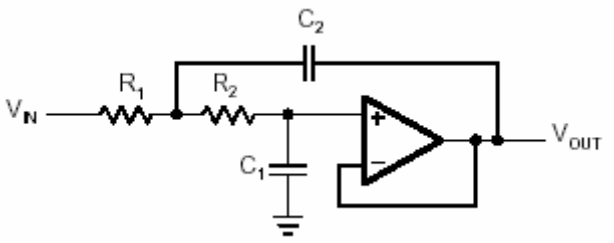


Figura 4.13 Diagrama para filtro de 2º orden tipo Sallen-Key

Siendo su función de transferencia la mostrada en la ecuación (4.2.6).

$$A(s) = \frac{1}{1 + \omega_C C_1 (R_1 + R_2)s + \omega_C{}^2 R_1 R_2 C_1 C_2 s^2} \qquad (4.2.6)$$

Comparando los coeficientes de la función de transferencia obtenida en la ecuación (4.2.5) con los de la ecuación (4.2.6) se observa que:

$$A_0 = 1$$
$$a_1 = \omega_C C_1 (R_1 + R_2)$$
$$b_1 = \omega_C{}^2 R_1 R_2 C_1 C_2$$

Si se fija el valor de C_1 y C_2, los valores de R_1 y R_2 se calculan a partir de la ecuación (4.2.7).

$$R_{1,2} = \frac{a_1 C_2 \mp \sqrt{a_1{}^2 C_2{}^2 - 4 b_1 C_1 C_2}}{4\pi f_C C_1 C_2} \tag{4.2.7}$$

Para que los valores de las resistencias sean reales se debe satisfacer la condición mostrada en (4.2.6):

$$C_2 \geq C_1 \frac{4b_1}{a1^2} \tag{4.2.8}$$

Considerando estos parámetros de diseño a continuación se calculan los valores de capacitares y resistencias para cada etapa.

Etapa 1

Se selecciona el valor de C_1=56 nF, obteniendo el valor de C_2 a partir de (4.2.8):

$C_2 \geq 56x10^{-9} * \frac{(4*1)}{1.9319^2}$ = 60nF, el valor comercial resulta de 68nF

Obteniendo el valor de R_1 y R_2 a partir de (4.7) queda:

$$R_1 = \frac{1.9319 * 68x10^{-9} - \sqrt{(1.9319 * 68x10^{-9})^2 - 4 * 1 * 56x10^{-9} * 68x10^{-9}}}{4\pi * 20 * 56x10^{-9} * 68x10^{-9}} = 90.2k\Omega$$

$$R_2 = \frac{1.9319 * 68x10^{-9} + \sqrt{(1.9319 * 68x10^{-9})^2 - 4 * 1 * 56x10^{-9} * 68x10^{-9}}}{4\pi * 20 * 56x10^{-9} * 68x10^{-9}} = 184.3k\Omega$$

Estos valores se aproximan a partir de los valores comerciales de precisión:

R_1= **90.9 kΩ**
R_2= **182 kΩ**

Etapa 2

Se selecciona el valor de C_3=22 nF, obteniendo el valor de C_4 a partir de (4.2.8):

$C_4 \geq 22x10^{-9} * \frac{(4*1)}{1.4142^2}$ = 44 nF, el valor comercial resulta de 56nF

Obteniendo el valor de R_3 y R_4 a partir de (4.2.7) queda:

$$R_3 = \frac{1.4142*56x10^{-9} - \sqrt{(1.4142*56x10^{-9})^2 - 4*1*22x10^{-9}*56x10^{-9}}}{4\pi*20*22x10^{-9}*56x10^{-9}} = 137.4k\Omega$$

$$R_4 = \frac{1.4142*56x10^{-9} + \sqrt{(1.4142*56x10^{-9})^2 - 4*1*22x10^{-9}*56x10^{-9}}}{4\pi*20*22x10^{-9}*56x10^{-9}} = 374.2k\Omega$$

Los valores anteriores se aproximan a partir de los valores comerciales de precisión:

R_3= **137 kΩ**
R_4= **374 kΩ**

Etapa 3

Se selecciona el valor de C_5=10 nF, obteniendo el valor de C_6 a partir de (4.2.8):

$C_6 \geq 10x10^{-9} * \frac{(4*1)}{0.5176^2}$ = 149.3 nF, el valor comercial resulta de 150nF

Obteniendo el valor de R_5 y R_6 a partir de (4.2.7) queda:

$$R_5 = \frac{0.5176*150x10^{-9} - \sqrt{(0.5176*150x10^{-9})^2 - 4*1*10x10^{-9}*150x10^{-9}}}{4\pi*20*10x10^{-9}*150x10^{-9}} = 191.9k\Omega$$

$$R_5 = \frac{0.5176*150x10^{-9} + \sqrt{(0.5176*150x10^{-9})^2 - 4*1*10x10^{-9}*150x10^{-9}}}{4\pi*20*10x10^{-9}*150x10^{-9}} = 220k\Omega$$

Estos valores se aproximan a partir de los valores comerciales de precisión:

R_3= **191 kΩ**
R_4= **220 kΩ**

El circuito con las tres etapas y los valores comerciales aproximados se muestra en la figura 4.14.

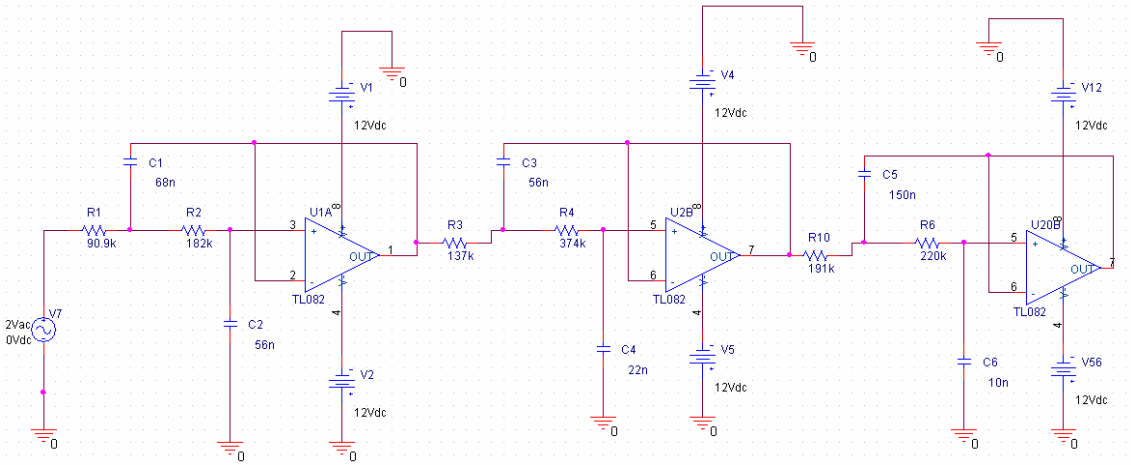


Figura 4.14 Diagrama del filtro Butterworth de 6º orden

La respuesta en frecuencia de esta configuración es la mostrada en la figura 4.15, en la cual se puede apreciar que la frecuencia no deseada tiene una amplitud aproximada a 100 mV

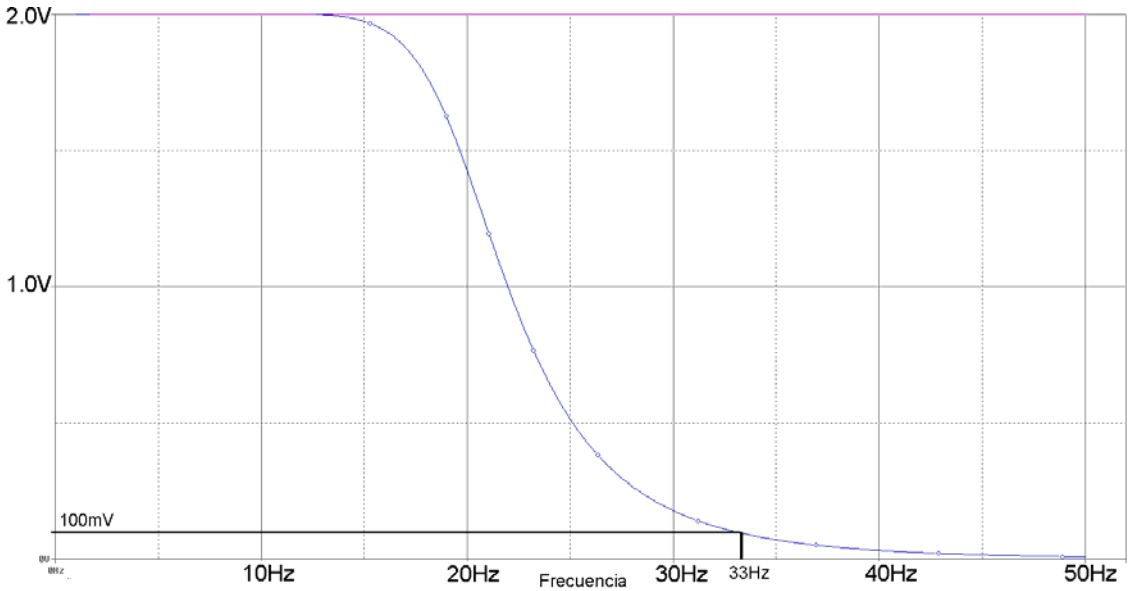


Figura 4.15 Respuesta en frecuencia de la señal filtrada

La respuesta al escalón unitario se muestra en la figura 4.16. En esta se puede apreciar que el tiempo de asentamiento del sensor es de 65ms.

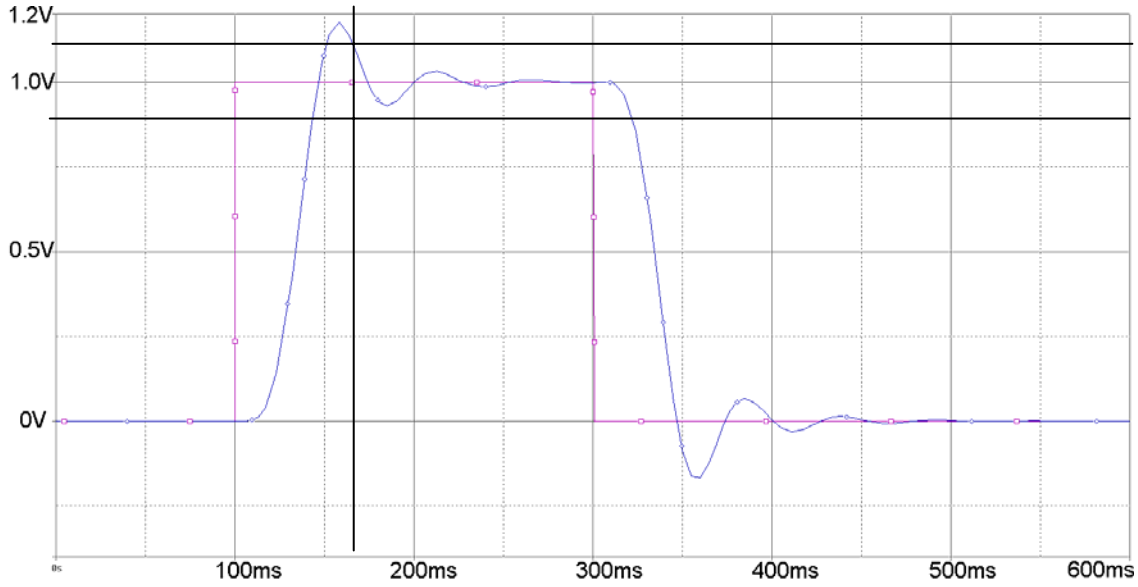


Figura 4.16 Respuesta en tiempo a escalón unitario

Posteriormente, se realizaron nuevas pruebas para verificar el contenido de ruido en la señal de voltaje del sensor de inclinación pero esta vez con la etapa de filtrado.

Los resultados se muestran en la figura 4.17.

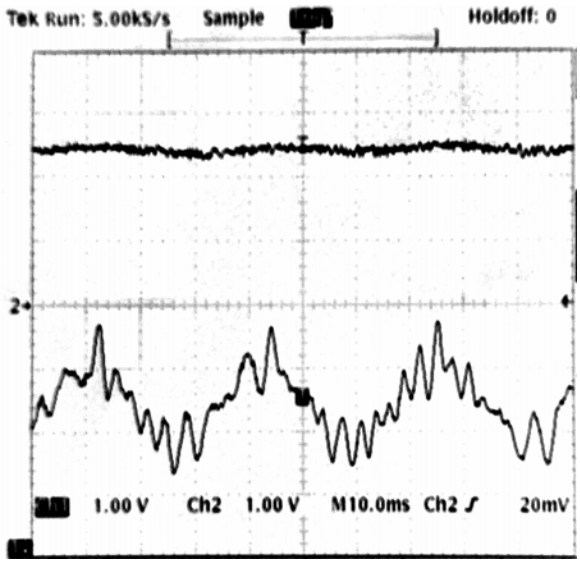


Figura 4.17 Oscilograma comparativo entre la señal con ruido y sin ruido

Una vez que se comprobó que el nivel de la señal de voltaje entregada por el sensor de inclinación era el adecuado, se integraron a la tarjeta final desarrollada anteriormente las etapas de filtrado. Para lo cual fue necesario construir una nueva tarjeta, la cual se muestra en la figura 4.18.

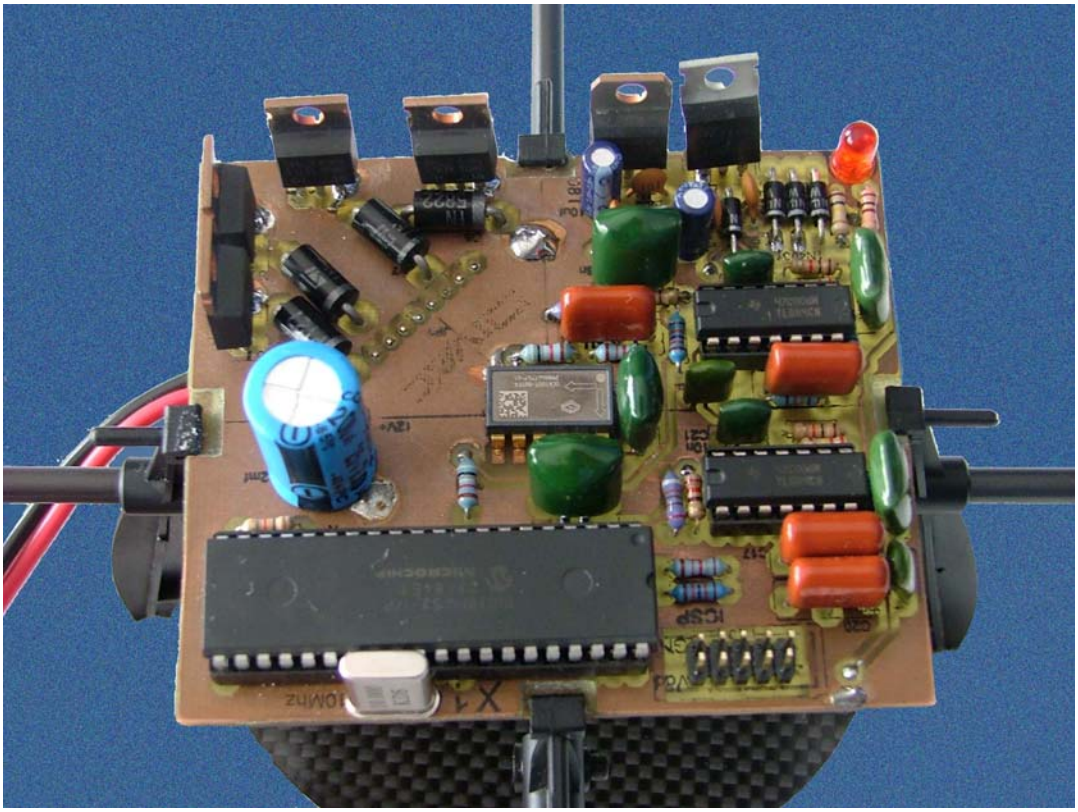


Figura 4.18 Tarjeta final con etapa de filtrado

4.2.1 Pruebas para un eje

Para realizar el nuevo conjunto de pruebas se decidió aplicar el controlador a un sólo eje, con lo cual sólo operarían dos motores observando así el comportamiento y la eficiencia de éste. De esta forma, el proceso de sintonización se realizaría de manera más sencilla dado que se aplican las mismas reglas y conjuntos para todos los motores. Una vez que se obtuviera un resultado aceptable, se podría aplicar en el otro eje.

Dado que las pruebas se tenían que hacer de forma autónoma, se construyó una plataforma que permitiera el libre movimiento angular en un eje. Al otro eje se le retiraron las hélices y sirvió de soporte de la estructura del DIVAC. Esto se muestra en la figura 4.19.

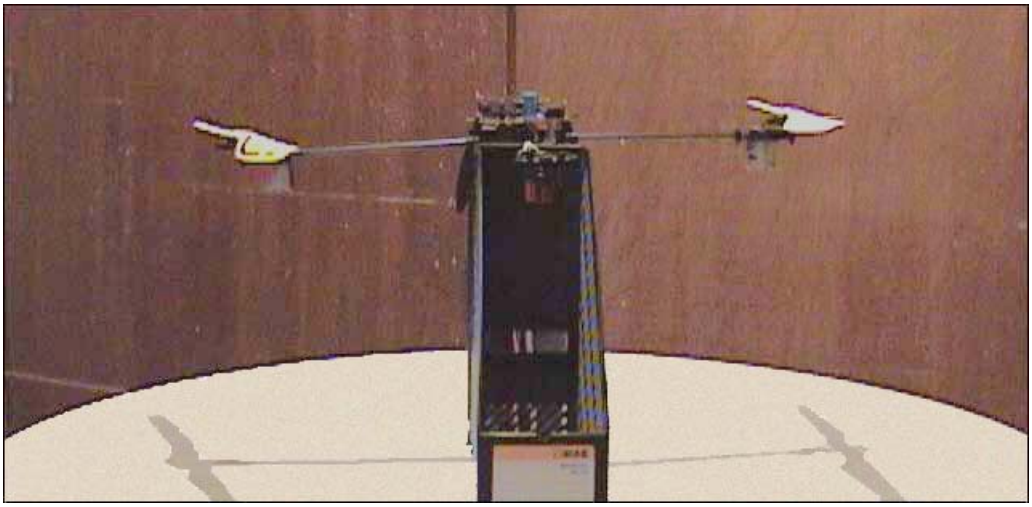


Figura 4.19 Pruebas para el DIVAC en un sólo eje

Las primeras pruebas de control se realizaron con el diseño de conjuntos difusos y reglas inicialmente propuesto. A partir de estas se observó una respuesta constantemente oscilatoria, es decir, se mantenía balanceándose. Se notaba que el controlador estaba generando las acciones de control aunque no lograba permanecer en posición horizontal.

El proceso de sintonización requirió de aproximadamente 25 conjuntos de pruebas diferentes para llegar al resultado final en los cuales se modificaron tanto los conjuntos de entrada y de salida como las reglas, obteniendo gran experiencia sobre las condiciones de vuelo.

Como resultado se obtuvo un diseño que lograba mantener en posición horizontal al DIVAC, presentando pequeñas perturbaciones que lo hacían desestabilizarse y descender o subir por encima de esta posición. A pesar de estas perturbaciones, el controlador efectuaba las acciones pertinentes para recuperar el nivel inicial de manera rápida. El funcionamiento de este diseño se puede observar en la figura 4.20.

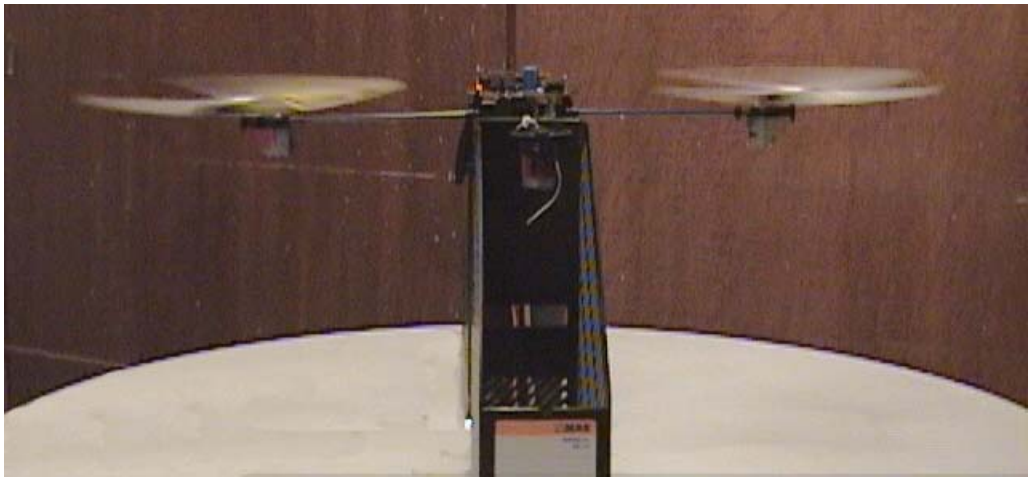


Figura 4.20 Pruebas en un sólo eje

Las figuras 4.21 y 4.22 muestran los conjuntos difusos de entrada finales de estas pruebas.

ERROR				
Punto 1	Punto 2	Punto 3	Punto 4	V. lingüísticos
0	0	50	88	Muy muy abajo
50	88	88	117	Muy abajo
88	117	117	127	Abajo
117	127	127	137	Horizontal
127	137	137	166	Arriba
137	166	166	205	Muy arriba
166	205	255	255	Muy muy arriba
0	0	0	0	

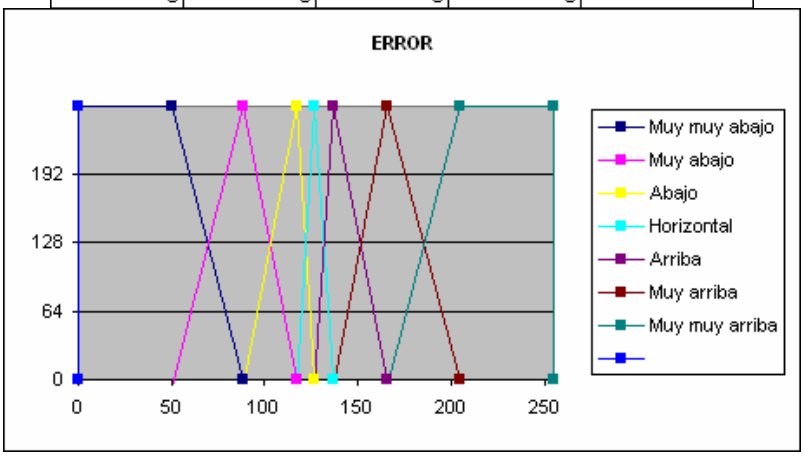


Figura 4.21 Diseño final para los conjuntos de la entrada error en pruebas de un eje

DERIVADA DEL ERROR				
Punto 1	Punto 2	Punto 3	Punto 4	V. lingüísticos
0	0	40	70	Baja muy rápido
40	70	70	107	Baja rápido
70	107	107	127	Baja lento
107	127	127	147	Estatico
127	147	147	184	Sube lento
147	184	184	215	Sube rapido
184	215	255	255	Sube muy rapido
0	0	0	0	

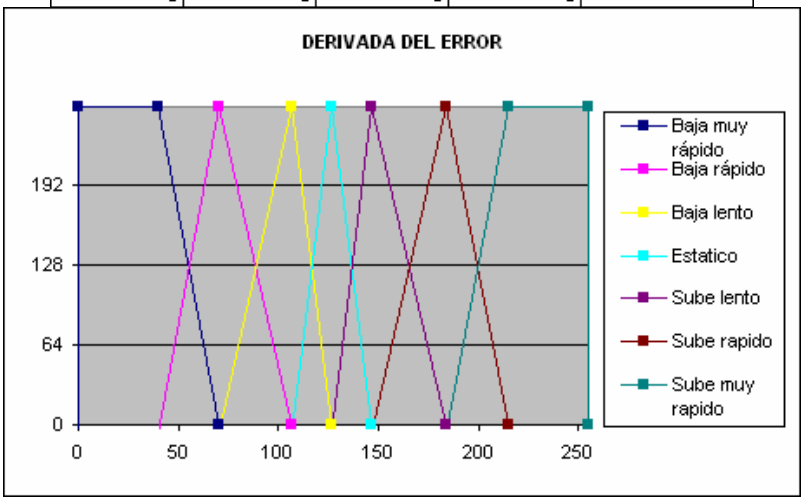


Figura 4.22 Diseño final para los conjuntos de la entrada derivada del error en pruebas de un eje

El conjunto de salida se muestra en la figura 4.23.

Potencia de Salida	
Potencia cero	159
Potencia sustentación	160
Potencia pequeña	163
Potencia sustentación	166
Potencia grande	169
Potencia muy grande	174
Potencia maxima	180
	0

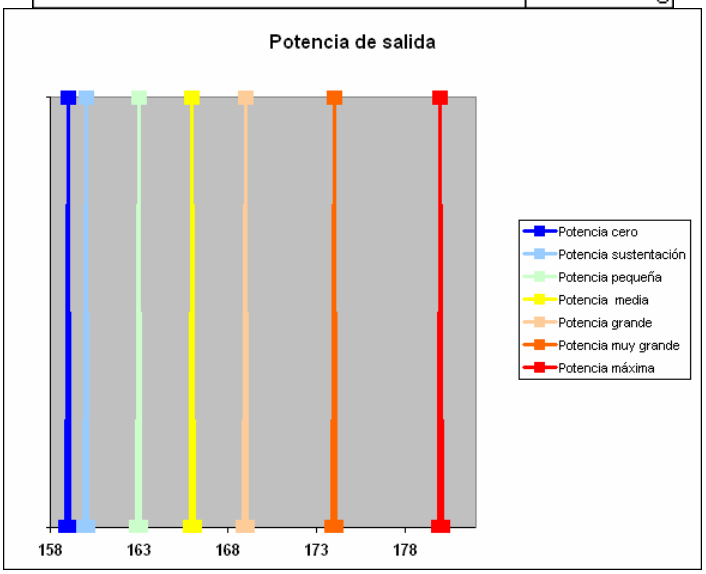


Figura 4.23 Diseño final para los conjuntos de salida en pruebas de un eje

Y por último el mapa asociativo y la superficie de control se muestran en las figuras 4.24 y 4.25 respectivamente.

ERROR \ DERIVADA DEL ERROR	Baja muy rápido	Baja rápido	Baja lento	Estatico	Sube lento	Sube rapido	Sube muy rapido
Muy muy abajo	7	6	6	6	5	4	2
Muy abajo	6	6	6	5	4	2	2
Abajo	6	5	4	3	2	2	1
Horizontal	4	4	3	2	2	1	1
Arriba	3	3	2	2	1	1	1
Muy arriba	2	2	2	2	1	1	1
Muy muy arriba	2	2	1	1	1	1	1

Figura 4.24 Mapa asociativo final en pruebas de un eje

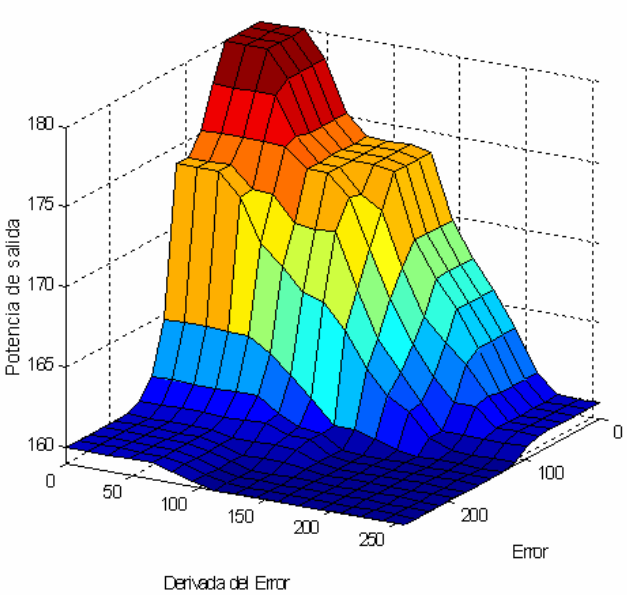


Figura 4.25 Superficie de control final en pruebas de un eje

4.2.2 Pruebas en dos ejes

Una vez que se concluyeron las pruebas para un solo eje, se procedió a realizar las pruebas considerando los dos ejes, es decir, con los cuatro motores. Para ello, se colocó en la base del DIVAC un cordón que limitaría la elevación por encima del nivel esperado para las pruebas además de evitar que sufriera algún daño en caso de que se saliera completamente de control. Por otro lado, estas pruebas se realizaron sobre una superficie lisa para evitar rupturas en las hélices por roces contra una superficie rugosa, además de que servía como protección para la persona que estuviera sosteniendo el cordón de elevación.

La figura 4.26 muestra la forma en que se realizaron estas pruebas.

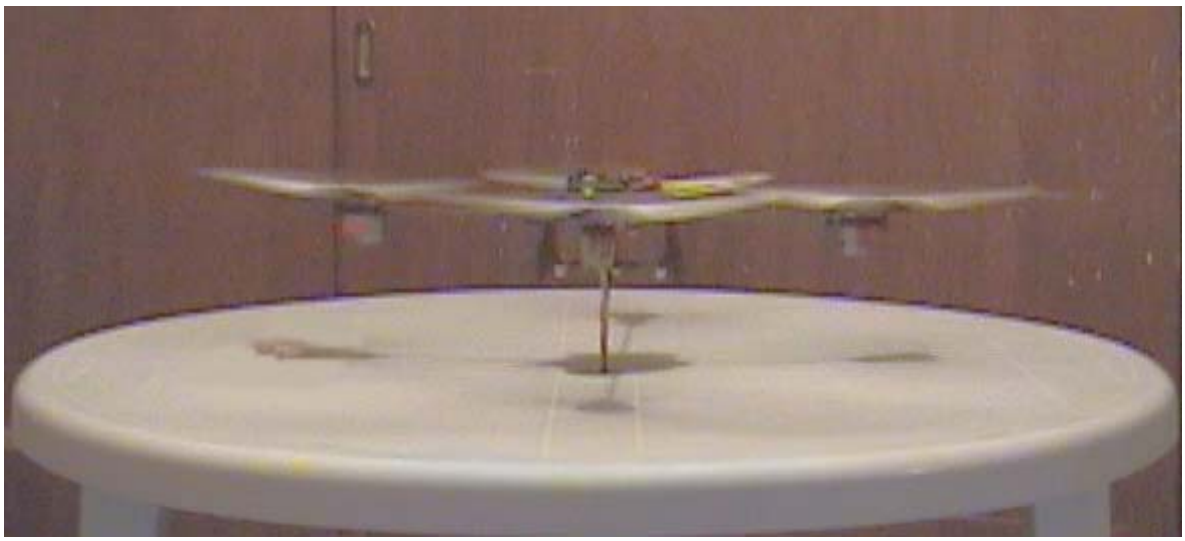


Figura 4.26 Pruebas para el DIVAC en dos ejes

El diseño del controlador difuso con el que se realizaron las primeras pruebas para dos ejes fue el obtenido como resultado final en las pruebas de un eje.

La primera prueba reveló que el controlador difuso del DIVAC presentaba el comportamiento esperado a partir de las pruebas con un eje. Se procedió a modificar únicamente el conjunto de reglas, con lo cuál se obtuvo un mejor desempeño.

