



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO DE FINAL DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

VOLUMEN I: MEMORIA

Diseño e implementación de un inversor trifásico dual para tracción eléctrica

Autor:

David Redondo

Director:

Prof. Alfonso Conesa Roca

Convocatoria: Junio 2024



Resumen

Este Trabajo de Fin de Grado se enfoca en el diseño y prototipado de un inversor trifásico dual de 80 kW (2x40 kW) y 600 V con control vectorial (FOC) para motores síncronos de imanes permanentes (PMSMs). Este inversor bidireccional busca ser una solución compacta y de alto rendimiento para aplicaciones de tracción eléctrica en vehículos de Formula Student.

El proyecto ha alcanzado una notable densidad de potencia de 30 kW/L mediante la implementación de tecnologías de vanguardia como los semiconductores de carburo de silicio (SiC). Se ha trabajado para lograr estos objetivos mediante un diseño que considera la correcta gestión térmica, la disposición eficiente de los componentes y la selección adecuada de conectores, entre otros aspectos fundamentales.

Además, el código del inversor permite controlar de forma completamente independiente dos motores con un solo MCU, e implementa un lazo de control de par que permite operar en la región de debilitamiento de campo. El control eficiente de la máquina eléctrica permite maximizar la extracción de potencia en un amplio rango de velocidades, llegando de forma segura a sus límites operativos.

Abstract

This Bachelor's Thesis focuses on the design and prototyping of an 80 kW dual three-phase inverter (2x40 kW) operating at 600 V, equipped with field-oriented control (FOC) for Permanent Magnet Synchronous Motors (PMSMs). This bidirectional inverter aims to serve as a compact, high-performance solution for electric traction applications in Formula Student vehicles.

The project has achieved a remarkable power density of 30 kW/L by incorporating state-of-the-art technologies such as silicon carbide (SiC) semiconductors. Efforts have been directed towards realizing a design that meticulously addresses thermal management, efficient component arrangement, and optimal connector selection, among other critical considerations.

Moreover, the inverter's code allows for the completely independent control of two motors with a single MCU, and implements a torque control loop enabling operation within the field weakening region. By ensuring efficient control of the electric machine, the system maximizes power extraction across a broad range of speeds, safely reaching its operational limits.

Agradecimientos

A mi familia y amistades, de quien siempre recibí el apoyo que necesitaba, a mi tutor Alfonso Conesa, por los muy necesarios consejos, al CITCEA-UPC, por todo el apoyo técnico y a e-Tech Racing y las personas que lo forman por hacer de un taller un hogar.

Glosario

1D One-Dimensional — Unidimensional

2D Two-Dimensional — Bidimensional

3D Three-Dimensional — Tridimensional

AC Alternating Current — Corriente alterna

BEMF Back Electromotive Force — Fuerza contra-electromotriz

CS Constant Speed — Velocidad constante

CT Constant Torque — Par constante

CVL Current and Voltage Limits — Límites de corriente y voltaje

DC Direct Current — Corriente continua

DTC Direct Torque Control — Control de par directo

ECU Electronic Control Unit — Unidad de control electrónico

EM Electric Machine — Máquina eléctrica

EMI Electromagnetic Interference — Interferencia electromagnética

EMR Energetic Macroscopic Representation — Representación macroscópica de la energía

EV Electric Vehicle — Vehículo eléctrico

FOC Field Oriented Control — Control orientado al campo

FW Flux Weakening — Debilitamiento del flujo

GaN Gallium Nitride — Nitruro de galio

HW Hardware — Hardware

I Integral — Integral

ICE Internal Combustion Engine — Motor de combustión interna

IGBT Insulated Gate Bipolar Transistor — Transistor Bipolar de Puerta Aislada

IM Induction Machine — Máquina de inducción

IPMSM Interior Permanent Magnet Synchronous Machine — Máquina síncrona de imanes permanentes interiores

LBEMF Low Back Electromotive Force — Baja fuerza contra-electromotriz

LPF Low-Pass Filter — Filtro pasa-bajos

LUT Look-Up Table — Tabla de búsqueda

MOSFET Metal Oxide Semiconductor Field-Effect Transistor — Transistor de efecto de campo semiconductor de óxido metálico

MTPA Maximum Torque Per Ampere — Par máximo por amperio

MTPV Maximum Torque Per Volt — Par máximo por voltio

NPC Neutral Point Clamped — Punto neutro fijado

PI Proportional-Integral — Proporcional-Integral

PID Proportional-Integral-Derivative — Proporcional-Integral-Derivativo

PLL Phase-Locked Loop — Lazo de fase bloqueada

PMSM Permanent Magnet Synchronous Machine — Máquina síncrona de imanes permanentes

PWM Pulse Width Modulation — Modulación de ancho de pulso

RMS Root Mean Square — Media cuadrática

Si Silicon — Silicio

SiC Silicon Carbide — Carburo de silicio

SoC State of Charge — Estado de carga

SPMSM Surface Permanent Magnet Synchronous Machine — Máquina síncrona de imanes permanentes superficiales

SPWM Sinusoidal Pulse Width Modulation — Modulación de ancho de pulso sinusoidal

SVPWM Space Vector PWM — PWM por vector espacial

SW Software — Software

T-NPC T-type Neutral Point Clamped — Punto neutro fijado tipo T

THD Total Harmonic Distortion — Distorsión armónica total

VSI Voltage Source Inverter — Inversor de fuente de voltaje

WBG Wide Bandgap — Banda prohibida ancha

Índice general

1. Introducción y Objetivos	9
1.1. Introducción	9
1.2. Objetivos	9
1.3. Contexto y justificación	10
1.4. Aplicación práctica	10
2. Estado del Arte	12
3. Metodología	13
3.1. Modelo en V	13
3.2. Gestión del Proyecto con Git	16
3.2.1. Introducción a Git	16
3.2.2. Funcionamiento de Git	16
3.2.3. Por qué utilizar un sistema de control de versiones	16
3.2.4. Uso de Git en el proyecto	17
3.2.5. GitHub	17
3.2.6. Automatización de la documentación y <i>releases</i>	17
4. Desarrollo	18
4.1. Modelo matemático del PMSM	18
4.1.1. Marco de referencia estático	18
4.1.2. Representación en el espacio vectorial	19
4.1.3. Transformadas	20
Transformada de Clarke ortonormal	20
Transformada de Clarke de Amplitud Constante	22
Transformada de Park	23
4.1.4. Marco de referencia rotatorio	24
4.1.5. Curvas características del PMSM	26
Curva de par-velocidad y potencia-velocidad	26
CLC (Círculo de límite de corriente)	28
TH (Hipérbolas de par)	29
VLE (Elipses de límite de voltaje)	30
4.2. Control del PMSM en el espacio d-q	31
4.2.1. Trayectorias de control	31
MTPA (Máximo Par por Amperio)	31
CTC (Curva de Par Constante)	32
MTPV (Máximo Par por Voltio)	33
CVL (Límites de Corriente y Voltaje)	34
4.2.2. Diseño y simulación del control	36
EMR (Representación macroscópica energética)	36
Lazos de corriente y modelo promediado del inversor	38
Implementación de las trayectorias de control	40
Modelo comutado	43

4.3. <i>Hardware</i>	52
4.3.1. Requisitos y pre-concepto	52
Potencia	52
Velocidad	54
Normativa	55
Resumen de Requisitos de <i>Hardware</i>	58
Requisitos Adicionales	59
Boceto del empaquetado	59
4.3.2. Topología y concepto	61
4.3.3. Semiconductores	63
Tecnología de semiconductores	63
Módulos de potencia	63
Análisis de pérdidas	64
4.3.4. Sistema de refrigeración	70
4.3.5. <i>Gate drivers</i>	71
Funcionamiento genérico	71
Criterios de selección	72
Comparativa de alternativas	72
Cálculos del <i>gate driver</i> seleccionado	73
4.3.6. Bus de condensadores	75
Función del bus de condensadores	75
Dimensionamiento del bus de condensadores	76
Selección de condensadores	77
4.3.7. Conectores de potencia	78
4.3.8. Sensor de posición	79
4.3.9. Microcontrolador	79
<i>Software</i> de CAD electrónico	81
4.3.10. PCB de potencia	81
Concepto y <i>layout</i>	81
Restricciones y enrutado	83
Bloques funcionales	86
Circuitos importantes	87
Resultado final	94
4.3.11. PCB de control	95
Concepto y <i>layout</i>	95
Restricciones y enrutado	96
Bloques funcionales	97
Configuración de <i>hardware</i> del MCU	98
Circuitos importantes	99
Resultado final	107
4.3.12. Ensamblaje del convertidor	107
Diseño en CAD	107
Ensamblaje real	108
4.4. <i>Firmware</i>	113
4.4.1. Desarrollo del <i>firmware</i>	113
Objetivos del <i>firmware</i>	113
Herramientas utilizadas	113

Plataforma de desarrollo	115
Lenguaje de programación	115
Estilo de programación	116
4.4.2. Arquitectura del <i>firmware</i>	118
Estructura del <i>firmware</i>	118
Módulo A	118
Módulo B	118
Módulo C	118
Módulo D	118
Módulo E	118
4.4.3. Implementación del <i>firmware</i>	118
Configuración del MCU	118
Manejo de interrupciones	118
Algoritmos de control	118
5. Resultados y validación	119
5.1. Introducción	119
5.2. Validación de <i>hardware</i>	119
5.2.1. Pre-inspección de la placa de potencia	119
5.2.2. Primer contacto con la placa de potencia	121
Alimentación del <i>gate driver</i>	121
Funcionalidad del <i>gate driver</i>	123
Sensado de corriente	123
Descarga	124
Sensado de voltaje	125
Resumen de errores encontrados y revisión de la PCB	126
5.2.3. Conmutación de una rama como DC-DC síncrono	127
5.2.4. Pruebas térmicas y de alta tensión	127
5.2.5. Validación de la placa de control	127
5.3. Validación de <i>firmware</i>	127
5.3.1. Verificación de los periféricos	127
5.3.2. Conmutación de las tres ramas	127
5.3.3. Lazo abierto de tensión con carga R-L	127
5.3.4. Lazo abierto de tensión con un motor	127
5.3.5. Lazo cerrado de corriente con carga R-L	127
5.3.6. Adquisición de la posición del motor	127
5.3.7. Lazo cerrado de corriente con un motor	127
5.3.8. Trayectorias de control	127
5.3.9. Control dual	127
5.4. Integración	127
A. Apéndices	1
A.1. Obtención de los Parámetros de un PMSM	1
A.2. Documentación del <i>hardware</i>	2
A.3. Documentación del <i>firmware</i>	28
B. Bibliografía	273

1. Introducción y Objetivos

1.1. Introducción

La Formula Student es una competición internacional que desafía a estudiantes de ingeniería de todo el mundo a diseñar, construir y competir con monoplazas diseñados, construidos y pilotados por los mismos estudiantes. Esta competición proporciona una plataforma excepcional para que los futuros ingenieros pongan en práctica sus conocimientos y adquieran experiencia práctica en todos los ámbitos de la ingeniería, desde la manufactura hasta la gestión de proyectos. El énfasis en la innovación, la eficiencia y la adaptabilidad aporta un valor incalculable a la formación de los jóvenes ingenieros.

Dentro de los puntos clave de la Formula Student en los últimos años, se encuentra el interés creciente en la tracción eléctrica. En un mundo cada vez más preocupado por la sostenibilidad y la eficiencia energética, los motores eléctricos han surgido como una opción atractiva para la propulsión de vehículos. Este cambio de paradigma plantea la cuestión fundamental de cómo diseñar y controlar eficazmente estos sistemas eléctricos para lograr un alto rendimiento sin dejar de ser accesibles para el público general. La optimización de costes no suele ser un reto en los deportes de motor, incluida la Formula Student, pero la innovación que se lleva a cabo en estos contextos de libertad absoluta permite traer ideas de las competiciones a los vehículos de calle. En este contexto, se establece un puente fundamental entre el presente y el futuro de la movilidad sostenible, explorando los elementos técnicos que impulsan el rendimiento de los motores eléctricos y su control, contribuyendo a dar forma al panorama de la movilidad del mañana. Además, la tracción eléctrica pura no es la única rama de la industria beneficiada por ingenieros conocedores de estos sistemas. Los trenes de potencia híbridos, los generadores en centrales energéticas e incluso los sistemas de gestión de la energía en hogares pueden volverse mucho más eficientes gracias a la investigación y el conocimiento en baterías, motores eléctricos, electrónica de potencia e integración.

Este proyecto se sitúa en el corazón de esta revolución en los deportes de motor, donde la ingeniería se fusiona con la sostenibilidad y la competición para forjar una nueva generación de soluciones de tracción eléctrica. A través de un enfoque riguroso en el diseño y control de motores eléctricos y controladoras, este proyecto busca avanzar en el conocimiento y la aplicación de tecnologías de vanguardia, contribuyendo así a la formación de ingenieros y al desarrollo de soluciones de movilidad más ecológicas.

1.2. Objetivos

El objetivo principal de este proyecto es diseñar y prototipar un inversor trifásico dual de alto rendimiento para motores síncronos de imanes permanentes (PMSM), aplicado al entorno de la Formula Student. Los objetivos específicos son:

- Adquirir **conocimiento** sobre control de motores eléctricos y diseño de convertidores de potencia.
- Definir unos **requisitos** para el inversor de tracción ideal para el equipo e-Tech Racing de la UPC-EEBE.
- **Diseñar** un inversor de tracción en base a esos requisitos.
- Asegurar la **integración** del inversor con los futuros monoplazas del equipo e-Tech Racing de la UPC-EEBE.
- Implementar un **control vectorial** (FOC) que permita el control independiente de dos motores **dos motores** con un solo microcontrolador (MCU).
- Evaluar y **validar** el rendimiento del inversor.

1.3. Contexto y justificación

El desarrollo de sistemas de control para motores eléctricos no es una tarea trivial y demanda un profundo conocimiento sobre electrónica de potencia, las características del motor, la teoría del control y la programación de dispositivos electrónicos.

La electrónica de potencia está viviendo grandes avances para aplicaciones de todos los rangos de potencia. Los dispositivos semiconductores de carburo de silicio (SiC) están permitiendo mayores densidades de potencia en convertidores que van desde los centenares de vatios hasta alcanzar el megavatio, debido a la reducción de pérdidas y la alta conductividad térmica del material en comparación con sus equivalentes de silicio tradicional. De forma similar, los dispositivos de nitruro de galio (GaN) consiguen miniaturizar convertidores desde unos pocos vatios hasta casi el kilovatio. Otros avances en componentes, como la tecnología de condensadores de película, aceleran el proceso de miniaturización, permitiendo conseguir densidades de potencia más elevadas. Junto a estos dispositivos de nueva generación, avances en las técnicas de control y modulación permiten reducir las pérdidas y aumentar la eficiencia, reduciendo los requisitos de gestión térmica y permitiendo diseñar convertidores aún más pequeños.

1.4. Aplicación práctica

En este trabajo se explorará el diseño y control de un inversor trifásico dual de alto rendimiento para motores PMSM, aplicado al entorno de la Formula Student. Se abordarán los aspectos técnicos y prácticos de este proyecto, desde la selección de componentes hasta la implementación de algoritmos de control avanzados.

En particular, el diseño de esta controladora tendrá en cuenta las necesidades específicas de e-Tech Racing, el equipo de Formula Student de la UPC-EEBE. Desde su fundación en 2013, este equipo ha construido monoplazas año tras año para competir en eventos en España, República Checa, Italia, Holanda y Alemania.



Figura 1.1: ETR-08, el monoplaza con el que e-Tech Racing participó en las competiciones del verano de 2023.

Tras cinco años de evolución de la misma plataforma, el equipo se encuentra diseñando y construyendo un nuevo concepto en el momento en que se redacta este trabajo. El cambio principal son los motores eléctricos, cuyo nuevo diseño permitirá integrarlos en las propias ruedas del monoplaza debido a su compacticidad, liberando así espacio del chasis. Se usarán dos motores, uno para cada rueda trasera, aunque el plan a largo plazo es implementar otros dos motores en las ruedas delanteras. Para controlarlos, se utilizarán dos inversores Bamocar D3 700-400 de la empresa alemana Unitek. Estos inversores están sobredimensionados en potencia y ocupan un espacio considerable dentro del chasis. Además, existen limitaciones en los parámetros modificables, lo que impide programar un control óptimo para el motor.

Para que el inversor sea fácilmente integrable con los futuros monoplazas del equipo, necesita contar con algunos componentes usados actualmente por el equipo. Por ejemplo, se usará el mismo microcontrolador que para el resto de ECUs, los mismos conectores de potencia y comunicación, y PCBs de estilos similares al resto de circuitos del monoplaza, con tal de que los fabricantes que colaboran con el equipo puedan fabricar todos los circuitos impresos del inversor. Esto permitirá una integración fluida del inversor como una ECU más del monoplaza, con el añadido de la electrónica de potencia personalizada.

2. Estado del Arte

esto ya en su momento

3. Metodología

En este capítulo se detalla la metodología seguida para el desarrollo del proyecto. Se incluyen las etapas del modelo en V y se describe el uso de un repositorio de GitHub para gestionar el proyecto.

3.1. Modelo en V

El modelo en V es un enfoque de desarrollo y validación que organiza las etapas del proyecto en forma de una 'V' invertida, donde cada etapa de desarrollo tiene una contraparte de validación. Esto asegura que la validación se considere desde el principio del proyecto y que cada fase de desarrollo tenga su correspondiente prueba o verificación asociada.

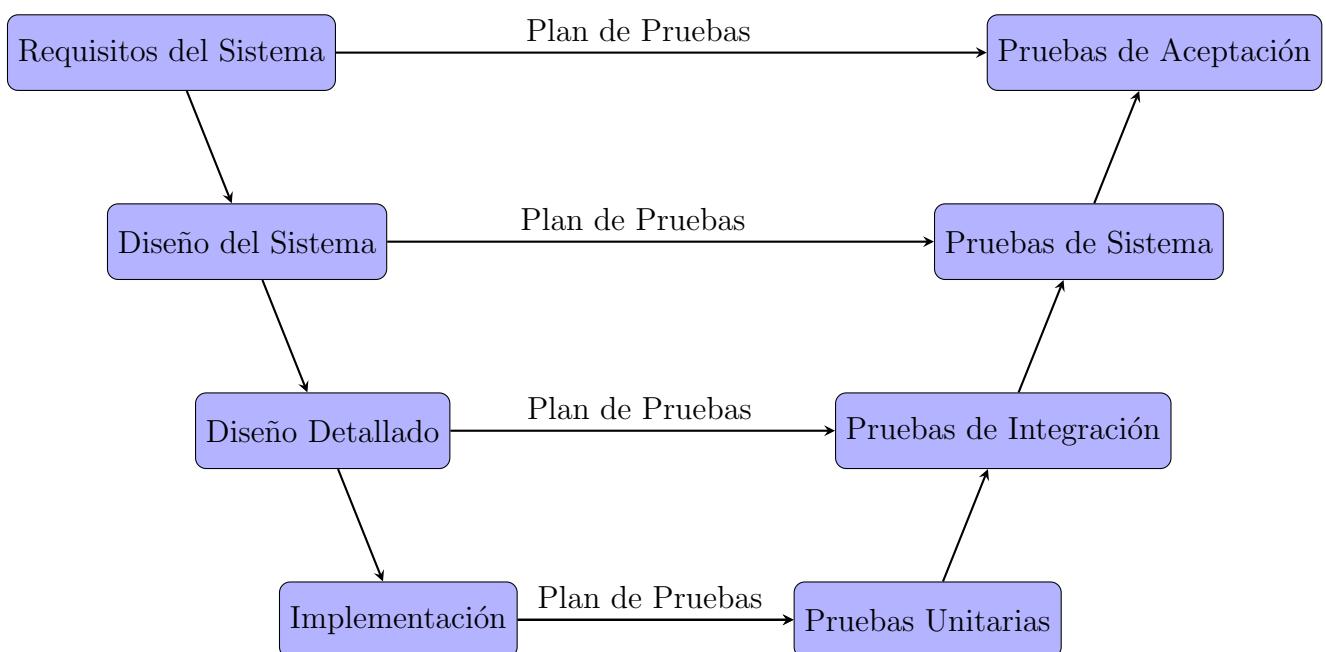


Figura 3.1: Modelo en V para el desarrollo y validación de sistemas.

En el modelo en V, las etapas de desarrollo se encuentran en el lado izquierdo de la 'V', comenzando desde los requisitos del sistema hasta la implementación y codificación. Por otro lado, las etapas de validación se encuentran en el lado derecho de la 'V', comenzando desde las pruebas unitarias y de integración hasta las pruebas de sistema y aceptación.

A continuación, se detallan las diferentes etapas del modelo en V:

- **Requisitos del Sistema:** En esta etapa se definen y documentan todos los requisitos del sistema, que sirven como la base para el diseño y desarrollo del

convertidor. Estos requisitos incluyen características eléctricas, interfaces, restricciones mecánicas, etc.

- **Diseño del Sistema:** Aquí se traducen los requisitos del sistema en un diseño detallado del convertidor. Se definen las arquitecturas de *hardware* y *software*, así como los diagramas de bloques y especificaciones técnicas. También se escogen los componentes más importantes.
- **Diseño Detallado:** En esta etapa se realiza un diseño profundo de cada parte y subsistema del convertidor. Se diseñan los esquemas eléctricos y circuitos impresos, se escriben los algoritmos de control y se crean listas de materiales para preparar el ensamblado.
- **Implementación:** Aquí se lleva a cabo la implementación física del diseño. Se fabrican los circuitos impresos, se programa el *firmware* y se ensamblan las piezas.
- **Pruebas Unitarias:** En esta etapa se realizan pruebas individuales en cada componente o subcircuito o módulo del convertidor para verificar su funcionamiento según lo especificado. Se asegura que cada unidad funcione correctamente antes de la integración.
- **Pruebas de Integración:** Se combinan los subcircuitos individuales para formar subsistemas y se verifican las interfaces entre ellos. Se asegura que los subsistemas funcionen correctamente juntos y se detectan posibles problemas de compatibilidad.
- **Pruebas de Sistema:** Aquí se evalúa el sistema completo en su conjunto. Se realizan pruebas de funcionamiento global.
- **Pruebas de Aceptación:** Finalmente, se llevan a cabo las pruebas finales en las que se evalúan los requisitos marcados inicialmente.

El modelo en V proporciona una estructura clara y sistemática para el desarrollo y validación del convertidor, asegurando que cada etapa tenga su correspondiente prueba de validación y que los resultados sean coherentes con los objetivos del proyecto. Sin embargo, es necesario ser pragmático y eficaz con la validación, puesto que tiende a llevar más tiempo del necesario si se sigue una metodología de forma estricta. Por ello, a lo largo de todo el proceso de diseño y de verificación se ha tomado la libertad de usar el modelo en V para aquello para lo que es útil.

A continuación se enumeran las etapas del proyecto, aplicando pragmáticamente el modelo en V a este trabajo en específico.

1. Definición de requisitos

La primera etapa del proyecto se centra en la definición de los requisitos del inversor trifásico y su control. Aquí se establecen los objetivos, las especificaciones técnicas y los criterios de rendimiento que guiarán todo el desarrollo. Además, se identifican las necesidades y expectativas del equipo de Formula Student, asegurando que el proyecto cumpla con sus requerimientos específicos. La duración de esta etapa es excepcionalmente larga, pues requiere de mucha familiaridad con el entorno de la Formula Student y conocimiento sobre las necesidades reales del equipo.

2. Modelo continuo y simulación del control

La siguiente etapa es el diseño del modelo en continuo y el desarrollo del control en Simulink. Aquí, se crea un modelo matemático del inversor y del motor PMSM y se implementa el control vectorial (FOC). Este proceso implica una comprensión profunda de la teoría detrás de los motores eléctricos y el diseño del control. Se basará en la representación de la energía macroscópica (EMR) con el fin de ilustrar la aplicación final del motor. La duración estimada para esta fase es de aproximadamente 2 meses.

3. Discretización del modelo y simulación de la conmutación

Al acabar la etapa anterior, se trabaja en la discretización del modelo y la simulación de la conmutación de los interruptores de potencia en PLECS. Aquí se tiene en cuenta la naturaleza discreta de la electrónica de potencia y se simula el comportamiento del inversor en el dominio del tiempo discreto. Además, esta simulación incorpora también el modelo térmico, con lo que se pueden extraer las pérdidas del inversor. Esta fase dura alrededor de 2 semanas, pues gran parte de lo ya modelado se puede reutilizar para el nuevo modelo discreto.

4. Desarrollo del *hardware*

Con el diseño del control y la simulación de la conmutación como base, se procede al diseño del *hardware*. Esto implica seleccionar componentes, diseñar esquemáticos y PCB y diseñar los tests de validación de *hardware*. La duración estimada para esta etapa es de 3 a 4 meses, y se solapa parcialmente con el diseño del *firmware*.

5. Desarrollo del *firmware*

Simultáneamente con el diseño del *hardware*, se trabaja en el desarrollo del *software*. Esto incluye programar el microcontrolador que controlará el inversor y la implementación del algoritmo de control. Se utilizará una placa de evaluación antes de tener la placa de control propia con tal de acelerar el desarrollo. La duración estimada para esta fase es de aproximadamente 4 meses.

6. Validación del *hardware*

Una vez que el *hardware* se está construyendo, se procede a la validación con los tests diseñados anteriormente. La duración estimada para esta etapa es de aproximadamente 3 a 4 meses y se superpone con el desarrollo del *software*.

7. Validación del *firmware*

La validación del *firmware* se realiza una vez que el *hardware* está validado, y mayoritariamente, mientras se desarrolla el *firmware*. Aquí, se llevan a cabo pruebas unitarias para garantizar que cada módulo de código funcione según lo previsto y ver que las simulaciones y cálculos previos se ajusten a la realidad. La duración estimada para esta etapa es de 2 meses.

8. Documentación y preparación para la implementación

La fase final del proyecto se enfoca en la documentación y la preparación para su implementación en los monoplazas de e-Tech Racing. La duración estimada para

esta etapa es de aproximadamente un par de semanas.

Es importante destacar que las etapas de diseño del *hardware* y del *firmware* pueden superponerse y solaparse con otras etapas, lo que permite un desarrollo más ágil y eficiente del proyecto. La superposición de estas etapas es esencial para cumplir con los plazos y garantizar que el proyecto avance de manera constante. El trabajo se ha realizado en el marco de un año académico, de septiembre a junio, aunque la definición de requisitos y los primeros pasos del modelo continuo se realizaron antes del comienzo del trabajo.

3.2. Gestión del Proyecto con Git

3.2.1. Introducción a Git

Git es un sistema de control de versiones distribuido, diseñado para rastrear cambios en archivos y coordinar el trabajo entre múltiples personas en proyectos de desarrollo de *software*. Utiliza un enfoque descentralizado, lo que significa que cada desarrollador tiene una copia completa del historial de cambios del proyecto. Git facilita el trabajo colaborativo, la gestión de cambios y la integración continua en proyectos de cualquier tamaño.

3.2.2. Funcionamiento de Git

Git trabaja mediante la creación de instantáneas (*commits*) de los archivos en un repositorio. Cada vez que se realiza un cambio significativo en los archivos del proyecto, se crea un nuevo *commit* que contiene una instantánea del estado de esos archivos en ese momento. Estos *commits* se organizan en ramas (*branches*), que permiten trabajar en paralelo en diferentes características o correcciones de errores sin afectar al código principal. Las ramas se fusionan (*merge*) cuando el trabajo está completo y se quiere incorporar al código principal. En este proyecto solo se ha usado una rama puesto que solo hay un desarrollador.

3.2.3. Por qué utilizar un sistema de control de versiones

El uso de un sistema de control de versiones como Git proporciona numerosos beneficios para el desarrollo de proyectos de ingeniería, incluyendo:

- **Historial de cambios:** Permite mantener un registro detallado de todas las modificaciones realizadas en el código y otros archivos del proyecto, lo que facilita la identificación de errores y el seguimiento del progreso del desarrollo.
- **Colaboración:** Facilita el trabajo en equipo al permitir que múltiples desarrolladores trabajen simultáneamente en diferentes aspectos del proyecto, sin temor a sobrescribir los cambios de los demás.

- **Seguridad:** Proporciona una copia de seguridad del código en caso de pérdida o corrupción de los archivos locales.
- **Experimentación:** Permite probar nuevas ideas y características en ramas separadas sin afectar al código principal.

3.2.4. Uso de Git en el proyecto

En el desarrollo de este proyecto, Git se utilizó para gestionar no solo el *firmware*, sino también las simulaciones, la documentación y el diseño del *hardware*. Cada aspecto del proyecto se organizó en una carpeta del mismo repositorio de Git, lo que permitió un seguimiento preciso de los cambios.

3.2.5. GitHub

GitHub es una plataforma de alojamiento de código que utiliza Git como sistema de control de versiones. Ofrece funciones adicionales como seguimiento de problemas, gestión de proyectos y revisión de código. En este proyecto, se utilizó GitHub para alojar el repositorio de Git.

3.2.6. Automatización de la documentación y *releases*

Una de las ventajas de utilizar Git y GitHub es la capacidad de automatizar tareas como la generación de documentación y la creación de *releases*. En este proyecto, se generó automáticamente una *wiki* utilizando los contenidos de esta memoria, lo cual facilitó la creación y el mantenimiento de documentación actualizada.

Además, se realizaron *releases* de *hardware* mediante el etiquetado de versiones en Git. Cada vez que se solicitaban PCBs, se creaba una nueva *release* que incluía los archivos de fabricación de las PCBs, una lista detallada de los cambios realizados desde la última versión, y se iban actualizando las *release notes* con los errores que se iban encontrando.

4. Desarrollo

4.1. Modelo matemático del PMSM

En esta sección, se abordará de manera gradual la introducción al control vectorial de máquinas eléctricas. Esta sección se asemeja más a un conjunto de pasos para comprender completamente el control utilizado en los inversores. Es importante tener en cuenta que existen muchas simplificaciones matemáticas, suposiciones no justificadas y pasos omitidos ya que la extensión de la sección sería extremadamente larga en caso de razonar desde cero todos los resultados.

4.1.1. Marco de referencia estático

Los motores síncronos de imanes permanentes (PMSM) están hechos de hierro, cobre y imanes. Se limita el análisis a máquinas de tres fases, aunque el análisis para sistemas con más de tres fases es similar. El cobre se distribuye en devanados, que tienen una resistencia y una inductancia iguales entre ellas si la máquina está equilibrada. Además, cuando el rotor gira, los imanes permanentes inducen una tensión en los devanados, lo que se conoce como fuerza contraelectromotriz (FCEM).



Figura 4.1: Motor síncrono de imanes permanentes desmontado, de manera que se pueden ver los empilados del rotor y el estátor, los imanes y el bobinado.

Se trabajará con motores cuyos devanados estén conectados en configuración estrella, ya que se utiliza con mucha más frecuencia que la configuración delta. De todas maneras, es sencillo extender el modelo y el control a motores con conexión delta.

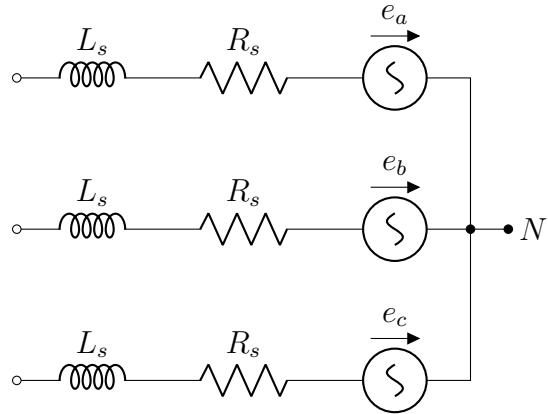


Figura 4.2: Circuito eléctrico equivalente de un PMSM trifásico en configuración estrella.

4.1.2. Representación en el espacio vectorial

Cuando el PMSM gira, las formas de onda de corriente y voltaje son aproximadamente sinusoidales. Siendo un sistema trifásico y equilibrado, se vería algo como en figura 4.3.

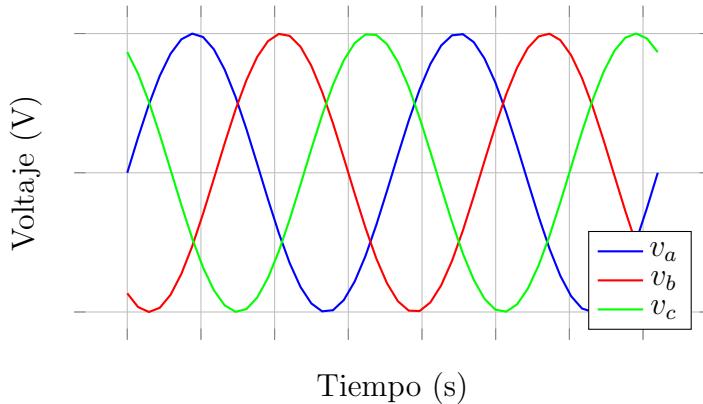


Figura 4.3: Sistema trifásico representado en el tiempo.

Todas las ecuaciones se pueden definir a partir de este modelo, pero se vuelve muy poco práctico para análisis más complejos. Se introduce una forma de representar estas formas de onda en el espacio tridimensional \mathbb{R}^3 , en el que se escogen los ejes (a, b, c). Lo que debería esperarse ver es una forma tridimensional en el espacio \mathbb{R}^3 . Se comienza en $t = 0$ y se dibuja un punto en las coordenadas $[v_a(0), v_b(0), v_c(0)]$. Luego, se continúa trazando puntos mientras se avanza a lo largo del eje del tiempo. La forma se puede expresar como curva paramétrica como $[v_a(t), v_b(t), v_c(t)]$. Cuando se representa la forma resultante, puede sorprender, ya que es un círculo perfectamente plano. Cabe destacar que el orden de los ejes depende del contexto, y en este trabajo se toma el sistema de referencia mostrado en la figura 4.4.

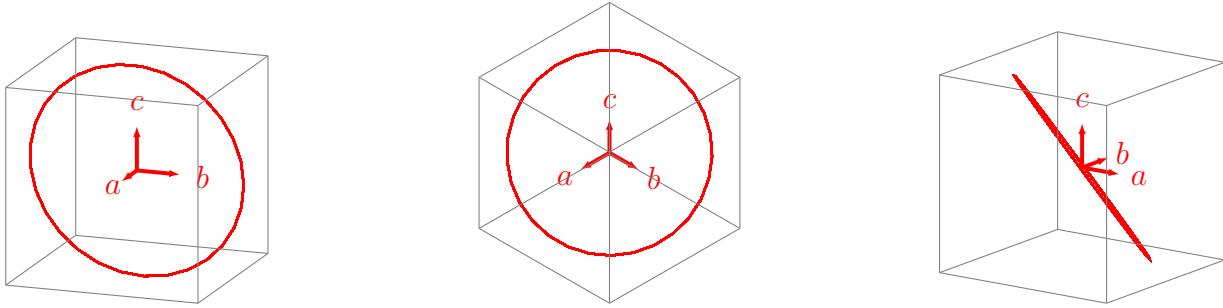


Figura 4.4: Sistema trifásico representado en el Espacio Vectorial.

Se puede observar que el sistema trifásico, cuando está equilibrado, se puede representar con solo dos variables, ya que la forma resultante es bidimensional. Se explorará cómo simplificar esto para facilitar el control del PMSM.

4.1.3. Transformadas

Se puede sospechar que lo que se necesita para representar el sistema trifásico con dos variables es un cambio de base, en particular, una rotación. El enfoque general sería utilizar la transformada de Concordia. La transformada de Clarke es el caso particular para 3 dimensiones de la transformada de Concordia, que se utiliza para cualquier número de dimensiones.

Se pueden encontrar dos transformadas de Clarke: la transformada de Clarke ortonormal y la transformada de Clarke de amplitud constante o módulo invariante. Además, se pueden considerar el eje α avanzado 90° respecto al eje β , o viceversa. Para el análisis y control de una máquina eléctrica se suele utilizar la transformada de Clarke de amplitud constante con α avanzada.

Transformada de Clarke ortonormal

Esta transformada se utiliza cuando la potencia del sistema debe permanecer inalterada después de la transformación. Se aplican dos rotaciones al marco de referencia abc y se crea el marco de referencia $\alpha\beta\gamma$. En este nuevo marco de referencia, la trayectoria del vector espacial está completamente contenida en el plano $\alpha\beta$ cuando el sistema está equilibrado, y el eje γ se utiliza para explicar el componente homopolar del sistema (la suma de los tres componentes debe ser igual a 0 si el sistema está equilibrado, de lo contrario, aparece el componente homopolar).

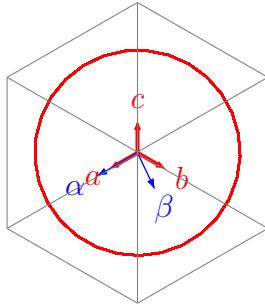


Figura 4.5: Sistema trifásico representado en el Espacio Vectorial con la transformada de Clarke ortonormal superpuesta.

Representar esta transformada en un eje temporal refleja de forma más intuitiva el resultado.

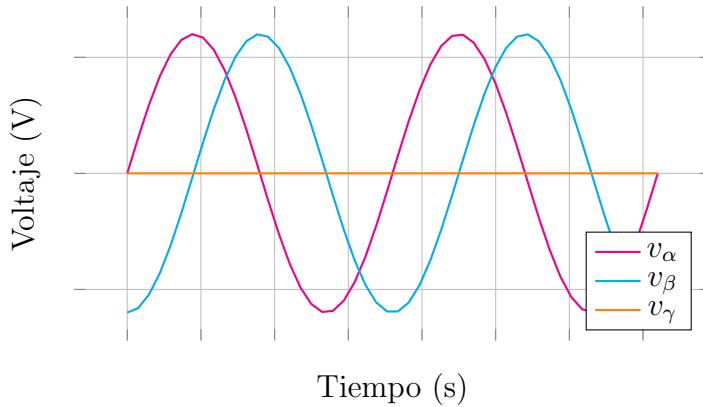


Figura 4.6: Transformada de Clarke ortonormal representada en el tiempo.

No se derivará la transformada aquí, aunque es necesario comprender lo que hace y conocer la matriz de transformación.

Lo que hace la transformada es colocar dos de los ejes del sistema de referencia en el plano formado por el círculo generado por el vector espacial. El eje restante es perpendicular a ese plano y representa el componente homopolar.

La forma matricial de la transformada es

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}. \quad (4.1)$$

La forma matricial de la transformada inversa es

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}. \quad (4.2)$$

Esta transformada tiene la particularidad de mantener constante la potencia del sistema, de modo que se cumple que

$$p(t) = p_{abc} = p_{\alpha\beta\gamma, \text{ortonormal}} = v_a \cdot i_a + v_b \cdot i_b + v_c \cdot i_c = v_\alpha \cdot i_\alpha + v_\beta \cdot i_\beta + v_\gamma \cdot i_\gamma. \quad (4.3)$$

Transformada de Clarke de Amplitud Constante

Mantener constante la potencia a lo largo de las transformadas puede ser útil en algunos contextos, pero lo que se suele implementar es una variante de la transformada de Clarke que mantiene constante la amplitud de la magnitud.

La transformada no es ortonormal, ya que ajusta las magnitudes para que el módulo de las variables sea el adecuado, pero la rotación se mantiene igual.

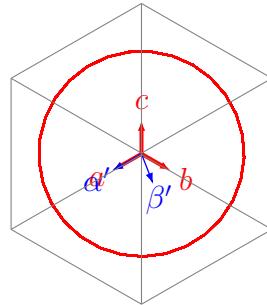


Figura 4.7: Sistema trifásico representado en el Espacio Vectorial con la transformada de Clarke de amplitud constante superpuesta.

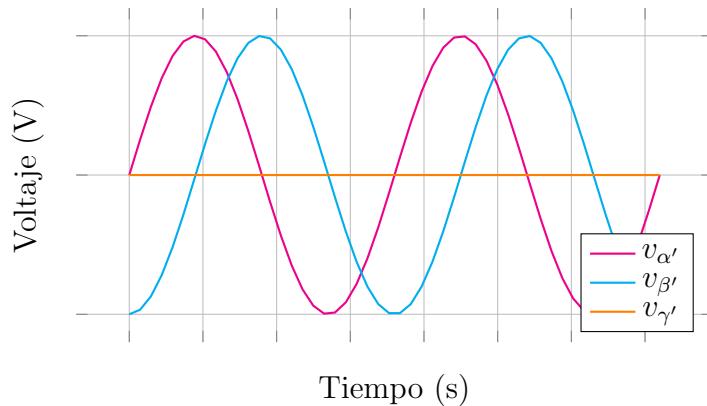


Figura 4.8: Transformada de Clarke de amplitud constante representada en el tiempo.

De esta manera, la forma matricial de la transformada de amplitud constante es

$$\begin{bmatrix} \alpha' \\ \beta' \\ \gamma' \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}. \quad (4.4)$$

Y la de la transformada inversa es

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix} \begin{bmatrix} \alpha' \\ \beta' \\ \gamma' \end{bmatrix}. \quad (4.5)$$

Se puede derivar que

$$p_{abc} = \frac{3}{2} \cdot (v_{\alpha'} \cdot i_{\alpha'} + v_{\beta'} \cdot i_{\beta'} + v_{\gamma'} \cdot i_{\gamma'}). \quad (4.6)$$

Transformada de Park

Después de aplicar la transformada de Clarke, todavía quedan dos variables sinusoidales ($\alpha\beta$ si se considera $\gamma = 0$), lo que dificulta el análisis para que el control sea sencillo. Por lo tanto, se aplica otra transformada para convertir estas cantidades sinusoidales en constantes.

Ahora, considerando que el vector espacial gira a una velocidad de $\omega = 2\pi f$, si se aplica continuamente una rotación alrededor del eje γ con un ángulo $\theta = \omega t$, se puede representar el vector espacial como una composición de dos variables continuas (en lugar de sinusoides). Además, si se sincroniza esta rotación con el ángulo del vector espacial (el eje d apuntando al vector espacial), se obtiene una variable continua (q) y una segunda variable de valor nulo (d).

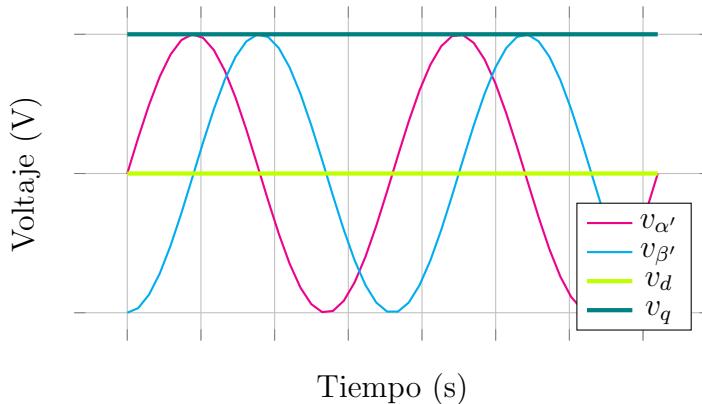


Figura 4.9: Transformada de Park representada en el tiempo junto a la transformada de Clarke de amplitud constante.

Se puede ver que la transformada es simplemente una rotación a lo largo de uno de los ejes de la base, de manera que

$$\begin{bmatrix} d \\ q \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}. \quad (4.7)$$

4.1.4. Marco de referencia rotatorio

En esta sección, se convertirá el modelo trifásico inicial del PMSM en un modelo continuo en el marco de referencia $d - q$. Antes, se omitieron las ecuaciones del modelo trifásico, pero utilizando estas y aplicando las transformadas de Clarke y Park, se obtienen las ecuaciones que se derivarán en esta sección.

El enfoque general es utilizar la transformada de amplitud constante, lo que lleva a estas ecuaciones para las tensiones y corrientes del estator

$$\begin{bmatrix} v_d \\ v_q \\ v_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} \quad (4.8)$$

$$\begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}. \quad (4.9)$$

El modelo de circuito equivalente se divide en los circuitos del eje d y el eje q .

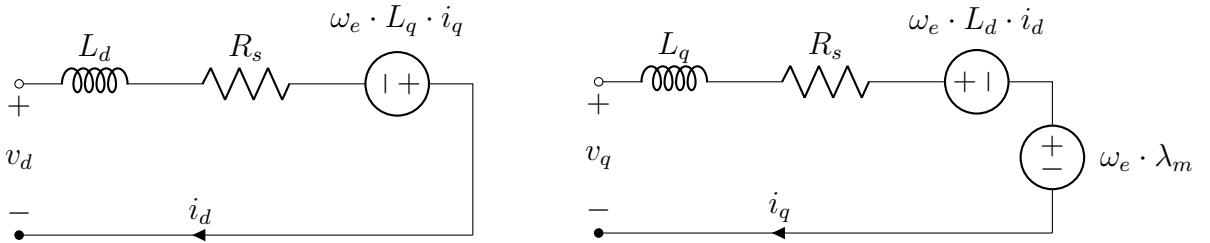


Figura 4.10: Circuitos eléctricos equivalentes del modelo $d - q$ de un PMSM.

Se puede observar que

$$v_d = v_{R_s} - \omega_e \cdot L_q \cdot i_q + v_{L_d} \quad (4.10)$$

$$v_d = R_s \cdot i_d - \omega_e \cdot L_q \cdot i_q + L_d \cdot \frac{di_d}{dt} \quad (4.11)$$

y

$$v_q = v_{R_s} - \omega_e \cdot L_d \cdot i_d + \omega_e \cdot \lambda_m + v_{L_q} \quad (4.12)$$

$$v_q = R_s \cdot i_q - \omega_e \cdot L_d \cdot i_q + \omega_e \cdot \lambda_m + L_q \cdot \frac{di_q}{dt} \quad (4.13)$$

En estado estacionario, es decir, sin cambios muy bruscos de la corriente, el término diferencial puede ser eliminado, de manera que

$$v_d = R_s \cdot i_d - \omega_e \cdot L_q \cdot i_q \quad (4.14)$$

$$v_q = R_s \cdot i_q - \omega_e \cdot L_d \cdot i_q + \omega_e \cdot \lambda_m, \quad (4.15)$$

donde

- v_d y v_q son las tensiones en los ejes d y q respectivamente.
- i_d y i_q son las corrientes en los ejes d y q respectivamente.
- L_d y L_q son las inductancias en los ejes d y q respectivamente.
- ω_e es la velocidad eléctrica, que es la velocidad mecánica multiplicada por el número de pares de polos del PMSM ($\omega_e = \omega_m \cdot pp = \omega_m \cdot \frac{n}{2}$).
- λ_m es el flujo magnético generado por los imanes permanentes. La magnitud del flujo magnético generado afecta directamente la magnitud de la tensión inducida en las fases del estator. Este parámetro se puede transformar fácilmente en k_E , que es la relación entre la velocidad mecánica del rotor y la tensión generada en las 3 fases.

Hay PMSM cuyos imanes están montados dentro del rotor (IPM) y otros cuyos imanes están en la superficie del rotor (SPM). Esta diferenciación juega un papel importante en el desarrollo del modelo y el control, porque en los SPM se cumple $L_d = L_q$, a menudo escrito solo como L . Además, si se trata de un IPM, la orientación de los imanes puede cambiar las trayectorias del control, de manera que si son imanes tangenciales se da $L_d > L_q$ y si son imanes radiales $L_d < L_q$. Se desarrollarán las ecuaciones para motores IPM con imanes radiales, pero se ha de tener en cuenta que se pueden realizar muchas simplificaciones si el motor es un SPM. Una situación que se dará es que algunas ecuaciones tienen alguna forma de $L_d - L_q$ como denominador, lo cual es bastante problemático si se está tratando de implementar la ecuación directamente. Por ese motivo, es mejor diferenciar el tipo de motor antes de implementar las ecuaciones. Una solución válida es tener en cuenta la anisotropía magnética que suelen presentar la mayoría de SPMs, con lo que $L_d < L_q$ y se pueden aplicar las ecuaciones del IPM a costa de potencia de cálculo, de ejecutarse el control en tiempo real.

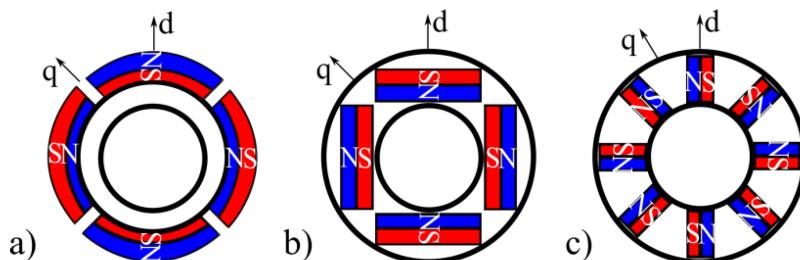


Figura 4.11: Tipos de PMSMs según la disposición de los imanes. a) SPM, b) IPM radial, c) IPM tangencial.

