UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

DISEÑO DE UN RECTIFICADOR VIENNA APLICADO AL CONTROL DE UN AEROGENERADOR DE 2,2KW

16/11/2018

Autor: **Fornes Castro, Jose Vicente**

Directores: **Gimeno Sales, Francisco José**

Orts Grau, Salvador

AGRADECIMIENTOS

Quisiera agradecer en primer lugar a mis dos directores de trabajo de fin de grado Voro y Paco por el apoyo recibido y por introducirme en un proyecto enfocado en la electrónica de potencia y el estudio de las energías renovables, el cual es el ámbito profesional donde quiero crecer y continuar con mis estudios futuros.

Agradecer como no a mis padres Paquita y Armando los cuales sin su apoyo incondicional no habría podido llegar a realizar el grado en ingeniería electrónica y automática industrial.

También agradecer a mi abuela Alicia por todos los sermones y por preguntarme cada mes cuando iba a terminar la carrera que estaba haciendo, y agradecer a su hermana, mi tía Fina por el apoyo recibido durante todos mis estudios.

RESUMEN

Hoy en día el consumo de energía, está abriendo puertas a nuevas metodologías para adquirir otro tipo de energía, diferente a la convencional, energía limpia, energía que nos permita avanzar con el ritmo actual sin dañar al medioambiente, ahí es donde aparecen las energías renovables, y donde la energía eólica se abre camino.

Las dificultades que la red eléctrica encuentra con este tipo de energía, ya que es una energía que depende de factores como el viento o la dirección que este tome, son la estabilidad de potencia que inyecta a la red, para ello y con el propósito de solucionar este problema la electrónica de potencia juega un papel fundamental en la conversión y control de la energía que suministran los campos eólicos.

Aquí nace un punto de partida para mejorar los sistemas actuales y obtener una energía más limpia y con mayor eficiencia que la que teníamos en años anteriores, con ello presente trabajo de fin de grado tiene como objetivo indagar en esta área y aportar su estudio en un campo en desarrollo.

ABSTRACT

Nowadays we are consuming all kind of energy for our daily life and hence we need to open new points of view to get other type of energy, more clean than the older and energy that let us to improve in our society without injuring our environment, there is where renewable energy appears and where wind power systems become more important than ever.

The main issues appears in the electrical grid with the connection of this energy from wind parks, because the grid needs a stable power but the energy that we get from a wind turbine depends on the wind and its direction, there is where power electronics can help to behave stable the power from a wind turbine and is able to manage and convert it to the right energy for the grid.

There is where born a starting point to improve the actual systems and get a cleaner and efficient energy that the older, hence the present final project have an objective in mind, get deep in that matter and can help with the present research about this new field of researching.

INDICE

LISTA DE ILUSTRACIONES

1. INTRODUCCIÓN

En los últimos años la demanda energética continúa en auge, lo cual está provocando que se adquieran nuevas metodologías de adquisición de energía con un bajo índice de contaminación. Aquí es donde hacen su entrada la generación de energías renovables como una de las alternativas para dar solución a los problemas de consumo de hoy en día. Una de las energías renovables más empleada y eficaz es la eólica, la cual convierte energía mecánica proveniente de las corrientes del viento en energía eléctrica, la cual se puede transportar por la red eléctrica.

Para su fácil transporte y posterior consumo, la energía que proporcionan los aerogeneradores en los parques eólicos debe ser tratada mediante sistemas electrónicos de potencia, los cuales convierten una energía inicial mecánica en una energía eléctrica adecuada a las características de la red.

Se muestra un diagrama de bloques para la aclaración de las etapas generales en un sistema aerogenerador en el cual se pueden ver diferenciadas las etapas principales para el tratamiento y la transformación de energía producida por un aerogenerador.

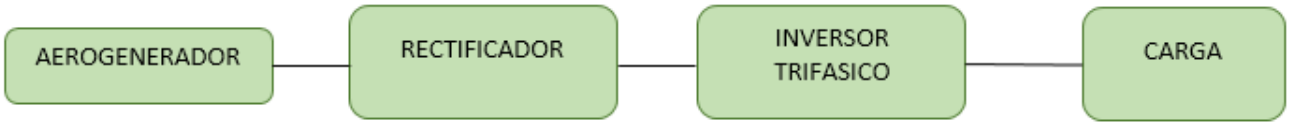


Ilustración 1.Diagrama de bloques genérico de un sistema aerogenerador

En el presente proyecto se va a profundizar en la primera etapa de conversión de energía, la cual hace referencia a la etapa del rectificador. Pará realizar esta etapa existen multitud de configuraciones y se pueden elegir dependiendo del uso de dicha aplicación, dependiendo del tipo de control aplicado al generador y de las especificaciones de potencia de la carga así como de los niveles de tensión que se requieran a la salida.

En nuestro caso, en esta primera etapa vamos a incorporar un rectificador activo trifásico elevador con corrección de factor de potencia, la topología empleada para el diseño y simulación es la topología Vienna tipo T, la cual nos ofrece una alta eficiencia, una tensión en los interruptores de potencia reducida, ya que trabajamos en tres niveles con dos interruptores por cada nivel y la posibilidad de elevar el voltaje de salida dependiendo de los requerimientos del sistema.

La placa de desarrollo de la cual se va a realizar la simulación del sistema y su diseño es la placa de Texas Instruments (TIDM-1000).

El entorno de desarrollo está basado en el rectificador de Vienna tipo T, el cual realiza el control mediante la familia de microcontroladores C2000 de 32-bits y doble núcleo TMS320F28377D de Texas Instruments, el entorno de desarrollo también contiene todos los sensores y componentes necesarios para la adquisición de datos de la placa y su tratamiento.

2. OBJETIVOS

El objetivo del presente trabajo de fin de grado, consiste en la realización de la simulación de un rectificador de Vienna, basado en la plataforma de desarrollo TIDM-1000 de Texas Instruments la cual incorpora un rectificador trifásico elevador con la topología Vienna tipo T. La plataforma TIDM-1000, tiene como objetivo ser implementada en la etapa de electrónica de potencia y rectificación (AC/DC) de un sistema de conversión de energía eólica la cual está compuesto por una maquina eléctrica sincrónica de imanes permanentes de 3 kW-120V.

Mediante el siguiente diagrama se pueden observar las etapas de las cuales está compuesto un sistema eólico completo como el que se ha comentado.

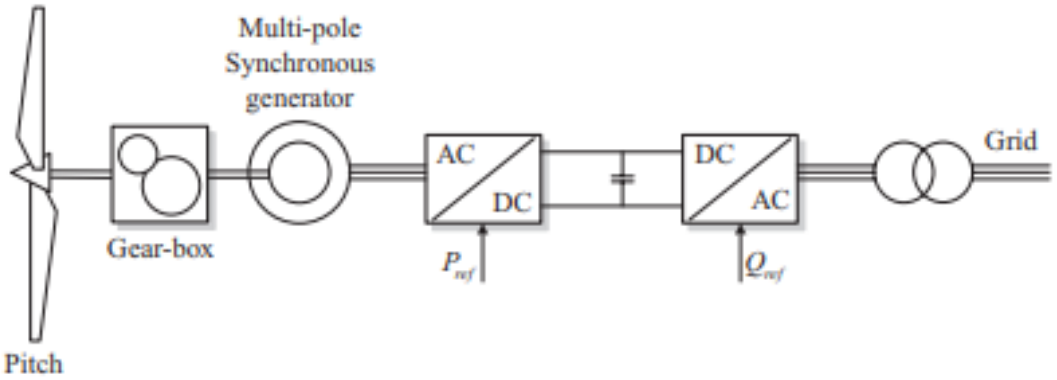


Ilustración 2.Diagrama de bloques para un sistema de conversión de energía eólico [9]

El propósito de la implementación de la plataforma TIDM-1000, consiste en mejorar las características de conversión de energía, ya que implementando este modelo podemos conseguir factores de potencia cercanos a la unidad además de una distorsión armónica total reducida y una señal de continua a la salida del rectificador que conecta con la siguiente etapa de potencia, él conversor DC/AC con mínimo rizado a la salida del sistema.

El desarrollo de la simulación citada, estará basada en la topología Vienna implementada en la placa de desarrollo TIDM-1000, con el objetivo de verificar el sistema sin necesidad de realizar pruebas experimental y/o para prevenir respuestas inesperadas mediante la realización de las mismas.

En la realización del mismo se pretende realizar un estudio del rectificador Vienna, además de las metodologías empleadas en otros estudios referentes a la topología del rectificador Vienna, así como las aplicaciones en energías renovables en las que se ha implementado dicha topología.

3. REQUERIMIENTOS

3.1 Definición y alcance

El objeto del presente proyecto es realizar una simulación para un sistema de conversión de energía mediante un convertidor electrónico, el cual se basa en un rectificador trifásico activo tipo boost con corrección de factor de potencia, el modelo que se elige para el rectificador es la topología Vienna tipo T ,con aplicación a una maquina sincrónica de imanes permanentes de 3 kW-120 V la cual se implementa como aerogenerador.

La realización de citado proyecto conlleva también realizar un estudio de las metodologías del rectificador Vienna e implementaciones para sistemas de energías renovables ya realizados con aplicaciones orientadas a la conversión de energía de sistemas eólicos.

Dicho sistema de conversión de energía electrónico, se debe documentar, tanto para el diseño hardware, cómo para toda la documentación que se refiera al software con el cual se realiza la programación del sistema en el convertidor electrónico de potencia TIDM-1000.

La simulación del rectificador Vienna se implementa mediante el software Matlab-Simulink en la cual se incluye el modelo físico del rectificador Vienna TIDM-1000, así como el control necesario para el funcionamiento de dicho rectificador, con el que poder verificar su funcionamiento, tanto para los parámetros generales de funcionamientos que se emplean en las redes eléctricas actuales, tanto la europea como la americana, cómo su funcionamiento con los parámetros extraídos de ensayo de la máquina sincrónica de imanes permanentes Bornay 3kW-120 V [14]realizada en estudios anteriores con conexión a dicha máquina.

3.2 Requerimientos generales

Las condiciones que se incluyen en este apartado, son las que con carácter general deberá cumplir tanto la simulación del convertidor electrónico como la documentación que se aporta en el presente documento.

3.2.1 Simulación

La simulación deberá de incorporar un modelo físico de la plataforma de desarrollo TIDM-1000 de Texas Instruments el cual será una aproximación del modelo real con el objetivo de obtener una simulación del rectificador Vienna con el que ser capaces de verificar su funcionamiento ante múltiples situaciones en las cuales el sistema real TIDM-1000 puede verse involucrado.

Mediante la simulación se debe verificar que se cumplen las características generales de funcionamiento del TIDM-1000 para las alimentaciones de red de 208 V_{rms} / 60 Hz y 400V_{rms} / 50 Hz, realizando un análisis en carga. También será ámbito de aplicación de la simulación realizar simulaciones mediante los datos que se proporcionan en estudios anteriores de la maquina sincrónica de imanes permanentes Bornay 3 kW-120 V.

Las características que se pretenden cumplir , incluirán resultados tanto a nivel teórico en la simulación como a nivel real, ya que existen parámetros los cuales pueden destruir el dispositivo o no permitir a la placa TIDM-1000 funcionar correctamente con el diseño realizado, por ello se enumeran las características que se deben cumplir en el sistema .

1. Intensidad de entrada al rectificador menor de 4 A de valor eficaz.
2. Intensidad de salida del rectificador menor de 5 A de valor eficaz.
3. Factor de potencia unitario.
4. Eficiencia del convertidor cercana al 98%.
5. Bajos niveles de distorsión armónica total en la intensidad de entrada del sistema THDi con niveles inferiores al 5 % a plena carga.
6. Compensación en la tensión de bus de los condensadores a la salida del sistema.
7. Tensiones reducidas en los elementos de potencia, tanto en los diodos de rectificación con en los transistores Mosfet de potencia en la conmutación de la mitad de la tensión aplicada a la salida.
8. Alcanzar la referencia de tensión en el bus de continua con el rizado que se obtiene en el diseño del sistema.
9. Conseguir elevar la tensión en el bus de continua por encima de la tensión que obtendríamos en un rectificador no elevador.
10. En los resultados obtenidos se deberá mostrar información de la variación de THDi, el factor de potencia y la eficiencia que está obteniendo el rectificador de Vienna, realizando un análisis de cargas a la salida del sistema.

3.2.2 Estudios anteriores

Durante la ejecución del estado del arte, se deben comentar estudios referentes a la construcción y diseño de la topología rectificador de Vienna, así como las diversas técnicas que están siendo empleadas para el control de dicho rectificador, también se estudiaran estudios que presenten interés con el fin de aportar información complementaria en la comprensión del presente proyecto.

Finalmente se observara las aplicaciones de esta topología en el campo de las energías renovables el cual tiene el ámbito el presente proyecto.

3.2.3 Documentación TIDM-1000

La documentación que se debe incluir en el presente proyecto, incluirá el diseño del hardware implementado en la placa de desarrollo TIDM-1000, así como un desglose del hardware utilizado para su funcionamiento.

La documentación referente al software deberá incluir con que programas se implementa el control y la adquisición de datos en el microcontrolador el TSM320F28377D, el cual realiza el control en la placa TIDM-1000 para su correcto funcionamiento.

3.2.4 Documentación complementaria

En el primer anexo se incluirá los resultados que se hayan obtenido de los análisis en carga para las simulaciones citadas, con el fin de aportar la información de funcionamiento del rectificador de Vienna.

En el segundo anexo se adjuntara un ejemplo del código generado realizando la configuración software para el rectificador de Vienna implementado en la placa TIDM-1000.

Por ultimo en el tercer anexo, se deben adjuntar los esquemáticos que contiene el modelo real del rectificador de Vienna en la placa TIDM-1000.

4. ESTADO DEL ARTE

A través de este apartado se va realizar un estudio en que se va a mostrar el rectificador de Vienna y las topologías que puede adoptar, así como las características de funcionamiento que lo están convirtiendo en un rectificador popular para la conversión de energía, para potencias elevadas y con buenas prestaciones.

También vamos a hacer un repaso de las metodologías de control que investigadores de todo el mundo están realizando respecto a esta materia, así como veremos el sistema aerogenerador y como el rectificador de Vienna se está incorporando en la etapa AC/DC de dicho sistema.

4.1 RECTIFICADOR DE VIENNA

Lo primero que vamos a hacer respecto al rectificador de Vienna es situarlo dentro de la clasificación existente de todo tipo de rectificadores trifásicos, con la cual tenemos un punto de partida para comprensión.

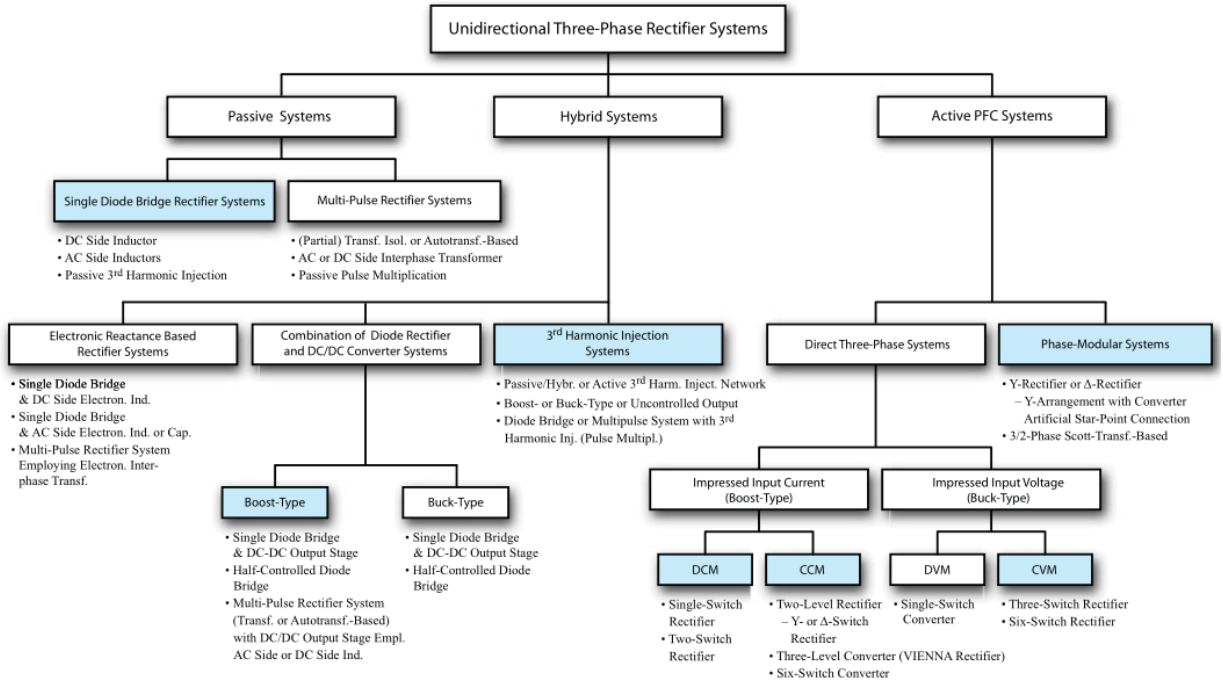


Ilustración 3. Clasificación general rectificadores trifásicos [1]

Mediante esta clasificación podemos ver como el rectificador de Vienna dentro de los sistemas activos de corrección del factor de potencia “PFC”, en los que se diverge en la rama de sistemas directos trifásicos del tipo elevador “boost” con modo de corriente continua “CCM”, siendo un convertidor electrónico de tres niveles.

La configuración del rectificador de Vienna nace a través de la modificación de otro tipo de rectificador trifásico con el mismo nivel de clasificación el rectificador con interruptores en triangulo “Δ-Switch Rectifier”.

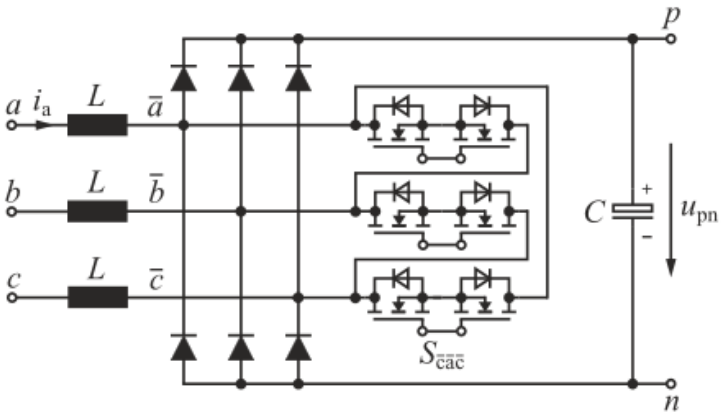


Ilustración 4. Rectificador con conexión en triangulo [1]

Con esta topología en mente si ahora realizamos una conversión de los interruptores con conexión en estrella y si además el punto de esta conexión en estrella se convierte en un punto medio a la salida, partiendo así el condensador del bus de la salida de continua en dos, obteniendo un punto neutro entre las dos ramas, obtenemos lo que se conoce como rectificador de Vienna en su configuración original [1].

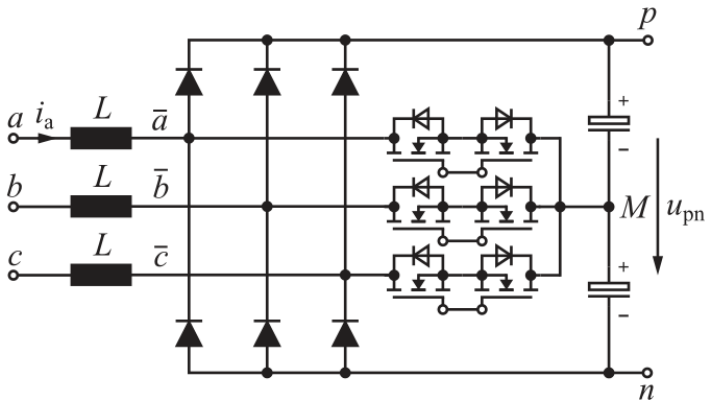


Ilustración 5. Rectificador de Vienna configuración original tipo T [1]

Además de la configuración original existen otro tipo de conexiones posibles en la configuración de los niveles del rectificador de Vienna, en la siguiente ilustración se muestran tres configuraciones que se pueden implementar en esta topología de rectificador.

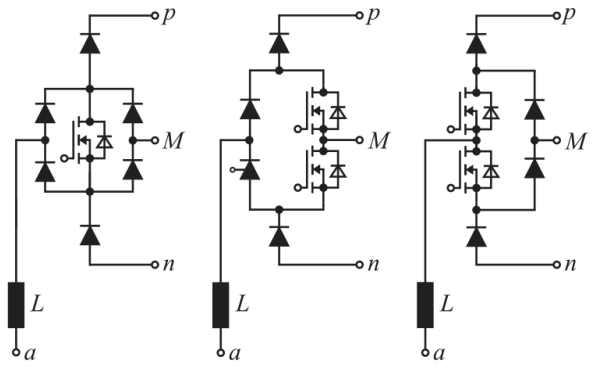


Ilustración 6. Configuraciones alternativas para el rectificador de Vienna [1]

Las principales características de la que dispone esta topología es la considerable disminución de la tensión de bloqueo que tienen que soportar los transistores de potencia, ya que se reduce a la mitad del bus de continua alargando así su vida útil.

Por otro lado con el control adecuado se pueden conseguir factores de potencia unitarios"UPF", empleando la corrección de del factor de potencia "PFC", a su vez esta configuración alcanza altos niveles de eficiencia.

En lo referente a los niveles de distorsión armónica total en las corrientes de entrada THDi, a bajas potencias obtenemos un valor más elevado que en altas potencias donde mediante las pertinentes simulaciones y estudios se puede verificar como tienen un orden bajo de THDi, dependiendo de las características del diseño podemos estar por debajo del 5% y el 3%, también para ciertas aplicaciones que necesitan niveles muy bajos de THDi, se aplican frecuencias de conmutación elevadas con las que conseguir este resultado.

4.2 METODOLOGIAS DE CONTROL PARA EL RECTIFICADOR VIENNA

En los últimos años los rectificadores activos de tres niveles que implementan la topología de Vienna, están alcanzando factores de potencia muy cercanos a la unidad además de conseguir una alta eficiencia con entradas de corriente prácticamente sinusoidales, en recientes investigaciones [2] , se llegan a conseguir potencias elevadas de hasta 10 kW , mediante controles digitales de corriente y frecuencias de muestreo de hasta 25 MHz, alcanzadas gracias a los últimos modelos de FPGA de alta velocidad.

Se están consiguiendo además niveles de distorsión armónica total "THD" con resultados por debajo del 3 % y llegando a niveles de THD de hasta el 1,4 % para frecuencias de red de 50 Hz.

Las partes de las que se compone el citado estudio, se pueden observar en la siguiente ilustración.

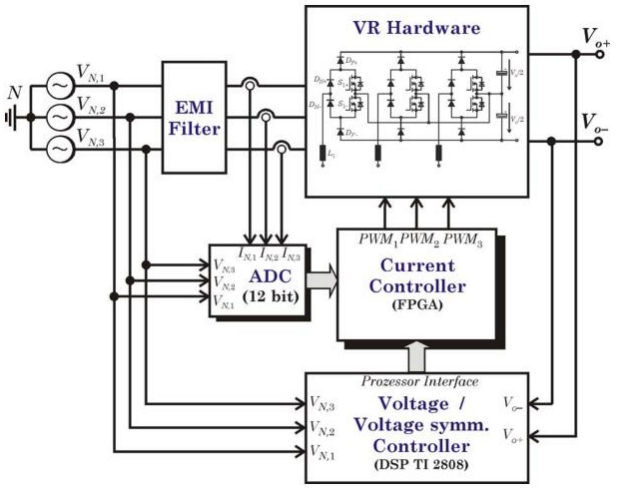


Ilustración 7. Estructura rectificador de Vienna con control digital mediante FPGA de alta velocidad [2]

Otro de los grandes hitos que se están alcanzando, gracias a la implementación de los rectificadores de tipo Vienna es el bajo voltaje de bloqueo que se genera en los componentes de conmutación, lo cual garantiza una mayor fiabilidad y duración [3], siendo así ampliamente utilizada para el suministro en alta tensión de corriente continua "HVDC". También se está adoptando por el control con modulación de vectores espaciales "SVM" mediante el modo de control d-q.

Además en el estudio citado se implementa una nueva estrategia de control denominada compensación de la relación de trabajo mediante realimentación "QRP", que consigue reducir la distorsión armónica de corriente de red al introducir un compensador para armónicos en paralelo, dicha técnica se propone para llegar a eliminar la distorsión armónica total a un valor cercano a 0, THDi ≈ 0, dicha estrategia de control se verifica para una potencia de 5,3 kW en una placa prototipo.

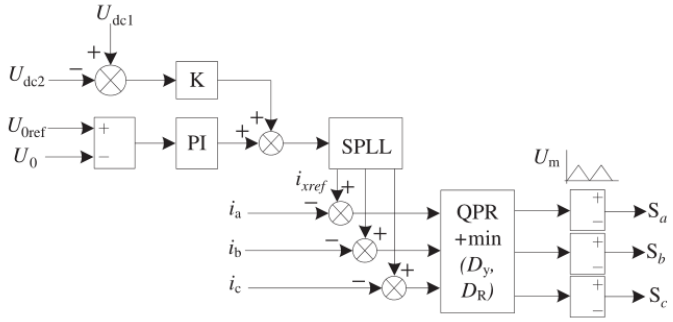


Ilustración 8. Estructura del diagrama de control [3]

La topología de Vienna se hace interesante para la mejora del factor de potencia el cual podríamos denominar, como el proceso de aumento del factor de potencia sin variar las cargas conectadas o variando el voltaje o la corriente que circula por la carga.

La topología VIENNA mantiene su interés debido a que la tensión que tienen que soportar los elementos de conmutación son la mitad del voltaje final en la carga lo cual permite la utilización de transistores tipo MOSFET para una velocidad de conmutación mayor.

Mediante un estudio [4] realizado para conseguir una alta eficiencia, se obtiene mediante la implementación de transistores tipo Mosfet, una alta frecuencia para la señal triangular de comparación de 10 kHz y el método "SPWM", modulación sinusoidal de ancho de pulso, se consigue un factor de potencia mayor a 0,94 y con un THDi por debajo del 0,1 % lo cual es inferior de lo que se pide en el estándar de la norma IEEE 519.

El rectificador de Vienna está diseñado para operar en el modo de conducción continua(CCM) a plena potencia, pero por el contrario se estudia [5], cuando la carga es reducida que se puede dar el caso de que nos adentremos en el modo de conducción discontinua(DCM), además se estudia , como incluso cuando nuestro rectificador está a máxima potencia se puede dar este caso si las inductancias de entrada son de reducido valor, por lo que se implementa un método digitalizado de compensación por realimentación para eliminar el problema del modo de conducción mixta(MCM).

El estudio al cual se hace referencia se realiza con inductancias de entrada de 120 µH y los análisis son efectuados para un prototipo de 5 kW, el cual se verifica con una señal de entrada de 220 V y salida 720 V, empleando por cada rama dos transistores del tipo CoolMOS a una frecuencia de conmutación de 65 kHz.

Los resultados de funcionamiento sin el método propuesto, para una potencia de 5002 W son de un factor de potencia "PF" 0,971 % y una distorsión armónica total de entrada de 16,71 %, aplicando la digitalización de compensación por realimentación los resultados son más favorables alcanzando así, para la misma potencia un factor de potencia del 0,995 % y una distorsión armónica total en la entrada del 4,8%.

El control empleado para cada fase se puede ver en la siguiente ilustración.

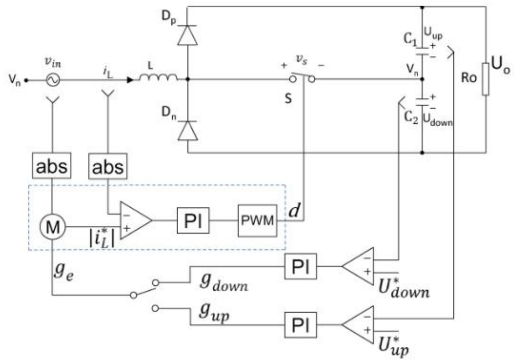


Ilustración 9. Control de fase para el rectificador de Vienna [5]

Nuevas técnicas de control también están surgiendo, obteniendo resultados similares con metodologías de control más sencillas que las basadas en rectificadores Vienna por control de vectores espaciales, en la realización de un estudio de estas nuevas técnicas [6], se propone un método basado en la modulación por ancho de pulso de forma discontinua "CB-DPWM", el cual consigue un factor de potencia unitario y garantiza trabajar con todos los valores de modulación M_a.

Otros estudios proponen métodos para poder mitigar la distorsión de corriente en el rectificador Vienna con voltajes de corriente continua balanceados y no balanceados [7], el método de secuencia con compensación por eliminación de corriente de armónicos, simplifica la identificación en intervalos anormales de vectores espaciales simétricos y asimétricos, además este método no ejerce ningún efecto sobre el equilibrio del voltaje en el punto neutro.

El control implementado en el citado estudio es el siguiente.

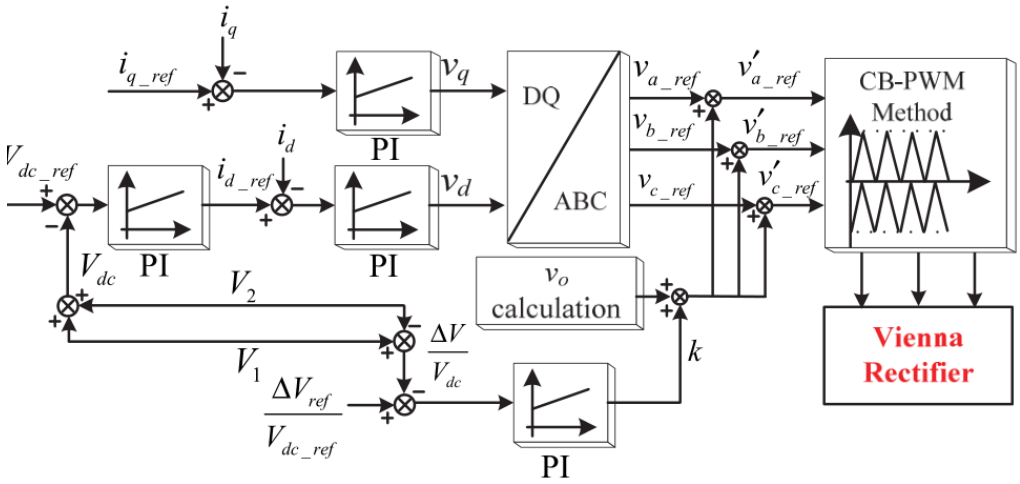


Ilustración 10. Diagrama de control con secuencia cero para el rectificador de Vienna [7]

Uno de los problemas que pueden surgir, es cuando conectamos cargas no lineales en paralelo con el convertidor , un reciente estudio nos habla de la calidad de la potencia [8] y el método que se utiliza para compensar la potencia reactiva y eliminar los armónicos, función de calidad de potencia inyectada (PQAF).

Realizados los experimentos oportunos utilizando dicho método se verifica que la potencia activa no lineal puede ser de hasta 1,228 veces si el rectificador Vienna tiene compensados los niveles de armónicos, en lo referente a la potencia reactiva puede alcanzar hasta un máximo del 10,6% de la potencia activa, que se obtiene cuando la potencia activa del rectificador es la mitad que en la carga no lineal.

En las siguientes ilustraciones podemos observar el sistema que se está estudiando y el tipo de control que se está empleando para su funcionamiento.

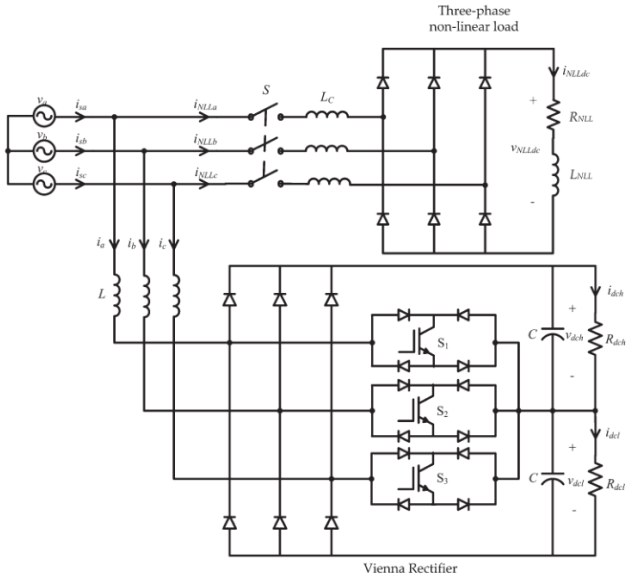


Ilustración 11. Rectificador de Vienna con carga no lineal en red [8]

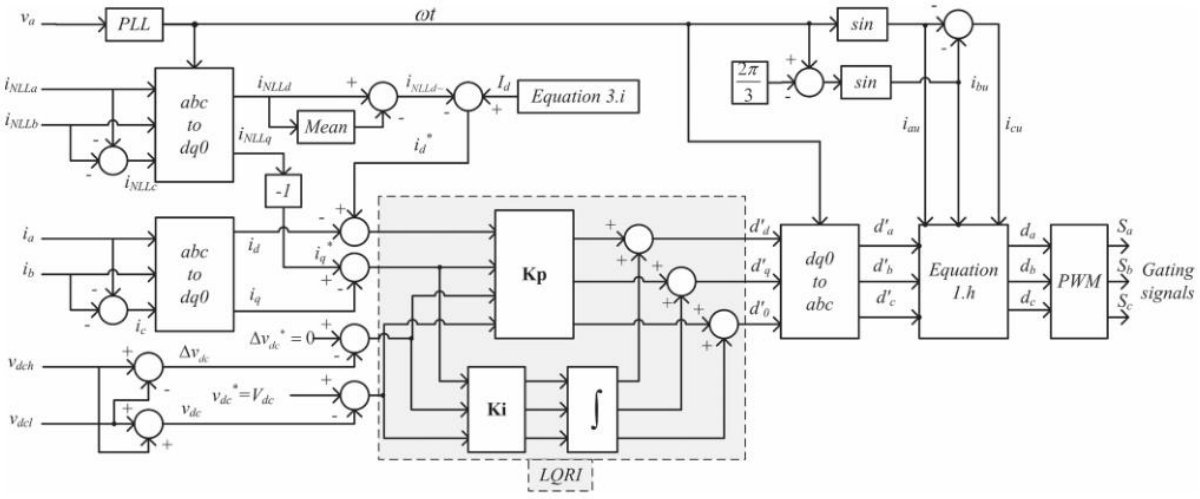


Ilustración 12.Diagrama de bloques para el control del rectificador de Vienna con carga no lineal en red [8]

4.3 ESTRUCTURA DEL SISTEMA AEROGENERADOR EN CONEXIÓN A RED

Las conexiones que se producen de los sistemas aerogenerador en la red, son un tema delicado a tratar, ya que los parques eólicos pueden crear problemas en la red eléctrica, ya que está diseñada para una potencia constante, funcionando de manera contraria los parques eólicos, ya que dependen de las velocidades y direcciones que tome el viento.

Visto el problema que plantean las instalaciones de parques eólicos, se dispone a encontrar una solución para poder convertir un sistema que no aporta una potencia estable, en uno que si cumpla esta característica fundamental.

Aquí es donde entran los las configuraciones de potencia para sistemas aerogeneradores, las configuraciones básicas, cuentan con dos partes fundamentales, una parte mecánica y otra eléctrica.

La primera parte obtiene su energía del viento convirtiendo energía cinética en energía de rotación gracias a las palas con las que cuenta un sistema aerogenerador, la segunda parte es la involucrada en convertir esta energía mecánica que permite girar a cierta velocidad un generador, en energía eléctrica regulada para su conexión a red, en esta parte es donde se integran los convertidores electrónicos de potencia, capaces de realizar esta función de conversión de energía.

Se muestra un esquema básico de un sistema de control para un sistema aerogenerador, el cual cuenta con las etapas necesarias para dicho tratamiento y conversión de la energía proveniente del viento.

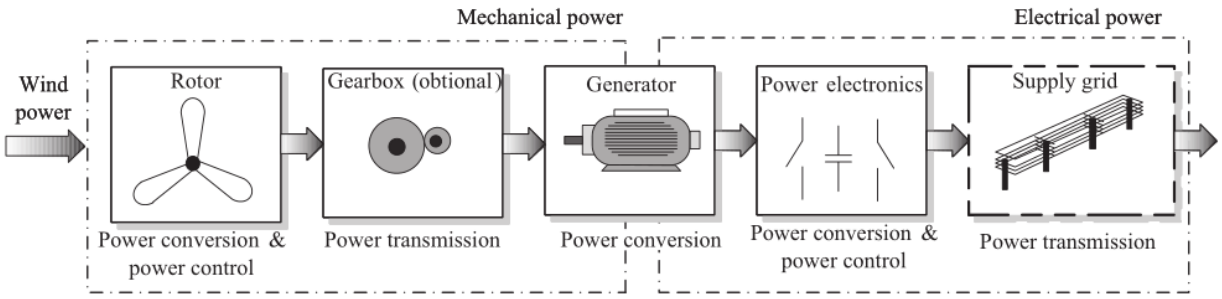


Ilustración 13. Diagrama de un sistema aerogenerador conectado a red [9]

Nuestra aportación a este sistema de gran complejidad, está dirigida a la etapa de electrónica de potencia en la cual contamos con dos tipos de convertidores electrónicos, el primero un rectificador, convertidor AC/DC y después un inversor, convertidor DC/AC, en el presente trabajo de fin de grado, hacemos hincapié en el estudio de esta primera etapa de rectificación.

4.4 APLICACIONES EN ENERGIAS RENOVABLES DEL RECTIFICADOR VIENNA

En este aparado vamos a incidir en las aplicaciones directamente aplicadas mediante el rectificador de Vienna en sistemas aerogenerador.

En recientes estudios se observa en base a un generador síncrono de imanes permanentes "PMSG" [10], un sistema de conversión de la energía eólica con velocidad variable, basado en el rectificador de Vienna.

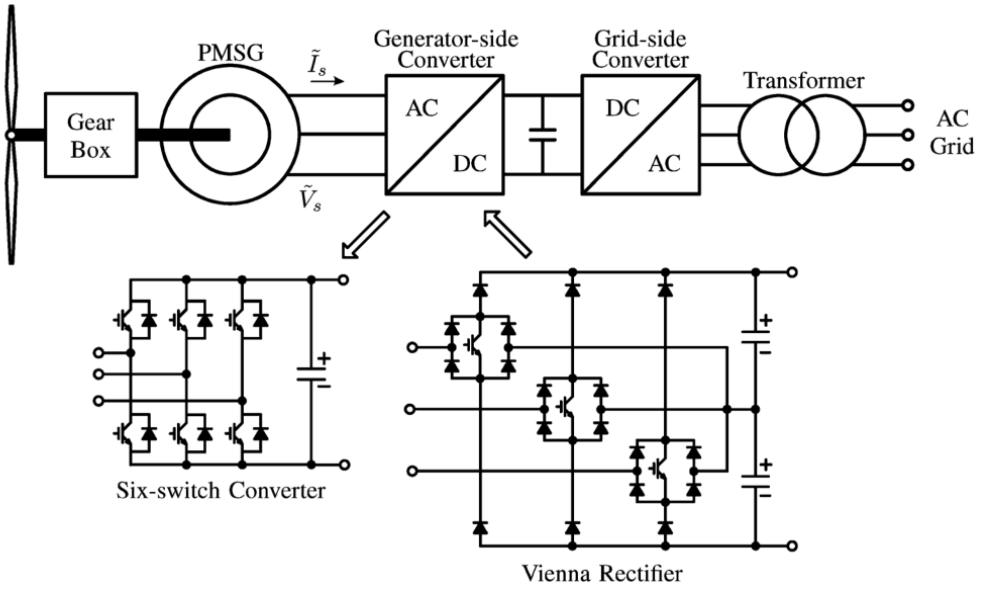


Ilustración 14.Topologia para sistemas de conversión eólica PMSG Rectificador Vienna [10]

Para poder identificar la región de operación disponible, se diseña un control modo corriente para obtener el par que se desea del generador, ajustando los límites de corriente, voltaje y factor de potencia.

Otro tipo de generador para el cual se está aplicando la topología de rectificador de Vienna es en máquinas eléctricas de inducción mediante jaula de ardilla con variación de velocidad [11], en el cual se comprueba mediante simulaciones del sistema como se obtiene mayor eficiencia y mayor fiabilidad respecto a un convertidor de seis niveles y dos niveles debido a reducida tensión de bloqueo que soportan los semiconductores.

En este caso concreto el sistema incorpora un filtro LC en el sistema de conversión de energía

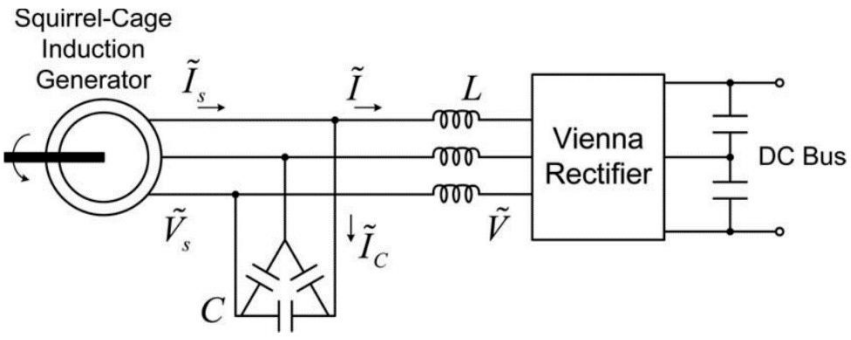


Ilustración 15.Configuración para generador de jaula de ardilla con rectificador Vienna [11]

Están saliendo a la luz nuevas metodologías de control para generadores de imanes permanentes síncronos "PMSG", mediante el rectificador de Vienna, como es el control sobre el par directo del generador "DTC" [12], esta metodología de control cobra especial importancia de estudio para aplicaciones de baja potencia debido a la rápida respuesta del par, a la eliminación de un sensor en el rotor para el control de posición y disminución de los cálculos.

Este nuevo método se prueba experimentalmente para un prototipo de 1kW, los cuales confirman las simulaciones y resultados teóricos anteriores.

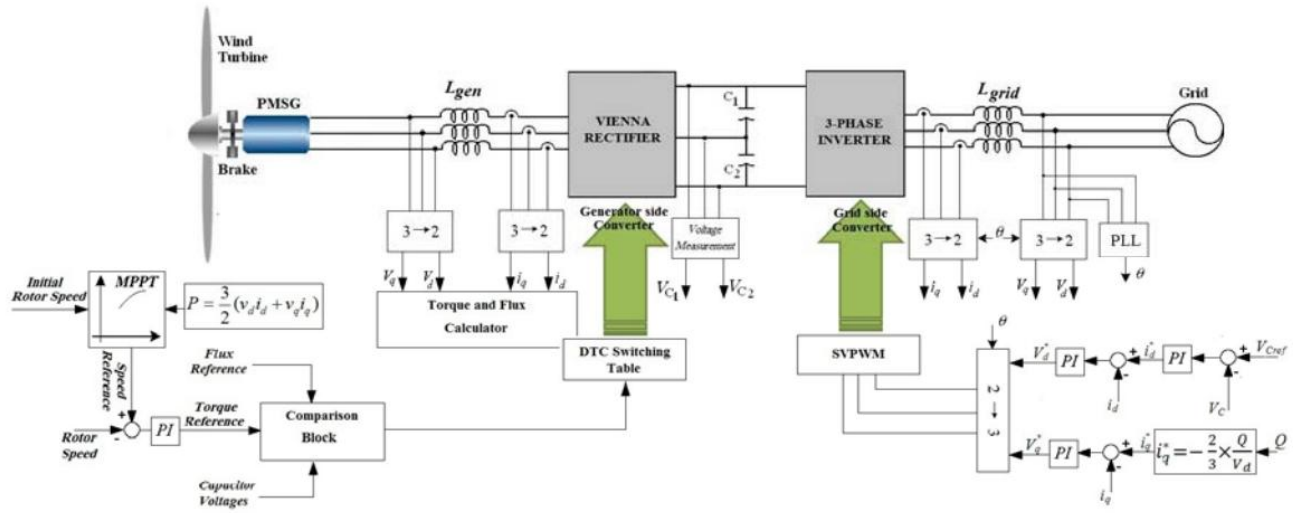


Ilustración 16. Diagrama de bloques de generador de imanes permanentes con control de par directo para rectificador de Vienna [12]

4.5 RECTIFICADOR DE VIENNA IMPLEMENTADO EN “TIDM-1000”

La topología del rectificador Vienna se encuentra dentro del marco de rectificadores activos trifásicos unidireccionales, con corrección activa del factor de potencia, del tipo elevador (Boost-Type), además de trabajar en el modo de conducción continua y de tres niveles.

La topología para sistemas de potencia del rectificador Vienna, [13] es usada para altas potencias, las cuales requieren de un sistema trifásico de corrección de factor de potencia, las aplicaciones van desde cargadores para vehículos eléctricos, usos industriales, hasta aplicación en energías renovables como es nuestro caso particular para el control de un aerogenerador de 3 kW.

El control de dicha placa se realiza mediante la familia de microcontrooladores C2000 de Texas Instruments con el modelo TMS320F28377D.

Las características de la placa TIDM-1000 son las siguientes:

- Sistema trifásico de entrada 208 VL-L a 60 Hz, con una salida de 600 V en corriente continua, con una potencia de 1,2kW
- Sistema trifásico de entrada 400 VL-L a 50 Hz, con una salida de 700 V en corriente continua, con una potencia de 2,4kW.
- Frecuencia de conmutación, modulación de ancho de pulso (PWM), de 50 kHz.
- La eficiencia máxima del rectificador Vienna es mayor del 98%.
- Distorsión armónica total (THD), menor del 2 % tanto para carga máxima como para carga mínima.

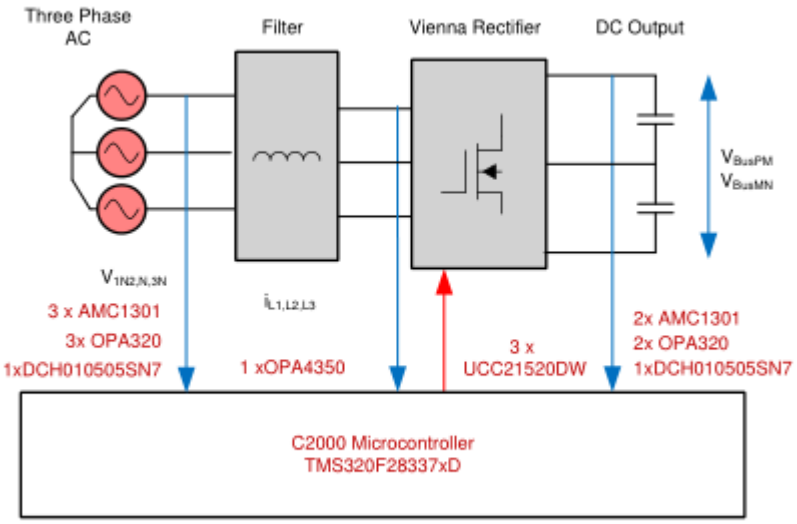


Ilustración 17.Diagrama de bloques de la placa TIDM-1000 de TI

5. DISEÑO DEL RECTIFICADOR VIENNA TIDM-1000

5.1 DISEÑO INDUCTOR ENTRADA

El correcto diseño del inductor a la entrada del rectificador es capaz de filtrar los armónicos de conmutación que se generan, para realizar el diseño del inductor (L_i), se va a partir de la corriente de rizado que se considere tolerable ,en nuestro diseño será del 10% para la corriente de rizado a entrada. También se debe tener en cuenta que el núcleo que contiene al inductor sea capaz de tolerar la corriente que por el inductor va a circular así como su rizado.

En la siguiente figura se observa el rizado la forma que puede optar el rizado de corriente.

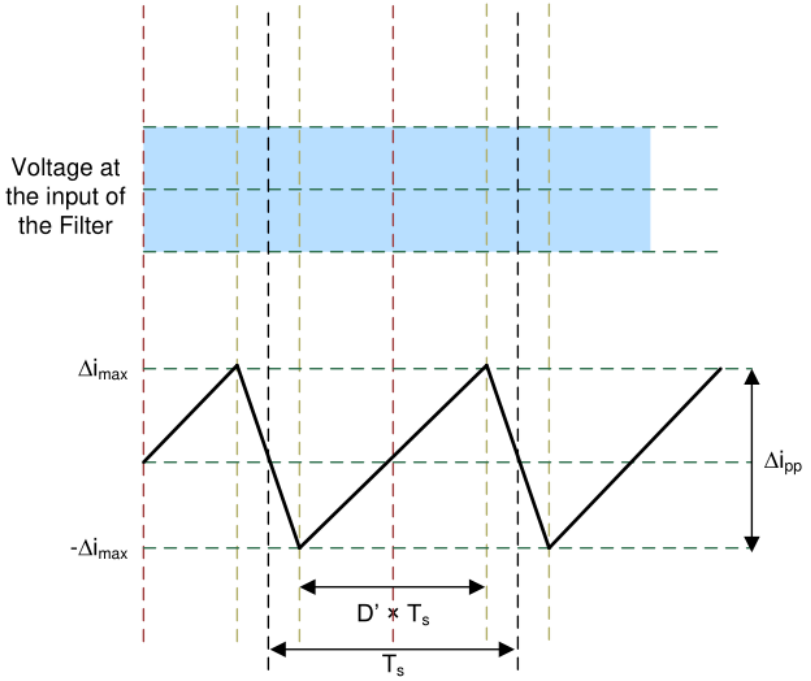


Ilustración 18. Forma de corriente en el rizado del inductor [13]

El voltaje que se produce en un inductor viene dado por la siguiente ecuación:

$$V = L_i \frac{di}{dt} \tag{1}$$

En nuestro caso particular para el diseño del rectificador Vienna se puede rescribir como:

$$\frac{V_{bus}}{2} - V_{in} = L_i \cdot \frac{\Delta i_{pp}}{D\prime \cdot T_s} \tag{2}$$

Para esta ecuación tenemos en consideración que $T_s = \frac{1}{F_{sw}}$, siendo T_s el periodo de conmutación y F_{sw} la frecuencia de conmutación, también nos encontramos con D` que es el ciclo de trabajo para el cual los Mosfet's en la conmutación están activados. Pará realizar el diseño en el control se tiene que D es el voltaje en el otro terminal del inductor por lo tanto con D = 1 los interruptores de potencia están apagados y se cumple que D´=1-D.

La ecuación con la que relacionamos tanto el ciclo de trabajo como el voltaje en el terminal del inductor es la siguiente

$$V_{xiN} = D \cdot \frac{V_{bus}}{2} \tag{3}$$

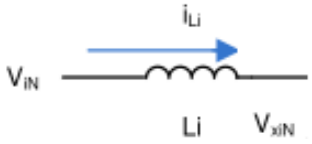


Ilustración 19.Inductor de entrada en el rectificado de Vienna [13]

Si reformulamos la ecuación anterior para conocer la forma que tiene el rizado de corriente para cualquier instante podemos llegar a:

$$\Delta i_{pp} = \frac{D\prime \cdot T_s \cdot (\frac{V_{bus}}{2} - V_{in})}{L_i} \tag{4}$$

Si ahora asumimos que el índice de modulación es m_a el ciclo de trabajo se puede ver como $D'=m_a \cdot \sin(\omega t)$ y si además se asume que el voltaje de entrada viene dado por:

$$V_{in} = D' \cdot \frac{V_{bus}}{2} \tag{5}$$

Podemos ver el rizado de corriente de la siguiente forma:

$$\Delta i_{pp} = \frac{\frac{V_{bus}}{2} \cdot T_s \cdot m_a \cdot \sin(\omega t) \cdot (1 - m_a \sin(\omega t))}{L_i} \tag{6}$$

De esta última ecuación podemos extraer como el pico de corriente está relacionado con el voltaje de entrada y la forma sinusoidal que el voltaje tiene. Pará obtener el valor máximo derivaremos respecto el tiempo.

$$\frac{d(\Delta i_{pp})}{dt} = K\{\cos(\omega t)\,(1 - m_a \cdot \sin(\omega t)) - m_a \cdot \sin(\omega t)\} = 0 \tag{7}$$

El valor máximo lo obtenemos cuando se cumple:

$$\sin(\omega t) = \frac{1}{2 \cdot m_a} \tag{8}$$

Desarrollando estas dos ecuaciones nos encontramos con dos ecuaciones las cuales nos aportan tanto el rizado de la corriente como el valor para el inductor con las características del sistema.

$$\Delta i_{pp} = \frac{\frac{V_{bus}}{2}\cdot T_s}{4\cdot L_i} \qquad\qquad L_i = \frac{\frac{V_{bus}}{2}}{4\cdot F_{sw}\cdot \Delta i_{pp_{max}}} \tag{9-10}$$

Con todas estas ecuaciones en mente, podemos seleccionar el inductor adecuado y el núcleo que lo contiene.

Las características con las que cuenta el diseño del rectificador de Vienna son las siguientes:

- Frecuencia de conmutación: 50 kHz
- Rizado de corriente tolerable: 10 %
- Potencia para 208 V_{RMS} : 1200 W
- Potencia para 208 V_{RMS} por cada fase: 400 W
- Corriente para 208 V_{RMS} : 3,33 A
- Potencia para 400 V_{RMS} : 2300 W
- Potencia para 400 V_{RMS} por cada fase : 766,66 W
- Corriente para 400 V_{RMS} : 3,33 A
- El valor de diseño del inductor aplicando (9) será de : 3,18 mH

5.2 DISEÑO DEL CONDENSADOR DE BUS

El condensador del bus de continua es el responsable de eliminar todo el rizado que pueda ser generado, para ello aplicaremos la siguiente ecuación, con la que hallaremos el valor mínimo de capacidad que debe poseer el condensador.

$$C = \left(\frac{1}{3}\right)\cdot \frac{P_{ac}}{4\cdot f\cdot (V^2-(V-\Delta V)^2} \tag{11}$$

Las características con las que cuenta el correcto diseño del condensador son las siguientes:

- Rizado de tensión en el bus de continua tolerable: 2%
- Rizado para el bus de continua a 208 V_{RMS} : 12 V
- Valor mínimo de rizado para el bus de continua a 208 V_{RMS} : 588 V
- Valor mínimo de capacidad en el condensador para 208 V_{RMS}: 115,7 µF
- Rizado para el bus de continua a 400 V_{RMS} : 14 V
- Valor mínimo de rizado para el bus de continua a 400 V_{RMS} : 686 V
- Valor mínimo de capacidad en el condensador para 400 V_{RMS}: 195,5 µF

5.3 DISEÑO DEL SENSADO PARA EL VOLTAJE DE ENTRADA AC

El diseño de esta etapa se realiza mediante dos amplificadores, él AMC1301DWR para la adquisición de señal, junto con una red de resistores para conseguir un divisor con una ganancia a la entrada del sistema para a ajustar el rango de voltaje permisible con las características que ofrece el operacional.

$$V_{Rin} = \frac{V_{AMC1301}}{G_{divisor}} \tag{12}$$

Después de esta primera etapa se llega a una etapa con la que se puede regular la ganancia del sistema mediante una red de resistencias y el amplificador operacional OPA320A, en configuración de amplificador diferencial.

$$G_{dif} = \frac{R_h}{R_g} \tag{13}$$

A la salida de la etapa de amplificación se añade un filtro RC paso bajo para eliminar componentes de altas frecuencias que se puedan añadir así como el ruido que se pudiese originar.

Las características de los componentes del diseño de sensado de ac son las siguientes:

- Resistencia de entrada Re: 3 MΩ
- Resistencias de entrada Rf: 2 kΩ
- Ganancia del divisor de tensión $G_{divisor}$: 0,0006 V/V
- Rango de voltaje del AMC1301: ± 0,25 V
- Rango de entrada de voltaje V_{Rin}: ± 416,9 V
- Resistencias etapa de amplificación Rg: 9,4 kΩ
- Resistencias etapa de amplificación Rh: 7,5 kΩ
- Ganancia diferencial de la etapa amplificación G_{dif} : 0,79 V/V
- Resistencia del filtro pasa bajas Rfltr: 68,1 Ω
- Condensador del filtro pasa bajas Cfltr: 0,1 µF

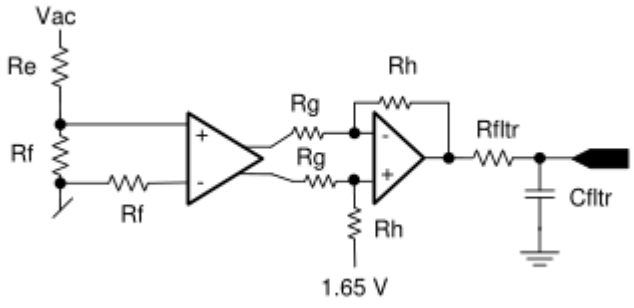


Ilustración 20.Sensado voltaje de entrada ac [13]

5.4 DISEÑO DEL SENSADO PARA EL VOLTAJE DEL BUS DE SALIDA DC

Para el diseño del sensado del voltaje de bus, se emplea una red de resistencias con la que se consigue una ganancia a la entrada del sistema de adquisición de señal, mediante el amplificador operacional AMC1301.

$$V_{Rin} = \frac{V_{AMC1301}}{G_{divisor}} \tag{14}$$

En la segunda etapa se implementa una etapa para ajustar la amplificación con el diseño de la red de resistencias y con el amplificador OPA320A, el amplificador está configurado como amplificador diferencial.

$$G_{dif} = \frac{R_d}{R_c} \tag{15}$$

A la salida del amplificador diferencial se añade un filtro paso bajo, con el cual eliminar las altas frecuencias o el ruido que se pueda acoplar.