Los resultados obtenidos fueron aceptables ya que en algunas ocasiones el DIVAC logró mantener la posición horizontal durante pequeños periodos de tiempo. Aunque el controlador genera las correcciones adecuadas corrigiendo diferentes niveles de inclinación para cada uno de los motores, no se obtiene un control perfecto sobre el dispositivo.

Finalmente, el conjunto de reglas modificado y su respectiva superficie de control se muestran en las figuras 4.27 y 4.28.

ERROR \ DERIVADA DEL ERROR	Baja muy rapido	Baja rapido	Baja lento	Estatico	Sube lento	Sube rapido	Sube muy rapido
Muy muy abajo	7	7	6	6	5	4	3
Muy abajo	7	6	6	5	4	3	2
Abajo	6	5	4	3	3	2	1
Horizontal	4	3	3	2	2	1	1
Arriba	3	2	2	2	1	1	1
Muy arriba	2	2	2	2	1	1	1
Muy muy arriba	2	2	1	1	1	1	1

Figura 4.27 Diseño final del mapa asociativo en pruebas de dos ejes

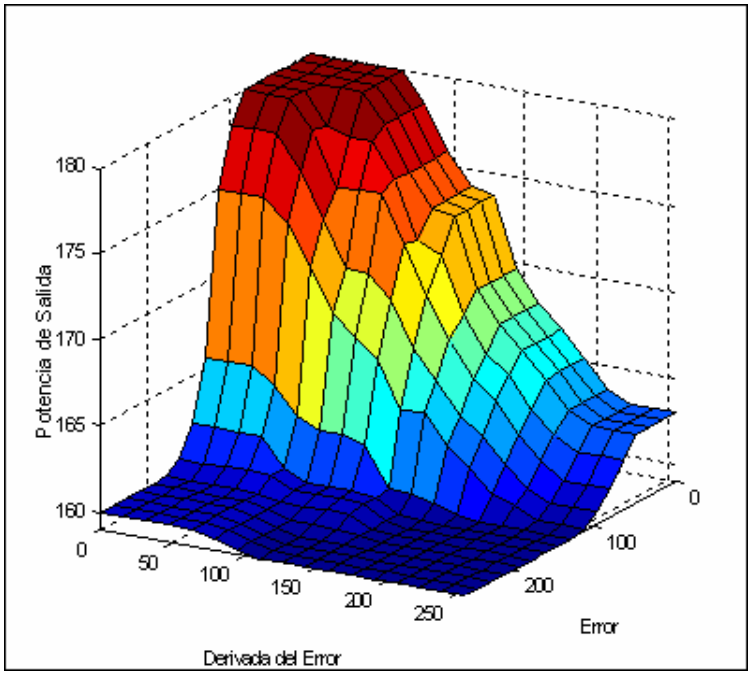


Figura 4.28 Resultado final de la superficie de control en pruebas de dos ejes

Capítulo V Conclusiones

En este proyecto se presento el diseño y la implementación de un controlador difuso tipo Mamdani con acciones de control proporcionales-derivativas para aplicarlo al vuelo autónomo de un mini-helicóptero. Se seleccionó esta aplicación en específico, debido a que representa un reto considerable dado el nivel de complejidad que implican la dinámica y el modelo matemático de este tipo de sistemas.

- A lo largo de esta tesis fue necesario implementar diversos circuitos electrónicos, entre ellos un programador para el microcontrolador PIC18F452, una tarjeta de desarrollo y finalmente para probar el comportamiento del controlador una tarjeta con los siguientes módulos: adquisición y acondicionamiento de la señal de posición angular, potencia para los motores, regulación de voltaje y programación ICSP, así como la configuración del oscilador para que el microcontrolador funcione en modo de alta velocidad. Por otro lado se desarrollaron herramientas de software para realizar el diseño del controlador difuso que permitieron simular las acciones de control y observar gráficamente el comportamiento del mismo. Hasta que se concluyeron los módulos y las herramientas antes mencionadas, se logró realizar las pruebas para observar el desempeño real del sistema.

- Debido a que dentro del software utilizado para diseñar y simular controles difusos existen muy pocos que generan un código de programa para la familia de microcontroladores PIC18FXXX, se requirió programar un kernel difuso que acepta hasta ocho entradas y cuatro salidas crisp con ocho conjuntos cada una, el cual con algunas modificaciones podría emplearse para otras aplicaciones.

- Durante el proceso de selección del sensor de inclinación se optó por utilizar el SCA100T-D01, el cual resultó ser el más adecuado a las necesidades y presupuesto del proyecto aunque el tiempo de respuesta era considerable.

- En la etapa de pruebas se descubrió la presencia de ruido con frecuencia de 33 Hz y amplitud de hasta 2 Vpp en la señal de posición angular. El ruido se originaba por la vibración de la estructura de fibra de carbón propiciada por la operación de los motores a velocidad nominal para la sustentación del DIVAC. Como solución se desarrolló por hardware un filtro paso bajas analógico con aproximación Butterworth de sexto orden para atenuar las frecuencias superiores a 20 Hz. Por software se programó una rutina que calcula el promedio de cuatro valores, cada uno de los cuales se obtiene en periodos de 11 ms, con el fin de realizar la derivada.

- Para simplificar las pruebas, se comenzó a sintonizar el controlador difuso utilizando como referencia un sólo eje, debido a que se reducían las variables que actúan sobre el sistema, teniendo la certeza de que al obtener un resultado aceptable para un solo eje se obtendría un comportamiento análogo para el otro, puesto que el control de los motores se realiza de forma secuencial.

- Para evitar que durante las pruebas realizadas en dos ejes se dañaran la estructura y las hélices del DIVAC, se sujetó por la parte inferior de la estructura con un cordón para limitar el nivel de elevación, sin embargo, se observó que este generaba una tensión sobre la estructura causando una perturbación en el sistema que como consecuencia modificaba su comportamiento.

- Por otro lado, durante esta etapa de pruebas también se observó que el DIVAC tendía a girar sobre su propio eje, haciendo que el cable de alimentación se enredara, generando una torsión sobre la estructura que afectaba de manera significativa el control. Este giro se debe a que la modulación del ancho de pulso para dos motores se realiza mediante el módulo embebido en el microcontrolador para este propósito, mientras que para los dos restantes se realiza a través de una rutina programada para funcionar por medio de interrupciones. A pesar de que el diseño de la rutina por software está realizado para operar de manera similar al módulo de hardware, las pequeñas variaciones entre estos influyen en la potencia entregada a ambos pares de motores por lo que la suma de fuerzas en ese plano es diferente de cero.

- A pesar de las medidas de seguridad tomadas para evitar daños en la estructura y partes del DIVAC, durante la etapa de pruebas se dañaron tres hélices. Esto tuvo como consecuencia que la fuerza con la que se sustentaba el dispositivo fuera diferente en cada una de ellas, afectando el control.

- De las pruebas realizadas para un sólo eje se puede concluir que el diseño del controlador difuso resulta funcional ya que la mayor parte del tiempo tiende a mantenerse horizontal y cuando presenta inclinaciones, realiza las correcciones pertinentes para regresar a dicha posición.

- Sobre la operación del DIVAC considerando ambos ejes se puede concluir que el resultado a pesar de no ser el deseado, resulta aceptable y aunque no se mantiene sustentado por largos periodos de tiempo, el controlador difuso desarrollado realiza las acciones de control de manera razonable.

- Finalmente, se concluye que la lógica difusa es una herramienta de gran utilidad en el control de este sistema porque en general, el resultado de las pruebas muestra que el controlador basado en reglas difusas es el adecuado para generar las acciones requeridas. Las fallas que se presentaron son atribuibles a otros factores tales como el tiempo de adquisición de la señal de posición angular, la falta de una mejor estructura para la realización de pruebas y el daño sufrido en un par de hélices. Por lo anterior, se proponen las siguientes recomendaciones:

Recomendaciones

- Utilizar un sensor con una respuesta en tiempo más rápida ya que con esto se podría saber la posición de cada motor en el instante en el que se corrige y así generar una salida más adecuada.

- Diseñar una estructura de pruebas que le permita mayor grados de libertad al DIVAC reduciendo así las fuerzas externas producidas por la sujeción.

- Agregar conjuntos difusos de entrada a la variable lingüística “Error” debido a que el área de los conjuntos trapezoidales ubicados en los extremos es más grande en comparación con los conjuntos triangulares, lo que origina que el controlador produzca cambios drásticos en las acciones de control para las reglas que incluyen estos conjuntos, ya que presentan un mayor grado de pertenencia.

- Conforme al punto anterior, sería necesario modificar el kernel difuso desarrollado en este trabajo para aceptar más de ocho conjuntos difusos de entrada.

APÉNDICE A El microcontrolador

Un microcontrolador es un dispositivo lógico programable que tiene la capacidad de ejecutar funciones aritméticas y lógicas. Está diseñado para funcionar de manera independiente por lo cual usualmente su arquitectura incluye:

- **Unidad aritmética lógica**: se le llama así a la unidad encargada de realizar las operaciones aritméticas y lógicas sobre operandos que provienen de la memoria principal y que pueden estar almacenados de forma temporal en algunos registros de la propia unidad.

- **Memoria RAM**: significa memoria de acceso aleatorio por sus siglas en inglés (Random Acces Memory). Se trata de un tipo de memoria en la que se puede tanto leer como escribir, datos. Se considera un tipo de memoria volátil, debido a que pierde su contenido al desconectar la energía eléctrica. Se utilizan normalmente como memorias temporales para almacenar resultados intermedios y datos similares no permanentes.

- **Memoria ROM**: se refiere a memoria de sólo lectura por sus siglas en inglés (Read-Only Memory). Es una memoria no destructible, debido a que no se puede escribir sobre ella por lo que conserva intacta la información almacenada, incluso en el caso de interrupción de corriente (memoria no volátil).

- **Puertos de entrada y salida:** son los elementos empleados por el microcontrolador para enviar o recibir señales hacia o provenientes de dispositivos externos.

- **Convertidor analógico-digital**: es un dispositivo que produce un código digital de salida en función de un voltaje analógico de entrada y de un voltaje de referencia de entrada.

- **Oscilador**: es el elemento empleado como componente de control de la frecuencia del microcontrolador, en algunas ocasiones se puede configurar el cristal empleado para aumentar o disminuir la frecuencia de operación.

- **Temporizador**: son contadores digitales los cuales pueden ser programados por software y que realizan un conteo descendente hasta cero o bien comparar un número con un contador, una vez que se ha alcanzado este valor, generan una interrupción en el dispositivo.

Estas características le permiten a un solo circuito integrado adquirir, procesar, almacenar y transmitir datos o acciones de control. Por los motivos anteriores y debido a su bajo costo son muy utilizados en sistemas de control y equipos electrónicos en general.

Por la forma en que se organiza la memoria, la arquitectura de un microcontrolador puede clasificarse como Von Neumann o Harvard. La primera se caracteriza por contar con un solo modulo de memoria. Este único módulo es utilizado como memoria de datos y de programa simultáneamente. Por tal motivo al realizar consultas sucesivas, la ALU debe alternar el uso del bus para leer las instrucciones y escribir los resultados.

Por otro lado, la Harvard cuenta con dos o más módulos de memoria organizados en almacenamiento de datos y de programa. Cada uno de estos posee un bus dedicado que permite a la ALU leer y escribir en cada uno de ellos de manera simultanea.

Debido a la cantidad y complejidad de las instrucciones que permite un microprocesador se puede clasificar como:

CISC (Complex Instruction Set Computer): se refiere a que cuenta con un juego de instrucciones complejas y un número elevado de las mismas, algunas de las instrucciones son muy potentes y realizan operaciones complicadas. La desventaja que presenta es que requiere de varios ciclos de máquina para ejecutar una instrucción.

RISC (Reduced Instruction Set Computer): en este caso se cuenta con un juego de instrucciones reducido y con un número inferior en cuanto a la cantidad, las instrucciones con muy simples y normalmente se ejecutan en un ciclo de máquina.

La velocidad de operación es uno de los parámetros más importantes en la selección de un microcontrolador, pues de esta depende el tiempo que tardará en ejecutar el programa contenido.

El microcontrolador PIC18F452

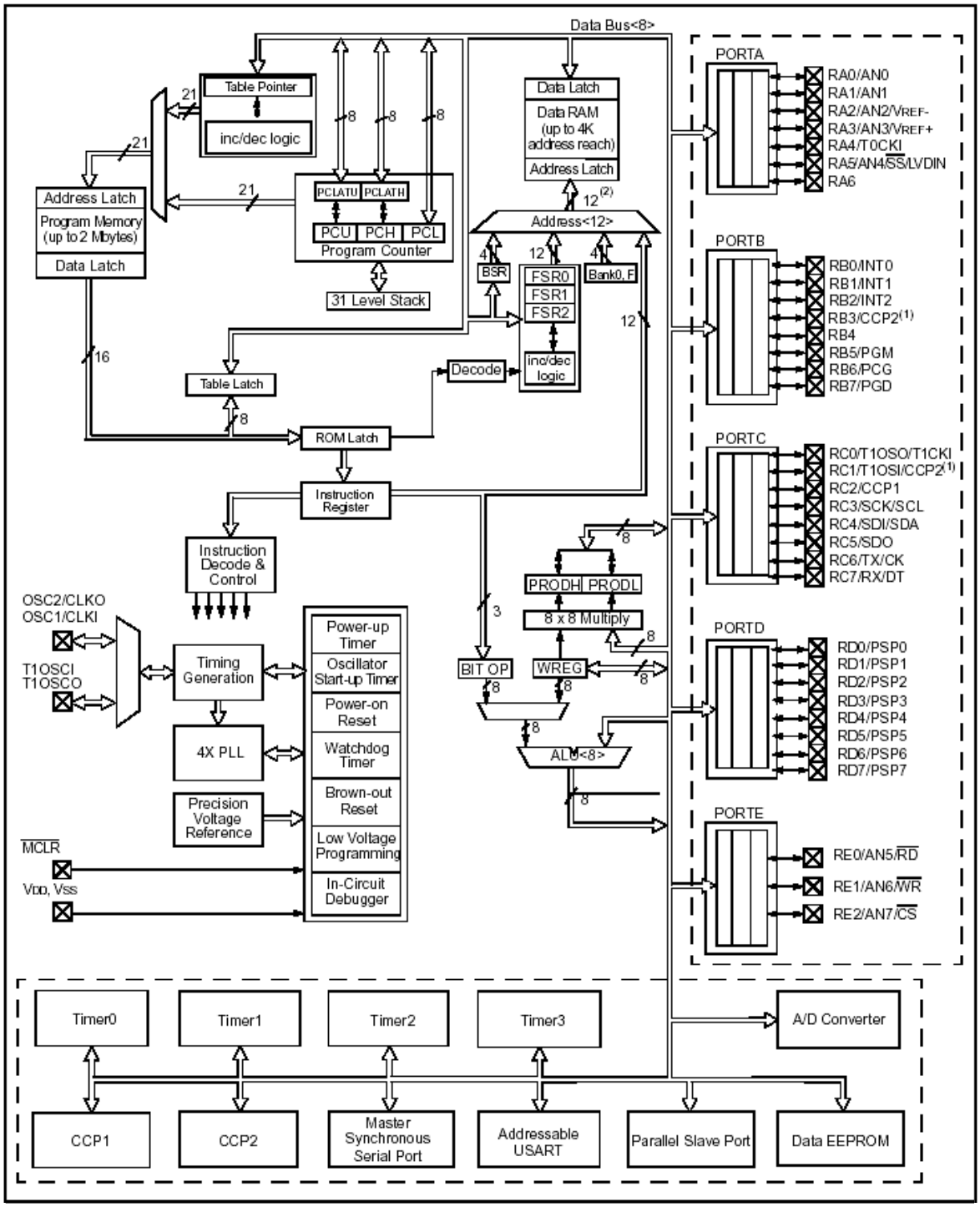


Figura A1 Diagrama de bloques del microcontrolador PIC 18F452

Este microcontrolador pertenece a la familia de dispositivos de alto desempeño desarrollados por MICROCHIP; debido a que esta construido en base a la tecnología CMOS cuenta con un bajo consumo de potencia, un amplio rango de voltaje para su operación (2.0V a 5.5V) además de memoria Flash y EEPROM de alta velocidad, la cual está distribuida como se muestra en la tabla A1:

Dispositivo	Memoria de Programa	Memoria RAM (bytes)	Memoria EEPROM para datos (bytes)
	Flash (bytes)		
PIC18F452	32K	1536	256

Tabla A1 Distribución del espacio de memoria en el PIC18F452

Puede operar en un amplio rango de frecuencias teniendo como limite superior 40 MHz, para alcanzar esta velocidad, se debe activar un PLL interno que multiplica cuatro veces la frecuencia de oscilación, en este caso, el cristal máximo permitido debe ser de 10 Mhz.

Organización de la memoria

La memoria se encuentra organizada en tres bloques, los cuales son:

- Memoria de programa
- Memoria de datos RAM
- Memoria de datos EEPROM

La memoria de datos y la memoria de programa utilizan buses independientes, por lo que se permite el acceso simultáneo a estos bloques.

Organización de la memoria de programa

El microcontrolador es capaz de direccionar hasta 2-Mbytes de memoria haciendo uso de un contador de programa de 21-bits. Tiene implementada una memoria flash de 32 Kbytes en la cual se pueden almacenar hasta 16k instrucciones de una sola palabra. Al leer una localidad que se encuentre más allá de la memoria implementada, dará como resultado ‘0’s. La organización de esta memoria se muestra en bloques en la figura A2.

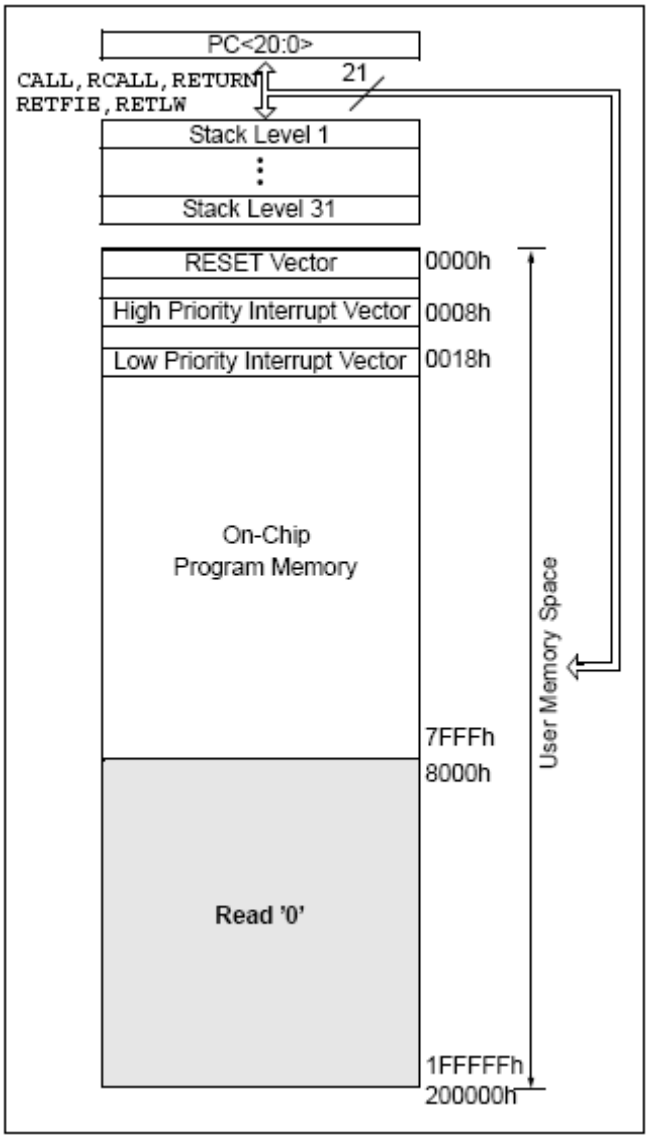


Figura A2 Organización de la memoria de programa

El vector de RESET se encuentra ubicado en la dirección 0000h mientras que los vectores de interrupción están la dirección 0008h y 0018h

En este microcontrolador la memoria de programa está implementada como memoria flash, esta puede ser leída, escrita o borrada durante la operación normal del sistema.

La lectura de la memoria de programa se ejecuta un byte a la vez, mientras que la escritura se realiza en bloques de 8 bytes. El proceso de borrado se realiza en bloques de 64 bytes. Las operaciones de escritura y borrado conllevarán a que las instrucciones sean detenidas hasta que esta última se complete, debido a que la memoria no puede ser accesada durante tal proceso.

Es posible escribir datos en la memoria de programa que no sean instrucciones válidas, en este caso el resultado será una instrucción NOP.

Lectura y escritura de la memoria de programa

Para poder leer y escribir la memoria de datos, existen dos operaciones que permiten al procesador mover bytes entre la memoria de datos y la memoria RAM:

- Lectura de tabla (TBLRD)
- Escritura de tabla (TBLWT)

El tamaño de palabra de la memoria de datos es de 16-bits mientras que el de la memoria RAM es de 8-bits. Las instrucciones de lectura y escritura mueven los datos entre estos dos tipos de memoria a través del registro de ocho bits llamado TABLAT.

La operación TBLRD envía información de la memoria de programa a la memoria RAM, según se muestra en la figura A3.

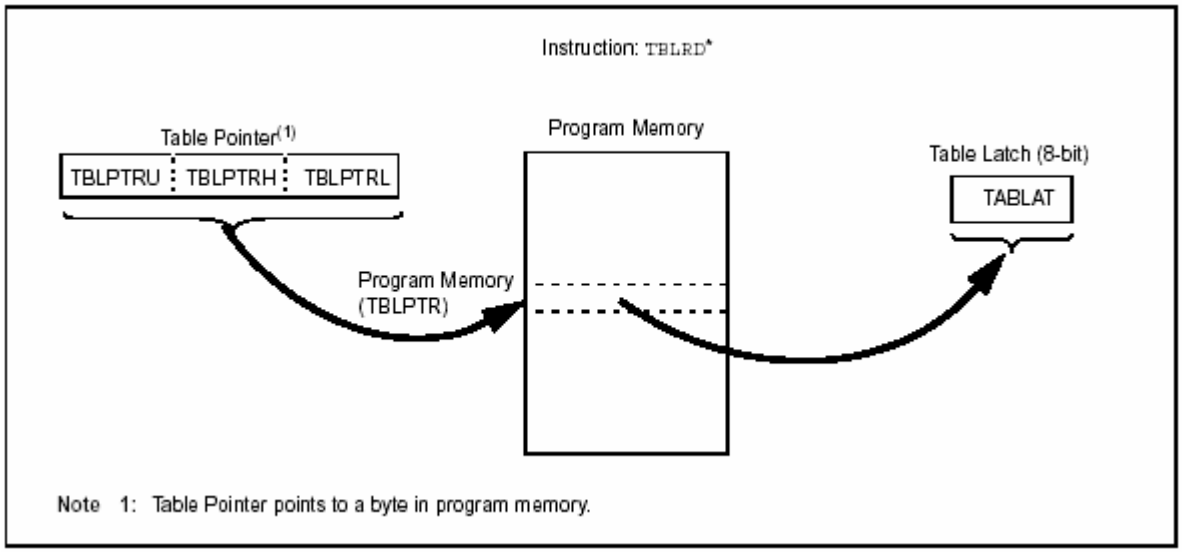


Figura A3 Operación de la instrucción TBLRD

La operación TBLWT escribe datos desde la memoria RAM a la memoria de programa.

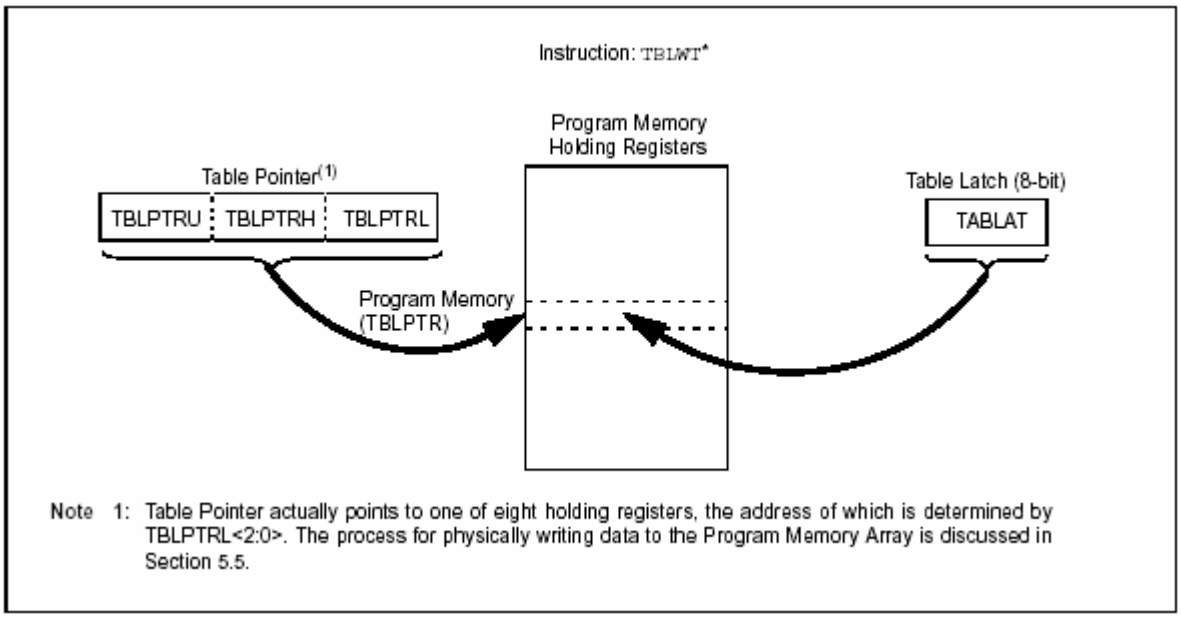


Figura A4 Operación de la instrucción TBLWT

Registros de control

Asociado a las instrucciones TBLRD y TBLWT es necesario el uso de lo siguiente registros:

- EECON1
- EECON2
- TABLAT
- TBLPTR

Registros EECON1 y EECON2

El primero de estos registros es utilizado para controlar el acceso a la memoria. El segundo no es un registro físico, leerlo conllevaría a leer ‘0’s. El registro EECON2 es usado exclusivamente por las operaciones de escritura y por la secuencia de borrado.

El bit de control WR, inicia la operación de escritura y no puede ser limpiado por ninguna operación de software. Sólo puede ser borrado por hardware una vez que ha sido terminada la operación de escritura, esto para prevenir que accidentalmente se termine de forma prematura la operación de escritura.

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

Bit 7 **EEPGD**: Selecciona el tipo de memoria a accesar. Cuando se escribe un 1 se accede a la memoria FLASH. Con un 0 a la memoria de datos EEPROM.
Bit 6 **CFGS**: Selecciona entre memoria y configuración. Cuando se escribe un 1 se accede a los registros de configuración. Con un 0 a la memoria flash o memoria EEPROM.

Bit 5 no implementado.

Bit 4 **FREE**: Habilita el borrado de fila. Cuando se escribe un 1 se borra el contenido de la línea apuntada por TBLPTR en la memoria de programa durante la siguiente instrucción de escritura. Con un 0 se realiza únicamente el borrado.

Bit 3 **WRERR**: Bandera de estado para las memorias flash y EEPROM. Cuando se encuentra activado se ha terminado prematuramente la operación de escritura. Con un 0 se indica que se completó correctamente.

Bit 2 **WREN**: Habilita la escritura en la memoria flash y EEPROM. Cuando se escribe un 1 se permite el ciclo de escritura. Con un 0 se inhibe la escritura en la EEPROM.

Bit 1 **WR**: Control de escritura. Cuando se escribe un 1 se inicia el ciclo de borrado y escritura en la memoria de datos EEPROM o en la memoria de programa. Cuando se lee un 0, el ciclo de escritura se ha completado.

Bit 0 **RD**: Control de lectura. Cuando se escribe un 1 se inicia la lectura de la EEPROM y se borra por hardware.

Registro temporal de tabla TABLAT

Es un registro de ocho bits mapeado en los registros de funciones especiales. Se utiliza para mantener datos de ocho bits durante la transferencia entre la memoria de programa y la memoria RAM.

Registro apuntador de tabla TBLPTR

Direcciona un byte cualquiera dentro de la memoria de programa. Está compuesto por tres registros de funciones especiales: alto, medio y bajo (TBLPTRU:TBLPTRH:TBLPTRL). Estos registros se unen para formar un apuntador de 22 bits. Los primeros 21 bits permiten al dispositivo direccionar hasta 2 Mbytes de memoria. El bit número 22 permite el acceso a la identificación del dispositivo, la del usuario y a los bits de configuración.

El apuntador de tabla es usado por los comandos de escritura y lectura TBLRD y TBLWT. Estas instrucciones pueden modificar el valor del apuntador, incrementando o

decrementando una unidad la dirección de la localidad a la que éste apunta, esta acción puede realizarse antes o después de haber consultado el dato. La tabla A2 muestra los modificadores y las acciones que ejerce cada uno de ellos.

Instrucción	Operación en el apuntador de tabla
TBLRD*	No se modifica el TBLPTR
TBLWT*	
TBLRD*+	Se incrementa el TBLPTR después de una operación de lectura/escritura
TBLWT*+	
TBLRD*-	Se decrementa el TBLPTR después de una operación de lectura/escritura
TBLWT*-	
TBLRD+*	Se incrementa el TBLPTR antes de una operación de lectura/escritura
TBLWT+*	

Tabla A2 Modificadores del apuntador TBLPTR

Proceso de lectura de la memoria de programa

La instrucción TBLRD se usa para enviar datos desde la memoria de programa hacia la memoria RAM. La lectura de la memoria de programa se hace un byte a la vez.
Cuando se ejecuta la instrucción TBLRD, se coloca el byte apuntado por TBLPTR en el registro temporal TABLAT.
La memoria de programa interna está organizada en palabras. El bit menos significativo de la dirección selecciona entre la parte alta y baja de la palabra de 16 bits. El proceso de lectura se muestra de forma gráfica en la figura A5.