La potencia eléctrica se define como

$$p_e = \frac{3}{2} \cdot (v_d i_d + v_q i_q). \quad (4.16)$$

Si se supone que la máquina es perfectamente eficiente, se puede decir que la potencia mecánica es igual a la potencia eléctrica, $p_e = p_m$. Sabiendo que $p_m = \omega_m T_{em}$, donde T_{em} es el par electromagnético producido, se puede derivar que

$$T_{em} = \frac{3}{2 \cdot \omega_m} \cdot (v_d i_d + v_q i_q) = \frac{3}{2 \cdot \omega_e} \frac{n}{2} \cdot (v_d i_d + v_q i_q) \quad (4.17)$$

$$T_{em} = \frac{3}{2} \frac{n}{2} \cdot ((L_d - L_q) i_q i_d + \lambda_m i_q). \quad (4.18)$$

También se puede establecer un límite de voltaje, porque el PMSM generalmente se controla con un inversor de fuente de voltaje (VSI) modulado por SVPWM y la tensión de salida V_s está limitada a $\frac{V_{DC}}{\sqrt{3}}$. En esta ecuación, no se considera la caída de tensión por la resistencia del estator R_s , ya que haría que las ecuaciones fueran muy densas y el efecto de esta caída de voltaje es despreciable a altas velocidades. Por ello,

$$V_s = \sqrt{v_d^2 + v_q^2} \leq \frac{V_{DC}}{\sqrt{3}} \quad (4.19)$$

$$\sqrt{(R_s \cdot i_d - \omega_e \cdot L_q \cdot i_q)^2 + (R_s \cdot i_q - \omega_e \cdot L_d \cdot i_q + \omega_e \cdot \lambda_m)^2} \leq \frac{V_{DC}}{\sqrt{3}}. \quad (4.20)$$

Despreciando los términos con R_s ,

$$\sqrt{(-\omega_e \cdot L_q \cdot i_q)^2 + (-\omega_e \cdot L_d \cdot i_q + \omega_e \cdot \lambda_m)^2} \leq \frac{V_{DC}}{\sqrt{3}}. \quad (4.21)$$

Y reordenando,

$$\left(\frac{\frac{V_{DC}}{\sqrt{3}}}{\omega_e} \right)^2 \geq (\lambda_m + L_d \cdot i_d)^2 + (L_q \cdot i_q)^2. \quad (4.22)$$

4.1.5. Curvas características del PMSM

Las ecuaciones del PMSM son muy útiles cuando se diseña el control y la implementación real, pero antes de eso, es muy recomendable verlas en un gráfico para comprender mejor algunas de las intuiciones detrás de ellas.

Curva de par-velocidad y potencia-velocidad

Las primeras curvas estudiadas son las de par-velocidad y potencia-velocidad. Definen la intención de diseño del PMSM, ilustrando el rendimiento deseado. El objetivo al diseñar el control es tratar de igualar o incluso superar estas curvas.

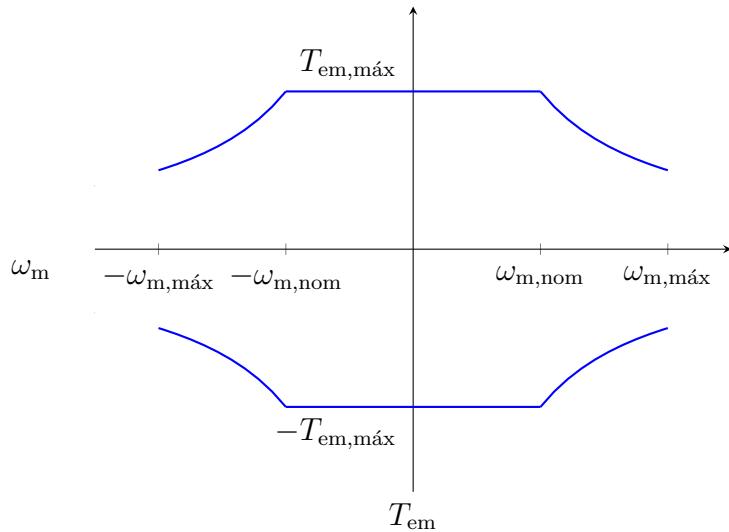


Figura 4.12: Curva de par-velocidad del PMSM.

La curva es una función por tramos, que toma $T_{\text{em},\text{máx}}$, $\omega_{m,\text{máx}}$ y $P_{m,\text{máx}}$ como sus parámetros.

La curva es constante desde $\omega_m = 0$ hasta $\omega_m = \omega_{m,\text{nom}} = \frac{P_{m,\text{máx}}}{T_{\text{em},\text{máx}}}$, donde su valor es $T_{\text{em},\text{máx}}$. Esta porción es lo que se conoce como la zona de par constante. Desde $\omega_m = \omega_{m,\text{nom}}$ hasta $\omega_m = \omega_{m,\text{máx}}$, T_{em} se define como $T_{\text{em}} = \frac{P_{m,\text{máx}}}{\omega_m}$, lo que da una curva de tipo $y = \frac{a}{x}$. Esto se llama la zona de potencia constante.

$$T_{\text{em}} = \begin{cases} T_{\text{em},\text{máx}} & -\omega_{m,\text{nom}} < \omega_m < \omega_{m,\text{nom}} \\ -T_{\text{em},\text{máx}} & -\omega_{m,\text{nom}} < \omega_m < \omega_{m,\text{nom}} \\ \frac{P_{m,\text{máx}}}{\omega_m} & \omega_{m,\text{nom}} \leq \omega_m \leq \omega_{m,\text{máx}} (T_{\text{em},\text{máx}}), -\omega_{m,\text{máx}} \leq \omega_m \leq -\omega_{m,\text{nom}} (-T_{\text{em},\text{máx}}) \\ -\frac{P_{m,\text{máx}}}{\omega_m} & \omega_{m,\text{nom}} \leq \omega_m \leq \omega_{m,\text{máx}} (-T_{\text{em},\text{máx}}), -\omega_{m,\text{máx}} \leq \omega_m \leq -\omega_{m,\text{nom}} (T_{\text{em},\text{máx}}) \end{cases}$$

Además P_m aumentará linealmente con ω_m hasta $\omega_{m,\text{nom}}$, ya que $P_m = T_{\text{em}} \cdot \omega_m$. Desde $\omega_{m,\text{nom}}$ hasta $\omega_{m,\text{máx}}$, P_m es una recta de valor $P_{m,\text{máx}}$.

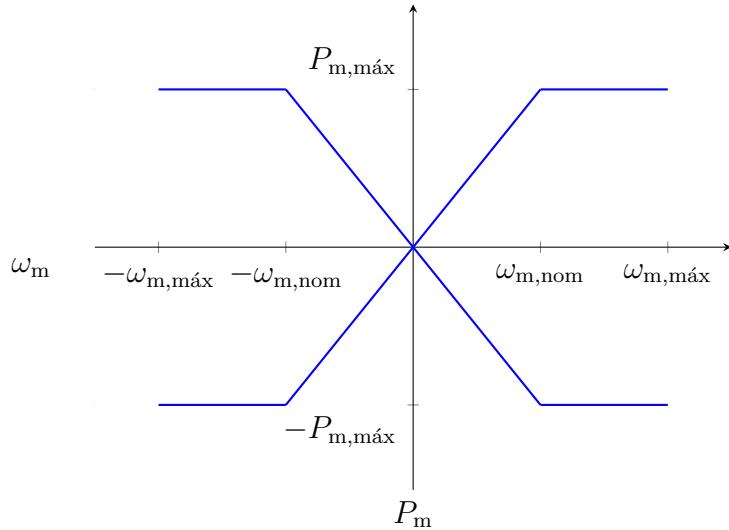


Figura 4.13: Curva de potencia-velocidad del PMSM.

CLC (Círculo de límite de corriente)

Es obvio que la corriente eléctrica suministrada al PMSM debe estar limitada. Por lo general, el fabricante del motor establecerá la corriente alterna máxima, lo cual se traduce en un límite para i_d y i_q .

A continuación, se presenta un gráfico muy útil para conocer los límites del motor. Se establecen los ejes como i_d e i_q . Por ejemplo, si un motor está funcionando con $i_d = -1$ A e $i_q = 5$ A, se dibuja un punto en $(-1, 5)$. Como se puede apreciar, este es un sistema de coordenadas cartesianas. También puede convertirse en un sistema de coordenadas polares, que utiliza una magnitud y un ángulo. La magnitud del vector será entonces:

$$I_s = \sqrt{i_d^2 + i_q^2} \quad (4.23)$$

Y el ángulo:

$$\gamma = \arctan \left(\frac{i_q}{i_d} \right) \quad (4.24)$$

Se puede observar que la corriente máxima puede expresarse más fácilmente como I_s , independientemente del ángulo γ (no debe confundirse con el γ de la transformada de Clarke). Por lo tanto, se puede representar $I_s = I_{s,\max}$, $\forall \gamma \in [0, 2\pi]$

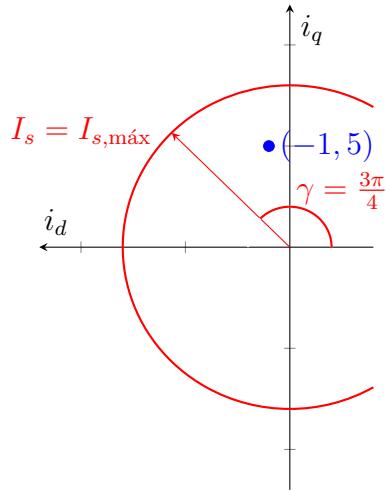


Figura 4.14: Círculo de límite de corriente con punto añadido.

Resulta ser un círculo, lo cual tiene sentido, ya que es un vector de magnitud constante. Además, el vector de corriente $I_s \angle \gamma = I_{s,\max} \angle \frac{3\pi}{4}$ se representa para facilitar su comprensión.

TH (Hipérbolas de par)

Si se estudia la ecuación del par 4.18, es evidente que T_{em} es una función de (i_d, i_q) . El resto de parámetros son constantes, por lo que se puede establecer un valor fijo de par y deslizar alrededor de (i_d, i_q) para generar una curva. La forma de la curva resultante es una hipérbola, y a continuación se presenta en su forma polar:

$$T_{em} = \frac{3}{2}pp \cdot ((L_d - L_q) \cdot I_s^2 \cdot \sin(\gamma) \cos(\gamma) + \lambda_m \cdot I_s \cdot \sin(\gamma)) \quad (4.25)$$

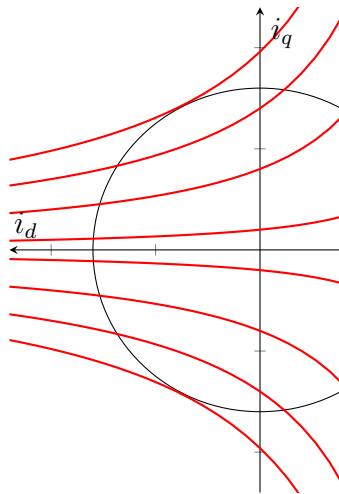


Figura 4.15: Hipérbolas de par.

El gráfico se limita a los cuadrantes 2 y 3 por un motivo ilustrado con estas hipérbolas: solo los valores negativos de i_d contribuyen a la generación de par. Cuando $i_d > 0$, se necesita más corriente para generar la misma cantidad de par. Cuanto más alejada está la hipérbola del eje i_d , más par representa en valor absoluto. Aquellas hipérbolas que quedan por encima del eje i_d , es decir, $i_q > 0$ son par positivo, mientras que si $i_q < 0$, el par es de sentido opuesto.

VLE (Elipses de límite de voltaje)

Tomando la ecuación de voltaje 4.22, se puede demostrar que es una elipse. Del mismo modo que con las hipérbolas de par, se pueden establecer una velocidad y una tensión, y deslizar valores de (i_d, i_q) para generar la curva.

$$1 \geq \frac{\left(\frac{\lambda_m}{L_d} + i_d\right)^2}{\left(\frac{V_{DC}}{\sqrt{3} L_d \omega_e}\right)^2} + \frac{i_q^2}{\left(\frac{V_{DC}}{\sqrt{3} L_q \omega_e}\right)^2} \quad (4.26)$$

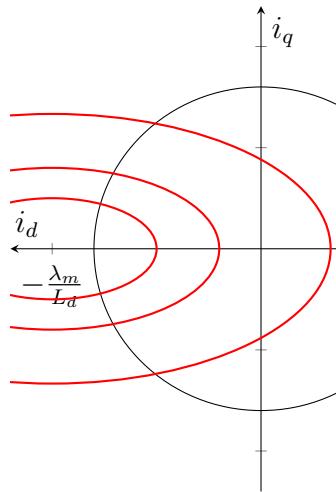


Figura 4.16: Elipses de límite de voltaje con $I_{sc} > I_{s,\text{máx}}$.

Al representar estas elipses, normalmente se anotan los valores de velocidad en RPM mecánicas, ya que es mucho más fácil hacerse una idea de los límites del motor junto al resto de curvas ($\omega_m[\text{RPM}] = \frac{1}{pp} \omega_e [\frac{\text{rad}}{\text{s}}] \cdot \frac{60}{2\pi}$).

Las elipses se reducen a medida que la velocidad aumenta. El foco de las elipses está ubicado exactamente en $(i_d, i_q) = (-I_{sc}, 0) = \left(-\frac{\lambda_m}{L_d}, 0\right)$. En el gráfico anterior, el foco está fuera del círculo de límite de corriente, pero no siempre es el caso. Si $I_{sc} \leq I_{s,\text{máx}}$, teóricamente el motor puede alcanzar una velocidad infinita, ya que las elipses colapsan en un solo punto.

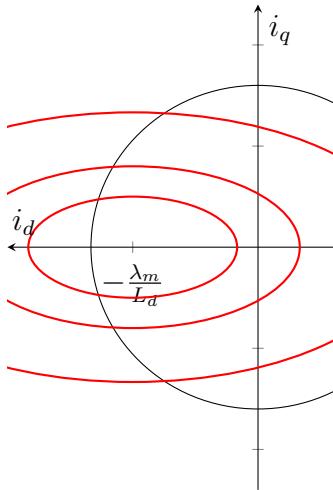


Figura 4.17: Elipses de límite de voltaje con $I_{sc} \leq I_{s,\text{máx}}$.

4.2. Control del PMSM en el espacio d-q

4.2.1. Trayectorias de control

Después de conocer los límites de la máquina, se pueden establecer criterios para decidir cuánto i_d y i_q (o I_s y γ) se deben aplicar al PMSM para que se comporte mecánicamente como se desee. El conjunto de puntos de trabajo que definen un comportamiento se llama trayectoria y existen una multitud de ellas. Por ejemplo, se puede desear que el motor produzca la mayor cantidad de par posible con la mínima corriente. Pero también se podría querer que tenga un cierto factor de potencia o que mantenga el par constante subiendo la velocidad, etc.

En un monoplaza de Formula Student, se desea que la salida de par esté perfectamente controlada y conocida para que el algoritmo de dinámica vehicular pueda estimar correctamente las fuerzas en los neumáticos. También es deseable que el motor pueda girar más rápido cuando no se requiere más par, ya que no es necesaria mucha tracción a altas velocidades del vehículo. Además, es necesario que sea eficiente para aprovechar mejor la energía de la batería. Con estos requisitos en mente, se estudian 4 trayectorias de control adecuadas para esta aplicación.

MTPA (Máximo Par por Amperio)

La trayectoria de control más utilizada es el MTPA, o Máximo Par por Amperio. Como su nombre indica, minimiza la corriente para entregar un par determinado. La condición que se debe cumplir es

$$\frac{\partial T_{em}}{\partial \gamma} = 0. \quad (4.27)$$

La expresión analítica se desarrolla partiendo de la ecuación de par en forma polar 4.25,

$$T_{em} = \frac{3}{2}pp \cdot ((L_d - L_q) \cdot I_s^2 \cdot \sin(\gamma) \cos(\gamma) + \lambda_m \cdot I_s \cdot \sin(\gamma))$$

$$\frac{\partial T_{em}}{\partial \gamma} = \frac{\partial}{\partial \gamma} \frac{3}{2}pp \cdot (I_s^2 \cos(\gamma) \sin(\gamma) \cdot ((L_d - L_q) + \lambda_m I_s \sin(\gamma))) = 0 \quad (4.28)$$

$$I_{s,MTPA} = -\frac{\lambda_m \cos(\gamma)}{(2 \cdot \cos(\gamma)^2 - 1) \cdot (L_d - L_q)}. \quad (4.29)$$

Para la aplicación de esta trayectoria en el control se busca el ángulo como función de la corriente, y para ello se debe despejar γ_{MTPA} de la expresión.

$$\gamma_{MTPA} = \frac{\pi}{2} + \arcsin\left(\frac{\lambda_m \pm \sqrt{8(L_d - L_q)^2 \cdot I_s^2 + \lambda_m^2}}{4 \cdot I_s(L_d - L_q)}\right) \quad (4.30)$$

El resultado de graficar esta expresión sobre el plano (i_d, i_q) deja a la vista que el módulo de corriente es mínimo para cada hipérbola de par.

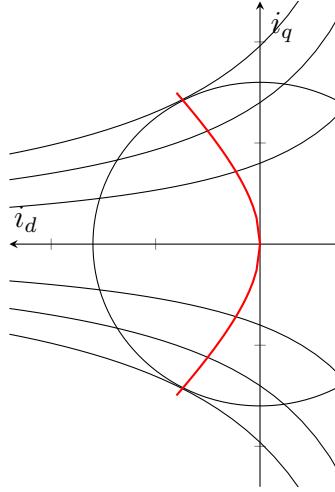


Figura 4.18: Trayectoria MTPA.

CTC (Curva de Par Constante)

Como se puede observar, las hipérbolas de par definen una trayectoria la cual permite mantener un par constante. Recordando las elipses de tensión, para un mismo valor de V_s las elipses se contraen hacia el foco a medida que la velocidad aumenta. Esto significa que siguiendo la curva de par constante de derecha a izquierda se puede mantener el par aumentando la velocidad. Usando la expresión 4.25 se puede obtener directamente

$$T_{em} = \frac{3}{2}pp \cdot ((L_d - L_q) \cdot I_s^2 \cdot \sin(\gamma) \cos(\gamma) + \lambda_m \cdot I_s \cdot \sin(\gamma))$$

$$I_{s,CTC} = \frac{\lambda_m}{L_d} \cdot \frac{\sqrt{\sin(\gamma)^2 + \frac{2 \cdot \frac{L_d - L_q}{L_d} \cdot \sin(2\gamma) \cdot T_{em} \cdot 2L_d}{3 \cdot pp \cdot \lambda_m^2}} - \sin(\gamma)}{\sin(2\gamma) \cdot \left(\frac{L_d - L_q}{L_d}\right)}. \quad (4.31)$$

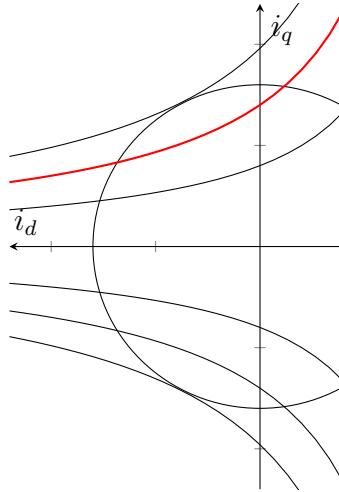


Figura 4.19: Trayectoria CTC.

MTPV (Máximo Par por Voltio)

Existe una trayectoria que permite maximizar el par entregado por el motor en rangos de velocidad muy altos donde el límite es la tensión que puede sintetizar la controladora. La condición que se debe cumplir es que

$$\frac{\partial T_{em}}{\partial \delta} = 0. \quad (4.32)$$

Donde δ es el ángulo del vector de tensión V_s , de la misma manera que γ es el ángulo del vector de corriente I_s . La expresión analítica se desarrolla a partir de la expresión de par en coordenadas cartesianas (4.18), de manera que

$$T_{em} = \frac{3}{2} pp \cdot ((L_d - L_q)i_q i_d + \lambda_m i_q).$$

Se aíslan i_d e i_q de 4.14 y 4.15, neglijiendo la caída de tensión resistiva del estator,

$$i_d = \frac{v_q}{\omega_e \cdot L_d} - \frac{\lambda_m}{L_d}; i_q = -\frac{v_d}{\omega_e \cdot L_q} \quad (4.33)$$

$$T_{em} = \frac{3}{2} pp \cdot \left((L_d - L_q) \left(-\frac{v_d}{\omega_e \cdot L_q} \right) \left(\frac{v_q}{\omega_e \cdot L_d} - \frac{\lambda_m}{L_d} \right) + \lambda_m \left(-\frac{v_d}{\omega_e \cdot L_q} \right) \right) \quad (4.34)$$

$$T_{em} = \frac{3}{2}pp \cdot \left((L_d - L_q) \left(-\frac{V_s \cdot \cos(\delta)}{\omega_e \cdot L_q} \right) \left(\frac{V_s \cdot \sin(\delta)}{\omega_e \cdot L_d} - \frac{\lambda_m}{L_d} \right) + \lambda_m \left(-\frac{V_s \cdot \cos(\delta)}{\omega_e \cdot L_q} \right) \right) \quad (4.35)$$

$$\begin{aligned} \frac{\partial T_{em}}{\partial \delta} &= \frac{3}{2}pp \cdot \left((L_d - L_q) \cdot \left(\frac{V_s \cdot \sin(\delta)}{\omega_e \cdot L_q} \right) \cdot \left(\frac{V_s \cdot \sin(\delta)}{\omega_e \cdot L_d} - \frac{\lambda_m}{L_d} \right) \right. \\ &\quad \left. - \left(\frac{V_s \cdot \cos(\delta)}{\omega_e} \right)^2 \cdot \frac{L_d - L_q}{L_d \cdot L_q} \right. \\ &\quad \left. - \frac{\lambda_m \cdot V_s \cdot \sin(\delta)}{L_q \cdot \omega_e} \right) = 0 \end{aligned} \quad (4.36)$$

Definiendo la saliencia

$$\xi = \frac{L_q}{L_d} \quad (4.37)$$

se obtiene

$$\begin{aligned} I_{s,MTPV} &= \frac{\lambda_m}{L_d} \left(\frac{-(2-\xi) \cos(\gamma)}{2(1-\xi)(1+(\xi)^2) \cos(\gamma)^2 - 2(1-\xi)(\xi)^2} \right. \\ &\quad \left. - \frac{\sqrt{(2-\xi)^2 \cos(\gamma)^2 - 4(1-\xi)(1+(\xi)^2) \cos(\gamma)^2 - 4(1-\xi)(\xi)^2}}{2(1-\xi)(1+(\xi)^2) \cos(\gamma)^2 - 2(1-\xi)(\xi)^2} \right) \end{aligned} \quad (4.38)$$

Cabe destacar que esta trayectoria solamente se puede ejecutar si se cumple la condición de que $I_{sc} \leq I_{s,\max}$.

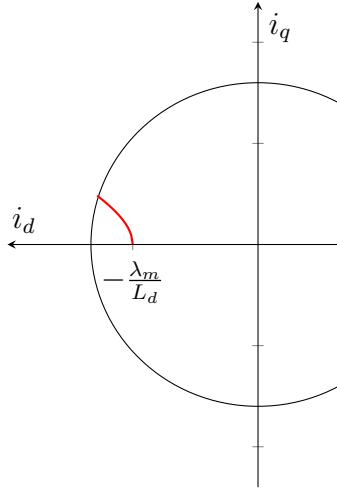


Figura 4.20: Trayectoria MTPV.

CVL (Límites de Corriente y Voltaje)

Por último, se presentan los límites eléctricos del motor y del convertidor. El límite de corriente consiste simplemente en saturar la magnitud de la corriente de manera que

no sobrepase el valor máximo establecido. La trayectoria sería sencillamente seguir el círculo de corriente anteriormente presentado (CLC), con la siguiente expresión:

$$I_{s,\text{CLC}} = I_{s,\text{máx}}, \forall \gamma \in [0, 2\pi]$$

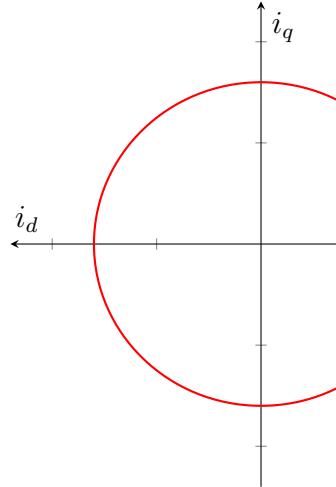


Figura 4.21: Trayectoria CLC.

El límite de tensión del motor en realidad se puede entender como la velocidad máxima a la que se puede llegar con una determinada tensión. Por ello, se usa la forma polar de la expresión de las elipses de tensión (VLE),

$$1 \geq \frac{\left(\frac{\lambda_m}{L_d} + I_s \cdot \cos(\gamma)\right)^2}{\left(\frac{V_{DC}}{L_d \cdot \omega_e}\right)^2} + \frac{(I_s \cdot \sin(\gamma))^2}{\left(\frac{V_{DC}}{L_q \cdot \omega_e}\right)^2}. \quad (4.39)$$

Igual que para el resto de trayectorias, se debe obtener una expresión de la elipse en función de I_s y γ . Ya que no es trivial despejar estas variables de la expresión anterior, se manipula usando la ecuación polar de la elipse desplazada del origen, de manera que

$$\rho(\theta) = \frac{b^2 x \cos(\theta) + a^2 y \sin(\theta) \pm ab \sqrt{(a^2 - x^2) \sin^2(\theta) + (b^2 - y^2) \cos^2(\theta) + 2xy \sin(\theta) \cos(\theta)}}{a^2 \sin^2(\theta) + b^2 \cos^2(\theta)}. \quad (4.40)$$

Dado que estas elipses tan solo están desplazadas en el eje x , se pueden eliminar todos los términos referentes al desplazamiento en y .

$$\rho(\theta) = \frac{b^2 x \cos(\theta) \pm ab \sqrt{(a^2 - x^2) \sin^2(\theta) + (b^2) \cos^2(\theta)}}{a^2 \sin^2(\theta) + b^2 \cos^2(\theta)} \quad (4.41)$$

Sustituyendo por los términos conocidos y simplificando,

$$= \frac{\left(\frac{1}{L_q}\right)^2 (-I_{sc}) \cos(\gamma) \pm \frac{1}{L_d \cdot L_q} \sqrt{\left(\left(\frac{V_s}{L_d \cdot \omega_e}\right)^2 - (-I_{sc})^2\right) \sin^2(\gamma) + \left(\frac{V_s}{L_q \cdot \omega_e}\right)^2 \cos^2(\gamma)}}{\left(\frac{1}{L_d}\right)^2 \sin^2(\gamma) + \left(\frac{1}{L_q}\right)^2 \cos^2(\gamma)}. \quad (4.42)$$

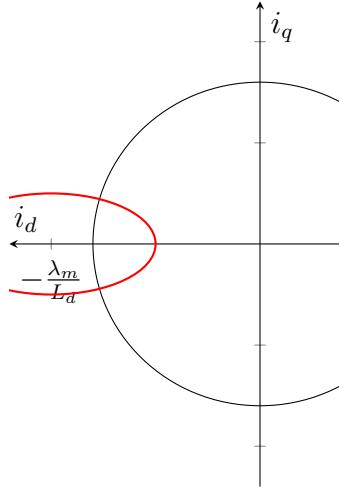


Figura 4.22: Trayectoria VLE.

Además, el inversor es capaz de sintetizar un máximo de $V_s = \frac{V_{DC}}{\sqrt{3}}$ utilizando SVPWM, por lo tanto, se debe saturar la consigna de tensión a ese valor. Adicionalmente, por seguridad, se multiplica por un factor de seguridad $K_{FW} \in (0, 1)$.

$$V_{s,\max} \leq \frac{V_{DC}}{\sqrt{3}} \cdot K_{FW} \quad (4.43)$$

4.2.2. Diseño y simulación del control

En esta sección, se aborda la implementación del modelo matemático del PMSM en un entorno de simulación. Además, se detallan los pasos cruciales en el diseño del control, destacando la implementación del lazo de control de corriente, el modelo promediado y commutado del inversor, o la integración de las trayectorias y la estrategia de debilitamiento de campo. Disponer de un modelo de simulación completo permite ganar mucha comprensión sobre el sistema estudiado, pero por el coste computacional suele ser inviable juntar muchos sistemas en una misma simulación.

EMR (Representación macroscópica energética)

En primer lugar, se creará un modelo que permita simular la dinámica electromecánica del PMSM, así como los entornos mecánico (vehículo) y eléctrico (batería) en los que se encuentra, para posteriormente integrar el control vectorial. Para ello se

usará un estándar para modelizar sistemas de potencia, la representación macroscópica energética o EMR por sus siglas en inglés. El concepto se basa en agrupar o dividir las diferentes etapas en las que la potencia se transforma, utilizando el principio de acción-reacción y el principio causal (el efecto causa-efecto causa efecto porque la causa del efecto causa-efecto es a su vez causa y efecto).

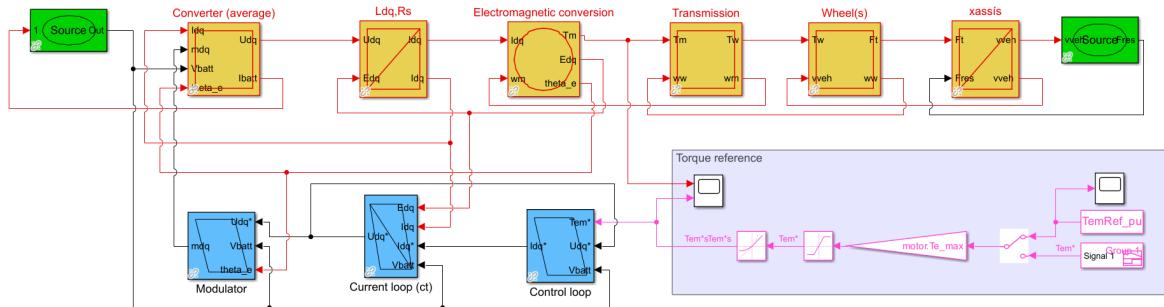


Figura 4.23: Modelo EMR completo.

En primer lugar se modeliza la planta eléctrica del PMSM. Se utiliza el modelo con el marco de referencia rotativo $d - q$ por su sencillez. Para ello se implementan las diferentes ecuaciones del motor en bloques separados siguiendo el estándar de la EMR.

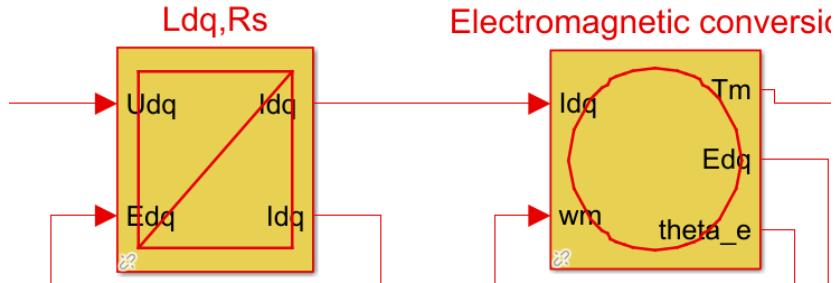


Figura 4.24: Bloques que representan el PMSM.

También se modela la planta mecánica del motor, así como la transmisión de la potencia mecánica a las ruedas y al vehículo.

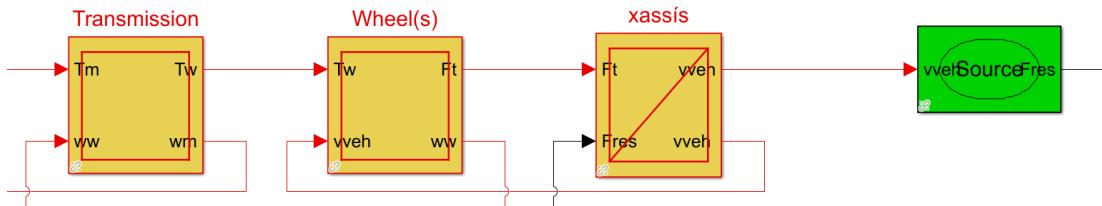


Figura 4.25: Planta mecánica.

Lazos de corriente y modelo promediado del inversor

Como se ha visto hasta ahora, es práctico utilizar la corriente para controlar el motor. Por ello, se implementa un lazo de corriente utilizando controladores PI para el eje d y para el eje q por separado. Como el inversor se utiliza como fuente de tensión, la salida de estos PI es la consigna de tensión. Dado que el motor genera una fuerza contraelectromotriz, se añade como *feed-forward* a los controladores. La salida del controlador no se satura directamente, sino que se implementa una saturación posterior la cual se realimenta al controlador para usar una técnica de *anti-windup*. Las constantes de los controladores se ajustan de la siguiente manera:

$$M_p = 15\%$$

$$t_s = T_s \cdot 20$$

Donde M_p es el sobre-impulso deseado en la respuesta a una entrada de escalón, t_s es el tiempo de establecimiento deseado, y T_s es la inversa de la frecuencia de control.

$$\xi = \sqrt{\frac{\log(M_p)^2}{\pi^2 + \log(M_p)^2}} \quad (4.44)$$

$$\omega_n = \frac{3}{\xi \cdot t_s} \quad (4.45)$$

$$Kp_{id} = 2 \cdot \xi \cdot \omega_n \cdot L_d - R_s \quad (4.46)$$

$$Ki_{id} = \omega_n^2 \cdot L_d \quad (4.47)$$

$$Kp_{iq} = 2 \cdot \xi \cdot \omega_n \cdot L_q - R_s \quad (4.48)$$

$$Ki_{iq} = \omega_n^2 \cdot L_q \quad (4.49)$$

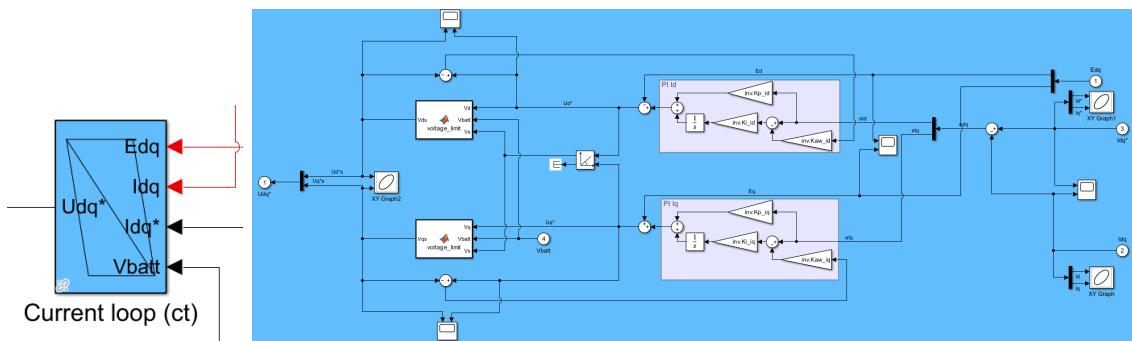


Figura 4.26: Bloque de los lazos de corriente.

Para las simulaciones se han implementado los siguientes parámetros de motor, ya que aunque a fecha de redacción de este documento no está fabricado, se han estimado de manera que se cumplan los requisitos del motor.

Parámetros del Motor			
Parámetro	Valor	Unidades	Descripción
pp	3	ad	Número de pares de polos
λ_m	52.615	mWb	Flujo magnético de los imanes permanentes
L_d	188.7	μH	Inductancia en el eje d
L_q	283.1	μH	Inductancia en el eje q
R_s	150	$\text{m}\Omega$	Resistencia de fase del estator
$\omega_{\text{m,máx}}$	20000	RPM	Velocidad angular máxima del motor
$T_{\text{em,máx}}$	26	N·m	Par máximo del motor
V_{bat}	540	V	Voltaje de la batería DC
$I_{s,\text{máx}}$	108	A	Corriente máxima en los ejes d-q

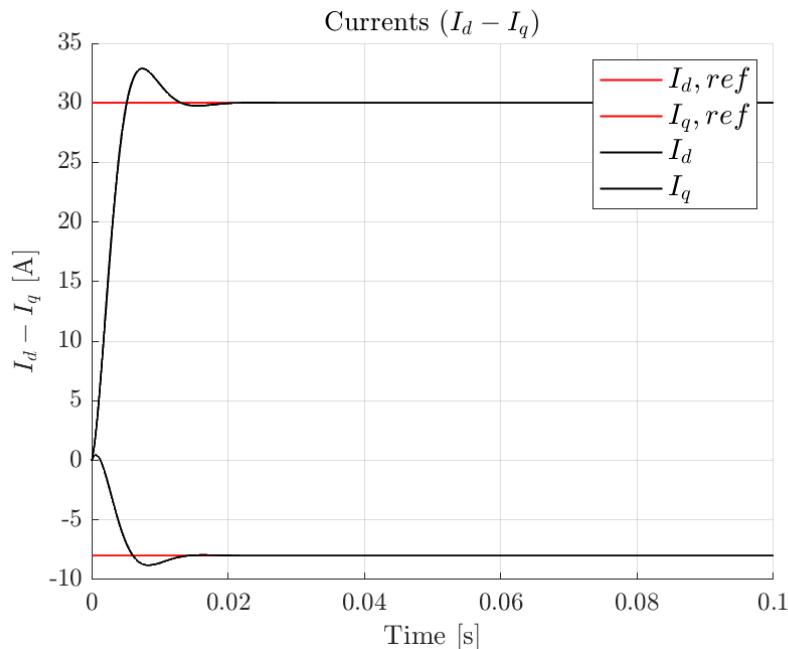


Figura 4.27: Simulación de los lazos de corriente, con una consigna de $(i_d, i_q) = (-8, 30)$ A.

Tras obtener las consignas de tensión, se modela el inversor VSI con SVPWM con un modelo promediado, es decir, sin llegar a generar una señal conmutada por PWM. Se usan relaciones básicas para convertir las magnitudes eléctricas del espacio $d - q$ a DC. Además se incorpora la fuente de energía del sistema, la batería, con un simple modelo RC.

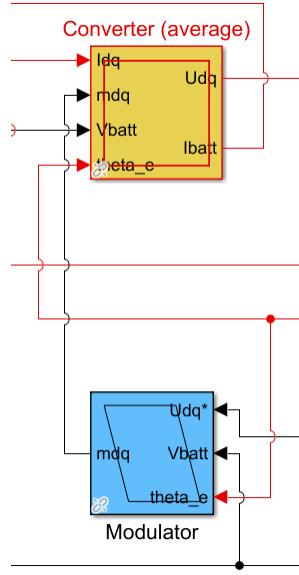


Figura 4.28: Bloques que contienen el modelo promediado del VSI con SVPWM.

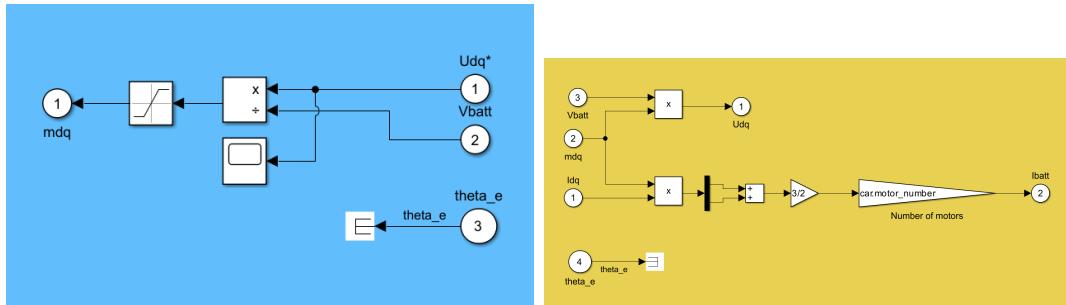


Figura 4.29: Detalle de los bloques del VSI.

Implementación de las trayectorias de control

Con lo anteriormente desarrollado solamente se pueden consignar las corrientes i_d e i_q manualmente, pero el objetivo es consignar el par y que el propio control sea capaz de gestionar el debilitamiento de campo. Por ello, se implementan las ecuaciones presentadas en el apartado anterior en bloques de código.

La estrategia es la siguiente: Se implementan las ecuaciones de las trayectorias cuya salida es una corriente (CLC, CTC, MTPV y VLE) y se selecciona la mínima, o en caso de estar en zona de debilitamiento de campo, la óptima, que no necesariamente es la mínima. En paralelo, se calcula el ángulo que correspondería a la trayectoria del MTPA, y se añade un control integral que aumenta el valor del ángulo controlando la tensión para poder entrar en el resto de trayectorias. Este integrador sería justamente el controlador de debilitamiento de campo y se encarga de que la consigna de ángulo no haga sobrepasar el límite de tensión establecido por el bus DC con un cierto factor de seguridad. Se trabaja con módulos de corriente siempre positivos, y ángulos comprendidos entre $\gamma \in [\frac{\pi}{2}, \pi]$. Para obtener par negativo, simplemente se multiplica el ángulo

γ por el signo de la consigna de par, atendiendo específicamente al caso de par igual a cero.

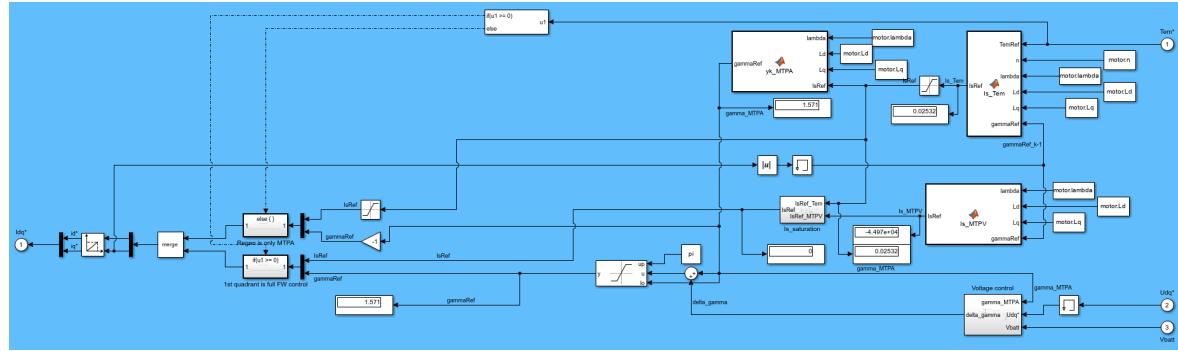


Figura 4.30: Detalle del bloque del lazo de control vectorial.

Para comprobar el funcionamiento y la estabilidad del control, se realiza una simulación en la que la consigna de par está extraída de un perfil de conducción real.

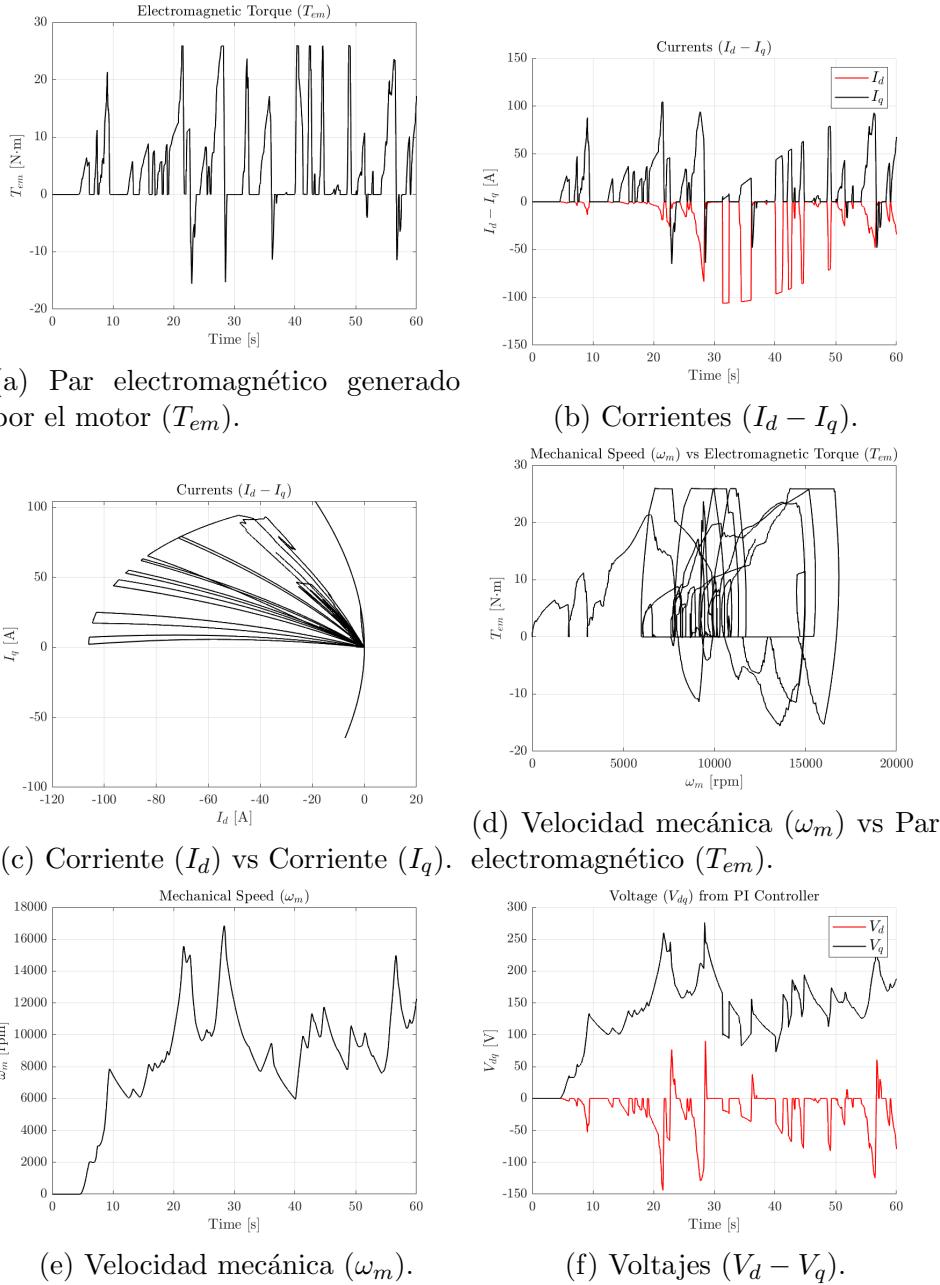


Figura 4.31: Resultados de la simulación.

Se puede observar que en esta simulación se ha limitado el comportamiento de la frenada regenerativa a la trayectoria del MTPA. Además, se pueden observar ciertos problemas en la implementación del lazo de tensión. Más adelante se realiza una simulación más enfocada en el control que en la aplicación, y en ella se revisan estas situaciones.

Modelo conmutado

Dado que el inversor realmente es una fuente conmutada, se debe modelar utilizando una herramienta que lo permita. Lo único que es necesario discretizar realmente es el control y la generación de tensiones, ya que la planta es continua. Por ello, el primer paso es recalcular las constantes de los lazos de control. Hasta ahora, se habían usado PI continuos, descritos con la expresión

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) d\tau. \quad (4.50)$$

Cuando se discretiza esta expresión usando la transformada Z, se obtiene

$$u[k] = K_p \cdot e[k] + K_i \cdot I[k], \quad (4.51)$$

donde la integral $I[k]$ se calcula con una aproximación trapezoidal como

$$I[k] = I[k - 1] + \frac{(e[k] + e[k - 1]) \cdot \Delta T}{2}, \quad (4.52)$$

donde ΔT es el tiempo de ejecución del PI. Para simplificar el cálculo, se introducen las constantes K_0 y K_1 , que están relacionadas con K_p y K_i .

$$K_0 = K_p + K_i \cdot \frac{\Delta T}{2} \quad (4.53)$$

$$K_1 = K_i \cdot \frac{\Delta T}{2} \quad (4.54)$$

Así, la ecuación discreta del controlador PI trapezoidal se expresa como

$$u[k] = u[k - 1] + K_0 \cdot e[k] + K_1 \cdot e[k - 1]. \quad (4.55)$$

Utilizando el modelo EMR generado en Simulink se ha implementado la conmutación, pero el tiempo de simulación es demasiado grande como para que sea una herramienta práctica para el desarrollo.