Las características de los componentes del diseño de sensado del bus de continua son las siguientes:

- Resistencia de entrada Ra: 3 MΩ
- Resistencias de entrada Rb: 2,74 kΩ
- Ganancia del divisor de tensión $G_{divisor}$: 0,00079 V/V
- Rango de voltaje del AMC1301: ± 0,5 V
- Rango de entrada de voltaje V_{Rin}: ± 631,27 V
- Resistencias etapa de amplificación Rc: 9,4 kΩ
- Resistencias etapa de amplificación Rd: 7,5 kΩ
- Ganancia diferencial de la etapa amplificación G_{dif} : 0,79 V/V
- Resistencia del filtro pasa bajas Rfltr: 68,1 Ω
- Condensador del filtro pasa bajas Cfltr: 0,1 µF

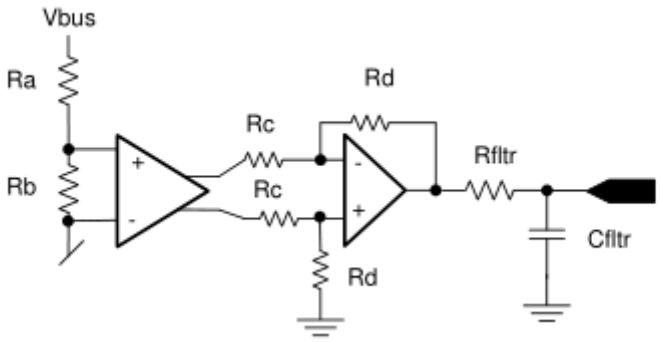


Ilustración 21.Sensado del voltaje de bus [13]

5.5 DISEÑO DEL FILTRO RC DE SALIDA SENSADO

Se emplea un filtro paso bajo RC en la salida de las dos configuraciones de sensado tanto para el sensado ac, como para el bus dc, antes de su conexión al microcontrolador.

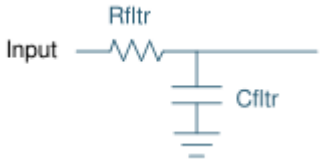


Ilustración 22.Filtro RC sensado ac y bus dc [3]

$$G_{fltr} = \frac{1}{1+R_{fltr}+C_{fltr}} \quad (16)$$

5.6 DISEÑO DEL SENSADO DE LA CORRIENTE DEL INDUCTOR

En el diseño del sensado de la corriente en los tres inductores de entrada del rectificador, cuenta con el sensor de efecto hall LTSR 6-NP y el amplificador operacional OPA4350UA, además de los elementos pasivos para su correcta configuración.

Ya que el sensor de efecto hall tiene un desplazamiento incorporado, este se ajusta con la inyección de un voltaje de offset.

Las características del diseño del sensado de corriente del inductor son las siguientes:

- Resistencia Re: 500 kΩ
- Resistencia Rf: 330 kΩ
- Voltaje nominal $V_{nominal}$: 0,625 V
- Intensidad nominal máxima $I_{nominal\_max}$: 12 A
- Voltaje de offset V_{offset}: 3,3 V

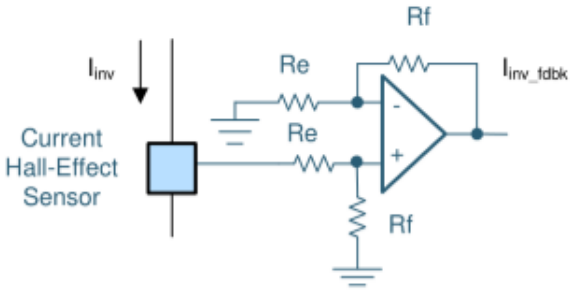


Ilustración 23.Diseño sensado de corriente del inductor [13]

$$I_{inv\_fdbk} = \frac{R_f}{R_e}\left(I_{inv} \cdot \frac{V_{nominal}}{I_{nominal\_max}} + V_{offset}\right) \quad (17)$$

6. RECTIFICADOR DE VIENNA TIPO T (TIDM-1000)

6.1 HARDWARE

6.1.1 Microcontrolador

6.1.1.1 MCU de la familia *C2000 F28377D*

Los microcontroladores “MCU” de la familia C2000 están optimizados para aplicaciones que requieren de control en tiempo real.

Cuenta con un convertidor A/D de alta velocidad y de alta calidad el cual nos permite una medición precisa de las señales a tratar en forma de corriente y voltaje. Además lleva incorporado un subsistema de comparación “CMPSS”, el cual nos permite la protección para sobre corrientes y sobretensiones que puedan surgir.

El microcontrolador ”MCU” implementado es el TSM320F28377D este microcontrolador cuenta con una potente unidad de coma flotante de 32-bits,diseñada para aplicaciones avanzadas en bucle cerrado tales como vehículos industriales, inversores solares o procesamiento de sensores y señales.

El TMS320F28377D cuenta con una arquitectura de doble núcleo la cual trabaja a una frecuencia de 200 MHz. Las CPU de la serie C28x además incorporan un nuevo acelerador TMU, con el que se pueden realizar de forma muy veloz la ejecución de algoritmos con operaciones trigonométricas, también dispone de un acelerador VCU con el que se reduce el tiempo en operaciones complejas comunes en aplicaciones codificadas.

El TMS320F28377D admite hasta 1 MB de memoria flash en la cual incorpora también código de corrección de errores “ECC” y cuenta con hasta 204 KB de SRAM, además contamos con dos secciones seguras disponibles de hasta 128-bits en cada CPU para la protección del código.

Se cuenta también con hasta cuatro convertidores A/D de en los que se puede elegir el modo de trabajo con 12-bits o 16-bits, en el subsistema analógico nos encontramos además una unidad de “S/H” para cada ADC.

Lleva incorporado en unidades periféricas de control 24 canales “PWM” y 16 canales de alta resolución (HRPWM).

También se dispone de 6 canales dobles de acceso directo a memoria “DMA” en los sistemas periféricos del microcontrolador como hasta 169 entradas de propósito general programables I/O “GPIO”.

En lo referente a las comunicaciones que incorpora se dispone de un puerto USB 2.0 además de comunicación CAN y tres puertos SPI , también cuenta con cuatro interfaces de comunicación serie (SCI/UART) y 2 interfaces I^2C.

La estructura del diagrama de bloques de TMS320F28377D y los pines de conexión del microcontrolador se muestra a continuación.

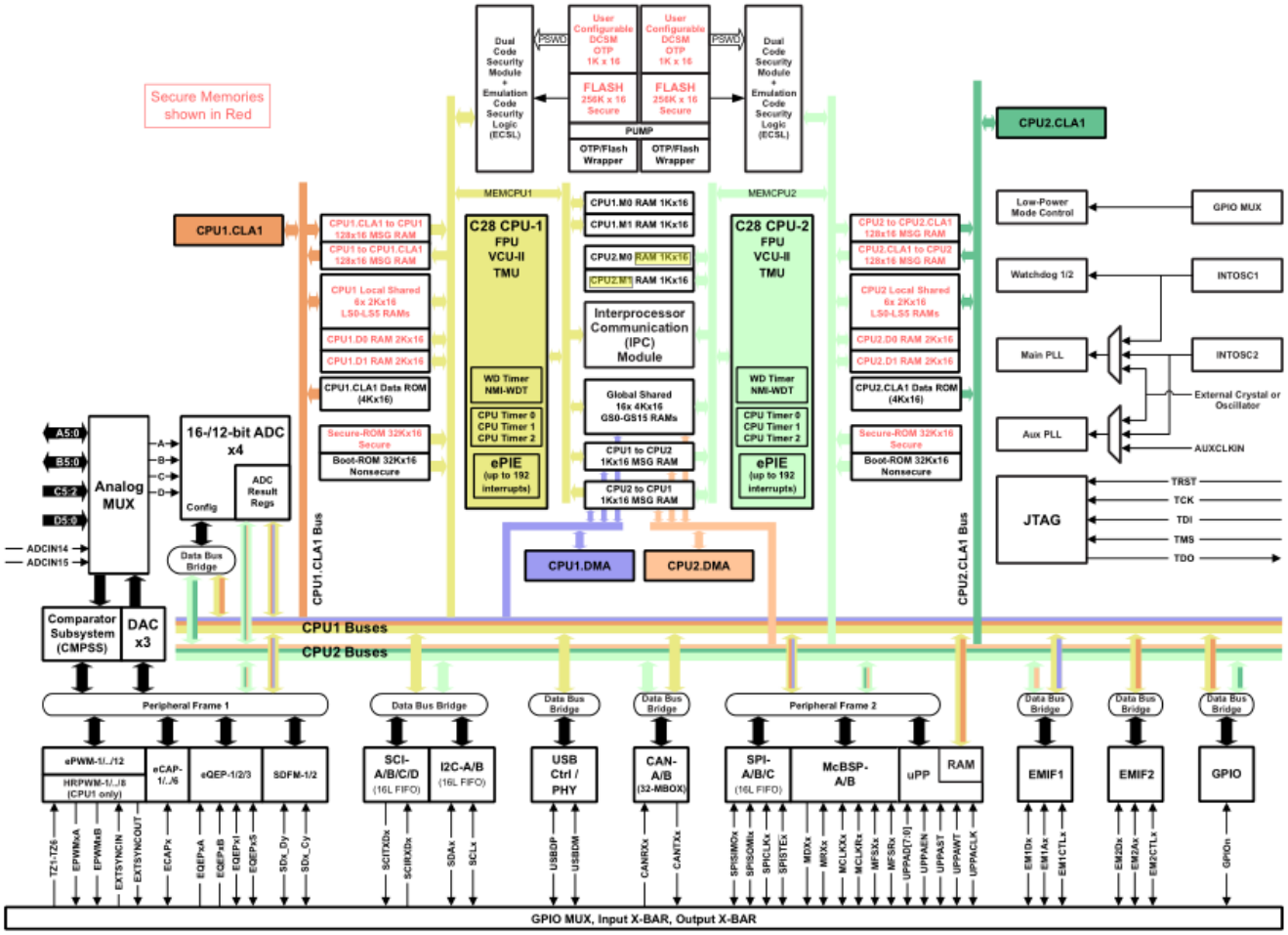


Ilustración 24. Diagrama de bloques del microcontrolador TMS320F2837D

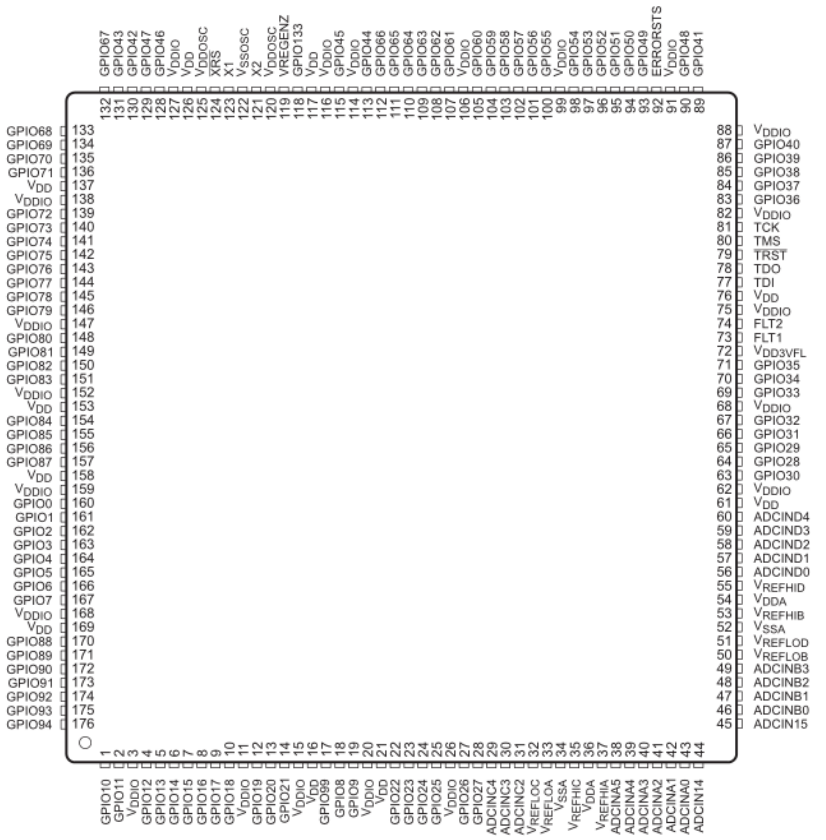


Ilustración 25.Distribución pines del TMS320F28377D

6.1.2 Adquisición de señales e instrumentación.

6.1.2.1 Amplificador de aislamiento AMC1301DWVR

El AMC1301 es un amplificador de aislamiento de precisión, el cual está aislado con una salida alejada de los circuitos que componen la entrada por una barrera de aislamiento capaz de bloquear las interferencias electromagnéticas .La barrera proporciona un aislamiento galvánico de hasta 7 kV_p(pico).

El uso de este dispositivo junto con fuentes de alimentación que cuenten con aislamiento consigue evitar que una línea de alto voltaje en modo común pueda interferir a los circuitos o los dañe.

La entrada del AMC1301 esta optimizada para realizar medidas de corriente empleando resistencias shunt o señales de voltaje de baja señal de hasta $\pm$ 250 mV

El amplificador de aislamiento, cuenta también con un bajo error de offset de $\pm$ 200µV a 25º y con un desvió de la señal de $\pm$ 2µ$V/ºC$.

Cuenta con una ganancia fija de 8,2 y con error de ganancia y de desviación muy reducido de $\pm$ 0,3µV a 25ºC.

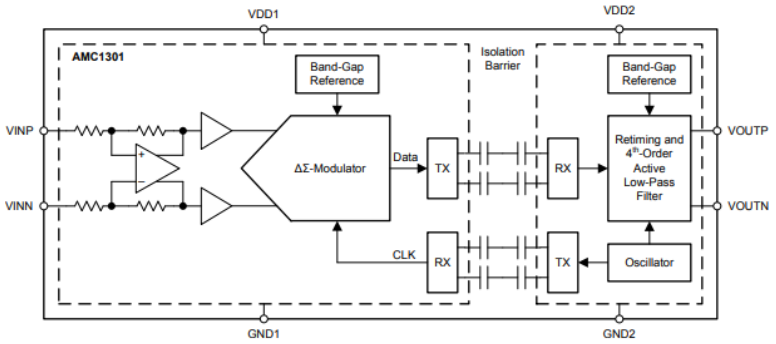
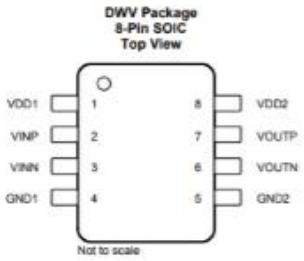


Ilustración 26.Diagrama de bloques AMC1301



Pin Functions

PIN		I/O	DESCRIPTION
NAME	NO.		
GND1	4	—	High-side analog ground
GND2	5	—	Low-side analog ground
VDD1	1	—	High-side power supply, 3.0 V to 5.5 V. See the *Power Supply Recommendations* section for decoupling recommendations.
VDD2	8	—	Low-side power supply, 3.0 V to 5.5 V. See the *Power Supply Recommendations* section for decoupling recommendations.
VINN	3	I	Inverting analog input
VINP	2	I	Noninverting analog input
VOUTN	6	O	Inverting analog output
VOUTP	7	O	Noninverting analog output

Ilustración 27.Configuración y función de los pines del amplificador AMC1301

6.1.2.2 Amplificador operacional OPA320AIDBVR

El amplificador operacional OPA320A se encuentra en una familia de amplificadores CMOS con un bajo voltaje y precisión diseñados para trabajar con ruidos muy reducidos del orden de 7nV/$\sqrt{Hz}$ para 10 kHz y con un ancho de banda muy amplio, llegando así a los 20 MHz, el offset que nos ofrece este operacional es muy reducido alcanzando 150µV como máximo.

Además cuenta con un rechazo en modo común (CMRR) de 114 dB en el rango de entrada y nos permite tener un voltaje rail-to-rail, el slew rate del que dispone este operacional es de 10 V/µs.

El OPA320A se ajusta perfectamente para aplicaciones de baja potencia y se puede alimentar con tan solo un potencial el cual se encuentra en el rango de 1,8 hasta 5,5 V, lo cual los hace ideales para aplicaciones en la que se alimente solamente con baterías sin regulación, así también opera con corrientes muy pequeñas del orden de 1,45 mA por cada canal.

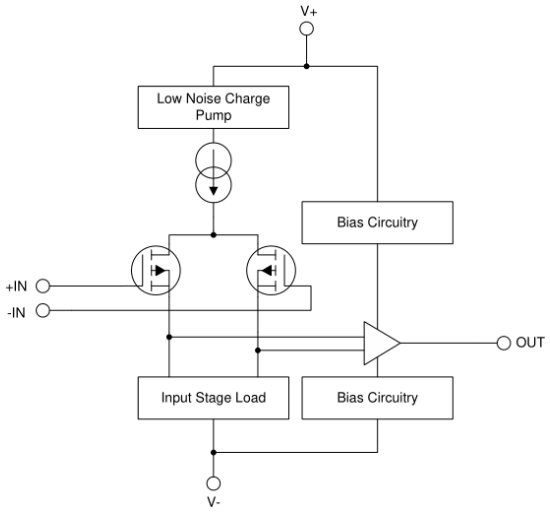


Ilustración 28.Diagrama de bloques del amplificador operacional OPA320A

6.1.2.3 Amplificador operacional OPA4350UA

El OPA4350 es un amplificador operacional el cual contiene cuatro amplificadores operacionales con sus respectivas entradas y salidas para cada uno.

La serie en la que se encuentra el OPA4350, son amplificadores operacionales CMOS que cuentan voltaje rail-to-rail tanto en la entrada como a la salida, también están diseñados para trabajar con una baja tensión y una alimentación de hasta 7 V en forma simple.

Tienen un bajo ruido definido como 5nV/$\sqrt{Hz}$ y un bajo THD+ruido de 0,0006% de la señal, el ancho de banda del OPA4350 es muy amplio llegando a 38 MHz lo cual permite trabajar a frecuencias muy elevadas, también cuenta con un slew rate elevado de 22V/µs.

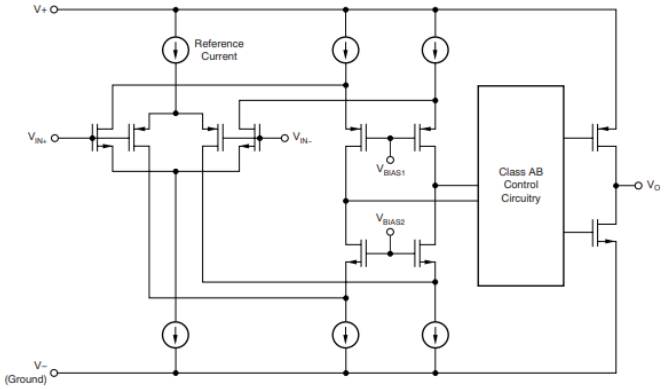


Ilustración 29.Diagrama de bloques OPA4350 para un canal

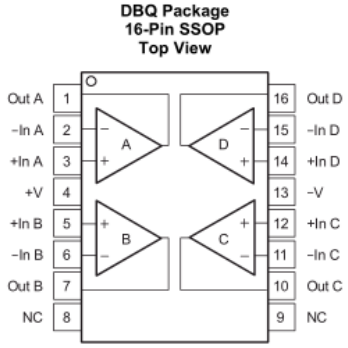


Ilustración 30.Configuración de pines del amplificador operacional OPA4350

6.1.2.4 Transductor de corriente LTSR 6-NP

El LTSR 6 –NP es un transductor de corriente para la medición electrónica tanto de corriente continua, alterna o tren de pulsos, el cual cuenta con una separación galvánica entre el circuito primario y secundario.

Es un transductor de corriente que emplea el efecto hall y esta compensado en lazo cerrado, la alimentación del transductor se realiza de forma unipolar, incorpora medida de resistencia y acceso al voltaje interno de referencia.

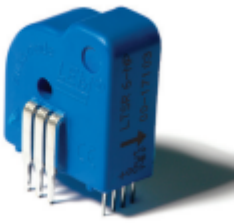


Ilustración 31.Transductor de corriente LTSR 6-NP

6.1.3 Convertidores DC/DC y reguladores de tensión.

6.1.3.1 Convertidor de aislamiento DC/DC DCH010505

El DCH010505 es un convertidor aislado DC/DC de 3 kVDC de aislamiento con una potencia de salida de 1 W, está encapsulado en formato SIP de 7 pines, la serie DCH01 solamente requiere los componentes externos mínimos para su funcionamiento.

El diseño del convertidor nos permite una salida con un canal de salida o una salida doble, alcanzando hasta un 78% de eficiencia en el convertidor.

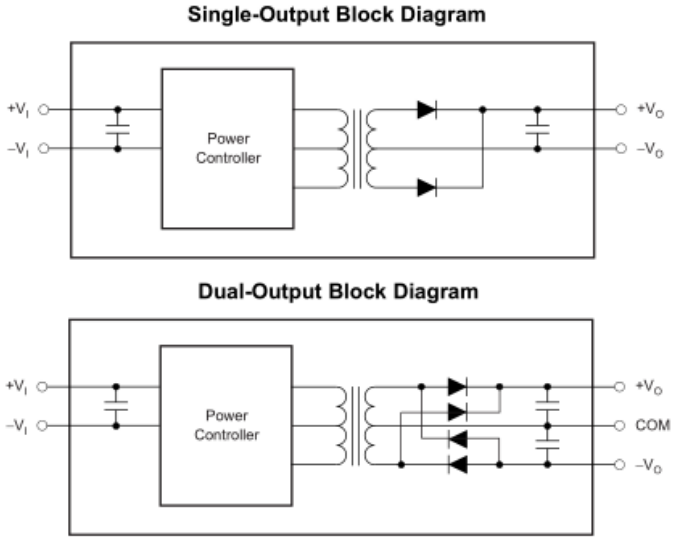


Ilustración 32.Diagramas de bloques del convertidor DCH010505 para salida simple o doble

6.1.3.2 Convertidor DC/DC RP-1212S

El RP-1212S es un convertidor DC/DC con un alto voltaje de aislamiento de 5,2 kVDC en un encapsulado muy reducido. El convertidor se ajusta a las necesidades de aplicaciones con IGBT o Mosfet's.

El convertidor cuenta con una eficiencia típica del 75 % hasta el 78 % con una posibilidad de voltaje de entrada de [5, 9, 12, 15,24] VDC, con una salida de 12 VDC y una intensidad de salida de 84 mA.

El convertidor soporta 1 W a la salida, además de contar con salida simple y salida doble.

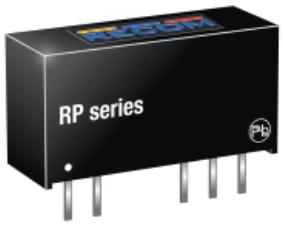


Ilustración 33.Convertidor DC/DC RP-1212S

6.1.3.3 Convertidor conmutado DC/DC PTH08080WAH

El PTH08080 es un convertidor DC/DC conmutado el cual es capaz de entregar hasta un máximo de 2,25 A de corriente de salida a una temperatura de 85ºC. Las fuentes conmutadas PTH08080W permiten aportar una corriente con una eficiencia mayor a como lo hacen los convertidores DC/DC lineales, eliminando así el uso de un disipador de calor para el circuito integrado.

El convertidor PTH08080 nos permite un rango de voltaje de salida que se puede ajustar desde 0,9 a 5,5 V, tan solo utilizando una resistencia externa. Las características del convertidor permiten también el bloqueo de bajo voltaje "UVLO", inhibición de encendido y apagado además de protección tanto como para sobrecorriente como por sobretemperatura.

El rango de voltaje de entrada que admite este convertidor va desde 4,5 hasta 18 V, contando con una eficiencia de hasta el 93%.

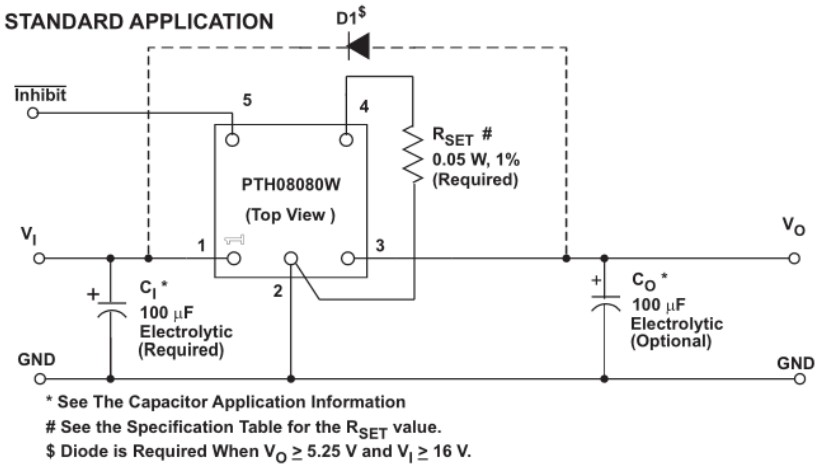


Ilustración 34.Aplicación standard con el convertidor PTH08080

6.1.3.4 Regulador de tensión TVL1117-33IDCY

EL TVL1117 es un regulador de voltaje positivo con diseño el cual permite proporcionar hasta 800 mA de corriente de salida. El regulador permite el ajuste de la tensión de salida, se puede encontrar con tensiones regulables o como tensiones fijas [1,5 V, 1,8 V 2,5 V 3,3 V y 5 V]

La regulación máxima que se puede realizar de carga es del 0,4 % y el 0,2 % en regulación en línea.

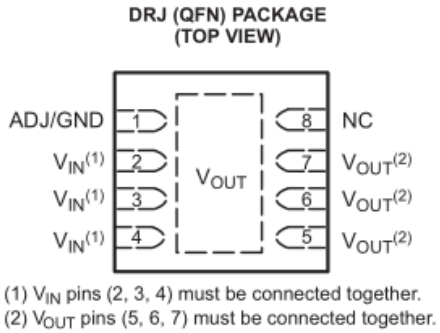


Ilustración 35.Configuración pines TLV117

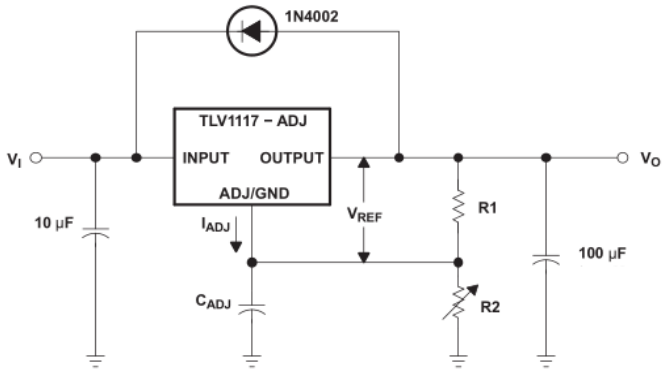


Ilustración 36.Ejemplo de aplicación con el TVL1117

La ecuación que se emplea para ajustar el regulador de tensión con esta configuración típica es la siguiente:

$$V_{out} = V_{ref} \cdot \left(1 + \frac{R1}{R2}\right) + \left(I_{adj} \cdot R2\right)$$

6.1.4 Semiconductores

6.1.4.1 Diodo de potencia C4D08120A

El C4D08120A es un diodo Schottky de carburo de silicio el cual soporta 1,2 kV de voltaje de pico repetitivo inverso, con un funcionamiento en altas frecuencias y coeficiente de temperatura positivo, para corriente directa alcanza hasta 12 A para 135 ºC.

Ilustración 37.Encapsulado TO-220-2 para el C4D08120A

6.1.4.2 Diodo de potencia MBRM130LT1G

El MBRM130LT1G es un diodo Schottky de potencia de montaje superficial (SMD), el cual soporta un voltaje de pico repetitivo inverso de hasta 30 V y una corriente directa de 1 A.

Es un diodo Schottky ideal para emplear con baja tensión, también es capaz de trabajar en altas frecuencias, además de tener un baja caída de tensión V_F alrededor de 0,4 V.

SOD−123
CASE 425
STYLE 1

Ilustración 38.Encapsulado diodo MBRM130LT1G

6.1.4.3 Transistor Mosfet de potencia IPP65R190C7FKSA1

El IPP65R190C7FKSA1 es un transistor Mosfet de potencia el cual soporta un alto voltaje, el diseño del MOSFET está libre de plomo y halógenos, soporta un voltaje entre los terminales de drenador-surtidor V_{DS} de hasta 700 V y soporta una intensidad de 13 A en conducción continua e intensidades pulsantes de hasta 49 A.

Ilustración 39. Encapsulado para el transistor Mosfet IPP65R190C7FKSA1

6.1.5 Driver de disparo

6.1.5.1 Driver de disparo aislado de dos canales UCC21520DWR

El driver UCC21520, es un driver de puerta aislado de doble canal, está diseñado para disparar transistores Mosfet de potencia, IGBT y SiC Mosfet's hasta una frecuencia de 5 MHz.

El lado de la entrada del driver cuenta con un aislamiento de hasta 5,7 kV_{rms}, la tensión de alimentación que permite este driver, llega hasta los 25 V, el driver tiene altos niveles de eficiencia y densidad de potencia.

También incorpora entradas compatibles con tecnología TTL y CMOS.

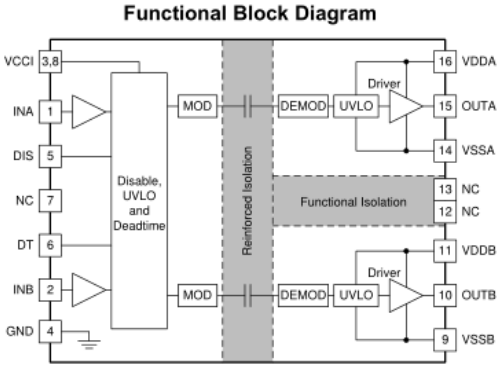


Ilustración 40.Diagrama de bloques del driver UCC21520

6.1.6 Placa de circuito impreso TIDM-1000

La placa de circuito impreso (PCB) en la que se montan todos los componentes para el funcionamiento del rectificador Vienna es la TIDM-1000, la cual incluye todo tipo de componentes.

Cuenta con amplificadores operacionales tales como AMC1301DWVR, OPA320AIDBVR, OPA4350UA los cuales adquirirán las señales a tratar o el transductor de corriente por efecto hall LTSR 6-NP, además de convertidores DC/DC aislados y conmutados como puede ser el DCH010505SN7, RP-1212S o el convertidor conmutado PTH08080WAH, para reguladores de tensión podemos encontrarnos con el circuito integrado TVL1117-33IDCY.

Se emplean también semiconductores tales como los diodos de potencia C4D08120A, MBRM130LT1G y los MOSFET de potencia para la conmutación del circuito IPP65R190C7FKSA1.

La regulación del control de rectificador Vienna se implementa con el microcontrolador TMS320F28377D que cuenta con la potencia y la velocidad necesaria para su correcto funcionamiento.

En el diseño del prototipo TIDM-1000, cuenta con elementos pasivos tales como bobinas, condensadores y resistencias así como con terminales de prueba alrededor de todo el circuito impreso para verificar su funcionamiento.

Ilustración 41.Disposicion de elementos en la placa de circuito impreso TIDM-1000 [13]

6.2 SOFTWARE

6.2.1 Plataforma de programación

Para poder configurar y cargar la programación del microcontrolador TMS320F28377D, se utiliza la plataforma "Code Composer Studio", aquí será donde se abra el proyecto del rectificador Vienna implementado en la placa TIDM-1000.

Para acceder al modelo adoptado, tendremos que utilizar dos herramientas que contiene CCS, se emplearan las herramientas "controlSUITE" y "powerSUITE"

6.2.2 Modelo adoptado "Vienna Rectifier Three Phase PFC using F2837x"

El modelo de programación empleado para la placa TIDM-1000 "Vienna Rectifier Three Phase PFC using F2837x", podemos acceder mediante controlSUITE.

Cuando tenemos abierto el buscador de controlSUITE vemos que nos aparecen diversas soluciones para todo tipo de aplicaciones de conversión de energía.

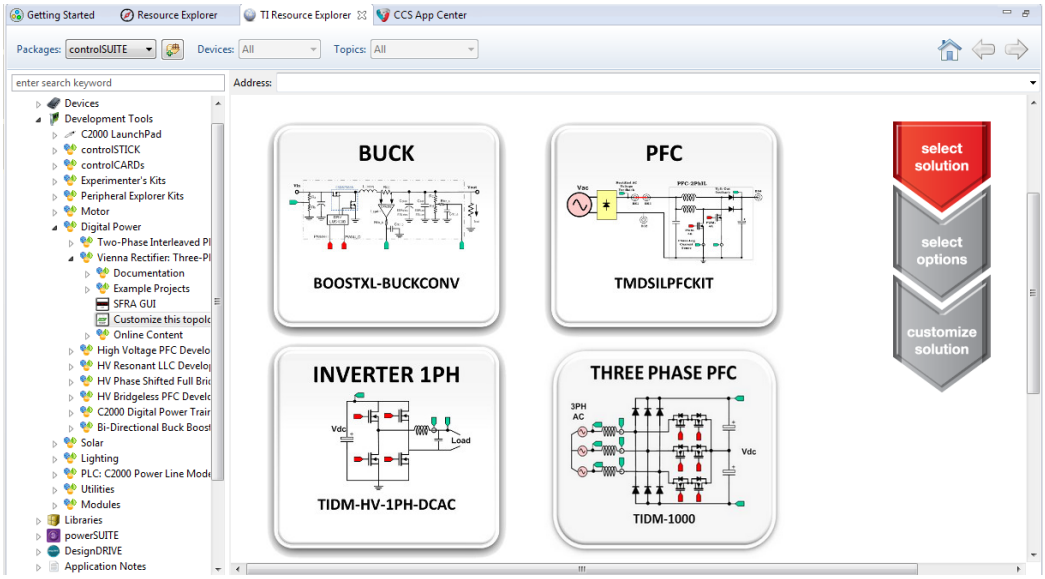


Ilustración 42.Selección modelo de programación para TIDM-1000

Ya dentro de la solución seleccionada del TIDM-1000, continuamos con seleccionar las opciones que tenemos disponibles.

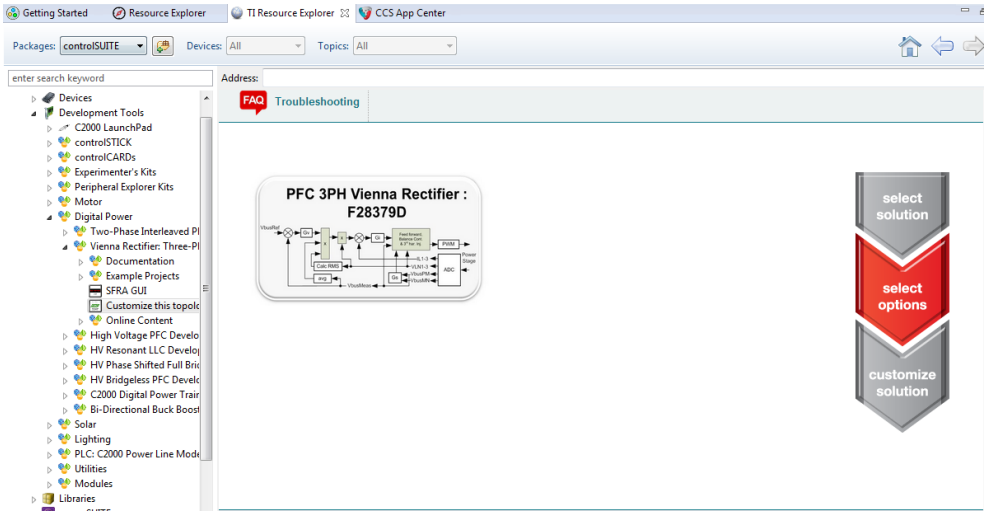


Ilustración 43.Selección de opciones para TIDM-1000

Seleccionamos la opción que tenemos disponible y la guardamos para poder importar el modelo adoptado para el TIDM-1000.

Ya importado el proyecto nos aparece el archivo “main.cfg”, desde el cual podremos configurar las características que va a tener nuestro sistema.

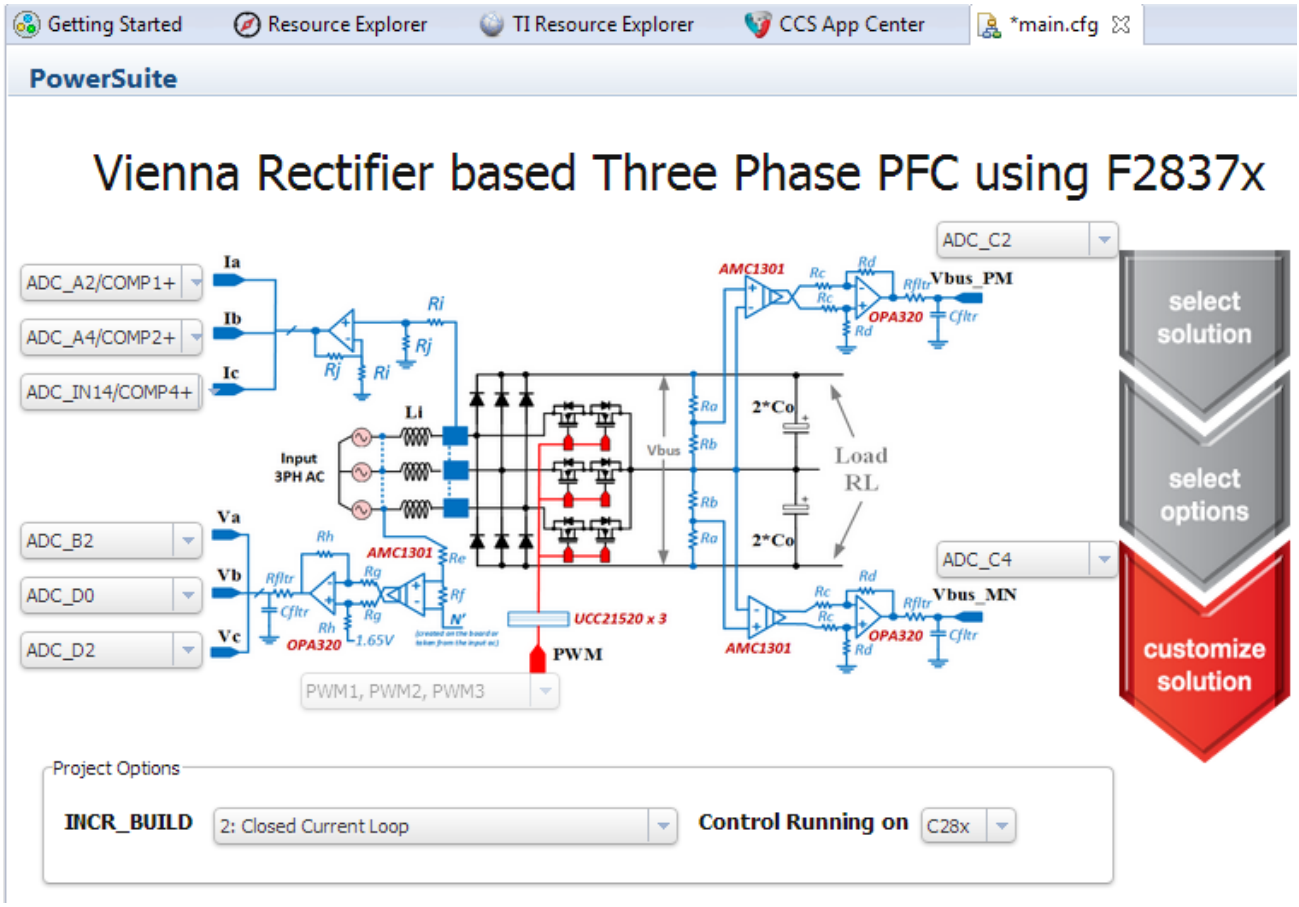


Ilustración 44.Configuración Software TIDM-1000

Ahora ya podemos empezar a configurar la programación del sistema, vemos aparece un esquemático, en el que se pueden apreciar los puntos de sensado y donde están ubicados así como la unidad que dispara los transistores Mosfet.

También observamos como aparecen las opciones de proyecto, en las cuales podremos elegir si trabajamos en bucle abierto o aplicando los lazos de control, que dispone el modelo de programación, lazo por corriente, lazo por tensión y lazo de compensación en la tensión de los condensadores.

Mediante la siguiente ilustración se puede ver como contamos con herramientas para poder ajustar y compensar el diseño de los lazos de control que podemos seleccionar.

Contamos con una adquisición de los parámetros de la etapa de potencia con lo que va a operar la placa TIDM-1000, para ajustar la programación a dichos parámetros.

Podremos elegir desde la frecuencia de conmutación de los drivers de los transistores Mosfet hasta que tensión va ser aplicada a la entrada del sistema o la potencia que deseamos que se disipe a la salida del sistema.

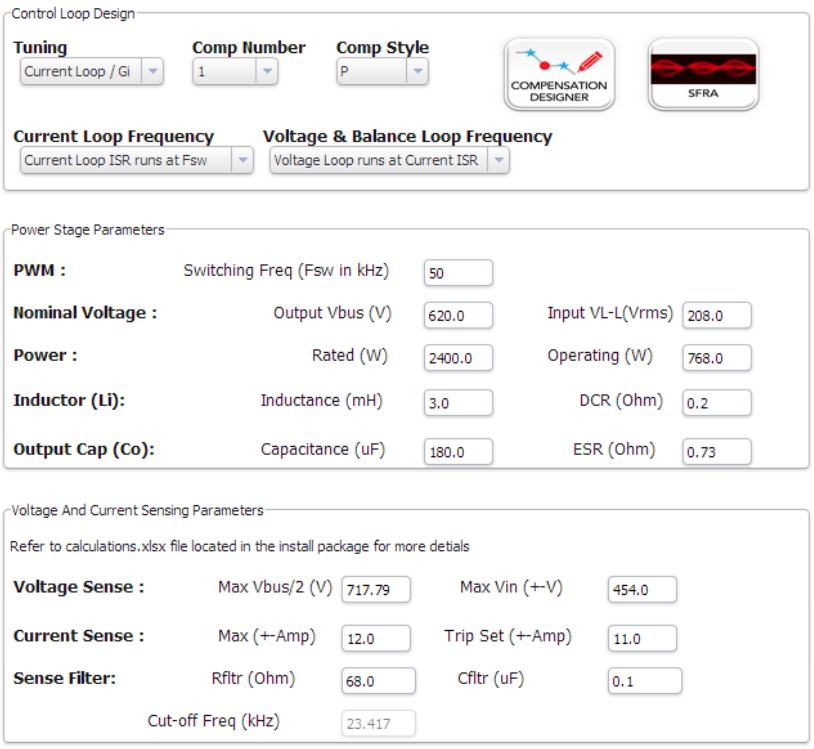


Ilustración 45.Configuración parámetros TIDM-1000

Además de los parámetros de la etapa de potencia, se observa cómo podemos ajustar los parámetros de sensado de la señal, con los que el microcontrolador TMS320F28377D, realiza la adquisición de datos.

Cuando se tiene claro que parámetros vamos a escoger, nos disponemos a guardar los ajustes del rectificador de Vienna implementado en la placa TIDM-1000.

Con ello podemos ver el explorador de proyectos donde nos aparece la estructura de todos los archivos con los que cuenta el proyecto.

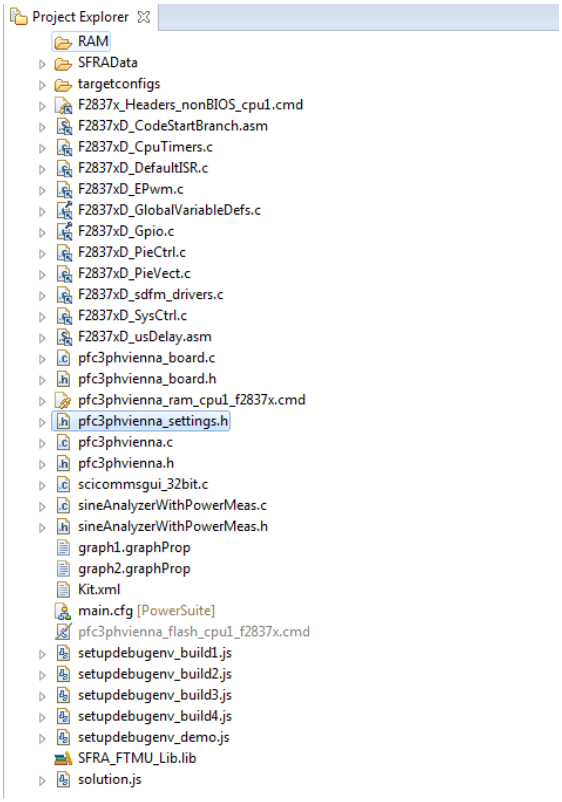


Ilustración 46.Estructura del proyecto "pfc3phvienna" para TIDM-1000

En el proyecto se encuentran todos los archivos necesarios para poder cargar y correr la programación, contamos con archivos de configuración del microcontrolador, así como archivos específicos de la solución que configuramos en etapas anteriores.

Los archivos que modificamos continuamente se realizan mediante el archivo “main.cfg”, con el que modificamos los archivos “pf3phvienna” y “solution.js”, para cada configuración elegida, se recomienda no modificar los archivos, ya que puede darse el caso que el proyecto no funcione correctamente.

Un ejemplo del código principal que se genera de la solución adoptada “pf3phvienna” del rectificador de Vienna implementado en la placa TIDM-1000, se adjuntan en el anexo II Software.

6.2.3 Modos de operación

En este apartado vamos a ver las opciones de proyecto con las que contamos a la hora de realizar la configuración, las cuales son las siguientes:

1. Lazo abierto
2. Lazo cerrado de corriente
3. Lazo cerrado de corriente y voltaje
4. Lazo cerrado de corriente, voltaje y balance de bus de continua

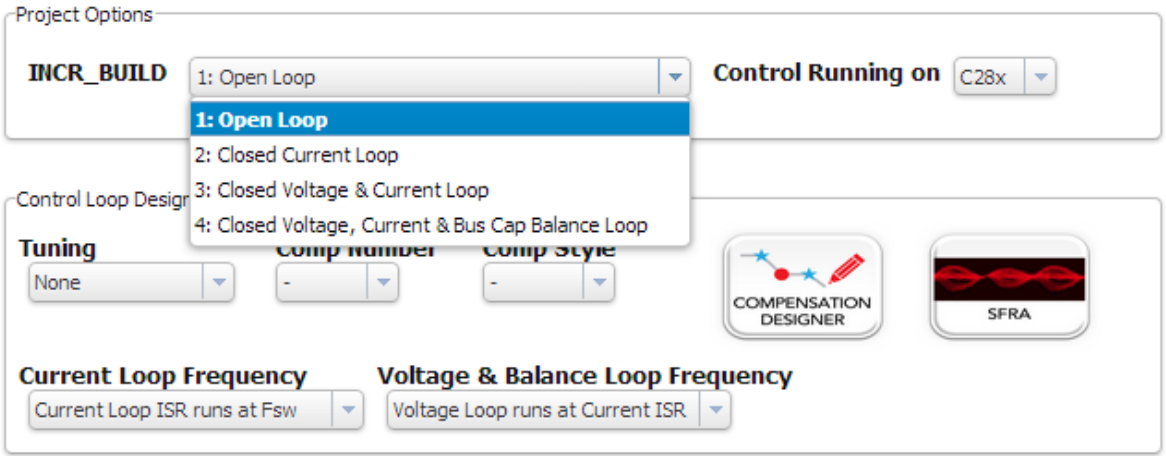


Ilustración 47.Elección modo de operación del sistema

6.2.4 Ajuste de los lazos de control

Una de las herramientas que nos ofrece la solución adoptada, es poder ajustar los lazos de control, dependiendo del control que estemos implementado con uno de los lazos o con todos ellos, para ello accedemos a “compensation designer” en el archivo main.cfg , desde donde podremos ajustar los lazos de control elegidos, analizando su efecto tanto en magnitud como en fase.

En todos ellos podremos ajustar el regulador PID, dependiendo del lazo de control que se vaya a implementar en el control, también se observa como disponemos de los polos y los ceros del sistema planta y la lista de los coeficientes que se están empleando.

En la siguiente ilustración se puede ver un ejemplo del lazo de tensión y corriente en bucle cerrado, en el que nos muestra si el sistema es estable o no y los márgenes de ganancia y de fase.

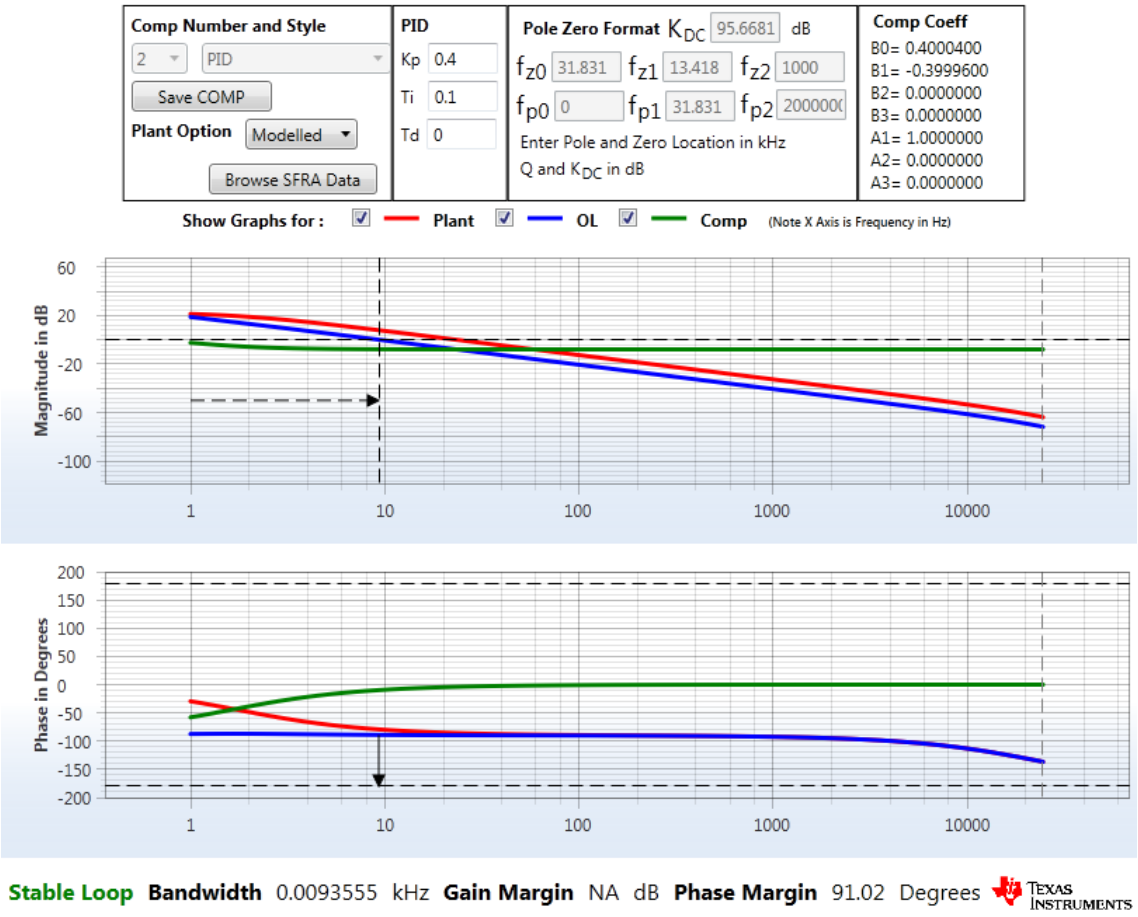


Ilustración 48.Ajuste lazos de control de corriente y tensión

6.2.5 Flujograma TIDM-1000

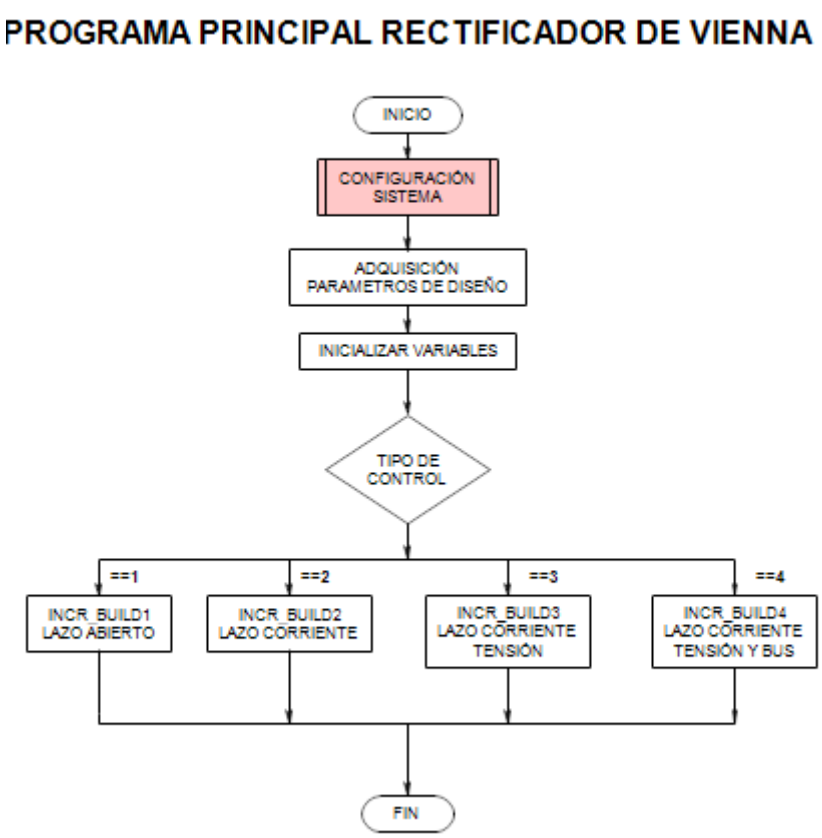


Ilustración 49.Flujograma principal TIDM-1000

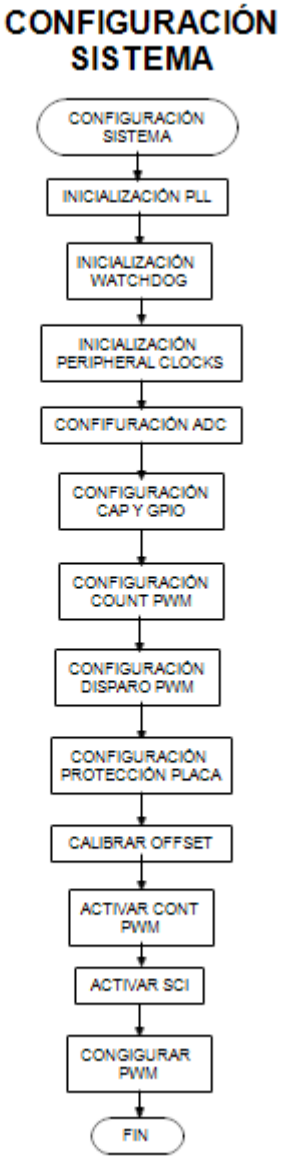


Ilustración 50.Flujograma configuración TIDM-1000

7. SIMULACIÓN

7.1 Modelo rectificador de Vienna tipo T

En este apartado, se muestra el modelo que se ha empleado para la realización de las simulaciones del rectificador de Vienna consideradas en la ejecución de este proyecto.

El modelo está basado en el rectificador de Vienna tipo T, con los parámetros hardware empleados en el modelo físico de la placa TIDM-1000, las características de esta configuración se enumeran a continuación:

- Inductor primario de filtrado: 3 mH
- Resistencia inductor: 0,02 Ω
- Malla de resistencias equivalente en la salida: 765 kΩ
- Condensador de bus: 220 µF
- Frecuencia de conmutación: 50 kHz

Estos parámetros son con los que cuenta el modelo físico y que se han implementado en el modelo de la simulación, para realizar las simulaciones conforme a la placa TIDM-1000.

En nuestro modelo además se modelan unas resistencias de precarga que ayudan a los condensadores realizar su carga y así limitar la corriente en el momento del arranque.