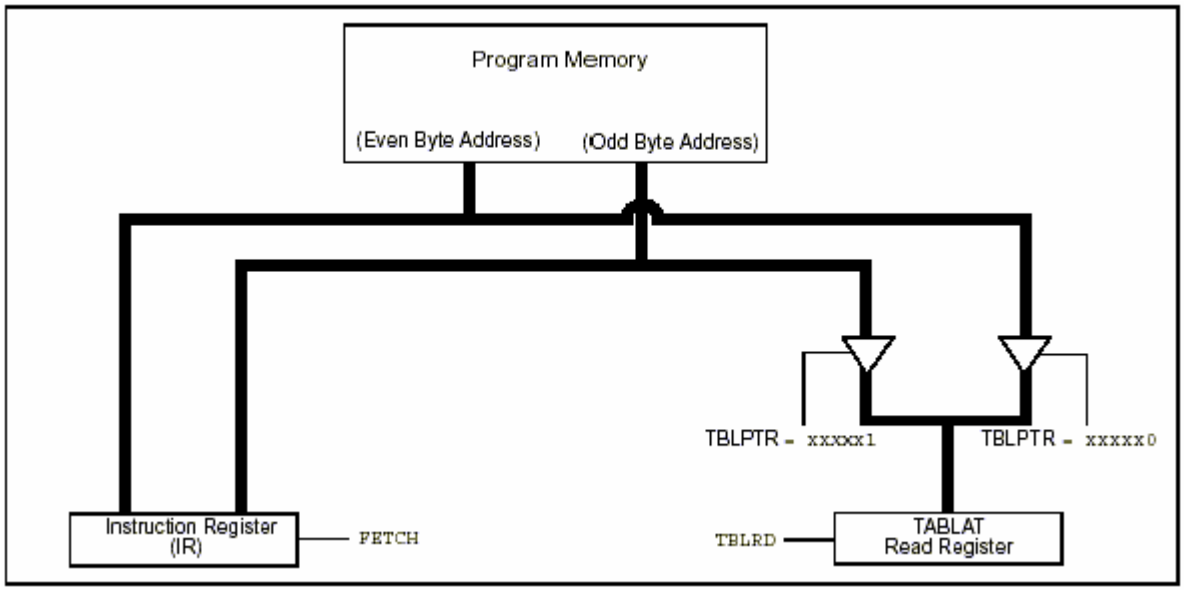


Figura A5 Proceso de lectura de la memoria de programa

Organización de la memoria de datos

La memoria de datos está implementada como RAM estática. Cada registro tiene una dirección de 12 bits permitiendo hasta 4096 bytes de esta. El mapa de memoria se encuentra dividido en 16 bancos, que contienen cada uno 256 bytes, los cuales son seleccionados a través del registro de selección de bancos BSR (Bank Select Register). La organización de la memoria por bancos se muestra en la figura A6.

Dentro de esta memoria podemos encontrar registros de función especial SFR (Special Function Registers), lo cuales se muestran en la figura A7 y registros de propósito general GPR (General Purpose Registers) mostrados en la figura A6.

Los registros de función especial son registro usados por el CPU y por los módulos periféricos para controlar las diferentes operaciones del dispositivo. Pueden ser clasificados en dos conjuntos, aquellos que se encuentran asociados con la función del “núcleo” y los relacionados con la operación de las características periféricas. Se encuentran implementados a partir de la última localidad del banco 15 (0xFFF) y se extienden en sentido descendente.

Los registros de propósito general son empleados para almacenamiento de datos y de variables temporales en las aplicaciones de usuario. Están implementados a partir de la primera localidad del banco 0 y crecen en sentido ascendente hasta la localidad 0x600.

El acceso a la memoria de datos se puede realizar de manera directa o indirecta. El direccionamiento directo requiere del uso del registro BSR, mientras que el direccionamiento indirecto requiere el uso del FSR y del correspondiente operando de archivo indirecto (INDF). Cada FSR contiene una dirección de 12 bits que puede ser utilizada para acceder cualquier localidad en el mapa de memoria de datos sin la necesidad de utilizar los bancos.

El conjunto de instrucciones y la arquitectura de este microcontrolador, permiten operaciones a través de todos los bancos, lo cual se logra utilizando direccionamiento indirecto o bien la instrucción MOVFF, la cual mueve un valor de un registro a otro.

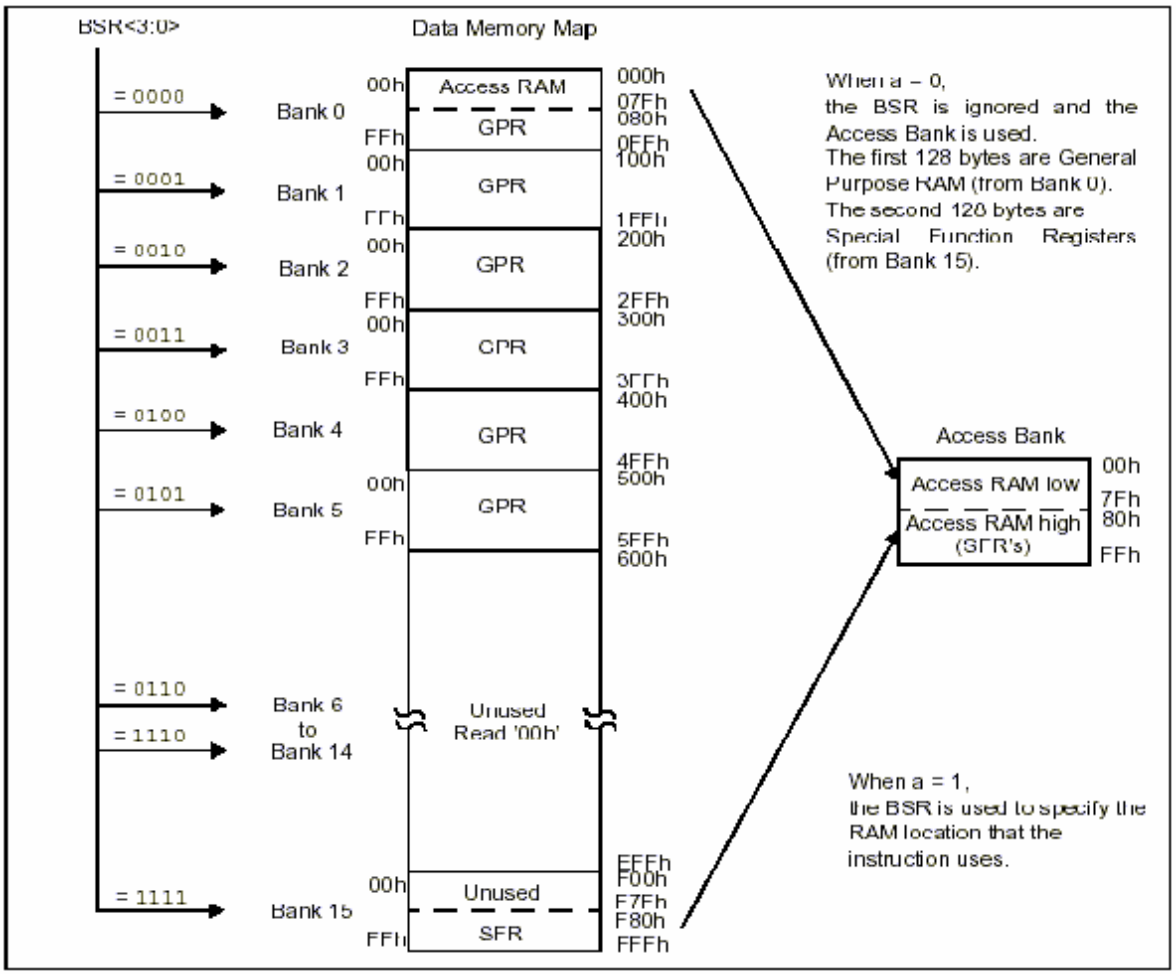


Figura A6 Mapa de memoria de datos

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(3)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(3)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(3)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(3)	FBCh	CCPR2H	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2(3)	FBBh	CCPR2L	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—	F97h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—	F96h	TRISE(2)
FF5h	TABLAT	FD5h	T0CON	FB5h	—	F95h	TRISD(2)
FF4h	PRODH	FD4h	—	FB4h	—	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—
FF0h	INTCON3	FD0h	RCON	FB0h	—	F90h	—
FEFh	INDF0(3)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—
FEEh	POSTINC0(3)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—
FEDh	POSTDEC0(3)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(2)
FECh	PREINC0(3)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(2)
FEBh	PLUSW0(3)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—
FE7h	INDF1(3)	FC7h	SSPSTAT	FA7h	EECON2	F87h	—
FE6h	POSTINC1(3)	FC6h	SSPCON1	FA6h	EECON1	F86h	—
FE5h	POSTDEC1(3)	FC5h	SSPCON2	FA5h	—	F85h	—
FE4h	PREINC1(3)	FC4h	ADRESH	FA4h	—	F84h	PORTE(2)
FE3h	PLUSW1(3)	FC3h	ADRESL	FA3h	—	F83h	PORTD(2)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	—	FA0h	PIE2	F80h	PORTA

Figura A7 Mapa de registros de función especial

Banco de acceso

El banco de acceso es una mejora en la arquitectura muy útil para la optimización de código en compiladores de lenguaje C.
Esta región de la memoria de datos puede ser utilizada para:

- Valores intermedios de cómputo
- Variables locales en subrutinas
- Variables comunes
- Evaluación más rápida / control de los registros SFR

El banco de acceso está comprendido a partir de los 128 bytes superiores del banco 15 (RAM de acceso alto) y de los 128 bytes inferiores en el banco 0 (RAM de acceso bajo).

Mediante un bit en la instrucción del microcontrolador, se especifica si la operación se realizará en el banco seleccionado por el registro BSR o en el banco de acceso. Este bit es denotado por la “a” en los bits de acceso

Cuando se ocupa el banco de acceso (a=0), la última dirección en la RAM de acceso bajo es seguida por la primera dirección de la RAM de acceso alto. La RAM de acceso alto mapea los registros SFR de manera que pueden ser accesados sin la necesidad de software escrito en el encabezado. Esto es útil para comprobar las banderas del registro de estado y para modificar los bits de control.

Registro de selección de bancos

La necesidad de contar con una gran cantidad de memoria de propósito general obliga la implementación de un esquema donde la memoria RAM está banqueada. La memoria de datos se encuentra dividida en 16 bancos. Cuando se utiliza direccionamiento directo, el registro BSR debe estar configurado para el banco deseado. En este registro los bits BSR<3:0> contiene los cuatro bits superiores de la dirección en RAM que es de 12 bits. Los bits BSR<7:4> siempre se leerán como ceros y su escritura no tendrá ningún efecto. La figura A8 muestra un diagrama donde se muestra la forma en que se realiza el direccionamiento directo.

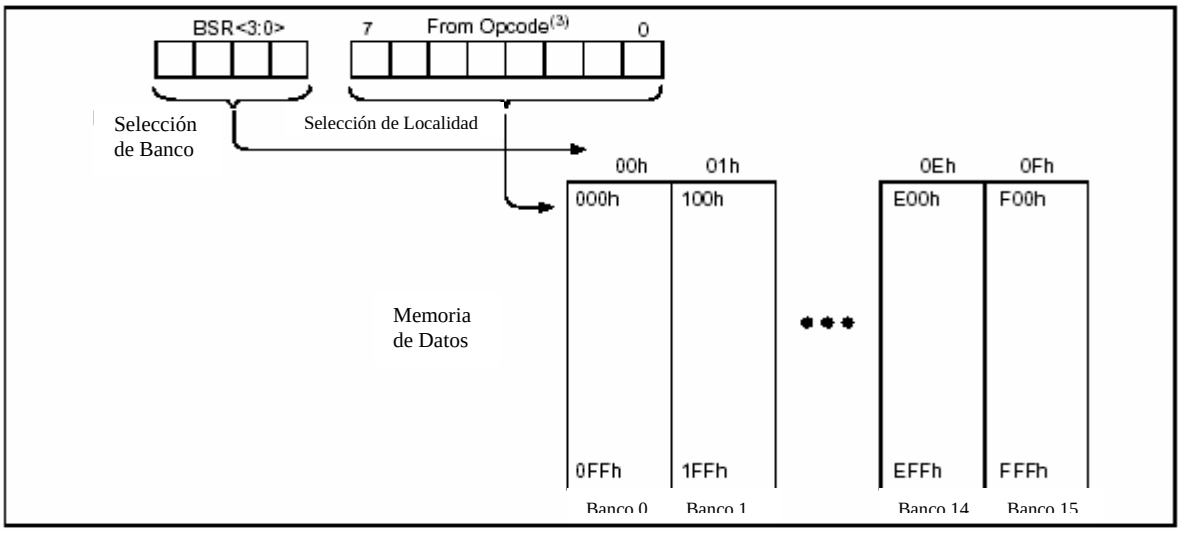


Figura A8 Diagrama a bloques de la operación del direccionamiento directo

Para facilitar la selección de bancos, el microcontrolador cuenta con la instrucción MOVLB. En caso de que el banco seleccionado no se encuentre implementado, la lectura de este será registrada como cero y las escrituras serán ignoradas.

Direccionamiento indirecto y registros INDF y FSR

El direccionamiento indirecto es un modo de direccionar la memoria de datos. Se utiliza un registro denominado FSR como apuntador a la localidad de la memoria de datos que va a ser leída o escrita. Debido a que el apuntador se encuentra en memoria RAM, el contenido puede ser modificado por el programa, lo cual resulta muy útil para implementar tablas de datos y stacks por software en la memoria de datos. El direccionamiento indirecto se realiza como se muestra en el diagrama de la figura A8.

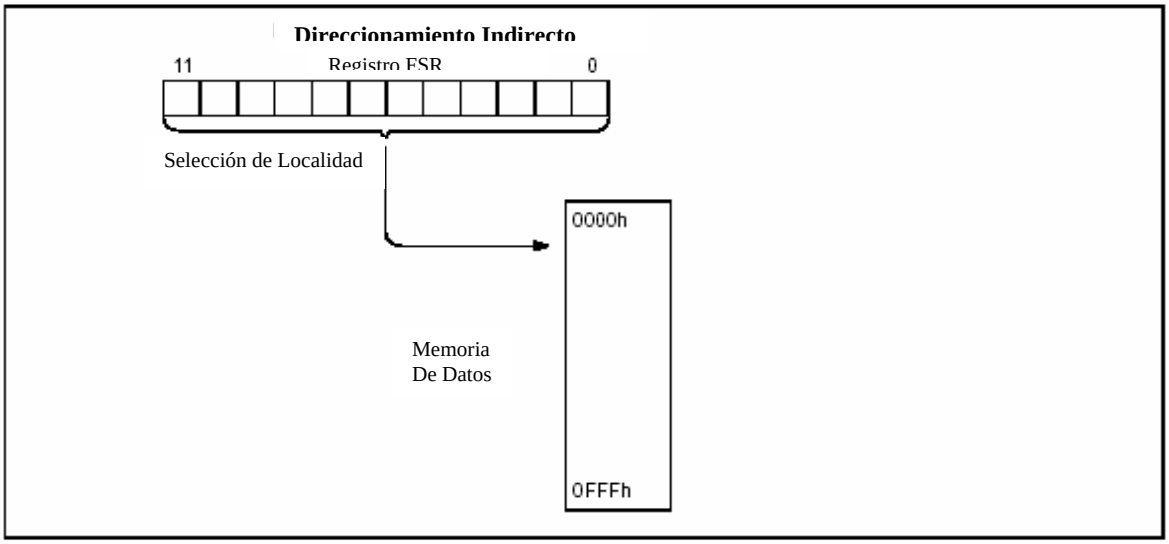


Figura A9 Diagrama a bloques de la operación del direccionamiento Indirecto

Este direccionamiento es posible mediante la utilización de los registros INDF. Cualquier instrucción que utilice el registro INDF puede acceder al registro apuntado por el registro de selección de archivos FSR. Los registros INDF no son registros físicos, sino que direccionando estos, se accede a las direcciones contenidas en el registro FSR funcionando como apuntador, por lo que a este proceso se le conoce como direccionamiento indirecto.
Existen tres registros para realizar el direccionamiento indirecto. Con el fin de direccionar completamente el espacio destinado para memoria de datos, es decir, 4096 bytes, estos registros son de 12 bits por lo que debido a la arquitectura del microcontrolador se deben utilizar dos registros de 8 bits. Estos registros son:

- FSR0: compuesto de las localidades FSR0H: FSR0L
- FSR1: compuesto de las localidades FSR1H: FSR1L
- FSR2: compuesto de las localidades FSR2H: FSR2L

Además, existen tres registros INDF0, INDF1 e INDF2 los cuales no se encuentran físicamente implementados. Las operaciones de lectura y escritura a estos registros activan el direccionamiento indirecto, con el valor en el registro FSR correspondiente como dirección del dato. Cuando una instrucción escribe un valor en el registro INDF0, este en realidad será escrito en la dirección apuntada por FSR0H:FSR0L.

Cuando los registros INDF0, INDF1 e INDF2 son leídos indirectamente por medio del registro FSR, se modifica el registro de estado del microcontrolador, mientras que cuando se efectúan operaciones de escritura, esto será equivalente a una instrucción NOP y el registro de estado no será afectado.

Operaciones en el direccionamiento indirecto

Cada registro FSR cuenta con un registro INDF asociado además de cuatro registros de dirección adicionales. Realizando una operación sobre cualquiera de estos cinco registros se determina la forma en que se modificará el FSR durante el direccionamiento indirecto.
Cuando se ha accesado el dato a alguno de los cinco registros INDF. La dirección seleccionada configurará el registro FSR para realizar cualquiera de las siguientes acciones:

- INDFn: sin acción, no cambia el valor en el FSRn después de un acceso indirecto.
- POSTDECn (post-decremento): auto decrementa el FSRn después de un acceso indirecto.
- POSTINCn (post-incremento): auto incrementa el FSRn después de un acceso indirecto.
- PREINCn (pre-incremento): auto incrementa el FSRn antes de un acceso indirecto.
- PLUSWn: utiliza el valor en el registro w como un offset al FSRn. Esta operación no modifica el valor del WREG ni del FSRn después de un acceso indirecto.

Cuando se utilizan las características de auto-incremento o auto-decremento, el efecto causado en el registro FSR no afecta el registro de estado del microcontrolador, pero sí se modifican los 12 bits de la dirección. Cuando se presenta un desbordamiento después de incrementar el FSRnL, se incrementará automáticamente el FSRnH.

Cada FSR cuenta con una dirección asociada, mediante la cual se puede realizar un acceso indirecto indizado. Cuando ocurre el acceso de un dato a la localidad INDFn, el FSR es configurado para sumar el valor signado almacenado en el registro W al valor guardado en el FSR y así formar la dirección para realizar el direccionamiento indirecto y el valor del FSR no es modificado. La figura A10 muestra la forma en que se realiza el direccionamiento indirecto.

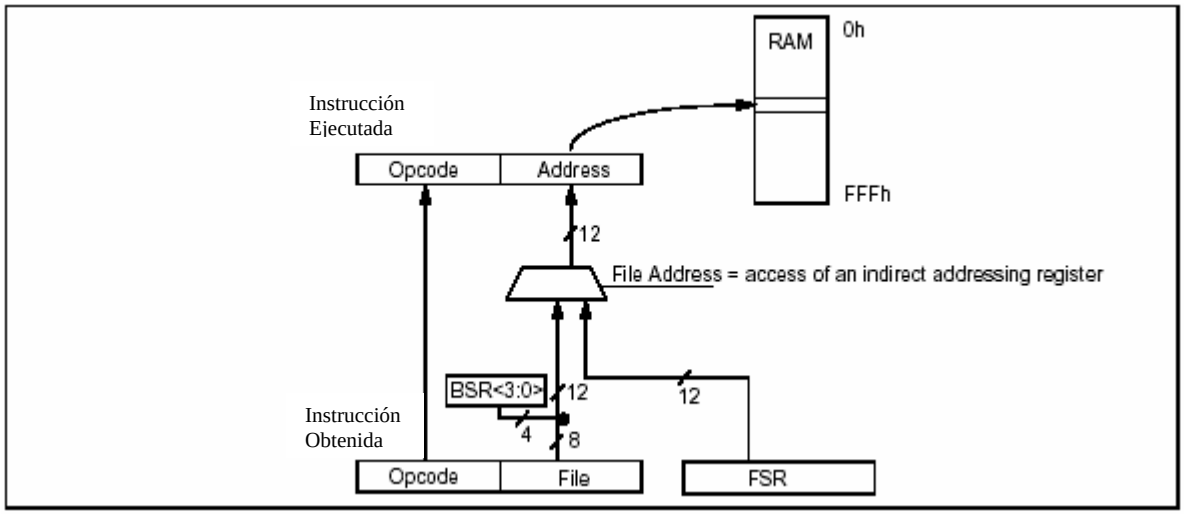


Figura A10 Operación del direccionamiento indirecto

Memoria de datos EEPROM

La memoria EEPROM puede ser leída, escrita o borrada durante la operación normal del microcontrolador. La memoria de datos no se encuentra directamente mapeada, por lo que debe ser direccionada de forma indirecta a través de los siguientes registros de funciones especiales.

- EECON1
- EECON2
- EEDATA
- EEADR

Cuando se interactúa con un bloque de datos, el registro especial EEDATA guarda los ocho bits del dato que se esté leyendo o escribiendo y el registro EEADR guarda la dirección de la localidad de la memoria EEPROM que se está accesando.

Cuando se escribe un byte en la EEPROM, primero se borra la localidad y posteriormente se escribe el nuevo dato. El tiempo de escritura se controla por un temporizador embebido en el chip. Este tiempo varía en función del voltaje y la temperatura de uso.

Registros EECON1 y EECON2

Los bits de control RD y WR inician la lectura y escritura en la memoria de datos EEPROM respectivamente. Estos bits solo pueden ser habilitados mediante software. Se borran por hardware en el momento en que son completadas las tareas de lectura y escritura.
Sólo se permite la escritura cuando el bit WREN se encuentra activo. Cuando se enciende el microcontrolador, el bit es borrado.

Proceso de lectura de memoria de datos EEPROM

Para poder leer una localidad de la memoria de datos el usuario primero debe escribir la dirección en el registro especial EEADR, borrar los bits de control EEPGD, CFGS y posteriormente habilitar el bit RD

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

Una vez realizada esta secuencia, los datos se encuentran disponibles a partir del siguiente ciclo de instrucción, en este momento se puede leer el registro especial EEDATA y hasta que se efectúe una nueva lectura o se realice una escritura por el usuario.

Interrupciones

El PIC18F452 tiene múltiples fuentes de interrupción que pueden ser generadas a partir de hardware interno o bien por dispositivos periféricos. Cada una de estas, cuenta con una característica de control que permite asignarles un determinado nivel de prioridad, el cual puede ser alto o bajo.

Dispone de diez registros que controlan la operación de las interrupciones. Estos registros son:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2
- PIE1, PIE2
- IPR1, IPR2

Cada fuente de interrupción, usa tres bits para el control de su operación.

- **Bit de bandera:** Indica cuando la interrupción ha ocurrido.
- **Bit de habilitación:** Permite que el programa en ejecución salte a la dirección del vector de interrupción cuando la bandera esta activada.
- **Bit de prioridad:** Asigna el nivel de prioridad.

Cuando se requiere asignar niveles de prioridad a las fuentes de interrupción, se debe activar el bit siete (IPEN) del registro RCON, en el caso de las interrupciones de baja prioridad se debe habilitar el bit seis (GIEL) del registro INTCON y su vector se localiza en la dirección 000018h. Para las de alta prioridad se tiene que habilitar el bit siete (GIEH) del mismo registro y su vector se encuentra en la dirección 000008h. Si no se asignan niveles de prioridad, todas las interrupciones saltan a la dirección 000008h y los bits seis (PEIE) y siete (GIE) del registro INTCON son los encargados de habilitar las fuentes de interrupción periféricas e internas, respectivamente.

En el momento en que se presenta una interrupción se activa la bandera de la fuente que la generó. El microcontrolador almacena en el stack la dirección a la que va a regresar el programa en ejecución después de atender la rutina de servicio y el contador de programa carga la dirección del vector de interrupción dependiendo de su nivel. Por último se borra el bit de interrupción global para evitar otras interrupciones a menos que se trate de interrupciones de alta prioridad, las cuales, en cualquier momento pueden interrumpir a las de baja.

Es necesario desactivar por software la bandera de interrupción durante la ejecución de la rutina de servicio, para evitar interrupciones recursivas. Una vez que se ha atendido completamente, se habilita nuevamente el bit de interrupción global para esperar la próxima interrupción, esta acción se realiza mediante la instrucción de regreso de interrupción RETFIE.

Registros INTCON

Este tipo de registros son de lectura y escritura, contienen diferentes bits para el control de las prioridades, habilitación de interrupciones y banderas de estado.

- **Registro INTCON**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

Bit 7 **GIE/GIEH:** Bit de control de Interrupción Global. Este bit depende directamente del bit IPEN. Cuando se encuentra activo, habilita o deshabilita las interrupciones de alta prioridad, en caso contrario, actúa de la misma forma sobre las interrupciones no mascarables.

Bit 6 **PEIE/GIEL:** Bit de control de Interrupción para periféricos. Este bit depende directamente del bit IPEN. Cuando se encuentra activo, habilita o deshabilita las interrupciones de baja prioridad, en caso contrario, actúa de la misma forma sobre las interrupciones periféricas no mascarables.

Bit 5 **TMR0IE:** Habilita/deshabilita la interrupción por desbordamiento en el Timer0.

Bit 4 **INT0IE:** Habilita/deshabilita la interrupción externa INT0 que se activa por el bit 0 del puerto B.

Bit 3 **RBIE:** Habilita/deshabilita la interrupción por cambio de estado en la entrada de los pines 4,5,6 y 7 del puerto B.

Bit 2 **TMR0IF:** Bandera de estado del Timer0. Indica si se presentó una interrupción por desbordamiento del Timer0.

Bit 1 **INT0IF:** Bandera de estado del INT0. Indica si se presentó una interrupción por INT0.

Bit 0 **RBIF:** Bandera de estado por cambio de entraba en el puerto B. Indica si se presentó una interrupción por cambio en la entrada del puerto B.

- **Registro INTCON 2**

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7							bit 0

Bit 7 $\overline{\text{RBPU}}$ **:** Habilita/deshabilita el pull-up interno para el puerto B.

Bit 6 **INTEDG0:** Selecciona el flanco para que se genere la interrupción externa INT0. Escribiendo un 1 será por flanco de subida y con un 0 por flanco de bajada.

Bit 5 **INTEDG1:** Selecciona el flanco para que se genere la interrupción externa INT1. Escribiendo un 1 será por flanco de subida y con un 0 por flanco de bajada.

Bit 4 **INTEDG0:** Selecciona el flanco para que se genere la interrupción externa INT2. Escribiendo un 1 será por flanco de subida y con un 0 por flanco de bajada.

Bit 3 no implementado.

Bit 2 **TMR0IP:** Selecciona la prioridad para la interrupción por desbordamiento en el temporizador 0. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

Bit 1 no implementado.

Bit 0 **RBIP:** Selecciona la prioridad para la interrupción por cambio de estado en el puerto B. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

- **Registro INTCON 3**

R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
bit 7							bit 0

Bit 7 **INT2IP:** Selecciona la prioridad para la interrupción externa INT2. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

Bit 6 **INT1IP:** Selecciona la prioridad para la interrupción externa INT1. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

Bit 5 no implementado.

Bit 4 **INT2IE:** Habilita/deshabilita la interrupción externa INT2.

Bit 3 **INT1IE:** Habilita/deshabilita la interrupción externa INT1.
Bit 2 no implementado.

Bit 1 **INT2IF:** Bandera de estado de la interrupción externa INT2. Indica si se presentó la interrupción externa en el pin 2 del puerto B.

Bit 0 **INT1IF:** Bandera de estado de la interrupción externa INT1. Indica si se presentó la interrupción externa en el pin 1 del puerto B.

Registros PIR

Contienen los bits de bandera de estado para las interrupciones periféricas. Debido al número de fuentes de interrupción externa, existen dos registros para su control (PIR1 y PIR2).

- **Registro PIR1**

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

Bit 7 **PSPIF:** Bandera de estado que indica si se ha efectuado una lectura o una escritura en el puerto paralelo esclavo.

Bit 6 **ADIF:** Bandera de estado que indica cuando se ha completado una conversión A/D.

Bit 5 **RCIF:** Bandera de estado que indica cuando se recibe información por el USART, en este caso el búfer se encuentra lleno.

Bit 4 **TXIF:** Bandera de estado que indica cuando se transmite información por el USART, en este caso el búfer se encuentra vacío.

Bit 3 **SSPIF:** Bandera de estado que indica cuando la transmisión o la recepción en el puerto síncrono serial maestro se ha completado.

Bit 2 **CCP1IF:** Bandera de estado para el CCP1. Indica cuando ha ocurrido una captura o una comparación en el registro del TMR1.

Bit 1 **TMR2IF:** Bandera de estado que indica cuando el valor del registro TMR2 alcanza el valor del PR2 y se produce la interrupción en el temporizador 2.

Bit 0 **TMR1IF:** Bandera de estado que indica si la interrupción por desbordamiento en el Temporizador 1 se ha efectuado.

- **Registro PIR2**

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF
bit 7							bit 0

Bit 7-5 no implementados

Bit 4 **EEIF:** Bandera de estado que indica cuando se ha completado la operación de escritura de datos en la EEPROM o la FLASH.

Bit 3 **BCLIF:** Bandera de estado que indica cuando existe una colisión en el bus del I^2C.

Bit 2 **LVDIF:** Bandera de estado que indica cuando se detecta bajo voltaje y se produce la interrupción asociada.

Bit 1 **TMR3IF:** Bandera de estado que indica si la interrupción por desbordamiento en el Temporizador 3 se ha efectuado.

Bit 0 **CCP2IF:** Bandera de estado para el CCP2. Indica cuando ha ocurrido una captura o una comparación en el registro del TMR1.

Registros PIE

Contienen los bits para habilitar o deshabilitar las interrupciones por periféricos. Debido al número de fuentes de interrupción externa, existen dos registros para su control (PIE1 y PIE2). Cuando el bit IPEN se encuentra deshabilitado se debe activar el bit PEIE para poder tener acceso a este tipo de interrupciones.

- **Registro PIE1**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Bit 7 **PSPIE:** Habilita/deshabilita la interrupción para la lectura y escritura del puerto paralelo esclavo.

Bit 6 **ADIE:** Habilita/deshabilita la interrupción en el convertidor A/D.

Bit 5 **RCIE:** Habilita/deshabilita la interrupción en la recepción del módulo USART.

Bit 4 **TXIE:** Habilita/deshabilita la interrupción en la transmisión del módulo USART.

Bit 3 **SSPIE:** Habilita/deshabilita la interrupción en el puerto síncrono serial maestro.

Bit 2 **CCP1IE:** Habilita/deshabilita la interrupción en el módulo CCP1.

Bit 1 **TMR2IE:** Habilita/deshabilita la interrupción cuando el valor del TMR2 alcanza el valor del PR2.

Bit 0 **TMR1IE:** Habilita/deshabilita la interrupción por desbordamiento en el temporizador

- **Registro PIE2**

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE
bit 7							bit 0

Bit 7-5 no implementados.

Bit 4 **EEIE:** Habilita/deshabilita la interrupción por escritura de datos en la EEPROM o la FLASH.

Bit 3 **BCLIE:** Habilita/deshabilita la interrupción cuando existe una colisión en el bus del módulo I^2C.

Bit 2 **LVDIE:** Habilita/deshabilita la interrupción por detección de bajo voltaje.

Bit 1 **TMR3IE:** Habilita/deshabilita la interrupción por desbordamiento en el temporizador 3.

Bit 0 **CCP2IE:** Habilita/deshabilita la interrupción en el módulo CCP2.

Registros IPR

Controlan los bits de prioridad para las interrupciones por periféricos. Debido al número de fuentes de interrupción externa, existen dos registros para su control (IPR1 e IPR2). Para utilizar la prioridad en las interrupciones se debe activar el bit IPEN.