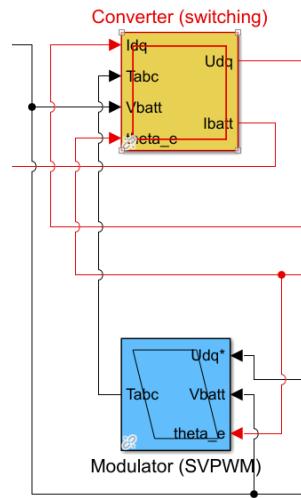


Figura 4.32: Bloques que contienen el modelo conmutado del VSI con SVPWM.

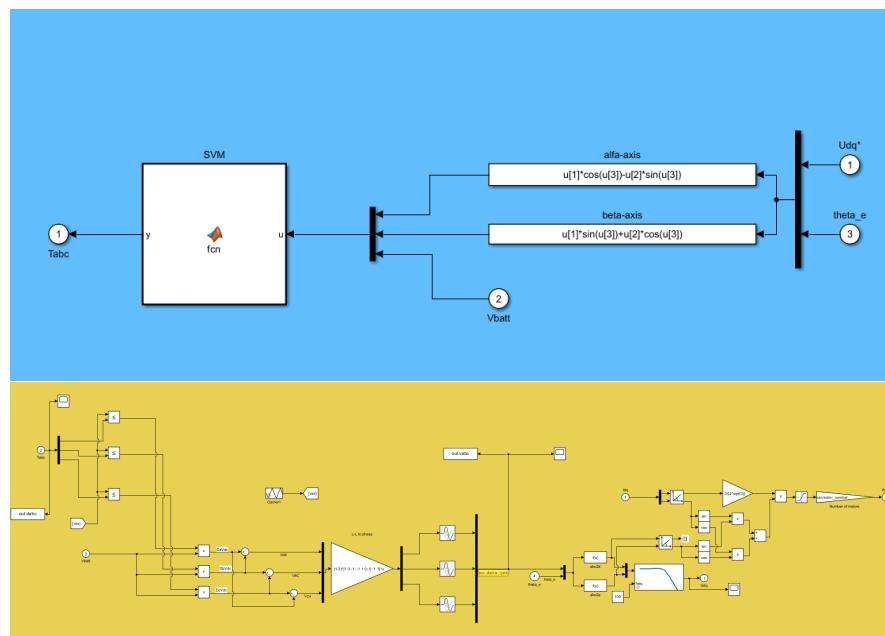
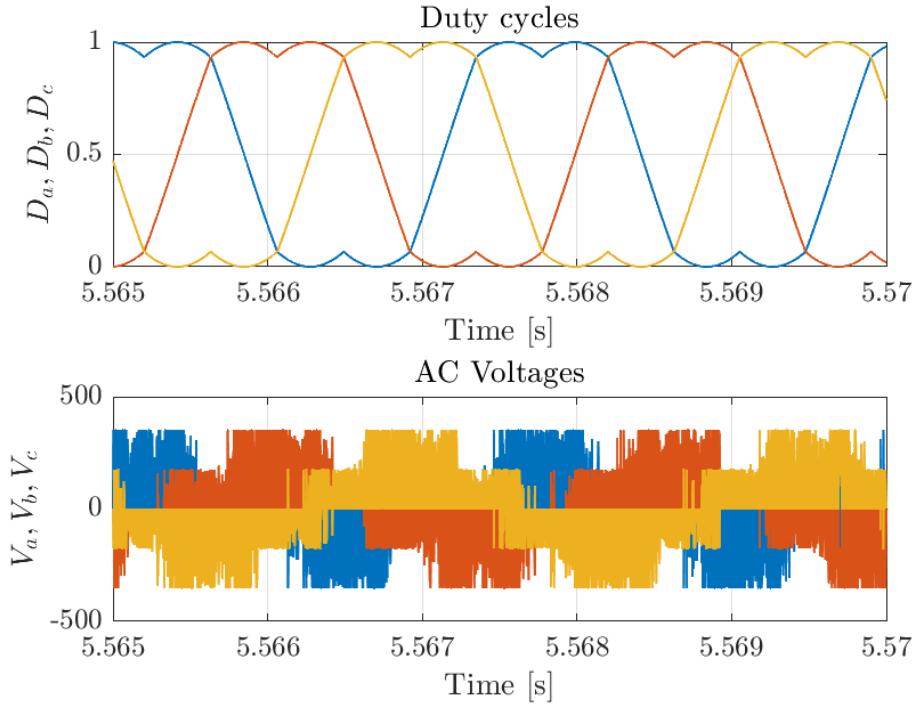


Figura 4.33: Detalle de los bloques del VSI conmutado.

Figura 4.34: *Duty cycles* y tensiones alternas.

Por ello se desarrolla un modelo en PLECS que incorpora el lazo de control, el inversor con MOSFETs, la planta mecánica simplificada, y a la cual se le discretiza la adquisición y el control, de manera que es una aproximación muy realista de la posterior implementación en un microcontrolador. Además se incorporan cálculos térmicos de pérdidas y eficiencia.

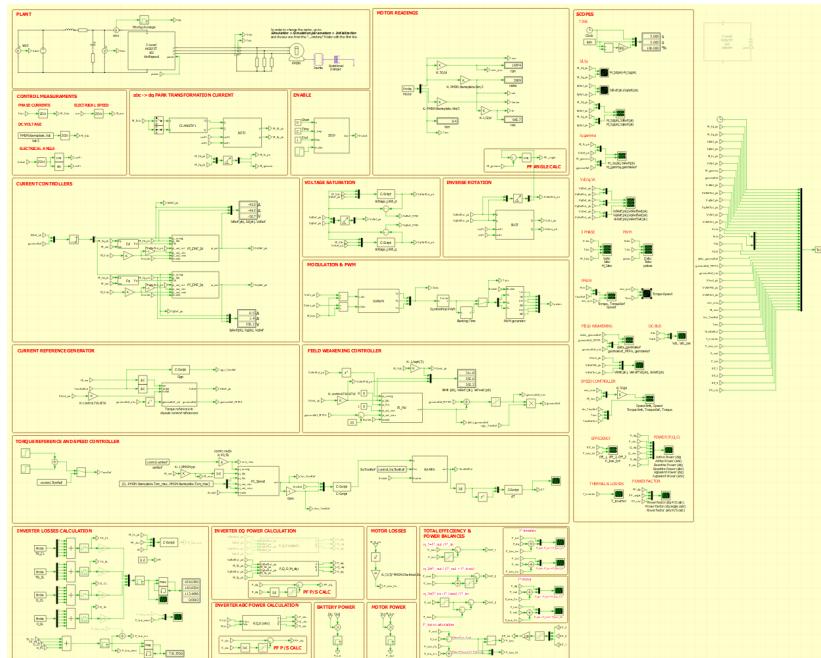


Figura 4.35: Modelo de simulación en PLECS.

Se puede observar que la representación de los bloques en este modelo no sigue el EMR, ya que se elige una implementación más práctica y realista. Destaca la implementación de las funciones de control utilizando la librería PERGAMON, desarrollada por el CITCEA-UPC. Esta librería reproduce completamente el código de las funciones matemáticas implementadas en el control, como los PI, el SVPWM..., en el *software* PLECS, lo que facilita la correlación de cualquier modelo con la implementación final en un sistema discreto.

A continuación se presentan los diferentes bloques que forman el modelo.

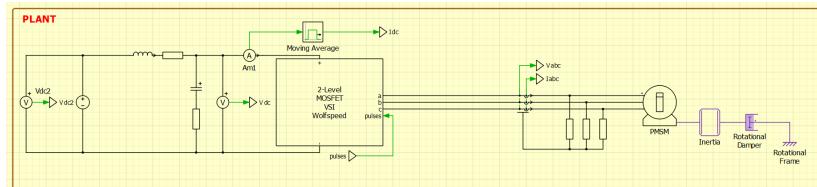


Figura 4.36: Planta electromecánica del conjunto fuente-inversor-motor-carga.

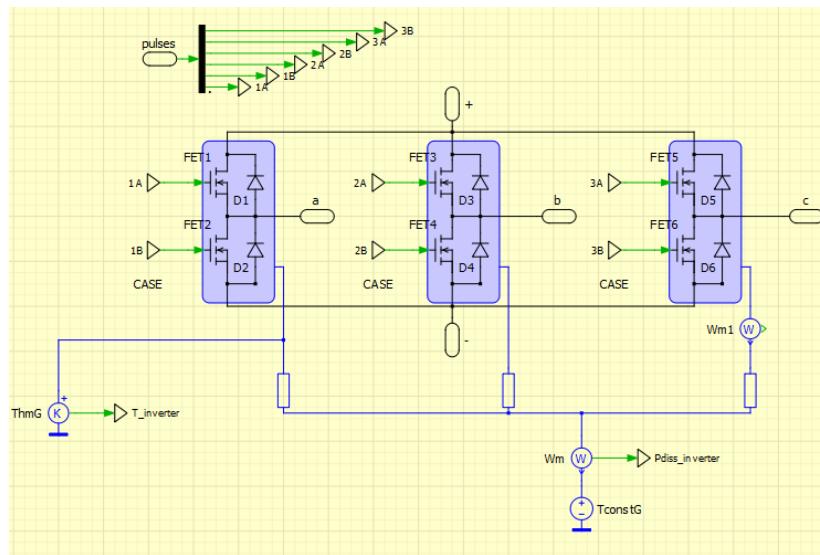


Figura 4.37: Detalle del subbloque VSI, donde se realiza la conexión electrotérmica de los módulos de potencia a la fuente DC, al motor y a la *coldplate*.

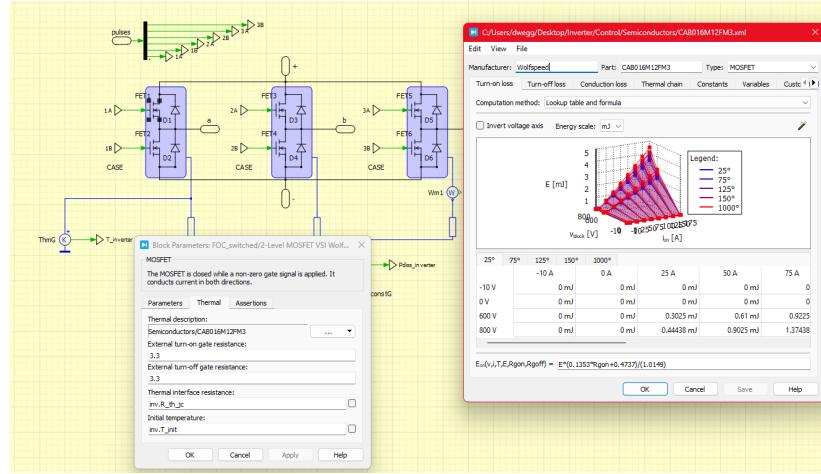


Figura 4.38: Tanto los MOSFETs como los diodos integrados están modelados térmicamente de mano del fabricante, lo que facilita mucho la estimación de pérdidas y la simulación térmica.

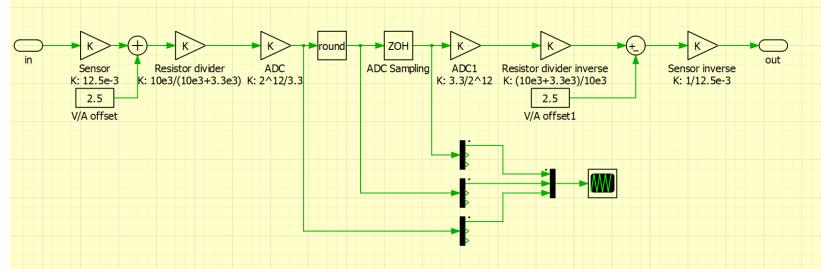


Figura 4.39: Se modela el ADC mediante el uso de *zero order holds* y cuantizando la adquisición y aplicando las ganancias adecuadas.

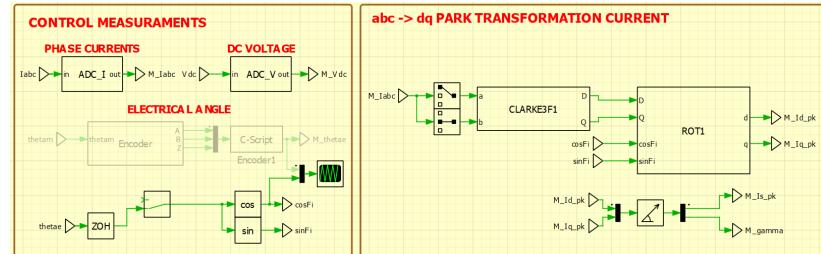


Figura 4.40: Se implementa la transformada de las corrientes, tanto de a, b, c al espacio $d - q$ como de coordenadas cartesianas a coordenadas polares.

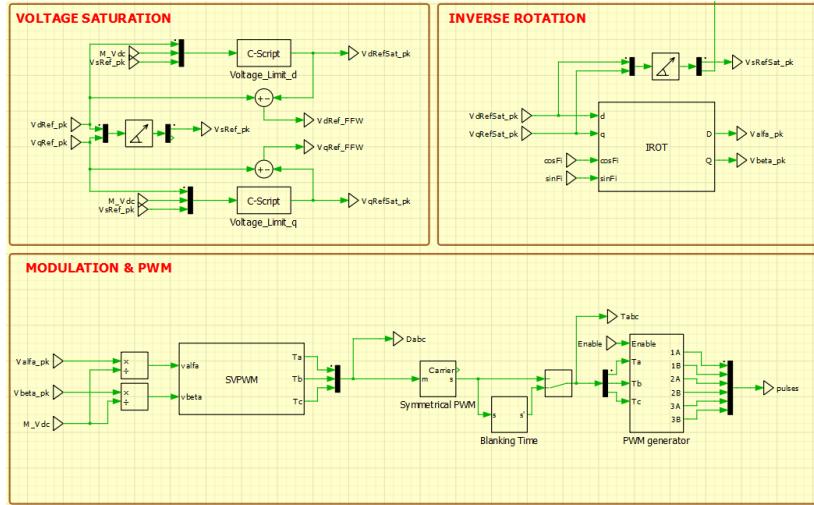


Figura 4.41: Saturación y síntesis de las tensiones V_d y V_q mediante la transformada de Clarke inversa y la modulación SVPWM de la librería PERGAMON. Se modelan también los tiempos muertos de la modulación.

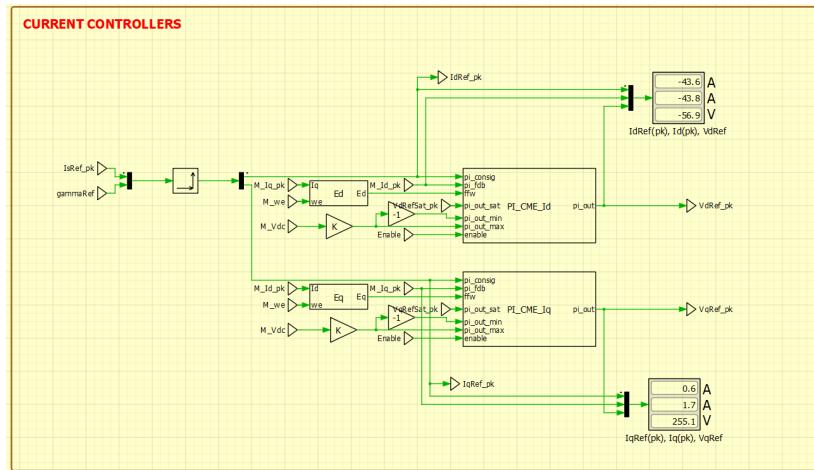


Figura 4.42: Los lazos de corriente están implementados con PI's discretos de la librería PERGAMON y afinados analíticamente con el procedimiento mostrado anteriormente.

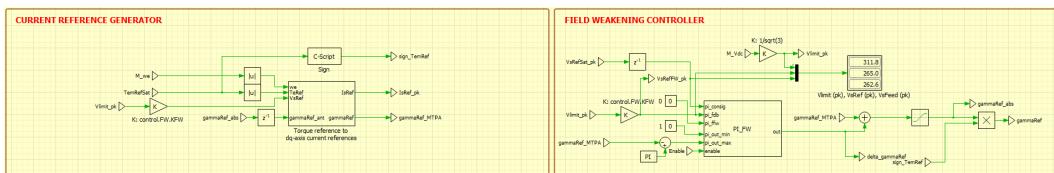


Figura 4.43: Consigna de corriente con debilitamiento de campo.

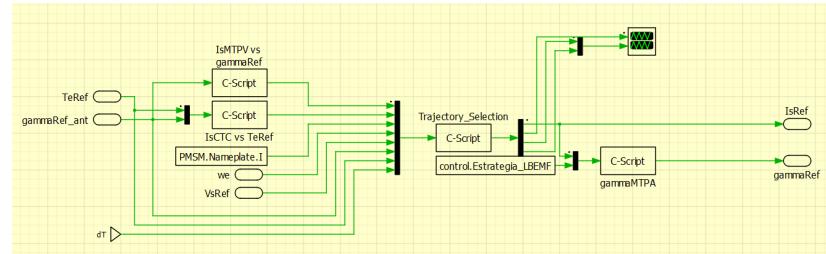


Figura 4.44: Detalle del bloque de cálculo de corriente, donde se implementan las trayectorias de control del PMSM como ecuaciones analíticas resueltas.

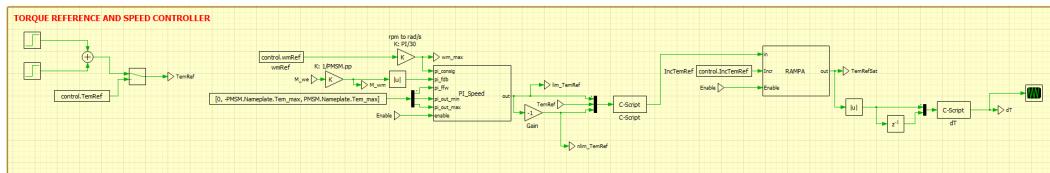


Figura 4.45: Consigna de par y lazo de velocidad como saturación de la consigna de par.

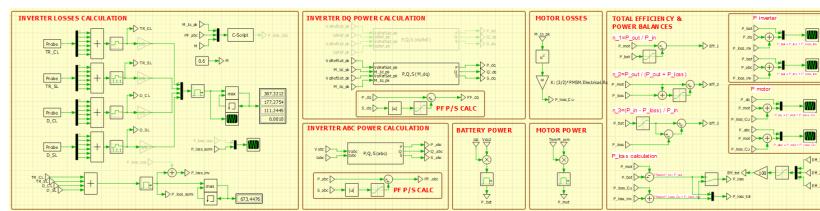


Figura 4.46: Cálculo de pérdidas, eficiencia y factor de potencia.

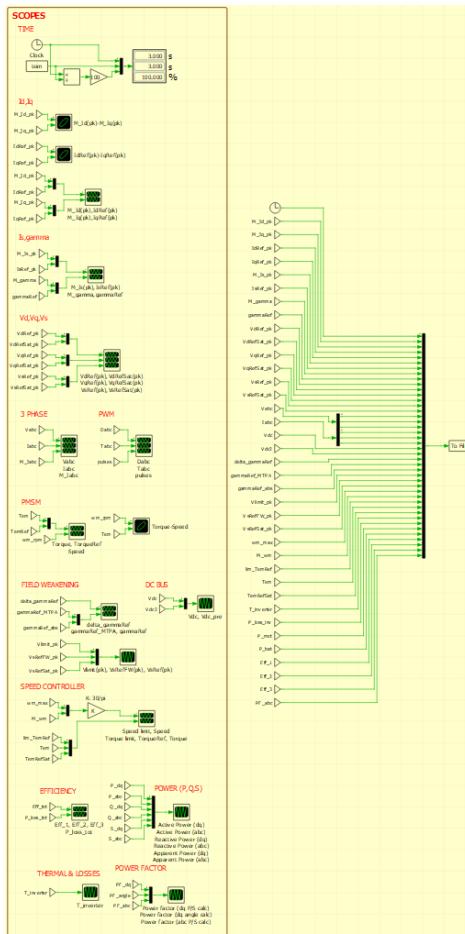


Figura 4.47: Visualización y registro de las variables de interés para el análisis y el procesamiento de datos.

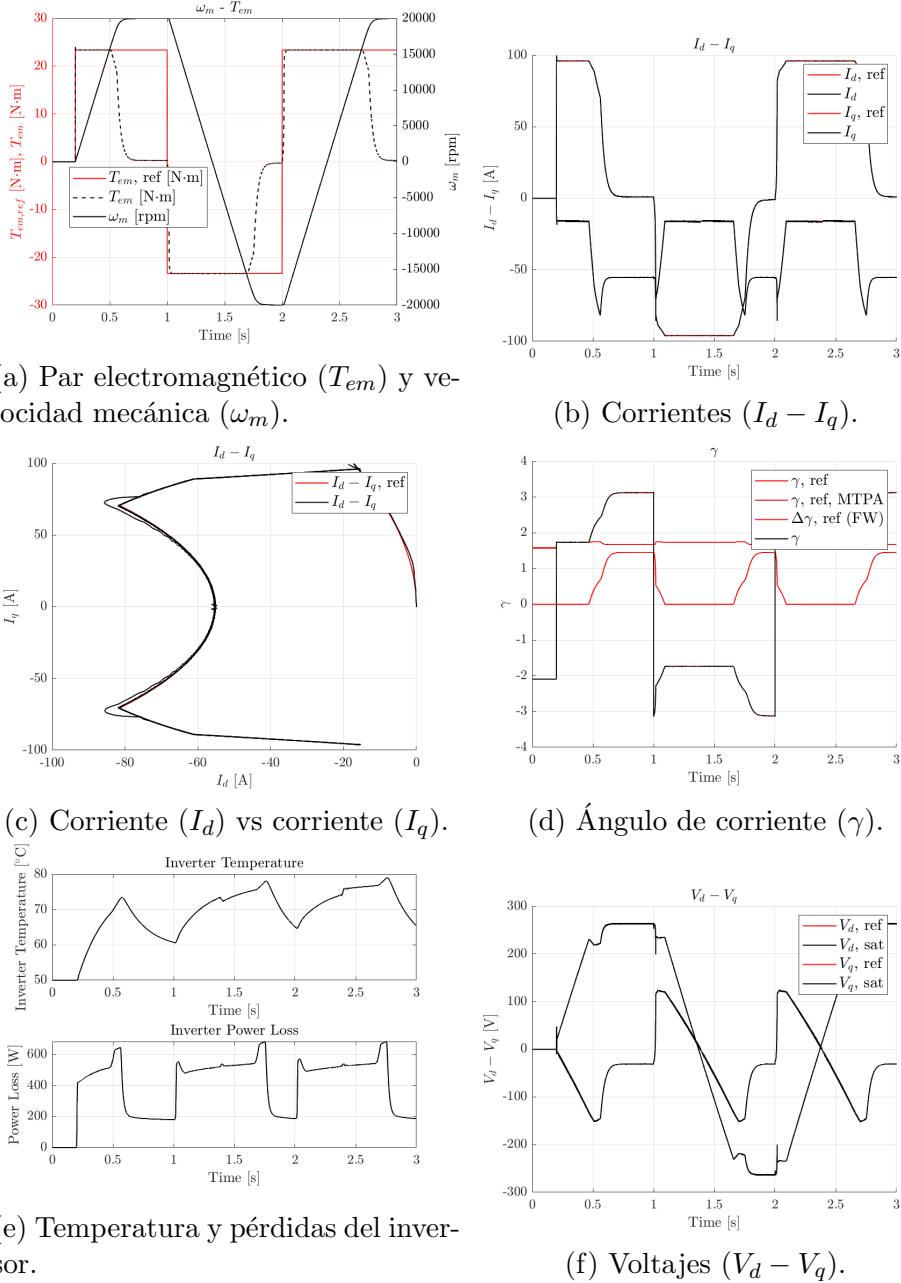


Figura 4.48: Resultados de la simulación.

Como se puede observar, se ha optado por consignar un escalón de par del 90 % del par máximo hasta llegar a la velocidad máxima, donde el mismo control es capaz de limitarla. Posteriormente, se consigna un escalón de par igual pero de signo opuesto, para generar una situación extrema y verificar la robustez del control. De esta manera, se evalúa el comportamiento del control vectorial en las circunstancias más adversas, así como el rendimiento teórico del inversor a potencia máxima en los cuatro cuadrantes de operación del motor eléctrico.

Destaca la claridad con la que esta simulación permite apreciar las trayectorias de control en la figura (c), que muestra la consigna y el valor medido de i_d e i_q .

4.3. Hardware

4.3.1. Requisitos y pre-concepto

Para definir los requisitos del inversor de tracción, es crucial considerar las restricciones y exigencias del tren de potencia que impone la normativa de la competición [1], así como los parámetros eléctricos del vehículo en particular en el cual se va a implementar.

Potencia

La norma **EV2.2.1** dicta que la potencia en la salida de la batería no debe exceder los 80 kW. Esta restricción es crucial en el diseño del inversor, ya que se prevé que el vehículo sea impulsado por un solo convertidor con esta potencia máxima. Por ende, el inversor debe estar dimensionado para manejar máximos de potencia de hasta 80 kW. Dado que se trata de un inversor doble, la asignación de potencia se divide en 40 kW por motor, aunque se podría necesitar más potencia en uno de los motores durante la aceleración en una curva. Esta decisión se toma pensando en el futuro del equipo, cuando implemente 4 motores, lo que dotaría al vehículo de hasta 160 kW de máximo. Sin embargo, en ese escenario, solo se podrían utilizar 80 kW repartidos entre las 4 ruedas, con un máximo de 40 kW por rueda.

La prueba donde se espera utilizar esta potencia máxima es la *Acceleration*, donde se podría requerir de los 80 kW durante un máximo de 10 segundos, siendo esta una aproximación muy conservadora. Por lo tanto, la potencia máxima queda como

$$P_{\text{out, máx, tot}} = 80 \text{ kW} (2 \cdot 40 \text{ kW}) (10 \text{ s máx.})$$

Por otro lado, el requisito de potencia media es distinto. Sería un error dimensionar la potencia media del inversor como el valor máximo, ya que esto conduciría a un sobredimensionamiento excesivo de la refrigeración, los conductores, los conectores y los dispositivos de potencia. En cambio, se opta por determinar el valor medio de potencia requerido durante la prueba más exigente, la *Endurance*. En esta prueba, el vehículo debe recorrer aproximadamente 22 km, lo que equivale a alrededor de media hora de uso continuo. En la *Endurance*, considerar la frenada regenerativa es esencial, ya que aumenta el valor medio de la potencia total, sumándose a la potencia de tracción. Se ha calculado el valor medio de potencia a partir de una simulación de la *Endurance* que llega a máximos de 80 kW:

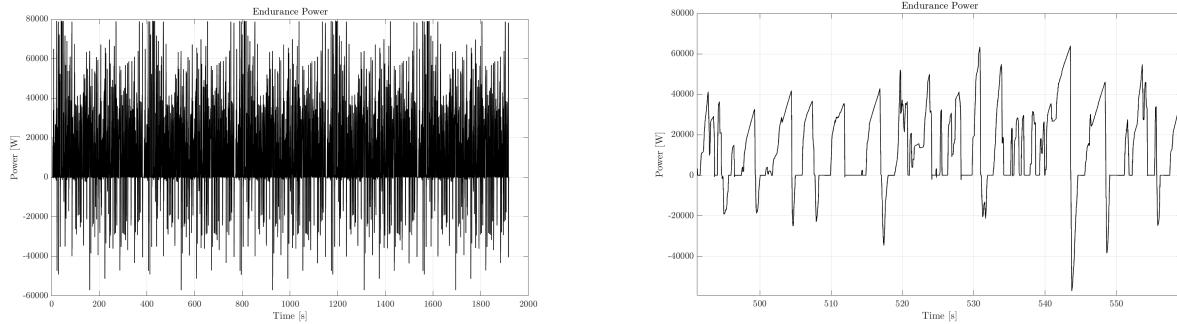


Figura 4.49: Potencia instantánea que sale de la batería en un evento de *Endurance* (Tiempo completo y detalle de un fragmento).

A continuación se muestra el cálculo del valor RMS de la potencia requerida, basado en los datos de potencia:

```
% Calcular el cuadrado de los valores de potencia
power_squared = power_time.Data.^2;

% Calcular el promedio de los valores cuadrados
mean_power_squared = mean(power_squared);

% Tomar la raíz cuadrada del promedio de los valores cuadrados para obtener el valor RMS
power_rms = sqrt(mean_power_squared);

power_rms =
2.3193e+04
```

Esto demuestra que el valor medio de potencia requerido es considerablemente menor que el pico máximo, en concreto un $\frac{23,2\text{kW}}{80\text{kW}} = 29\%$. Por lo tanto, se dimensiona el inversor considerando una potencia media de 35 kW, es decir, 17.5 kW por motor. Esta cifra proporciona un margen de seguridad de 11.8 kW respecto a la simulación, lo que permite algo de juego en el reparto de potencia entre ambos motores. De esta manera, la potencia constante queda como:

$$P_{\text{out, const, tot}} = 35\text{kW}(2 \cdot 17,5\text{kW})$$

Cabe destacar que se está calculando la potencia aparente del convertidor, ya que la potencia activa y la reactiva se reparten en función del factor de potencia, que no es constante y depende de la planta mecánica y la situación del motor. La potencia activa es la que usa el motor para acelerar o frenar, y la reactiva la consume o suministra el bus de condensadores. Otra nota es que la simulación es poco realista puesto que utiliza picos de potencia máxima, cuando en realidad, en la prueba de la *Endurance* se suele limitar el valor de estos picos con tal de extender la autonomía.

Además, existe la norma **EV4.1.1** *The maximum permitted voltage that may occur between any two electrical connections is 600 V,DC and for motor controller/inverters internal low power control signals 630 V,DC.*, que impone el límite de 600 V para la tensión de bus máxima. Es de interés que el inversor esté dimensionado a esta

tensión de funcionamiento, 600 V,DC, para permitir máxima flexibilidad con el diseño de la batería, por ejemplo. Al usar la tensión más alta posible, se obtiene un beneficio en la reducción de la corriente, que implica menos pérdidas en los conductores y permite reducir la sección de cables, pletinas, conectores y otros conductores.

Ya que la estrategia de modulación es SVPWM y la tensión máxima es de 600 V, la tensión alterna fase-neutro de un motor se expresa como:

$$V_{\text{ph-n, pico, } 600 \text{ V,DC}} = \frac{V_{\text{DC}}}{\sqrt{3}} = \frac{600 \text{ V}}{\sqrt{3}} = 346 \text{ V}$$

$$V_{\text{ph-n, RMS, } 600 \text{ V,DC}} = \frac{V_{\text{ph-n, pico}}}{\sqrt{2}} = \frac{346 \text{ V}}{\sqrt{2}} = 245 \text{ V}$$

Sin embargo, la batería no estará constantemente a 600 V,DC, por lo que para poder entregar la potencia de 40 kW pico por inversor en un rango de tensiones adecuado se debería calcular la corriente con una tensión menor. Se escoge 450 V,DC como tensión mínima a partir de la cual se puede entregar la potencia máxima de 40 kW por inversor. Entonces, la corriente de fase de un motor queda:

$$V_{\text{ph-n, pico, } 450 \text{ V,DC}} = \frac{V_{\text{DC}}}{\sqrt{3}} = \frac{450 \text{ V}}{\sqrt{3}} = 261 \text{ V}$$

$$V_{\text{ph-n, RMS, } 450 \text{ V,DC}} = \frac{V_{\text{ph-n, pico}}}{\sqrt{2}} = \frac{261 \text{ V}}{\sqrt{2}} = 184 \text{ V}$$

$$I_{\text{ph, RMS, máx}} = \frac{P_{\text{out, máx}}}{3 \cdot V_{\text{ph-n, RMS, } 450 \text{ V,DC}}} = \frac{40 \text{ kW}}{3 \cdot 184 \text{ V}} = 72 \text{ A}$$

$$I_{\text{ph, pico, máx}} = I_{\text{ph, RMS, máx}} \cdot \sqrt{2} = 72 \text{ A} \cdot \sqrt{2} = 102 \text{ A}$$

De la misma manera que con la potencia, se puede obtener el valor de corriente constante a una tensión menor, ya que el resultado será más restrictivo:

$$I_{\text{ph, RMS, const}} = \frac{P_{\text{out, const}}}{3 \cdot V_{\text{ph-n, RMS, } 450 \text{ V,DC}}} = \frac{17,5 \text{ kW}}{3 \cdot 184 \text{ V}} = 32 \text{ A}$$

$$I_{\text{ph, pico, const}} = I_{\text{ph, RMS, const}} \cdot \sqrt{2} = 32 \text{ A} \cdot \sqrt{2} = 45 \text{ A}$$

Iterando el cálculo de potencia, se puede ver que si se dimensiona a estas corrientes, las potencias máxima y constante calculadas a 600 V,DC son considerablemente mayores:

$$P_{\text{out, máx}} = 3 \cdot V_{\text{ph-n, RMS, } 600 \text{ V,DC}} \cdot I_{\text{ph, RMS, máx}} = 3 \cdot 245 \text{ V} \cdot 72 \text{ A} = 53 \text{ kW}$$

$$P_{\text{out, const}} = 3 \cdot V_{\text{ph-n, RMS, } 600 \text{ V,DC}} \cdot I_{\text{ph, RMS, const}} = 3 \cdot 245 \text{ V} \cdot 32 \text{ A} = 23,5 \text{ kW}$$

$$P_{\text{out, máx, tot}} = 2 \cdot P_{\text{out, máx}} = 2 \cdot 53 \text{ kW} = 106 \text{ kW}$$

$$P_{\text{out, const, tot}} = 2 \cdot P_{\text{out, const}} = 2 \cdot 23,5 \text{ kW} = 47 \text{ kW}$$

Velocidad

Puesto que se va a controlar un motor con posibilidad de operación en debilitamiento de campo, es necesario conocer la frecuencia eléctrica máxima que se va a tener que poder sintetizar. Este requisito va a dimensionar la velocidad de los lazos de control, y en última instancia, la frecuencia de conmutación. El motor en estudio tiene 3 pares de polos y una velocidad angular mecánica máxima de 20000 RPM. Por

lo tanto, la frecuencia eléctrica máxima que se debe sintetizar se calcula de la siguiente manera:

$$f_{e,\text{máx}} = \frac{pp \cdot \omega_{m, \text{máx}, \text{RPM}}}{60} = \frac{3 \cdot 20000}{60} = 1000 \text{ Hz}$$

Dejando un poco de margen para la compatibilidad con motores de más polos, se escoge una frecuencia máxima de 1200 Hz. Para sintetizar esta frecuencia eléctrica con poca distorsión armónica, se debe escoger una frecuencia de conmutación al menos 30 veces superior, para tener como mínimo 30 pulsos por periodo a estas frecuencias.

$$f_{\text{conm}} > 30 \cdot f_{e,\text{máx}} = 30 \cdot 1200 \text{ Hz} = 36 \text{ kHz} \rightarrow f_{\text{conm}} = 40 \text{ kHz}$$

Se debe considerar también que con una frecuencia de conmutación mayor, las pérdidas de conmutación serán más grandes, pero el tamaño del bus de condensadores será más pequeño. Además, el lazo de control debe ir a 40 kHz, con lo que el microcontrolador que lo implemente debe ser capaz de ejecutar todos los cálculos en menos de $\frac{1}{40\text{kHz}} = 25\mu\text{s}$.

Es importante que el control sea mucho más rápido que la planta eléctrica, cuya constante de tiempo es de:

$$\min\left(\frac{L_d}{R_s}, \frac{L_q}{R_s}\right) = \min\left(\frac{188,7\mu\text{H}}{0,15\Omega}, \frac{283,1\mu\text{H}}{0,15\Omega}\right) = \min(1,258\text{ms}, 1,887\text{ms}) = 1,258\text{ms}$$

$$1,258\text{ms} \gg 25\mu\text{s}$$

Normativa

La normativa de la Formula Student no impone normativas muy restrictivas, ya que la mayoría están enfocadas a la seguridad. Estas son las normas que afectarán al diseño del inversor:

- **EV2.2.2** *Regenerating energy is allowed and unrestricted.* La posibilidad de regenerar energía implica que el inversor debe ser capaz de consignar par en el sentido opuesto a la velocidad de los motores para frenar, lo que tendrá un impacto fundamental en el diseño del control. La topología de VSI es bidireccional en sí misma.
- **EV2.2.3** *Wheels must not be spun in reverse.* Podría impactar la lógica de control del inversor y en la adquisición de los sensores de posición para evitar la marcha atrás. Se pueden implementar algoritmos de dependencia de dirección para que ambos motores traccionen el vehículo en la misma dirección.
- **EV3.1.1** *TS enclosures, see EV1.1.2, must consist of either*
 - *a grounded solid layer made of at least 0.5 mm thick electrically conductive material, aluminium or better, having a resistance below 300 mΩ, measured with a current of 1 A, to LVS ground and able to continuously carry at least 10 % of the TS accumulator main fuse current rating or*
 - *be fully made out of electrically insulating materials having an isolation resistance of at least 2 MΩ, measured with a voltage of 500*

V. The enclosure must be rigid and must prevent possible mechanical penetrations. Protruding electrically conductive parts, such as fasteners or connectors, must follow EV3.1.2

The TSAC might use at least 0.9 mm thick steel layer as the grounded layer.

Tendrá efectos en el diseño del empaquetado del convertidor y en la selección de materiales para garantizar la seguridad eléctrica y mecánica del convertidor.

- **EV3.2.5** All overcurrent protection devices that are part of the TS must not rely on programmable logic. The overcurrent protection function of motor controllers/inverters for the motor outputs may rely on programmable logic. Los dispositivos de protección contra sobrecorriente en el sistema de tracción para el motor pueden depender de lógica programable, lo que permite utilizar el sensado de corriente para garantizar esta protección.
- **EV4.1.2** All components in the TS must be rated for the maximum TS voltage. The TS area of a PCB, see EV4.3.6, is considered as one component. Every input connected to the TS must be rated to the maximum TS voltage. Tendrá efectos en la selección de los componentes del circuito de potencia del inversor.
- **EV4.1.3** All components must be rated for the maximum possible temperature that may occur during usage. Impactará en la selección de componentes que puedan manejar las temperaturas esperadas.
- **EV4.2.1** TS enclosures, see EV1.1.2, must be labeled with (a) reasonably sized sticker(s) according to “ISO 7010-W012” (triangle with a black lightning bolt on a yellow background). The sticker must also contain the text “High Voltage” if the voltage is more than 60 V,DC or 50 VACRMS. Impacta en los requisitos de señalización y seguridad del empaquetado.
- **EV4.3.1** The entire TS and LVS must be galvanically isolated, see EV1.2.1 and IN4.1.1. Afecta el diseño de la separación y el aislamiento entre ambos sistemas.
- **EV4.3.4** Where both TS and LVS are present within an enclosure, they must be separated by barriers made of moisture-resistant insulating materials or maintain the following spacing through the air, or over a surface:

Cuadro 4.1: Voltage Spacing Requirements

Voltage	Spacing
$U < 100\text{V,DC}$	10 mm
$100\text{V,DC} < U < 200\text{V,DC}$	20 mm
$U > 200\text{V,DC}$	30 mm

Impacta en el diseño del empaquetado del inversor. Se usarán aislantes como el Nomex en caso de que no se pueda garantizar el espaciado por aire.

- **EV4.3.6** If TS and LVS are on the same PCB, they must be on separate well-defined areas of the board, meeting the spacing requirements in table 5, each area

clearly marked with “TS” or “LV”. The outline of the area required for spacing must be marked. “Conformal coating” refers to a coating insulator, solder resist is not a coating.

Cuadro 4.2: Voltage Spacing Requirements

Voltage	Over Surface	Through Air (Cut in board)	Conformal Coating
0 to 50	1.6 mm	1.6 mm	1.0 mm
50 to 150	6.4 mm	3.2 mm	2.0 mm
150 to 300	9.5 mm	6.4 mm	3.0 mm
300 to 600	12.7 mm	9.5 mm	4.0 mm

Tendrá efectos en el diseño de la PCB de potencia.

- **EV4.5.3** *The temperature rating for TS wiring, connections, and insulation must be appropriate for the expected surrounding temperatures but at least 85 °C.* Impactará en la selección de cables y materiales aislantes.
- **EV4.5.4** *TS components and containers must be protected from moisture in the form of rain or puddles, see IN9.* Impactará en el diseño de la caja.
- **EV4.5.6** *All TS wiring must be completed to professional standards with appropriately sized conductors and terminals and with adequate strain relief and protection from loosening due to vibration etc.* Tendrá efectos en el diseño de las conexiones y terminales.
- **EV4.5.10** *Every TS connector outside of a housing must include a pilot contact/interlock line which is part of the SDC. Housings only used to avoid interlocks are prohibited.* Impactará en el diseño de los conectores y sistemas de interconexión.
- **EV4.5.11** *All TS connections must be designed so that they use intentional current paths through conductors such as copper or aluminium and must not rely on steel bolts to be the primary conductor.* Tendrá efectos en el diseño de las conexiones y la selección de materiales adecuados.
- **EV4.5.12** *All TS connections must not include compressible material such as plastic in the stack-up or as a fastener. FR-4 is allowed.* Impactará en el diseño de las conexiones y la selección de materiales adecuados.
- **EV4.5.13** *TS connectors outside of TS enclosures must be designed in a way, that the TS cannot be activated, see EV4.11, if connected in any way other than the design intent configuration.* Tendrá efectos en el diseño de los conectores y la seguridad.
- **EV4.5.14** *All electrical connections, including bolts, nuts, and other fasteners, in the high current path of the TS must be secured from unintentional loosening. Fasteners must use positive locking mechanisms, see T10.2, that are suitable for high temperatures, see EV4.5.3.* Impactará en el diseño de los sistemas de fijación y bloqueo, especialmente en las conexiones de potencia.

- **EV4.5.16** *Soldered connections in the high current path are only allowed if all of the following are true:*
 - *connections on PCBs*
 - *the connected devices are not cells or wires*
 - *the devices are additionally mechanically secured against loosening*

Tendrá efectos en el diseño de las conexiones y el uso de soldaduras. El uso de sistemas alternativos de conexión en placa como el *press-fit*, facilita el cumplimiento de esta norma.

- **EV4.9.1** *If a discharge circuit is required to meet EV6.1.5, it must be designed to handle the maximum TS voltage permanently. After three subsequent discharges within 15 s in total, the discharge time specified in EV6.1.5 may be exceeded. Full discharging functionality must be given after a reasonable time with a deactivated discharge circuit.* Se deberá integrar un circuito de descarga en el inversor que cumpla con estos requisitos.
- **EV4.9.2** *The discharge circuit must be wired in a way that it is always active whenever the SDC is open. Furthermore, the discharge circuit must be fail-safe such that it still discharges the intermediate circuit capacitors if the HVD has been opened or the TS accumulator is disconnected.* Tendrá efectos en la implementación del circuito de descarga.
- **EV2.2.1** *EV4.10.2 The TSAL itself must have a red light, flashing continuously with a frequency between 2 Hz and 5 Hz and a duty cycle of 50 %, active if and only if the LVS is active and the voltage across any DC-link capacitor exceeds*
 - *60 V,DC or 50 VACRMS*
 - *Half the nominal TS voltage*

whichever is lower

Se deberá implementar esta detección.

- **EV4.10.4** *The mentioned voltage detection must be performed inside the respective TS enclosure.* Obliga a situar la detección de alta tensión en el propio bus de condensadores.

Estas restricciones no solo definen los límites de diseño del inversor, sino que también influyen en la selección de componentes y la estrategia de control.

Resumen de Requisitos de *Hardware*

Con toda la información que se ha recopilado se pueden listar los requisitos más importantes del convertidor:

- **Topología del convertidor:** VSI doble.
- **Potencia pico:** 80 kW ($2 \cdot 40$ kW).

- **Potencia constante:** 35 kW ($2 \cdot 17,5$ kW).
- **Tensión DC máxima:** 600 V.
- **Tensión fase-neutro sintetizada máxima:** 245 V,RMS (SVPWM).
- **Corriente de fase máxima:** 80 A,RMS.
- **Aislamiento galvánico entre TS y LVS.**
- **Detección de tensión.**
- **Circuito de descarga integrado.**
- **Interfaz por comunicación CAN.**

Requisitos Adicionales

Además de las restricciones normativas, los requisitos del inversor también deben tener en cuenta las demandas específicas del sistema de tracción y del equipo para el cual se está diseñando. Entre estos requisitos se encuentran:

- **Densidad de Potencia:** El inversor debe ser compacto para facilitar la integración en el vehículo.
- **Eficiencia:** Se busca una eficiencia óptima para maximizar la autonomía del monoplaza y llevar el máximo de energía de la batería a los neumáticos.
- **Fiabilidad:** Dado el entorno de competición, el inversor debe ser altamente confiable y capaz de operar en las condiciones de una competición, como temperaturas ambientales y humedades altas.
- **Flexibilidad:** El diseño del inversor debe ser lo suficientemente flexible para adaptarse a posibles cambios en el sistema de tracción y comunicación con el vehículo.
- **Facilidad de mantenimiento:** Consideraciones para facilitar el mantenimiento y la evolución del proyecto a largo plazo.

Boceto del empaquetado

Antes de entrar en el diseño detallado del convertidor, se realizaron algunos bocetos preliminares para visualizar la disposición de los componentes y planificar la distribución espacial. Estos bocetos iniciales sirven como punto de partida para el diseño final del empaquetado del inversor.

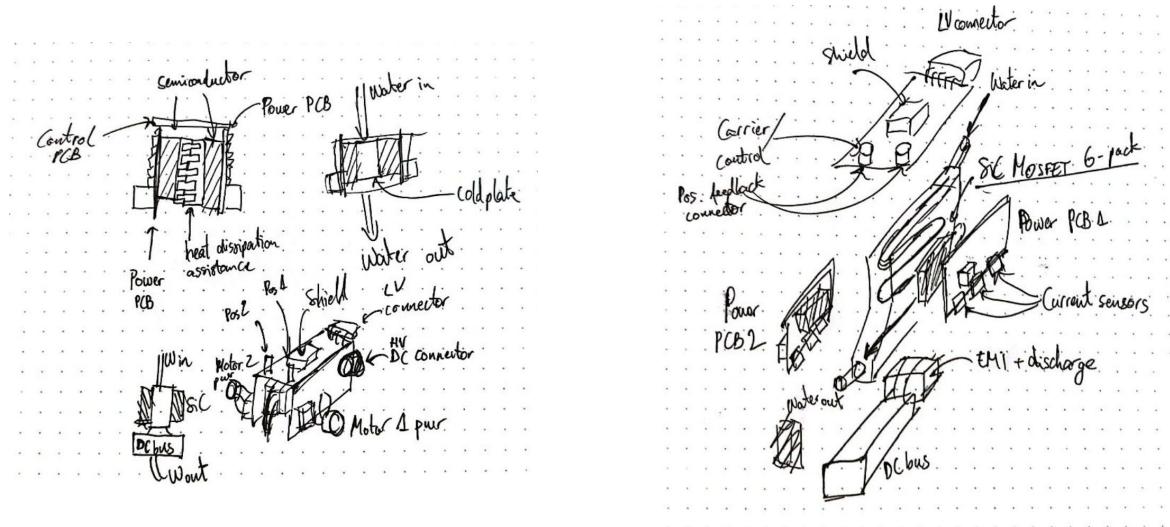


Figura 4.50: Bocetos preliminares del empaquetado del inversor.

En estos dos primeros conceptos se explora el uso de una sola *coldplate* para refrigerar ambos semiconductores. Adicionalmente se propone refrigerar el bus de condensadores, aunque posteriormente se encuentra que no es necesario. Se ha optado por una configuración que separa claramente los módulos de potencia de la placa de control, facilitando así el mantenimiento y la disipación de calor.