- Resistencia de precarga: 50 Ω

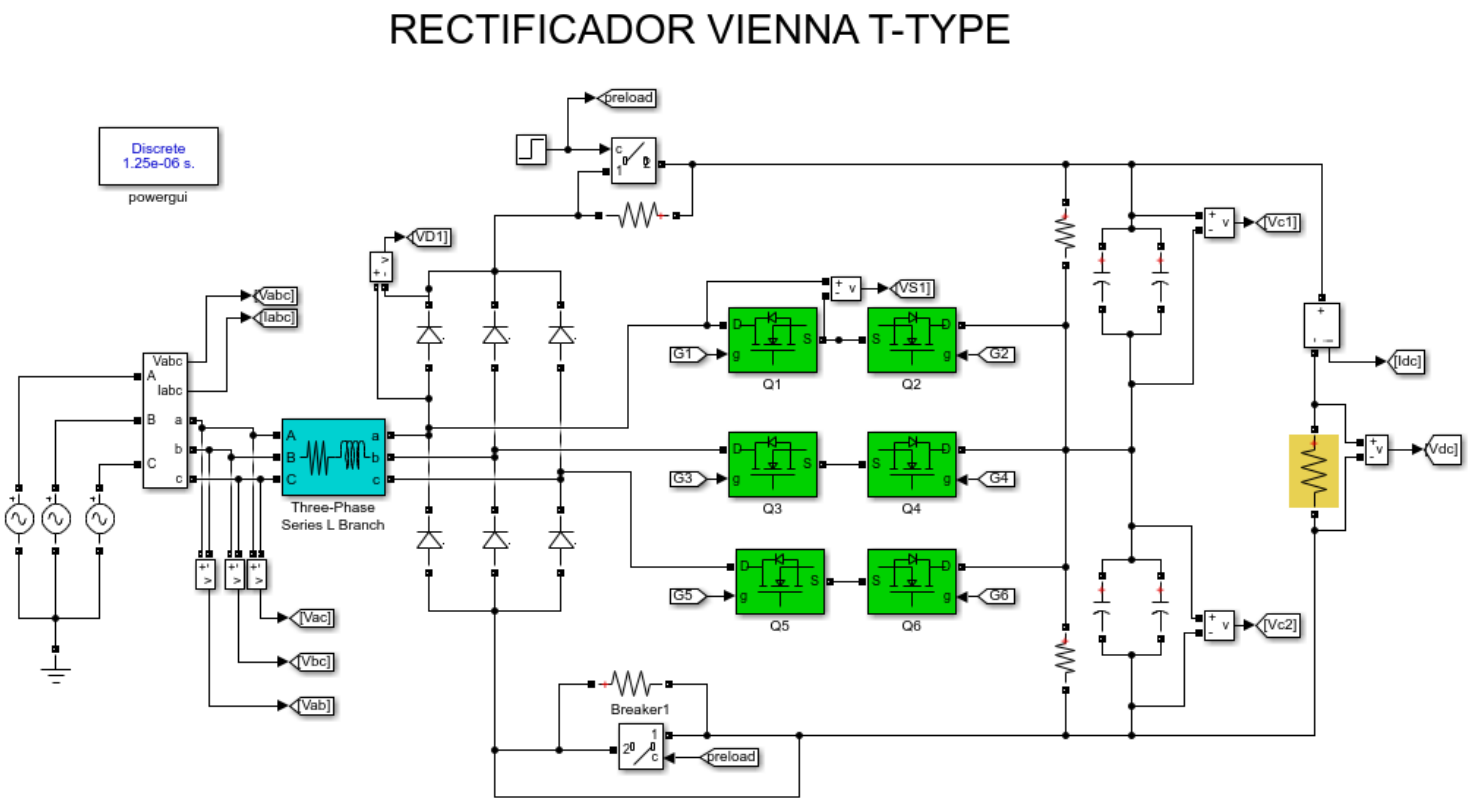


Ilustración 51. Modelo rectificador Vienna tipo T configurado para placa TIDM-1000

7.2 Control rectificador

El rectificador de Vienna requiere de tres tipos de control:

- Lazo de corriente
- Lazo de regulación del bus de continua
- Lazo de compensación de los condensadores

En el modelo de control se pueden observar los lazos implementados así como la etapa de modulación de ancho de pulso PWM, que nos permite realizar el disparo de los Mosfet's con la implementación de dicho control.

A continuación se muestra una aclaración grafica del disparo de los Mosfet's.

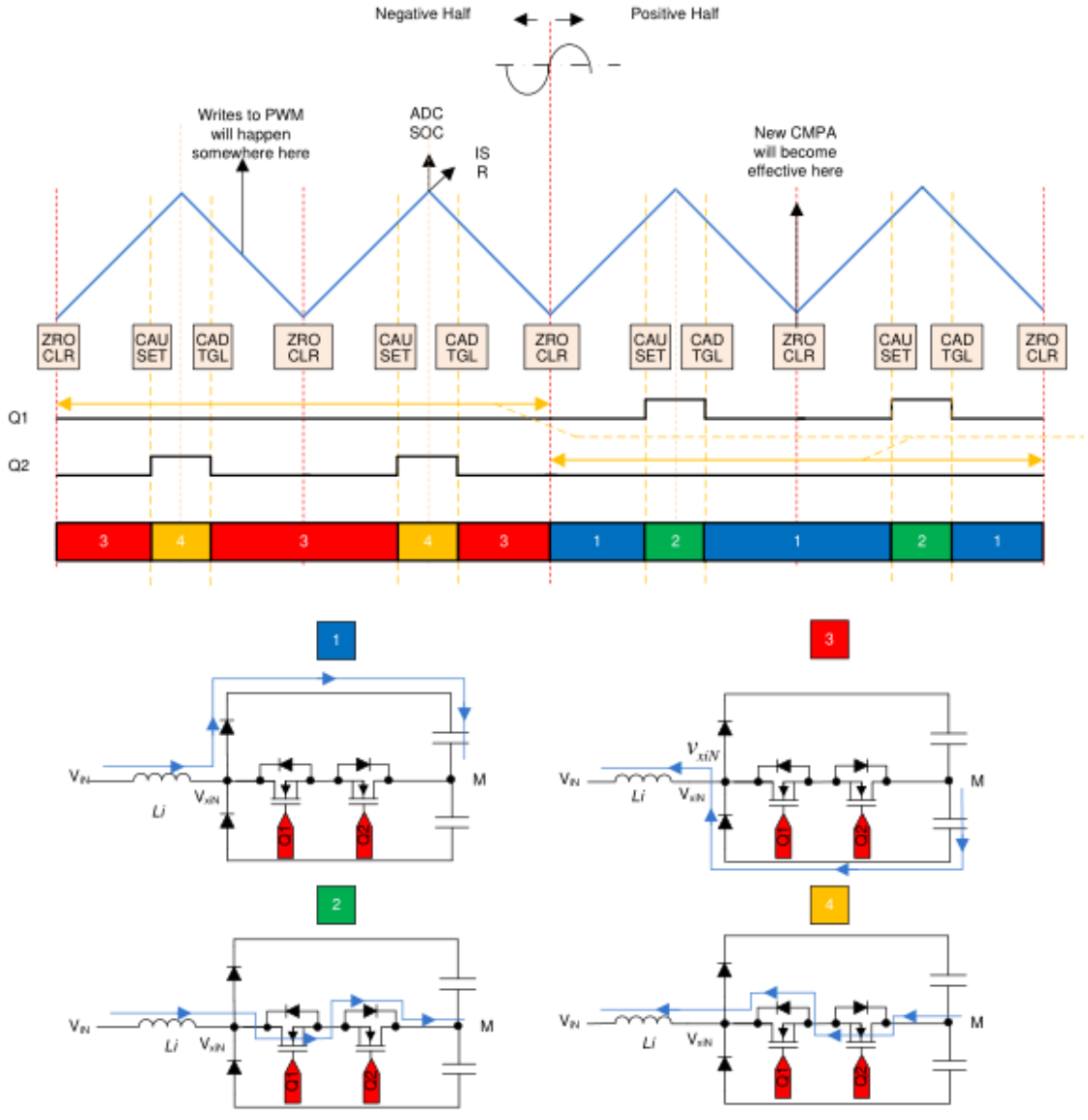


Ilustración 52.Ejemplo disparo de los Mosfet para el modelo del rectificador Vienna tipo T [13]

En la siguiente ilustración se puede observar el bloque donde se ha implementado el control, se ilustran los elementos de entrada al sistema de control y las salidas, que en nuestro caso será el disparo de los transistores Mosfet de potencia.

En la etapa de control podemos ver como se incorpora un disparo para su activación, la activación se da unos instantes después de haberse realizado la precarga de los condensadores.

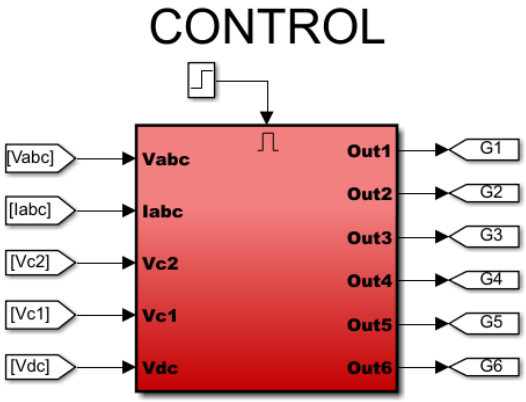


Ilustración 53.Entradas/Salidas de la etapa de control

En siguiente ilustración se puede observar los lazos de control que se han implementado para el correcto funcionamiento del modelo implementado en la simulación, así como la modulación del PWM de disparo de los transistores Mosfet de potencia.

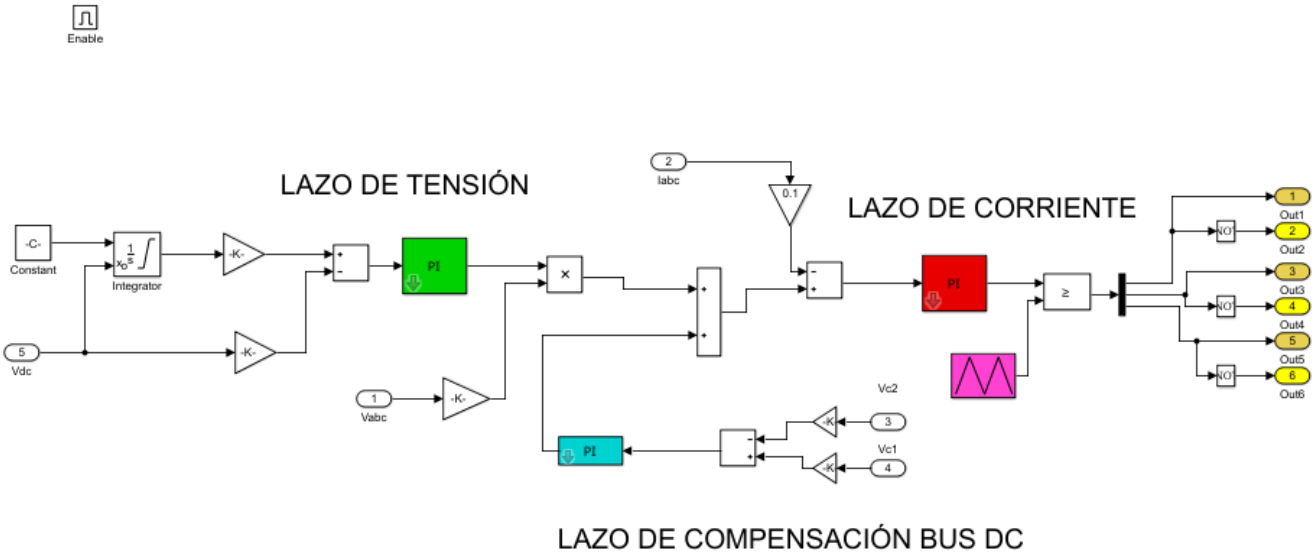


Ilustración 54.Control para el modelo rectificador Vienna tipo T

7.3 Adquisición de datos

Realizada la simulación, debemos adquirir la información que nos permita analizar si el rectificador de Vienna está funcionando conforme a lo resultados esperados, para ello se implementan bloques de adquisición con los que poder obtener toda la información necesaria para su análisis y comprensión del sistema rectificador de Vienna tipo T.

A través de los monitores implementados podemos conocer, las medidas de tensión y corriente del sistema así como sus potencias y factores de interés tales como el factor de potencia o la distorsión armónica total THD que estamos obteniendo.

Otro de los factores que pueden ser de relevante importancia en el modelo real son las intensidades de salida y entrada ya que la placa TIDM-1000 tiene un límite admisible máximo, así como las tensiones que van a tener que soportar los semiconductores en funcionamiento nominal.

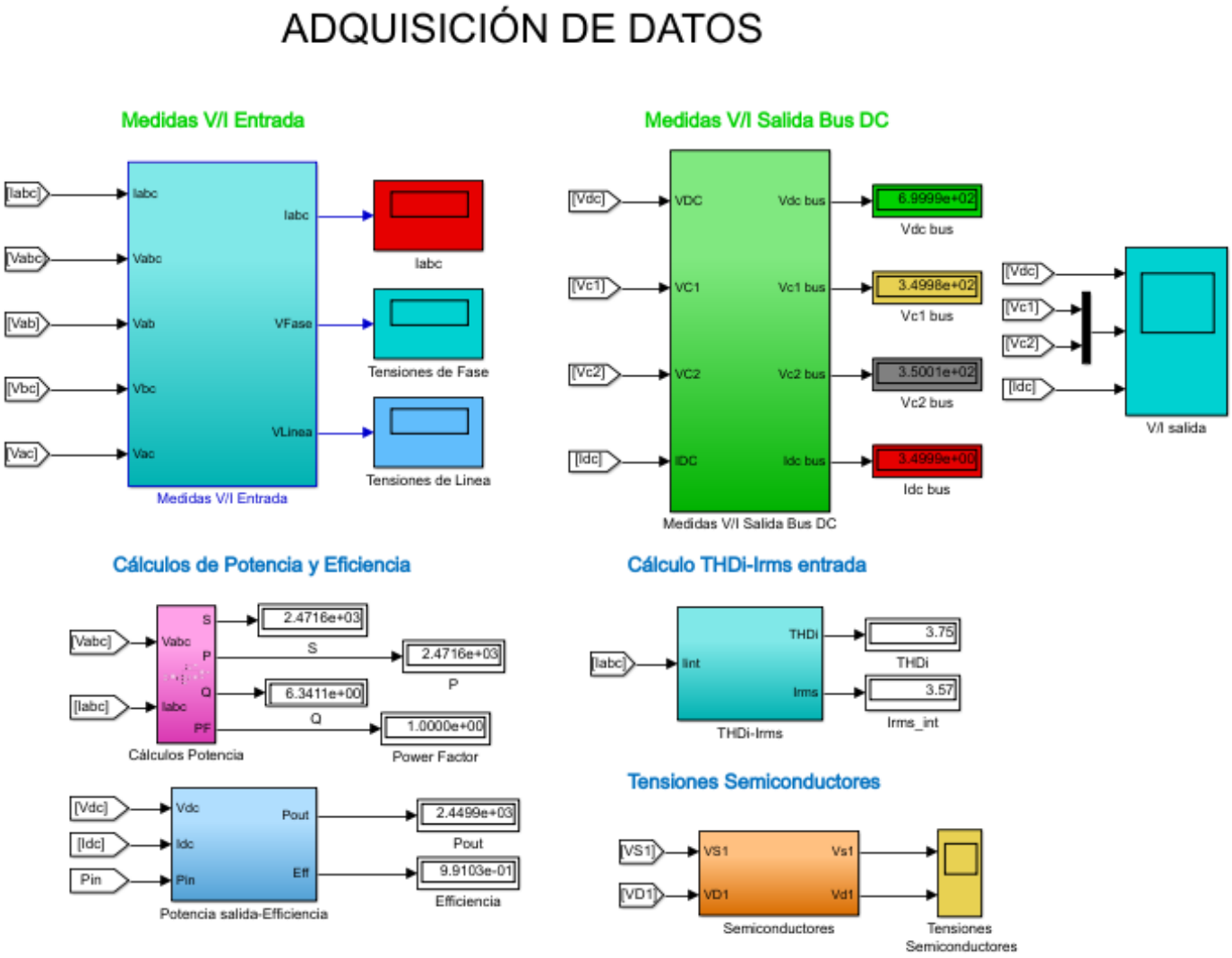


Ilustración 55.Adquisición de datos del rectificador de Vienna tipo T

7.4 Configuración simulación

En el desarrollo de este apartado se muestra como inicializar los parámetros para realizar una simulación con las características de diseño que necesitemos.

Primero vamos a ir al bloque del modelo del rectificador de Vienna tipo T, aquí podemos parametrizar tensiones así como los elementos de los que se compone el modelo.

Empezamos por introducir las tensiones de fase a la entrada del sistema.

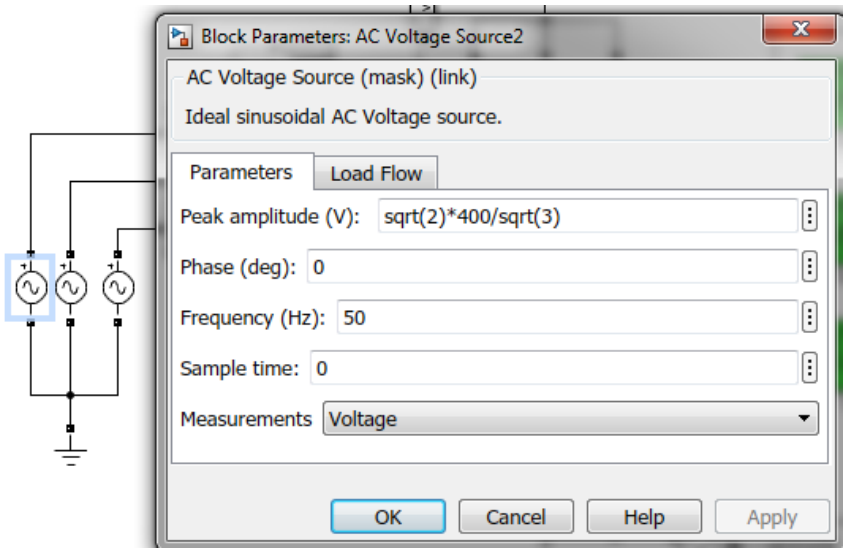


Ilustración 56.Configuración de tensión a la entrada del rectificador de Vienna

Ahora introduciremos los valores del inductor de filtrado primario con las características de nuestro diseño.

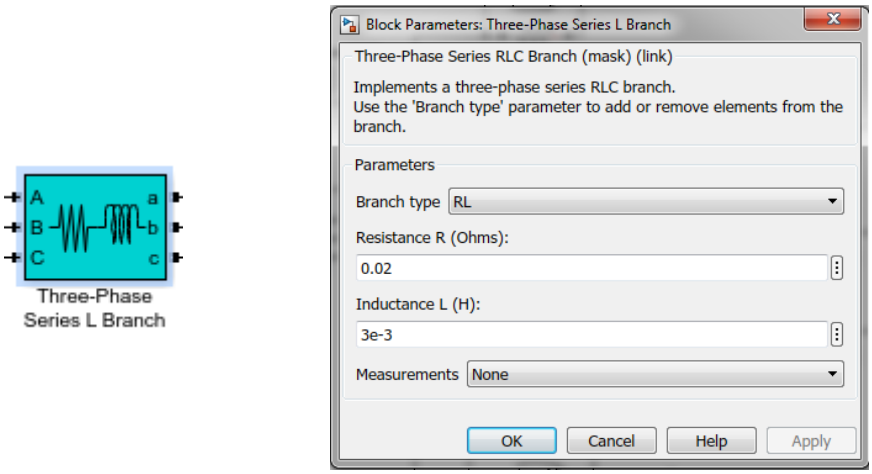


Ilustración 57.Configuración inductores entrada.

En el siguiente paso realizaremos la configuración de los elementos a la salida tales como la malla de resistencias, los condensadores del bus de salida y la carga que vamos a tener a la salida del rectificador.

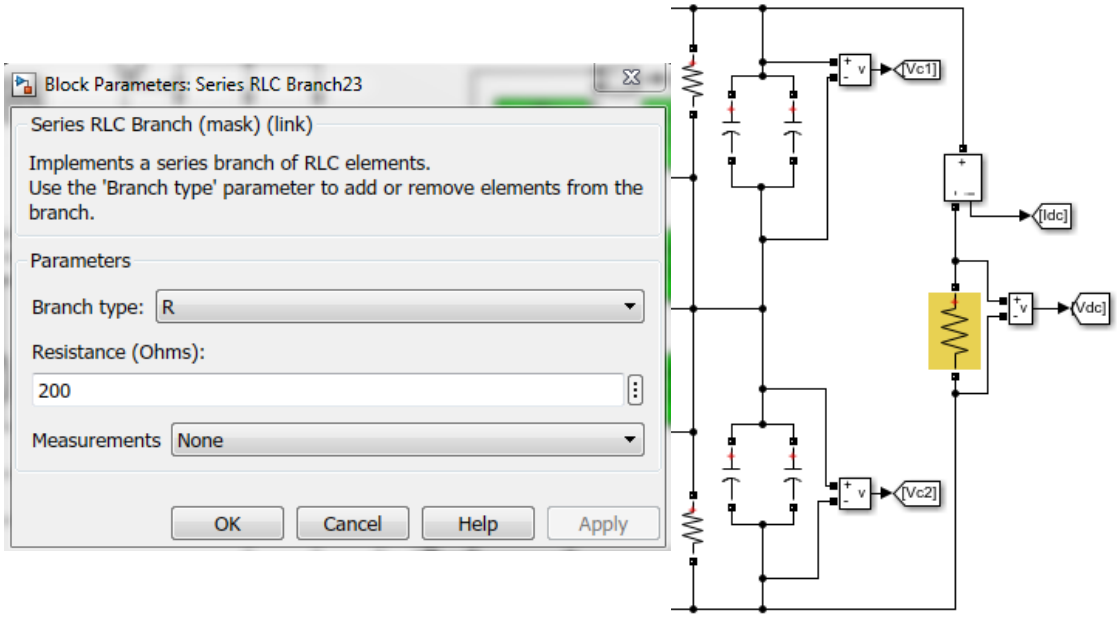


Ilustración 58.Configuracion componentes de salida

Por último en esta etapa seleccionaremos el tiempo de muestreo para realizar la simulación.

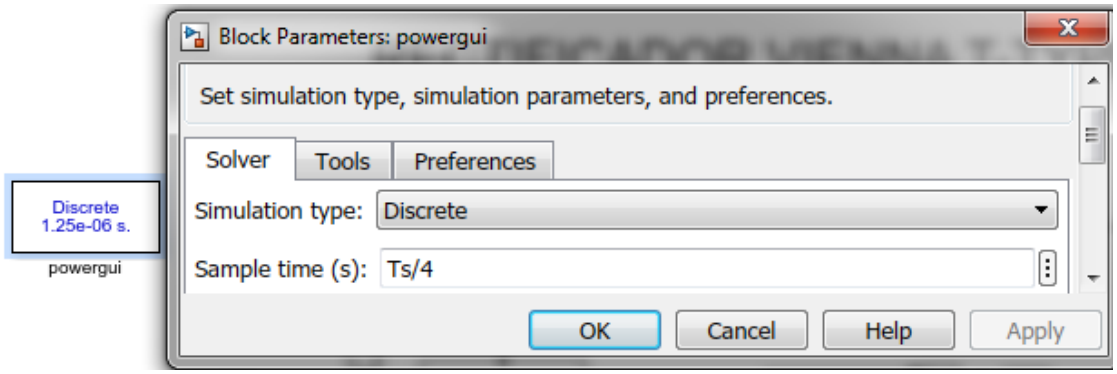


Ilustración 59.Configuración tiempo de muestreo

Ahora vamos a proceder a elegir la frecuencia de conmutación para el disparo de los transistores Mosfet así como la referencia de tensión que queremos que el sistema siga a la salida del bus de continua.

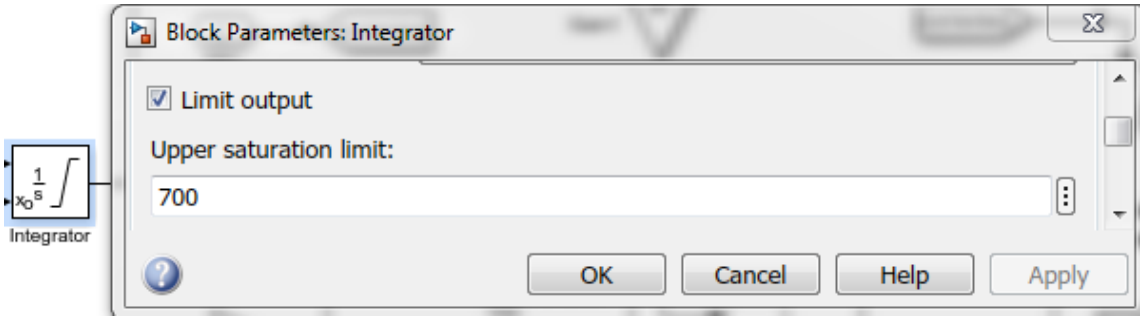


Ilustración 60.Configuracion tensión de salida del bus de continua

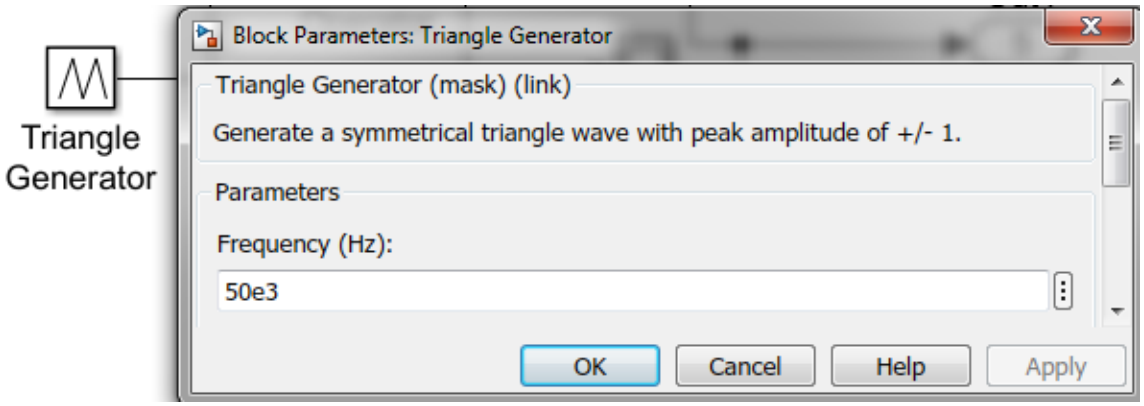


Ilustración 61.Configuración frecuencia de conmutación

Si cambiamos la frecuencia de muestreo deberemos incluirla también en los reguladores PI para su correcto funcionamiento, así como el ajuste que se pretenda realizar.

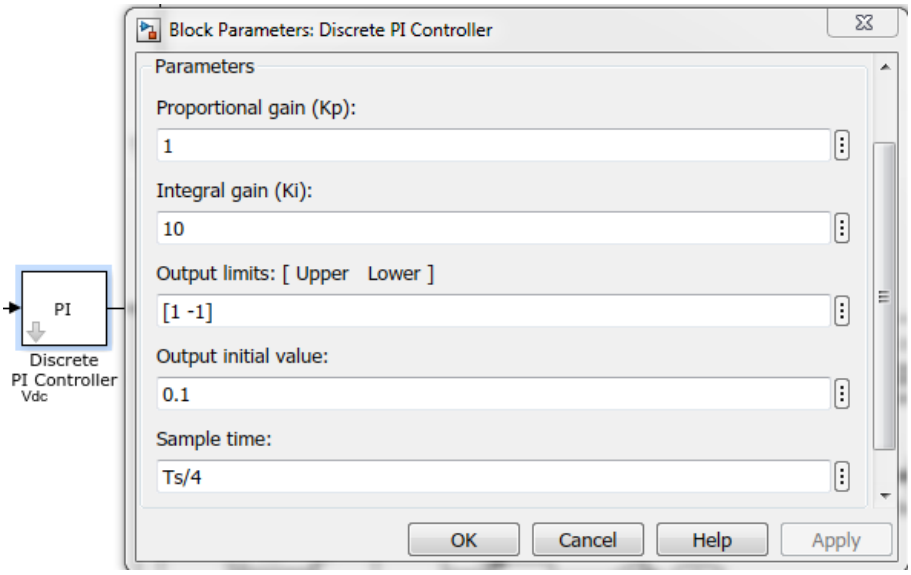


Ilustración 62.Configuracion reguladores PI

7.5 Simulación con parámetros de red

En este apartado se van a realizar las simulaciones para la red europea con tensión trifásica 400 V / 50 Hz y la red americana 208 V/ 60 Hz.

Mediante las simulaciones realizadas verificamos el funcionamiento con ambas entradas de tensión trifásicas y comprobamos como se cumplen las características de funcionamiento del rectificador de Vienna implementado en la placa TIDM-1000.

7.5.1 Red europea 400 V / 50 Hz

A lo largo de este apartado se podrá ver los resultados obtenidos para una tensión trifásica de la red eléctrica empleada en la red europea 400 V/ 50 Hz, conectando a la salida una carga de 2450 W, carga puramente resistiva de 200 Ω, VDC de referencia 700 V.

Lo primero que vamos a medir son las tensiones de fase de entrada y tensiones de línea y verificar que son las que corresponden a una red europea.

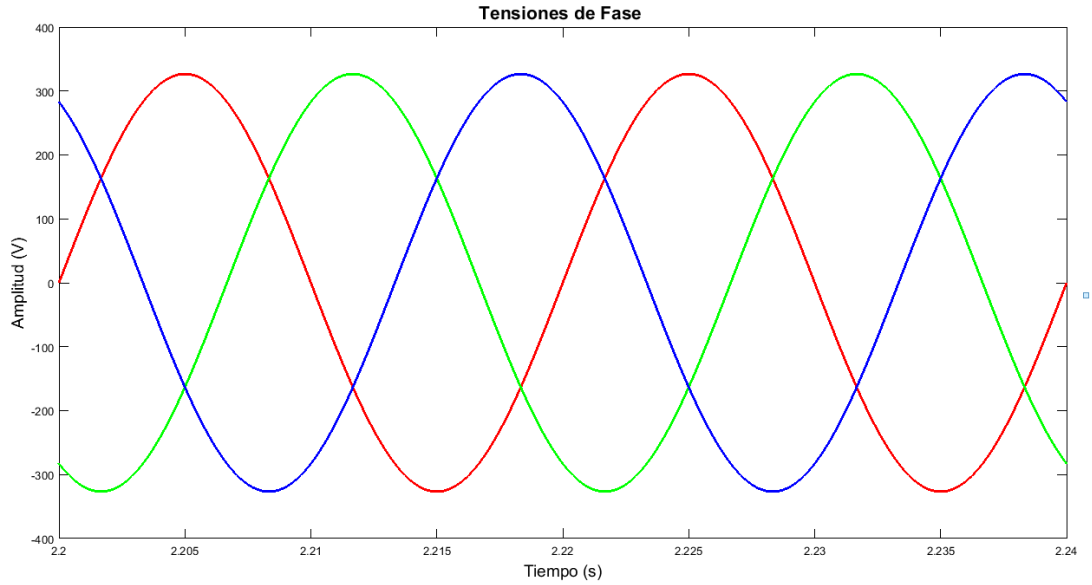


Ilustración 63.Tensiones de fase 400 V/ 50 Hz 2450 W

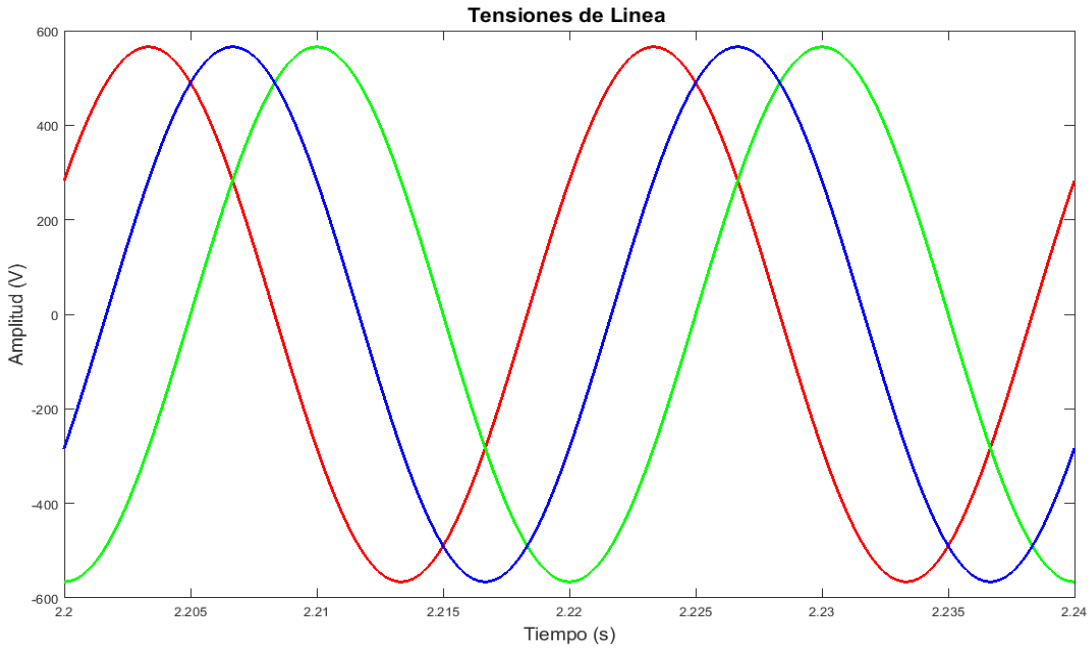


Ilustración 64. Tensiones de línea 400 V/ 50 Hz 2450 W

En las siguientes medidas vamos a ver las intensidades de entrada con las que podemos verificar que no se llega al límite en el rectificador Vienna implementado en TIDM-1000, el cual su valor máximo de entrada I_{rms} son 4 A.

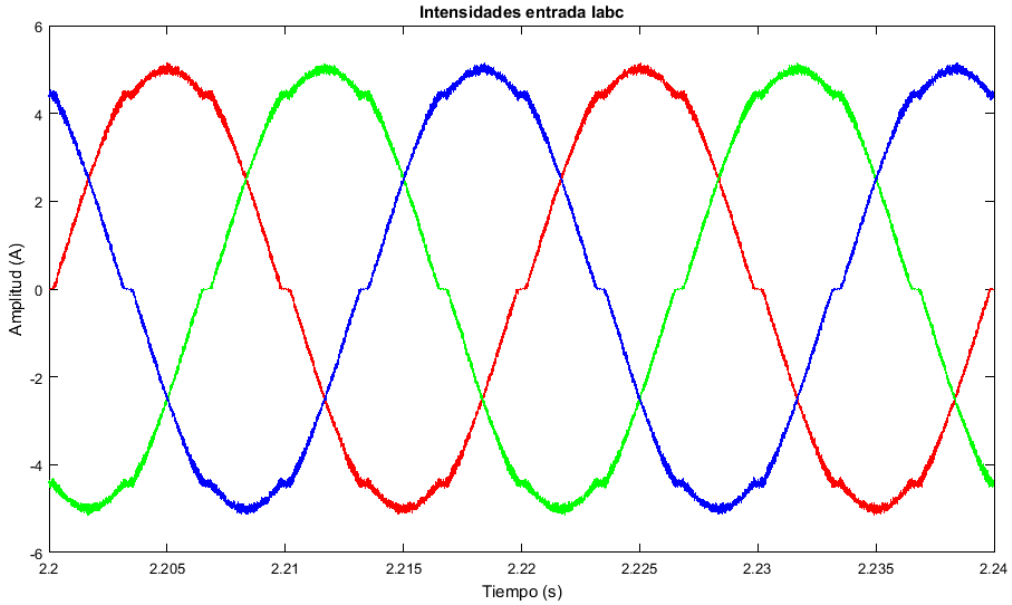


Ilustración 65.Intensidades de entrada del rectificador Vienna 400 V/ 50Hz 2450 W

Con los datos obtenidos vemos como la intensidad a plena carga es casi sinusoidal con una pequeña incorporación de armónicos, mediante la simulación comprobamos que el valor que obtenemos I_{rms} para este caso con carga 2450 W es de I_{rms}=3,57 A, con lo que estamos todavía en el rango admisible por el rectificador de Vienna para la placa TIDM-1000.

Ahora vamos a ver el espectro de la señal que se obtiene durante el tiempo que se realiza la simulación con lo vamos a poder ver que armónicos que están adheridos a la intensidad de entrada.

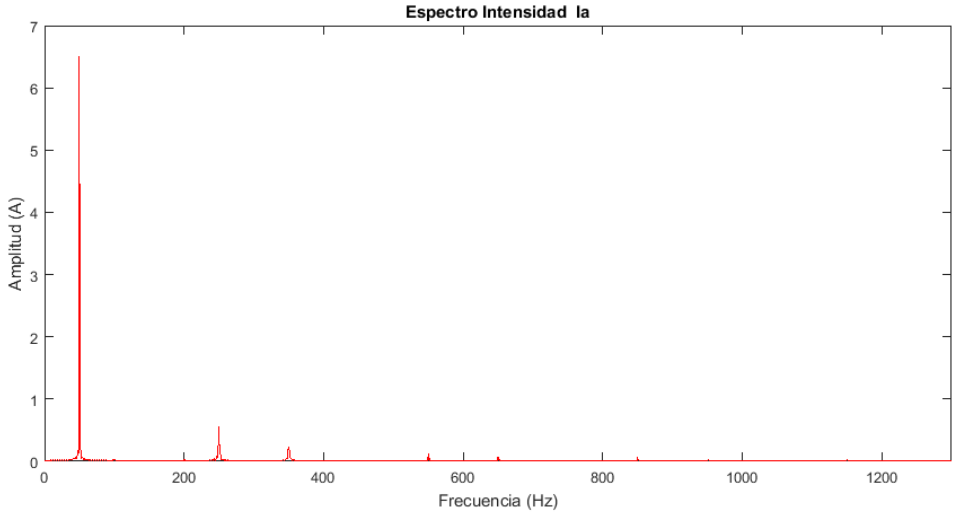


Ilustración 66.Espectro en frecuencia de la Intensidad de entrada Ia 400 V/ 50Hz 2450 W

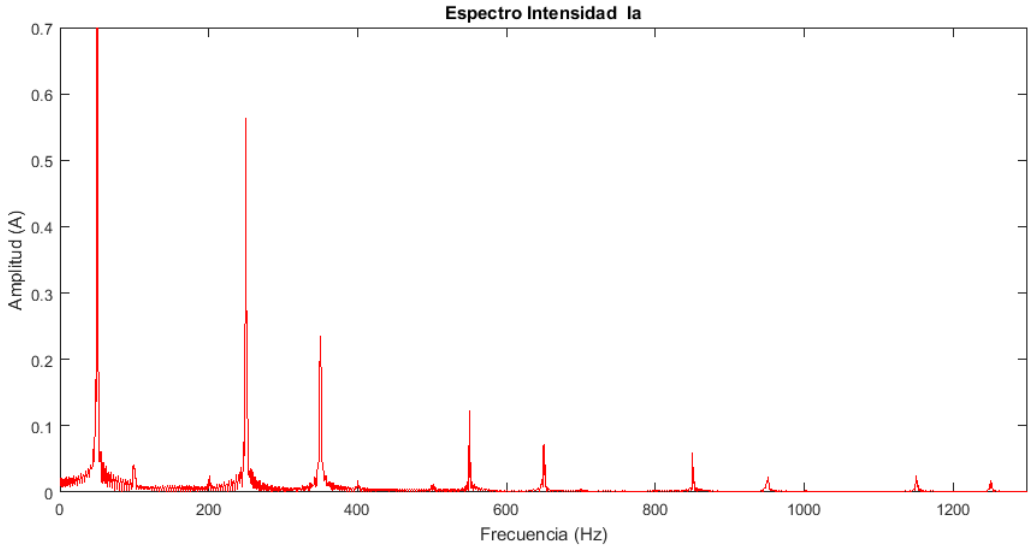


Ilustración 67.Espectro en frecuencia de la Intensidad de entrada Ia 400 V/ 50Hz 2450 W (2)

Mediante la adquisición del espectro en frecuencia de una de las intensidades de entrada podemos ver de qué tipo son los armónicos obtenidos y de qué forma van a afectar a nuestro sistema.

Podemos observar como solamente aparecen los armónicos: 5,7,11,13,17,19,23,25.Esto se debe a que nuestro sistema esta balanceado, si no lo estuviera tendríamos también perdidas por los armónicos 3,9,15,21.

La distorsión armónica total en las intensidades, que obtenemos en esta simulación es de THDi= 3,745 %.

Verificada la entrada del sistema, nos disponemos a adquirir las señales que estamos obteniendo a la salida y si cumplen con los resultados esperados.

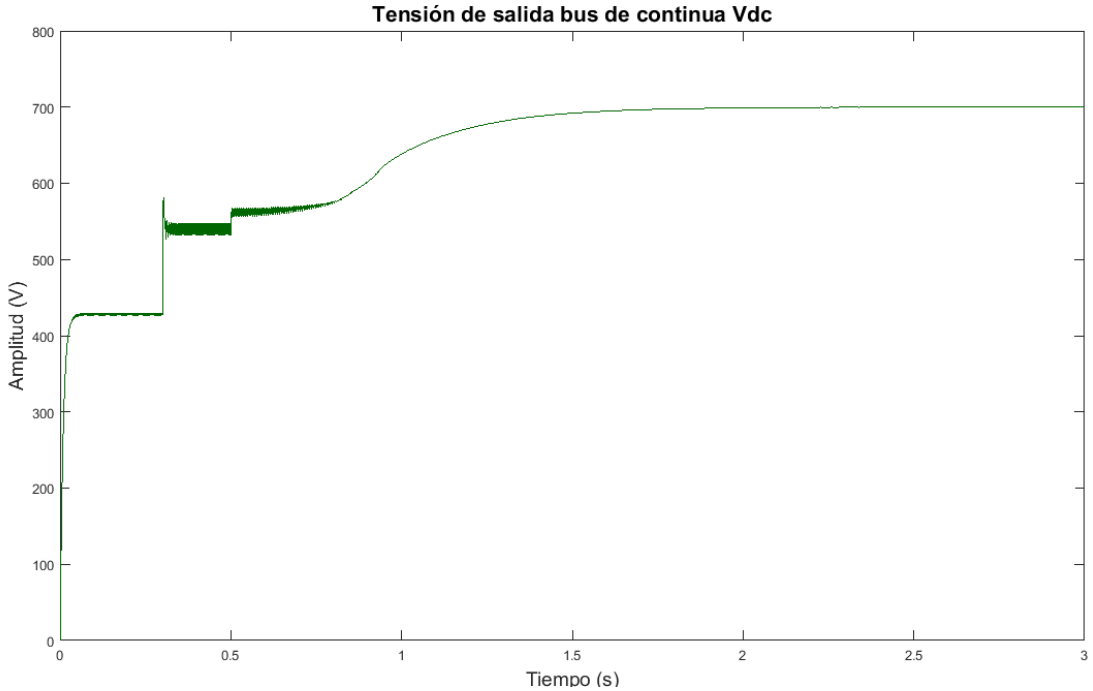


Ilustración 68.Tensión de salida del bus continua Vdc 400 V/ 50Hz 2450 W

En la adquisición de la tensión de salida del bus de continua podemos ver dos fases, la primera fase hasta 0,5 s , donde se realiza la precarga de los condensador y la segunda fase a partir de 0,5 s, donde activamos el control del rectificador de Vienna.Se observa como el tiempo de establecimiento de la referencia es inferior a 1,5 s y que alcanzamos la referencia de 700 V ,para esta simulacion con 2450 W, la tension de salida Vdc=700 V.

Ahora vamos a ver el rizado de la señal de obtenida, ya que para el diseño de esta simulación no deberiamos sobrepasar V_{rizado}>14 V.

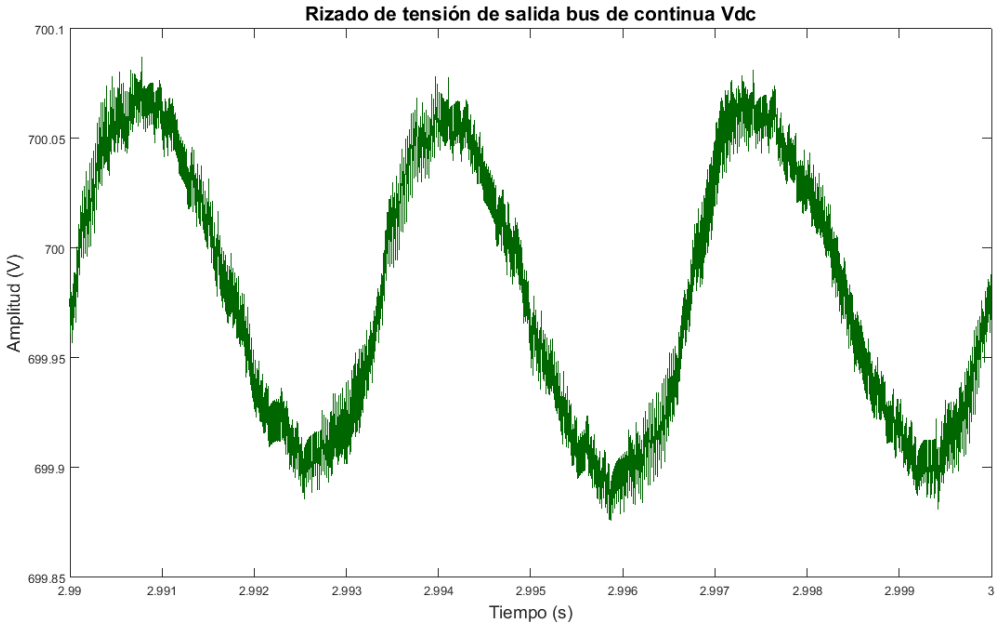


Ilustración 69.Tensión de rizado a la salida del bus continua Vrizado 400 V/ 50Hz 2450 W

Se puede ver como la tension de rizado que se obtiene es muy inferior al máximo que hemos diseñado con lo cual el diseño del condensador es valido, alcanzando asi una tension de rizado V_{rizado}<0,2 V.

Ahora pasamos a analizar la intensidad de salida si se encuentra dentro del rango maximo persisible por el rectificador de Vienna implementado en la placa TIDM-1000, ya que nos informa de que el valor maximo de la intensidad a la salida sera de I_{out}< 5 A.

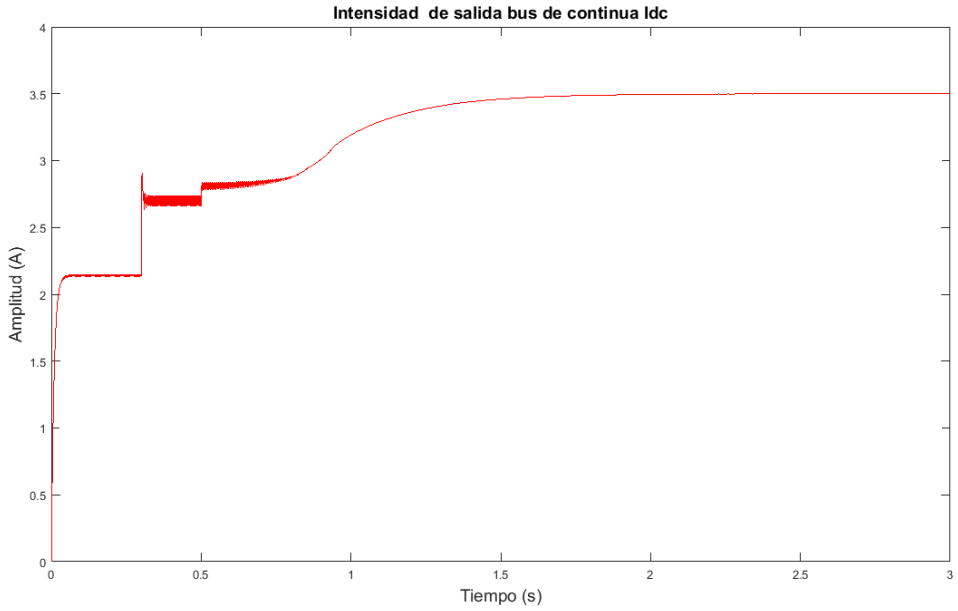


Ilustración 70.Intensidad a la salida del bus de continua Idc 400 V/ 50 Hz 2450 W

Podemos observar como la intensidad de salida es inferior a los 5 A, I_{out}=3,5 A, con lo cual el rectificador podrá operar con esta potencia y niveles de intensidad a la salida de la placa TIDM-1000.

Es interesante ver el rizado a la salida del rectificador, ya que veremos entre que valores de tensión continua vamos a operar.

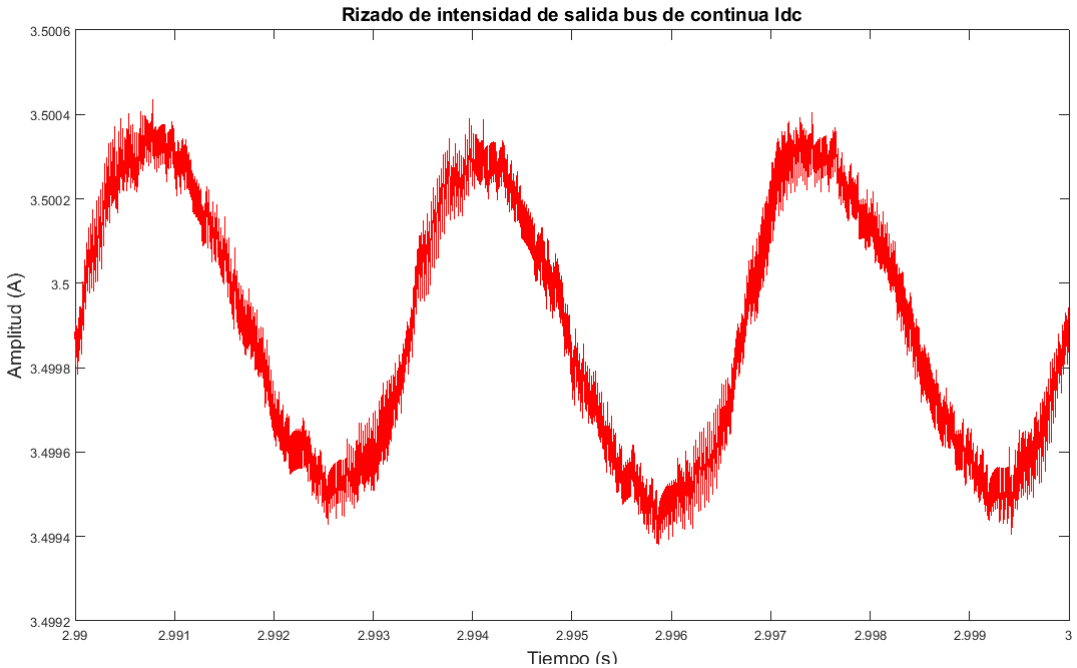


Ilustración 71. Intensidad de rizado a la salida del bus continua Vrizado 400 V/ 50Hz 2450 W

Se obtiene una intensidad de rizado mínima de aproximadamente I_{rizado}=1 mA, con lo que la señal obtenida se considera una corriente continua.

Por ultimo en esta simulación observaremos si el lazo de control de compensación en los condensadores está realizando su función.

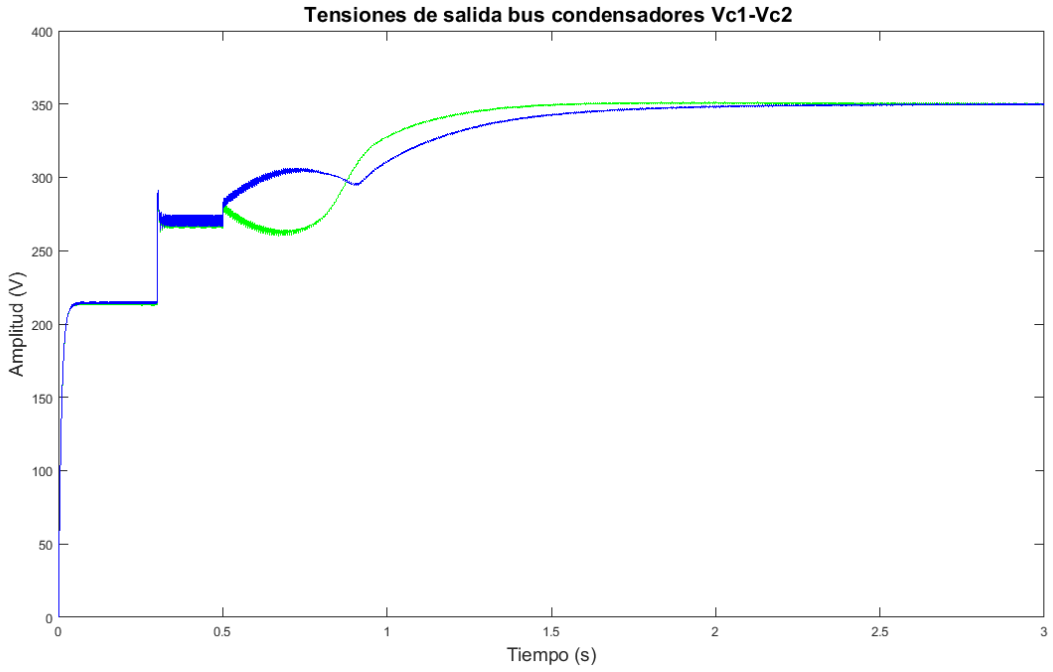


Ilustración 72.Tensión de salida en los condensadores del bus continua Vdc 400 V/ 50Hz 2450 W

7.5.2 Red americana 208 V / 60 Hz

En este apartado se pueden observar los resultados obtenidos para una tensión trifásica de la red eléctrica empleada en la red americana 208 V/ 60 Hz, conectando a la salida una carga de 1199,5 W, carga puramente resistiva de 300 Ω, VDC de referencia 600 V.

Lo primero que vamos a medir son las tensiones de fase de entrada y tensiones de línea y verificar que son las que corresponden a una red americana.

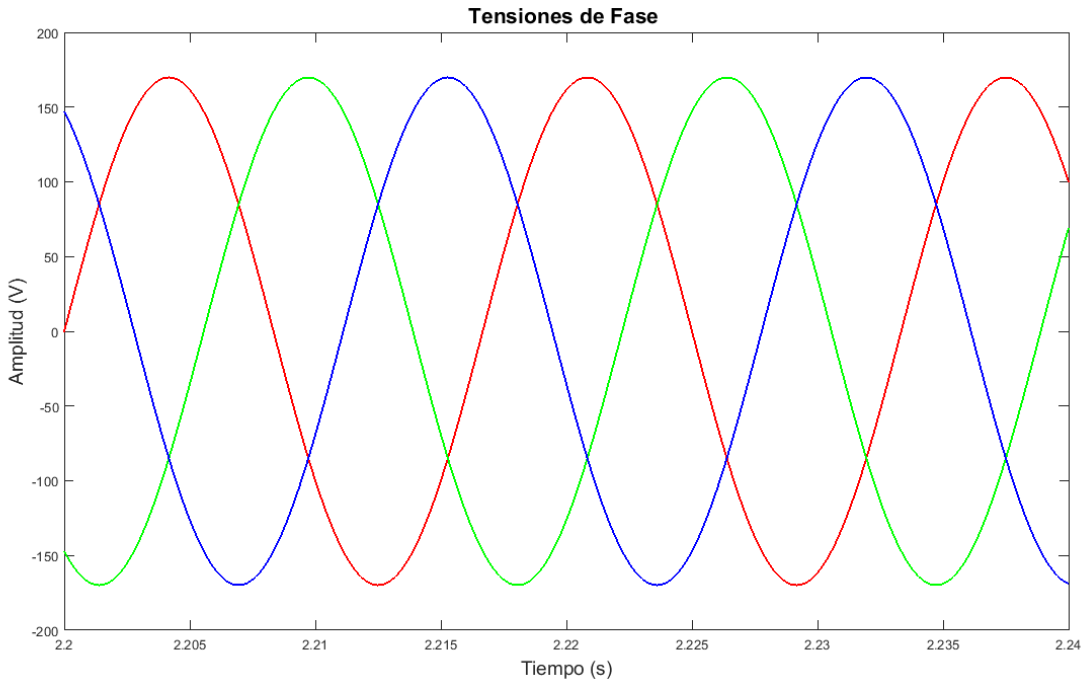


Ilustración 73.Tensiones de fase 208 V/ 60 Hz 1199,5 W

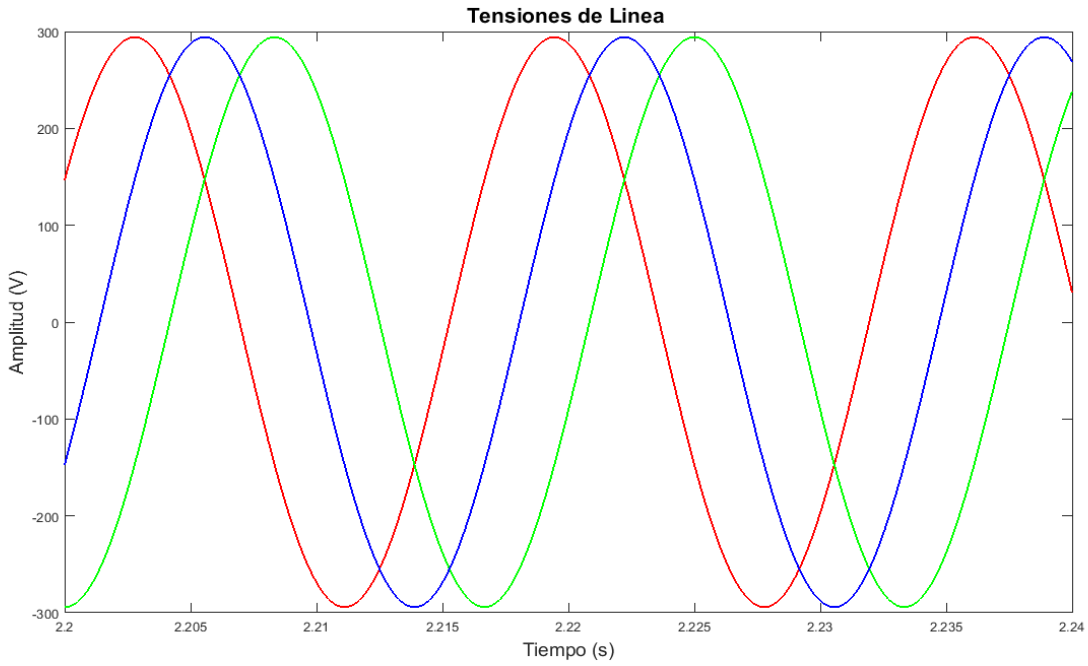


Ilustración 74.Tensiones de línea 208 V/ 60 Hz 1199,5 W

En la adquisición de las siguientes medidas adquirimos las intensidades de entrada con las que podemos verificar que no se llega al límite en el rectificador Vienna implementado en TIDM-1000, el cual su valor máximo de entrada I_{rms} son 4 A.