- **Registro IPR1**

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

Bit 7 **PSPIP:** Selecciona la prioridad para la interrupción en el puerto paralelo esclavo. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

Bit 6 **ADIP:** Selecciona la prioridad para la interrupción en el convertidor A/D. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

Bit 5 **RC2IP:** Selecciona la prioridad para la interrupción en la recepción del módulo USART. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.
Bit 4 **TXIP:** Selecciona la prioridad para la interrupción en la transmisión del módulo USART. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

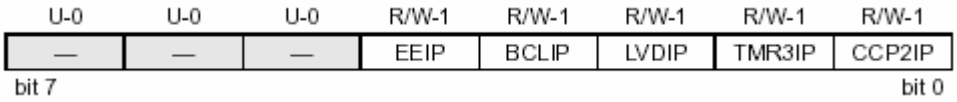
Bit 3 **SSPIP:** Selecciona la prioridad para la interrupción en el puerto síncrono serial maestro. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

Bit 2 **CCP1IP:** Selecciona la prioridad para la interrupción en el módulo CCP1. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

Bit 1 **TMR2IP:** Selecciona la prioridad para la interrupción cuando el valor del TMR2 alcanza el valor del PR2. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

Bit 0 **TMR1IP:** Selecciona la prioridad para la interrupción por desbordamiento en el temporizador 1. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

- **Registro IPR2**



Bit 7-5 no implementados.

Bit 4 **EEIP:** Selecciona la prioridad para la interrupción por escritura de datos en la EEPROM o la FLASH. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

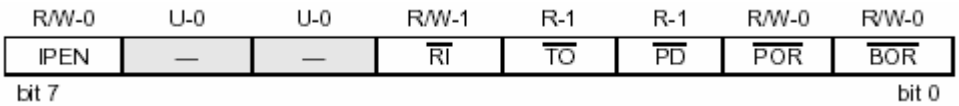
Bit 3 **BCLIP:** Selecciona la prioridad para la interrupción cuando existe una colisión en el bus del módulo I^2C. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

Bit 2 **LVDIP:** Selecciona la prioridad para la interrupción por detección de bajo voltaje. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

Bit 1 **TMR3IP:** Selecciona la prioridad para la interrupción por desbordamiento en el temporizador 3. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

Bit 0 **CCP2IP:** Selecciona la prioridad para la interrupción en el módulo CCP2. Escribiendo un 1 se le asigna alta prioridad y con un 0 baja prioridad.

- **Registro RCON:** Registro que contiene el bit con el cuál se habilita la prioridad en las interrupciones (IPEN).



Bit 7 **IPEN:** Habilita/deshabilita los niveles de prioridad en las interrupciones.

Bit 6-5 no implementados.

Bit 4 $\overline{\textbf{RI}}$ **:** Bandera de estado para la instrucción de RESET.

Bit 3 $\overline{\textbf{TO}}$ **:** Bandera de estado que indica la finalización del tiempo de espera por el Watchdog.

Bit 2 $\overline{\textbf{PD}}$ **:** Bandera de estado cuando se ejecuta la instrucción SLEEP.

Bit 1 $\overline{\textbf{POR}}$ **:** Bandera de estado para el Power-on Reset.

Bit 0 $\overline{\textbf{BOR}}$ **:** Bandera de estado para el Brown-out Reset.

Puertos de entrada y salida

Este microcontrolador cuenta con cinco puertos denominados con las letras A, B, C, D y E. Las funciones de cada uno de ellos varían según la habilitación o inhibición de los diferentes dispositivos de hardware embebidos en el microcontrolador. Esto se debe a que algunos de sus pines están multiplexados. De manera general, cuando un periférico se encuentra activo, el pin no puede ser utilizado como entrada o salida de propósito general.
Cada puerto cuenta con tres registros para su funcionamiento, a partir de los cuales se configura su operación.

- Registro TRIS: registro de dirección de datos, cuando se trata de un puerto bidireccional se puede especificar cuando un pin tendrá la función de entrada escribiendo un uno en el registro del pin correspondiente al puerto o como salida cuando se escribe un cero.
- Registro PORT: registro donde se leen los niveles presentes en los pines del microcontrolador.
- Registro LAT: latch de salida, utilizado para operaciones de lectura, modificación y escritura del valor que presentarán los pines del puerto.

Puerto A

Se trata de un puerto bidireccional de 7 bits; sus registros son conocidos como TRISA, PORTA y LATA. El pin cuatro (RA4) está multiplexado con la entrada de reloj del modulo del temporizador 0. Los demás pines están multiplexados con las entradas analógicas y los voltajes de referencia para la operación del convertidor analógico-digital.

La operación de cada uno de ellos se selecciona escribiendo en los bits del registro de control ADCON1. Aún cuando estos pines del puerto se están ocupando como entradas analógicas, puede ser modificada su dirección escribiendo en el registro TRISA, por lo que se debe asegurar que están configurados como entradas mientras se emplea el convertidor A/D.La tabla A3 muestra la función de cada pin del puerto A.

Nombre	Bit	Función
RA0/AN0	0	Entrada/Salida o Entrada Analógica
RA1/AN1	1	Entrada/Salida o Entrada Analógica
RA2/AN2/V_{REF-}	2	Entrada/Salida o Entrada Analógica o Referencia de voltaje negativo para el convertidor A/D.
RA3/AN3/V_{REF+}	3	Entrada/Salida o Entrada Analógica o Referencia de voltaje positivo para el convertidor A/D.
RA4/T0CKI	4	Entrada/Salida o Entrada de reloj externo para temporizador 0
RA5/$\overline{SS}$ /AN4/LVDIN	5	Entrada/Salida o Selección de entrada como esclavo para puerto serial síncrono o Entrada analógica o Entrada para detección de bajo voltaje.
OSC2/CLKO/RA6	6	OSC2 o Entrada de reloj o Entrada/Salida

Tabla A3 Funciones del Puerto A

Puerto B

Se trata de un puerto bidireccional de 8 bits; sus registros son conocidos como TRISB, PORTB y LATB. Cada pin cuenta con un pull-up interno que puede ser habilitado escribiendo un cero en el bit de control RBPU para que se active en todos los pines. Se desactiva automáticamente cuando el puerto es configurado como salida y deshabilitados en cada Power on Reset quedando como entradas digitales.

Cuatro de los pines de este puerto RB7:RB4 tiene una característica especial que les permite generar una interrupción del dispositivo cuando se detecta un cambio de nivel lógico en ellos, cuando se encuentran configurados como entradas. Esto se realiza comparando el valor actual en los pines con el viejo valor del latch registrado en la última lectura del puerto y cuando se descubre la existencia de un cambio, se levanta la bandera de interrupción RBIF ubicada en el registro INTCON0. Esta interrupción puede despertar al microcontrolador del modo SLEEP.

La condición de diferencia en la lectura del puerto mantendrá levantada la bandera. Mediante la lectura del registro PORTB se puede terminar esta condición permitiendo restablecer la condición de la bandera.

El bit RB3 puede ser configurado mediante el bit CCP2MX como el pin periférico alterno para el modulo de captura/comparación/pwm.

La tabla A4 muestra la función de cada pin del puerto B.

Nombre	Bit	Función
RB0/INT0	0	Entrada/Salida o Entrada de interrupción externa 0 ó pull-up interno
RB1/INT1	1	Entrada/Salida o Entrada de interrupción externa 1 ó pull-up interno
RB2/INT2	2	Entrada/Salida o Entrada de interrupción externa 2 ó pull-up interno
RB3/CCP2	3	Entrada/Salida o Entrada de captura 2/ Salida de comparación 2/ Salida de PWM cuando el bit de configuración CCP2MX es uno o pull-up interno.
RB4I	4	Entrada/Salida con interrupción por cambio de estado o pull-up interno.
RB5/PGM	5	Entrada/Salida con interrupción por cambio de estado o pull-up interno. Pin para habilitar ICSP de bajo voltaje.
RB6/PGC	6	Entrada/Salida con interrupción por cambio de estado o pull-up interno. Reloj de programación serial
RB7/PGD	7	Entrada/Salida con interrupción por cambio de estado o pull-up interno. Datos de programación serial.

Tabla A4 Funciones del Puerto B

Puerto C

Es un puerto bidireccional de 8 bits; cuyos registros son conocidos como TRISC, PORTC y LATC. Este puerto está multiplexado con varias funciones periféricas, por lo que se debe tener especial cuidado cuando se define la dirección de los pines de este puerto en el registro TRISC, dado que algunos periféricos sobrescribirán este registro para que los bits funcionen como entrada, mientras que otros periféricos lo harán para que funcionen como salida. Por lo que se debe consultar detalladamente las secciones donde se describen los periféricos a utilizar y lograr una correcta configuración.

La tabla A5 muestra la función de cada pin del puerto C.

Nombre	Bit	Función
RC0/T1OSO/T1CKI	0	Entrada/Salida o Entrada de interrupción externa 0 ó pull-up interno
RC1/T1OSI/CCP2	1	Entrada/Salida o Entrada de interrupción externa 1 ó pull-up interno
RC2/CCP1	2	Entrada/Salida o Entrada de interrupción externa 2 ó pull-up interno
RC3/SCK/SCL	3	Entrada/Salida o Entrada de captura 2/ Salida de comparación 2/ Salida de PWM cuando el bit de configuración CCP2MX es uno o pull-up interno.
RC4/SDI/SDA	4	Entrada/Salida con interrupción por cambio de estado o pull-up interno.
RC5/SDO	5	Entrada/Salida con interrupción por cambio de estado o pull-up interno. Pin para habilitar ICSP de bajo voltaje.
RC6/TX/CK	6	Entrada/Salida con interrupción por cambio de estado o pull-up interno. Reloj de programación serial
RC7/RX/DT	7	Entrada/Salida con interrupción por cambio de estado o pull-up interno. Datos de programación serial.

Tabla A5 Funciones del puerto C

Puerto D

Es un puerto de 8 bits, bidireccional y para configurarlo cuenta con los registros TRISD, PORTD y LATD. Además, a la entrada presenta un buffer tipo Schmitt Trtigger. Puede ser configurado como un puerto de microprocesador de 8 bits, es decir como un puerto paralelo esclavo mediante la habilitación del bit de control PSPMODE, de este modo los buffers de entrada son TTL.

La tabla A6 muestra la función de cada pin del puerto D.

Nombre	Bit	Función
RD0/PSP0	0	Bit de entrada/salida o Bit de puerto paralelo esclavo 0.
RD1/PSP1	1	Bit de entrada/salida o Bit de puerto paralelo esclavo 1.
RD2/PSP2	2	Bit de entrada/salida o Bit de puerto paralelo esclavo 2.
RD3/PSP3	3	Bit de entrada/salida o Bit de puerto paralelo esclavo 3.
RD4/PSP4	4	Bit de entrada/salida o Bit de puerto paralelo esclavo 4.
RD5/PSP5	5	Bit de entrada/salida o Bit de puerto paralelo esclavo 5.
RD6/PSP6	6	Bit de entrada/salida o Bit de puerto paralelo esclavo 6.
RD7/PSP7	7	Bit de entrada/salida o Bit de puerto paralelo esclavo 7.

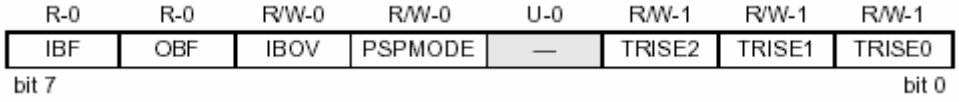
Tabla A6 Funciones del Puerto D

Puerto E

Este puerto cuenta con sólo tres bits, bidireccionales RE0/RD/AN5, RE1/WR/AN6 y RE2/CS/AN7). Su configuración se realza a través de los registros de control TRISE, PORTE y LATE. Cada uno de los pines pueden ser configurados individualmente como entradas o como salidas teniendo buffers de entrada tipo Schmitt Trigger. Los pines de este puerto están multiplexados con las entradas analógicas del convertidor A/D, por lo que cuando funcionan como tales son leídos como ceros.

Ya que sólo es necesario configurar tres bits, el registro TRISE es empleado para configurar otros procesos.

Registro TRISE



Bit 7 **IBF**: Bit de estado Completo del buffer de entrada
1 = Se ha recibido una palabra y esta en espera a ser leída por el CPU.
0 = No se ha recibido ninguna palabra

Bit 6 **OBF**: Bit de estado completa del buffer de salida
1 = El buffer de salida aún contiene la palabra escrita anteriormente
0 = El buffer de salida ha sido leído.

Bit 5 **IBOV**: Bit de detección de sobreflujo en el buffer de entrada (en modo microprocesador)
1 = Se ha realizado una escritura sin leer la palabra de entrada previa (debe borrarse por software)
0 = No ha ocurrido sobrefuljo

Bit 4 **PSPMODE**: Bit de selección para el modo puerto paralelo esclavo
1 = Modo puerto paralelo esclavo
0 = Modo de entrada/salida de propósito general

Bit 3 No implementado, es leído como cero.

Bits **RE2: RE0:** bits de control de dirección del registro TRISE2:TRISE0.

La tabla A7 muestra la función de cada pin del puerto E.

Nombre	Bit	Función
RE0/$\overline{RD}$/AN5	0	Bit de entrada/salida o Control de lectura de entrada en modo de puerto paralelo esclavo o entrada analógica.
RE1/$\overline{WR}$/AN6	1	Bit de entrada/salida o Control de escritura de entrada en modo de puerto paralelo esclavo o entrada analógica.
RE2/$\overline{CS}$ /AN7	2	Bit de entrada/salida o Control de selección de chip de entrada en modo de puerto paralelo esclavo o entrada analógica..

Tabla A7 Funciones del Puerto E

Temporizador

El microcontrolador cuenta con 4 temporizadores los cuales tienen propiedades específicas. Éstos pueden ser seleccionados de forma independientemente según su aplicación. De acuerdo a sus características principales son empleados como contadores de eventos o como fuentes para generar interrupciones.

Módulo del Temporizador 0

Este módulo, puede operar como temporizador o como contador, activando o desactivando el bit cinco T0CS del registro de control T0CON. En el modo de temporizador, éste se incrementa en cada ciclo de instrucción. Tiene un registro llamado TMR0L, donde se escribe el valor a partir del cuál comenzará el conteo; una vez escrito, el incremento se inhibe durante los dos siguientes ciclos de instrucción. Cuando se emplea como contador, se puede configurar en el bit cuatro T0SE para que los incrementos se realicen por la detección de flancos de subida o bajada en el pin cuatro del puerto A.

Se puede modificar la velocidad del conteo, mediante un prescalador el cuál se habilita borrando el bit tres PSA del registro T0CON. Es configurable para trabajar con valores predeterminados que se asignan mediante tres bits T0PS2:T0PS0 del mismo registro. Si se escribe en el registro TMR0L cuando el prescalador esta activado, se borrará su cuenta sin afectar la relación de prescalamiento.

La cuenta para este temporizador puede realizarse en ocho o dieciséis bits que se habilita con el bit de control T08BIT. Su fuente de interrupción se activa cuando el registro TMR0 se desborda de FFh a 00h en el modo de ocho bits ó de FFFFh a 0000h en el de dieciséis, levantando la bandera de interrupción TMR0IF. Cuando el microcontrolador se encuentra en el modo SLEEP, éste no puede despertarse por medio de la interrupción del temporizador, debido a que se encuentra apagado durante este estado.

Para las operaciones de lectura y escritura en modo de 16 bits se utiliza el registro TMR0H, que no es directamente el byte superior del temporizador. Este se ocupa como un búfer durante la lectura del TMR0L, actualizando el valor almacenado por el byte alto.

De esta forma, se pueden leer los 16 bits del temporizador sin la necesidad de verificar que la lectura de los bytes inferior y superior sea válida. Cuando se desea escribir la parte alta del temporizador, se debe realizar primero en el registro TMR0H y una vez escrito el valor del TMR0L se ejecuta la actualización de los 16 bits. De manera gráfica, la operación en 8 bits se aprecia en la figura A11 y para 16 bits en la figura A12.

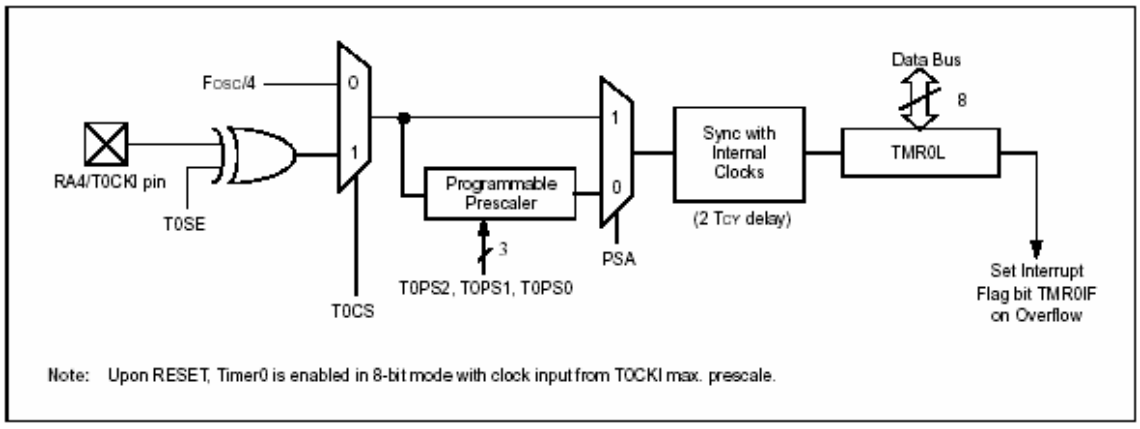


Figura A11 Diagrama de bloques del Temporizador 0 para funcionamiento en modo de 8-bits.

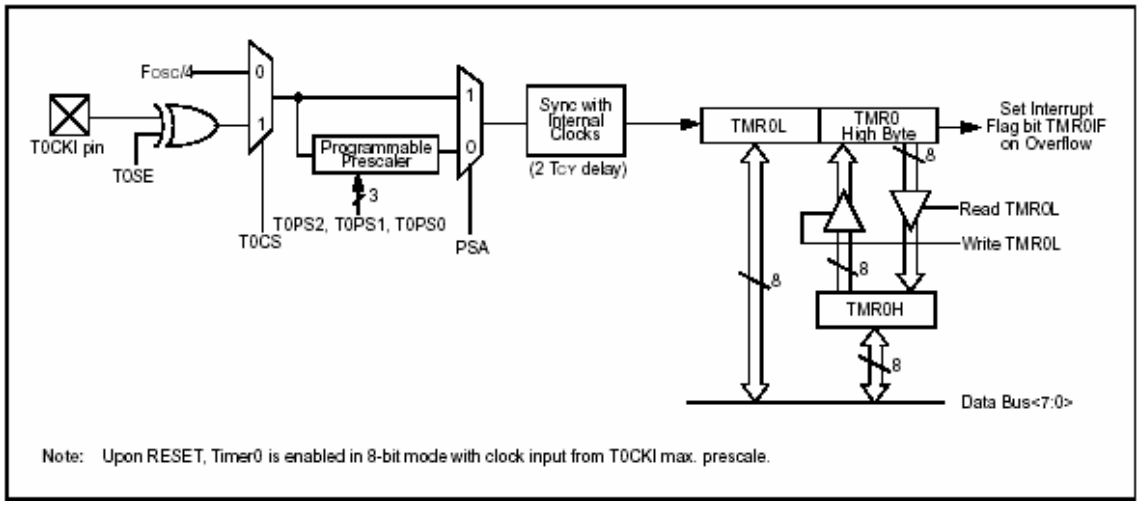


Figura A12 Diagrama de bloques del Temporizador 0 para funcionamiento en modo de 16-bits.

- **Registro T0CON:** Registro de control para el temporizador 0.

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

Bit 7 **TMR0ON:** Activa/desactiva el temporizador 0.

Bit 6 **T08BIT:** Configura el temporizador para su operación. Escribiendo un 1 en modo de 8 bits y con un 0 en 16 bits.

Bit 5 **T0CS:** Selecciona la fuente de reloj para su uso en modo de temporizador o de contador. Escribiendo un 1 se utiliza un reloj externo y con un 0 el ciclo de instrucción interno.
Bit 4 **TOSE:** Selecciona el flanco con el cuál incrementa el conteo. Escribiendo un 1 incrementa en el flanco de bajada y con un 0 en el de subida.

Bit 3 **PSA:** Escribiendo un 1 deshabilita el prescalador y con un 0 lo habilita.

Bit 2-0 **T0PS2:T0PS0:** Selecciona el valor del prescalador.

111=1:256
110=1:128
101=1:64
100=1:32
011=1:16
010=1:8
001=1:4
000=1:2

Módulo del Temporizador 1

Opera en modo de 16 bits utilizando dos registros de 8 bits TMR1H y TMR1L, que se pueden leer y escribir en 8 o 16 bits para modificar el valor de la cuenta en cualquier momento del programa. Puede funcionar en modo de temporizador, de contador síncrono y de contador asíncrono. Cualquiera de éstas funciones se asignan mediante el bit TMR1CS, cuando se escribe un 0 funciona como temporizador e incrementa en cada ciclo de instrucción y con un 1 opera como contador e incrementa en cada flanco de subida. El diagrama de bloques se muestra en la figura A13.

Su fuente de interrupción se activa cuando sus registros se desbordan de FFFFh a 0000h, levantando la bandera de interrupción TMR1IF. Cuando el microcontrolador se encuentra en el modo SLEEP, el temporizador puede funcionar con un cristal externo en modo de contador debido a que tiene construido un circuito oscilador entre los pines T1OSI y T1OSO. Éste oscilador trabaja hasta una frecuencia de 200 kHz.
Se puede modificar la velocidad del conteo, mediante un prescalador el cuál es configurable para trabajar con valores predeterminados que se asignan mediante dos bits

T1CKPS1:T1CKPS0 del registro T1CON. Si se escribe en los registros cuando el prescalador esta activado, se borrará su cuenta sin afectar la relación de prescalamiento.

Si el módulo CCP1 está configurado para operar en modo de comparación y para generar una señal interna de disparo, ésta reiniciará el Temporizador 1 y generará una conversión A/D siempre que el módulo haya sido habilitado. Para aprovechar esta característica, debe estar operando en modo de temporizador o de contador síncrono.

En el caso en que coincida una escritura en el temporizador y un evento del CCP1, la escritura tendrá mayor prioridad.

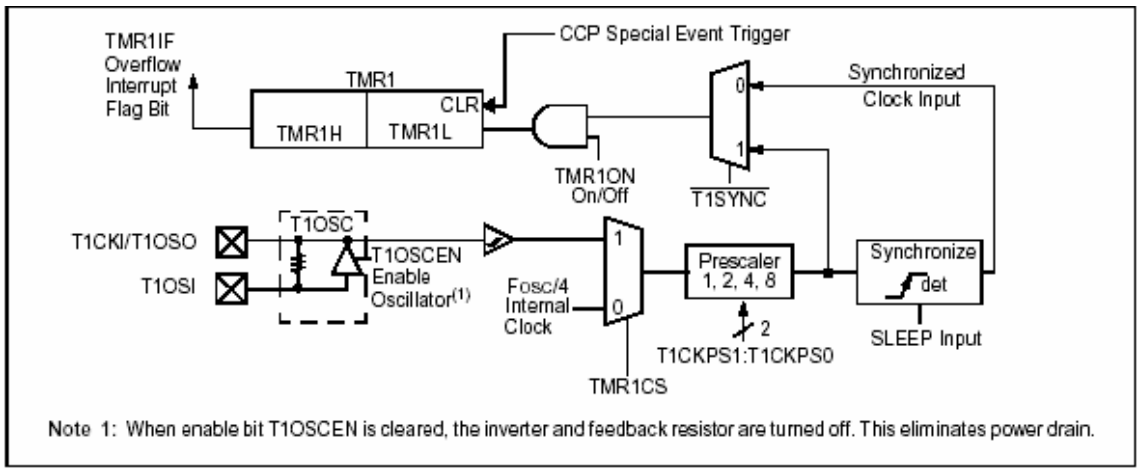


Figura A13 Diagrama de bloques del Temporizador 1

La operación del temporizador 1 se muestra en la figura A14.

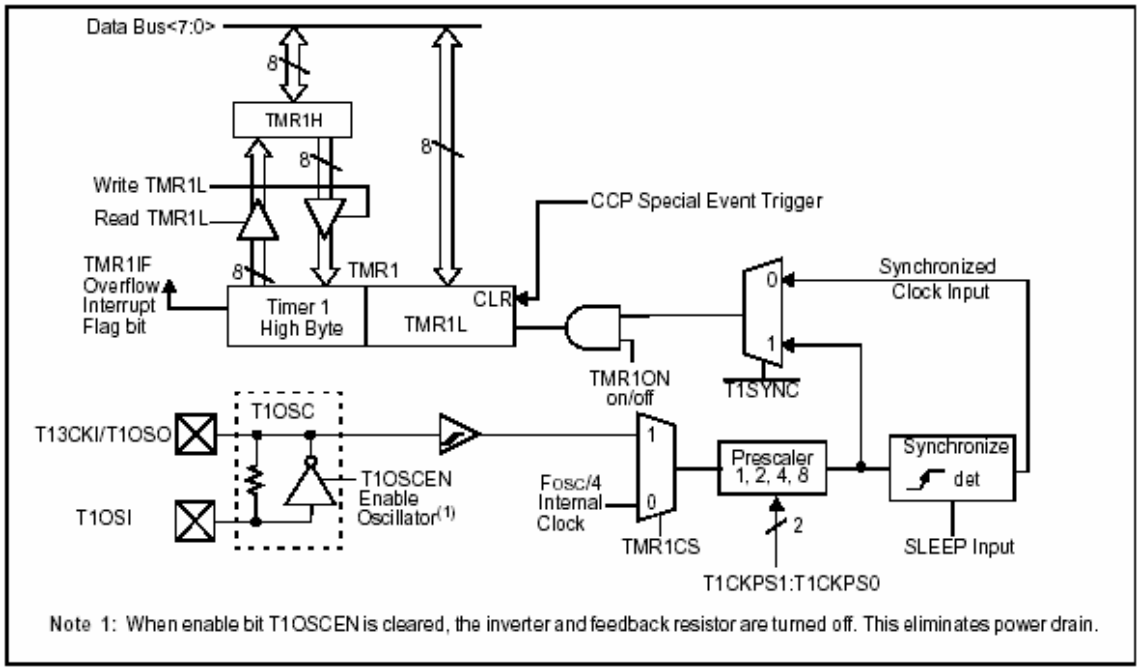


Figura A14 Diagrama de bloques del Temporizador 1 para funcionamiento en modo de 16-bits.

- **Registro T1CON:** Registro de control para el temporizador 1.

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

Bit 7 **RD16:** Escribiendo un 1 se habilita el modo de lectura y escritura en el temporizador en una operación de 16 bits y con un 0 en dos de 8 bits.

Bit 6 no implementado.

Bit 5-4 **T1CKPS1:T1CKPS0:** Selecciona el valor del prescalador.
11=1:8
10=1:4
01=1:2
00=1:1

Bit 3 **T1OSCEN:** Activa/desactiva el circuito oscilador del temporizador 1.

Bit2 $\overline{\text{T1SYNC}}$ **:** Selecciona la sincronización con un reloj externo cuando se activa el bit TMR1CS. Escribiendo un 1 la comunicación se hace sin sincronía y con un 0 con sincronía.

Bit 1 **TMR1CS:** Selecciona el reloj con el que opera el temporizador, escribiendo un 1 se selecciona un reloj externo y con un 0 el reloj interno.

Bit 0 **TMR1ON:** Activa/desactiva el temporizador 1.

Módulo del Temporizador 2

Este módulo sólo funciona como temporizador a diferencia de los otros temporizadores y opera en modo de 8 bits. Utiliza dos registros TMR2 y PR2 que se pueden leer y escribir en cualquier momento del programa. Puede usarse como el periodo para generar las señales de PWM cuando se ha configurado este modo en el módulo CCP. El diagrama de bloques se muestra en la figura A15.

Su fuente de interrupción se activa cuando el valor del registro TMR2 alcanza el valor de PR2 levantando la bandera de interrupción TMR2IF y reiniciando el valor de TMR2 a 00h en el siguiente ciclo. Ésta característica se utiliza para generar los PWM's. El registro PR2 se inicializa a FFh una vez que se ha reiniciado.

Se puede modificar la velocidad del conteo mediante un prescalador el cuál es configurable para trabajar con valores predeterminados que se asignan mediante dos bits T2CKPS1:T2CKPS0 del registro T2CON. También cuenta con un postescalador de cuatro

bits a la salida del TMR2 que sirve para alimentar el módulo del puerto síncrono serial. Se borrará la cuenta del prescalador o del postescalador cuando se escribe en el registro TMR2, en el registro de control T2CON o bajo cualquier evento que reinicie el microcontrolador sin afectar la relación de prescalamiento.

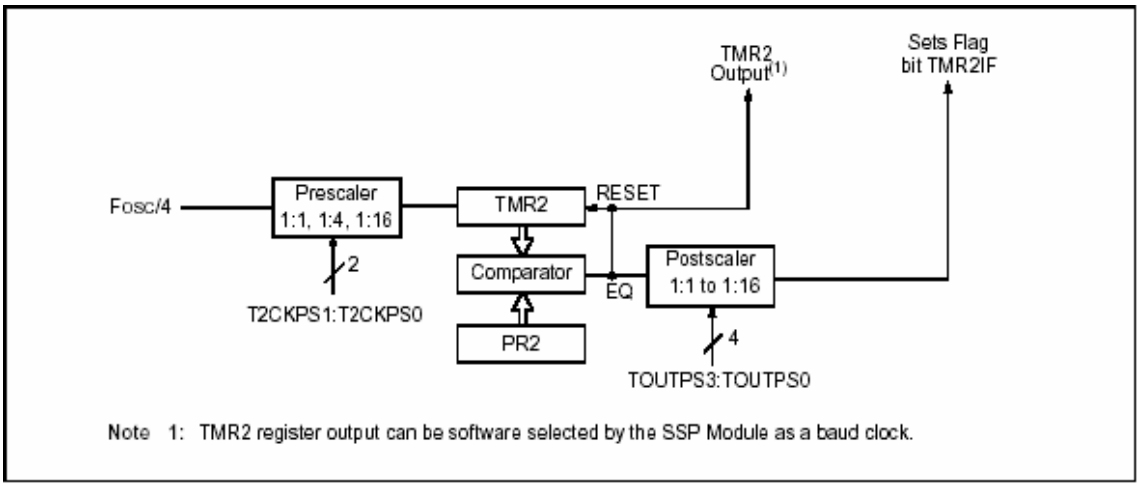


Figura A15 Diagrama de bloques del Temporizador 2.

- **Registro T2CON:** Registro de control para el temporizador 2

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

Bit 7 no implementado.

Bit 6-3 **TOUTPS3:TOUTPS0:** Selecciona el valor para el postescalador.
0000=1:1
0001=1:2
*
*
*
1111=1:16

Bit 2 **TMR2ON:** Activa/desactiva el temporizador 2.

Bit 1-0 **T2CKPS1:T2CKPS0:** Selecciona el valor del prescalador.
00=1:1
01=1:4
1X=1:16

Módulo del Temporizador 3

Este módulo tiene las mismas características que el temporizador 1 con la excepción de que se puede seleccionar que la fuente de reloj para la operación del módulo CCP sea este temporizador o el temporizador 1. El diagrama a bloques se muestra en la figura A16..

