E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Fuente de alimentación Flyback con control en pico de corriente controlada por el dsPIC33FJ16GS502

Grado en Ingeniería
en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Autor: Cristian Vidondo Pérez

Director: Peio Gil Izco

Pamplona, 10 de Junio de 2019

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Índice

1. RESUMEN 4
2. INTRODUCCIÓN 5
3. JUSTIFICACIÓN Y OBJETIVOS 6
4. CONTEXTO TECNOLÓGICO 7
   4.1. CONVERTIDORES CC CONMUTADOS 7
      4.1.1. Convertidor Flyback 7
   4.2. CONTROL EN MODO CORRIENTE 8
      4.2.1. Introducción y fundamentos de la teoría de control. 8
      4.2.2. Control en modo corriente por pico de corriente modo de conducción continuo 10
      4.2.3. Oscilaciones cuando D>0,5. 12
      4.2.4. Modelado dinámico de los convertidores con control por pico de corriente. 15
      4.2.6. Diseño del regulador de la fuente. 19
      4.2.7. Criterios generales de diseño del regulador. 19
   4.3. LOS DSP 23
5. FLYBACK CONTROLADO CON EL DSPIC33FJ16GS502 25
   5.1. CARACTERÍSTICAS DEL DSPIC33FJ16GS502 25
      5.1.1. EL PWM 26
      5.1.2. REGISTROS ESPECÍFICOS USADOS EN LA GENERACIÓN PWM 28
      5.1.3. EL CONVERSOR A/D 29
      5.1.4. SINCRONIZACIÓN DEL ADC Y LA PWM 31
      5.1.5. REGISTROS DEL ADC ESPECIAL INTERÉS Y SU CONFIGURACIÓN 31
      5.1.6. EL COMPARADOR 33
   5.2. LA PLACA 16-BIT 28-PIN STARTER DEVELOPMENT BOARD 34
   5.3. EL FLYBACK A CONTROLAR 36
      5.3.1. CALCULO DE LOS PARÁMETROS DEL REGULADOR 39
   5.4. EL PROGRAMA DE CONTROL CON EL DSPIC33F16J502GS 43
      5.4.1. INTRODUCCIÓN 43
      5.4.2. main.c 44
      5.4.3. init.c 46
      5.4.4. El isr_asm.s 53

5.4.5. EL pid2.s 54
5.5. EL PROGRAMA CON COMUNICACIÓN SERIE 64
5.5.1. INTRODUCCIÓN 64
5.5.2. MODIFICACIONES EN EL INIT.C 65
5.5.2.1. FLYBACKReferenceRoutine 65
5.5.2.2. Delay_ms 66
5.5.3. MODIFICACIONES EN main.c 67
5.5.4. EL ARCHIVO rs232.c 67
5.5.4.1. U1RXInterrupt() 68
5.5.5. PIC18f2450 70
5.5.6. MODIFICACIÓN DE LA PLACA 72
5.5.7. EL PROGRAMA DEL PIC18F2450 74
5.5.8. El PROGRAMA EN C# 75
6. RESULTADOS 77
7. BIBLIOGRAFÍA Y REFERENCIAS 80

1. RESUMEN

El convertidor Flyback es de entre todas las topologías de convertidores CC conmutados con aislamiento la más usada debido a que la cuenta de componentes para su realización es la más corta de todas ellas.

El modelo dinámico de esta topología en el modo de conducción continuo viene caracterizado por un sistema de segundo orden que cuenta además con la presencia de un cero en el plano derecho. Este cero dificulta el diseño del regulador para conseguir los objetivos de estabilidad y respuesta transitoria que se le pueden llegar a exigir. Es por esto que para simplificar el diseño del regulador o incluso para hacer posible los objetivos de control se hace necesario recurrir a técnicas de control que se denominan de modo corriente. De entre ellas la que se va a usar en este proyecto es la llamada técnica de control por pico de corriente que básicamente consiste en sustituir la forma de onda en diente de sierra obtenida de un oscilador por la rampa de corriente que circula por el transistor de la fuente durante el encendido del mismo.

En la parte de control de la fuente toda la lógica de control PWM se puede hacer tanto de modo analógico, utilizando operacionales para realizar los reguladores y comparadores o a través de un procesador digital de señal donde se programa toda la lógica necesaria para realizar de modo efectivo el control por pico de corriente. Cada vez más se prefiere usar esta segunda opción debido a múltiples razones entre las que se pueden citar las siguientes: flexibilidad a la hora de cambiar las condiciones de trabajo, hacer interfaces de comunicación con el usuario para el ajuste de parámetros y precisión en la elección de los parámetros del regulador.

El objetivo del proyecto es hacer un Flyback de pequeña potencia que cumpla con las características que se han descrito con el objetivo de que posteriormente pueda ser usado como equipo de prácticas en asignaturas de electrónica de potencia relacionadas con el estudio de las fuentes de alimentación conmutadas utilizadas para proporcionar las tensiones continuas necesarias para el funcionamiento de los equipos electrónicos

2. INTRODUCCIÓN

En este TFG se diseña un convertidor CC Flyback de pequeña potencia controlado digitalmente por un dsPIC33FJ16GS502, con el objetivo de que pueda ser utilizado como circuito de referencia para las prácticas de diseño de convertidores electrónicos de energía en asignaturas relacionadas con la conversión de potencia.

El convertidor Flyback es un convertidor DC-DC conmutado muy usado en sistemas de alimentación de equipos electrónicos para aplicaciones de baja potencia debido a su bajo coste y sencillez. Su estructura es similar a la del convertidor Buck-boost pero cuenta con un transformador de aislamiento con alta inductancia entre las etapas de entrada y de salida.

La fuente trabaja con control por pico de corriente. Esto de algún modo complica el marco teórico, dado que implica trabajar con lazos múltiples lo que requiere incrementar el estudio de los aspectos de control de los convertidores: después de explicar el control con lazo de tensión obliga a modificar esta explicación para añadir el lazo de corriente y sus efectos. Sin embargo, es necesario realizarlo así en el sentido de que actualmente el control de todos los convertidores DCDC prácticos se hacen usando este doble lazo de control. Más aún en el caso de un convertidor de este tipo que en el modo de conducción continuo presenta un cero en el plano derecho.

En este diseño los reguladores se programarán en un procesador digital de señal y además de realizar estos programas para poder trabajar en tiempo real a esta relativamente alta frecuencia de tiempo de muestreo de 10 µs (determinada por la frecuencia de conmutación de 100 kHz del transistor), se pretende que la fuente controlada digitalmente se pueda programar directamente desde un ordenador con el que se comunicará vía USB pudiendo cambiar los coeficientes del regulador y su tensión de referencia.

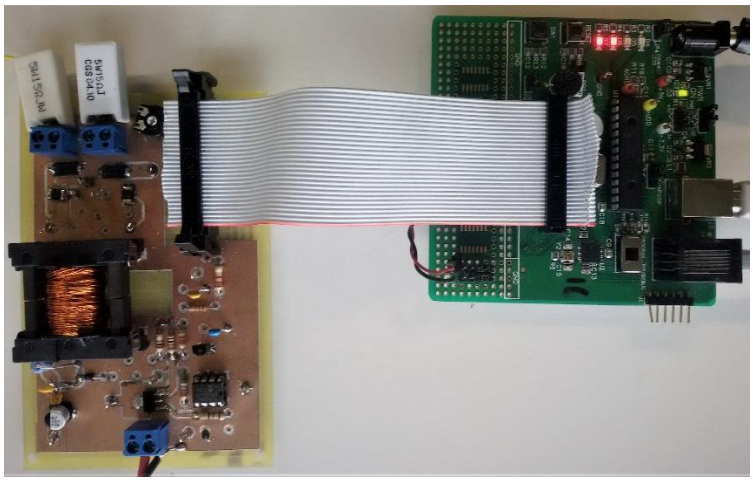


Figura 1: Flyback controlado por dsPIC33FJ16GS502

3. JUSTIFICACIÓN Y OBJETIVOS

Como ya se ha comentado anteriormente, el diseño y desarrollo de este convertidor se realiza dentro del marco de la docencia, es por ello que además de cumplir con las características de funcionamiento adecuado se debe incluir otros aspectos relacionados con la formación del alumnado.

El objetivo es que se incluyan guiones y documentación necesaria de modo que estos puedan ser utilizados por los alumnos dentro del contexto de la educación orientada a proyectos, que se basa en estos principios.

1.-El método consiste en la realización de un proyecto, normalmente de cierta envergadura y en grupo.

2.-El proyecto ha sido analizado previamente con el suficiente detalle como para asegurarse de que el alumno tiene todo lo necesario para resolverlo y que en su resolución desarrollará todas las destrezas que se desea.

El empleo de este convertidor en este contexto será capaz de aportar a los alumnos los siguientes aspectos básicos en la formación en el campo de la electrónica de potencia y también relacionados con el trabajo con microprocesadores en aplicaciones de sistemas embebidos con el valor añadido de que serán adquiridos con la experiencia práctica.

•Hardware electrónico digital: AD, temporizadores, comparadores…

•Conocimientos sobre conversión continua-continua, modelado dinámico de convertidores y teoría de control digital, incluyendo el aspecto práctico del cálculo en coma fija.

•Programación en lenguaje C (compilador C30) y ensamblador de los dsPIC.

•Criterios de diseño y selección de los componentes y dispositivos que forman parte del diseño

•Estudio y aplicación de la teoría de control automático adaptada a la aplicación de la conversión DCDC.

•Entorno de desarrollo MPLAB de Microchip.

•La colección de la familia de DSCs dsPIC30/33F.

•Manuales de las tarjetas de desarrollo: 16bit 28pin Development Board

4. CONTEXTO TECNOLÓGICO

4.1. CONVERTIDORES CC CONMUTADOS

Los Convertidores de continua a continua se emplean en una variedad de aplicaciones incluyendo fuentes de alimentación para cargadores de teléfonos móviles, equipamiento de Oficina, ordenadores portátiles, equipos de telecomunicaciones, etc. La entrada de un conversor de continua a continua es una tensión sin regular. El convertidor produce una salida de tensión regulada que tiene una magnitud y posiblemente polaridad que es diferente a la de la entrada.

La eficiencia siempre es un requisito ya que refrigerar un conversor de potencia ineficiente es difícil y caro. Un conversor cc ideal tiene una eficiencia del 100% pero en la práctica se obtienen eficiencias de entre un 70% y un 95%. Estas eficiencias se consiguen usando un modo conmutado o circuitos cortadores cuyos elementos disipan una cantidad de potencia despreciable.

4.1.1. Convertidor Flyback

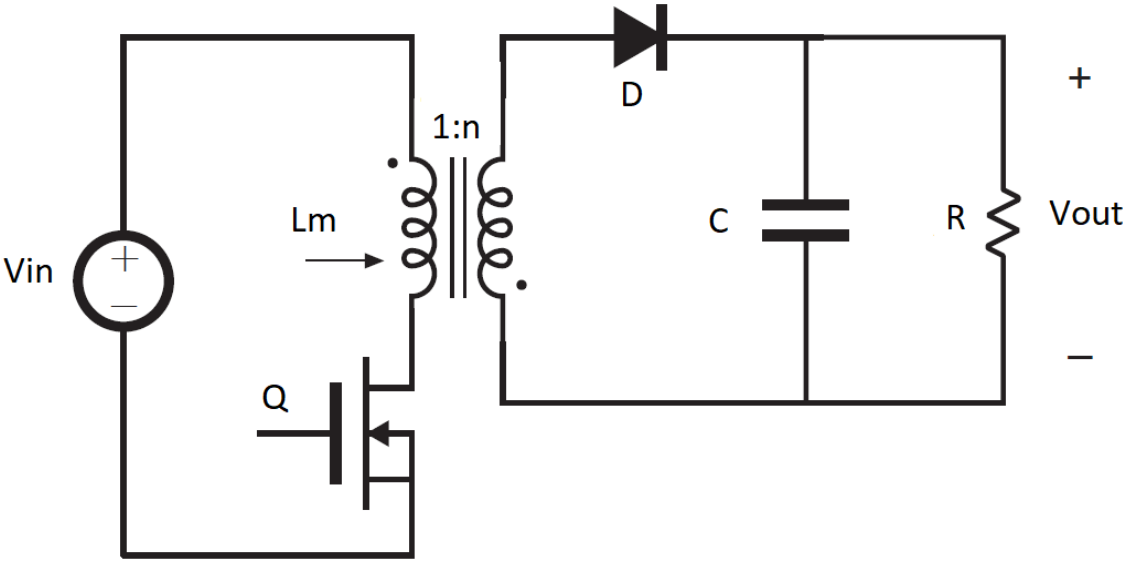


Figura 2 esquema del circuito de un convertidor Flyback

El convertidor Flyback es un convertidor CC aislado basado en el Buck-boost. Aunque el dispositivo magnético de 2 devanados se representa usando el mismo símbolo que un transformador, un nombre más descriptivo sería bobina de 2 devanados. Este dispositivo a veces también se llama transformador Flyback. A diferencia de lo que ocurre con un transformador ideal, en el transformador del Flyback la corriente no fluye simultáneamente en ambos devanados.

Cuando el transistor conduce el diodo está polarizado en inversa. En ese momento el devanado del primario funciona como una bobina conectada a la entrada VG. La energía se almacena en el campo magnético del transformador. Durante el Toff la corriente deja de circular por el primario. La corriente inducida en el devanado del secundario está en

ese momento polarizada directamente con el diodo y la energía almacenada en el campo magnético del transformador es transferida a la carga.

La ganancia de un Flyback es:

$$M(D) = \frac{V_{out}}{V_{in}} = n \cdot \frac{D}{1-D}$$

Ec. 1

Donde $n = \frac{N_1}{N_2}$ es la relación de vueltas entre primario y secundario.

4.2. CONTROL EN MODO CORRIENTE

4.2.1. Introducción y fundamentos de la teoría de control.

Dado el carácter progresivo en la introducción de la dificultad, normalmente en el estudio del control de los convertidores DCDC se estudia en primer lugar el control con solo lazo de control de la tensión de salida, sin embargo, debe quedar claro que el control más usado, especialmente en el MCC, es el llamado control en modo corriente. La esencia de este método de control es incorporar en los lazos de control, además de la medida de la tensión de salida, también una medida de la corriente en la bobina, con lo que en definitiva este control se enmarca dentro de la teoría general de control automático en el campo de los controles con lazos múltiples. Esta técnica es muy habitual en control automático, y en su forma más sencilla, queda representada con los lazos de control anidados que se muestran en la Figura 3.

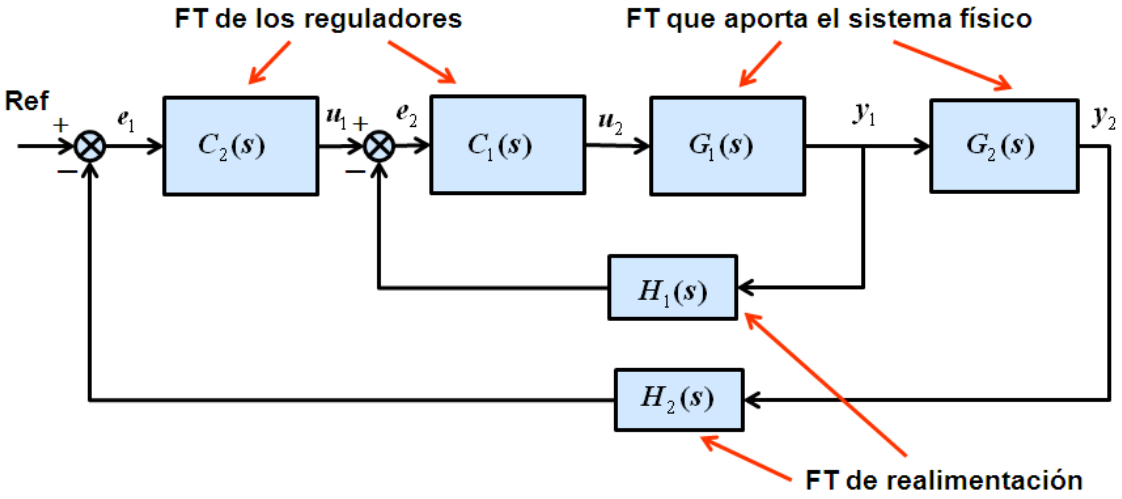


Figura 3. Diagrama de bloques de funciones de transferencia de un sistema general de control de lazos múltiples,

En ella los bloques $G_1(s)$ y $G_2(s)$ son los que contienen la dinámica del sistema, y normalmente, si esta viniera dada por una sola FT $G(s)$ que representara toda la dinámica del sistema sería un FT más complicada de controlar. El hecho de añadir el lazo de realimentación adicional a través de $H_1(s)$, permite repartir esa dinámica entre las dos FT $G_1(s)$ y $G_2(s)$ más simples. Además, para poder sacar mejor provecho de este esquema, la dinámica de las variables de lazo interno debe ser mucho más rápida que las del lazo externo. Esto se refleja en que el diagrama de Bode del módulo de la ganancia la FT $|G_1(j\omega)C_1(j\omega)H_1(j\omega)|$ es mucho mayor que la unidad:

$$|G_1(j\omega)C_1(j\omega)H_1(j\omega)| >> 1$$

Ec. 2

Esto implica que para el rango de frecuencias de interés de las señales del lazo externo, la FT en cadena cerrada del lazo interno verifica:

$$\left|\frac{y_1(j\omega)}{u_1(j\omega)}\right| = \frac{|G_1(j\omega)C_1(j\omega)|}{|1+G_1(j\omega)C_1(j\omega)H_1(j\omega)|} \approx \frac{1}{|H_1(j\omega)|}$$

Ec. 3

Donde el bloque de realimentación $|H_1(j\omega)|$ es muchas veces un valor constante, y en cualquier caso una función que controla el diseñador. Como resultado de esto, en el rango de frecuencias de interés para el diseño del controlador de la variable $y_2(s)$, el diagrama de bloques se reduce al de la Figura 4, para el que como se ha comentado va a resultar mucho más fácil diseñar el regulador del lazo de control.

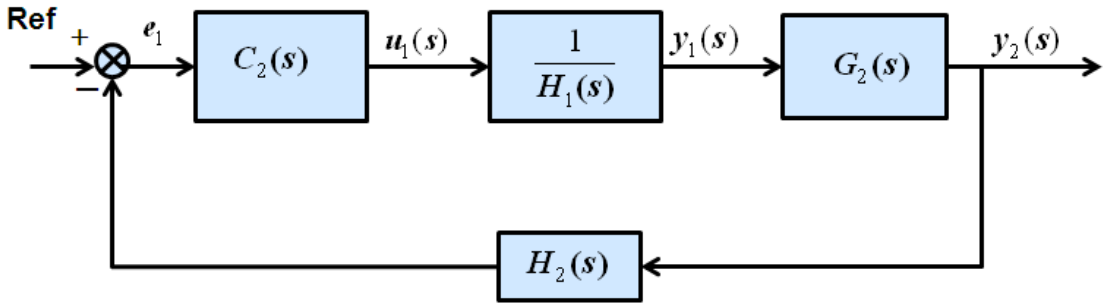


Figura 4. Diagrama de bloques de FT resultante de sustituir el lazo interno de control por su equivalente de cadena cerrada para el rango de frecuencias de interés de la variable $y_2(s)$.

Estos principios de control con lazos múltiples son en los que se basa también el control en modo corriente que se comenta en este apartado. Para el control de fuentes de alimentación DC-DC hay varios esquemas posibles, que por citarlos serían los siguientes:

- Corriente de Pico
- Corriente Promediada
- Corriente de Valle
- Tiempo de Conducción Constante y de Bloqueo Variable
- Tiempo de Bloqueo Constante y de Conducción Variable
- Histéresis constante

Los más usados son los dos primeros, entre otras razones porque son esquemas de control en los que el convertidor trabaja a frecuencia constante con control por PWM, mientras que para los otros la variable de control es la frecuencia. Esto supone un notable inconveniente, ya que el diseño viene condicionado por la frecuencia más baja de trabajo prevista, lo que hace que no pueda optimizarse el diseño de la fuente respecto al tamaño de los componentes magnéticos y capacitivos. En lo que sigue se va a hacer el estudio del control por pico de corriente que es el más usado y que es también el que se va a usar en el diseño del convertidor del TFG.

4.2.2. Control en modo corriente por pico de corriente modo de conducción continuo

La forma de trabajar de este modo de control de la tensión de salida se describe en lo siguiente. Módulo PWM fija pulsos a una frecuencia constante fs, que será la de conmutación del transistor. Cada vez que se produce un pulso se pone a uno la salida del driver MOSFET, que a través del circuito de gobierno cierra el transistor del convertidor. Cuando se cierra el transistor la corriente de la bobina circula por el transistor, y como se sabe, la corriente crece con pendiente aproximadamente constante: esta corriente escalada para dar voltios a través de una resistencia R_s. Cuando el valor de esta medida de corriente $R_s i_s(t)$ alcanza el valor de la tensión de control a la salida del regulador del lazo externo de control de la tensión, $v_c(t)$, se deben cortar los pulsos poniendo el transistor del convertidor en corte. Como en la modulación PWM, el transistor permanece abierto hasta que pasado el tiempo T desde el inicio del ciclo, no se produce un nuevo pulso hasta el siguiente periodo.

Conforme a lo descrito, la diferencia con la modulación PWM normal, consiste en que como señal portadora no se usa una rampa fija, sino que es la corriente que circula por el transistor la que hace esta función de rampa. De este modo, se consigue que también la corriente de la bobina y no sólo la tensión del condensador de filtro formen parte de las señales que se realimentan, lográndose así el control de lazos múltiples que se comentó en el apartado anterior introductorio al estudio de este tipo de control. De acuerdo con la descripción de su implementación, está claro que la consecución del esquema de control por lazos múltiples se hace de un modo muy específico:

aprovechando que la forma de onda de la corriente del transistor es una rampa. Con esto se consigue hacerlo lo más sencillo posible en cuanto al número de componentes usados, pero esto añade cierta complejidad a su análisis, ya que lo aleja de lo que pudiera ser un estudio sistemático típico de la teoría general de control automático.

Entre las ventajas del control en modo corriente por pico de corriente se pueden destacar las siguientes:

- La ventaja más importante es que simplifica el diseño del regulador del lazo de control de tensión, haciendo posible alcanzar un mejor comportamiento dinámico de la fuente. En los convertidores trabajando en el MCC esto se debe a que este control introduce desde el punto de vista de la FT un amortiguamiento, haciendo que los dos polos complejos de la planta pasen a ser reales, desplazando uno de ellos a una frecuencia próxima a la mitad de la frecuencia de conmutación. Esto permite que el regulador deba diseñarse para compensar el efecto de un solo polo.

- Implementando este tipo de control, se hace de modo inherente el control de la corriente máxima, sustituyendo así a la protección adicional de sobrecorriente o cortocircuito que hay incluir en control con sólo lazo de tensión. Además esta protección es muy efectiva ya que se hace ciclo a ciclo.

- También de modo implícito el control por pico de corriente lleva incluida la prealimentación para la compensación de las perturbaciones en la tensión de entrada. Esto es debido a que un cambio del valor de la tensión de entrada implica un cambio simultáneo de la pendiente de crecimiento de la corriente en la bobina, lo que inmediatamente acortará o alargará el ciclo de trabajo del transistor en cada pulso. De este modo, la variación de la tensión de entrada no tiene que afectar en primer lugar a la tensión de salida para luego poder ser corregida.

- Otro beneficio añadido es la reducción o eliminación de los problemas de saturación del transformador en los convertidores con aislamiento en medio puente, puente completo o push-pull. En el caso de estos convertidores, la introducción de una componente de continua en la teóricamente tensión alterna que se aplica al transformador hace que se provoque un proceso que lleva a aumentar el desequilibrio, que lleva a la saturación del transformador, dado que no hay ningún mecanismo de corrección. Para evitar esto, en los convertidores con control con un solo lazo de control se hace necesario conectar en serie con el devanado primario del transformador un condensador que filtra la tensión continua y que fuerza a que la tensión en el transformador sea alterna. Sin embargo, con el control por pico de corriente también de modo implícito al método, el mecanismo de corrección de una posible aplicación de componente de tensión continua se hace pulso a pulso. Efectivamente, si aparece una componente de tensión continua en el transformador, esto se traduce en la aparición de un incremento inmediato de la corriente por causa de esta

componente de continua. Pero a su vez, esto hace que el pico de corriente llegue antes a cortarse con la tensión de control, y por tanto a reducir la duración del ciclo de trabajo con respecto a si esta componente no deseada de corriente continua no existiera. Por último, lo que completa la explicación del método de corrección del control por pico de corriente, esta reducción del ciclo de trabajo hace que se corrija el desequilibrio anulando la tensión media aplicada de modo que la tensión vuelva a ser completamente alterna.

Por otra parte, antes de poder deducir los modelos dinámicos de este control en modo corriente, es conveniente poner de manifiesto una particularidad del mismo que consiste en que éste método de control es, si no se toma la correspondiente medida correctora, inherentemente inestable y por lo tanto oscila si la pendiente de decrecimiento de la corriente en la bobina es superior a la pendiente de crecimiento, o lo que es lo mismo cuando D>0,5. Esto se demuestra, principalmente de un modo gráfico, en el siguiente apartado.

4.2.3. Oscilaciones cuando D>0,5.

La Figura 5 muestra con detalle cómo se hace la PWM en un control de pico de corriente. La demostración del comportamiento de esta forma de determinar el ciclo de trabajo respecto a la estabilidad se hace suponiendo que se rompe el equilibrio, y analizando en qué condiciones el desequilibrio inicial se corrige, o si en cambio tiende a aumentar en cada ciclo de conmutación, lo que significa una condición de inestabilidad.