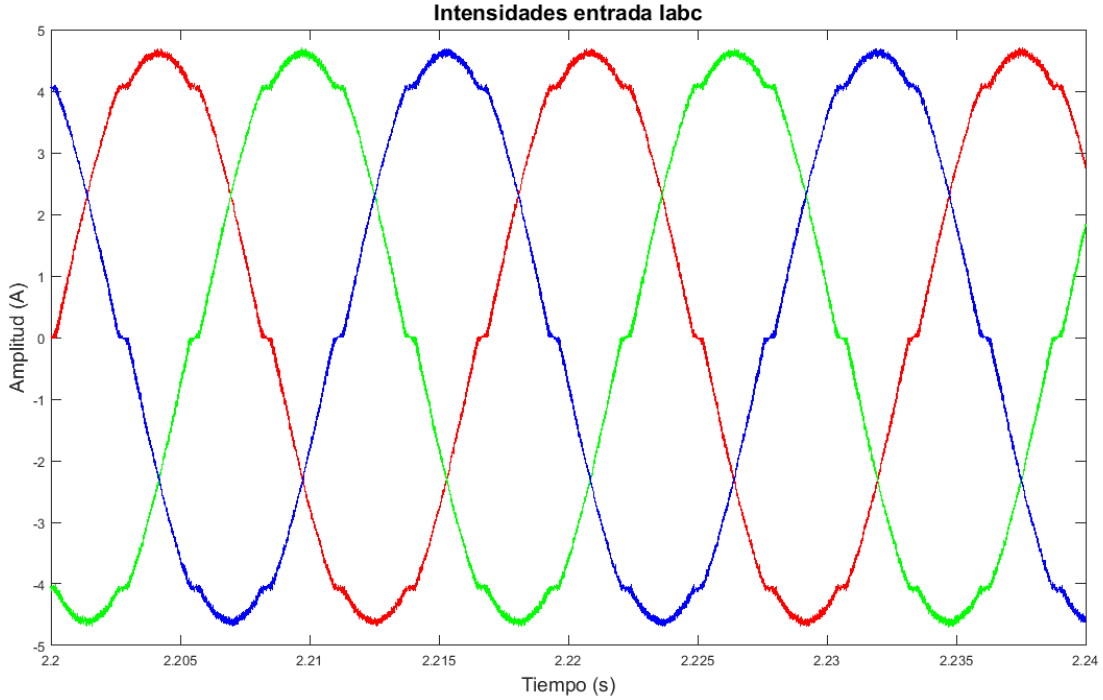


Ilustración 75.Intensidades de entrada del rectificador Vienna 208 V/ 60Hz 1199,5 W

En la anterior grafica podemos ver como la intensidad con una carga de 1199,5 W es casi sinusoidal con una pequeña incorporación de armónicos, mediante la simulación comprobamos que el valor que obtenemos de intensidad eficaz es de I_{rms}=3,382 A, con lo que estamos todavía en el rango admisible por el rectificador de Vienna para la placa TIDM-1000 con una entrada trifásica de 208 V/ 60 Hz.

En las siguientes ilustraciones vamos a ver el espectro en frecuencia para la señal 208 V/ 60 Hz, ya que estamos trabajando a una frecuencia de 60 Hz veremos otro tipo de armónicos acoplados a la intensidad de entrada.

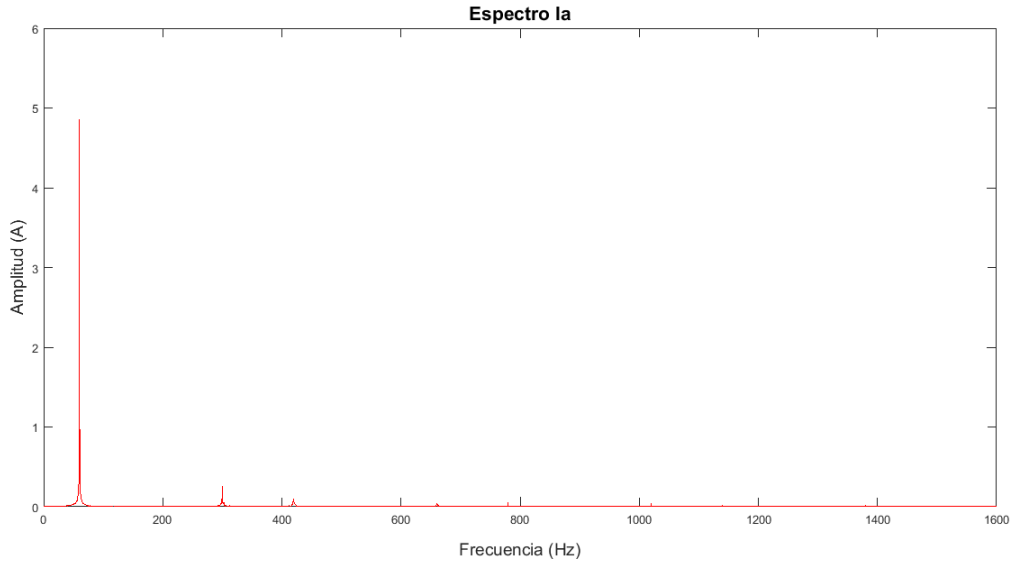


Ilustración 76. Espectro en frecuencia de la Intensidad de entrada Ia 208 V/ 60Hz 1199,5 W

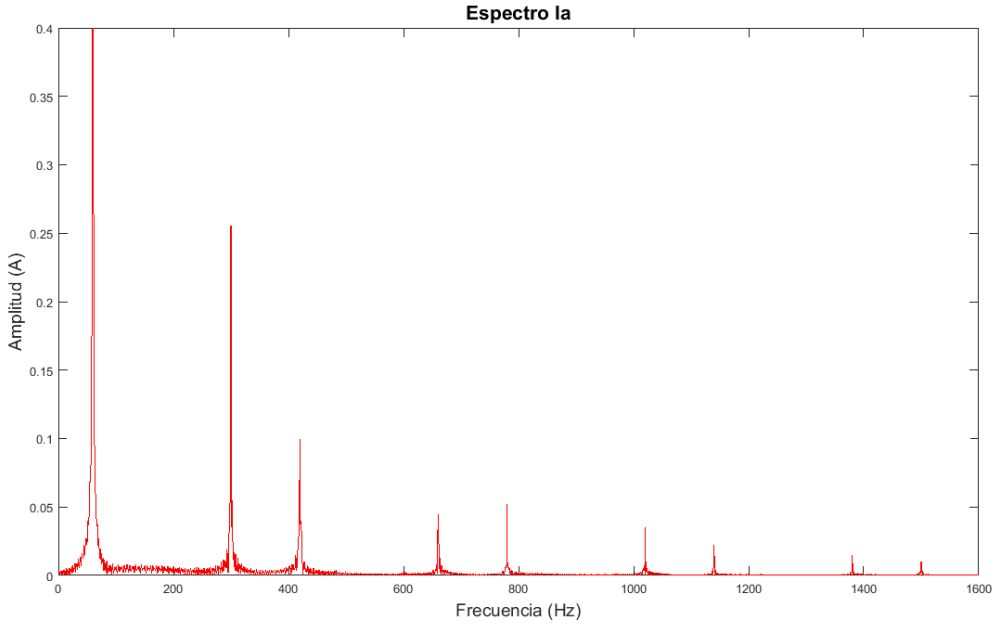


Ilustración 77. Espectro en frecuencia de la Intensidad de entrada Ia 208 V/ 60Hz 1199,5 W (2)

Con la obtención del espectro en frecuencia podemos ver como en este caso de 208 V/ 60 Hz los armónicos que obtenemos son: 5,7,11,13,17,19,23,25, pero la diferencia que tenemos con el anterior caso, es que ahora son múltiplos de 60 Hz con lo que llegamos hasta 1500 Hz.

La distorsión armónica total que obtenemos en las intensidades de entrada en este caso es THDi=3,592 %.

Realizado el análisis de las variables de entrada, pasamos a ver los resultados que hemos obtenido en la salida del sistema rectificador de Vienna.

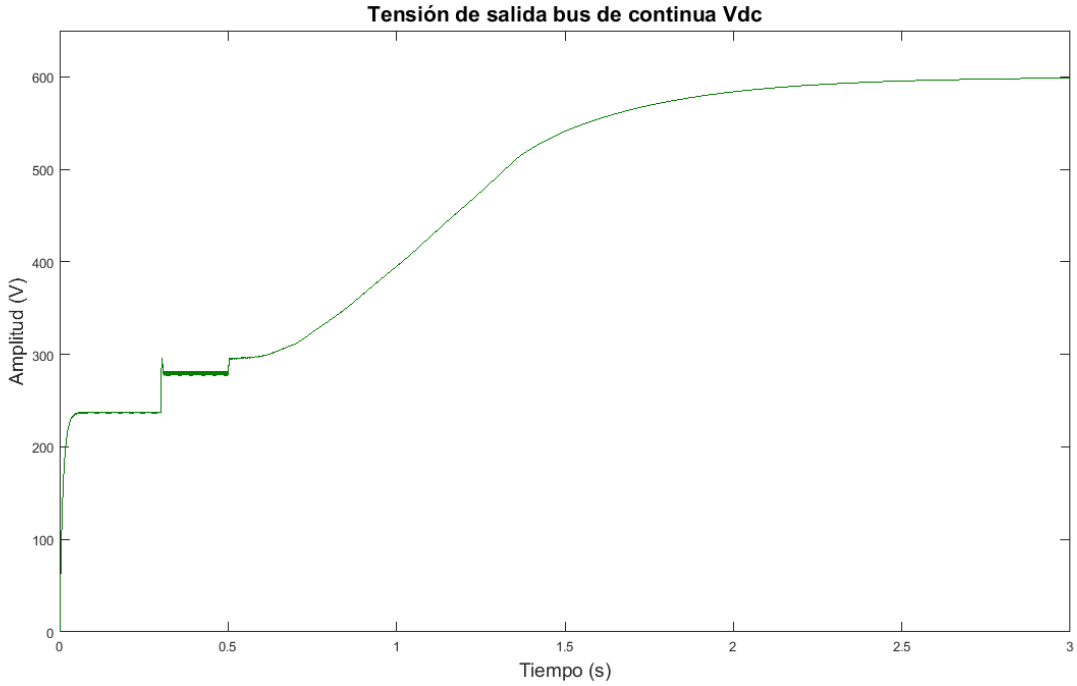


Ilustración 78. Tensión de salida del bus continua Vdc 208 V/ 60Hz 1199,5 W

Con esta configuracion se observa como el tiempo de establecimiento de la referencia varia un poco con respectro el caso de 400 V/ 50 Hz ahora nos desplazamos a 2 s, vemos como se alcanza la tensión de referencia de 600 V en este caso con una carga de1199,5W, Vdc=599,87V.

Ahora vamos a ver el rizado de la señal de obtenida, ya que para el diseño de esta simulación no deberiamos sobrepasar V_{rizado}>12 V.

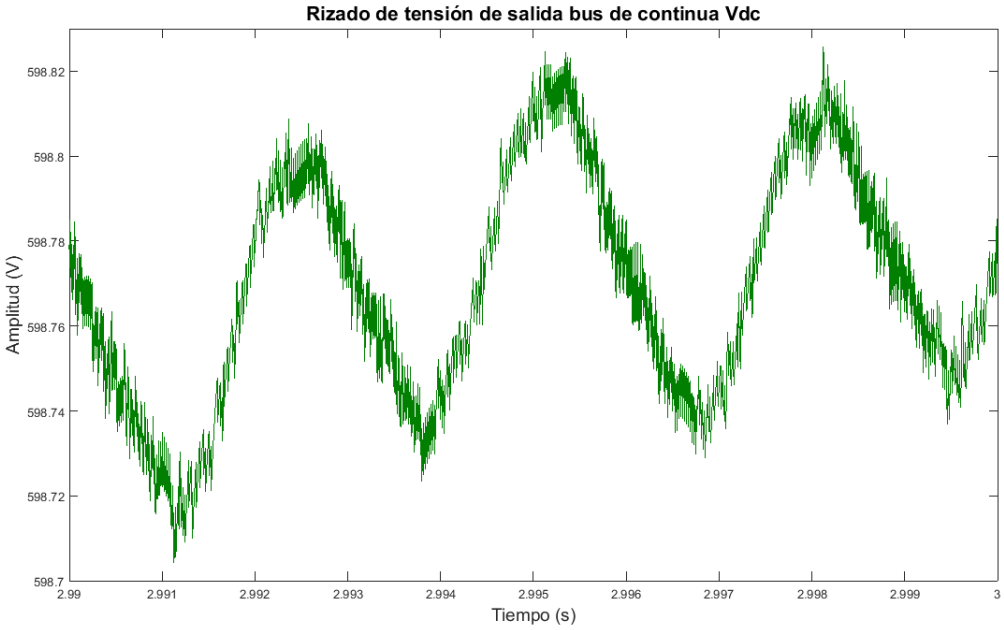


Ilustración 79. Tensión de rizado a la salida del bus continua Vrizado 208 V/ 60Hz 1199,5 W

Ahora puede ver como la tension de rizado que se obtenemos , es muy inferior al limite que hemos diseñado, con lo cual el diseño del condensador es el correcto,en esta simulación alcanzamos una tension de rizado V_{rizado}<0,15 V.

Ahora analizamos la intensidad que obtenemos a la salida y comprobar que encuentra dentro del rango maximo permitido por el rectificador de Vienna implementado en la placa TIDM-1000, ya que el valor maximo de la intensidad en la salida sera de I_{out}< 5 A.

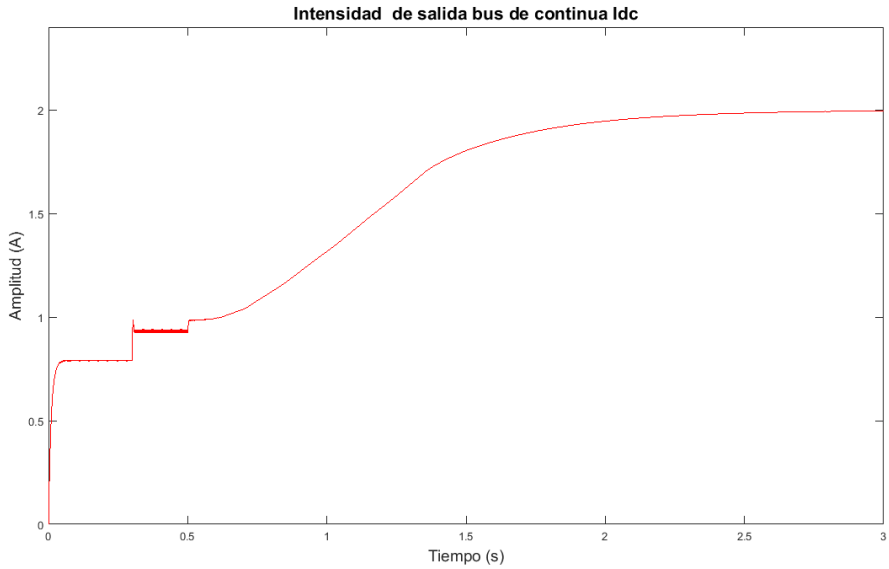


Ilustración 80. Intensidad a la salida del bus de continua Idc 208 V/ 60 Hz 1199,5 W

Podemos ver como la intensidad de salida es menor de la mitad de la permitida por el rectificador de Vienna implementado en la placa TIDM-1000, por lo tanto podremos operar en este rango de potencias e intensidades, para esta simulacion la intensidad de salida es Iout= 1,9996 A.

A continuación vamos a ver el rizado que tenemos en la intensidad, para comprobar que obtenemos un intensidad continua o muy aproximada.

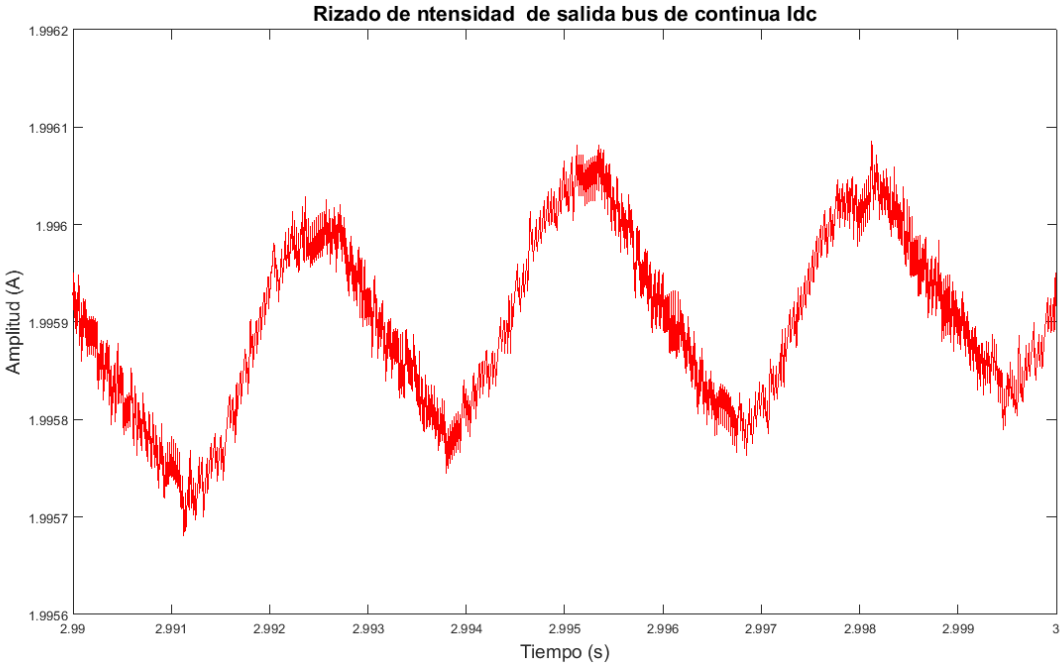


Ilustración 81. Rizado en la intensidad a la salida del bus de continua Idc 208 V/ 60 Hz 1199,5 W

Pode.En la siguiente ilustración comprobaremos que se produce el balance de tensión en los condensadores a la salida del rectificador de Vienna.

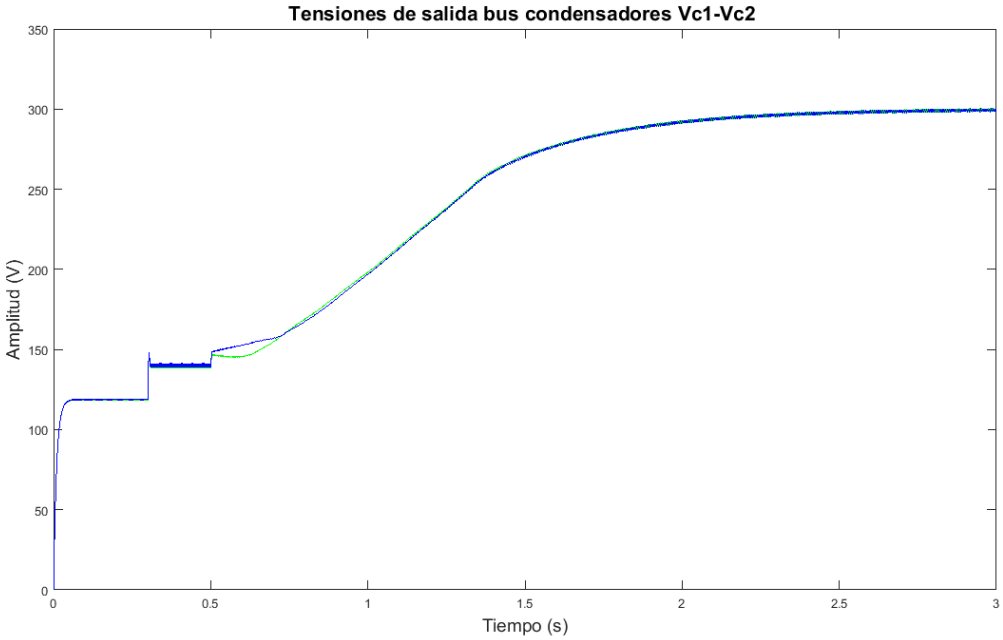


Ilustración 82. Tensión de salida en los condensadores del bus continua Vdc 208 V/ 60Hz 1199,5 W

7.6 Simulación con parámetros de máquina sincrónica Bornay 3 kW /120 V

En este apartado se van a realizar la simulaciones referidas al funcionamiento de la maquina sincronica Bornay, con la que simulamos podemos simular la salida del sistema aerogeneror.

Las simulaciones se realizan con resultados en funcionamiento nominal proporcionado por el ensayo realizado de dicha máquina [14].

Se documentan dos simulaciones de dichos resultados la primera con un régimen nominal de 120 V / 50 Hz, en la que fijamos una tensión de referencia a alcanzar de 600 V, con el fin de posteriormente convertir la energía obtenida mediante un inversor trifásico en parámetros de red europea 400 V/ 50 Hz.

La segunda simulación se realiza con un funcionamiento nominal de 149 V / 60 Hz, con la misma referencia de tensión continua en la salida y el mismo objetivo.

Mediante las simulaciones que se realizan podremos determinar hasta que potencias es capaz de trabajar el rectificador de Vienna implementado en la placa TIDM-1000 con la tensión de salida de referencia y parametrizar los resultados, con el fin de tener una idea más amplia de que va a suceder con la conexión del rectificador de Vienna a la máquina sincrónica.

7.6.1 Simulación 120 V / 50 Hz

La siguiente simulación se realiza con los valores obtenidos de anteriores estudios, con una tensión entrada trifasica de 120 V / 50 Hz, la carga que se coloca a la salida disipa una potencia de 599,52W, con una carga puramente resistiva de 600 Ω.

Lo primero que vamos a ver son las tensiones de fase producidas por la maquina sincronica y las tensiones de linea.

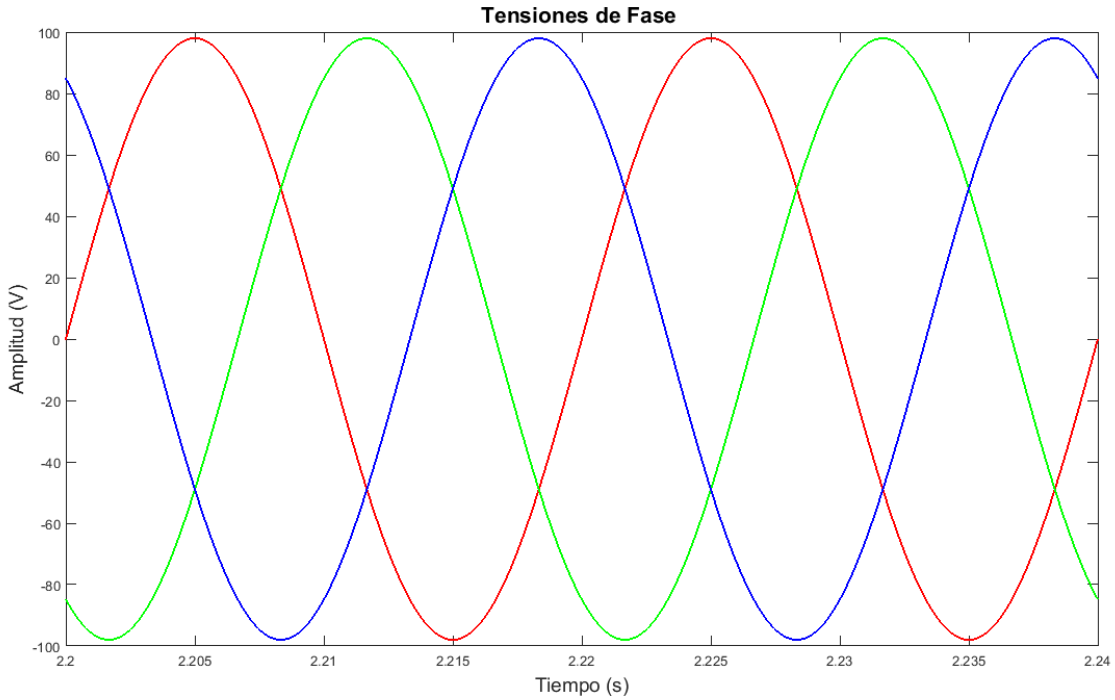


Ilustración 83. Tensiones de fase 120 V/ 50 Hz 599,52 W

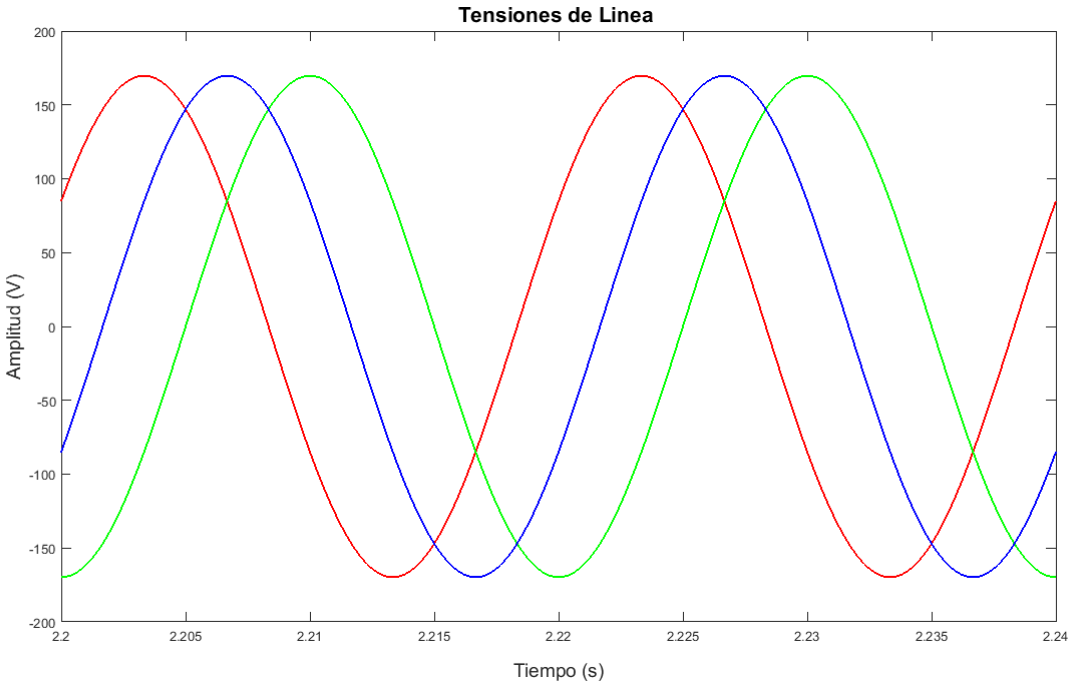


Ilustración 84. Tensiones de línea 120 V/ 50 Hz 599,52 W

Ahora vamos adquirir las intensidades de entrada, con lo podremos ver si se encuentran dentro del rango de funcionamiento del rectificador de Vienna, tambien realizaremos el espectro en frecuencia de dichas intensidades para ver el contenido de los armonicos que nos puedan aparecer en la señal.

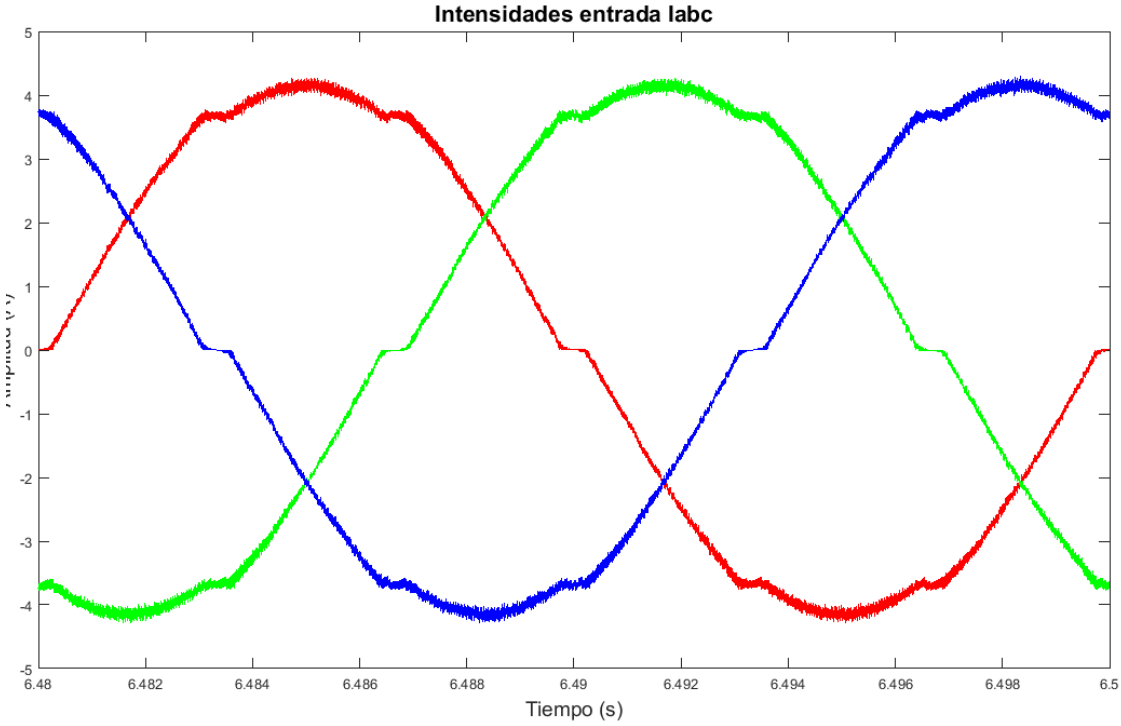


Ilustración 85. Intensidades de entrada del rectificador Vienna 120 V/ 50 Hz 599,52 W

Las señales que obtenemos vemos que entran dentro del rango que nos permite la placa TIDM-1000, para intensidad de entrada,con una intensidad eficaz I_{rms}=2,961 A , también se observan la aparición de armónicos los cuales vamos a ver su importancia en el espectro en frecuencia y su medida de la distorsión armónica total.

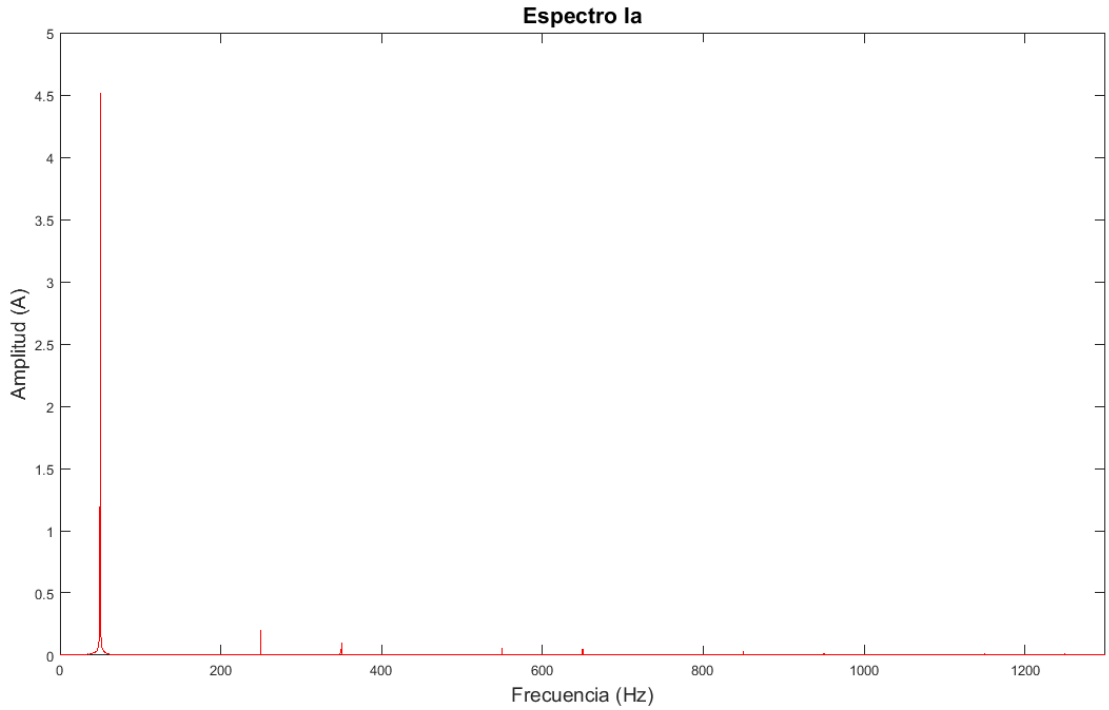


Ilustración 86. Espectro en frecuencia de la Intensidad de entrada Ia 120 V/ 50 Hz 599,52 W

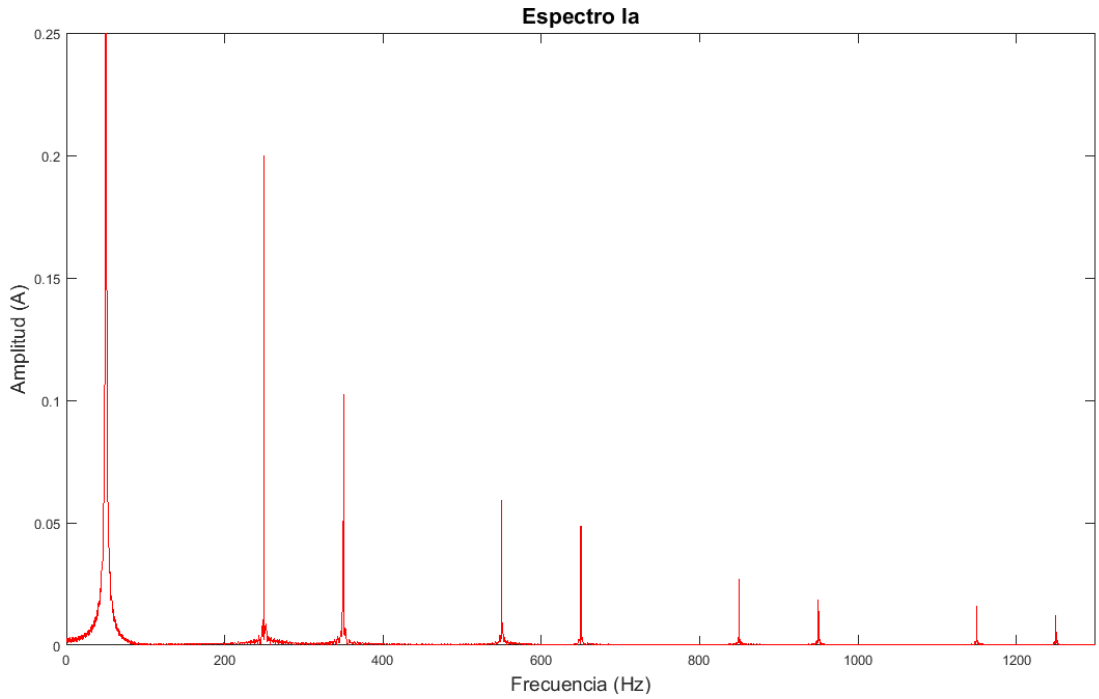


Ilustración 87. Espectro en frecuencia de la Intensidad de entrada Ia 120 V/ 50 Hz 599,52 W (2)

En la adquisición del espectro en frecuencia se puede que los armónicos que se obtienen al ser un sistema balanceado son los mismos que en casos estudiados anteriormente.

En esta simulación la distorsión armónica total que obtenemos en las intensidades de entrada es THDi= 4,407 %.

Con estos análisis en mente, vamos a ver los resultados obtenidos a la salida del sistema rectificador de Vienna y ver si se cumple con los resultados esperados.

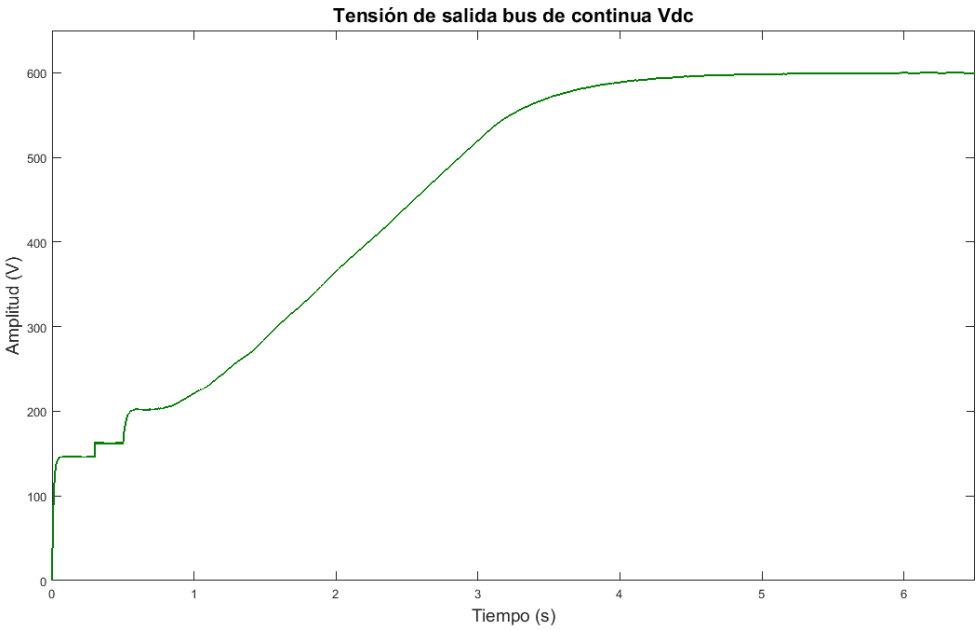


Ilustración 88. Tensión de salida del bus continúa Vdc 120 V/ 50 Hz 599,52 W

Se verifica que con la salida que nos proporciona la maquina sincrónica, podemos alcanzar la tensión continua fijada en la configuración del rectificador Vienna, con una potencia de 599,52 W y Vdc= 599,76 V.

Ahora pasamos a ver el rizado en la tensión continua generada y si está en un rango admisible.

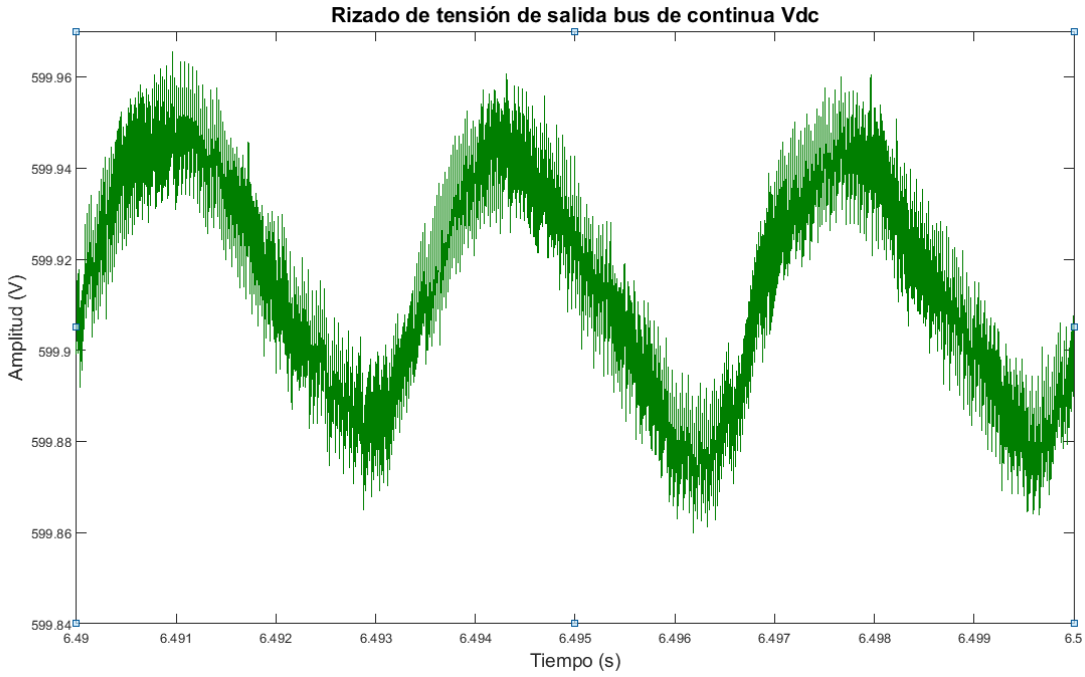


Ilustración 89. Tensión de rizado a la salida del bus continúa Vrizado 120 V/ 50 Hz 599,52 W

La tensión de rizado que obtenemos tiene una pequeña variación I_{rizado}≈0,1 V, en los niveles de tensión que estamos trabajando podemos considerar que en la salida del rectificador de Vienna la tensión obtenida es continua.

Vamos a ver la forma que tiene la intensidad en la salida del bus de continua y la variación que tiene a en la salida.

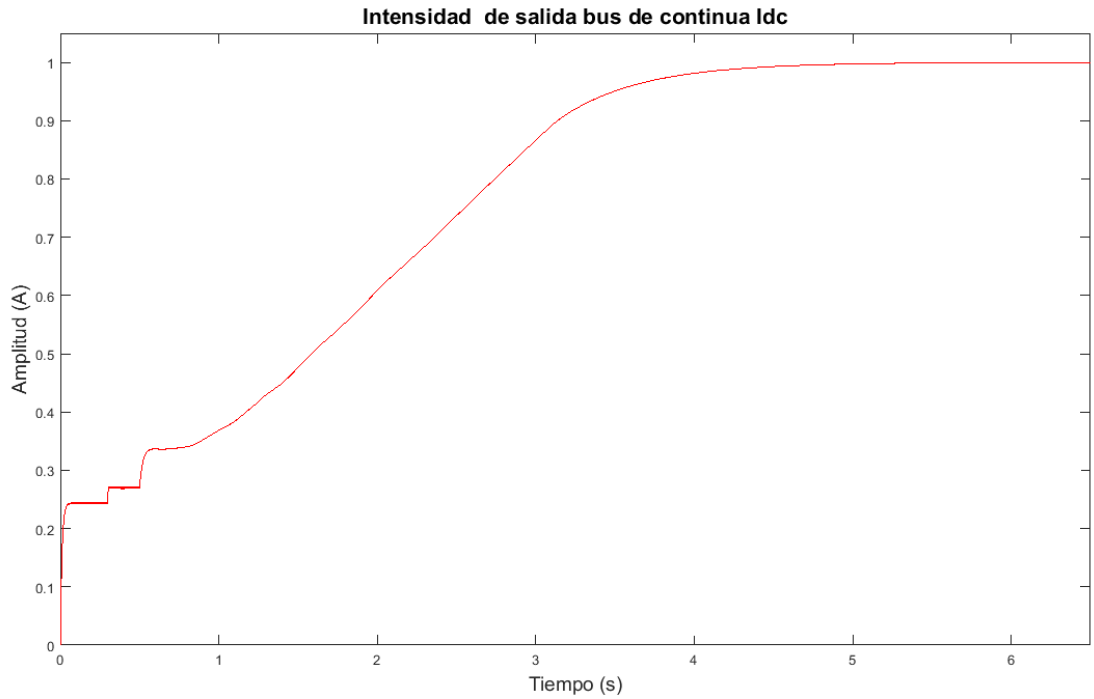


Ilustración 90. Intensidad a la salida del bus de continua Idc 120 V/ 50 Hz 599,52 W

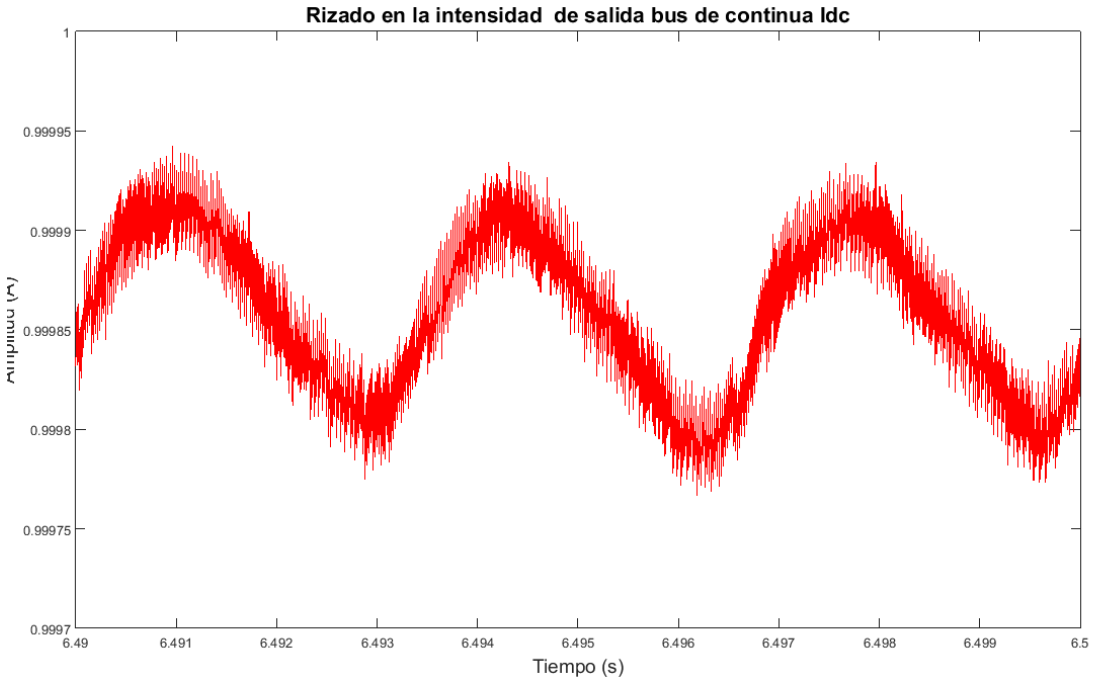


Ilustración 91. Rizado en la intensidad a la salida del bus de continua 120 V/ 50 Hz 599,52 W

Mediante la obtención de la intensidad a la salida del bus de continua, vemos que la intensidad que va a circular por la carga es I_{out}= 0,9996 A, por lo tanto estamos todavía muy lejos del límite de corriente que admite el rectificador de Vienna a la salida de la placa TIDM-1000.

En lo referente al rizado vemos que también se puede considerar una corriente continua ya que la variación es solamente de I_{rizado}≈0,12 mA.

La última adquisición la realizamos para ver si existe balance en los condensadores del bus de continua.

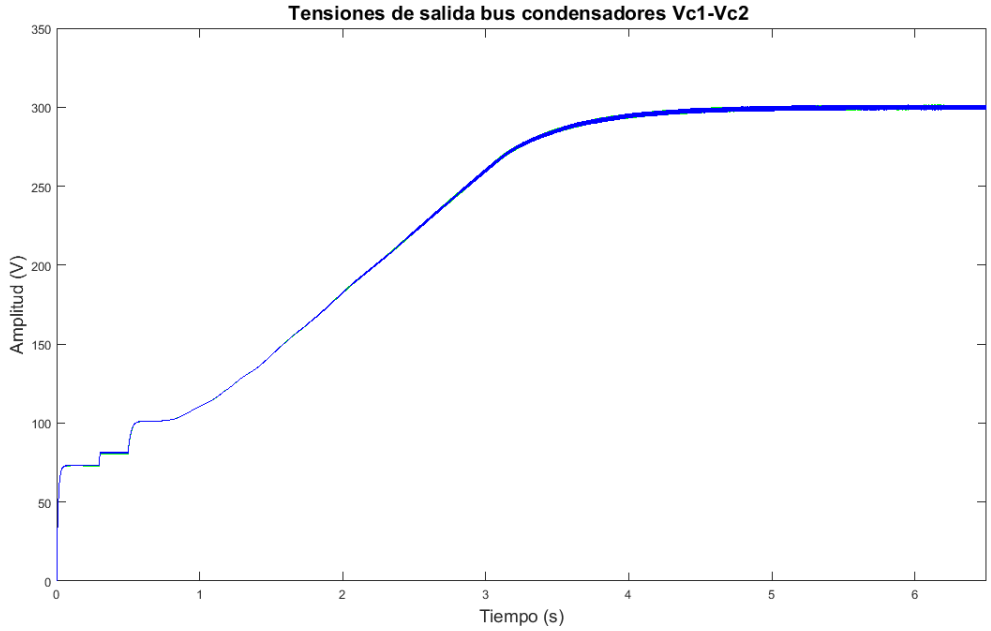


Ilustración 92. Tensión de salida en los condensadores del bus continúa Vdc 120 V/ 50 Hz 599,52 W

Se obtiene que las dos señales se encuentra superpuestas con lo que en todo momento siguen la referencia tienen el mismo potencial y no hay desniveles de tensión entre los capacitores.

7.6.2 Simulación 149 V / 60 Hz

La siguiente simulación se realiza con los valores obtenidos de anteriores estudios, con una tensión entrada trifasica de 149 V / 60 Hz, la carga que se coloca a la salida disipa una potencia de 854,9 W, con una carga puramente resistiva de 420 Ω.

Vamos a comprobar las señales de entrada, tanto las tensiones de linea y de fase que genera la maquina sincronica como las intensidades de entrada

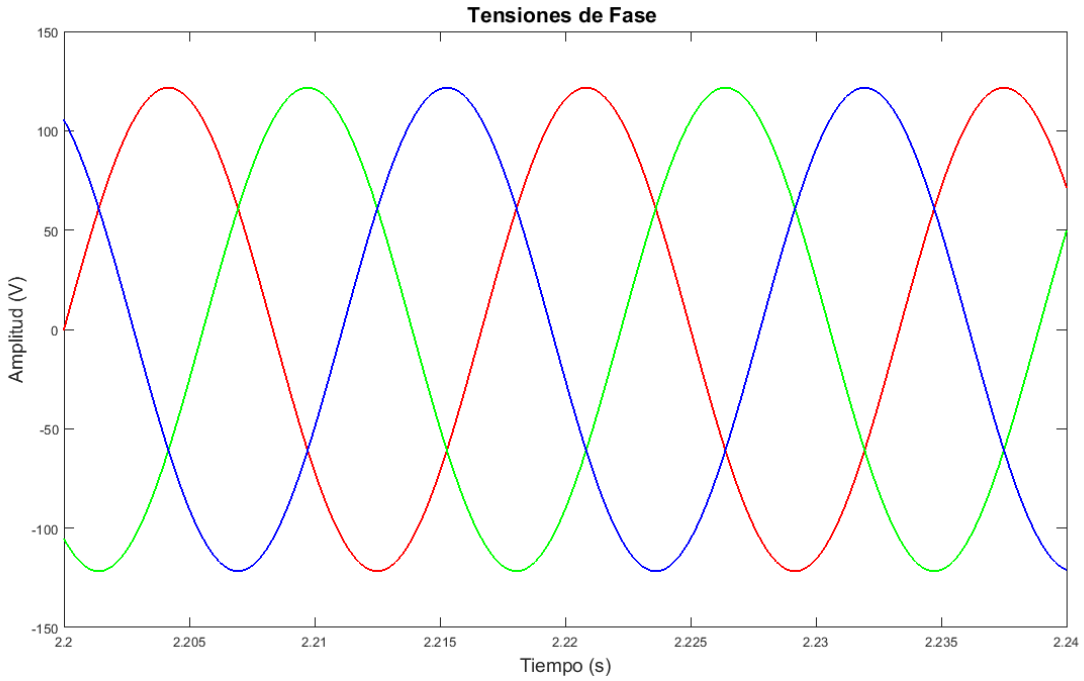


Ilustración 93. Tensiones de fase 149 V/ 60 Hz 854,9 W

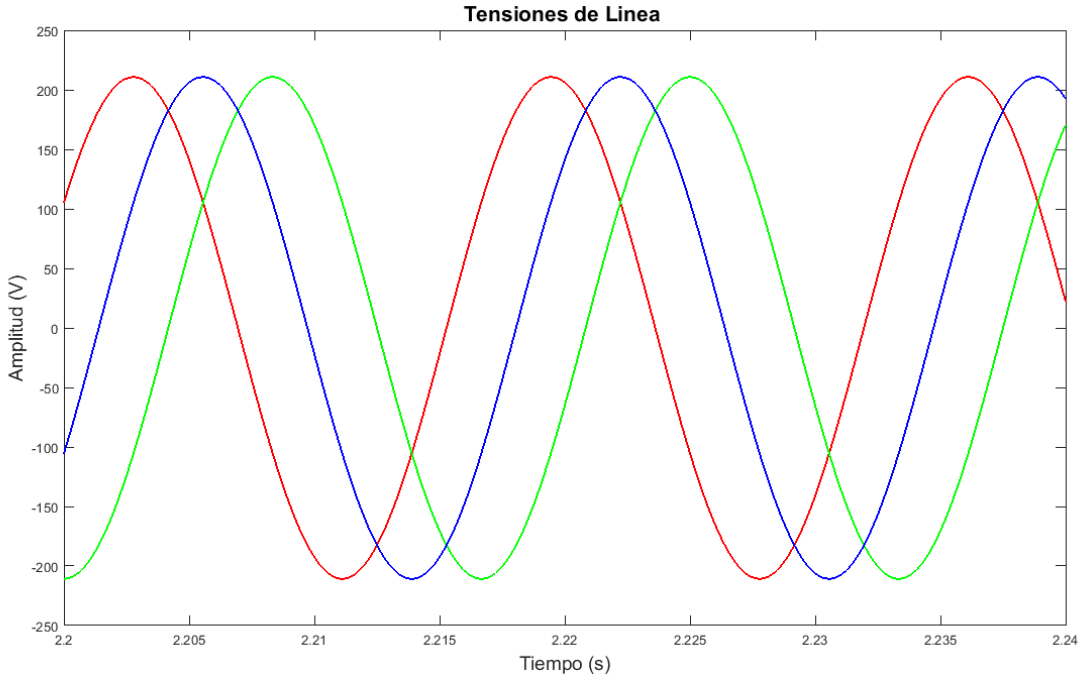


Ilustración 94. Tensiones de línea 149 V/ 60 Hz 854,9 W

En la adquisición de las corrientes de entradas, verificamos si se encuentran dentro del rango admisible de entrada de la placa TIDM-1000 y los niveles de armónicos acoplados a la corriente.

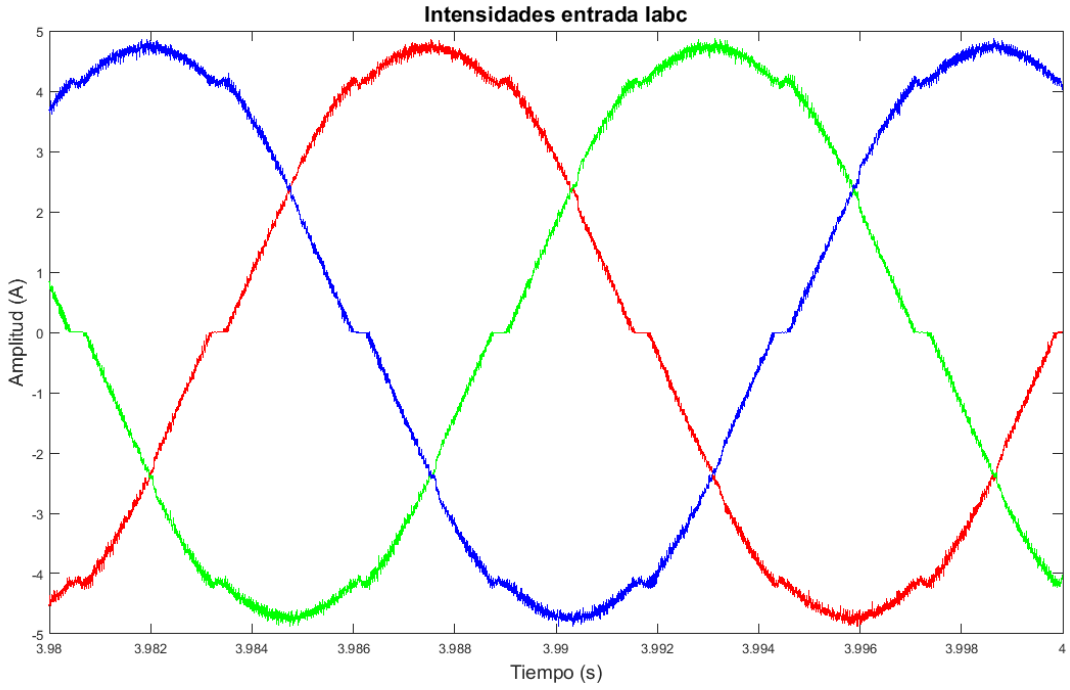


Ilustración 95. Intensidades de entrada del rectificador Vienna 149 V/ 60 Hz 854,9 W

Se oberva como la señal tiene forma sinusoidal, pero no es completamente pura, esto se debe a los armonicos acoplados en ella, tambien observamos que la intensidad se encuentran dentro del rango que nos permite la placa TIDM-1000, con una intensidad eficaz de I_{rms}=3,385A .

