Escola Universitària
Politècnica de Mataró

Ingeniería Técnica Industrial: Especialidad Electrónica Industrial

TECNOLOGIA DSP EN UN INVERSOR DE TRES NIVELES CON CONTROL PREDICTIVO

CARLOS FERNÁNDEZ CAMPOS
SALVADOR ALEPUZ

PRIMAVERA 2009

RESUMEN:

El proyecto desarrollado contiene dos objetivos principales:

1. Guía para emplear las herramientas de programación del DSC TMS320F28335.
2. Realización de una aplicación en tiempo real

Para conseguir el primer objetivo se ha realizado un estudio de las herramientas de programación contenidas en el software CCStudio v3.3, desarrollado por Texas Instruments. En el proyecto se ha utilizado un kit de evaluación eZdsp F28335, el cual contiene el Controlador de Señales Digitales (DSC) TMS320F28335, una tarjeta de evaluación con los conectores necesarios para su conexión al ordenador, y un CD con el software y drivers necesarios.

Además se ha desarrollado una aplicación en tiempo real con la tarjeta de evaluación. La aplicación desarrollada realiza el control predictivo de un inversor de tres niveles. Para desarrollar dicha aplicación se ha tenido que realizar la puesta en marcha de la tarjeta de evaluación y crear el hardware necesario para acondicionar las señales de entrada y salida del DSC al inversor de tres niveles.

RESUM:

El projecte desenvolupat conte dos objectius principals:

1. Guia per utilitzar les eines de programació del DSC TMS320F28335.
2. Realització d'una aplicació en temps real

Per aconseguir el primer objectiu s'ha desenvolupat un estudi de les eines de programació contingudes en el software CCStudio v3.3, desenvolupat per Texas Instruments. En el projecte s'ha utilitzat un kit d'avaluació eZdsp F28335, el qual conte el Controlador de Senyals Digitals (DSC) TMS320F28335, una targeta d'avaluació amb els connectores necessaris per la seva connexió a l'ordinador, i un CD amb el software i drivers necessaris.

També s'ha desenvolupat una aplicació en temps real amb la targeta d'avaluació. L'aplicació desenvolupada realitza el control predictiu d'un inversor de tres nivells. Per poder realitzar aquesta aplicació s'ha tingut que fer la posada en marxa de la targeta d'avaluació i crear el hardware necessari per condicionar les senyals d'entrada i sortida del DSC amb l'inversor de tres nivells.

ABSTRACT:

The project developed has two main objectives:

1. User's guide to use programming tools of TMS320F28335 DSC.
2. Making a real time application.

To achieve the first objective, it has been made a study of the programming tools of the software CCStudio v3.3, developed by Texas Instruments.

In this project, it has been used an evaluation kit, which contains the (DSC) TMS320F28335 digital signal controller, an eZdsp F28335 evaluation card with the connectors necessary to connect it to the computer, and a CD with proper software & drivers that are needed.

A real time application has been developed with the evaluation card as well.

The application developed carries out the predictive control of a three-level inverter.

To develop this application, the start-up of the evaluation card has been done together with the design and implementation of a set-up to connect the input and output signals of the DSC to the three-level inverter.

ÍNDICE:

1 OBJETIVO DEL PROYECTO:

Propósito: Realizar una guía de ayuda para la programación de la tecnología DSP de Texas instrumentes implementada en la tarjeta de evaluación eZdsp F28335 y utilizarla en una aplicación de tiempo real como ejemplo de programación.

Finalidad: Poder utilizar las herramientas de programación del software Code Composer Studio sobre la tarjeta de evaluación eZdsp F28335 para usuarios con conocimientos básicos de programación.

Alcance: Realización de una guía de programación para la tarjeta de evaluación eZdsp F28335 y creación del software de programación de un inversor de tres niveles, sin necesidad de realizar el hardware del equipo experimental.

Objeto: Redacción del presente documento que incluye una guía de programación de la tarjeta de evaluación eZdsp F28335, una explicación de las tecnologías utilizadas y una descripción detallada de la aplicación desarrollada con la tarjeta de evaluación eZdsp F28335.

2 TECNOLOGIAS DEL PROYECTO:

2.1 Introducción:

La aplicación en tiempo real desarrollada dentro de este proyecto como ejemplo de programación, tiene la finalidad de realizar el control predictivo de un inversor de tres niveles implementando tecnología DSC (Controlador de Señales Digitales) con la tarjeta de evaluación eZdsp F28335.

La decisión de implementar la aplicación desarrollada sobre la tarjeta de evaluación eZdsp F28335 viene dada por el conjunto de recursos que la caracterizan. Estas características se describen en el apartado 2.2 del presente capítulo. La falta de experiencia de los programadores con conocimientos básicos sobre la utilización de las herramientas básicas de la tarjeta de evaluación eZdsp F28335, genera la necesidad de crear una guía de usuario de programación. Esta guía se incluye en el capítulo 3.

En el desarrollo de la aplicación es necesarios conocer las tecnologías empleadas para realizar el control del sistema. Las tecnologías empleadas en la aplicación se dividen en dos partes, tecnología del control y tecnología del equipo experimental. Estas tecnologías se describen en el apartado 2.3 del presente capítulo. La tecnología del control del sistema se denomina control predictivo, y el equipo experimental usado para su implementación es un inversor de tres niveles. La funcionalidad del control predictivo sobre el inversor de tres niveles, es realizar predicciones dentro de cada tiempo de conmutación. A diferencia de otros sistemas más comunes en el control de inversores de potencia, el control predictivo no sigue una secuencia marcada, sino que realiza predicciones para el siguiente estado de conmutación del inversor.

La secuencia del control predictivo implementada en el inversor de tres niveles es la siguiente:

- Adquisición de las tensiones y corrientes del inversor de tres niveles.
- Realización de la predicción para el siguiente estado de conmutación.
- Aplicación del estado de conmutación.

2.2 Tarjeta de evaluación eZdsp F28335:

La tarjeta de evaluación eZdsp F28335 (Figura 2.2.1) permite desarrollar y ejecutar el funcionamiento de aplicaciones en tiempo real. Esta tarjeta de evaluación se suministra con el controlador de señales digitales DSC TMS320F28335.

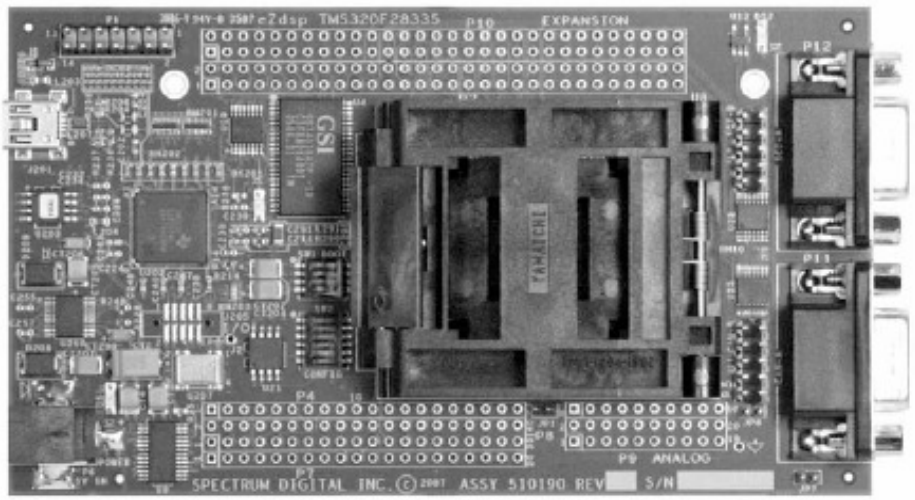

Figura 2.2.1 Tarjeta de evaluación eZdsp F28335.

La tarjeta de evaluación eZdsp F28335 contiene las siguientes características:

Características hardware:

- DSC TMS320F28335
- Unidad de punto flotante de 32 bits
- 68Kb de RAM
- Chip de memoria Flash de 512Kb
- Chip de memoria SRAM de 256 Kb
- Convertidor ADC de 12 bits con 16 canales de entrada
- Reloj de entrada 30 MHz
- Conector RS-232 con línea de drivers.
- Interface CAN 2.0 con línea de drivers y conector.
- Conector de expansión múltiple
- Controlador USB JTAG embedded.
- Funcionamiento a 5 voltios suministrados por adaptador AC
- Conector de emulador IEEE 1149.1 JTAG en placa.

Características software:

- TI F28xx Code Composer Studio version 3.3.
- Soporte F28335 Flash APIs.
- Archivos de encabezamientos y ejemplos de software F28335.

En la figura 2.2.2 se muestra un bloque de diagrama de la configuración de la tarjeta de evaluación eZdsp F28335.

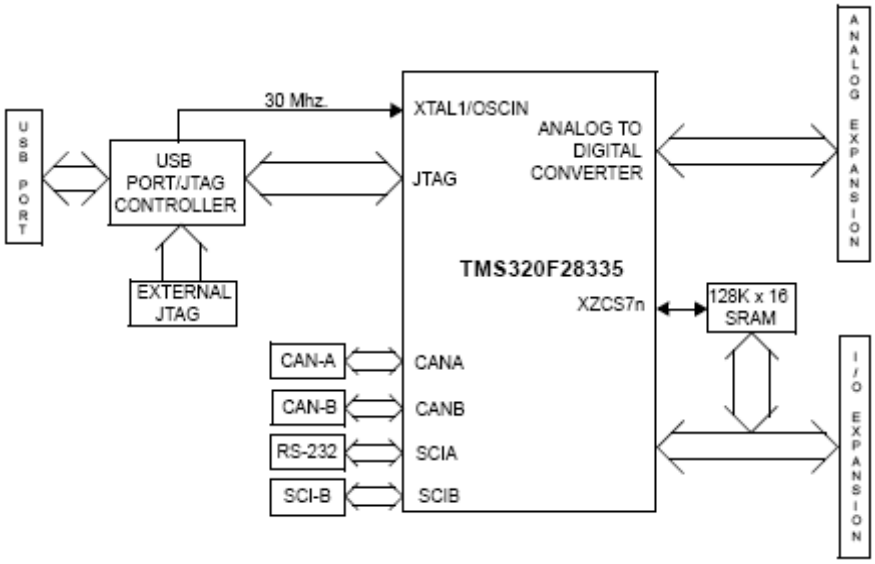

Figura 2.2.2 Diagrama de Bloques eZdsp F28335.

La tarjeta de evaluación eZdsp F28335 está compuesta por diferentes recursos. Estos recursos se describen a continuación mostrándose en las figuras 2.2.3 y 2.2.4.

- P1, Interface JTAG (emulador).
- P2, Interface de expansión.
- P4/P8/P7, Interface de I/O digitales.
- P5/P9, Interface Input Analógica.
- P6, Conector de alimentación (5V).
- P10, Interface de expansión.
- P11, Conector CANA.
- P12, Conector RS-232.
- J11, Conector CANB.
- J12, Conector SCIB.
- J201, Interface Embedded USB JTAG.
- JP1, Selector de ADCREFIN.

- JR2, Selector de voltaje XTPD.
- JR4, Selector de voltaje para conector P4 y P8
- JR5, Selector de voltaje para conector P2 y P10
- JR6, Selector MUX GPIO22_24
- JP7, Selector de resistencia de terminación de CANA
- JP8, Selector de resistencia de terminación de CANB
- SW1, Interruptor de selección Boot.
- SW2, Interruptor de configuración del procesador
- DS1, DS201, DS2, Leds

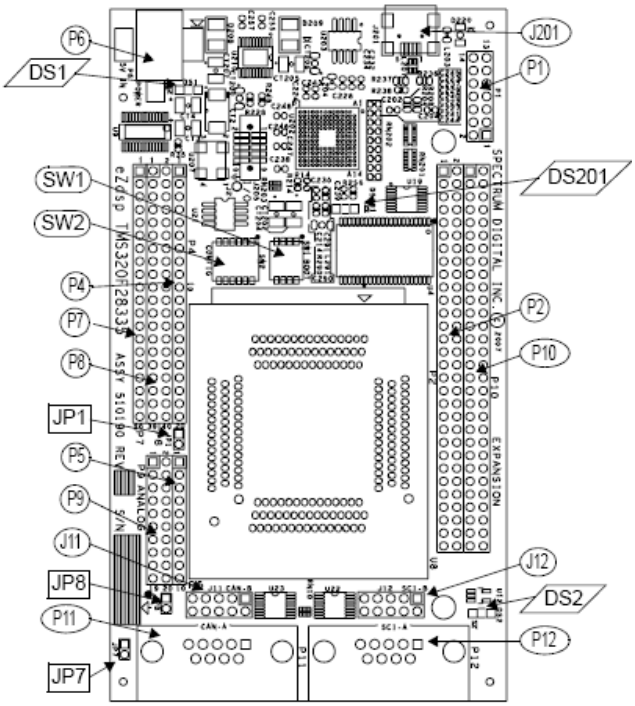

Figura 2.2.3: eZdsp F28335 PCB (Top)

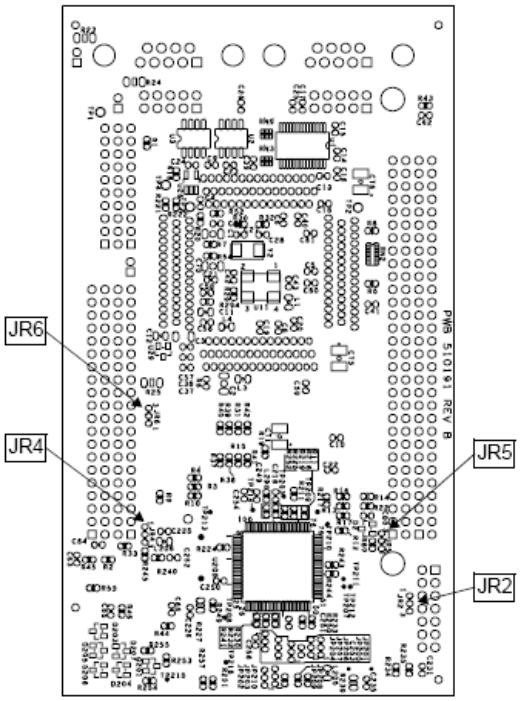

Figura 2.2.4: eZdsp F28335 PCB (Bottom)

Una Información más detallada sobre la tarjeta de evaluación eZdsp F28335 se encuentra en el archivo *ezdspf28335.pdf* del CD que incluye este documento.

Esta tarjeta de evaluación contiene el Controlador de Señales Digitales (DSC) TMS320F28335. Los controladores de Texas Instruments se clasifican por familias dependiendo de su numeración, el DSC TMS320F28335 pertenece a la familia de procesadores C2000. Los controladores de señales digitales son diseñados para realizar soluciones de alto rendimiento en aplicaciones de control exigentes.

El DSC TMS320F28335 contiene 176 pines (Figura 2.2.5). Este controlador esta conectado en la tarjeta de evaluación eZdsp F28335 a partir de un zócalo de presión. Este zócalo permite el cambio del controlador sin necesidad de realizar grandes esfuerzo.

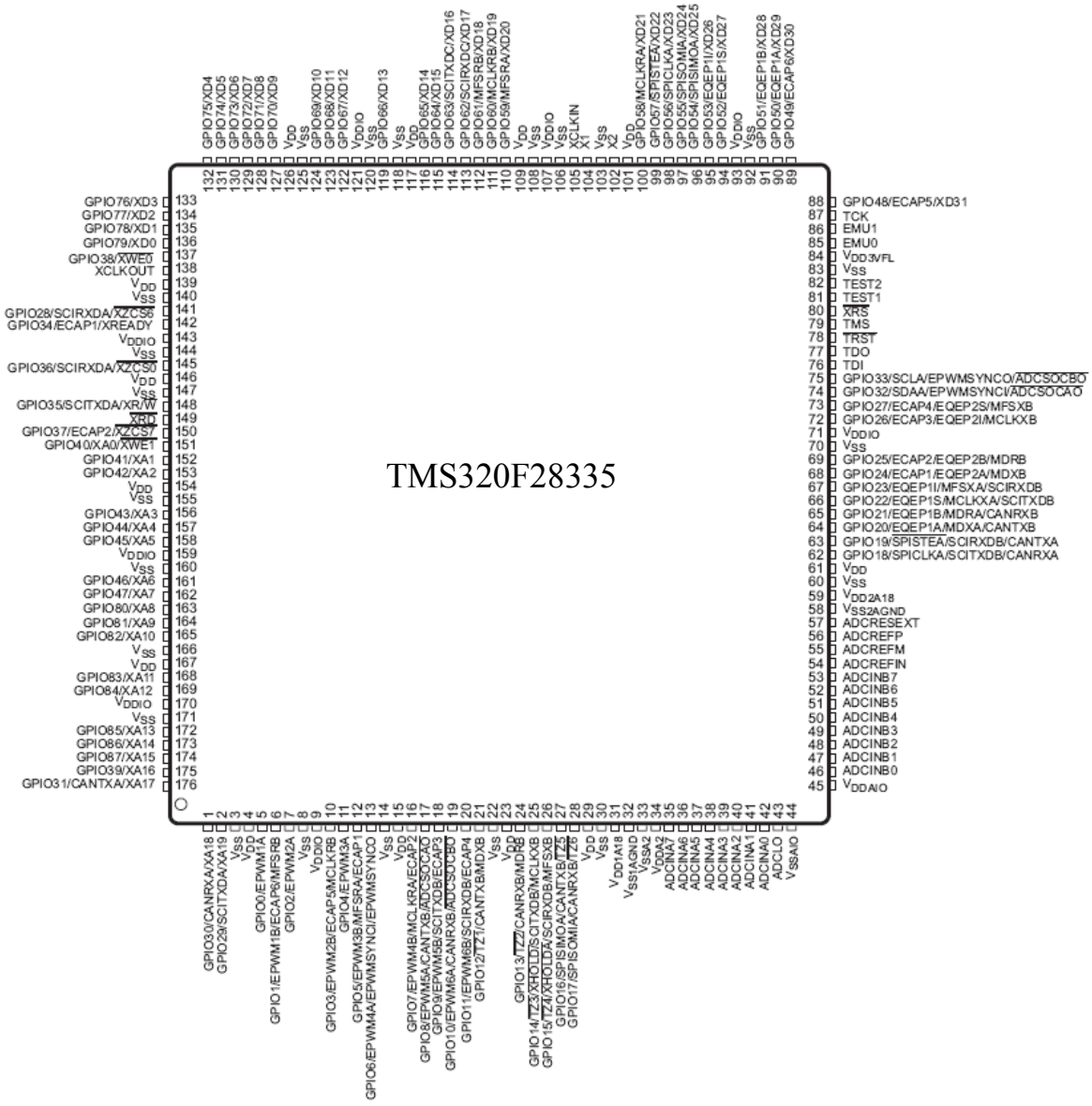

Figura 2.2.5: Pines del controlador TMS320F28335.

Las características que contiene el DSC TMS320F28335 son las siguientes:

- Velocidad de operación 150 MHz
- CPU de 32 bits.
- Controlador DMA de seis canales.
- Interface externo de 16 o 32 bits.
- Memoria 256K x 16 Flash, 34K x 16 SARAM.
- 1K x 16 OTP ROM.
- BOOT ROM (8K x 16).
- Control del sistema y relojes.
- Pines I/O (GPIO0 a GPIO63).

- Expansión interrupciones de periféricos.
- Código de seguridad de 128 bits.
- 18 salidas PWM.
- 6 salidas HRPWM con 150ps de resolución MEP.
- 3 temporizadores de la CPU de 32 bits.
- 3 puertos periféricos.
- ADC de 12 bits con 16 canales.

La funcionalidad del sistema interno del controlador DSC TMS320F28335 se muestra en la figura 2.2.6.

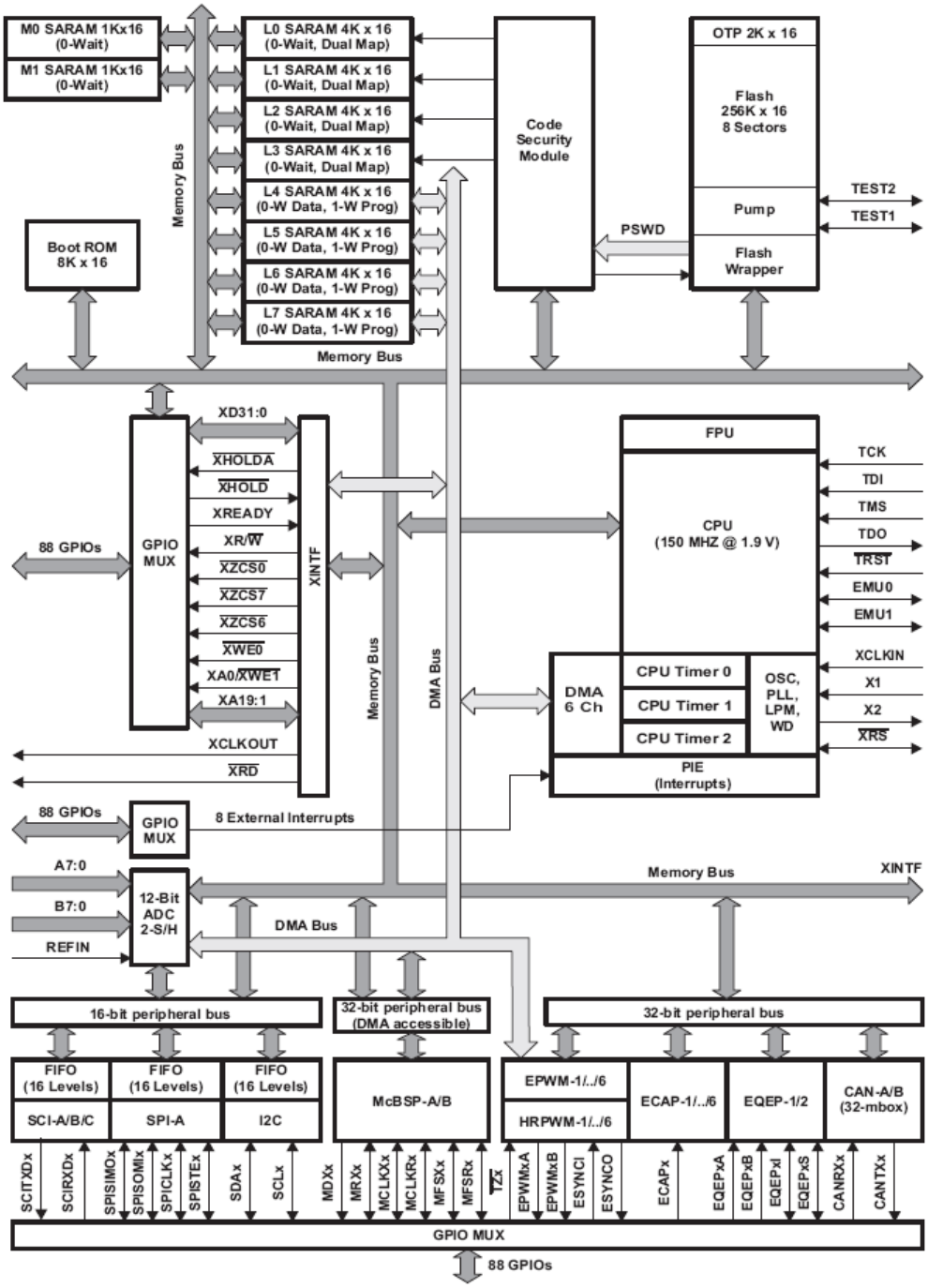

Figura 2.2.6: Diagrama de bloques del sistema interno de DSC TMS320F28335.

El mapa de memoria y la estructura del sistema son muy importantes a la hora de utilizar los recursos de los Controladores de Señales Digitales (DSCs). Una buena utilización de los recursos de DSC TMS320F28335 requiere el entendimiento de la estructura interna del DSC junto el mapa de memoria del mismo. El mapa de memoria de DSC TMS320F28335 se muestra en la figura 2.2.7.

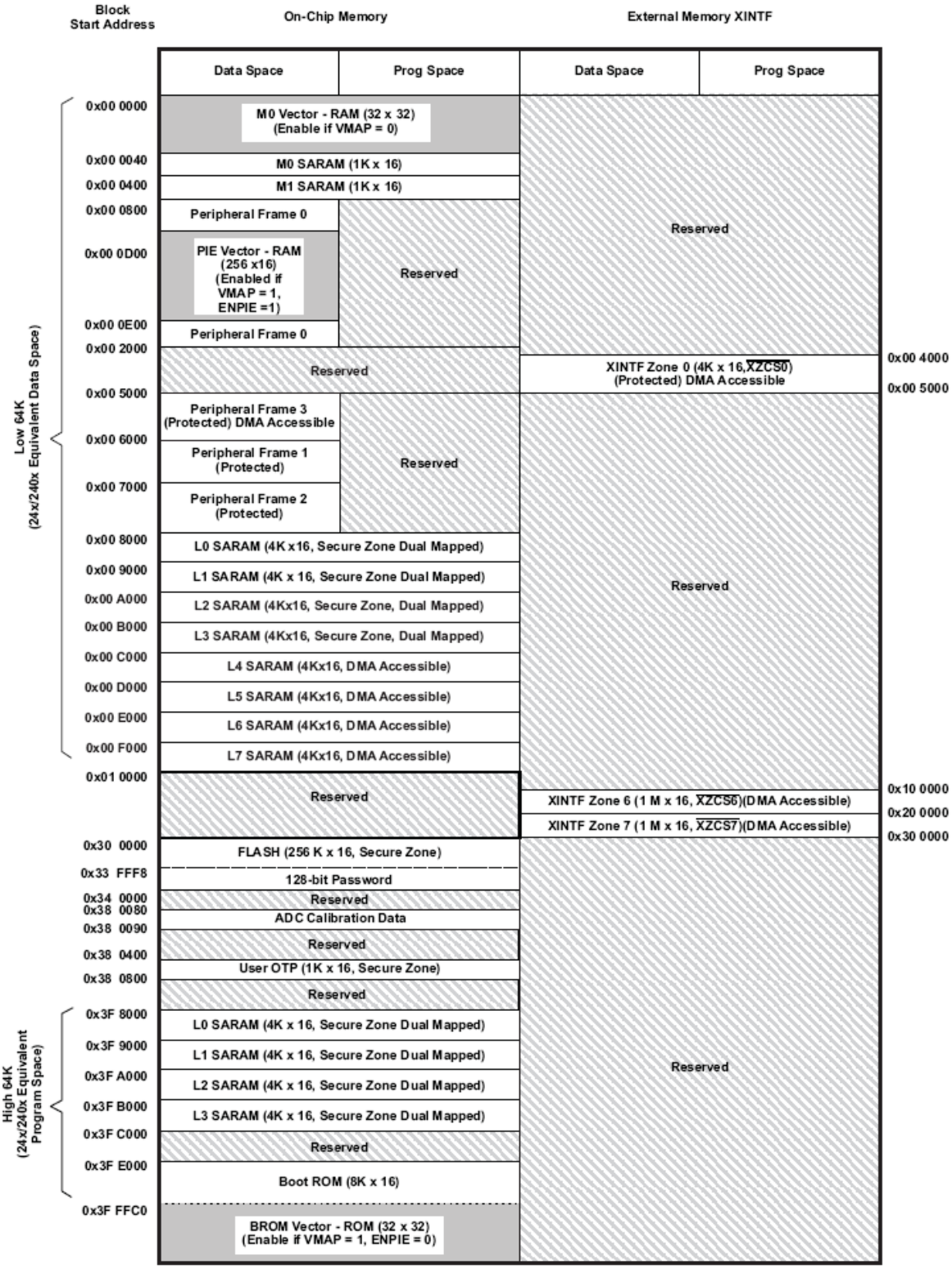

Figura 2.2.7: Mapa de memoria DSC TMS320F28335.

En la programación de los controladores de señales digitales (DSCs) es necesario un archivo de código con la extensión *.cmd.* Este archivo contiene la descripción del mapa de memoria del DSC utilizado, y define cual es el uso de cada módulo de memoria en la ejecución de la aplicación desarrollada.

Una Información más detallada sobre el Procesador de Señales Digitales TMS320F28335 se encuentra en el archivo *Data_manual_tms320f28334.pdf* del CD que incluye este documento.

Los recursos de la tarjeta de evaluación eZdsp F28335 utilizados en el desarrollo de la aplicación de tiempo real, se clasifican en los siguientes módulos:

- Sistema de control e interrupciones del DSC TMS320F28335.
- Módulo Convertidor ADC de DSC TMS320F28335.
- Módulo ePWM de DSC TMS320F28335.

Cualquier aplicación que se realice en tiempo real es necesaria la utilización de interrupciones del sistema. Las interrupciones permiten la interacción de un sistema físico (hardware) con un sistema de control (software). Gracias a las interrupciones es posible realizar aplicaciones reales con procesadores.

El controlador DSC TMS320F28335 contiene 17 vectores de interrupción a nivel de CPU. También contiene periféricos que pueden realizar más de una interrupción. El control de las interrupciones de sus periféricos se realiza a partir del módulo PIE, este módulo centraliza y arbitra las preferencias de las peticiones de interrupción hacia la CPU. La figura 2.2.8 muestra la configuración de las interrupciones del DSC TMS320F28335.

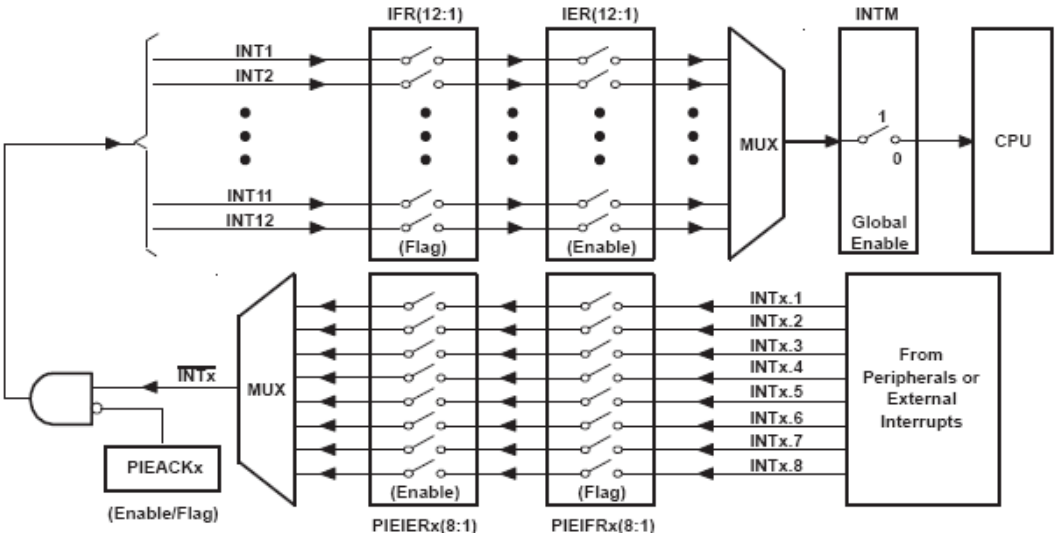

Figura 2.2.8: Diagrama de bloques de las interrupciones DSC TMS320F28335.