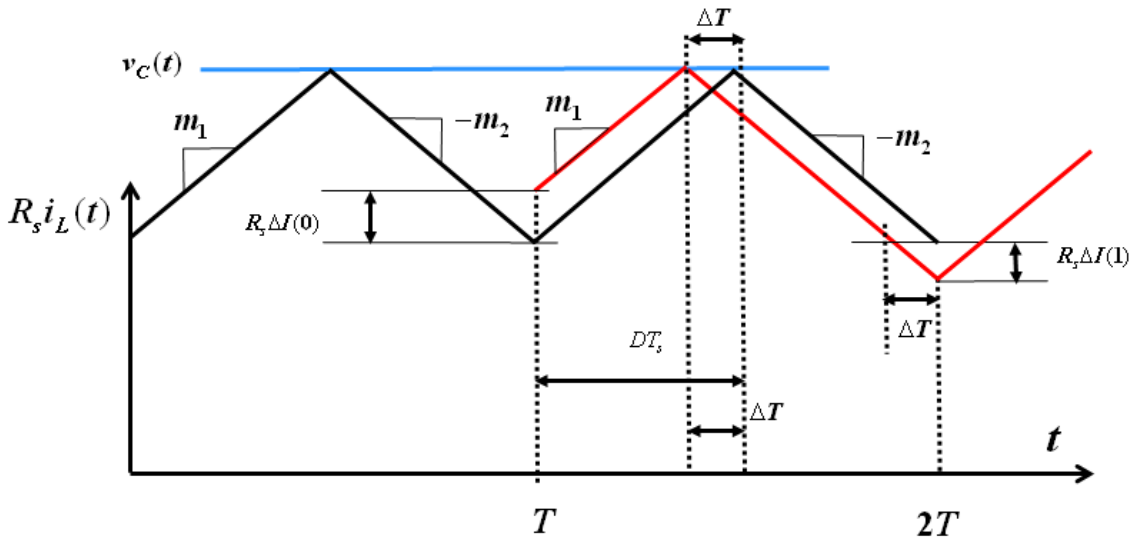


Figura 5. Efecto de una perturbación inicial $R_s\Delta I(0)$ en la corriente $R_s i_L(t)$.

Para ello, como se muestra en la referida Figura 5 en el ciclo T se introduce una perturbación en la corriente de valor $R_s\Delta I(0)$, esto hace que la corriente alcance antes

el valor de la señal de control reduciendo así el ciclo de trabajo. Esta perturbación se propaga como muestra la línea roja y cuándo el tiempo es 2T toma un valor igual a $R_s\Delta I(1)$. Estudiando la geometría de las rampas de corriente de la figura es fácil ver que se cumple la siguiente relación entre las pendientes y el incremento negativo del tiempo de encendido del transistor ΔT :

$$\frac{R_s\Delta I(0)}{\Delta T}=m_1,\frac{R_s\Delta I(1)}{\Delta T}=m_2 \quad \Rightarrow \Delta I(1)=\Delta I(0)\frac{m_2}{m_1}$$

Ec. 4

Razonando del mismo con los siguientes ciclos, se llega a la siguiente expresión para el ciclo n

$$\Delta I(n)=\Delta I(n-1)\frac{m_2}{m_1}=\Delta I(n-2)\left(\frac{m_2}{m_1}\right)^2=\ldots=\Delta I(0)\left(\frac{m_2}{m_1}\right)^n$$

Ec. 5

Por tanto, se deduce inmediatamente, que para anular la perturbación inicial la razón de las pendientes de crecimiento y decrecimiento tiene que ser menor que la unidad:

$$\frac{m_2}{m_1}<1$$

Ec. 6

Desigualdad, que usando el ciclo de trabajo es expresa de este modo:

$$\frac{m_2}{m_1}=\frac{D}{(1-D)}<1 \quad \Rightarrow \quad D<\frac{1}{2}$$

Ec. 7

Fácilmente se puede comprobar que esta expresión es válida para todas las topologías de convertidores. Por supuesto, como no es práctico condicionar el diseño a que el convertidor trabaje siempre con un ciclo de trabajo $D<0,5$, se recurre a implementar una solución que consiste en introducir una rampa de compensación, bien en la pendiente de la corriente que se mide para controlar, o bien en la señal de control $v_c(t)$. Cualquiera de las dos proporciona el mismo resultado respecto a la estabilidad de la fuente, por lo que el estudio de su efecto sobre la estabilidad se hará suponiendo que la rampa de compensación se aplica sobre la señal $v_c(t)$. Esta compensación y su efecto sobre la corrección en el desequilibrio de la corriente se han representado en la Figura 6. Examinando la geometría de la forma de onda de la corriente una vez perturbada,

como se ha hecho con ayuda de los triángulos en esta figura, se llega a las siguientes dos expresiones para $\Delta I(0)$ y $\Delta I(1)$:

$$\Delta I(0) = BD = BC + CD = m_c \Delta T(0) + m_1 \Delta T(0) = (m_c + m_1)\Delta T(0)$$

Ec. 8

$$\Delta I(1) = AB = AC - BC = m_2 \Delta T(0) - m_c \Delta T(0) = (m_2 - m_c)\Delta T(0)$$

Ec. 9

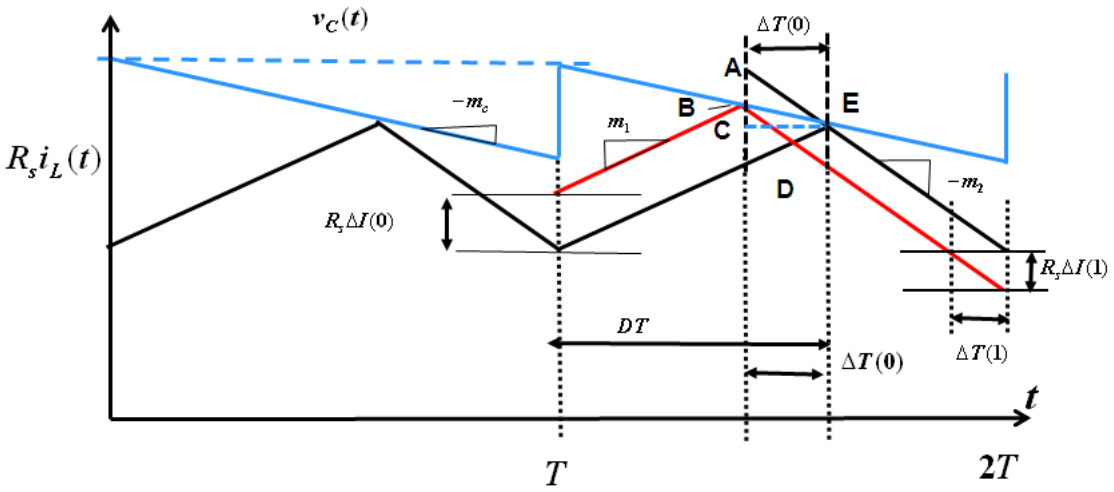


Figura 6. Formas de onda de la corriente en la bobina en régimen permanente y perturbado en presencia de una rampa artificial de compensación.

Nuevamente, haciendo el cociente entre $\Delta I(0)$ y $\Delta I(1)$, ahora que se ha introducido la rampa de compensación, se obtiene la siguiente expresión:

$$\Delta I(1) = \Delta I(0) \frac{m_2 - m_c}{m_c + m_1}$$

Ec. 10

Y como se hizo en el caso de no aplicar rampa de compensación en la Ec. 5, al cabo de n ciclos, la expresión de la perturbación es:

$$\Delta I(n) = \Delta I(n-1) \frac{m_2 - m_c}{m_c + m_1} = \Delta I(n-2) \left(\frac{m_2 - m_c}{m_c + m_1} \right)^2 = \ldots = \Delta I(0) \left(\frac{m_2 - m_c}{m_c + m_1} \right)^n$$

Ec. 11

Por tanto, como antes, la perturbación tenderá a desaparecer si:

$$|\alpha| = \left| \frac{m_2 - m_c}{m_c + m_1} \right| < 1$$

Ec. 12

Se toma en valor absoluto porque si $m_c > m_2$ α se hará negativo, pero esto no tiene importancia respecto a la suposición de estabilidad. Se puede comprobar que eligiendo:

$$m_c = \frac{1}{2} m_2$$

Ec. 13

Entonces $|\alpha| < 1$ para todos los valores del ciclo de trabajo 0<D<1, lo que asegura la estabilidad en todas las condiciones. Además de este valor mínimo de m_c para asegurar la estabilidad, otra elección común es hacer:

$$m_c = m_2$$

Ec. 14

Con lo que $\alpha = 0$ y por tanto siempre vale lo mismo independientemente del valor del ciclo de trabajo. Con este valor se consigue que $\Delta I(n+1) = 0$ independientemente del valor de $\Delta I(n)$. Este tipo de corrección de la perturbación en un solo ciclo se conoce en la literatura de control con el nombre de "control de golpe muerto" traducido del inglés "dead beat control".

4.2.4. Modelado dinámico de los convertidores con control por pico de corriente.

Una vez puesta de manifiesto la necesidad de la compensación necesaria para poder hacer factible el control por pico de corriente, el siguiente paso es obtener el modelo dinámico, en este caso del convertidor Flyback, que será la base para diseñar el regulador del lazo de control con el que conseguir los objetivos de control. Para ello se procede del siguiente modo.

En primer lugar, se obtiene el modelo promedio del convertidor Flyback, que como se muestra en la Figura 7, a efectos de control coincide con la del convertidor reductor-elevador Una vez obtenido este circuito promediado, por ser no lineal hay que proceder a la linealización.

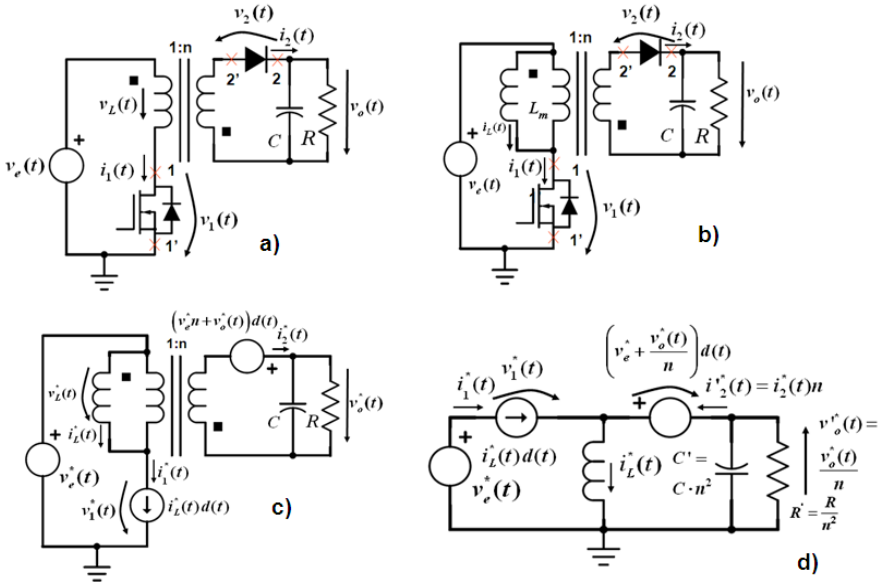


Figura 7. Obtención del modelo promediado del convertidor Flyback en el MCC.

4.2.5. Tensión promedio en la bobina del convertidor elevador:

$$v_L^*(t) = v_e^*(t)d(t) + (v_e^*(t) - v'^*_o(t))(1-d(t))$$

Ec. 15

Relación entre la tensión promedio y la corriente promedio en la bobina es a través de la derivada:

$$\frac{di_L^*(t)}{dt} = \frac{v_L^*}{L}$$

Ec. 16

Corriente promedio en el diodo:

$$i_D^* = i_L^*(1-d(t))$$

Ec. 17

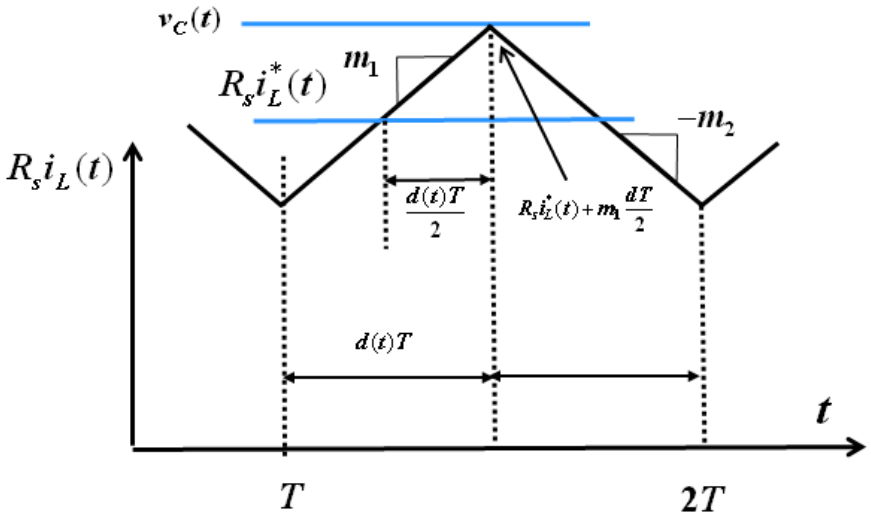


FIGURA 8 FORMAS DE ONDA EN LA ENTRADA DEL MODULADOR EN EL CONTROL POR PICO DE CORRIENTE

Como se deduce del examen de la Figura 8, la corriente de pico en el transistor verifica:

$$i_c(t) \approx i_L^* + d(t)\frac{v_e^*}{2L}d(t)T + (1-d(t))\frac{v'^*_o}{2L}(1-d(t))T + M_a d(t)T$$

Ec. 18

En el desarrollo hay que tener en cuenta igualdad entre la tensión de control y la corriente de pico de la bobina:

$$v_c(t) = R_s i_c(t) = R_s i_p(t)$$

Ec. 19

Linealización de la Ec. 18:

$$\hat{i}_c(t) \approx \hat{i}_L(t) + M_a T\hat{d}(t) + \frac{D^2 T}{2}\frac{\hat{v}_e}{L} - \frac{D'^2 T}{2}\frac{\hat{v}'_o}{L}$$

Ec. 20

Interesa despejar $\hat{d}(t)$ para usar la notación típica ligada a este tipo de control de corriente por pico de corriente:

$$\hat{d}(t) = F_m\left[\hat{i}_c(t) - \hat{i}_L(t) - F_g\hat{v}_e - F_v\hat{v}'_o\right]$$

Ec. 21

En la que para el caso del Flyback:

$$F_m = \frac{1}{M_a T} \quad F_g = \frac{D^2 T}{2L} \quad F_v = -\frac{D'^2 T}{2L}$$

Teniendo todo lo anterior en cuenta, y linealizando también el circuito promediado del convertidor Flyback representado en la Figura 7, se obtiene el circuito lineal representado en la Figura 8 que incorpora los bloques consecuencia de aplicar el control por pico de corriente.

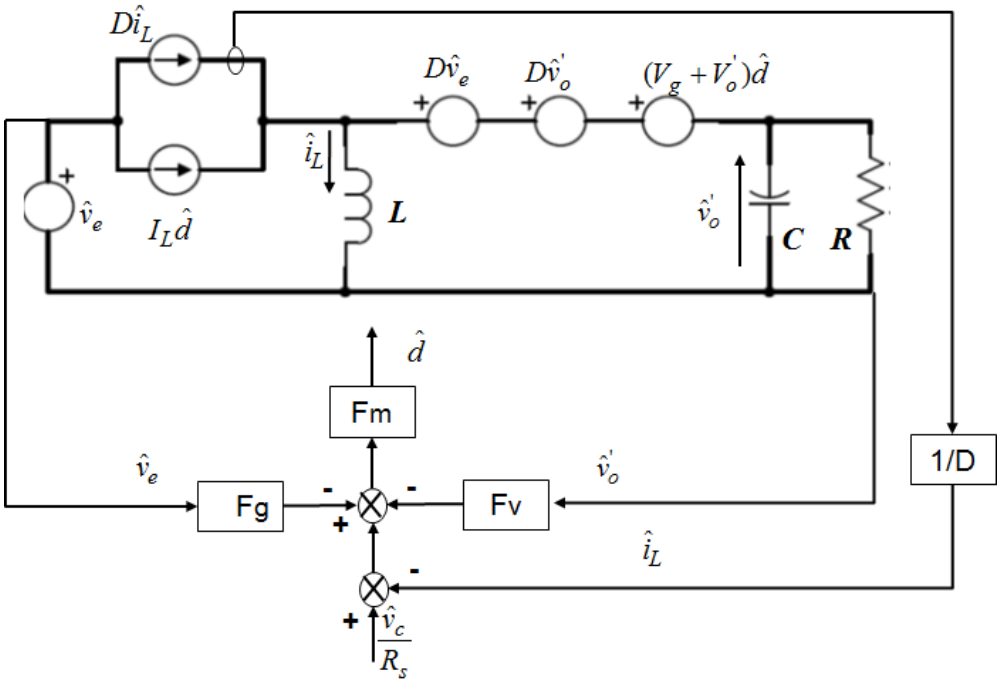


Figura 9. Modelo de pequeña señal incorporando los bloques de ganancia consecuencia de aplicar el control por pico de corriente.

Analizando el circuito anterior se obtiene la siguiente función de transferencia entre la tensión de salida y la tensión de control $\frac{\hat{v}'_o}{\hat{v}_c}$:

$$\frac{\hat{v}'_o}{\hat{v}_c} = G_{co} \frac{\left(1 - s\frac{DL}{D'^2 R}\right)}{1 + \frac{s}{Q_c \omega_c} + \left(\frac{s}{\omega_c}\right)^2} \frac{1}{R_s}$$

Ec. 22

Donde los coeficientes que aparecen en la ecuación anterior vienen dados por las siguientes ecuaciones:

$$G_{co} = \frac{V'_o}{DD'} \frac{F_m}{\left(1 + \frac{F_m V'_o (1+D)}{DD'^2 R} - \frac{F_m F_v V'_o}{DD'}\right)}$$

Ec. 23

$$\omega_c = \frac{D'}{\sqrt{LC}} \sqrt{1 + \frac{F_m V'_o (1+D)}{DD'^2 R} - \frac{F_m F_v V'_o}{DD'}}$$

Ec. 24

$$Q_c = D' R' \sqrt{\frac{C}{L}} \frac{\sqrt{1 + \frac{F_m V'_o (1+D)}{DD'^2 R} - \frac{F_m F_v V'_o}{DD'}}}{\left(1 + \frac{F_m V'_o RC}{DL} + \frac{F_m F_v V'_o}{D'}\right)}$$

Ec. 25

4.2.6. Diseño del regulador de la fuente.

4.2.7. Criterios generales de diseño del regulador.

Dadas las características de la planta, para conseguir los objetivos de control que se concretan en los siguientes aspectos:

- Error cero en continua, lo que exige un polo en el origen.
- Selección de la frecuencia de cruce con un margen de fase objetivo: normalmente 50º.

Una estructura de regulador adecuada es la siguiente:

$$R(s) = G \frac{(s+z)}{s \quad (s+p)}$$

Ec. 26

Que se conoce, especialmente en la literatura sobre el control de fuentes conmutadas, como regulador Tipo II. Este es el método de diseño del regulador que se seguirá, y para una mayor comprensión de la extensión del mismo se pide examinar lo que se expone en El diagrama de Bode correspondiente a esta FT se ha representado en la Figura 10:Bode de un regulador de tipo2. Puesto que en este regulador hay únicamente un cero, el retraso de fase mínimo que éste introduce, teniendo en cuenta que por efecto del polo en el origen se parte con una fase inicial de -90º, es de 0º. En la práctica, la separación entre el cero y el polo se determina para dar la elevación de fase deseada, y el par cero-polo se disponen simétricamente a los dos lados de la frecuencia de cruce

del lazo que se ha fijado como objetivo, lo que coloca el máximo de la elevación de fase coincidiendo precisamente con esta frecuencia de cruce.

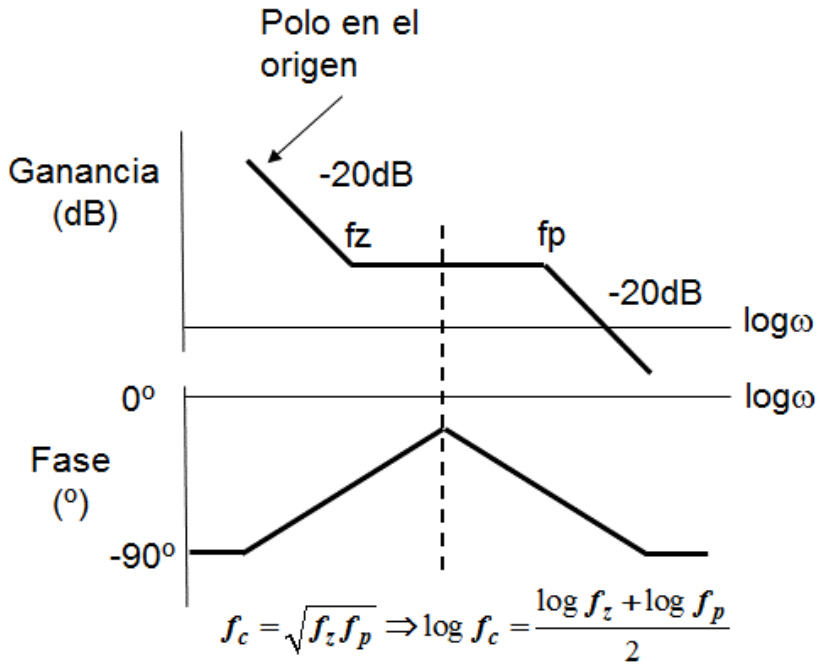


Figura 10:Bode de un regulador de tipo2

La implementación de este amplificador en el diseño que se ha realizado en este TFG se realiza con el método del factor K, en el que la frecuencia del polo y del cero se escogen, como se ha representado en Figura 10, para que la frecuencia de cruce se encuentre en la media geométrica de las frecuencias del polo y del cero:

$$f_c = \sqrt{f_z \cdot f_p} \Rightarrow f_c^2 = f_z \cdot f_p \Rightarrow \frac{f_c}{f_z} = \frac{f_p}{f_c} = K$$

Ec. 27

Por otra parte, en el dominio de la frecuencia, sustituyendo $s = j\omega$, la FT del regulador se expresa como:

$$R(j\omega) = G\frac{\omega_z}{\omega_p}\frac{(1+\frac{j\omega}{\omega_z})}{j\omega\left(1+\frac{j\omega}{\omega_p}\right)}$$

Ec. 28

El desfase introducido por esta función de transferencia a la pulsación de cruce $\omega = \omega_c$, que es el punto donde este alcanza el máximo valor, y que además se usará para determinar el margen de fase, viene dado por la siguiente ecuación:

$$\angle \frac{\hat{v}_c}{\hat{v}_{o2}}(\omega_c) = -90^o + \mathrm{tg}^{-1}\left(\frac{\omega_c}{\omega_z}\right) - \mathrm{tg}^{-1}\left(\frac{\omega_c}{\omega_p}\right) = -90^o + \mathrm{tg}^{-1}(K) - \mathrm{tg}^{-1}\left(\frac{1}{K}\right)$$

Ec. 29

Que nos dará el valor del desfase en función de la separación del par cero-polo. La ecuación anterior se ha representado en la Figura 1.

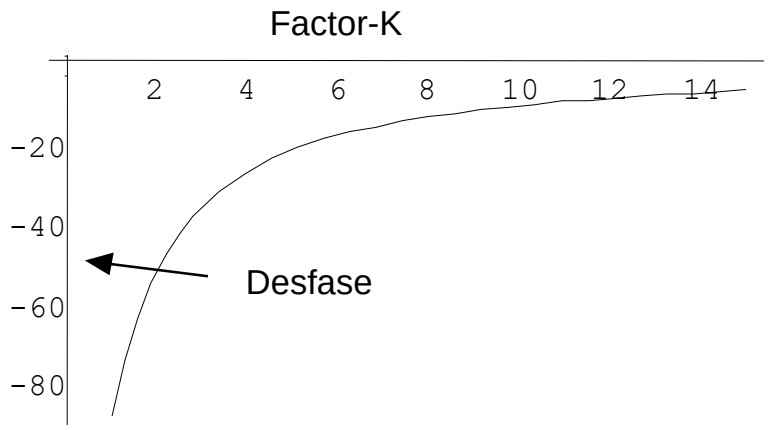


Figura 1. Desfase introducido por el regulador de estrategia 2 en función de la separación del par cero-polo.

Esta Figura 1 se puede usar para calcular el valor de K, y por lo tanto la relación de frecuencias del par polo cero. En cualquier caso, despejando K en la Ec. 29 la expresión del valor de K en función de la fase que introduce el regulador de estrategia 2 a la frecuencia de cruce es:

$$K = -\frac{1}{\tan\left(\frac{\angle R(j\omega_c)}{2}\right)}$$

Ec. 30

Para completar la explicación sobre la elección de los parámetros del regulador con este método del factor K, únicamente falta comentar que la pulsación de cruce del sistema completo sea ω_c, se logra escogiendo adecuadamente el valor del parámetro *G* del regulador.

4.3. LOS DSP

Los DSPs son microprocesadores especializados para el procesamiento de señales que aparecen en el mercado a principios de la década de los 80, y son la clave para el desarrollo de multitud de productos en todos los campos donde hace falta tratamiento de señal como las comunicaciones, navegación, automóvil, aviación, médicas, industriales, de tratamiento de la energía y otras.