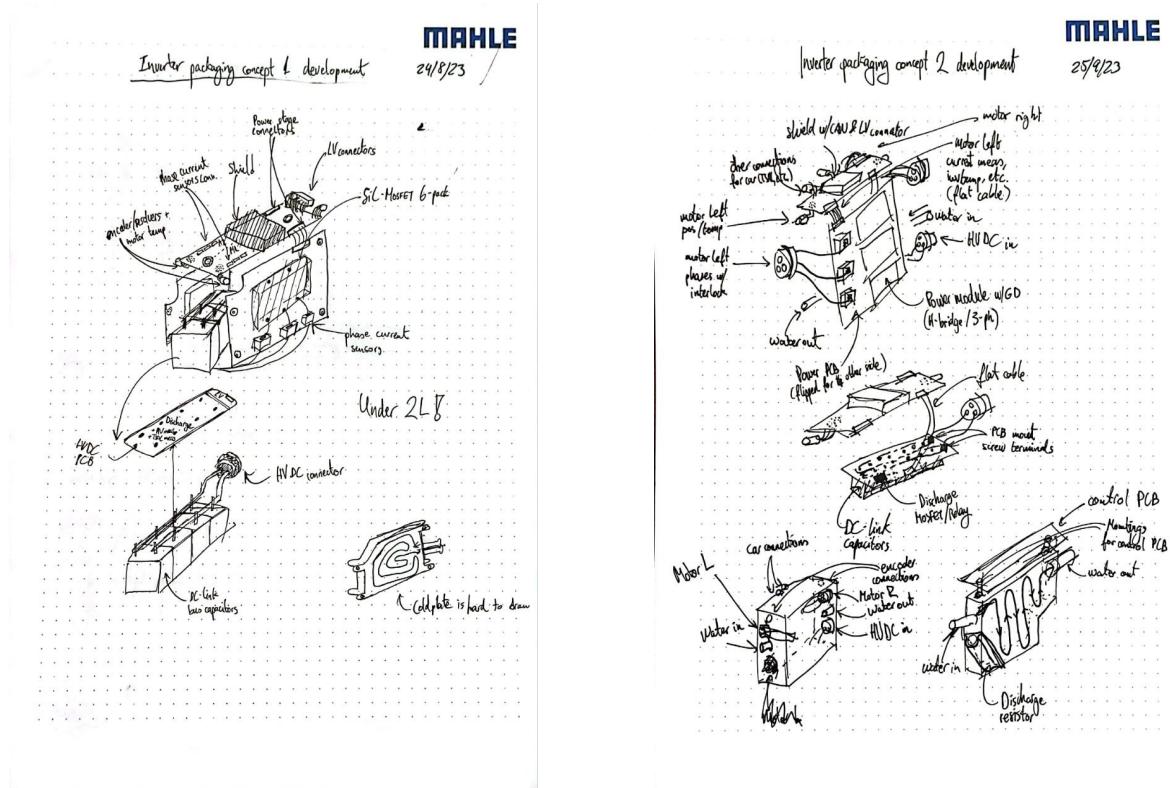


Figura 4.51: Evolución de los bocetos del empaquetado del inversor.

En los bocetos posteriores, se ha refinado la disposición de los componentes para

mejorar la accesibilidad y la eficiencia de refrigeración. La disposición de los conectores y la distribución de los conductores también se ha ajustado para optimizar la densidad de potencia y la eficiencia del diseño. Estos cambios reflejan un enfoque hacia un empaquetado más eficiente y funcional del inversor. De todas maneras, todavía presentan muchas complicaciones en el ensamblaje y fabricación de algunos elementos como la *coldplate*.

Estos bocetos iniciales proporcionan una idea visual de cómo se podría organizar el empaquetado del inversor. A medida que avanza el diseño, se realizarán ajustes a la vez que se encuentren limitaciones con los componentes específicos seleccionados y se hagan consideraciones eléctricas y térmicas que en este punto no se han tenido.

4.3.2. Topología y concepto

Con el fin de producir las tensiones trifásicas que calcula el control para los dos motores se implementa un ondulador trifásico fuente de voltaje doble (*dual VSI*). Poniendo el foco en uno solo de los convertidores, existen varias topologías que permiten hacer un ondulador trifásico, pero la más utilizada es el VSI de 2 niveles, formado por tres ramas de medio puente. Otras topologías de más niveles logran sintetizar las tensiones con menos distorsión armónica, pero dado que el motor eléctrico es una carga inductiva, y que el control está basado en consignas de corriente, la distorsión armónica del convertidor tiene un impacto muy poco significativo en el rendimiento global. De esta manera, se justifica la implementación del VSI dual.

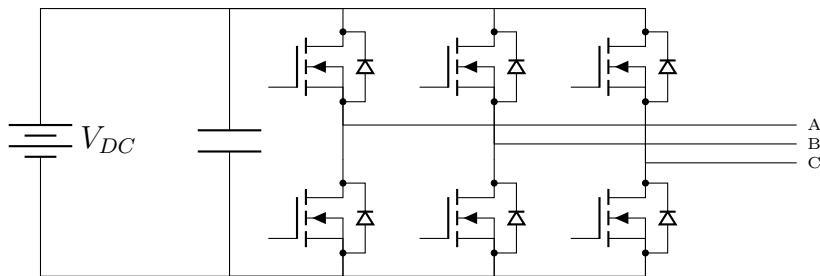


Figura 4.52: VSI con MOSFETs.

Ya que el convertidor implementa código propio, se decide implementar dos VSI en el mismo convertidor, de modo que se opta por unificar algunos elementos de ambos como la refrigeración con el fin de optimizar el peso y el espacio. Esta decisión evita duplicados de medidas, componentes y *software*, sin embargo, conlleva otros retos, como por ejemplo, que el microcontrolador deberá ser capaz de ejecutar los dos lazos de control al mismo tiempo.

Para adquirir las variables necesarias para el control y la interfaz con el vehículo y sus periféricos, se requieren los submódulos que se presentan en el diagrama de bloques 4.53.

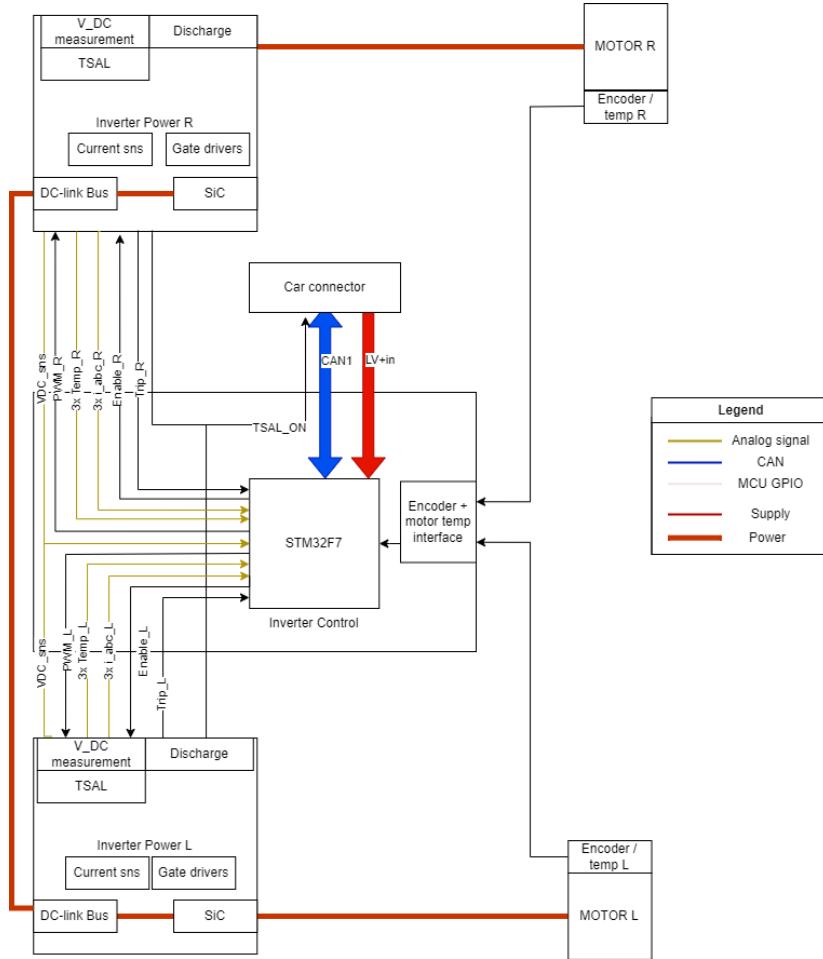


Figura 4.53: Diagrama de bloques del sistema completo.

El convertidor se divide en 3 PCBs, *Inverter Power*, que se instala por duplicado (una por motor), e *Inverter Control*. Como su nombre indica, *Inverter Power* alojará todos los componentes de potencia, incluyendo los *gate drivers*, los sensores de corriente, los semiconductores, el bus de condensadores y los conectores de potencia. Además, en esta placa se puede encontrar el sensado de la tensión de los buses, los circuitos de descarga, y la detección de alta tensión para la *TSAL*. El sensado de tensión sería el único subcircuito repetido, pero esto solamente aporta redundancia y tolerancia al fallo.

Por su parte, *Inverter Control* contiene el microcontrolador de la familia STM32F7 para consignar la conmutación a los *gate drivers*, así como para realizar las lecturas de corriente, tensión y temperatura, las interfaces con los sensores de posición de los motores o los protocolos comunicación CAN y USB.

El montaje de estas placas se realiza de forma compacta y eficiente, de manera que cabe una *coldplate* entre medias de las dos de potencia para refrigerar los semiconductores de ambos inversores. Se encajan de manera que la placa de control se puede conectar a las placas de potencia mediante conectores placa a placa. Las conexiones de potencia facilitan la integración con el cableado del monoplaza debido a su posicionamiento.

4.3.3. Semiconductores

Tecnología de semiconductores

La decisión de emplear MOSFETs de carburo de silicio (SiC) como interruptores se fundamenta en consideraciones específicas de la aplicación y en los requisitos de la competición. En un entorno donde la reducción de peso y volumen es crucial, el SiC emerge como una opción destacada frente a alternativas como el IGBT de silicio, el FET de GaN o el MOSFET de silicio. Aunque el precio no es una limitación primordial en este caso, el proyecto ha atraído el interés de empresas que han ofrecido muestras de semiconductores de forma gratuita para el desarrollo del convertidor. Esto ha sido beneficioso, ya que aunque se cuenta con un presupuesto, recibir estas muestras adicionales ayuda a reducir costos y aprovechar al máximo los recursos disponibles.

Las ventajas inherentes del SiC, como su rendimiento superior o su resistencia a temperaturas más altas, permiten un diseño más compacto y robusto del inversor de tracción. Estas características son esenciales para cumplir con los requisitos de un monoplaza de competición, donde lograr altas densidades de potencia es muy beneficioso para la integración del resto de componentes del vehículo y aligeramiento del mismo.

Módulos de potencia

En el diseño del inversor de tracción, se optó por módulos *half-bridge* debido a su idoneidad para el rango de potencias y tensiones del convertidor. Dos modelos de semiconductores se consideraron para su integración: el DFS05HF12EYR1 de Leapers Semiconductor y el CAB016M12FM3 de Wolfspeed.

Ambos modelos cumplen estrictamente con los requisitos de la aplicación, con un voltaje de ruptura ($V_{DS,\text{breakdown}}$) de 1200V y una corriente máxima ($I_{DS,\text{máx}}$) que excede los 80 A,RMS. La elección de dos modelos distintos se basa en la intención de realizar pruebas comparativas para verificar las diferencias entre ambos.

El DFS05HF12EYR1 ofrece especificaciones muy buenas en su datasheet, aunque Leapers Semiconductors no lleva muchos años en la industria y no han logrado crear la confianza que otras empresas han conseguido con su experiencia. Por otro lado, el CAB016M12FM3, de la reconocida marca Wolfspeed (antiguamente CREE), aporta la confiabilidad asociada a una empresa con amplia experiencia en el campo.

Según sus respectivos datasheets, ambos modelos permiten alcanzar sin mucho esfuerzo una frecuencia de conmutación de 40 kHz, lo que contribuye significativamente a la reducción del tamaño del bus de continua y optimiza el empaquetado del inversor. La placa de potencia se diseñará para permitir la prueba de ambos modelos, ya que comparten *footprint*, facilitando la adaptabilidad y la evaluación comparativa.

A continuación se muestra una tabla comparativa con los detalles de cada semiconductor:

Parámetro	DFS05HF12EYR1	CAB016M12FM3
$V_{DS,\text{breakdown}}$ [V]	1200	1200
R_{on} [$m\Omega$]	5.5 - 13	16.0 - 28.8
$V_{f,D}$ [V]	3.3 - 4	4.9 - 5.5
T_{rr} [ns]	41.5 - 45	20.0
Q_{rr} [μC]	2.19 - 3.94	1.30
$R_{th,jc}$ [K/W]	0.12 - 0.15	0.543
Q_G [nC]	520	236
C_{in} [nF]	14.5	6.6
$R_{G,\text{int}}$ [Ω]	1.9	2.4
$V_{GS,\text{th}}$ [V]	2.8 - 4.8	1.8 - 3.6
$I_{DS,\text{max}}$ [A]	150	89
		

Cuadro 4.3: Comparación de parámetros entre DFS05HF12EYR1 y CAB016M12FM3.

Análisis de pérdidas

Los dispositivos semiconductores experimentan pérdidas de energía que influyen significativamente en su eficiencia y desempeño, y se transforman en calor, limitando así la potencia de salida. Estas pérdidas pueden clasificarse en dos categorías fundamentales: las pérdidas de conducción y las pérdidas de conmutación. Las **pérdidas de conducción** surgen cuando el dispositivo se encuentra en estado activo, conduciendo corriente a través de él. Esto genera una caída de voltaje y una disipación de potencia asociada a la resistencia interna del dispositivo. Por otro lado, las **pérdidas de conmutación** se manifiestan durante los ciclos de activación y desactivación del dispositivo, cuando la energía almacenada en la capacitancia del mismo se disipa durante la transición entre los estados de conducción y corte.

En el análisis de la eficiencia de los semiconductores, es imperativo considerar las pérdidas totales, las cuales se definen como la suma de las pérdidas de conducción y las de conmutación:

$$P_{\text{tot}} = P_{\text{cond}} + P_{\text{comm}} \quad (4.56)$$

A continuación, se examinarán detalladamente las pérdidas de conducción, seguidas por un análisis exhaustivo de las pérdidas de conmutación en los módulos seleccionados.

Pérdidas de conducción

Las pérdidas de conducción tienen su origen en la resistencia entre drenador y fuente cuando el MOSFET está en estado de conducción, o en la caída de tensión del diodo cuando es el diodo quien está conduciendo. Según la ley de Ohm

$$P_{\text{cond,MOSFET}} = I_{\text{DS}}^2 \cdot R_{\text{DS, on}}, \quad (4.57)$$

y la definición de potencia

$$P_{\text{cond,D}} = I_{\text{SD}} \cdot V_f. \quad (4.58)$$

Sin embargo, estas expresiones se corresponden con las pérdidas instantáneas y no son útiles a la hora de dimensionar el convertidor. Para ello son necesarias las pérdidas promediadas, pero tienen una expresión analítica muy compleja, ya que dependen de la estrategia de modulación, el índice de modulación instantáneo, el factor de potencia instantáneo, etc. Para el inversor se utiliza SVPWM, pero el cálculo de pérdidas tiene demasiados parámetros instantáneos. Muchas referencias asemejan las pérdidas en SVPWM a las de una modulación PWM sinusoidal (SPWM), con lo que las pérdidas se simplifican.

$$P_{\text{cond,MOSFET}} \approx I_{\text{RMS}} V_{Q0} \left(\frac{1}{\pi\sqrt{2}} + \frac{m \cdot \cos(\phi)}{2\sqrt{8}} \right) + I_{\text{RMS}}^2 R_Q \left(\frac{1}{4} + \frac{2 \cdot m}{3\pi \cdot \cos(\phi)} \right) \quad (4.59)$$

$$P_{\text{cond,D}} \approx I_{\text{RMS}} V_{D0} \left(\frac{1}{\pi\sqrt{2}} - \frac{m \cdot \cos(\phi)}{2\sqrt{8}} \right) + I_{\text{RMS}}^2 R_D \left(\frac{1}{4} - \frac{2 \cdot m}{3\pi \cdot \cos(\phi)} \right) \quad (4.60)$$

Como se puede apreciar, aunque son más compactas que otras en la bibliografía, estas expresiones son difíciles de abordar, puesto que casi todos los parámetros son instantáneos. Por ello, se ha usado PLECS para estimar las pérdidas de conducción debido a que facilita la simulación detallada de los semiconductores y permite obtener las pérdidas de forma diseccionada por cada dispositivo y tipo de pérdida. PLECS utiliza modelos detallados de los dispositivos, considerando sus características de conmutación y conducción reales.

Con los parámetros de estos modelos, PLECS calcula las pérdidas de conmutación y conducción de manera precisa. Sin embargo, como no tiene en cuenta el circuito completo de los *gate drivers*, no se pueden tomar directamente las pérdidas de conmutación. Se usa un modelo de MOSFET y diodo de Wolfspeed proporcionado directamente por el fabricante. Este modelo cuenta con una parametrización de las pérdidas y los efectos térmicos mucho más detallada que la que se puede obtener a partir de la hoja de datos. El fabricante utiliza tablas de búsqueda en función de la tensión, corriente y temperatura, e incorpora una relación matemática con las resistencias de puerta. Lamentablemente, Leapers no ofrece esta parametrización con sus semiconductores, por lo que el análisis en PLECS se realiza únicamente con el modelo de Wolfspeed, siendo este el más restrictivo en cuanto a las pérdidas de conducción.

Para sacarle el máximo partido, se ha obtenido un perfil de pérdidas a máxima potencia para el inversor en diferentes zonas de control del PMSM, que se puede observar a continuación.

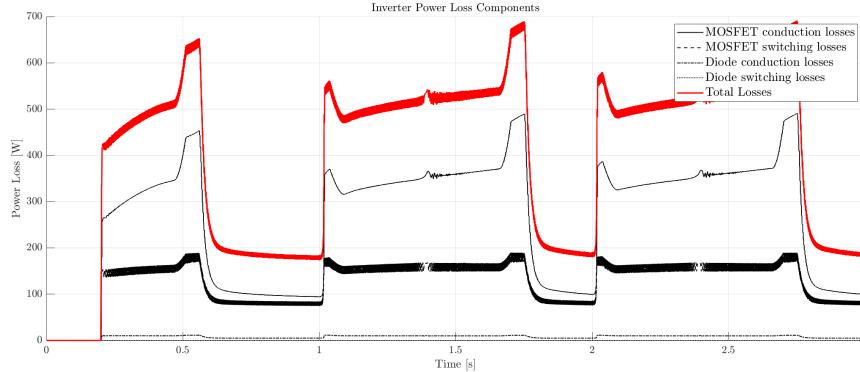


Figura 4.54: Descomposición de las pérdidas de los semiconductores de una simulación en PLECS.

Se puede ver como las pérdidas de conducción son más significativas que las de conmutación, y varían significativamente. Las zonas de par constante (de 1,05 s a 1,4 s por ejemplo), muestran unas pérdidas casi constantes con la corriente, pero crecen debido a la temperatura. Se anota para una comparación posterior que las pérdidas de conmutación de los MOSFETs están entre 150 y 200 W. Al entrar en la zona de límite de tensión (de 1,4 s a 1,45 s), se observa un pico que se corresponde con la zona de potencia constante, que en esta simulación es la potencia máxima. Para obtener un valor constante de pérdidas con el que diseñar la refrigeración, se parte de este valor de pico (450 W aproximadamente) y se trata de la misma manera que la relación de potencia media respecto a la potencia pico calculada en la sección de requisitos.

$$\frac{P_{\text{RMS}}}{P_{\text{pico}}} = \frac{35 \text{ kW}}{80 \text{ kW}} = 0,4375 \approx 0,5$$

Con este razonamiento, se da un valor de pérdidas de conducción máximas de un inversor de $450 \text{ W} \cdot 0,5 = 225 \text{ W}$. Como hay dos inversores anclados a la misma *coldplate*, las pérdidas de conducción que debe disipar es de $225 \text{ W} \cdot 2 = 450 \text{ W}$. Es importante recordar que estas pérdidas son para el semiconductor de Wolfspeed. Dado que las pérdidas de conducción son proporcionales a $R_{DS, \text{on}}$, se pueden aproximar las pérdidas de conducción del semiconductor de Leapers.

$$P_{\text{cond, Wolfspeed}} = 450 \text{ W}$$

$$P_{\text{cond, Leapers}} \approx P_{\text{cond, Wolfspeed}} \cdot \frac{R_{DS, \text{on, Leapers}}}{R_{DS, \text{on, Wolfspeed}}} = 450 \text{ W} \cdot \frac{5,5 \text{ m}\Omega}{16 \text{ m}\Omega} \approx 155 \text{ W}$$

Pérdidas de conmutación

El cálculo de las pérdidas de conmutación es más fácil de determinar de forma analítica, aunque depende del circuito de *gate driver* y de sus parásitos. Si no se tienen en cuenta estos dos factores, se pueden calcular utilizando los valores de E_{on} y E_{off} ,

que representan la energía necesaria para encender o apagar el dispositivo respectivamente. Cuando las hojas de datos de los semiconductores proporcionan los valores E_{on} y E_{off} , las pérdidas de conmutación pueden calcularse de forma directa. Estas pérdidas están influenciadas por la tensión de conmutación, la corriente de conmutación, la temperatura de la unión y la resistencia de puerta. La fórmula comúnmente utilizada para calcular las pérdidas es

$$E_{\text{conm}}(t) = (E_{on} + E_{off}) \left(\frac{I_{Q,\text{env}}(t)}{I_{\text{test}}} \right)^{K_i} \left(\frac{V_{DC}}{V_{\text{test}}} \right)^{K_v}, \quad (4.61)$$

donde $I_{Q,\text{env}}(t)$ es la corriente de conmutación en cada instante de tiempo, I_{test} y V_{test} son los valores de prueba, y K_i y K_v son coeficientes de ajuste que normalmente son igual a 1. La justificación se puede encontrar en los gráficos proporcionados en las hojas de datos de ambos semiconductores, donde la relación de las energías de encendido y apagado con la corriente son aproximadamente lineales.

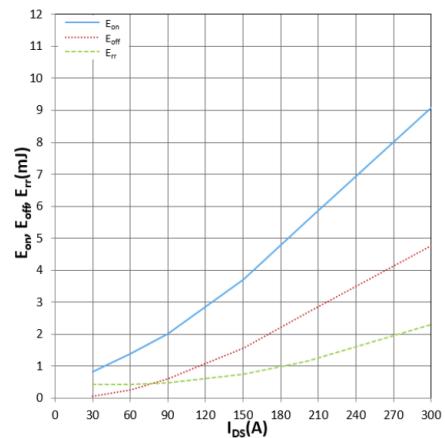


Figure 10. E_{on} , E_{off} , E_{rr} vs I_{DS}
 $T_j = 25^\circ\text{C}$, $R_G = 3.3\Omega$, $V_{GS} = +18\text{V}/0\text{V}$

(a) Leapers

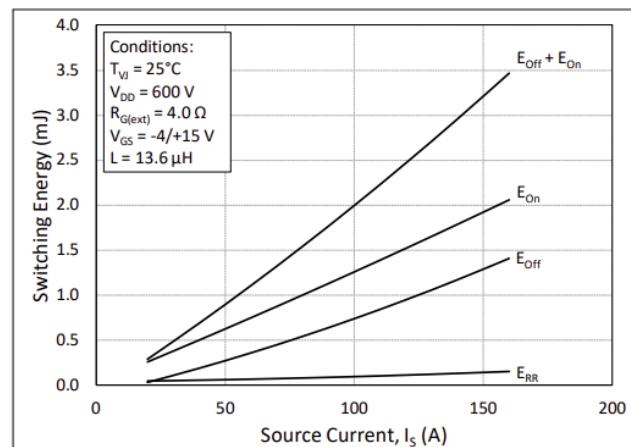


Figure 11. Switching Energy vs. Drain Current ($V_{DD} = 600\text{ V}$)

(b) Wolfspeed

Figura 4.55: Energías de encendido y apagado en función de la corriente para Leapers y Wolfspeed.

Si la corriente es continua,

$$P_{\text{conm}} \approx (E_{on} + E_{off}) f_{\text{conm}} \frac{I_{\text{out}}}{I_{\text{test}}} \frac{V_{DC}}{V_{\text{test}}}. \quad (4.62)$$

Y si la corriente es una sinusoidal semi-rectificada con un valor de pico de $\sqrt{2}I_{\text{RMS}}$ (como en el caso del SVPWM),

$$P_{\text{conm}} \approx (E_{on} + E_{off}) f_{\text{conm}} \frac{\sqrt{2}I_{\text{RMS}}}{\pi I_{\text{test}}} \frac{V_{DC}}{V_{\text{test}}}. \quad (4.63)$$

Para obtener los valores de E_{on} y E_{off} se pueden usar los gráficos que aparecen en las hojas de datos, ya que principalmente dependen de la resistencia de puerta que se escoja.

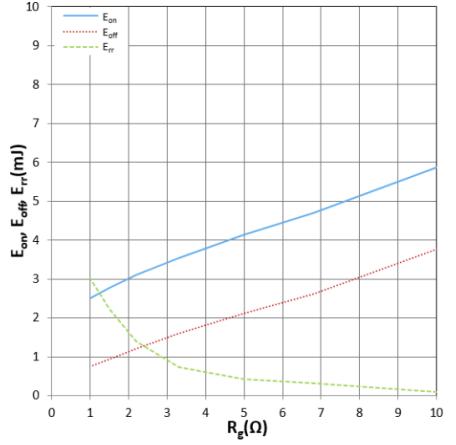


Figure 9. E_{on} , E_{off} , E_{tr} vs R_g
 $T_j = 25^\circ\text{C}$, $I_D = 150\text{A}$, $V_{GS} = +18\text{V}/0\text{V}$

(a) Leapers

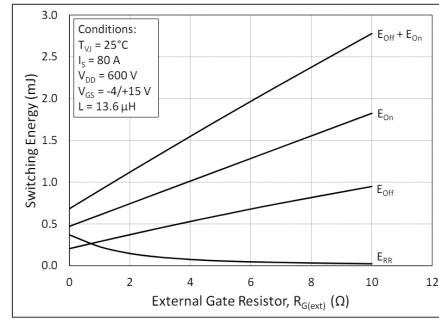


Figure 15. MOSFET Switching Energy vs. External Gate Resistance

(b) Wolfspeed

Figura 4.56: Energías de encendido y apagado en función de la resistencia de puerta externa para Leapers y Wolfspeed.

Se escogerán los valores más grandes de resistencia de puerta para hacer este cálculo, pues un valor más grande produce menos sobrepico de tensión en la conmutación, pero aumenta las pérdidas. Sustituyendo por los valores del convertidor en diseño,

- Generales: $f_{conm} = 40 \text{ kHz}$, $I_{RMS} = 80 \text{ A}$, $V_{DC} = 600 \text{ V}$
- Leapers: $E_{on} = 5,8 \text{ mJ}$, $E_{off} = 3,7 \text{ mJ}$, $V_{test} = 600 \text{ V}$, $I_{test} = 150 \text{ A}$
- Wolfspeed: $E_{on} = 1,8 \text{ mJ}$, $E_{off} = 0,9 \text{ mJ}$, $V_{test} = 600 \text{ V}$, $I_{test} = 80 \text{ A}$

Sustituyendo para los dispositivos Leapers,

$$P_{conm, \text{Leapers}} \approx (5,8 \text{ mJ} + 3,7 \text{ mJ}) \cdot 40 \text{ kHz} \cdot \frac{80 \text{ A}}{150 \text{ A}} \cdot \frac{600 \text{ V}}{600 \text{ V}} \approx 91,23 \text{ W},$$

y para los dispositivos Wolfspeed,

$$P_{conm, \text{Wolfspeed}} \approx (1,13 \text{ mJ} + 0,54 \text{ mJ}) \cdot 40 \text{ kHz} \cdot \frac{80 \text{ A}}{80 \text{ A}} \cdot \frac{600 \text{ V}}{600 \text{ V}} \approx 48,61 \text{ W}.$$

Dado que estas pérdidas son para un solo MOSFET, se deben multiplicar por 12 para obtener las pérdidas de conmutación totales, dado que hay 6 dispositivos de potencia en la topología VSI y hay dos VSIs en el convertidor.

$$P_{\text{conm, Leapers, tot}} = 12 \cdot P_{\text{conm, Leapers}} \approx 1095 \text{ W}$$

$$P_{\text{conm, Wolfspeed, tot}} = 12 \cdot P_{\text{conm, Wolfspeed}} \approx 583 \text{ W}$$

Con tal de verificar estas pérdidas, se puede revisar la simulación realizada anteriormente, mostrada en la figura 4.54. En ella, se pueden apreciar unos 150 W de pérdidas de conducción para el semiconductor de Wolfspeed, lo cual se alinea con el cálculo analítico realizado, siendo este último un tanto más conservador (hay que tener en cuenta que la simulación usa solamente un VSI).

Resumen de pérdidas

El siguiente gráfico presenta una comparación detallada de las pérdidas de conducción y conmutación para los semiconductores de Leapers y Wolfspeed.

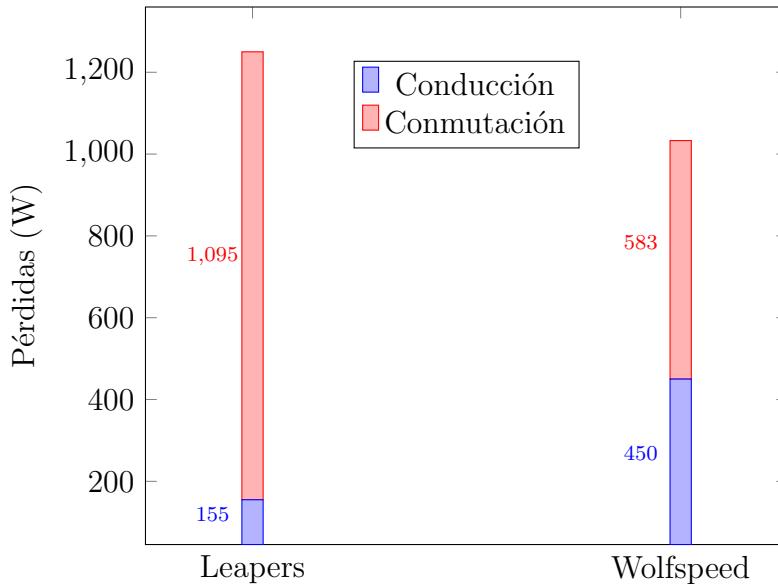


Figura 4.57: Comparación de pérdidas de conducción y conmutación

Fabricante	Pérdidas de conducción (W)	Pérdidas de conmutación (W)	Pérdidas totales (W)
Leapers	155	1095	1250
Wolfspeed	450	583	1033

Cuadro 4.4: Comparación de pérdidas entre semiconductores

Se observa que en el caso de Leapers, las pérdidas de conducción exhiben una significativa disminución en comparación con las de Wolfspeed, mientras que las pérdidas de conmutación muestran un incremento dramático para Leapers en relación con Wolfspeed. Este fenómeno sugiere que, en líneas generales, los dispositivos desarrollados por Leapers podrían ofrecer un camino de corriente más favorable en el semiconductor,

y los dispositivos de Wolfspeed son capaces de conmutar más rápido debido a que su carga Q_G es mucho menor, reduciendo así E_{on} y E_{off} .

La suma total de las pérdidas de ambos inversores, independientemente del semiconductor, es como máximo de 1300 W, valor con el cual se puede diseñar el sistema de refrigeración.

4.3.4. Sistema de refrigeración

A continuación se presenta una comparación entre las opciones comunes para refrigerar el inversor:

Opción de Refrigeración	Ventajas	Desventajas
Convección Natural	Económica, Sin partes móviles	Limitada disipación de calor, Depende del ambiente
Disipador de Calor	Económico, Fácil de instalar, No requiere energía adicional	Limitado en aplicaciones de alta potencia, Depende del ambiente
Ventilador	Buena disipación de calor, Control de temperatura	Limitado en aplicaciones de alta potencia, Ruido
<i>Coldplate</i> de Agua	Excelente capacidad de disipación, Buen control de temperatura	Coste, Mantenimiento, Posible riesgo de fugas, Instalación complicada
<i>Coldplate</i> de Aceite	Buena capacidad de disipación, No corrosivo, Alto rango de temperaturas	Coste, Mantenimiento, Posible riesgo de fugas, Instalación complicada

Cuadro 4.5: Comparación de Opciones de Refrigeración

Para el sistema de refrigeración de los semiconductores se ha optado por un sistema de refrigeración líquida consistente en una *coldplate*, una bomba y un radiador. Esta elección se basa en varios factores clave que se detallan a continuación:

- **Mantenimiento de la temperatura:** A diferencia de una refrigeración por aire, una *coldplate* permite mantener una interfaz de temperatura fija para los semiconductores. Esto es crucial para garantizar un funcionamiento estable y confiable de los componentes, especialmente en aplicaciones de alta potencia como la que se está diseñando.
- **Experiencia previa:** El equipo tiene experiencia previa en sistemas de refrigeración líquida, lo que facilita la implementación y mantenimiento de este tipo de sistemas.
- **Flexibilidad del packaging:** La refrigeración líquida, especialmente con agua, permite un diseño más compacto en comparación con otras opciones. Un disipador haría mucho más voluminoso el convertidor, mientras que la refrigeración líquida

permite una distribución más libre de los componentes por el monoplaza. Por ejemplo, se puede situar el radiador en una zona donde tenga mucho contacto con el aire exterior, y conectarlo mediante mangueras a la bomba y a la *coldplate*.

Aunque el carburo de silicio tiene una temperatura máxima de la unión ($T_{j(max)}$) notablemente alta, se ha establecido un objetivo de mantener la interfaz térmica del semiconductor a 80°C como máximo. Este enfoque busca garantizar una operación óptima y una vida útil prolongada de los semiconductores, además de mitigar los posibles efectos negativos derivados del calor.

Se ha elegido agua como el fluido de refrigeración por varias razones:

- **Norma T7.2.2:** Según la norma T7.2.2 [1], los sistemas de refrigeración solo pueden utilizar agua, aire u aceite como refrigerante. Con lo cual, no es posible utilizar líquidos refrigerantes específicos.
- **Propiedades térmicas:** El agua tiene una alta capacidad calorífica y conductividad térmica, lo que la hace eficaz para disipar el calor generado por los semiconductores. El aceite tiene peor conductividad térmica, y su amplio rango de temperaturas no es útil para la aplicación.
- **Seguridad:** El agua es un fluido seguro y no inflamable, lo que reduce los riesgos de seguridad en caso de fugas o accidentes. Además, es un fluido fácilmente disponible y de bajo costo en comparación con otros refrigerantes. Si se usa agua destilada, el riesgo de corrosión es más bajo, y los sistemas electrónicos no serán tan susceptibles a posibles fugas.

En el presente trabajo no se desarrolla más acerca el sistema de refrigeración puesto que la complejidad de diseñarlo excede el alcance del proyecto. Sin embargo, se reconoce su importancia, ya que al no disponer de este no se pueden realizar pruebas de potencia constante.

4.3.5. Gate drivers

Los gate drivers son componentes críticos en un convertidor de potencia, puesto que son los encargados de conectar las señales débiles de un microcontrolador con los semiconductores. Además, suelen llevar funcionalidades adicionales como la detección de cortocircuitos en el semiconductor, la lectura de señales analógicas de forma aislada, o mecanismos de protección avanzados.

Funcionamiento genérico

Los *gate drivers* desempeñan un papel esencial en la commutación eficiente de los MOSFETs, proporcionando los niveles de tensión y corriente adecuados para activar y desactivar rápidamente los transistores. En esencia, es una etapa transistorizada que "amplifica" la fuerza de la señal del microcontrolador, y a menudo lo hace de forma aislada.

Criterios de selección

Al seleccionar un *gate driver*, varios factores deben tenerse en cuenta para garantizar un rendimiento óptimo del sistema:

- **Topología:** A pesar de que hay soluciones que incluyen 6 *drivers* en un solo encapsulado, las más comunes son los circuitos integrados diseñados para controlar un solo dispositivo o dos en configuración de medio puente. Sin embargo, encontrar esta última opción para el rango de tensiones de 600 V con características óptimas es difícil. Por lo tanto, la topología más adecuada suele ser la de un solo *driver* por encapsulado.
- **Tensión de aislamiento:** Dado que las señales de puerta vienen del sistema de baja tensión, deben aislarse antes de llegar al *gate*, con lo que será necesario buscar un componente aislado a una tensión adecuada de al menos tres veces superior a la tensión máxima del inversor, 600 V.
- **Corriente de salida:** Es crucial elegir un *gate driver* que pueda suministrar la corriente necesaria para cargar y descargar rápidamente las capacidades de compuerta de los MOSFETs. La corriente necesaria depende de la resistencia de puerta, pero se puede determinar una primera cota de $\frac{V_{GS,typ}}{R_{G,int}} = \frac{20V}{2,4\Omega} = 8,3A$
- **Protecciones integradas:** Funciones como la lectura de señales analógicas, la posibilidad de paralelizar salidas o la inclusión de protecciones son muy beneficiosas para la integración del convertidor.

Comparativa de alternativas

A continuación se presenta una comparativa entre varios modelos de *gate drivers*, considerando sus características principales:

Características	ADUM4146	UCC21710	GD3160
Voltaje de Operación	-15 V hasta 30 V	-17.5 V hasta 36 V	-12 V hasta 25V
Corriente de Salida	4.61 A	10 A	15 A
Protecciones	Desaturación, UVLO	Sobrecorriente, Desaturación, <i>Mil</i> <i>ler Clamp</i>	Desaturación, Sobrecorriente, Apagado suave
Tiempo de Retardo	75 ns	90 ns	-
Aislamiento	5 kV	5.7 kV	8 kV
Extras	-	Sensor analógico aislado	Sensor analógico aislado
CMTI	100 V/ns	150 V/ns	100 V/ns

Cuadro 4.6: Comparación de *gate drivers*

Considerando los criterios de selección, se observa que el UCC21710 se presenta como una opción intermedia adecuada. En primer lugar, la corriente de salida que ofrece es de hasta 10 A, lo que lo hace adecuado para cargar y descargar rápidamente las

capacidades de compuerta de los MOSFETs, cumpliendo así con uno de los requisitos fundamentales. Además, integra protecciones contra sobrecorriente y desaturación, así como la funcionalidad de *Miller Clamp*. Además, la inclusión de un sensor analógico aislado proporciona la capacidad de monitorear la temperatura de los semiconductores con sus sensores integrados sin ningún componente adicional.

Aunque las protecciones de sobrecorriente y desaturación son elementos fundamentales para garantizar la seguridad y el correcto funcionamiento de un inversor, en este caso se ha optado por no implementarlas debido a las complicaciones que pueden surgir en el diseño del *layout*. Estas protecciones requieren la inclusión de componentes adicionales con *footprints* grandes y un enrutamiento más complejo de las señales, lo cual puede resultar en problemas de interferencia y aumento de la complejidad del circuito.

Es importante destacar que este inversor se encuentra en una fase inicial de prototipado y desarrollo, donde se prioriza la funcionalidad básica y la viabilidad del diseño. En futuras revisiones y etapas de desarrollo, se considerará la inclusión de estas protecciones, ya que son cruciales para proteger tanto el inversor como los dispositivos conectados. La decisión actual de omitir estas protecciones se basa en la necesidad de simplificar el diseño y garantizar una implementación más eficiente y manejable en esta fase del proyecto.

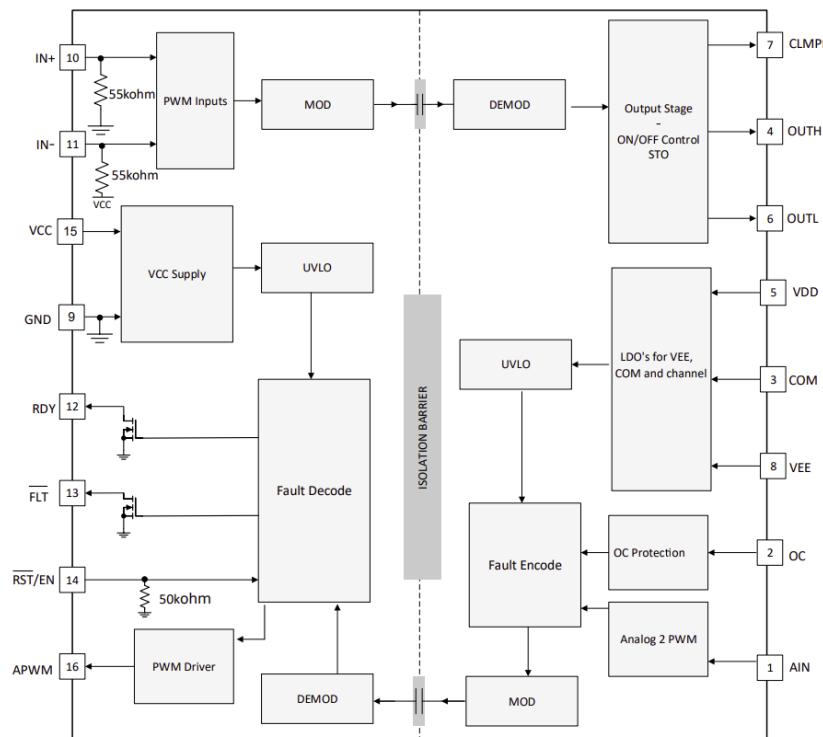


Figura 4.58: Diagrama de bloques funcional del *gate driver* seleccionado.

Cálculos del *gate driver* seleccionado

- **Tensión gate-source:** Uno de los valores más importantes a determinar es el nivel de tensión de puerta para encender y apagar el MOSFET. En una aplicación

hard-switched como la del convertidor en diseño, es beneficioso usar una tensión de encendido tan alta como sea posible.

Dada la naturaleza rápida del carburo de silicio, se prevé una dV/dt considerable, la cual puede causar varios problemas. Uno de ellos sería la activación accidental de un MOSFET, la cual causaría la condición de *shoot-through*. Una estrategia para mitigar parcialmente este problema es el uso de un *gate driver* con *Miller Clamp*. Además, se suele apagar el dispositivo utilizando una tensión negativa, que amortigua posibles interferencias en la puerta del dispositivo afectado. El valor de tensión negativo al cual se encuentre la puerta aumenta la tensión que debe inducirse en ese nodo para llegar a la tensión umbral $V_{G,threshold}$ del MOSFET. Dado que el valor óptimo no es trivial, conviene implementar alguna manera de modificarlo, por ejemplo, mediante el uso de un regulador lineal.

Los dispositivos seleccionados tienen un rango de tensiones V_{GS} un tanto distintos, siendo el de Leapers de +21 V a -2 V, y el de Wolfspeed de +15 V a -4 V.

- **Potencia de alimentación del *gate driver*:** Se estima en función de la frecuencia de conmutación y la carga de la compuerta de los MOSFETs, además de la tensión V_{GS} .

$$I_{supply,min} = f_{conm} \cdot Q_G \quad (4.64)$$

Donde

- $I_{supply,min}$ es al corriente mínima que debe poder suministrar la fuente de alimentación del *gate driver*,
- f_{conm} es la frecuencia de conmutación, y
- Q_G es la carga en la puerta.

Utilizando el valor más alto de Q_G entre los dos semiconductores,

$$I_{supply,min} = f_{conm} \cdot Q_G = 40 \text{ kHz} \cdot 520 \text{ nC} = 20,8 \text{ mA}$$

se puede calcular la potencia necesaria de la fuente.

$$P_{supply,min} = \Delta V_{GS} \cdot I_{supply,min} \quad (4.65)$$

Donde:

- $P_{supply,min}$ es la potencia mínima que debe poder suministrar la fuente de alimentación del *gate driver* y
- ΔV_{GS} es la diferencia entre la tensión de puerta de encendido y apagado.

$$P_{supply,min} = \Delta V_{GS} \cdot I_{supply,min} = 20 \text{ V} \cdot 20,8 \text{ mA} = 0,416 \text{ W}$$

Con esta estimación de potencia mínima requerida, es esencial seleccionar una fuente adecuada para los *gate drivers*. En este contexto, se considera la serie MGJ2 de Murata, que ofrece características como:

- Tensión de entrada de 5 V, 12 V, 15 V o 24 V, lo que permite adaptarse a diferentes fuentes de alimentación disponibles. En esta aplicación, la entrada de 5 V simplifica mucho la arquitectura de *hardware*.
- Rango de tensiones de salida muy variados, que permite intercambiar fuentes para probar distintos niveles de tensión.
- Aislamiento reforzado hasta 5.2 kV,DC, garantizando la seguridad y el cumplimiento de la normativa.
- Caracterización de CMTI mayor a 200 V/ns, lo que garantiza una respuesta rápida y efectiva ante transitorios de alta velocidad.
- **Valores de Resistencias de Puerta ($R_{G_{on/off}}$):** Reducir el valor de las resistencias de puerta conlleva la disminución de las pérdidas de conmutación, ya que los MOSFETs cambiarán más rápido de estado y, por lo tanto, pasarán menos tiempo en la etapa de conmutación. Esta rapidez en el cambio implica un mayor dV/dt, lo que puede ser responsable del aumento de la interferencia electromagnética (EMI) y de sobrepico de tensión entre el *drain* y el *source* del propio MOSFET. El valor de estas resistencias se estimará mejor con pruebas empíricas, pero se parte del valor más alto considerado de $10\ \Omega$.

4.3.6. Bus de condensadores

La tarea del bus de condensadores es equilibrar la potencia instantánea fluctuante en los nodos positivo y negativo de la batería, resultante de los semiconductores que conmutan esta tensión continua en la carga. Estos condensadores, a menudo referidos como bus o enlace de continua, mitigan el rizado creado por la conmutación de alta frecuencia, lo que garantiza un suministro de tensión más estable para el sistema eléctrico en su conjunto.

Función del bus de condensadores

En un inversor de fuente de voltaje (VSI), el bus de condensadores tiene dos funciones principales:

- **Minimizar el rizado de tensión:** El rizado de tensión en el bus de continua está causado por la conmutación de los dispositivos semiconductores. El bus de condensadores ayuda a reducir este rizado de tensión, lo que es crucial para evitar fluctuaciones en la tensión máxima que se puede consignar al motor.

El rizado de tensión se calcula como:

$$V_{\text{ripple}} = \frac{I_{C,\text{RMS}}}{C \cdot f_{\text{conm}}} \quad (4.66)$$

Donde:

- V_{ripple} es el rizado de tensión producido,
- $I_{C,\text{RMS}}$ es la corriente alterna del bus de condensadores,

- C es la capacitancia del bus de condensadores, y
- f_{conm} es la frecuencia de conmutación.

- **Proporcionar la potencia reactiva:** Además de la reducción del rizado de tensión, los condensadores en el bus también proporcionan potencia reactiva necesaria para compensar la carga inductiva del motor.

Dimensionamiento del bus de condensadores

Para dimensionar adecuadamente el bus de condensadores en un inversor de fuente de voltaje, se deben considerar las siguientes condiciones:

- $V_C > 1,1 \cdot V_{\text{max}}$

Donde:

- V_C es el voltaje del bus de condensadores,
- V_{max} es la tensión máxima DC.

- $I_{C,\text{RMS}} \approx 0,65 \cdot I_{\text{fase,RMS}}$

Donde:

- $I_{C,\text{RMS}}$ es la corriente efectiva del bus de condensadores,
- $I_{\text{fase,RMS}}$ es la corriente efectiva de fase,
- El factor 0.65 se utiliza para estimar la corriente RMS máxima del bus de condensadores en el peor caso, que con una modulación SVPWM se da cuando el índice de modulación es de aproximadamente 60 % [2].

- $C > \frac{I_{C,\text{RMS}}}{V_{\text{ripple}} \cdot f_{\text{conm}}}$

Donde:

- C es la capacitancia del bus de condensadores,
- V_{ripple} es el rizado de tensión permitido,
- f_{conm} es la frecuencia de conmutación.

Es fácil ver como reducir la frecuencia de conmutación aumentará proporcionalmente el rizado de tensión para la misma corriente de salida.

Ahora, aplicando estas consideraciones a los valores del diseño:

- $V_C > 1,1 \cdot V_{\text{max}} = 1,1 \cdot 600 \text{ V} = 660 \text{ V} \rightarrow 850 \text{ V}$

Escoger condensadores con un límite de tensión más grande es más seguro para transitorios y el régimen de trabajo en debilitamiento de campo.