Las peticiones de interrupción hacia la CPU esta multiplexada con un total de 12 vectores de interrupción desde INT1 a INT12 (Figura 2.2.8). Los 12 vectores de interrupción de la CPU son nuevamente multiplexadas por el modulo PIE que se compone de 8 vectores de interrupción. De esta forma se obtiene 8 vectores de interrupción de periféricos por cada vector de interrupción de la CPU, con un total de 96 vectores interrupción de periféricos (12 * 8 = 96 vectores). La figura 2.2.9 muestra la tabla de vectores de interrupción del módulo PIE.

	INTx.8	**INTx.7**	**INTx.6**	**INTx.5**	**INTx.4**	**INTx.3**	**INTx.2**	**INTx.1**
INT1.y	*WAKEINT*	*TINT0*	*ADCINT*	*XINT2*	*XINT1*	Reserved	*SEQ2INT*	*SEQ1INT*
	(LPM/WD)	(TIMER 0)	(ADC)	Ext. int. 2	Ext. int. 1	-	(ADC)	(ADC)
	0xD4E	0xD4C	0xD4A	0xD48	0xD46	0xD44	0xD42	0xD40
INT2.y	Reserved	Reserved	*EPWM6_ TZINT*	*EPWM5_TZINT*	*EPWM4_TZINT*	*EPWM3_TZINT*	*EPWM2_TZINT*	*EPWM1_TZINT*
	-	-	(ePWM6)	(ePWM5)	(ePWM4)	(ePWM3)	(ePWM2)	(ePWM1)
	0xD5E	0xD5C	0xD5A	0xD58	0xD56	0xD54	0xD52	0xD50
INT3.y	Reserved	Reserved	*EPWM6_ INT*	*EPWM5_INT*	*EPWM4_INT*	*EPWM3_INT*	*EPWM2_INT*	*EPWM1_INT*
	-	-	(ePWM6)	(ePWM5)	(ePWM4)	(ePWM3)	(ePWM2)	(ePWM1)
	0xD6E	0xD6C	0xD6A	0xD68	0xD66	0xD64	0xD62	0xD60
INT4.y	Reserved	Reserved	*ECAP6_INT*	*ECAP5_INT*	*ECAP4_INT*	*ECAP3_INT*	*ECAP2_INT*	*ECAP1_INT*
	-	-	(eCAP6)	(eCAP5)	(eCAP4)	(eCAP3)	(eCAP2)	(eCAP1)
	0xD7E	0xD7C	0xD7A	0xD78	0xD76	0xD74	0xD72	0xD70
INT5.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	*EQEP2_INT*	*EQEP1_INT*
	-	-	-	-	-	-	(eQEP2)	(eQEP1)
	0xD8E	0xD8C	0xD8A	0xD88	0xD86	0xD84	0xD82	0xD80
INT6.y	Reserved	Reserved	*MXINTA*	*MRINTA*	*MXINTB*	*MRINTB*	*SPITXINTA*	*SPIRXINTA*
			(McBSP-A)	(McBSP-A)	(McBSP-B)	(McBSP-B)	(SPI-A)	(SPI-A)
	0xD9E	0xD9C	0xD9A	0xD98	0xD96	0xD94	0xD92	0xD90
INT7.y	Reserved	Reserved	*DINTCH6*	*DINTCH5*	*DINTCH4*	*DINTCH3*	*DINTCH2*	*DINTCH1*
			(DMA6)	(DMA5)	(DMA4)	(DMA3)	(DMA2)	(DMA1)
	0xDAE	0xDAC	0xDAA	0xDA8	0xDA6	0xDA4	0xDA2	0xDA0
INT8.y	Reserved	Reserved	*SCITXINTC*	*SCIRXINTC*	Reserved	Reserved	*I2CINT2A*	*I2CINT1A*
	-	-	(SCI-C)	(SCI-C)	-	-	(I2C-A)	(I2C-A)
	0xDBE	0xDBC	0xDBA	0xDB8	0xDB6	0xDB4	0xDB2	0xDB0
INT9.y	*ECAN1INTB*	*ECAN0INTB*	*ECAN1INTA*	*ECAN0INTA*	*SCITXINTB*	*SCIRXINTB*	*SCITXINTA*	*SCIRXINTA*
	(CAN-B)	(CAN-B)	(CAN-A)	(CAN-A)	(SCI-B)	(SCI-B)	(SCI-A)	(SCI-A)
	0xDCE	0xDCC	0xDCA	0xDC8	0xDC6	0xDC4	0xDC2	0xDC0
INT10.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	0xDDE	0xDDC	0xDDA	0xDD8	0xDD6	0xDD4	0xDD2	0xDD0
INT11.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	0xDEE	0xDEC	0xDEA	0xDE8	0xDE6	0xDE4	0xDE2	0xDE0
INT12.y	*LUF*	*LVF*	Reserved	*XINT7*	*XINT6*	*XINT5*	*XINT4*	*XINT3*
	(FPU)	(FPU)		Ext. Int. 7	Ext. Int. 6	Ext. Int. 5	Ext. Int. 4	Ext. Int. 3
	0xDFE	0xDFC	0xDFA	0xDF8	0xDF6	0xDF4	0xDF2	0xDF0

Figura 2.2.9.: Tabla vectores de interrupción del módulo PIE.

Las prioridades de los vectores de interrupción del controlador DSC TMS320F28335 abarcan desde prioridad 1 hasta prioridad 19. Las interrupciones que contengan un valor más bajo de prioridad serán las primeras en ser servidas por la CPU. Más información del sistema de control e interrupciones del DSC TMS320F28335 se incluye en el archivo *System control and interrupts.pdf* del CD que incluye este documento.

En la aplicación desarrollada del inversor de tres niveles se implementa el vector de interrupción SEQ1INT del módulo ADC con prioridad 5, y el vector de interrupción EPWM3_INT del módulo ePWM con prioridad 6.

El vector de interrupción SEQ1INT del módulo ADC pertenece al grupo de interrupciones INT1 de la CPU. El vector SEQ1INT se muestra en la figura 2.9 en la posición INT1.1 de la tabla de vectores. Este vector realiza una petición de interrupción cuando se finaliza la conversión de las 8 entradas analógicas del convertidor ADC utilizadas en la adquisición de las señales del inversor de tres niveles.

El vector de interrupción EPWM3_INT del módulo ePWM pertenece al grupo de interrupciones INT3 de la CPU. El vector EPWM3_INT se muestra en la figura 2.2.9 en la posición INT3.3 de la tabla de vectores. Este vector realiza una petición de interrupción cuando el tiempo modulación de la salida ePWM3 finaliza.

El módulo ADC de DSC TMS320F28335 es un recurso de la tarjeta de evaluación eZdsp F28335 utilizado en la adquisición de las señales de control del inversor de tres niveles. Este módulo está compuesto por dos convertidores independientes de 8 canales cada uno (ADCINA[0-7] y ADCINB[8-15]). Los dos convertidores que componen el módulo ADC son configurables en modo cascada, obteniendo un total de 16 canales para una sola conversión. La figura 2.2.10 muestra el bloque de diagrama del módulo ADC.

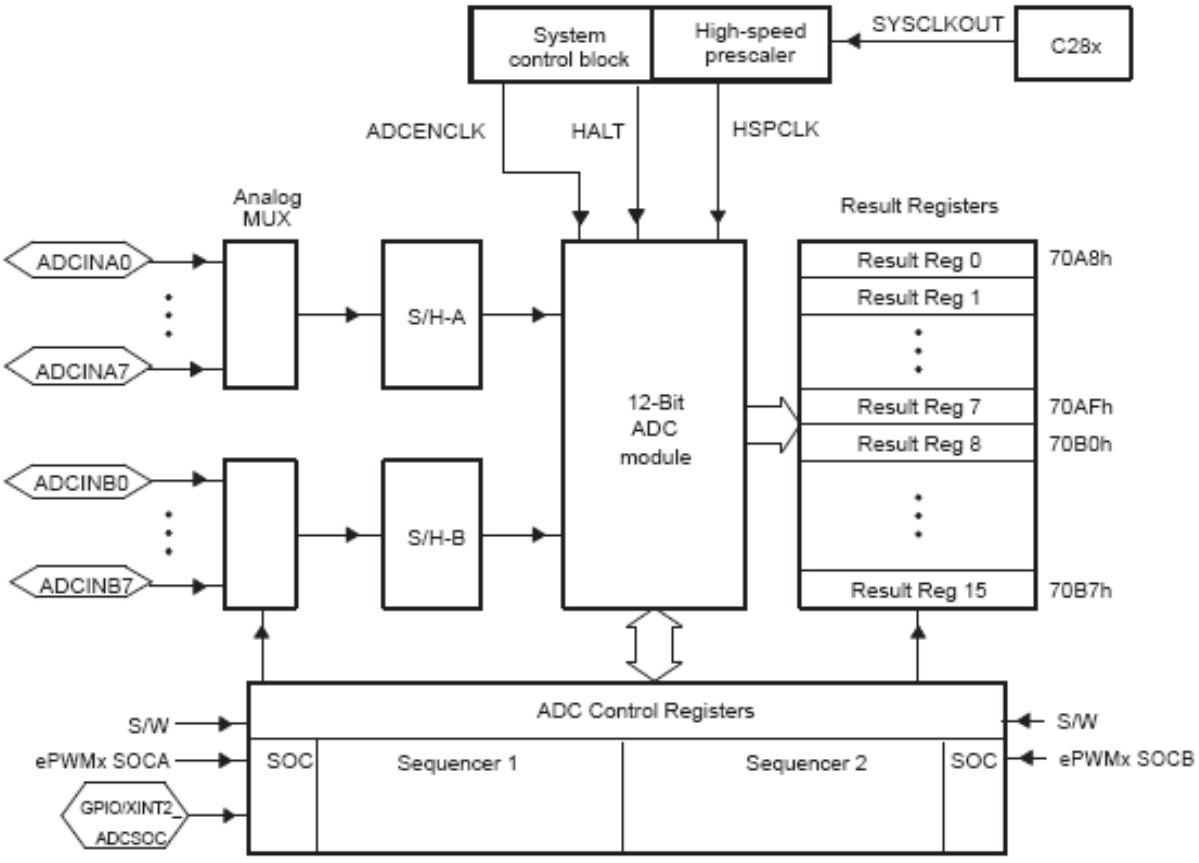

Figura 2.2.10.: Diagrama de bloques módulo ADC.

El módulo ADC realiza las conversiones de las señales de entrada con una resolución de 12 bits (valor máximo 0x0FFFh). El rango de tensión analógica de entrada que soporta cada canal es de 0V a 3V. La frecuencia del reloj de conversión de este módulo es de 12.5 MHz, esta frecuencia es variable a partir de un registro de configuración.

Este módulo contiene 16 registros independientes de memoria para el almacenamiento de los resultados de la conversión de los valores adquiridos.

La configuración del modo de adquisición y conversión de los canales del módulo ADC abarca tres configuraciones posibles. El primer modo de configuración se realiza por software, a partir de una instrucción se indica el disparo de comienzo de adquisición de las señales de entrada y conversión de las mismas. El segundo modo de configuración se realiza a partir de una señal de interrupción del módulo ePWM. Y el tercer modo de configuración se realiza a partir de la señal de interrupción XINT2, esta señal de interrupción contiene un pin de entrada en el puerto de los periféricos de la tarjeta de evaluación eZdsp F28335.

Las peticiones de interrupción de final de conversión (EOS) de los canales del módulo ADC se pueden configurar en dos modos diferentes. El primer modo de configuración es realizar una petición de interrupción en cada conversión finalizada. El segundo modo de configuración es realizar una petición de interrupción después de finalizar varias conversiones. En la aplicación desarrollada dentro de este proyecto la configuración utilizada para las peticiones de interrupción del módulo ADC ha sido realizar una petición de interrupción cada una conversión finalizada, de esta manera cada vez que se finaliza una conversión de los 8 canales que forman un convertidor del módulo ADC, se realiza una petición de interrupción hacia la CPU.

El módulo ADC necesita una tensión de referencia para realizar las conversiones de los valores adquiridos. La configuración de esta tensión de referencia tiene dos modos, puede ser configurada como tensión externa o tensión interna. La configuración de la tensión de referencia en modo interna, significa que el módulo ADC adquiere el valor de esta tensión a partir del valor cargado en un registro de memoria. La configuración de la tensión de referencia en modo externa, significa que el módulo ADC adquiere el valor de esta tensión a partir de una entrada externa del DSC TMS320F28335, este DSC soporta tres valores de tensiones de referencia distintas: 2.048V, 1.5V y 1.024V. La figura 2.2.11 muestra el esquema de conexionado de la tensión de referencia de la tarjeta eZdsp F28335.

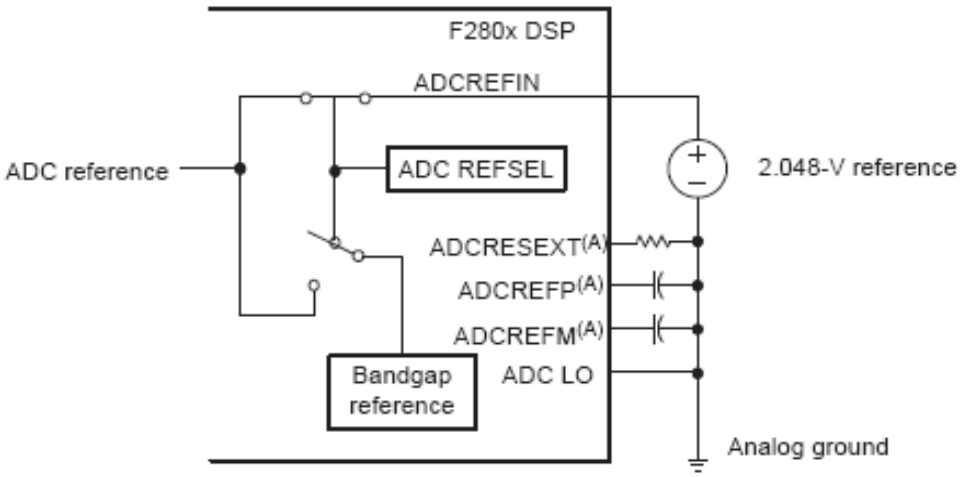

Figura 2.2.11.: Esquema de tensión de referencia del módulo ADC (eZdsp F28335).

En aplicaciones reales que utilizan convertidores ADC es necesaria la realización de una calibración inicial de los canales del módulo ADC utilizados. La calibración de los canales utilizados en la aplicación real desarrollada dentro de este proyecto, se realiza a partir de un registro de memoria. Este registro ajusta el resultado de la conversión aplicando un offset.

La calibración inicial de los canales del módulo ADC utilizados, se realiza conectando estos a la masa de la tarjeta de evaluación eZdsp F28335 (GND). Seguidamente se realiza una conversión de los canales conectados a masa (GND). El resultado de la conversión de estos canales debe de ser 0 (0x0000h) ya que la tensión de entrada de los canales corresponde a la masa del circuito de la tarjeta de evaluación. Si se obtiene un valor diferente a 0 se puede corregir cargando un valor al registro de offset. El valor cargado en el registro de offset se sumará o se restará al resultado obtenido de la conversión. De esta manera se obtiene una calibración de los canales del módulo ADC a 0 (0x0000h). Más información del módulo ADC de DSC TMS320F28335 se incluye en el archivo *Analog-to-Digital Converter.pdf* que incluye este documento.

La configuración del módulo ADC sobre la tarjeta de evaluación eZdsp F28335 en el desarrollo de la aplicación del control del inversor de tres niveles, es la siguiente:

- Utilización de 8 canales del convertidor **A** del módulo ADC.
- Configuración de la tensión de referencia interna.
- Interrupción de final de conversión (EOC) cada una conversión finalizada.
- Calibración de entradas analógicas del ADC sin utilizar Offset.
- Reloj del ADC a 12.5 MHz.

El módulo ePWM de DSC TMS320F28335 está compuesto por 6 canales PWM. Cada PWM está formado por dos salidas: ePWMAx y ePWMBx. En la figura 2.2.12 se muestra el módulo ePWM.

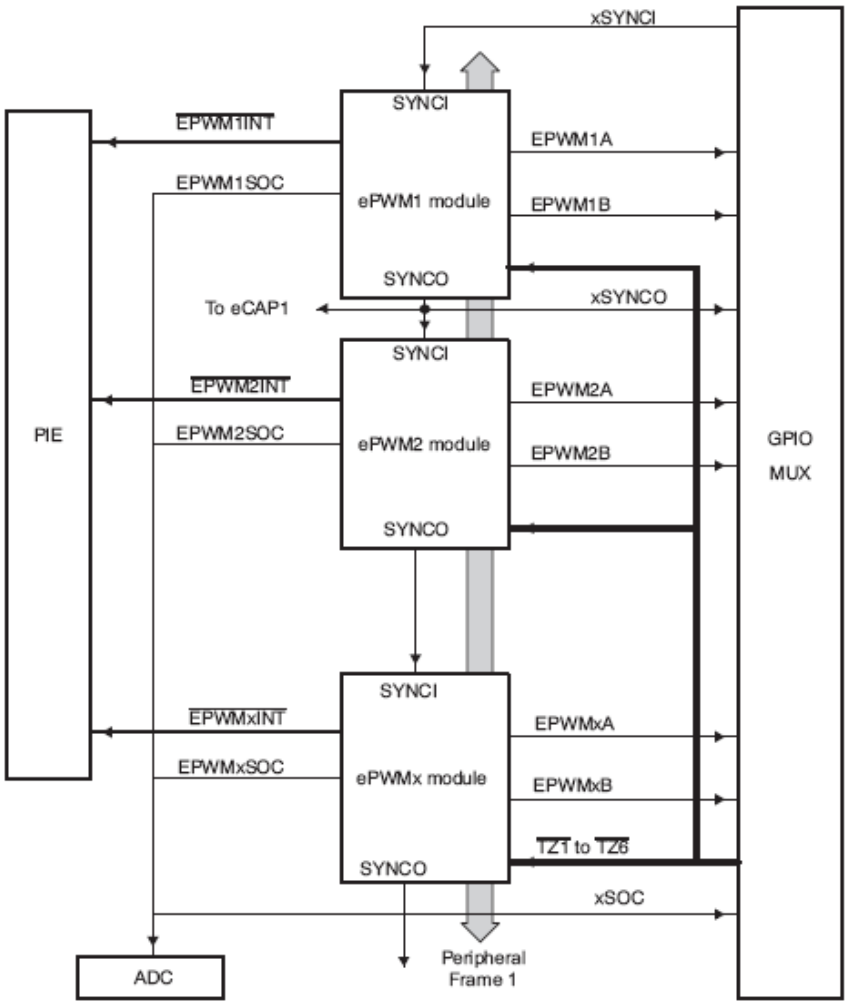

Figura 2.2.12.: Módulo ePWM

El módulo ePWM contiene un registro de memoria para la configuración de la base de tiempo de los canales PWM. Esta base de tiempo tiene una resolución de 16 bits y es configurable e independiente para cada canal PWM.

Las 12 salidas ePWM (ePWMAx y ePWMbx) de los 6 canales PWM pueden ser configuradas en diferentes modos. Un modo de configuración posible es configurar las dos salidas ePWM de un mismo canal PWM como dos salidas simétricas. Otro modo de configuración es configurar las dos salidas ePWM de un mismo canal PWM como dos salidas asimétricas. Y el último modo de configuración es configurar las dos salidas ePWM de un mismo canal PWM como dos salidas independientes.

El módulo ePWM contiene registros para la configuración de la banda muerta entre los cambios de conmutaciones de los 6 canales PWM. Esta configuración genera un tiempo de espera entre la conmutación actual y la siguiente conmutación. En una aplicación real el tiempo de espera entre conmutaciones llamado como banda muerta es crucial para el buen funcionamiento del sistema que se desea controlar.

Este módulo contiene registros llamados comparadores. Estos comparadores son independientes para cada canal ePWM. A partir de la configuración de estos registros se controla los cambios de conmutación de las salidas ePWM. La figura 2.12 muestra el diagrama de bloques del módulo ePWM del DSC TMS320F28335. Más información del módulo ePWM se incluye en el archivo *Enhanced Pulse Width Modulator (ePWM) Module.pdf* que incluye este documento.

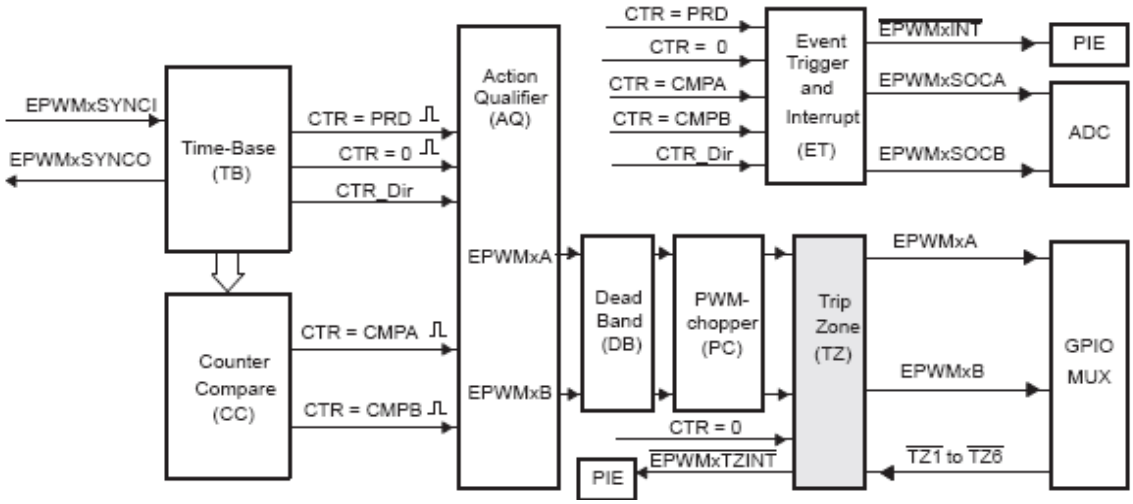

Figura 2.2.13: Diagrama de bloques del módulo ePWM del DSC TMS320F28335.

En la aplicación real del inversor de tres niveles se ha utilizado los canales del módulo ePWM correspondientes del 1 al 3 (ePWM1A, ePWM1B, ePWM2A, ePWM2B, ePWM3A y ePWM3B). El modo de configuración utilizado para estas salidas ha sido el modo de salidas independientes. Las 6 salidas ePWM controlan la conmutación de los IGBTs del inversor de tres niveles.

En la aplicación del inversor de tres niveles no ha sido necesaria la configuración de la banda muerta de las salidas ePWM. La configuración de la banda muerta en la aplicación se ha realizado a partir de una placa externa a la tarjeta de evaluación eZdsp F28335. Esta placa además de generar la banda muerta entre los estados de conmutación, genera 12 salidas ePWM a partir de las 6 salidas de la tarjeta de evaluación.

2.3 Inversor de tres niveles con control predictivo:

Los inversores de tres niveles son utilizados en muchas aplicaciones industriales para la conversión y conducción de potencia.

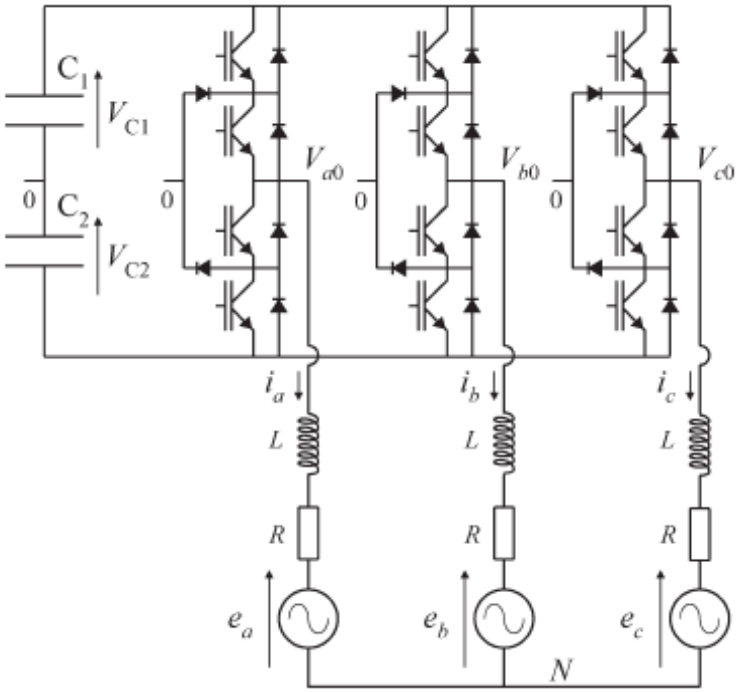

Figura 2.3.1: Esquema eléctrico de un inversor de tres niveles conectado a una carga activa R-L.

El equilibrio del punto neutro en esta topología es un problema que ha sido estudiado en las últimas décadas. Los métodos más utilizados en este tipo de tipologías son los métodos lineales como el uso de controladores Integrales Proporcionales (PI) conjuntamente con una modulación de ancho de pulso (PWM).

El control predictivo es una teoría de control que fue desarrollada a finales de 1970s. La variación de este tipo de estrategia de control permite realizar aplicaciones de control en convertidores de potencia. El control predictivo se utiliza en control de corriente, en la corrección del factor de potencia y en filtros activos. Todos estos controles son considerados modelos lineales y usados en técnicas de modulación para la generación de voltaje. La idea básica del control predictivo es considerar el inversor de tres niveles como un sistema lineal, esta consideración es contraria a su naturaleza discreta y su proceso de control.

El control predictivo es una estrategia de control que consiste en generar predicciones a partir de un modelo del sistema. Las predicciones realizadas en el modelo del sistema son evaluadas a partir de una función de calidad. Esta función de calidad minimiza la aplicación de los diferentes estados del inversor de tres niveles, teniendo en cuenta la naturaleza discreta del mismo.

En el modelo del sistema del inversor de tres niveles se ha considerado una carga activa RL (Figura 2.3.1). Esta carga representa una de las aplicaciones más comunes en la industria, representa una máquina de inducción. Este modelo de sistema abarca aplicaciones como la conexión del inversor de tres niveles a la red o realizar la conexión de cargas pasivas en la salida del inversor.

El modelo del sistema del inversor de tres niveles está compuesto por 19 vectores de tensión. Estos 19 vectores de tensión contienen 27 estados de corriente. Los vectores del sistema están representados en la figura 2.3.2

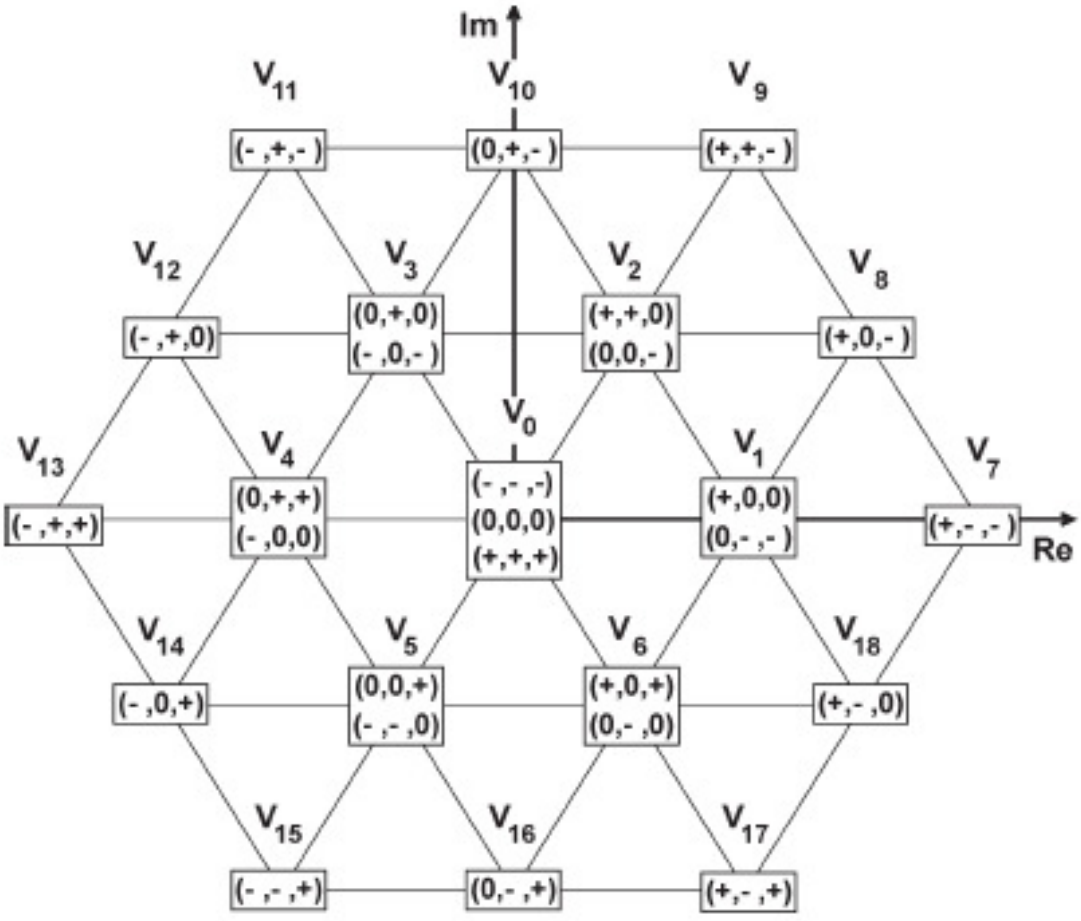

Figura 2.3.2.: Vectores de tensión y estados posibles de corriente de un inversor de tres niveles.

El control predictivo consiste en realizar predicciones de la corriente para el estado siguiente a aplicar en el inversor de tres niveles. En el modelo del sistema del inversor de tres niveles mostrado en la figura 2.3.2 corresponde a la ecuación de una carga activa, inductiva y resistiva de tres fases que cumple la ecuación (1).

$$L\frac{d\mathbf{i}(t)}{dt} = \mathbf{v}(t) - R\mathbf{i}(t) - \mathbf{e}(t) \qquad (1)$$

Las variables **R** y **L** son el valor de las cargas resistiva e inductiva, respectivamente. La variable **v** es el vector de voltaje generado en el inversor de tres niveles. La variable **i** es el vector de corriente de carga y la variable **e** es la fuerza electromotriz (EMF) de la carga. Las variables **v**, **i** y **e** son vectores que corresponden a las funciones (2), (3) y (4), respectivamente.

$$\mathbf{v} = \frac{2}{3}(V_{a0} + aV_{b0} + a^2V_{c0}) \qquad (2)$$

$$\mathbf{i} = \frac{2}{3}(i_a + ai_b + a^2i_c) \qquad (3)$$

$$\mathbf{e} = \frac{2}{3}(e_a + ae_b + a^2e_c) \qquad (4)$$

La variable **a** es igual a: $a = e^{j(2\pi/3)}$.