Los procesadores digitales de señal (DSPs) son microprocesadores con las siguientes características:

- Están pensados para realizar operaciones matemáticas con señales en el mínimo tiempo posible, lo que incluye tanto operaciones individuales: suma, multiplicación, división, exponenciales y trigonométricas, como las compuestas de un conjunto de ellas en una sola instrucción. Por ejemplo, una instrucción típica es la instrucción MAC que reúne en una sola instrucción el siguiente conjunto de operaciones: sumar, multiplicar, sumar al resultado anterior y cargar nuevos valores para repetir la operación.
- Funcionar de modo determinista. El tiempo de ejecución de las partes críticas de los programas tienen un tiempo de ejecución predefinido exacto. Por ejemplo, en el caso del regulador que se diseñará en este TFG, este debe comenzar y acabar su ejecución en el tiempo prefijado. Esto no ocurre con los ordenadores personales dónde el sistema operativo lleva el control de la ejecución de números procesos que se ejecutan simultáneamente.
- Reprogramabilidad por software. La aplicación última de un DSP depende de lo que se programa sobre él. De todos modos, cada vez hay un mayor grado de especialización en función de la aplicación con el objetivo de optimizar en tiempos de ejecución y costes. Frente a este concepto de configuración por software se encuentra la de basarse para realizar la aplicación en el hardware diseñado específicamente para la función, concepto englobado en los llamados dispositivos específicamente para la aplicación (ASICs. Applied Specific Integrated Circuits).

Por lo tanto, los procesadores digitales de señal tienen que tener una gran capacidad para manejar un caudal grande de datos a gran velocidad con los que se hacen operaciones matemáticas en tiempo real. Esto condiciona numerosos aspectos de su diseño, por lo que no es objetivo de este texto hacer una descripción completa de las mismas. Aquí sólo se citarán dos aspectos básicos como son la arquitectura de la memoria en los DSP y el formato de los números que manejan, esto último con el fin de clasificar a los DSP en función de si manejan números en coma fija o en coma flotante.

El requisito de velocidad de transferencia de datos entre la CPU y la memoria hace que se use la llamada arquitectura Harvard en el diseño de los procesadores digitales de señal, en donde los buses de líneas de instrucciones y direcciones de programa y de datos están separadas, frente a la arquitectura Von Neumann, en el que los buses de direcciones y datos

es compartido por instrucciones y datos de memoria, lo que aumenta el número de ciclos de trabajo necesarios para ejecutar una instrucción. Esta última arquitectura Von-Neuman dispone de un único espacio de memoria en el que se almacena tanto el código a ejecutar como los datos del programa. Los accesos a este espacio de memoria se realizan a través de un único grupo de buses (bus de direcciones y bus de datos). Si bien se trata de una estructura muy simple, el procesador sólo puede realizar un acceso (de lectura o escritura) a memoria durante cada ciclo de instrucción.

5. FLYBACK CONTROLADO CON EL DSPIC33FJ16GS502

5.1. CARACTERÍSTICAS DEL DSPIC33FJ16GS502

El dspic33fj16gs502 es un DSP de 28 pines útil para aplicaciones de conversión de potencia que cuenta con 4 generadores PWM de alta velocidad con 2 salidas (H y L), 4 comparadores y un conversor, implementando muchas funcionalidades para la generación de pulsos utilizadas para encender y apagar los transistores de potencia en fuentes de alimentación.

Figura 11 dsPIC33FJ16GS502

Microchip proporciona multitud de documentos con toda la información necesaria para su programación y montaje entre ellos cabe destacar:

1.- dsPIC33FJ06GS101/X02 and dsPIC33FJ16GSX02/X04 datasheet: Un documento de 398 páginas editado en 2014 que describe completamente los dsPICs de esa familia, tanto a nivel eléctrico como de programación (registros, memoria, programación…). Es un documento que, aunque describe todos los registros del dsPIC, lo hace muy brevemente sin concretar su función y no incluye ejemplos, por ello, aunque es un documento imprescindible conviene completarlo con los siguientes.

2.- dsPIC33/PIC24 Family Reference Manual, High-Speed PWM Module: Editado el 07 de septiembre de 2017, este documento de 130 páginas es más reciente que el datasheet y a diferencia de este cuenta con ejemplos varios y una mejor descripción de los registros relacionados con los generadores PWM.

3.- dsPIC30F/33F Programmer's Reference Manual: Este documento de 364 páginas está hecho para servir de referencia para los desarrolladores del dsPIC33F. El manual describe detalladamente los sets de instrucciones en ensamblador y también proporciona información general para ayudar al usuario en el desarrollo de software para las familias de dsPIC30F y dsPIC33F. A lo largo de este manual también podemos encontrar una buena cantidad de código de ejemplo válido para cualquier dispositivo de la familia del 33F. Imprescindible para programar a bajo nivel.

4.- dsPIC33/PIC24 Family Reference Manual, UART: Es de 2014 al igual que el primer documento y cuenta con 56 páginas que describen el módulo UART y sus registros incluyendo instrucciones de como configurarlos para la transmisión o recepción. En el datasheet también se enumeran y describen brevemente pero no se explica cómo configurarlos. No es necesario para hacer funcionar el Flyback, pero lo será para permitir la comunicación entre el dsPIC33FJ16GS502 y el PIC18F2450.

5.1.1. EL PWM

El periférico para la gestión de la PWM es específico de este DSP y está especialmente diseñado para su uso en aplicaciones de conversión de energía eléctrica, AC/DC, DC/DC, corrección del factor de potencia (PFC) etc. Como se puede comprobar, principalmente en la primera de las referencias, el fabricante proporciona una amplia lista de característica y prestaciones, pero sólo se comentarán algunas de ellas, y además de entre estas sólo se profundizará más en aquellas que tienen una especial relevancia en la programación del funcionamiento de la PWM de cara a la aplicación del control de las fuentes conmutadas del proyecto de este TFG. Refiriéndonos en concreto al caso del dsPIC33FJ16GS502, este dispone de 4 generadores PWM con 8 salidas complementarias por parejas 4 H “HIGH” y 4 L “LOW”. La arquitectura del generador de PWM se puede ver en la Figura 12: Diagrama de bloques del PWM. Cada uno de estos generadores tiene su propio contador de tiempos, que en la documentación se le llama generador PWM. Dispone de distintos modos de funcionamiento, “complementario”, “independiente”, “push-pull”, “multifase” y otros más, pero los dos que han sido usados son el complementario para el reductor cuando se usa la posibilidad de rectificación síncrona y el independiente para el caso del convertidor elevador, y serán explicados más adelante. Una de las opciones más importantes de este módulo PWM, es el disparo sincronizado con las lecturas del ADC, ya que es especialmente importante para fijar el instante de muestro con el inicio de los pulsos de disparo de los transistores. La resolución mínima de la PWM es de 1.04ns, y es aplicable tanto al ancho del ciclo de trabajo, como para los tiempos muertos, los desplazamientos de fase etc.

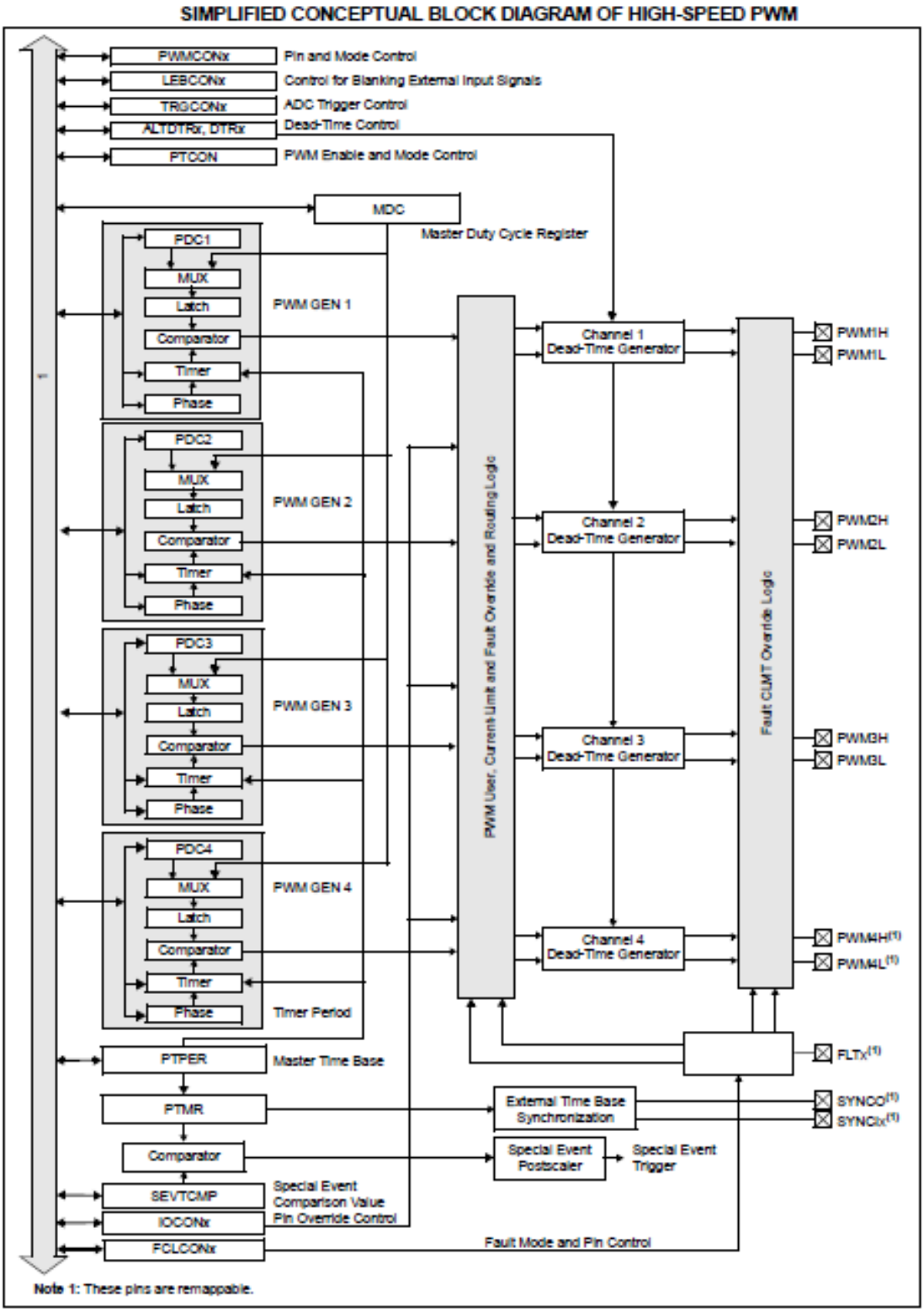


Figura 12: Diagrama de bloques del PWM

5.1.2. REGISTROS ESPECÍFICOS USADOS EN LA GENERACIÓN PWM

Para generar la PWM, el dsPIC33F usa una base de tiempos que va contando en sentido ascendente y que podemos conceptualmente asimilar a una forma de onda en rampa de subida. El valor de este contador se almacena en el registro PTMR que incrementa su cuenta directamente del reloj interno del DSP. En base a este, el periodo de la PWM se fija a través del registro PTPER, que es un registro que delimita el valor al cual la rampa o contador se detiene e inicia otra rampa. El PTPER es configurable por el programador, por lo que la PWM puede tener la frecuencia configurable y con ello definir la frecuencia de conmutación del convertidor. Además del PTPER para definir esta frecuencia de conmutación hay otros métodos más complejos especialmente pensados para convertidores multifase, que no se usarán. Para programar la PWM de un solo convertidor el programador fija un valor de 16 bits en el registro PTPER, que se compara con el valor de la cuenta del temporizador PTMR de modo que cuando el valor en PTMR iguala al programado en PTPER la rampa que genera la PWM finaliza, e inicia de nuevo otra rampa. El diagrama de bloques de la lógica de generación de la PWM se muestra en la Figura 12. Esta cuenta que genera el periodo se activa a través del bit PTEN del registro PTCON que a efectos prácticos supone la activación del toda la PWM.

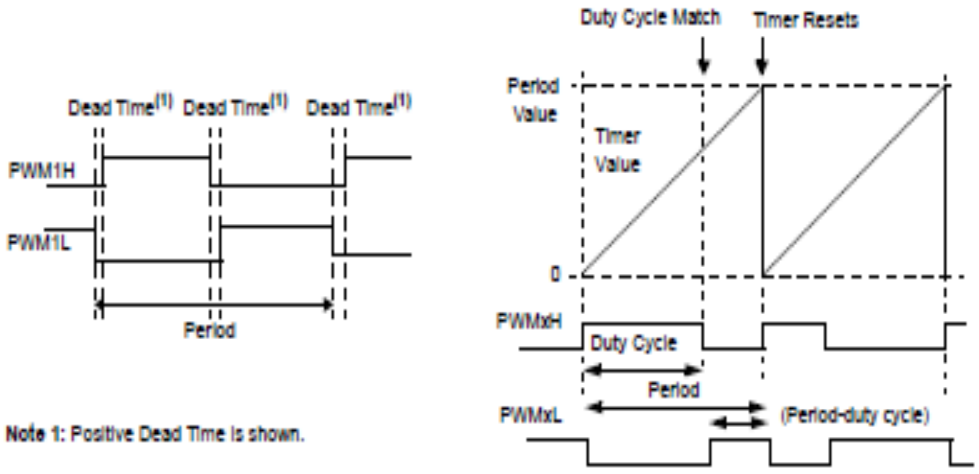


Figura 13: Ciclo de trabajo con la PWM programada en modo complementario.

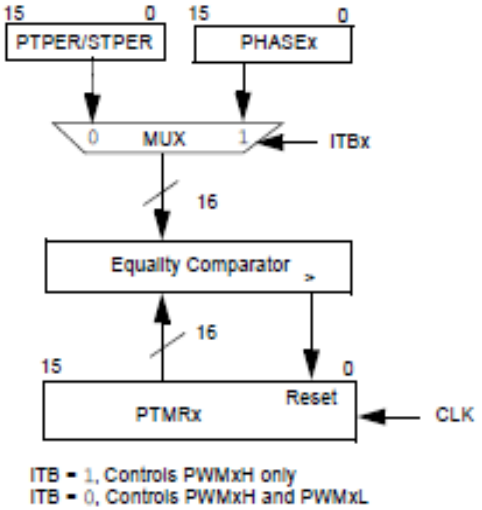


Figura 14: Diagrama de bloques en torno al comparador lógico usado para determinar el período de la señal PWM

Para terminar de definir la PWM se encuentra el conjunto de registros mostrados en la Figura 14: Diagrama de bloques en torno al comparador lógico usado para determinar el período de la señal PWMque permiten la determinación del ciclo de trabajo. Para ello se usa el registro MDC para hacer que los ciclos de trabajos de todos los generadores PWM trabajen de igual modo, o los registros PDCx, para fijar de forma independiente para cada uno de los 4 bloques PWM su modo de trabajo.

5.1.3. EL CONVERSOR A/D

Para las aplicaciones de la electrónica de potencia, es muy importante el tiempo o retardo que se produce entre la solicitud de la toma de medida y el resultado de la misma, y también es importante en qué instante se solicita la medida dentro de un ciclo de conmutación del convertidor. Para conseguir este objetivo, como con los dsPIC33F, es imprescindible disponer de la opción de que la conversión AD trabaje en sincronismo con la PWM y a gran velocidad. Respecto a la velocidad de conversión, el convertidor AD 10 bits de resolución del dsPIC33F realiza la conversión en 0.5 µs. Hay que tener en cuenta para valorar este dato, que trabajando a 300 kHz el período de la PWM para ejecutar todas las acciones de actualización del regulador es sólo de 3,3 µs.

El funcionamiento general del ADC está organizado por pares (AN0-AN1, AN2-AN3…) y cada uno de ellos puede iniciar la conversión mediante 16 fuentes distintas, es decir, que pueden programar hasta 16 motivos por los que se inicia la conversión. Estas fuentes que solicitan la lectura y conversión pueden ser por ejemplo las PWM, un temporizador que se desborda, por software etc. Esta capacidad resulta de especial utilidad ya que es la que permite sincronizar el ADC con la PWM.

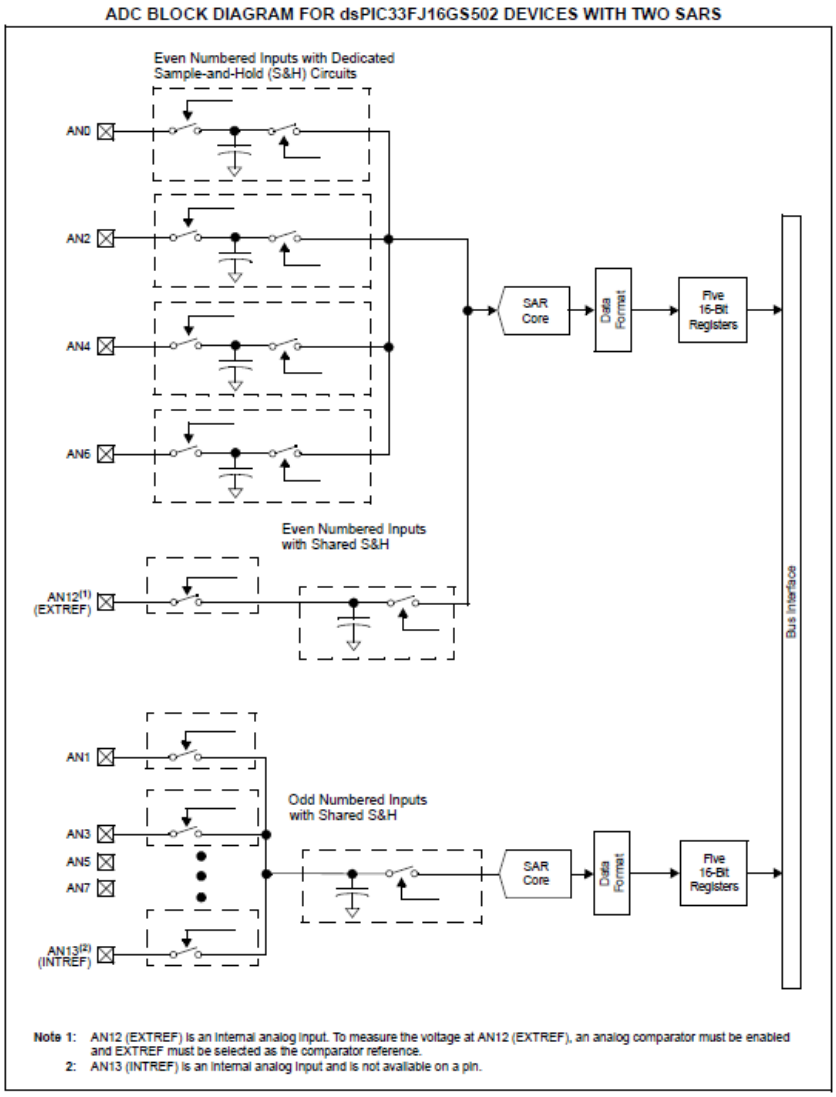


Figura 15 Diagrama de bloques del ADC del dsPIC33FJ16GS502

En cada instante solo se realiza la conversión de uno de los pares ya que sólo hay dos SAR (Successive Approximation Registers). El par sobre el que se hace la conversión es seleccionado por un multiplexor, pero la conversión entre el par y el impar se realiza a la vez. Si se explica el proceso por pasos queda de la siguiente manera:

- Primero se selecciona con el multiplexor un par para realizar la conversión. Esta entrada analógica se conecta con un circuito sample&hold (S&H), que toma el valor analógico de tensión mediante un condensador que se carga hasta el valor analógico de la entrada.

A continuación, se desconecta el circuito de (S&H) dejando el condensador en abierto y cargado.

- Finalmente otro multiplexor selecciona el circuito (S&H) que procede y se realiza la conversión a digital mediante un conversor de tipo SAR que mediante aproximaciones sucesivas, ajusta un valor digital a la lectura analógica que hay

en el condensador del S&H, guardándolo en el buffer correspondiente como último paso.

5.1.4. SINCRONIZACIÓN DEL ADC Y LA PWM

Como ya se ha comentado en repetidas ocasiones, las aplicaciones de electrónica de potencia con lazos de control de alta frecuencia requieren del funcionamiento sincronizado del módulo ADC y la PWM. De una forma simplificada se puede decir que el ADC lee el estado actual del convertidor y la PWM actúa sobre este para corregir el comportamiento dinámico o mantener el régimen permanente. Por lo tanto, cuanto menor sea el retardo entre la entrada al control (lectura ADC) y la actuación de este, (modificación de la PWM) mejor será el control en el sentido de que tendrá unas características más próximas al caso ideal en el que no existe retardo entre el sensado y la actuación. El módulo PWM contiene varios modos de actuar sobre la activación del ADC que se detallarán más adelante. Pero como idea general debe quedar claro que hay tres elementos necesarios sobre los que hay que actuar, estos son: la PWM, el ADC y la interrupción. En general el proceso sucederá así: la PWM fija en qué momento se produce el evento que provoca la interrupción por la que el convertidor ADC comienza la conversión. A su vez, el ADC se programará de modo que una vez que ha terminado la conversión y tiene el dato listo en su correspondiente "buffer", este evento disparará su propia interrupción de modo que el gestor de interrupciones interrumpa el flujo del programa para saltar a la interrupción solicitada por el ADC. En la rutina de gestión de esta interrupción se ejecutará los cálculos correspondientes al regulador con los que se obtiene el valor actualizado de la señal de control con la que se determina el ciclo de trabajo necesario para mantener constante la tensión de salida.

5.1.5. REGISTROS DEL ADC ESPECIAL INTERÉS Y SU CONFIGURACIÓN

Para seleccionar una de las 16 fuentes que provocan la lectura, conversión e interrupción del ADC, se deben configurar los 4 bits de TRGSRCx (Trigger Source) que están en el registro de 16 bits ADCPCx. Mediante estos 4 bits podemos seleccionar la fuente que reclama la conversión. Este disparo se puede hacer individual por software para un ADC, global por software para todos los ADC, por PWM *SpecialEvent*, por Timer1 ó 2 y por *PWM generator* primario, secundario y por corriente límite, consultar [10] para mayor información. Por otra parte, hay que tener presente que se está trabajando con fuentes conmutadas y por tanto hay transitorios a la frecuencia de conmutación que provocan cambios más o menos bruscos en las variables físicas a controlar. Por lo tanto, no es sólo importante la sincronización, sino también el instante en el que se hace la medida. Como ejemplo ilustrativo se muestran en la Figura 16 las formas de onda típicas que se tienen en un convertidor como resultado de la conmutación de los transistores. Durante los transitorios de conmutación las derivadas de tensión y corriente en diferentes partes del circuito son muy importantes y esto provoca sobretensiones transitorias durante las conmutaciones que distorsionan las formas de ondas de las señales que se están midiendo. Estos transitorios se superponen al valor más idealizado de las señales que se quieren medir, pero no deben aparecer en la medida realizada por el convertidor ADC puesto que daría lugar a valores erróneos en la medida de las

variables, lo que tiene un efecto muy negativo sobre el control. Por esta razón en los dsPIC33F se ha incluido la posibilidad a través del registro TRIGx de retrasar el inicio de la conversión respecto al instante de generación de la interrupción por parte del generador de PWM las decenas de nanosegundos que el diseñador considere necesarios hasta que ha pasado el transitorio, y por tanto se está ya en condiciones de medir el valor correcto de la señal. Como se muestra en la Figura 16 también existen posibilidades adicionales para controlar la sincronización del ADC con la señal PWM a través de los registros TRGDIV y TRGSTRT. Con el valor del registro TRGDIV se programa cada cuántos pulsos se hace la conversión y con TRGSTRT se fija, contando desde el primer pulso de la PWM, a partir de que numero de pulso empieza a trabajar el DAC.

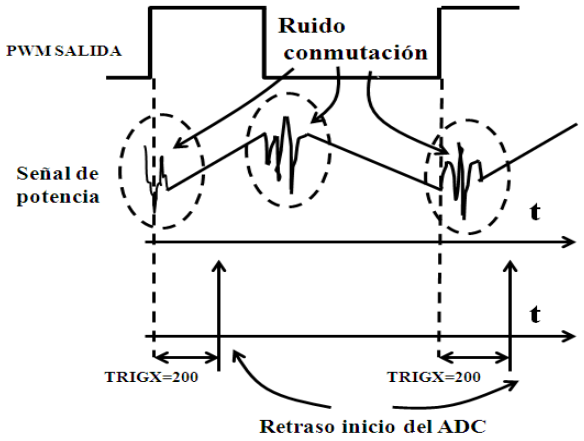


Figura 16 Justificación del retraso en el inicio de la conversión AD para evitar que el ruido introducido en el circuito por la conmutación no enmascare la medida de la señal.

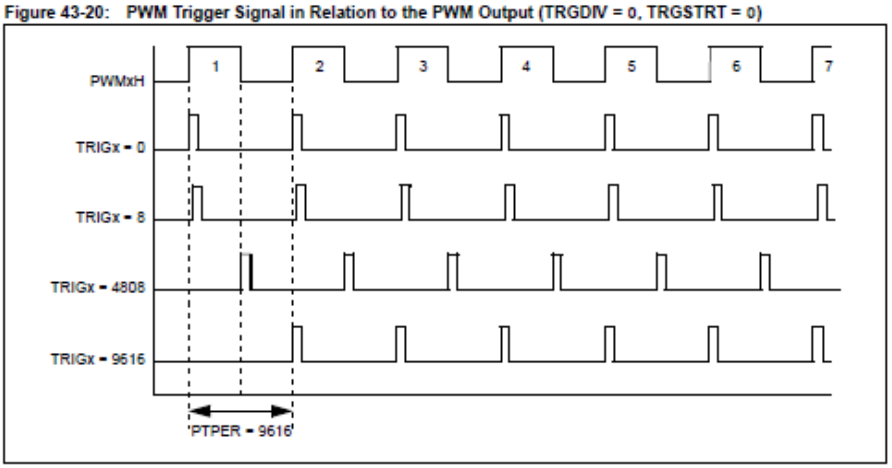


Figura 17 control del tiempo entre el flanco positivo del pulso de salida PWM y el de interrupción de inicio de la conversión del ADC en función TRGDIV=0, TRGSTRT=0 y TRIGx.