- $I_{C,\text{RMS}} \approx 0,65 \cdot I_{\text{fase,RMS}} = 0,65 \cdot 80 \text{ A,RMS} = 52 \text{ A,RMS}$

Se debe asegurar que el condensador puede soportar esta corriente, idealmente sin refrigeración. En caso de usar múltiples condensadores para formar el bus, la corriente se reparte de forma proporcional entre cada uno de ellos.

$$\blacksquare C > \frac{I_{C,\text{RMS}}}{V_{\text{ripple}} \cdot f_{\text{comm}}} = \frac{52 \text{ A,RMS}}{15 \text{ V} \cdot 40 \text{ kHz}} \approx 87 \mu\text{F} \rightarrow 100 \mu\text{F}$$

Conviene sobredimensionar la capacitancia tanto como sea posible atendiendo a límites mecánicos y de ensamblaje. Para el valor de V_{ripple} se suele considerar un 5 % de la tensión mínima de funcionamiento, que en este caso es de entorno a los 350 V ($\frac{15\text{V}}{350\text{V}} = 4,3\%$).

Selección de condensadores

Se ha optado por un diseño con 10 condensadores del modelo FHA85Y106KS de la serie FH de Murata. A continuación se detallan las razones para esta elección:

- **Material:** La serie FH de Murata tiene condensadores de película, cuya resistencia interna es muy inferior a la de un condensador electrolítico, lo cual es crucial para minimizar las pérdidas en el condensador.
- **Densidad de capacitancia:** Esta serie de condensadores presenta una alta densidad de capacitancia de entorno a $0,6 \frac{\mu\text{F}}{\text{cm}^3}$. Esto va a permitir compactificar el diseño bastante.
- **Tensión máxima:** Con una gama de 500 V y otra 850 V, estos últimos están dentro del rango necesario para operar de manera segura, incluso en debilitamiento de campo, donde en caso de error, la tensión de bus podría ser superior a 600 V.
- **Temperatura de Operación:** Tienen un rango de temperatura de operación amplio, que va desde -40°C hasta +125°C. Dado que se prefiere evitar su refrigeración, el hecho de que aguanten altas temperaturas permite tener más confiabilidad en la corriente que pueden soportar y el calentamiento que esta va a provocar.
- **Frecuencia de Operación:** Los condensadores FHA85Y106KS están diseñados para operar en frecuencias de hasta 200 kHz, cuatro veces superior a la frecuencia de comutación escogida.



Figura 4.59: Apariencia del condensador seleccionado.

Se evalúa el comportamiento térmico utilizando la hoja de datos:

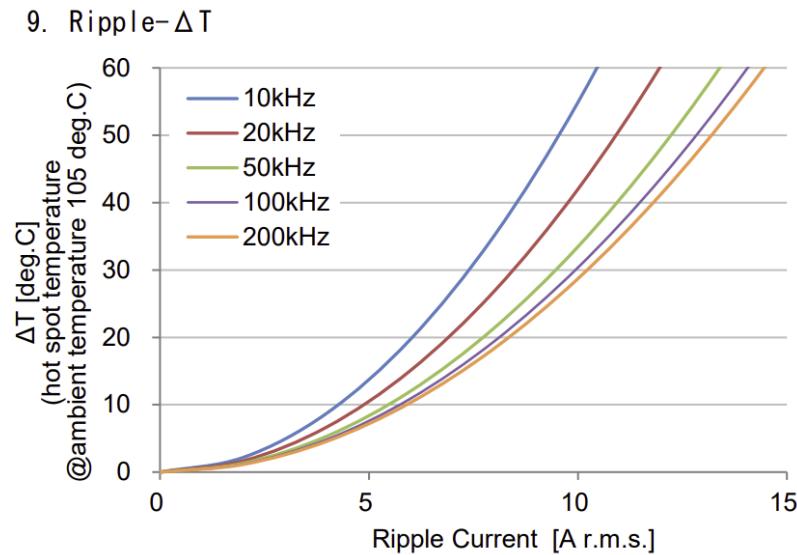


Figura 4.60: Calentamiento propio del condensador en función de la corriente.

Dado que hay 10 condensadores, la corriente máxima de cada uno es de $\frac{52 \text{ A,RMS}}{10} = 5,2 \text{ A,RMS}$. Por tanto, el incremento de temperatura es de 10°C como máximo. No será necesaria ninguna consideración térmica ni de refrigeración para estos condensadores.

4.3.7. Conectores de potencia

Para asegurar una conexión segura y eficiente del bus de continua con la batería y de los semiconductores con las fases de los motores, se requiere una cuidadosa selección de conectores. Dada la alta corriente que deben soportar, las conexiones soldadas sin soportes adicionales no son adecuadas según la normativa. Por ello, se opta por la tecnología de conectores *press-fit*.

Un modelo que destaca en esta aplicación es el 7461063 de Würth Elektronik, diseñado para conexiones de hasta 160 A. Está fabricado en latón con una superficie estañada, y su montaje *press-fit* asegura una conexión mecánica y eléctrica sólida.

La tecnología *press-fit* utilizada en estos conectores REDCUBE de Würth Elektronik proporciona una serie de ventajas significativas. Al presionar los pines en la PCB, se genera una soldadura fría homogénea entre el pin y el orificio metalizado, asegurando una resistencia de contacto inferior a $200 \mu\text{Ohm}$.

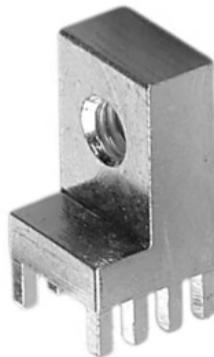


Figura 4.61: Conector REDCUBE PRESS-FIT - Modelo 7461063.

Estos conectores permiten una flexibilidad de integración muy grande, puesto que la conexión a ellos se realiza mediante una unión roscada, un tornillo, lo cual permite conectar cables con terminales de anilla, pletinas y otros elementos conductores. Gracias a esto es muy sencillo implementar mecanismos de bloqueo positivo para evitar la desconexión por vibraciones.

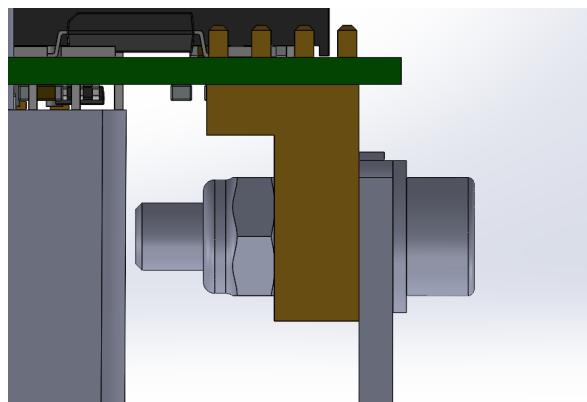


Figura 4.62: Ejemplo de conexión con bloqueo positivo utilizando una tuerca DIN 980V.

4.3.8. Sensor de posición

4.3.9. Microcontrolador

La unidad de procesamiento del convertidor es una de las piezas clave, pues será la encargada de gestionar lecturas analógicas, alarmas, comunicaciones, y por supuesto, ejecutar el control. Por ello su selección debe ser meticulosamente estudiada. En primer lugar se seleccionan unos requisitos:

- **Velocidad de reloj mayor a 200 MHz:** En base a la experiencia del autor con otros inversores, el cálculo de todos los PIs, filtros, transformadas y trayectorias de control puede tomar hasta 2500 ciclos de reloj. Por lo tanto, dado que hay que ejecutar dos lazos de control para los dos motores, $\frac{2000 \cdot 2}{200 \text{ MHz}} = 25 \mu\text{s} \leftrightarrow 40 \text{ kHz}$, 200 MHz es la mínima velocidad para poder conmutar a 40 kHz.
- **Unidad de punto flotante:** Para evitar trabajar en coma fija, lo cual puede causar muchos problemas de integración y añade una capa de complejidad innecesaria, se pone como requisito que el MCU soporte el uso de *float* de forma nativa.
- **Dos timers avanzados:** Son necesarios dos *timers*, uno para cada inversor, con características como la salida PWM bipolar con varios canales, configuración sencilla del tiempo muerto, etc.
- **Lecturas analógicas:** Se necesitan al menos un par de ADCs de 12 bits suficientemente rápidos con múltiples canales cada uno, con tal de poder capturar las corrientes de fase y las tensiones de bus adecuadamente.
- **Interfaces de comunicación CAN, SPI, I2C y USB:** Dado que se requerirá la interacción con otros dispositivos, es esencial contar con varias interfaces de comunicación. Obviamente CAN para la comunicación con el vehículo, pero también SPI/I2C para utilizar memorias externas, o UART/USB como manera de conectar el inversor de forma sencilla a un ordenador.
- **Capacidades de depuración avanzadas:** Se requerirá de herramientas de depuración que permitan detener el código en puntos específicos, monitorizar en tiempo real el estado de las variables, etc.
- **Memoria flash superior a 500 kB:** El binario de un programa muy extenso no debería pesar mucho más de 100 kB, sin embargo, si en el futuro se programa un *bootloader* u otras implementaciones como tablas de búsqueda grandes, la memoria *flash* debería ser el último de los problemas.

Teniendo en cuenta estos requisitos, se ha evaluado una variedad de familias y marcas de MCUs y DSPs, y se ha llegado a la conclusión de que el MCU STM32F777VI de la familia F7 de STMicroelectronics cumple con todos los criterios establecidos. Sus características incluyen:

- Procesador ARM Cortex-M7 con FPU, funcionando a una velocidad de hasta 216 MHz.
- Amplio conjunto de interfaces de comunicación, incluyendo I2C, USART, SPI, CAN, USB y Ethernet.
- 3 ADCs (24 canales máximo) de 12 bits y hasta 2.4 MSPS.
- 18 *timers*, incluyendo *timers* de 16 y 32 bits, dos *timers* con PWM avanzado.
- Soporte para depuración mediante interfaces SWD y JTAG, incluyendo el modo de *Trace Asynchronous Serial Wire*, que incorpora un pin de comunicación serie independiente de las capacidades de *debug* de SWD.

- Hasta 2 MB de memoria *flash*, SRAM de 512 Kbytes y 16 Kbytes de RAM TCM para datos críticos en tiempo real.

El STM32F777VI cumple con todos los requisitos establecidos y proporciona capacidades adicionales que pueden ser beneficiosas para el desarrollo y la operación del convertidor. Además, es el mismo microcontrolador que se usa en el resto de ECUs del monoplaza, convirtiéndose en la elección más cómoda para el futuro desarrollo.

Software de CAD electrónico

El diseño de las PCBs se realizó utilizando el *software* Altium Designer, una herramienta ampliamente utilizada en la industria para diseños electrónicos. Altium Designer ofrece una amplia gama de funcionalidades que facilitan el proceso de diseño, desde la creación de esquemáticos hasta la disposición de componentes y el enrutado de pistas.

- **Creación de esquemáticos:** Altium Designer proporciona un entorno intuitivo para crear el esquemático del circuito. Primero se crean los componentes usando la información de la hoja de datos para hacer un buen *footprint* y símbolo, y después se incluyen los símbolos de los componentes en el esquemático y se realizan las conexiones eléctricas.
- **Diseño de PCB:** Una vez completado el esquemático, Altium Designer facilita la transferencia de diseño a la PCB. Los componentes del esquemático quedan vinculados y de forma muy sencilla se importan los cambios del esquemático a la PCB. Posteriormente se realizan las conexiones eléctricas en la PCB en un proceso conocido como enrutado.
- **Verificación de reglas:** Altium Designer permite definir y verificar reglas de diseño para garantizar que el PCB cumpla con las restricciones de los fabricantes. Esto incluye reglas de espaciado, de colisiones, etc.
- **Generación de archivos de fabricación:** Una vez completado el diseño de la PCB, Altium Designer facilita la generación de archivos necesarios para la fabricación, como los archivos *Gerber*, la lista de materiales (BOM) o el archivo de taladros. Además, permite exportar un modelo 3D de la PCB con todos los componentes, lo cual facilita mucho la integración mecánica.

4.3.10. PCB de potencia

Esta PCB es una placa de 4 capas que integra componentes de potencia y circuitos de adquisición para el control de un solo motor eléctrico. Se requieren de dos de estas PCBs para montar el convertidor al completo.

Concepto y *layout*

La PCB de potencia es el sustrato físico sobre el cual se montan y conectan los componentes de potencia. En ella están situados los semiconductores, el bus de

condensadores, las medidas de tensión, corriente y temperatura, los *gate drivers*, y algunos circuitos de seguridad. A continuación se muestran unos bocetos iniciales que se usaron para tantear el emplazamiento de los componentes y conexiones de potencia:

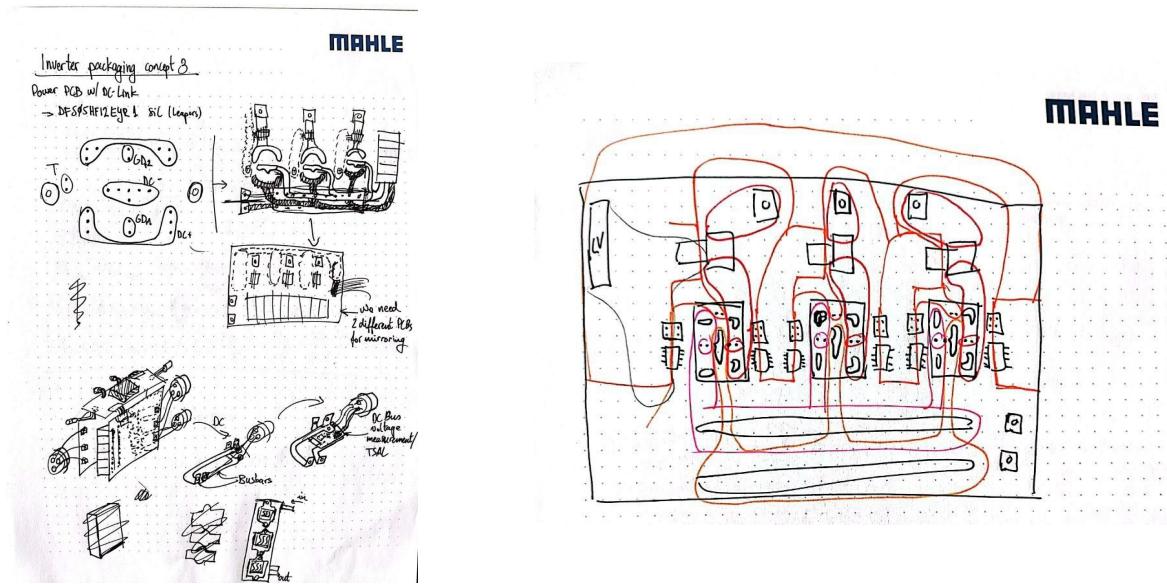


Figura 4.63: Bocetos del *layout* del inversor.

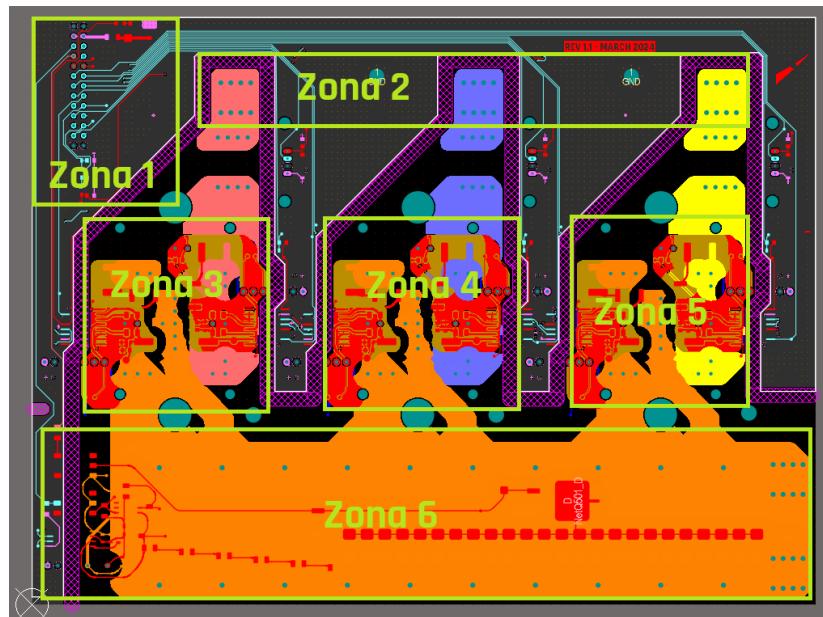


Figura 4.64: *Layout* de la PCB de potencia, con las zonas más importantes marcadas.

En el *layout* se distinguen varias zonas.

- **Zona 1:** Conector a la PCB de control y protección de la alimentación de baja tensión.
- **Zona 2:** Conectores de las fases del motor y montajes mecánicos.

- **Zonas 3, 4 y 5:** Semiconductores, sensores de corriente, *gate drivers* y *snubbers* para cada fase.
- **Zona 6:** Bus de condensadores, conectores de alta tensión DC, circuito de descarga y sensado de tensión.

Restricciones y enrutado

Para dotar al convertidor de una alta densidad de potencia se decidió un formato de 150 mm por 200 mm, dejando espacio más que suficiente para distribuir todos los componentes de forma más o menos holgada.

Dado que esta PCB implementa partes del circuito del sistema de tracción y partes del sistema de baja tensión, se debe crear una separación física de 4 mm ya que estará acabada con un revestimiento conformal acrílico (*conformal coating*), acorde con la norma EV 4.3.6.

Los espaciados entre conductores de la parte de alta tensión son de 3 mm, ya que la tensión que deben aislar es de 600 V. Este valor se obtiene de la norma genérica IPC-2221B, que especifica una separación de $2,5 + (V - 500) \cdot 0,005$ mm = 3 mm. De todas maneras, se usará un revestimiento conformal acrílico para asegurar una mejor característica dieléctrica y prevenir la aparición de arcos eléctricos.

El dimensionamiento de las pistas para la corriente que debe pasar es algo complicado, y de hecho condiciona el apilado de la PCB. Por ello, se selecciona un apilado de 4 capas de 70 micras (2 onzas por pulgada cuadrada).

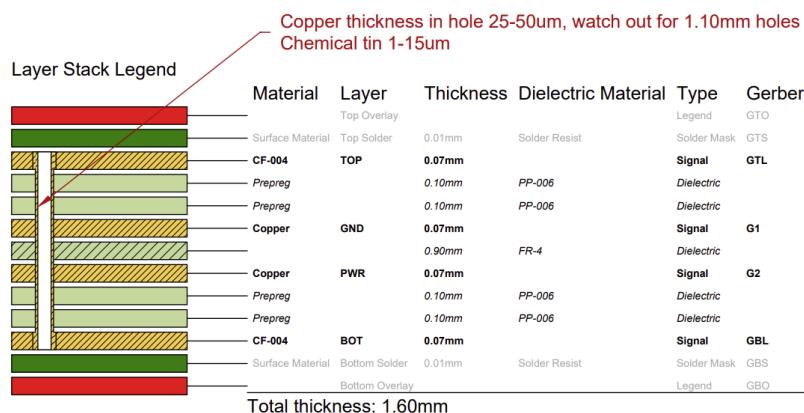


Figura 4.65: Apilado de la PCB de potencia.

Para dimensionar el ancho de estas pistas se ha considerado la corriente RMS de fase, ya que es la corriente máxima en toda la PCB. Después, se ha usado este ancho para todas las conexiones de potencia, aunque la corriente que vean sea menor. El cálculo se ha realizado usando la norma IPC-2152, que especifica unas relaciones de capacidad de corriente respecto a incremento de temperatura y grosor de capa. Ya que el cálculo es algo tedioso, se opta por el uso de una calculadora específica para este propósito:

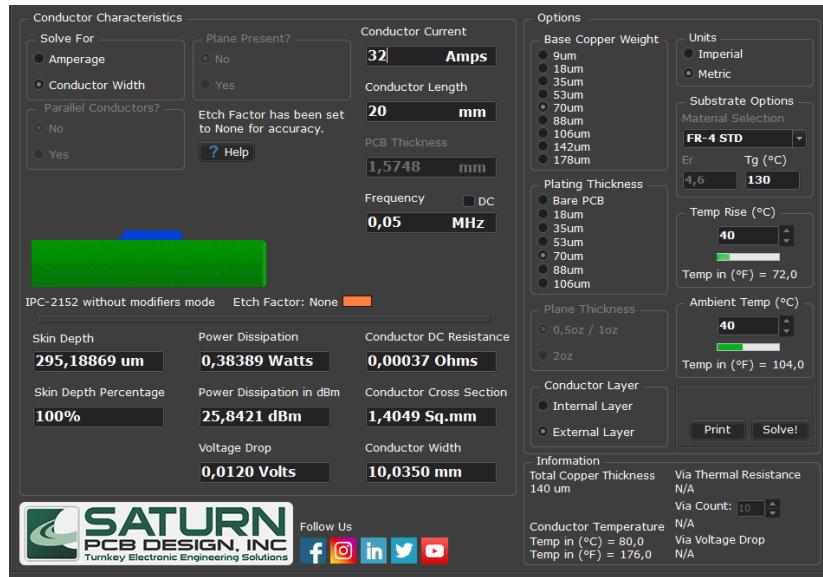
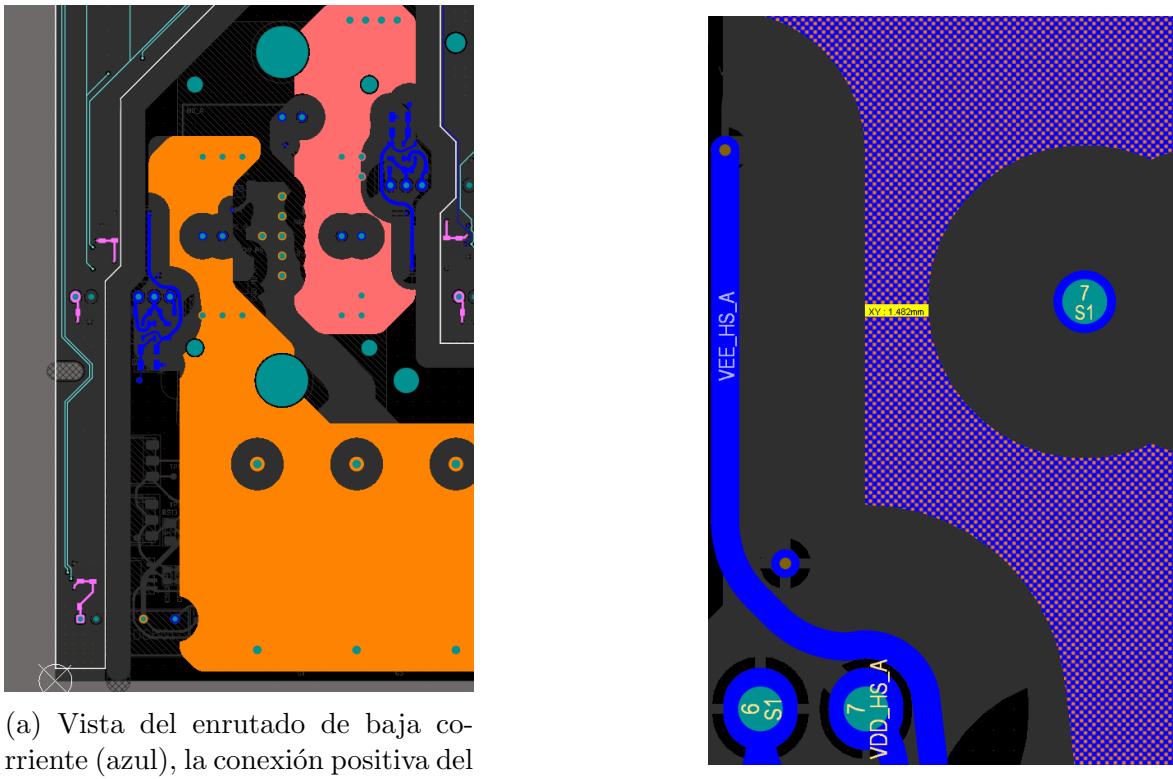


Figura 4.66: Cálculo del ancho de pista necesario.

El resultado es de 10 mm, lo cual es exagerado, pero sin embargo, solamente se está teniendo en cuenta una capa. Por ello, se decide enrutar las conexiones de potencia por un mínimo de 2 capas con un ancho mínimo de 5 mm. A continuación se detalla el enrutado de la conexión DC positiva de uno de los 3 módulos de medio puente:



(a) Vista del enrutado de baja corriente (azul), la conexión positiva del bus de continua (naranja) y de la fase A (rosado).

(b) Detalle de una sección más estrecha.

Figura 4.67: Ejemplo de enrutado de la conexión DC positiva del medio puente de la fase A.

Como se puede apreciar, el espaciado que hay que dejar por aislamiento compite por espacio con el ancho de las conexiones. En la mayoría de casos se ha podido respetar el ancho mínimo de 5 mm, sin embargo, en algunas situaciones esto no es posible. Por ejemplo, lo ilustrado en la subfigura (b) muestra una estrechez de 1.5 mm, pero como esa conexión está repetida en dos capas, el ancho total es de 3 mm. Además, la conexión positiva en el semiconductor se realiza a través de dos grupos de pines, y como antes de realizar esa estrechez ya se ha conectado un grupo de pines, se puede asumir que la corriente que circula por ese trozo de cobre es la mitad que la corriente continua entera que va a recibir ese módulo, que a su vez, es aproximadamente un tercio de la corriente continua total ($\frac{1}{3} \frac{17.5 \text{ kW}}{450 \text{ V}} = 13 \text{ A}$), pasado por la calculadora, tan solo son necesarios 2.8 mm de cobre para esa corriente, lejos de los 10 mm que se han intentado mantener.

Esto es tan solo un ejemplo del análisis que se ha realizado con cualquier mínima sospecha de falta de cobre en todo el enrutado de potencia. Un camino de corriente muy estrecho podría presentar sobrecalentamiento y en casos extremos, levantamiento de la pista o plano.

Además, se ha especificado el acabado y el espesor de cobre en las vías para asegurar un buen montaje de los componentes *press-fit* (los conectores de potencia y los semiconductores). Se ha optado por un acabado ENIG (oro electrolítico) por ser una opción con la que se pueden hacer este tipo de conexiones, además de aguantar mucho tiempo sin oxidarse. El ancho de cobre en las vías es algo que se debe consultar directamente con el fabricante porque no todos son capaces de asegurar una tolerancia. Para saber cuánto cobre es necesario, se han consultado los documentos explicativos de Würth Elektronik (para los conectores de potencia) y de Wolfspeed (para los semiconductores).

	Minimum	Typical	Maximum
Hole drill diameter	1.12 mm	1.15 mm	
Copper thickness in hole	25 μm		50 μm
Metalization in hole			15 μm
End hole diameter	1.00 mm	1.05 mm	1.18 mm
Copper thickness of conductors	35 μm	70 μm – 105 μm	400 μm
Metalization of circuit board		Tin (chemical) recommended	
Metalization of pin		Tin (galvanic)	

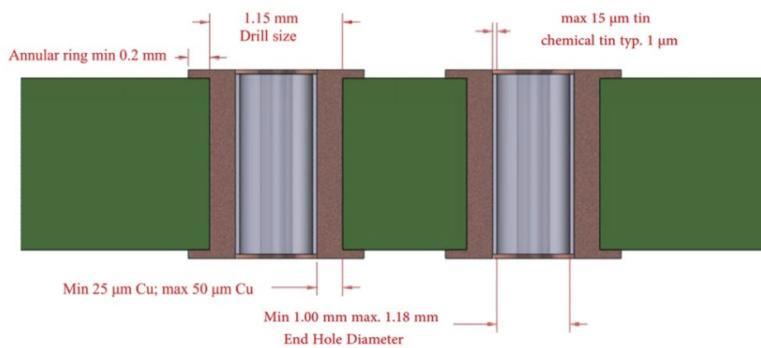


Figura 4.68: Requisitos para la PCB según Wolfspeed.

Materials and tolerances

REDCUBE PRESS-FIT from Würth Elektronik are manufactured from the material CuZn39Pb3 and are therefore RoHS-compliant according to the RoHS stipulation concerning copper alloys.

The circuit board thickness should ideally be between 1.6 and 3.2 mm. Tested surfaces are chemical tin, HAL and ENIG. The **immersion tin** coating process is recommended. Using this process usually guarantees that the tin is evenly distributed in the case whereby the tolerances can be complied with more easily and thus chip formation can be prevented. Due to the uneven distribution of the tin in the case for the HAL process, we recommend the immersion tin process for circuit board thickness of 2.4 mm and greater. ENIG can be used but not recommended for Press-Fit technology.

Unless otherwise noted in the corresponding drawing, Würth Elektronik **REDCUBE** PRESS-FIT have quadratically designed press-fit pins. The through-hole plating in the PCB must therefore have the following characteristics:

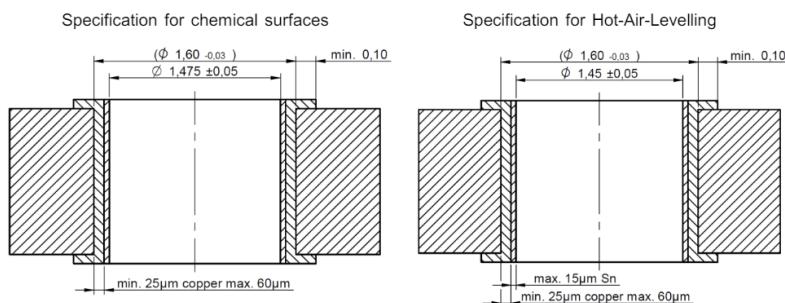


Figura 4.69: Requisitos para la PCB según Würth Elektronik.

Ambos documentos apuntan que un acabado de estaño químico es lo ideal para realizar esta conexión, sin embargo, no se pudo conseguir un fabricante que lo pudiera realizar a un precio razonable. El grosor de cobre que se marcó como requisito al fabricante fue de 25 µm a 50 µm, por ser el más restrictivo de ambos.

Bloques funcionales

La PCB de potencia se divide en varios bloques funcionales, que se juntan en un esquemático jerárquico, que además incluye la conexión entre estos bloques, algunas notas indicativas y componentes que no pertenecen a ningún bloque en específico:

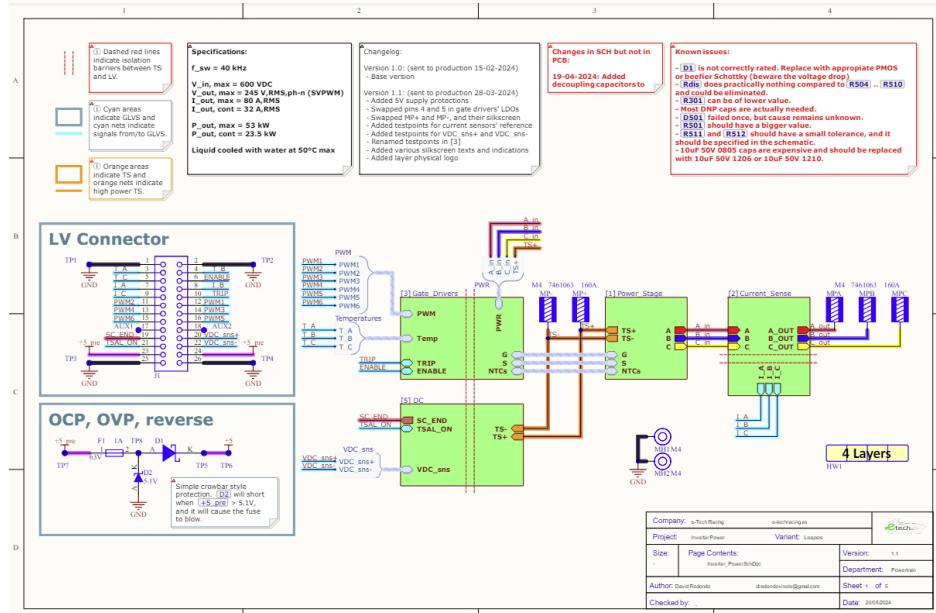


Figura 4.70: Esquemático jerárquico de la PCB de potencia.

En el esquemático se pueden distinguir el conector a la PCB de control, la protección de alimentación de baja tensión y unas notas indicativas, así como una leyenda de colores. Se anotan también los cambios y la fecha en la que se envía a producción. Los bloques que se pueden ver incluyen:

- Etapa de Potencia:** En este bloque aparecen los semiconductores, el bus de condensadores y una descarga pasiva.
- Sensado de corriente:** Se incluyen los sensores de corriente y su interfaz analógica con la PCB de control.
- Gate Drivers:** En este bloque se recogen las señales necesarias para la commutación, así como la configuración de los *gate drivers* en su conjunto. Este bloque contiene 3 subcircuitos correspondientes con cada fase. En ellos se encuentran los *gate drivers* en sí, junto a su alimentación aislada, los *snubbers* y la medida de temperatura.
- DC:** Por último, hay un bloque dedicado al bus de continua, que incluye el circuito de descarga, la medida de tensión y la detección de 60 V requerida por la normativa.

Circuitos importantes

VSI Ya que los módulos son estructuras *half-bridge*, se deben conectar entre sí para formar la topología VSI.

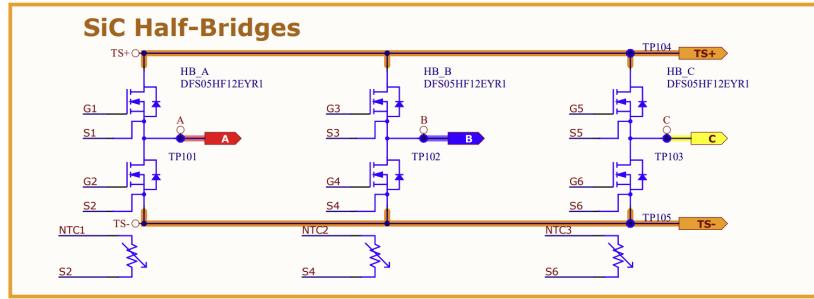


Figura 4.71: Conexión de los módulos configurando un VSI.

Protección de alimentación Aunque en principio es regulada, la entrada de 5 V necesita de un mínimo de protecciones sencillas para evitar retrasos ocasionados por fallos mientras se realizan pruebas. En este caso se ha usado un fusible como protección contra sobrecorriente, un diodo *zener* a modo de protección contra sobretensión (hará saltar el fusible cuando la tensión de entrada supere la tensión umbral), y un diodo *schottky* como protección frente a tensión inversa, ya que tiene poca caída de tensión, actúa bastante rápido y es barato.

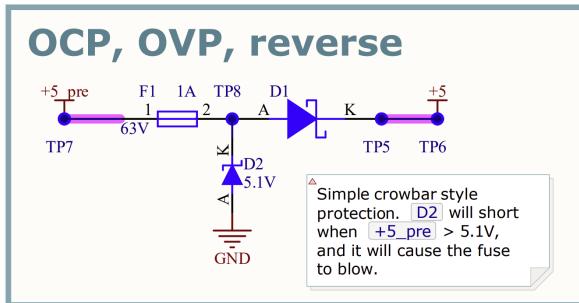
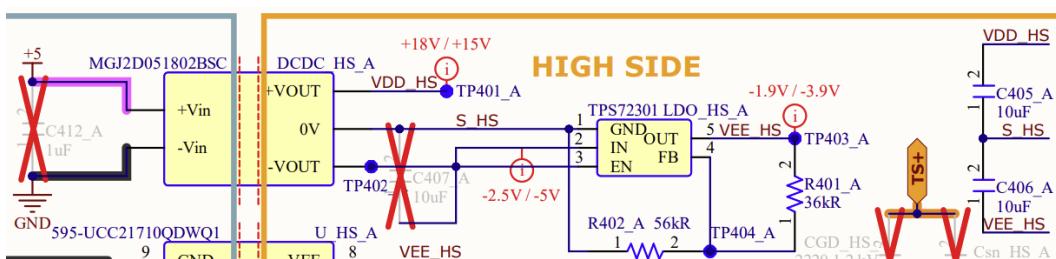


Figura 4.72: Circuito de protección de alimentación.

Alimentación del *gate driver*

Figura 4.73: Alimentación bipolar aislada del *gate driver*.

En función del modelo de DC-DC específico escogido de la serie MGJ2 de Murata, se tendrá una tensión positiva y negativa diferente. La tensión positiva no es muy problemática puesto que se pueden seleccionar DC-DCs con salidas de +18 V o de +15 V, valores que encajan con los recomendados por los fabricantes de los semiconductores.

Sin embargo, ese no es el caso para las tensiones negativas, ya que el valor debería poder ajustarse para ser más versátil en el desarrollo. Por ello, se implementan

reguladores lineales para ajustar $V_{EE,HS}$ y $V_{EE,LS}$ durante las pruebas para afinar el voltaje negativo necesario.

V_{EE} se calcula

$$V_{EE,LS} = -1,186 \cdot \left(1 + \frac{R1}{R2}\right).$$

Donde $R1+R2 \approx 100 \text{ k}\Omega$ según la hoja de datos del LDO. Para las configuraciones específicas de Leapers y Wolfspeed, los valores de las resistencias son

- Leapers: $R1 = 36 \text{ k}\Omega$, $R2 = 56 \text{ k}\Omega$
 - Wolfspeed: $R1 = 68 \text{ k}\Omega$, $R2 = 30 \text{ k}\Omega$

Este proceso permite ajustar de forma precisa los voltajes negativos necesarios para el correcto funcionamiento del circuito de conmutación.

Gate driver

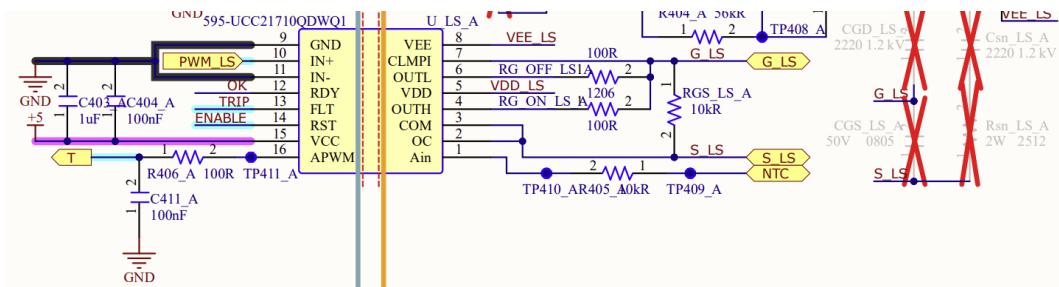


Figura 4.74: *Gate driver* con medida de temperatura y *snubbers*.

Uno de los componentes más importantes de todo el diseño tiene una de las implementaciones más sencillas, ya que la hoja de datos del UCC21710 proporciona mucha información útil para la aplicación final.

El *gate driver* UCC21710 se implementa con varias características que lo hacen adecuado para su uso en aplicaciones de potencia. Las señales TRIP y OK están configuradas en *open drain*, lo que permite que se paralelicen con los otros *gate drivers* para facilitar su integración. La entrada IN- no se utiliza y está conectada a tierra. El pin ENABLE se controla desde el MCU, y cuando se fuerza un estado bajo durante más de 1 μ s, se resetea la señal TRIP. La señal TRIP se lee como una interrupción externa desde el MCU, y sirve para detectar un error en los *gate drivers* y actuar rápidamente.

Para la medición de temperatura, se utilizan los *drivers* de los MOSFETS del *low-side*. Estos proporcionan una corriente de salida de $200 \mu\text{A}$ en su salida, a través de la cual se puede interpretar la medida. Esta señal se convierte a PWM y luego a analógica mediante un filtro RC para ser recibida directamente por el ADC del MCU.

El circuito de medición de temperatura utiliza el sensor NTC del semiconductor para obtener una lectura de tensión, la cual se convierte en una lectura de bits mediante el ADC del MCU. La relación entre la resistencia del NTC y la temperatura se modela utilizando la fórmula de Steinhart-Hart, que toma en cuenta el valor de Beta del NTC,

la resistencia a una temperatura de referencia, y la temperatura ambiente. La fórmula utilizada es:

$$NTC = R_0 \cdot e^{[-\beta \cdot (\frac{1}{T_0} - \frac{1}{T})]}.$$

El voltaje leído por el ADC se calcula

$$V_{\text{read}} = V_{CC,\text{GD}} \cdot \left(\frac{-20 \cdot I_{AIN} \cdot (R_{\text{filt}} + NTC)}{100} \right) = 5 \text{ V} \cdot \left(\frac{-20 \cdot 200 \mu\text{A} \cdot (10 \text{ k}\Omega + NTC)}{100} \right).$$

El circuito también incluye protección de *Miller clamp* para evitar activaciones accidentales de los MOSFETs. Aunque los *gate drivers* tienen un *pull-down* activo, se ha implementado un *pull-down* externo con $R_{GS,HS}$ y $R_{GS,LS}$. Sin embargo, la detección de sobrecorriente no está implementada en este diseño por los motivos mencionados en un apartado anterior.

Adicionalmente, se incluyen unos *snubbers* consistentes en un RC entre drenador y fuente, un condensador entre puerta y drenador y un condensador entre puerta y fuente. En caso de encontrar problemas excesivos con la conmutación, pueden ayudar a amortiguar transitorios.

Cabe destacar que el enrutado de este subcircuito es especialmente crítico por los parásitos inductivos que se pueden ocasionar. En particular, la inductancia en el camino de retorno de puerta-fuente puede ser fatal para el comportamiento de la conmutación, razón por la cual se ha tirado un plano conectado a la fuente de cada MOSFET en el área donde se enruta su *driver*, para minimizar esa inductancia. En estas líneas, se ha enrutado en orden de prioridad, empezando por la conexión de las puertas de los MOSFETs a los *drivers* y la alimentación aislada, y acabando con los *snubbers*.

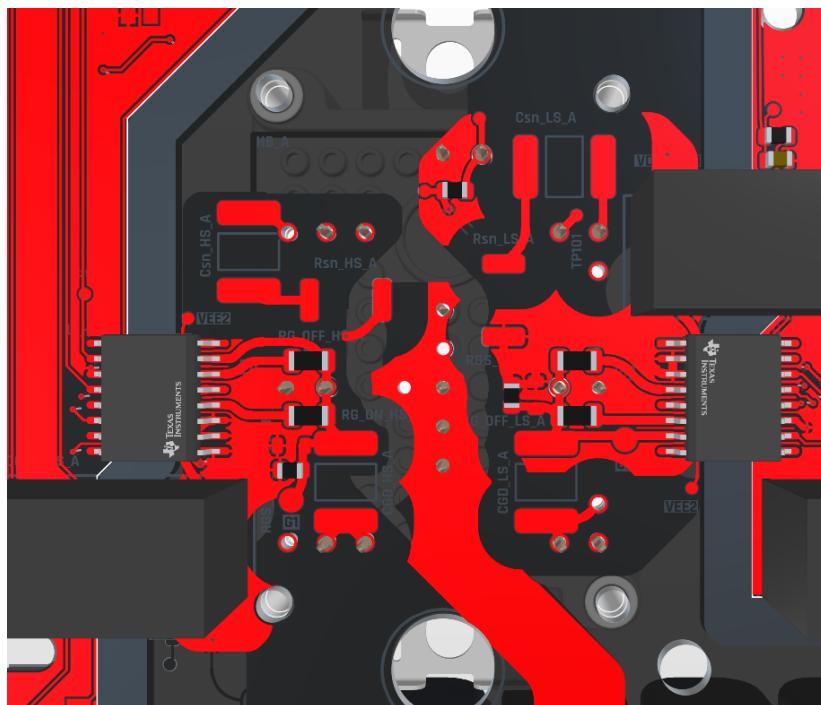


Figura 4.75: Enrutado en la capa superior de un *half-bridge*.

Debido a los parásitos que puede presentar el *layout*, los MOSFETs pueden presentar un sobrepico de tensión excesivo al conmutar. La figura 4.76 explica el circuito que pueden formar. Si la capacidad parásita es insuficiente, la inductancia parásita provocará una subamortiguación. Por ello, se ha minimizado el área que dejan estas conexiones para minimizar la inductancia parásita. Además, es importante añadir un condensador de desacoplo entre los terminales positivo y negativo de cada *half-bridge*.

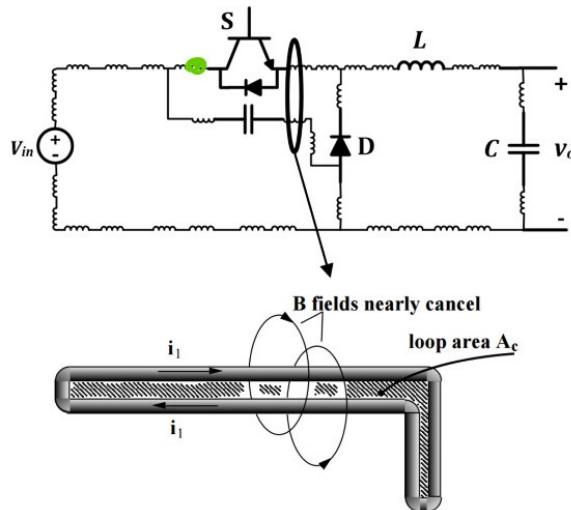


Figura 4.76: Circuito de parásitos en un medio puente, que cuando la corriente es positiva, actúa como un convertidor *buck*.

Descarga

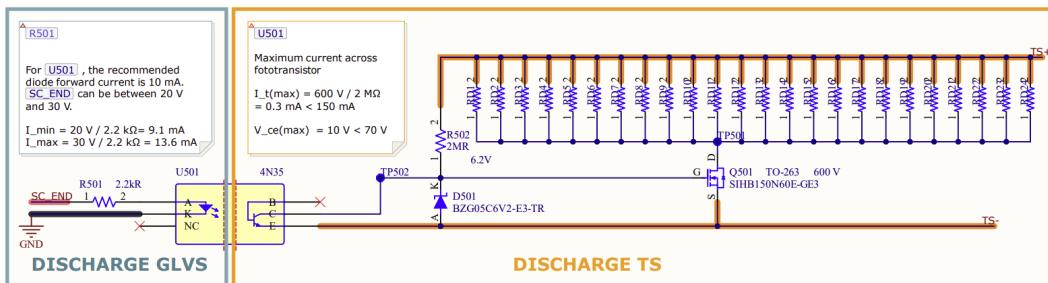


Figura 4.77: Circuito de descarga.

Una de las partes más importantes en cuanto a la seguridad eléctrica es la implementación del circuito de descarga. Este circuito se utiliza para descargar la energía almacenada en el bus de condensadores cuando el convertidor no está operativo. Los requisitos de la normativa son que el circuito debe ser capaz de descargar el bus de la tensión máxima hasta 60 V en menos de 5 segundos. Adicionalmente, el circuito debe ser capaz de aguantar de forma constante la tensión máxima.

Con tal de cumplir estos requisitos se ha optado por un circuito basado en un NMOS, cuya puerta se conecta al terminal positivo del bus a través de una resistencia de valor elevado y se limita la tensión con un diodo zener. La puerta del MOSFET está

controlada por un optoacoplador de forma aislada desde el circuito de baja tensión. Mientras el bus tenga una tensión superior a 10 V y el optoacoplador esté desactivado, el MOSFET quedará cerrado, conectando el banco de resistencias a los condensadores, descargándolos.

La selección de la resistencia se ha basado en minimizar el tamaño de la misma. La solución óptima se ha encontrado en utilizar un banco de 24 resistencias SMD de 470 kΩ, en encapsulado 2512, que es capaz de disipar 1 W. Se conectan en paralelo para llegar al valor necesario para cumplir con el requisito de tiempo de descarga. Para el cálculo del tiempo de descarga se utiliza la siguiente expresión:

$$t_{\text{dis}} = R_{\text{dis}} \cdot C \cdot \ln \left(\frac{V_{\text{inicial}}}{V_{\text{final}}} \right) \quad (4.67)$$

$$t_{\text{dis}} = R_{\text{dis}} \cdot C \cdot \ln \left(\frac{V_{\text{inicial}}}{V_{\text{final}}} \right) = \left(\frac{470 \text{ k}\Omega}{24} \right) \cdot (100 \mu\text{F}) \cdot \ln \left(\frac{600 \text{ V}}{60 \text{ V}} \right) = 4,509 \text{ s}$$

La potencia disipada en cada resistencia de descarga se calcula como:

$$P(R_{\text{dis}}, \text{máx}) = \frac{V_{\text{máx}}^2}{R_{\text{dis}}} \quad (4.68)$$

$$P(R_{\text{dis}}, \text{máx}) = \frac{V_{\text{máx}}^2}{R_{\text{dis}}} = \frac{(600 \text{ V})^2}{470 \text{ k}\Omega} = 0,766 \text{ W} < 1 \text{ W}$$

Medida de corriente

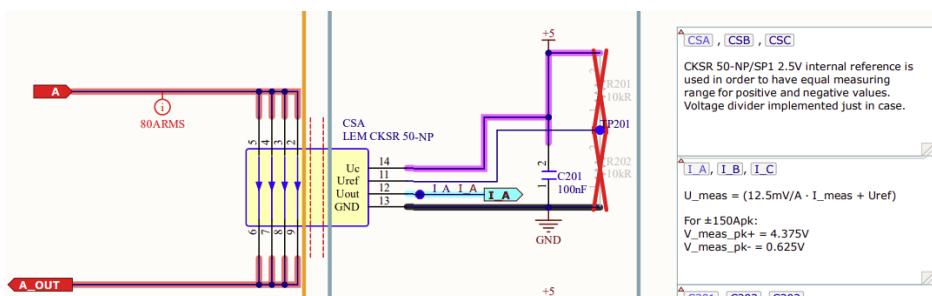


Figura 4.78: Sensor de corriente de fase.