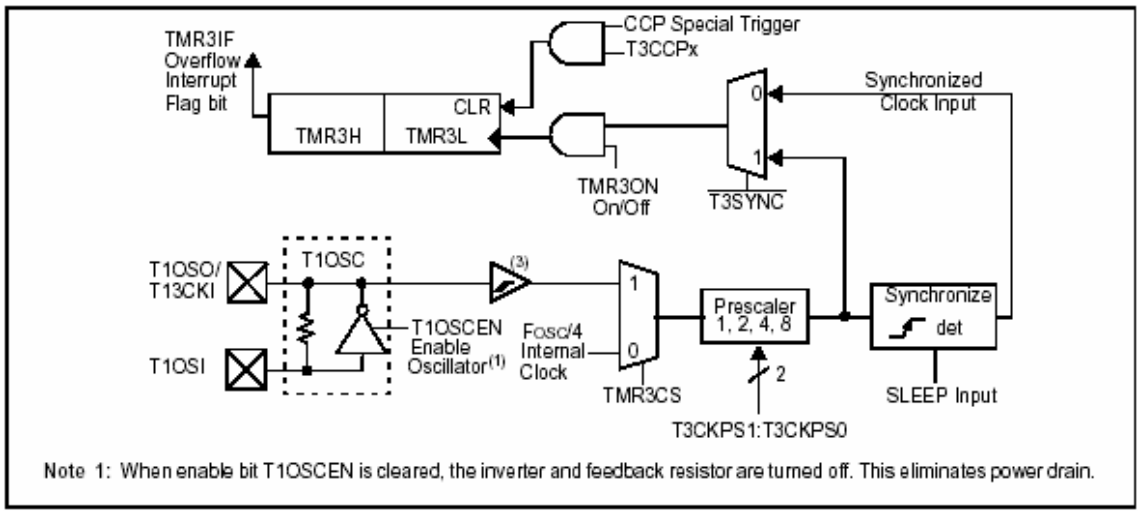


Figura A16 Diagrama de bloques del Temporizador 3

La figura A17 muestra la operación del temporizador 3.

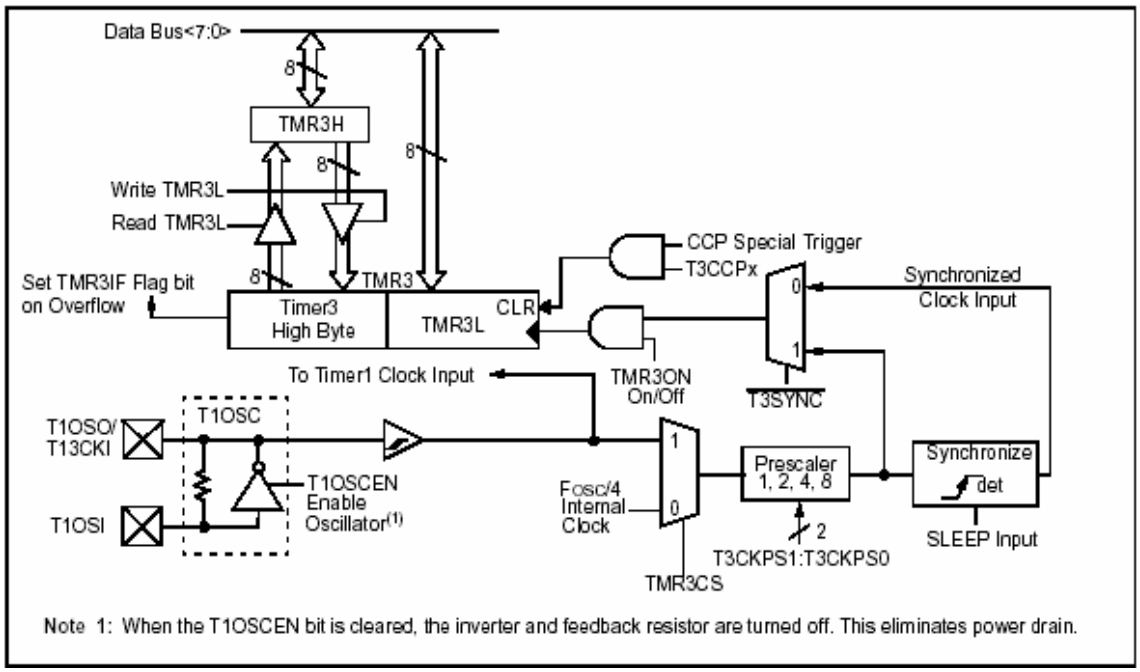
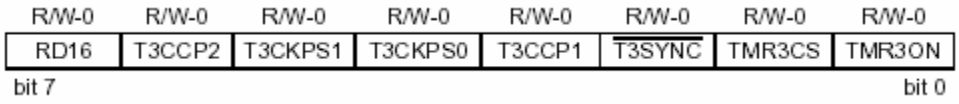


Figura A 17 Diagrama de bloques del Temporizador 3 para funcionamiento en modo de 16-bits.

- **Registro T3CON:** Registro de control para el temporizador 3.



Bit 7 **RD16:** Escribiendo un 1 se habilita el modo de lectura y escritura en el temporizador en una operación de 16 bits y con un 0 en dos de 8 bits.

Bit 6 y 3 **T3CCP2:T3CCP1:** Habilita la fuente de reloj para los módulos CCPx
1X = Temporizador 3 es la fuente de reloj para los módulos CCP
01 = Temporizador 3 es la fuente de reloj para el módulo CCP2
Temporizador 1 es la fuente de reloj para el módulo CCP1
00 = Temporizador 1 es la fuente de reloj para los módulos CCP

Bit 5-4 **T3CKPS1:T3CKPS0:** Selecciona el valor del prescalador.
11=1:8
10=1:4
01=1:2
00=1:1

Bit2 **$\overline{\text{T3SYNC}}$:** Selecciona la sincronización con un reloj externo cuando se activa el bit TMR1CS. Escribiendo un 1 la comunicación se hace sin sincronía y con un 0 con sincronía.

Bit 1 **TMR3CS:** Selecciona el reloj con el que opera el temporizador, escribiendo un 1 se selecciona un reloj externo y con un 0 el reloj interno.

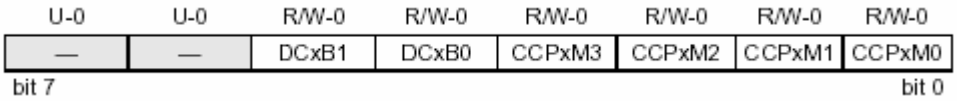
Bit 0 **TMR3ON:** Activa/desactiva el temporizador 1.

Módulos de Captura/Comparación/PWM (CCP)

Cada módulo CCP contiene un registro de 16 bits, que puede operar como un registro de captura, registro de comparación ó como un registro de 10 bits para el ciclo de trabajo de una señal de PWM. Los temporizadores 1 y 3 pueden seleccionarse para los modos de comparación y captura; y el temporizador 2 es la base de tiempo para el PWM. La operación de los módulos CCP1 y CCP2 es idéntica.

Los registros CCP1CON y CCP2CON controlan el modo de operación de los módulos CCP.

- **Registros de control CCPxCON**



Bit 7-6 no implementado

Bit 5-4 **DCxB1:DCxB0:** Representan los dos bits menos significativos del ciclo de trabajo del PWM.

Bit 3-0 **CCPxM3:CCPxM0:** Selección del modo de operación del CCPx

0000 = Deshabilita los modos de Captura/Comparación/PWM.
0001 = Reservado
0010 = Modo de captura. Levanta la bandera de interrupción CCPxIF
0011 = Reservado
0100 = Modo de captura en cada flanco de bajada.
0101 = Modo de captura en cada flanco de subida.
0110 = Modo de captura en cada 4º flanco de subida.
0111 = Modo de captura en cada 16º flanco de subida.

1000 = Modo de comparación. Inicializa el pin del CCP en bajo y cuando se iguala el valor a comparar cambia el estado del pin a alto. Levanta la bandera de interrupción.

1001 = Modo de comparación. Inicializa el pin del CCP en alto y cuando se iguala el valor a comparar cambia el estado del pin a bajo. Levanta la bandera de interrupción.

1010 = Modo de comparación. Genera una interrupción por software y el pin del CCP no se afecta.

1011 = Modo de comparación. Genera un disparo interno para habilitar la operación del convertidor A/D.

11xx = Modo de PWM

En la tabla A8 se describe la interacción entre los módulos de captura y comparación.

Modo del CCPx	Modo del CCPy	Interacción
Captura	Captura	Temporizador 1 o 3 son la base del tiempo y pueden ser diferentes para cada CCP.
Captura	Comparación	La comparación puede ser configurada para generar un disparo interno, el cuál borra el valor en la base de tiempo, dependiendo del temporizador seleccionado.
Comparación	Comparación	Las comparaciones pueden ser configuradas para generar los disparos internos, los cuales borraran el valor en la base de tiempo, dependiendo del temporizador seleccionado.
PWM	PWM	Los PWMs tendrán la misma frecuencia y la misma velocidad de actualización.
PWM	Captura	Ninguna.
PWM	Comparación	Ninguna.

Tabla A8 Interacción entre los dos módulos CCP

Modo de PWM

En este modo el pin del CCP1 que se encuentra multiplexado con el pin dos del puerto C, produce una salida de PWM con una resolución de 10-bits. Se tienen que borrar los bits correspondientes en el registro TRISC para que operen como salidas. El postescalador del temporizador 2 no se usa para determinar la frecuencia del PWM.

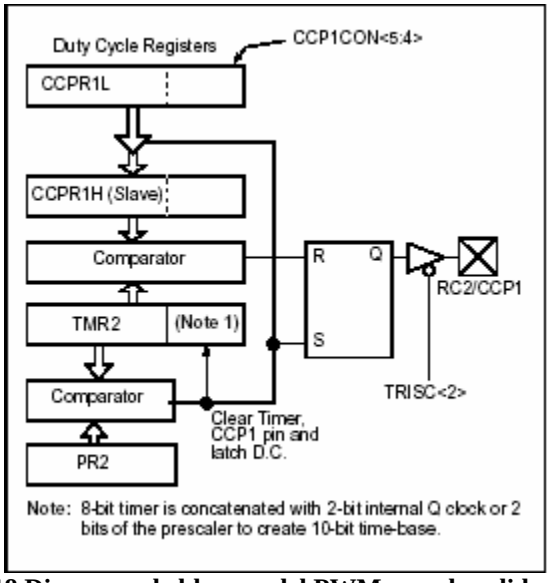


Figura A18 Diagrama de bloques del PWM para la salida por CCP1

La salida del PWM tiene una base de tiempo (periodo) y un tiempo en el cual la salida se encuentra en alto (ciclo de trabajo). La frecuencia del PWM es el inverso de su periodo, en la figura A19 se muestran gráficamente el periodo y el ciclo de trabajo.

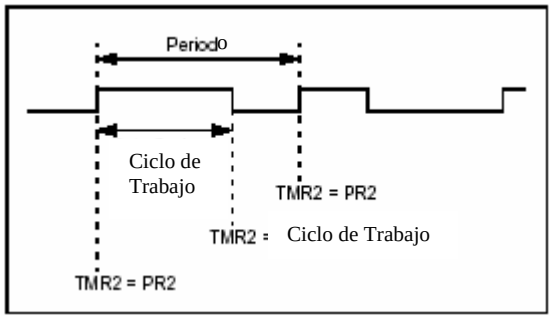


Figura A19 Operación de la salida por PWM

El periodo del PWM se especifica escribiéndolo en el registro PR2. Éste puede ser calculado usando la ecuación (A.1).

$$PWM_{periodo} = [PR2 + 1] * 4 * Tosc * (TMR2\,valor\ del\ prescalador) \quad \text{(A.1)}$$

Cuando el temporizador 2 alcanza el valor del registro PR2, ocurren tres eventos durante el siguiente ciclo.

- Se borra el valor de la cuenta del temporizador 2.
- Se activa el pin del CCP1, a excepción de que el ciclo de trabajo sea igual a 0%.
- El ciclo de trabajo del PWM es almacenado de forma temporal por el registro CCPR1L y transferido al CCPR1H.

El ciclo de trabajo del PWM se especifica escribiendo el valor en el registro CCPR1L y en los bits 4:5 del CCP1CON, para generar un PWM de 10 bits de resolución. Se puede escribir en los registros en cualquier momento, pero el ciclo de trabajo no es transferido hasta que se iguala el valor en los registros PR2 y TMR2. Para calcular el ciclo de trabajo en el tiempo se usa la ecuación (A.2).:

$$PWM_{ciclo\,de\,t} = [CCPR1L : CCP1CON < 5:4 >] * Tosc * (TMR2\,valor\ del\ prescalador) \quad \text{(A.2)}$$

El registro CCPR1H es de solo lectura durante el modo de PWM. Es usado para hacer un doble búfer del ciclo de trabajo y así evitar fallas en la operación. Cuando el valor de este registro es igualado por TMR2, se borra el pin CCP1. La máxima resolución en bits para una frecuencia de PWM dada se calcula mediante la ecuación (A.3).

$$PWM_{máxima\,resolución} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} bits \quad \text{(A.3)}$$

Debido a la similitud que presentan los módulos CCP1 y CCP2 sólo se detalló la configuración para el primero siendo análoga la configuración para el segundo.

Convertidor Analógico-Digital

El PIC18F452 cuenta con un convertidor analógico-digital de 10 bits configurable, para funcionar con un oscilador interno o con la frecuencia de operación del microprocesador. Tiene ocho canales de conversión distribuidos en los puertos A y E, que pueden ser activados o desactivados de manera independiente.

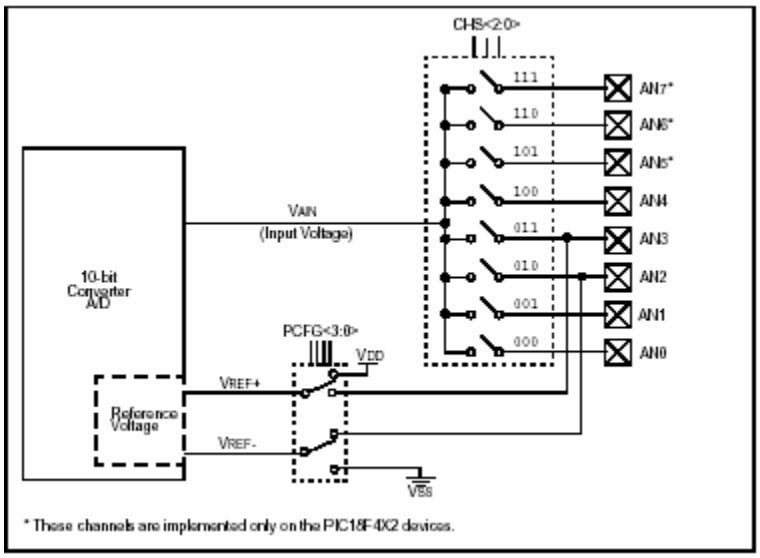


Figura A20 Diagrama de bloques del convertidor A/D

El voltaje de referencia se puede tomar del voltaje de la fuente de alimentación o de los pines dos y tres del puerto A.

Adquisición de la señal analógica

El convertidor cuenta con un circuito conectado en la entrada analógica, que se basa en un capacitor de retención de carga el cual es desconectado de la entrada antes de que la conversión se inicie.

Para que el convertidor A/D adquiera la señal de entrada de manera precisa, se debe esperar un tiempo T_{ADQ}, para que el capacitor de retención (C_{HOLD}) se cargue completamente con el nivel de voltaje del canal de entrada, de lo contrario se presentaran errores en el resultado de la conversión.

El modelo analógico de entrada se muestra en la siguiente figura:

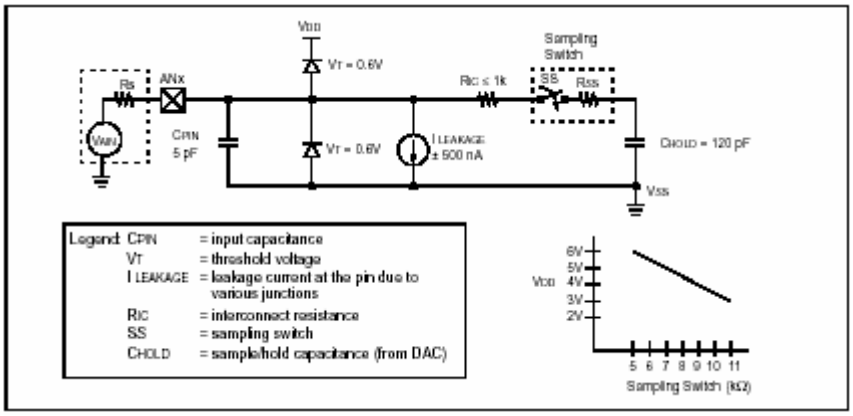


Figura A21 Modelo analógico de entrada

La impedancia de la fuente (Rs) y la impedancia del interruptor de muestreo (Rss) afectan directamente el tiempo requerido para la carga del capacitor. Se recomienda que la impedancia de la fuente no sea mayor de 2.5 kΩ.

Para calcular el tiempo mínimo de adquisición *(T_{ADQ})* se debe considerar la ecuación (A.4):

$$T_{ADQ} = T_{AMP} + T_C + T_{COEF} \tag{A.4}$$

donde:

T_{AMP} = tiempo de acople del amplificador
T_C = tiempo de carga del capacitor
T_{COEF} = coeficiente de temperatura

Y T_C se muestra en (A.5):

$$T_C = -(120\ pF)(1k\Omega + R_{SS} + R_S)\ln(1/2048) \tag{A.5}$$

Tiempo de conversión

El tiempo de conversión por bit se define como $T_{AD.}$ Se requiere de 12 T_{AD} para una conversión de 10 bits y para asegurar una correcta conversión se necesita un T_{AD} mínimo de 1.6 μs. Es importante mencionar que se debe esperar 2 T_{AD} antes de realizar la siguiente adquisición. La descripción gráfica del tiempo de espera, se muestra en la figura A22.

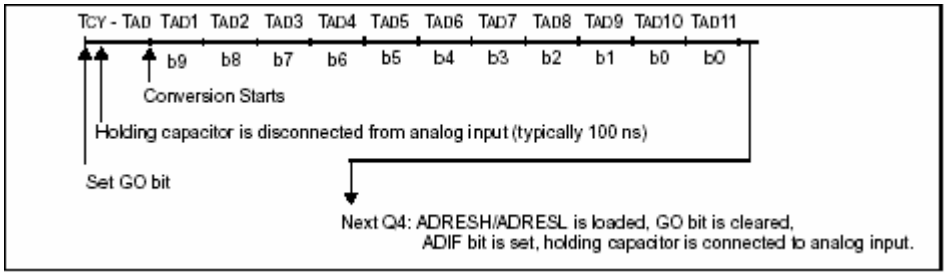


Figura A22 Tiempo de conversión por bit

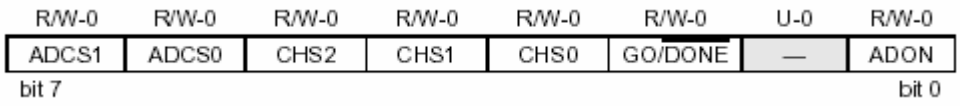
A continuación, en la tabla 9 se muestra la configuración de bits para seleccionar el T_{AD} mínimo dependiendo de la frecuencia a la que este operando el microprocesador.

Fuente de reloj para el convertidor AD (TAD)		**Frecuencia máxima del dispositivo**
Operación	*ADCS2:ADCS0*	*PIC18F452*
2 Tosc	000	1.25 MHz
4 Tosc	100	2.50 MHz
8 Tosc	001	5.00 MHz
16 Tosc	101	10.00 MHz
32 Tosc	010	20.00 MHz
64 Tosc	110	40.00 MHz
RC	011	----------

Tabla A9 Selección del TAD según la frecuencia del microcontrolador

Para configurar el convertidor se utilizan dos registros de control ADCON0 y ADCON1.

Registro ADCON0



Bit 7-6 **ADCS1:ADCS0:** Selección de la fuente de reloj para realizar la conversión tomando en cuenta las siguientes opciones.

ADCON0 <ADCS1:ADCS0>	**Reloj de conversión**
00	Fosc/2
01	Fosc/8
10	Fosc/32
11	FRC (reloj derivado del oscilador interno del convertidor)
00	Fosc/4
01	Fosc/16
10	Fosc/64
11	FRC (reloj derivado del oscilador interno del convertidor)

Tabla A10 Configuración de la fuente de reloj

Bit 5-3 **CHS2:CHS0:** Bits de selección para canales analógicos.

000 Canal 0 (AN0)
001 Canal 1 (AN1)
010 Canal 2 (AN2)
011 Canal 3 (AN3)
100 Canal 4 (AN4)
101 Canal 5 (AN5)
110 Canal 6 (AN6)
111 Canal 7 (AN7)

Tabla A11 Configuración de bit para selección de canal del convertidor A/D

Bit 2 **GO/$\overline{\text{DONE}}$**: Bit de estado de conversión. Al activar este bit se comienza la conversión y una vez terminada, éste se borra por hardware y la bandera de interrupción del convertidor se habilita. El resultado se almacena en dos localidades de memoria: ADRESH y ADRESL.

Bit 0 **ADON:** Bit que enciende o apaga el convertidor.

Registro ADCON1

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

Bit 7 **ADFM:** Bit de formato de resultado. Cuando se activa, el resultado es justificado a la derecha y los seis bits más significativos del registro ADRESH son leídos como ‘0’. Cuando se desactiva, el resultado es justificado a la izquierda y los seis bits menos significativos del registro ADRESL son leídos como ‘0’. Esto se representa gráficamente en la figura A22.

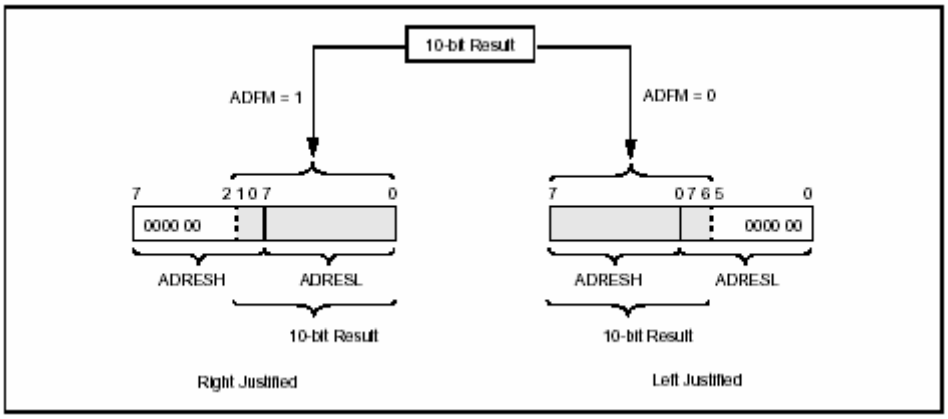


Figura A23 Operación del formato de resultado

Bit 6 **ADCS2:** Bit de selección de fuente de reloj. Este bit debe ser configurado en conjunto con los bits ADCS1:ADCSO que se encuentran en el registro ADCON0.

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Reloj de conversión
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (reloj derivado del oscilador interno del convertidor)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (reloj derivado del oscilador interno del convertidor)

Tabla A12 Configuración de la fuente de reloj

Bit 3-0 **PCFG3:PCFG0:** Bits de control para configuración de canales.

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	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	AN3	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	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

A = Analog input D = Digital I/O
C/R = # of analog input channels / # of A/D voltage references

Figura A24 Descripción de bits para configuración del convertidor A/D

Conjunto de Instrucciones

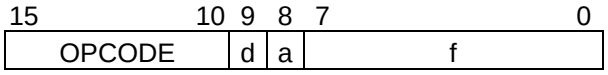
El conjunto de instrucciones para el PIC 18F452 presenta grandes mejorías con respecto a sus antecesores. La mayoría de las instrucciones son palabras simples en memoria de programa de 16 bits, pero existen tres instrucciones que requieren de dos localidades en dicha memoria.

Cada instrucción es una palabra de 16 bits dividida dentro de un código de operación (OPCODE), en el cual se especifica el tipo de instrucción del que se trata, así como uno o más operandos que especificarán la operación que realizará la instrucción.

El conjunto de instrucciones es altamente ortogonal, es decir, y se encuentra dividido en cuatro categorías básicas:

- Operaciones orientadas a bytes
- Operaciones orientadas a bits
- Operaciones a literales
- Operaciones de control

La mayoría de las instrucciones orientadas a bytes cuentan con tres operandos:

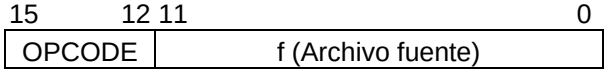


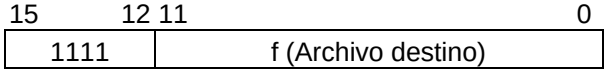
1.- El registro de archivo, especificado por la letra “f”. En este operando se declara el registro que va a ser utilizado por la instrucción.

2.- El destino del resultado, especificado por la letra “d”. En este operando se declara donde va a ser colocado el resultado de la operación. Cuando d=0 el resultado se almacenará en el registro W, si d=1 el resultado será almacenado en el registro de archivo especificado en la instrucción.

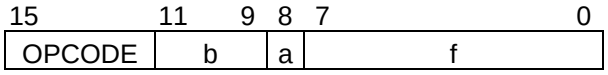
3.- La memoria accesada, especificada por la letra “a”.

El caso especial trata las operaciones de movimiento entre bytes (instrucción de dos palabras), lo cual se realiza de la siguiente forma:





Por lo que en este caso f es un registro de archivo de 12 bits.
Todas las instrucciones orientadas a bits cuentan con tres operandos:

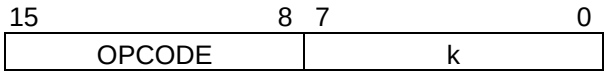


1.- El registro de archivo, especificado por la letra “f”. En este operando se declara el número de archivo donde se encuentra el bit del siguiente inciso.

2.- El bit en el registro de archivo, especificado por la letra “b”. Este operando selecciona el número de bit que va a ser afectado por la operación.

3.- La memoria accesada, especificada por la letra “a”.

Las instrucciones de operación a literales pueden usar alguno de los siguientes operandos:



1.- El valor de la literal a ser almacenada dentro de un registro de archivo, especificado por la letra “k”.

2.- El registro FSR deseado para almacenar el valor de una literal, especificado por la letra “f”.

3.- Sin operando, sólo se escribe la instrucción sin la necesidad de agregar operandos.

Las instrucciones de control pueden usar alguno de los siguientes operandos:

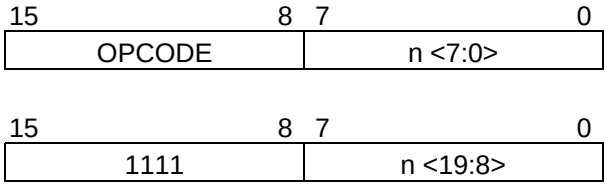
1.- Una dirección de memoria de programa, especificada por la letra “n”

2.- El modo para las instrucciones de lectura y escritura de tabla, especificada por la letra “m”.

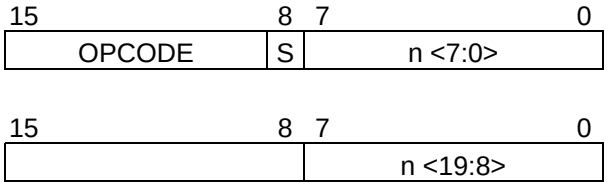
3.- Sin operando, sólo se escribe la instrucción sin la necesidad de agregar operandos.

La información utilizada por estas instrucciones y sus operandos varía según el tipo de instrucción de la que se trate:

Instrucción GOTO (dos palabras)

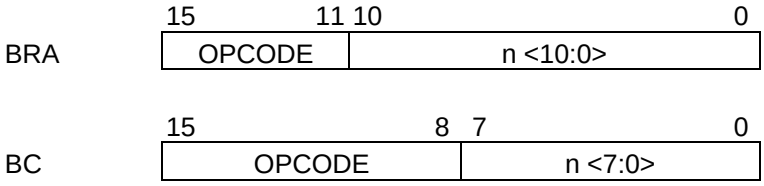


Instrucción CALL (dos palabras)



donde S es un bit rápido, cuando su valor es uno, se respalda el valor almacenado en el registro W, el registro de estado y el BSR.

Instrucciones de salto (instrucción de una palabra)



Existen tres instrucciones que son de doble palabra, estas requieren de toda la información almacenada en los 32 bits correspondientes. Para asegurar que no se presenten errores, los bits más significativos de la segunda palabra son 1's, de esta forma si la segunda palabra es ejecutada como una instrucción, el microcontrolador no efectuará ninguna operación (NOP).

Todas las instrucciones de una sola palabra son ejecutadas en un ciclo de instrucción, a menos que se realice una prueba condicional y sea verdadera o que el contador de programa sea modificado como resultado de la instrucción. En estos casos, la ejecución se realiza en dos ciclos de instrucción y la instrucción adicional es ejecutada como un NOP. Por otro lado, las instrucciones de dos palabras son ejecutadas en dos ciclos de instrucción.

Un ciclo de instrucción consiste de cuatro periodos de oscilación, por lo que el tiempo en el que se ejecutará una instrucción depende de la frecuencia del cristal seleccionado para el microcontrolador.

APÉNDICE B Sensor de inclinación

Es un transductor que produce una señal eléctrica de salida proporcional a su posición angular. En la figura B1 se muestra el empaque plástico del sensor de inclinación SCA100T-D01 y la dirección de sensado para cada eje.

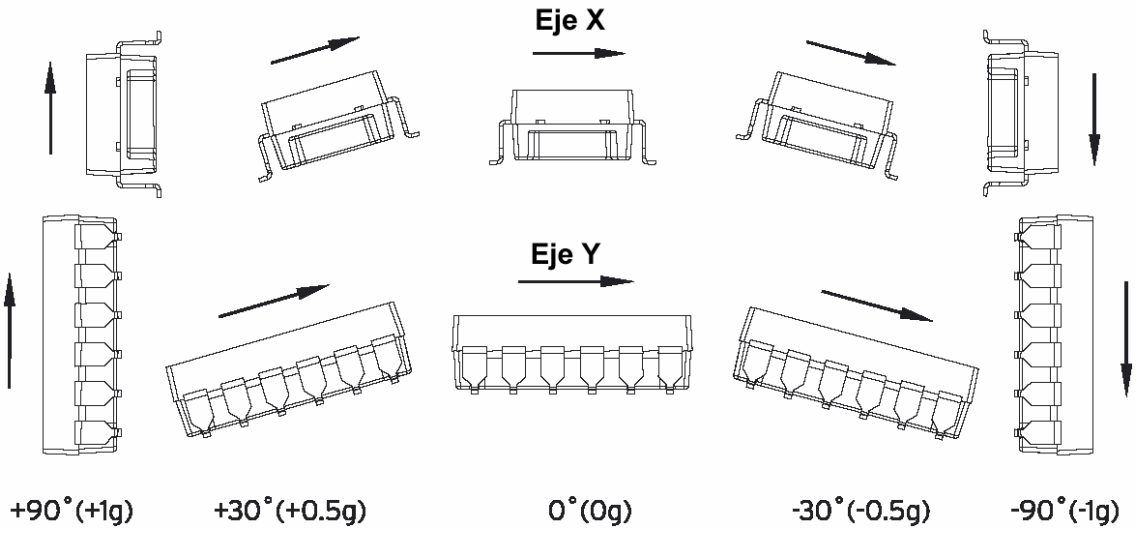


Figura B1 Sensor de Inclinación SCA100T-D01

En la industria, un sensor de inclinación se refiere estrictamente al transductor. Cuando a este se le agregan etapas electrónicas de acondicionamiento de señal o transmisión digital de datos, el sistema se denomina inclinómetro.