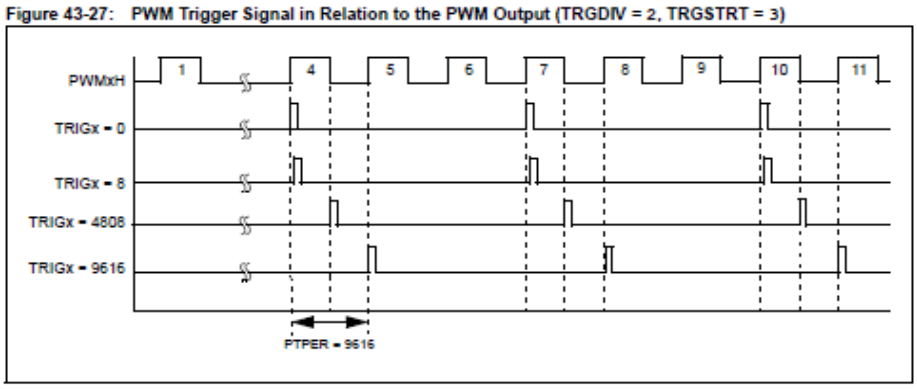


Figura 18 tiempo entre el flanco positivo del pulso de salida PWM y el de interrupción de inicio de la conversión del ADC en función TRGDIV=2, TRGSTRT=3 y TRIGx.

5.1.6. EL COMPARADOR

El Módulo de comparador de alta velocidad del dsPIC33F proporciona una manera de monitorizar corriente y voltaje en aplicaciones de conversión de potencia. El comparador analógico da la posibilidad al usuario de poder implementar el control en modo corriente en fuentes conmutadas. El dsPIC tiene cuatro comparadores de alta velocidad con conversores A/D dedicados a los que se puede proporcionar hasta una referencia de voltaje programable. Estos comparadores tienen también la capacidad de generar interrupciones.

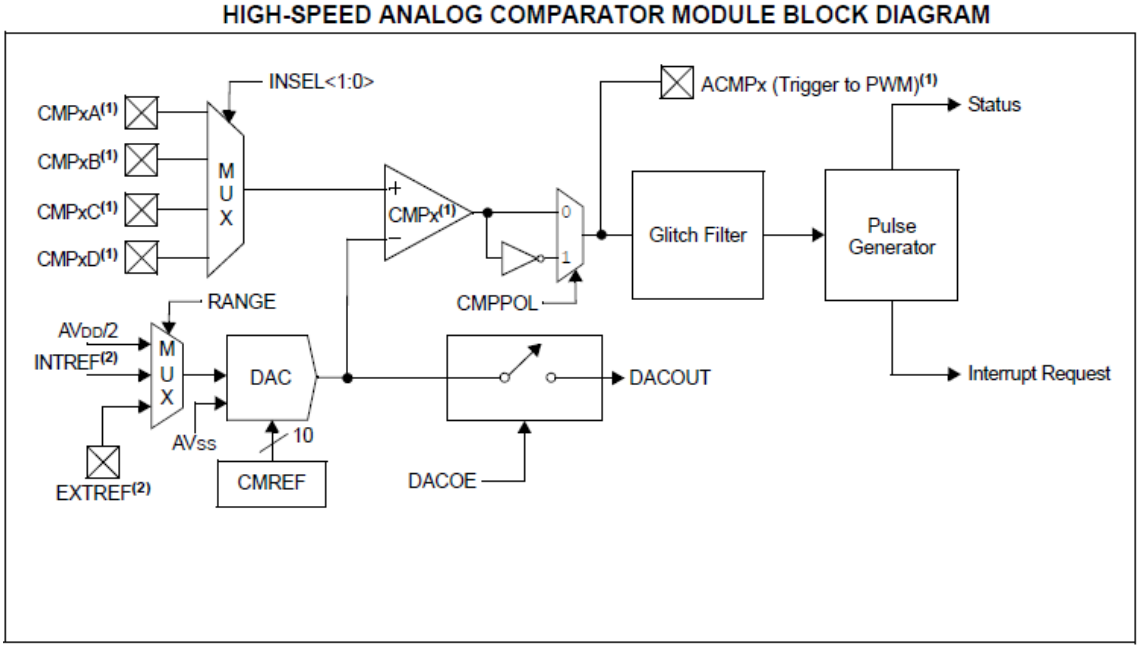


Figura 19 Diagrama de bloques del comparador

Tabla 1

CMPCONx: COMPARATOR CONTROL x REGISTER

R/W-0	U-0	R/W-0	r-0	r-0	r-0	r-0	R/W-0
CMPON	—	CMPSIDL	r	r	r	r	DACOE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	r-0	R/W-0	r-0	R/W-0	R/W-0
INSEL1	INSEL0	EXTREF	r	CMPSTAT	r	CMPPOL	RANGE
bit 7							bit 0

El registro CMPCON se usa para configurar la fuente de la referencia de tensión, el pin de entrada y la polaridad de salida.

Sus bits más importantes para el tema de este TFG son:

CMPON-Bit de modo de operación: puesto a 1 habilita el comparador, a 0 lo deshabilita

CMPSIDL- Bit de paro del comparador en reposo: si vale 1 el comparador se apagará en reposo y no volverá a funcionar.

DACOE-Bit de habilitar la salida del conversor D/A: puesto a 1 conecta la salida del DAC a DACOUT.

INSEL- Bits de selección de entrada: como son dos bits puede tomar valores de entre 0 y 3 para elegir la entrada A, B, C o D, en orden

EXTREF- Bit de referencia externa: 1 elige una referencia externa, 0 la interna que puede ser de hasta Vdd/2 en caso de que RANGE este puesto as 1.

5.2. LA PLACA 16-BIT 28-PIN STARTER DEVELOPMENT BOARD

Es una placa de desarrollo de Microchip para los microcontroladores de 16 bits PIC24 y los dsPIC DSC. Es una buena herramienta que ayuda a evaluar estos DSCs y microcontroladores. Tiene un potenciómetro conectado a la entrada A5, un pulsador SW1 conectado a la entrada digital RB8 y cuenta con 4 leds de estado conectados a los pines entre la pata 23 y la 27. También incluye una pequeña zona perforada para soldar componentes.

Se suministra con un CD con sus manuales y un programa de demostración que hace parpadear los leds de la placa mediante interrupciones de tiempo (Blinking) y que devuelve caracteres por el USB mediante UART.

En este trabajo se utiliza esta placa para conectar el convertidor Flyback mediante un cable plano de 20 pines con el dsPIC33f16JGS502 que se encargará de controlarlo. También se utilizan el pulsador y el potenciómetro para cambiar la referencia de tensión del Flyback y poder subir o bajar su tensión.

La placa de demostración incluye un pic18F2450 encargado de controlar el USB. Este tiene precargado un programa que, si previamente se ha instalado el controlador en el PC y se ha puesto el interruptor SW2 en posición “USB”, aparte de hacer lo necesario

para ser reconocida por el PC hace ecos al dsPIC funcionando como un interfaz USB-UART. En un apartado posterior veremos que, aunque la placa no esté diseñada para ello es posible reprogramar el pic18F2450 utilizando el ICD3 realizando algunas modificaciones simples en la placa para hacer cosas más avanzadas con ella.

Figura 20: Foto de la placa 16bits 28 pin Development Board

Para programar el dsPIC33f16JGS502 en este trabajo se ha utilizado el MPLAB ICD 3 In-Circuit Debugger. Se trata de un dispositivo que permite tanto depurar como programar multitud de microcontroladores de microchip. Se utiliza conectando su puerto RJ45 al puerto RJ45 de la placa 16-Bit 28-Pin Starter Development Board y su puerto USB al puerto USB del PC. El software que se ha utilizado es el MPLAB 8 aunque también sería posible realizar toda la parte del dsPIC33f16JGS502 de este proyecto mediante el software MPLABX sin ninguna modificación en el código, pero a la hora de programar el pic18f2450 se debe utilizar el software MPLAB 8 ya que las librerías periféricas de microchip necesarias para este microcontrolador no están disponibles para las versiones actuales de los compiladores y las versiones legacy que se han probado simplemente no funcionan. Tampoco sirve el entorno gráfico de configuración de periféricos que Microchip está desarrollando en los últimos años llamado Microchip Code Configurator incluido para trabajar con MPLABX en el que de momento no está incluidos ninguno de los dos microprocesadores que se van a programar.

5.3. EL FLYBACK A CONTROLAR

En este proyecto se pretende controlar un convertidor Flyback que estaba previamente diseñado y montado como variante digital de los convertidores Flyback analógicos que se usan en asignaturas de electrónica de potencia.

Sus especificaciones eran las siguientes:

Tensión de entrada.

- Tensión nominal: 20 VDC
- Tensión Máxima: 25 VDC
- Tensión mínima: 15 VDC.

Tensiones de salida. Dos salidas:

- Tensión de salida +5 VDC;
- Corriente de salida máxima: 1,2 A. P_{o1max}= 6 W;
- Corriente de salida mínima: 0,6 A. P_{o1max}= 3 W;
- Tensión de salida -5 VDC;
- Corriente de salida máxima: 0,8 A. P_{o2max}= 4,5 W.
- Corriente de salida mínima: 0,4 A. P_{o1max}= 2 W;

Frecuencia de conmutación 100kHz.

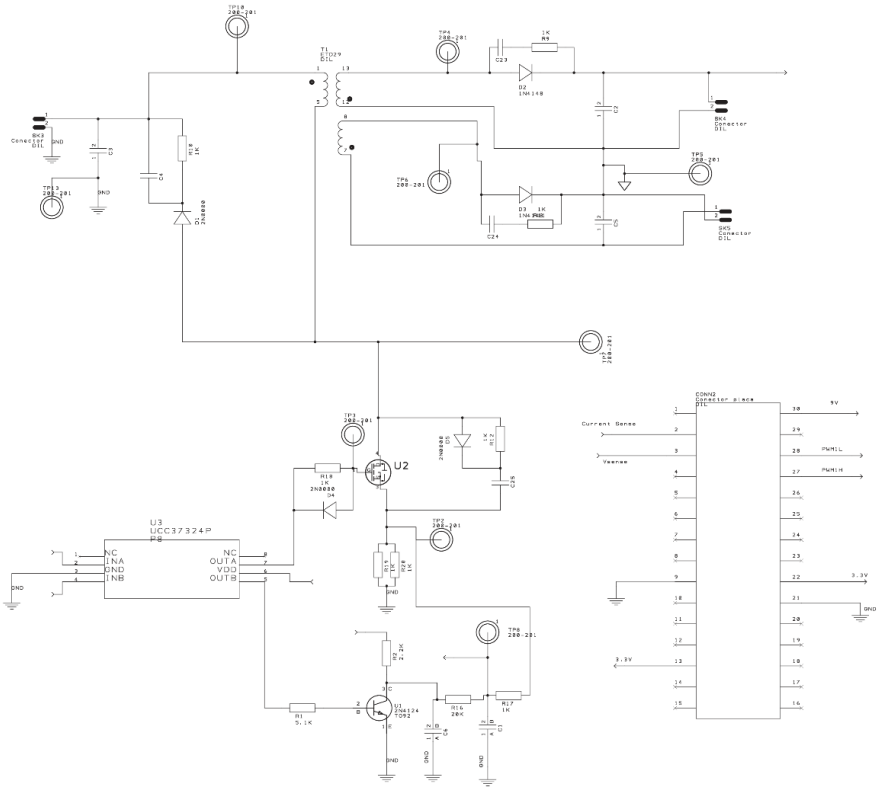


Figura 21 Esquema del Flyback que se va a controlar

Se trata de un circuito que cuenta con un conector de 20 pines pensado para ser conectado a la placa de microchip 16-Bit 28-Pin Starter Development Board. De estos pines, se utilizaban los correspondientes al canal 1 del PWM del dsPIC33FJ16GS502 que van conectados al driver MOSFET siendo el canal h el que controlara el MOSFET que da los pulsos al primario y el canal L el que controlara el transistor que introduce la rampa de compensación. También se utiliza el pin correspondiente a +9V para alimentar el driver y los pines correspondientes a las entradas analógicas A0 y A1 del microcontrolador que serán las que se utilicen para sensar la corriente que circula por el MOSFET y la tensión de salida respectivamente. Por supuesto también se utilizan las masas para qué la masa del Flyback este al mismo potencial que la de la placa 16-Bit 28-Pin Starter Development Board.

Como se puede ver en el esquema el transformador tiene dos devanados en su etapa de salida, de esta manera el convertidor proporcionará dos tensiones invertidas.

El diseño original del PCB era el siguiente

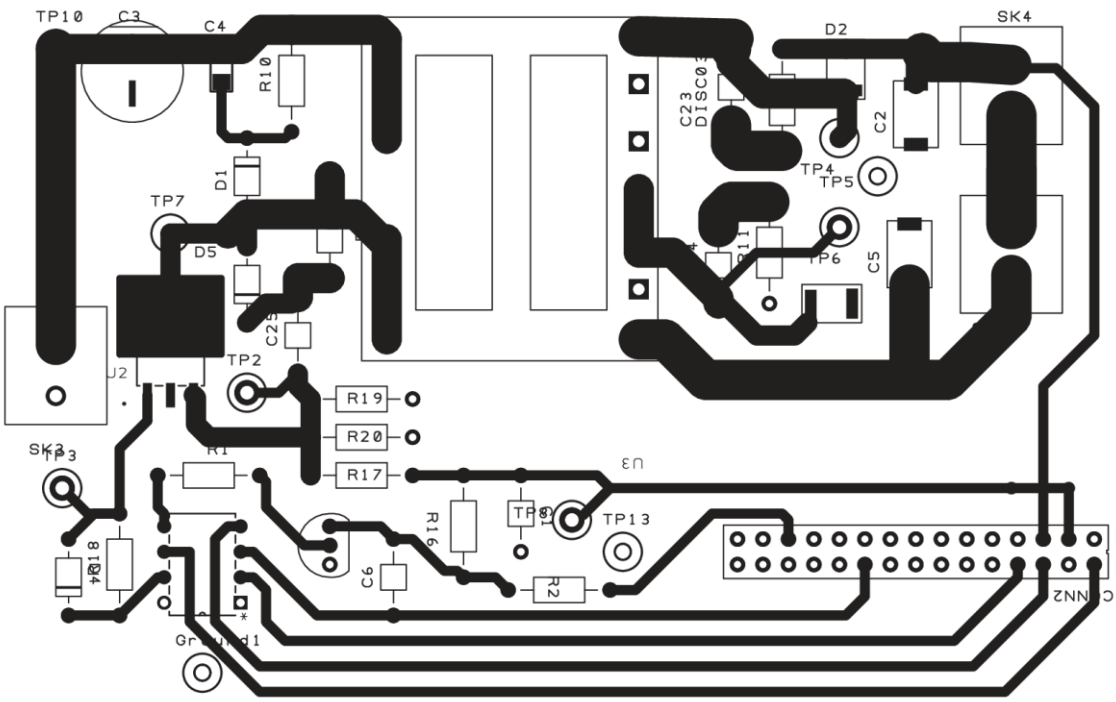


Figura 22:diseño del PCB

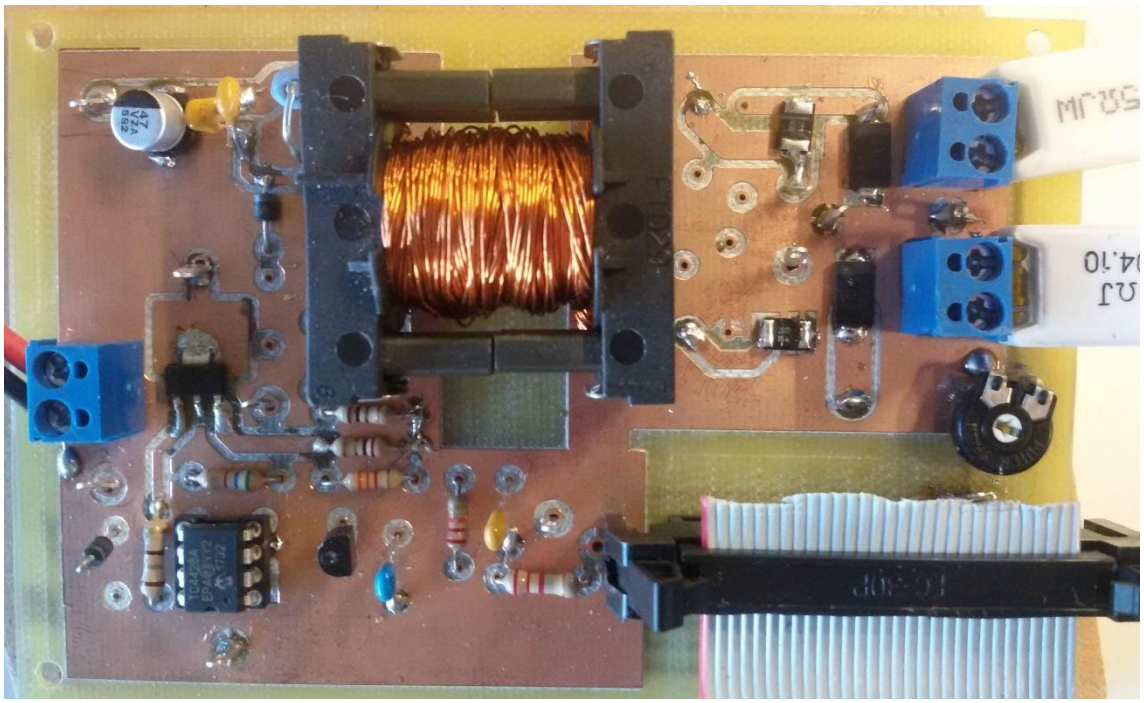


Figura 23: Circuito tras modificaciones

Como durante las pruebas la placa no funcionaba como era de esperar ya que se producían picos de tensión en el MOSFET durante la conmutación, con ciertos ciclos de trabajo se cortaban los pulsos inexplicablemente y al resetear la placa 16-Bit 28-Pin Starter Development Board se producían cortocircuitos porque dejaba a nivel alto ambos canales PWM se le realizaron las siguientes modificaciones:

1.- Se añadió un condensador en paralelo al condensador de snubber para que recortase bien los pulsos en el MOSFET ya que el que tenía no trabajaba correctamente. Con esto se consiguió que se recortaran adecuadamente los picos de tensión que se producían durante la conmutación

2.-Se cortaron las pistas correspondientes al canal 1 PWM y se hicieron coincidir en el conector con el canal 2. Sin mejorías.

3.-Se añadió un condensador de desacoplo al driver MOSFET ya que se observó que por alguna razón al trabajar con ciertos ciclos de trabajo se producían unos rizados en la alimentación de 9 voltios que impedían el correcto funcionamiento del driver resultando en ruidos y un extraño corte de los pulsos que le impedían regular adecuadamente. Con esto se consiguió un correcto funcionamiento del driver.

4.-Se cambió el driver UCC37324 de Texas Instruments por un TC1428CPA de Microchip. Este driver tiene su salida A negada lo que evita que durante los reinicios de la placa 16-Bit 28-Pin Starter Development Board se produzcan cortocircuitos.

5.-Se añadió un divisor de tensión con un potenciómetro al pin del lazo de tensión para calibrar la salida.

5.3.1. CALCULO DE LOS PARÁMETROS DEL REGULADOR

Siguiendo los principios del control en modo corriente explicados en 4.2 y aplicándolos al Flyback que se va a controlar se calcularán en este apartado los parámetros del regulador.

El circuito cuenta con las siguientes características:

- Pendiente de compensación

$$m_a = 22 \cdot 10^3 V/s$$

- Periodo de conmutación:

$$T = \frac{1}{100^3} = 10\mu s$$

- Resistencia en valores del primario:

$$R_p = 226\Omega$$

- Resistencia serie equivalente de los condensadores expresada en valores del primario:

$$ESR_p = 605m\Omega$$

- Resistencia de medida de la corriente en la fuente del MOSFET:

$$R_{corriente} = 0{,}5\Omega$$

- Inductancia:

$$L = 260\mu H$$

- Capacidad de salida en valores del primario:

$$Cp = 4{,}495\mu F$$

- Tensión de salida en valores del primario:

$$V_{op} = 27{,}5V$$

- Ciclo de trabajo:

$$D = 0{,}55 \qquad D' = 1 - D = 0{,}45$$

Con ello se procede a calcular la función de transferencia entre la tensión de salida y la tensión de control del regulador, empezando por los bloques de ganancia Fm, Fg y Fv:

$$Fm = \frac{1}{m_a \cdot T} = 4{,}54V^{-1}$$

Ec. 31

$$F_g = \frac{D^2 \cdot T}{2L} = 0{,}0058$$

Ec. 32

$$F_v = \frac{D'^2}{2L} = 0{,}0039$$

Ec. 33

$$G_{co} = \frac{V_{op}}{DD'} \frac{F_m}{1 + \frac{F_M V_{op}(1+D)}{DD'^2 R_p} - \frac{F_m F_v V_{op}}{DD'}} = 47.3594$$

Ec. 34

$$\omega_c = \frac{D'}{\sqrt{LC_p}} \cdot \sqrt{1 + \frac{F_m V_{op}(1+D)}{DD'^2 R} - \frac{F_m F_v V_{op}}{DD'}} = 4{,}2982 \cdot 10^{-4}$$

Ec. 35

$$Q_c = D' R_p \sqrt{\frac{C_p}{L}} \cdot \frac{\sqrt{1 + \frac{F_m V_{op}(1+D)}{DD'^2 R} - \frac{F_m F_v V_{op}}{DD'}}}{1 + \frac{F_m V_{op} RC}{DL} + \frac{F_m F_v V_{op}}{D'}} = 0{,}0492$$

Ec. 36

Con lo anterior se obtiene la siguiente función de transferencia entre la tensión de salida y la tensión de control. El factor ½ de la FT está porque el conversor D/A del registro CMDAC del comparador del dsPIC tiene referencia 1.6V mientras que el convertidor D/A que se usa para pasar a digital la Tensión de salida tiene 3.3V

$$\frac{\hat{v_o}}{\hat{v_c}} = \frac{1}{2} G_{co} \cdot \frac{\left(1 - s\frac{DL}{DD'^2 R_p}\right)}{1 + \frac{s}{Q_c \omega_c} + \left(\frac{s}{\omega_c}\right)^2} \cdot \frac{1}{R_{corriente}} =$$

$$= 0.131 \cdot \frac{-s^2 - 4{,}76 \cdot 10^4 s + 1{,}176 \cdot 10^{11} s}{s^2 + 0{,}874 \cdot 10^7 + 0{,}1847 \cdot 10^{11}}$$

Ec. 37

Cuyo diagrama de Bode se ha representado en la Figura 24

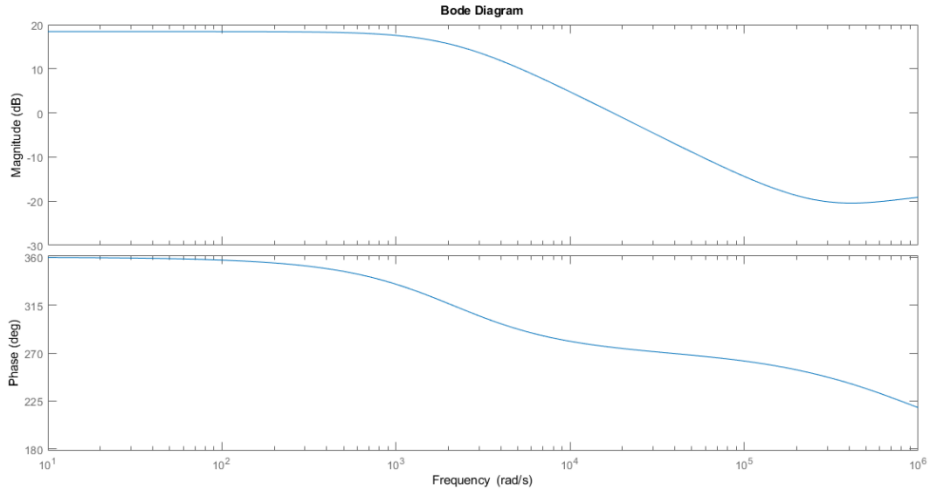


Figura 24: Diagrama de Bode de la función de transferencia del convertidor Flyback.