Uno de los componentes más críticos para asegurar el funcionamiento óptimo del inversor es la medición precisa de corriente. En inversores de bajo coste, comúnmente se emplean resistencias *shunt* junto con amplificadores aislados para esta tarea. Sin embargo, esta solución puede presentar diversos desafíos de integración.

Una opción destacada para abordar esta necesidad de medición de corriente es la gama de sensores CKSR-NP de LEM. Este sensor utiliza la tecnología de transductores de corriente basados en *fluxgate*. Sus características incluyen un rango de medición amplio de hasta ±150 A para modelo seleccionado, junto con una alta precisión del

0.8 %. Además, su diseño compacto y montaje en PCB lo hacen ideal para aplicaciones donde se requiere una medición precisa y confiable de corriente en un espacio limitado.

Dado que el sensor tiene un rango de medida de ± 150 A y está alimentado a 5 V, se puede pasar por un divisor resistivo con tal de adecuar la señal de salida al rango de entrada del ADC del MCU. Adicionalmente, se puede modificar la tensión de referencia (salida a 0 A) para ajustar todavía más la medida.



Figura 4.79: Sensor de corriente LEM CKSR 50-NP

Medida de tensión

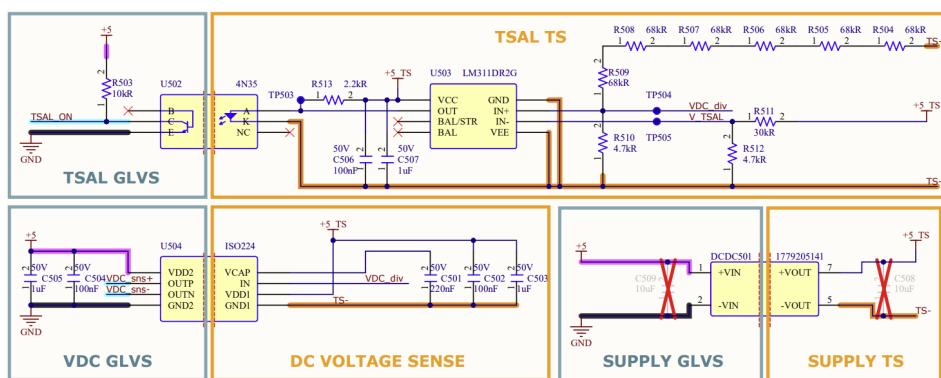


Figura 4.80: Circuito de medida y detección de tensión DC.

El sensado de la tensión es una tarea crítica en el sistema, ya que no solo es esencial para el control del motor, sino también para la detección y alerta de niveles altos de tensión. Para ambos propósitos, se emplea un divisor de tensión utilizando una serie de resistencias.

$$V_{DC,div} = (TS^+ - TS^-) \cdot \left(\frac{4,7 \text{ k}\Omega}{4,7 \text{ k}\Omega + 6 \cdot 68 \text{ k}\Omega} \right)$$

Sustituyendo los valores de 600 V y 60 V para TS+ y TS-, respectivamente,

$$V_{DC,div} = \frac{600 \text{ V} \cdot 4,7 \text{ k}\Omega}{4,7 \text{ k}\Omega + 6 \cdot 68 \text{ k}\Omega} = 6,833 \text{ V}$$

$$V_{DC,div} = \frac{60 \text{ V} \cdot 4,7 \text{ k}\Omega}{4,7 \text{ k}\Omega + 6 \cdot 68 \text{ k}\Omega} = 683 \text{ mV} .$$

El valor de potencia disipada en las resistencias de 68 kΩ se calcula

$$P_{R68k} = I_{R68k}^2 \cdot 68 \text{ k}\Omega = \left(\frac{V_{DC,div}}{68 \text{ k}\Omega} \right)^2 \cdot 68 \text{ k}\Omega = 144 \text{ mW.}$$

Se elige un encapsulado 1206 para las resistencias de 68 kΩ, que es capaz de disipar por lo menos 250 mW.

El amplificador aislado ISO224 se utiliza para medir la tensión en el bus de continua. Según la hoja de datos, la salida se calcula como un tercio del voltaje de entrada del divisor de tensión:

$$(V_{DC,sns+} - V_{DC,sns-}) = \frac{1}{3} \cdot V_{DC,div} = \frac{1}{3} \cdot \left((TS^+ - TS^-) \cdot \frac{4,7 \text{ k}\Omega}{4,7 \text{ k}\Omega + 6 \cdot 68 \text{ k}\Omega} \right)$$

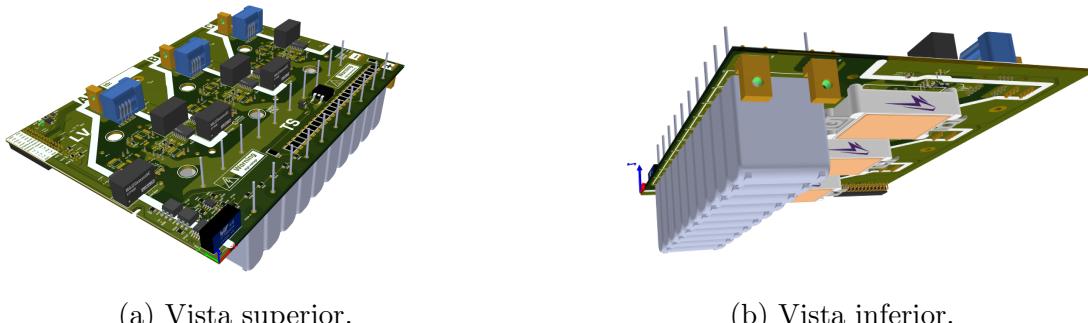
$$(V_{DC,sns+} - V_{DC,sns-}) = \frac{1}{3} \cdot 0,011388 \cdot (TS^+ - TS^-)$$

$$(V_{DC,sns+} - V_{DC,sns-}) = \frac{1}{3} \cdot 0,011388 \cdot 600 \text{ V} = 2,278 \text{ V}$$

El amplificador aislado ISO224 es una elección eficaz para esta aplicación, ya que permite medir de manera precisa la diferencia de potencial entre los terminales, proporcionando un voltaje de salida proporcional a la tensión medida y cuadrando con el rango de adquisición del ADC. Adicionalmente, la salida es diferencial, lo que permite mayor inmunidad frente a interferencias al ser enrutada hasta la placa de control. En ella se deberá incluir un amplificador diferencial o similar para llevar la lectura al ADC.

Resultado final

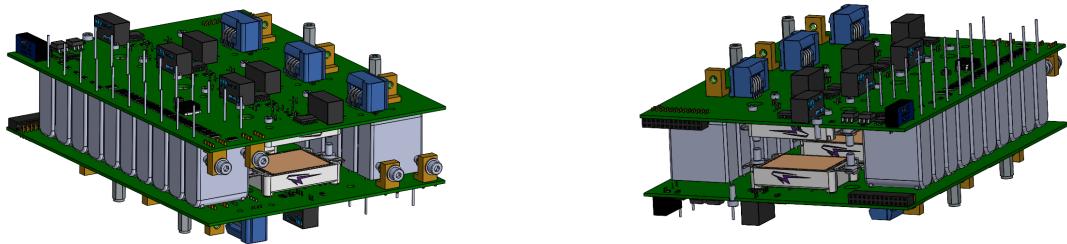
Tras haber creado todos los subcircuitos, haber emplazado cada componente y enrutado cada nodo, se ha completado el diseño de la placa de potencia. La distribución de componentes permite empaquetar dos de estas placas enfrentadas de forma muy compacta, dejando el espacio justo entre medias para la inserción de la *coldplate*. La placa de control se podrá conectar a las dos de potencia mediante conectores placa a placa, quedando a uno de las caras libres. Por la otra cara se deben incluir las conexiones de potencia DC y la entrada y salida de agua.



(a) Vista superior.

(b) Vista inferior.

Figura 4.81: Vistas 3D de la PCB de potencia.

(a) Vista frontal, conexión del bus DC y *coldplate*.

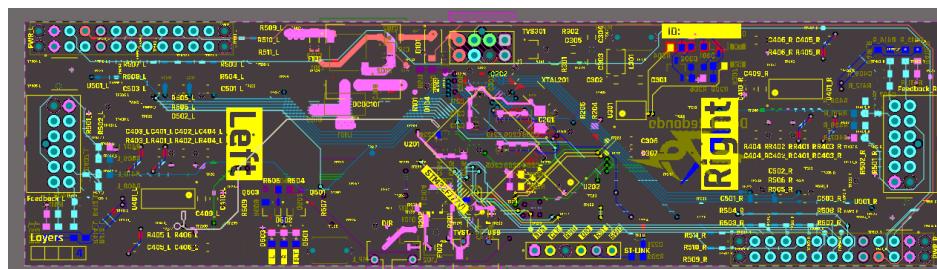
(b) Vista trasera, conexión de la placa de control.

Figura 4.82: Vistas 3D del ensamblaje de dos PCBs de potencia.

4.3.11. PCB de control

Concepto y *layout*

En esta placa se alojará el microcontrolador con todos los componentes necesarios para interactuar con las placas de potencia y el exterior. Dado que el esfuerzo de integración es mucho menor, las restricciones mecánicas y la facilidad de montaje son las que rigen el concepto. Por ello no se han tenido muchos miramientos para la disposición de los componentes, y simplemente se ha puesto atención en utilizar muchas simetrías para hacer que los circuitos repetidos para los dos motores/inversores sean exactamente iguales.

Figura 4.83: *Layout* de la PCB de control.

Como se puede observar, el microcontrolador es la pieza central y está orientado a 45° para que los pines se puedan enrutar de forma óptima hacia los extremos de la placa. Ambos lados de la placa son simétricos a excepción de los componentes que no están duplicados por el control dual.

Restricciones y enrutado

Se ha mencionado ya que esta placa no tiene muchas restricciones más allá de las mecánicas, lo cual ha permitido crearla en un formato de 150 mm por 40 mm, encajando perfectamente en ángulo recto con las placas de potencia en el espacio que dejan entre medias.

El apilado escogido es un estándar de 4 capas, con 35 micras de espesor en las externas (1 onza por pulgada cuadrada), y 17 micras en las internas (0.5 onzas por pulgada cuadrada). Esto asegura una producción barata, y la posibilidad de crear planos de referencia y alimentación en las capas internas.

Layer Stack Legend	Material	Layer	Thickness	Dielectric Material	Type	Gerber
Surface Material	Top Solder	Top Overlay	0.010mm	Solder Resist	Solder Mask	GTO
CF-004	TOP		0.035mm		Signal	GTL
Prepreg			0.100mm	PP-006	Dielectric	
Prepreg			0.100mm	PP-006	Dielectric	
Copper	GND		0.035mm		Signal	G1
			1.040mm	FR-4	Dielectric	
Copper	PWR		0.035mm		Signal	G2
Prepreg			0.100mm	PP-006	Dielectric	
Prepreg			0.100mm	PP-006	Dielectric	
CF-004	BOT		0.035mm		Signal	GBL
Surface Material	Bottom Solder	Bottom Overlay	0.010mm	Solder Resist	Solder Mask	GBS
					Legend	GBO
Total thickness: 1.600mm						

Figura 4.84: Apilado de la PCB de control.

Dado que no hay grandes magnitudes eléctricas, el enrutado se ha centrado en minimizar el área de los caminos de corriente entre cada señal y la masa de la placa. Utilizando polígonos conectados a la masa en múltiples capas (tanto la de la propia señal como las que queden por encima o por debajo) se minimiza muchísimo este área.

También se ha procurado mantener todas las señales rápidas lo más separadas posibles para evitar el *cross-talk* entre ellas, causado por el acople capacitivo que presentan dos conductores paralelos. En ocasiones no se ha podido evitar y tan solo queda confiar en el resto de buenas prácticas de enrutado.

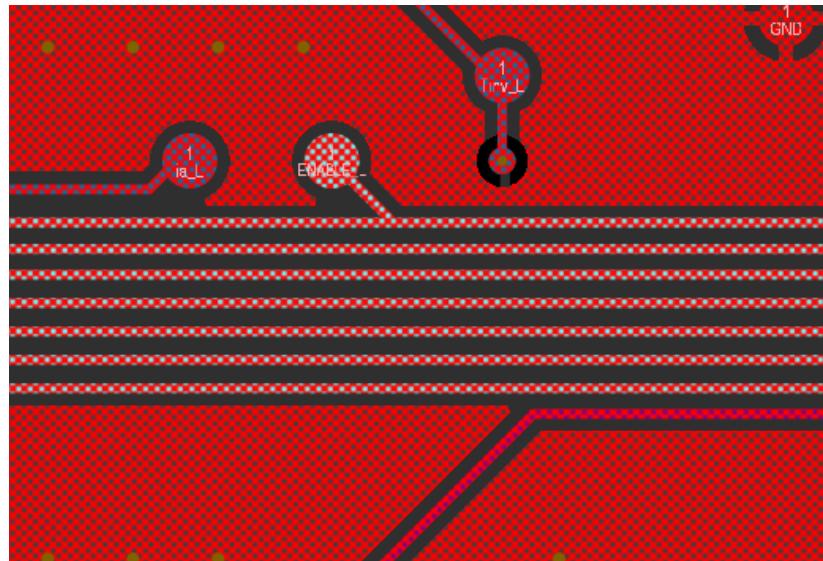


Figura 4.85: Ejemplo de grupo de señales que podrían presentar *cross-talk*.

Bloques funcionales

La PCB de control se divide en varios bloques funcionales, que igual que la PCB de potencia, se juntan en un esquemático jerárquico.

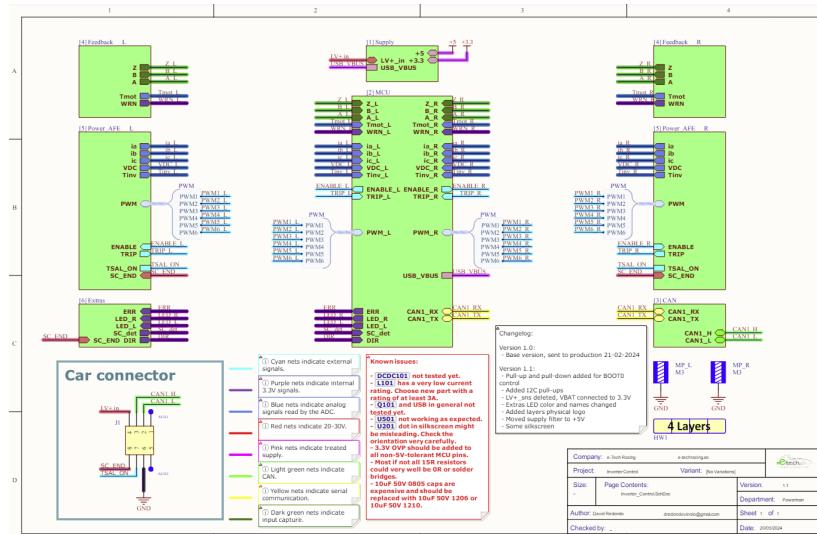


Figura 4.86: Esquemático jerárquico de la PCB de control.

En el esquemático se pueden apreciar el conector al vehículo, las monturas de la placa, unas notas indicativas con los cambios y una leyenda de colores. Los bloques son los siguientes:

- **Alimentación:** Dado que esta placa no tiene conexión al sistema de alta tensión, se alimenta exclusivamente del sistema de baja tensión del monoplaza, cuya tensión es de entre 20 V y 30 V. Por ello se implementan distintas protecciones,

un convertidor para obtener un bus de 5 V y un regulador lineal para obtener 3.3 V estables para el MCU.

- **MCU:** Es el bloque central, que implementa el STM32F777VI, un puerto USB, una memoria externa EEPROM y el conector de programación y depuración.
- **CAN:** Incluye un transceptor CAN para habilitar la comunicación con el vehículo.
- **Retroalimentación:** En estos bloques se integra la conexión del *encoder* incremental y el *front-end* analógico de la lectura de temperatura del motor.
- **Front-end analógico de la placa de potencia:** Se tratan las distintas señales que provienen de las placas de potencia, adaptándolas a los rangos de tensión que admiten los ADCs del MCU.
- **Extras:** Tan solo se colocan unos LEDs y un par de entradas digitales para el MCU.

Configuración de *hardware* del MCU

Con tal de asignar de forma adecuada la función de cada pin, se ha hecho uso de una herramienta proporcionada por STMicroelectronics llamada CubeMX. Este programa permite generar un código base con todos los periféricos del microcontrolador configurados según se escoja. Tiene una interfaz muy intuitiva que hace muy sencilla la implementación de *hardware*, permitiendo un desarrollo muy ágil.

De esta manera, se han decidido las conexiones de la placa de control con base en las siguientes consideraciones:

- **Funcionalidad del Pin:** Cada pin del microcontrolador se ha configurado para cumplir una función específica de acuerdo al mapeado que tenga cada periférico, ya que normalmente un canal específico de un periférico específico solamente se puede conectar a uno o dos pines.
- **Compatibilidad con Periféricos:** Se ha tenido en cuenta la compatibilidad entre los periféricos del microcontrolador y los dispositivos externos conectados a la placa de control.
- **Optimización del Enrutado:** Se han escogido algunos pines a medida que se ha ido enrutando la placa, por facilitar algunas conexiones.

Finalmente, se logra que el esquemático y el programa CubeMX muestren la misma distribución de pines:

4.3.11. PCB de control

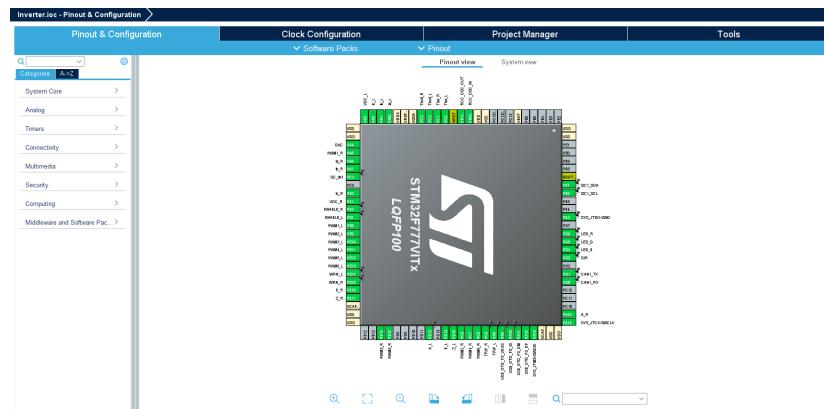


Figura 4.87: Pantalla principal de CubeMX con todos los pines usados configurados con los periféricos adecuados.

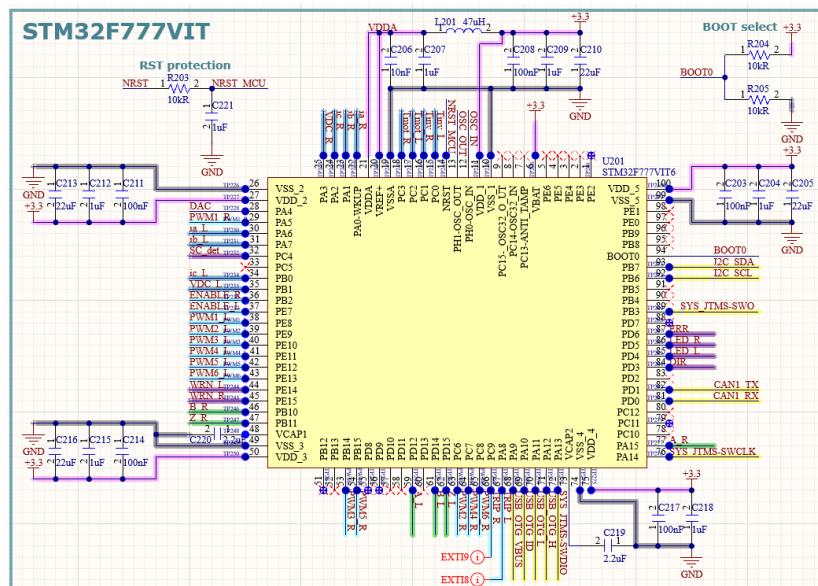


Figura 4.88: Símbolo esquemático del MCU con todas las conexiones realizadas.

Circuitos importantes

Alimentación

Ya que se propone alimentar la placa directamente desde el sistema de baja tensión del vehículo, se debe implementar un tratamiento de esta alimentación. Se incluye protección contra descargas electrostáticas, polaridad inversa y sobrecorriente. Se implementa un DC-DC de Recom de 15 W para generar el bus principal de 5 V, sin embargo, dado a problemas de disponibilidad y ensamblaje no se ha podido probar. Además, se incluye alimentación del conector USB, y ambos buses de 5 V se pasan por un filtro Pi con frecuencia de corte $f_c = \frac{1}{2\pi\sqrt{C \cdot L}} = \frac{1}{2\pi\sqrt{10 \mu\text{F} \cdot 47 \mu\text{H}}} = 7,34 \text{ kHz}$ para evitar acoplos de ruido en la alimentación. También se puede observar un regulador lineal fijo de 3.3 V para proporcionar una tensión estable al MCU, y un pequeño LED indicativo de que la alimentación está activa.

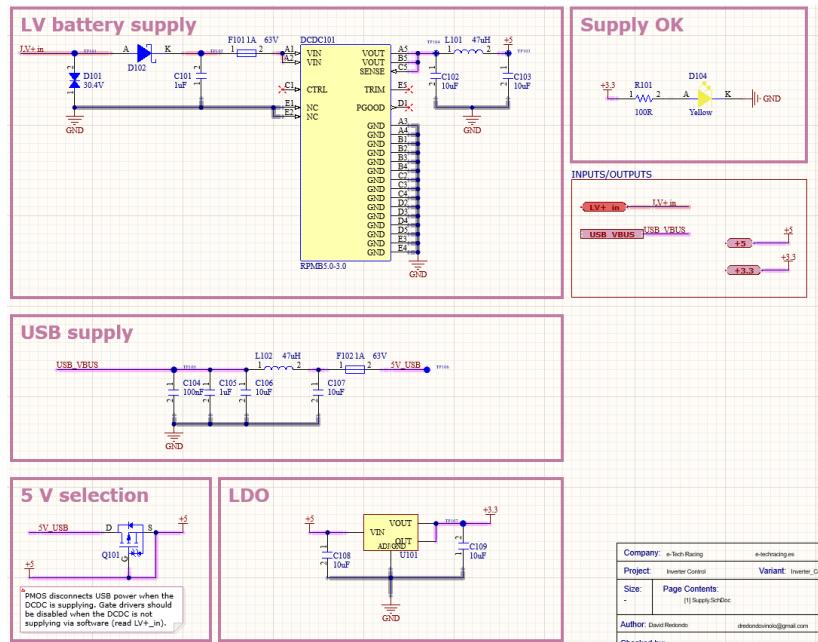


Figura 4.89: Esquemático de los circuitos de alimentación de la placa de control.

MCU

Aunque ya se ha visto la implementación del MCU en sí, su esquemático contiene otras partes como una memoria externa, el conector USB, notas sobre qué periféricos están mapeados a qué pines y el conector de programación y depuración.

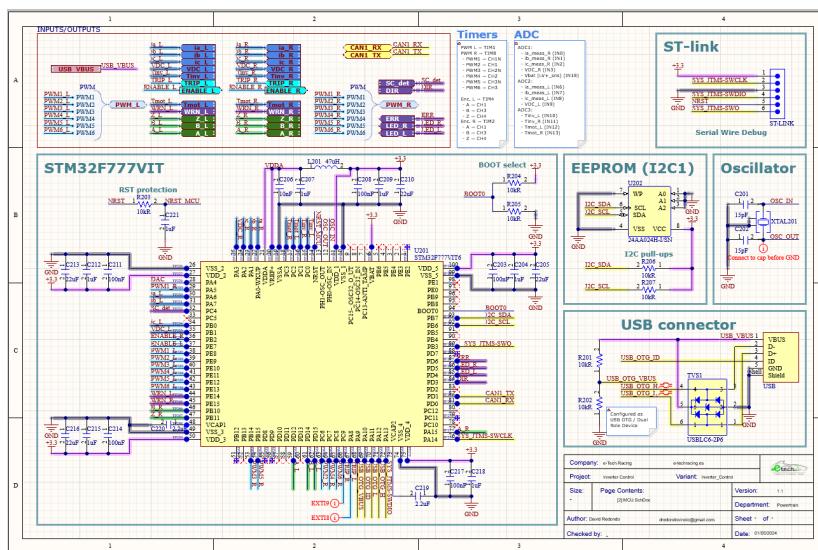


Figura 4.90: Esquemático de los circuitos relacionados con el MCU.

CAN

Para implementar comunicación CAN es necesario incorporar un *transceiver* que pueda comunicar el MCU con un bus de CAN real. Se escoge el MCP2551 por su coste, simplicidad y robustez. Se añade también un filtro de línea consistente en un *choke*

para las interferencias en modo común, y algunos condensadores. Se incluye también un final de línea por si fuera necesario, y protección contra ESD. Adicionalmente, se han implementado un par de luces LED para indicar el correcto funcionamiento del envío y recepción de datos.

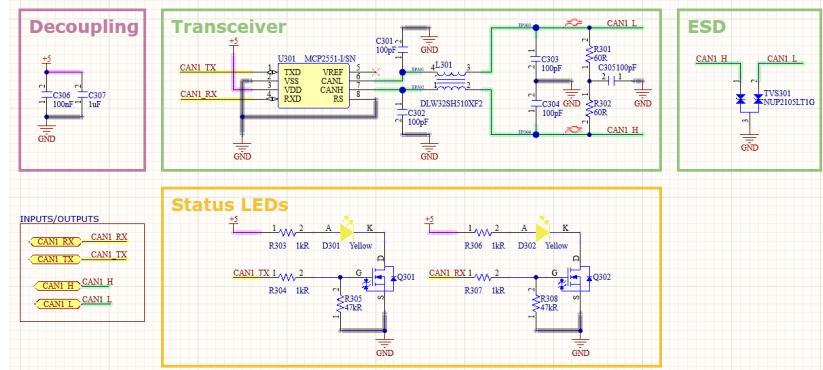
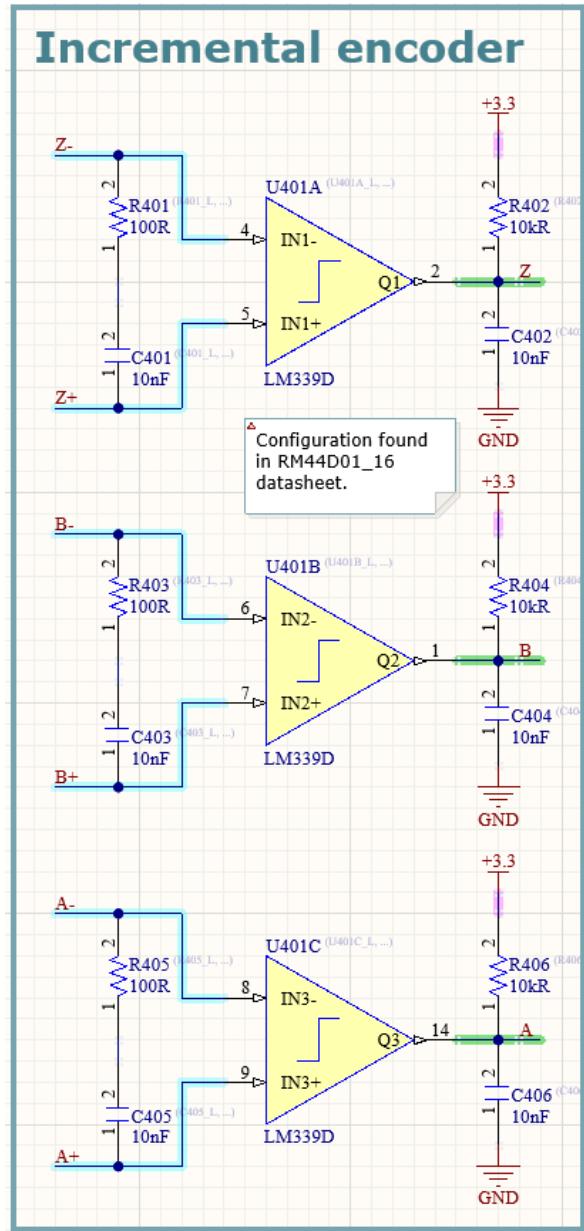


Figura 4.91: Esquemático referente a la comunicación CAN.

Retroalimentación

Este bloque se implementa por duplicado, un bloque por cada motor que se controla. Contiene los componentes necesarios para leer su posición y temperatura.

Ya que se requiere de leer la posición con un *encoder* incremental, se equipan los componentes necesarios para ello. En primer lugar, se sabe que el modelo de *encoder* en concreto tiene tres canales diferenciales, con lo que se usa el LM339, un comparador cuádruple con salida HiZ/GND para obtener la lectura desde una interrupción externa en el MCU. Cabe destacar que esta configuración de *hardware* hace compatible este inversor con sensores de efecto *Hall* para la lectura de posición, aunque no se desarrollará el código que los interprete.

Figura 4.92: Esquemático del *front end* del *encoder* incremental.

A fecha de la redacción de este trabajo todavía no se conoce el sensor de temperatura que montarán los motores, de modo que se implementa un circuito modificable para la lectura de cualquier tipo de sensor de temperatura resistivo. Aprovechando el comparador restante del LM339 se añade también una alarma configurable por *hardware* para cualquier tipo de sensor.

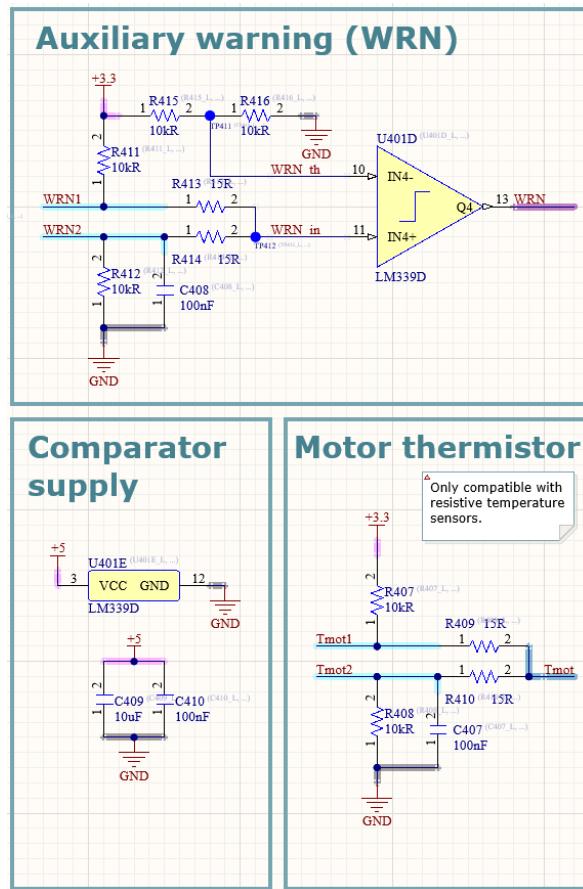


Figura 4.93: Esquemático de la lectura de temperatura del motor y alarma arbitraria.

Por último, se añade un conector para el sensor que incluye también una alimentación de 5 V, el sensor de temperatura y el sensor arbitrario junto con unas notas sobre la implementación.

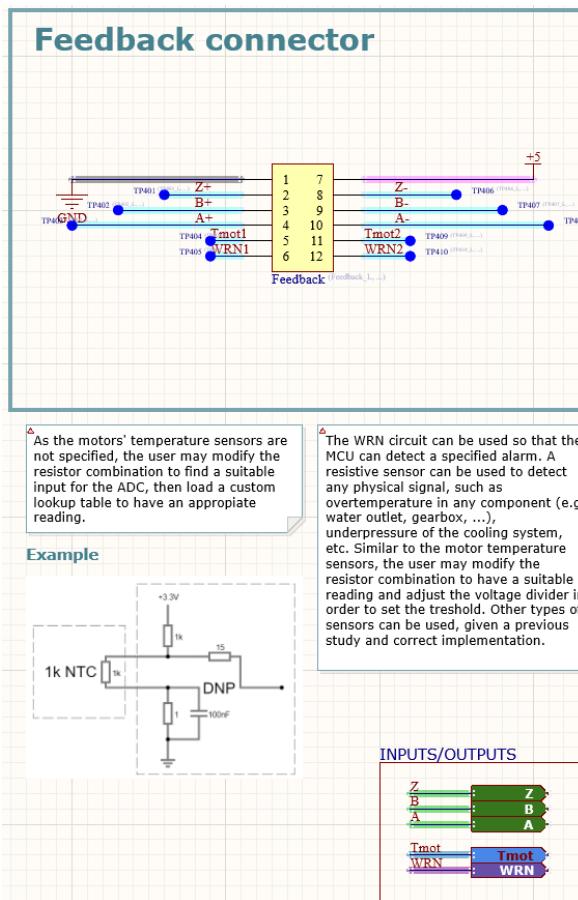


Figura 4.94: Esquemático del conector del sensor de posición y notas.

Front end de la placa de potencia

Este bloque se implementa por duplicado, un bloque por cada placa de potencia que tiene el inversor.

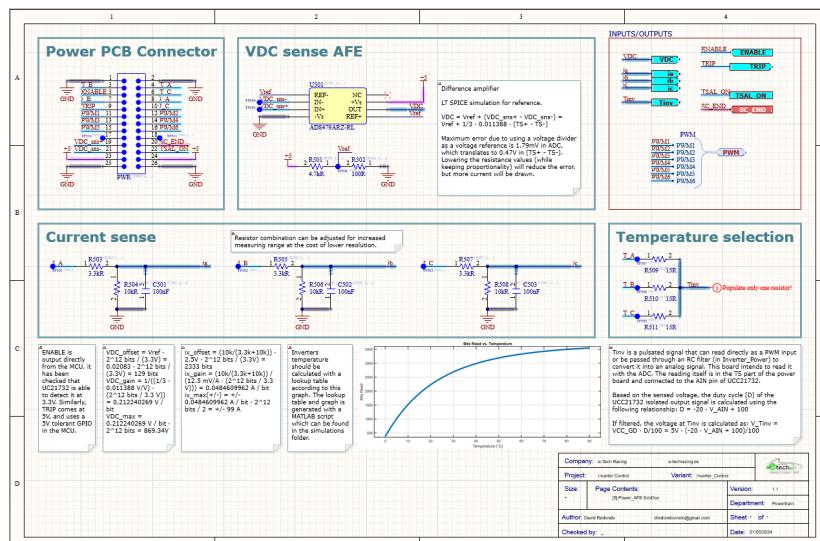


Figura 4.95: Esquemático del *front end* de la placa de potencia.

En primer lugar, *ENABLE* es una salida directa del MCU, y se ha comprobado que el UCC21710 es capaz de detectarla a 3.3V, igual que los PWMs. De manera similar, *TRIP* se lee a 5V y utiliza un GPIO tolerante a 5V en el MCU.

Para que el ADC del MCU reciba correctamente todas las señales analógicas, se deben tratar adecuadamente. Hay tres grupos de señales analógicas en cada placa de potencia: las corrientes de fase, la tensión DC y la temperatura de los semiconductores.

Corrientes de Fase

Como se ha visto en el apartado de la placa de potencia, se utilizan sensores referenciados a 5 V, y por tanto, sus señales podrían exceder el rango de 3.3 V del ADC. Por ello se implementa un simple divisor de tensión. La combinación de resistencias se puede ajustar para aumentar el rango de medición a costa de una menor resolución.

El *offset* de la corriente en bits se calcula

$$\text{offset}_i = \left(\frac{10 \text{ k}\Omega}{3,3 \text{ k}\Omega + 10 \text{ k}\Omega} \right) \cdot 2,5 \text{ V} \cdot \frac{2^{12} \text{ bits}}{3,3 \text{ V}} = 2333 \text{ bits},$$

y la ganancia de la medida de corriente

$$\text{gain}_i = \frac{\left(\frac{10 \text{ k}\Omega}{3,3 \text{ k}\Omega + 10 \text{ k}\Omega} \right)}{12,5 \text{ mV/A} \cdot \left(\frac{2^{12} \text{ bits}}{3,3 \text{ V}} \right)} = 0,0484609962 \text{ A/bit.}$$

La corriente máxima que se puede medir es

$$\pm 0,0484609962 \text{ A/bit} \cdot \frac{2^{12} \text{ bits}}{2} = \pm 99 \text{ A.}$$

Tensión de Bus

Dado que el amplificador aislado utilizado saca una señal diferencial, se debe convertir a *single-ended* usando un amplificador diferencial integrado.

Puesto que el modelo escogido presentaría una pequeña zona muerta, se debe añadir un *offset* de muy pocos milivoltios.

No existen referencias de tensión de tan poco nivel, así que se usa un divisor resistivo de valores bajos, a sabiendas de que esta decisión causa error en la medida.

El *offset* de la medida de tensión se calcula

$$\text{offset}_V = V_{ref} \cdot \frac{2^{12} \text{ bits}}{3,3 \text{ V}} = 129 \text{ bits},$$

y la ganancia de la medida de tensión en voltios por bit se calcula

$$\text{gain}_V = \frac{1}{\left(\frac{1}{3} \cdot 0,011388 \text{ V/V} \cdot \left(\frac{2^{12} \text{ bits}}{3,3 \text{ V}} \right) \right)} = 0,212240269 \text{ V/bit.}$$

La tensión máxima que se puede medir es

$$0,212240269 \text{ V/bit} \cdot 2^{12} \text{ bits} = 869,34 \text{ V}$$

Temperaturas de los semiconductores

Sería poco práctico leer las tres temperaturas de cada inversor ya que ocuparían muchos pines del MCU. Por ello, se añaden unas pequeñas resistencias, dos de las cuales no se montan, lo cual permite escoger una de las tres medidas de temperatura. Las temperaturas de los inversores deben calcularse con una *lookup table* para ahorrar tiempo de computación. Como se ha visto, la señal que sale del UC21710 es una señal pulsada que puede leerse directamente como una entrada PWM o pasar a través de un filtro RC en la placa de potencia para convertirla en una señal analógica. Esta placa pretende leerla con el ADC.

La lectura en sí misma está en la parte de alta tensión de la placa de potencia y se conecta al pin AIN de UC21710. Basándose en el voltaje leído, se calcula el ciclo de trabajo (D) de la salida analógica aislada de UC21710 utilizando la relación

$$D = -20 \cdot V_{AIN} + 100.$$

Si se filtra, el voltaje en leído por el ADC del MCU se calcula

$$V_{Tinv} = VCC_{GD} \cdot \frac{D}{100} = 5V \cdot (-20 \cdot V_{AIN} + 100) / 100.$$

Extras

En este bloque aparecen tres LEDs informativos controlados por el MCU, un interruptor para cambiar la dirección de giro de los motores, y una lectura de la cadena de seguridad.

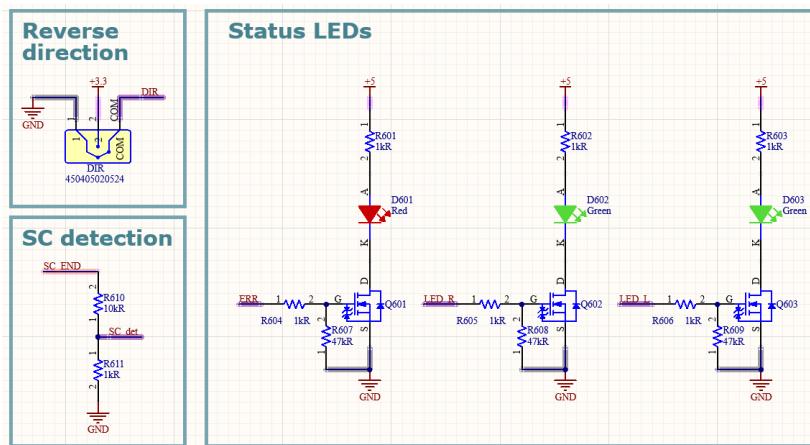


Figura 4.96: Esquemático de extras.

Resultado final

Después de diseñar todos los circuitos, emplazado todos los componentes y enrutado todos los nodos, se completa el diseño de la placa de control.



(a) Vista superior.

(b) Vista inferior.

Figura 4.97: Vistas 3D de la PCB de control.

4.3.12. Ensamblaje del convertidor

El ensamblaje del convertidor se realizó siguiendo un proceso meticuloso para garantizar la correcta integración de todos los componentes del mismo. Se utilizó un enfoque basado en diseño asistido por computadora (CAD) para planificar y visualizar el montaje antes de la construcción física. Durante el diseño de las PCBs se fue comprobando mediante CAD que no existían colisiones y que el convertidor se podía montar, atendiendo a razones como el acceso de herramientas.

Diseño en CAD

Se utilizó Solidworks para crear un modelo tridimensional detallado del convertidor. Se empezó por importar los archivos de ambas PCBs generados por Altium.

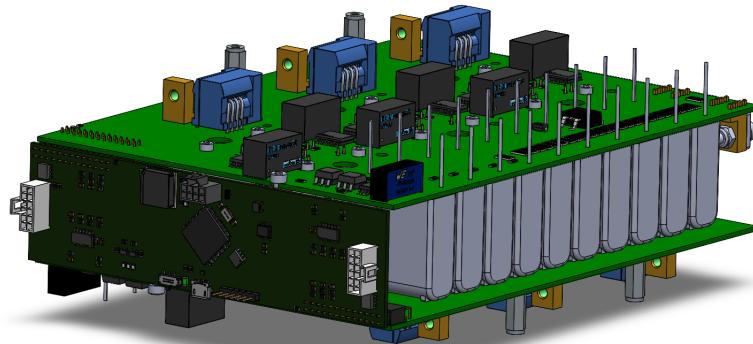


Figura 4.98: Ensamblaje de la placa de control con las dos placas de potencia.

A este ensamblaje se le añadieron unas pletinas para conectar entre sí los buses de continua, ya que van conectados a la misma batería. Además, se planteó un concepto

de *coldplate* atendiendo únicamente a las restricciones mecánicas. Este pre-diseño no fue estudiado ni simulado, simplemente marca una primera referencia para desarrollar la *coldplate* en el futuro. También se incluyó la tornillería necesaria para el montaje.

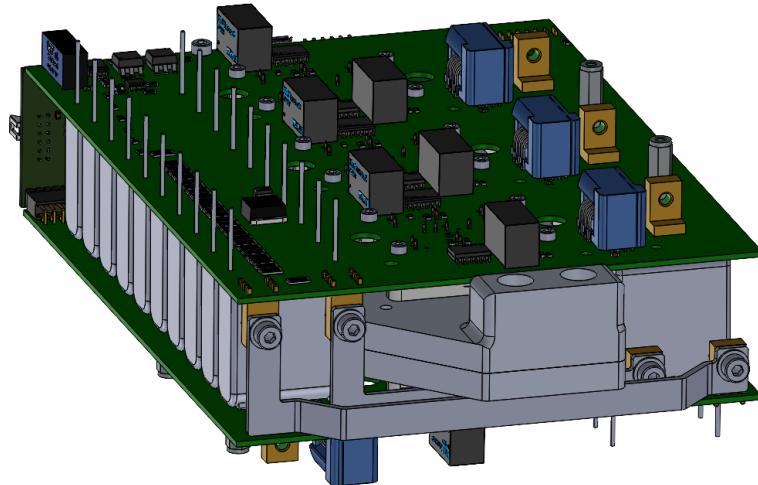


Figura 4.99: Ensamblaje con las pletinas, concepto de *coldplate* y tornillería.

Ensamblaje real

Como se discutirá en el capítulo de resultados, el convertidor fue montado poco a poco para validar sección a sección. Se soldaron los componentes manualmente con un lápiz de soldadura e hilo de estaño. Ocasionalmente se usó una pistola de calor para extraer componentes con conexiones con mucha masa térmica.

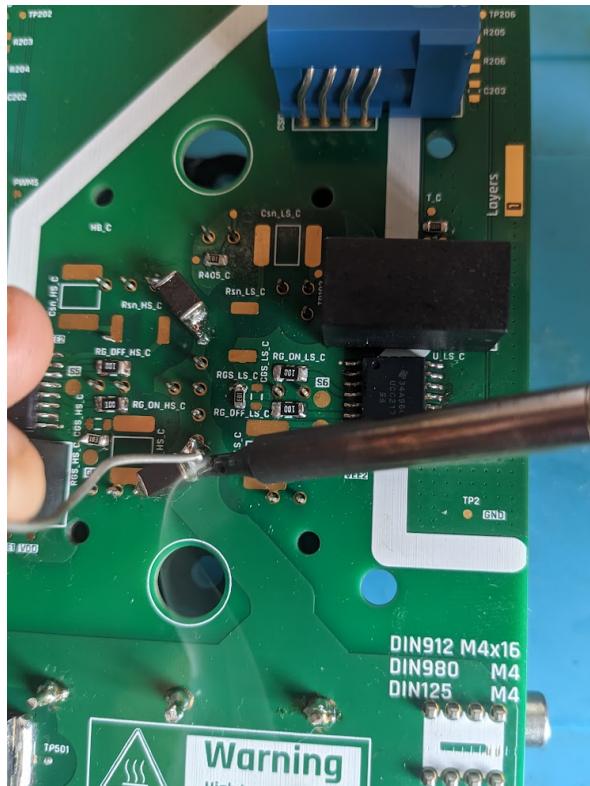


Figura 4.100: Soldadura manual con hilo de estaño.

Se montó una PCB de control y dos de potencia. Aprovechando que se dispone de los dos semiconductores seleccionados, se montó una placa de potencia con cada uno. A parte de los semiconductores, a penas hay que cambiar el valor de un par de resistencias.

Para montar los componentes *press-fit* se usó un tornillo de banco y un utilaje que permita apretar los componentes. Cabe destacar que se priorizó el montaje de los componentes con este tipo de conexión puesto que al realizar el proceso se generan flexiones en la placa que podrían dañar componentes como los condensadores cerámicos.

El caso de los conectores de potencia fue sencillo puesto que no son muchos pines y se pudo insertar fácilmente usando herramientas comunes. Para los semiconductores sin embargo, se tuvo que imprimir un utilaje personalizado para evitar colisiones con otros componentes.

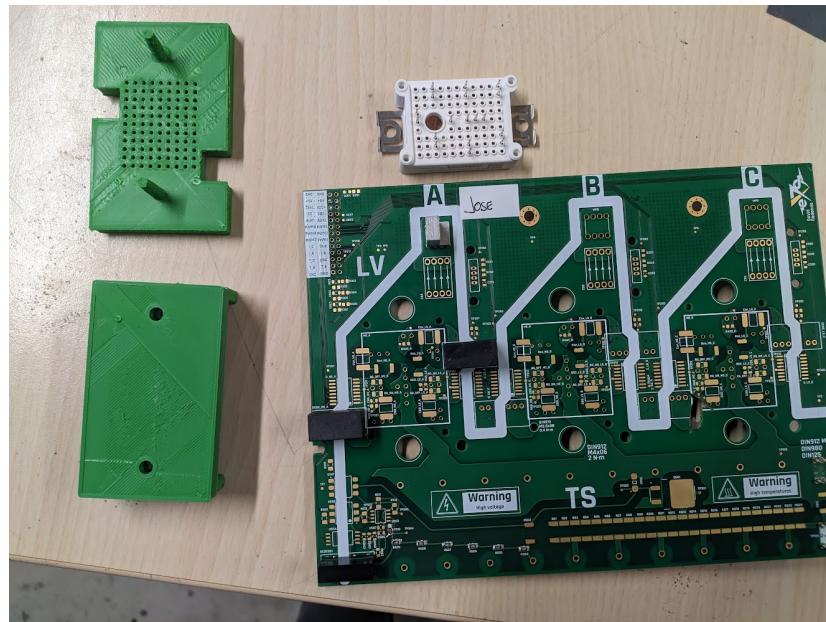


Figura 4.101: Placa lista para el montaje de un componente *press-fit* con un utilaje personalizado.