Aplicando un periodo de muestreo Ts en el modelo del sistema del inversor de tres niveles que corresponde a la función (1), la derivada **di(t)/dt** se puede aproximar a la función (5).

$$\frac{d\mathbf{i}(t)}{dt} \approx \frac{\mathbf{i}(k) - \mathbf{i}(k-1)}{T_s}. \qquad (5)$$

Aplicando la aproximación (5) en la función (1) se obtiene una nueva función (6) en un tiempo discreto Ts.

$$\mathbf{i}(k+1) = \frac{T_s}{RT_s + L}\left[\frac{L}{T_s}\mathbf{i}(k) + \mathbf{v}(k+1) - \mathbf{e}(k+1)\right]. \qquad (6)$$

A partir de la función (6) se obtiene predicciones del siguiente valor de la corriente de carga **i(k+1)**, para el siguiente estado Ts. En la función (6) se considera todas las

posibilidades de los vectores **v** generados en el inversor, y medidas de corriente obtenidas en el estado de muestreo actual **k**.

La estrategia de control a su vez realiza estimaciones de la siguiente corriente de referencia. Dependiendo del tiempo de muestreo (Ts) aplicado y de la restricción computacional, la estimación puede ser obtenida en una extrapolación de segundo orden mostrada en la función (7).

$$\mathbf{i}^*(k+1) = 3\mathbf{i}^*(k) - 3\mathbf{i}^*(k-1) + \mathbf{i}^*(k-2) \qquad (7)$$

La función (7) se puede considerar **i*(k+1) ≈ i*(k)**, de esta manera el tiempo de muestreo y los esfuerzos computacionales son más pequeños. La realización de la consideración **i*(k+1) ≈ i*(k)** evita la realización de la extrapolación de la función (7).

La función (6) requiere una estimación de la siguiente carga de fuerza electromotriz (EMF) e(k+1). La estimación de e(k+1) valúa cual es el siguiente caso analógico de la corriente de referencia, esta evaluación dependerá del tiempo de muestreo (Ts) y el programa utilizado para su implementación. Las estimaciones de **e** del presente estado cambiante y anteriores estados se obtienen en base tiempo en la función (6) y en carga de corriente en la función (8).

$$\hat{\mathbf{e}}(k) = \mathbf{v}(k) + \frac{L}{T_s}\mathbf{i}(k-1) - \frac{RT_s + L}{T_s}\mathbf{i}(k). \qquad (8)$$

La carga de cada capacidad del enlace DC del inversor de tres niveles se realiza a partir de la función (9).

$$V_c(k+1) = V_c(k) + \frac{1}{C} i_c(k) T_s \qquad (9)$$

En la función (9) el valor de $\mathbf{i_c(k)}$ es la corriente a través de la capacidad, $\mathbf{V_c(k)}$ es el voltaje de la capacidad y **C** es la capacidad. Realizando la función (9) con las corrientes que circulan por las capacidades en el presente estado se obtiene los valores de tensión de las capacidades para el siguiente estado. De esta forma no son necesarias medidas adicionales.

El valor del siguiente estado de muestreo de la corriente de carga y voltaje de las capacidades se prevé para los 27 estados de conmutación que forman el inversor de tres niveles. La realización de la previsión se realiza por medio de las funciones (6) y (9). El uso de las funciones (6) y (9) es necesario medir la corriente de carga actual y el voltaje en las capacidades. Una vez obtenidas las predicciones para el siguiente estado de muestreo, una función de calidad **g** (10) evalúa todas las predicciones realizadas determinando una de las 27 para el siguiente estado de muestreo. El estado de muestreo que minimiza el valor de **g** se implementa en el siguiente estado de conmutación del inversor.

$$g = f\left(\mathbf{i}^*(k+1), \mathbf{i}(k+1)\right) + h(Vc_{12}(k+1), n_c) \qquad (10)$$

En la función (10) el valor de $\mathbf{n_c}$ representa el número de conmutaciones de los semiconductores de potencia para llegar al proceso de evaluación del siguiente cambio de estado. El primer término de la función (10) **f (i*, i)**, realiza la cuantificación de la diferencia entre la corriente de referencia y la predicción de la corriente del siguiente estado de muestreo. De esta forma la nueva composición de la función (10) se muestra en la ecuación (11).

$$f\left(\mathbf{i}^*(k+1), \mathbf{i}(k+1)\right) = \left|i_\alpha^*(k+1) - i_\alpha(k+1)\right| + \left|i_\beta^*(k+1) - i_\beta(k+1)\right| \qquad (11)$$

Las variable **iα y iβ** de la función (11) son los componentes reales e imaginarios del vector de corriente **i**, respectivamente. Las variables **i∗α y i∗β** son los componentes reales e imaginarios del vector de corriente de referencia **i***.

El objetivo del segundo término en la función (10) **h (Vc12, nc)** es para aplicar el estado de redundancia de un inversor de tres niveles. El seguimiento del término **f** de la función (10) solo depende del voltaje del vector seleccionado. La composición del segundo término de la función (10) se muestra en la ecuación (12).

$$h(Vc_{12}(k+1), n_c) = \lambda_{\mathrm{dc}} \cdot \left|Vc_1(k+1) - Vc_2(k+1)\right| + \lambda_n \cdot n_c. \qquad (12)$$

El primer elemento $\mathbf{h(V_{c12}(k+1),n_c)}$ de la función (12) se suma a la función **g** (10), este término representa el valor proporcional de la diferencia absoluta entre las predicciones de

tensión de las dos capacidades del inversor de tres niveles. El segundo elemento en **$h(V_{c12}(k+1),n_c)$** de la función (12) es proporcional al número de conmutaciones a realizar en el próximo cambio de estado **nc**. En la función de calidad se busca un cambio de estado para el siguiente estado de muestreo que genere pequeñas diferencias entre las tensiones de las capacidades del inversor de tres niveles y realice el mínimo número de conmutaciones de los semiconductores de potencia. La función **$h(V_{c12}(k+1),n_c)$** tiene un efecto directo en la frecuencia de conmutación del inversor de tres niveles. Los factores **λdc** y **λn** controlan la relación entre los términos dedicados al seguimiento de la referencia, la tensión de equilibrio, y la reducción de la frecuencia de conmutación dentro de **g**. En la figura 2.3.3 se muestra un diagrama que representa la implementación de la estrategia de control predictivo.

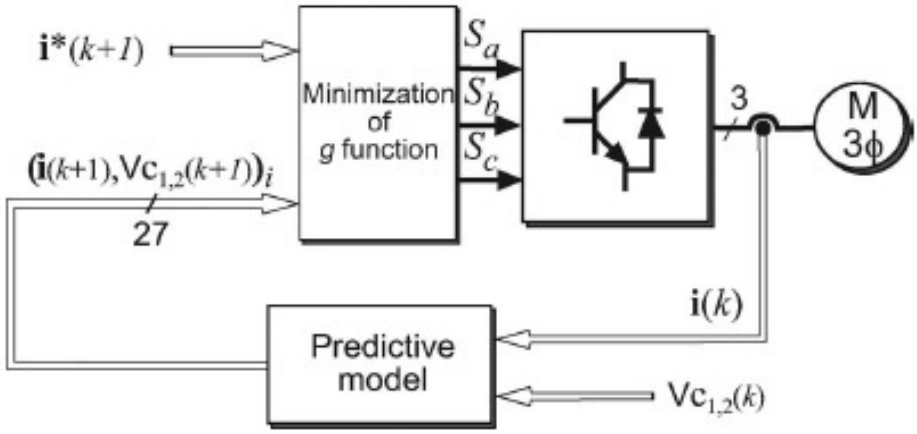

Figura 2.3.3: Método de control predictivo.

3 MANUAL DE USUARIO:

3.1 Introducción:

El seguimiento de desarrollo en las aplicaciones basadas en DSC (Controlador de Señales Digitales) consiste en cuatro fases básicas que se muestran en la figura 3.1.1.

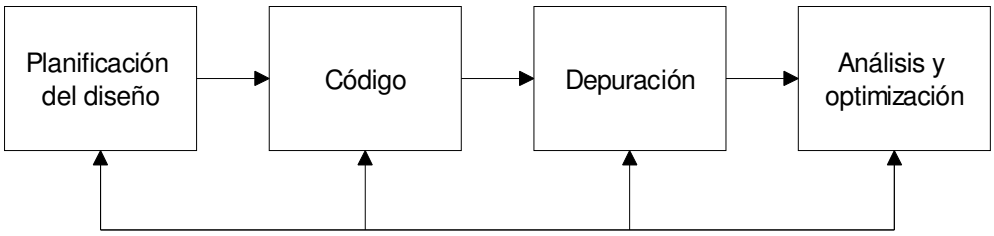

Figura 3.1.1: Diagrama de desarrollo

Estas cuatro fases básicas son consecutivas, pero se puede modificar cada una de las fases en paralelo. De esta forma se puede actualizar cada fase básica sin necesidad de recorrer hacia tras el diagrama.

Este documento pretende ser una guía que permita emplear las herramientas de programación del DSC TMS320F28335 a usuarios con conocimientos básicos o nulos de programación de DSCs. En la guía se describen los pasos necesarios y las herramientas básicas para la utilización de *Code Composer Studio* con la tarjeta de evaluación eZdsp F28335.

3.2 Puesta en marcha eZdsp F28335:

Antes de utilizar el software de programación *Code Composer Studio IDE* se requiere una previa instalación de los drivers de la tarjeta de evaluación utilizada, en este caso eZdsp F28335. Para ello es necesario entrar en *Setup Code Composer Studio* para instalar los drivers necesarios. En este programa esta definidos los drivers de varias tarjetas de evaluación.

Al entrar en *Setup Code Composer Studio* se debe de seleccionar los drivers de la tarjeta de evaluación considerada, los cuales están predefinidos de fábrica. Para realizar dicha configuración de los drivers es necesario seguir los pasos que a continuación se describen.

1.- Doble clic en el icono *Setup CCStudio* (Figura 3.2.1). Aparece el cuadro de dialogo de configuración del sistema.

Figura 3.2.1: Icono Setup CCStudio

2.- Elegir del listado de tarjetas de fábrica disponibles, la tarjeta con el nombre de F28335 eZdsp.

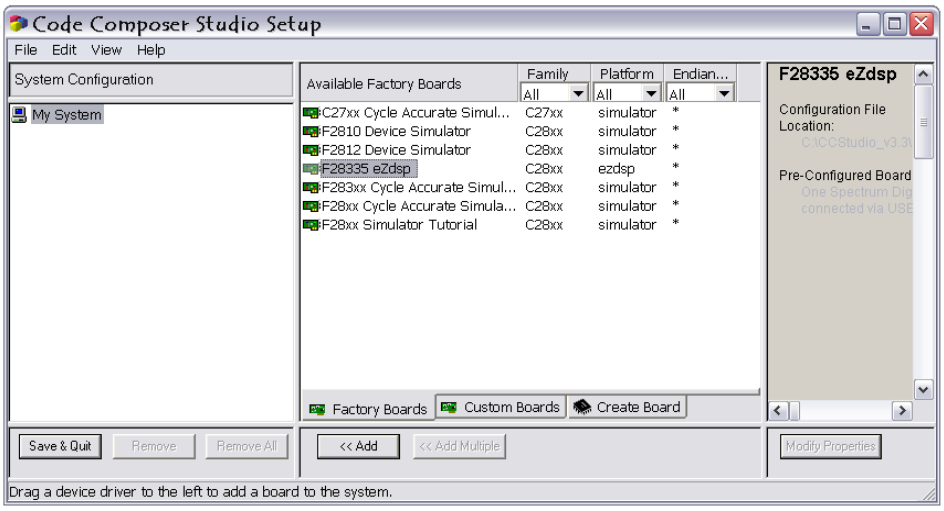

Figura 3.2.2: Cuadro de dialogo de configuración del sistema.

3.- Clicar en el botón de añadir (<<*Add*), para importar la configuración de la tarjeta a la configuración del sistema.

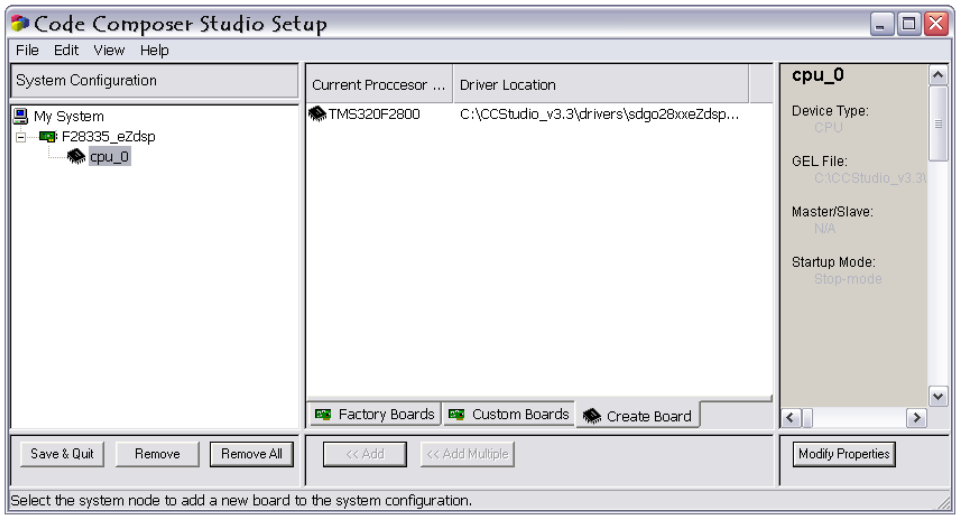

Figura 3.2.3: Cuadro de dialogo de configuración del sistema con la configuración realizada.

4.- Guardar y quitar (*Save & Quit*). Después de realizar esta operación se abre un cuadro de dialogo el cual pregunta se quiere abrir *Code Composer Studio IDE*.

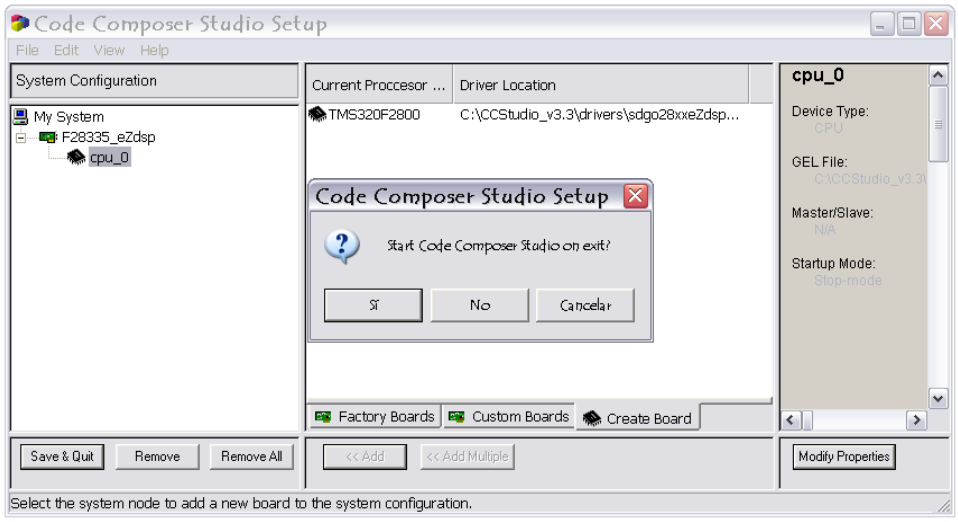

Figura 3.2.4: Opciones de guardar la configuración del sistema

5.- Elegir la opción más adecuada:

- **Si** => Se guarda la configuración del sistema de la tarjeta y seguidamente se abre *Code Composer Studio IDE.*
- **No** => Se guarda la configuración del sistema de la tarjeta sin abrir *Code Composer Studio IDE.*
- **Cancelar** => No se guarda la configuración del sistema de la tarjeta.

Para seguir con la configuración del sistema la opción más adecuada es la primera opción "**Si**". Esta opción guarda la configuración del sistema y a continuación se abre *Code Composer Studio*.

3.3 Entorno Code Composer Studio:

Para poder realizar una utilización adecuada de *Code Composer Studio*, antes se debe de entender el entorno junto las posibilidades que ofrece el mismo software. Por este motivo antes de generar un nuevo proyecto, se realiza una descripción de las herramientas que son más útiles a la hora de utilizar este software. A continuación se describen los iconos más utilizados a la hora de realizar código de aplicación y herramientas de visualización las cuales son útiles para la comodidad de utilización del entorno de este software.

Code Composer Studio tiene un entorno de programación similar a otros tipos de procesadores ajenos a Texas Instruments. La figura 3.3.1 muestra la ventana de inicio de *Code Composer Studio*.

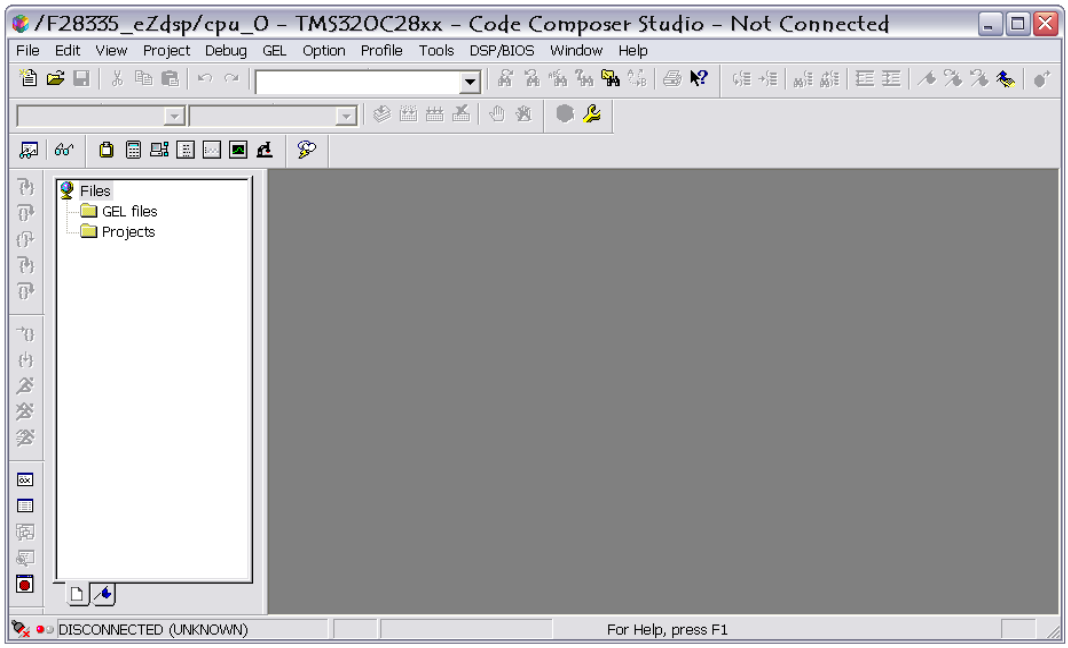

Figura 3.3.1.: Entorno *Code Composer Studio*.

Los iconos que se describen en este apartado, son los iconos mínimos que se deben de entender para la correcta utilización del entorno. Los iconos se agrupan dependiendo en función de su uso, por este motivo los iconos que pertenecen al mismo grupo se agrupan por barras de herramientas.

Las dos barras de herramientas que se describen a continuación son las principales para realizar una correcta aplicación.

La barra de herramientas de construcción sirve para crear el archivo código fuente y la barra de herramientas de ejecución sirve para poder depurar el código de la aplicación.

Barra de herramientas de construcción:

 Compilación del archivo que está en uso. Solo compila el archivo en el cual se encuentra el cursor de escritura.

 Construcción del archivo que está en uso. Solo construye el archivo en el cual se encuentra el cursor de escritura.

 Compilación y construcción de todos los archivos que forman parte del proyecto para crear el archivo ejecutable del proyecto.

 Parar la compilación y construcción del archivo ejecutable.

Barra de herramienta de ejecución:

 Ejecuta el programa hasta la siguiente función de código C.

 Ejecuta el programa hasta la siguiente línea de código C.

 Ejecuta el programa hasta la siguiente función de código ensamblador.

 Ejecuta el programa hasta la siguiente línea de código ensamblador.

 Coloca el puntero PC del programa en la ubicación donde se encuentra el cursor de escritura.

 Ejecuta el programa hasta que el puntero PC se ubique en la misma posición que el cursor de escritura.

 Ejecutar el programa.

 Parar la ejecución del programa.

 Colocar puntos de parada de programa (breakpoints).

 Borrar puntos de parada de programa.

 Visualizar las variables del programa en ejecución.

3.4 Creación nuevo proyecto:

Un proyecto está compuesto por diferentes tipos de archivos. Estos archivos son necesarios para poder generar un único programa ejecutable a partir del código escrito de la aplicación a realizar. Existen cuatro tipos de archivos principales para poder realizar un único programa ejecutable a partir de estos archivos.

- Archivos con extensión *.c*: Son documentos construidos a partir de lenguaje C.
- Archivos con extensión *.asm*: Son documentos construidos a partir de lenguaje ensamblador.
- Archivos con extensión *.h*: Son documentos llamados librerías en los cuales están definidas todas las variables y funciones que se utilizan en los documentos de código, ya sea *.c* o *.asm.*
- Archivo con extensión *.cmd*: Es un archivo el cual contiene la memoria mapeada del DSC utilizado.

Un proyecto está formado por varios archivos, cada uno de estos archivos puede estar construido con un tipo de lenguaje de programación diferente, ya que el compilador de *Code Composer Studio* es capaz de generar un único programa ejecutable de un proyecto que contenga archivos construidos con diferentes tipos de lenguaje de programación.

Después de realizar un nuevo proyecto se debe de configurar las opciones de compilación y construcción del archivo ejecutable. Esta configuración evita que el compilador a la hora de construir un único archivo ejecutable no encuentre ninguna incompatibilidad entre los archivos que forman el mismo proyector.

Un proyecto siempre debe incluir un archivo con la extensión *.cmd*. Este archivo contiene la memoria mapeada del DSC utilizado y tiene definido en qué dirección de memoria del DSC se va a ubicar el código ejecutable, igual que las variable utilizadas en este mismo código ejecutable.

La creación de un nuevo proyecto requiere la realización de diversos pasos. Hay que tener en cuenta que cada proyecto debe de tener un nombre único lo que permite abrir varios proyectos simultáneamente. La información de cada proyecto se guarda en un único archivo con la extensión **.pjt*.

1.- En la barra de herramientas se selecciona el menú *Project*, dentro de este menú seleccionar *New*. La figura 3.4.1 muestra la ventana de creación de nuevos proyectos.

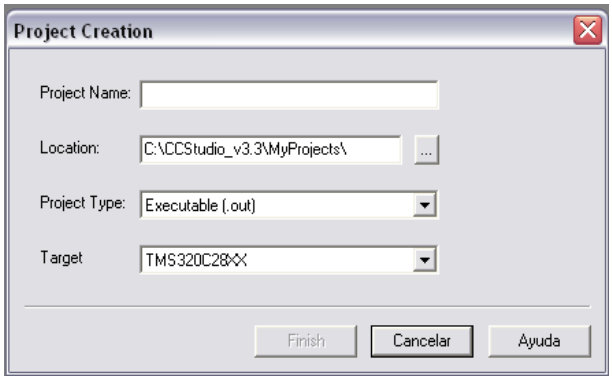

Figura 3.4.1.: Cuadro de dialogo de creación de proyectos

2.- En el campo *Project Name* se define el nombre del proyecto. Cada proyecto debe de tener un nombre único, esto permite la opción de abrir varios proyectos simultáneamente.

3.- En el campo *Location* se especifica el directorio en el que se almacena todos los archivos que forman el proyecto a crear.

4.- En el campo *Project Type* se selecciona el tipo de proyecto. Existen dos opciones del tipo del proyecto, los *Ejecutables* con la extensión **.out* y *Librería* con la extensión **.lib*. Los proyectos Ejecutables generan un archivo el cual el DSP puede ejecutar como una aplicación real. Los proyectos Librería generan un archivo el cual contiene variables y definiciones de funciones que sirven para usarlas en los archivos ejecutables de código con las extensiones *.c* y *.asm*.

5.- En el campo *Target* se selecciona el tipo de tarjeta para el cual se va a generar el código del proyecto. En este caso para la tarjeta de evaluación utilizada en este proyecto es TMS320C28XX.

6.- Seleccionar *Finish*. En la realización de este paso se genera un archivo de proyecto llamado *elnombredelproyecto*.pjt. En este archivo se almacena todos los archivos y ajustes del proyecto utilizados en el mismo.

Al finalizar los anteriores pasos, el entorno *Code Composer Studio* es el que se muestra en la figura 3.4.2.

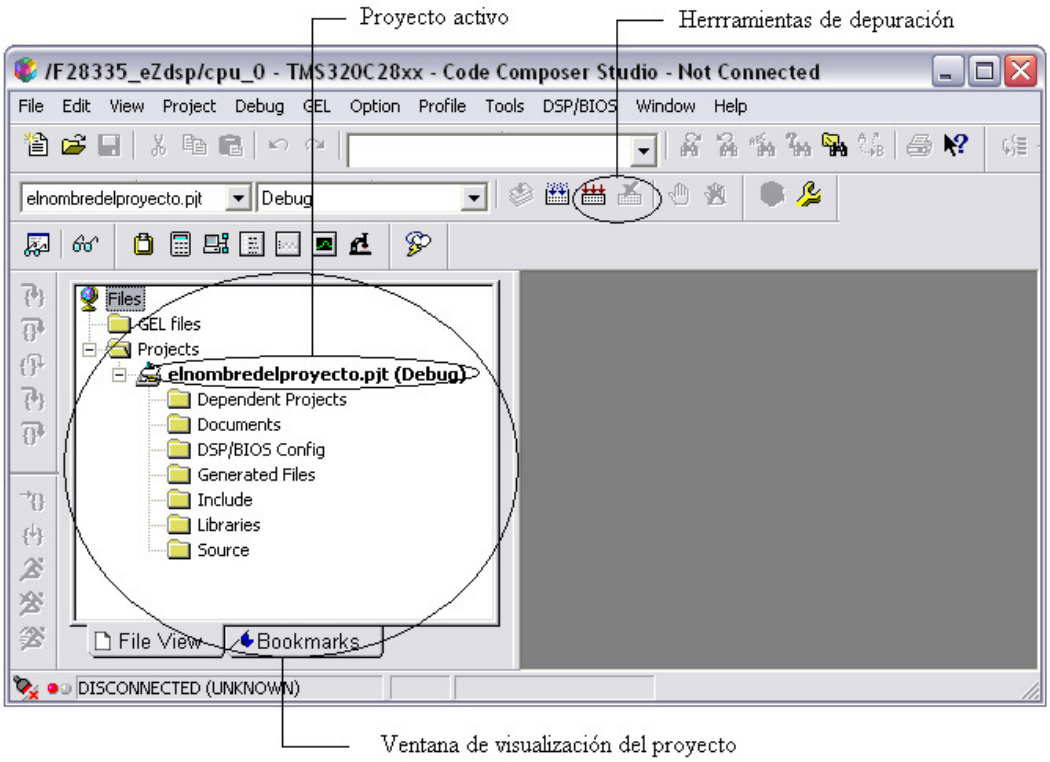

Figura 3.4.2.: Visualización del proyecto creado en *Code Composer Studio*.

Después de crear el proyecto se debe de realizar la configuración de compilación y construcción del archivo ejecutable. Esto evita errores en el compilador a la hora de generar el archivo ejecutable (*.out*).

A continuación se describe como configurar las principales opciones de compilación y construcción del proyecto. Las siguientes opciones de configuración son las mínimas a realizar para que no surja ningún error ni ningún warning a la hora de construir el archivo ejecutable (*.out*).

Para entrar en las opciones de configuración de compilación y construcción del proyecto, se debe de abrir el menú de *Project*, seleccionando la opción de *Build Options*.

Aparece un cuadro de dialogo el cual tiene varias opciones de pestañas y cada opción de pestaña está compuesta por varias opciones de categorías. No es necesario configurar todas las opciones.

Solo se configura las opciones de pestaña del compilador (*Compiler*) y del Constructor (*Linker*), de las cuales se configurara las opciones de categoría que a continuación se muestran.

El cuadro de dialogo de las opciones de configuración es el que se muestra a continuación en la figura 3.4.3. Se empieza por la categoría básica del compilador.

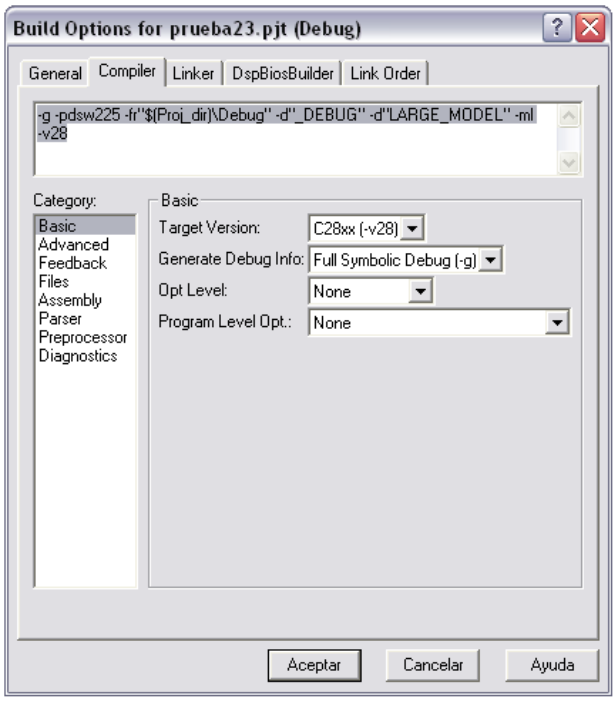

Figura 3.4.3.: Cuadro de dialogo categoría básica del compilador.

En el campo *Target Version* se selecciona de la versión de la tarjeta que se utiliza en el proyecto. Esta opción por defecto al crear el nuevo proyecto ya se define, pero permite cambiarla en cualquier momento modificando este campo antes de generar el archivo ejecutable. Para cambiar esta opción se selecciona la lista desplegable del campo como se muestra en la figura 3.4.4.