Como criterio de diseño establecemos una frecuencia de cruce f_d de 500Hz y un margen de fase de 50º, mirando en el bode anterior en una frecuencia de 3141rad/s tenemos una magnitud M_c de 13'4dB y una fase Fc de 303º, con ello podemos calcular la ganancia del controlador como:

$$G_{control} = \frac{1}{10^{\frac{13{,}38}{20}}} = 0{,}2142$$

Ec. 38

Ahora se procede a calcular la fase que hemos de introducir con el controlador a partir de la especificación del margen de fase

$$Fase_{control} = MargenFase - 180 - Fc = 50 - 180 - 303 = -433^{o}$$

Ec. 39

Ahora se procede a calcular el valor del factor K. La fórmula empleada está directamente despejada de la de la Ec. 29

$$K = -\frac{1}{tan\left(\frac{Fase_{control}}{2}\right)} = 1{,}333870417375217$$

Ec. 40

Fase que introduce el regulador

$$Fase_{reg} = \left(-\frac{\pi}{2} + atan(K) - atan\left(\frac{1}{K}\right)\right) \cdot \frac{180}{\pi} = -73{,}7176$$

Ec. 41

Margen de fase resultante

$$Margen_{FaseResultante} = Fase_{reg} + 180 + Fc = 410$$

Ec. 42

Una vez conocido el factor K podemos hallar la frecuencia del cero y del polo.

$$fz = \frac{fd}{K} = 374{,}849Hz$$

Ec. 43

$$f_p = f_d * K = 666{,}9352Hz$$

Ec. 44

Con la ganancia y las frecuencias del cero y del polo ya se puede hallar la función de transferencia del controlador

$$R(s) = G_{control} \cdot \frac{\sqrt{(Wd^2 + 2\pi fp^2)} \cdot Wd}{\sqrt{(Wd^2 + 2\pi fz^2)}} \cdot \frac{s + 2\pi fz}{(s + 2\pi fp) \cdot s}$$

Ec. 45

Cuyo bode es:

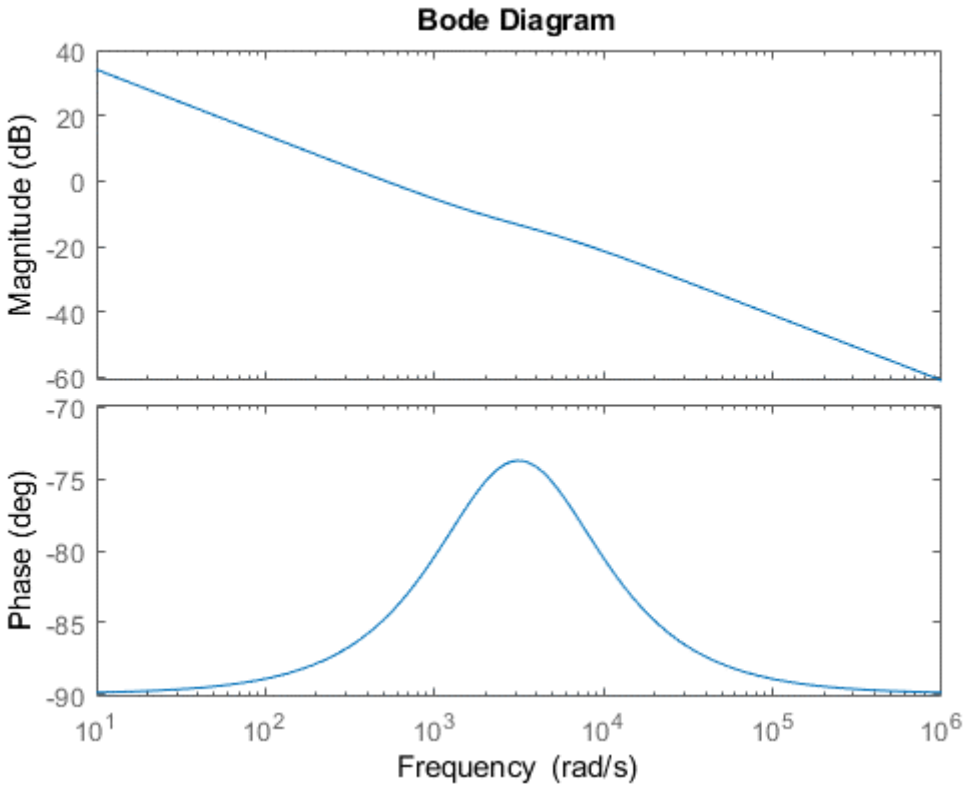


Figura 25

Discretizando R(s) con un periodo de muestreo de 10us se obtiene la función de transferencia a implementar de manera digital con el dsPIC33:

$$\boldsymbol{R(z) = \frac{0.004447\,z^2 + 0.0001035\,z - 0.004344}{z^2 - 1.959\,z + 0.959}}$$

Ec. 46

5.4. EL PROGRAMA DE CONTROL CON EL DSPIC33F16J502GS

5.4.1. INTRODUCCIÓN

En este apartado se explicarán el conjunto de subprogramas que forman parte del proyecto de MPLAB 8 que es capaz de controlar el convertidor Flyback de manera autónoma. Este proyecto de MPLAB sigue la misma estructura de subprogramas que se pueden encontrar en los ejemplos de la placa PICtail™ Plus Daughter Board, aunque esos ejemplos controlan convertidores Buck o Boost y utilizan control en modo tensión mientras que este proyecto controla un Flyback en modo pico de corriente.

El archivo workspace del proyecto se llama Flyback_TFG.mcw y los subprogramas están en tres carpetas. En la carpeta "src" están los subprogramas principales y en las carpetas "h" e "inc" las cabeceras e includes.

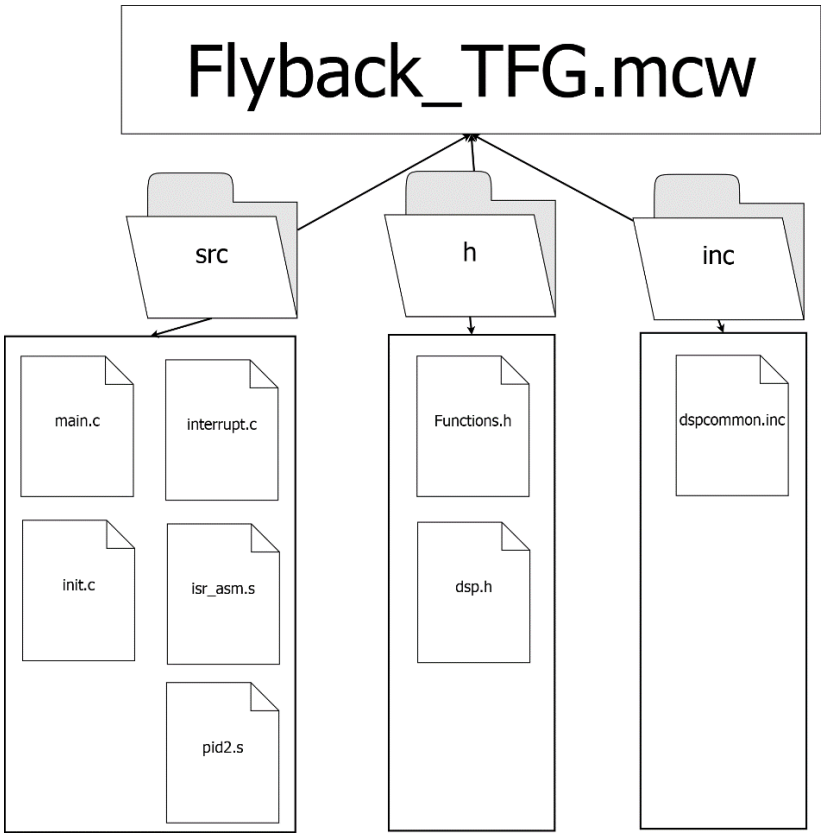


Figura 26. Diagrama con los archivos que forman parte del entorno del proyecto Flyback_TFG.mcw

Explicado brevemente, el proyecto de MPLAB consta de los siguientes archivos:

Archivos fuente: escritos en lenguaje C o ensamblador

Subprograma main.c: Como su nombre indica es el programa principal y está escrito en lenguaje C. Realiza los procesos de configuración de osciladores, llama a las funciones del init.c necesarias para que el controlador funcione y habilita el PWM y el conversor A/D:

Subprograma init.c: contiene las funciones que son llamadas desde main.c para hacer funcionar el Flyback.

Interrupt.c: contiene la interrupción necesaria para cambiar la referencia de tensión del Flyback mediante el potenciómetro.

Isr_asm.s: escrito en ensamblador para maximizar la velocidad. Contiene las instrucciones que se ejecutaran cuando se produzca la interrupción asociada al convertidor analógico digital 0, cargando los valores necesarios en el comparador para realizar el control por pico de corriente.

Pid2.s: Esta rutina escrita en ensamblador es llamada desde Isr_asm.s y su función es ejecutar el algoritmo de cálculo del regulador. Está escrita en ensamblador para tener el máximo control del programa.

Cabeceras e includes:

Dsp.h: Es un archivo muy largo, de más de 1600 líneas, proporcionado por Microchip que contiene la definición de multitud de constantes, estructuras y prototipos. Sirve de interfaz para las librerías dsp de microchip y facilita la programación de los microcontroladores dsPIC en aplicaciones de procesado digital de señal (DSP). A este archivo se le ha añadido la estructura tPID que almacena los parámetros del regulador.

Functions.h: Este fichero se limita a enunciar los nombres de las funciones que son llamadas desde la función main.c.

Dspcommon.inc: Archivo proporcionado por Microchip. Contiene macros escritas en ensamblador para facilitar la configuración del CORCON (core configuration).

5.4.2. main.c

Es un archivo escrito en C y como su nombre indica es el subprograma principal del proyecto y el primero en ejecutarse al iniciar el programa. En él nos encontramos las siguientes instrucciones:

En las primeras líneas del programa se comienza incluyendo los ficheros necesarios para la ejecución del mismo mediante la instrucción #include.

#include "p33FJ16GS502.h"

Esta cabecera contiene todos los nombres necesarios para programar los registros del dsPIC utilizando identificación nemotécnica.

#include "Functions.h"

En el archivo Functions.h se enuncian los nombres de las funciones que son llamadas desde la función main().

#include “dsp.h”

Contiene las Definiciones de múltiples tipos de variables, estructuras y prototipos de funciones que pueden ser usadas en operaciones de procesado digital de señal (DSP). En el programa se usa la estructura tPID.

Las siguientes instrucciones sirven para configurar los osciladores del dsPIC y permitir la depuración mediante el icd3

_FOSCSEL(FNOSC_FRC);

Selecciona el oscilador RC interno en el Encendido durante el Power On reset

_FOSC(FCKSM_CSECMD & OSCIOFNC_ON)

Configura el Registro de configuración del oscilador (Oscillator Configuration Register) con las siguientes opciones:

FCKSM_CSECMD: Habilita la generación de la señal de reloj

OSCIOFNC_ON: Configura el pin OSC2 como Digital I/O

_FWDT(FWDTEN_OFF)

Deshabilita el reloj del watchdog, si estuviera habilitado tendríamos que reiniciar el watchdog para evitar que este reinicie el programa.

_FPOR(FPWRT_PWR128)

Establece un retardo de 128ms en el Power On reset,

_FICD(ICS_PGD2 & JTAGEN_OFF)

Elige el canal de comunicación ICD conectando PGC2/EMUC2 y PGD2/EMUD2 y deshabilita el JTAG

Después de establecer esos bits de configuración, ya dentro de la función main(), se configura el PLL del oscilador, se espera a que enganche y se configura otro oscilador auxiliar del que dependerán el conversor ADC y el generador PWM haciendo que funcione a una frecuencia de unos 120MHz.

Una vez se ha configurado el reloj del microprocesador, se define la frecuencia a la que va a funcionar el generador PWM, es decir la frecuencia de conmutación de los transistores. A continuación se realizan las llamadas a las funciones de configuración que están en el fichero init.c:

FLYBACKDrive();

CurrentandVoltageMeasurements();

FLYBACKRefVoltValInit();

FLYBACKVoltageLoop();

Antes de entrar en bucle se activan tanto el PWM como el ADC que previamente se habían configurado en las funciones citadas anteriormente, por lo que a partir de este momento quedan habilitadas las interrupciones del ADC y se producirán periódicamente siguiendo el disparo provocado durante inicio de cada ciclo de la PWM.

Para la cuenta de tiempo se usa el temporizador Timer 1, que genera una interrupción en cada cuenta del tiempo programado. La rutina de gestión de esta interrupción se encuentra en el fichero isr.c

Finalmente se entra en el bucle infinito, while(1) en el que no hace nada excepto esperar interrupciones.

5.4.3. init.c

Este archivo contiene todas las funciones a las que se llama desde la función main(). Estas funciones mencionadas en el apartado son: FLYBACKDrive(), CurrentandVoltageMeasurements(), FLYBACKVoltageLoop() y FLYBACKRefVoltValInit() que realizan la configuración de todos los registros del dsPIC necesarios para programar su modo de trabajo.

El fichero comienza con dos líneas en la que se definen dos variables que tienen una gran importancia en el algoritmo de cálculo del regulador. Estas variables son dos arrays cuya longitud es igual a la cantidad de coeficientes de la FDT del regulador a implementar. Las variables se definen de la siguiente manera:

```
fractional FLYBACKVoltageABC[5] __attribute__ ((section (".xbss, bss, xmemory")));

fractional FLYBACKVoltageHistory[5] __attribute__ ((section (".ybss, bss, ymemory")));
```

La primera línea es la correspondiente a los valores de la FDT y se encarga de reservar memoria en el banco x para almacenarlos.

La segunda línea reserva en la zona de memoria Y espacio para ubicar el error y d actuales y anteriores

La introducción en la configuración de la memoria de datos de los espacios de memoria X e Y en los dsPIC de Microchip permite que las instrucciones identificadas como de tipo DSP, se ejecuten más rápido permitiendo que en operaciones con dos operadores ubicados en bancos de memoria distintos se carguen ambos al mismo tiempo en los registros del microcontrolador en lugar de hacerlo secuencialmente. La lista de instrucciones de este tipo se puede encontrar por ejemplo en el apartado 3.6 del manual dsPIC33FJ06GS101/X02 and dsPIC33FJ16GSX02/X04. Una de ellas es la instrucción MAC que juega un papel muy importante en algoritmo de cálculo del regulador. La sintaxis de esta instrucción es la siguiente:

MAC W4*W5, a ,[W8], W4 {,[W11], W5

Y ejecuta en un sólo ciclo de instrucción las siguientes funciones:

Multiplica el contenido del registro W4 con W5, suma el resultado al dato que se encuentra en el acumulador "a", a la vez carga en W4 el dato almacenado en la dirección apuntada por el registro W8, y en W5 el dato al que señala el registro W11. Todo este conjunto de instrucciones, que se repite se hace en un sólo ciclo de instrucción (40 MIPs, 25 ns), es lo que permite hacer un algoritmo como el del regulador que se va a hacer trabajar a una frecuencia de 100kHz (10μs). Este resultado no es sólo algo que tenga que ver con el software, sino que se sobre todo se apoya en una arquitectura de registros y buses hardware del DSP que soporta este tipo de instrucciones

Como se puede observar las variables FLYBACKVoltageHistory y FLYBACKVoltageABC son de tipo fractional aunque fractional no es un tipo estándar en lenguaje C. Este tipo está definido dentro del archivo dsp.h y buscando su definición en el archivo se puede encontrar lo siguiente:

```
typedef int          fractional;
```

Con esto se puede comprobar que las variables de tipo fractional son en realidad enteros con signo, y por lo tanto ocupan 16 bits. Uno de esos bits, el más significativo, es el bit de signo por tanto el rango de números que se pueden usar es de -2^15 a +2^15, es decir [-32768,+32767].

A continuación nos encontramos con las líneas que definen los coeficientes de la FDT del regulador con el signo cambiado, utilizadas para implementar la función:

$$R(z) = \frac{0.004447\,z^2 + 0.0001035\,z - 0.004344}{z^2 - 1.959\,z + 0.959}$$

Ec. 47

```
#define PID_FLYBACK_A1 Q15(+0.959)

#define PID_FLYBACK_A2 Q15(-0.959)

#define PID_FLYBACK_B0 Q15(+0.00444)

#define PID_FLYBACK_B1 Q15(+0.00010)

#define PID_FLYBACK_B2 Q15(-0.00434)
```

Estas se definen con el signo cambiado y solo contienen la parte decimal de la FDT. Hay que considerar un aspecto muy importante de la forma de trabajar de este DSP, y es que las operaciones matemáticas no se pueden realizar en formato de coma flotante ya que para ello el microcontrolador debería contar con una unidad de coma flotante (FPU) implementada por hardware y no es el caso, por tanto, se deben realizar las operaciones matemáticas en formato de coma fija. El formato de coma fija, aunque proporciona una

mayor precisión para un número determinado de bytes y puede ser más rápida de procesar, cubre un menor rango de valores. Esto condiciona la manera de trabajar con las variables del sistema a controlar obligando a escalarlas respecto al valor máximo para evitar desbordamientos. De este modo se trabaja en formato Q15 que cubre un rango de [-1, 0.999969482). Este formato utiliza 16 bits: 1 para el signo y 15 para la parte fraccional y cuenta con una precisión de $3.05176x10^{-5}$.

En el resto del archivo están las funciones a las que se les llama desde main.c.

5.4.3.1. FLYBACKDrive()

Al principio de esta función se configura el comparador interno del dsPIC que se utilizara para comparar la corriente en la fuente del transistor MOSFET con la tensión de control del lazo de tensión. Esto se hace configurando los bits del registro CMPCON1bits que es el encargado de controlar el comparador 1.

```
CMPCON1bits.CMPON=1;
CMPCON1bits.CMPSIDL=0;
CMPCON1bits.DACOE=0
CMPCON1bits.INSEL=0b00;
CMPCON1bits.EXTREF=0;
CMPCON1bits.CMPPOL=0;
CMPCON1bits.RANGE=1;
```

Lo primero que se hace es habilitar el comparador, luego deshabilita la opción de que este se apague cuando el dispositivo esté en reposo. A continuación desconecta la salida del pin DACOUT. Mediante los bits INSEL<1:0> se selecciona la entrada A del comparador 1 que se corresponde con el pin2 del dsPIC. Este pin es la entrada Analógica A0 por la que entra la tensión mediante la cual se obtiene la corriente que entraría al terminal + del comparador. La entrada - del comparador que es usada como referencia para la comparación está conectada a la salida de un DAC de 10 bits cuyo valor debe ser programado a través del registro CMREF sino se usa la referencia externa. A través de EXTREF=0 y RANGE=1 hace que use como máximo de tensión de referencia la mitad de la tensión de la alimentación del dsPIC por lo que se saturará con voltajes superiores a 1’65V. CMPPOL configura la polaridad para que no se invertida.

Después de haber configurado el comparador se configura el PWM mediante las siguientes líneas a través de distintos bits del registro IOCON2bits.

```
IOCON2bits.PENH = 1;

IOCON2bits.PENL = 1;

IOCON2bits.PMOD = 1;

IOCON2bits.POLH = 1;

IOCON2bits.POLL = 1;

IOCON2bits.FLTDAT = 0x00;
```

En estas líneas, en primer lugar, se habilitan los puertos que se van a usar como PWM2H y PWM2L mediante los bits PENH y PENL respectivamente. Estos canales que se corresponden con los pines 23 y 24 del dsPIC funcionarán como puertos PWM en lugar de hacerlo como GPIO (General Purpose I O) y trabajarán en modo redundante ya que el driver tiene la salida A negada y la B sin negar. Para hacerlos trabajar en modo redundante se pone el valor 01 en los bits PMOD. POLH y POLL configuran la polaridad de los canales H y L respectivamente, POLH que está puesto a 1 está invertido y POLL que es 0 no lo está. Finalmente, FLTDAT establecerá el estado en el que se pondrán los canales PWM cuando se produzca una falta, en este caso para que el MOSFET este a nivel bajo y el transistor que controla la rampa de compensación esté a nivel alto se debe establecer un valor de 0x11.

A continuación a través del registro PWMCON2bits se establece la configuración para que las actualizaciones del ciclo de trabajo no se produzcan en el momento de haber calculado el nuevo valor de control sino en el siguiente ciclo para evitar inestabilidades.

Lo siguiente es asignar la falta. Para entender esto hay que saber que cuando la tensión en el terminal positivo del comparador sea igual a la de su pata negativa, la salida ACMP1 pasará a nivel alto. Esto hará que se produzca la condición de falta que debe cortar los pulsos en el transistor de potencia del Flyback. Para conseguir este resultado, previamente hay que entender varios aspectos de funcionamiento del dsPIC. En primer lugar hay que tener en cuenta que la salida del comparador ACMP1 es un pin interno del dsPIC. Para hacer este pin interno operativo es necesario conectarlo a uno de los 36 pines también internos que dispone el dsPIC que al estar configurados como salidas reciben el nombre de pines remapeables. Esto se hace con la instrucción:

```
RPORbits.RP33R= 0b100111;
```

Con esta instrucción el pin remapeable 33 es una salida que se ha asociado a ACMP1 que es la salida del comparador. Después hay que programar el pin para que su activación se considere como un fallo o falta.

Para completar esta configuración es necesario configurar el pin RP33 para que sea también una entrada asociada a la falta 1. Esto se hace programando el registro que conforman los pines virtuales y sabiendo que también tienen la función de servir de

entradas. Estos registros se denominan registros de selección de entrada de pines periféricos (RPINR). Para asignar la falta 1 (FLT1) al pin virtual RP33 se usa la siguiente línea de código.

```
RPINR29bits.FLT1R=33.
```

Después de esto se configura la funcionalidad de Leading Edge Blanking a través del registro LEBCON2bits. Esta característica del DSPIC sirve para retrasar levemente la medida de la corriente tras el cierre del MOSFET y así poder evitar que el pico de corriente inicial proporcione una medida de corriente que pueda recortar el ciclo de trabajo de forma prematura.

En la parte final de la función a través de los registros FCLCON2bits y TRGCON2bits se establece el número de ciclos PWM que se tienen que dar para que se produzca la interrupción por eventos del generador PWM y que a su vez dispara también la interrupción del conversor ADC. En la configuración elegida en cada inicio de pulso de la PWM se inicia una conversión TRGDIV = 0 y también se define que la primera interrupción que provoca la conversión se hace desde el primer pulso TRGSTRT = 0. Finalmente se inicializa el ciclo de trabajo con el valor mínimo de tiempo muerto posible en el PWM2 mediante el PDC2 y se fija el retardo entre la interrupción de inicio de la PWM y el inicio de la conversión en el ADC TRIG2.

Este retardo permite evitar que tensiones transitorias que se produzcan en el convertidor puedan llegar a distorsionar las medidas. De esta manera el generador PWM2 del microprocesador queda configurado.

5.4.3.2. CurrentandVoltageMeasurements()

El propósito de esta función es configurar los registros necesarios relacionados con el conversor analógico digital para que el microcontrolador pueda obtener mediante las entradas analógicas las medidas de tensión y corriente necesarias para controlar el Flyback. Para obtener estas medidas se utilizan los puertos del dsPIC asignados a las entradas analógicas A0 y A1 correspondientes a los pines 2 y 3 respectivamente.

En primer lugar configura el registro ADCON del conversor analógico digital:

```
ADCONbits.FORM = 0;

ADCONbits.EIE = 0;

ADCONbits.ORDER = 1;

ADCONbits.SEQSAMP = 0;

ADCONbits.ADCS = 2;
```

El bit FORM puesto a cero establece el formato de salida del comparador como entero. EIE (Early Interrupt Enable bit) es un bit que hace que se produzca la interrupción cuando los dos datos hayan sido convertidos, es decir en el siguiente ciclo de la señal PWM evitando así las actualizaciones inmediatas. El bit ORDER con un valor de 1 se utiliza para

que se convierta primero el canal impar y luego el par. Las dos últimas líneas de ese bloque sirven para activar el muestreo simultaneo y para establecer el divisor del reloj en FADC/3 de manera que el reloj del conversor quedara configurado de la siguiente manera:

$$RelojADC = \frac{F_{adc}}{3} = \frac{120MHz}{3} = 40MHz$$

Como según el dsPIC33FJ06GS101/X02 and dsPIC33FJ16GSX02/X04 datasheet de Microchip el tiempo de conversión típico de estos microcontroladores es de $tcon = 14 \cdot T_{AD}$, se tendrá una tasa de muestreo de $f_s = \frac{1}{tcon} = \frac{1}{14 \cdot 25ns} = 2'86MSPS$ pero como el dsPIC33FJ16GS502 es un microcontrolador que cuenta con dos registros de aproximaciones sucesivas (SAR) y tiene 2 circuitos S&H separados para los canales pares e impares, en total, duplica esa tasa llegando a las 5'72 MSPS con esta configuración.

Después de esto se configura el registro IEC6 (Interrupt Enable Control Register 6) y el IPC27 (Interrupt Priority Control Register 27):

```
IEC6bits.ADCP0IE = 1;

IFS6bits.ADCP0IF = 0;

IPC27bits.ADCP0IP = 5;
```

Las dos primeras líneas sirven para habilitar las interrupciones y limpiar el flag de interrupción del par 0 del conversor analógico digital. El flag es un bit que se pone a 1 cuando se produzca una interrupción y debe ser puesto a 0 de nuevo durante esa interrupción para que pueda volver a darse. La siguiente línea establece la prioridad de las interrupciones del par 0 del conversor AD. La prioridad máxima es 7 y la mínima es 1 ya que establecer una prioridad de 0 implica que la interrupción no se produciría.

Para seguir con la configuración del conversor es necesario establecer el registro ADPCFG (Analog-To-Digital Port Configuration Register):

```
ADPCFGbits.PCFG0 = 0;

ADPCFGbits.PCFG1 = 0;
```

Los bits PCFG0 y PCFG1 permiten la lectura analógica de la corriente que entra por A0 y de la tensión que entra por A1 respectivamente.

Finalmente en las últimas tres líneas se pone el bit de dato preparado a 0, este bit debería empezar a cero y cambiar cuando el conversor tenga un valor. Después se habilitan las solicitudes de interrupciones para el bucle de control y se hace que la conversión en el par 0 sea disparada por el generador PWM2.

```
ADSTATbits.P0RDY = 0;

ADCPC0bits.IRQEN0 = 1

ADCPC0bits.TRGSRC0 = 5;
```

Con todo lo mencionado anteriormente se pueden ver todos los elementos implicados en la sincronización del conversor AD y la configuración de las interrupciones vinculadas a la conversión que permitirán que el algoritmo de control sea calculado donde interesa. Todos los eventos se producirán sincronizadamente con el generador PWM, el conversor A/D producirá interrupciones al darse los eventos necesarios del PWM y esta interrupción será la que corte el flujo de ejecución del programa para que con las variables a controlar ya leídas puedan realizarse todos los cálculos que permitirán hallar el valor de control que entrará en funcionamiento en el siguiente ciclo de trabajo del generador PWM.