Los sensores de inclinación pueden ser agrupados dentro de tres categorías:

- Fuerza balanceada
- Estado sólido (MEMS)
- Llenados con líquido

Los primeros proveen un desempeño superior, a un costo elevado. Los dispositivos basados en MEMS se caracterizan por contar con una etapa de acondicionamiento de señal que facilita su implementación. Los últimos son los más utilizados en la industria gracias a su relación costo-beneficio en aplicaciones donde la velocidad de repuesta no sea un factor privativo.

El principio de operación generalizado se basa en que estos dispositivos generan un horizonte artificial y miden la inclinación angular con respecto de este. Son utilizados principalmente en cámaras, en controles para máquinas voladoras y sistemas de seguridad para automóviles entre otros.

Las especificaciones de mayor importancia para considerar durante la selección de sensores de inclinación e inclinómetros son el ángulo de inclinación, el número de ejes, la precisión y el ancho de banda que es el rango de frecuencia sobre el cual el dispositivo cumple con las especificaciones de precisión, ya que la precisión se ve reducida.

Las salidas eléctricas comúnmente utilizadas incluyen voltaje y corriente analógicos, salida digital, por puerto serial, puerto paralelo y por frecuencia.

Sensor capacitivo SCA100T-D01

Este sensor está basado en un sistema micro electrónico-mecánico altamente integrado. El elemento de sensado y el circuito de medición integrado, están ensamblados en un empaque plástico (DIL) con pines para montaje de superficie. Se encuentran protegidos de las condiciones ambientales con gel de silicio lo que provee un excelente desempeño y confiabilidad en ambientes húmedos y con cambios de temperatura. La dirección de las mediciones se encuentra en el plano paralelo al de montaje.

Principio de operación

Este dispositivo sensa la aceleración de una manera simple y confiable mediante la inercia de la masa, de acuerdo a la segunda ley de Newton. Los elementos básicos del acelerómetro son el cuerpo, el resorte y la masa de prueba. Cuando cambia la velocidad del cuerpo del sensor, la masa es forzada a seguir este cambio a través del acoplamiento del resorte. Se requiere de una fuerza para cambiar el movimiento de la masa de prueba, debido a esta fuerza el resorte es tensado y la distancia entre el cuerpo y la masa de prueba cambia en proporción a la aceleración del cuerpo.

El movimiento entre el cuerpo y la masa es detectado debido a que estos se encuentran aislados uno del otro y su capacitancia es medida. Mientras la distancia decrementa, la capacitancia se incrementa y la corriente eléctrica fluye hacia el sensor, por el otro lado, mientras la distancia se incrementa sucede lo contrario.

El sensor convierte la aceleración del cuerpo en un voltaje o corriente eléctrica, la tecnología empleada para su construcción los hace adecuados para detectar pequeños cambios de movimiento. El elemento para sensar la aceleración esta construido de un cristal único de silicio y de vidrio, lo que le proporciona una gran precisión y principalmente estabilidad en términos de tiempo y temperatura. El elemento de sensado capacitivo mide la aceleración en sentido positivo y negativo sobre el mismo eje, además de ser sensible a la aceleración estática y a la vibración.

Sensor electrolítico

Este tipo de sensor pertenece a los denominados sumergidos en líquido. Su principio de operación está basado en un fluido eléctricamente conductivo (electrolito) sellado en una cavidad de vidrio o cerámica para conducir entre un electrodo positivo, negativo y un común, la figura B2 muestra gráficamente este principio.

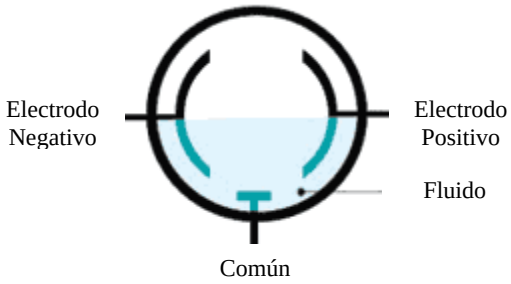


Figura B2 Sensor electrolítico

Mientras los electrodos positivo y negativo se encuentren sumergidos en el fluido se mantiene un nivel eléctrico nulo, lo que produce una señal de salida balanceada entre los electrodos positivo, negativo y común.

Una vez que el sensor es rotado sobre el eje sensible, el área sumergida de un electrodo en el líquido se incrementará y simultáneamente decrementará para el otro creando un desbalanceo en la salida, como se muestra en la figura B3. La relación de desbalanceo de un electrodo con respecto al otro, es directamente proporcional al ángulo de rotación.

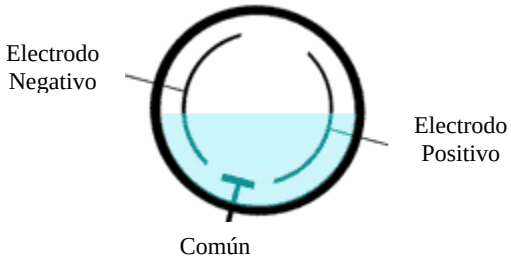


Figura B3 Sensor Electrolítico

Los sensores que funcionan con este principio, están limitados a un rango de operación menor a ±90º debido a que los electrodos estarían con ausencia o presencia total de líquido cuando se acercan a este valor, lo cual provocaría una saturación y no produciría cambios en la salida. Mientras que por el otro lado, mediciones por debajo de los ±15º requieren de un aumento significativo en el radio de la cavidad del fluido.

Alterando el perfil de la cavidad interna se puede lograr atenuar la sensibilidad del dispositivo a la vibración, lo que se ha detectado un problema típico en los sensores llenados con líquido.

Cuando es necesario sensar en dos ejes X e Y, se requiere dos conjuntos de electrodos colocados ortogonalmente entre sí.

Aplicaciones

La aplicación de estos dispositivos depende en gran medida del tipo de sensor que se utiliza, aunque en general son empleados en procesos donde se requiere mantener un cierto nivel angular.

Existen aplicaciones en el área de la construcción para sistemas de seguridad en grúas de carga, en la industria automotriz se emplean en máquinas de alineación. Entre otras aplicaciones se encuentran el posicionamiento de antenas satelitales y la compensación de inclinación en brújulas electrónicas.

APÉNDICE C Memoria flash

La memoria de flash es una forma de EEPROM que permite que múltiples localidades de memoria sean borradas o escritas en una sola operación de programación.
Esta memoria se deteriora después de un cierto número de operaciones de borrado, debido a la degradación en la capa de óxido que rodea el mecanismo de carga a través del cual se almacenan los datos.

La memoria flash está basada en la tecnología de semiconductor de óxido metálico, con compuerta flotante y efecto de inyección de avalancha (FAMOS transistor), que consiste esencialmente en un transistor NMOS al cual se le ha adicionado un conductor flotante suspendido entre el Gate y el canal, su construcción y representación simbólica se muestran en la figura C1.

La memoria flash almacena la información en un arreglo de transistores llamado celda, cada una de estas almacena un BIT información. En la flash negada o NOR flash, cada celda está compuesta por un dispositivo muy parecido a un transistor MOSFET, excepto que el de éstas tiene dos Gates en lugar de uno. El primero es el Gate de control (CG), como el de cualquier MOSFET, y el segundo es un Gate flotante, que está aislado completamente por una capa de óxido de silicio. El Gate flotante se ubica en medio del Gate de control y del sustrato. Gracias a que el Gate flotante se encuentra aislado por la capa de óxido de silicio, es capaz de mantener cualquier carga de electrones que quede atrapado en ella, siendo este mecanismo ocupado para el almacenamiento de datos. Cuando existen electrones atrapados en el Gate flotante el campo eléctrico generado por la compuerta de control es parcialmente cancelando, modificando el voltaje de disparo Vt de la celda.

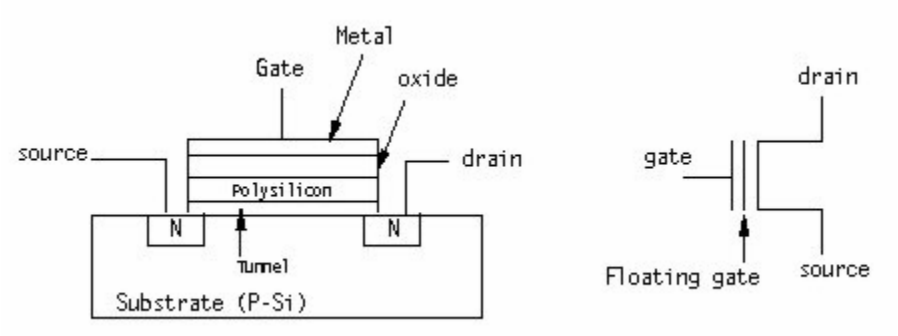


Figura C1 Construcción y representación simbólica del transistor NMOS

La celda de una memoria NOR flash es programada (“0” lógico) haciendo fluir una corriente de electrones provenientes del source (fuente) hacia el drain (drenaje), mientras se aplica un voltaje alto en el Gate de control, para generar un campo eléctrico fuerte que inyecte algunos electrones desde el canal hasta el Gate flotante. El proceso de borrado (escribir un “1” lógico) consiste en aplicar un voltaje alto entre el gate de control y el source mientras se deja abierto el drain, este voltaje genera un campo eléctrico que expulsa los electrones almacenados en el gate flotante gracias al efecto de túnel Fowler-Nordheim.

Las celdas están unidas en arreglos que suelen llamarse bloques como se muestra en la figura C2. Aunque es posible escribir y leer un dato cualquiera de forma aleatoria, no es posible borrar una celda de manera individual, siendo este proceso realizado por bloques.

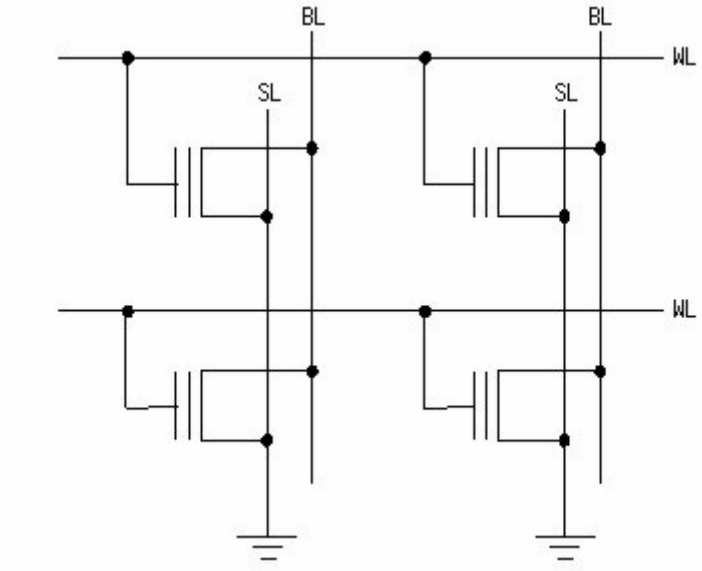


Figura C2 Arreglo de celdas

En estos bloques los datos están almacenados por palabras de datos, cuando se lee una de estas palabras se selecciona la línea de la palabra correspondiente (WL) y se prueba el voltaje del drain de cada una de las celdas que la conforman. Los transistores inhibidos por la acción de la compuerta flotante presentan voltajes altos mientras que el resto son bajados a tierra. Por ser negada los voltajes altos son leídos como “0” y los bajos como “1”

APÉNDICE D Programación del microcontrolador PIC18F452

El microcontrolador puede ser programado usando la tecnología de programación en el circuito mismo ICSP (In-Circuit Serial Programming) en las modalidades de alto y bajo voltaje.

Programación ICSP de alto voltaje

Los pines utilizados para la programación del microcontrolador son descritos en la tabla D1.

Nombre del Pin	Función
VPP	Habilitación de programación
VDD	Polarización
VSS	Tierra
SCLK	Reloj serial
SDATA	Datos seriales

Tabla D1 Pines utilizados en el modo de programación ICSP

Donde V_{DD}=13v , V_{DD}=5V.

Mapa de memoria

La memoria de programa se extiende desde la localidad 0000h hasta la 7FFFh (32 Kbytes) y esta agrupada en cuatro bloques de 8-Kbytes, las primeras localidades de la 0000h y 01FFh son un bloque especial llamado boot block (bloque de arranque) como se muestra en la figura D1.

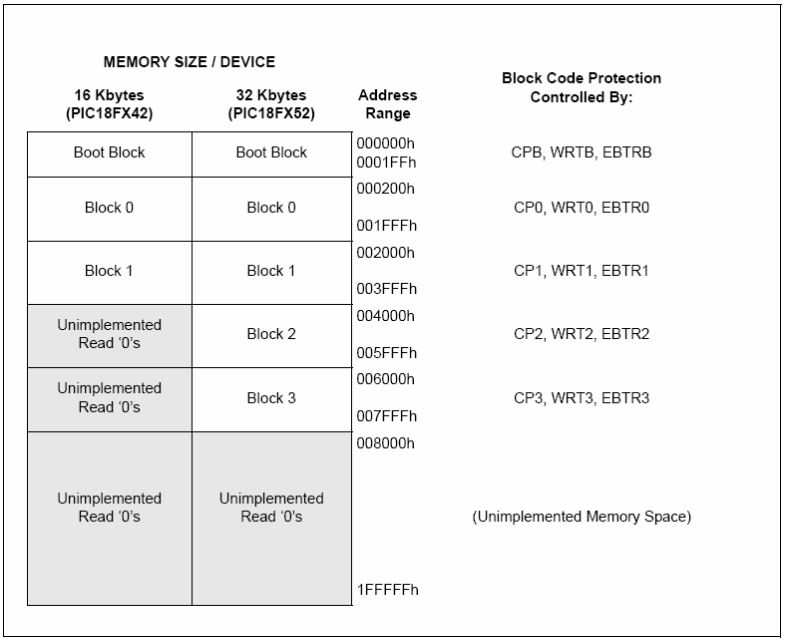


Figura D1 Distribución de la memoria de programa

Aunado al espacio de memoria de programa existen otros 3 bloques especiales. El primero es de identificación de usuario ID (Identification), el segundo de registros de configuración y el último, guarda información sobre el dispositivo. El contenido y ubicación de estos bloques se muestra en la figura D2.

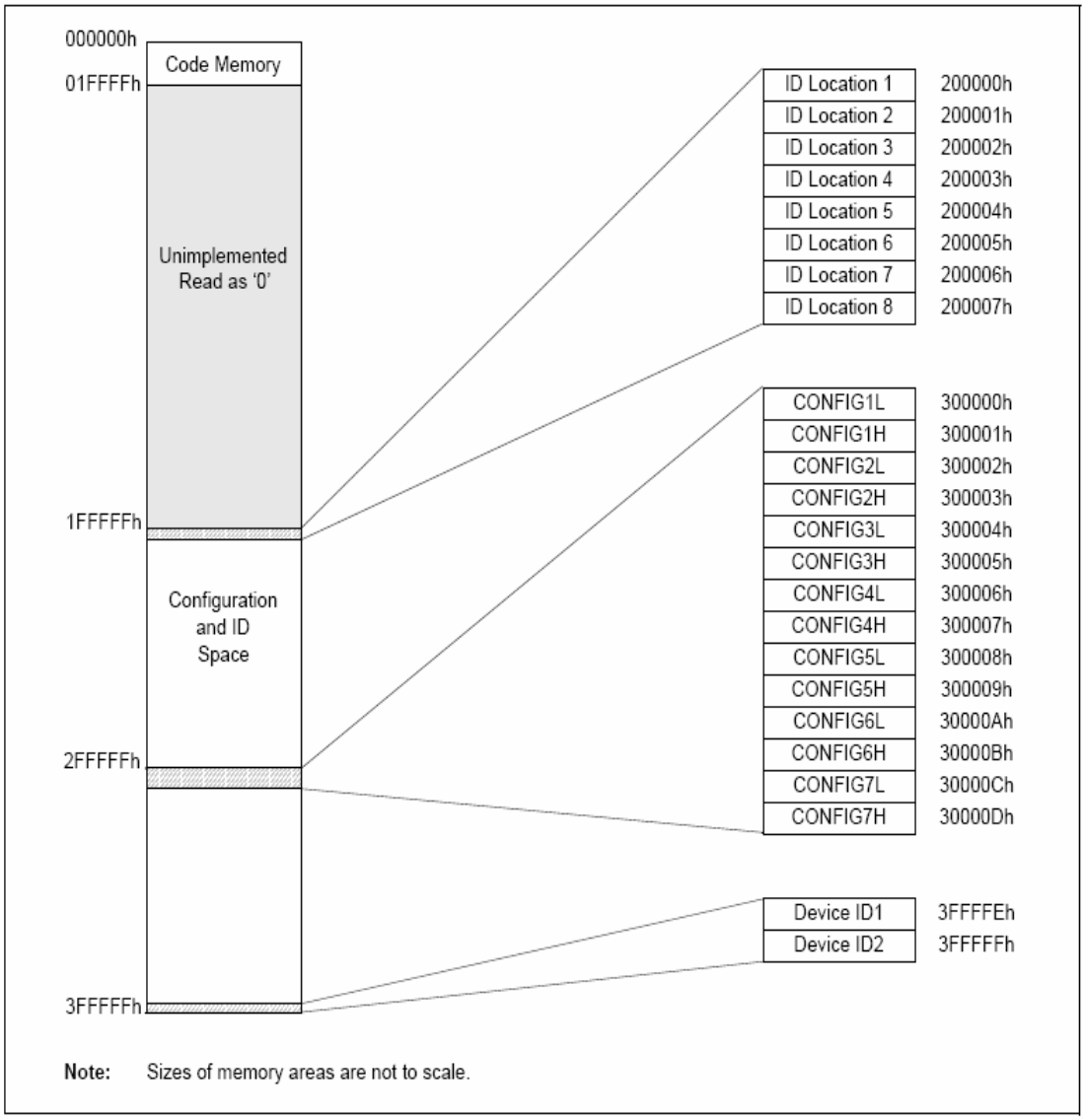


Figura D2 Descripción de los bloques especiales

Apuntador de memoria

La memoria (000000h -3FFFFFh) es direccionada a través del apuntador de tabla TBLPTR (Table pointer) mediante sus tres registros, superior, alto e inferior los cuales están distribuidos en memoria como se muestra en la tabla D2..

TBLPTRU	TBLPTRH	TBLPTRL
Addr[21:16]	Addr[15:8]	Addr[7:0]

Tabla D2 Registros superior, alto e inferior del TBLPTR

La instrucción '0000' es usada para ejecutar instrucciones en núcleo del microcontrolador, tales como MOVWF o MOVLW que son útiles para poder cargar la dirección deseada en el apuntador TBLPTR y otros registros.

Operación en modo de programación serial

Para entrar al modo de programación serial debe mantenerse el reloj SCLK y el canal de datos SDATA en unible bajo, para posteriormente elevar el voltaje Vpp a un nivel muy alto (13v). Una vez hecho esto se puede direccionar y programar la memoria de programa, las localidades ID, los bits de configuración y la memoria EEPROM.

El pin SCLK es usado como reloj de sincronía para la transmisión serial. Los datos de entrada, salida y los comandos son transmitidos a través del SDATA. Los comandos y los datos son trasmitidos en el flanco de subida y mantenidos durante el flanco de bajada del SCLK.

Todas las instrucciones son de 20 bits de largo, y están constituidas por un comando con extensión de 4 bits seguido de un operador de 16 bits cuya función depende del comando que se esté ejecutando, pudiendo ser 16 bits de datos de entrada u 8 bits de entrada y 8 de salida.

Description	4-Bit Command
Core Instruction (Shift in16-bit instruction)	0000
Shift out TABLAT register	0010
Table Read	1000
Table Read, post-increment	1001
Table Read, post-decrement	1010
Table Read, pre-increment	1011
Table Write	1100
Table Write, post-increment by 2	1101
Table Write, post-decrement by 2	1110
Table Write, start programming	1111

Figura D3 Comandos con extensión de cuatro bits

Las instrucciones son enviadas empezando por el bit menos significativo del comando, para posteriormente enviar el operador iniciando también por el bit menos significativo. Las figuras D4 y D5 muestran ejemplos de la construcción de las instrucciones y de la transmisión de los comandos respectivamente.

4-Bit Command	Data Payload	Core Instruction
1101	3C 40	Table Write, post-increment by 2

Figura D4 Ejemplo de la construcción de las instrucciones

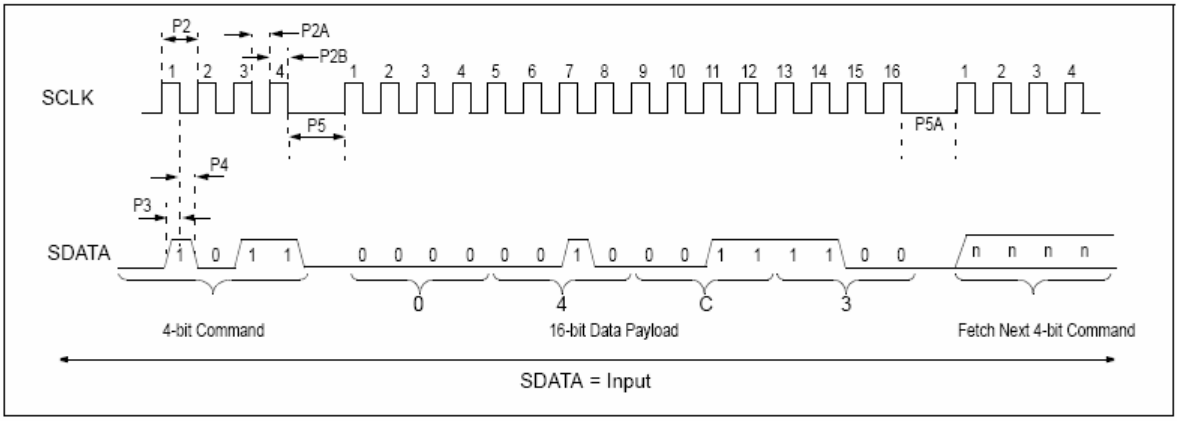


Figura D5 Ejemplo de transmisión de comandos

El diagrama de flujo de la figura D6 ilustra el proceso completo de programación del microprocesador.

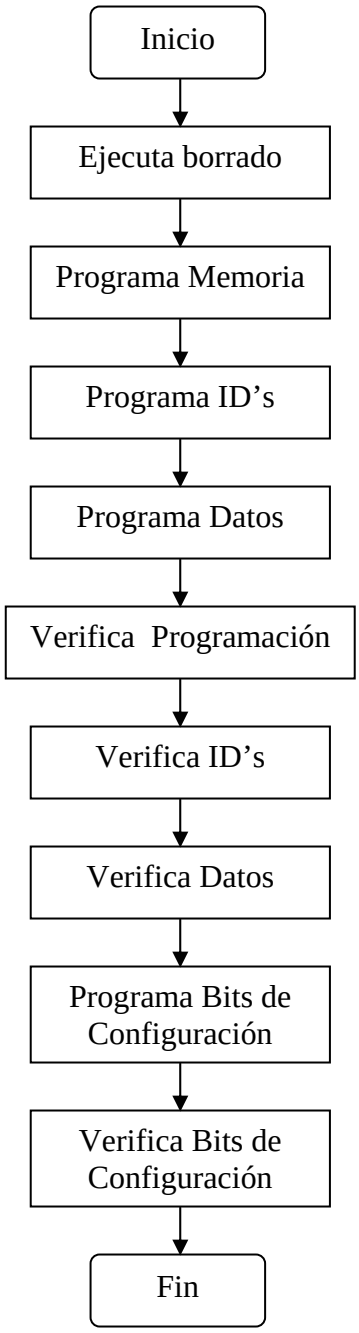


Figura D6 Diagrama de flujo de programación

La figura D7 especifica las condiciones de operación para el modo de programación y verificación.

Standard Operating Conditions
Operating Temperature: 10°C to 50°C unless otherwise indicated

Param No.	Sym	Characteristic	Min	Max	Units	Conditions
D110	VIHH	High Voltage Programming Voltage on MCLR/VPP	9.00	13.25	V	
D110A	VIHL	Low Voltage Programming Voltage on MCLR/VPP	2.00	5.50	V	
D111	VDD	Supply Voltage during programming	2.00	5.50	V	Normal programming
			4.50	5.50	V	Bulk erase operations
D112	IPP	Programming Current on MCLR/VPP	—	300	µA	
D113	IDDP	Supply Current during programming	—	5	mA	
D031	VIL	Input Low Voltage	VSS	0.2 VSS	V	
D041	VIH	Input High Voltage	0.8 VDD	VDD	V	
D080	VOL	Output Low Voltage	—	0.6	V	IOL = 8.5 mA
D090	VOH	Output High Voltage	VDD – 0.7	—	V	IOH = -3.0 mA
D012	CIO	Capacitive loading on I/O pin (SDATA)	—	50	pF	To meet AC specifications
P2	Tsclk	Serial Clock (SCLK) period	100	—	ns	VDD = 5.0V
			1	—	µs	VDD = 2.0V
P2A	TsclkL	Serial Clock (SCLK) Low time	40	—	ns	VDD = 5.0V
			400	—	ns	VDD = 2.0V
P2B	TsclkH	Serial Clock (SCLK) High time	40	—	ns	VDD = 5.0V
			400	—	ns	VDD = 2.0V
P3	Tset1	Input Data Setup Time to serial clock ↓	15	—	ns	
P4	Thld1	Input Data Hold Time from SCLK ↓	15	—	ns	
P5	Tdly1	Delay between 4-bit command and command operand	20	—	ns	
P5A	Tdly1a	Delay between 4-bit command operand and next 4-bit command	20	—	ns	
P6	Tdly2	Delay between last SCLK ↓ of command byte to first SCLK ↑ of read of data word	20	—	ns	
P9	Tdly5	SCLK High time (minimum programming time)	1	—	ms	
P10	Tdly6	SCLK Low time after programming (high voltage discharge time)	5	—	µs	
P11	Tdly7	Delay to allow self-timed data write or bulk erase to occur	5	—	ms	
P12	Thld2	Input Data Hold time from MCLR/VPP ↑	2	—	µs	
P13	Tset2	VDD ↑ Setup time to MCLR/VPP ↑	100	—	ns	
P14	Tvalid	Data Out Valid from SCLK ↑	10	—	ns	
P15	Tset3	PGM ↑ Setup time to MCLR/VPP ↑	2	—	µs	

Figura D7 Tabla de requerimientos de AC, DC y tiempo, para el modo de programación y verificación