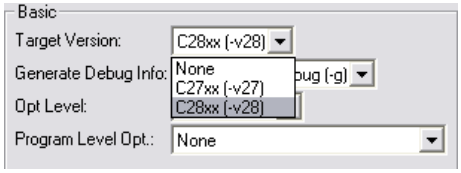

Figura 3.4.4.: Opciones de versión de tarjeta

En *Generate Debug info* se selecciona que información se va a generar del depurador cuando el compilador compile los archivos que forman el proyecto. Esta opción se muestra en la figura 3.4.5.

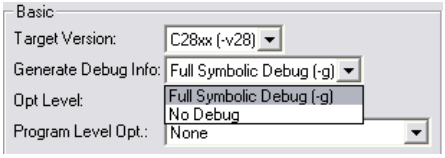

Figura 3.4.5.: Opciones de generación de información del depurador

Dentro de la lista desplegable de esta opción se encuentran dos opciones *Full Symbolic Debug* y *No Debug*. La selección de la opción *Full Symbolic Debug* genera información de todos los símbolos que se utilizan en el depurador y la selección de la opción *No Debug* no genera ninguna información sobre el depurador.

Con las anteriores configuraciones ya se ha abarcado las mínimas opciones de la categoría básica del compilador. La siguiente categoría del compilador a configurar es la categoría de archivos *Files*.

En la categoría de archivos solo es necesario definir en qué directorio se va a ubicar el archivo objeto del proyecto. La categoría archivos tiene el cuadro de dialogo mostrado en la figura 3.4.6.

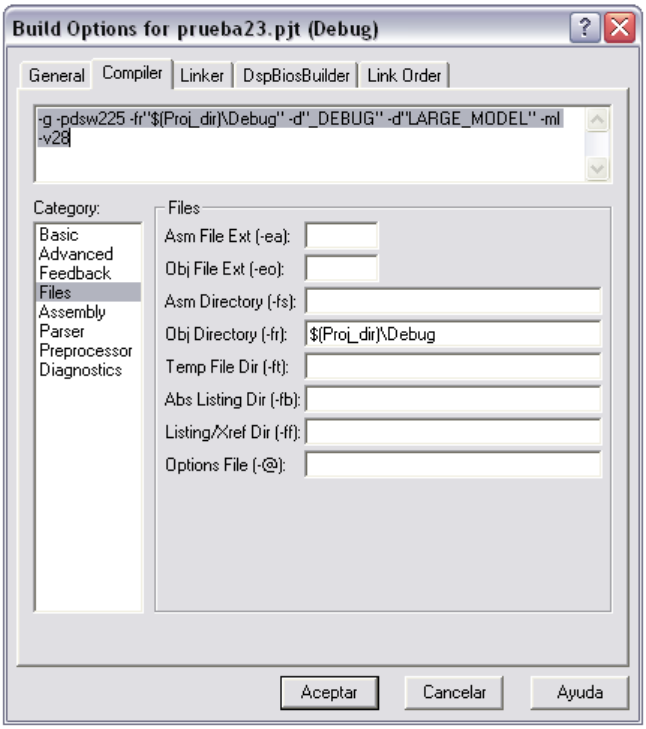

Figura 3.4.6.: Cuadro de dialogo categoría archivos del compilador.

Para definir la ubicación del directorio donde se va ubicar el archivo objeto del proyecto, se debe de escribir el directorio en el campo de escritura de la opción *Obj Directory*, dentro de la categoría archivos del compilador. El archivo objeto tiene la extensión **.obj*.

Una vez realizada la anterior configuración se finaliza la configuración de la categoría de archivos del compilador. A su vez la configuración mínima de las opciones del compilador está finalizada.

Siguiendo con las opciones de configuración del proyecto se realiza a continuación la configuración de las opciones del constructor (*Linker*). Dentro de las opciones del

constructor solo se va a realizar la configuración de la categoría básica. El constructor (*Linker)* tiene la ventana de dialogo que se muestra en la figura 3.4.7.

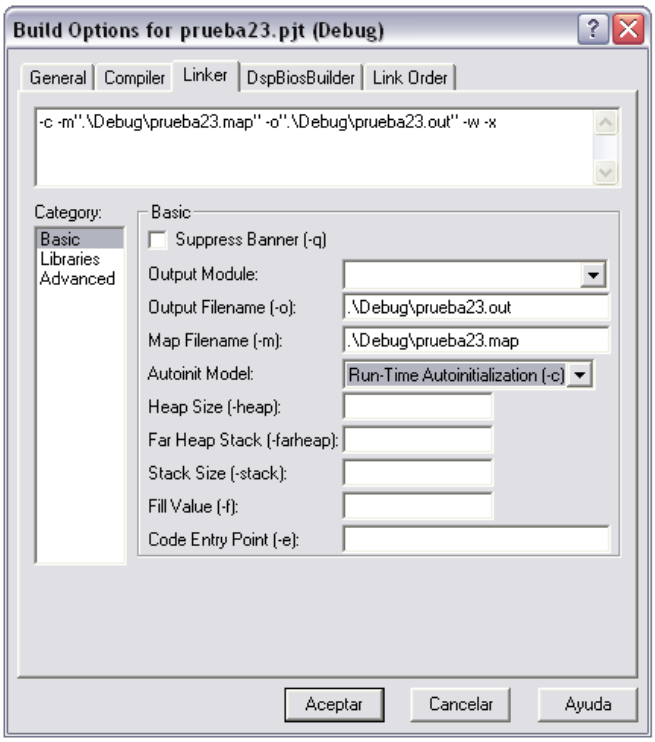

Figura 3.4.7.: Cuadro de dialogo categoría básica del constructor (*Linker*).

En la opción *Output Module* se selecciona el tipo de modulo del archivo ejecutable. Esta opción permite elegir qué tipo de ubicación tendrá el programa ejecutable dentro de la memoria del DSC. Existen tres posibilidades que se muestran en la figura 3.4.8.

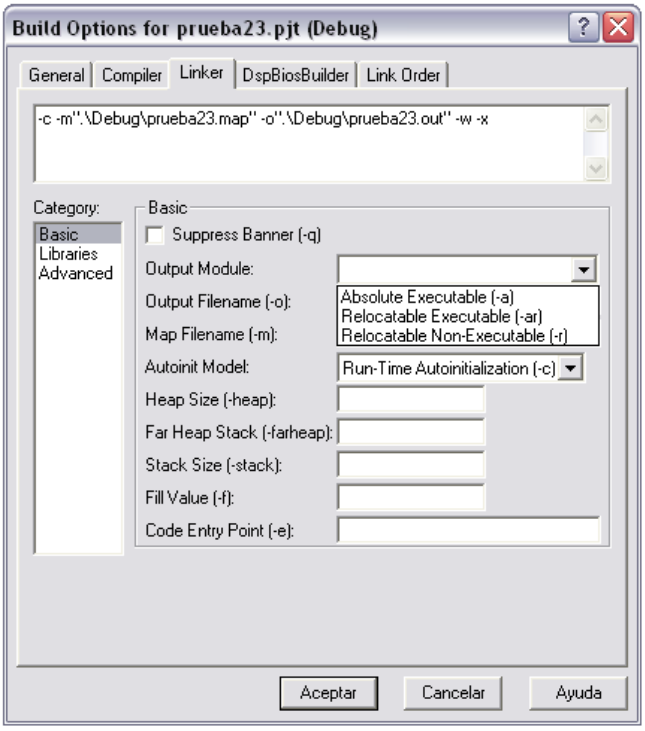

Figura 3.4.8.: Opciones del tipo de modulo del archivo ejecutable (*.out*).

Opción *Absolute Executable,* ubica el código ejecutable en el modulo de memoria que el programador define. Esta ubicación es única y no se puede ampliar, el programador define en que modulo de memoria se va a ubicar el código ejecutable en el archivo *.cmd*. En esta opción hay que tener en cuenta el tamaño que adquiere el archivo ejecutable y el tamaño del modulo de memoria en el que se va a ubicar ya que puede darse el caso que el código ejecutable tenga un tamaño mayor que el modulo de memoria en el que se va a ubicar.

Opción *Relocatable Executable*, esta opción ubica el código ejecutable a partir del modulo de memoria que el programador define para el código ejecutable. A diferencia de la opción *Absolute Executable* si el código ejecutable tiene un tamaño mayor que el modulo de memoria del DSC seleccionado para su ubicación, el código sigue ubicándose en el siguiente módulo de memoria hasta que se ubique todo el código ejecutable.

Opción *Relocatable Non-Executable*, realiza la misma operación que la opción de *Relocatable Executable*, pero el código que se almacena en los módulos de memoria no es un código ejecutable.

Siguiendo con las opciones de la categoría básica de la configuración del constructor (*Linker*), en la opción *Output Filename* se describe el directorio donde se va a ubicar el archivo ejecutable del proyecto seguidamente del nombre del archivo ejecutable. Se recomienda que los archivos ejecutables de un proyecto tengan el mismo nombre que el proyecto al que pertenecen. El archivo ejecutable adquiere la extensión *.out*, que se ubicara en el directorio definido en la opción *Output Filename*

En la opción *Map Filename* se describe el directorio donde se va a ubicar el archivo de mapa del proyecto seguidamente del nombre del archivo de mapa. Se recomienda que los archivos de mapa de un proyecto tengan el mismo nombre que el proyecto al que pertenecen. El archivo de mapa adquiere la extensión *.map*, que se ubicara en el directorio definido en la opción *Map Filename*. Este archivo es importante ya que ayuda al programador a depurar el programa de la aplicación a realizar.

Opción *Autoinit Model*, esta opción contiene tres formas de construir el modulo de arranque del código ejecutable. Dentro de las cuales se define en qué punto se va a ubicar el puntero de programa al cargar el código ejecutable dentro de la memoria del DSC.

La opción *Stack Size* permite definir el tamaño de la pila (Stack) para la ubicación de las variables cuando el programa está en ejecución o de las direcciones de retorno cuando el programa entra en una subrutina

La opción *Code Entry Point* permite definir cuál es el primer modulo que se va a ejecutar del programa ejecutable de la aplicación. En esta opción se describe el nombre del bloque de programa seleccionado para el comienzo de la aplicación. En la figura 3.4.9 se muestra un ejemplo.

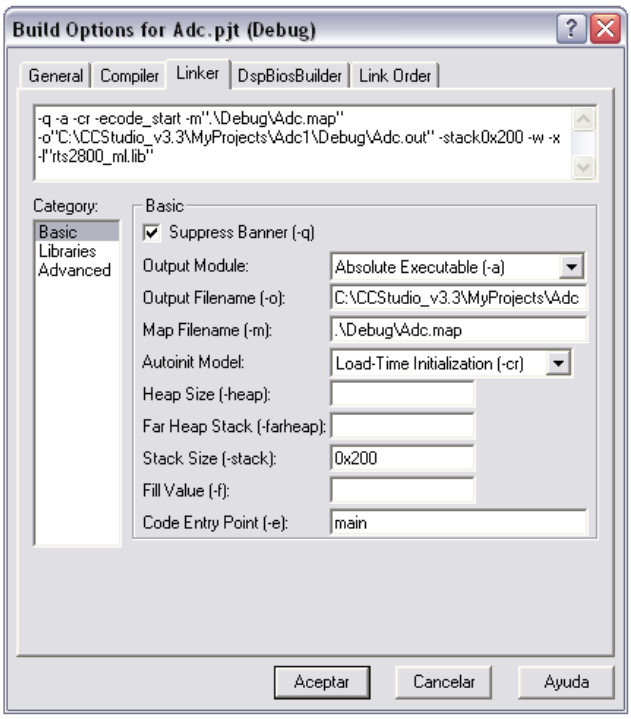

Figura 3.4.9.: Opciones del punto de entrada del código.

Una vez realizadas las anteriores configuraciones se finaliza la configuración del compilador (*Compiler*) y constructor (*Linker*) del proyecto.

Para finalizar la configuración del proyecto, se describen a continuación una opción que permite definir opciones de creación para diferentes fases de desarrollo del programa. Las opciones especificadas en la configuración del proyecto se aplican a cada uno de los archivos del proyecto.

Cada proyecto se genera con dos configuraciones por defecto: *Debug* y *Release*.

- La configuración *Debug* se utiliza para la depuración de la aplicación.
- La configuración *Release* se utiliza para la creación de la aplicación acabada.

La selección del tipo de configuración del proyecto se realiza a partir de las pestañas desplegables de la barra *Project*. La barra *Project* se muestra en la figura 3.4.10.

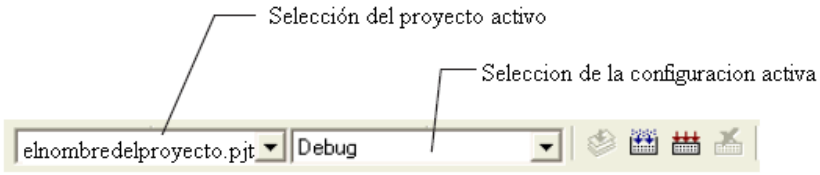

Figura 3.4.10.: Barra de herramientas proyecto.

Después de la creación del proyecto y la realización de las configuraciones necesarias, el último paso es añadir al proyecto creado el archivo *.cmd* que se va a utilizar. El cual contendrá toda la memoria mapeada del DSC utilizado para la aplicación.

Al utilizar eZdsp F28335 que es una tarjeta estándar de fábrica, el archivo *.cmd* viene definido con el software *Code Composer Studio*. Solo hay que añadirlo dentro de nuestro proyecto.

Para añadir el archivo *.cmd* de la tarjeta de evaluación eZdsp F28335 se debe de realizar los siguientes pasos descritos. Dentro de la ventana de visualización del proyecto clicar con el botón secundario del ratón del ordenador en la carpeta *Source*. Seleccionando la opción *Add Files to Project…*, mostrado en la figura 3.4.11.

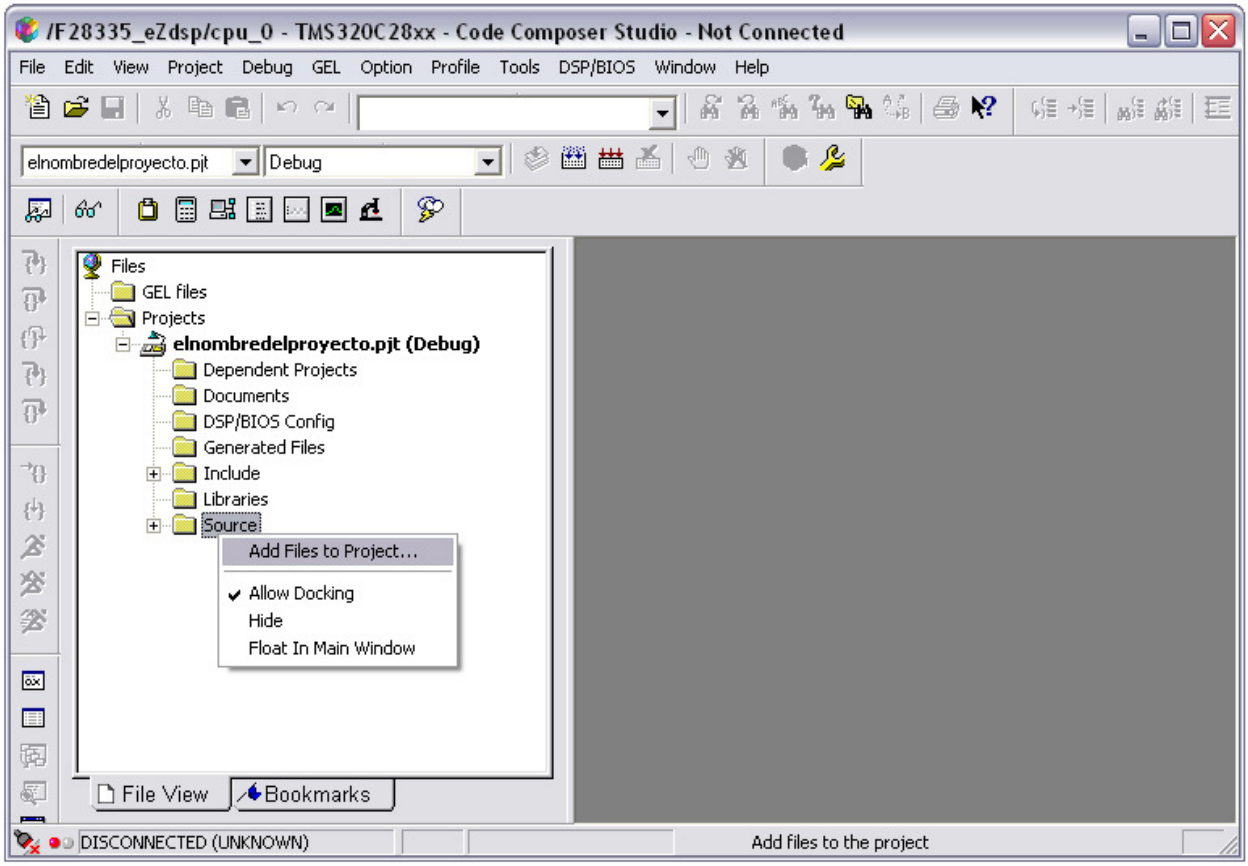

Figura 3.4.11.: Añadir archivo *.cmd*.

Se abre el siguiente cuadro de dialogo en el cual se busca el archivo *.cmd* que corresponde a la tarjeta de evaluación eZdsp F28335. En el cual seleccionamos *All Files (*.*)* del menú desplegable del tipo de archivo que buscamos, de la manera que se muestra en la figura 3.4.12.

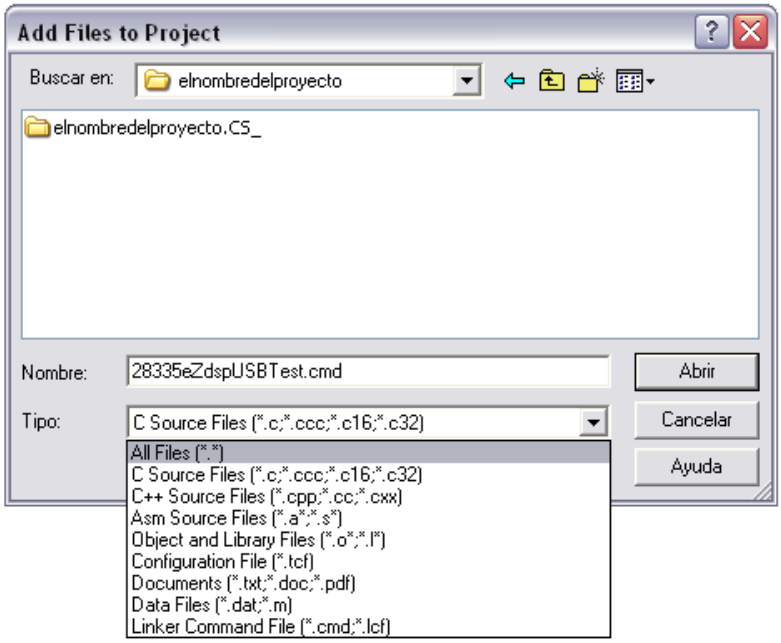

Figura 3.4.12.: Selección de visualización de todos los archivos.

Una vez seleccionado en "Tipo" la opción *All Files (*.*)*, buscamos en el menú desplegable de "Buscar en:" la carpeta 28335eZdspUSBTest. La carpeta está en la ubicación"C:CCStudio_v3.3/Boards/28335eZdspUSBDiags/Target/28335eZdspUSBTest", como se muestra en la figura 3.4.13.

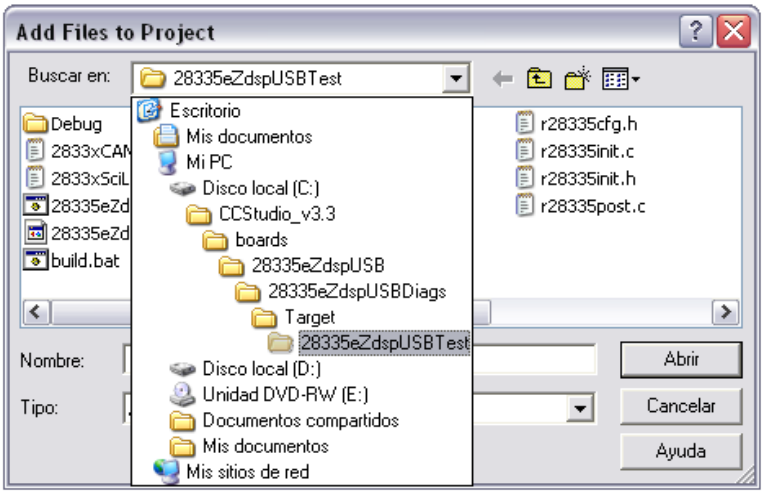

Figura 3.4.13.: Ubicación de la carpeta 28335eZdspUSBTest

Dentro de esta carpeta se selecciona el archivo "28335eZdspUSBTest.cmd" y "Abrir". En la ventana de visualización del proyecto se observa que se ha añadido el archivo "28335eZdspUSBTest.cmd" después de la carpeta *Source.*

3.5 Opciones de visualización del entorno:

Para una mayor comodidad en el entorno del programa *Code Composer Studio*, existen herramientas de visualización que mejoran la comodidad a la hora de utilizar este programa. Son herramientas de visualización las cuales mejoran incluso a la hora de depurar la aplicación. Estas opciones se encuentran en la barra de *Option >> Editor >> View Setup*. Las opciones de visualización se muestran en el cuadro de dialogo de la figura 3.5.1.

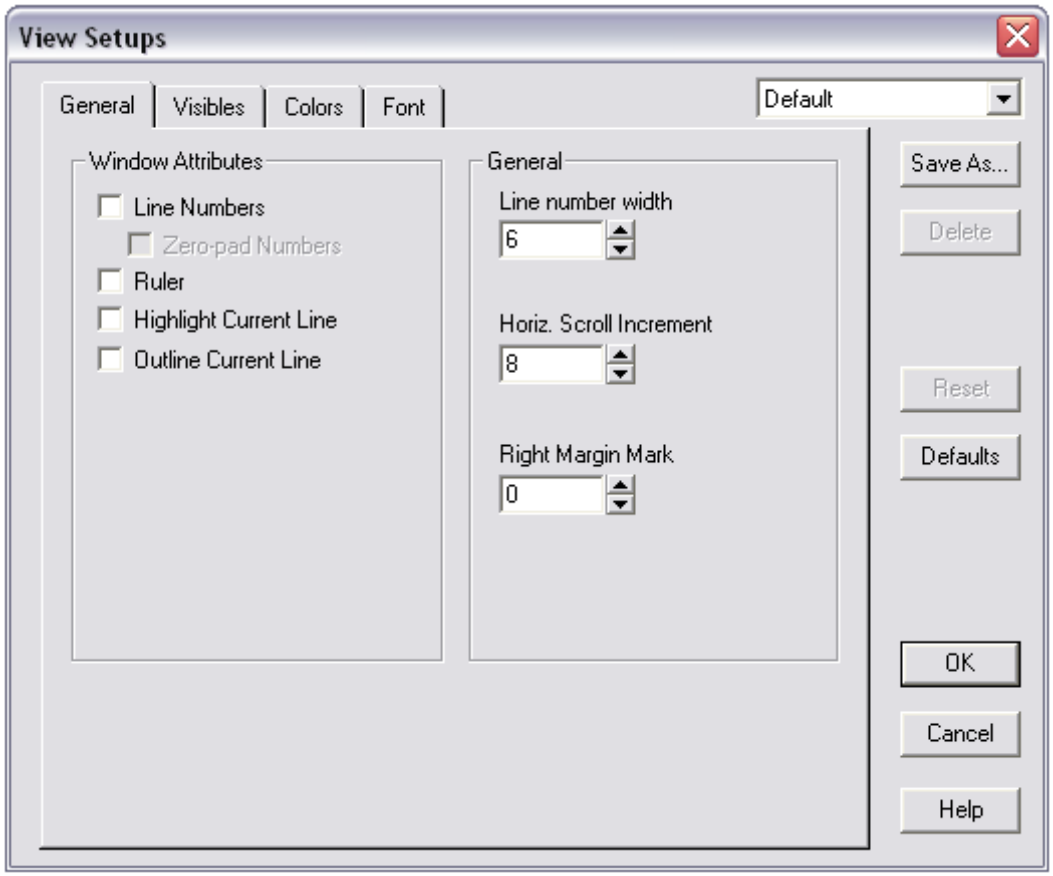

Figura 3.5.1.: Cuadro de dialogo opciones de visualización.

En la pestaña de opciones generales encontramos la ventana de atributos (*Windows Attributs*), en la cual se muestra cuatro opciones sin seleccionar: *Line Numbers*, *Ruler*, *Highlight Current Line* y *Outline Current Line*.

- *Line Numbers* (Figura 3.5.2): Esta opción numera las líneas de cada archivo que forman el proyecto. Esto es muy útil ya que el compilador cuando encuentra un error o algún warning describe cual es la causa del error, en que archivo del proyecto se encuentra y en qué línea del archivo está ubicado.

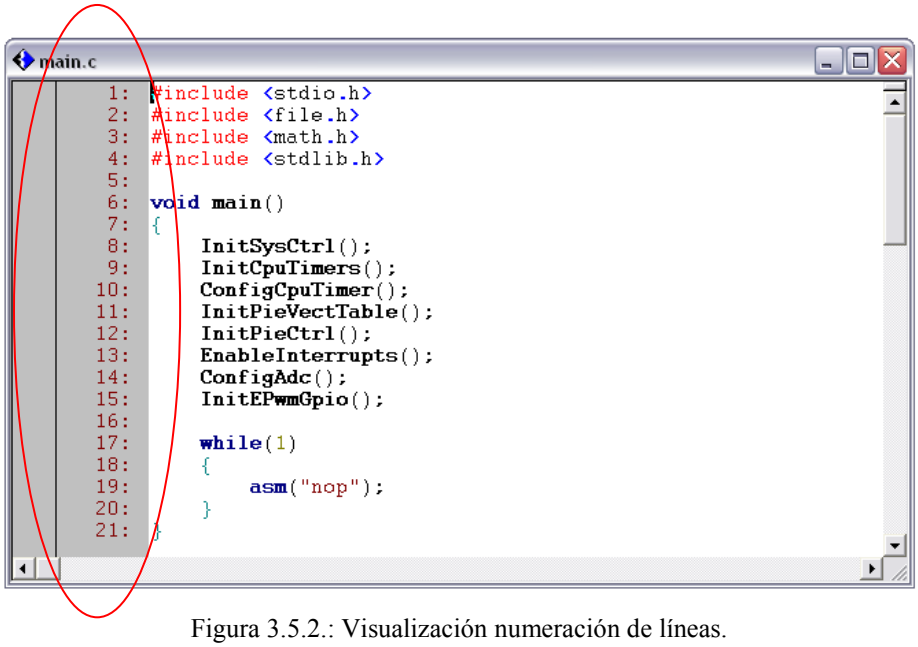

Figura 3.5.2.: Visualización numeración de líneas.

- *Ruler* (Figura 3.5.3): Esta opción muestra una regla en cada archivo que forma parte del proyecto. La regla se utiliza para encuadrar la escritura del código creado dependiendo del folio que se utilice para imprimir en papel los documentos del proyecto.

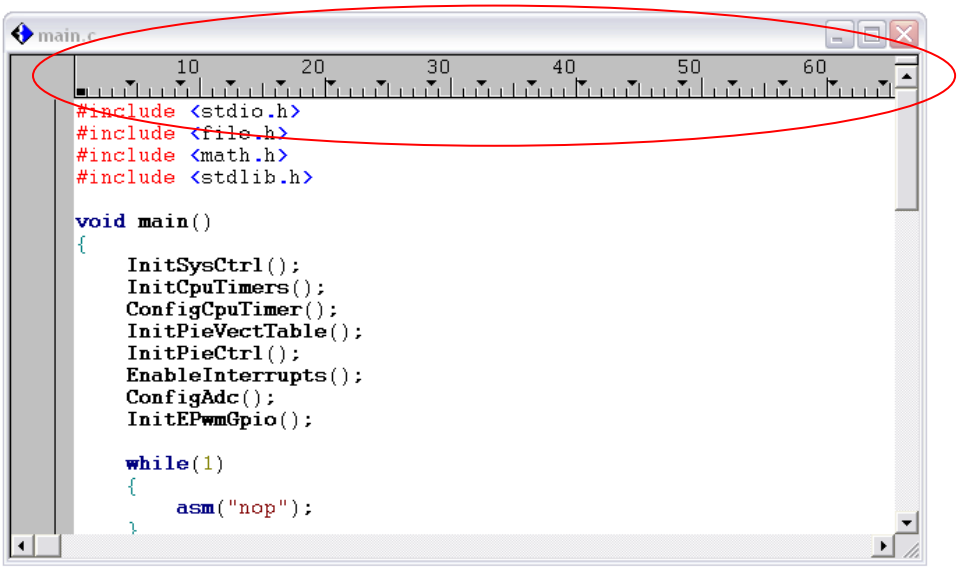

Figura 3.5.3.: Visualización regla.

- *Highlight Current Line* (Figura 3.5.4): Esta opción resalta la línea actual con una franja amarilla. Visualmente es muy llamativo y en cualquier momento se sabe cuál es la línea actual para escribir.

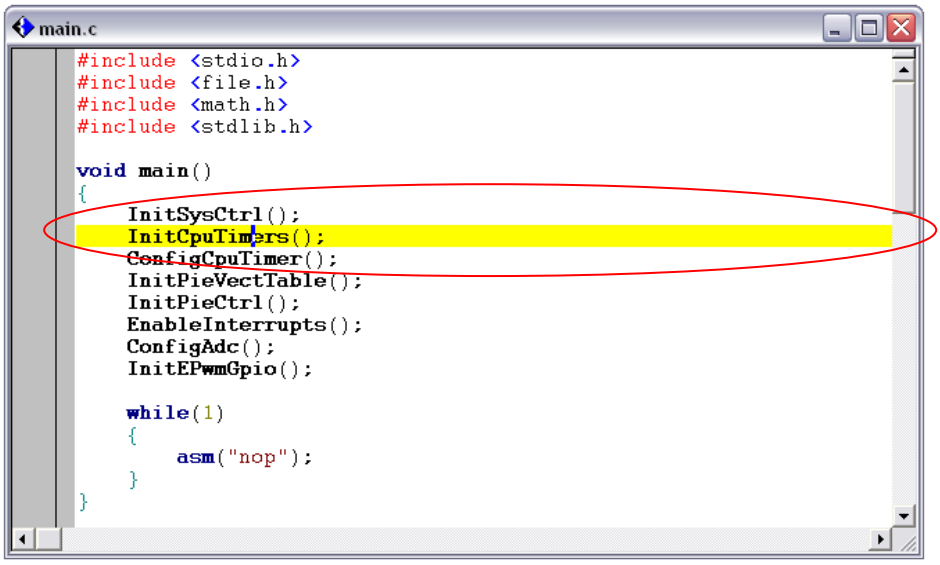

Figura 3.5.4.: Visualización resalto de línea actual.