!!!!!! TORNILLO DE BANCO

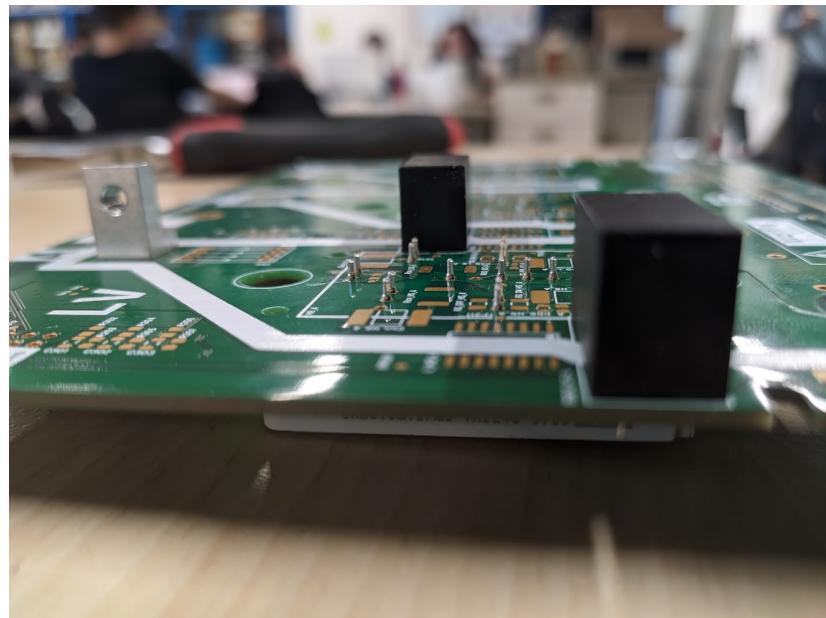


Figura 4.102: Resultado de la inserción del componente.

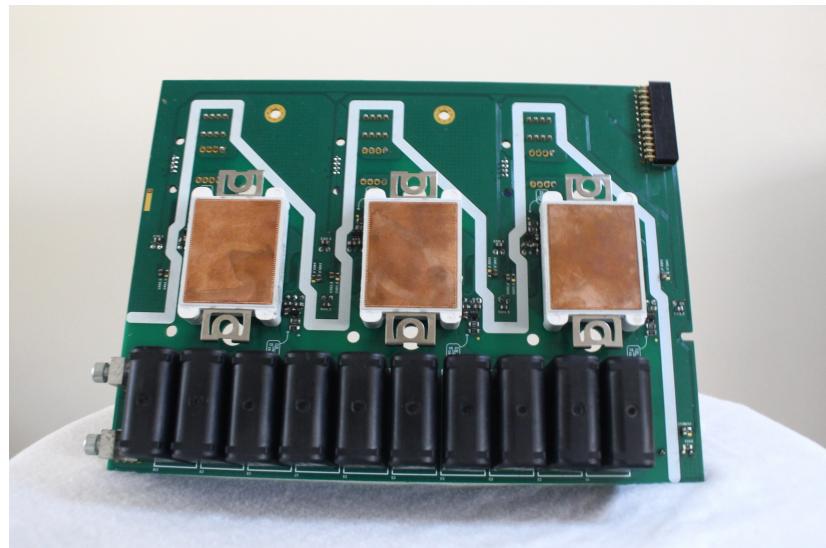


Figura 4.103: Cara trasera de la PCB de potencia (variante Wolfspeed) montada.

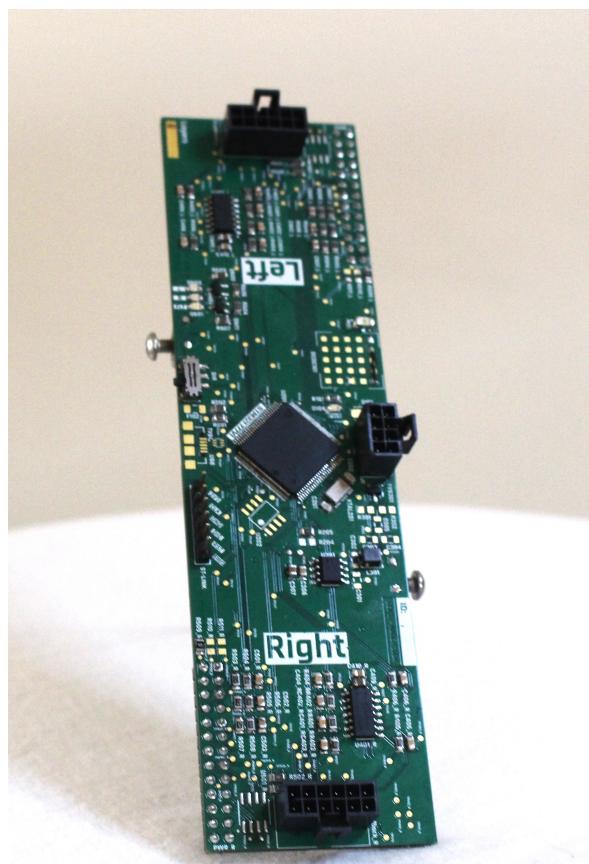


Figura 4.104: PCB de control montada.

Tras montar todos los componentes de todas las placas se procedió a ensamblarlas. A fecha de cuando se tomaron las siguientes imágenes no se dispuso de las pletinas.

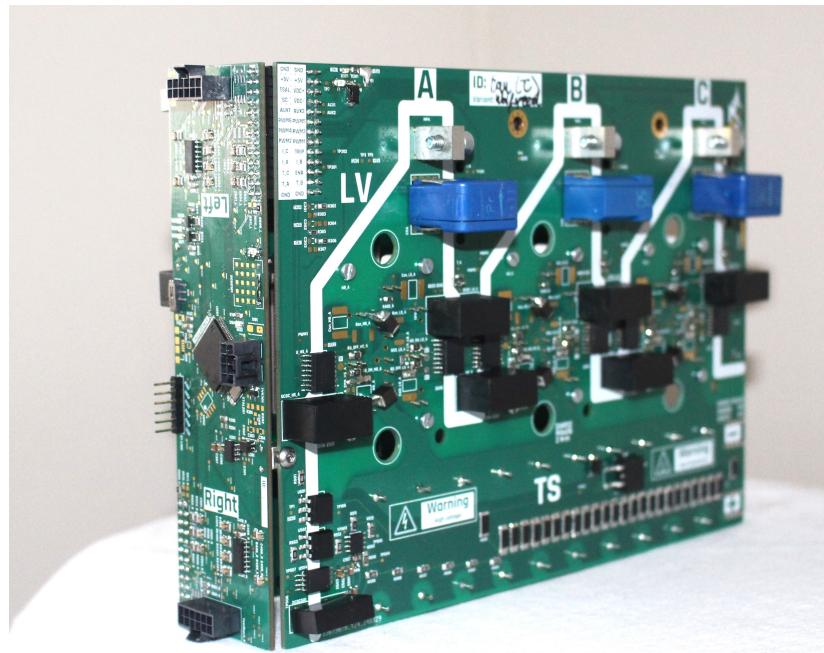


Figura 4.105: Placa de control y placa de potencia (variante Wolfspeed).

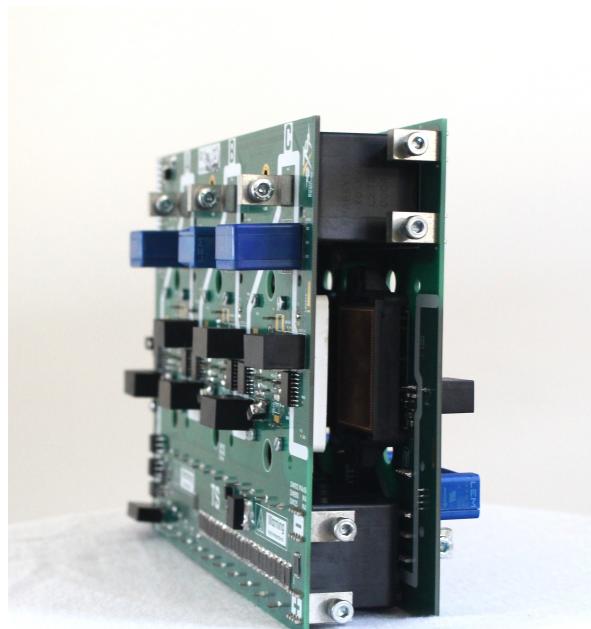


Figura 4.106: Vista en la que se aprecian ambas placas de potencia con sus respectivos semiconductores, dejando el espacio para una *coldplate*.

4.4. Firmware

4.4.1. Desarrollo del *firmware*

El *firmware* de un proyecto como este no se puede desarrollar al estilo *maker* de manera informal o improvisada. Requiere un enfoque metodológico y estructurado para garantizar la estabilidad, la eficiencia y la fiabilidad del sistema en su conjunto. Es fundamental seguir prácticas de desarrollo de *software* bien establecidas, como el diseño modular, la documentación detallada del código y la gestión adecuada de versiones. Además, el cumplimiento de estándares de codificación y la realización de análisis estáticos son pasos críticos para garantizar la calidad y la seguridad del *firmware*. Este enfoque profesional y riguroso es esencial para crear una base de código ampliable, portable y eficiente.

Objetivos del *firmware*

El desarrollo del *firmware* para el inversor tiene como objetivo principal garantizar un control preciso y eficiente de los motores PMSM en el contexto de la Formula Student. Los objetivos específicos del *firmware* son los siguientes:

- Integrar correctamente el *firmware* con el *hardware* del inversor, asegurando una interacción fluida y eficaz entre los diferentes componentes del sistema.
- Implementar un control vectorial (FOC) que permita el control independiente de los dos motores conectados al inversor.
- Implementar funciones de protección y seguridad que garanticen el funcionamiento seguro del sistema y prevengan posibles fallos o daños.

Herramientas utilizadas

Análisis estático: Cppcheck

Cppcheck es una herramienta de análisis estático de código C/C++ que se utiliza para detectar errores y problemas potenciales en el código. Realiza un escaneo exhaustivo del código para identificar posibles problemas como fugas de memoria, uso incorrecto de punteros, variables no inicializadas y otros errores comunes de programación. La integración de Cppcheck en el flujo de trabajo de desarrollo ayuda a identificar y corregir estos problemas en una etapa temprana del proceso de desarrollo, lo que contribuye a mejorar la calidad y fiabilidad del *firmware*.

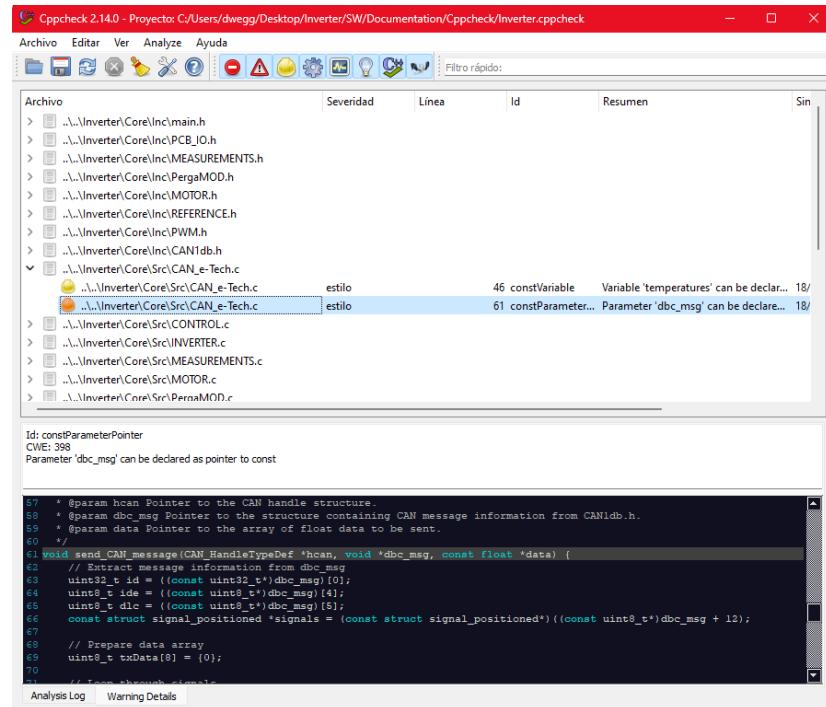


Figura 4.107: Cppcheck detectando un potencial problema.

Documentación del código: Doxygen

Doxygen es una herramienta de generación de documentación que se utiliza para crear documentación automática a partir del mismo código. Permite documentar el código usando comentarios especiales incrustados en el código mismo, utilizando una sintaxis sencilla y clara. Doxygen procesa estos comentarios para generar documentación en varios formatos, incluyendo HTML (como una pequeña web), y PDF (para una documentación más tradicional). Esta documentación incluye detalles sobre la estructura del código, la descripción de las funciones y variables, así como relaciones y dependencias entre diferentes partes del código. La generación automática de documentación con Doxygen facilita la comprensión y el mantenimiento del código, y proporciona una referencia útil para futuros desarrolladores.

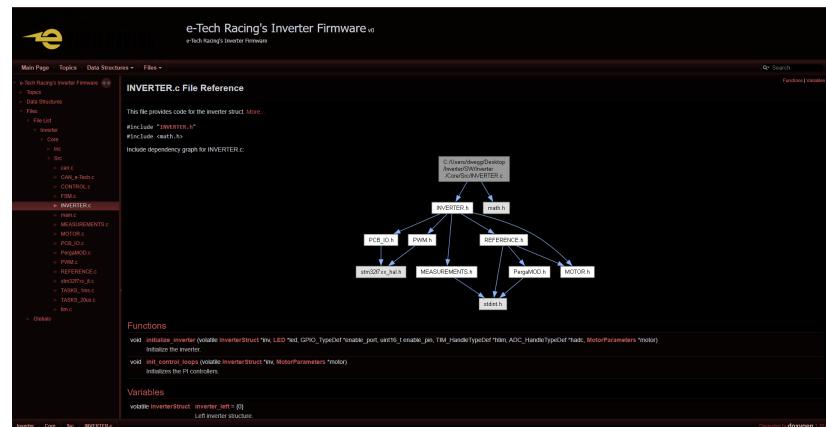


Figura 4.108: Documentación del código en formato HTML generada automáticamente por Doxygen.

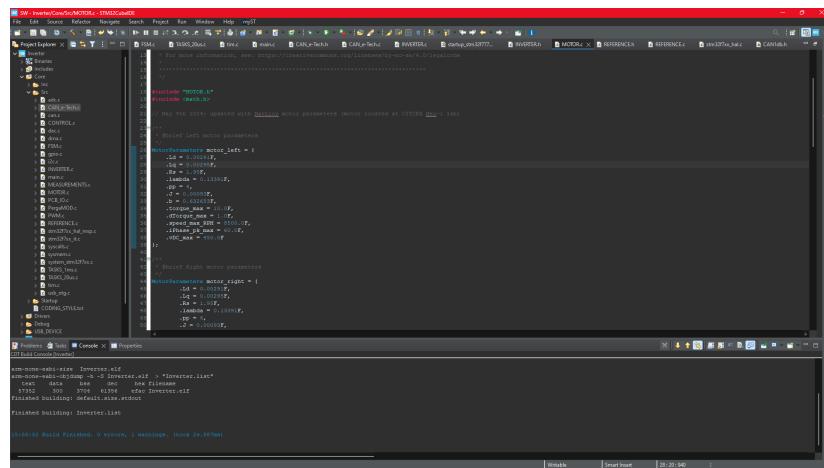


Figura 4.109: STM32CubeIDE en la perspectiva de código.

Plataforma de desarrollo

Para el desarrollo del *firmware*, se emplea STM32CubeIDE de STMicroelectronics, una plataforma de desarrollo integrado (IDE) basada en Eclipse y diseñada específicamente para trabajar con microcontroladores STM32. CubeIDE proporciona un entorno de desarrollo completo que incluye un editor de código, un compilador, un depurador y otras herramientas útiles para la programación de microcontroladores STM32. Además, integra STM32CubeMX, que facilita la configuración de periféricos y la generación de código inicial mediante una interfaz gráfica intuitiva.

El uso de CubeIDE simplifica el proceso de desarrollo de *firmware* para microcontroladores STM32, ofreciendo características como autocompletado de código, resaltado de sintaxis, depuración paso a paso y visualización de variables en tiempo real.

Lenguaje de programación

El lenguaje de programación seleccionado para desarrollar el *firmware* de este proyecto es C. Esta elección se basa en varias consideraciones relacionadas con la naturaleza de la aplicación embebida en un microcontrolador STM32.

En primer lugar, el lenguaje C es ampliamente compatible y soportado por la mayoría de los microcontroladores, incluidos los STM32. La vasta cantidad de recursos, bibliotecas y herramientas disponibles para el desarrollo en C facilita la implementación de funcionalidades complejas y la integración con el *hardware* específico del microcontrolador.

Además, el lenguaje C es altamente eficiente en términos de uso de recursos de memoria y velocidad de ejecución, lo cual es crucial en aplicaciones embebidas donde se deben cumplir estrictas restricciones de recursos.

Estilo de programación

Para mantener un código claro y coherente, se seguirán las siguientes convenciones de nomenclatura y estilo de programación:

1. Convenciones generales:

- Se usará el inglés como idioma a la hora de programar para facilitar el entendimiento.

2. Convenciones de nomenclatura para variables/estructuras/enumeraciones:

- Se utilizará **camelCase** para los nombres de variables y la declaración de estructuras/enumeraciones.
- Se utilizará **PascalCase** para la definición de estructuras/enumeraciones.
- Se emplearán nombres descriptivos que indiquen claramente el propósito de la variable.
- Se agregarán extensiones **_left** y **_right** a las variables que representen componentes del inversor izquierdo y derecho, respectivamente.
- Se evitará añadir más extensiones con guión bajo.
- Las constantes se nombrarán en **MAYÚSCULAS**.

Ejemplo:

- Para el *offset* y el *slope* del ADC:

```
#define CURRENT_SLOPE 54.4217687f /**< [A/V] (10/(4.7+10)) * ( 1 / (12.5 mV / A)) */
#define CURRENT_OFFSET 1.70068027211f /**< [V] (10/(4.7+10))* 2.5 V */
```

- Para la referencia de par:

```
torqueRef_left
torqueRef_right
```

- Para la definición y declaración de la estructura Encoder:

```
typedef struct {} Encoder; // Definición de la estructura Encoder
Encoder encoder_left; // Declaración de una estructura Encoder
```

3. Convenciones de nomenclatura para funciones:

- Se utilizará **snake_case** para los nombres de funciones.
- Se emplearán nombres descriptivos que indiquen claramente el propósito de la función, incluso si parecen demasiado largos.
- El nombre de todas las funciones debe empezar por un verbo.

Ejemplo:

- `initialize_inverter()`
- `limit_torque_to_prevent_overspeed()`
- `handle_direction()`

4. Convenciones de nomenclatura para archivos:

- Los pares de archivos .c/.h desarrollados se nombrarán en MAYÚSCULAS, facilitando su distinción de los archivos generados automáticamente.

Ejemplo:

- `MEASUREMENTS.c/.h` → Desarrollado
- `adc.c/.h` → Generado automáticamente

5. Convenciones para Abreviaturas:

- Se utilizarán abreviaciones comunes solo cuando sean ampliamente entendidas dentro del contexto del proyecto.
- Se evitarán abreviaciones ambiguas o poco claras.

Ejemplo:

- ADC (Convertidor Analógico-Digital)
- FSM (Máquina de Estados Finitos)
- LUT (Tabla de Búsqueda)

6. Comentarios:

- Cada archivo y función debe ir precedido por un comentario estilo Doxygen que explique claramente su propósito.

Ejemplo:

```
/**
 * @brief Computes d-q currents from current measurements and electrical angle.
 *
 * This function computes the d-q currents from phase currents (ABC), theta_e, and stores
 * the results in the provided pointers.
 *
 * @param[in] ia Phase A current in A.
 * @param[in] ib Phase B current in A.
 * @param[in] ic Phase C current in A.
 * @param[in] theta_e Electrical rotor position in radians.
 * @param[out] idMeas Pointer to store the D-axis current.
 * @param[out] iqMeas Pointer to store the Q-axis current.
 */
void get_idiq(float ia, float ib, float ic, float theta_e, float *idMeas, float *iqMeas);
```

4.4.2. Arquitectura del *firmware*

Estructura del *firmware*

Módulo A

Módulo B

Módulo C

Módulo D

Módulo E

4.4.3. Implementación del *firmware*

Configuración del MCU

Manejo de interrupciones

Algoritmos de control

5. Resultados y validación

5.1. Introducción

La validación del sistema es una etapa crítica en el proceso de desarrollo de cualquier proyecto, especialmente en un proyecto de la magnitud y complejidad del convertidor diseñado. La validación se encarga de verificar que todos los componentes y subsistemas del convertidor funcionen correctamente y cumplan con los requisitos y especificaciones establecidos previamente.

Dada la naturaleza multifacética del convertidor, que abarca desde aspectos eléctricos y electrónicos hasta aspectos de control y *firmware*, es esencial planificar una estrategia de validación exhaustiva y efectiva. En este sentido, se ha adoptado un enfoque metodológico basado en el modelo en V, que permite validar los subsistemas desde los más específicos hasta los más generales, siguiendo un flujo lógico y sistemático.

En este capítulo se desarrolla la parte derecha de la 'V', diseñando y ejecutando las pruebas a varios niveles. La estrategia de validación se divide en tres grandes bloques: validación de *hardware*, validación de *firmware* e integración. Cada bloque tiene sus propios desafíos y requisitos específicos, pero todos contribuyen al objetivo final de asegurar el funcionamiento correcto y robusto del convertidor.

5.2. Validación de *hardware*

La parte más crítica en la validación del convertidor es la de *hardware* puesto que cada iteración cuesta tiempo y dinero. Por ello, son las pruebas de *hardware* las que se diseñan y ejecutan más meticulosamente. En particular, se prestó especial atención a la placa de potencia, puesto que es el diseño más complicado de los dos.

Inicialmente, se elaboraron hojas de cálculo detalladas para cada uno de los tests, abarcando desde la inspección visual y pruebas de subcircuitos, hasta la evaluación de sistemas algo más complejos. Sin embargo, a medida que avanzaba el proceso de validación y se integraban los subcircuitos unos con otros, se hizo evidente que un enfoque más dinámico y adaptativo era necesario. Esto se debió a que los problemas encontrados y las soluciones implementadas requerían una validación continua y menos estructurada, lo que permitía realizar ajustes y mejoras en menos tiempo.

5.2.1. Pre-inspección de la placa de potencia

Antes de realizar cualquier prueba funcional, se llevó a cabo una inspección visual exhaustiva de la PCB de potencia para identificar cualquier defecto físico o de manufactura. Esta inspección incluyó la verificación del acabado superficial, la serigrafía, el grosor del cobre en las capas y las dimensiones de la placa. En la primera

iteración se encontraron algunos problemas menores, como una serigrafía invertida en los conectores del bus de continua, pero no afectaron las pruebas iniciales.

Se validó el grosor de cobre en las vías para aportar confianza a la hora de montar los componentes *press-fit*. Para ello, se extrajo una sección de la PCB que cortaba transversalmente una vía y se pulió. Posteriormente, se tomó una fotografía con un microscopio y se hicieron medidas digitales.

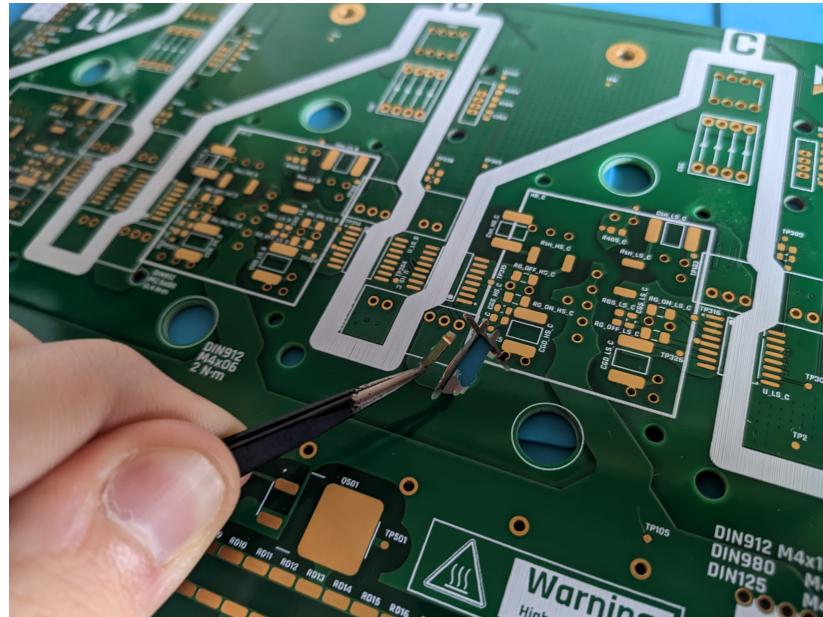


Figura 5.1: Extracción de una muestra de la PCB.

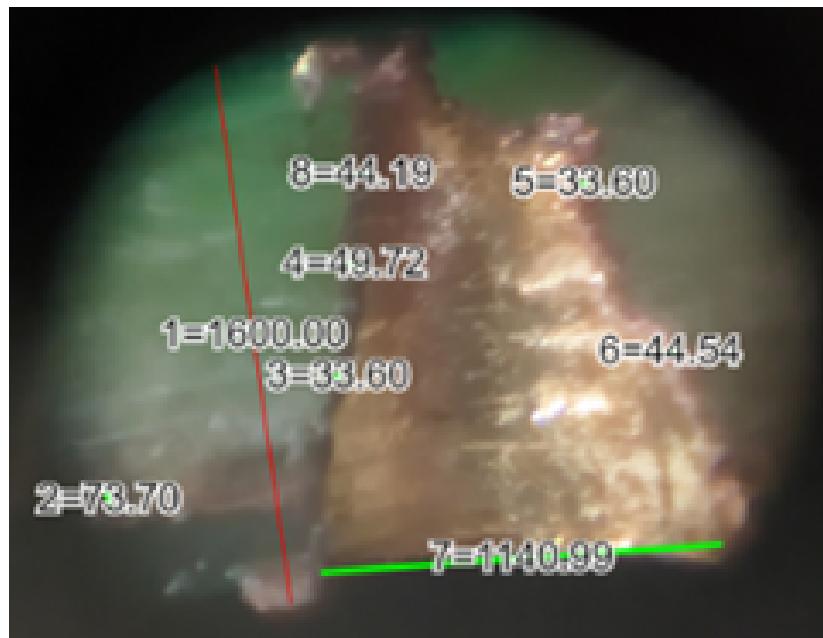


Figura 5.2: Micrografía de una vía de la placa de potencia. Se pueden apreciar los grososres de cobre en las 4 capas y en la vía.

Se pudo comprobar que el grosor de las capas era correcto (70 micras), y que el de las vías estaba dentro de la tolerancia establecida (de 25 a 50 micras).

5.2.2. Primer contacto con la placa de potencia

Durante la primera interacción con la placa de potencia, se realizaron diversas pruebas para validar el funcionamiento de los distintos componentes y subcircuitos de forma aislada. Principalmente se busca validar una funcionalidad básica de cada uno de forma separada, de manera que si se encuentran problemas de integración se sepa que son de integración y no de los componentes o circuitos individuales. Se ordenaron estas pruebas de "dentro hacia afuera", empezando por el circuito de *driving* y acabando con la interacción externa del mismo. También se validó la adquisición de variables analógicas y se probó la funcionalidad básica del circuito de descarga.

Alimentación del *gate driver*

Para validar la alimentación del *gate driver* se separó en una verificación del LDO para la tensión negativa por separado, y en otra para validar el DC-DC.

En primer lugar se pobló únicamente el propio LDO con los componentes necesarios para su funcionamiento. Se anotó el primer error de la placa, que consiste en una equivocación en el mapeado de los pines del esquemático al *footprint*. De todas formas, se realizó un *dead-bugging* para conectar el componente de forma correcta y poder verificar el circuito de forma exitosa.



Figura 5.3: LDO *dead-bugged* con resina para evitar daños con las siguientes pruebas.

- **Valor esperado:** $-4 \text{ V} \leq V \leq -3.5 \text{ V}$
- **Resultado:** -3.78 V

La dificultad de realizar este *rework* fue motivo suficiente para corregir el diseño, pero se esperó a finalizar la mayoría de pruebas básicas antes de pedir placas nuevas.

Después de verificar el funcionamiento del LDO se probaron los DC-DCs aislados. Funcionaron según lo previsto, aunque se notó que se calentaban ligeramente, incluso en vacío (sin carga). Esto se debe a su curva de rendimiento, que con cargas bajas es muy poca. Además se anotaron variaciones en las tensiones de salida, pero resultó que eran función de la carga, y está explicado en la hoja de datos.

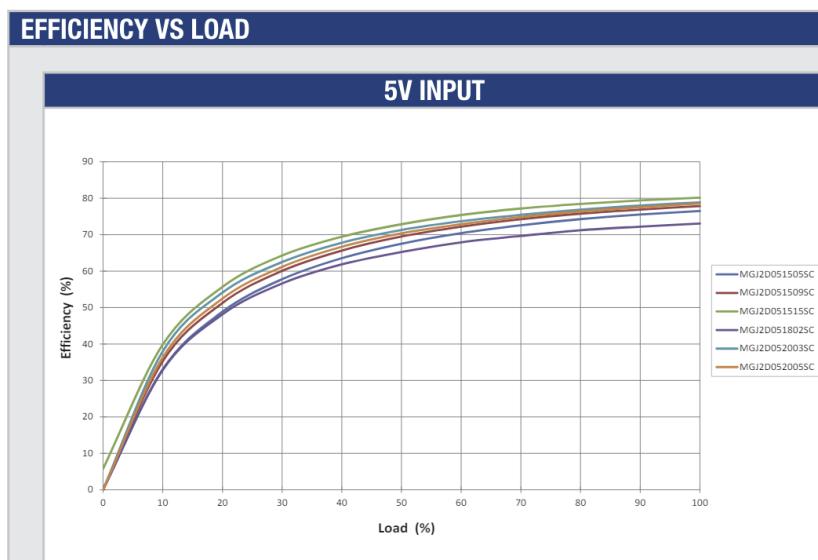


Figura 5.4: Curva de rendimiento del DC-DC aislado.

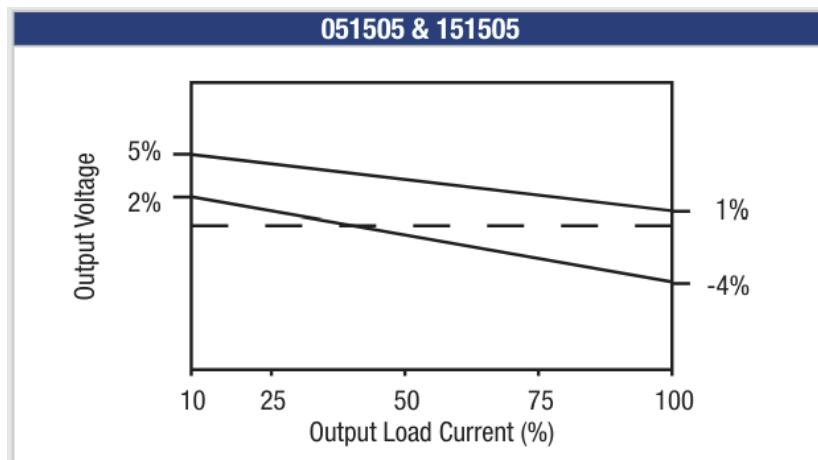


Figura 5.5: Variación de la tensión de salida en función de la carga.

- **Valor esperado:** $15.5 \text{ V} \leq \text{VDD} \leq 14.5 \text{ V}$; $-4 \text{ V} \leq \text{VEE} \leq -3.5 \text{ V}$
- **Resultado:** 15.94 V ; -3.78 V

Funcionalidad del *gate driver*

Para validar la funcionalidad básica del *gate driver*, se montó un módulo de potencia y los dos *gate drivers* necesarios para controlar los dos MOSFETs.

- **OK y TRIP:** Se poblaron los componentes necesarios y se alimentó la placa desde el lado de LV. Se midieron los nodos TRIP y OK. Ambas señales tenían un valor de 5V, confirmando su correcto funcionamiento sin errores. Posteriormente se forzaron fallas en el lado del semiconductor y ambas bajaron a 0V.
- **Dispositivos apagados en reposo:** Se montaron las resistencias de puerta y se verificó el estado de ambos MOSFETs midiendo la resistencia entre *drain* y *source*. Ambos dispositivos se encontraban en estado *OFF*, confirmando que no se encendían.
- **Enable no enciende los dispositivos:** Se introdujo una señal de 3.3V el nodo ENABLE. De nuevo, ambos dispositivos se encontraban en estado *OFF*, lo cual es el resultado esperado, ya que ambas señales de PWM estaban en estado bajo, es decir, consignando que los MOSFETs estuvieran abiertos.
- **Medición de temperatura:** Se verificó la capacidad del *gate driver* de tomar una medida analógica, en concreto, de la NTC interna del semiconductor. Se contrastaron las medidas obtenidas con los cálculos previos, arrojando resultados satisfactorios.

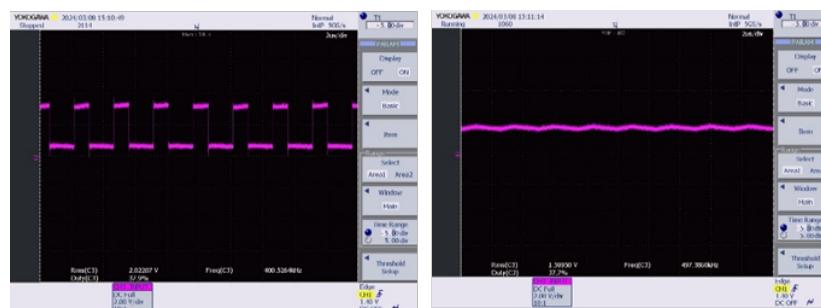


Figura 5.6: Medición de temperatura ambiente. Se puede ver la señal de salida sin filtrar (PWM) y filtrada.

Sensado de corriente

Para validar el funcionamiento del sensado de corriente, se realizaron las siguientes pruebas:

- **Standby:** Se esperaba una salida de 2.5V con un margen de error de ± 5 mV. La salida medida fue de 2.499 V, lo que se considera dentro del rango aceptable.
- **Operación básica:** Se esperaba una salida de 2.5625V para una corriente de +5A y 2.4375V para una corriente de -5 A, con un margen de error de ± 5 mV. Las mediciones obtenidas fueron de 2.557 V para +5 A y 2.436 V para -5 A, ambas dentro del rango esperado.

Descarga

Para verificar la funcionalidad de la descarga, se realizaron las siguientes pruebas:

- **Descarga pasiva:** Se precalculó una descarga de 24 V a 10 V en 35 segundos usando un 20 % de la capacidad del bus. La medición obtenida fue de 5.5 segundos, debido a que no se tuvo en cuenta el resto de conexiones entre los terminales positivo y negativo. La resistencia aparente entre estos terminales era más baja, principalmente por el divisor de tensión usado para tomar la medida de voltaje del bus. El divisor de voltaje para la adquisición ya es de $6 \cdot 68 \text{ k}\Omega + 4,7 \text{ k}\Omega = 412,7 \text{ k}\Omega$, y está en paralelo con la resistencia de $2 \text{ M}\Omega$, lo que resulta en una resistencia equivalente de

$$\frac{1}{\frac{1}{2 \text{ M}\Omega} + \frac{1}{412,7 \text{ k}\Omega}} = 342 \text{ k}\Omega,$$

lo que equivale a aproximadamente 6 segundos de descarga. Al agregar cualquier otra resistencia en paralelo, como la de los propios semiconductores u otros componentes, los cálculos coinciden.

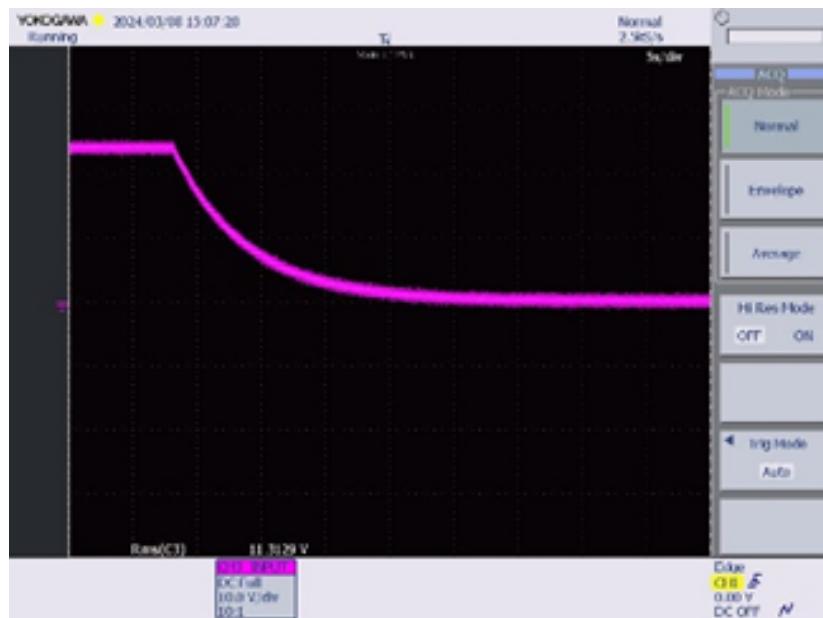


Figura 5.7: Descarga pasiva a 24 V con $20 \mu\text{F}$.

- **Control de la descarga:** Se aplicó un voltaje de 20V en el nodo *SC-END* y se suministraron 24 V entre *TS+* y *TS-*. Se esperaba que la diferencia de potencial entre *TS+* y *TS-* fuera de 10 V 0.35 segundos después de retirar el voltaje, que coincidió perfectamente con el cálculo previo.

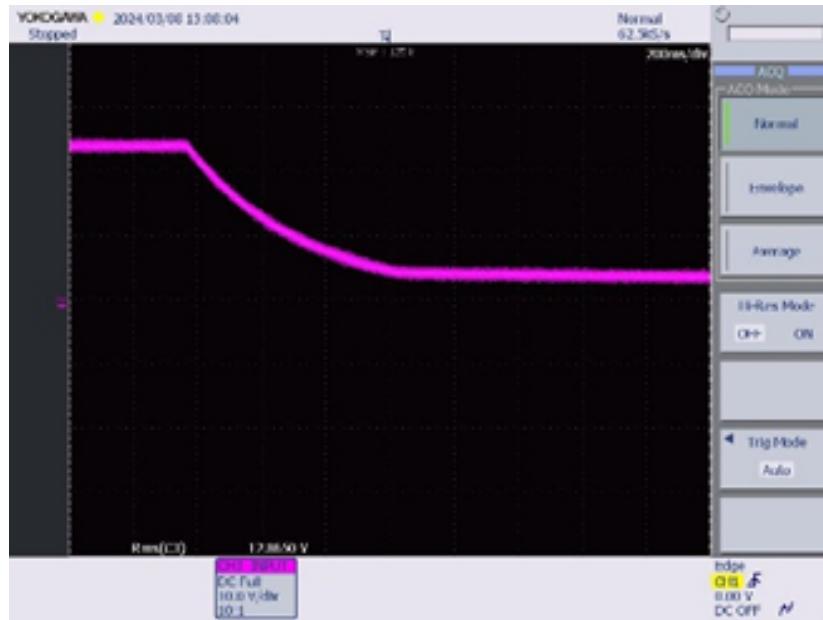


Figura 5.8: Descarga activa a 24 V con $20 \mu\text{F}$. Se puede ver como a 10 V (la tensión del zener) se desactiva la descarga activa.

Sensado de voltaje

Para verificar el sensado de voltaje, se llevaron a cabo las siguientes pruebas:

- **Divisor de voltaje:** Se aplicaron 24 V entre $TS+$ y $TS-$. Se capturó el voltaje en el divisor de tensión. La tensión esperada era de $0,011388 \cdot 24 \text{ V} = 0,2733 \text{ V}$ con un margen de error de $\pm 1 \text{ mV}$. La medición obtenida fue de 0.272 V, dentro del rango aceptable.
- **Alimentación aislada:** Se montó el convertidor DCDC501 y se alimentó la placa desde el lado LV para medir el voltaje entre $+5_TS$ y $TS-$. Se esperaba una lectura de 5V con un margen de error de $\pm 0.5 \text{ V}$. Sin embargo, la medición fue de 5.5 V, indicando que se deberá ajustar el divisor de voltaje para el umbral de detección debido a la inexactitud.
- **Detección de voltaje:** Se aplicaron 50 V entre $TS+$ y $TS-$, y se midió $TSAL_ON$. Se repitió la prueba con 60 V. Se esperaba que V_{TSAL} fuera de 677 mV y que $TSAL_ON$ fuera 5 V con 50 V de tensión de bus y 0 V con 60 V. Sin embargo, la medición fue incorrecta, ya que el umbral aproximado para la transición fue de 63 V. Se encontró que la causa podría estar la tolerancia del divisor de resistencia y el hecho de que la salida del convertidor DC-DC era medio voltio superior.
- **Sensado de voltaje:** Se aplicaron 13V entre $TS+$ y $TS-$. Se capturó el valor entre VDC_{sns+} y VDC_{sns-} . Se esperaba que la diferencia de potencial fuera de 0.0495 V. La medición obtenida fue de 0.05 V, dentro del rango aceptable.

Resumen de errores encontrados y revisión de la PCB

Se encontraron los siguientes errores durante el proceso de prueba y montaje:

1. **Problema:** El serigrafiado de los conectores de alimentación (+ y -) está invertido. Esto no afecta las pruebas.
Solución propuesta: Nombrar los conectores según corresponda y cambiar los textos del serigrafiado. Pedir nuevas PCBs.
2. **Problema:** Se conectó accidentalmente el nodo de 5 V de alimentación a *SC-END* (20 V), lo que hizo que todos los componentes alimentados a 5 V murieran.
Solución propuesta: Añadir protecciones a la alimentación de 5 V.
3. **Problema:** El *footprint* del LDO tiene los pines 4 y 5 intercambiados (-*VEE*, *FB*).
Solución propuesta: Montar el LDO verticalmente con conexiones flotantes cruzadas.

Para realizar el resto de pruebas se pidieron nuevas placas, con las siguientes modificaciones:

- Protecciones para la alimentación de 5 V.
- Se intercambiaron los pines 4 y 5 en los LDO de los *gate drivers*.
- Se intercambiaron *MP+* y *MP-* junto con su serigrafiado.
- Se añadieron *testpoints* para la referencia de los sensores de corriente.
- Se añadieron *testpoints* para *VDC_sns+* y *VDC_sns-*.
- Se renombraron los *testpoints* en el esquemático de los *gate drivers* para agilizar las pruebas.
- Se añadieron varios textos e indicaciones en el serigrafiado.

5.2.3. Comutación de una rama como DC-DC síncrono**5.2.4. Pruebas térmicas y de alta tensión****5.2.5. Validación de la placa de control****5.3. Validación de *firmware*****5.3.1. Verificación de los periféricos****5.3.2. Comutación de las tres ramas****5.3.3. Lazo abierto de tensión con carga R-L****5.3.4. Lazo abierto de tensión con un motor****5.3.5. Lazo cerrado de corriente con carga R-L****5.3.6. Adquisición de la posición del motor****5.3.7. Lazo cerrado de corriente con un motor****5.3.8. Trayectorias de control****5.3.9. Control dual****5.4. Integración**



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO DE FINAL DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y
AUTOMÁTICA

VOLUMEN II: APÉNDICES

Diseño e implementación de un inversor trifásico dual para tracción eléctrica

Autor:

David Redondo

Director:

Prof. Alfonso Conesa Roca

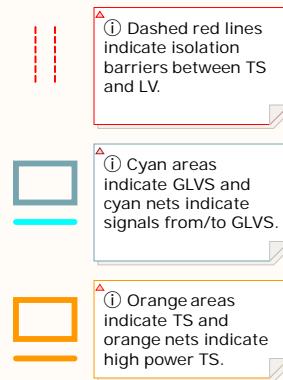
Convocatoria: Junio 2024



A. Apéndices

A.1. Obtención de los Parámetros de un PMSM

A.2. Documentación del *hardware*



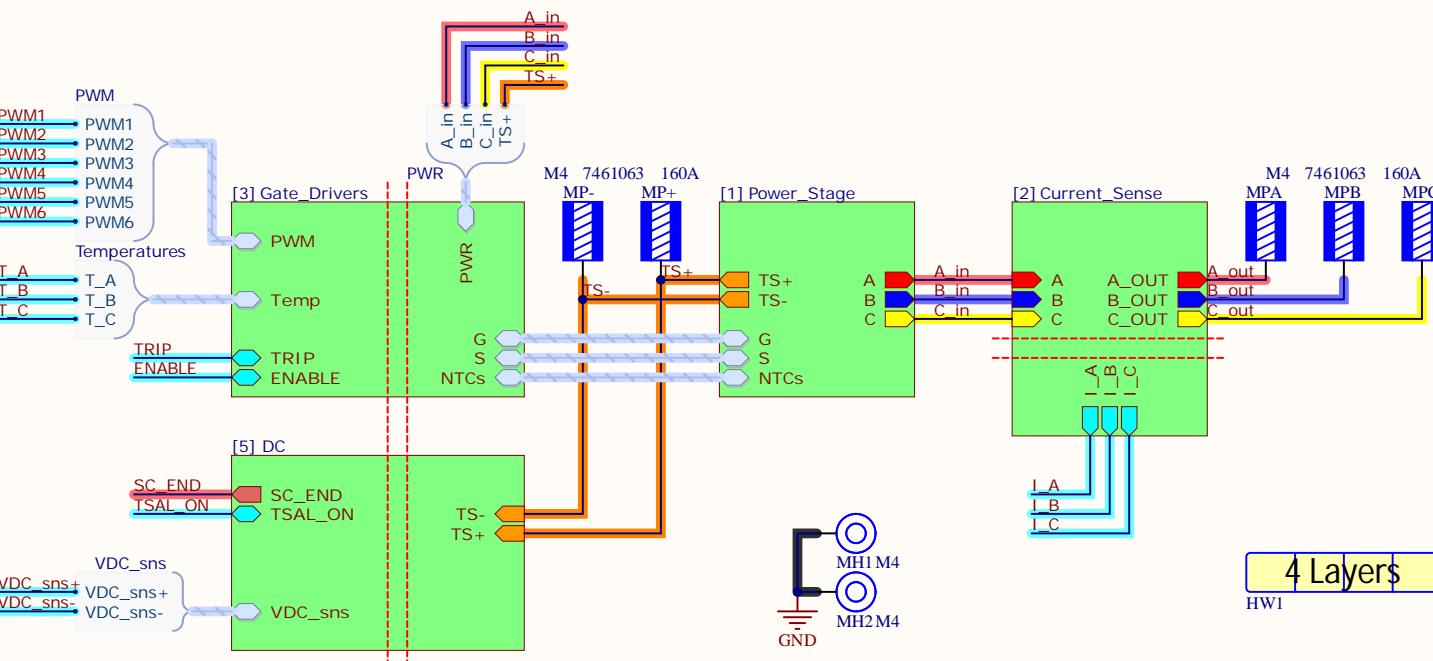
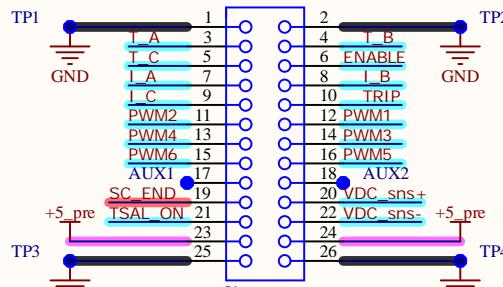
Specifications:
 $f_{sw} = 40 \text{ kHz}$
 $V_{in, max} = 600 \text{ VDC}$
 $V_{out, max} = 245 \text{ V, RMS, ph-n (SVPWM)}$
 $I_{out, max} = 80 \text{ A, RMS}$
 $I_{out, cont} = 32 \text{ A, RMS}$
 $P_{out, max} = 53 \text{ kW}$
 $P_{out, cont} = 23.5 \text{ kW}$
Liquid cooled with water at 50°C max

Changelog:
Version 1.0: (sent to production 15-02-2024)
- Base version
Version 1.1: (sent to production 28-03-2024)
- Added 5V supply protections
- Swapped pins 4 and 5 in gate drivers' LDOs
- Swapped MP+ and MP-, and their silkscreen
- Added testpoints for current sensors' reference
- Added testpoints for VDC_sns+ and VDC_sns-
- Renamed testpoints in [3]
- Added various silkscreen texts and indications
- Added layer physical logo

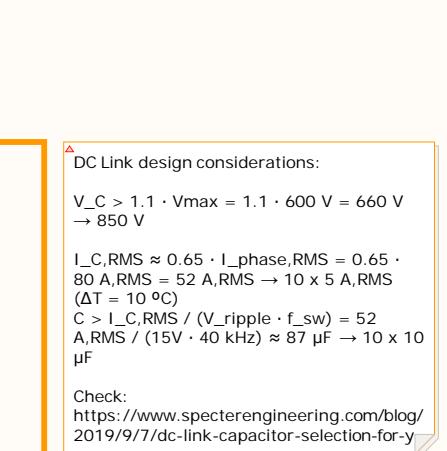
Changes in SCH but not in PCB:
19-04-2024: Added decoupling capacitors to

Known issues:
- **D1** is not correctly rated. Replace with appropriate PMOS or beefier Schottky (beware the voltage drop)
- **Rdis** does practically nothing compared to **R504** ... **R510** and could be eliminated.
- **R301** can be of lower value.
- Most DNP caps are actually needed.
- **D501** failed once, but cause remains unknown.
- **R501** should have a bigger value.
- **R511** and **R512** should have a small tolerance, and it should be specified in the schematic.
- 10uF 50V 0805 caps are expensive and should be replaced with 10uF 50V 1206 or 10uF 50V 1210.