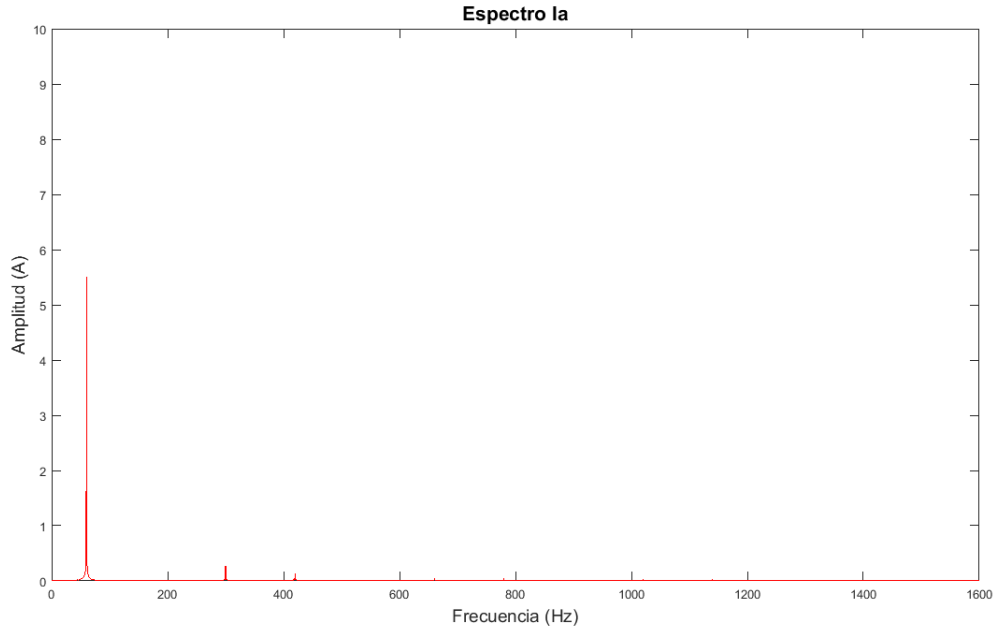


Ilustración 96. Espectro en frecuencia de la Intensidad de entrada Ia 149 V/ 60 Hz 854,9 W

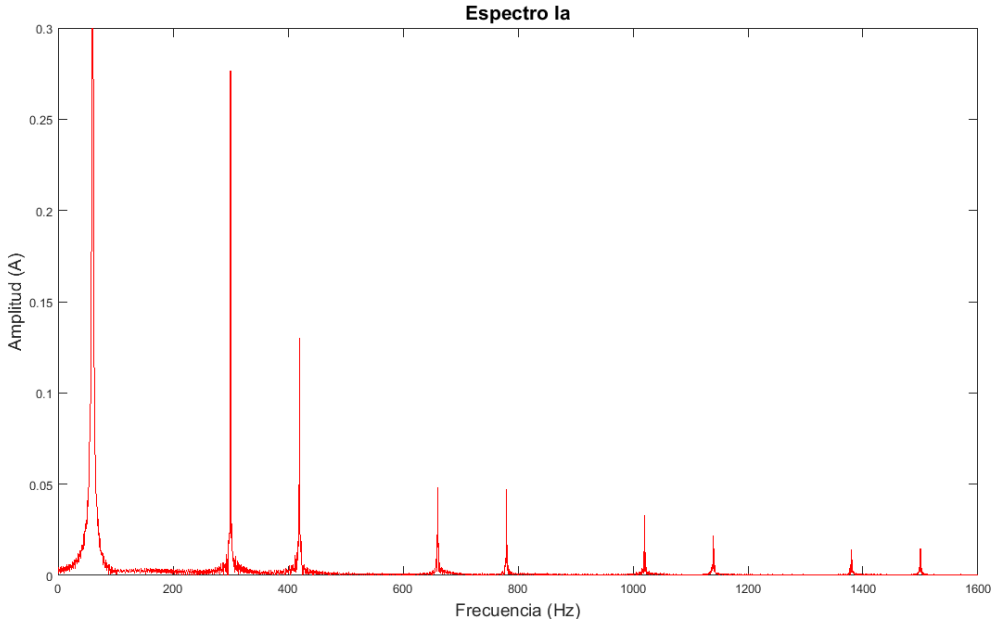


Ilustración 97. Espectro en frecuencia de la Intensidad de entrada Ia 149 V/ 60 Hz 854,9 W (2)

En el espectro en frecuencia que se obtiene, se observa como los armónicos que están acoplados, son el : 5,7,11,13,17,19,23,25 ,al tener un sistema balanceado, también vemos como al ser múltiplo de la frecuencia fundamental de 60 Hz llegamos hasta los 1550 Hz de componente de armónicos, también aunque no se ven en este espectro, tenemos acoplados los referentes a la frecuencia de conmutación 50 kHz y sus múltiplos.

En esta simulación la distorsión armónica total que obtenemos en las intensidades de entrada es THDi= 3,664 %.

Con estos datos ya analizados, pasamos a ver los resultados obtenidos a la salida del sistema, primero veremos la tensión de salida en el bus continua y su rizado de tensión.

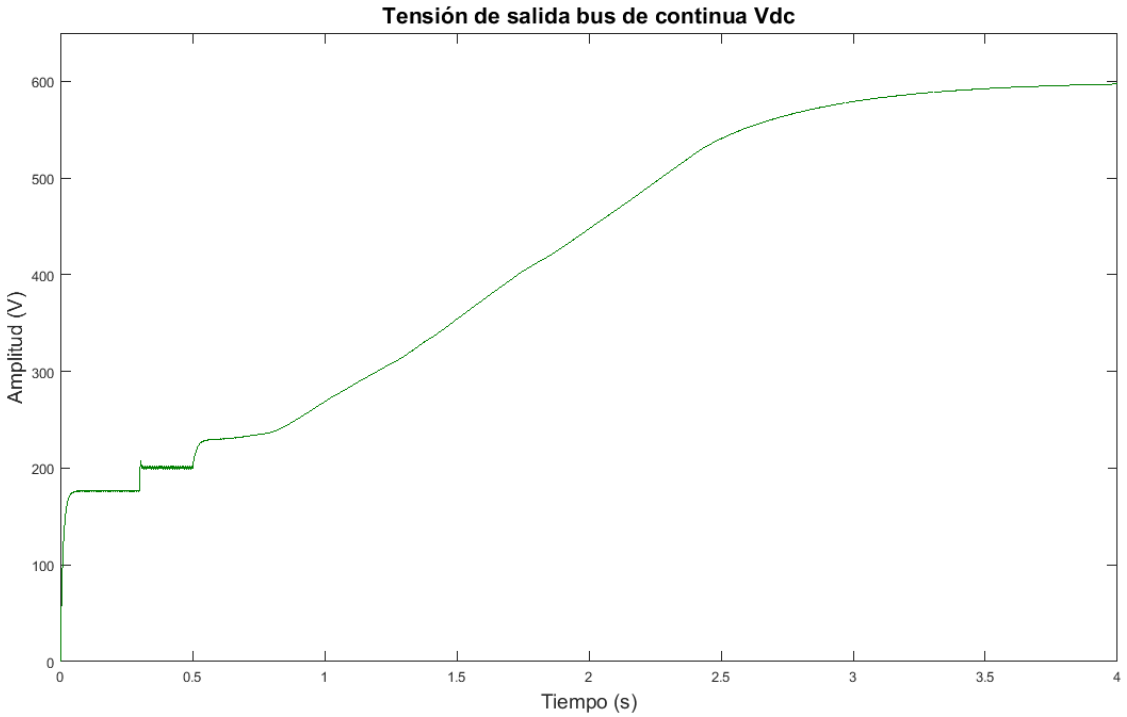


Ilustración 98. Tensión de salida del bus continúa 149 V/ 60 Hz 854,9 W

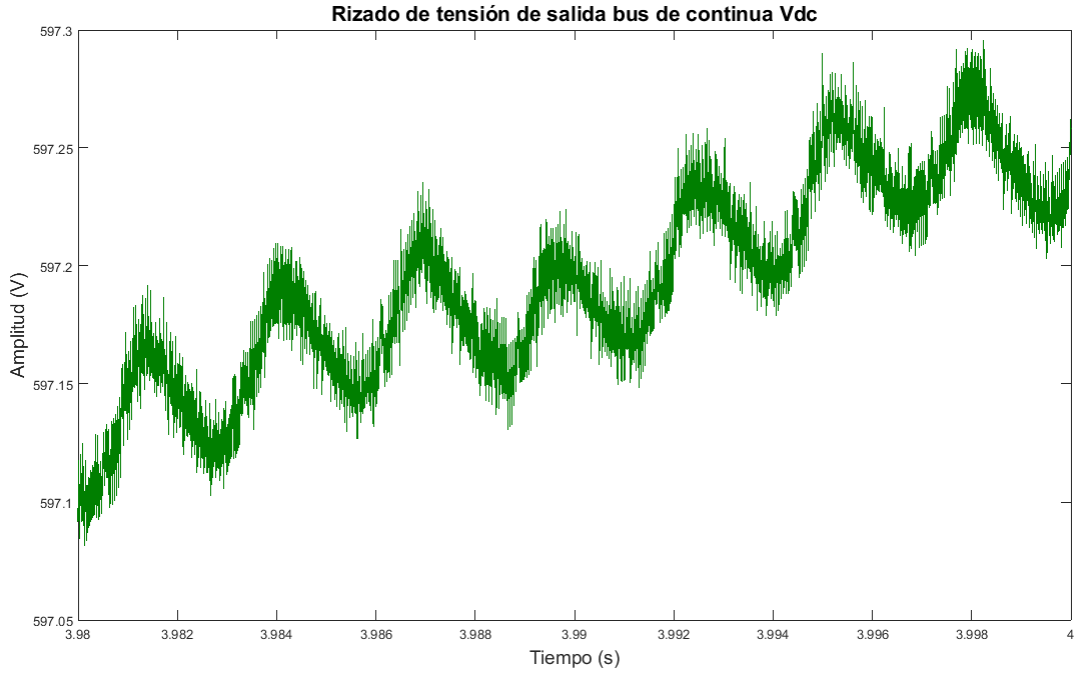


Ilustración 99. Tensión de rizado a la salida del bus continúa Vrizado 149 V/ 60 Hz 854,9 W

Con los datos obtenidos, ser verifica que el sistema alcanza la referencia de 600 V alcanzando Vdc= 599,21 V suministrando una potencia de 854,9 W.

En la tensión de rizado a la salida, se observa como la variación de la tensión es mínima, siendo así $V_{rizado} \approx$ 0,25 V, por lo que se considera que obtenemos una señal continua a la salida.

Ahora vamos a observar el comportamiento que tiene la intensidad de salida en el bus de continua y el rango que tiene su corriente de rizado.

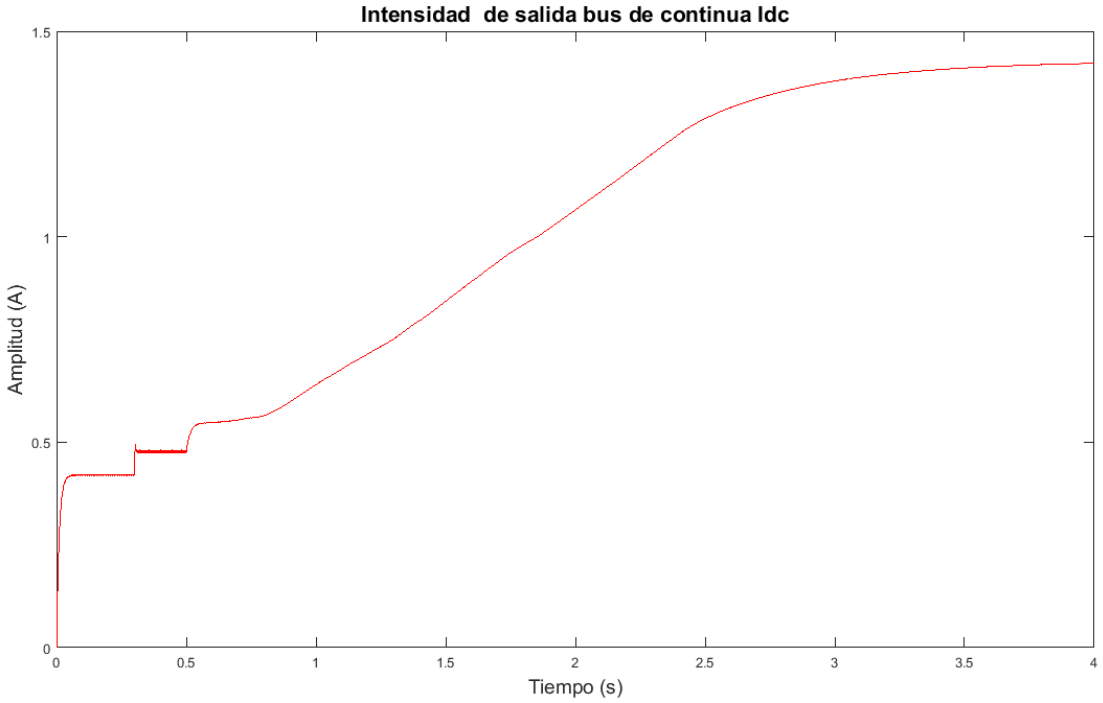


Ilustración 100. Intensidad a la salida del bus de continua Idc 149 V/ 60 Hz 854,9 W

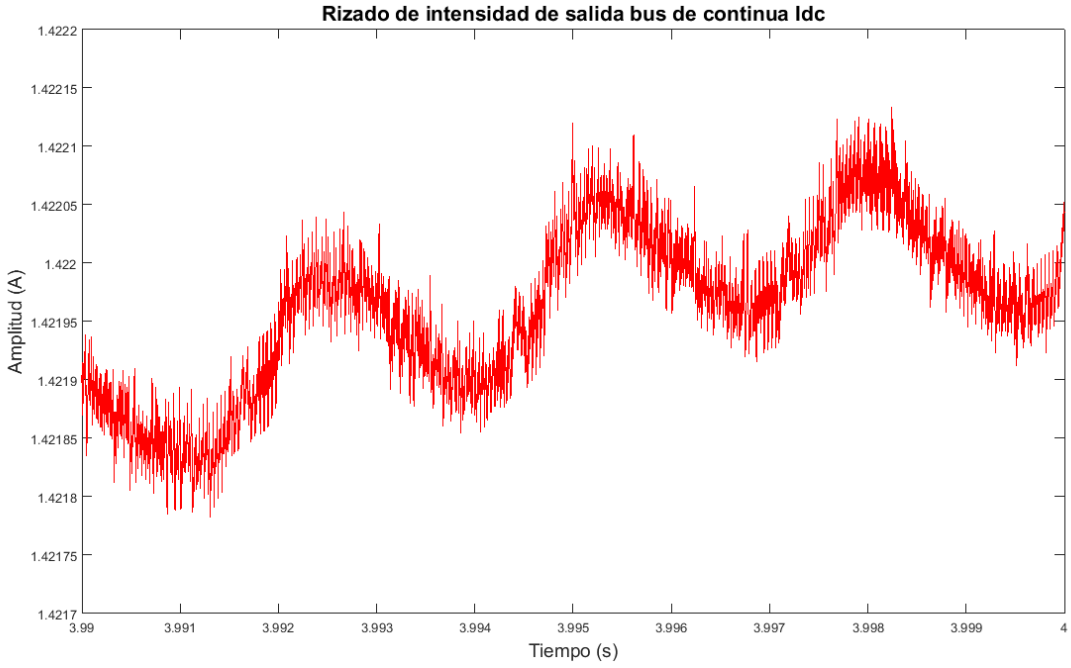


Ilustración 101. Rizado en la intensidad a la salida del bus de continua 149 V/ 60 Hz 854,9 W

Con obtención de la intensidad a la salida del bus de continua, observamos que la intensidad que circula por la carga es I_{out}= 1,4267 A, por lo tanto nos encontramos dentro del límite de corriente que admite el rectificador de Vienna a la salida de la placa TIDM-1000, el cual permita hasta 5 A.

Si analizamos el rizado se verifica que se puede considerar como corriente continua ya que la variación es tan solo de I_{rizado}≈0,3 mA.

Para comprobar que existe balance en los condensadores vemos las tensiones de bus en las dos ramas de condensadores.

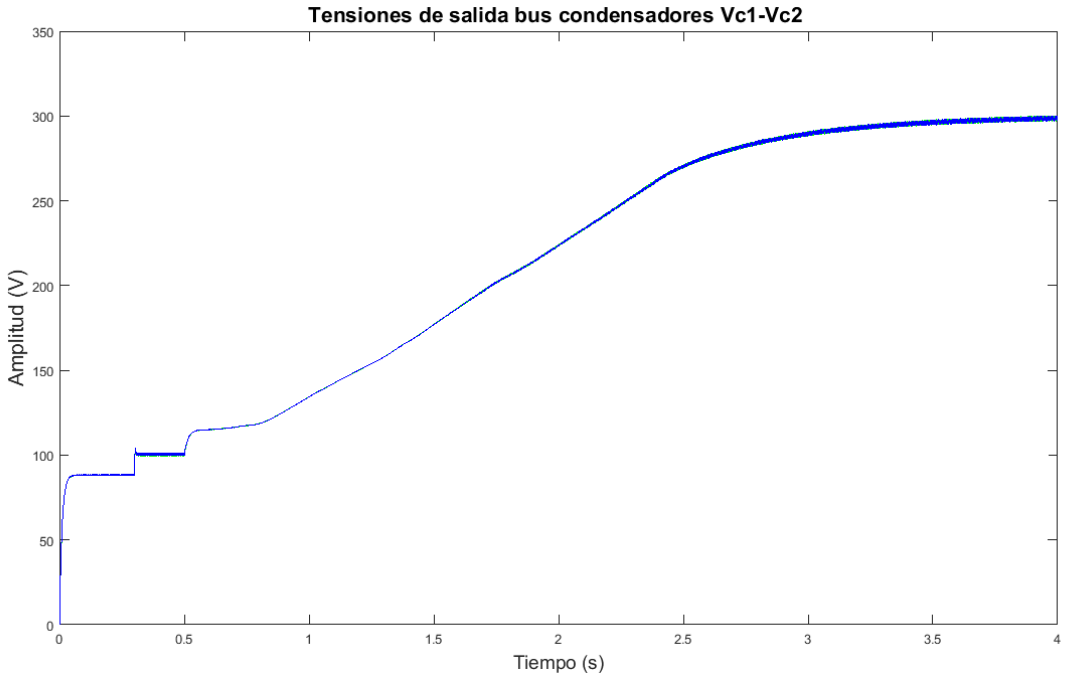


Ilustración 102. Tensión de salida en los condensadores del bus continúa Vdc 149 V/ 60 Hz 854,9 W

Comprobamos como los condensadores en todo momento se encuentran al mismo potencial, sí aplicásemos mayor potencia al sistema, se podría observar como ya no aparecería dicho efecto y tardarían cierto tiempo en compensarse.

7.7 Simulación de escalones en carga

En este apartado se estudia el comportamiento del sistema rectificador de Vienna cuando tenemos en la entrada los parametros de la máquina sincrónica con 149 V/ 60 Hz.

Así realizamos un analisis con escalones de carga primero con 300 W-1200 Ω y pasado un tiempo con 600 W-600 Ω,el tiempo de activación del escalon se produce con 3,8 s, estando ya estabilizado el sistema a la salida.

Lo primero que vamos observar será la evolución de la corriente de entrada en el rectificador de Vienna respecto a los escalones.

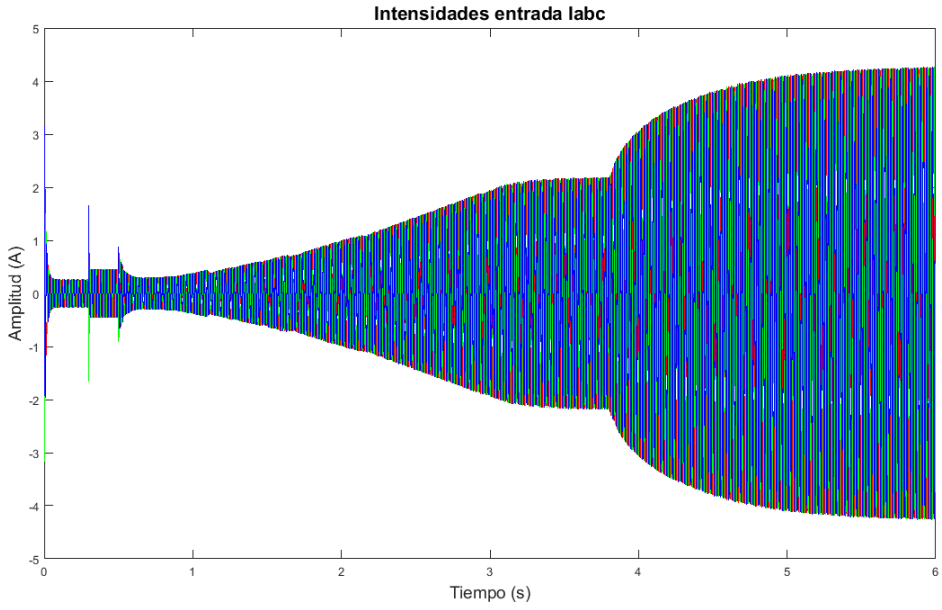


Ilustración 103.Evolución temporal intensidad de entrada 149 V /60 Hz

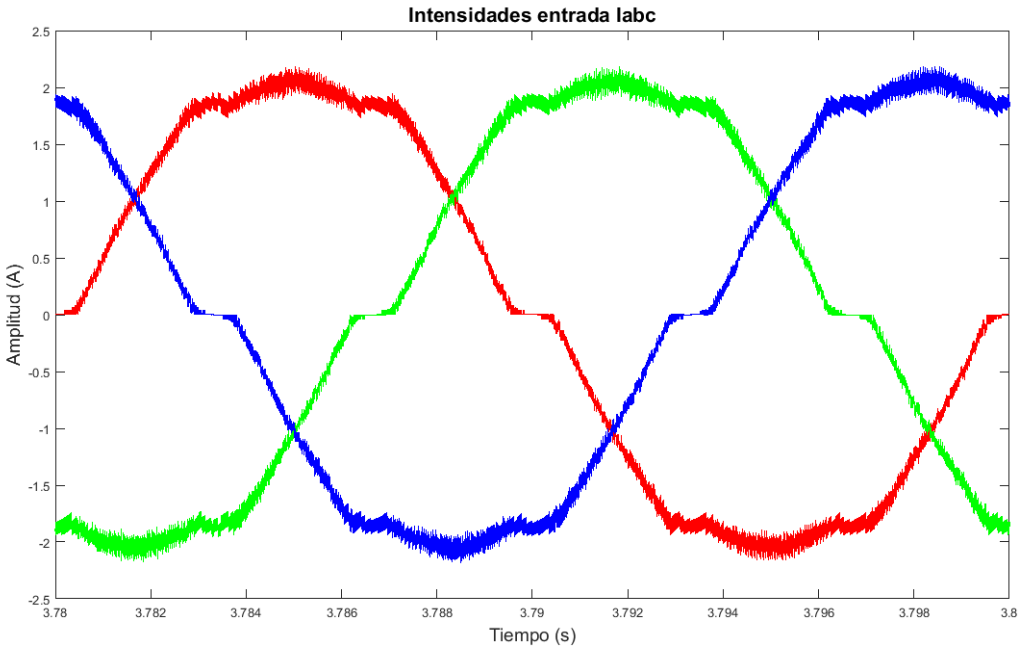


Ilustración 104.Intensidad de entrada para 300 W, antes de escalón 149 V /60 Hz

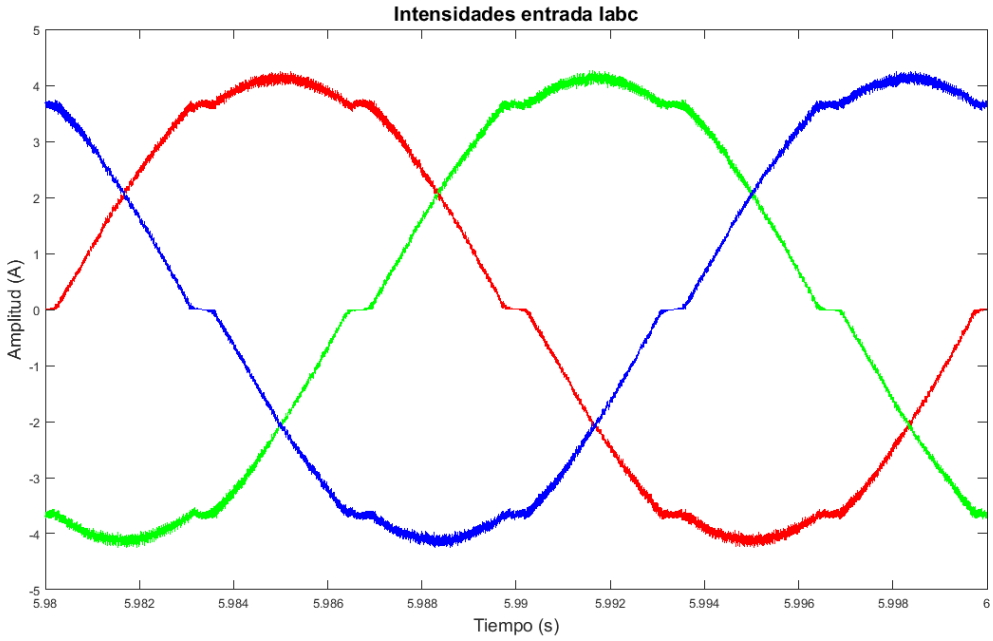


Ilustración 105.Intensidad de entrada 600 W, después de escalón 149 V /60 Hz

Se observa como empieza a aumentar la intensidad en la entrada del rectificador de Vienna, al producirse el escalon en la carga, y como tarda aproximadamente 1,5 s en estabilizarse para la carga aplicada, con mayor escalon el tiempo seria mayor , ya que tardaria mas en llegar al valor de referencia de intensidad.

Ahora nos disponemos a ver el comportamiento de la tensión de salida del bus de continua y la intensidad del bus.

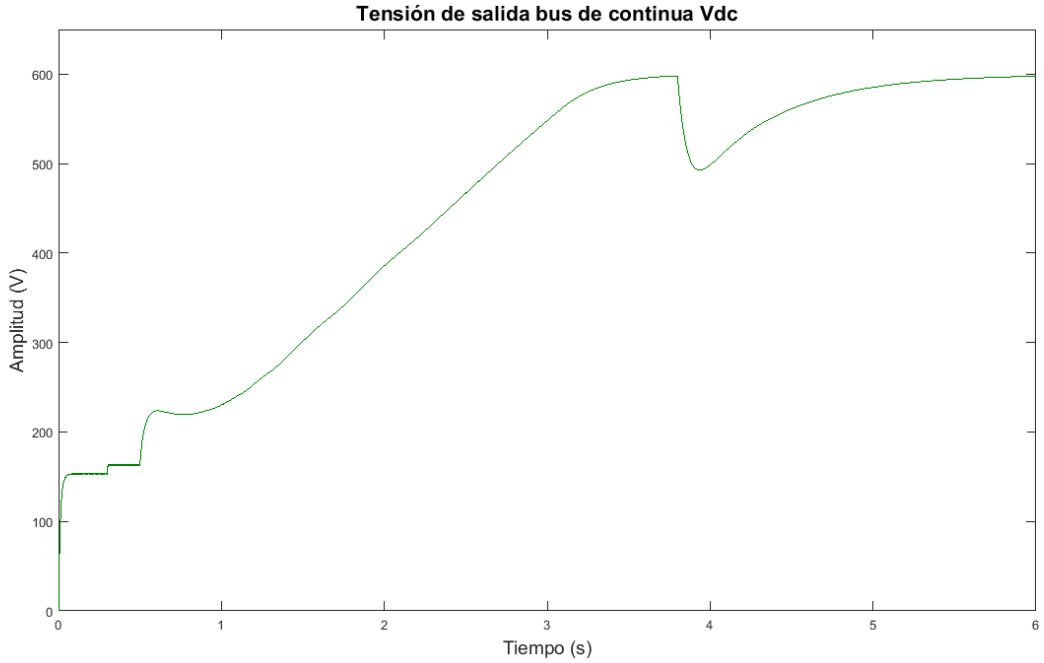


Ilustración 106.Evolución temporal Vdc escalón de carga 149 V / 60 Hz

Se comprueba como al conectar la carga el sistema hay una caida en la tensión del bus de aproximadamente 100 V.

Despues de aproximadamente 1 s, la tensión de salida del bus de continua se observa como vuelve a compensar la caida y sigue la referencia fijada.

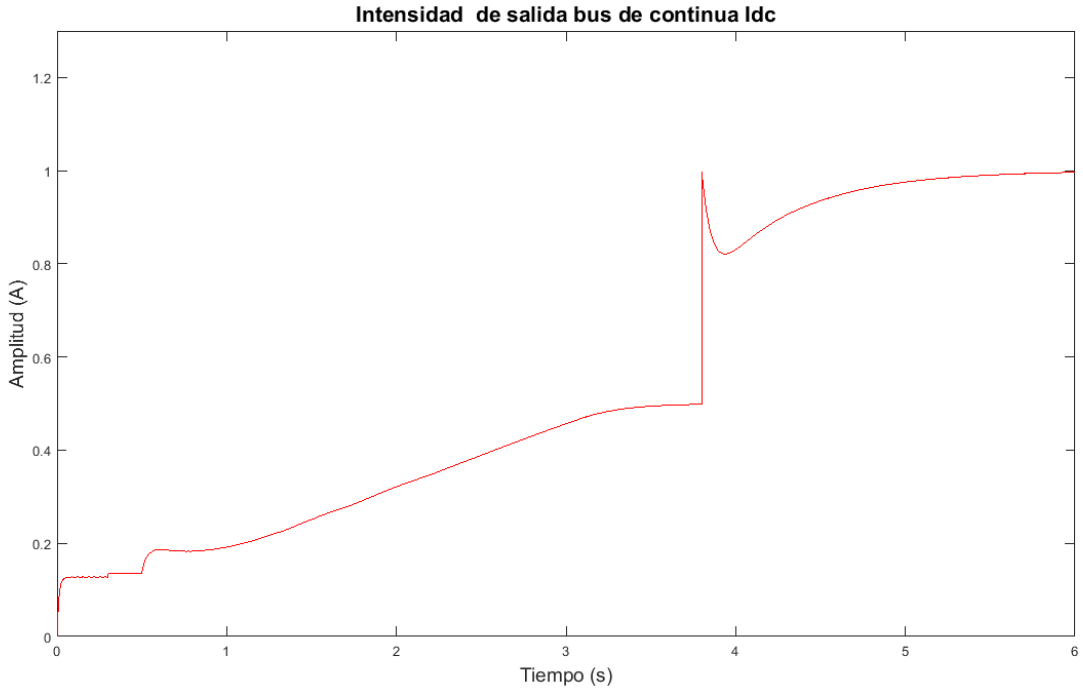


Ilustración 107. Evolución temporal Idc escalón de carga 149 V / 60 Hz

En al intensidad a la salida del bus de continua, se observa como en el momento del escalon se produce un pico de corriente, equilibrandose cerca de un segundo mas tarde.

7.8 Simulación semiconductores

Para ver el comportamiento y las tensiones que van a tener que soportar los semiconductores realizamos una simulación en el caso más desfavorable respecto a niveles de tensión estudiado en el presente proyecto, por lo tanto tomamos la simulación de parámetros de red europea con tensión 400V y 50 Hz, así como la carga que será de carga a máxima potencia de 2450 W suministrados en la salida.

Primero observamos la tensión que se produce en unos de los transistores Mosfet de potencia.

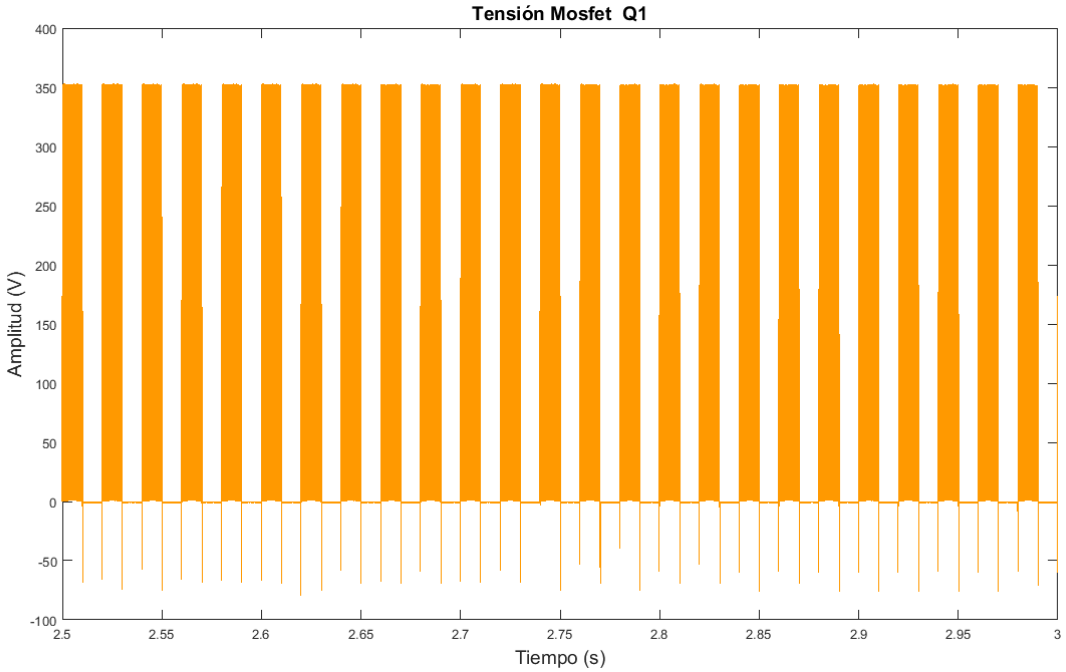


Ilustración 108.Tensión en bornes de transistor Mosfet de potencia 400 V/ 50Hz 2450 W

Se puede observar cómo se cumple una de las características, que cobran importancia en la configuración de Vienna es que la tensión que deben de soportar los interruptores de potencia, así vemos como estamos suministrando una tensión de 700 V en continua y como el transistor solamente está soportando la mitad de ella 350 V, con lo que de esta forma la vida de los elementos de conmutación puede alargarse.

Ahora veremos las tensiones que producen en uno de los diodos de potencia.

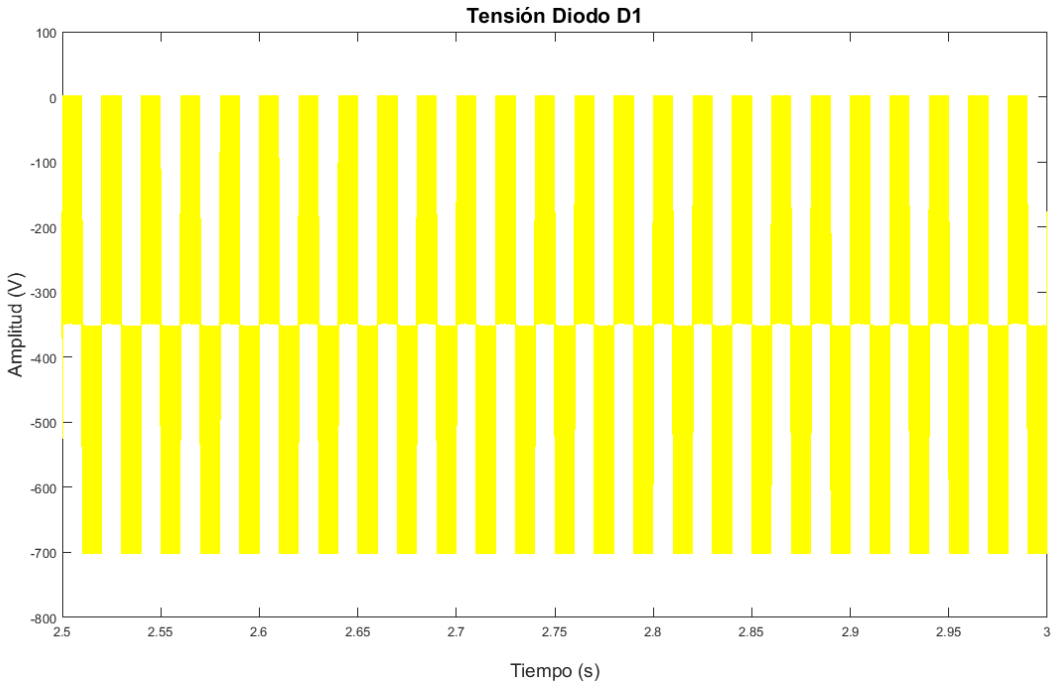


Ilustración 109.Tensión en bornes del diodo D1 de potencia 400 V/ 50Hz 2450 W

En el caso de las tensiones que deben soportar los diodos de potencia, se puede observar como esta en dos tramos de 350 V para el caso estudiado, con lo cual la tensión que está soportando en cada tramo de funcionamiento es la mitad de la tensión aplicada a la salida, alargando así su vida útil.

8. PRESUPUESTO

En el presente apartado se incluye el presupuesto del coste de la realización y estudio del trabajo de fin de grado, “Simulación y diseño de un rectificador de Vienna tipo T aplicado al control de un aerogenerador de 3 kW”.

En el presupuesto se incluyen las horas dedicadas a la realización del proyecto, así como a su estudio previo y simulación.

Se añade también el coste del software de licencias necesario para la ejecución del presente trabajo de fin de grado.

El coste de las horas dedicadas se estima con la tabla salarial 2018 del convenio de la industria, la tecnología y los servicios del metal de Valencia, en el grupo 1, clasificación profesional “Ingenieros y Licenciados”.

Presupuesto :Simulación y diseño de un rectificador de Vienna tipo T aplicado al control de un aerogenerador de 3 kW

Mano de obra	**Cantidad[horas]**	**Precio[€/h]**	**Total[€]**
Estudio previo	160	14,46	2313,6
Simulación	320	14,46	4627,2
Redacción	120	14,46	1735,2
	Total mano de obra		8676
Licencias Software	**Cantidad[unidades]**	**Precio[€/u]**	**Total[€]**
Microsoft Windows 7	0,3	74,99	22,497
Office profesional plus	0,3	75,99	22,797
Matlab standard	0,3	800	240
Matlab simulink	0,3	1200	360
Simscape	0,3	800	240
	Total licencias software		885,294
	Total		9561,294
	Impuestos		
	21%		2007,8717
	Total presupuesto		**11569,166**

9. CONCLUSIONES Y RESULTADOS

Gracias a los estudios realizados del rectificador de Vienna se puede observar como existen diversas topologías de este tipo de rectificador así como un abanico de métodos de control que se pueden aplicar para el funcionamiento de dicho rectificador, como su aparición e importancia en sistemas de conversión de energía eólica.

Tras la realización del presente proyecto, se puede observar la importancia de los convertidores electrónicos de potencia para sistemas de conversión de energía renovable, donde todavía existe un campo abierto donde poder indagar más a fondo.

Con la finalización de este trabajo de fin de grado, se da pie a continuar con la labor de investigación del sistema estudiado, con conexiones experimentales a un sistema aerogenerador para realizar su control y complementando el sistema con un inversor trifásico a la salida del rectificador de Vienna con el que conseguir tensiones trifásicas con posible conexión a red.

Como conclusión en lo referente a la simulación verificamos el funcionamiento del rectificador de Vienna con los parámetros para las redes eléctricas europea y americana con las que se cumplen los rangos de potencias en los que se encuentra la placa implementada TIDM-1000, verificando así como para la red europea funcionamos correctamente con 2,4 kW y con la red americana a 1,2 kW.

La verificación de la simulación en el caso de 400 V / 50 Hz, nos permite observar como el rectificador de Vienna es capaz de operar hasta potencias cercanas a 2,8 kW sin superar los límites de corriente de entrada y de salida establecidos en el modelo implementado, también se verifica como el factor de potencia es prácticamente unitario durante el análisis en carga y alcanzamos con esta simulación distorsiones armónicas totales en las corrientes de entrada THDi cercanas al 3 % en el rango de funcionamiento de la placa TIDM-1000, además de mantener una salida en el bus de continua que sigue la referencia marcada y prácticamente continua sin apenas tensión y corriente de rizado.

En simulaciones fuera del rango de la TIDM-1000, se observa cómo se consiguen valores de THDi cercanos al 1%, con potencias elevadas de 7 kW y 10 kW, lo cual deja abierto el camino para altas potencias, con otro modelo distinto.

La conclusión que obtenemos de las simulaciones realizadas con 208 V /60 Hz, son que el rectificador de Vienna con esta tensión de entrada, puede alcanzar potencias de hasta 1, 8 kW, cumpliendo con las especificaciones de funcionamiento del sistema sin superar los límites de corrientes, además de conseguir un THDi inferior al 5 % y cercano al 2 % para 1,8 kW, se consigue factor de potencia prácticamente unitario y tensión a la salida del bus de continua capaz de seguir la referencia marcada y sin apenas rizado de tensión y de corriente, y también manteniendo una eficiencia superior al 98%.

Para las simulaciones realizadas con los parámetros analizados de ensayos anteriores de la maquina sincrónica de imanes permanentes, la cual simula a un sistema aerogenerador, obtenemos que para 120 V/50 Hz somos capaces de operar hasta potencias cercanas a los 800W, cumpliendo con los límites de corriente de la placa TIDM-1000, con un factor de potencia unitario y alta eficiencia, superior al 97 %, el THDi que conseguimos en este rango de potencias supera ligeramente el 3 %.

La simulación realizada con 149 V/ 60 Hz nos permite trabajar con un rango mayor de potencias de salida que en el caso de 120 V/ 50 Hz , llegando así a poder suministrar potencias cercanas a 1 kW, con niveles de factor de potencia unitarios y eficiencias superiores al 98 %, las potencias que es capaz de suministrar dentro de los límites de corrientes de entrada y de salida son cercanas a 1kW de potencia de salida, los niveles de THDi que conseguimos en el rango de operación con estas características se mantiene inferior al 4 %.

En el análisis de los resultados comentados, extraemos como el diseño que se ha implementado en la placa TIDM-1000, funciona conforme a lo esperado para 400 V/ 50 Hz y 208 V / 60 Hz, con el sistema aerogenerador conectado a la entrada alcanzamos rangos de potencia de 800 W para 120 V / 50 Hz y de 1 kW para 149 V / 60 Hz.

Si pretendemos extraer más potencia del aerogenerador con los mismos valores de tensiones, vemos que nuestro diseño está limitado ya que a partir del rango de potencias citado, superamos los límites de corrientes de entrada y salida que estable la placa TIDM-1000.

Para sacar más partido al aerogenerador, necesitaríamos una placa con mayores prestaciones referentes a los rangos de intensidades e implementar un control para regular la velocidad de giro de la máquina, con el fin de conseguir un sistema estable de conversión de energía.

10. BIBLIOGRAFÍA

[1] Johan W.Kolar, Thomas Friedli"The Essence of Three-Phase PFC Rectifier Systems-Part I", VOL. 28, 1, January 2013.

[2] Hartman, S.D.Round , H.Ertl, J.W.Kolar "Digital Current Controller for a 1MHz, 10kW Three-Phase VIENNA Rectifier" .Vol.24,NO.11,November 2009.

[3] C.Dang, X.Tong, J.Huang, Q.Wang, H.Zhang "QPR and Duty Ratio Feed-Forward Control for Vienna Rectifier of HVDC Supply System". DOI: 10.1002.

[4] S.A.Shaon, K.M.A.Salam "Study of Vienna Rectifier and a Highly Efficient Single Phase Two Stage Inverter with low THD". 20-22 December, 2014, Dhaka, Bangladesh.

[5] L.Hang,M.Zhang,L.M.Tolbert,Z.Lu "Digitized Feedforward Compensation Method for High-Power-Density Three-Phase Vienna PFC Converter".Vol.60,NO.4,April 2013.

[6] J-S.Lee, K-B.Lee "Carried-Based Discontinuous PWM Method for VIENNA Rectifiers".DOI:10.1109.

[7] W.Ding, C.Zhang, F.Gao, B.Duan, H.Qiu "A Zero-Sequence Component Injection Modulation Method with Compensation for Current Harmonic Mitigation of Vienna Rectifier".DOI:10.1109.

[8] B.Kedjar, H.Y.Kanaan, K.Al-Haddad "Vienna Rectifier With Power Quality Added Function".Vol.61, NO.8, August 2014.

[9] Remus Teodorescu, Marco Liserre, Pedro Rodríguez "Grid Converters for Photovoltaic and Wind Power Systems"2011, Wiley.

[10] H.Chen, N.David, D.C.Aliprantis "Analysis of Permanent-Magnet Synchronous Generator With Vienna Rectifier for Wind Energy Conversion System" .Vol. 4, NO.1, January 2013.

[11] H.Chen, D.C.Aliprantis "Analysis of Squirrel-Cage Induction Generator With Vienna Rectifier for Wind Energy Conversion System".Vol.26, No.3, September 2011.

[12] A.Rajaei, M.Mohamadian, A.Y.Varjani "Vienna Rectifier-Based Direct Torque Control of PMSG for Wind Energy Application".12-0184-TIE.

[13]Vienna Rectifier-Based, Three-Phase Power Factor Correction (PFC) Reference Design Using C2000™ MCU, Texas Instruments, November 2016-Revised July 2017.

[14] Elías Hurtado Pérez "Ensayos y resultados obtenidos de la máquina sincrónica de imanes permanentes mod.3000 120 V, Departamento de ingeniería eléctrica, Universidad Politécnica de Valencia.

[15] Muhammad H.Rashid, "Electrónica de potencia: Circuitos, dispositivos y aplicaciones", Tercera Edición, Pearson Prentice Hall, 2004.

[16] Julián J.Salt Llobregat, Ángel Cuenca Lacruz, Vicente Casanova Calvo, Antonio Correcher Salvador "Control automático: Tiempo Continuo y Tiempo Discreto" Editorial Reverte ED|UPV, 2015.

[17] Fco.J.Gimeno Sales, Salvador Seguí Chilet, Salvador Orts Grau "Convertidores Electrónicos: Energía Solar Fotovoltaica, Aplicaciones y Diseño" Editorial Universidad Politécnica de Valencia.

[18] Fang Lin Luo, Hong Ye “Power Electronics: Advanced Conversion Technologies” CRC Press.

[19] Johan W.Kolar, Thomas Friedli”The Essence of Three-Phase PFC Rectifier Systems-Part II”, VOL. 28, 1, January 2013.

[20] Tiara R.S de Freitas, Paulo J.M.Menegáz, Domingos S.L Simonetti “Rectifier topologies for permanent magnet synchronous generator on wind energy conversion system: A review “Power Electronics and Drives Laboratory-Department of Electrical Engineering Federal University of Espírito Santo, Brazil.

[21] Rixin Lai,Fei(Fred) Wang, Rolando Burgos,Dushan Boroyevich,Dong Jiang, and Di Zhang”Average Modeling and Control Desing for VIENNA-Type Rectifiers Considering the DC-Link Voltage Balance”VOL.24,NO.11.November 2009.

[22] Lin Ma, Kai Tian, River TinHo Li, and Ken KuenFaat Yuen “A simple DC Bus voltage balancing control algorithm utilized in Vienna rectifiers” Corporate Research Center ABB(China) Limited,Beijing,China,2017.

[23] Hui Ma, Yunxiang Xie,Yubo Yang Zeyu Shi”Voltage Balance Control of Vienna-Type Rectifier Using SVPWM based On 60º Cordinate System” School of Electrical Power,South China University of Technology,China.

[24] Misha Kumar, Laszlo Huber, and Milan M.Jovanovic “Start-up Procedure for Three-Phase Six-Switch Boost PFC Rectifier” Delta Products Corporation, 2014.

[25] Amir Hosseing Rajaei,Mustafa Mohamadian,Seyed Mohamad Dehghan and Ali Yazdian “PMSG-based variable speed wind energy conversion system using Vienna rectifier”Department of Electrical Engineering,Tarbiat Modares University,Tehran 14115-111,Iran,2010.

[26] Raúl Fernández Burguillos “Emulador de una bancada eólica educacional mediante una plataforma con sistemas embebidos (TMSDSC28035)” Máster Universitario en Ingeniería Mecatrónica, Director:Dr.Francisco José Gimeno Sales, Septiembre de 2016.

[27] Wanfeng Zhang, Guang Feng, Yan-Fei Liu and Bin Wu “A Digital Power Factor Correction (PFC) Control Strategy Optimized for DSP” VOL.19, NO.6.November 2004.

[28] Johann W.Kolar, Uwe Drofenik, F.C. Zach “Current Handling Capability of the Neutral Point of a Three-Phase/Switch/Level Boost-Type PWM (VIENNA) Rectifier”Technical University Vienna, Power Electronics Section 359.5,1996.

ANEXO I.RESULTADOS SIMULACIONES RECTIFICADOR DE VIENNA TIPO T

A1.1 Simulación red europea 400 V / 50 Hz

Simulación para 50Hz-400VL-L-700Vdc_out										
R_Load	Vdc_out	Vc1_top	Vc2_botton	Pin	Pout	Iout	THD %	PF	Eficiencia	Irms_in
7200	700,01	350,04	349,96	72,268	68,057	0,097223	44,07	1	0,94173	0,114
3600	700,02	349,96	350,07	140,87	136,12	0,19445	28,15	1	0,96614	0,2112
1800	700,12	350,03	350,09	278,12	272,31	0,38895	18,19	1	0,97854	0,408
1200	699,9	349,96	349,94	414,87	408,22	0,58325	15,05	1	0,98375	0,6056
1020	700,11	350,14	349,97	487,33	480,54	0,68638	13,46	1	0,98604	0,7097
780	700,08	350,01	350,07	636,15	628,35	0,89754	11,29	0,99999	0,98734	0,924
600	699,98	350,07	349,91	826,23	816,61	1,1666	9,438	1	0,98833	1,198
420	699,96	350,39	349,57	1177,7	1166,5	1,6666	7,115	1	0,99042	1,704
300	699,83	350,33	349,5	1647,4	1632,5	2,3328	5,368	1	0,99075	2,381
220	699,98	350,23	349,76	2247,2	2227,2	3,1817	4,105	1	0,99121	3,246
200	700	350	350	2471,6	2450	3,5	3,745	1	0,99104	3,57
160	700,09	349,49	350,59	3089,5	3063,3	4,375	3,101	1	0,9913	4,461
140	700,07	350,75	349,33	3530,9	3500,7	5,0005	2,725	1	0,99143	5,098
100	699,92	349,81	350,11	4944,2	4898,9	6,9992	2,013	1	0,99123	7,138

Ilustración 110. Análisis de carga 400 V/ 50 Hz 700 Vdc_out

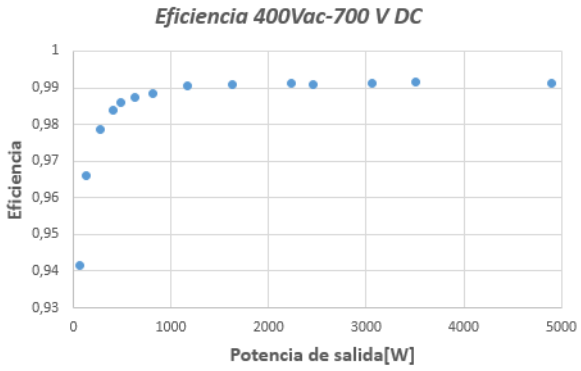


Ilustración 111.Eficiencia 400 V/ 50 Hz

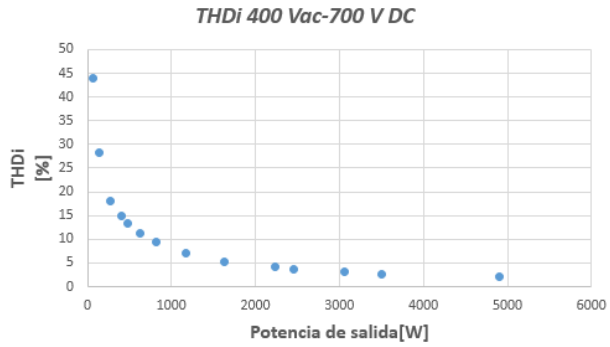


Ilustración 112.THDi 400 V/ 50 Hz

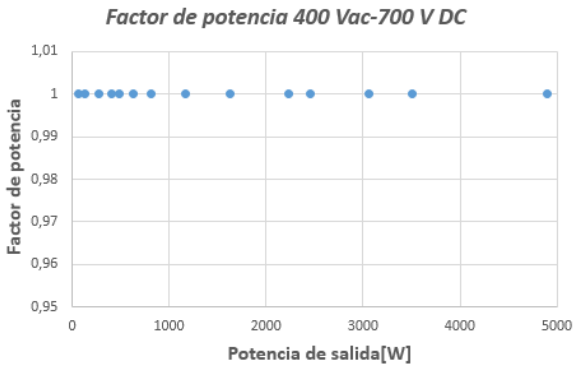


Ilustración 114.Factor de potencia 400 V/ 50 Hz

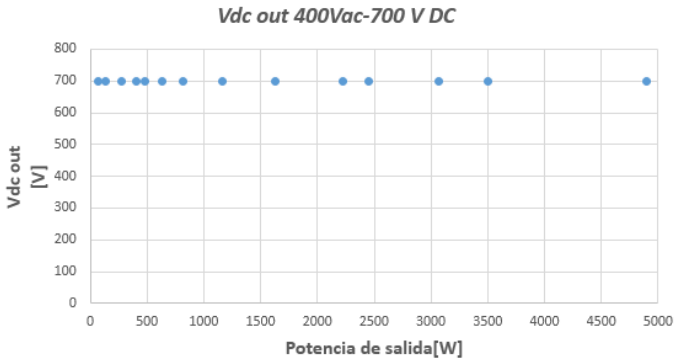


Ilustración 113.Vdc out 400 V/ 50 Hz

A1.2 Simulación red americana 208 V / 60 Hz

Simulación para 60Hz-208VL-L-600Vdc_out										
R_Load	Vdc_out	Vc1_top	Vc2_botton	Pin	Pout	Iout	THD %	PF	Eficiencia	Irms_in
7200	599,98	299,99	299,99	52,574	49,997	0,083331	29,29	0,99999	0,95093	0,1521
3600	599,7	299,91	299,79	101,88	99,899	0,16658	20,35	1	0,98047	0,2886
1800	600	300,03	299,97	203,98	200	0,33333	13,08	0,99999	0,98055	0,571
1200	600	299,18	300,18	304,88	300	0,5	10,59	0,99999	0,98404	0,851
1020	599,96	299,75	300,21	358,5	352,89	0,58819	9,557	0,99999	0,98437	0,9996
780	600	299,96	200,34	468,7	461,54	0,76923	7,889	0,99999	0,98478	1,305
600	599,95	299,6	300,35	608,44	599,89	0,99991	6,394	0,99999	0,98606	1,692
420	599,99	299,42	300,57	869,64	857,11	1,4285	4,84	0,99999	0,98558	2,417
300	599,87	299,03	300,85	1217,7	1199,5	1,9996	3,592	0,99999	0,98518	3,382
200	598,77	297,95	300,82	1822,6	1792,6	2,9938	2,515	0,99999	0,98361	5,061

Ilustración 115.Análisis de carga 208 V/ 60 Hz 600 Vdc_out

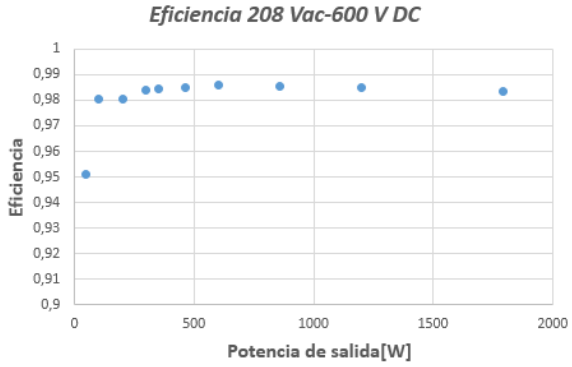


Ilustración 117.Eficiencia 208 V/ 60 Hz

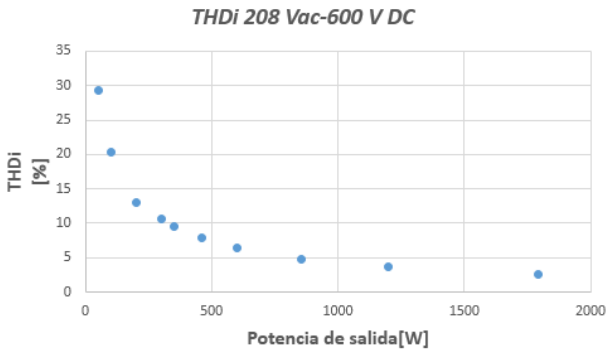


Ilustración 116.THDi 208 V/ 60 Hz

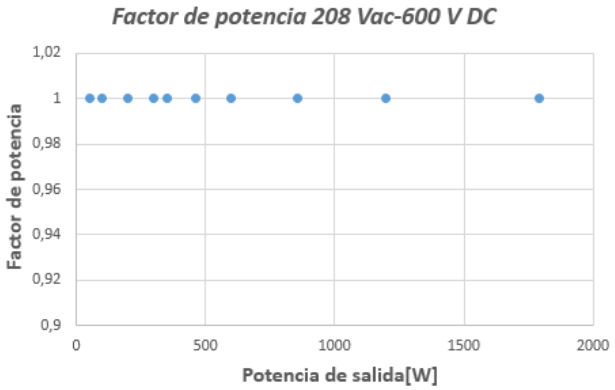


Ilustración 118.Factor de potencia 208 V/ 60 Hz

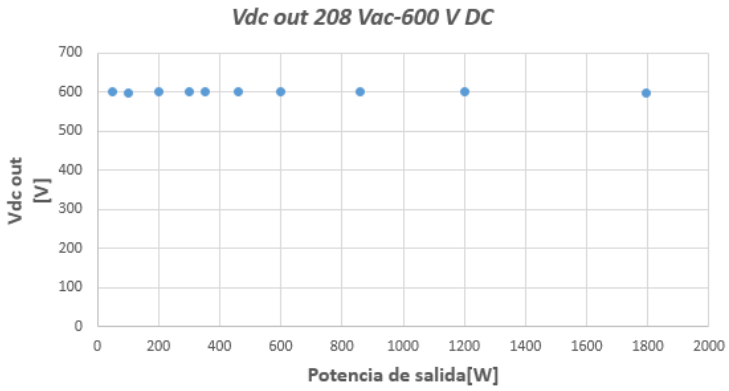


Ilustración 119. Vdc out 208 V/ 60 Hz

A1.3 Simulación máquina sincrónica 120 V / 50 Hz

Simulación para 50Hz-120VL-L-600Vdc_out										
R_Load	Vdc_out	Vc1_top	Vc2_botton	Pin	Pout	Iout	THD %	PF	Eficiencia	Irms_in
7200	600,7	300,34	300,36	53,439	50,117	0,083431	24,96	1	0,9378	0,265
3600	599,5	299,54	299,96	103,61	99,833	0,16653	16,39	1	0,96353	0,5051
1800	602,29	301,18	301,11	206,59	201,53	0,3346	10,62	0,99999	0,97545	0,9996
1200	599,82	299,25	300,57	307,22	299,82	0,49985	7,921	0,99999	0,97586	1,483
1020	598,82	298,54	300,28	360,47	351,55	0,58708	6,99	0,99999	0,9754	1,739
780	599,06	299,01	300,05	471,64	460,09	0,76803	5,557	1	0,9757	2,273
600	599,76	299,37	300,39	614,75	599,52	0,9996	4,407	0,99999	0,97505	2,961
420	599,62	299,62	300,46	879,85	856,06	1,4277	3,193	1	0,97281	4,235
300	597,74	295,83	301,92	1229,4	1191	1,9925	2,291	1	0,96894	5,916
250	595,43	294,11	301,32	1467,8	1418,1	2,3817	1,926	1	0,96639	7,063
200	590,46	290,82	299,63	1811,8	1743,2	2,9523	1,579	0,99999	0,96262	8,718