- *Outline Current Line* (Figura 3.5.5): Esta opción resalta la línea actual con un recuadro. Visualmente es menos llamativo que la anterior opción, pero de también resalta la línea actual a diferencia de las que no lo son.

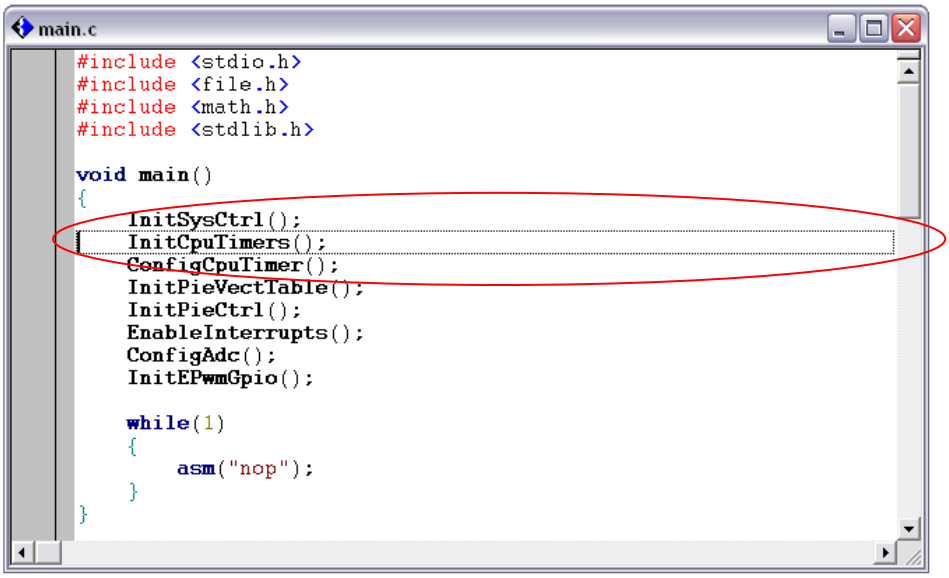

Figura 3.5.5.: Visualización recuadro de línea actual.

Las tres primeras opciones son muy recomendables activarlas a la hora de realizar cualquier aplicación real con *Code Composer Studio*.

4 APLICACIÓN:

4.1 Introducción:

La realización del control predictivo del inversor de tres niveles sobre la tarjeta de evaluación eZdsp F28335 es una aplicación real. La realización de esta aplicación tiene como finalidad realizar un ejemplo de programación de la tarjeta de evaluación eZdsp F28335 sobre un sistema físico.

El trabajo desarrollado dentro del proyecto se compone de varias partes. Las diferentes partes se dividen en la realización del software de control del sistema, la realización de los conectores necesarios para la conexión del equipo experimental a la tarjeta de evaluación eZdsp F28335, la puesta en marcha de la tarjeta de evaluación y creación de una placa de tratamiento de las señales generadas por los sensores del inversor de tres niveles.

La documentación desarrollada sobre la aplicación se divide en tres puntos. En el primer punto se explica el software desarrollado en la aplicación, en el segundo punto se explica el hardware que compone el sistema experimental utilizado, y el tercer punto se muestra los resultados obtenidos de la aplicación desarrollada.

En el apartado de software se explica la introducción de cada archivo de código creado con un diagrama de bloques. Los archivos de código se encuentran en este documento en el apartado de anexos.

En el apartado de hardware se explica los diferentes componentes que engloba el sistema experimental utilizado para el desarrollo de la aplicación. Todos los elementos del equipo experimental explicados se muestran en fotos para su localización en el laboratorio.

Dentro del apartado de resultados experimentales se recoge todas las conclusiones obtenidas de la aplicación realizada del inversor de tres niveles y se describen posteriores mejoras al trabajo realizado dentro de este proyecto.

4.2 Software:

La aplicación del control predictivo del inversor de tres niveles se implementa para la generación de corriente trifásica hacia la red. En la figura 4.2.1 se muestra el esquema de control del sistema.

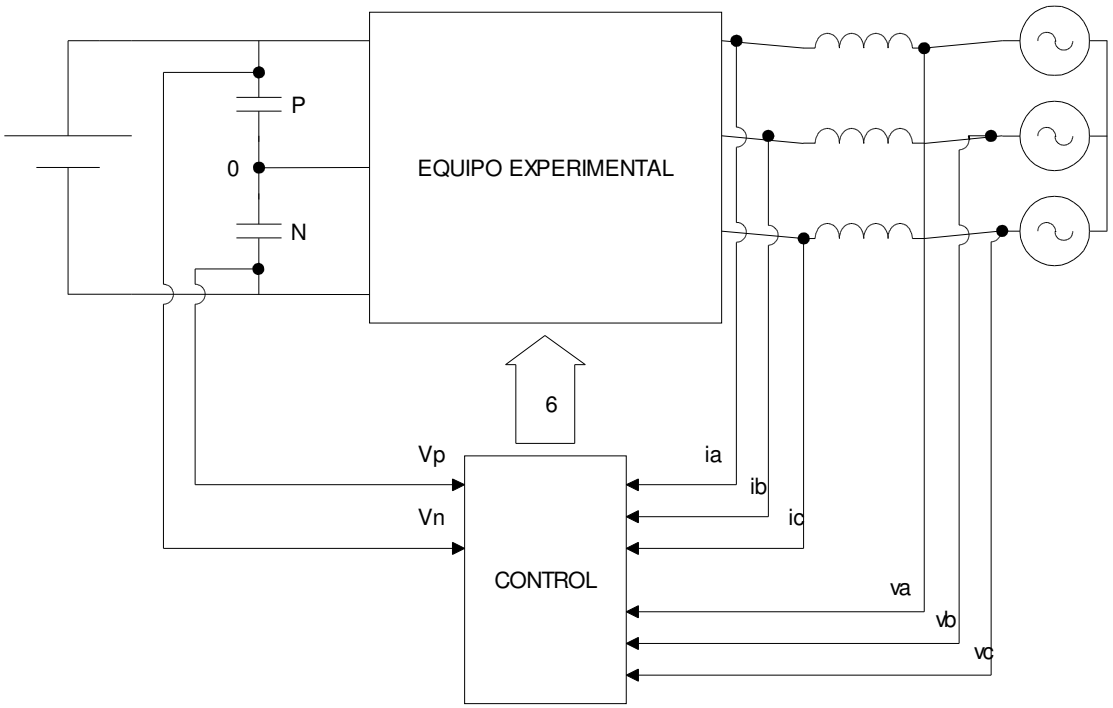

Figura 4.2.1.: Esquema de control del sistema.

El software desarrollado en Code Composer Studio para la realización de la aplicación se compone de 14 archivos de código con extensión *.c,* llamados módulos de programa. Esta extensión indica que los módulos de programa están desarrollados con lenguaje de programación C. Los 14 módulos de programa que contiene el software de la aplicación son:

- *main.c*
- *DSP2833x_AdcCtrl.c*
- *DSP2833x_AdcRef_Alfa-Beta.c*
- *DSP2833x_ConIGBTs.c*
- *DSP2833x_DefaultIsr.c*
- *DSP2833x_EPwm.c*
- *DSP2833x_GlobalVariableDefs.c*
- *DSP2833x_ik.c*

- *DSP2833x_Ik+1.c*
- *DSP2833x_PieCtrl.c*
- *DSP2833x_PieVect.c*
- *DSP2833x_SysCtrl.c*
- *DSP2833x_Vck+1_g[x].c*
- *DSP2833x_Vk+1.c*

El diagrama de bloque de la figura 4.2.2 muestra la secuencia que realiza el programa de la aplicación.

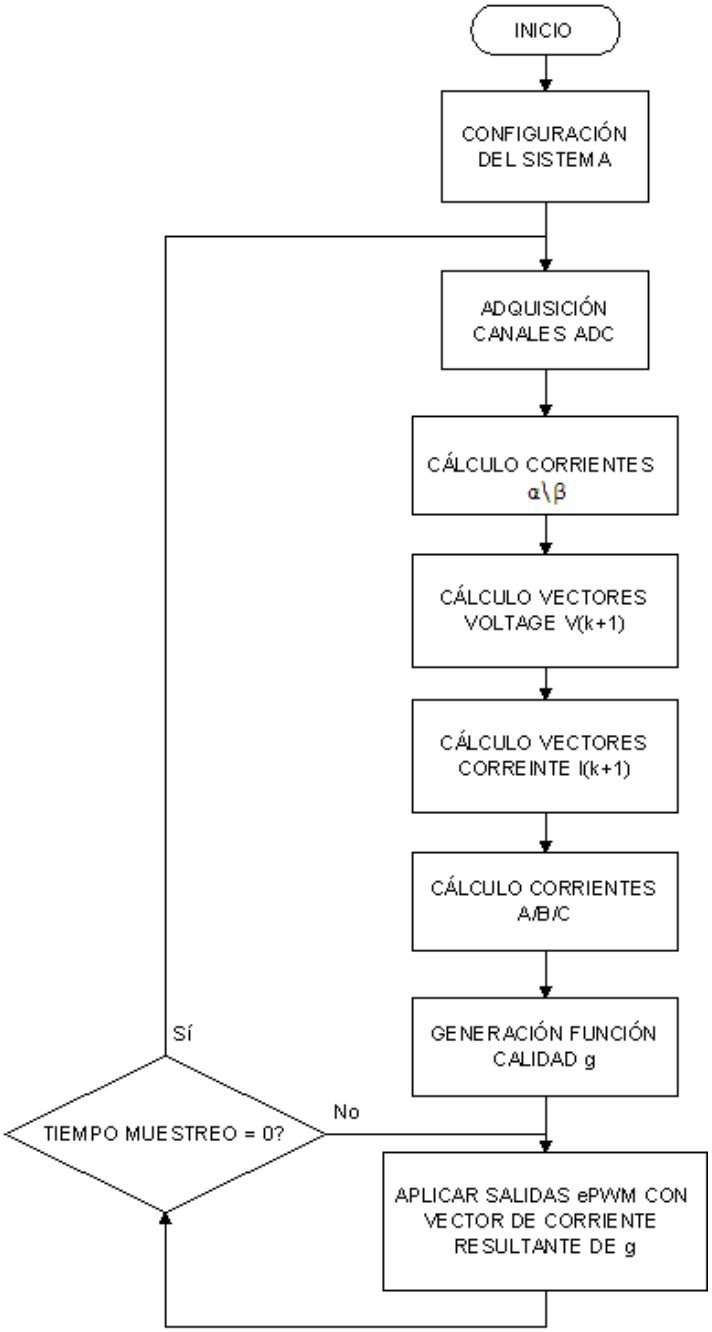

Figura 4.2.2.: Diagrama de bloques secuencia del programa.

La configuración del sistema en el diagrama de bloques de la figura 4.2.2 se realiza en el módulo de programa *main.c*. Este módulo de programa es el primero en ejecutarse al arrancar la aplicación. Cuando se compila y se carga el programa en la memoria de la tarjeta de evaluación eZdsp F28335 automáticamente el puntero de programa (PC) se sitúa en la primera instrucción de este módulo de programa.

La configuración del sistema consiste en configurar todos los recursos del controlador DSC TMS320F28335 que se van a utilizar en el programa. En este caso se realiza la configuración de los relojes del sistema, inicialización de la tabla de vectores, habilitación de las interrupciones, configuración del módulo ADC, inicialización de las salidas ePWM, configuración módulo ePWM y el cálculo de las variables fijas del programa. En la figura 4.2.3 se muestra el diagrama de bloques de la configuración del sistema que se realiza en el módulo de programa *main.c*. En el anexo 1 se encuentra el código del módulo programa main.c.

Figura 4.2.3.: Diagrama de bloques del módulo de programa *main.c*.

El módulo de programa *main.c* realiza la configuración del sistema a partir de los siguientes módulos de programa, estos módulos de programa solo son de configuración:

- *DSP2833x_AdcCtrl.c*: Configuración del módulo ADC
- *DSP2833x_EPwm.c*: Configuración del módulo ePWM
- *DSP2833x_GlobalVariableDefs.c*: Definición de variables y registros en memoria.
- *DSP2833x_PieVect.c*: Definición de la tabla de vectores del módulo PIE.
- *DSP2833x_PieCtrl.c*: Habilitación del módulo PIE (interrupciones de periféricos).
- *DSP2833x_SysCtrl.c*: Configuración de los relojes del sistema.

La adquisición de los canales ADC en el diagrama de bloques 4.2.2 se realiza en el módulo de programa *DSP2833x_AdcRef_Alfa-Beta.c*. Este módulo de programa realiza dos funciones principales, el ajuste del valor 0 en las señales alternas adquiridas y cálculo de las corrientes en alfa y beta. Los valores adquiridos por el módulo ADC recorren un rango de tensión de 0V a 3V con una resolución de 12 bits, estos valores son todos positivos. Varias señales adquiridas en los canales ADC son señales alternas, las cuales contienen valores positivos y negativos. La solución para obtener valores positivos y negativos en los canales utilizados para adquirir las señales senosoidales es ajustar la tensión 1,5V del rango de tensión soportada por los canales ADC a valor 0. La segunda función es calcular los valores de las corrientes α y β a partir de los valores adquiridos en el ajuste del valor medio de 1,5V a 0.

El ajuste de la tensión 1,5V al valor 0 se realiza del siguiente modo. Los valores adquiridos por el módulo ADC tienen una resolución de 12 bits, esto significa que los valores adquiridos están dentro del rango de valores decimales de 0 a 4095. Estos 4096 valores decimales corresponden al rango de tensión de entrada de 0V a 3V, a partir de estos rangos de tensión se obtiene que para una tensión de entrada de 1,5V corresponde al valor decimal 2048. El ajuste de la tensión 1,5V a valor 0 decimal se realiza restando el valor decimal de 2048 a los valores obtenidos en los canales del ADC. Esta operación solo se realiza para las entradas del módulo ADC que adquieren valores de señales alternas.

En la figura 4.2.4 se muestra el diagrama de bloques del módulo de programa *DSP2833x_AdcRef_Alfa-Beta.c*. En el anexo 2 se encuentra el código del módulo programa DSP2833x_AdcRef_Alfa-Beta.c.

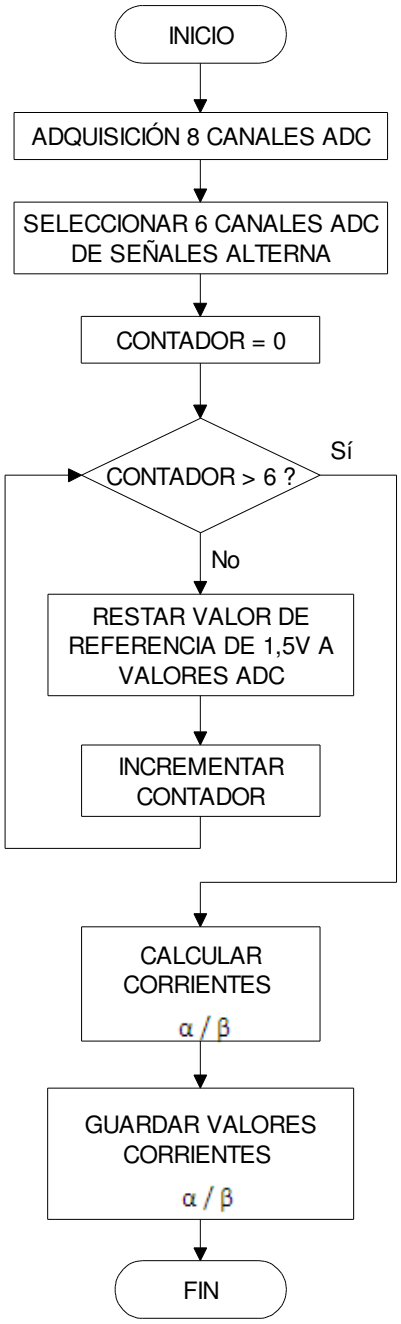

Figura 4.2.4.: Diagrama de bloques del módulo de programa *DSP2833x_AdcRef_Alfa-Beta.c*.

El cálculo de los vectores de voltaje V(k+1) del diagrama de bloques 4.2.2 se realiza en el módulo de programa *DSP2833x_Vk+1.c*. Este módulo de programa realiza el cálculo de los vectores de tensión del inversor de tres niveles para el siguiente estado de conmutación a partir de la diferencia de tensión que hay entre los condensadores de entrada del inversor. Realizando la diferencia de las tensiones de las capacidades de entrada al inversor de tres niveles se obtiene la tensión en el punto O del inversor. Con la tensión del punto O el módulo de programa *DSP2833x_Vk+1.c* realiza 19 operaciones que corresponden a los 19

vectores de tensión del inversor de tres niveles. Las funciones para realizar los cálculos de los 19 vectores de tensión son:

- V0 = 0 + 0j
- V1 =(Vdc / 3) + 0j
- V2 = (Vdc / 3) * (0,5 + 0,866j)
- V3 = (Vdc / 3) * (-0,5 + 0,866j)
- V4 = (Vdc / 3) * (-1 + 0j)
- V5 = (Vdc / 3) * (-0,5 - 0,866j)
- V6 = (Vdc / 3) * (0,5 - 0,866j)
- V7 = ((2 * Vdc) / 3) * (1 + 0j)
- V8 = (Vdc / Raíz(3)) * (0,866 + 0,5j)
- V9 = ((2 * Vdc) / 3) * (0,5 + 0,866j)
- V10 = (Vdc / Raíz(3)) * (0 + 1j)
- V11= ((2 * Vdc) / 3) * (-0,5 + 0,866j)
- V12 =(Vdc / Raíz(3)) * (-0.866 + 0,5j)
- V13 = ((2 * Vdc) / 3) * (-1 + 0j)
- V14=(Vdc / Raíz(3)) * (-0,866 - 0,5j)
- V15 = ((2 * Vdc) / 3) * (-0,5 - 0,866j)
- V16 =(Vdc / Raíz(3)) * (0 - 1j)
- V17= ((2 * Vdc) / 3) * (0,5 - 0,866j)
- V18 = (Vdc / Raíz(3)) * (0,866 - 0,5j)

El valor de Vdc corresponde a la tensión del punto O de la entrada del inversor de tres niveles. Los valores obtenidos a partir de las anteriores funciones son números complejos, para representar números complejos en el programa se ha utilizado la nomenclatura de α y β. La parte real de los valores complejos corresponden a la nomenclatura α y la parte imaginaria de los valores complejos corresponden a la nomenclatura β. Por tanto cada vector de tensión de los 19 vectores tendrá un primer valor con la nomenclatura α y un segundo valor con la nomenclatura β.

La figura 4.2.5 muestra un diagrama de bloques del módulo de programa *DSP2833x_Vk+1.c*. En el anexo 3 se encuentra el código del módulo programa DSP2833x_Vk+1.c.

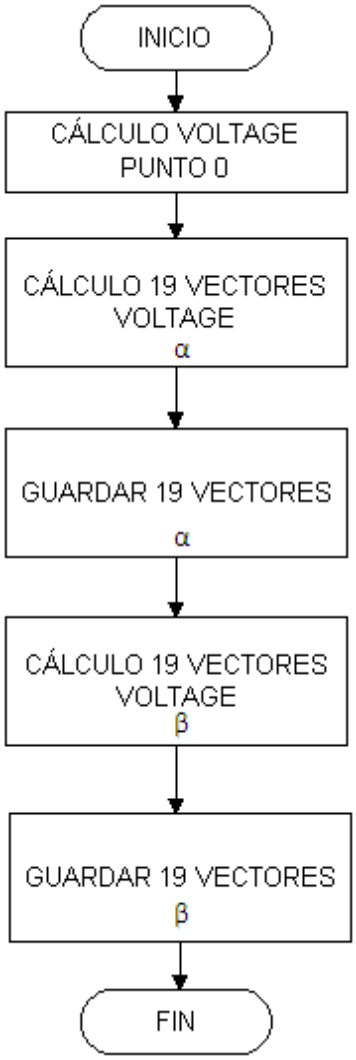

Figura 4.2.5.: Diagrama de bloques del módulo de programa *DSP2833x_Vk+1.c*.

El cálculo de los vectores de corriente I(k+1) del diagrama de bloques 4.2.2 se realiza en el módulo de programa *DSP2833x_Ik+1.c*. Este módulo de programa realiza el cálculo de los vectores de corriente del inversor de tres niveles para el siguiente estado de conmutación. El cálculo de los vectores de corriente I(k+1) se realiza a partir de la siguiente función:

$$I(k+1) = \frac{Ts}{R*Ts + L} * \left(\frac{L}{Ts} * i(k) + V(k+1) \right)$$

Los valores de Ts, R y L son variables fijas correspondientes al tiempo de conmutación, la carga resistiva y la carga inductiva, respectivamente. La variable i(k) es el valor de referencia de la corriente para el siguiente estado, esta variable se obtiene del módulo de programa *DSP2833x_ik.c*. Los valores de los vectores de corriente V(k+1) se obtienen del

módulo de programa *DSP2833x_Vk+1.c*. Los valores obtenidos a partir de la función I(k+1) son valores complejos con lo cual se utiliza la nomenclatura α y β.

Para aplicar el cálculo de la función I(k+1) en el módulo de programa *DSP2833x_Ik+1.c*, se realiza un bucle de 19 repeticiones, cada repetición corresponde a un vector de corriente diferente. En la figura 4.2.6 se muestra un diagrama de bloques que corresponde al módulo de programa *DSP2833x_Ik+1.c*. En el anexo 4 se encuentra el código del módulo programa DSP2833x_Ik+1.c.

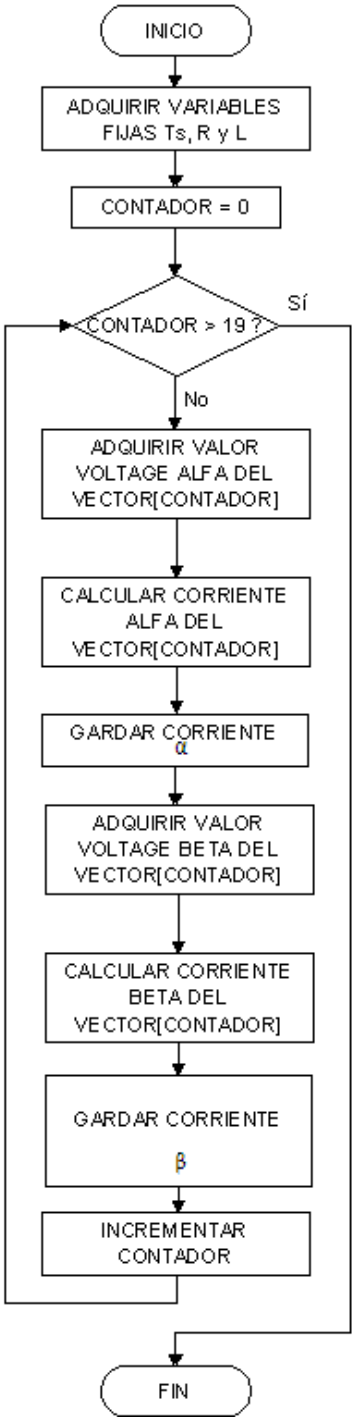

Figura 4.2.6.: Diagrama de bloques del módulo de programa *DSP2833x_Ik+1.c*.

El cálculo de las variables fijas del diagrama de bloques 4.2.3 se realiza en el módulo de programa *DSP2833x_ik.c*. La aplicación de este módulo es generar una tabla de corriente de referencia para el inversor de tres niveles. Este módulo multiplica un factor de la consigna de corriente de referencia deseada con una tabla de referencia de la corriente a 1A. La figura 4.2.7 muestra un diagrama de bloques del módulo de programa *DSP2833x_ik.c*. En el anexo 5 se encuentra el código del módulo programa DSP2833x_ik.c.

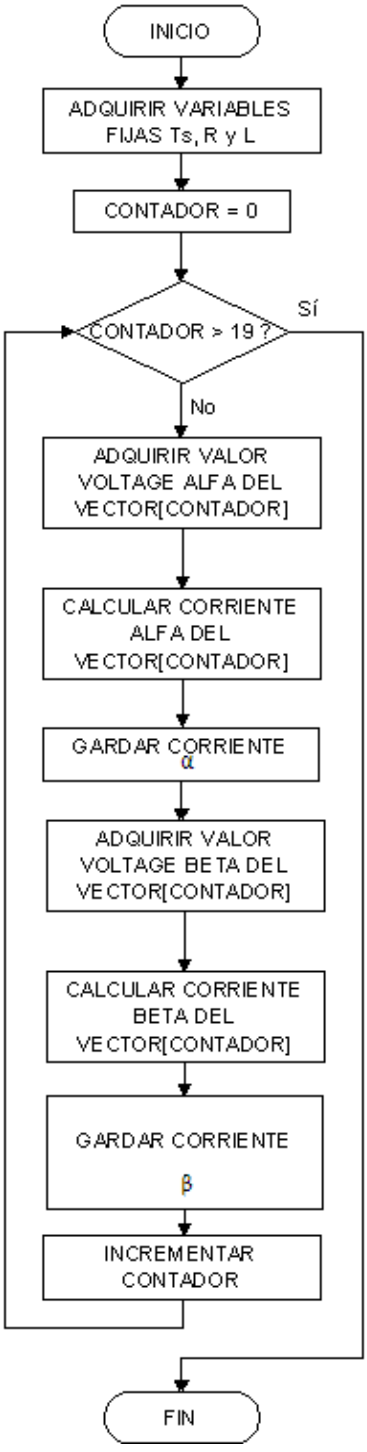

Figura 4.2.7.: Diagrama de bloques del módulo de programa *DSP2833x_ik.c*.

El cálculo de las corrientes A, B, C y la generación de la función de calidad del diagrama de bloques 4.2.2 se realiza en el módulo de programa *DSP2833x_Vck+1_g[x].c*. Este módulo realiza la función de calidad para determinar cuál es el vector de corriente que se debe de aplicar en el siguiente estado de conmutación del inversor de tres niveles, para realizar la función de calidad es necesario calcular los valores de las corrientes en las fases A, B y C de cada vector de corriente estimado. El cálculo de las corrientes A, B y C se realizan a partir de las siguientes funciones:

- Corriente A = Raíz (2/3) * Iα(k+1)
- Corriente B = ((-Raíz (6) / 6) * Iα(k+1) + ((Raíz (2) / 2) * Iβ(k+1)
- Corriente C = ((-Raíz (6) / 6) * Iα(k+1) + ((-Raíz (2) / 2) * Iβ(k+1)

El cálculo de las anteriores funciones se realiza para cada vector de corriente estimado en I(k+1) realizado en el módulo de programa *DSP2833x_Ik+1.c*. Después de obtener los valores de las corrientes A, B y C de cada vector se obtiene las corrientes de las capacidades de la entrada al inversor de tres niveles dependiendo del vector que se está utilizando. Las corrientes de las capacidades son factores necesarios para realizar la estimación de las tensiones de los condensadores en el siguiente estado de conmutación del inversor. El cálculo de las tensiones de las capacidades se obtiene a partir de las siguientes funciones:

- Vp = Vdcp + ((Ts / C) * ICp);
- Vn = Vdcn + ((Ts / C) * ICn);

Los valores de las corrientes de las capacidades corresponden a la variable ICp y ICn que se calculan a partir de las corrientes A, B y C. Las variables Vdcp y Vdcn corresponden a las tensiones de los condensadores adquiridas en las entradas del módulo ADC. I las variables Ts y C corresponden al tiempo de conmutación y el valor de las capacidades de los condensadores de la entrada al inversor de tres niveles, respectivamente.

Seguidamente el módulo de programa *DSP2833x_Vck+1_g[x].c* realiza la función de calidad g. Una vez obtenidas las tensiones de los condensadores para cada vector de corriente se aplica la siguiente función que realiza la estimación del vector que se debe aplicar en el siguiente estado:

- g = abs | iα(k) – Iα(k+1)| + abs | iβ(k) – Iβ(K+1)| + abs | Vp – Vn|

Los factores iα(k) y iβ(k) corresponden a los valores de la corriente de referencia que se ha realizado en el módulo de programa *DSP2833x_ik.c*. Los factores Iα(k+1) y Iβ(K+1) corresponden a los valores de la corriente estimada para el siguiente estado de conmutación del inversor de tres niveles realizado en el módulo de programa *DSP2833x_Ik+1.c*. La función de calidad g genera 27 valores diferentes correspondientes los 27 valores de corriente del sistema, el valor más pequeño obtenido indica el vector que se debe de aplicar en el siguiente estado de conmutación. La figura 4.2.8 muestra un diagrama de bloques del módulo de programa *DSP2833x_Vck+1_g[x].c.*

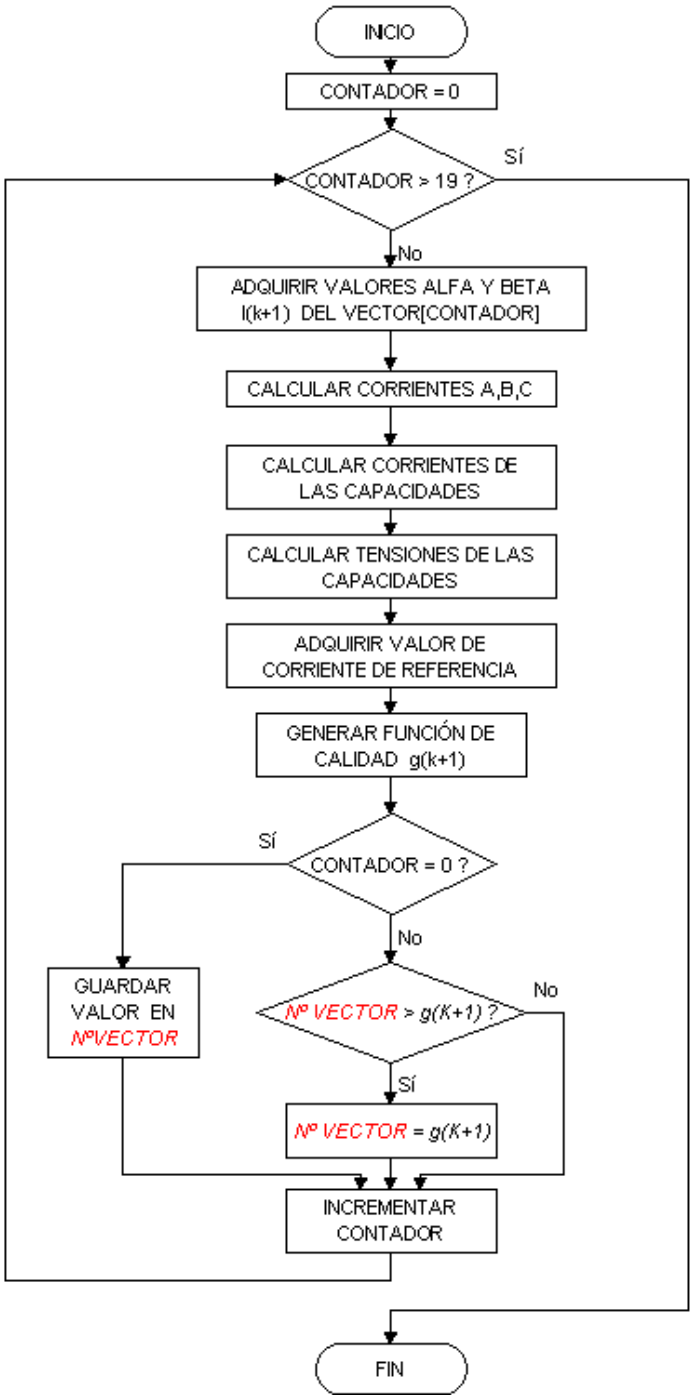

Figura 4.2.9.: Diagrama de bloques del módulo de programa *DSP2833x_Vck+1_g[x].c.*

El número de vector resultante de la función de calidad g que se aplicara en el siguiente estado de conmutación se guarda en la variable Nº VECTOR.