En resumen, con toda la configuración anterior en cada ciclo de la señal PWM se hará una lectura de la corriente que entra por el terminal A0 del microcontrolador y se comparará con el valor de control para decidir cuándo cortar el pulso y fijar el ciclo de trabajo. Este valor no podrá hacerse efectivo inmediatamente ya que el bit EIE del registro ADCONbits está puesto a 0 y por lo tanto el cambio se producirá en el siguiente periodo del generador PWM. Tras haber aplicado el ultimo valor calculado en el algoritmo de control se repetirá nuevamente todo el proceso anterior a lo largo del siguiente periodo y así sucesivamente de manera que, en definitiva, en cada uno de los ciclos de la señal PWM se producirá todo el proceso de actualización y cálculo del control a aplicar en el siguiente ciclo.

5.4.3.3. FLYBACKRefVoltValInit();

Esta función será la encargada de habilitar el cambio de referencia del Flyback a través del potenciómetro. Al presionar el pulsador ws1 de la placa se producirá una interrupción que hará que el dsPIC33fj16gs502 lea la tensión que entra en la pata7 conectada al potenciómetro de la placa 16bits 28 pin Development Board.

5.4.3.4. FLYBACKVoltageLoop();

La función FLYBACKVoltageLoop() será la encargada de obtener las zonas de memoria que contienen los valores de error y control y los coeficientes del regulador a través de las direcciones de los punteros *control y *coefficients obtenidas de la memoria que apuntan respectivamente a esas zonas. Estos valores serán los que se usen más adelante en el algoritmo de control. Tras haber obtenido las direcciones que apuntan a estos punteros y a través de la llamada a la función PDInit esas zonas de memoria se borran para después llenarse con los valore A1, A2, B0, B1 y B2 definidos anteriormente.

Estos valores obtenidos a través del script de MATLAB son modificados de manera que solo se utiliza su parte decimal con el signo cambiado. Esos números que se han definido en la primera línea deberían ser fraccionarios y estar comprendidos entre cero y uno, pero por si no lo fueran el programa lo comprueba y en el caso de que estén en un rango inválido para el formato Q15 el programa entraría mediante la instrucción while(1), en

un bucle infinito del que no podría salir al no haberse habilitado el PWM en este punto del programa.

Una vez se han introducido correctamente los coeficientes del regulador se dejan el resto de los campos de las variables tPID a cero a excepción de la referencia que se pone a un valor mínimo establecido en la constante PID_FLYBACK_VOLTAGE_REFERENCE.

5.4.4. El isr_asm.s

Isr_asm.s es un fichero escrito en ensamblador en el que se encuentra la subrutina que será la encargada de atender a la interrupción del par 0 del conversor A/D del dsPIC. Está escrita en ensamblador para tener un mayor control sobre las instrucciones del que se podría obtener programando en lenguaje C y también para poder conseguir una mayor velocidad de cálculo. Es en esta subrutina en la que se empieza todo el proceso de cálculo necesario para obtener el valor de la variable de control y por ello es necesario tener suficiente control sobre las instrucciones del programa para poder garantizar que estas instrucciones y todo el algoritmo de control en general duran siempre la misma cantidad de tiempo. Todos los cálculos e instrucciones que están presentes en este archivo tardan menos de un periodo la señal PWM en completarse para la frecuencia de 100kHz tal y como se ha podido comprobar experimentalmente.

Esta subrutina comienza introduciendo en el interior del registro de trabajo W0 la dirección que apunta al comienzo de la variable FLYBACKVoltagePID que es del tipo tPID definido en dsp.h y guarda en su interior, entre otras cosas, el puntero a los coeficientes del regulador, el puntero al historial de valores anteriores del regulador y su valor actual de control. De este modo la subrutina podrá desplazarse por todas esas zonas de memoria partiendo del valor guardado en ese registro de trabajo. Después en el registro de trabajo W1 se guarda el valor guardado en el buffer ADCBUF1. Este buffer es el lugar donde el conversor A/D guarda el valor leído en la entrada A1 del dsPIC que se corresponde con la medida de la tensión de salida. El valor leído en el buffer del conversor A/D tiene una resolución de 10 bits y como los valores con los que va a trabajar el dsPIC tienen una longitud de 16 bits es necesario escalar el registro mediante un desplazamiento de 5 bits hacia la izquierda de manera que el valor queda en las posiciones más significativas del registro de trabajo W1. Después de esto se guarda este valor medido y escalado en la posición de memoria del tPID que apunta a measuredOutput a partir de la dirección de inicio de la estructura que previamente se había almacenado en el registro W0.

Una vez se han ejecutado estas instrucciones se llama a la rutina PID que es la encargada de calcular el valor de control. La rutina PID se describe con detalle en el apartado siguiente.

Después de haber ejecutado la rutina PID se habrá introducido en la memoria de datos el nuevo valor de control que se ha calculado dentro de ella. A continuación el valor de la variable controlOutput se extrae y se desplaza 5 bits hacia la izquierda para adecuarlo a 15 bits de Q15 de CMPDAC1 este escalado produce una ganancia en el regulador. El

valor de control introducido en el registro producirá una señal PWM con un ciclo de trabajo que vendrá dado por la comparación entre este valor y el valor de la corriente.

Finalmente se debe limpiar el flag de interrupción de convertidor A/D para permitir que sea posible que se pueda volver a generar otra interrupción de este tipo más adelante. Al tratarse de una rutina de interrupción se hace necesario terminarla con la instrucción RTFIE que permite recuperar los registros que controlan la prioridad de la interrupción además de los que se recuperarían si se terminase con la instrucción de retorno RETURN.

5.4.5. EL pid2.s

En esta rutina que se llama desde la rutina Isr_asm.s se ejecuta el algoritmo de cálculo del regulador. Debido a que es parte del algoritmo de cálculo de control, se escribe en ensamblador para tener el máximo control posible sobre la ejecución del programa.

La función de transferencia del regulador expresada en transformada z y normalizada para que el coeficiente A0 sea la unidad es la siguiente:

$$R(z) = \frac{D(z)}{E(z)} = \frac{0.004447 - 0.0001035z^{-1} - 0.004344z^{-2}}{1 - 1.959z^{-1} + 0.959z^{-2}}$$

Ec. 48

La ecuación anterior expresada en ecuación de diferencias es la siguiente:

$$d_n = 0.004447e_n - 0.0001035e_{n-1} - 0.004344e_{n-2} + 1.959d_{n-1} - 0.959d_{n-2}$$

Ec. 49

Donde los coeficientes de la FDT son los que van desde b0 a b2 y de a1 hasta a2. Los valores de error son los que van desde e_n hasta e_{n-2} y los del ciclo de trabajo van de d_n a d_{n-2}, donde d_n será el nuevo valor de control que se va a calcular.

En lo siguiente se explica la programación del algoritmo que realiza la ecuación de diferencias dada por la Ec. 49 y que implica el uso de la ecuación DSP MAC.

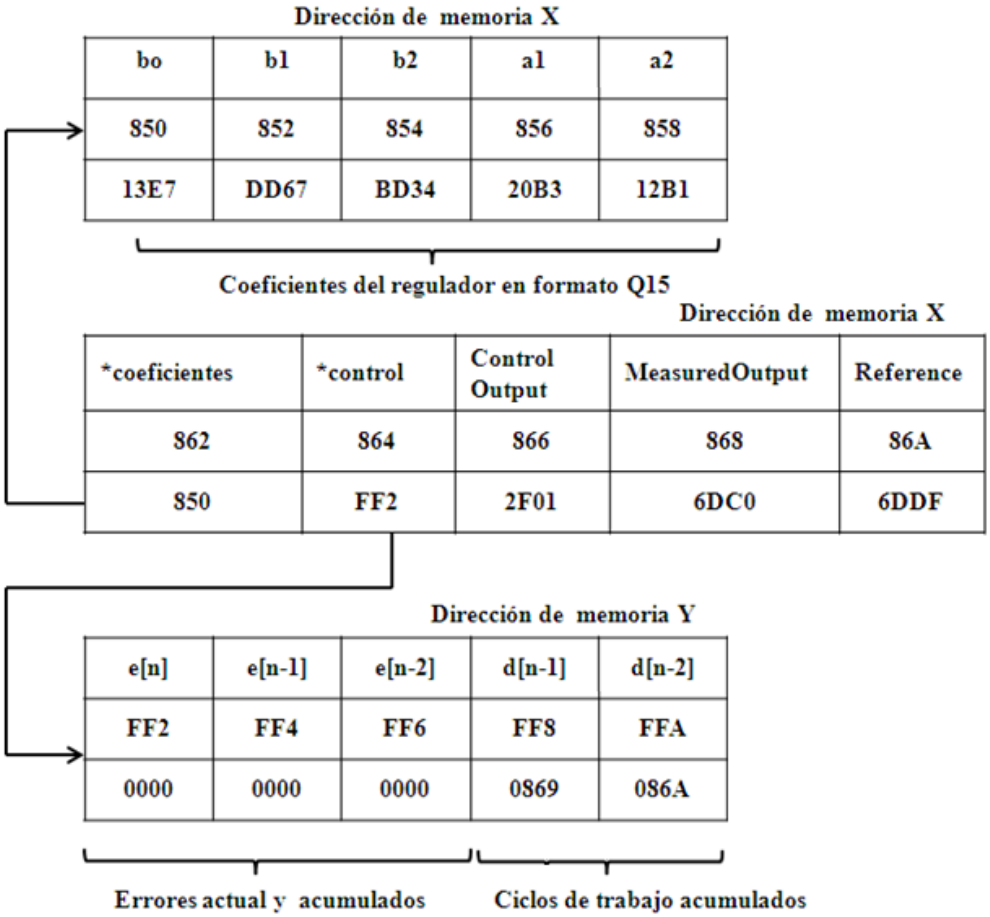


Figura 58. Relación entre la estructura tPID y las matrices de variables y coeficientes del regulador.

Desplazamiento de los valores de error.

Lo primero que se hace es desplazar los valores de error para que el nuevo valor del error se pueda colocar en la posición inicial, posición cero, de la matriz FlybackVoltajeHistory[0]

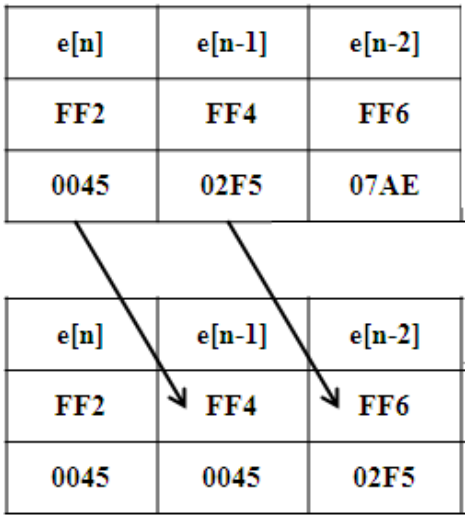


Figura 59. Recolocación de los términos de error en la matriz FlybackVoltajeHistory[]

El siguiente paso es hacer la resta entre el valor de referencia y el escalado del ultimo valor medido de la tensión de salida que se encuentra en la variable MeasuredOutput. El error se almacena en FlybackVoltajeHistory[0] como indica la Figura 60. Se calcula el error y se almacena en la posición FlybackVoltajeHistory[0]

*coeficientes	*control	Control Output	MeasuredOutput	Reference
862	864	866	868	86A
850	FF2	2F01	6DC0	6DDF

6DDF-6DC0=1F

e[n]	e[n-1]	e[n-2]
FF2	FF4	FF6
001F	0045	02F5

Figura 60. Se calcula el error y se almacena en la posición FlybackVoltajeHistory[0].

Algoritmo de controlador.

A continuación se realiza el cálculo del algoritmo del regulador que en transformada z tiene la siguiente expresión:

$$R(z) = \frac{D(z)}{E(z)} = \frac{0.004447 - 0.0001035z^{-1} - 0.004344z^{-2}}{1 - 1.959z^{-1} + 0.959z^{-2}}$$

El algoritmo del regulador consiste en implementar su correspondiente ecuación de diferencias, que es la siguiente:

$$d_n = 0.004447e_n - 0.0001035e_{n-1} - 0.004344e_{n-2} + 1.959d_{n-1} - 0.959d_{n-2}$$

Como se ve, esta ecuación en diferencias es un sumatorio con un total de 5 términos, en el cual para formar cada termino hay que realizar una multiplicación cuyos factores son un parámetro del regulador y su correspondiente termino variable, bien sea de error o de ciclo de trabajo. Es precisamente para este tipo de cálculos para lo que se ha diseñado la instrucción MAC, multiplicar y acumular, que recordemos no es una mera instrucción software, dado que lleva asociado diseño hardware de la estructura de memoria con el objetivo de que se haga con la máxima rapidez. Respecto a la descripción de la función que realiza, se trata de una operación de tipo DSP, que multiplica los datos ubicados en dos registros de trabajo, uno cargado desde la memoria X (coeficientes) y otro desde la memoria Y (Error y ciclo de trabajo anteriores). Posteriormente suma este valor a lo que hay en el acumulador, que en la primera multiplicación es 0 y en las

sucesivas es el valor de la cuenta previa. Adicionalmente, esta instrucción guarda en unos registros de trabajo, las nuevas posiciones de memoria a donde apuntar en sucesivas cuentas (apuntando al siguiente coeficiente y al siguiente error-ciclo de trabajo), estos son W8 y W10, que hacen esta operación mediante un post-incremento. Sin embargo, los valores que son apuntados por estos, es decir los números útiles a multiplicar, los mete en W4 y W5 y una vez se meten ahí esos valores, es cuando se hace el post-incremento de w8 y w10, para apuntar a las siguientes posiciones. Un dato significativo sobre la potencialidad de esta instrucción es que el conjunto de operaciones que se han descrito se hace en 1 ciclo de instrucción. Para tener una información completa sobre esta instrucción conviene leer la explicación sobre la misma en el documento: 16-bit MCU and DSC Programmer's Reference Manual High-Performance Microcontrollers (MCU)and Digital Signal Controllers (DSC).

El uso de esta instrucción se puede ver en esta copia que se adjunta de la parte inicial del programa de regulador. Nótese que los comentarios son las líneas precedidas por ";" y que las instrucciones están en negrita. Como se muestra en los comentarios que acompañan a las instrucciones cada vez que se ejecuta la instrucción MAC se incrementa en uno los términos que se incorporan al sumatorio.

```
;-----------------------------------------------------------------
                ; se multiplica y se prepara la siguiente multiplicación
                ; a = b0 * e[n]
                ; w4 = b1, w5 = e[n-1]
                mac w4*w5, a, [w8]+=2, w4, [w10]+=2, w5

                ; a = b0 * e[n] + b1 * e[n-1]
                ; w4 = b2, w5 = e[n-2]
                mac w4*w5, a, [w8]+=2, w4, [w10]+=2, w5

                ; a = b0 * e[n] + b1 * e[n-1] + b2 * e[n-2]
                ; w4 = a1, w5 = d[n-1]
                mac w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
```

Una vez que ha terminado este bloque de instrucciones, como se muestra, ya todos los términos de la variable de error están incorporados a la suma, y a partir de este punto lo que toca es incorporar los términos correspondientes a valores anteriores del ciclo de trabajo. En la función de transferencia el parámetro a1 del ciclo de trabajo es mayor que 1, pero tenemos que recordar que en la matriz de los parámetros del regulador solo se ha almacenado la parte decimal del número, reservando para esta parte del programa

la incorporación de su parte entera. Esta descomposición del parámetro en su parte entera y decimal es necesaria porque por la forma en la que trabaja la instrucción MAC, solo se pueden multiplicar partes fraccionarias ya que los dos números tienen que estar representados en el formato Q15. La forma de introducir la parte entera del parámetro es a través de las instrucciones de suma con acumulador ADD si la parte entera es positiva o SUB en caso contrario, repitiendo la instrucción tantas veces como el valor del entero, con lo que el código que continua la secuencia de instrucciones anteriores y completa el algoritmo del regulador queda:

```
;-----------------------------------------------------------------

                ; se mete d[n-1] en b

                lac w5, b

                ;a = b0 * e[n] + b1 * e[n-1] + b2 * e[n-2] + d[n-1]

                add a

                ; a = b0 * e[n] + b1 * e[n-1] + b2 * e[n-2] +  d[n-1] + a1 * d[n-1]

                ; en este caso el coeficiente a1, tiene parte entera '1' y parte fraccionaria
a1

                ; w4 = a2, w5 = d[n-2]

                mac w4*w5, a, [w8], w4, [w10]-=2, w5

                ; se mete d[n-2] en el registro intermedio del acumulador b (high)

;-------------------------------------------------------------------------

                ;antes [w8]+=2 ahora [w8] Y [w10] retrocede de pos[6] a pos[4] en la
matriz de historia

                mac w4*w5, a

                ; a = b0 * e[n] + b1 * e[n-1] - b2 * e[n-2] + (1 + a1) * d[n-1] - (a2) * d[n-2]

                ; se tiene en a, el valor de la ecuación en diferencias del regulador tipo2

                ; se redondea el valor de d que hay en acc a (40 bits) y se mete la parte
más significativa en w1(16 bits)

                sac.r a, w1

;-------------------------------------------------------------------------
```

En esta última instrucción, el valor de la ecuación en diferencias del regulador tipo2 contenido en el acumulador a es redondeado y guardado en un registro de trabajo. Esto es así porque el acumulador es de 40 bits pero los registros de trabajo y posiciones de memoria son de 16, por lo tanto se redondea para tomar la parte más significativa del acumulador (ACCH) y meterla en un registro de 16 bits. Sobre cómo se hace el redondeo

en el dsPIC33 hay que consultar el apartado 3.6.3.1 de las hojas de especificaciones del dsPIC33FJ06GS101/X02, documento DS70318F de Microchip. También hay que tener en cuenta que durante el algoritmo al hacer operaciones de sumas y restas no se puede descartar la posibilidad de desbordamiento en la representación Q31 de los números. Este aspecto es muy importante y se comenta al final de este apartado, de momento hay que saber que está configurado para que si hay desbordamiento el valor resultante se limite entre 0X7FFF +0,999969482 y -1.

Con esto el rango de la variable de ciclo de trabajo en el regulador trabaja en el intervalo [0, +0,999969482]. El valor que se almacena en la variable controlOutput puede ser saturado, este bloque realimenta su salida a la entrada de la ecuación de diferencias con objeto de mejorar el comportamiento cuando el regulador entra en condiciones de saturación. Con este tipo de estrategias "antisaturación" se limitan las excursiones en la variable de control durante los transitorios, haciendo más rápida la vuelta a las condiciones lineales de funcionamiento una vez que ha pasado el transitorio.

Finalmente, del mismo modo que al principio de esta rutina pid2.s se relocalizan las muestras anteriores del error, ahora se hace los mismo con las muestras anteriores del actual

```
;-------------------------------------------------------

mov [w10],w11

mov w11, [w10 + #+2] ; d[n-1,anterior]==>d[n-2]

mov w1,[w10]          ;w1==>d[n-1]

mov w1, [w0 + #offsetcontrolOutput]

; se recupera de la pila lo que había en los registros que han sido usados

pop CORCON

pop w10

pop w8

pop w5

pop w4

pop.s

return
```

Para terminar de comentar esta rutina, merece prestar atención el tratamiento que se hace del registro CORCON en la rutina (Core Configuration). Al principio de la misma se salva en la pila con la instrucción pop CORCON junto con el resto de los registros de trabajo (Wx) que se usan en la rutina porque se modifica su contenido en la misma, y al final se recupera del mismo modo que el resto de los registros antes de la instrucción RETURN que pone fin a la rutina.

La modificación de este registro se hace a través de una macro que se encuentra en el archivo “include” dspcommon.inc. A esta macro se le invoca al principio de la rutina pid2.s a través de una macro del siguiente modo:

fractsetup w8

Tal como la lee el ensamblador las instrucciones que realiza son las siguientes:

mov #FRACT_MODE, W8

mov W8,CORCON

El valor de la constante FRACT_MODE en binario se define en el mismo archivo dspcommon.inc con los siguientes bits:

FRACT_MODE=0000000011110000

Esta configuración de bits se carga en el registro CORCON, que queda configurado como muestra la siguiente tabla

CORCON: CORE CONTROL REGISTER

U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-0	R-0
—	—	—	US	EDT(1)	DL2	DL1	DL0
bit 15							bit 8

R/W-0	R/W-0	R/W-1	R/W-0	R/C-0	R/W-0	R/W-0	R/W-0
SATA	SATB	SATDW	ACCSAT	IPL3(2)	PSV	RND	IF
bit 7							bit 0

En este punto se hace necesaria la explicación sobre la configuración del registro CORCON para examinar el modo en que hace la asignación de valores al símbolo FRAC_MODE en el archivo dspcommon.inc porque puede llegar a ser confuso ya que en las directivas usadas para asignar valores se están mezclando directivas de ensamblador con instrucciones de lenguaje C. Este método tan particular de asignar valores es posible porque en el proceso de compilación del programa primero se ejecuta el compilador C que traduce las instrucciones a ensamblador. Las instrucciones de asignación son las siguientes:

```
;-----------------------------------------------------------------------------------------------------------
; Operational modes for fractional computation:
.equ    SATA_ON,1<<7            ; AccA sat. enabled
```

```
.equ    SATB_ON,1<<6            ; AccB sat. enabled
.equ    SATDW_ON,1<<5           ; data write sat. enabled
.equ    ACCSAT_ON,1<<4          ; Accs sat. 9.31 enabled
.equ    FRACT_SET,SATA_ON|SATB_ON|SATDW_ON|ACCSAT_ON ; set mask
.equ    RND_OFF,~(1<<1)         ; convergent rnd. enabled
.equ    IF_OFF,~(1<<0)          ; multiply fractional enabled
.equ    FRACT_RESET,RND_OFF&IF_OFF     ; reset mask
.equ    FRACT_MODE,FRACT_SET&FRACT_RESET       ; fractional mask
mov     #FRACT_MODE,\wx
mov     \wx,CORCON
;-----------------------------------------------------------------------------------------------------------
```

Examinando las líneas anteriores, la directiva **.equ SATA_ON,1<<7**, implica que **SATA_ON** va a ser sustituida por la instrucción de lenguaje C: 1<<7 cuyo resultado será escribir el número 0x80 en un byte de memoria. Cuando se hace la función lógica OR en la instrucción

.equ FRACT_SET,SATA_ON|SATB_ON|SATDW_ON|ACCSAT_ON

Es equivalente a la siguiente igualdad:

FRACT_SET=0x80|0x40|0x20|0x01=0xF0

mientras que la directiva **.equ RND_OFF,~(1<<1)** primero escribe el número 0x02 para luego invertirlo, con lo que **RND_OFF** será 0XFC. De este modo, siguiendo toda la cadena de asignaciones en el registro CORCON el valor que se programa en CORCON es:

CORCON=0x00F0

Con esta configuración del acumulador el tratamiento matemático de los números obtenidos en las operaciones de multiplicación, suma y resta que usan el acumulador es el siguiente:

- **CORCON.US=0** bit 12 US: Signo de los multiplicadores del motor DSP

1= Los multiplicadores no tienen signo.

0=Los multiplicadores tienen signo.

En este caso, el algoritmo de multiplicación tiene en cuenta el signo de los multiplicandos, luego multiplica números en complemento a 2, y por lo tanto con signo. Como además son números fraccionarios, como se explica más adelante en este apartado el bit IF (Integer Fractional) hay que ponerlo a 0 para que tenga esto en cuenta cuando se hace la multiplicación.

- **CORCON.SATA=1** bit 7 bit de saturación habilitada en el ACCA.

1= Saturación del acumulador A habilitada.

0= Saturación del acumulador A deshabilitada.

Cuando el número vaya a ser mayor que 0x7FFFFFFFFF si es positivo o menor que 0x8000000000 si es negativo, el resultado se satura en uno de estos valores. Como este bit está puesto a 1 no ocurre desbordamiento catastrófico. Si el bit OVATE del registro INTCON1 está puesto a 1 se produce una interrupción de tipo TRAP, que es una interrupción de la máxima prioridad.

- **CORCON.SATB=1** bit 6 bit de saturación habilitada en el ACCB.

1= Saturación del acumulador B habilitada.

0= Saturación del acumulador B deshabilitada.

En el caso de este bit se aplica al B lo mismo que se ha explicado para el acumulador A, naturalmente cambiando las referencias al registro A por lo que se aplica al B.

- **CORCON.SATDW=1** bit 4 bit de selección del modo de saturación

1= Escritura en la memoria de datos trabaja con saturación habilitada.

0= Escritura en la memoria de datos trabaja con saturación deshabilitada.

Con este registro se configura como trabajará el dsPIC33 con las instrucciones de almacenamiento del acumulador de 44 bits en los registros de 16 bits. Cuando la saturación para el almacenamiento está habilitada. Si el número que se encuentra en acumulador tiene parte entera se almacena el máximo número fraccional positivo=0x7FFF, y si la parte entera es negativa entonces se almacena 0x8000, esto es "-1", que es el máximo número negativo en el formato Q15.

- **CORCON.ACCSAT=1** bit 4 bit de selección del modo de saturación

1= 9.31 saturación (alta saturación).

0= 1.31 saturación (saturación norma).

Permite que en las sumas y restas los números se extiendan hacia la parte superior del acumulador, bits de guarda sin que el bit de saturación SA del registro de estado de la CPU (registro SR) pase a nivel alto. Si esto ocurre sí pasara a nivel alto el bit de desbordamiento (OA). Con esto se permite que los números después de una operación de suma o resta tengan parte entera no nula sin que tenga necesariamente que considerarse un error cubriendo el rango de extensión en el intervalo [-28, 28]= [-256,256].

- **CORCON.RND=0** bit 1 bit de selección del modo de redondeo

1=Redondeo con desviación (convencional).

0=Redondeo sin desviación (convergente).

En el programa se ha optado por el redondeo convergente. Para saber en qué consiste consultar el apartado 3.6.3.1 de las hojas de especificaciones del dsPIC33FJ06GS101/X02, documento DS70318F de Microchip.