Ilustración 120. Análisis de carga 120V/ 50 Hz 600 Vdc_out

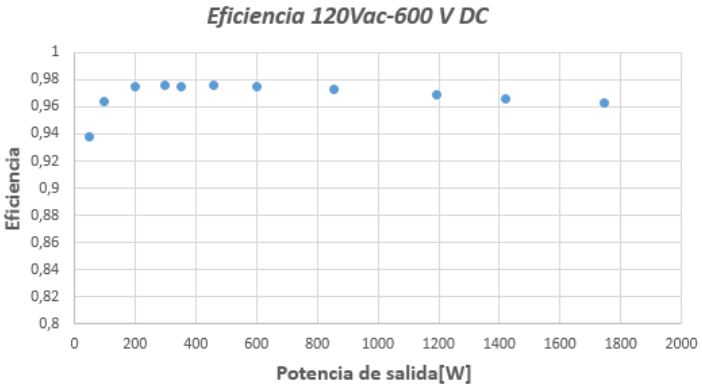


Ilustración 121.Eficiencia 120 V/ 50 Hz

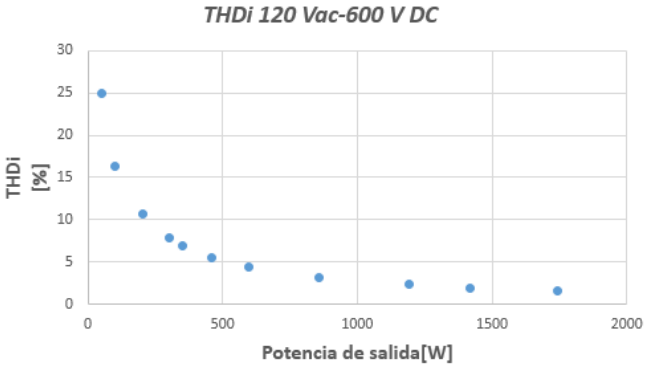


Ilustración 122.THDi 120 V/ 50 Hz

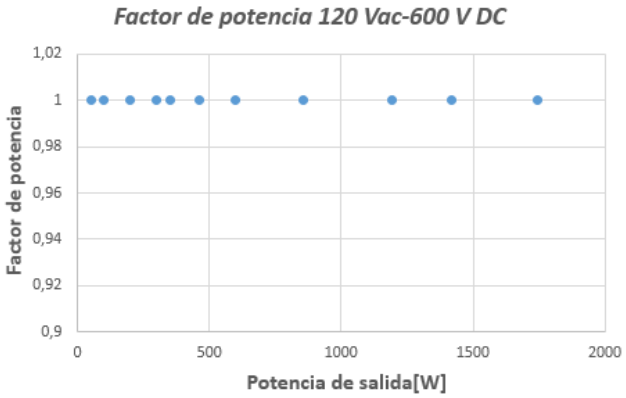


Ilustración 123.Factor de potencia 120 V/ 50 Hz

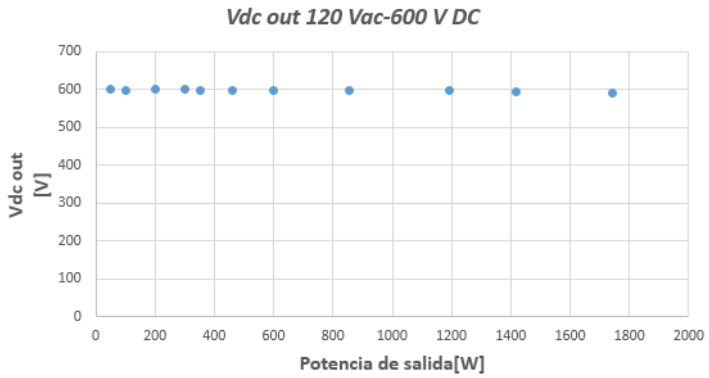


Ilustración 124. Vdc out 120 V/ 50 Hz

A1.4 Simulación máquina sincrónica 149 V / 60 Hz

Simulación para 60Hz-149VL-L-600Vdc_out										
R_Load	Vdc_out	Vc1_top	Vc2_botton	Pin	Pout	Iout	THD %	PF	Eficiencia	Irms_in
7200	600,01	300	300,01	52,978	50,002	0,083335	26,01	0,99999	0,94377	0,2121
3600	599,89	299,98	299,92	103,24	99,964	0,16664	17,55	0,99999	0,96835	0,4062
1800	600,08	300,01	300,07	204,24	200,06	0,33338	11,45	1	0,97949	0,7966
1200	600,01	299,99	300,02	306,08	300,01	0,50001	8,493	1	0,98008	1,19
1020	599,97	299,53	300,45	359,98	352,91	0,58821	7,489	1	0,98037	1,399
780	599,55	299,73	299,82	469,78	460,85	0,76866	6,061	1	0,98092	1,824
600	599,51	298,81	300,7	611,08	599,02	0,99918	4,957	0,99999	0,98024	2,371
420	599,21	299,43	299,78	873,08	854,9	1,4267	3,664	0,99999	0,97914	3,385
300	598,2	298,82	299,38	1221	1192,8	1,994	2,798	0,99999	0,97708	4,733
250	599,54	297,56	301,97	1473,6	1438,07	2,3981	2,400	0,99999	0,97589	5,712
200	598,05	299,81	298,23	1837,3	1788,3	2,9902	1,968	0,99999	0,9733	7,121

Ilustración 125. Análisis de carga 149V/ 60 Hz 600 Vdc_out

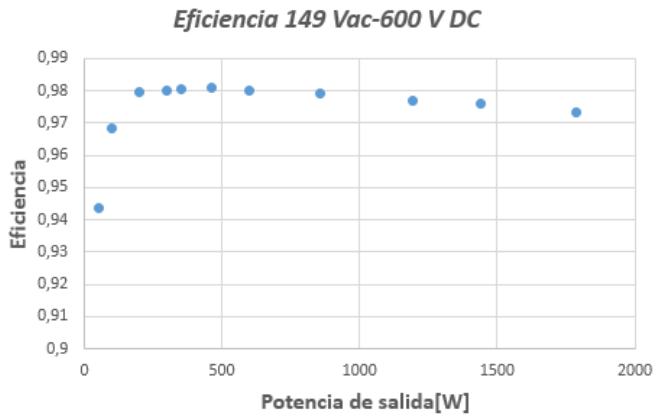


Ilustración 126.Eficiencia 149V/ 60 Hz

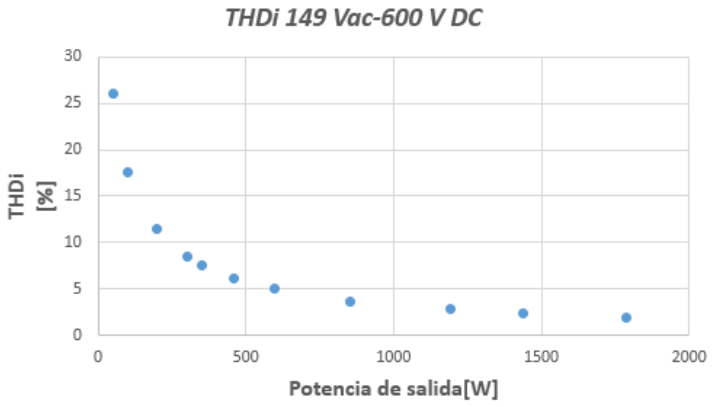


Ilustración 127.THDi 149V/ 60 Hz

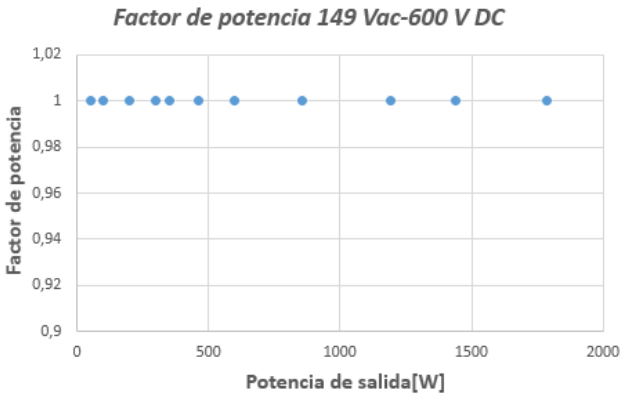


Ilustración 128.Factor de potencia 149V/ 60 Hz

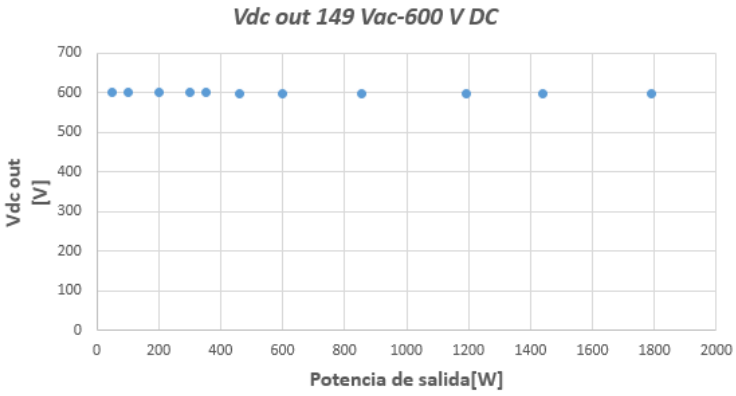


Ilustración 129. Vdc out 149 V/ 60 Hz

ANEXO II.SOFTWARE TIDM-1000

A2.1 pfc3phvienna_board.c

```
/*
=============================================================================
=
System Name:

File Name:

Target:                  TIDM_HV_1PH_DCAC R3, F2837xD

Author:                  Manish Bhardwaj

Description: This file consists of board related initialization, this file is
used to make the
                         main file more readable

Copyright (C) {2015} Texas Instruments Incorporated - http://www.ti.com/
 * ALL RIGHTS RESERVED*
=============================================================================
====  */
#include "pfc3phvienna_board.h"

//********************** USER Variables
***********************************//
// Global variables used that are applicable to this board
//*****************************************************************************
*//

// Variables used to indirectly address the peripheral
volatile struct EPWM_REGS *ePWM[] = {
            &EPwm1Regs,              //intentional: (ePWM[0] not used)
            &EPwm1Regs, &EPwm2Regs, &EPwm3Regs, &EPwm4Regs, &EPwm5Regs,
&EPwm6Regs,
            &EPwm7Regs, &EPwm8Regs, &EPwm9Regs, &EPwm10Regs, &EPwm11Regs,
&EPwm12Regs};

volatile struct SDFM_REGS *SDFM[] =
{     0, &Sdfm1Regs, &Sdfm2Regs};

// Used to indirectly access eQEP module
volatile struct EQEP_REGS *eQEP[] =
                           { &EQep1Regs,
                               &EQep1Regs,
                               &EQep2Regs,
                           };

// Used to indirectly access ADC module
volatile struct ADC_REGS *ADC[] =
                           { &AdcaRegs,
                               &AdcaRegs,
                               &AdcbRegs,
                               &AdccRegs,
                               &AdcdRegs
                           };

// Used to indirectly access CMPSS module
volatile struct CMPSS_REGS *CMPSS[] =
```

```
                                { &Cmpss1Regs,
                                    &Cmpss1Regs,
                                    &Cmpss2Regs,
                                    &Cmpss3Regs,
                                    &Cmpss4Regs,
                                    &Cmpss5Regs,
                                    &Cmpss6Regs
                                };


/*  This routine sets up the basic device ocnfiguration such as initializing
PLL
    copying code from FLASH to RAM */
//TODO device_setup()
void setupDevice(void)
{
#ifdef FLASH
// Copy time critical code and Flash setup code to RAM
// The  RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
// symbols are created by the linker. Refer to the linker files.
    memcpy(&RamfuncsRunStart, &RamfuncsLoadStart, (Uint32)&RamfuncsLoadSize);
#endif

      //DeviceInit();    // Device Life support & GPIO
      // Step 1. Initialize System Control:
      // PLL, WatchDog, enable Peripheral Clocks
      // This example function is found in the F28M3Xx_SysCtrl.c file.
      InitSysCtrl();

#ifdef FLASH
// Call Flash Initialization to setup flash waitstates
// This function must reside in RAM
      InitFlash(); // Call the flash wrapper init function
#endif //(FLASH)

      // Step 2. Initialize GPIO:
      // This example function is found in the F28M3Xx_Gpio.c file and
      // illustrates how to set the GPIO to it's default state.
      //InitGpio(); // Skipped for this example
      // Step 3. Clear all interrupts and initialize PIE vector table:
      // Disable CPU interrupts
      DINT;

      // Initialize the PIE control registers to their default state.
      // The default state is all PIE interrupts disabled and flags
      // are cleared.
      // This function is found in the F28M3Xx_PieCtrl.c file.
      InitPieCtrl();

      // Disable CPU interrupts and clear all CPU interrupt flags:
      IER = 0x0000;
      IFR = 0x0000;

      // Initialize the PIE vector table with pointers to the shell
Interrupt
      // Service Routines (ISR).
      // This will populate the entire table, even if the interrupt
      // is not used in this example.  This is useful for debug purposes.
      // The shell ISR routines are found in F28M3Xx_DefaultIsr.c.
```

```
	// This function is found in F28M3Xx_PieVect.c.
	InitPieVectTable();

	// initialize CPU timers
	InitCpuTimers();
	ConfigCpuTimer(&CpuTimer0, 200, 10000);
	ConfigCpuTimer(&CpuTimer1, 200, 20000);
	CpuTimer0Regs.TCR.all = 0x4001; // Use write-only instruction to set
TSS bit = 0
	CpuTimer1Regs.TCR.all = 0x4001; // Use write-only instruction to set
TSS bit = 0

}

//TODO setupADC()
void setupADC(void)
{
	//Write ADC configurations and power up the ADC for both ADC A
	Uint16 i;

	EALLOW;
	//write configurations for ADC-A
	// External REFERENCE must be provided
	AdcaRegs.ADCCTL2.bit.PRESCALE = 6; //set ADCCLK divider to /4
	AdcaRegs.ADCCTL2.bit.RESOLUTION = RESOLUTION_12BIT;
	AdcaRegs.ADCCTL2.bit.SIGNALMODE = SIGNAL_SINGLE;

	//Set pulse positions to late
	AdcaRegs.ADCCTL1.bit.INTPULSEPOS = 1;

	//power up the ADC
	AdcaRegs.ADCCTL1.bit.ADCPWDNZ = 1;

	//write configurations for ADC-B
	// External REFERENCE must be provided
	AdcbRegs.ADCCTL2.bit.PRESCALE = 6; //set ADCCLK divider to /4
	AdcbRegs.ADCCTL2.bit.RESOLUTION = RESOLUTION_12BIT;
	AdcbRegs.ADCCTL2.bit.SIGNALMODE = SIGNAL_SINGLE;

	//Set pulse positions to late
	AdcbRegs.ADCCTL1.bit.INTPULSEPOS = 1;

	//power up the ADC
	AdcbRegs.ADCCTL1.bit.ADCPWDNZ = 1;

	//    //write configurations for ADC-C
	// External REFERENCE must be provided
	AdccRegs.ADCCTL2.bit.PRESCALE = 6; //set ADCCLK divider to /4
	AdccRegs.ADCCTL2.bit.RESOLUTION = RESOLUTION_12BIT;
	AdccRegs.ADCCTL2.bit.SIGNALMODE = SIGNAL_SINGLE;

	//Set pulse positions to late
	AdccRegs.ADCCTL1.bit.INTPULSEPOS = 1;

	//power up the ADC
	AdccRegs.ADCCTL1.bit.ADCPWDNZ = 1;
	//
	//    //write configurations for ADC-D
	// External REFERENCE must be provided
```

```
        AdcdRegs.ADCCTL2.bit.PRESCALE = 6; //set ADCCLK divider to /4
        AdcdRegs.ADCCTL2.bit.RESOLUTION = RESOLUTION_12BIT;
        AdcdRegs.ADCCTL2.bit.SIGNALMODE = SIGNAL_SINGLE;

        //Set pulse positions to late
        AdcdRegs.ADCCTL1.bit.INTPULSEPOS = 1;

        //power up the ADC
        AdcdRegs.ADCCTL1.bit.ADCPWDNZ = 1;

        //delay for > 1ms to allow ADC time to power up
        for(i = 0; i < 1000; i++){
        asm("   RPT#255 || NOP");
        }

        EDIS;
}



//TODO setupSDFM()
void setupSDFM(uint16_t PWM_ticks, uint16_t PWM_ticks_in_sdfm_osr,  uint16_t
SD_clk_ecap_sys_ticks, uint16_t sdfmosr)
{

        // setup CAP to generate CLK for SDFM
        configureECAPforSDFMClk(SD_clk_ecap_sys_ticks);

        // setup PWM 11 for syncing up the SD filter windows with motor
control PWMs
        configurePWMsyncforSDFM(PWM_ticks,PWM_ticks_in_sdfm_osr);

        // Setup GPIO for SD
        //SD Filter 1 , iL1
        GPIO_SetupPinOptions(48, GPIO_INPUT, GPIO_SYNC);
        GPIO_SetupPinMux(48,0,7);

        GPIO_SetupPinOptions(49, GPIO_INPUT, GPIO_SYNC);
        GPIO_SetupPinMux(49,0,7);

        //SD Filter 2 , iL2
        GPIO_SetupPinOptions(50, GPIO_INPUT, GPIO_SYNC);
        GPIO_SetupPinMux(50,0,7);

        GPIO_SetupPinOptions(51, GPIO_INPUT, GPIO_SYNC);
        GPIO_SetupPinMux(51,0,7);

        //SD Filter 3 , iL3
        GPIO_SetupPinOptions(52, GPIO_INPUT, GPIO_SYNC);
        GPIO_SetupPinMux(52,0,7);

        GPIO_SetupPinOptions(53, GPIO_INPUT, GPIO_SYNC);
        GPIO_SetupPinMux(53,0,7);

        /*******************************************************/
        /* Input Control Module */
        /*******************************************************/
        //Configure Input Control Mode: Modulator Clock rate = Modulator data
rate
```

```
        Sdfm_configureInputCtrl(1,FILTER1,MODE_0);
        Sdfm_configureInputCtrl(1,FILTER2,MODE_0);
        Sdfm_configureInputCtrl(1,FILTER3,MODE_0);

        /*****************************************************/
        /* Comparator Module */
        /*****************************************************/
        // comparator module is setup in the board protection function

        /*****************************************************/
        /* Sinc filter Module */
        /*****************************************************/
        //Configure Data filter modules filter type, OSR value and enable /
disable data filter
        // 16 bit data representation is chosen for OSR 128 using Sinc3, from
the table in the TRM
        // the max value represented for OSR 128 using sinc 3 is +/-2097152
i.e. 2^21
        // the max value represented for OSR 64 using sinc 3 is +/-262144 i.e.
2^18
        // to represent this in 16 bit format where the first bit is sign
shift by 6 bits
        Sdfm_configureData_filter(1, FILTER1, FILTER_ENABLE, SINC3, sdfmosr-1,
DATA_32_BIT, SHIFT_0_BITS );
        Sdfm_configureData_filter(1, FILTER2, FILTER_ENABLE, SINC3, sdfmosr-1,
DATA_32_BIT,  SHIFT_0_BITS );
        Sdfm_configureData_filter(1, FILTER3, FILTER_ENABLE, SINC3, sdfmosr-1,
DATA_32_BIT,  SHIFT_0_BITS );

        //PWM11.CMPC, PWM11.CMPD, PWM12.CMPC and PWM12.CMPD signals cannot
synchronize the filters. This option is not being used in this example.
        Sdfm_configureExternalreset(1,FILTER_1_EXT_RESET_ENABLE,
FILTER_2_EXT_RESET_ENABLE, FILTER_3_EXT_RESET_ENABLE,
FILTER_4_EXT_RESET_ENABLE);

        /*****************************************************/
        /* Enable interrupts */
        /*****************************************************/
        //Following SDFM interrupts can be enabled / disabled using this
function.
        //    Enable / disable comparator high threshold
        //  Enable / disable comparator low threshold
        //  Enable / disable modulator clock failure
        //  Enable / disable filter acknowledge
        Sdfm_configureInterrupt(1, FILTER1, IEH_ENABLE, IEL_ENABLE,
MFIE_ENABLE, AE_ENABLE);
        Sdfm_configureInterrupt(1, FILTER2, IEH_ENABLE, IEL_ENABLE,
MFIE_ENABLE, AE_ENABLE);
        Sdfm_configureInterrupt(1, FILTER3, IEH_ENABLE, IEL_ENABLE,
MFIE_ENABLE, AE_ENABLE);

        // Enable master filter bit of the SDFM module 1
        Sdfm_enableMFE(1);
        Sdfm_enableMIE(1);
}


//TODO configureECAPforSDFMClk()
void configureECAPforSDFMClk(uint16_t SD_clk_ecap_sys_ticks)
```

```
{
	//Use CAP as source for the SD Clock
	// PWM3A -> OUTPUTXBAR3 -> GPIO4
	// PWM4A -> OUTPUTXBAR4 -> GPIO6

	EALLOW;

	OutputXbarRegs.OUTPUT1MUX0TO15CFG.bit.MUX0 = 3;     // Select ECAP1.OUT
on Mux0
	OutputXbarRegs.OUTPUT1MUXENABLE.bit.MUX0 = 1;// Enable MUX0 for
ECAP1.OUT

	GPIO_SetupPinOptions(24, GPIO_OUTPUT, GPIO_SYNC);
	GPIO_SetupPinMux(24,0,1); // set to OUTPUTXBAR1



	// Setup APWM mode on CAP1, set period and compare registers
	ECap1Regs.ECCTL2.bit.CAP_APWM = 1;     // Enable APWM mode
	ECap1Regs.CAP1 = SD_clk_ecap_sys_ticks-1;                  // Set
Period value
	ECap1Regs.CAP2 = SD_clk_ecap_sys_ticks>>1;                 // Set
Compare value
	// set next duty cycle to 50%
	ECap1Regs.CAP3 = SD_clk_ecap_sys_ticks-1;
	ECap1Regs.CAP4 = SD_clk_ecap_sys_ticks>>1;
	ECap1Regs.ECCLR.all = 0x0FF;                  // Clear pending
__interrupts
	// Start counters
	ECap1Regs.ECCTL2.bit.TSCTRSTOP = 1;

	EDIS;
}

//TODO configurePWMsyncforSDFM()
void configurePWMsyncforSDFM(uint16_t PWM_ticks, uint16_t
PWM_ticks_in_sdfm_osr)
{
	configurePWM1chUpCnt(11,PWM_ticks);

	EPwm3Regs.TBCTL.bit.SYNCOSEL=TB_SYNC_IN;
	EPwm3Regs.TBCTL.bit.PHSEN=TB_ENABLE;
	EPwm3Regs.TBPHS.bit.TBPHS=2;
	EPwm3Regs.TBCTL.bit.PHSDIR=TB_UP;

	SyncSocRegs.SYNCSELECT.bit.EPWM4SYNCIN=0;     //EPwm1SyncOut

	EPwm4Regs.TBCTL.bit.SYNCOSEL=TB_SYNC_IN;
	EPwm5Regs.TBCTL.bit.SYNCOSEL=TB_SYNC_IN;

	SyncSocRegs.SYNCSELECT.bit.EPWM10SYNCIN=0;  //EPwm1Sync Out
	EPwm10Regs.TBCTL.bit.SYNCOSEL=TB_SYNC_IN;

	EPwm11Regs.TBCTL.bit.PHSEN=TB_ENABLE;
	EPwm11Regs.TBPHS.bit.TBPHS=2;
	EPwm11Regs.TBCTL.bit.PHSDIR=TB_UP;
```

```
	//EPwm11Regs.CMPC=(EPwm11Regs.TBPRD>>1-PWM_ticks_in_sdfm_osr-
PWM_ticks_in_sdfm_osr>>1);
	//EPwm11Regs.CMPD=(EPwm11Regs.TBPRD>>1-PWM_ticks_in_sdfm_osr-
PWM_ticks_in_sdfm_osr>>1);


//	EPwm11Regs.CMPC=(EPwm11Regs.TBPRD>>1-PWM_ticks_in_sdfm_osr-
PWM_ticks_in_sdfm_osr>>1);
//	EPwm11Regs.CMPD=(EPwm11Regs.TBPRD>>1-PWM_ticks_in_sdfm_osr-
PWM_ticks_in_sdfm_osr>>1);

//	EPwm11Regs.CMPA.bit.CMPA= EPwm11Regs.TBPRD>>1 + PWM_ticks_in_sdfm_osr+
(PWM_ticks_in_sdfm_osr>>1) - 10; // 40 cycles advance so the ISR can begin
executing as soon as the data for the current conversion is ready


	EPwm11Regs.CMPC = 1750;
	EPwm11Regs.CMPD = 1750;
	EPwm11Regs.CMPA.bit.CMPA= 1250;
}



//TODO configurePWM1chUpCnt()
void configurePWM1chUpCnt(int16 n, Uint16 period)
{
	EALLOW;
	// Time Base SubModule Registers
	(*ePWM[n]).TBCTL.bit.PRDLD = TB_IMMEDIATE; // set Immediate load
	(*ePWM[n]).TBPRD = period-1; // PWM frequency = 1 / period
	(*ePWM[n]).TBPHS.bit.TBPHS = 0;
	(*ePWM[n]).TBCTR = 0;
	(*ePWM[n]).TBCTL.bit.CTRMODE = TB_COUNT_UP;
	(*ePWM[n]).TBCTL.bit.HSPCLKDIV = TB_DIV1;
	(*ePWM[n]).TBCTL.bit.CLKDIV = TB_DIV1;

	(*ePWM[n]).TBCTL.bit.PHSEN = TB_DISABLE;
	(*ePWM[n]).TBCTL.bit.SYNCOSEL = TB_CTR_ZERO; // sync "down-stream"

	// Counter Compare Submodule Registers
	(*ePWM[n]).CMPA.bit.CMPA = 0; // set duty 0% initially
	(*ePWM[n]).CMPCTL.bit.SHDWAMODE = CC_SHADOW;
	(*ePWM[n]).CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;

	// Action Qualifier SubModule Registers
	(*ePWM[n]).AQCTLA.bit.CAU = AQ_CLEAR;
	(*ePWM[n]).AQCTLA.bit.ZRO = AQ_SET;
	EDIS;
}

void setupComparatorSubsystem(uint16_t cmpss_no, float current_limit, float
current_max_sense)
{
	EALLOW;
	// LEM Current goes ADC A2/ CMPSS 1
	//Enable CMPSS1
	CMPSS[cmpss_no]->COMPCTL.bit.COMPDACE=1;

	//NEG signal comes from DAC for the low comparator
```

```
        CMPSS[cmpss_no]->COMPCTL.bit.COMPLSOURCE=NEGIN_DAC;

        //NEG signal comes from DAC for the high comparator
        CMPSS[cmpss_no]->COMPCTL.bit.COMPHSOURCE=NEGIN_DAC;

        //Use VDDA as the reference for comparator DACs
        CMPSS[cmpss_no]->COMPDACCTL.bit.SELREF= REFERENCE_VDDA;

        //Set DAC to H~75% and L ~25% values
        CMPSS[cmpss_no]-
>DACHVALS.bit.DACVAL=2048+(int16_t)((float)current_limit*(float)2048.0/(float
)current_max_sense);
        CMPSS[cmpss_no]->DACLVALS.bit.DACVAL=2048-
(int16_t)((float)current_limit*(float)2048.0/(float)current_max_sense);

        // comparator oputput is "not" inverted for high compare event
        CMPSS[cmpss_no]->COMPCTL.bit.COMPHINV=0;

        // Comparator output is inverted for for low compare event
        CMPSS[cmpss_no]->COMPCTL.bit.COMPLINV=1;

        // Configure Digital Filter
        //Maximum CLKPRESCALE value provides the most time between samples
        CMPSS[cmpss_no]->CTRIPHFILCLKCTL.bit.CLKPRESCALE = 0x10;

        //Maximum SAMPWIN value provides largest number of samples
        CMPSS[cmpss_no]->CTRIPHFILCTL.bit.SAMPWIN        = 0x1F;

        //Maximum THRESH value requires static value for entire window
        //  THRESH should be GREATER than half of SAMPWIN
        CMPSS[cmpss_no]->CTRIPHFILCTL.bit.THRESH         = 0x0F;

        //Reset filter logic & start filtering
        CMPSS[cmpss_no]->CTRIPHFILCTL.bit.FILINIT        = 1;

        // Configure CTRIPOUT path
        //Digital filter output feeds CTRIPH and CTRIPOUTH
        CMPSS[cmpss_no]->COMPCTL.bit.CTRIPHSEL           = CTRIP_FILTER;
        CMPSS[cmpss_no]->COMPCTL.bit.CTRIPOUTHSEL        = CTRIP_FILTER;
        CMPSS[cmpss_no]->COMPCTL.bit.CTRIPLSEL           = CTRIP_FILTER;
        CMPSS[cmpss_no]->COMPCTL.bit.CTRIPOUTLSEL        = CTRIP_FILTER;

        // Make sure the asynchronous path compare high and low event
        // does not go to the OR gate with latched digital filter output
        CMPSS[cmpss_no]->COMPCTL.bit.ASYNCHEN=0;
        CMPSS[cmpss_no]->COMPCTL.bit.ASYNCLEN=0;

        //Comparator hysteresis control , set to 2x typical value
        CMPSS[cmpss_no]->COMPHYSCTL.bit.COMPHYS=2;
        // Dac value is updated on sysclock
        CMPSS[cmpss_no]->COMPDACCTL.bit.SWLOADSEL=0;
        // ramp is bypassed
        CMPSS[cmpss_no]->COMPDACCTL.bit.DACSOURCE=0;

        // Clear the latched comparator events
        CMPSS[cmpss_no]->COMPSTSCLR.bit.HLATCHCLR=1;
        CMPSS[cmpss_no]->COMPSTSCLR.bit.LLATCHCLR=1;

        EDIS;
```

```
}

void setupPWMTrip(uint16_t pwm_no )
{
	// Now enable TRIP 4 to trip the PWM of the vienna rectifier
	EALLOW;

	// TRIP4 is the source for DCAEVT1
	// TRIP5 is the source for DCAEVT2

	(*ePWM[pwm_no]).DCTRIPSEL.bit.DCAHCOMPSEL=3; //Trip 4 is the input to
the DCAHCOMPSEL
	(*ePWM[pwm_no]).TZDCSEL.bit.DCAEVT1=TZ_DCAH_HI;
	(*ePWM[pwm_no]).DCACTL.bit.EVT1SRCSEL= 0;
	(*ePWM[pwm_no]).DCACTL.bit.EVT1FRCSYNCSEL=DC_EVT_SYNC  ; // this
forces scnchronization before accepting the trip
//	(*ePWM[pwm_no]).TZSEL.bit.DCAEVT1=1;
	(*ePWM[pwm_no]).TZCTL.bit.DCAEVT1 = TZ_NO_CHANGE;


	(*ePWM[pwm_no]).DCTRIPSEL.bit.DCBHCOMPSEL=4; //Trip 5 is the input to
the DCBHCOMPSEL
	(*ePWM[pwm_no]).TZDCSEL.bit.DCBEVT1=TZ_DCBH_HI;
	(*ePWM[pwm_no]).DCBCTL.bit.EVT1SRCSEL= 0;
	(*ePWM[pwm_no]).DCBCTL.bit.EVT1FRCSYNCSEL=DC_EVT_SYNC  ; //this forces
synchronization before accepting the trip
	(*ePWM[pwm_no]).TZSEL.bit.DCBEVT1=1;
//	(*ePWM[pwm_no]).TZCTL.bit.DCBEVT1 = TZ_NO_CHANGE; // EPWMxB will go
low

	(*ePWM[pwm_no]).TZSEL.bit.CBC6=0x1; // Emulator Stop

	// What do we want the OST/CBC events to do?
	// TZA events can force EPWMxA
	// TZB events can force EPWMxB

	(*ePWM[pwm_no]).TZCTL.bit.TZA = TZ_FORCE_LO; // EPWMxA will go low
	(*ePWM[pwm_no]).TZCTL.bit.TZB = TZ_FORCE_LO; // EPWMxB will go low

	// Clear any spurious trip
	(*ePWM[pwm_no]).TZCLR.bit.DCBEVT1=1;
	(*ePWM[pwm_no]).TZCLR.bit.DCAEVT1=1;
	(*ePWM[pwm_no]).TZCLR.bit.OST = 1;


	(*ePWM[pwm_no]).TZEINT.bit.DCAEVT1=1;
	//(*ePWM[pwm_no]).TZEINT.bit.DCBEVT1=1;
	// force a PWM Trip
	(*ePWM[pwm_no]).TZFRC.bit.OST=1;

	EDIS;
}

//TODO setupBoardProtection()
void setupBoardProtection(uint16_t current_sense, float current_limit, float
current_max_sense)
{
	EALLOW;
	// configure TRIP4 for SDFM
```

```
        // Reset the MUX CFG
        EPwmXbarRegs.TRIP4MUX0TO15CFG.all=0x0000;
        EPwmXbarRegs.TRIP4MUX16TO31CFG.all=0x0000;
        // Disable all the muxes first
        EPwmXbarRegs.TRIP4MUXENABLE.all=0x0000;
        EALLOW;

        Sdfm_clearFlagRegister(1,0x8000FFFF);

        Sdfm_configureComparator ( 1, FILTER1, SINC3, OSR_32,

        ((32768/2)+(int16_t)((float)current_limit*(float)16384/(float)I_MAX_SE
NSE_SD)),
                        ((32768/2)-
(int16_t)((float)current_limit*(float)16384/(float)I_MAX_SENSE_SD)));

        Sdfm_configureComparator(1, FILTER2, SINC3, OSR_32,

        ((32768/2)+(int16_t)((float)current_limit*(float)16384/(float)I_MAX_SE
NSE_SD)),
                        ((32768/2)-
(int16_t)((float)current_limit*(float)16384/(float)I_MAX_SENSE_SD)));

        Sdfm_configureComparator(1, FILTER3, SINC3, OSR_32,

        ((32768/2)+(int16_t)((float)current_limit*(float)16384/(float)I_MAX_SE
NSE_SD)),
                        ((32768/2)-
(int16_t)((float)current_limit*(float)16384/(float)I_MAX_SENSE_SD)));

        Sdfm_clearFlagRegister(1,0x8000FFFF);

        EALLOW;
        EPwmXbarRegs.TRIP4MUX16TO31CFG.bit.MUX16=1; // select the 2nd option
of MUx 16
        EPwmXbarRegs.TRIP4MUX16TO31CFG.bit.MUX18=1; // select the 2nd option
of MUx 18
        EPwmXbarRegs.TRIP4MUX16TO31CFG.bit.MUX20=1; // select the 2nd option
of MUx 20

        // Enable Mux 16,18 and 20, option 2 SD1FLT1/2/3.COMPH_OR_COMPL to
generate TRIP4
        EPwmXbarRegs.TRIP4MUXENABLE.bit.MUX16=1;
        EPwmXbarRegs.TRIP4MUXENABLE.bit.MUX18=1;
        EPwmXbarRegs.TRIP4MUXENABLE.bit.MUX20=1;
        EDIS;



        // configure TRIP5 for ADC comparator based trip

        // Reset the MUX CFG
        EPwmXbarRegs.TRIP5MUX0TO15CFG.all=0x0000;
        EPwmXbarRegs.TRIP5MUX16TO31CFG.all=0x0000;
        // Disable all the muxes first
        EPwmXbarRegs.TRIP5MUXENABLE.all=0x0000;
        EDIS;
```

```
        setupComparatorSubsystem(1, (current_limit), current_max_sense);
        setupComparatorSubsystem(2, (current_limit), current_max_sense);
        setupComparatorSubsystem(4, (current_limit), current_max_sense);

        EALLOW;

        // see TRM ePWM X-Bar MUX Configuration Table
        // Enable Mux 0,option 2 CMPSS1.CTRIPH_OR_CTRIPL to generate TRIP5
        EPwmXbarRegs.TRIP5MUX0TO15CFG.bit.MUX0=1;
        EPwmXbarRegs.TRIP5MUXENABLE.bit.MUX0=1;

        // Enable Mux 2,option 2 CMPSS2.CTRIPH_OR_CTRIPL to generate TRIP5
        EPwmXbarRegs.TRIP5MUX0TO15CFG.bit.MUX2=1;
        EPwmXbarRegs.TRIP5MUXENABLE.bit.MUX2=1;

        // Enable Mux 6,option 2 CMPSS4.CTRIPH_OR_CTRIPL to generate TRIP5
        EPwmXbarRegs.TRIP5MUX0TO15CFG.bit.MUX6=1;
        EPwmXbarRegs.TRIP5MUXENABLE.bit.MUX6=1;

        EDIS;



        setupPWMTrip(1);
        setupPWMTrip(2);
        setupPWMTrip(3);


//************************* End of Prot. Conf. ***************************//
}



//TODO calibrateOffset()
void calibrateOffset(volatile float *iL1MeasOffset, volatile float
*iL2MeasOffset, volatile float *iL3MeasOffset ,
             volatile float *v1MeasOffset, volatile float *v2MeasOffset,
volatile float *v3MeasOffset ,
             float k1, float k2)
{
        int16_t offsetCalCounter=0;

        EALLOW;

        EPwm1Regs.ETSEL.bit.SOCASEL = ET_CTRU_CMPA; // select CMPC when
counting up to trigger the ADC
        EPwm1Regs.ETSEL.bit.SOCASELCMP = 1; // enable the CMPC and CMPD Pulses
for the ADC trigger
        EPwm1Regs.CMPC= EPwm1Regs.TBPRD - (ACQPS_SYS_CLKS * 6);
        EPwm1Regs.ETPS.bit.SOCAPRD = ET_1ST; /* Generate pulse on 1st even*/
        EPwm1Regs.ETSEL.bit.SOCAEN = 1; /* Enable SOC on A group*/

        //PWM1 INT is used to trigger the ISR
        EPwm1Regs.ETSEL.bit.INTSEL = ET_CTRD_CMPB;    // INT on Zero event
        EPwm1Regs.CMPB.bit.CMPB= EPwm1Regs.TBPRD - (ACQPS_SYS_CLKS <<1);
        EPwm1Regs.ETSEL.bit.INTEN = 1;                // Enable INT
        EPwm1Regs.ETPS.bit.INTPRD = ET_1ST;           // Generate INT on every
event
```

```
	EPwm1Regs.ETCLR.bit.INT=1;

	EDIS;

	offsetCalCounter=0;
	*iL1MeasOffset=0;
	*iL2MeasOffset=0;
	*iL3MeasOffset=0;

	*v1MeasOffset=0;
	*v2MeasOffset=0;
	*v3MeasOffset=0;

	while(offsetCalCounter<25000)
	{
		if(EPwm1Regs.ETFLG.bit.INT==1)
		{
			if(offsetCalCounter>1000)
			{
				// offset of the inductor current sense
				*iL1MeasOffset = k1*(*iL1MeasOffset) +
k2*(IL1_ADC_READ1+IL1_ADC_READ2+IL1_ADC_READ3+IL1_ADC_READ4)*0.25*ADC_PU_SCAL
E_FACTOR;
				*iL2MeasOffset = k1*(*iL2MeasOffset) +
k2*(IL2_ADC_READ1+IL2_ADC_READ2+IL2_ADC_READ3+IL2_ADC_READ4)*0.25*ADC_PU_SCAL
E_FACTOR;
				*iL3MeasOffset = k1*(*iL3MeasOffset) +
k2*(IL3_ADC_READ1+IL3_ADC_READ2+IL3_ADC_READ3+IL3_ADC_READ4)*0.25*ADC_PU_SCAL
E_FACTOR;

				// offset of the inductor current sense
				*v1MeasOffset = k1*(*v1MeasOffset) +
k2*(V1_ADC_READ1+V1_ADC_READ2+V1_ADC_READ3+V1_ADC_READ4)*0.25*ADC_PU_SCALE_FA
CTOR;
				*v2MeasOffset = k1*(*v2MeasOffset) +
k2*(V2_ADC_READ1+V2_ADC_READ2+V2_ADC_READ3+V2_ADC_READ4)*0.25*ADC_PU_SCALE_FA
CTOR;
				*v3MeasOffset = k1*(*v3MeasOffset) +
k2*(V3_ADC_READ1+V3_ADC_READ2+V3_ADC_READ3+V3_ADC_READ4)*0.25*ADC_PU_SCALE_FA
CTOR;
			}
			EPwm1Regs.ETCLR.bit.INT=1;
			offsetCalCounter++;
		}
	}


// NO use using the PPB because of the over sampling

}

//TODO disablePWMCLKCounting
void disablePWMCLKCounting(void)
{
	EALLOW;
	CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 0;
	EDIS;
}
```

```
//TODO enablePWMCLKCounting
void enablePWMCLKCounting(void)
{
	EALLOW;
	CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 1;
	EDIS;
}

//TODO setupLEDGPIO()
void setupLEDGPIO(void)
{
	GPIO_SetupPinOptions(31, GPIO_OUTPUT, GPIO_ASYNC);
	GPIO_SetupPinMux(31,0,0);

	GPIO_SetupPinOptions(34, GPIO_OUTPUT, GPIO_ASYNC);
	GPIO_SetupPinMux(34,0,0);
}

//TODO toggleLED()
void toggleLED(void)
{
	static uint16_t ledCnt1=0;

	if(ledCnt1==0)
		{
		GpioDataRegs.GPBTOGGLE.bit.GPIO34=1;
		ledCnt1=10;
		}
	else
		ledCnt1--;
}

//TODO setupProfilingGPIO()
void setupProfilingGPIO(void)
{	//To profile use GPIO 40 and GPIO41
	GPIO_SetupPinOptions(12, GPIO_OUTPUT, GPIO_ASYNC);
	GPIO_SetupPinMux(12,0,0);

	GPIO_SetupPinOptions(13, GPIO_OUTPUT, GPIO_ASYNC);
	GPIO_SetupPinMux(13,0,0);
}

//TODO setupRelayGPIO();
void setupRelayGPIO(void)
{
	// Configure GPIO 5 as Output, this goes to the Relay
	GPIO_SetupPinOptions(5,GPIO_OUTPUT, GPIO_ASYNC);
	GPIO_SetupPinMux(5,0,0);;
}

//TODO setupSCIconnectionForSFRA()
void setupSCIconnectionForSFRA()
{

	// Use Gpio 28 and 29 for the SCI comms through JTAG
	EALLOW;
	GPIO_SetupPinMux(28, GPIO_MUX_CPU1, 1);
	GPIO_SetupPinOptions(28, GPIO_INPUT, GPIO_SYNC);
```

```
	GPIO_SetupPinMux(29, GPIO_MUX_CPU1, 1);
	GPIO_SetupPinOptions(29, GPIO_OUTPUT, GPIO_SYNC);
	EDIS;

	// 50000000 is the LSPCLK or the Clock used for the SCI Module
	// 57600 is the Baudrate desired of the SCI module
	SCIA_Init(50000000, 57600);
}

void configure3phViennaPWM(uint16_t inv_pwm_no, uint16_t pwm_period_ticks)
{
	EALLOW;
	// PWM clock on F2837x is divided by 2
	// ClkCfgRegs.PERCLKDIVSEL.bit.EPWMCLKDIV=1
	configurePWMUpDwnCnt(inv_pwm_no,pwm_period_ticks);
	configurePWMUpDwnCnt(inv_pwm_no+1,pwm_period_ticks);
	configurePWMUpDwnCnt(inv_pwm_no+2,pwm_period_ticks);

	(*ePWM[inv_pwm_no]).TBCTL.bit.PHSEN = TB_DISABLE;
	(*ePWM[inv_pwm_no]).TBCTL.bit.SYNCOSEL = TB_CTR_ZERO; // sync "down-
stream"

	// configure next PWM  as slaves and let it pass the sync in pulse
from PWM1
	ePWM[inv_pwm_no+1]->TBCTL.bit.SYNCOSEL=TB_SYNC_IN;
	ePWM[inv_pwm_no+1]->TBCTL.bit.PHSEN=TB_ENABLE;
	ePWM[inv_pwm_no+1]->TBPHS.bit.TBPHS=2;
	ePWM[inv_pwm_no+1]->TBCTL.bit.PHSDIR=TB_UP;

	ePWM[inv_pwm_no+2]->TBCTL.bit.SYNCOSEL=TB_SYNC_IN;
	ePWM[inv_pwm_no+2]->TBCTL.bit.PHSEN=TB_ENABLE;
	ePWM[inv_pwm_no+2]->TBPHS.bit.TBPHS=2;
	ePWM[inv_pwm_no+2]->TBCTL.bit.PHSDIR=TB_UP;

	EDIS;
}


//TODO configurePWM1chUpDwnCnt()
void configurePWMUpDwnCnt(uint16_t inv_pwm_no, uint16_t pwm_period_ticks)
{
	EALLOW;

	// Time Base SubModule Registers
	(*ePWM[inv_pwm_no]).TBCTL.bit.PRDLD = TB_SHADOW;
	(*ePWM[inv_pwm_no]).TBPRD = pwm_period_ticks >>1 ; // PWM frequency =
1 / period
	(*ePWM[inv_pwm_no]).TBPHS.bit.TBPHS = 0;
	(*ePWM[inv_pwm_no]).TBCTR = 0;
	(*ePWM[inv_pwm_no]).TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;
	(*ePWM[inv_pwm_no]).TBCTL.bit.HSPCLKDIV = TB_DIV1;
	(*ePWM[inv_pwm_no]).TBCTL.bit.CLKDIV = TB_DIV1;

	// Counter Compare Submodule Registers
	(*ePWM[inv_pwm_no]).CMPA.bit.CMPA =    (*ePWM[inv_pwm_no]).TBPRD ; //
set duty 0% initially
	(*ePWM[inv_pwm_no]).CMPCTL.bit.SHDWAMODE = CC_SHADOW;
	(*ePWM[inv_pwm_no]).CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
```

```
      // Action Qualifier SubModule Registers
      // assuming positive half cycle comes first i.e. dhl>0
      (*ePWM[inv_pwm_no]).AQCTLA.bit.CAU = AQ_SET; // CTR = CMPA@UP , set to
1
      (*ePWM[inv_pwm_no]).AQCTLA.bit.CAD = AQ_CLEAR; // CTR = CMPA@Down ,
toggle
      (*ePWM[inv_pwm_no]).AQCTLA.bit.ZRO = AQ_CLEAR; // CTR=0, clear to 0

      (*ePWM[inv_pwm_no]).AQCTLB.bit.CAU = AQ_SET; // CTR = CMPA@UP , set to
1
      (*ePWM[inv_pwm_no]).AQCTLB.bit.CAD = AQ_CLEAR; // CTR = CMPA@Down ,
toggle
      (*ePWM[inv_pwm_no]).AQCTLB.bit.ZRO = AQ_CLEAR; // CTR=0, clear to 0

      EDIS;
}
```

A2.2 pfc3phvienna_board.h

```
/*
=================================================================================
=
System Name:

File Name:          <BoardName>_board.h

Target:

Author:

Description: This file consits of common variables for a particular hardware
board
                          Like variables and functions to read current and
voltage signals on the board
                          functions to setup the basic peripherals of the
board
                          These common set is independent of the control
method used and hence helps
                          in eliminating duplication of this code when
implementing different
                          control schemes on the same design
                          This file must be settings independent, an settings
dependent code should reside
                          in the parent solution project.

Copyright (C) {2015} Texas Instruments Incorporated - http://www.ti.com/
 * ALL RIGHTS RESERVED*
=================================================================================
====  */
#ifndef BOARD_H
#define BOARD_H

#ifdef __cplusplus

extern "C" {
#endif


//*************************************************************************
**
// the includes
//*************************************************************************
**

// Include the IQmath Library First, define Global Q
#define MATH_TYPE 1
#include "IQmathLib.h"

// Include files for device support, F2806x in this case
#include "F2837xD_Device.h"        // F2837xD Headerfile Include File
#include "F2837xD_Examples.h"    // F2837xD Examples Include File
#include "F2837xD_sdfm_drivers.h"

#include "pfc3phvienna_settings.h"

//*************************************************************************
**
```

```
//defines
//*********************************************************************
**
// Timer definitions based on System Clock
// 150 MHz devices
#define            mSec0_5            0.5*SYSTEM_FREQUENCY*1000       // 0.5
mS
#define            mSec1        1*SYSTEM_FREQUENCY*1000         // 1.0 mS
#define            mSec2        2.0*SYSTEM_FREQUENCY*1000       // 2.0 mS
#define            mSec5        5*SYSTEM_FREQUENCY*1000         // 5.0 mS
#define            mSec7_5            7.5*SYSTEM_FREQUENCY*1000       // 7.5
mS
#define            mSec10       10*SYSTEM_FREQUENCY*1000        // 10 mS
#define            mSec20       20*SYSTEM_FREQUENCY*1000        // 20 mS
#define            mSec50       50*SYSTEM_FREQUENCY*1000        // 50 mS
#define            mSec100            100*SYSTEM_FREQUENCY*1000       // 100
mS
#define            mSec500            500*SYSTEM_FREQUENCY*1000 // 500 mS
#define            mSec1000     1000*SYSTEM_FREQUENCY*1000      // 1000 mS

#ifndef TRUE
#define FALSE 0
#define TRUE  1
#endif

#define HistorySize 8
//#define DLOG_SIZE   200

#define ADC_PU_SCALE_FACTOR  (float32)(0.000244140625)
#define HI_RES_SHIFT (float32)(65536.0)
#define ADC_PU_PPB_SCALE_FACTOR 0.000488281250 //1/2^11
#define SD_PU_SCALE_FACTOR  0.000030517578125 // 1/2^15


//definitions for selecting DACH reference
#define REFERENCE_VDDA     0

//definitions for COMPH input selection
#define NEGIN_DAC          0
#define NEGIN_PIN          1

//definitions for CTRIPH/CTRIPOUTH output selection
#define CTRIP_ASYNCH       0
#define CTRIP_SYNCH        1
#define CTRIP_FILTER       2
#define CTRIP_LATCH        3

// ADC Configuration
#define REFERENCE_INTERNAL 0 //internal reference (12-bit only)
#define REFERENCE_EXTERNAL 1 //external reference
//definitions for selecting ADC resolution
#define RESOLUTION_12BIT 0 //12-bit resolution
#define RESOLUTION_16BIT 1 //16-bit resolution (not supported for all
variants)
//definitions for selecting ADC signal mode
#define SIGNAL_SINGLE 0 //single-ended channel conversions (12-bit mode only)
#define SIGNAL_DIFFERENTIAL 1 //differential pair channel conversions

#define ACQPS_SYS_CLKS       16
```

```
#define REFERENCE_VDAC     0
#define REFERENCE_VREF     1

#define HWREG32(x) (*((volatile Uint32 *)(x)))
#define HWREG16(x) (*((volatile Uint16 *)(x)))


//*****************************************************************************
**
//globals
//*****************************************************************************
**

extern volatile struct EPWM_REGS *ePWM[];
extern volatile struct SDFM_REGS *SDFM[];
extern volatile struct ADC_REGS *ADC[];

//*****************************************************************************
**
// the function prototypes
//*****************************************************************************
**

void setupDevice(void);
void configure3phViennaPWM(uint16_t inv_pwm_no, uint16_t pwm_period_ticks);
void configurePWMUpDwnCnt(uint16_t inv_pwm_no, uint16_t pwm_period_ticks);
void setupADC(void);
void setupPWMTrip(uint16_t pwm_no );
void setupSDFM(uint16_t PWM_ticks, uint16_t PWM_ticks_in_sdfm_osr, uint16_t
SD_clk_ecap_sys_ticks, uint16_t sdfmosr);
void configureECAPforSDFMClk(uint16_t SD_clk_ecap_sys_ticks);
void configurePWMsyncforSDFM(uint16_t PWM_ticks, uint16_t
PWM_ticks_in_sdfm_osr);
void configurePWM1chUpDwnCnt(int16_t n, int16_t period, int16_t mode, int16_t
phase, int16_t db_red, int16_t db_fed,int16_t phase_dir);
void configurePWM1chUpCnt(int16_t n, uint16_t period);
void setupBoardProtection(uint16_t current_sense, float current_limit, float
current_max_sense);
void initGlobalVariables(void);
void calibrateOffset(volatile float *iL1MeasOffset, volatile float
*iL2MeasOffset, volatile float *iL3MeasOffset ,
            volatile float *v1MeasOffset, volatile float *v2MeasOffset,
volatile float *v3MeasOffset ,
            float k1, float k2);
void setupProfilingGPIO(void);
void setupRelayGPIO(void);
void toggleLED(void);
void setupLEDGPIO(void);
void disablePWMCLKCounting(void);
void enablePWMCLKCounting(void);
void setupSCIconnectionForSFRA(void);
void updateBoardStatus(void);
inline void closeRelay(void);
inline void openRelay(void);
inline void setProfilingGPIO(void);
inline void resetProfilingGPIO(void);
inline void readCurrVolADCSignals(void);
inline void readCurrVolSDFMSignals(float sdfm_scale_factor);
```

```
void SCIA_Init(long SCI_VBUS_CLOCKRATE, long SCI_BAUDRATE);
void DeviceInit(void);
void MemCopy(void);
void InitFlash(void);
void SerialHostComms(void);
interrupt void controlISR(void);
interrupt void tenkHzISR(void);

// Inline functions

//TODO updateInverterPWM()
inline void updatePFCviennaPWM(float duty1PU, float duty2PU, float duty3PU,
uint16_t inv_pwm_no)
{
	float pwm_period;
	uint16_t duty1, duty2, duty3;

	pwm_period= (*ePWM[inv_pwm_no]).TBPRD;

	duty1=  (uint16_t) ( (float) pwm_period * (float) fabs(duty1PU) );
	duty2=  (uint16_t) ( (float) pwm_period * (float) fabs(duty2PU) );
	duty3=  (uint16_t) ( (float) pwm_period * (float) fabs(duty3PU) );

	(*ePWM[inv_pwm_no]).CMPA.bit.CMPA = duty1;

	(*ePWM[inv_pwm_no+1]).CMPA.bit.CMPA = duty2;

	(*ePWM[inv_pwm_no+2]).CMPA.bit.CMPA = duty3;

}

//TODO closeRelay
inline void closeRelay(void)
{
	GpioDataRegs.GPASET.bit.GPIO5=1;
}

//TODO openRelay
inline void openRelay(void)
{
	GpioDataRegs.GPACLEAR.bit.GPIO5=1;
//	invVoRef=0;
//	invIiRef=0;
}

//TODO setProfilingGPIO
inline void setProfilingGPIO(void)
{
	GpioDataRegs.GPASET.bit.GPIO12 = 1;
}

//TODO resetProfilingGPIO
inline void resetProfilingGPIO(void)
{
	GpioDataRegs.GPACLEAR.bit.GPIO12=1;
}

//TODO setProfilingGPIO
inline void setProfilingGPIO2(void)
```

```
{
	GpioDataRegs.GPASET.bit.GPIO13 = 1;
}

//TODO resetProfilingGPIO
inline void resetProfilingGPIO2(void)
{
	GpioDataRegs.GPACLEAR.bit.GPIO13=1;
}

#define SDFM_READ_FILTER1_DATA_32BIT   *(int32_t *)0x5E16
#define SDFM_READ_FILTER2_DATA_32BIT   *(int32_t *)0x5E26
#define SDFM_READ_FILTER3_DATA_32BIT   *(int32_t *)0x5E36
#define SDFM_READ_FILTER4_DATA_32BIT   *(int32_t *)0x5E46




//TODO setupADCconversion()
inline void setupADCconversion(uint16_t adc_module_no, uint16_t
adc_soc_offset, uint16_t adc_channel, uint16_t trig_sel, uint16_t acqps)
{
	EALLOW;
	*( ( (uint32_t *) ADC[adc_module_no] ) +
(uint32_t)(adc_soc_offset>>1)) = ((uint32_t)trig_sel <<20) +
((uint32_t)adc_channel <<15) + acqps ;
	EDIS;
}


//TODO clearPWMTripFlags()
inline void clearPWMTripFlags(uint16_t pwm_no)
{
		// clear all the configured trip sources for the PWM module
		EALLOW;
		(*ePWM[pwm_no]).TZCLR.bit.OST=0x1;
		(*ePWM[pwm_no]).TZCLR.bit.CBC=0x1;
		(*ePWM[pwm_no]).TZCLR.bit.DCAEVT1=0x1;
		(*ePWM[pwm_no]).TZCLR.bit.DCBEVT1=0x1;
		(*ePWM[pwm_no]).TZCLR.bit.INT=0x1;
		EDIS;
}

//TODO clearInterrupt
inline void clearInterrupt(void)
{
#if SENSING_OPTION ==SDFM_BASED_SENSING
    EPwm11Regs.ETCLR.bit.INT=1;
#elif SENSING_OPTION ==ADC_BASED_SENSING
    EPwm1Regs.ETCLR.bit.INT=1;
    EPwm11Regs.ETCLR.bit.INT=1;
#endif
    PieCtrlRegs.PIEACK.all=PIEACK_GROUP3;
}

inline void setupInterrupt(void)
{
#if SENSING_OPTION ==SDFM_BASED_SENSING
```

```
        EALLOW;
        //PWM11 INT is used to trigger the ISR
        PieVectTable.EPWM11_INT = &inverterISR;
        EPwm11Regs.ETSEL.bit.INTSEL = ET_CTRU_CMPA;  // INT on PRD event
        EPwm11Regs.ETSEL.bit.INTEN = 1;              // Enable INT
        EPwm11Regs.ETPS.bit.INTPRD = CNTRL_ISR_FREQ_RATIO;
        clearInterrupt();

        PieCtrlRegs.PIEIER3.bit.INTx11 = 1; // Enable PWM11INT in PIE group 3
        EPwm11Regs.ETCLR.bit.INT=1;

        IER |= M_INT3; //Enable group 3 interrupts
#else
        /*EALLOW;
        //PWM1 INT is used to trigger the ISR
        PieVectTable.EPWM1_INT = &inverterISR;
        EPwm1Regs.ETSEL.bit.INTSEL = ET_CTR_PRD;     // INT on PRD event
        EPwm1Regs.ETSEL.bit.INTEN = 1;              // Enable INT
        EPwm1Regs.ETPS.bit.INTPRD = CNTRL_ISR_FREQ_RATIO;         // Generate
INT on every event
        clearInterrupt();

        PieCtrlRegs.PIEIER3.bit.INTx1 = 1; // Enable PWM11INT in PIE group 3
        EPwm1Regs.ETCLR.bit.INT=1;

        IER |= M_INT3; //Enable group 3 interrupts
        */
        EALLOW;
        //PWM11 INT is used to trigger the ISR
        PieVectTable.EPWM11_INT = &controlISR;
        EPwm11Regs.ETSEL.bit.INTSEL = ET_CTRU_CMPA;  // INT on PRD event
        EPwm11Regs.ETSEL.bit.INTEN = 1;              // Enable INT
        EPwm11Regs.ETPS.bit.INTPRD = CNTRL_ISR_FREQ_RATIO;

        PieVectTable.TIMER2_INT = &tenkHzISR;        // 1kHz interrupt from
CPU timer 2
        ConfigCpuTimer(&CpuTimer2, 200, 100);
        CpuTimer2Regs.TCR.all = 0x4001; // Use write-only instruction to set
TSS bit = 0
        clearInterrupt();
        PieCtrlRegs.PIEIER3.bit.INTx11 = 1; // Enable PWM11INT in PIE group 3
        EPwm11Regs.ETCLR.bit.INT=1;

        IER |= M_INT3; //Enable group 3 interrupts
        IER |= M_INT14;                              // CPU timer 2 is
connected to CPU INT 14

#endif

        EINT;  // Enable Global interrupt INTM
        ERTM;  // Enable Global realtime interrupt DBGM
        EDIS;
}
#ifdef __cplusplus
}
#endif                                  /* extern "C" */
#endif
```