La aplicación del vector resultante de la función g en las salidas ePWM se realiza en el módulo de programa *DSP2833x_ConIGBTs.c*. A partir del valor de la variable Nº VECTOR, este módulo de programa realiza la configuración de las salidas ePWM para el siguiente estado de conmutación de los IGBTs del inversor de tres niveles. En el anexo 6 se encuentra el código del módulo programa DSP2833x_Vck+1_g[x].c.

4.3 Hardware:

El diagrama de bloques del equipo experimental utilizado en la aplicación del inversor de tres niveles se muestra en la figura 4.3.1.

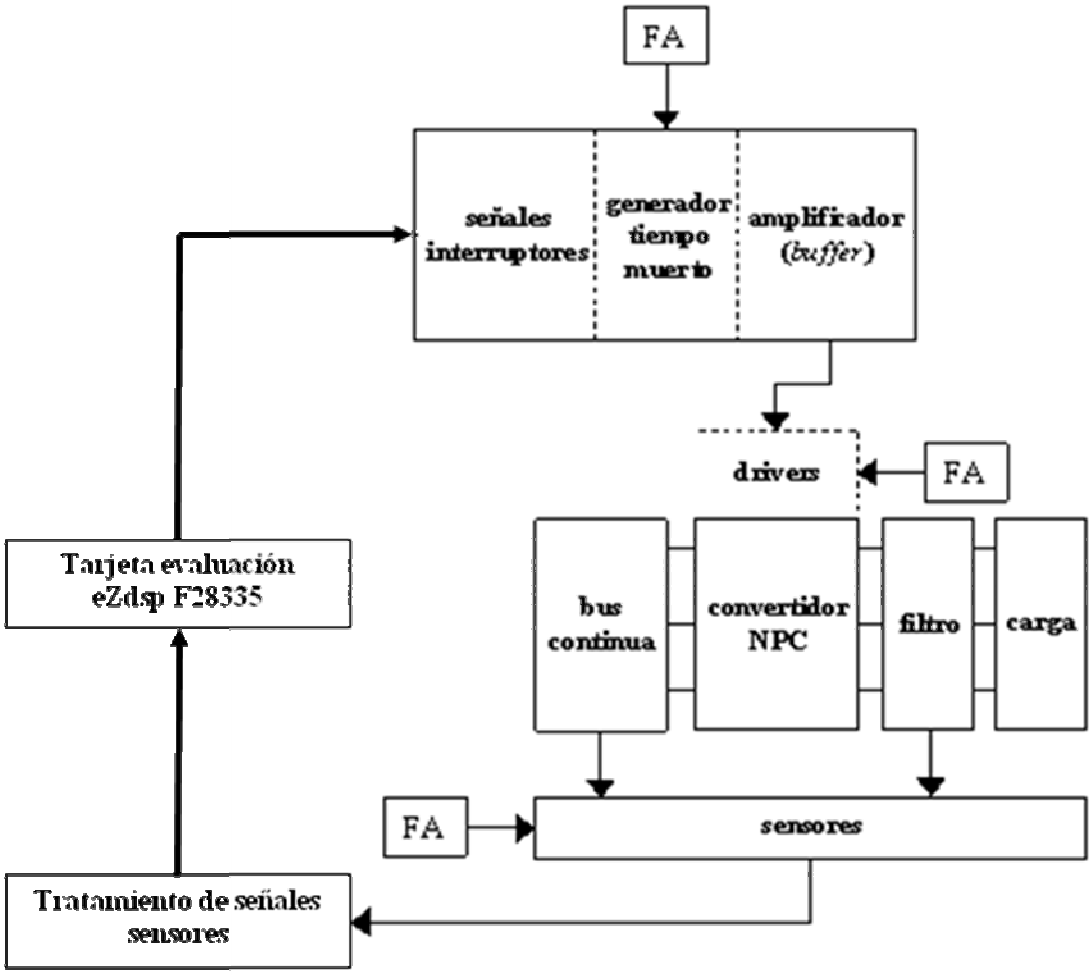

Figura 4.3.1.: Diagrama de bloques correspondiente al equipo experimental.
(FA indica fuente de alimentación)

Los elementos que forman parte del equipo experimental no se han construido dentro del desarrollo de este proyecto. Por este motivo se ha necesitado realizar una placa de tratamiento de señales intermedia entre los sensores del equipo experimental y la tarjeta de evaluación eZdsp F28335, con el fin de que la tarjeta de evaluación reconozca las señales generadas por los sensores. Igual que la realización de los conectores necesarios para la interconexión de la tarjeta de evaluación eZdsp F28335 con el equipo experimental.

La función de la placa de tratamientos de las señales generadas por los sensores es acondicionar estas señales para que la tarjeta de evaluación eZdsp F28335 realice una correcta conversión con el módulo ADC.

Los sensores del equipo experimental generan una señal que oscila entre ± 10V, esta tensión es un rango que el módulo ADC de la tarjeta de evaluación no puede soportar. El controlador de señales digitales DSC TMS320F28335 que se incluye en la tarjeta de evaluación eZdsp F238335, utiliza un módulo ADC de 12 bits de resolución con un rango de tensión de entrada por canal de 0V a 3V. En la figura 4.3.2 se muestra la placa de tratamiento de señales.

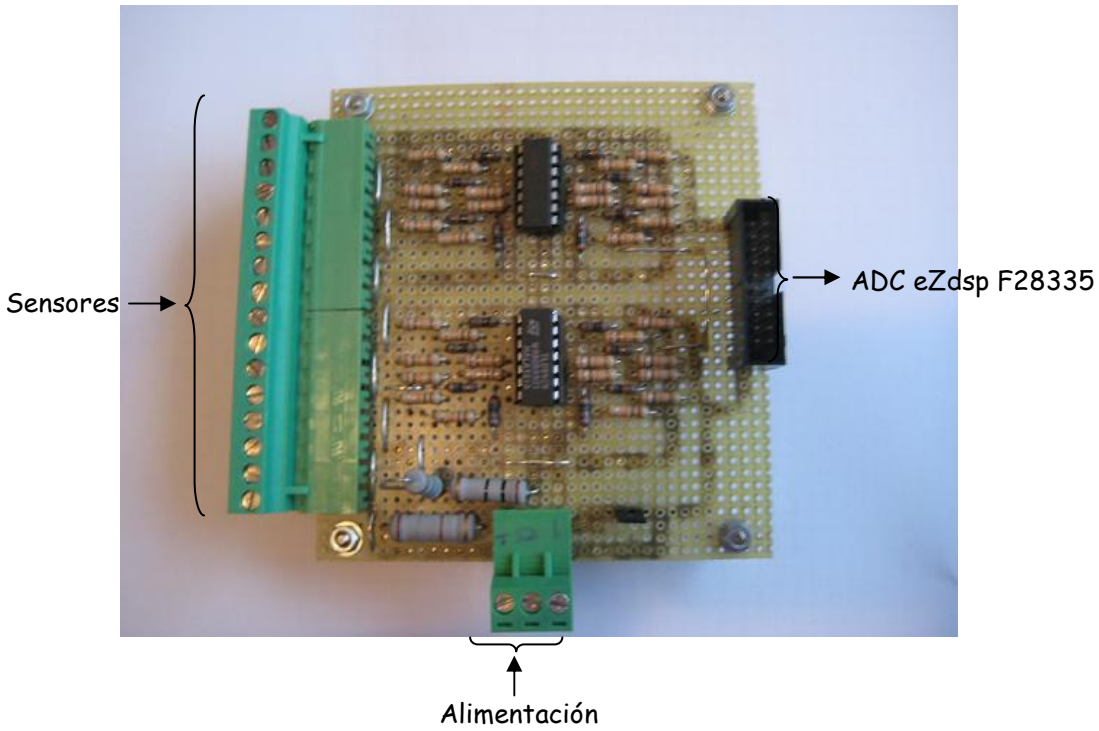

Figura 4.3.2.: Placa de tratamiento de las señales generadas por los sensores

La placa de tratamiento de señales realiza una atenuación de las señales generadas por los sensores del equipo experimental, además añade un offset a estas señales obteniendo como resultado tensiones que oscilan entre 0V y 3V. Este tratamiento se realiza a partir de un circuito con operacionales TL084CN. En la figura 4.3.3 se muestra el esquema empleado para cada señal que genera los 8 sensores del equipo experimental.

El consumo de potencia disipada en las resistencias R1 y R2 del esquema 4.3.3 es elevado. Por esta razón se ha descrito dentro de este capítulo en el apartado de resultados experimentales, una mejora de la placa de tratamiento de señales para que su consumo sea mínimo.

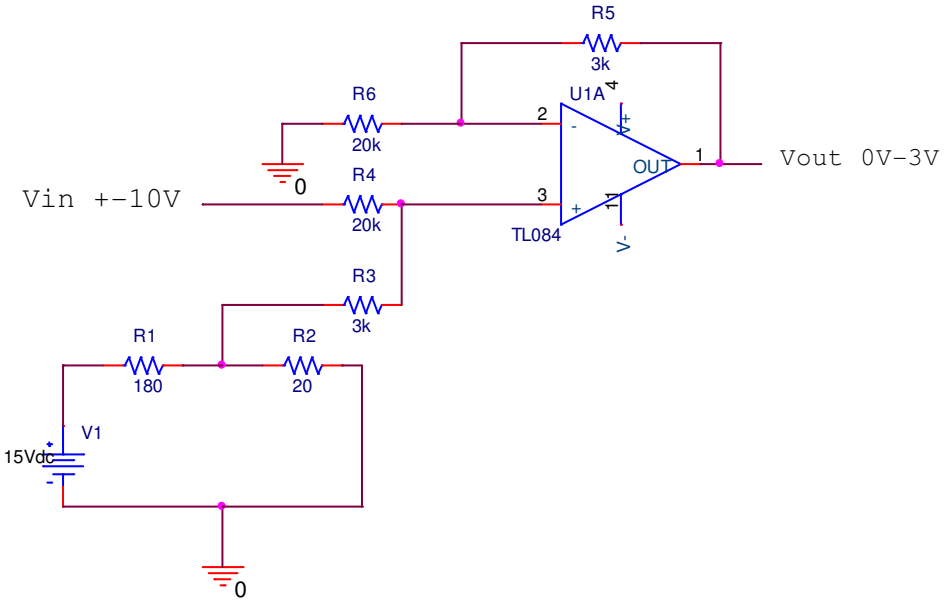

Figura 4.3.3.: Esquema de una entrada de la placa de tratamiento de señales.

La simulación resultante del esquema mostrado en la figura 4.3.3 se muestra en la figura 4.3.4.

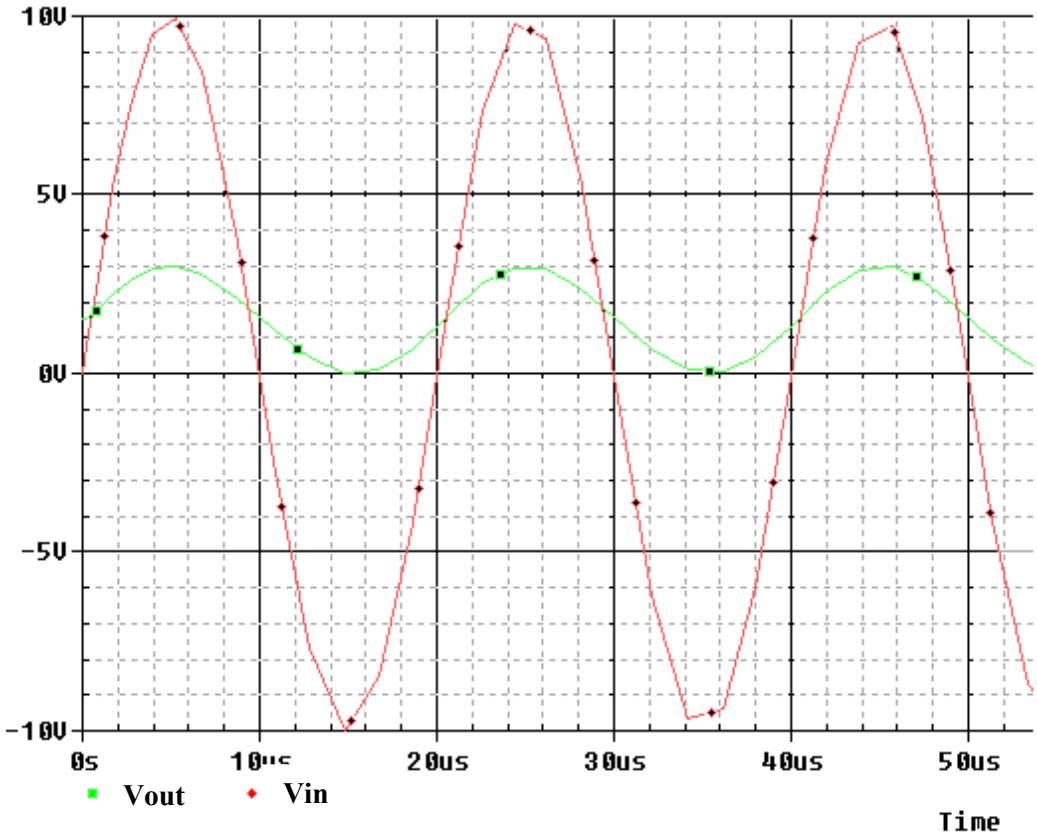

Figura 4.3.4.: Simulación de una entrada de la placa de tratamiento de señales.

En la figura 4.3.4 se observa que los sensores generan una tensión de ±10V y la placa de tratamiento de las señales genera una salida 0V a 3V.

Para una correcta conexión del equipo experimental con la tarjeta de evaluación eZdsp F28335 se ha realizado los conectores necesarios.

En la figura 4.3.5 se muestra el conector realizado para la conexión de la placa de tratamientos de señales al periférico del módulo ADC de la tarjeta de evaluación eZdsp F28335. Este conector se compone de 20 pines de conexión.

Figura 4.3.5: Conector placa tratamiento de señales al periférico del módulo ADC eZdsp F28335.

En la figura 4.3.6 se muestra los conectores necesarios para la conexión de la tarjeta de evaluación eZdsp F28335 con la placa generadora de señales ePWM. Este conexionado se compone de dos conectores, y una placa de conexión. Los conectores son de 10 y 40 pines de conexión.

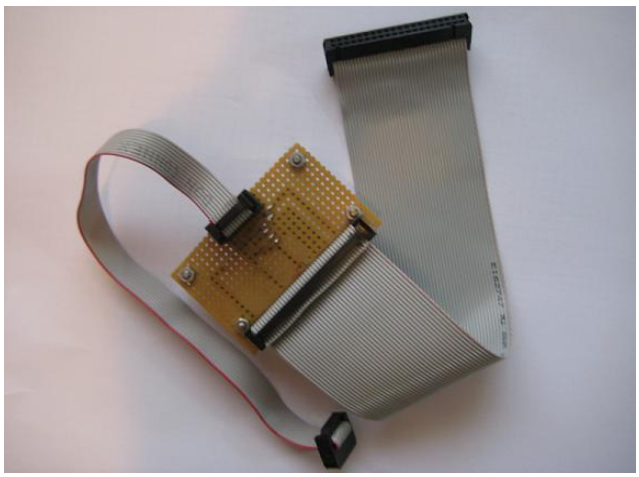

Figura 4.3.6.: Conexionado de la tarjeta de evaluación eZdsp F28335 a la placa de drivers de los IGBTs

La tarjeta de evaluación eZdsp F28335 entrega seis señales ePWM, que suponen seis señales de conmutación de los doce interruptores del inversor de tres niveles. Es necesario generar las seis señales ePWM restantes a partir de las señales de salida ePWM de la

tarjeta de evaluación. La generación se realiza mediante puertas inversoras. En la figura 4.3.7 se muestra la placa que genera las señales ePWM restantes.

La placa que genera las señales ePWM restantes realiza la generación de tiempos muertos para evitar posibles cortocircuitos causados por los cambios de conmutación de los semiconductores. En cada señal de conmutación se ha incorporado una línea de retardo que retrasa el flanco de subida (puesta en conducción).

La salida de la línea de retardo (una puerta inversora) no entrega corriente suficiente para atacar la etapa de entrada del *driver* de cada interruptor, por este motivo contiene una etapa de amplificación (*buffer*) para cada interruptor.

La figura 4.3.7 se muestra la placa que realiza el conjunto de generación de las señales ePWM restantes, generación de tiempos muertos y amplificador para los interruptores.

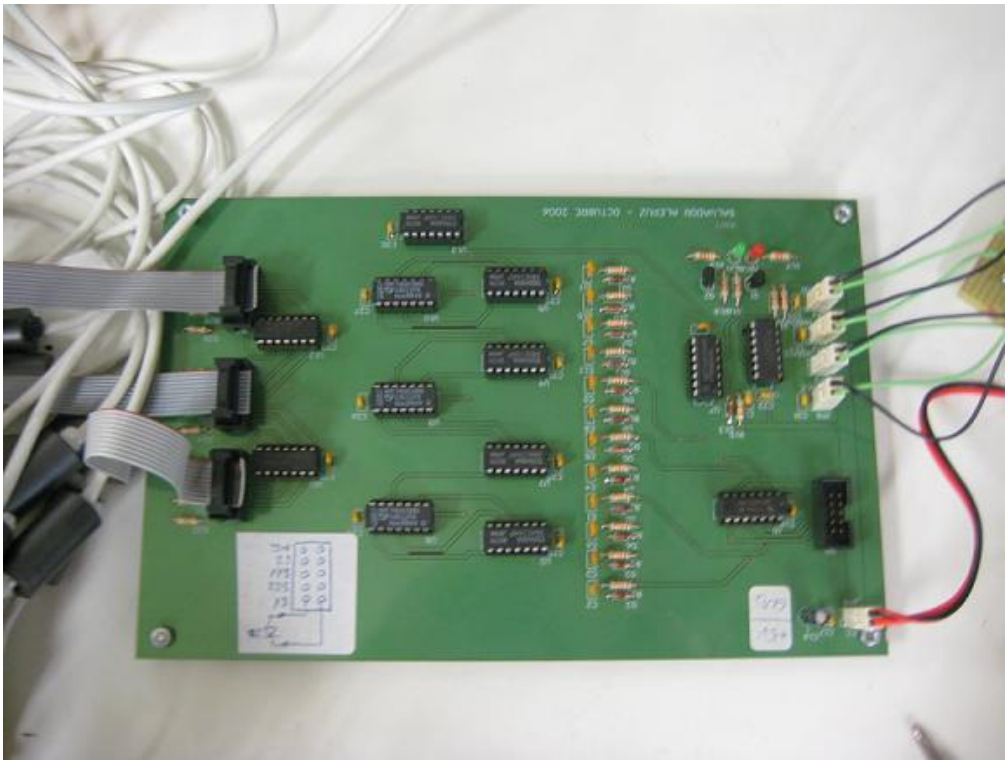

Figura 4.3.7.: Adaptación de las señales entre la tarjeta de evaluación eZdsp F278335 y los *drivers* del convertidor NPC.

El inversor de tres niveles tiene la potencia de 1kW a una tensión de $250V_{DC}$ y con una corriente máxima de $4A_{DC.}$ Los resultados experimentales de proyecto se han obtenido con este inversor que se describe a continuación.

La figura 4.3.8 muestra el inversor de tres niveles utilizado en este proyecto, contiene cuatro placas de circuito impreso, una placa para el bus de continua en parte superior más tres placas, una por rama del inversor. Su singular estructura tipo '*sandwich*', ver figura 4.3.9, permite compactar los elementos del convertidor y reducir su volumen considerablemente.

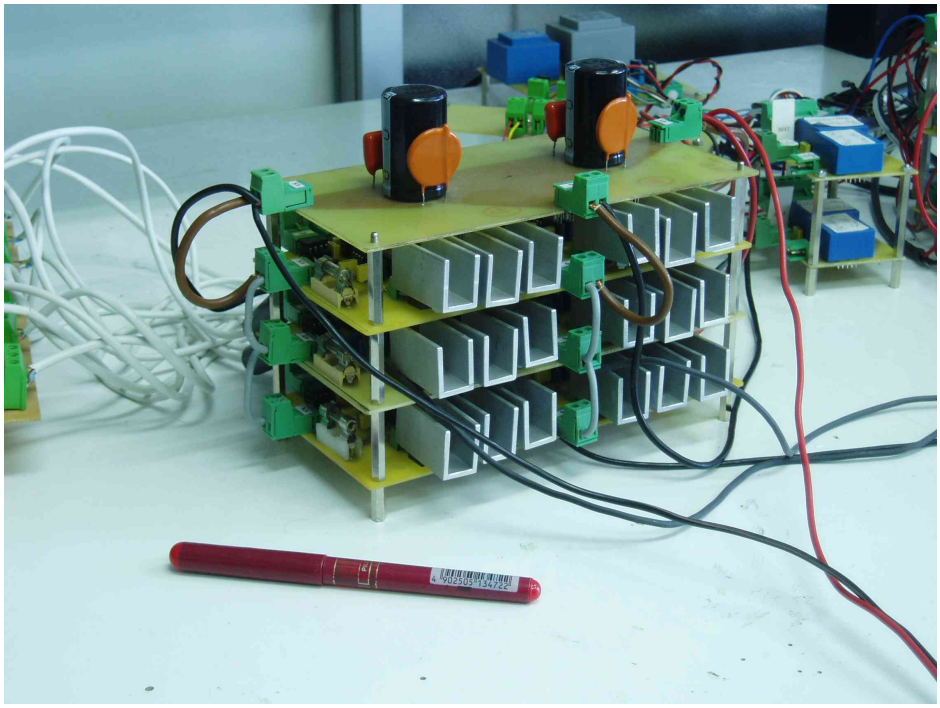

Figura 4.3.8. Inversor de tres niveles.

La figura 4.3.10 muestra el detalle del bus de continua, que contiene dos capacidades electrolíticas de 470 µF (250 V), en paralelo con dos capacidades de polipropileno metalizado (MKT) de 470 nF (400 V), que permiten eliminar los rizados de alta frecuencia. La protección de las capacidades frente a sobretensiones se realiza con un varistor de 250 V, en paralelo con cada capacidad.

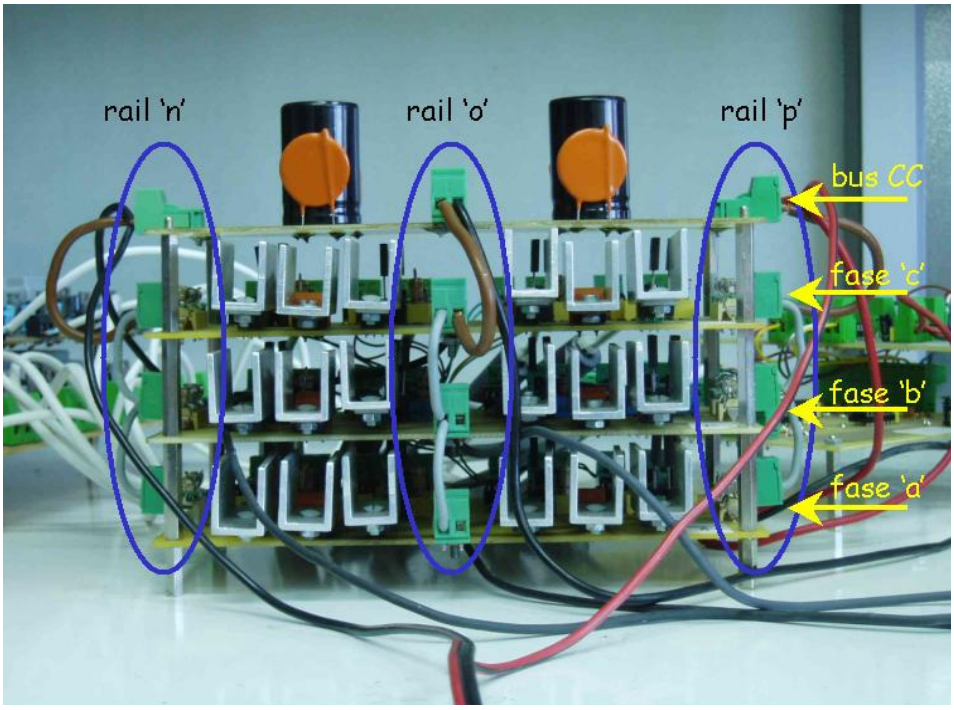

Figura 4.3.9. Inversor de tres niveles. Vista frontal.

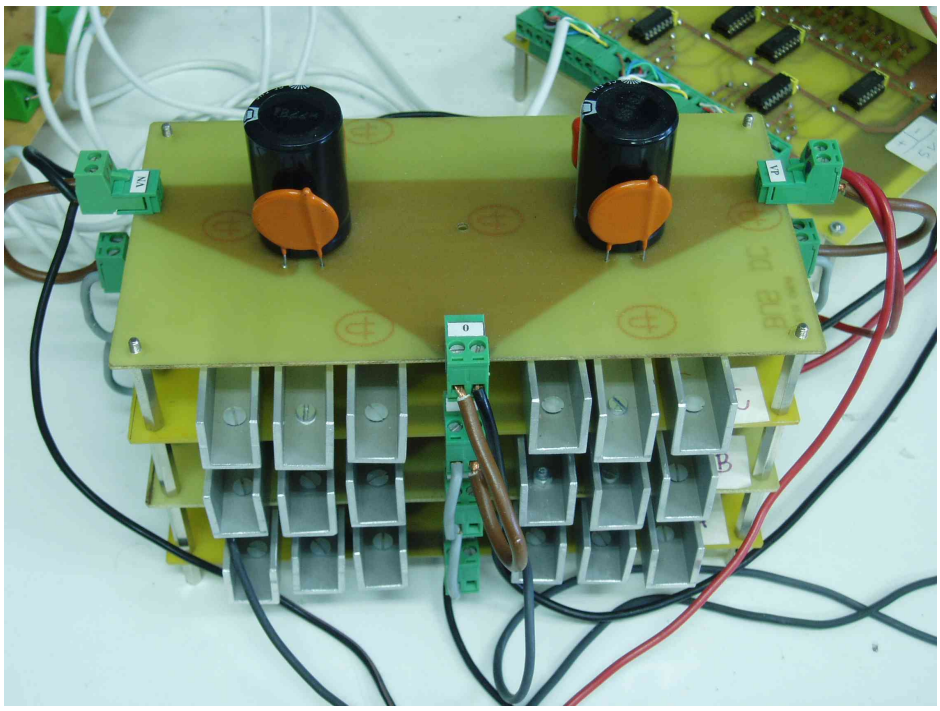

Figura 4.3.10.: Inversor de tres niveles. Vista superior.

Retirando la placa del bus de continua la figura 4.3.11 muestra la rama correspondiente a la fase 'c' del inversor de tres niveles. Se observan los raíles 'p', 'o' y 'n' procedentes del bus de continua, las entradas de las señales de conmutación y de la alimentación flotante ± 15 V de los *drivers* (circuitos de mando) de los cuatro interruptores. La salida de la fase 'c' está

ubicada debajo de la placa, mediante el tornillo situado en el centro de ella. Cada radiador está atornillado sobre un semiconductor de la rama. Los dispositivos que hay montados son el IGBT HGTP12N60C3D de Harris (600 V – 24 A), que incorpora un diodo en antiparalelo y el diodo MUR1560 de Intersil (600 V – 15 A).

Los cables que conectan las fuentes de alimentación con los *drivers* contiene toroides cuya función es evitar interferencias electromagnéticas (*spike killer*). En cada rama hay instalado fusibles en las conexiones a los raíles 'p' y 'n'.

Todas las entradas al convertidor se realizan a través de regletas enchufables, y los semiconductores de potencia están fijados mediante regletas atornillables. Aquellos dispositivos susceptibles de averiarse están montados sobre zócalos, lo que facilita su sustitución.

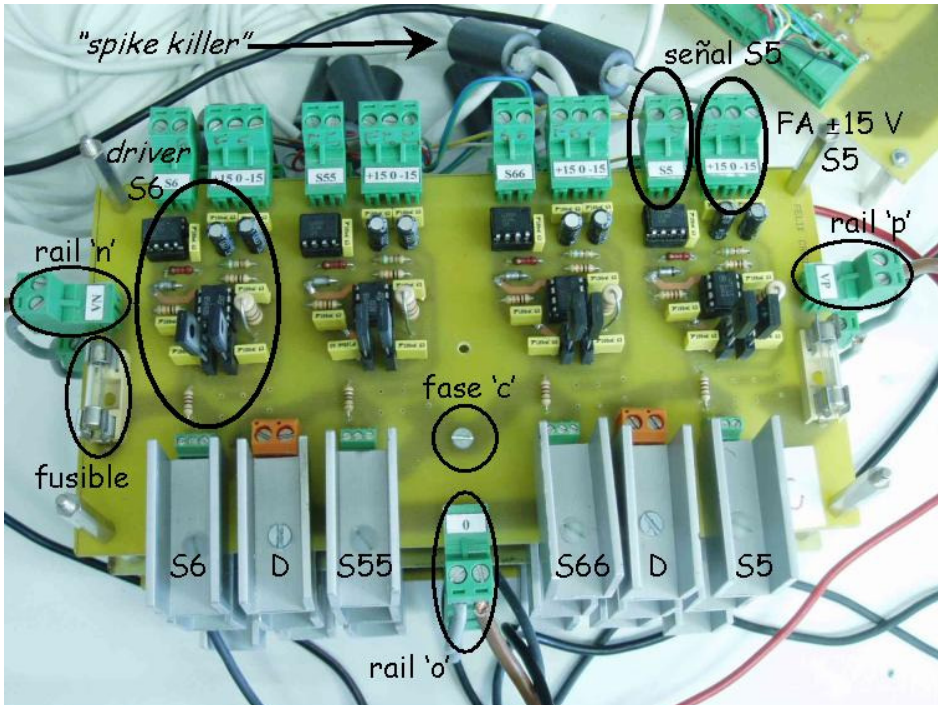

Figura 4.3.11.: Rama del convertidor NPC (fase 'c'). Vista superior.