APÉNDICE E Código de programa

```
;*****************************************************************************
;  THIS FILE IS A BASIC TEMPLATE FOR CREATING RELOCATABLE ASSEMBLY CODE FOR
*
;  A PIC18F452. COPY THIS FILE INTO YOUR PROJECT DIRECTORY AND MODIFY OR
*
;  ADD TO IT AS NEEDED. CREATE A PROJECT WITH MPLINK AS THE LANGUAGE TOOL
*
;  FOR THE HEX FILE. ADD THIS FILE AND THE 18F452.LKR FILE TO THE PROJECT.   *
;                                                *
;  THE PIC18FXXX ARCHITECTURE ALLOWS TWO INTERRUPT CONFIGURATIONS. THIS      *
;  TEMPLATE CODE IS WRITTEN FOR PRIORITY INTERRUPT LEVELS AND THE IPEN BIT   *
;  IN THE RCON REGISTER MUST BE SET TO ENABLE PRIORITY LEVELS. IF IPEN IS    *
;  LEFT IN ITS DEFAULT ZERO STATE, ONLY THE INTERRUPT VECTOR AT 0X008 WILL   *
;  BE USED AND THE WREG_TEMP, BSR_TEMP AND STATUS_TEMP VARIABLES WILL NOT    *
;  BE NEEDED.                                    *
;                                                *
;  REFER TO THE MPASM USER'S GUIDE FOR ADDITIONAL INFORMATION ON THE         *
;  FEATURES OF THE ASSEMBLER AND LINKER.                 *
;                                                *
;  REFER TO THE PIC18FXX2 DATA SHEET FOR ADDITIONAL INFORMATION ON THE       *
;  ARCHITECTURE AND INSTRUCTION SET.                 *
;                                                *
;*****************************************************************************
;                                                *
;   FILENAME:                                   *
;   DATE:                                     *
;   FILE VERSION:                                 *
;                                                *
;   AUTHOR:                                   *
;   COMPANY:                                    *
;                                                *
;*****************************************************************************
;                                                *
;   FILES REQUIRED:       P18F452.INC                     *
;                   18F452.LKR                  *
;                                                *
;*****************************************************************************

        LIST P=18F452, F=INHX32 ;DIRECTIVE TO DEFINE PROCESSOR AND FILE FORMAT
        #INCLUDE <P18F452.INC>          ;PROCESSOR SPECIFIC VARIABLE DEFINITIONS

;*****************************************************************************
;CONFIGURATION BITS
; THE __CONFIG DIRECTIVE DEFINES CONFIGURATION DATA WITHIN THE .ASM FILE.
; THE LABELS FOLLOWING THE DIRECTIVE ARE DEFINED IN THE P18F452.INC FILE.
; THE PIC18FXX2 DATA SHEET EXPLAINS THE FUNCTIONS OF THE CONFIGURATION BITS.
; CHANGE THE FOLLOWING LINES TO SUIT YOUR APPLICATION.

        __CONFIG        _CONFIG1H, _OSCS_ON_1H & _HSPLL_OSC_1H
        __CONFIG        _CONFIG2L, _BOR_OFF_2L & _BORV_20_2L & _PWRT_ON_2L
        __CONFIG        _CONFIG2H, _WDT_OFF_2H & _WDTPS_128_2H
        __CONFIG        _CONFIG3H, _CCP2MX_ON_3H
        __CONFIG        _CONFIG4L, _STVR_ON_4L & _LVP_OFF_4L & _DEBUG_OFF_4L
        __CONFIG        _CONFIG5L, _CP0_OFF_5L & _CP1_OFF_5L & _CP2_OFF_5L & _CP3_OFF_5L
        __CONFIG        _CONFIG5H, _CPB_OFF_5H & _CPD_OFF_5H
        __CONFIG        _CONFIG6L, _WRT0_OFF_6L & _WRT1_OFF_6L & _WRT2_OFF_6L &
                        _WRT3_OFF_6L
        __CONFIG        _CONFIG6H, _WRTC_OFF_6H & _WRTB_OFF_6H & _WRTD_OFF_6H
```

```
        __CONFIG    _CONFIG7L, _EBTR0_OFF_7L & _EBTR1_OFF_7L & _EBTR2_OFF_7L &
                    _EBTR3_OFF_7L
        __CONFIG    _CONFIG7H, _EBTRB_OFF_7H


;###############################################################################################
; DECLARACION DE VARIABLES

VARIABLE_DIFUSA_A                   EQU 0X00
VARIABLE_DIFUSA_B                   EQU 0X01
INDICE_SUMA                         EQU 0X02
NUMERADOR_SUPERIOR                  EQU 0X03
NUMERADOR_MEDIO                     EQU 0X04
NUMERADOR_INFERIOR                      EQU 0X05
DENOMINADOR_ALTO                    EQU 0X06
DENOMINADOR_BAJO                    EQU 0X07
CONTADOR_CONJUNTOS_DIFUSOS          EQU 0X08
CONTADOR_ENTRADAS_CRISP             EQU 0X09
CONTADOR_SALIDAS                    EQU 0X0A
PRERESULTADO_DIV_2                  EQU 0X0B
PRERESULTADO_DIV_1                  EQU 0X0C
PRERESULTADO_DIV_0                  EQU 0X0D
PRERESULTADO_DIV_22                 EQU 0X0E
PRERESULTADO_DIV_11                 EQU 0X0F
RESIDUO_SUPERIOR                    EQU 0X10
RESIDUO_MEDIO                       EQU 0X11
RESIDUO_INFERIOR                    EQU 0X12
CARRY_DIV                           EQU 0X13
CONTADOR_CICLOS_DE_DIVISION         EQU 0X14
COCIENTE                            EQU 0X15        ;OCUPA 8 ESPACIOS DE
MEMORIAERROR_ACTUAL_X1              EQU 0X20
DERIVADA_DE_ERROR_X1                EQU 0X21
INDICE_X1                           EQU 0X22
ERROR_ANTERIOR_X1                   EQU 0X23
ERROR_ACTUAL_X2                     EQU 0X24
DERIVADA_DE_ERROR_X2                EQU 0X25
INDICE_X2                           EQU 0X26
ERROR_ANTERIOR_X2                   EQU 0X27
ERROR_ACTUAL_Y1                     EQU 0X28
DERIVADA_DE_ERROR_Y1                EQU 0X29
INDICE_Y1                           EQU 0X2A
ERROR_ANTERIOR_Y1                   EQU 0X2B
ERROR_ACTUAL_Y2                     EQU 0X2C
DERIVADA_DE_ERROR_Y2                EQU 0X2D
INDICE_Y2                           EQU 0X2E
ERROR_ANTERIOR_Y2                   EQU 0X2F
TEMP_X                              EQU 0X30
TEMP_Y                              EQU 0X31
APUNTADOR_CICLO_0                   EQU 0X32
APUNTADOR_CICLO_1                   EQU 0X33
APUNTADOR_CICLO_2                   EQU 0X34
APUNTADOR_CICLO_3                   EQU 0X35
APUNTADOR_CICLO_4                   EQU 0X36
SENSOR_X                            EQU 0X37
SENSOR_Y                            EQU 0X38
CONTADOR_MAESTRO                    EQU 0X39
INDICE_DE_SALIDA                    EQU 0X3A
TABLA_PWM                           EQU 0X3B
PWM_0                               EQU 0X3B
PWM_1                               EQU 0X3C
```

```
PWM_2                                   EQU 0X3D
PWM_3                                   EQU 0X3E
PWM_INTERRUP_1                          EQU 0X3F
PWM_INTERRUP_3                          EQU 0X40
ENTRADA_ACTUAL_1                        EQU 0X41
ENTRADA_ACTUAL_2                        EQU 0X42
SALIDA_CRISP                            EQU 0X43
ENTRADAS_DIFUSAS                        EQU 0X50        ;IGUAL AL NUMERO DE ENTRADAS
CRISP*NUMERO DE CONJUNTOS DIFUSOS
SALIDA_DIFUSA                           EQU 0X60         ;IGUAL AL NUMERO DE SALIDAS
CRISP*NUMERO DE CONJUNTOS DE SALIDA

CONTADOR_RETARDO_CONVERSION             EQU 0X70

INICIO_TABLA_CONJUNTOS_DIFUSOS          EQU 0X100       ;IGUAL A 32 POR NUMERO DE
ENTRADAS CRISP, PARA 8 CONJUNTOS DIFUSOS
INICIO_TABLA_SINGLETON                  EQU 0X160       ;IGUAL AL NUMERO DE CONJUNTOS
DE SALIDA*NUMERO DE SALIDAS CRISP
INICIO_TABLA_REGLAS                     EQU 0X168       ;REGLAS
RAM                                     EQU 0X100       ;LOCALIDAD  A PARTIR DE LA CUAL
SERAN ESCRITOS LOS VALORES
CONTADOR_RAM                            EQU 0X1F        ;LOCALIDAD DONDE SE GUARDARA
EL CONTADOR
FLASH_U                                 EQU 0X00        ;LOS VALORES FLASH APUNTAN A
LA PRIMERA LOCAOLIDAD EN LA QUE FUE ALMACENADA LA TABLA DE DATOS
FLASH_H                                 EQU     0X01    ;POR LO CUAL SI SE CAMBIA LA
UBICACION DE LA TABLA DEBE
FLASH_L                                 EQU 0X00        ;CAMBIARSE TAMBIEN EL LUGAR A
DONDE APUNTA FLASH

LONGITUD_TABLA_RAM                      EQU 0X7E        ;ESTE VALOR CONTROLA EL
NUMERO DE CICLOS, CADA CICLO ESCRIBE 2 BYTES, NUMERO DECICLOS=NO.BYTES/2
NUMERO_CONJUNTOS_DIFUSOS_ENTRADA        EQU 0X07        ;CONJUNTOS DE ENTRADA-1
(CAMBIAR POR EL NUMERO A UTILIZAR)
NUMERO_DE_CONJUNTOS_DE_SALIDA           EQU 0X07        ;CONJUNTOS DE SALIDA DIFUSOS
(CAMBIAR POR EL NUMERO A UTILIZAR)

;*****************************************************************************************************
;RESET VECTOR
; THIS CODE WILL START EXECUTING WHEN A RESET OCCURS.

;RESET_VECTOR
                ORG     0X0000
                GOTO    MAIN            ;GO TO START OF MAIN CODE

;*****************************************************************************************************
;HIGH PRIORITY INTERRUPT VECTOR
; THIS CODE WILL START EXECUTING WHEN A HIGH PRIORITY INTERRUPT OCCURS OR
; WHEN ANY INTERRUPT OCCURS IF INTERRUPT PRIORITIES ARE NOT ENABLED.

;HI_INT_VECTOR          CODE
                ORG     0X0008

                BRA     HIGHINT                 ;GO TO HIGH PRIORITY INTERRUPT ROUTINE


;*****************************************************************************************************
;HIGH PRIORITY INTERRUPT ROUTINE
; THE HIGH PRIORITY INTERRUPT CODE IS PLACED HERE.

                ;CODE
```

```
HIGHINT:
;           *** HIGH PRIORITY INTERRUPT CODE GOES HERE ***
                BTFSC   INTCON,TMR0IF           ;CHECA LA BANDERA DE INTERRUPCIÓN, SALE DE
                                                LA INT SI NO ESTA PRENDIDA
                BRA     INTERRUPCION_CERO

                BTFSC   PIR1,TMR1IF             ;CHECA LA BANDERA DE INTERRUPCIÓN, SALE DE
                                                LA INT SI NO ESTA PRENDIDA
                BRA     INTERRUPCION_UNO

                BTFSC   PIR2,TMR3IF             ;CHECA LA BANDERA DE INTERRUPCIÓN, SALE DE
                                                 LA INT SI NO ESTA PRENDIDA
                BRA     INTERRUPCION_TRES

                BRA     SALIDA_INTERRUPHIGH


INTERRUPCION_CERO
                BSF     PORTC,3
                BSF     PORTC,0
                MOVLW 0XFE                      ;VALOR PARA CONTEO DE TIMER0_ALTO
                MOVWF TMR0H
                MOVLW 0XFE                      ;VALOR PARA CONTEO DE TIMER0_BAJO
                MOVWF TMR0L                     ;EL PERIODO DEL TIMER_0 ES MAS GRANDE QUE
                                                TIMER_1 Y 3
                BSF     T0CON,TMR0ON            ;PRENDE EL TIMER_0


                COMF    PWM_INTERRUP_1,W        ;PWM_INTERRUP_1,W     ;----------------------------------------
                MULLW 0X02
                MOVLW0XFE                       ;AJUSTA EL VALOR PARA QUE EMPIECE A CONTAR
                                                EL TIMER_1
                ADDWF PRODH,W
                MOVWFTMR1H                      ;CONSIDERANDO UN PRESCALADOR DE 1:8 EN UNA
                                                 CUENTA DE 16 BITS
                INCF    PRODL
                MOVFF  PRODL,TMR1L              ;----------------------------------------

                MOVLW 0X31                      ;0X11 0X01 [8BITS READ/WRITE, 1:8 PRESCALE
VALUE, OSCILLATOR
                                                SHUT-OFF, INTERNAL CLOCK, TIMER ON]
                MOVWF T1CON
                BCF     INTCON,TMR0IF           ;BORRA BANDERA DE INTERRUPCIÓN TIMER_0
                BSF     PIE1,TMR1IE             ;HABILITA INTERRUPCIÓN POR DESBORDAMIENTO
                                                PARA TIMER1

                COMF    PWM_INTERRUP_3,W        ;PWM_INTERRUP_3,W     ;----------------------------------------
                MULLW 0X02
                MOVLW0XFE                       ;AJUSTA EL VALOR PARA QUE EMPIECE A CONTAR
                                                EL TIMER_3
                ADDWF PRODH,W
                MOVWFTMR3H                      ;CONSIDERANDO UN PRESCALADOR DE 1:8 EN UNA
                                                 CUENTA DE 16 BITS
                INCF    PRODL
                MOVFF  PRODL,TMR3L              ;----------------------------------------


                MOVLW 0X71                      ;0111 XX01 [8BITS READ/WRITE, TIMER3 IS THE
                                                CLOCK SOURCE, 1:8 PRESCALE VALUE, INTERNAL
                                                CLOCK, TIMER ON]
```

```
                MOVWF T3CON
                BSF     PIE2,TMR3IE             ;HABILITA INTERRUPCIÓN POR DESBORDAMIENTO
                                                PARA TIMER3

                BRA     SALIDA_INTERRUPHIGH

INTERRUPCION_UNO
                BCF     PORTC,0
                BCF     PIR1,TMR1IF             ;BORRA BANDERA DE INTERRUPCIÓN TIMER_1
                BCF     PIE1,TMR1IE             ;DESHABILITA INTERRUPCIÓN POR
                                                 DESBORDAMIENTO PARA TIMER1
                BRA     SALIDA_INTERRUPHIGH

INTERRUPCION_TRES
                BCF     PORTC,3
                BCF     PIR2,TMR3IF             ;BORRA BANDERA DE INTERRUPCIÓN TIMER_3
                BCF     PIE2,TMR3IE             ;DESHABILITA INTERRUPCIÓN POR
                                                DESBORDAMIENTO PARA TIMER1

SALIDA_INTERRUPHIGH
                RETFIE  FAST


;#############################################################################################
;**********INICIALIZACION DE LOS CONJUNTOS DIFUSOS EN RAM**************************

MAIN:           MOVLW LONGITUD_TABLA_RAM
                MOVWF CONTADOR_RAM
                MOVLW FLASH_H
                MOVWF TBLPTRH
                MOVLW FLASH_L
                MOVWF TBLPTRL
                MOVLW FLASH_U
                MOVWF TBLPTRU
                LFSR    FSR0,RAM
CICLO   MOVF    POSTINC0,W
                TBLRD*+
                MOVF    TABLAT,W
                MOVWF POSTDEC0
                TBLRD*+
                MOVF    TABLAT,W
                MOVWF POSTINC0
                MOVF    POSTINC0
                DECF    CONTADOR_RAM
                BZ              DIF
                BRA     CICLO
DIF             GOTO INICIO_GENERAL

                ORG     0X100                   ;SI SE MODIFICA ESTA DIRECCION SE TIENE
                                                QUE CAMBIAR EL LUGAR A DONDE APUNTA
                                                EL TBLPTR

;CONJUNTOS DIFUSOS DE ENTRADA

        DA      0X0000,0X5211                   ;NIVEL -3
        DA      0X5211,0X6111                   ;NIVEL -2
        DA      0X6111,0X7011                   ;NIVEL -1
        DA      0X7011,0X7F11                   ;NIVEL CERO
        DA      0X7F11,0X8E11                   ;NIVEL 1
        DA      0X8E11,0X9D11                   ;NIVEL 2
        DA      0X9D11,0XFF00                   ;NIVEL 3
```

```
        DA      0X0000,0X0000                   ;0
        DA      0X0000,0X5211                   ;BAJA MUY RAPIDO
        DA      0X5211,0X6111                   ;BAJA RAPIDO
        DA      0X6111,0X7011                   ;BAJA LENTO
        DA      0X7011,0X7F11                   ;ESTATICO
        DA      0X7F11,0X8E11                   ;SUBE LENTO
        DA      0X8E11,0X9D11                   ;SUBE RAPIDO
        DA      0X9D11,0XFF00                   ;SUBE MUY RAPIDO
        DA      0X0000,0X0000                   ;0

        DA      0X0000,0X0000                   ;E3 CONJUNTO_1
        DA      0X0000,0X0000                   ;E3 CONJUNTO_2
        DA      0X0000,0X0000                   ;E3 CONJUNTO_3
        DA      0X0000,0X0000                   ;E3 CONJUNTO_4
        DA      0X0000,0X0000                   ;E3 CONJUNTO_5
        DA      0X0000,0X0000                   ;E3 CONJUNTO_6
        DA      0X0000,0X0000                   ;E3 CONJUNTO_7
        DA      0X0000,0X0000                   ;E3 CONJUNTO_8

        ; CONJUNTOS DIFUSOS DE SALIDA

        DA      0X0046                          ;POTENCIA CERO, POTENCIA MUY PEQUEÑA
        DA      0X6482                          ;POTENCIA PEQUEÑA  ,POTENCIA
SUSTENTACIÓN
        DA      0XA0BE                          ;POTENCIA GRANDE  ,POTENCIA MUY
GRANDE
        DA      0XD200                          ;POTENCIA MAXIMA  ,

        ; REGLAS DIFUSAS

        DA      0X0008;     NIVEL -3   Y        BAJA MUY RAPIDO   Y
        DA      0X8600;     7.- POTENCIA MAXIMA         NIVEL -3   Y
        DA      0X0986;     BAJA RAPIDO   Y         7.- POTENCIA MAXIMA
        DA      0X000A;      NIVEL -3   Y         BAJA LENTO   Y
        DA      0X8400;     5.- POTENCIA GRANDE         NIVEL -3   Y
        DA      0X0B84;      ESTATICO   Y         5.- POTENCIA GRANDE
        DA      0X000C;      NIVEL -3   Y         SUBE LENTO   Y
        DA      0X8400;     5.- POTENCIA GRANDE         NIVEL -3   Y
        DA      0X0D84;      SUBE RAPIDO   Y         5.- POTENCIA GRANDE
        DA      0X000E;      NIVEL -3   Y         SUBE MUY RAPIDO   Y
        DA      0X8301;     4.- POTENCIA  SUSTENTACIÓN          NIVEL -2   Y
        DA      0X0885;     BAJA MUY RAPIDO   Y         6.- POTENCIA MUY GRANDE
        DA      0X0109;     NIVEL -2   Y         BAJA RAPIDO   Y
        DA      0X8501;     6.- POTENCIA MUY GRANDE          NIVEL -2   Y
        DA      0X0A84;      BAJA LENTO   Y         5.- POTENCIA GRANDE
        DA      0X010B;      NIVEL -2   Y         ESTATICO   Y
        DA      0X8401;     5.- POTENCIA GRANDE         NIVEL -2   Y
        DA      0X0C84;      SUBE LENTO   Y         5.- POTENCIA GRANDE
        DA      0X010D;      NIVEL -2   Y         SUBE RAPIDO   Y
        DA      0X8301;     4.- POTENCIA  SUSTENTACIÓN          NIVEL -2   Y
        DA      0X0E82;      SUBE MUY RAPIDO   Y         3.- POTENCIA PEQUEÑA
        DA      0X0208;     NIVEL -1   Y         BAJA MUY RAPIDO   Y
        DA      0X8502;     6.- POTENCIA MUY GRANDE          NIVEL -1   Y
        DA      0X0984;     BAJA RAPIDO   Y         5.- POTENCIA GRANDE
        DA      0X020A;      NIVEL -1   Y         BAJA LENTO   Y
        DA      0X8402;     5.- POTENCIA GRANDE         NIVEL -1   Y
        DA      0X0B84;      ESTATICO   Y         5.- POTENCIA GRANDE
        DA      0X020C;      NIVEL -1   Y         SUBE LENTO   Y
        DA      0X8302;     4.- POTENCIA  SUSTENTACIÓN          NIVEL -1   Y
        DA      0X0D82;      SUBE RAPIDO   Y         3.- POTENCIA PEQUEÑA
        DA      0X020E;      NIVEL -1   Y         SUBE MUY RAPIDO   Y
```

```
        DA      0X8103;     2.- POTENCIA MUY PEQUEÑA        NIVEL CERO   Y
        DA      0X0884;     BAJA MUY RAPIDO   Y       5.- POTENCIA GRANDE
        DA      0X0309;     NIVEL CERO   Y       BAJA RAPIDO   Y
        DA      0X8403;     5.- POTENCIA GRANDE         NIVEL CERO   Y
        DA      0X0A83;      BAJA LENTO   Y       4.- POTENCIA  SUSTENTACIÓN
        DA      0X030B;     NIVEL CERO   Y       ESTATICO   Y
        DA      0X8303;     4.- POTENCIA  SUSTENTACIÓN        NIVEL CERO   Y
        DA      0X0C83;     SUBE LENTO   Y       4.- POTENCIA  SUSTENTACIÓN
        DA      0X030D;     NIVEL CERO   Y       SUBE RAPIDO   Y
        DA      0X8203;     3.- POTENCIA PEQUEÑA         NIVEL CERO   Y
        DA      0X0E82;     SUBE MUY RAPIDO   Y       3.- POTENCIA PEQUEÑA
        DA      0X0408;     NIVEL 1   Y       BAJA MUY RAPIDO   Y
        DA      0X8404;     5.- POTENCIA GRANDE         NIVEL 1   Y
        DA      0X0984;     BAJA RAPIDO   Y       5.- POTENCIA GRANDE
        DA      0X040A;     NIVEL 1   Y       BAJA LENTO   Y
        DA      0X8304;     4.- POTENCIA  SUSTENTACIÓN        NIVEL 1   Y
        DA      0X0B82;     ESTATICO   Y       3.- POTENCIA PEQUEÑA
        DA      0X040C;     NIVEL 1   Y       SUBE LENTO   Y
        DA      0X8204;     3.- POTENCIA PEQUEÑA         NIVEL 1   Y
        DA      0X0D82;     SUBE RAPIDO   Y       3.- POTENCIA PEQUEÑA
        DA      0X040E;     NIVEL 1   Y       SUBE MUY RAPIDO   Y
        DA      0X8105;     2.- POTENCIA MUY PEQUEÑA        NIVEL 2   Y
        DA      0X0884;     BAJA MUY RAPIDO   Y       5.- POTENCIA GRANDE
        DA      0X0509;     NIVEL 2   Y       BAJA RAPIDO   Y
        DA      0X8405;     5.- POTENCIA GRANDE         NIVEL 2   Y
        DA      0X0A83;      BAJA LENTO   Y       4.- POTENCIA  SUSTENTACIÓN
        DA      0X050B;     NIVEL 2   Y       ESTATICO   Y
        DA      0X8205;     3.- POTENCIA PEQUEÑA         NIVEL 2   Y
        DA      0X0C82;     SUBE LENTO   Y       3.- POTENCIA PEQUEÑA
        DA      0X050D;     NIVEL 2   Y       SUBE RAPIDO   Y
        DA      0X8105;     2.- POTENCIA MUY PEQUEÑA        NIVEL 2   Y
        DA      0X0E80;     SUBE MUY RAPIDO   Y       1.- POTENCIA CERO
        DA      0X0608;     NIVEL 3   Y       BAJA MUY RAPIDO   Y
        DA      0X8406;     5.- POTENCIA GRANDE         NIVEL 3   Y
        DA      0X0984;     BAJA RAPIDO   Y       5.- POTENCIA GRANDE
        DA      0X060A;     NIVEL 3   Y       BAJA LENTO   Y
        DA      0X8306;     4.- POTENCIA  SUSTENTACIÓN        NIVEL 3   Y
        DA      0X0B82;     ESTATICO   Y       3.- POTENCIA PEQUEÑA
        DA      0X060C;     NIVEL 3   Y       SUBE LENTO   Y
        DA      0X8206;     3.- POTENCIA PEQUEÑA         NIVEL 3   Y
        DA      0X0D81;     SUBE RAPIDO   Y       2.- POTENCIA MUY PEQUEÑA
        DA      0X060E;     NIVEL 3   Y       SUBE MUY RAPIDO   Y
        DA      0X80FF;     1.- POTENCIA CERO

;#############################################################################################
                                        INICIO_GENERAL
;**********************INICIALIZACIÓN DE PWM CON INTERRUPCIONES**************************
;************LA CONFIGURACIÓN DE LAS INTERRUPCIONES DEBE IR AL INICIO DEL CODIGO**********

        CLRF    TRISC                   ;PUERTO C COMO SALIDA
        CLRF    PORTC

        BSF     RCON,IPEN               ;HABILITA PRIORIDAD EN LASINTERRUPCIONES
        MOVLW 0XA0                      ;SE HABILITA INTERRUPCIÓN GLOBAL, SE
                                        HABILITA INTERRUPCIÓN POR DESBORDAMIENTO
                                        PARA TIMER0
        MOVWF INTCON                    ;1X1X X0XX[GIENABLE, TMR0I ENABLE, SE BORRA
                                        BANDERA DE INTERRUPCIÓN]

        BSF     INTCON2,TMR0IP          ;ALTA PRIORIDAD PARA LA INTERRUPCIÓN DEL
TIMER0
```

```
        MOVLW 0XFE
        MOVWF TMR0H                     ;VALOR PARA CONTEO DE TIMER0_ALTO
        MOVLW0XFB
        MOVWF TMR0L                     ;VALOR PARA CONTEO DE TIMER0_BAJO
        MOVLW 0X83                      ;1000 0011 [TIMER0 ON, 16BITS, INTERNAL CLOCK,
LOW TO
                                        HIGH TRANSITION, PRESCALER ASIGNED, 1:16]
        MOVWF T0CON
        BCF     PIR2,TMR3IF             ;BORRA LA BANDERA DE INTERRUPCIÓN
                                        PARA TIMER3
        BSF     IPR2,TMR3IP             ;HABILITA PRIORIDAD PARA LA INTERRUPCIÓN EN
                                        TIMER3 (0=BAJA 1=ALTA)

        BCF     PIR1,TMR1IF             ;BORRA LA BANDERA DE INTERRUPCIÓN PARA
                                        TIMER1
        BSF     IPR1,TMR1IP             ;HABILITA PRIORIDAD PARA LA INTERRUPCIÓN EN
                                        TIMER1 (0=BAJA 1=ALTA)

;****************************************** PWM ************************************************
;**************SE TIENE QUE CONFIGURAR EL _CCP2MX_ EN ON PARA TENER EL PWM EN RC1**********

        MOVLW 0XFF                      ;CAMBIAR SEGUN LA FRECUENCIAREQUERIDA
        MOVWF PR2                       ;INICIALIZAMOS EL PERIODO PR2 PARA EL TIMER2
        MOVLW 0X07                      ;CAMBIAR SEGUN EL TIEMPO DEL PRESCALADOR
                                        DESEADO 0X07=16  0X05=4   0X04=1
        MOVWF T2CON                     ;TMR2 ENCENDIDO, TIMER PRESCALER (16) PWM
                                        FREC= 2.44KHZ
        MOVLW 0X0C
        MOVWF CCP1CON                   ;INICIALIZAMOS EL REGISTRO PARA OPERACIÓN DE
PWM
        MOVWF CCP2CON                   ;INICIALIZAMOS EL REGISTRO PARA OPERACIÓN DE
PWM

;*****************************INICIALIZACION DEL CONVERTIDOR A/D*****************************
;INIAD
        MOVLW 0XFF
        MOVWF TRISA                     ;INICIALIZO PUERTOA COMO ENTRADAS;
        CLRF PORTA
        MOVLW 0X42
        MOVWF ADCON1                    ;INICIALIZO CONVERSION DEL RELOJ (FOSC/64),
JUSTIFICADO A LA
                                        IZQUIERDA Y 8 ENTRADAS ANALOGICAS

;*********************************************************************************************
        MOVLW 0X04
        MOVWF CONTADOR_MAESTRO

; RELLENA CON VALORES INICIALES ERROR_ACTUAL Y PWMS
        MOVLW 0X80
        MOVWF ERROR_ACTUAL_X1
        MOVWF ERROR_ACTUAL_X2
        MOVWF ERROR_ACTUAL_Y1
        MOVWF ERROR_ACTUAL_Y2

        MOVWF PWM_0
        MOVWF PWM_1
        MOVWF PWM_2
        MOVWF PWM_3
```

```
CICLO_PRINCIPAL

;********************************************** INICIO **************************************

        MOVLW0X04
        CPFSLT CONTADOR_MAESTRO
        CALL   ORDENADOR_DE_PRIORIDAD
        CALL   TRANSFERENCIA_DE_ENTRADA
        CALL   RUTINA_DIFUSA
        CALL   TRANSFERENCIA_DE_SALIDA
        CALL   ACTUALIZACION_DE_PWMS
        INCF   CONTADOR_MAESTRO
        BRA    CICLO_PRINCIPAL

;#############################################################################################
;********************** SELECCIONA LA PRIORIDAD PARA LA CORRECIÓN EN LOS EJES
************************
;********************** CALCULA LA DERIVADA DEL ERROR ************************
ORDENADOR_DE_PRIORIDAD     ;SUBRUTINA

;TRANSFERENCIA DE ERRORES ACTUALES A ANTERIORES
        MOVFF  ERROR_ACTUAL_X1,ERROR_ANTERIOR_X1
        MOVFF  ERROR_ACTUAL_X2,ERROR_ANTERIOR_X2
        MOVFF  ERROR_ACTUAL_Y1,ERROR_ANTERIOR_Y1
        MOVF   ERROR_ACTUAL_Y2,ERROR_ANTERIOR_Y2

;CARGA NUEVOS ERRORES ACTUALES
        MOVFF  SENSOR_X,ERROR_ACTUAL_X1
        COMF   ERROR_ACTUAL_X1,W
        MOVWF ERROR_ACTUAL_X2
        MOVFF  SENSOR_Y,ERROR_ACTUAL_Y1
        COMF   ERROR_ACTUAL_Y1,W
        MOVWF ERROR_ACTUAL_Y2

;CALCULA LA DERIVADA DE X1
        MOVLW0X00
        MOVWF INDICE_X1
        MOVF   ERROR_ANTERIOR_X1,W
        SUBWF  ERROR_ACTUAL_X1,W          ;ERROR ACTUAL-ERROR ANTERIOR
        BTFSC  STATUS,C                   ;SI ES NEGATIVO SALTA UNA INSTRUCCION
        BRA    DERIVANDO_02_X1            ;SI LA RESTA ES POSITIVA SALTA A
                                          DERIVANDO_02_X1
        MOVWF DERIVADA_DE_ERROR_X1        ;SI ES NEGATIVA, INVESTIGA SI ES MENOR A 80
        MOVLW0X80
        CPFSLT DERIVADA_DE_ERROR_X1       ;SI ES NEGATIVA Y MENOR A 80 SALTA UNA
                                          INSTRUCCION
        BRA    DERIVANDO_01_X1            ;SI ES NEGATIVA Y MAYOR A 80 SALTA A
                                          DERIVANDO_01_X1
        MOVLW 0X80                        ;SI ES NEGATIVA Y MENOR A 80, HACE LA
                                          RESTA IGUAL A 80

;EN ESTE PUNTO POD DRIA UNA DESBORDARSE POR ARRIBA
        MOVWF DERIVADA_DE_ERROR_X1

DERIVANDO_01_X1
        MOVLW0X80
        ADDWF DERIVADA_DE_ERROR_X1,1      ;SUMA OFFSET A LA DERIVADA
        BRA    DERIVANDO_03_X1

DERIVANDO_02_X1
        MOVWF DERIVADA_DE_ERROR_X1
```

```
        MOVLW0X80
        ADDWF DERIVADA_DE_ERROR_X1,1    ;SUMA OFFSET A LA DERIVADA
        BTFSS   STATUS,OV               ;SALTA UNA INSTRUCCION SI LA SUMA SE
                                        DESBORDA
        BRA     DERIVANDO_03_X1         ;SI NO SE DESBORDA SALTA A
                                        DERIVANDO_03_X1
        MOVLW0XFF                       ;SI SE DESBORDA ENTONCES LO HACE IGUAL A FF
        MOVWF DERIVADA_DE_ERROR_X1

DERIVANDO_03_X1
;CALCULA LA DERIVADA DE X2
        MOVLW0X01
        MOVWF INDICE_X2
        MOVF    ERROR_ANTERIOR_X2,W
        SUBWF  ERROR_ACTUAL_X2,W        ;ERROR ACTUAL-ERROR ANTERIOR
        BTFSC   STATUS,C                ;SI ES NEGATIVO SALTA UNA INSTRUCCION
        BRA     DERIVANDO_02_X2         ;SI LA RESTA ES POSITIVA SALTA A
                                        DERIVANDO_02_X2
        MOVWF DERIVADA_DE_ERROR_X2      ;SI ES NEGATIVA, INVESTIGA SI ES EMNOR A 80
        MOVLW0X80
        CPFSLT  DERIVADA_DE_ERROR_X2    ;SI ES NEGATIVA Y MENOR A 80 SALTA UNA
                                        INSTRUCCION
        BRA     DERIVANDO_01_X2         ;SI ES NEGATIVA Y MAYOR A 80 SALTA A
                                        DERIVANDO_01_X2
        MOVLW0X80                       ;SI ES NEGATIVA Y MENOR A 80, HACE LA RESTA
                                        IGUAL A 80

;EN ESTE PUNTO POD DRIA UNA DESBORDARSE POR ARRIBA
        MOVWF DERIVADA_DE_ERROR_X2

DERIVANDO_01_X2
        MOVLW 0X80
        ADDWF DERIVADA_DE_ERROR_X2,1    ;SUMA OFFSET A LA DERIVADA
        BRA     DERIVANDO_03_X2

DERIVANDO_02_X2
        MOVWF DERIVADA_DE_ERROR_X2
        MOVLW0X80
        ADDWF DERIVADA_DE_ERROR_X2,1    ;SUMA OFFSET A LA DERIVADA
        BTFSS   STATUS,OV               ;SALTA UNA INSTRUCCION SI LA SUMA SE
                                        DESBORDA
        BRA     DERIVANDO_03_X2         ;SI NO SE DESBORDA SALTA A
                                        DERIVANDO_03_X2
        MOVLW0XFF                       ;SI SE DESBORDA ENTONCES LO HACE IGUAL A FF
        MOVWF DERIVADA_DE_ERROR_X2

DERIVANDO_03_X2
;CALCULA LA DERIVADA DE Y1
        MOVLW0X02
        MOVWF INDICE_Y1
        MOVF    ERROR_ANTERIOR_Y1,W
        SUBWF  ERROR_ACTUAL_Y1,W        ;ERROR ACTUAL-ERROR ANTERIOR
        BTFSC   STATUS,C                ;SI ES NEGATIVO SALTA UNA INSTRUCCION
        BRA     DERIVANDO_02_Y1         ;SI LA RESTA ES POSITIVA SALTA A
                                        DERIVANDO_02_Y1
        MOVWF DERIVADA_DE_ERROR_Y1      ;SI ES NEGATIVA, INVESTIGA SI ES EMNOR A 80
        MOVLW0X80
        CPFSLT  DERIVADA_DE_ERROR_Y1    ;SI ES NEGATIVA Y MENOR A 80 SALTA UNA
                                        INSTRUCCION
        BRA     DERIVANDO_01_Y1         ;SI ES NEGATIVA Y MAYOR A 80 SALTA A
                                         DERIVANDO_01_Y1
```

```
        MOVLW0X80                       ;SI ES NEGATIVA Y MENOR A 80, HACE LA RESTA
                                        IGUAL A 80

;EN ESTE PUNTO POD DRIA UNA DESBORDARSE POR ARRIBA
        MOVWF DERIVADA_DE_ERROR_Y1

DERIVANDO_01_Y1
        MOVLW0X80
        ADDWF DERIVADA_DE_ERROR_Y1,1    ;SUMA OFFSET A LA DERIVADA
        BRA   DERIVANDO_03_Y1

DERIVANDO_02_Y1
        MOVWF DERIVADA_DE_ERROR_Y1
        MOVLW0X80
        ADDWF DERIVADA_DE_ERROR_Y1,1    ;SUMA OFFSET A LA DERIVADA
        BTFSS STATUS,OV                 ;SALTA UNA INSTRUCCION SI LA SUMA SE
                                        DESBORDA
        BRA   DERIVANDO_03_Y1           ;SI NO SE DESBORDA SALTA A
                                        DERIVANDO_03_Y1
        MOVLW0XFF                       ;SI SE DESBORDA ENTONCES LO HACE IGUAL A FF
        MOVWF DERIVADA_DE_ERROR_Y1

DERIVANDO_03_Y1
;CALCULA LA DERIVADA DE Y2
        MOVLW0X03
        MOVWF INDICE_Y2
        MOVF  ERROR_ANTERIOR_Y2,W
        SUBWF ERROR_ACTUAL_Y2,W         ;ERROR ACTUAL-ERROR ANTERIOR
        BTFSC STATUS,C                  ;SI ES NEGATIVO SALTA UNA INSTRUCCION
        BRA   DERIVANDO_02_Y2           ;SI LA RESTA ES POSITIVA SALTA A
                                        DERIVANDO_02_Y2
        MOVWF DERIVADA_DE_ERROR_Y2      ;SI ES NEGATIVA, INVESTIGA SI ES EMNOR A 80
        MOVLW0X80
        CPFSLT DERIVADA_DE_ERROR_Y2     ;SI ES NEGATIVA Y MENOR A 80 SALTA UNA
                                        INSTRUCCION
        BRA   DERIVANDO_01_Y2           ;SI ES NEGATIVA Y MAYOR A 80 SALTA A
                                        DERIVANDO_01_Y2
        MOVLW0X80                       ;SI ES NEGATIVA Y MENOR A 80, HACE LA RESTA
                                        IGUAL A 80

;EN ESTE PUNTO PODRIA DESBORDARSE POR ARRIBA
        MOVWF DERIVADA_DE_ERROR_Y2

DERIVANDO_01_Y2
        MOVLW0X80
        ADDWF DERIVADA_DE_ERROR_Y2,1    ;SUMA OFFSET A LA DERIVADA
        BRA   DERIVANDO_03_Y2

DERIVANDO_02_Y2
        MOVWF DERIVADA_DE_ERROR_Y2
        MOVLW0X80
        ADDWF DERIVADA_DE_ERROR_Y2,1    ;SUMA OFFSET A LA DERIVADA
        BTFSS STATUS,OV                 ;SALTA UNA INSTRUCCION SI LA SUMA SE
                                        DESBORDA
        BRA   DERIVANDO_03_Y2           ;SI NO SE DESBORDA SALTA A
                                        DERIVANDO_03_Y2
        MOVLW0XFF                       ;SI SE DESBORDA ENTONCES LO HACE IGUAL A FF
        MOVWF DERIVADA_DE_ERROR_Y2

DERIVANDO_03_Y2
; LA RUTINA COMIENZA CALCULANDO EN EL EJE X
```

```
        MOVLW0X80
        SUBWF  SENSOR_X,W
        BTFSC   STATUS, C                   ; SALTA UNA INSTRUCCION SI EL ANGULO
                                                    ES NEGATIVO
        BRA     X_POSITIVO                  ;SALTA
        MOVF    SENSOR_X,W
        SUBLW  0X80
        MOVWFTEMP_X
        MOVLWERROR_ACTUAL_X1
        MOVWFAPUNTADOR_CICLO_1
        MOVLWERROR_ACTUAL_X2
        MOVWFAPUNTADOR_CICLO_3
        BRA     EMPIEZA_Y

X_POSITIVO
        MOVWFTEMP_X
        MOVLWERROR_ACTUAL_X2
        MOVWFAPUNTADOR_CICLO_1
        MOVLWERROR_ACTUAL_X1
        MOVWFAPUNTADOR_CICLO_3

EMPIEZA_Y
        MOVLW0X80
        SUBWF  SENSOR_Y,W
        BTFSC   STATUS, C                   ; SALTA UNA INSTRUCCION SI EL ANGULO ES
                                            NEGATIVO
        BRA     Y_POSITIVO                  ;SALTA SI Y2 ES EL MOTOR MAS BAJO
        MOVF    SENSOR_Y,W                  ;Y1 ES EL MOTOR MAS BAJO
        SUBLW  0X80
        MOVWFTEMP_Y
        CPFSGT TEMP_X                       ;SI TEMP_X > TEMP_Y  =>  SALTA UNA INSTRUCCION
        BRA     Y1_ES_MAYOR                 ;SALTA SI Y1 ES EL MOTOR MAS BAJO
        MOVLWERROR_ACTUAL_Y1                ;Y1 ES EL SEGUNDO MOTOR MAS BAJO
        MOVWFAPUNTADOR_CICLO_2
        MOVLWERROR_ACTUAL_Y2
        MOVWFAPUNTADOR_CICLO_4
        BRA     FINALIZA_Y

Y1_ES_MAYOR                                 ;Y1 ES EL MOTOR MAS BAJO
        MOVLWERROR_ACTUAL_Y1
        MOVWFAPUNTADOR_CICLO_0
        MOVLWERROR_ACTUAL_Y2
        MOVWFAPUNTADOR_CICLO_2
        MOVFF  APUNTADOR_CICLO_3,APUNTADOR_CICLO_4
        MOVFF  APUNTADOR_CICLO_2,APUNTADOR_CICLO_3
        MOVFF  APUNTADOR_CICLO_1,APUNTADOR_CICLO_2
        MOVFF  APUNTADOR_CICLO_0,APUNTADOR_CICLO_1
        BRA     FINALIZA_Y

Y_POSITIVO
        MOVWFTEMP_Y
        CPFSGT TEMP_X                       ;SI TEMP_X > TEMP_Y  =>  SALTA UNA  INSTRUCCION
        BRA     Y2_ES_MAYOR                 ;SALTA SI Y2 ES EL MOTOR MAS BAJO
        MOVLWERROR_ACTUAL_Y2                ;Y2 ES EL SEGUNDO MOTOR MAS BAJO
        MOVWFAPUNTADOR_CICLO_2
        MOVLWERROR_ACTUAL_Y1
        MOVWFAPUNTADOR_CICLO_4
        BRA     FINALIZA_Y

Y2_ES_MAYOR                                 ;Y2 ES EL MOTOR MAS BAJO
        MOVLWERROR_ACTUAL_Y2
```

```
        MOVWF APUNTADOR_CICLO_0
        MOVLW ERROR_ACTUAL_Y1
        MOVWF APUNTADOR_CICLO_2
        MOVFF  APUNTADOR_CICLO_3,APUNTADOR_CICLO_4
        MOVFF  APUNTADOR_CICLO_2,APUNTADOR_CICLO_3
        MOVFF  APUNTADOR_CICLO_1,APUNTADOR_CICLO_2
        MOVFF  APUNTADOR_CICLO_0,APUNTADOR_CICLO_1
        BRA    FINALIZA_Y

FINALIZA_Y
        CLRF    CONTADOR_MAESTRO

        RETURN

;#############################################################################################
;********************** ACTUALIZA LAS ENTRADAS PARA COLOCARLAS EN LA RUTINA
DIFUSA************************
TRANSFERENCIA_DE_ENTRADA  ;SUBRUTINA

        FSR     FSR0,APUNTADOR_CICLO_1
        MOVF    CONTADOR_MAESTRO,W
        MOVFF   PLUSW0,FSR1L
        CLRF    FSR1H
        MOVFF   POSTINC1,ENTRADA_ACTUAL_1
        MOVFF   POSTINC1,ENTRADA_ACTUAL_2
        MOVFF   INDF1,INDICE_DE_SALIDA
        RETURN

;#############################################################################################
;********************** REALIZA LA DIFUSIÓN, EVALUACIÓN DE REGLAS Y DESDIFUSIÓN
************************

RUTINA_DIFUSA

         MOVLW 0X89
         MOVWFADCON0                    ;(FOSC/64), CANAL 1 (AN1), ENCENDEMOS EL
CONVERTIDOR

;****************************************** DIFUSIÓN ******************************************

DIFUSIÓN
        MOVLW0X02                       ;CAMBIAR POR NUMERO DE ENTRADAS
                                                CRISP A UTILIZAR
        MOVWF CONTADOR_ENTRADAS_CRISP   ;INICIALIZA CONTADOR DE ENTRADAS
                                                CRISP
        LFSR    FSR0,ENTRADAS_DIFUSAS   ;FSR0 APUNTA A ENTRADAS DIFUSAS
        LFSR    FSR1,INICIO_TABLA_CONJUNTOS_DIFUSOS     ;FSR1 APUNTA AL INICIO DE
                                                        LOS CONJUNTOS DIFUSOS
        LFSR    FSR2,ENTRADA_ACTUAL_1   ;FSR2 APUNTA A LA ENTRADA CRISP ACTUAL
SIG_ENT
        MOVLW NUMERO_CONJUNTOS_DIFUSOS_ENTRADA  ; # DE CONJUNTOS DE ENTRADA
                                                DIFUSOS-1
        MOVWF CONTADOR_CONJUNTOS_DIFUSOS        ;INICIALIZA CONTADOR PARA LOS
                                                CONJUNTOS DE ENTRADA
GRAD_LOOP
        CLRF VARIABLE_DIFUSA_A                  POR SI GRADO=0
        MOVLW 0X02
        MOVF PLUSW1,W                           ;W=PUNTO2
        SUBWF INDF2,W                           ;RESTA ENTRADA CRISP-PT2 (XI-PT2)
        BTFSS STATUS,C
        BRA SEG_1                               ;SI ES NEGATIVO SALTA A SEGMENTO 1
```

```
        BZ PEND_0

COMP
        MOVWF VARIABLE_DIFUSA_A          ;ALMACENA EL VALOR DE LA RESTA EN EL
                                         REGISTRO VARIABLE_DIFUSA_A
        MOVLW 0X03
        MOVF PLUSW1,W                    ;W=PENDIENTE2
        MULWF VARIABLE_DIFUSA_A          ;M2(XI-PT2)
        MOVF PRODH,W                     ;MUEVE LA PARTE ALTA DE LA
                                         MULTIPLICACION A W
        BZ      MAX_GRAD                 ;SALTA SI LA MULTIPLICACION NO GENERO
                                          UN NUMERO DE 16 BITS
        MOVLW 0X00                       ;SI LA MULTIPLICACION ES MAYOR A FF

ENTONCES GRADO=0
        BRA OBTEN_GRAD                   ;SALTA A OBTEN GRADO

MAX_GRAD
        MOVF PRODL,W                     ;MUEVE LA PARTE BAJA DE LA
                                         MULTIPLICACION A W
        BZ      OBTEN_GRAD1
        SUBLW 0XFF                       ;OBTIENE EL MAX [FF-(XI-PT2)*M2]
        BRA OBTEN_GRAD                   ;SALTA A OBTEN GRADO
SEG_1
        MOVWF VARIABLE_DIFUSA_A          ;MUEVE EL VALOR DE LA RESTA A
                                         VARIABLE_DIFUSA_A
        MOVLW 0X02
        MOVF PLUSW1,W                    ;W=PUNTO2
        ADDWF VARIABLE_DIFUSA_A,1        ;SUMA EL PUNTO2+VARIABLE_DIFUSA_A
                                         PARA RECONSTRUIR EL VALOR DE LA
                                         ENTRADA CRISP
        MOVF INDF1,W                     ;W=PUNTO1
        SUBWF VARIABLE_DIFUSA_A,W        ;RESTA ENTRADA CRISP-PT1 (XI-PT1)
        BTFSS STATUS,C
        BRA OBTEN_GRAD1                  ;SI ES NEGATIVO SALTA A OBTEN_GRAD1
        MOVWF VARIABLE_DIFUSA_A          ;MUEVE EL VALOR DE LA RESTA A
                                         VARIABLE_DIFUSA_A
        MOVLW 0X01
        MOVF PLUSW1,W                    ;W=PENDIENTE1
        BZ PEND_0                        ;CHECA SI LA PENDIENTE ES IGUAL A CERO,
                                         PARA EL CASO DEL PRIMER CONJUNTO
        MULWF VARIABLE_DIFUSA_A          ;M1(XI-PT1)
        MOVF PRODH,W                     ;MUEVE LA PARTE ALTA DE LA
                                         MULTIPLICACION A W
        BZ      NOT_Z                    ;SALTA A NOT_Z SI LA MULTIPLICACION NO
                                         GENERO UN NUMERO DE 16 BITS
        BNZ PEND_0                       ;PARA EL CASO DEL ULTIMO CONJUNTO

OBTEN_GRAD1
        MOVLW 0X00
        BRA OBTEN_GRAD
NOT_Z
        MOVF PRODL,W                     ;OBTIENE EL MIN [(XI-PT1)*M1]
        BRA OBTEN_GRAD                   ;SALTA OBTEN_GRAD

PEND_0
        MOVLW 0XFF                       ;SI PENDIENTE CERO ENTONCES GRADO
                                         IGUAL A FF
OBTEN_GRAD
        MOVWF POSTINC0                   ;SALVA EL VALOR DE LA ENTRADA DIFUSA
```

```
INCREMENTA APUNTADOR
        MOVF POSTINC1,1
        MOVF POSTINC1,1
        MOVF POSTINC1,1
        MOVF POSTINC1,1                 ;APUNTA AL SIGUIENTE CONJUNTO DIFUSO
        DECF CONTADOR_CONJUNTOS_DIFUSOS ;DECREMENTA CONTADOR DE CONJUNTOS
                                        DIFUSOS
        BNN GRAD_LOOP                   ;SALTA SI NO ES NEGATIVO
        MOVF POSTINC2,1                 ;INCREMENTA APUNTADOR DE ENTRADA
                                        CRISP
        DECF CONTADOR_ENTRADAS_CRISP    ;DECREMENTA CONTADOR DE ENTRADAS
                                        CRISP
        BNZ SIG_ENT                     ;SALTA SI EL VALOR ES DIFERENTE DE CERO

        BSF ADCON0,2                    ;INICIA CONVERSIÓN "X"

;****************************** EVALUACIÓN DE REGLAS ****************************************


        LFSR FSR0,SALIDA_DIFUSA         ;FSR0 APUNTA A SALIDAS DIFUSAS
        MOVLW NUMERO_DE_CONJUNTOS_DE_SALIDA ;# CONJUNTOS DE SALIDA A UTILIZAR
        MOVWF CONTADOR_SALIDAS          ;INICIALIZA CONTADOR PARA NUMERO DE
                                        SALIDAS DIFUSAS
BORRA
        CLRF POSTINC0                   BORRA LAS LOCALIDADES DE LAS SALIDAS
                                        DIFUSAS
        DECF CONTADOR_SALIDAS           ;DECREMENTA CONTADOR
        BNZ BORRA                       ;SALTA SI NO ES CERO
        LFSR FSR1,INICIO_TABLA_REGLAS   ;FSR1 APUNTA A REGLAS
RULE_TOP
        MOVLW 0XFF
        MOVWF VARIABLE_DIFUSA_A         ;RULE_TOP LDAA #$FF
SI
        MOVF INDF1,W                    CARGA BYTE DE REGLA
        BN ENTONCES                     ;SALTA SI EL BIT MAS SIGNIFICATIVO =1
        MOVF POSTINC1,1                 ;APUNTA AL SIGUIENTE BYTE DE REGLA
        LFSR FSR2,ENTRADAS_DIFUSAS      ;FSR2 APUNTA A ENTRADAS DIFUSAS
        MOVF PLUSW2,W                   ;CARGA EN W EL VALOR QUE APUNTA FSR2
                                        MAS UN OFFSET
        CPFSGT VARIABLE_DIFUSA_A        ;COMPARA SI VARIABLE_DIFUSA_A ES
                                        MAYOR AL VALOR DE W, SALTA SI CUMPLE
        BRA SI
        MOVWF VARIABLE_DIFUSA_A         ;ACTUALIZA EL VALOR DE
                                        VARIABLE_DIFUSA_A
        BNZ SI                          ;SALTA SI EL VALOR ES DIFERENTE DE CERO
BUSCA_ENT
        MOVF INDF1,W                    CARGA EL VALOR DE LA REGLA EN W
        BN BUSCA_SI                     ;BRINCA SI EL BIT MAS SIGNIFICATIVO =1
        MOVF POSTINC1,1                 ;APUNTA AL SIGUIENTE BYTE DE REGLA
        BRA BUSCA_ENT                   ;SALTA A BUSCA_ENTONCES
BUSCA_SI
        MOVF POSTINC1,1                 APUNTA AL SIGUIENTE BYTE DE REGLA
        MOVF INDF1,W                    CARGA EL VALOR DE LA REGLA EN W
        BNN RULE_TOP                    SALTA SI EL VALOR DEL BIT MAS
                                        SIGNIFICATIVO =0
        SUBLW 0XFF                      ;BUSCA FIN DE REGLAS "FF"
        BNZ BUSCA_SI                    SI NO ES CERO CONTINUA BUSCANDO POR
                                        EL FIN DE LAS REGLAS
        BRA DESDIFUSION                 ;BRINCA A DESDIFUSION CUANDO TERMINA
                                         DE EVALUAR LAS REGLAS
```

```
ENTONCES
        LFSR FSR0,SALIDA_DIFUSA                 ;FSR0 APUNTA A SALIDAS DIFUSAS
        ANDLW 0X7F                              ;OBTENEMOS EL VALOR DEL OFFSET
        MOVWF VARIABLE_DIFUSA_B                 ;GUARDA EL RESULTADO ANTERIOR EN LA
                                                LOCALIDAD VARIABLE_DIFUSA_B
        MOVF PLUSW0,W                           ;CARGA EN W EN VALOR DE LA SALIDA
                                                DIFUSA
        CPFSGT VARIABLE_DIFUSA_A                ;COMPARA SI ES MAYOR QUE EL VALOR
                                                GUARDADO EN VARIABLE_DIFUSA_A
        BRA NO_MAYOR                            ;BRINCA SI NO ES MAYOR
        MOVF VARIABLE_DIFUSA_B,W
        MOVFF VARIABLE_DIFUSA_A,PLUSW0          ;ACTUALIZA EL VALOR MAYOR EN LA
                                                SALIDA DIFUSA

NO_MAYOR
        MOVF POSTINC1,1                         APUNTA AL SIGUIENTE BYTE DE REGLA
        MOVF INDF1,W                            CARGA EL VALOR DE LA REGLA EN W
        BNN RULE_TOP                            SALTA SI EL VALOR DEL BIT MAS
                                                SIGNIFICATIVO =0

CHK_FIN
        SUBLW 0XFF                              ;BUSCA FIN DE REGLAS "FF"
        BNZ ENTONCES                            SI NO ES CERO BUSCA UN "ENTONCES"


;********************************** DESDIFUSIÓN ************************************************
DESDIFUSION
        MOVFF  ADRESH,SENSOR_X                  ;MOVEMOS EL VALOR DEL CONVERTIDOR A
                                                LA LOCALIDAD SENSOR_X
        MOVLW 0X91
        MOVWF ADCON0                            ;(FOSC/64), CANAL 2 (AN2), ENCENDEMOS EL
                                                CONVERTIDOR


        LFSR FSR1,INICIO_TABLA_SINGLETON        ;FSR1 APUNTA CONJUNTOS SINGLETON
        LFSR FSR0,SALIDA_DIFUSA                 ;FSR0 APUNTA A SALIDAS DIFUSAS
        MOVLW NUMERO_DE_CONJUNTOS_DE_SALIDA     ;(SINGLETON)
        MOVWF INDICE_SUMA                       ;INICIALIZA INDICE DE SUMA
        CLRF NUMERADOR_SUPERIOR
        CLRF NUMERADOR_MEDIO
        CLRF NUMERADOR_INFERIOR                 ;BORRA LOCALIDADES QUE SIRVEN PARA
                                                GUARDAR EL NUMERADOR
        CLRF DENOMINADOR_ALTO
        CLRF DENOMINADOR_BAJO                   ;BORRA LOCALIDADES QUE SIRVEN PARA
                                                GUARDAR EL DENOMINADOR
LAZO_SUMA
        MOVF INDF0,W                            ;CARGA EN W EL VALOR DE LA SALIDA
                                                DIFUSA
        ADDWF DENOMINADOR_BAJO,1                ;SUMA EL VALOR DE W CON EL
                                                DENOMINADOR BAJO
        MOVLW 0X00
        ADDWFC DENOMINADOR_ALTO,1               ;SUMA EL VALOR DEL DENOMINADOR ALTO
                                                MAS EL CARRY
        MOVF POSTINC0,W                         ;CARGA EL VALOR DE LA SALIDA DIFUSA EN
                                                W Y POSTINCREMENTA
        MOVWF VARIABLE_DIFUSA_A                 ;MUEVE EL VALOR DE W EN LA LOCALIDAD
                                                VARIABLE_DIFUSA_A
        MOVF POSTINC1,W                         ;CARGA EL VALOR DEL SINGLETON EN W
        MULWF VARIABLE_DIFUSA_A                 ;MULTIPLICA EL VALOR DEL SINGLETON Y
                                                EL VALOR DE LA SALIDA DIFUSA M(ZI)*ZI
        MOVF PRODL,W                            ;MUEVE LA PARTE BAJA DE LA
                                                MULTIPLICACION A W
```

```
        ADDWF NUMERADOR_INFERIOR,1          ;SUMA W CON LA PARTE BAJA DEL
                                            NUMERADOR
        MOVF PRODH,W                        ;MUEVE LA PARTE ALTA DE LA
                                            MULTIPLICACION A W
        ADDWFC NUMERADOR_MEDIO,1
        MOVLW 0X00
        ADDWFC NUMERADOR_SUPERIOR,1
        DECF INDICE_SUMA
        BNZ LAZO_SUMA
        CLRF VARIABLE_DIFUSA_A
        MOVLW 0X00
        CPFSEQ DENOMINADOR_ALTO
        BRA SIGUE
        CPFSEQ DENOMINADOR_BAJO
        BRA SIGUE
        BRA GUARDA_SAL

SIGUE
        BSF ADCON0,2                        ;INICIA CONVERSIÓN "Y"
        CALL DIVISION
GUARDA_SAL
        MOVFF COCIENTE,SALIDA_CRISP
        MOVFF  ADRESH,SENSOR_Y              ;MOVEMOS EL VALOR DEL CONVERTIDOR A
                                            LA LOCALIDAD SENSOR_Y
        RETURN

;#############################################################################################
;********************** ACTUALIZA LOS INDICES DE SALIDA PARA SELECCION DEL MOTOR
*************************
TRANSFERENCIA_DE_SALIDA                     ;SUBRUTINA
        LFSR FSR0,TABLA_PWM
        MOVF INDICE_DE_SALIDA,W
        MOVFF SALIDA_CRISP, PLUSW0
        RETURN

;#############################################################################################
;********************** ACTUALIZA PWM'S EN LA LOCALIDAD DE DONDE SE ENVIAN AL MOTOR
*************************
ACTUALIZACION_DE_PWMS       ;SUBRUTINA
        MOVFF PWM_0,PWM_INTERRUP_3          ;B3=X1
        MOVFF PWM_1,CCPR1L                  ;B2=X2
        MOVFF PWM_2,PWM_INTERRUP_1          ;B0=Y1
        MOVFF PWM_3,CCPR2L                  ;B1=Y2
        RETURN

;#############################################################################################
;********************** REALIZA LA DIVISIÓN PARA CALCULAR EL VALOR DE SALIDA DIFUSO
*************************

;*************************DIVISION*********************************************

;MULTIPLICACION DEL DENOMINADOR POR 80H
DIVISION
        BCF     STATUS,C
        MOVLW0X80
        MULWF DENOMINADOR_BAJO
        MOVFF  PRODL,PRERESULTADO_DIV_0
        MOVFF  PRODH,PRERESULTADO_DIV_1
        MULWF DENOMINADOR_ALTO
        MOVF    PRODL,W
        ADDWF PRERESULTADO_DIV_1
```

```
        MOVF    PRODH,W
        ADDWFC          PRERESULTADO_DIV_2

;CASO PARTICULAR DE LA DIVISION: COCIENTE=FF
        BCF STATUS,C
        RLCF PRERESULTADO_DIV_0
        RLCF PRERESULTADO_DIV_1
        RLCF PRERESULTADO_DIV_2
        CALL    RESTA
        BTFSS   CARRY_DIV,00                    ;SALTA SI LA RESTA FUE NEGATIVA
        BRA     EFEEFE                          ;SI POSITIVA, COCIENTE = FF
        CALL    CORRIMIENTO                     ;SI NEGATIVA, PRES= DENOMINADOR*80H

;EMPIEZA LA ITERACION DE RESTAS Y CORRIMIENTOS (DIVISION)
        MOVLW07H
        MOVWFCONTADOR_CICLOS_DE_DIVISION        ;CONTROLA EL NUMERO DE CICLOS EN LA
                                                DIVISION
        LFSR    FSR0,COCIENTE                   ;DIRECCIONAMIENTO INDIRECTO A
                                                "COCIENTE"
        MOVLW80H
        MOVWFINDF0                              ;RELLENANDO EL COCIENTE CON EL PRIMER
                                                VALOR

DIV
        CALL    RESTA
        BTFSC   CARRY_DIV,00                    ;SALTA SI LA RESTA FUE POSITIVA
        BRA     MITAD                           ;SI NEGATIVA, =(DENOMINADOR* Q1)/2,
                                                UNIVERSO NO ACTUALIZADO
                                                ;SI FUE POSITIVA ACTUALIZA EL NUEVO
                                                UNIVERSO
        MOVFF   RESIDUO_SUPERIOR,NUMERADOR_SUPERIOR
        MOVFF   RESIDUO_MEDIO,NUMERADOR_MEDIO
        MOVFF   RESIDUO_INFERIOR,NUMERADOR_INFERIOR
        MOVF    POSTINC0,W                      ;Q1 ENCONTRADO
        MOVWFINDF0                              ;AJUSTANDO UNA NUEVA LOCALIDAD PARA
                                                 EL NUEVO Q1

MITAD
        CALL    CORRIMIENTO                             ;  =(DENOMINADOR* Q1)/2
        RRNCF   INDF0           ;       Q1=Q1/2
        DECF    CONTADOR_CICLOS_DE_DIVISION     ;ESTE LOOP SOLO SE REPETIRA 8 VECES
        BZ              FINDIV                  ;SALTA DESPUES DE 8 CICLOS AL FINAL DE
                                                LA DIVISION
        BRA             DIV

;FINAL DE LA DIVISION: SE DETERMINA EL VALOR FINAL DEL COCIENTE
FINDIV
        MOVLWCOCIENTE
        CPFSGT FSR0L

AQUI
        RETURN                                  ;PUNTO FINAL DE LA DIVISION
        MOVF    POSTDEC0,W
        ADDWF INDF0,1
        BC              EFEEFE
        BRA             FINDIV

EFEEFE
        MOVLW0XFF
        MOVWFCOCIENTE
        BRA AQUÍ
```

```
;SUBRUTINA DE RESTA     =NUMERADOR - (DENOMINADOR* Q1)
RESTA
        CLRF    CARRY_DIV                               ;INDICADOR DE SIGNO DE RESTA
        MOVFF   PRERESULTADO_DIV_1,PRERESULTADO_DIV_11
        MOVFF   PRERESULTADO_DIV_2,PRERESULTADO_DIV_22
        MOVF    PRERESULTADO_DIV_0,W
        SUBWF   NUMERADOR_INFERIOR,W
        MOVWF   RESIDUO_INFERIOR
        BTFSC   STATUS,C
        BNN             RESTA_1                         ;SALTA SI NO HAY ACCARREO
        INCF    PRERESULTADO_DIV_11
        MOVLW   00H
        ADDWFC          PRERESULTADO_DIV_22,1

RESTA_1
        MOVF    PRERESULTADO_DIV_11,W
        SUBWF   NUMERADOR_MEDIO,W
        MOVWF   RESIDUO_MEDIO
        BTFSC   STATUS,C
        BNN             RESTA_2                         ;SALTA SI NO HAY ACCARREO
        BTFSS   STATUS,C                                ;CREO QUE CUANDO SE RESTA CERO SE
                                                        PRENDEN LAS BANDERAS DE NEGATIVO Y
                                                        CARRY AL MISMO TIEMPO
        INCF    PRERESULTADO_DIV_22

RESTA_2
        MOVF    PRERESULTADO_DIV_22,W
        SUBWF   NUMERADOR_SUPERIOR,W
        MOVWF   RESIDUO_SUPERIOR
        BTFSC   STATUS,C
        BNN             FINAL                   ;SALTA SI LA RESTA NO ES NEGATIVA
        MOVLW   01H  ;SI ES NEGATIVA LO INDICA COLOCANDO UN 1 EN EL ARCHIVO C
        MOVWF   CARRY_DIV

FINAL
        RETURN

;SUBRUTINA DE CORRIEMIENTO A LA DERECHA DE DENOMINADOR MULTIPLICADO, =PRES/2
CORRIMIENTO
        BCF STATUS,C
        RRCF PRERESULTADO_DIV_2
        RRCF PRERESULTADO_DIV_1
        RRCF PRERESULTADO_DIV_0
        RETURN

;*******************************************************************************
        END
```