A2.3 pfc3phvienna_settings.h

```
/*
=============================================================================
=
System Name:

File Name:         {ProjectName}-Settings.h

Target:

Author: Manish Bhardwaj, C2000 Systems Solutions Group

Description:

Copyright (C) {2015} Texas Instruments Incorporated - http://www.ti.com/
 * ALL RIGHTS RESERVED*
=============================================================================
====  */
#ifndef _PROJSETTINGS_H
#define _PROJSETTINGS_H

#ifdef __cplusplus

extern "C" {
#endif

//*****************************************************************************
**
//defines
//*****************************************************************************
**

/* Device Related Defines */
#define CPU_SYS_CLOCK (200*1000000)
#define PWMSYSCLOCK_FREQ (100*1000000)
#define ECAPSYSCLOCK_FREQ (200*1000000)

/* Project Options*/
//=============================================================================
=======
// Incremental Build options for System check-out
//=============================================================================
=======
// BUILD 1    Open Loop Check
// BUILD 2   Closed Current Loop Check
// BUILD 3   Voltage loop with inner current loop check
// BUILD 4   Balance loop with voltage and current loop closed
#define INCR_BUILD 3

// Check system under DC condition (cleanest SFRA curves) 0 is FALSE, 1 is
TRUE
#define DC_CHECK 0
// Select sensing option, 1 for ADC, 2 for SD
#define ADC_BASED_SENSING 1
#define SDFM_BASED_SENSING 2
#define SENSING_OPTION          1
#define AC_FREQ                 60

#if INCR_BUILD==4
```

```
#define INDUCTOR_VOLTAGE_DROP_FEEDFORWARD 1
#define THIRD_HARMONIC_INJECTION 1
#else
#define INDUCTOR_VOLTAGE_DROP_FEEDFORWARD 0
#define THIRD_HARMONIC_INJECTION 0
#endif

#define NON_LINEAR_VOLTAGE_LOOP 0

/* Power Stage Related Values*/
#define PFC3PH_PWM_SWITCHING_FREQUENCY ((float)50*1000)
#define PFC3PH_PWM_PERIOD (PWMSYSCLOCK_FREQ)/(PFC3PH_PWM_SWITCHING_FREQUENCY)

#define VAC_MAX_SENSE 454
#define VDCBUS_MAX_SENSE 717.79
#define V_MAX_SENSE (454)

#define I_MAX_SENSE 12
#define I_TRIP_LIMIT 11
#define I_MAX_SENSE_SD 15.625
#define VAC_TYPICAL 208

/* Control Loop Design */
#define      CNTRL_ISR_FREQ_RATIO      1
#define VOLTAGE_LOOP_RUN_RATIO  1

#define ISR_CONTROL_FREQUENCY
(PFC3PH_PWM_SWITCHING_FREQUENCY)/(CNTRL_ISR_FREQ_RATIO)
#define ISR_10KHZ_FREQUENCY (10000)/(CNTRL_ISR_FREQ_RATIO)

#define PFC_INDUCTOR_VALUE 0.003 // 3mH

//SFRA Options
//1 FRA for the Current Loop
//2 FRA for the Voltage Loop
//3 FRA for the Voltage Balance Loop
#define SFRA_CURRENT 1
#define SFRA_VOLTAGE 2
#define SFRA_BALANCECNTL 3
#if INCR_BUILD ==4
#define SFRA_TYPE               SFRA_BALANCECNTL
#elif INCR_BUILD==3
#define SFRA_TYPE               SFRA_VOLTAGE
#else
#define SFRA_TYPE               SFRA_CURRENT
#endif
#define SFRA_ISR_FREQ           ISR_CONTROL_FREQUENCY

#define PI 3.141592653589

#define GI_GAINKP 2.0000000000

#define CNTL_GV_A1 1.0000000000
#define CNTL_GV_A2 0.0000000000
#define CNTL_GV_B0 0.4000400000
#define CNTL_GV_B1 -0.3999600000
#define CNTL_GV_B2 0.0000000000
#define CNTL_GV_IMIN -0.25
#define CNTL_GV_MAX 0.9
```

```
#define CNTL_GV_MIN -0.25

#define GS_GAINKP 1.0000000000

/* SDFM Sensing Parameters*/

#define SDCLK_FREQ        (float)(20*1000000)
#define SD_CLK_COUNT ECAPSYSCLOCK_FREQ/SDCLK_FREQ
#define PWM_CLK_IN_SDFM_OSR
(int)(((float)PWMSYSCLOCK_FREQ/(float)SDCLK_FREQ)*(float)SDFM_OSR)
#define SDFM_FILTER_TYPE  3
#define SDFM_OSR                100


/* PWM pin, ADC, SDFM, Relay Selection related variables */
#define PWM_NO                  1
#define ADC_TRIG_SOURCE         5

#define ADC_TRIG_SOURCE         5

#define IL1_ADC_MODULE    1
#define IL1_ADC_PIN    2
#define IL1_CMPSS_NO      1
#define IL1_ADC_TRIG_SOURCE ADC_TRIG_SOURCE
#define IL1_ACQPS_SYS_CLKS 50
#define IL1_ADC_SOC1 0x10
#define IL1_ADC_READ1 AdcaResultRegs.ADCRESULT0
#define IL1_ADC_SOC2 0x16
#define IL1_ADC_READ2 AdcaResultRegs.ADCRESULT3
#define IL1_ADC_SOC3 0x1C
#define IL1_ADC_READ3 AdcaResultRegs.ADCRESULT6
#define IL1_ADC_SOC4 0x22
#define IL1_ADC_READ4 AdcaResultRegs.ADCRESULT9

#define IL2_ADC_MODULE    1
#define IL2_ADC_PIN    4
#define IL2_CMPSS_NO      2
#define IL2_ADC_TRIG_SOURCE ADC_TRIG_SOURCE
#define IL2_ACQPS_SYS_CLKS 50
#define IL2_ADC_SOC1 0x12
#define IL2_ADC_READ1 AdcaResultRegs.ADCRESULT1
#define IL2_ADC_SOC2 0x18
#define IL2_ADC_READ2 AdcaResultRegs.ADCRESULT4
#define IL2_ADC_SOC3 0x1E
#define IL2_ADC_READ3 AdcaResultRegs.ADCRESULT7
#define IL2_ADC_SOC4 0x24
#define IL2_ADC_READ4 AdcaResultRegs.ADCRESULT10

#define IL3_ADC_MODULE    1
#define IL3_ADC_PIN    14
#define IL3_CMPSS_NO      4
#define IL3_ADC_TRIG_SOURCE ADC_TRIG_SOURCE
#define IL3_ACQPS_SYS_CLKS 50
#define IL3_ADC_SOC1 0x14
#define IL3_ADC_READ1 AdcaResultRegs.ADCRESULT2
#define IL3_ADC_SOC2 0x1A
#define IL3_ADC_READ2 AdcaResultRegs.ADCRESULT5
#define IL3_ADC_SOC3 0x20
#define IL3_ADC_READ3 AdcaResultRegs.ADCRESULT8
```

```
#define IL3_ADC_SOC4 0x26
#define IL3_ADC_READ4 AdcaResultRegs.ADCRESULT11


#define V1_ADC_MODULE     2
#define V1_ADC_PIN      2
#define V1_ADC_TRIG_SOURCE ADC_TRIG_SOURCE
#define V1_ACQPS_SYS_CLKS 50
#define V1_ADC_SOC1 0x10
#define V1_ADC_READ1 AdcbResultRegs.ADCRESULT0
#define V1_ADC_SOC2 0x12
#define V1_ADC_READ2 AdcbResultRegs.ADCRESULT1
#define V1_ADC_SOC3 0x14
#define V1_ADC_READ3 AdcbResultRegs.ADCRESULT2
#define V1_ADC_SOC4 0x16
#define V1_ADC_READ4 AdcbResultRegs.ADCRESULT3

#define V2_ADC_MODULE     4
#define V2_ADC_PIN      0
#define V2_ADC_TRIG_SOURCE ADC_TRIG_SOURCE
#define V2_ACQPS_SYS_CLKS 50
#define V2_ADC_SOC1 0x10
#define V2_ADC_READ1 AdcdResultRegs.ADCRESULT0
#define V2_ADC_SOC2 0x14
#define V2_ADC_READ2 AdcdResultRegs.ADCRESULT2
#define V2_ADC_SOC3 0x18
#define V2_ADC_READ3 AdcdResultRegs.ADCRESULT4
#define V2_ADC_SOC4 0x1C
#define V2_ADC_READ4 AdcdResultRegs.ADCRESULT6


#define V3_ADC_MODULE     4
#define V3_ADC_PIN      2
#define V3_ADC_TRIG_SOURCE ADC_TRIG_SOURCE
#define V3_ACQPS_SYS_CLKS 50
#define V3_ADC_SOC1 0x12
#define V3_ADC_READ1 AdcdResultRegs.ADCRESULT1
#define V3_ADC_SOC2 0x16
#define V3_ADC_READ2 AdcdResultRegs.ADCRESULT3
#define V3_ADC_SOC3 0x1A
#define V3_ADC_READ3 AdcdResultRegs.ADCRESULT5
#define V3_ADC_SOC4 0x1E
#define V3_ADC_READ4 AdcdResultRegs.ADCRESULT7

#define VBUSPM_ADC_MODULE 3
#define VBUSPM_ADC_PIN        2
#define VBUSPM_ADC_TRIG_SOURCE ADC_TRIG_SOURCE
#define VBUSPM_ACQPS_SYS_CLKS 50
#define VBUSPM_ADC_SOC1 0x10
#define VBUSPM_ADC_READ1 AdccResultRegs.ADCRESULT0
#define VBUSPM_ADC_SOC2 0x14
#define VBUSPM_ADC_READ2 AdccResultRegs.ADCRESULT2
#define VBUSPM_ADC_SOC3 0x18
#define VBUSPM_ADC_READ3 AdccResultRegs.ADCRESULT4
#define VBUSPM_ADC_SOC4 0x1C
#define VBUSPM_ADC_READ4 AdccResultRegs.ADCRESULT6

#define VBUSMN_ADC_MODULE 3
#define VBUSMN_ADC_PIN        4
```

```
#define VBUSMN_ADC_TRIG_SOURCE ADC_TRIG_SOURCE
#define VBUSMN_ACQPS_SYS_CLKS 50
#define VBUSMN_ADC_SOC1 0x12
#define VBUSMN_ADC_READ1 AdccResultRegs.ADCRESULT1
#define VBUSMN_ADC_SOC2 0x16
#define VBUSMN_ADC_READ2 AdccResultRegs.ADCRESULT3
#define VBUSMN_ADC_SOC3 0x1A
#define VBUSMN_ADC_READ3 AdccResultRegs.ADCRESULT5
#define VBUSMN_ADC_SOC4 0x1E
#define VBUSMN_ADC_READ4 AdccResultRegs.ADCRESULT7

#define VBUS_REF_SET 620

#ifdef __cplusplus
}
#endif                                  /* extern "C" */

#endif //_PROJSETTINGS_H
```

A2.4 pfc3phvienna.c

```
/*
=============================================================================
=
System Name: Power Factor Correction Three Phase Vienna Rectifier

File Name:          pfc3phvienna.c

Target:                   F2837xD

Author:                   Manish Bhardwaj, 21/11/2016

Description: Project implements power factor correction control algorithm for
a three phase vienna rectifier

                          Current and voltages on the design are sensed using
the SDFM
                          or ADC selectable through the powerSUITE page.

                          The control ISR rate is 50Khz and is triggered by
EPWM1

                          following is the description of other files that
are used by this system
                          <solution name>.c -> This file, which has the
main.c of the project
                          <solution name>.h -> This is the main header file
for the project
                          <solution name>_settings.h -> powerSUITE generated
settings
                          <boad name>_board.c -> This is the file that is
used to keep common functions
                                                                related to the
board hardware like setting up the PWM, ADC, SDFM,
                                                                reading data and
common variables liked current, voltage etc.

Copyright (C) {2016} Texas Instruments Incorporated - http://www.ti.com/
 * ALL RIGHTS RESERVED*
=============================================================================
====  */

#include "pfc3phvienna.h"

//********************** USER Variables
***********************************//
// Global variables used in this system specific to this control method
// common variables are kept in <system_name>_board.c
//*****************************************************************************
*//

//TODO  Control Variables

// Measurement Variables
// Inductor Current Measurement
volatile float iL1MeasADC,iL2MeasADC,iL3MeasADC;
volatile float iL1MeasSD,iL2MeasSD,iL3MeasSD;
volatile float iL1Meas,iL2Meas,iL3Meas;
// Inductor Current Measurement Offset
```

```
volatile float iL1MeasOffset,iL2MeasOffset,iL3MeasOffset;

// Output Voltage Bus measurement
volatile float vBusMNMeas,vBusPMMeas, vBusMeas, vBusHalfMeas;
volatile float vBusMNMeasAvg,vBusPMMeasAvg, vBusMeasAvg;

// variables used for calibration of output voltage measurements
volatile float m_VBusMNMeas, b_VBusMNMeas;
volatile float m_VBusPMMeas, b_VBusPMMeas;

// Input Grid Voltage Measurement
volatile float v1Meas,v2Meas,v3Meas;
volatile float v1MeasOffset,v2MeasOffset,v3MeasOffset;

// Sine analyzer block for RMS Volt, Curr and Power measurements
sineAnalyzerWithPowerMeas sine_mains1, sine_mains2, sine_mains3;

volatile float vRmsMeasAvg;

// Variables used to calibrate measurement offsets
//Offset filter coefficient K1: 0.05/(T+0.05);
float k1 = 0.998;
//Offset filter coefficient K2: T/(T+0.05)
float k2 = 0.001999;
int16_t offsetCalCounter;
float offset165;

// Display Values
volatile float  guiVbusMN,guiVbusPM, guiVbus,
                        guiV1, guiV2, guiV3,
                        guiIL1, guiIL2, guiIL3,
                        guiIL1sd, guiIL2sd, guiIL3sd;

volatile float guiACFreq;
volatile float guiPrms1,guiPrms2, guiPrms3, guiPrmsTotal;
volatile float guiIrms1,guiIrms2, guiIrms3;
volatile float guiVrms1, guiVrms2, guiVrms3;
volatile float guiPF1, guiPF2, guiPF3;
volatile float guiVA1, guiVA2, guiVA3;

float guiVbusTripLimit=720.0;

uint16_t guiPowerStageStart;
uint16_t guiPowerStageStop;

// PFC Filtered DC bus measurement
volatile float vBusAvg;

volatile float iL1_CalibrationGain=1.0; //0.95999979;
volatile float iL2_CalibrationGain=1.0;
volatile float iL3_CalibrationGain=1.0; // 0.9850000;



//SD Trip Level
// 32 OSR and SINC3 is used for the comparator trip
// Max value is 32768
// As the SD is used to sense positive and negative values offset is 16384
Uint16  HLT = 0x7FFF;
```

```
Uint16  LLT = 0x0000;

// Control Variables

// CNTL 2p2z for DC Bus controller
#pragma DATA_SECTION(Gv_vars,"cntl_var_RAM")
#pragma DATA_SECTION(Gv_coeff,"cntl_coeff_RAM")
CNTL_2P2Z_F_VARS Gv_vars;
CNTL_2P2Z_F_COEFFS Gv_coeff;

CNTL_PI_F Gv;

// Reference variables
// Peak value of the Ii, Io current set point
volatile float iLRef, iL1Ref, iL2Ref, iL3Ref;
// DC Bus set point
volatile float vBusRef=0.0;
volatile float vBusRefSlewed=0.0;


// variables for third harmmonic injection
volatile float vMin, vMax;
volatile float thirdHarmonicInjection;

// individual duty cycles for each phase
volatile float duty1PU, duty2PU, duty3PU;
volatile float dutyPU_DC;

// Flags for clearing trips and closing the loops
int16_t closeGiLoop,closeGvLoop, closeGsLoop, clearTrip, firstTimeGvLoop;

// datalogger
DLOG_4CH_F dLog1;
float dBuff1[100], dBuff2[100], dBuff3[100], dBuff4[100];
float dVal1, dVal2, dVal3, dVal4;

volatile int updateCoeff=0;

volatile float Gi1_Out,Gi2_Out,Gi3_Out, Gs_Out;
volatile float Gs_GainKp=1.0;
volatile float Gi_GainKp=GI_GAINKP;

volatile float vBusDiff=0;
volatile float vBusZero=0;

volatile int16_t thirdHarmonicInjectionEnable=1;

volatile float iL1Ref_prev, iL2Ref_prev, iL3Ref_prev;
volatile float inductor_voltage_drop_feedforward1,
inductor_voltage_drop_feedforward2, inductor_voltage_drop_feedforward3;


int16_t slewState;

volatile uint16_t BusVoltageSlew=0;

// Stand Alone Flash Image Instrumentation

int16_t i;
```

```
int16_t timer1;

//--------------------------------- SFRA Related Variables -----------------
----------
float plantMagVect[SFRA_FREQ_LENGTH];
float plantPhaseVect[SFRA_FREQ_LENGTH];
float olMagVect[SFRA_FREQ_LENGTH];
float olPhaseVect[SFRA_FREQ_LENGTH];
float freqVect[SFRA_FREQ_LENGTH];

volatile SFRA_F sfra1;

//extern to access tables in ROM
extern long FPUsinTable[];
//--------------------------------- SFRA GUI Related Variables -------------
--------
volatile int16_t SerialCommsTimer;
volatile int16_t CommsOKflg;
//Flag for reinitializing SFRA variables
volatile int16_t initializationFlag;

//GUI support variables
// sets a limit on the amount of external GUI controls - increase as
necessary
volatile int16_t *varSetTxtList[16]; //16 textbox controlled variables
volatile int16_t *varSetBtnList[16]; //16 button controlled variables
volatile int16_t *varSetSldrList[16]; //16 slider controlled variables
volatile int16_t *varGetList[16]; //16 variables sendable to GUI
volatile int32_t *arrayGetList[16]; //16 arrays sendable to GUI
volatile uint32_t *dataSetList[16];    //16 32-bit textbox or label
controlled variables

//--------------------------- State Machine Related ------------------------
-------

int16_t      vTimer0[4];               // Virtual Timers slaved off CPU Timer
0 (A events)
int16_t      vTimer1[4];         // Virtual Timers slaved off CPU Timer 1 (B
events)

// Variable declarations for state machine
void (*Alpha_State_Ptr)(void);  // Base States pointer
void (*A_Task_Ptr)(void);       // State pointer A branch
void (*B_Task_Ptr)(void);       // State pointer B branch

//--------------------------------------------------------------------------
---------
// Enum for build level of software
enum enum_BuildLevel BuildInfo = BuildLevel1_OpenLoop ;

enum enum_boardState boardState = PowerStageOFF;

enum enum_boardStatus boardStatus = boardStatus_Idle;

//TODO Main
void main(void)
{
      // This routine sets up the basic device configuration such as
initializing PLL
```

```
        // copying code from FLASH to RAM,
        setupDevice();

        // Tasks State-machine init
        Alpha_State_Ptr = &A0;
        A_Task_Ptr = &A1;
        B_Task_Ptr = &B1;

        // Stop all PWM mode clock
        disablePWMCLKCounting();

        configure3phViennaPWM(PWM_NO,PFC3PH_PWM_PERIOD);

        // power up ADC on the device
        setupADC();
        // setup conversion on ADCA for inverter current and voltages
        // iL1Meas
        setupADCconversion(IL1_ADC_MODULE, IL1_ADC_SOC1, IL1_ADC_PIN,
IL1_ADC_TRIG_SOURCE, IL1_ACQPS_SYS_CLKS);
        // iL2Meas
        setupADCconversion(IL2_ADC_MODULE, IL2_ADC_SOC1, IL2_ADC_PIN,
IL2_ADC_TRIG_SOURCE, IL2_ACQPS_SYS_CLKS);
        // iL3Meas
        setupADCconversion(IL3_ADC_MODULE, IL3_ADC_SOC1, IL3_ADC_PIN,
IL3_ADC_TRIG_SOURCE, IL3_ACQPS_SYS_CLKS);

        // iL1Meas2
        setupADCconversion(IL1_ADC_MODULE, IL1_ADC_SOC2, IL1_ADC_PIN,
IL1_ADC_TRIG_SOURCE, IL1_ACQPS_SYS_CLKS);
        // iL2Meas2
        setupADCconversion(IL2_ADC_MODULE, IL2_ADC_SOC2, IL2_ADC_PIN,
IL2_ADC_TRIG_SOURCE, IL2_ACQPS_SYS_CLKS);
        // iL3Meas2
        setupADCconversion(IL3_ADC_MODULE, IL3_ADC_SOC2, IL3_ADC_PIN,
IL3_ADC_TRIG_SOURCE, IL3_ACQPS_SYS_CLKS);

        // iL1Meas3
        setupADCconversion(IL1_ADC_MODULE, IL1_ADC_SOC3, IL1_ADC_PIN,
IL1_ADC_TRIG_SOURCE, IL1_ACQPS_SYS_CLKS);
        // iL2Meas3
        setupADCconversion(IL2_ADC_MODULE, IL2_ADC_SOC3, IL2_ADC_PIN,
IL2_ADC_TRIG_SOURCE, IL2_ACQPS_SYS_CLKS);
        // iL3Meas3
        setupADCconversion(IL3_ADC_MODULE, IL3_ADC_SOC3, IL3_ADC_PIN,
IL3_ADC_TRIG_SOURCE, IL3_ACQPS_SYS_CLKS);

        // iL1Meas4
        setupADCconversion(IL1_ADC_MODULE, IL1_ADC_SOC4, IL1_ADC_PIN,
IL1_ADC_TRIG_SOURCE, IL1_ACQPS_SYS_CLKS);
        // iL2Meas4
        setupADCconversion(IL2_ADC_MODULE, IL2_ADC_SOC4, IL2_ADC_PIN,
IL2_ADC_TRIG_SOURCE, IL2_ACQPS_SYS_CLKS);
        // iL3Meas4
        setupADCconversion(IL3_ADC_MODULE, IL3_ADC_SOC4, IL3_ADC_PIN,
IL3_ADC_TRIG_SOURCE, IL3_ACQPS_SYS_CLKS);

        // v1Meas
        setupADCconversion(V1_ADC_MODULE, V1_ADC_SOC1, V1_ADC_PIN,
V1_ADC_TRIG_SOURCE, V1_ACQPS_SYS_CLKS);
```

```
       // v2Meas
       setupADCconversion(V2_ADC_MODULE, V2_ADC_SOC1, V2_ADC_PIN,
V2_ADC_TRIG_SOURCE, V2_ACQPS_SYS_CLKS);
       // v3Meas
       setupADCconversion(V3_ADC_MODULE, V3_ADC_SOC1, V3_ADC_PIN,
V3_ADC_TRIG_SOURCE, V3_ACQPS_SYS_CLKS);

       // v1Meas2
       setupADCconversion(V1_ADC_MODULE, V1_ADC_SOC2, V1_ADC_PIN,
V1_ADC_TRIG_SOURCE, V1_ACQPS_SYS_CLKS);
       // v2Meas2
       setupADCconversion(V2_ADC_MODULE, V2_ADC_SOC2, V2_ADC_PIN,
V2_ADC_TRIG_SOURCE, V2_ACQPS_SYS_CLKS);
       // v3Meas2
       setupADCconversion(V3_ADC_MODULE, V3_ADC_SOC2, V3_ADC_PIN,
V3_ADC_TRIG_SOURCE, V3_ACQPS_SYS_CLKS);

       // v1Meas3
       setupADCconversion(V1_ADC_MODULE, V1_ADC_SOC3, V1_ADC_PIN,
V1_ADC_TRIG_SOURCE, V1_ACQPS_SYS_CLKS);
       // v2Meas3
       setupADCconversion(V2_ADC_MODULE, V2_ADC_SOC3, V2_ADC_PIN,
V2_ADC_TRIG_SOURCE, V2_ACQPS_SYS_CLKS);
       // v3Meas3
       setupADCconversion(V3_ADC_MODULE, V3_ADC_SOC3, V3_ADC_PIN,
V3_ADC_TRIG_SOURCE, V3_ACQPS_SYS_CLKS);

       // v1Meas4
       setupADCconversion(V1_ADC_MODULE, V1_ADC_SOC4, V1_ADC_PIN,
V1_ADC_TRIG_SOURCE, V1_ACQPS_SYS_CLKS);
       // v2Meas4
       setupADCconversion(V2_ADC_MODULE, V2_ADC_SOC4, V2_ADC_PIN,
V2_ADC_TRIG_SOURCE, V2_ACQPS_SYS_CLKS);
       // v3Meas4
       setupADCconversion(V3_ADC_MODULE, V3_ADC_SOC4, V3_ADC_PIN,
V3_ADC_TRIG_SOURCE, V3_ACQPS_SYS_CLKS);

       // vBusMeas
       setupADCconversion(VBUSPM_ADC_MODULE, VBUSPM_ADC_SOC1, VBUSPM_ADC_PIN,
VBUSPM_ADC_TRIG_SOURCE, VBUSPM_ACQPS_SYS_CLKS);
       setupADCconversion(VBUSPM_ADC_MODULE, VBUSPM_ADC_SOC2, VBUSPM_ADC_PIN,
VBUSPM_ADC_TRIG_SOURCE, VBUSPM_ACQPS_SYS_CLKS);
       setupADCconversion(VBUSPM_ADC_MODULE, VBUSPM_ADC_SOC3, VBUSPM_ADC_PIN,
VBUSPM_ADC_TRIG_SOURCE, VBUSPM_ACQPS_SYS_CLKS);
       setupADCconversion(VBUSPM_ADC_MODULE, VBUSPM_ADC_SOC4, VBUSPM_ADC_PIN,
VBUSPM_ADC_TRIG_SOURCE, VBUSPM_ACQPS_SYS_CLKS);

       // vBusMidMeas
       setupADCconversion(VBUSMN_ADC_MODULE, VBUSMN_ADC_SOC1, VBUSMN_ADC_PIN,
VBUSMN_ADC_TRIG_SOURCE, VBUSMN_ACQPS_SYS_CLKS);
       setupADCconversion(VBUSMN_ADC_MODULE, VBUSMN_ADC_SOC2, VBUSMN_ADC_PIN,
VBUSMN_ADC_TRIG_SOURCE, VBUSMN_ACQPS_SYS_CLKS);
       setupADCconversion(VBUSMN_ADC_MODULE, VBUSMN_ADC_SOC3, VBUSMN_ADC_PIN,
VBUSMN_ADC_TRIG_SOURCE, VBUSMN_ACQPS_SYS_CLKS);
       setupADCconversion(VBUSMN_ADC_MODULE, VBUSMN_ADC_SOC4, VBUSMN_ADC_PIN,
VBUSMN_ADC_TRIG_SOURCE, VBUSMN_ACQPS_SYS_CLKS);

       setupSDFM(PFC3PH_PWM_PERIOD,PWM_CLK_IN_SDFM_OSR,
SD_CLK_COUNT,SDFM_OSR);
```

```
        //Profiling GPIO and led GPIOs
        setupProfilingGPIO();

        //configure lED GPIO
        setupLEDGPIO();

        //Control Variable specific to the voltage source inverter
Initialization

        CNTL_2P2Z_F_VARS_init(&Gv_vars);
        CNTL_2P2Z_F_COEFFS_init(&Gv_coeff);

        Gv_coeff.Coeff_A1 = (float)(CNTL_GV_A1);
        Gv_coeff.Coeff_A2 = (float)(CNTL_GV_A2);
        Gv_coeff.Coeff_B0 = (float)(CNTL_GV_B0);
        Gv_coeff.Coeff_B1 = (float)(CNTL_GV_B1);
        Gv_coeff.Coeff_B2 = (float)(CNTL_GV_B2);
        Gv_coeff.IMin       = (float)(CNTL_GV_IMIN);
        Gv_coeff.Min = (float)(CNTL_GV_MIN);
        Gv_coeff.Max = (float)(CNTL_GV_MAX);

        CNTL_PI_F_init(&Gv);
        Gv.Ki=0.04;
        Gv.Kp=1.0;
        Gv.Umax=CNTL_GV_MAX;
        Gv.Umin=CNTL_GV_MIN;

        setupSFRA();

        //sine analyzer initialization
        sineAnalyzerWithPowerMeas_init(&sine_mains1);
        sine_mains1.sampleFreq = (float32)(ISR_10KHZ_FREQUENCY);
        sine_mains1.threshold = (float32)(0.08);
        sine_mains1.nSamplesMax=ISR_10KHZ_FREQUENCY/UNIVERSAL_GRID_MIN_FREQ;
        sine_mains1.nSamplesMin=ISR_10KHZ_FREQUENCY/UNIVERSAL_GRID_MAX_FREQ;
        sine_mains1.emaFilterMultiplier=2.0/ISR_10KHZ_FREQUENCY;

        sineAnalyzerWithPowerMeas_init(&sine_mains2);
        sine_mains2.sampleFreq = (float32)(ISR_10KHZ_FREQUENCY);
        sine_mains2.threshold = (float32)(0.08);
        sine_mains2.nSamplesMax=ISR_10KHZ_FREQUENCY/UNIVERSAL_GRID_MIN_FREQ;
        sine_mains2.nSamplesMin=ISR_10KHZ_FREQUENCY/UNIVERSAL_GRID_MAX_FREQ;
        sine_mains2.emaFilterMultiplier=2.0/ISR_10KHZ_FREQUENCY;

        sineAnalyzerWithPowerMeas_init(&sine_mains3);
        sine_mains3.sampleFreq = (float32)(ISR_10KHZ_FREQUENCY);
        sine_mains3.threshold = (float32)(0.08);
        sine_mains3.nSamplesMax=ISR_10KHZ_FREQUENCY/UNIVERSAL_GRID_MIN_FREQ;
        sine_mains3.nSamplesMin=ISR_10KHZ_FREQUENCY/UNIVERSAL_GRID_MAX_FREQ;
        sine_mains3.emaFilterMultiplier=2.0/ISR_10KHZ_FREQUENCY;

        // Initialize global variables generic to the board like ones used to
read current values etc
        initGlobalVariables();

        // Enable PWM Clocks
        enablePWMCLKCounting();
```

```
	// Offset Calibration Routine
#if SENSING_OPTION ==ADC_BASED_SENSING
	calibrateOffset(&iL1MeasOffset, &iL2MeasOffset, &iL3MeasOffset,
			&v1MeasOffset, &v2MeasOffset, &v3MeasOffset, k1 , k2);
#endif

	// clear any spurious flags in the SDFM module
	// setup protection and trips for the board
	setupBoardProtection(SENSING_OPTION, I_TRIP_LIMIT,I_MAX_SENSE);

	// PWM were tripped low in the previoud routine

	// now it is safe to enable them as PWM, untill now they will be GPIO
input
	// and because of the res pull down all will be drivern low
	InitEPwm1Gpio();
	InitEPwm2Gpio();
	InitEPwm3Gpio();

	// ISR Mapping
	setupInterrupt();

// IDLE loop. Just sit and loop forever, periodically will branch into A0-A3,
B0-B3, C0-C3 tasks
// Frequency of this brnaching is set in setupDevice routine:
	for(;;)  //infinite loop
	{
		// State machine entry & exit point
		//===========================================================
		(*Alpha_State_Ptr)();	// jump to an Alpha state (A0,B0,...)
		//===========================================================
		if(initializationFlag == 1)
		{
		//	SFRA_F_INIT(&SFRA1);
			initializationFlag = 0;
			sfra1.start = 1;
		}
	}
} //END MAIN CODE



//===========================================================================
======
//	STATE-MACHINE SEQUENCING AND SYNCRONIZATION FOR SLOW BACKGROUND TASKS
//===========================================================================
======

//--------------------------------- SFRAMEWORK ----------------------------
-------
void A0(void)
{
	// loop rate synchronizer for A-tasks
	if(CpuTimer0Regs.TCR.bit.TIF == 1)
	{
		CpuTimer0Regs.TCR.bit.TIF = 1;  // clear flag

		//-----------------------------------------------------------
		(*A_Task_Ptr)();		// jump to an A Task (A1,A2,A3,...)
```

```
            //-------------------------------------------------------------

            vTimer0[0]++;                   // virtual timer 0, instance 0
(spare)
            SerialCommsTimer++;
      }


      Alpha_State_Ptr = &B0;          // Comment out to allow only A tasks
}

void B0(void)
{
      // loop rate synchronizer for B-tasks
      if(CpuTimer1Regs.TCR.bit.TIF == 1)
      {
            CpuTimer1Regs.TCR.bit.TIF = 1;                      // clear
flag

            //-------------------------------------------------------------
            (*B_Task_Ptr)();          // jump to a B Task (B1,B2,B3,...)
            //-------------------------------------------------------------
            vTimer1[0]++;                   // virtual timer 1, instance 0
(spare)
      }

      Alpha_State_Ptr = &A0;          // Allow C state tasks
}


//===========================================================================
======
//     A - TASKS (executed in every 1 msec)
//===========================================================================
======
//-----------------------------------------------------------
void A1(void) // SPARE (not used)
//-----------------------------------------------------------
{

      SerialHostComms();
      SFRA_F_BACKGROUND(&sfra1);


      //-------------------
      //the next time CpuTimer0 'counter' reaches Period value go to A2
      A_Task_Ptr = &A1;
      //-------------------
}

//===========================================================================
======
//     B - TASKS (executed in every 5 msec)
//===========================================================================
======

//--------------------------------- USER -----------------------------------
------
```

```
//---------------------------------------
void B1(void)
//---------------------------------------
{
      //updateBoardStatus();
#if INCR_BUILD==1
      BuildInfo=BuildLevel1_OpenLoop;
#elif INCR_BUILD==2
      BuildInfo=BuildLevel2_CurrentLoop;
#elif INCR_BUILD==3
      BuildInfo=BuildLevel3_VoltageAndCurrentLoop;
#elif INCR_BUILD==4
      BuildInfo=BuildLevel4_BalanceVoltageAndCurrentLoop;
#endif
      //-----------------
      //the next time CpuTimer1 'counter' reaches Period value go to B2
      B_Task_Ptr = &B2;
      //-----------------
}

//---------------------------------------
void B2(void) //  SPARE
//---------------------------------------
{
      toggleLED();

      if(EPwm1Regs.TZFLG.bit.DCBEVT1==1)
            {
                  if(boardStatus==boardStatus_NoFault)
                        boardStatus =      boardStatus_OverCurrentTrip;
            }

      //-----------------
      //the next time CpuTimer1 'counter' reaches Period value go to B3
      B_Task_Ptr = &B3;
      //-----------------
}

//---------------------------------------
void B3(void) //  SPARE
//---------------------------------------
{

      if(updateCoeff==1)
      {
            DINT;

            EINT;
            updateCoeff=0;
      }

      //-----------------
      //the next time CpuTimer1 'counter' reaches Period value go to B1
      B_Task_Ptr = &B1;
      //-----------------
}
```

```
//TODO control ISR Code
interrupt void controlISR(void)
{

	setProfilingGPIO();

	readCurrVolADCSignals();

	//readCurrVolSDFMSignals(SD32_PU_SCALE_FACTOR);

#if(ALL_PHASE_ENABLED==1)
	clearPWMTrip();
#else
	if (clearTrip==1 )
		{
		EALLOW;
#if CHECK_PHASE == 1
		// clear all the configured trip sources for the PWM
module 1
		EPwm1Regs.TZCLR.bit.OST=0x1;
		EPwm1Regs.TZCLR.bit.CBC=0x1;
		EPwm1Regs.TZCLR.bit.DCAEVT1=0x1;
#elif CHECK_PHASE == 2
		// clear all the configured trip sources for the PWM
module 2
		EPwm2Regs.TZCLR.bit.OST=0x1;
		EPwm2Regs.TZCLR.bit.CBC=0x1;
		EPwm2Regs.TZCLR.bit.DCAEVT1=0x1;
		clearTrip=0;
#elif CHECK_PHASE == 3
		// clear all the configured trip sources for the PWM
module 2
		EPwm3Regs.TZCLR.bit.OST=0x1;
		EPwm3Regs.TZCLR.bit.CBC=0x1;
		EPwm3Regs.TZCLR.bit.DCAEVT1=0x1;
#else
#warning define CHECK_PHASE in this build
#endif
		EDIS;
		}
#endif

	//TODO BUILD 1
	//-----------------------------------------------------------------------
----------------------
	#if (INCR_BUILD == 1 ) // Open Loop Check
	//-----------------------------------------------------------------------
----------------------
		duty1PU = SFRA_F_INJECT(dutyPU_DC);
		duty2PU = dutyPU_DC;
		duty3PU = dutyPU_DC;

	#endif // (INCR_BUILD == 1)


	//TODO BUILD 2
	//-----------------------------------------------------------------------
----------------------
```

```
      #if (INCR_BUILD == 2 || INCR_BUILD==3 || INCR_BUILD==4)   //Closed
Current Loop
      //-------------------------------------------------------------------
--------------------

      iL1Ref= iLRef * (v1Meas/vRmsMeasAvg);
      iL2Ref= iLRef* (v2Meas/vRmsMeasAvg);
      iL3Ref= iLRef* (v3Meas/vRmsMeasAvg);



#if INDUCTOR_VOLTAGE_DROP_FEEDFORWARD ==1
      // inductor voltage drop feed forward (L*di/dt)
      inductor_voltage_drop_feedforward1 = (iL1Ref - iL1Ref_prev)* (
PFC_INDUCTOR_VALUE *ISR_CONTROL_FREQUENCY * I_MAX_SENSE / V_MAX_SENSE);
      inductor_voltage_drop_feedforward2 = (iL2Ref - iL2Ref_prev)* (
PFC_INDUCTOR_VALUE *ISR_CONTROL_FREQUENCY * I_MAX_SENSE / V_MAX_SENSE);
      inductor_voltage_drop_feedforward3 = (iL3Ref - iL3Ref_prev)* (
PFC_INDUCTOR_VALUE *ISR_CONTROL_FREQUENCY * I_MAX_SENSE / V_MAX_SENSE);
#else
      inductor_voltage_drop_feedforward1 = 0;
      inductor_voltage_drop_feedforward2 = 0;
      inductor_voltage_drop_feedforward3 = 0;
#endif

            if(closeGiLoop==1)
            {

#if SFRA_TYPE==SFRA_CURRENT
                  Gi1_Out= (iL1Meas - SFRA_F_INJECT(iL1Ref))* Gi_GainKp;
#else
                  Gi1_Out= (iL1Meas - iL1Ref)* Gi_GainKp;
#endif
                  Gi2_Out= (iL2Meas - iL2Ref)* Gi_GainKp;

                  Gi3_Out= (iL3Meas - iL3Ref)* Gi_GainKp;


#if THIRD_HARMONIC_INJECTION == 0
      thirdHarmonicInjection=0;
#else
      vMin= (v1Meas<v2Meas)?v1Meas:v2Meas;
      vMin= (vMin<v3Meas)?vMin:v3Meas;
      vMax= (v1Meas>v2Meas)?v1Meas:v2Meas;
      vMax= (vMax>v3Meas)?vMax:v3Meas;

      thirdHarmonicInjection = (vMin+vMax)*0.5;
#endif


                  duty1PU = ( (Gi1_Out
                                          +
inductor_voltage_drop_feedforward1
                                          + v1Meas
                                          - thirdHarmonicInjection
                                          ) / vBusHalfMeas ) - Gs_Out;

                  duty2PU = ( (Gi2_Out
```

```
                                                +
inductor_voltage_drop_feedforward2
                                                + v2Meas
                                                - thirdHarmonicInjection
                                                ) / vBusHalfMeas ) - Gs_Out;


                        duty3PU = ( (Gi3_Out
                                                +
inductor_voltage_drop_feedforward3
                                                + v3Meas
                                                - thirdHarmonicInjection
                                                 ) / vBusHalfMeas ) - Gs_Out;
                }
                else
                {
                        duty1PU=1.0;
                        duty2PU=1.0;
                        duty3PU=1.0;
                }

        #endif

        //-------------------------------------------------------------------
------------
        // PWM Driver for three phase Vienna Rectifier PFC
        //-------------------------------------------------------------------
------------
        duty1PU = (duty1PU>1.0)?1.0:duty1PU;
        duty1PU = (duty1PU<-1.0)?-1.0:duty1PU;
        duty2PU = (duty2PU>1.0)?1.0:duty2PU;
        duty2PU = (duty2PU<-1.0)?-1.0:duty2PU;
        duty3PU = (duty3PU>1.0)?1.0:duty3PU;
        duty3PU = (duty3PU<-1.0)?-1.0:duty3PU;

        updatePFCviennaPWM(duty1PU, duty2PU, duty3PU, PWM_NO);

        iL1Ref_prev=iL1Ref;
        iL2Ref_prev=iL2Ref;
        iL3Ref_prev=iL3Ref;
//TODO BUILD 3
//-------------------------------------------------------------------------
--------------
#if(INCR_BUILD ==3 || INCR_BUILD ==4)
//-------------------------------------------------------------------------
--------------
        if(closeGvLoop==1)
        {
                if(firstTimeGvLoop==1)
                {
                        vBusRefSlewed=vBusMeas;
                        firstTimeGvLoop=0;
                }

#if SFRA_TYPE==SFRA_VOLTAGE
                Gv_vars.Ref = SFRA_F_INJECT(vBusRefSlewed);
#else
                Gv_vars.Ref = vBusRefSlewed;
#endif
```

```
            Gv_vars.Fdbk = vBusMeas;


            Gv_vars.Errn = Gv_vars.Ref - Gv_vars.Fdbk;

#if NON_LINEAR_VOLTAGE_LOOP
            if(fabs(Gv_vars.Errn) > 0.01)
            {
                  Gv_vars.Out = Gv_vars.Out1 +
Gv_coeff.Coeff_B0*3*(Gv_vars.Errn) + Gv_coeff.Coeff_B1*3*(Gv_vars.Errn1);
            }
            else
                  Gv_vars.Out = Gv_vars.Out1 +
Gv_coeff.Coeff_B0*(Gv_vars.Errn) + Gv_coeff.Coeff_B1*(Gv_vars.Errn1);

#else
            Gv_vars.Out = Gv_vars.Out1 + Gv_coeff.Coeff_B0*(Gv_vars.Errn) +
Gv_coeff.Coeff_B1*(Gv_vars.Errn1);
#endif

            Gv_vars.Out = ( Gv_vars.Out>Gv_coeff.Max )? Gv_coeff.Max :
Gv_vars.Out;
            Gv_vars.Out = ( Gv_vars.Out<Gv_coeff.Min )? Gv_coeff.Min :
Gv_vars.Out;

            Gv_vars.Out1 = Gv_vars.Out;
            Gv_vars.Errn1 = Gv_vars.Errn;

            // Output of the voltage loop is thought to be the Power
            iLRef=Gv_vars.Out * (vBusMeasAvg/(3*vRmsMeasAvg));
            //iLRef=Gv_vars.Out ;

      }
#endif



#if(INCR_BUILD ==4)
      if(closeGsLoop==1)
      {
            vBusDiff= vBusPMMeas-vBusMNMeas;

#if SFRA_TYPE==SFRA_BALANCECNTL
            Gs_Out = ( vBusDiff -SFRA_F_INJECT(vBusZero))*Gs_GainKp;
#else
            Gs_Out = (vBusDiff)*Gs_GainKp;
#endif
      }
      else
      {
            Gs_Out=0;
      }
#else
      Gs_Out=0;
#endif


      #if (SFRA_TYPE == SFRA_VOLTAGE )       //Running FRA on Voltage
            SFRA_F_COLLECT(&Gv_vars.Out,&vBusMeas);
```

```
        #elif(SFRA_TYPE ==SFRA_CURRENT) // running FRA on Current
              SFRA_F_COLLECT(&Gi1_Out,&iL1Meas);
        #elif(SFRA_TYPE == SFRA_BALANCECNTL)
              SFRA_F_COLLECT(&Gs_Out,&vBusDiff);
        #endif

        clearInterrupt();

    resetProfilingGPIO();

}// MainISR Ends Here


//TODO control ISR Code
interrupt void tenkHzISR(void)
{
        // Let the controlISR interrupt this routine
        EINT;
        setProfilingGPIO2();

        if(closeGvLoop==1)
        {
              if ( fabs(vBusRef - vBusRefSlewed) > 0.1)
              {
                    if(vBusRef>vBusRefSlewed)
                          vBusRefSlewed = vBusRefSlewed + 0.0001;
                    else
                          vBusRefSlewed = vBusRefSlewed - 0.0001;
              }
              else if (fabs(vBusRef - vBusRefSlewed) > 0.01)
              {
                    if(vBusRef>vBusRefSlewed)
                          vBusRefSlewed = vBusRefSlewed + 0.00005;
                    else
                          vBusRefSlewed = vBusRefSlewed - 0.00005;
              }
              else if (fabs(vBusRef - vBusRefSlewed) > 0.005)
              {
                    if(vBusRef>vBusRefSlewed)
                          vBusRefSlewed = vBusRefSlewed + 0.00001;
                    else
                          vBusRefSlewed = vBusRefSlewed - 0.00001;
              }
              else
              {
                    vBusRefSlewed = vBusRef;
              }
        }


        if(guiVbus > guiVbusTripLimit || guiVbusPM > 415.0  || guiVbusMN >
415.0)
        {
              EALLOW;
              EPwm1Regs.TZFRC.bit.OST = 1;
              EPwm2Regs.TZFRC.bit.OST = 1;
              EPwm3Regs.TZFRC.bit.OST = 1;
              EDIS;
```

```
            if(boardStatus==boardStatus_NoFault)
                  boardStatus =      boardStatus_OverVoltageTrip;
      }

      EMAVG_MACRO(vBusPMMeas, vBusPMMeasAvg);
      EMAVG_MACRO(vBusMNMeas, vBusMNMeasAvg);

      vBusMeasAvg= vBusPMMeasAvg+ vBusMNMeasAvg;


      guiVbusPM= vBusPMMeasAvg * V_MAX_SENSE;
      guiVbusMN= vBusMNMeasAvg * V_MAX_SENSE;
      guiVbus = guiVbusPM + guiVbusMN;

      guiV1 = v1Meas * V_MAX_SENSE;
      guiV2 = v2Meas * V_MAX_SENSE;
      guiV3 = v3Meas * V_MAX_SENSE;

      guiIL1 = iL1Meas * I_MAX_SENSE;
      guiIL2 = iL2Meas * I_MAX_SENSE;
      guiIL3 = iL3Meas * I_MAX_SENSE;

      /*guiIL1sd = iL1MeasSD * I_MAX_SENSE_SD;
      guiIL2sd = iL2MeasSD * I_MAX_SENSE_SD;
      guiIL3sd = iL3MeasSD * I_MAX_SENSE_SD;
*/
      // Sine Analyzer
      //Calculate RMS input voltage and input frequency
      sine_mains1.i = iL1Meas;
      sine_mains1.v = v1Meas;
      sineAnalyzerWithPowerMeas_calc(&sine_mains1);

      sine_mains2.i = iL2Meas;
      sine_mains2.v = v2Meas;
      sineAnalyzerWithPowerMeas_calc(&sine_mains2);

      sine_mains3.i = iL3Meas;
      sine_mains3.v = v3Meas;
      sineAnalyzerWithPowerMeas_calc(&sine_mains3);

      EMAVG_MACRO(sine_mains1.vRms, vRmsMeasAvg);

      guiIrms1=sine_mains1.iRms*I_MAX_SENSE;
      guiVrms1=sine_mains1.vRms*V_MAX_SENSE;
      guiPrms1=sine_mains1.pRms*V_MAX_SENSE*I_MAX_SENSE;
      guiPF1=sine_mains1.powerFactor;
      guiVA1=sine_mains1.vaRms*V_MAX_SENSE*I_MAX_SENSE;

      guiIrms2=sine_mains2.iRms*I_MAX_SENSE;
      guiVrms2=sine_mains2.vRms*V_MAX_SENSE;
      guiPrms2=sine_mains2.pRms*V_MAX_SENSE*I_MAX_SENSE;
      guiPF2=sine_mains2.powerFactor;
      guiVA2=sine_mains2.vaRms*V_MAX_SENSE*I_MAX_SENSE;

      guiIrms3=sine_mains3.iRms*I_MAX_SENSE;
      guiVrms3=sine_mains3.vRms*V_MAX_SENSE;
      guiPrms3=sine_mains3.pRms*V_MAX_SENSE*I_MAX_SENSE;
      guiPF3=sine_mains3.powerFactor;
      guiVA3=sine_mains3.vaRms*V_MAX_SENSE*I_MAX_SENSE;
```

```
	guiACFreq = sine_mains1.acFreqAvg;

	//TODO DLOG
	// ---------------------------------------------------------------------
-----------
	//    Connect inputs of the Datalogger module
	// ---------------------------------------------------------------------
-----------

	// check voltage and inductor current meas.
	dVal1 = guiV1;
	dVal2 = iL1Meas;
	dVal3 = iL2Meas;
	dVal4 = iL3Meas;

	DLOG_4CH_F_MACRO(dLog1);

	resetProfilingGPIO2();
}

//TODO setupSFRA
void setupSFRA(void)
{
	setupSCIconnectionForSFRA();

	//Specify the injection amplitude
	sfra1.amplitude=SFRA_AMPLITUDE;

	SFRA_F_INIT(&sfra1);

	//SFRA Related
	//SFRA Object Initialization

	//Specify the length of SFRA
	sfra1.Vec_Length=SFRA_FREQ_LENGTH;
	//Specify the SFRA ISR Frequency
	sfra1.ISR_Freq=SFRA_ISR_FREQ;
	//Specify the Start Frequency of the SFRA analysis
	sfra1.Freq_Start=SFRA_FREQ_START;
	//Specify the Frequency Step
	sfra1.Freq_Step=FREQ_STEP_MULTIPLY;
	//Assign array location to Pointers in the SFRA object
	sfra1.FreqVect=freqVect;
	sfra1.GH_MagVect=olMagVect;
	sfra1.GH_PhaseVect=olPhaseVect;
	sfra1.H_MagVect=plantMagVect;
	sfra1.H_PhaseVect=plantPhaseVect;

	// Re-initialize the frequency array to make SFRA sweep go fast
	i=0;

	#if SFRA_TYPE==SFRA_CURRENT// current loop
	sfra1.FreqVect[i++]=SFRA_FREQ_START;
	for(;i<sfra1.Vec_Length;i++)
	{
		sfra1.FreqVect[i]=sfra1.FreqVect[i-1]*sfra1.Freq_Step;
	}
	#else
```

```
sfra1.FreqVect[0]=2;
sfra1.FreqVect[1]=4;
sfra1.FreqVect[2]=6;
sfra1.FreqVect[3]=8;
sfra1.FreqVect[4]=10;
sfra1.FreqVect[5]=12;
sfra1.FreqVect[6]=14;
sfra1.FreqVect[7]=16;
sfra1.FreqVect[8]=18;
sfra1.FreqVect[9]=20;
sfra1.FreqVect[10]=22;
sfra1.FreqVect[11]=24;
sfra1.FreqVect[12]=26;
sfra1.FreqVect[13]=28;
sfra1.FreqVect[14]=30;
sfra1.FreqVect[15]=35;
sfra1.FreqVect[16]=40;
sfra1.FreqVect[17]=45;
sfra1.FreqVect[18]=55;
sfra1.FreqVect[19]=65;
sfra1.FreqVect[20]=70;
sfra1.FreqVect[21]=80;
sfra1.FreqVect[22]=90;
sfra1.FreqVect[23]=100;
sfra1.FreqVect[24]=120;
sfra1.FreqVect[25]=140;
sfra1.FreqVect[26]=160;
sfra1.FreqVect[27]=170;
sfra1.FreqVect[28]=210;
sfra1.FreqVect[29]=250;
#endif



//"Set" variables
// assign GUI Buttons to desired flag addresses
varSetBtnList[0] = (int16*)&initializationFlag;

//"Get" variables
//---------------------------------------
// assign a GUI "getable" parameter address
varGetList[0] = (int16*)&(sfra1.Vec_Length);
varGetList[1] = (int16*)&(sfra1.status);
varGetList[2] = (int16*)&(sfra1.FreqIndex);

//"Setable" variables
//---------------------------------------
// assign GUI "setable" by Text parameter address
dataSetList[0] = (Uint32*)&(sfra1.Freq_Start);
dataSetList[1] = (Uint32*)&(sfra1.amplitude);
dataSetList[2] = (Uint32*)&(sfra1.Freq_Step);

// assign a GUI "getable" parameter array address
arrayGetList[0] = (int32*)freqVect;
arrayGetList[1] = (int32*)olMagVect;
arrayGetList[2] = (int32*)olPhaseVect;
arrayGetList[3] = (int32*)plantMagVect;
arrayGetList[4] = (int32*)plantPhaseVect;
arrayGetList[5] = (int32*)&(sfra1.Freq_Start);
```

```
	arrayGetList[6] = (int32*)&(sfra1.amplitude);
	arrayGetList[7] = (int32*)&(sfra1.Freq_Step);

	CommsOKflg = 0;
	SerialCommsTimer = 0;
}

//===========================================================================
// No more.
//===========================================================================



//TODO initGlobalVariables()
void initGlobalVariables(void)
{
	// Gui Variables
	guiVbus=0;
	guiACFreq=0;
	guiPrms1=0;
	guiIrms1=0;
	guiVrms1=0;

	guiPrms2=0;
	guiIrms2=0;
	guiVrms2=0;

	guiPrms3=0;
	guiIrms3=0;
	guiVrms3=0;


	vBusPMMeasAvg=0;
	vBusMNMeasAvg=0;

	dLog1.input_ptr1 = &dVal1;
	dLog1.input_ptr2 = &dVal2;
	dLog1.input_ptr3 = &dVal3;  // spll.Out logged in buff3
	dLog1.input_ptr4 = &dVal4;  // ramp_sin logged in buff4
	dLog1.output_ptr1 = dBuff1;
	dLog1.output_ptr2 = dBuff2;
	dLog1.output_ptr3 = dBuff3;
	dLog1.output_ptr4 = dBuff4;
	dLog1.prev_value = (float32)(0);
	dLog1.trig_value = (float32)(0.05);
	dLog1.status = 1;
	dLog1.pre_scalar = 5;
	dLog1.skip_count = 0;
	dLog1.size = 100;
	dLog1.count = 0;

	//Variable initialization

	offsetCalCounter=0;

	closeGiLoop=0;
	closeGsLoop=0;
	closeGvLoop=0;
	clearTrip=0;
```

```
	duty1PU=0.0;
	duty2PU=0.0;
	duty3PU=0.0;

	iL1MeasOffset=0.0;
	iL2MeasOffset=0.0;
	iL3MeasOffset=0.0;

	m_VBusPMMeas=0.985769085;
	b_VBusPMMeas=0.003082908;

	m_VBusMNMeas=0.991549;
	b_VBusMNMeas=-0.00324;

	vRmsMeasAvg=0.0;

	guiPowerStageStart=0;
	guiPowerStageStop=0;

	dutyPU_DC=0.5;

	firstTimeGvLoop=1;
	closeGvLoop=0;
	vBusRef=((float)VBUS_REF_SET / (float)V_MAX_SENSE );
	iLRef=0.05;
}

void SPLL_LPF_Coeff_Update(SPLL_1ph_SOGI_F *spll, float Kp, float Ki, float
Ts)
{
	spll->lpf_coeff.B0_lf= ((2.0)*Kp+(float)Ki*(float)Ts)*0.5;
	spll->lpf_coeff.B1_lf= -((2.0)*Kp-(float)Ki*(float)Ts)*0.5;
	spll->lpf_coeff.A1_lf = 1;
}
//TODO updateBoardStatus()
void updateBoardStatus(void)
{
	if(EPwm1Regs.TZFLG.all==0 && EPwm2Regs.TZFLG.all==0)
		boardStatus=boardStatus_NoFault;
	else
	{
		if(EPwm1Regs.TZFLG.bit.DCAEVT1==1 ||
EPwm1Regs.TZFLG.bit.DCAEVT1==1)
		{
			boardStatus=boardStatus_OverCurrentTrip;
		}
		else if(EPwm1Regs.TZFLG.bit.CBC==1 ||
EPwm2Regs.TZFLG.bit.CBC==1)
		{
			boardStatus=boardStatus_EmulatorStopTrip;
			openRelay();
		}
		else if (EPwm1Regs.TZFLG.bit.OST==1 &&
EPwm2Regs.TZFLG.bit.OST==1 )
			boardStatus=boardStatus_Idle;
	}
}
```