La compacidad del inversor de tres niveles tiene sus desventajas en términos de evacuación de calor. Los radiadores son simples perfiles de aluminio, insuficientes si se desea hacer trabajar el convertidor de forma continua en la banda alta de su margen de potencia admisible y hace necesario el empleo de ventilación forzada.

Las cargas empleadas en las pruebas experimentales se muestran en la figura 4.3.12. En cada fase, se han conectado tres bobinas en serie de 1 mH, para ofrecer un total de 3 mH

por fase, con una corriente de saturación de 11 A. Se ha optado por una estructura '*sandwich*', con objeto de reducir el volumen ocupado por las bobinas

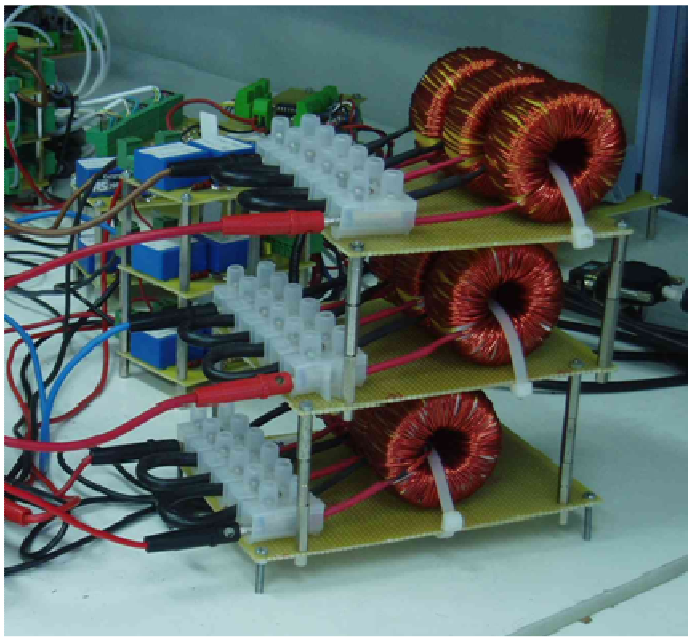

Figura 4.3.12.: Bobinas de 1 mH.

En la figura 4.3.13 se observa la carga resistiva empleada. Consta de 3 etapas trifásicas, conectables independientemente en paralelo, con un valor resistivo de 500 Ω por fase, con conexión estrella y una disipación máxima de 1000 W.

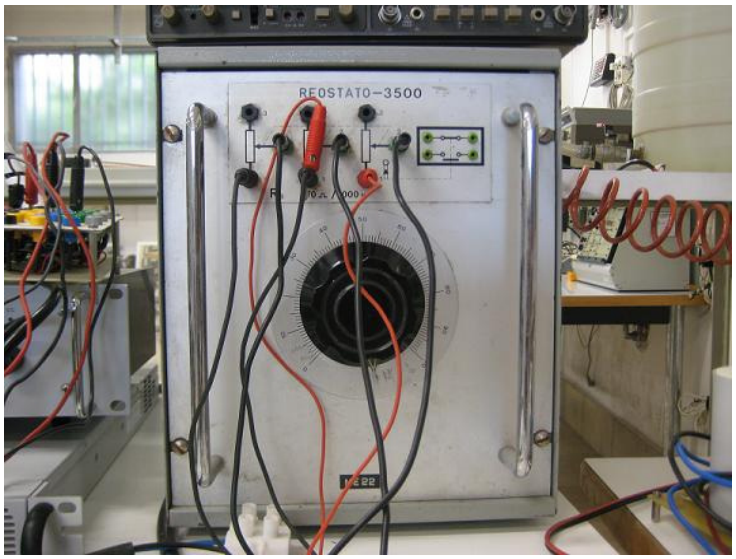

Figura 4.3.13.: Carga resistiva variable por tramos.

Los sensores que contiene el equipo experimental son sensores de efecto Hall LEM LV 25-P (tensión) y LEM LA 25-NP (corriente), ver figura 4.3.14. Cada circuito impreso contiene dos sensores, uno de tensión y otro de corriente. En función del número de tensiones y/o corrientes a medir, se emplea un número apropiado de circuitos impresos. El apilado de sensores mediante la estructura '*sandwich*' permite compactar los circuitos impresos,

reducir el volumen y se simplifica la distribución de la alimentación de los sensores. Ambos sensores se adaptan a las necesidades del sistema empleado, permiten ser configurados para medir diferentes rangos de tensiones y corrientes, ofrecen aislamiento, requieren alimentación bipolar ± 15 V y su salida se ha configurado apropiadamente para entregar tensiones dentro del rango ± 10 V.

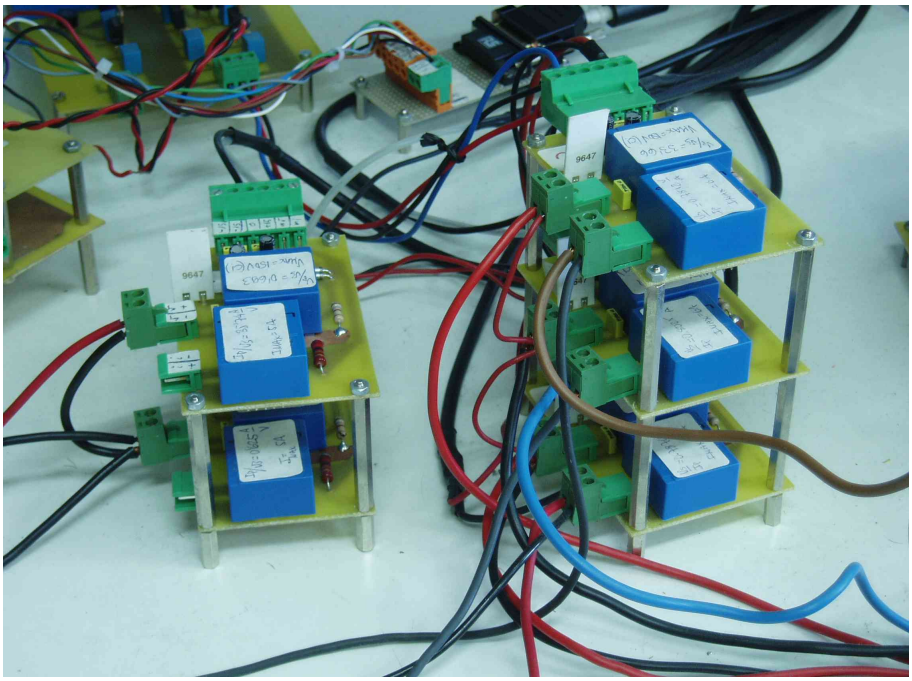

Figura 4.3.14.: Sensores de efecto Hall LEM LV 25-P y LEM LA 25-NP.

La figura 4.3.15 muestra una imagen de todo el equipo experimental.

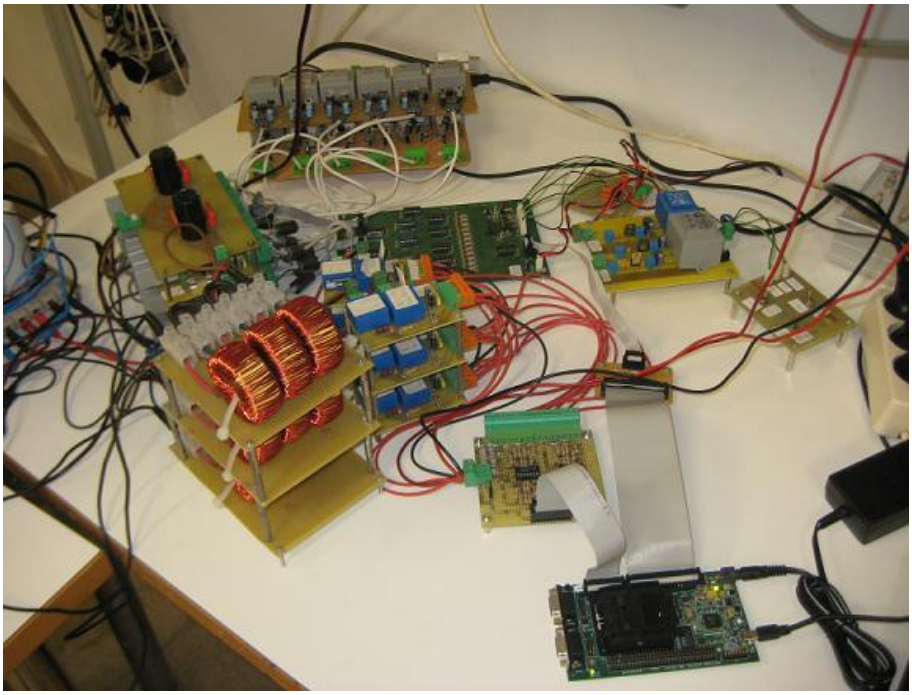

Figura 4.3.15.: Imagen del equipo experimental.

4.4 Resultado experimental:

En la aplicación experimental del inversor de tres niveles se ha alimentado con una tensión de bus de continua de 30V. I se ha fijado una corriente de referencia de 0,5A en el software desarrollado para la aplicación. A partir de estas dos variables, tensión del bus de continua y valor de la corriente de referencia, el inversor de tres niveles debe de generar tres señales de corriente de salida (Fase A, Fase B y Fase C) que contengan una frecuencia de 50Hz y una amplitud máxima de 0,5A. La razón de generar las tres señales a una frecuencia de 50Hz, es por la aplicación de esta corriente hacia la red. La amplitud máxima de 0,5A está limitada por la corriente de referencia fijada en el software de la aplicación.

El conexionado de las tres señales generadas por el inversor de tres niveles hacia la red de tomas de corriente, debe de cumplir las siguientes especificaciones:

1. Frecuencia de la señal generada 50Hz.
2. Amplitud de la señal generada dependiendo de la consigna de corriente de referencia fijada en software.
3. Desfase entre las señales de corriente generadas ha de ser de 120º
4. Generar señales senosoidales perfectas

La medición se ha realizado utilizando una pinza de corriente (100mV/1A) y un osciloscopio analógico con la configuración de 20mV división a una frecuencia de 5ms división.

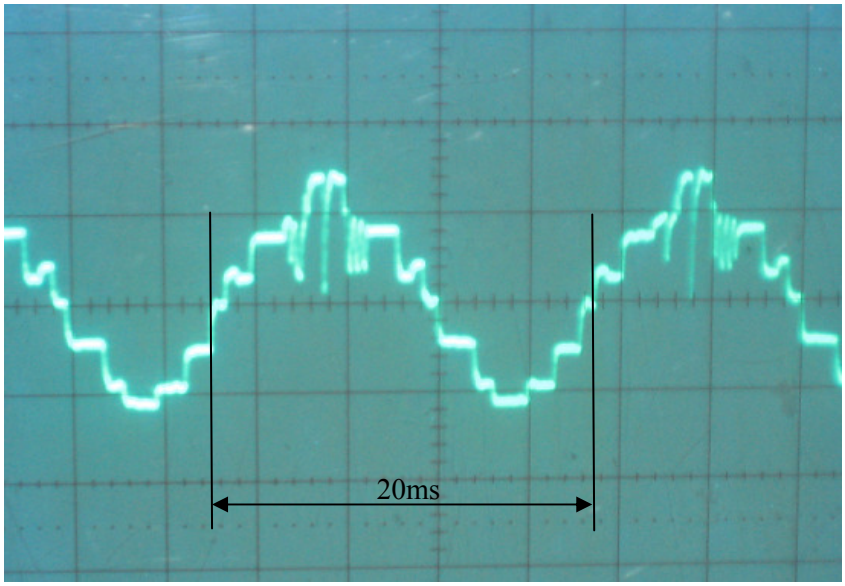

Figura 4.4.1.: Fase de corriente A (Osciloscopio: Canal = 20mV/div., Freq: 5ms/div).
(Pinza amperimétrica: 100mV/1A.)

La fase de corriente A mostrada en la figura 4.4.1 cumple con la amplitud de la corriente de referencia fijada de 0,5A máximo y la frecuencia a la que esta generada es de 50Hz. Esta señal no muestra una forma de onda senosoidal perfecta.

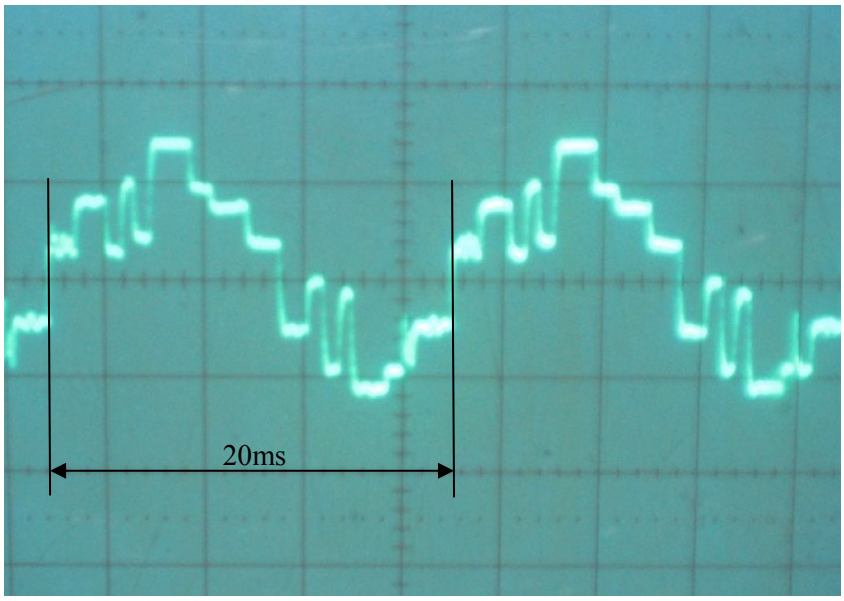

Figura 4.4.2.: Fase de corriente B (Osciloscopio: Canal = 20mV/div., Freq: 5ms/div).
(Pinza amperimétrica: 100mV/1A.)

La fase de corriente B mostrada en la figura 4.4.2 cumple con la amplitud de la corriente de referencia fijada de 0,5A máximo y la frecuencia a la que esta generada es de 50Hz. Esta señal no muestra una forma de onda senosoidal perfecta e incluso esta peor generada que la fase de corriente A.

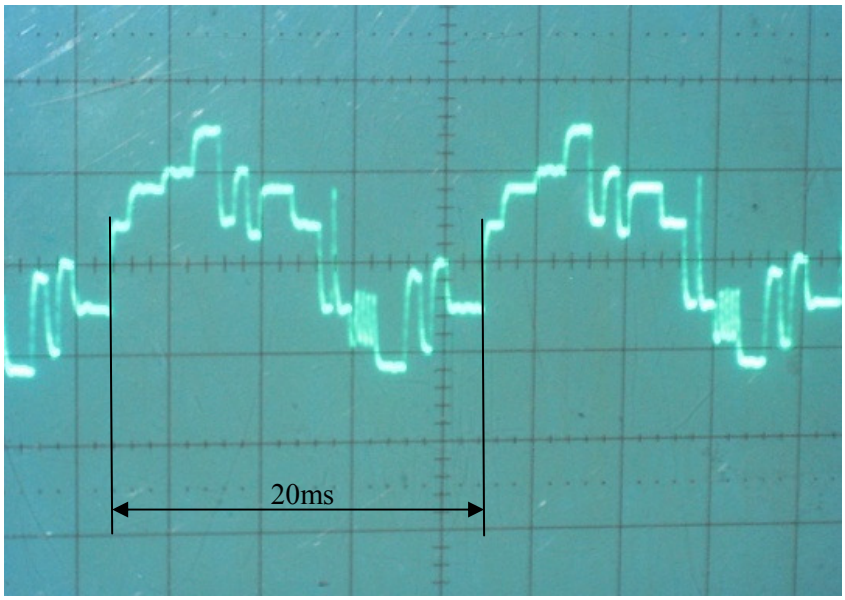

Figura 4.4.3.: Fase de corriente C (Osciloscopio: Canal = 20mV/div., Freq: 5ms/div).
(Pinza amperimétrica: 100mV/1A.)

La fase de corriente C mostrada en la figura 4.4.3 cumple con la amplitud de la corriente de referencia fijada de 0,5A máximo y la frecuencia a la que esta generada es de 50Hz. Esta señal no muestra una forma de onda senosoidal perfecta e incluso esta peor generada que la fase de corriente B.

El desfase entre las señales generadas para la fase A y la fase B se muestra en la figura 4.4.4. Se observa que el desfase entre las corrientes generadas es correcto, cumplen con un grado de desfase de 120º.

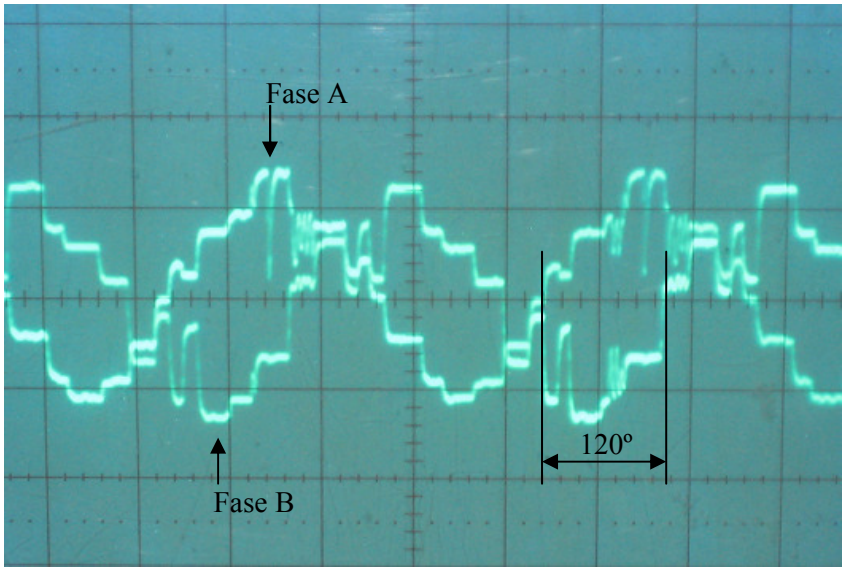

Figura 4.4.4.: Fase de corriente A y B (Osciloscopio: Canal = 20mV/div., Freq: 5ms/div).
(Pinza amperimétrica: 100mV/1A.)

Los resultados experimentales obtenidos de la aplicación no son óptimos, no cumplen los cuatro puntos anteriormente descritos para la conexión de las corrientes a la red. Por este motivo se ha decidido realizar una descripción de las posibles mejoras posteriores al trabajo realizado dentro de este proyecto, de esta manera es posible llegar a la realización de la aplicación del inversor de tres niveles en condiciones optimas.

La primera mejora a realizar sobre el trabajo desarrollado dentro de este proyecto es eliminar el escalonamiento de las señales de corrientes generadas por el inversor de tres niveles. Este escalonamiento se genera a causa de aplicar durante un número seguido de periodos de muestreo el mismo vector de conmutación. Una posible causa de la aplicación del mismo vector de conmutación durante varios estados seguidos podría ser la falta de resolución de los cálculos que realiza el software del sistema. Una mayor resolución en las variables y en los cálculos que realiza el software de la aplicación podría ser la solución

para la eliminación del escalonado de las señales generadas, pero a consecuencia de aplicar una mayor resolución en los cálculos se incrementa el tiempo necesario para realizar el cálculo del sistema. Por tanto si incrementamos la resolución de los cálculos, disminuimos la frecuencia de las señales de corriente generadas por el inversor de tres niveles por debajo de 50Hz.

Una segunda mejora del trabajo desarrollado dentro de este proyecto, es rediseñar la placa de tratamiento de señales de los sensores que se muestra en la figura 4.3.3. Esta placa de tratamiento de señales tiene un consumo alto en la generación de la tensión de referencia. El funcionamiento óptimo de cualquier placa es realizar su función con el mínimo consumo posible. El rediseño de la placa de tratamiento de señales incluye un regulador de tensión a 5V (LM7805), implementando este regulador las resistencias para generar la tensión de referencia son de 1/4W, en cambio en la figura 4.3.3 estas resistencias son de 3W. En la figura 4.4.5 se muestra el esquema de la mejora propuesta para la placa de tratamiento de señales de los sensores.

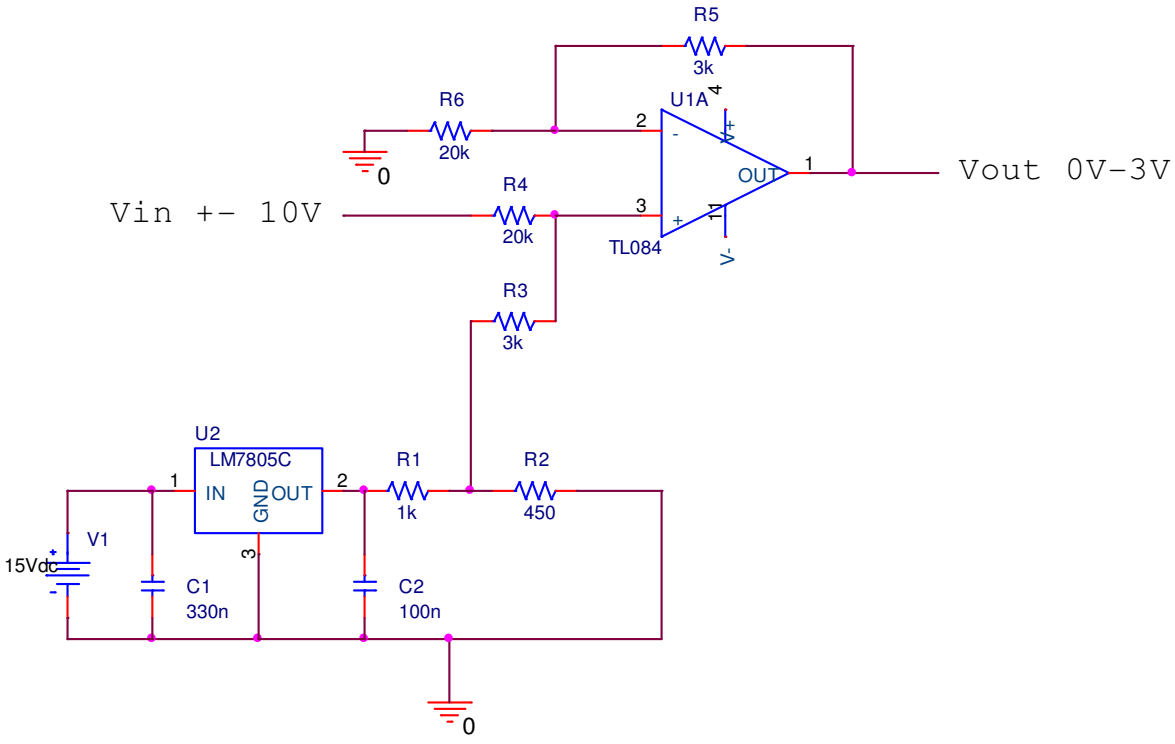

Figura 4.4.5.: Esquema de la placa de tratamiento de señales mejorada

En la figura 4.4.5 se puede observar la simulación del circuito mejorado para la placa de tratamiento de señales. Se observa que el resultado de la simulación es el mismo que en la figura 4.3.4.

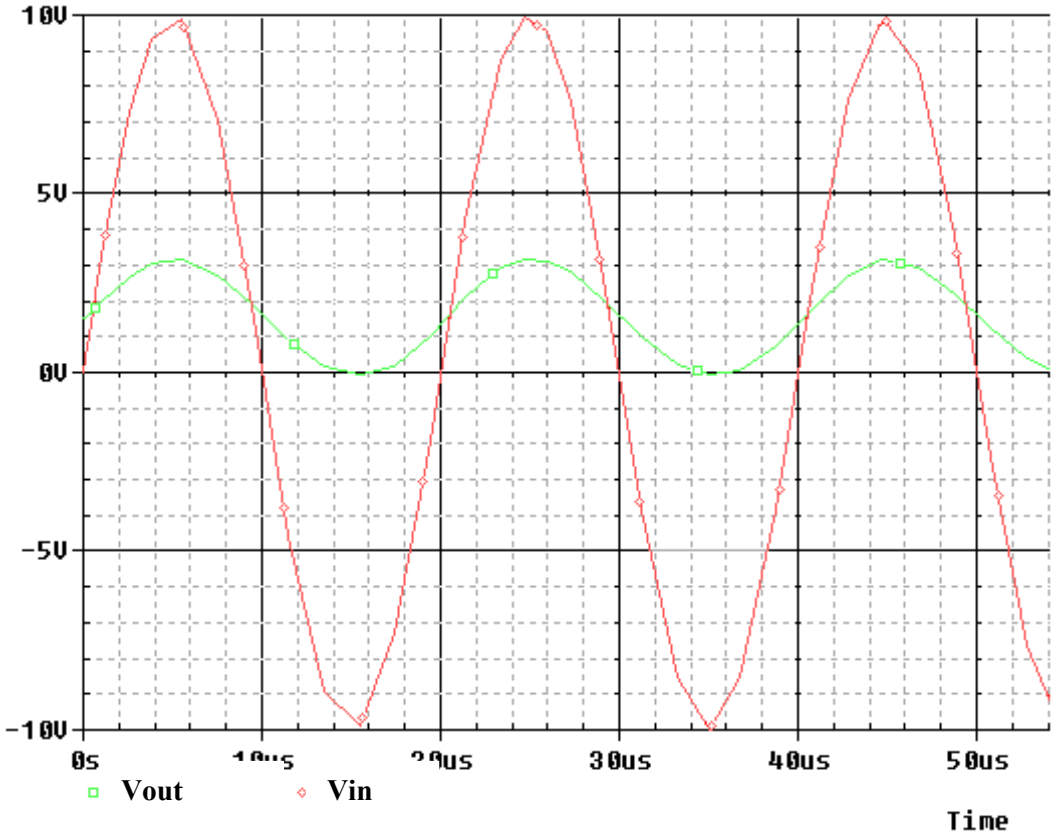

Figura 4.4.6.: Simulación del esquema mostrado en la figura 4.4.5

PRESUPUESTO:

El presupuesto incluye el coste de los materiales empleados en el proyecto, tales como el coste de la tarjeta de evaluación y su puesta a punto, el coste de realización de la placa de tratamiento de señales de entrada al ADC y el coste de los cables de conexionado realizados. Se incluye también los costes generados por el uso de recursos necesarios para el desarrollo del proyecto.

Descripción	**Unidades**	**Precio Unidad**	**Total**
Tarjeta de evaluación eZdsp F28335	1	373,92 €	373,72 €
Placa de tratamiento de señales ADC:			
- Componentes de Material	78	Varios	22,25 €
- Horas de trabajo	16h	15 €/h	240 €
Cables de conexión:			
- Componentes de Material	9	Varios	9,75 €
- Horas de trabajo	8h	15 €/h	120 €
Puesta a punto tarjeta evaluación:			
- Componentes de Material	3	1,50 €	4,50 €
- Horas de trabajo	4h	15 €/h	60 €
Otros:			
- Horas de uso de Ordenador	1280h	0,25 €/h	320 €
- Horas de uso de Osciloscopio	230h	0,15 €/h	34,50 €
		Subtotal	1184.72 €
		IVA 16%	189,56 €
		Total	**1374,28 €**

CONCLUSIONES:

El trabajo desarrollado dentro del proyecto se divide en dos partes principales. Realización de una guía de ayuda para utilizar los recursos de programación que incluye la tarjeta de evaluación eZdsp F28335. Y el desarrollo de una aplicación de control predictivo sobre un inversor de tres niveles con la implementación de la tarjeta de evaluación eZdsp F28335.

En el desarrollo del trabajo realizado en este proyecto, se ha necesitado tiempo para entender los recursos que engloba la aplicación del inversor de tres niveles. Principalmente estos recursos se dividen en tres partes. La primera parte ha sido entender los recursos de la tarjeta de evaluación eZdsp 28335. La segunda parte ha sido entender la secuencia de ejecución del control predictivo. Y la tercera parte ha sido conocer el equipo experimental del inversor de tres niveles.

Un gran tiempo del desarrollo del proyecto se ha empleado en entender los recursos de la tarjeta de evaluación eZdsp F28335, por este motivo se ha decido realizar una guía de inicio de programación para Code Composer Studio. Otro tiempo considerable se ha empleado en la puesta en marcha de la tarjeta de evaluación eZdsp F28335 junto el desarrollo de los conectores necesarios y una placa de tratamiento de señales para la interconexión del inversor de tres niveles a la tarjeta de evaluación eZdsp F28335. Y el resto del tiempo se ha dedicado en la realización de la aplicación del inversor de tres niveles y la documentación del proyecto.

La comercialización de la aplicación desarrollada dentro de este proyecto es posible por dos motivos. Uno de los motivos es el amplio número de aplicaciones dentro de la industria que se utilizan inversores de potencia para el control de motores de inducción. Otro de los motivos son las características que contiene la tarjeta de evaluación eZdsp F28335, esta tarjeta de evaluación es una tarjeta autónoma, esto significa que esta tarjeta de evaluación una vez programada para una aplicación es autosuficiente para la ejecución y control de la misma.

A partir de los resultados experimentales obtenidos se ha descrito posibles mejoras posteriores al trabajo desarrollado dentro de este proyecto.