Bibliografía

- Fuzzy Logic and Control. Software and Hardware Applications
  Jamshidi, Mohammad / Vadiee, Nader / Ross, Timothy J.
  Prentice Hall 1993

- Fuzzy Control of Industrial Systems. Theory and Applications
  Shaw, Ian
  Kluwer Academic Publishers 1998

- Fuzzy Logic with Engineering Applications
  Ross, Timothy J.
  McGraw Hill 1995

- Fuzzy Engineering
  Kosko, Bart
  Prentice Hall 1997

- FuzzyTech-MP. User's Guide
  Microchip Technology 1994

- Real Time systems and their programming languages
  Burns, Alan / Wellings, Andy
  Addison-Wesley Publishers Company 1992

- Real-Time and Embedded Systems
  Stankovic, John
  ACM Computing Surveys, Vol. 28, No. 1, March 1996

- Microelectronic Circuits
  Sedra, Adel
  Saunders College Publishing 1987

- Electronic Communication Techniques
  Young, Paul
  Mcmillan Publishing 1994

Consultas en internet

- VTI Technologies
  http://www.vti.fi/productsen/productsen_2.html

- Fuzzy Logic Toolbox Technical Literature
  http://www.mathworks.com/products/fuzzylogic/technicalliterature.jsp

- FuzzyTECH
  http://www.fuzzytech.com

- Microchip Technology Inc. PICmicro Microcontrollers
  http://www.microchip.com