A2.5 pfc3phvienna.h

```
/*
=================================================================================
=
System Name:

File Name:         voltagesourceinv_lcfltr.h

Target:

Author:

Description:

Copyright (C) {2015} Texas Instruments Incorporated - http://www.ti.com/
 * ALL RIGHTS RESERVED*
=================================================================================
====  */
#ifndef SOLUTION_H
#define SOLUTION_H

#ifdef __cplusplus

extern "C" {
#endif

//*********************************************************************************
**
// the includes
//*********************************************************************************
**

#include "pfc3phvienna_board.h"
// this is the file generated by the powerSUITE page for this system
#include "pfc3phvienna_settings.h"
#include "Solar_F.h"
#include "SFRA_F_Include.h"
#include "sineAnalyzerWithPowerMeas.h"

#define EMAVG_MACRO(in, out)                  \
       out = ((in - out)*0.000528) + out;

//=================================================================================
=======
// System Settings
//---------------------------------------------------------------------------
-------
//Add any system specific setting below
#if SFRA_TYPE==SFRA_VOLTAGE || SFRA_TYPE==SFRA_BALANCECNTL // voltage loop/
balance loop
#define SFRA_FREQ_START 2
#define SFRA_FREQ_LENGTH 30
//SFRA step Multiply = 10^(1/No of steps per decade(40))
#define FREQ_STEP_MULTIPLY (float)1.08
#define SFRA_AMPLITUDE (float)0.003
#elif SFRA_TYPE==SFRA_CURRENT  // current loop
#define SFRA_FREQ_START 200
#define SFRA_FREQ_LENGTH 100
//SFRA step Multiply = 10^(1/No of steps per decade(40))
```

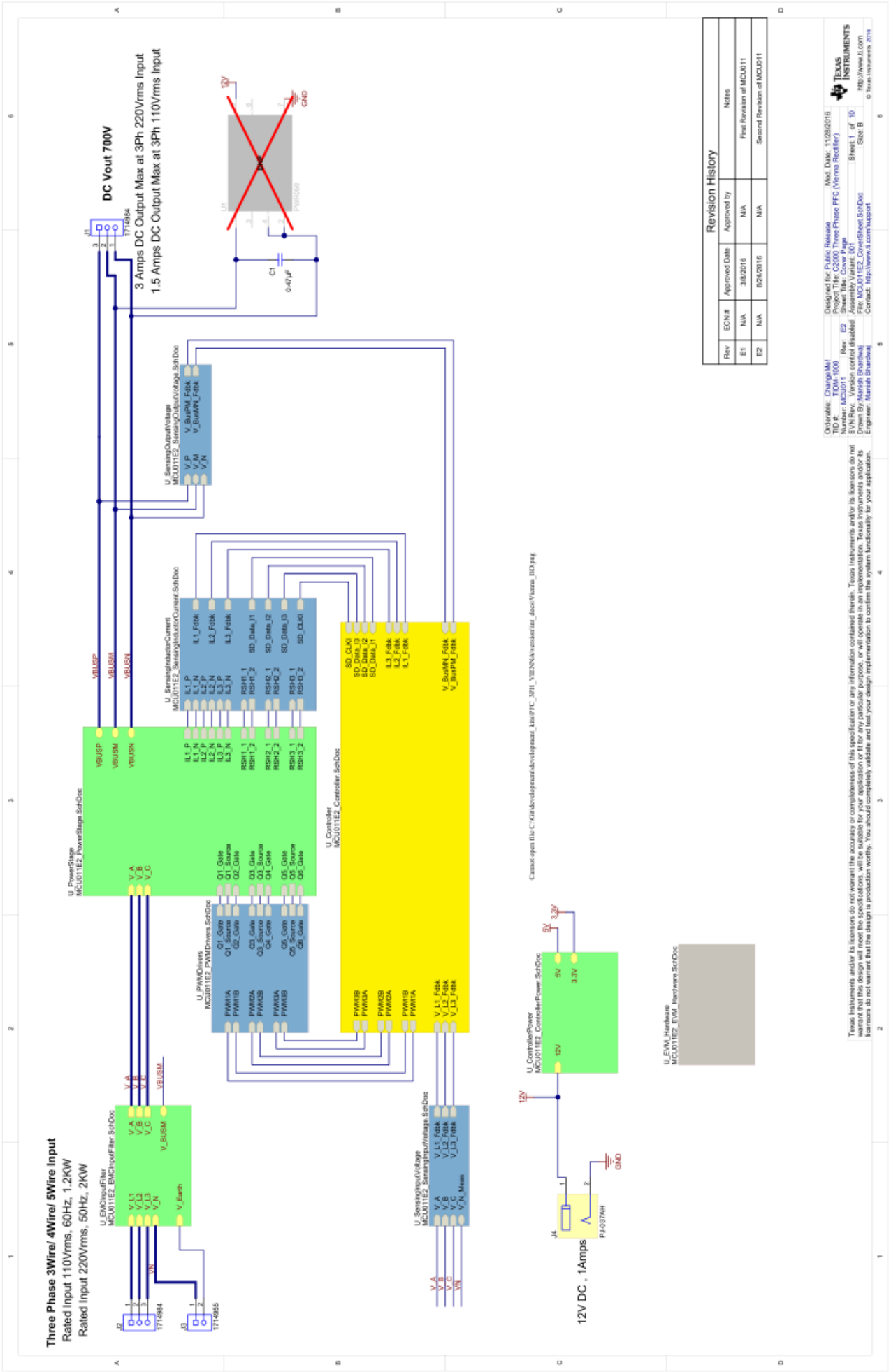
```
#define FREQ_STEP_MULTIPLY (float)1.045
#define SFRA_AMPLITUDE (float)0.005
#endif

#define UNIVERSAL_GRID_MAX_VRMS 240
#define UNIVERSAL_GRID_MIN_VRMS 80
#define UNIVERSAL_GRID_MAX_FREQ 110
#define UNIVERSAL_GRID_MIN_FREQ 40

#define SD32_PU_SCALE_FACTOR  (float) ( 1.0 /
((float)SDFM_OSR*(float)SDFM_OSR*(float)SDFM_OSR))

#define newPWM 1

#define ALL_PHASE_ENABLED 1
#define CHECK_PHASE 2

#define DUTY_MAX 0.8

//*****************************************************************************
**
//globals
//*****************************************************************************
**
// Enum for power Stage Status
enum enum_boardState {
      PowerStageOFF = 0,
      PowerStageON=1,
      TripCondition = 2,
};

enum enum_BuildLevel {
      BuildLevel1_OpenLoop = 0,
      BuildLevel2_CurrentLoop = 2,
      BuildLevel3_VoltageAndCurrentLoop = 3,
      BuildLevel4_BalanceVoltageAndCurrentLoop = 4,
};

enum enum_boardStatus {
      boardStatus_Idle = 0,
      boardStatus_NoFault=1,
      boardStatus_OverCurrentTrip = 2,
      boardStatus_OverVoltageTrip = 3,
      boardStatus_EmulatorStopTrip = 4,
};

// Measurement Variables
// Inductor Current Measurement
extern volatile float iL1MeasADC,iL2MeasADC,iL3MeasADC;
extern volatile float iL1MeasSD,iL2MeasSD,iL3MeasSD;
extern volatile float iL1Meas,iL2Meas,iL3Meas;
// Inductor Current Measurement Offset
extern volatile float iL1MeasOffset,iL2MeasOffset,iL3MeasOffset;

// Output Voltage Bus measurement
extern volatile float vBusMNMeas,vBusPMMeas, vBusMeas, vBusHalfMeas;
extern volatile float vBusMNMeasAvg,vBusPMMeasAvg, vBusMeasAvg;

// variables used for calibration of output voltage measurements
```

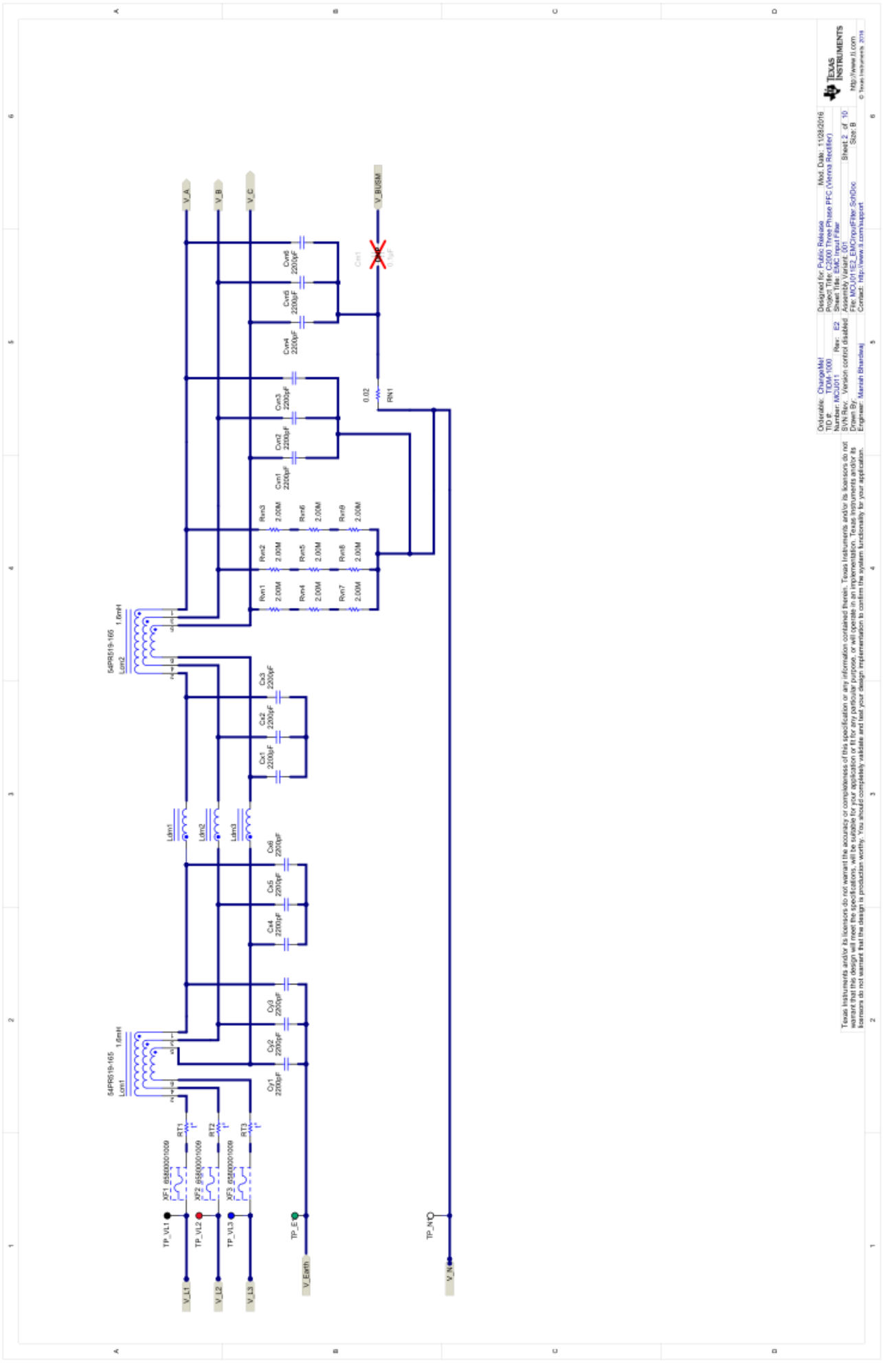
```
extern volatile float m_VBusMNMeas, b_VBusMNMeas;
extern volatile float m_VBusPMMeas, b_VBusPMMeas;

// Input Grid Voltage Measurement
extern volatile float v1Meas,v2Meas,v3Meas;
extern volatile float v1MeasOffset,v2MeasOffset,v3MeasOffset;

// Flags for clearing trips and closing the loops
extern int16_t closeGiLoop,closeGvLoop, closeGsLoop, clearTrip,
firstTimeGvLoop;

extern enum enum_BuildLevel BuildInfo;

extern enum enum_boardState boardState;

extern enum enum_boardStatus boardStatus;

//*************************************************************************
**
// the function prototypes
//*************************************************************************
**


// State Machine function prototypes
//-----------------------------------
// Alpha states
void A0(void);     //state A0
void B0(void);     //state B0
void C0(void);     //state C0

// A branch states
void A1(void);     //state A1
void A2(void);     //state A2
void A3(void);     //state A3

// B branch states
void B1(void);     //state B1
void B2(void);     //state B2
void B3(void);     //state B3


#ifdef FLASH
#pragma CODE_SECTION(controlISR,"ramfuncs");
#endif
#pragma INTERRUPT (controlISR, HPI)
interrupt void controlISR(void);
#ifdef FLASH
#pragma CODE_SECTION(tenkHzISR,"ramfuncs");
#endif
interrupt void tenkHzISR(void);
inline void clearInterrupt(void);
inline void setupInterrupt(void);
void setupSFRA();
void updateBoardStatus(void);

//TODO readCurrVolADCSignals()
inline void readCurrVolADCSignals(void)
{
```

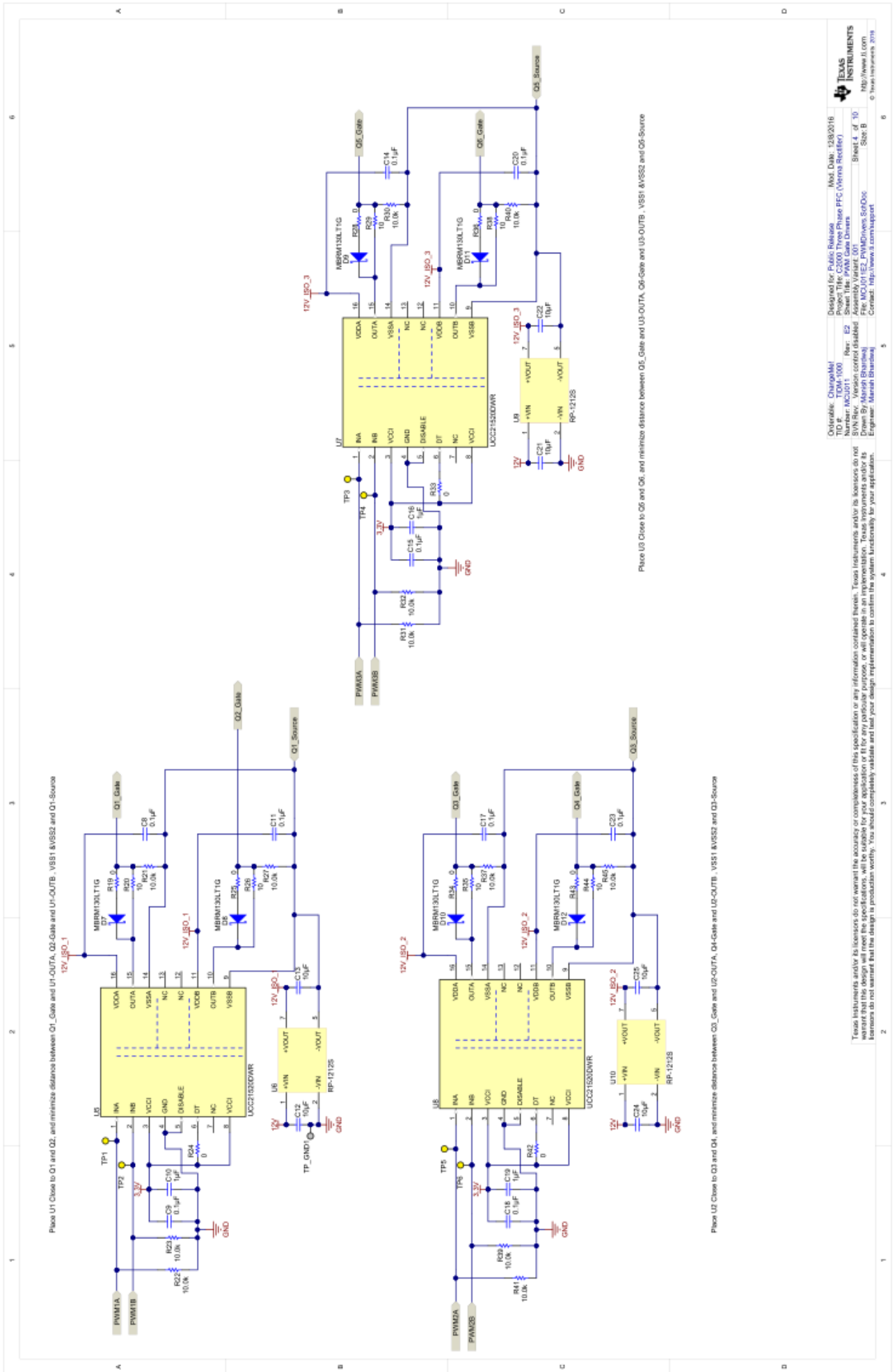
```
	/*offset165 = ((float32) (VREF165_FB))*ADC_PU_SCALE_FACTOR;*/

	iL1Meas =
(((float32)(IL1_ADC_READ1+IL1_ADC_READ2+IL1_ADC_READ3+IL1_ADC_READ4))*ADC_PU_
SCALE_FACTOR*0.25 - iL1MeasOffset )*2.0;
	iL2Meas =
(((float32)(IL2_ADC_READ1+IL2_ADC_READ2+IL2_ADC_READ3+IL2_ADC_READ4))*ADC_PU_
SCALE_FACTOR*0.25 - iL2MeasOffset )*2.0;
	iL3Meas =
(((float32)(IL3_ADC_READ1+IL3_ADC_READ2+IL3_ADC_READ3+IL3_ADC_READ4))*ADC_PU_
SCALE_FACTOR*0.25 - iL3MeasOffset )*2.0;

	v1Meas =
(((float32)(V1_ADC_READ1+V1_ADC_READ2+V1_ADC_READ3+V1_ADC_READ4))*ADC_PU_SCAL
E_FACTOR*0.25 - v1MeasOffset )*2.0;
	v2Meas =
(((float32)(V2_ADC_READ1+V2_ADC_READ2+V2_ADC_READ3+V2_ADC_READ4))*ADC_PU_SCAL
E_FACTOR*0.25 - v2MeasOffset )*2.0;
	v3Meas =
(((float32)(V3_ADC_READ1+V3_ADC_READ2+V3_ADC_READ3+V3_ADC_READ4))*ADC_PU_SCAL
E_FACTOR*0.25 - v3MeasOffset )*2.0;

	vBusPMMeas =
(((float32)(VBUSPM_ADC_READ1+VBUSPM_ADC_READ2+VBUSPM_ADC_READ3+VBUSPM_ADC_REA
D4))*ADC_PU_SCALE_FACTOR*0.25 )*VDCBUS_MAX_SENSE/VAC_MAX_SENSE;
	vBusMNMeas =
(((float32)(VBUSMN_ADC_READ1+VBUSMN_ADC_READ2+VBUSMN_ADC_READ3+VBUSMN_ADC_REA
D4))*ADC_PU_SCALE_FACTOR*0.25 )*VDCBUS_MAX_SENSE/VAC_MAX_SENSE;

	vBusPMMeas = m_VBusPMMeas*vBusPMMeas + b_VBusPMMeas; // y= mx+ b
equation used to reduce offset and gain error
	vBusMNMeas = m_VBusMNMeas*vBusMNMeas + b_VBusMNMeas;

	vBusMeas= vBusPMMeas+vBusMNMeas;
	vBusHalfMeas = (vBusMeas)*0.5;
	// clamp the vBusHalfMeas before dividing to avoid NaN
	vBusHalfMeas = (vBusHalfMeas<0.2)?0.2:vBusHalfMeas;
}

//TODO readCurrVolSDFMSignals()
inline void readCurrVolSDFMSignals(float sdfm_scale_factor)
{
	// temp variable used to read in the SD value
	int32_t sd1Flt1Read;
	int32_t sd1Flt2Read;
	int32_t sd1Flt3Read;

	sd1Flt1Read=SDFM_READ_FILTER1_DATA_32BIT;
	sd1Flt2Read=SDFM_READ_FILTER2_DATA_32BIT;
	sd1Flt3Read=SDFM_READ_FILTER3_DATA_32BIT;

	iL1MeasSD=sd1Flt1Read*sdfm_scale_factor;
	iL2MeasSD=sd1Flt2Read*sdfm_scale_factor;
	iL3MeasSD=sd1Flt3Read*sdfm_scale_factor;

}

//TODO clearPWMTrip()
inline void clearPWMTrip()
```

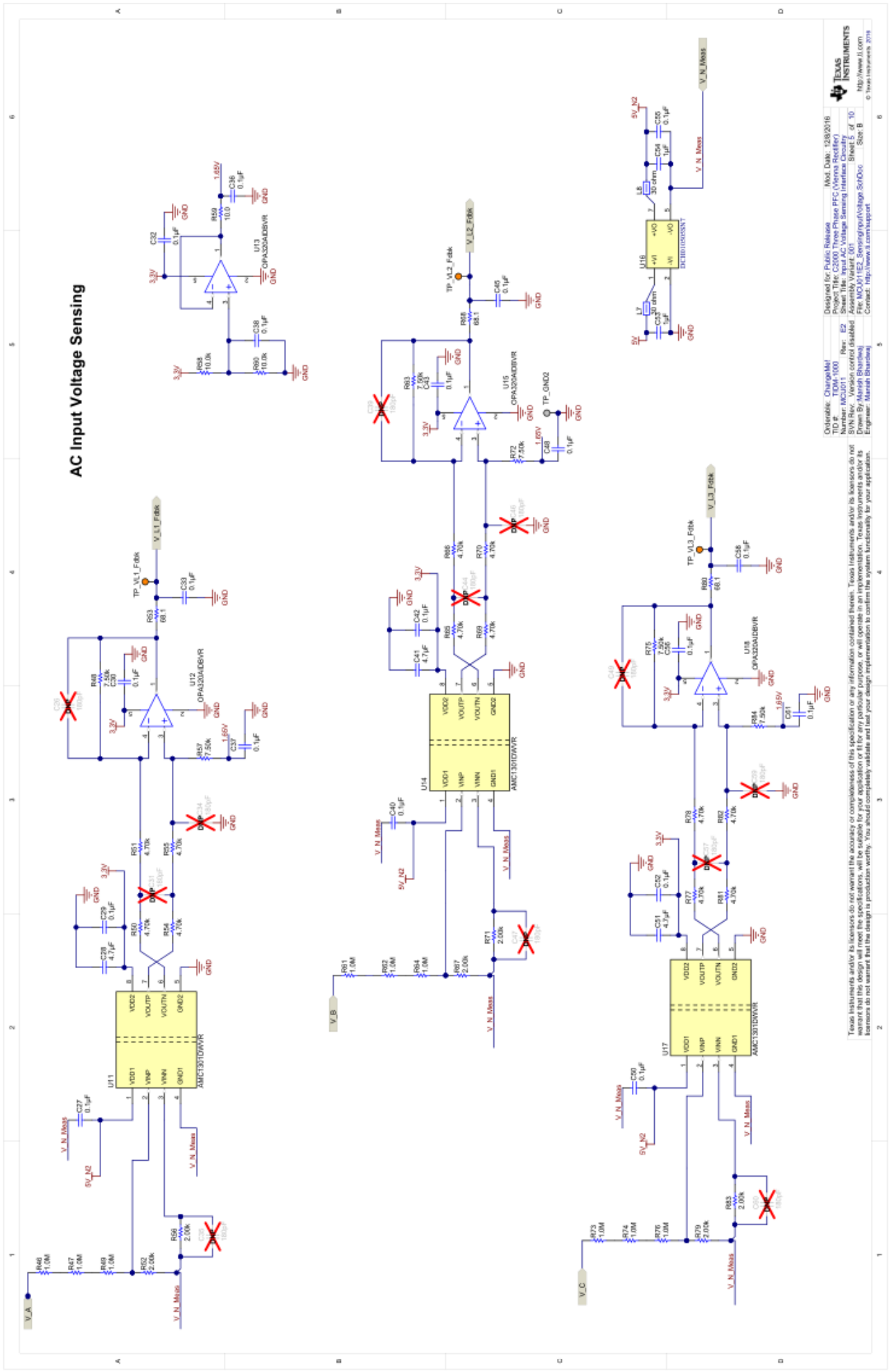
```
{

	if (clearTrip==1 )
	{
		// clear all the configured trip sources for the PWM module

		clearPWMTripFlags(PWM_NO);
		clearPWMTripFlags(PWM_NO+1);
		clearPWMTripFlags(PWM_NO+2);

		clearTrip=0;
		closeGiLoop=1;

#if INCR_BUILD==3 || INCR_BUILD==4
		closeGvLoop=1;
		closeGsLoop=1;
#else
		closeGvLoop=0;
#endif

		boardStatus= boardStatus_NoFault;
	}

}

//TODO computePIcontrollerCoeff()
// PI = (gain)*(s+z0)/s
void computePIcontrollerCoeff(CNTL_2P2Z_F_COEFFS *coef1, float gain, float
z0, float fsw)
{
	float temp1;
	temp1=gain/(2*fsw);

	coef1->Coeff_B0=(2*fsw+z0)*temp1;
	coef1->Coeff_B1=(-2*fsw+z0)*temp1;
	coef1->Coeff_B2=0;
	coef1->Coeff_A1=1;
	coef1->Coeff_A2=0;
	coef1->IMin=-1.0;
	coef1->Max=1.0;
	coef1->Min=-1.0;
}

#ifdef __cplusplus
}
#endif                                  /* extern "C" */


#endif
```

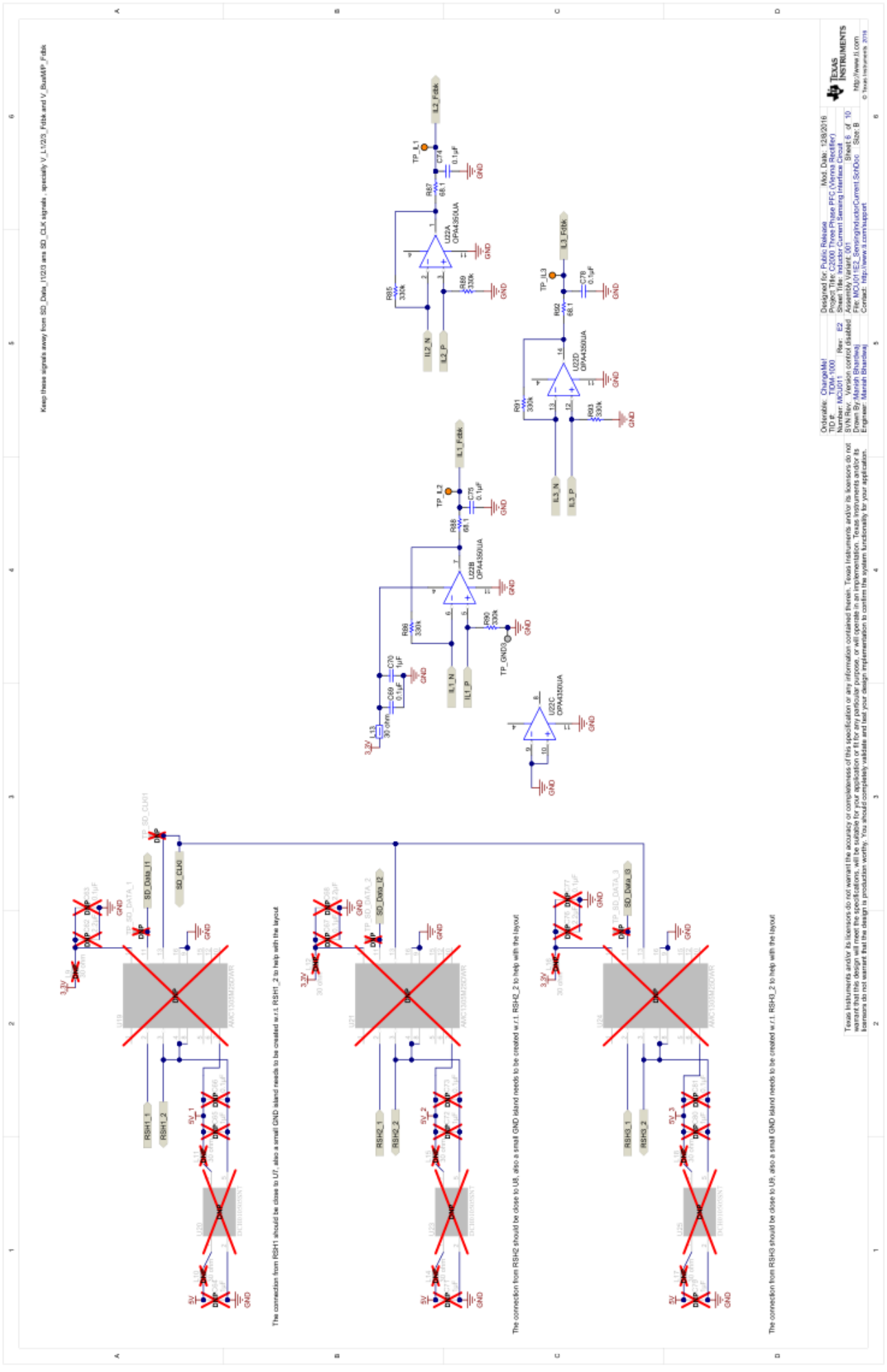
ANEXO III.ESQUEMATICOS TIDM-1000

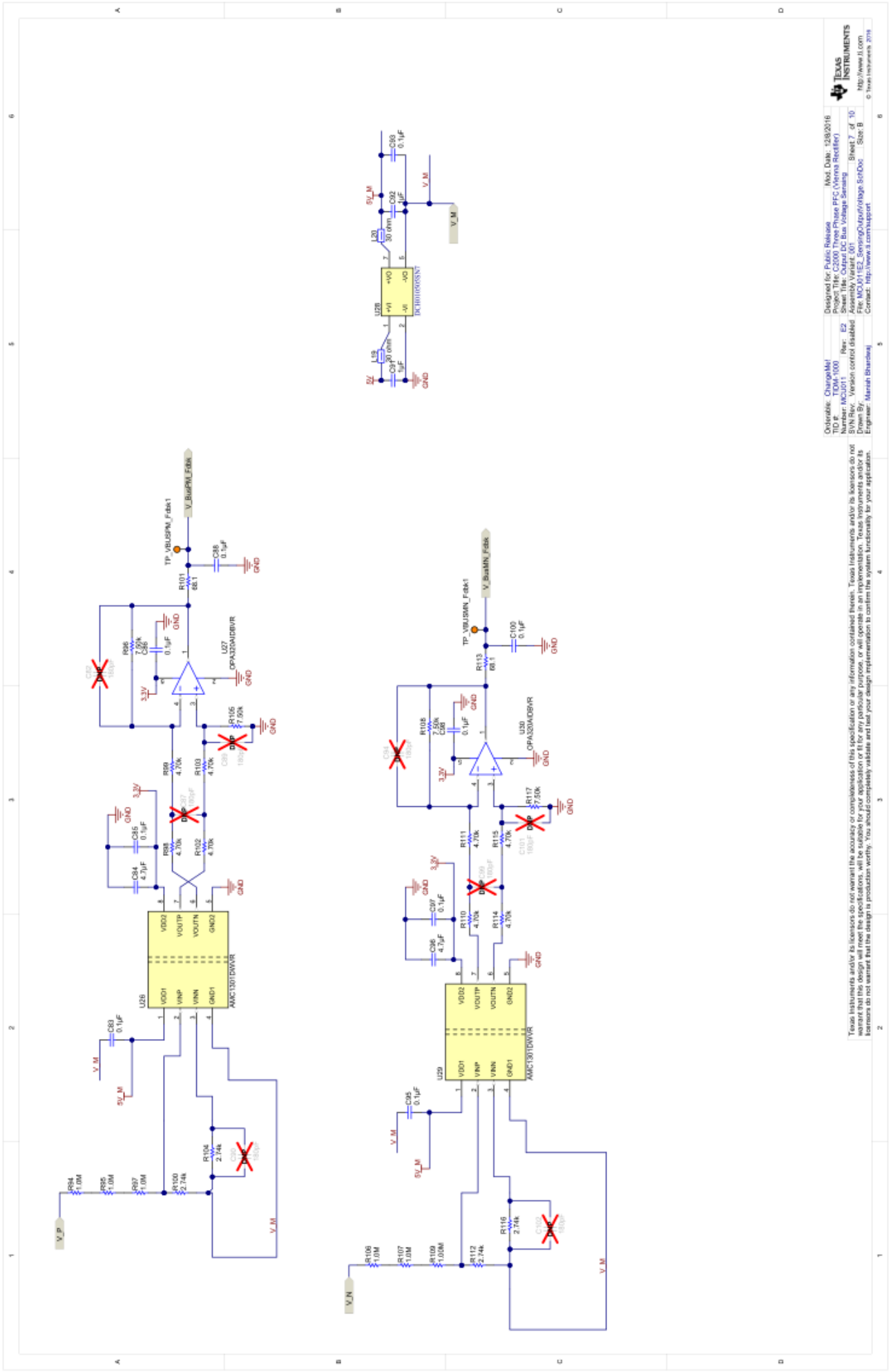

Three Phase 3Wire/ 4Wire/ 5Wire Input
Rated Input 110Vrms, 60Hz, 1.2KW
Rated Input 220Vrms, 50Hz, 2KW
DC Vout 700V
3 Amps DC Output Max at 3Ph 220Vrms Input
1.5 Amps DC Output Max at 3Ph 110Vrms Input
12V DC , 1Amps
Revision History
Rev
ECN #
Approved Date
Approved by
Notes
E1
N/A
3/8/2016
N/A
First Revision of MCU011
E2
N/A
8/24/2016
N/A
Second Revision of MCU011
U_PowerStage
U_PWMDrivers
U_Controller
U_SensingInductorCurrent
U_SensingOutputVoltage
U_SensingInputVoltage
U_EMCInputFilter
U_ControlerPower
U_EVM_Hardware

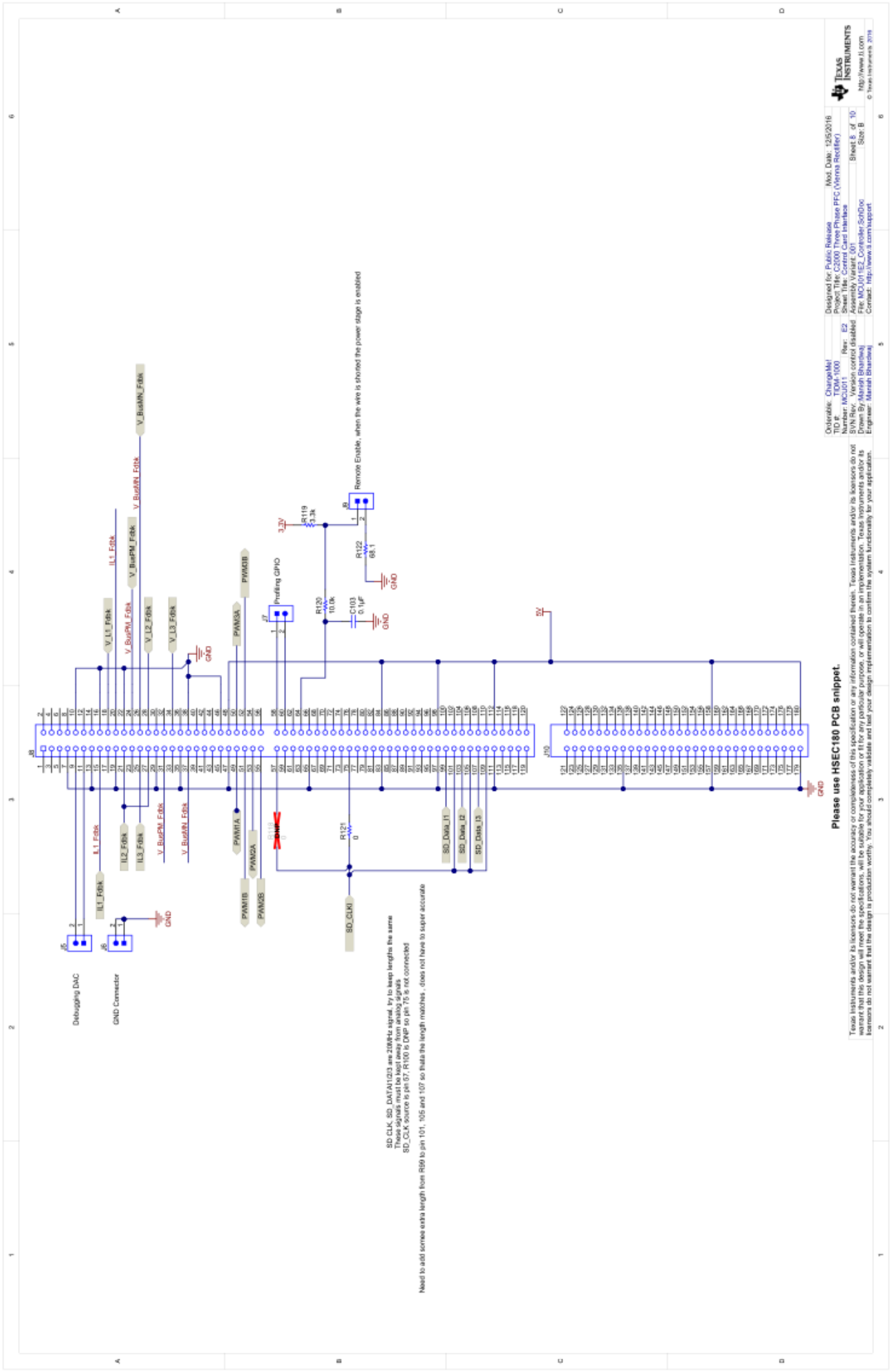

V_A
V_B
V_C
V_BUSM
Cm1
Cm2
Cm3
Cm4
Cm5
Cm6
RN1
0.02
Rvn1
Rvn2
Rvn3
Rvn4
Rvn5
Rvn6
Rvn7
Rvn8
Rvn9
Lcm1
Lcm2
Ldm1
Ldm2
Ldm3
Cx1
Cx2
Cx3
Cx4
Cx5
Cx6
Cy1
Cy2
Cy3
RT1
RT2
RT3
XF1
XF2
XF3
TP_VL1
TP_VL2
TP_VL3
TP_E
TP_N
V_L1
V_L2
V_L3
V_Earth
V_N
Designed for: Public Release
Project Title: C2000 Three Phase PFC (Vienna Rectifier)
Sheet Title: Inrush / Input Filter
Drawn By: Version control disabled
Engineer: Manish Bhardwaj
Mod. Date: 11/28/2016
Sheet 2 of 10
Size: B
Texas Instruments and/or its licensors do not warrant the accuracy or completeness of this specification or any information contained therein. Texas Instruments and/or its licensors do not warrant that this design will meet the specifications, will be suitable for your application or fit for any particular purpose, or will operate in an implementation. Texas Instruments and/or its licensors do not warrant that the design is production worthy. You should completely validate and test your design implementation to confirm the system functionality for your application.

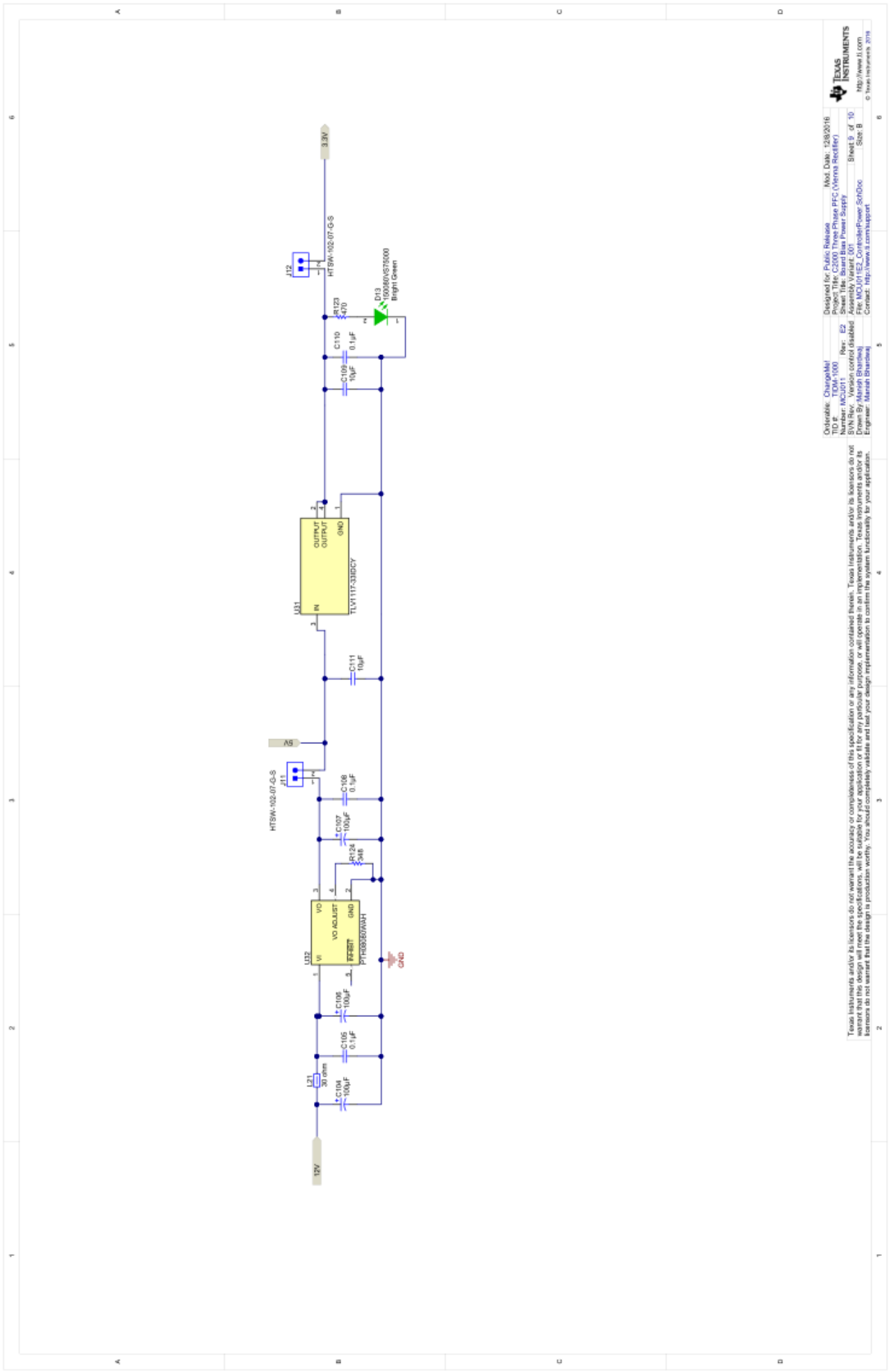

Place U1 Close to Q1 and Q2, and minimize distance between Q1_Gate and U1-OUTA, Q2-Gate and U1-OUTB , VSS1 &VSS2 and Q1-Source
Place U3 Close to Q5 and Q6, and minimize distance between Q5_Gate and U3-OUTA, Q6-Gate and U3-OUTB , VSS1 &VSS2 and Q5-Source
Place U2 Close to Q3 and Q4, and minimize distance between Q3_Gate and U2-OUTA, Q4-Gate and U2-OUTB , VSS1 &VSS2 and Q3-Source
Texas Instruments and/or its licensors do not warrant the accuracy or completeness of this specification or any information contained therein. Texas Instruments and/or its licensors do not warrant that this design will meet the specifications, will be suitable for your application or fit for any particular purpose, or will operate in an implementation. Texas Instruments and/or its licensors do not warrant that the design is production worthy. You should completely validate and test your design implementation to confirm the system functionality for your application.

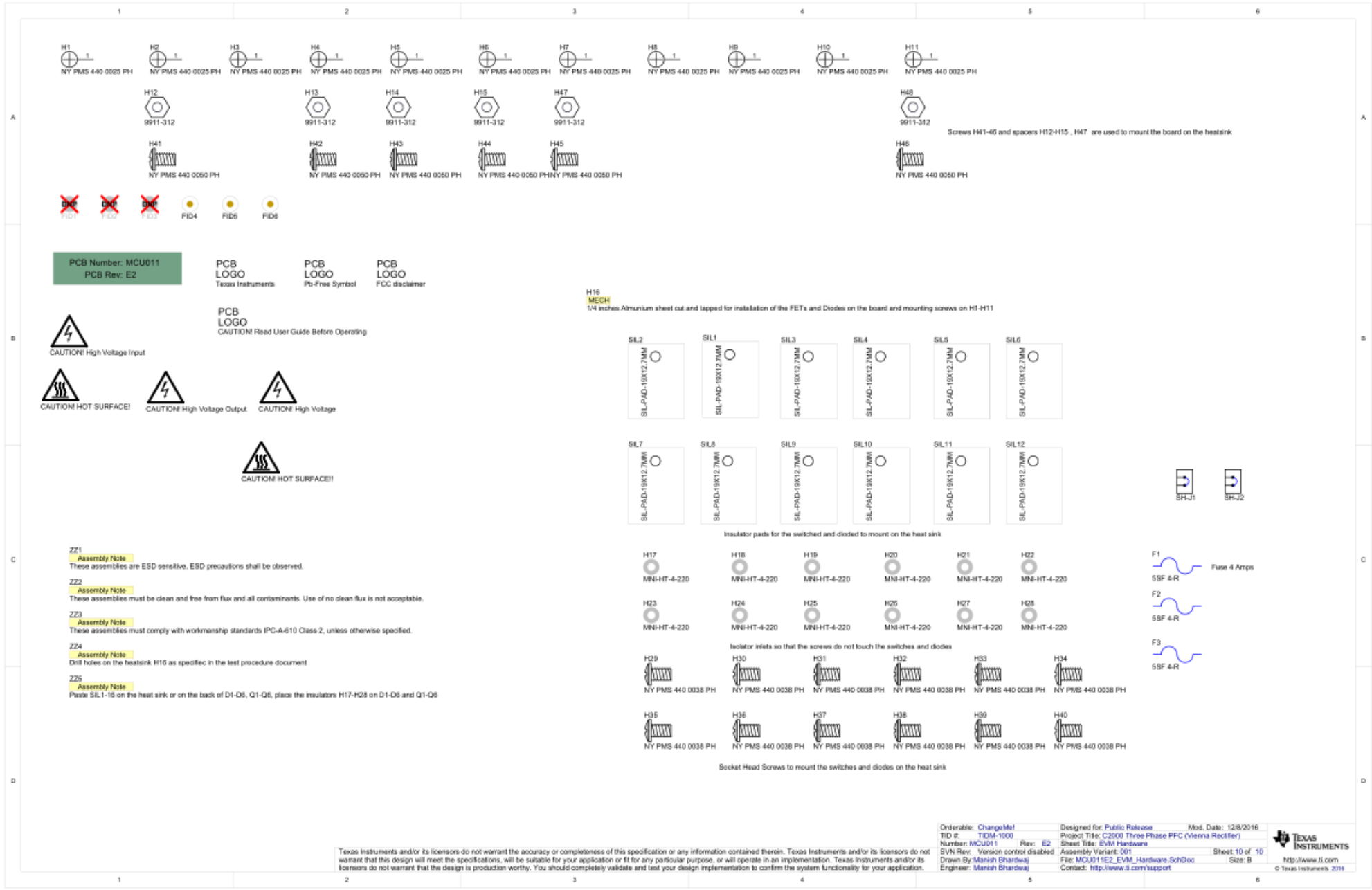

AC Input Voltage Sensing


Keep these signals away from SD_Data_I1/2/3 and SD_CLK signals - specially V_L1/2/3_Fdbk and V_BusMIP_Fdbk
The connection from RSH1 should be close to U7, also a small GND island needs to be created w.r.t. RSH1_2 to help with the layout
The connection from RSH2 should be close to U8, also a small GND island needs to be created w.r.t. RSH2_2 to help with the layout
The connection from RSH3 should be close to U9, also a small GND island needs to be created w.r.t. RSH3_2 to help with the layout
Texas Instruments and/or its licensors do not warrant the accuracy or completeness of this specification or any information contained therein. Texas Instruments and/or its licensors do not warrant that this design will meet the specifications, will be suitable for your application or fit for any particular purpose, or will operate in an implementation. Texas Instruments and/or its licensors do not warrant that the design is production worthy. You should completely validate and test your design implementation to confirm the system functionality for your application.




Please use HSEC180 PCB snippet.
Debugging DAC
GND Connector
Profiling GPIO
Remote Enable, when the wire is shorted the power stage is enabled
SD_CLK, SD_DATA1/2/3 are 20MHz signal, try to keep lengths the same
These signals must be kept away from analog signals
SD_CLK source is pin 57, R160 is DNP so pin 75 is not connected
Need to add some extra length from R89 to pin 101, 105 and 107 so that the length matches , does not have to super accurate


3.3V
J12
HTSW-102-07-G-S
D13
150060VS75000
Bright Green
R123
470
C110
0.1µF
C109
10µF
U31
TLV1117-33IDCY
OUTPUT
OUTPUT
GND
IN
C111
10µF
5V
J11
HTSW-102-07-G-S
C108
0.1µF
C107
100µF
R124
348
U32
PTH08080WAH
VO ADJUST
GND
C106
100µF
C105
0.1µF
L21
30 ohm
C104
100µF
12V
GND
Ordertitle: ChargeMe!
TID #:
Number: MCUJ011
Rev: E2
Designed for: Public Release
Project Title: C2000 Three Phase PFC (Vienna Rectifier)
Sheet Title: Board Bias Power Supply
Assembly Variant: 001
File: MCUJ011E2_ControlcardPower.SchDoc
Contact: http://www.ti.com/support
Mod. Date: 12/8/2016
Sheet 9 of 10
Size: B
TEXAS INSTRUMENTS
http://www.ti.com
© Texas Instruments 2016
Texas Instruments and/or its licensors do not warrant the accuracy or completeness of this specification or any information contained therein. Texas Instruments and/or its licensors do not warrant that this design will meet the specifications, will be suitable for your application or fit for any particular purpose, or will operate in an implementation. Texas Instruments and/or its licensors do not warrant that the design is production worthy. You should completely validate and test your design implementation to confirm the system functionality for your application.


H1 NY PMS 440 0025 PH
H2 NY PMS 440 0025 PH
H3 NY PMS 440 0025 PH
H4 NY PMS 440 0025 PH
H5 NY PMS 440 0025 PH
H6 NY PMS 440 0025 PH
H7 NY PMS 440 0025 PH
H8 NY PMS 440 0025 PH
H9 NY PMS 440 0025 PH
H10 NY PMS 440 0025 PH
H11 NY PMS 440 0025 PH
H12 9911-312
H13 9911-312
H14 9911-312
H15 9911-312
H47 9911-312
H48 9911-312
Screws H41-46 and spacers H12-H15 , H47 are used to mount the board on the heatsink
H41 NY PMS 440 0050 PH
H42 NY PMS 440 0050 PH
H43 NY PMS 440 0050 PH
H44 NY PMS 440 0050 PH
H45 NY PMS 440 0050 PH
H46 NY PMS 440 0050 PH
DNP FID1
DNP FID2
DNP FID3
FID4
FID5
FID6
PCB Number: MCU011
PCB Rev: E2
PCB LOGO Texas Instruments
PCB LOGO Pb-Free Symbol
PCB LOGO FCC disclaimer
PCB LOGO CAUTION! Read User Guide Before Operating
CAUTION! High Voltage Input
CAUTION! HOT SURFACE!
CAUTION! High Voltage Output
CAUTION! High Voltage
CAUTION! HOT SURFACE!!
H16 MECH
1/4 inches Almunium sheet cut and tapped for installation of the FETs and Diodes on the board and mounting screws on H1-H11
SIL2 SIL-PAD-19X12.7MM
SIL1 SIL-PAD-19X12.7MM
SIL3 SIL-PAD-19X12.7MM
SIL4 SIL-PAD-19X12.7MM
SIL5 SIL-PAD-19X12.7MM
SIL6 SIL-PAD-19X12.7MM
SIL7 SIL-PAD-19X12.7MM
SIL8 SIL-PAD-19X12.7MM
SIL9 SIL-PAD-19X12.7MM
SIL10 SIL-PAD-19X12.7MM
SIL11 SIL-PAD-19X12.7MM
SIL12 SIL-PAD-19X12.7MM
SH-J1
SH-J2
Insulator pads for the switched and dioded to mount on the heat sink
H17 MNI-HT-4-220
H18 MNI-HT-4-220
H19 MNI-HT-4-220
H20 MNI-HT-4-220
H21 MNI-HT-4-220
H22 MNI-HT-4-220
H23 MNI-HT-4-220
H24 MNI-HT-4-220
H25 MNI-HT-4-220
H26 MNI-HT-4-220
H27 MNI-HT-4-220
H28 MNI-HT-4-220
Isolator inlets so that the screws do not touch the switches and diodes
F1 SSF 4-R Fuse 4 Amps
F2 SSF 4-R
F3 SSF 4-R
H29 NY PMS 440 0038 PH
H30 NY PMS 440 0038 PH
H31 NY PMS 440 0038 PH
H32 NY PMS 440 0038 PH
H33 NY PMS 440 0038 PH
H34 NY PMS 440 0038 PH
H35 NY PMS 440 0038 PH
H36 NY PMS 440 0038 PH
H37 NY PMS 440 0038 PH
H38 NY PMS 440 0038 PH
H39 NY PMS 440 0038 PH
H40 NY PMS 440 0038 PH
Socket Head Screws to mount the switches and diodes on the heat sink
ZZ1 Assembly Note
These assemblies are ESD sensitive, ESD precautions shall be observed.
ZZ2 Assembly Note
These assemblies must be clean and free from flux and all contaminants. Use of no clean flux is not acceptable.
ZZ3 Assembly Note
These assemblies must comply with workmanship standards IPC-A-610 Class 2, unless otherwise specified.
ZZ4 Assembly Note
Drill holes on the heatsink H16 as specified in the test procedure document
ZZ5 Assembly Note
Paste SIL1-16 on the heat sink or on the back of D1-D6, Q1-Q6, place the insulators H17-H28 on D1-D6 and Q1-Q6
Texas Instruments and/or its licensors do not warrant the accuracy or completeness of this specification or any information contained therein. Texas Instruments and/or its licensors do not warrant that this design will meet the specifications, will be suitable for your application or fit for any particular purpose, or will operate in an implementation. Texas Instruments and/or its licensors do not warrant that the design is production worthy. You should completely validate and test your design implementation to confirm the system functionality for your application.
Orderable: ChangeMe! Designed for: Public Release Mod. Date: 12/8/2016
TID #: TIDM-1000 Project Title: C2000 Three Phase PFC (Vienna Rectifier)
Number: MCU011 Rev: E2 Sheet Title: EVM Hardware
SVN Rev: Version control disabled Assembly Variant: 001 Sheet: 10 of 10
Drawn By: Manish Bhardwaj File: MCU011E2_EVM_Hardware.SchDoc Size: B
Engineer: Manish Bhardwaj Contact: http://www.ti.com/support
Texas Instruments http://www.ti.com
© Texas Instruments 2016