BIBLIOGRAFÍA:

- **TMS320C28x Assembly Language Tools v5.0.0 User's Guide**, Texas Instruments 2007.
- **TMS320x2833x Analog-to-Digital Converter (ADC) Module Reference Guide,** Texas Instruments 2007.
- **Code Composer Studio Development Tools v3.3 Getting Started Guide,** Texas Instruments 2006.
- **TMS320F28335/28334/28332 Digital Signal Controllers (DSCs) Data Manual,** Texas Instruments 2008.
- **TMS320x280x, 2801x, 2804x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide,** Texas Instruments 2008.
- **eZdspTM F28335 Technical Reference**, Spectrum Digital 2007.
- **TMS320x2833x, 2823x System Control and Interrupts Reference Guide,** Texas Instruments 2008.
- **TMS320C28x Optimizing C/C++ Compiler v5.0.0 User's Guide**, Texas Instruments 2007.
- **Librarías ANSI C,** Steven R. Davidson y Salvador Pozo 2003.
- **Predictive Direct Torque Control of an Induction Machine,** Universidad Técnica Federico Santa María y Universitat Stuttgart, Institut für Leistungselektronik und Regelungstechnik Pfaffenwaldring
- **Predictive control of half-bridge single-phase active filter,** Antonio Dell.Aquila, Agostino Lecci, Marco Liser, Italia.
- **Efficient Predictive Current Control Technique for Multilevel Voltage Source Inverters,** NATIONAL TECHNICAL UNIVERSITY OF ATHENS, G.S. Perantzakis, F.H. Xepapas, S.N. Manias.
- **Predictive Current Control of a Voltage Source Inverter**, José Rodríguez*, Senior Member, IEEE*, Jorge Pontt*, Senior Member, IEEE*, César A. Silva*, Member, IEEE*, Pablo Correa, Pablo Lezana*, Member, IEEE*, Patricio Cortés*, Student Member, IEEE*, and Ulrich Ammann, 2007.
- **Predictive Control of a Three-Phase Neutral-Point-Clamped Inverter,** René Vargas, *Student Member, IEEE*, Patricio Cortés, *Student Member, IEEE*, Ulrich Ammann, *Member, IEEE*, José Rodríguez, *Senior Member, IEEE*, and Jorge Pontt, *Senior Member, IEEE*, 2007.

- **Cost Function-Based Predictive Control for Power Converters,** Patricio Cortés, José Rodríguez, René Vargas, Ulrich Ammann, Departamento de Electrónica Universidad Técnica Federico Santa María y Institut für Leistungselektronik und Elektrische Antriebe Universität Stuttgart.
- **Discrete-Time Current Control of Voltage-Fed Tree-Phase PWM Inveter,** Osman Kukrer, ***Member,*** *IEEE***,** 1996.
- **Predictive Power Control of an AC/DC/AC Converter,** José Rodríguez, Jorge Pontt, Pablo Correa, Pablo Lezana, Patricio Cortés, Departamento de Electrónica Universidad Técnica Federico Santa María y Institut für Leistungselektronik und Elektrische Antriebe Universität Siegen.
- **Predictive Current Control of Grid-Connected Neutral-Point-Clamped Converters to meet Low Voltage Ride-Through Requirements**, S. Alepuz, S. Busquets-Monge, J. Bordonau, P. Cortés, J. Rodríguez, R. Vargas, Dept. Electronic Engineering, Technical University of Catalonia, Barcelona, Spain y Dept. Electrónica, Universidad Técnica Federico Santa María, Valparaíso, Chile.

Anexo I: Código main.c.

```
//###########################################################################
// AUTOR: Carlos Fernández Campos
// FECHA: 12-06-2009
// ARCHIVO:  main.c
// DESCRIPCIÓN:     Módulo de programa principal
//###########################################################################
//---------------------------------------------------------------------------------------------------
// Incluir librerias:
//---------------------------------------------------------------------------------------------------

#include <stdio.h>
#include <file.h>
#include <math.h>
#include <stdlib.h>

//---------------------------------------------------------------------------------------------------
// main:
//---------------------------------------------------------------------------------------------------

void main()
{
//Inicialización del Control del Sistema
        InitSysCtrl();
//Inicialización de la tabla de vectores del módulo PIE
        InitPieVectTable();
//Inicialización del control del módulo PIE
        InitPieCtrl();
//Habilitación de las interrupciones
        EnableInterrupts();
//Configuración del módulo ADC
        ConfigAdc();
//Inicialización del puerto de salida ePWM
        InitEPwmGpio();
//Generación de la tabla de referencia i*(k)
        i_k();
//Configuración del módulo ePWM
        InitEPwm();
//Generación de un bucle infinito esperando interrupciones
        while(1)
        {
                asm("nop");
        }
}
//---------------------------------------------------------------------------------------------------
// Fin del programa
//---------------------------------------------------------------------------------------------------
```

Anexo II: Código DSP2833x_AdcRef_Alfa-Beta.c.

```
//###########################################################################
// AUTOR: Carlos Fernández Campos
// FECHA: 12-06-2009
// ARCHIVO:  DSP2833x_AdcRef_Alfa-Beta.c
// DESCRIPCIÓN:     Módulo de programa de adquisición de los canales ADC y
//                  cálculo de las corrientes Alfa/Beta
//###########################################################################
//---------------------------------------------------------------------------------------------------
// Incluir librerías:
//---------------------------------------------------------------------------------------------------

#include "DSP2833x_Device.h"
#include "DSP2833x_Examples.h"
#include <math.h>
#include <stdlib.h>

//---------------------------------------------------------------------------------------------------
// Definición de variables públicas:
//---------------------------------------------------------------------------------------------------

signed int abI[2] = {0,0};                  // Resultado de las corrientes Alfa y Beta

//---------------------------------------------------------------------------------------------------
// AdcRef_Alfa_Beta:
//---------------------------------------------------------------------------------------------------

void AdcRef_Alfa_Beta()
        {
        // Definición de variables del módulo del programa
        #define FACTOR          1000;           // Factor de multiplicación
        #define REFADC        0x0800;           // Valor de 1,5V de la entrada ADC
        signed int ADCResult[6] = {0,0,0,0,0,0}; // Declaración del vector de resultado
                                                 // del ADC
        signed int d = 0, e = 0, f = 0, h = 0, i = 0;  // Definición de variables para realizar
                                                       // la  matriz Alfa y Beta
        volatile Uint16 *pAdc, j = 0;            // Definir variable del bucle y puntero

        //-----------------------------------------------------------------------------------------
        // Ajuste de los valores adquiridos del ADC a la tensión de referencia 1,5V:
        //-----------------------------------------------------------------------------------------
        pAdc = &AdcMirror.ADCRESULT0;       // Posicionar el puntero pAdc a la
                                            // dirección de memoria del primer
                                            // resultado del ADC
        for (j = 0; j < 6; j++)             // Generar bucle de seis ciclos para las
                                            // seis entradas senosoidales
        {
        ADCResult[j]= *pAdc - REFADC;       // Restar el valor del canal ADC con el
```

```
                                        // valor de la tensión de referencia (0x0800).
                                        //El resultado se guarda en el vector de
                                        //resultados
        pAdc++;                         // Incrementar posición del puntero pAdc
        }

        //-------------------------------------------------------------------------------------------
        // Cáclulo de las Correintes Alfa y Beta:
        //-------------------------------------------------------------------------------------------

        ////////////////Correintes alfa y Beta/////////
        //  ---------------------------------------  //
        //    |Xa|    | Ma1 Ma2 Ma3  |   | XR |   //
        //    |  | =|                  | * | XS |   //
        //    |Xb|    | Mb1 Mb2 Mb3  |   | XT |   //
        //  ---------------------------------------  //
        ////////////////////////////////////////////////

        //CORRRIENTE ALFA//

        d = ADCResult[3] * 816 ;            // Resultado de multiplicar IR con Ma1
        e = ADCResult[4] * - 408 ;          // Resultado de multiplicar IS con Ma2
        f = ADCResult[5] * - 408 ;          // Resultado de multiplicar IT con Ma3

        abI[0] = (d + e + f) / FACTOR;      // Resultado Alfa de las corrientes

        //CORRIENTE BETA//

        h = ADCResult[4] * 707;             // Resultado de multiplicar IS con Mb2
        i = ADCResult[5] * - 707 ;          // Resultado de multiplicar IT con Mb3

        abI[1] = (h + i) / FACTOR;          // Resultado Beta de las corrientes
}
//-------------------------------------------------------------------------------------------
// Fin del módulo de programa
//-------------------------------------------------------------------------------------------
```

Anexo III: Código DSP2833x_Vk+1.c.

```
//###########################################################################
// AUTOR: Carlos Fernández Campos
// FECHA: 12-06-2009
// ARCHIVO:  DSP2833x_Vk+1.c
// DESCRIPCIÓN:     Módulo de programa de cálculo de los vectores de tensión
//                   para el siguiente estado de conmutación Ts.
//###########################################################################
//---------------------------------------------------------------------------------------------------
// Incluir librerías:
//---------------------------------------------------------------------------------------------------

#include "DSP2833x_Device.h"
#include "DSP2833x_Examples.h"
#include <math.h>
#include <stdlib.h>

//---------------------------------------------------------------------------------------------------
// Definición de variables públicas:
//---------------------------------------------------------------------------------------------------

//Vectores para los resultados de las tensiones del siguiente estado
signed long Va_Kadd1[19] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
signed long Vb_Kadd1[19] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

//---------------------------------------------------------------------------------------------------
// V_Kadd1:
//---------------------------------------------------------------------------------------------------

void V_Kadd1()
{
        ////////////Vectores de Voltage/////////////////
        ////////////////////////////////////////////////////
        /////                    1                 /////
        /////  V(k+1) = V(k) + --- * i(k) * Ts  /////
        /////                    C                 /////
        ////////////////////////////////////////////////////
        //                                        //
        //      v0=0+0j;                          //
        //      v1=Vdc/3+0j;                      //
        //      v2=Vdc/3*(0.5+0.866j);            //
        //      v3=Vdc/3*(-0.5+0.866j);           //
        //      v4=Vdc/3*(-1+0j);                 //
        //      v5=Vdc/3*(-0.5-0.866j);           //
        //      v6=Vdc/3*(0.5-0.866j);            //
        //      v7=2*Vdc/3*(1+0j);                //
        //      v8=Vdc/sqrt(3)*(0.866+0.5j);      //
        //      v9=2*Vdc/3*(0.5+0.866j);          //
        //      v10=Vdc/sqrt(3)*(0+1j);           //
```

```
//        v11=2*Vdc/3*(-0.5+0.866j);   //
//        v12=Vdc/sqrt(3)*(-0.866+0.5j); //
//        v13=2*Vdc/3*(-1+0j);         //
//        v14=Vdc/sqrt(3)*(-0.866-0.5j);  //
//        v15=2*Vdc/3*(-0.5-0.866j);   //
//        v16=Vdc/sqrt(3)*(0-1j);      //
//        v17=2*Vdc/3*(0.5-0.866j);    //
//        v18=Vdc/sqrt(3)*(0.866-0.5j);  //
//                                     //
/////////////////////////////////////////////////////

// Definición de variables del módulo del programa
signed long Vdc = 0;                    // Variable para la tensión del punto O

// Calcular la tension del punto O de las capacidades del bus de continua
Vdc = AdcMirror.ADCRESULT6 + AdcMirror.ADCRESULT7;

// Vectores de voltage ALFA
Va_Kadd1[0] = 0;
Va_Kadd1[1] = Vdc / 3;
Va_Kadd1[2] = Vdc / 6;
Va_Kadd1[3] = Vdc / - 6;
Va_Kadd1[4] = - Vdc / 3;
Va_Kadd1[5] = Vdc / - 6;
Va_Kadd1[6] = Vdc / 6;
Va_Kadd1[7] = (2 * Vdc) / 3;
Va_Kadd1[8] = Vdc / 2;
Va_Kadd1[9] = (2 * Vdc) / 6;
Va_Kadd1[10] = 0;
Va_Kadd1[11] = (2 * Vdc) / - 6;
Va_Kadd1[12] = Vdc / - 2;
Va_Kadd1[13] = (- 2 * Vdc) / 3;
Va_Kadd1[14] = Vdc / - 2;
Va_Kadd1[15] = (2 * Vdc) / - 6;
Va_Kadd1[16] = 0;
Va_Kadd1[17] = (2 * Vdc) / 6;
Va_Kadd1[18] = Vdc / 2;

// Vectores de voltage BETA
Vb_Kadd1[0] = 0;
Vb_Kadd1[1] = 0;
Vb_Kadd1[2] = (Vdc * 1000) / 3464;
Vb_Kadd1[3] = (Vdc * 1000) / 3464;
Vb_Kadd1[4] = 0;
Vb_Kadd1[5] = (Vdc * 1000) / - 3464;
Vb_Kadd1[6] = (Vdc * 1000) / - 3464;
Vb_Kadd1[7] = 0;
Vb_Kadd1[8] = (Vdc * 1000) / 3464;
Vb_Kadd1[9] = (2 * Vdc * 1000) / 3464;
Vb_Kadd1[10] = (Vdc * 1000) / 1732;
Vb_Kadd1[11] = (2 * Vdc * 1000) / 3464;
```

```
        Vb_Kadd1[12] = (Vdc * 1000) / 3464;
        Vb_Kadd1[13] = 0;
        Vb_Kadd1[14] = (Vdc * 1000) / - 3464;
        Vb_Kadd1[15] = (2 * Vdc * 1000) / - 3464;
        Vb_Kadd1[16] = (Vdc * 1000) / - 1732;
        Vb_Kadd1[17] = (2 * Vdc * 1000) / - 3464;
        Vb_Kadd1[18] = (Vdc * 1000) / - 3464;
}
//-------------------------------------------------------------------------------------------------------
// Fin del módulo de programa
//-------------------------------------------------------------------------------------------------------
```

Anexo IV: Código DSP2833x_Ik+1.c.

```
//###########################################################################
// AUTOR: Carlos Fernández Campos
// FECHA: 12-06-2009
// ARCHIVO:  DSP2833x_Ik+1.c
// DESCRIPCIÓN:    Módulo de programa de cálculo de las predicciones de las
//                 corrientes de las fases para el siguiente estado Ts.
//###########################################################################
//---------------------------------------------------------------------------------------------------
// Incluir librerias:
//---------------------------------------------------------------------------------------------------

#include "DSP2833x_Device.h"
#include "DSP2833x_Examples.h"
#include <math.h>
#include <stdlib.h>

//---------------------------------------------------------------------------------------------------
// Definición de variables públicas:
//---------------------------------------------------------------------------------------------------

long C  =     47;     // Capacidades del bus de continua de 470 uF
long L  =     30;     // Carga inductiva del sistema de 3 mH por fase
long R =      10;     // Carga resistiva del sistema de 10 R por fase
long Ts =      5;     // Tiempo de conmutación del sistema 100 us;

//Vectores para los resultados de las corrientes del siguiente estado
signed long Ia_Kadd1[19] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
signed long Ib_Kadd1[19] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

//---------------------------------------------------------------------------------------------------
// I_Kadd1:
//---------------------------------------------------------------------------------------------------

void I_Kadd1()
{
        ////////////////Función del Sistema//////////////////////////
        //                  Ts        | L                    |    //
        //   i(k+1) = ------------ * | ---- * i(k) + V(k+1)|    //
        //              R*Ts + L    | Ts                   |    //
        //                                                       //
        ///////////////////////////////////////////////////////////
```

```
        // Definición de variables del módulo del programa
        volatile Uint16 j = 0;                          // Definir variable del bucle
        signed long cons1 = 0, cons2 = 0, cons3 = 0; // Definición de variables para
                                                        // realizar las operaciones

        // Cálculo de las constantes
        cons1 = (L / Ts);                               // Valor de Constante 1
        cons2 = (R * Ts) + L;                           // Valor de Constante 2
        cons3 = (Ts * 100000) / cons2;                  // Valor de Constante 3

        for (j = 0; j < 19; j++)                // Bucle de 19 ciclos para los 19 vectores
        {
        Ia_Kadd1[j] = (cons3 * ((cons1 * abI[0]) + Va_Kadd1[j])) / 100000;
        Ib_Kadd1[j] = (cons3 * ((cons1 * abI[1]) + Vb_Kadd1[j])) / 100000;
        }
}
//-----------------------------------------------------------------------------------------------------
// Fin del módulo de programa
//-----------------------------------------------------------------------------------------------------
```

Anexo V: Código DSP2833x_ik.c.

```
//###########################################################################
// AUTOR: Carlos Fernández Campos
// FECHA: 12-06-2009
// ARCHIVO:  DSP2833x_ik.c
// DESCRIPCIÓN:     Módulo de programa de cálculo de la tabla de corriente de
//                  referencia.
//###########################################################################
//---------------------------------------------------------------------------------------------------
// Incluir librerías:
//---------------------------------------------------------------------------------------------------

#include "DSP2833x_Device.h"
#include "DSP2833x_Examples.h"
#include <math.h>
#include <stdlib.h>

//---------------------------------------------------------------------------------------------------
// Definición de variables públicas:
//---------------------------------------------------------------------------------------------------

//Vestores para los resultados de la tabla de corriente de referencia
signed int ia_K[200] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
signed int ib_K[200] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

//---------------------------------------------------------------------------------------------------
// i_k:
//---------------------------------------------------------------------------------------------------

void i_k()
{
        ////////Función de una onda senosoidal////////
        //      a(t) = Ao * sin (2*pi*f * t + B)        //
        ////////////////////////////////////////////////////////////

        // Tabla de la corriente de referencia para una consigna de 1A.
        signed int i_ak[200] = {411,410,409,408,406,404,402,399,397,393,389,385,
                                381,376,370,365,359,352,346,339,331,324,315,307,
                                298,290,280,271,260,251,241,230,220,209,197,186,
                                174,163,151,139,127,114,102,89,77,64,51,39,25,13,
                                0,-13,-25,-39,-51,-64,-77,-89,-102,-114,-127,-139,-151,
                                -163,-174,-186,-197,-209,-220,-230,-241,-251,-260,-271,
                                -280,-290,-298,-307,-315,-324,-331,-339,-346,-352,-359,
                                -365,-370,-376,-381,-385,-389,-393,-397,-399,-402,-404,
                                -406,-408,-409,-410,-411,-410,-409,-408,-406,-404,-402,
                                -399,-397,-393,-389,-385,-381,-376,-370,-365,-359,-352,
                                -346,-339,-331,-324,-315,-307,-298,-290,-280,-271,-260,
```

```
                                -251,-241,-230,-220,-209,-197,-186,-174,-163,-151,-139,
                                -127,-114,-102,-89,-77,-64,-51,-39,-25,-13,0,13,25,
                                39,51,64,77,89,102,114,127,139,151,163,174,186,
                                197,209,220,230,241,251,260,271,280,290,298,307,
                                315,324,331,339,346,352,359,365,370,376,381,385,
                                389,393,397,399,402,404,406,408,409,410};

        signed int i_bk[200] = {0,13,25,39,51,64,77,89,102,114,127,139,
                                151,163,174,186,197,209,220,230,241,251,260,271,280,
                                290,298,307,315,324,331,339,346,352,359,365,370,376,
                                381,385,389,393,397,399,402,404,406,408,409,410,411,
                                410,409,408,406,404,402,399,397,393,389,385,381,376,
                                370,365,359,352,346,339,331,324,315,307,298,290,280,
                                271,260,251,241,230,220,209,197,186,174,163,151,139,
                                127,114,102,89,77,64,51,39,25,13,0,-13,-25,-39,-51,-64,
                                -77,-89,-102,-114,-127,-139,-151,-163,-174,-186,-197,
                                -209,-220,-230,-241,-251,-260,-271,-280,-290,-298,-307,
                                -315,-324,-331,-339,-346,-352,-359,-365,-370,-376,-381,
                                -385,-389,-393,-397,-399,-402,-404,-406,-408,-409,-410,
                                -411,-410,-409,-408,-406,-404,-402,-399,-397,-393,-389,
                                -385,-381,-376,-370,-365,-359,-352,-346,-339,-331,-324,
                                -315,-307,-298,-290,-280,-271,-260,-251,-241,-230,-220,
                                -209,-197,-186,-174,-163,-151,-139,-127,-114,-102,-89,
                                -77,-64,-51,-39,-25,-13};

        // Definición de variables del módulo del programa
        volatile Uint16 j = 0;          // Definir variable del bucle
        int Ao = 5;                     // Valor nueva consigna de corriente (Consigna *10)

        for (j = 0; j < 200; j++ )      // Bucle de 200 ciclos para los 200 valores
        {
        ia_K[j] = (i_ak[j] * Ao) / 10;          // Valor de Alfa
        ib_K[j] = (i_bk[j] * Ao) / 10;          // Valor de Beta
        }
}
//---------------------------------------------------------------------------------------------------
// Fin del módulo de programa
//---------------------------------------------------------------------------------------------------
```

Anexo VI: Código DSP2833x_Vck+1_g[x].c.

```
//###########################################################################
// AUTOR: Carlos Fernández Campos
// FECHA: 12-06-2009
// ARCHIVO:  DSP2833x_Vk+1.c
// DESCRIPCIÓN:     Módulo de programa de cálculo de los vectores de tensión
//                          para el siguiente estado de conmutación Ts.
//###########################################################################
//---------------------------------------------------------------------------------------------------
// Incluir librerías:
//---------------------------------------------------------------------------------------------------

#include "DSP2833x_Device.h"
#include "DSP2833x_Examples.h"
#include <math.h>
#include <stdlib.h>

//---------------------------------------------------------------------------------------------------
// Definición de variables públicas:
//---------------------------------------------------------------------------------------------------

int sec_ref = {0};              // Variable puntero de seguimiento de la tabla de corriente de
                                //  referencia
int N_vector = {0};             // Número del vector a aplicar en el siguiente estado de
                                //  conmutación
int Redundante = {0};           // Número del estado redundante del vector que se debe de
                                // aplicar en el siguiente estado

//---------------------------------------------------------------------------------------------------
// VcKadd1_gx:
//---------------------------------------------------------------------------------------------------

void VcKadd1_gx()
{
        // Definición de variables del módulo del programa
        volatile Uint16 N_estados = 0, Estado = 0, j = 0, k = 0;
        signed long Ic1= 0, Ic2= 0, Ic1_1 = 0, Ic2_1 = 0;
        signed long Vc1_Kadd1 = 0, Vc2_Kadd1 = 0, Vc1_1_Kadd1 = 0,
                    Vc2_1_Kadd1 = 0, I_3f_A = 0, I_3f_B = 0, I_3f_C = 0;
        int      g_Kadd1_1 = 0, g_Kadd1 = 0;
        long cons4 = 0, cons5 = 0, cons6 = 0;
        long a = AdcMirror.ADCRESULT6, b = AdcMirror.ADCRESULT7;

        // Cálculo de las constantes
        cons4 = (Ts * 100000) / C;
        cons5 = a * 10000;
        cons6 = b * 10000;

        // Bucle de generación de la función de calidad
        for (j= 0; j < 19 ; j++)
```

```
{
        //Inicialización de las variables del bucle
        I_3f_A = 0;
        I_3f_B = 0;
        I_3f_C = 0;
        Vc1_Kadd1 = 0;
        Vc2_Kadd1 = 0;
        Vc1_1_Kadd1 = 0;
        Vc2_1_Kadd1 = 0;
        g_Kadd1 = 0;
        g_Kadd1_1 =0;

        N_estados = 1;

        //Cálculo de valores de las corrientes trifásicas A, B, y C.

        ////////////////Corrientes A B C////////////////
        //    ------------------------------------   //
        //    |Xa|    | Ma1      Mb1 |    | XR |   //
        //    |    | = | Ma2      Mb2 | * | XS |   //
        //    |Xb|    | Ma3      Mb3 |    | XT |   //
        //    ------------------------------------ //
        ////////////////////////////////////////////////

        I_3f_A = (816 * Ia_Kadd1[j]) / 1000;
        I_3f_B = ((- 408 * Ia_Kadd1[j]) + (707 * Ib_Kadd1[j])) / 1000;
        I_3f_C = ((- 408 * Ia_Kadd1[j]) + (- 707 * Ib_Kadd1[j])) / 1000;

        switch (j)
        {
        ////////////////////////////////
        //  En todos los casos:  //
        //      Ic1 = -ip        //
        //      Ic2 = in         //
        ////////////////////////////////
        case 0: //V0
                //Los tres estados redundantes => ip=0; i0=0; in=0;
                Ic1 = 0;
                Ic2 = 0;
                break;

        case 1: //V1
                // 1º estado redundante => ip=ia; i0=ic+ib; in=0;
                N_estados = 2;
                Ic1 = - I_3f_A;
                Ic2 = 0;
                // 2º estado redundante => ip=0; i0=ia; in=ib+ic;
                Ic1_1 = 0;
                Ic2_1 = I_3f_B + I_3f_C;
                break;
```

```
case 2: //V2
        // 1º estado redundante => ip=ia+ib; i0=ic; in=0;
        N_estados = 2;
        Ic1 = - (I_3f_A + I_3f_B);
        Ic2 = 0;
        // 2º estado redundante => ip=0; i0=ia+ib; in=ic;
        Ic1_1 = 0;
        Ic2_1 = I_3f_C;
        break;

case 3: //V3
        // 1º estado redundante => ip=ib; i0=ia+ic; in=0;
        N_estados = 2;
        Ic1 = - I_3f_B;
        Ic2 = 0;
        // 2º estado redundante => ip=0; i0=ib; in=ia+ic;
        Ic1_1 = 0;
        Ic2_1 = I_3f_A + I_3f_C;
        break;

case 4: //V4
        // 1º estado redundante => ip=ib+ic; i0=ia; in=0;
        N_estados = 2;
        Ic1 = - (I_3f_B + I_3f_C);
        Ic2 = 0;
        // 2º estado redundante => ip=0; i0=ib+ic; in=ia;
        Ic1_1 = 0;
        Ic2_1 = I_3f_A;
        break;

case 5: //V5
        // 1º estado redundante => ip=ic; i0=ia+ib; in=0;
        N_estados = 2;
        Ic1 = - I_3f_C;
        Ic2 = 0;
        // 2º estado redundante => ip=0; i0=ic; in=ia+ib;
        Ic1_1 = 0;
        Ic2_1 = I_3f_A + I_3f_B;
        break;

case 6: //V6
        // 1º estado redundante => ip=ia+ic; i0=ib; in=0;
        N_estados = 2;
        Ic1 = -(I_3f_A + I_3f_C);
        Ic2 = 0;
        // 2º estado redundante => ip=0; i0=ia+ic; in=ib;
        Ic1_1 = 0;
        Ic2_1 = I_3f_B;
        break;

case 7: // V7 => ip=ia; i0=0; in=ib+ic;
```

```
                Ic1 = - I_3f_A;
                Ic2 = I_3f_B + I_3f_C;
                break;

        case 8: // V8 => ip=ia; i0=ib; in=ic;
                Ic1 = - I_3f_A;
                Ic2 = I_3f_C;
                break;

        case 9: // V9 => ip=ia+ib; i0=0; in=ic;
                Ic1 = - (I_3f_A + I_3f_B);
                Ic2 = I_3f_C;
                break;

        case 10: // V10 => ip=ib; i0=ia; in=ic;
                Ic1 = - I_3f_B;
                Ic2 = I_3f_C;
                break;

        case 11: //  V11 => ip=ib; i0=0; in=ia+ic;
                Ic1 = - I_3f_B;
                Ic2 = I_3f_A + I_3f_C;
                break;

        case 12: // V12 => ip=ib; i0=ic; in=ia;
                Ic1 = - I_3f_B;
                Ic2 = I_3f_A;
                break;

        case 13: // V13 => ip=ib+ic; i0=0; in=ia;
                Ic1 = - (I_3f_B + I_3f_C);
                Ic2 = I_3f_A;
                break;

        case 14: // V14 => ip=ic; i0=ib; in=ia;
                Ic1 = - I_3f_C;
                Ic2 = I_3f_A;
                break;

        case 15: // V15 => ip=ic; i0=0; in=ia+ib;
                Ic1 = - I_3f_C;
                Ic2 = I_3f_A + I_3f_B;
                break;

        case 16: // V16 => ip=ic; i0=ia; in=ib;
                Ic1 = - I_3f_C;
                Ic2 = I_3f_B;
                break;

        case 17: // V17 => ip=ia+ic; i0=0; in=ib;
                Ic1 = - (I_3f_A + I_3f_C);
```

```
            Ic2 = I_3f_B;
            break;

case 18: // V18 => ip=ia; i0=ic; in=ib;
            Ic1 = - I_3f_A;
            Ic2 = I_3f_B;
            break;
}

//////////////Cálculo carga condensadores//////////////
//     Vc(k+1) = Vc(k) + (1 / C) * ic(k) * Ts      //
////////////////////////////////////////////////////////////////
Vc1_Kadd1 = cons5 + (cons4 * Ic1);
Vc2_Kadd1 = cons6 + (cons4 * Ic2);

//////////////////////Función de calidad/////////////////////////////////////////
// g = abs|i*a(k)-ia(k+1)| + abs|ib*(k)-ib(k+1)| + abs|Vc1-Vc2| //
////////////////////////////////////////////////////////////////////////////////
g_Kadd1 = abs(ia_K[sec_ref] - Ia_Kadd1[j]) + abs(ib_K[sec_ref] –
            Ib_Kadd1[j]) + abs((Vc1_Kadd1 - Vc2_Kadd1) / 10000);
Estado = 1;

//Se el vector contiene estado redundante (N_estados == 2)
if (N_estados == 2)
{
            //////////////Cálculo carga condensadores//////////////
            //     Vc(k+1) = Vc(k) + (1 / C) * ic(k) * Ts   //
            ////////////////////////////////////////////////////////////////
            Vc1_1_Kadd1 = cons5 + (cons4 * Ic1_1);
            Vc2_1_Kadd1 = cons6 + (cons4 * Ic2_1);

            //////////////////////////////Función de calidad/////////////////////////////////
            //g = abs|i*a(k)-ia(k+1)| + abs|ib*(k)-ib(k+1)| + abs|Vc1-Vc2|//
            ////////////////////////////////////////////////////////////////////////////////
            g_Kadd1_1 = abs(ia_K[sec_ref] - Ia_Kadd1[j]) + abs(ib_K[sec_ref]
                  - Ib_Kadd1[j]) + abs((Vc1_1_Kadd1 - Vc2_1_Kadd1) / 10000);

            //Se guarda el estado redundante más pequeño
            if (g_Kadd1 > g_Kadd1_1)
            {
            g_Kadd1 = g_Kadd1_1;
            Estado = 2;
            }
}
// Allar el vector a aplicar en el siguiente estado Ts
if (j == 0)
            {
            k = g_Kadd1;
            N_vector = j;
            Redundante = Estado;
            }
```

```
                if (g_Kadd1 < k)
                        {
                        k = g_Kadd1;
                        N_vector = j;
                        Redundante = Estado;
                        }
        }
        // Seguimiento de la tabla de la corriente de referencia
        if (sec_ref >= 200)
                sec_ref = 0;        // Si sec_ref contiene el valor 200 se pone a 0
                else
                sec_ref++;          // Si no incrementamos sec_ref
}
//-------------------------------------------------------------------------------------------------------
// Fin del módulo de programa
//-------------------------------------------------------------------------------------------------------
```