- **CORCON.IF=0** bit 0 bit de selección del modo de multiplicar entero o fraccionario

1=Modo entero para la multiplicación DSP habilitada.

0= Modo fraccionario para la multiplicación DSP habilitada.

Cuando este bit es cero, una vez realizada la multiplicación, el resultado se desplazará automáticamente en el acumulador un bit hacia la izquierda, generando un numero con 2 signos de bit y 31 bits fraccionales. Este formato lo usa el DSP para mantener correctamente el posicionamiento del punto decimal. Como resultado del desplazamiento, el bit menos significativo del resultado, para aclarar mejor, el de peso 2-31, va a ser siempre cero.

Cuando está activado el modo entero (IF=1) entonces se hace una multiplicación sin signo. En este caso, el resultado de la multiplicación no se desplaza y se almacena en el acumulador como un número de 32 bits. Para terminar la explicación sobre la multiplicación sin signo, indicar que para trabajar de este modo hay poner IF=1 y el bit US=1.

Como se comentó anteriormente, un aspecto muy importante que hay que tener en cuenta en las operaciones de cálculo del regulador es la posibilidad de que durante su ejecución, dado que las ecuaciones no son solo de multiplicación de números fraccionarios en formato Q15, sino que también hay que realizar sumas y restas, se supere el máximo valor que puede representarse en los 40 bits del acumulador, desbordamiento catastrófico, o también puede ser necesario supervisar si el número se hace mayor que la unidad, lo que ocurrirá en el caso de que para representar el número hagan falta más que los primeros 32 bits del acumulador. Para poder trabajar adecuadamente este aspecto, en primer lugar hay que familiarizarse con la forma de operar del dsPIC respecto a los desbordamientos de cálculo en el acumulador. Para ello se ha representado en Figura 27 en forma de flujograma la lógica de tratamiento del desbordamiento, donde se han marcado con fondo gris, la condición de desbordamiento de acuerdo con las opciones que se han elegido para la configuración de CORCON.

Por lo tanto, examinando la tabla donde se muestra la configuración de CORCON, vemos que tal y como se ha implementado el programa en respuesta a las operaciones con el acumulador es la siguiente:

- No se permiten interrupciones tipo trap porque CORCON.SATA y CORCON.SATB=1.

- La Supersaturación está habilitada, CORCON.ACCSAT=1, luego se permiten resultados intermedios mayores que la unidad.
- Como el bit CORCON.SATDW=1, cuando el resultado se pasa del acumulador a uno se los registros de la CPU de 16 bits, con la instrucción SAC.R (Store from ACcumulator with Rounding) si el número es mayor que la 0x7FFF o menor que 0x8000 se satura a estos límites.

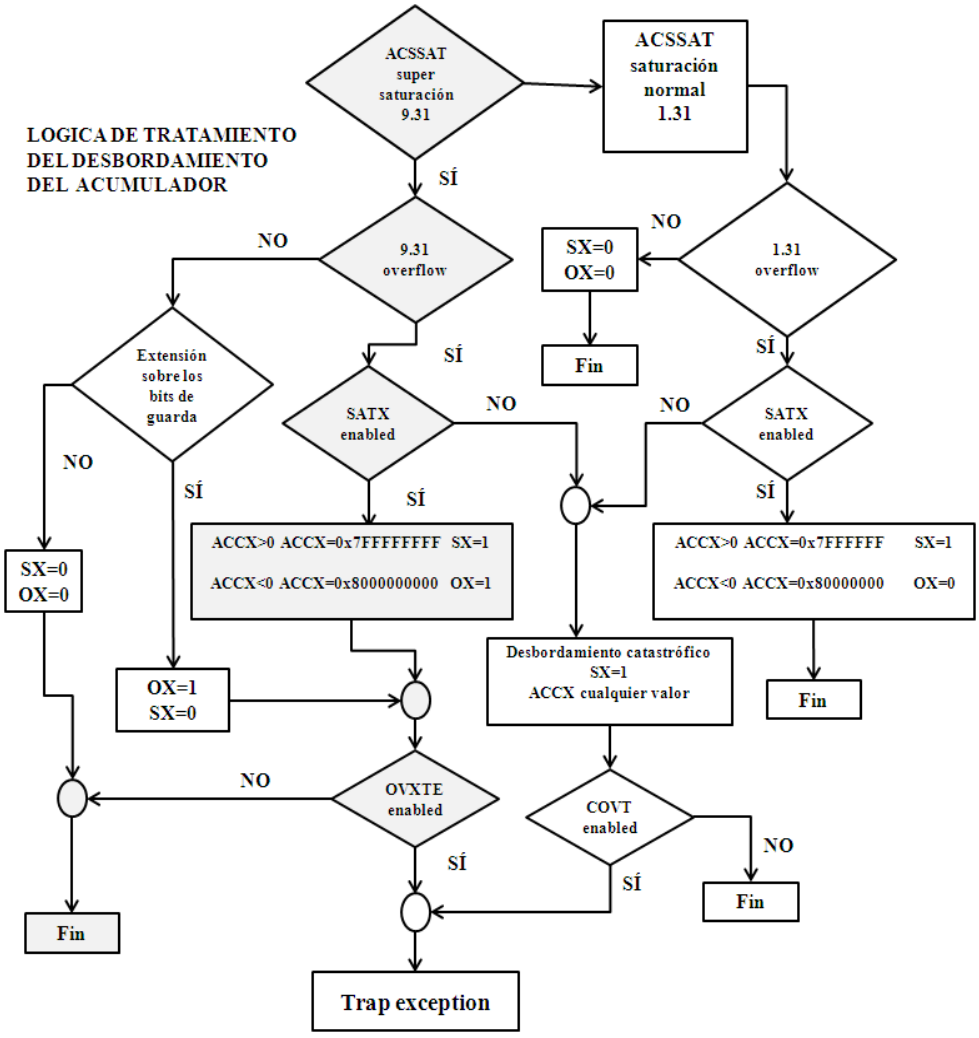


Figura 27: Diagrama de flujo con las diferentes opciones sobre el funcionamiento y la configuración del desbordamiento en el acumulador.

5.5. EL PROGRAMA CON COMUNICACIÓN SERIE

5.5.1. INTRODUCCIÓN

Hasta este punto se ha implementado el control digital del convertidor Flyback con control por pico de corriente con unos coeficientes y unas referencias de control fijas

utilizando el dsPIC33FJ502GS pero también es posible añadir comunicación serie a este programa para añadir las siguientes funcionalidades:

1.-Posibilidad de encender o apagar el convertidor sin necesidad de quitar la alimentación de microcontrolador.

2.-Cambiar la referencia para permitir tensiones variables.

3.-Modificar los coeficientes del regulador para estudiar su efecto sin necesidad de reprogramar el microchip.

4.- Obtener los valores de tensión medidos en el lazo externo.

Para realizar esto se han añadido un archivo rs232.c que contiene una función para iniciar el módulo UART del dsPIC y la interrupción que se vincula a la recepción de datos. Este archivo tiene en su include la cabecera uart.h proporcionada por microchip. Además de esto se han hecho ciertas modificaciones al init.c , al main.c y al interrupt.c.

5.5.2. MODIFICACIONES EN EL INIT.C

El init incluye dos nuevas funciones, la función FLYBACKReferenceRoutine() que permitirá cambiar la referencia poco a poco y la función Delay_ms que permitirá añadir los retrasos necesarios a esos incrementos.

Para ello el archivo empieza de la siguiente manera:

```
#define FLYBACK_SOFTSTART_INCREMENT          0x10

unsigned int FLYBACKReferenceNew,FLYBACKReferenceOld ;

extern unsigned int TimerInterruptCount;
```

La primera línea es la constante que definirá el salto que hará la referencia hasta llegar al valor nuevo cuando este se cambie. La segunda línea define la variable de la referencia actual y la nueva que se usarán para decidir cuándo actualizar la referencia en el main.c y la tercera línea es el contador que utiliza la interrupción asociada al timer 1 para darnos un retraso determinado por el parámetro de entrada.

5.5.2.1. FLYBACKReferenceRoutine

Es una función muy sencilla en la que se compara FLYBACKReferenceOld con la referencia que tenemos introducida en FLYBACKVoltagePID.controlReference de la siguiente manera:

```
if (FLYBACKVoltagePID.controlReference<FLYBACKReferenceOld)
{       while (FLYBACKVoltagePID.controlReference <= FLYBACKReferenceOld)
        {       Delay_ms(1);
                FLYBACKVoltagePID.controlReference += FLYBACK_SOFTSTART_INCREMENT;
        }
}else{   while (FLYBACKVoltagePID.controlReference > FLYBACKReferenceOld)
        {       Delay_ms(1);
                FLYBACKVoltagePID.controlReference -= FLYBACK_SOFTSTART_INCREMENT;
        }
        FLYBACKVoltagePID.controlReference = FLYBACKReferenceOld;
```

Si la referencia que tenemos es menor que el valor que se va a introducir se va incrementando poco a poco en interior del bucle while con un retardo de 1ms. Se saldrá del while en cuanto se haya superado la referencia deseada y a continuación se igualará.

Si la referencia del regulador es mayor que la que se pretende introducir, el proceso es el inverso. Se decrementará poco a poco con retardos de 1ms y cuando se alcance la referencia deseada o el valor inmediatamente inferior se igualará a esta.

5.5.2.2. Delay_ms

Recibe de la función FLYBACKReferenceRoutine el numero 1 como parámetro que será el número de interrupciones del timer1 que se producirán antes de parar el timer1 y sus interrupciones. Para hacer esto se prepara el timer1 para que cada interrupción del mismo se produzca cada milisegundo y se habilita su interrupción. De esta manera el timer1 se desbordará transcurrido 1 ms tras lo cual se desactivará el timer.

Con esto se consigue tener retardos de 1 ms que son llamados las veces necesarias por el FLYBACKReferenceRoutine. De este modo se construye una rampa de subida en la referencia a seguir hasta llegar a su nivel final en el que se quedará fijo.

Modificaciones en el interrupt.c

En el fichero interrupt.c se han escrito las instrucciones de la subrutina de atención a interrupción del timer 1 (retardo de 1ms) relacionadas con la función delay explicada anteriormente. Esta interrupción está programada en C, y para que el programa dsPIC la identifique como tal, debe declararse como una función de gestión de interrupción del siguiente modo:

void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt()

Donde _T1Interrupt() es un nombre reservado con el que se consigue que se asigne en la tabla de vectores de interrupción la interrupción de Timer1 con la posición de memoria donde carga el inicio de la rutina de gestión de interrupción. De no hacerse así, el contador de programa no podría encontrar su posición en el Interrupt Vector Table de la memoria de programa, y le resultaría imposible llegar a las subrutinas para poder ejecutar sus instrucciones.

5.5.3. MODIFICACIONES EN main.c

El archivo main incorpora los siguientes cambios respecto a la versión anterior:

1.-Añade las variables extern FLYBACKReferenceOld y FLYBACKReferenceNew que se declaran en el init.c

2.-Llama a InitUART1() para iniciar el UART y habilitar las interrupciones de recepción de datos.

3.-el bucle while(1) que estaba vacío ahora ejecuta:

```
if (FLYBACKReferenceOld!=FLYBACKReferenceNew)
{       FLYBACKReferenceOld=FLYBACKReferenceNew;
        FLYBACKReferenceRoutine();
}
```

FLYBACKReferenceNew será modificado desde la rutina de interrupción rs232.c el main comprueba en bucle si ha cambiado, si ha cambiado mete ese valor en FLYBACKReferenceOld y llama a FLYBACKReferenceRoutine(), explicada anteriormente, que se encargara de que la variable FLYBACKVoltagePID.controlReference cambie poco a poco hasta alcanzar ese valor.

5.5.4. EL ARCHIVO rs232.c

Es el archivo encargado de iniciar el módulo UART y de gestionar la rutina de recepción. Comienza con las líneas:

```
#include <p33FJ16GS502.h>
#include "uart.h"
#include "dsp.h"
```

La inclusión de los ficheros dsp.h y <p33FJ16GS502.h>, que ya se han explicado anteriormente, sirve para trabajar con la estructura PID y configurar los registros de forma nemónica respectivamente. Dentro de la cabecera uart.h proporcionada por Microchip que está incluida en rs232.c se ha añadido la siguiente enumeración:

```
enum {          NO_ORDEN= 0X00,
        ON_FLYBACK= 0XA0,
        OFF_FLYBACK= 0XA1,
        REF_FLYBACK= 0XA2,
        TENSION_SALIDA_FLYBACK=0XA3
        PARAM_REG =0xA4
}ordenUSB;
```

La única función que cumple es asignarle nombres a los valores numéricos que puede tomar ordenUSB para facilitar la programación. El resto del archivo uart.h incluye constantes y referencias a las funciones de la librería periférica uart. De esta manera se puede ver que el protocolo de comunicación funcionará enviando un byte de instrucción y otro con los datos asociados a esta.

5.5.4.1. U1RXInterrupt()

Se trata de la interrupción que se dispara al recibir un byte. Después de declarar variables empieza de la siguiente manera:

```
if (ordenUSB==NO_ORDEN)
ordenUSB = U1RXREG;
```

La variable ordenUSB comentada anteriormente será la que almacenará la orden recibida, si su valor indica que no hay ninguna orden lo que se reciba se guardará en ella. A confinación va un switch(ordenUSB) en el que se evalúa esa variable mediante las instrucciones case. Esto es equivalente a anidar ifs pero resulta en un código mucho más fácil de leer. Las primeras instrucciones son las de encendido y apagado del conversor Flyback.

Estas instrucciones se limitan a habilitar o deshabilitar el PWM mediante la asignación de un 0 o un 1 al bit PTEN del registro PTCONBITS. Después ponen el valor NO_ORDEN a la variable orden_USB. También incluyen una variable de estado de la fuente.

A continuación viene la orden de cambio de referencia que permitirá variar la tensión que proporcione el Flyback. Esto lo hace modificando el valor de la variable FLYBACKReferenceNew, esta variable es la que se compara con FLYBACKReferenceOld dentro de main.c y si son distintas las igualará y actualizará el valor a través de la función FLYBACKReferenceRoutine() del init. Es importante tener en cuenta que la referencia se envía en 2 bytes por tanto se entrará tres veces en ella: al recibir el código REF_FLYBACK responderá con el mismo código para pedir el byte bajo de la referencia y una vez recibido este volverá a mandar el mismo código para solicitar el byte alto. Así que para cambiar de referencia se producirán tres interrupciones en total y para contarlas se usa

la variable cont_RX_bits. Esta variable al declararse en la cabecera del archivo se inicializa con valor cero, dentro del case(REF_FLYBACK) la primera vez que entra está pues a 0 pero después de enviar el código 0XA2 para pedir el byte bajo se pone a 1 por lo que pasara al bloque de programa que guarda el byte bajo y pone el contador a 2 para que la tercera vez que se produzca la interrupción se guarde el byte alto y se reinicie el contador y se termine con ordenUSB=NO_ORDEN.

En la misma línea funciona el case PARAM_REG solo que en lugar de entrar tres veces se entrarán once veces: la primera para limpiar FLYBACKVoltagePID.controlHistory y responder con el mismo código y las demás para obtener los 5 coeficientes que ocuparan también 2 bytes cada uno. Se usa la variable cont_abcCoef para contar el índice del array que tiene los coeficientes y se incrementa después de obtener un byte par. La variable cont_TX_bits se utiliza para contar los bytes recibidos.

Finalmente case TENSION_SALIDA_FLYBACK manda la tensión de salida medida en ADCBUF1 que es el buffer del ADC, es decir, la tensión medida en la pata A1. Entra dos veces en esta interrupción la primera vez que recibe la orden guarda la tensión del buffer y manda el byte bajo y la segunda manda el byte alto.

5.5.4.2. void InitUART1()

Es la función que habilita el puerto UART1 del dsPIC que es el único puerto uart que posee. Para configurar el puerto hay que editar varios registros entre ellos U1MODEbits, U1STAbits y U1BRG. Siguiendo ejemplos de Microchip se han configurado los bits de la siguiente manera.

U1BRG: URT1 Baud rate generator: Define la velocidad del módulo UART siguiendo la siguiente expresión

$$U1BRG = \frac{fosc}{16 \cdot baudiosdeseados} - 1 = \frac{40MHz}{16 \cdot 11494} - 1 \sim 216$$

Ec. 50

U1STAbits-Registro de estado y control del UART1

Bit 14 UTXINV 1: el pin u1tx está a nivel bajo en estado de reposo

Bit 11 UTXBRK 0: deshabilitado el bit de sincronización

Bits 7 y 6 URXISEL 0: las interrupciones de recepción se producen cada vez que se recibe un carácter.

Resto de bits de U1STAbits a 0.

Se deben seleccionar los terminales mediante las instrucciones RPINR18bits.U1RXR=7 y

RPOR3bits.RP6R=3

U1MODEbits-Registro de modo de uart1: Todos sus bits a cero excepto UARTEN y UTXEN que se ponen a uno al final de la función, en ese orden.

Además de esto las interrupciones de recepción se habilitan con IEC0bits.U1RXIE = 1 y las de transmisión en principio no necesitan ser habilitadas.

Hay que tener en cuenta que el UART es un periférico que no tiene unos pines fijos asignados ya que es remapeable por tanto se debe vincular sus puertos de transmisión y recepción con alguna de las patas RP del chip. En este caso se deben utilizar la 17 y la 18 que son las que están conectadas al PIC18F2450

RPINR18bits.U1RXR=7 asigna la recepción de UART1 al terminal RP7 que se corresponde con pin 18 del dsPIC

RPOR3bits.RP6R=3; asigna la transmisión UART1 al terminal RP6 que se corresponde con pin 17del dsPIC

Con todas las modificaciones del programa original de control comentadas hasta ahora se ha conseguido dotarle de conectividad serie y nuevas funcionalidades que permitirán controlarlo mediante USB a través del PIC18F2450 y una aplicación C#.

5.5.5. PIC18f2450

El PIC18F2450 es un microcontrolador de 16 bits de Microchip de bajo coste que ofrece un alto rendimiento computacional. Aunque es un microcontrolador de propósito general incluye un módulo de comunicación bus serie universal (USB) que cumple todas las especificaciones de USB 2.0.

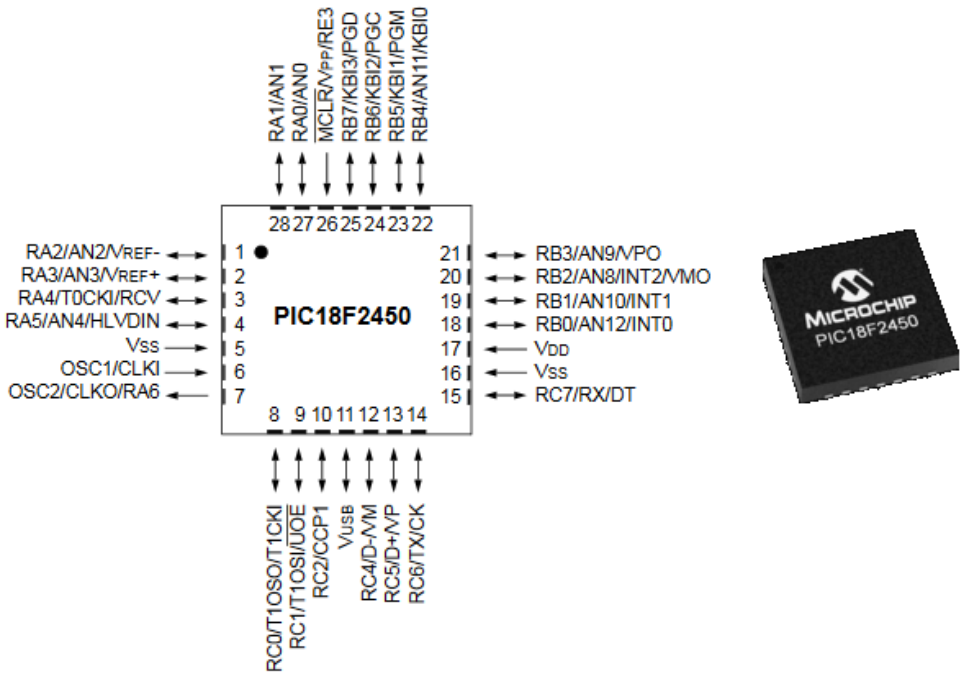


Figura 28: pinout del PIC18F2450 e imagen de su encapsulado

Como ya se ha comentado anteriormente en el apartado 5.2, la placa 16 bit 28 pins Development Board incluye un PIC18f2450 para controlar su puerto USB. Este microcontrolador se ocupa de conectar el puerto USB de la placa con el dsPIC33FJ16GS502 aunque tal y como viene de fabrica sirve más que nada para ejecutar el programa de ejemplo que se encarga de hacer ecos de caracteres.

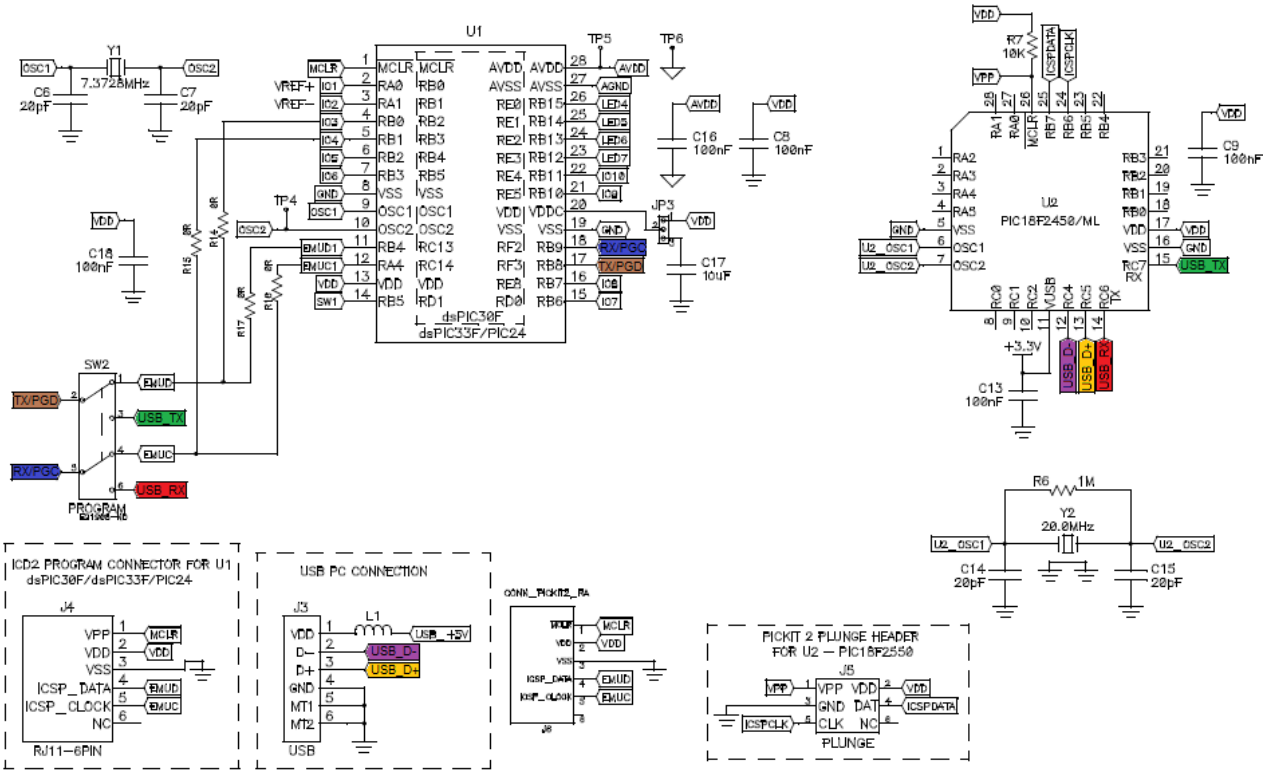


Figura 29: Esquema de conexiones de la placa 16bits 28 pin Development Board, con las conexiones relacionadas con el USB marcadas

Tal y como se puede ver en la Figura 29: Esquema de conexiones de la placa 16bits 28 pin Development Board, con las conexiones relacionadas con el USB marcadas con el interruptor SW2 de la placa en la posición que habilita el USB, los pines quedarían conectados de la siguiente manera:

Pin 17 del dsPIC33f al pin 15 del PIC18F: El pin 17 del dsPIC es el quedaba configurado como pin de transmisión de su modulo UART1 en la función initUART1 de rs232.c. En el caso del PIC18F se trata del pin de recepción USART que no puede ser cambiado al no ser un periférico remapeable.

Pin 18 del dsPIC al pin 14 del PIC18F; El pin 18 del dsPIC es el quedaba configurado como pin de recepción de su modulo UART1 en la función initUART1 de rs232.c. En el caso del PIC18F se trata del pin de transmisión USART que no puede ser cambiado al no ser un periférico remapeable.

Pin 13 del PIC18F al pin D+ del conector USB: aquí va la señal + USB que va al ordenador

Pin 12 del PIC18F al pin D- del conector USB: aquí va la señal - USB que va al ordenador.

De lo escrito anteriormente cabe aclarar dos cosas. La primera es que la diferencia entre un controlador UART y un USART es que el UART solo soporta transmisión asíncrona mientras que el USART soporta transmisión síncrona o asíncrona. Un USART trabajando asíncronamente es igual que un UART. La segunda es que en la conexión USB hay un data + y un data – porque el protocolo USB utiliza una señal diferencial para reducir los ruidos. Por data + va la señal de datos y por data – esa misma señal invertida. Como los cables están cerca el uno del otro el ruido en ambos será similar por lo que al restar la señal con su versión invertida se obtendrá la señal amplificada con gran parte de su ruido cancelado.

5.5.6. MODIFICACIÓN DE LA PLACA

Como se ha comentado hasta ahora tal y como está la placa 16 bits 28pin Development Board usando el ICD3 solo es posible programar el dsPIC33F pero el PIC18F2450 es un chip normal y corriente, es decir, no se puede programar porque Microchip no quiso añadir esa funcionalidad a su placa de desarrollo. No obstante, como los pines de programación del chip existen y están accesibles a través de la cabecera J5, es posible modificar la placa para poder programar ambos microcontroladores usando el ICD3.

Para hacer esto posible se han realizado las conexiones necesarias en J2, J5 y J6 soldando cables. Estas conexiones son:

1.-MCLR de J6 a VPP de J5. MCLR de J6 está conectada al pin VPP del puerto ethernet que utiliza el ICD3. VPP de J5 está conectada al MCLR del PIC18F. Permite al ICD3 resetear el microcontrolador en depuración.

2.-VDD de J6 a VDD de J5. Para que el ICD3 pueda alimentar el PIC18F2450

3.-EMUD y EMUC de J6 soldados a 3 pines cada uno en la zona protoboard. Son los pines que utiliza el ICD3 para la programación. Se dejan sin conectar a nada.

4.-DAT y CLK de J5 a la zona de prototipado, son los pines de programación del PIC18F. Van soldados en pines al lado de los anteriores, sin conectar a nada más ni entre ellos, procurando que el pin DAT quede frente al pin EMUD del ICD3 y que el pin CLK este frente al pin EMUC del ICD3.Serivirán paran programar el PIC18F.

5.-EMUD1 y EMUC1 de J2 conectadas a un cable rojo y uno negro respectivamente. Soldados a un conector Dupont hembra doble. Servirá para programar el dsPIC33F.

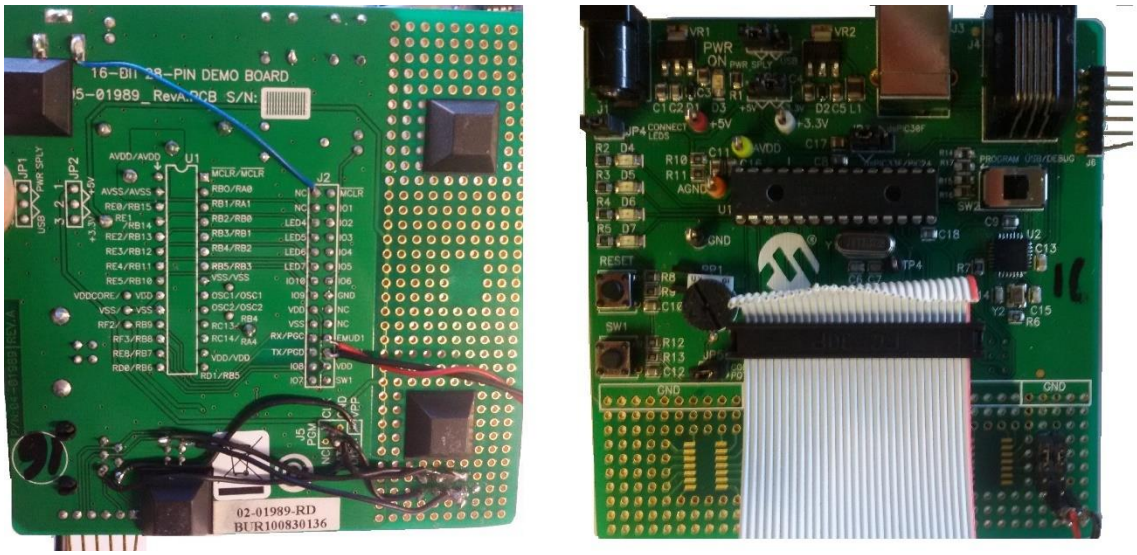


Figura 30 placa modificada para poder programar ambos microcontroladores

Con la placa así modificada SW2 deberá estar siempre en la posición USB/DEBUG ya que se conectarán EMUD y EMUC mediante el cable rojo y negro

Una vez hechas estas modificaciones el método para programar es el siguiente:

Para programar el dsPIC33F deben conectarse los cables rojo y negro que van a EMUD1 y EMUC1 del dsPIC33F a los pines de la zona protoboard que están conectados al EMUD y EMUC del puerto del ICD3.

Para programar el PIC18F se deja ese cable desconectado y mediante jumpers se unen en la zona de prototipado EMUD del icd3 con DAT de J5 y EMUC del ICD3 con CLK de J5.

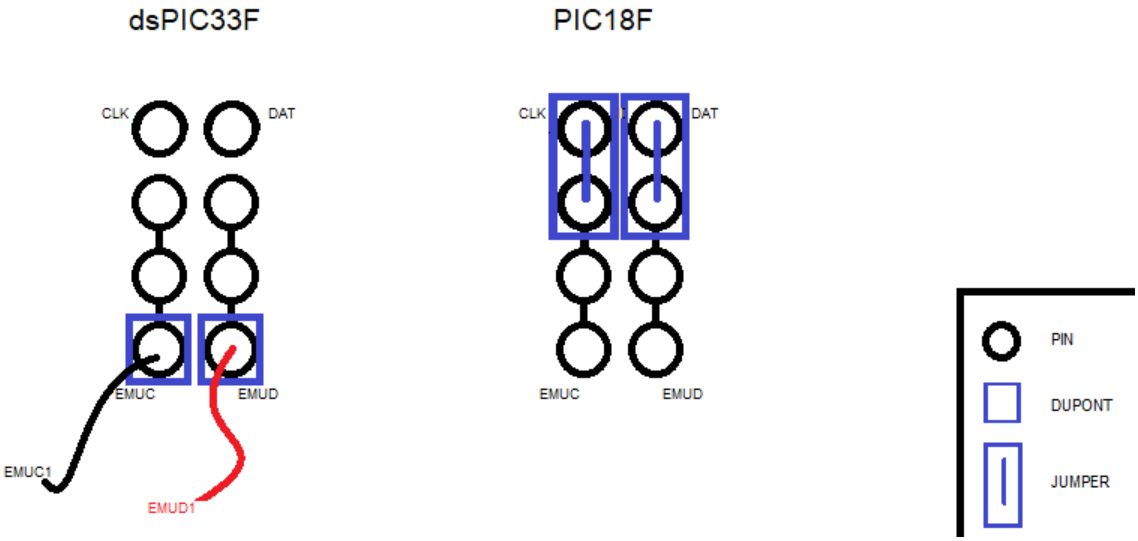


Figura 31: Conexiones para programar uno u otro chip

5.5.7. EL PROGRAMA DEL PIC18F2450

Con la placa modificada ya es posible programar y depurar el PIC18F utilizando el ICD3. Para programar las funcionalidades USB de este microcontrolador existen proyectos de demostración de Microchip y como realizar un programa de estas características es algo muy complejo que por sí mismo sería suficiente para realizar un TFG. El programa que se explica a continuación esta desarrollado a través de mínimas modificaciones realizadas a uno de ellos llamado Microchip USB C18 Firmware Versión 1.0. desarrollado por Rawin Rojvanit el 11/19/04

Para que este programa funcione y haga lo que se pretende deben editarse el user.c y el main.c para que el PIC18F sea capaz de reenviar las ordenes USB del PC al dsPIC.

En el fichero user.c se gestionan las órdenes recibidas por USB desde el PC. Dentro de la función ServiceRequest hay un switch(dataPacket.CMD) en el que se compara ese dataPacket.CMD que contiene el byte recibido desde el PC con las ordenes USB predefinidas, que coinciden con las del dsPIC. Estas órdenes están definidas dentro de la estructura CMD que está dentro de una unión dataPacket definida en el fichero user.h. Al recibir la orden vía USB se almacena en el valor de la variable ordenUSB y se reenvía al dsPIC33F lo que genera una interrupción en el dsPIC33F durante la cual responde. Al recibir la respuesta del dsPIC se produce una interrupción en el PIC18F que es gestionada en la función SerialReceiveInterrupt() que se encuentra en el main.c.

La función SerialReceiveInterrupt() del main.c es la que se ocupa de controlar las comunicaciones entre el PIC18F y el dsPIC33F tiene una estructura muy parecida a la interrupción de recepción del dsPIC. Se encarga de reenviar los datos del receiveBuffer.buffer USB mediante el TXREG del UART al dsPIC excepto en el caso de la tensión medida en la que lo hace es el dsPIC el que la envía. En ese caso lo que hace es guardar el valor en la variable tensionSalidaFLYBACK la cual pasara al dataPacket._byte que se enviara en la función ServiceRequest del user.c.

Con lo anteriormente mencionado se tiene ya una placa 16bits 28pin Development Board que será reconocida por el PC como Microchip Custom USB y podrán desarrollarse aplicaciones que la permitan controlar un Flyback

5.5.8. El PROGRAMA EN C#

Para probar que todo lo anteriormente escrito funciona y que es posible controlar el Flyback con un PC a través de USB en este TFG se ha desarrollado también una aplicación escrita en C# utilizando Microsoft Visual Studio 2017. Se trata de una solución del tipo "Aplicación de Windows Forms". Visual Studio es un entorno de desarrollo muy intuitivo que permite que la interfaz con el usuario se "programe" de manera gráfica

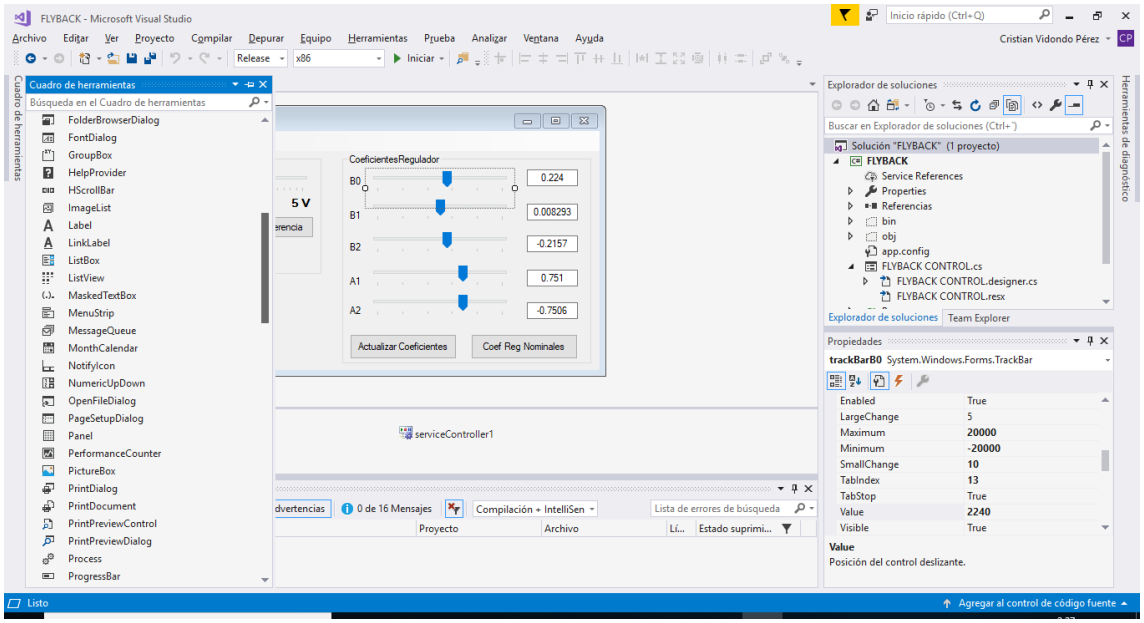


Figura 32: Vista de diseñador de Visual Studio

Tal y como se ve en la Figura 32: Vista de diseñador de Visual Studio al crear una nueva Solución, que es como se llaman los proyectos en Visual Studio, si se elige una solución de Windows Forms crear una ventana será tan sencillo como arrastrar componentes de cuadro de herramientas a la ventana. Después haciendo clic en cada elemento se podrán cambiar sus propiedades en el cuadro de la esquina inferior derecha. Esas propiedades pueden ser por ejemplo las etiquetas de texto si las tiene, el rango de valores si es un control deslizante, su color, el orden de tabulación, etc.

Dentro del cuadro de propiedades de los elementos hay un icono de un rayo amarillo, este sirve para asignar manejadores de eventos a los elementos. Por ejemplo, se puede hacer que se ejecute cierta función al hacer Clic en un botón, cuando un control de tipo radio cambie, cuando un checkbox se active, cuando se presione una tecla…

Además de la interfaz con el usuario se ha creado el archivo usb_interface_DCDC.cs. Este archivo contendrá esas funciones que se ejecutarán cuando se produzcan los eventos en la ventana y el API de la USB. Para implementar la comunicación USB hace uso de la

librería dinámica mpusbapi.dll. Se trata de una librería desarrollada por microchip que proporciona las funciones de acceso al puerto USB con microcontroladores de la familia PIC18Fxx5x para una facilitar el desarrollo de aplicaciones basadas en el bus USB. Para funcionar correctamente necesita el driver mchpusb.sys. Este archivo sirve tanto para Visual Basic como para Visual C, además de otros.

En el archivo usb_interface_DCDC.cs se define el espacio de nombres usb_api para organizar el código, a continuación la clase usb_interface que contendrá las funciones importadas de la librería mpusbapi.dll que son usadas a su vez en funciones más simples que vienen a continuación. Las funciones de usb_interface_DCDC.cs son:

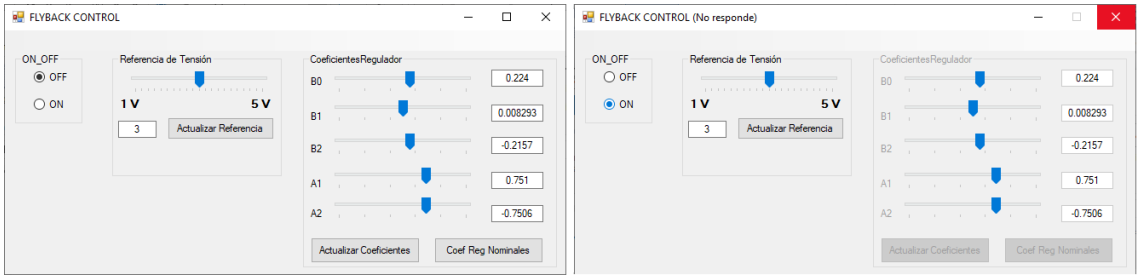
OpenPipes(): Selecciona el dispositivo a comunicar y abre el canal

ClosePipes(): cierra la comunicación.

SendReceivePacket: Manda un dato, se le meten como parámetros el dato a enviar, su longitud y la longitud esperada del dato a recibir. Si la longitud esperada en la recepción es correcta devuelve 1.

A continuación vienen las funciones que mandaran las ordenes de encendido apagado, cambio de referencia, y cambio de coeficientes del regulador.

El programa se ha diseñado de la siguiente manera:



Estando en OFF se pueden cambiar los coeficientes y la referencia, al pasar a on se envían los coeficientes y la referencia y la orden de encendido. Estando en ON se puede cambiar la referencia, pero el control de los coeficientes queda bloqueado.

Toda esa lógica se programa en el archivo FLYBACK CONTROL.cs que contiene las funciones que serán llamadas al producirse los eventos sobre los controles, estas a su vez harán uso del usb_interface para enviar los datos al dsPIC.

6. RESULTADOS

Con todo lo desarrollado anteriormente se ha podido verificar su correcto funcionamiento en el laboratorio. Comprobando que la fuente regula y que es posible cambiar su referencia y los coeficientes del regulador.

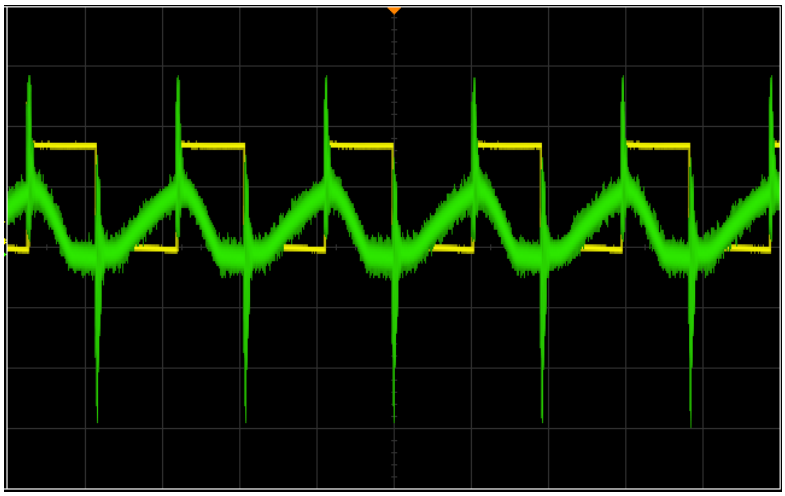


Figura 33: señal PWM2H y medida de la corriente

Con los coeficientes nominales se ha podido comprobar que el ciclo de trabajo además de cambiar en función de la tensión de entrada también se puede modificar mediante el programa escrito en C# FLYBACK CONTROL.

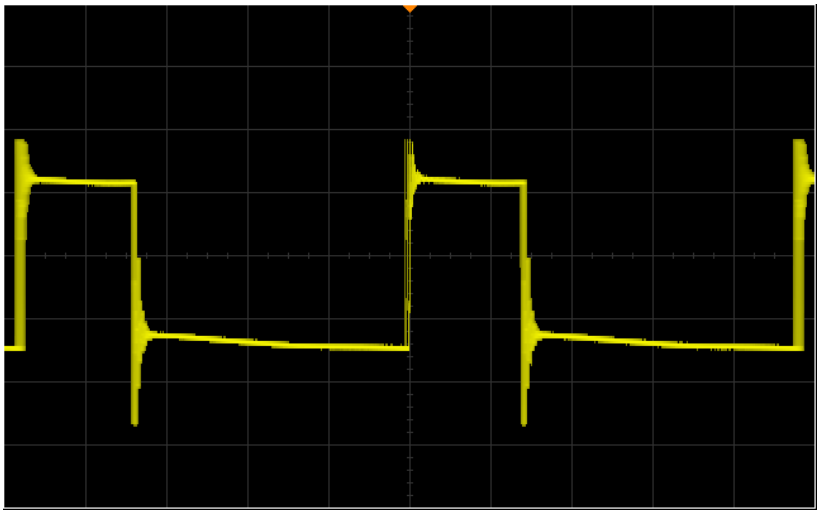


Figura 34: señal PWM con un ciclo de trabajo del 75%

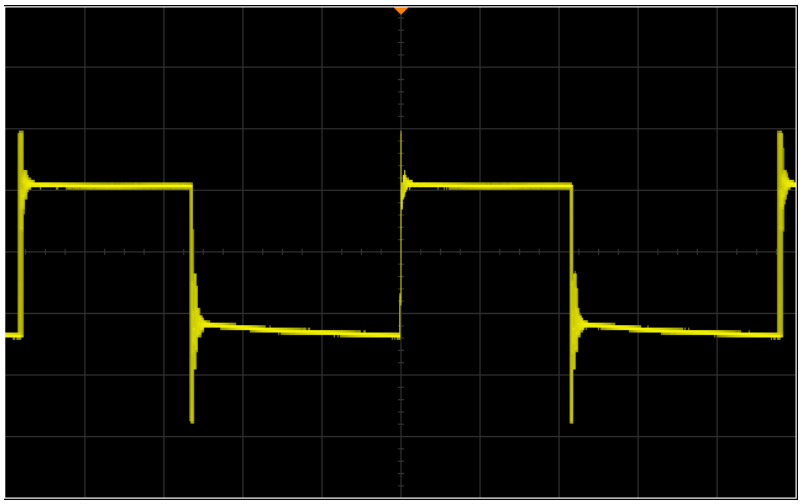


Figura 35: señal PWM2H con un ciclo de trabajo de cerca del 50%

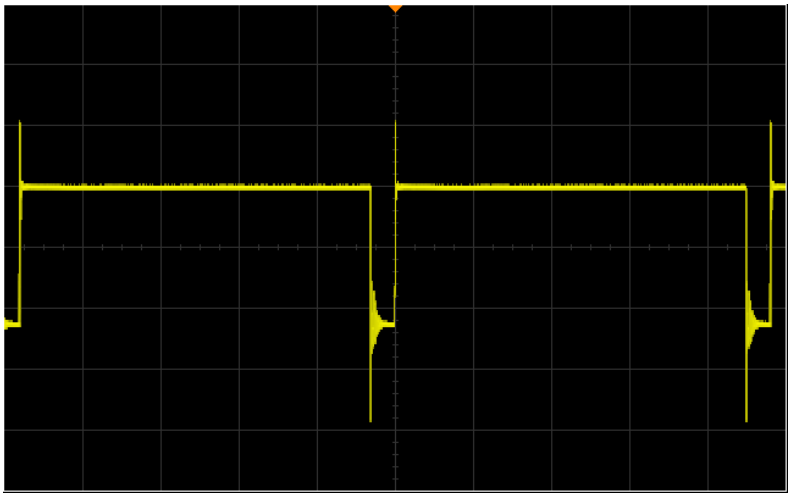


Figura 36:señal PWM con ciclo de trabajo máximo

También se ha podido comprobar que los coeficientes del regulador al ser cambiados con el programa de control por coeficientes configurados al azar desestabilizan la fuente que al restaurar los nominales vuelve a ser estable.

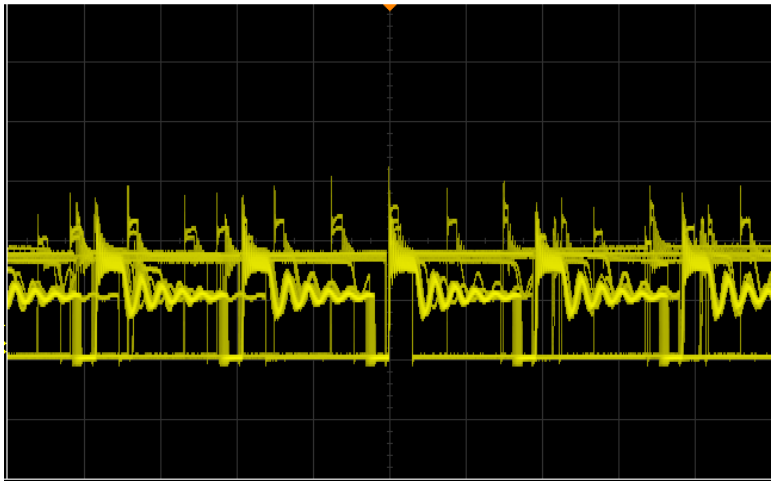


Figura 37: Pulsos en el Transistor con Coeficientes al azar

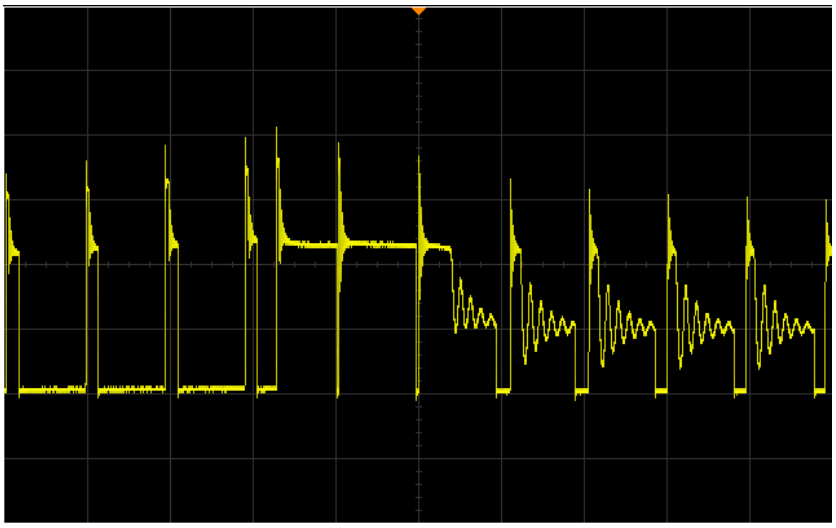


Figura 38: Detalle de los pulsos en el transistor con coeficientes al azar pulsando el STOP del osciloscopio

7. BIBLIOGRAFÍA Y REFERENCIAS

[1] 16-bit Embedded Control Solutions PIC24 Microcontrollers • dsPIC® Digital Signal Controllers.

[2] Gene F. Franklin, J. David Powell, Michael L 'Digital control of dynamic systems' Workman Addison-Wesley,

[3] Section 43. High-Speed PWM DS70323E.

[4] dsPIc33F/24H Family's Reference Manual.Section 41. Interrupts (Part IV).DS70300C

[5] dsPIC33FJ06GS101/X02 and dsPIC33FJ16GSX02/X04 Data Sheets.DS70318F.

[6] MPLAB® C30 C COMPILER USER'S GUIDE. Capitulo 7. Interrupciones. DS51284D

[7] 16-bit MCU and DSC Programmer's Reference Manual High-Performance Microcontrollers (MCU)and Digital Signal Controllers (DSC).

[8] TI embedded processing overview. Presentación de TI del 2010.

[9] TI Developer conference. C2000 roadmap applications.

[10] Section 16. Analog-to-Digital Converter (ADC). DS70183D.

[11] Buck/Boost Converter PICtail™ Plus Daughter BoardUser's Guide

[12]R. Severns and G. E. Bloom, Modern Dc-to-Dc Switchmode Power Converter Circuits, NewYork: Van Nostrand Reinhold, 1985.

[13] Robert W. Erickson, Dragan Maksimovic Fundamentals of Power ElectronicsSpringer Science & Business Media, 31/1/2001.

[14] Keith Billings, Switch Modern Power Supply Handbook. McGraw-Hill 1989.

[15] G.C.Goodwin, S.F. Graebe, M.E. Salgado, "Control System Design," Prentice Hall, 2001.

[16] R. D. Middlebrook, Power electronics: topologies, modeling, and measurement, Proc. IEEE Int. Symp. Circuits Syst., 1981.