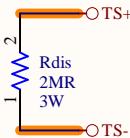
LV Connector



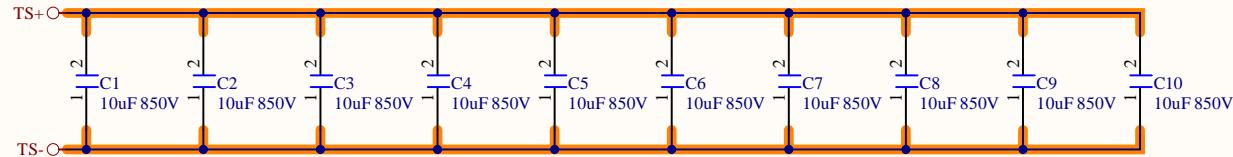
A



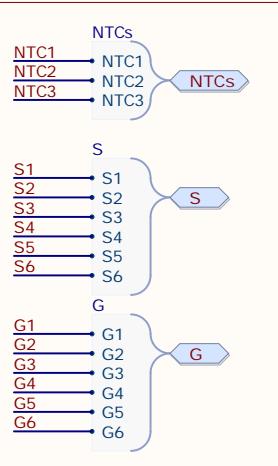
Passive discharge



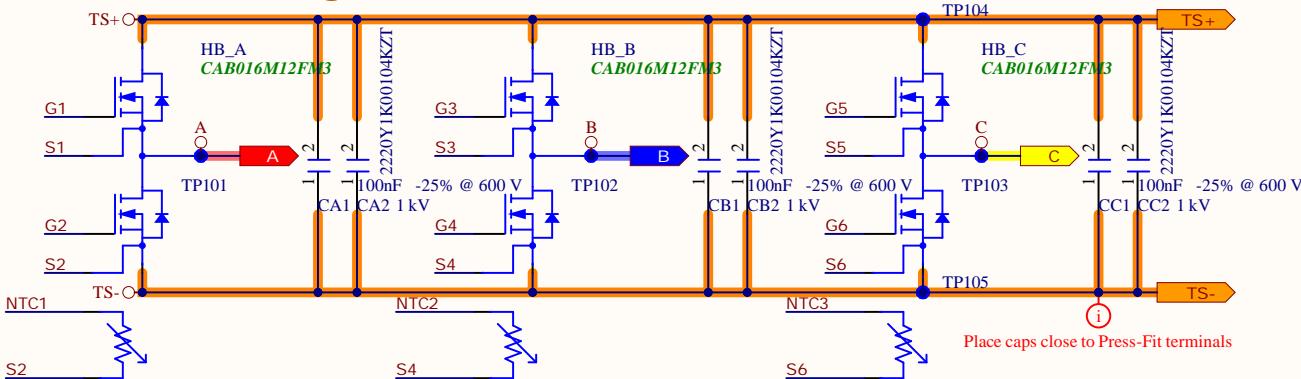
DC Bus capacitors, 100uF, Murata FHA85Y106KS



INPUTS/OUTPUTS



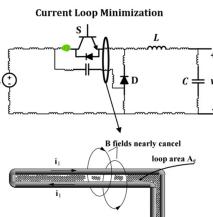
SiC Half-Bridges



Semiconductor details:

$V_{DSS}(\text{breakdown}) = 1200 V // 1200 V$
 $R_{on} = 5.5 .. 13 m\Omega // 16.0 .. 28.8 m\Omega$
 $V_{f,D} = 3.3 .. 4 V // 4.9 .. 5.5 V$
 $T_{rr} = 41.5 .. 45 ns // 20.0 ns$
 $Q_{rr} = 2.19 .. 3.94 \mu C // 1.30 \mu C$
 $R_{th,jc} = 0.12 .. 0.15 K/W // 0.543 K/W$
 $Q_G = 520 nC / 236 nC$
 $C_{in} = 14.5 nF // 6.6 nF$
 $R_G(\text{int}) = 1.9 \Omega // 2.4 \Omega$
 $V_{GS(th)} = 2.8 .. 4.8 V // 1.8 .. 3.6 V$

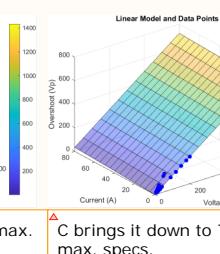
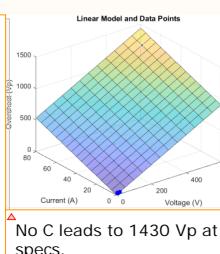
B



Current Loop Minimization

Current loop between top and bottom MOSFETs will cause excessive overshoot due to parasitic inductance and low parasitic capacitance. Increasing capacitance to hundreds of nF mitigates the effect. The capacitors essentially work as a switching decoupling. MLCCs with low DC bias or film, but MLCCs are way more compact. Place as close as possible to semiconductor terminals.

Overshoot analysis can be performed using a linear model proportional to DC bus voltage and output current. Easiest way to do it is using only one half bridge as a synchronous buck and varying input voltage with a fixed duty cycle and a R or RL load.



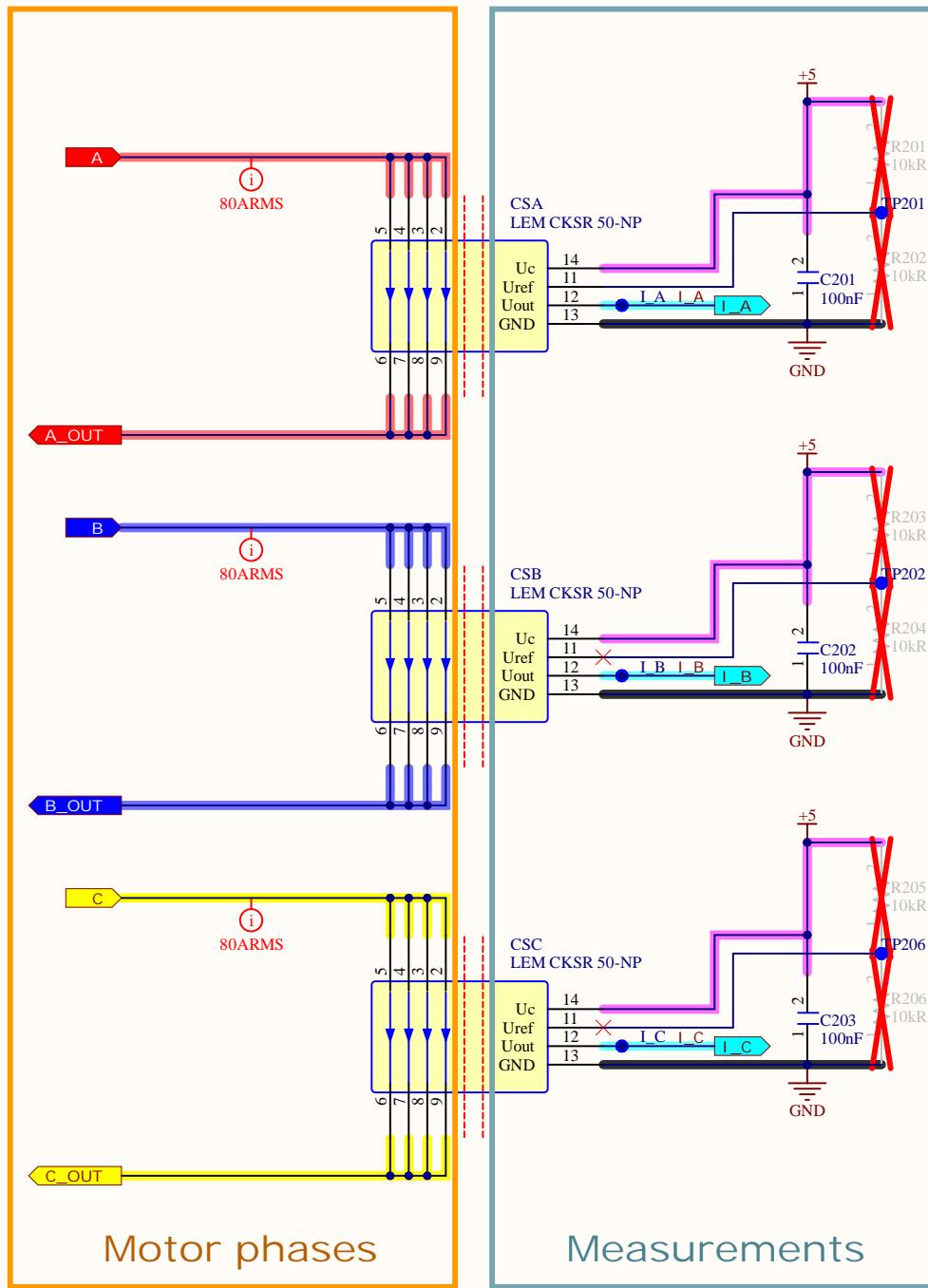
No C leads to 1430 Vp at max. specs.

C brings it down to 720 Vp at max. specs.

Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Power	Variant: Wolfspeed	
Size:	Page Contents:	[1]Power_Stage.SchDoc	Version: 1.1
Department:	Powetrain		
Author:	David Redondo	dredondovinolo@gmail.com	Sheet 2 of 5
Checked by:			Date: 20/05/2024

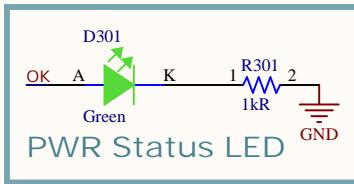
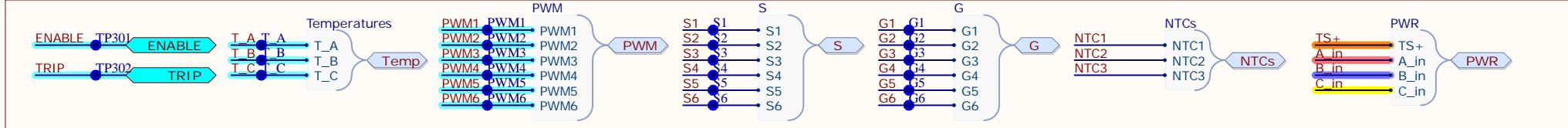
C

D



Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Power	Variant: Wolfspeed	
Size: -	Page Contents: [2]Current_Sense.SchDoc	Version: 1.1	
		Department:	Powertrain
Author:	David Redondo	dredondovinolo@gmail.com	Sheet 3 of 5
Checked by:	-	Date:	20/05/2024

INPUTS/OUTPUTS



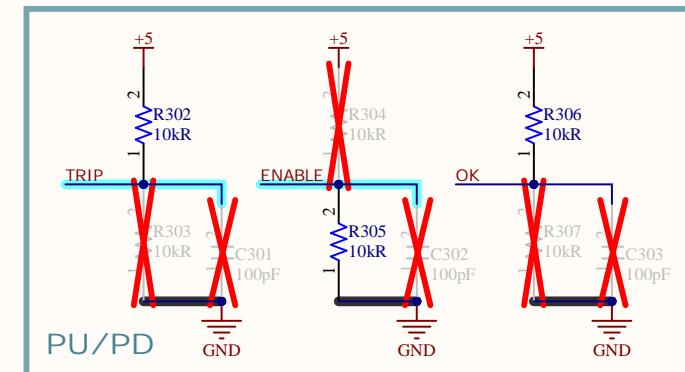
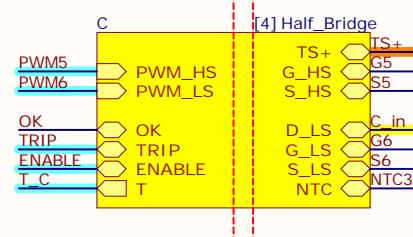
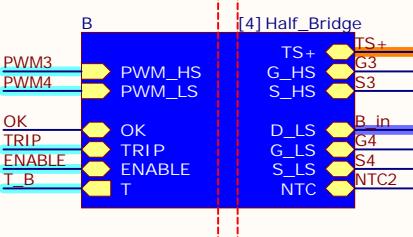
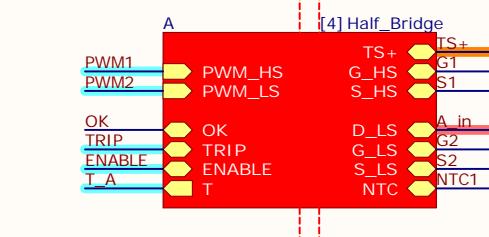
T_A, T_B, T_C

Look-up table obtained with MATLAB script which can be found in the simulations folder.

For different temperatures:
 $V_{meas}(0^\circ C) = 0.246V$
 $V_{meas}(25^\circ C) = 2V$
 $V_{meas}(50^\circ C) = 2.578V$
 $V_{meas}(90^\circ C) = 2.864V$

B

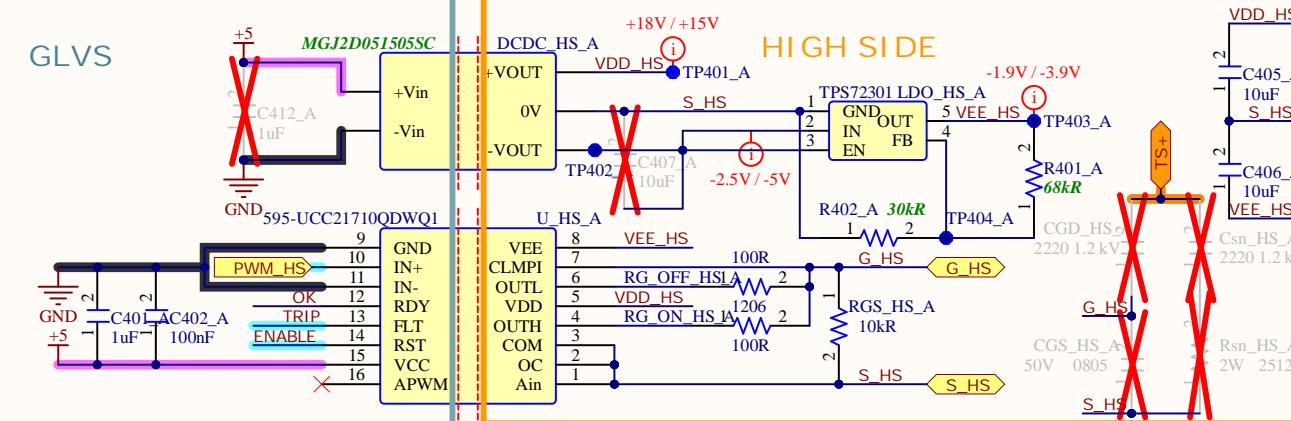
B



Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Power	Variant: Wolfspeed	
Size:	Page Contents: [3]Gate_Drivers.SchDoc	Version: 1.1	
		Department: Powertrain	
Author:	David Redondo	dredondovinolo@gmail.com	Sheet 4 of 5
Checked by:			Date: 20/05/2024

A [U_HS], [U_LS]

- [TRIP] and [OK] signals are in open drain configuration, so they can be paralleled.
- IN- is not used and tied to GND.
- [ENABLE] to be given by MCU in active-high mode. When set to low for more than 1 μ s, [TRIP] is reset.
- Temperature sensing using low-side drivers. Ain outputs a current of 200 μ A. PWM to analog using a RC filter, to be fed directly to MCU ADC. [R405], [R406] and [C411] from SPICE simulation.
- Miller clamp protection is used.
- [RGS_HS], [RGS_LS]: External gate pull-down is implemented even though the gate drivers implement an active pull-down.
- Overcurrent detection is not implemented.

GLVS**V_GS values:**

The values can be modified by replacing [DCDC_HS] and [DCDC_LS] with one from the following list: MGJ2D051505SC, MGJ2D051509SC, MGJ2D051515SC, MGJ2D051802SC, MGJ2D052003SC, MGJ2D052005SC. LDO voltages must also be adjusted.

Minimum gate driver current and power:
 $I_{GD(min)} = f_{sw} \cdot Q_G = 40 \text{ kHz} \cdot 520 \text{ nC} = 20.8 \text{ mA}$
 $P_{min} = \Delta V_{GS} \cdot I_{GD(min)} = 20 \text{ V} \cdot 20.8 \text{ mA} = 0.416 \text{ W} \rightarrow 2 \text{ W}$

B [LDO_HS], [LDO_LS]

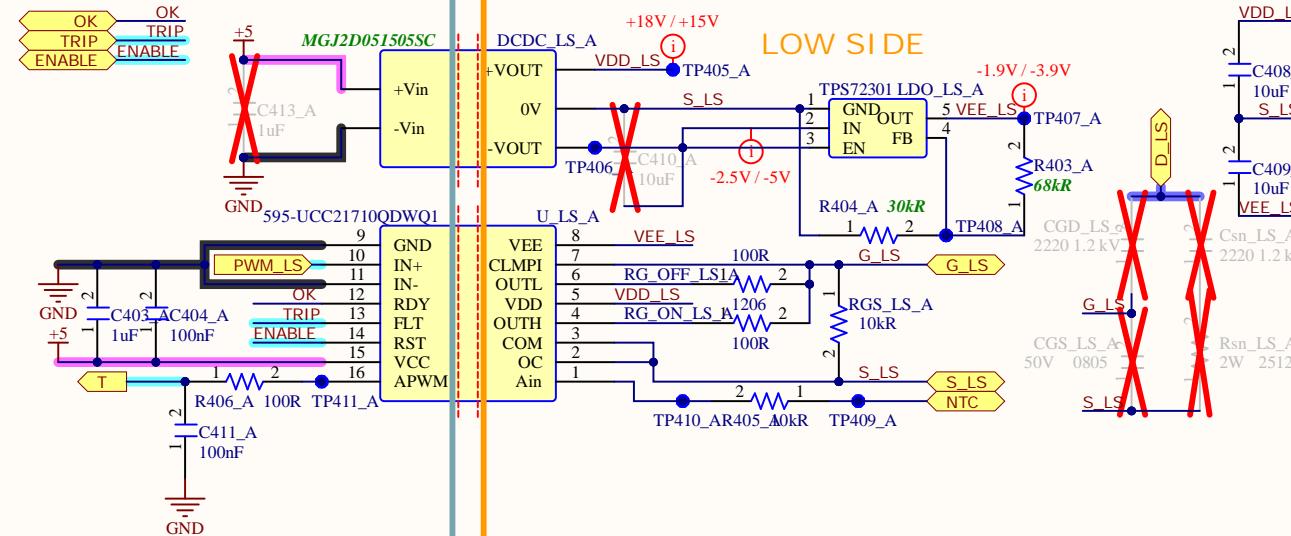
An LDO is implemented to trim [VEE_HS_A] and [VEE_LS_A] during testing to fine tune the necessary negative gate voltage. Feedback voltage divider adjusted with a Python script which can be found in the simulations folder.

$$VEE = -1.186 \cdot (1 + R1/R2)$$

$$R1 + R2 \approx 100 \text{ k}\Omega$$

Leapers $\rightarrow R1 = 36 \text{ k}\Omega$, $R2 = 56 \text{ k}\Omega$

Wolfspeed $\rightarrow R1 = 68 \text{ k}\Omega$, $R2 = 30 \text{ k}\Omega$

OK, TRIP, ENABLE**[RG_ON_HS], [RG_OFF_HS], [RG_ON_LS], [RG_OFF_LS]**

Essentially, a lower value for the gate resistors will reduce switching losses as the MOSFETs will switch faster and thus spend less time switching. Switching faster also means that the dV/dt will be higher, which can be responsible of EMI increase. The considered values of 3.3 Ω are recommended by the datasheet.

C [DCDC_HS], [DCDC_LS]

Isolation test voltage (Qualification tested for 1 minute): 5200 VDC

D [U_HS], [U_LS]

VIOTM ($t = 60 \text{ s}$ (qualification test)): 8000 VPK

[CGS_HS], [CGS_LS], [CGD_HS], [CGD_LS], [Csn_HS], [Csn_LS], [Rsn_HS], [Rsn_LS]

DNP, but they could be useful with EMI related issues to decrease dV/dt . Implementing them could result in further issues with the power limit for [DCDC_HS] and [DCDC_LS], as the gate charge would increase significantly. The maximum allowed capacitance would be:

$$CGS_{max} = 2 \cdot P_{DCDC} / (\Delta V_{GS}^2 \cdot f_{sw}) = 2 \cdot 2 \text{ W} / ((20 \text{ V})^2 \cdot 40 \text{ kHz}) = 250 \text{ nF}$$

Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Power	Variant: Wolfspeed	
Size:	Page Contents: [4]Half_Bridge.SchDoc	Version: 1.1	
		Department: Powertrain	
Author:	David Redondo	dredondovinolo@gmail.com	Sheet 5 of 5
Checked by:			Date: 20/05/2024

U_HS , U_LS

- TRIP** and **OK** signals are in open drain configuration, so they can be paralleled.
 - IN- is not used and tied to **GND**.
 - ENABLE** to be given by MCU in active-high mode. When set to low for more than 1 μ s, **TRIP** is reset.
 - Temperature sensing using low-side drivers. Ain outputs a current of 200 μ A. PWM to analog using a RC filter, to be fed directly to MCU ADC. **R405**, **R406** and **C411** from SPICE simulation.
 - Miller clamp protection is used.
 - RGS_HS**, **RGS_LS**: External gate pull-down is implemented even though the gate drivers implement an active pull-down.
 - Overshoot detection is not implemented.

△ LDO_HS , LDO_LS

An LDO is implemented to trim [VEE_HS_A](#) and [VEE_LS_A](#) during testing to fine tune the necessary negative gate voltage. Feedback voltage divider adjusted with a Python script which can be found in the simulations folder.

$$VEE = -1.186 \cdot (1 + R1/R2)$$

$$R_1 + R_2 \approx 100 \text{ k}\Omega$$

Leapers $\rightarrow R_1 = 36 \text{ k}\Omega$, $R_2 = 56 \text{ k}\Omega$

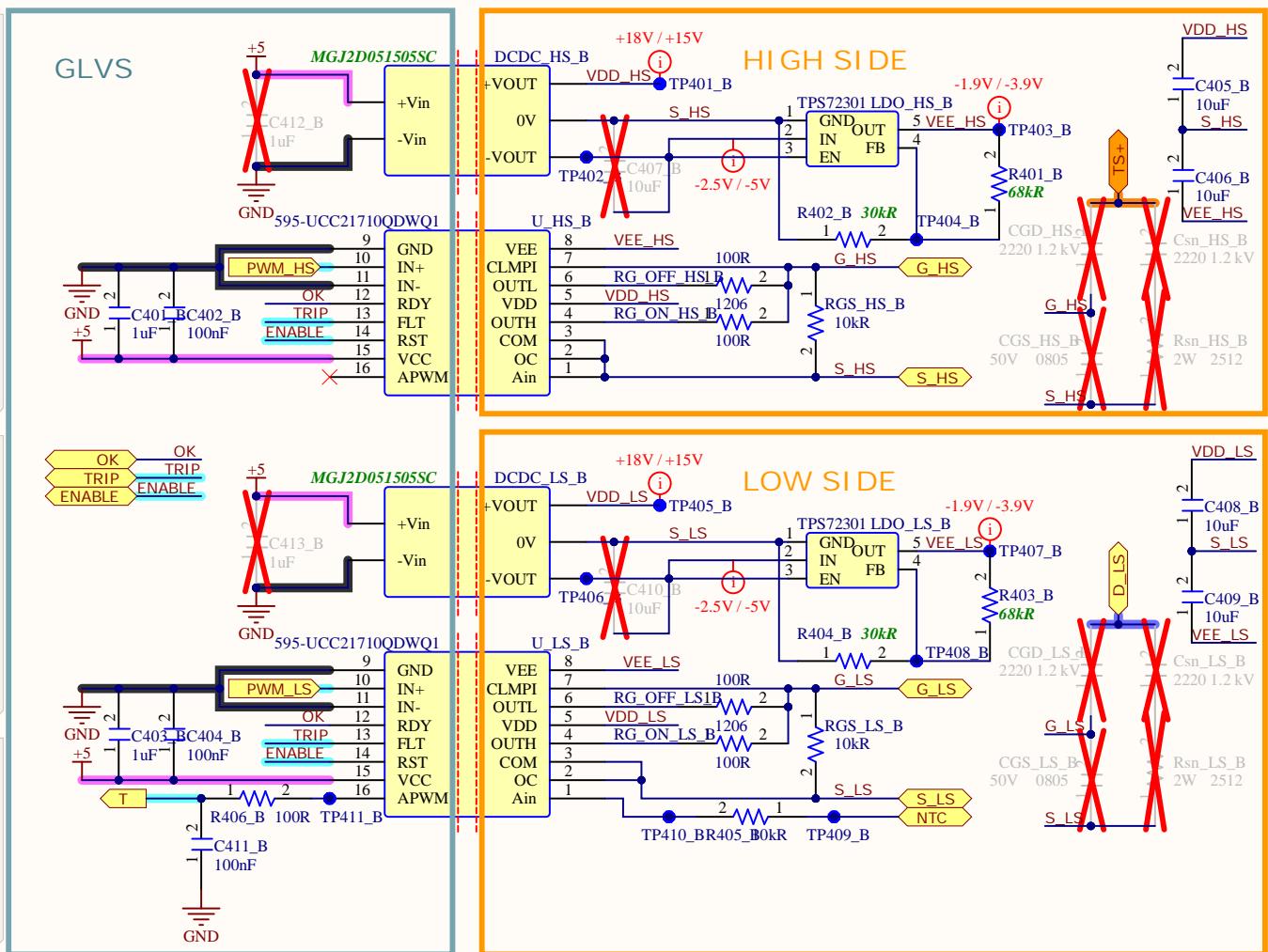
Wolfspeed $\rightarrow R_1 = 68 \text{ k}\Omega$, $R_2 = 30 \text{ k}\Omega$

DCDC HS DCDC LS

Isolation test voltage (Qualification tested for 1 minute): 5200 VDC

U_HS, U_LS

VIOTM ($t = 60$ s (qualification test))
8000 VPK



⚠ V_{GS} values:

The values can be modified by replacing [DCDC_HS](#) and [DCDC_LS](#) with one from the following list: MGJ2D051505SC, MGJ2D051509SC, MGJ2D051515SC, MGJ2D051802SC, MGJ2D052003SC, MGJ2D052005SC. LDO voltages must also be adjusted.

$$\begin{aligned} & \text{Minimum gate driver current and power:} \\ & I_{GD(\min)} = f_{sw} \cdot Q_G = 40 \text{ kHz} \cdot 520 \text{ nC} = \\ & 20.8 \text{ mA} \\ & P_{\min} = \Delta V_{GS} \cdot I_{GD(\min)} = 20 \text{ V} \cdot 20.8 \\ & \text{mA} = 0.416 \text{ W} \rightarrow 2 \text{ W} \end{aligned}$$

RG_ON_HS, RG_OFF_HS
RG_ON_LS, RG_OFF_LS

Essentially, a lower value for the gate resistors will reduce switching losses as the MOSFETs will switch faster and thus spend less time switching. Switching faster also means that the dV/dt will be higher, which can be responsible of EMI increase. The considered values of 3.3 Ω are recommended by the datasheet.

`CGS_HS`, `CGS_LS`, `CGD_HS`, `CGD_LS`,
`Csn_HS`, `Csn_LS`, `Rsn_HS`, `Rsn_LS`

DNP, but they could be useful with EMI related issues to decrease dV/dt. Implementing them could result in further issues with the power limit for [DCDC_HS](#) and [DCDC_LS](#), as the gate charge would increase significantly. The maximum allowed capacitance would be:

$$\text{CGS}_{\text{max}} = 2 \cdot P_{\text{DCDC}} / (\Delta V_{\text{GS}}^2 \cdot f_{\text{sw}})$$

Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Power	Variant: Wolfspeed	
Size: -	Page Contents: [4]Half_Bridge.SchDoc	Version: 1.1	
		Department:	Powertrain
Author:	David Redondo	dredondovinolo@gmail.com	Sheet 5 of 5
Checked by:			Date: 20/05/2024

- 1. **TRIP** and **OK** signals are in open drain configuration, so they can be paralleled.
- 2. IN- is not used and tied to **GND**.
- 3. **ENABLE** to be given by MCU in active-high mode. When set to low for more than 1 μ s, **TRIP** is reset.
- 4. Temperature sensing using low-side drivers. AIN outputs a current of 200 μ A. PWM to analog using a RC filter, to be fed directly to MCU ADC. **R405**, **R406** and **C411** from SPICE simulation.
- 5. Miller clamp protection is used.
- 6. **RGS_HS**, **RGS_LS**: External gate pull-down is implemented even though the gate drivers implement an active pull-down.
- 7. Overcurrent detection is not implemented.

An LDO is implemented to trim `VEE_HS_A` and `VEE_LS_A` during testing to fine tune the necessary negative gate voltage. Feedback voltage divider adjusted with a Python script which can be found in the simulations folder.

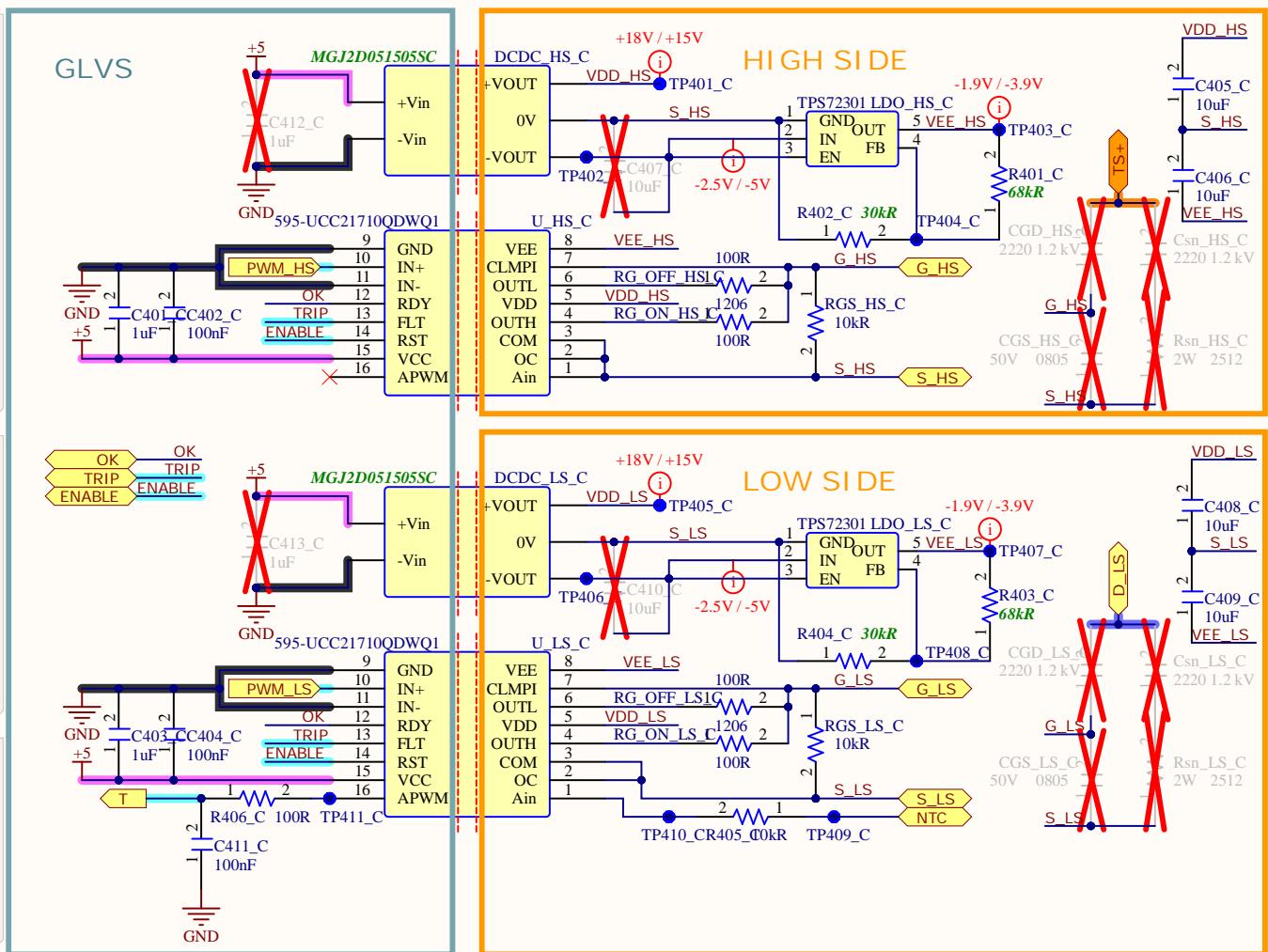
$$VEE = -1.186 \cdot (1 + R1/R2)$$

$$R_1 + R_2 \approx 100 \text{ k}\Omega$$

Leapers $\rightarrow R_1 = 36 \text{ k}\Omega$, $R_2 = 56 \text{ k}\Omega$
 Wolfspeed $\rightarrow R_1 = 68 \text{ k}\Omega$, $R_2 = 30 \text{ k}\Omega$

DCDC_HS, **DCDC_LS**
Isolation test voltage (Qualification tested for 1 minute): 5200 VDC

U_HS, U_LS
VIOTM ($t = 60$)
8000 VPK



V. CS values:

The values can be modified by replacing **DCDC_HS** and **DCDC_LS** with one from the following list: MGJ2D051505SC, MGJ2D051509SC, MGJ2D051515SC, MGJ2D051802SC, MGJ2D052003SC, MGJ2D052005SC. LDO voltages must also be adjusted.

$$\begin{aligned} & \text{Minimum gate driver current and power:} \\ & I_{GD(\min)} = f_{sw} \cdot Q_G = 40 \text{ kHz} \cdot 520 \text{ nC} = \\ & 20.8 \text{ mA} \\ & P_{\min} = \Delta V_{GS} \cdot I_{GD(\min)} = 20 \text{ V} \cdot 20.8 \\ & \text{mA} = 0.416 \text{ W} \rightarrow 2 \text{ W} \end{aligned}$$

RG_ON_HS, RG_OFF_HS
RG_ON_LS, RG_OFF_LS

Essentially, a lower value for the gate resistors will reduce switching losses as the MOSFETs will switch faster and thus spend less time switching. Switching faster also means that the dV/dt will be higher, which can be responsible of EMI increase. The considered values of 3.3 Ω are recommended by the datasheet.

`CGS_HS`, `CGS_LS`, `CGD_HS`, `CGD_LS`,
`Csn_HS`, `Csn_LS`, `Rsn_HS`, `Rsn_LS`

DNP, but they could be useful with EMI related issues to decrease dV/dt . Implementing them could result in further issues with the power limit for [DCDC_HS](#) and [DCDC_LS](#), as the gate charge would increase significantly. The maximum allowed capacitance would be:

$$\begin{aligned} \text{CGS}_{\text{max}} &= 2 \cdot P_{\text{DCDC}} / (\Delta V_{\text{GS}}^2 \cdot f_{\text{sw}}) \\ &= 2 \cdot 2 \text{ W} / ((20 \text{ V})^2 \cdot 40 \text{ kHz}) = 250 \text{ nF} \end{aligned}$$

Company:	e-Tech Racing	e-techracing.es	
Project:	InverterPower	Variant: Wolfspeed	
Size: -	Page Contents: [4]Half_Bridge.SchDoc	Version:	1.1
		Department:	Powertrain
Author:	David Redondo	dredondovinolo@gmail.com	Sheet 5 of 5
Checked by:		Date:	20/05/2024

Discharge resistors:

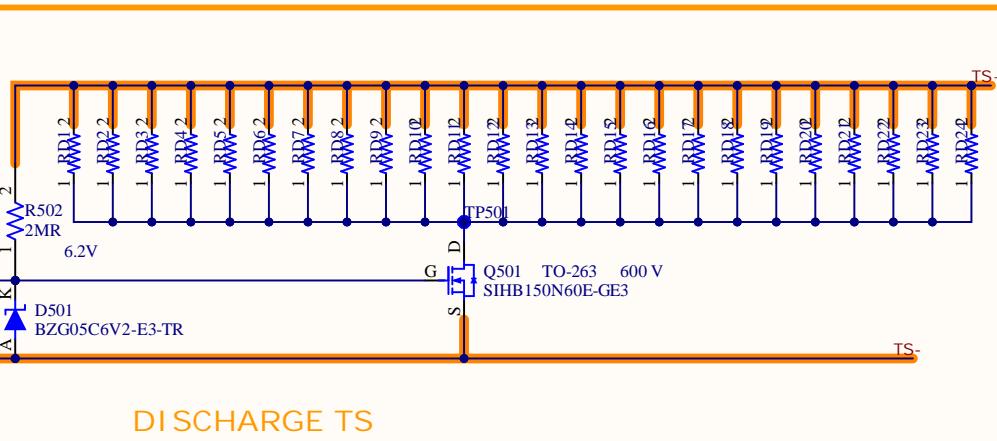
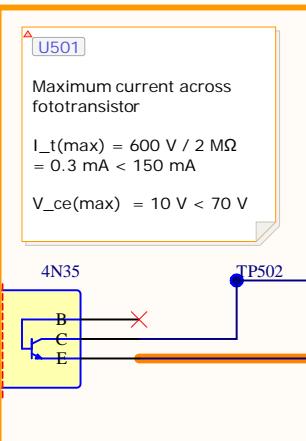
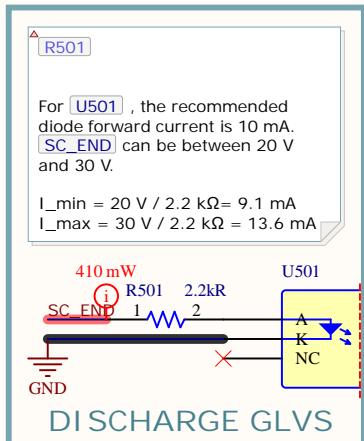
$$t_{dis} = R_{dis} \cdot C \cdot \ln(V_{initial}/V_{final}) = (470 \text{ k}\Omega / 24) \cdot (100 \mu\text{F}) \cdot \ln(600 \text{ V} / 60 \text{ V}) = 4.509 \text{ s}$$

$$P(R_{dis}, \text{max}) = V_{max}^2 / R_{dis} = 600 \text{ V}^2 / 470 \text{ k}\Omega = 0.766 \text{ W} < 1 \text{ W}$$

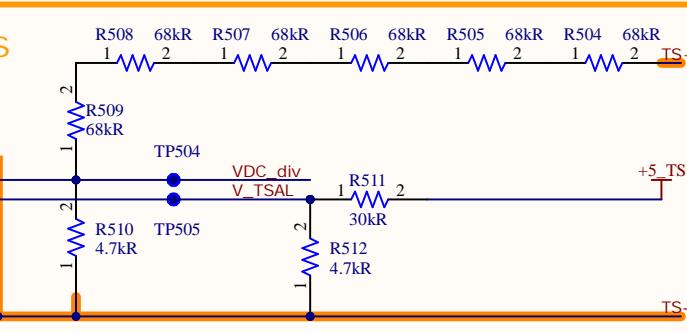
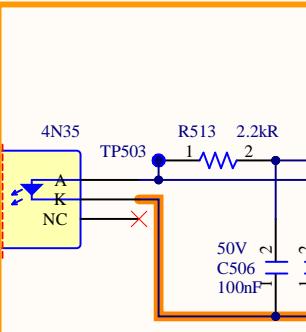
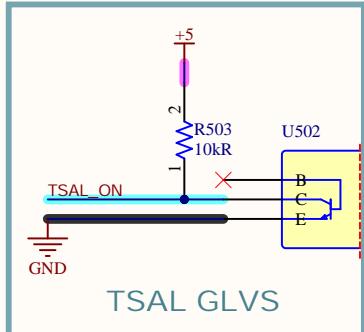
$$\Delta T \sim 110^\circ\text{C/W} \cdot 0.766 \text{ W} = 85^\circ\text{C}$$

$$I_{dis, \text{max}} = 600 \text{ V} / (470 \text{ k}\Omega / 24) = 30.64 \text{ mA}$$

U503
Single supply configuration as per datasheet.
Maximum differential input voltage = 6.833 V - 677 mV = 6.156 V < 30 V

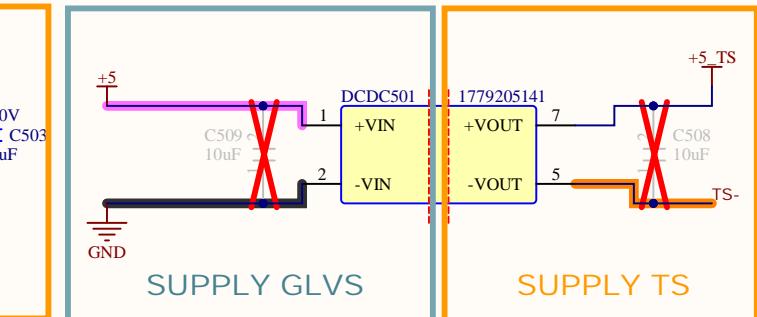
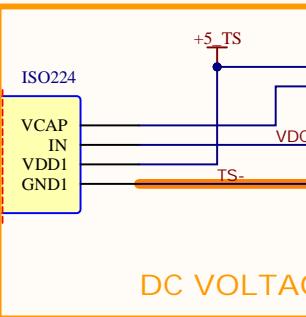
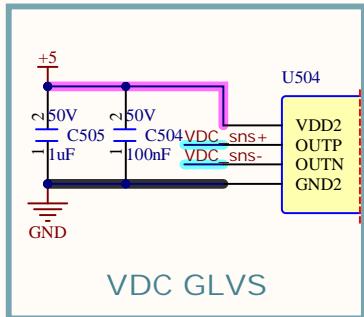


VDC_div = $(TS+ - TS-) \cdot 4.7 \text{ k}\Omega / (4.7 \text{ k}\Omega + 6 \cdot 68 \text{ k}\Omega)$
 $600 \text{ V} \cdot 4.7 \text{ k}\Omega / (4.7 \text{ k}\Omega + 6 \cdot 68 \text{ k}\Omega) = 6.833 \text{ V}$
 $60 \text{ V} \cdot 4.7 \text{ k}\Omega / (4.7 \text{ k}\Omega + 6 \cdot 68 \text{ k}\Omega) = 683 \text{ mV}$
 $P_{R4} = I_{R4} \cdot R4 = ((600 \text{ V} / (4.7 \text{ k}\Omega + 6 \cdot 68 \text{ k}\Omega)) / 68 \text{ k}\Omega)^2 \cdot 68 \text{ k}\Omega = 144 \text{ mW} \rightarrow 1206 \text{ package}$
V_TSAL = $5 \text{ V} \cdot 4.7 \text{ k}\Omega / (4.7 \text{ k}\Omega + 30 \text{ k}\Omega) = 677 \text{ mV} \equiv 59.46 \text{ V in } TS+ - TS-$



INPUTS/OUTPUTS

SC-END → SC-END
TP506 → VDC_sns
VDC_sns+ → VDC_sns+ → VDC_sns
TP507 → VDC_sns- → VDC_sns- → VDC_sns
TSAL_ON → TSAL_ON
TS+ → TS+
TS- → TS-



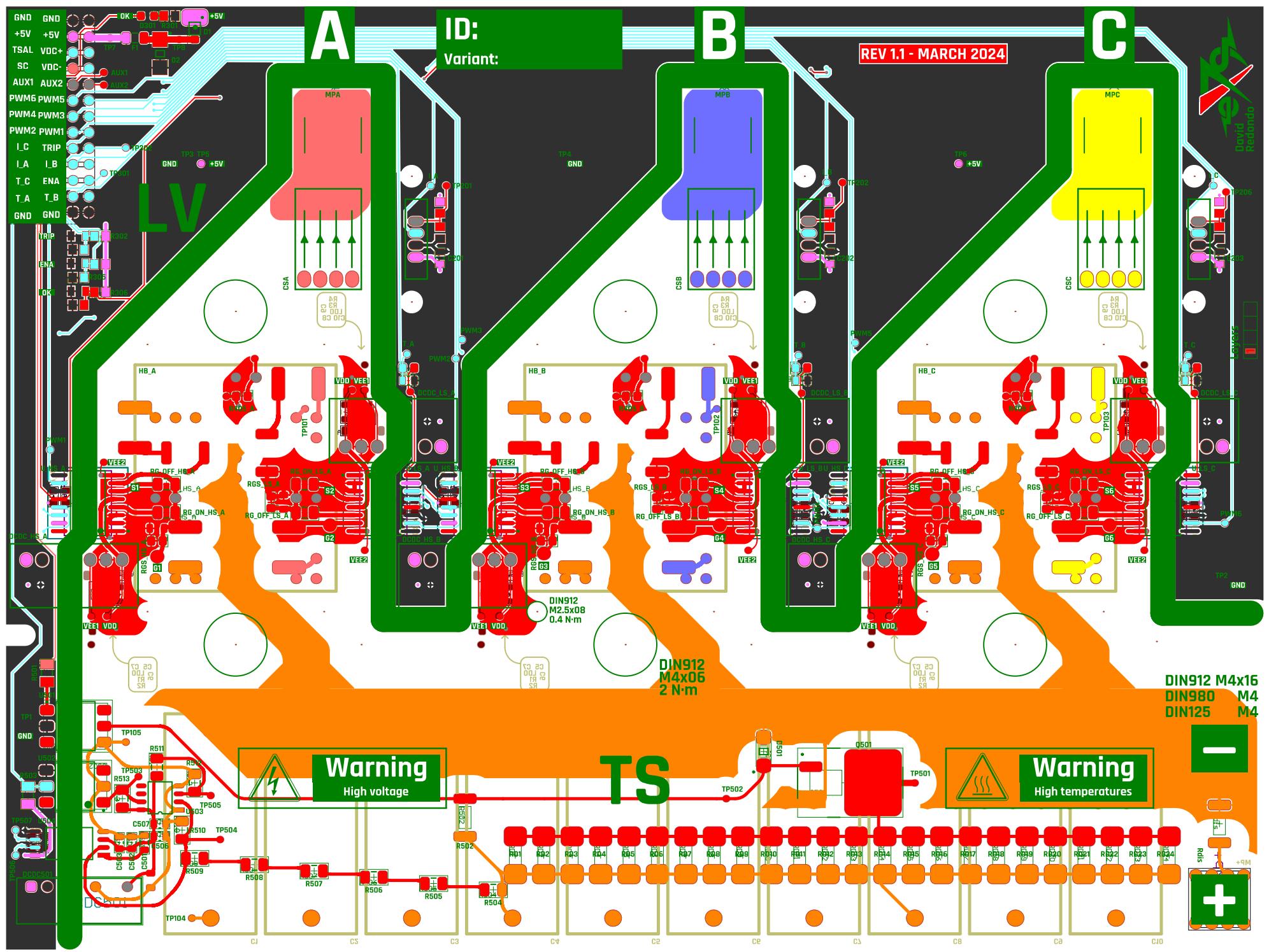
$(TS+ - TS-) > 60 \text{ V} \rightarrow TSAL_ON = 0 \text{ V}$
 $(TS+ - TS-) < 60 \text{ V} \rightarrow TSAL_ON = 5 \text{ V}$

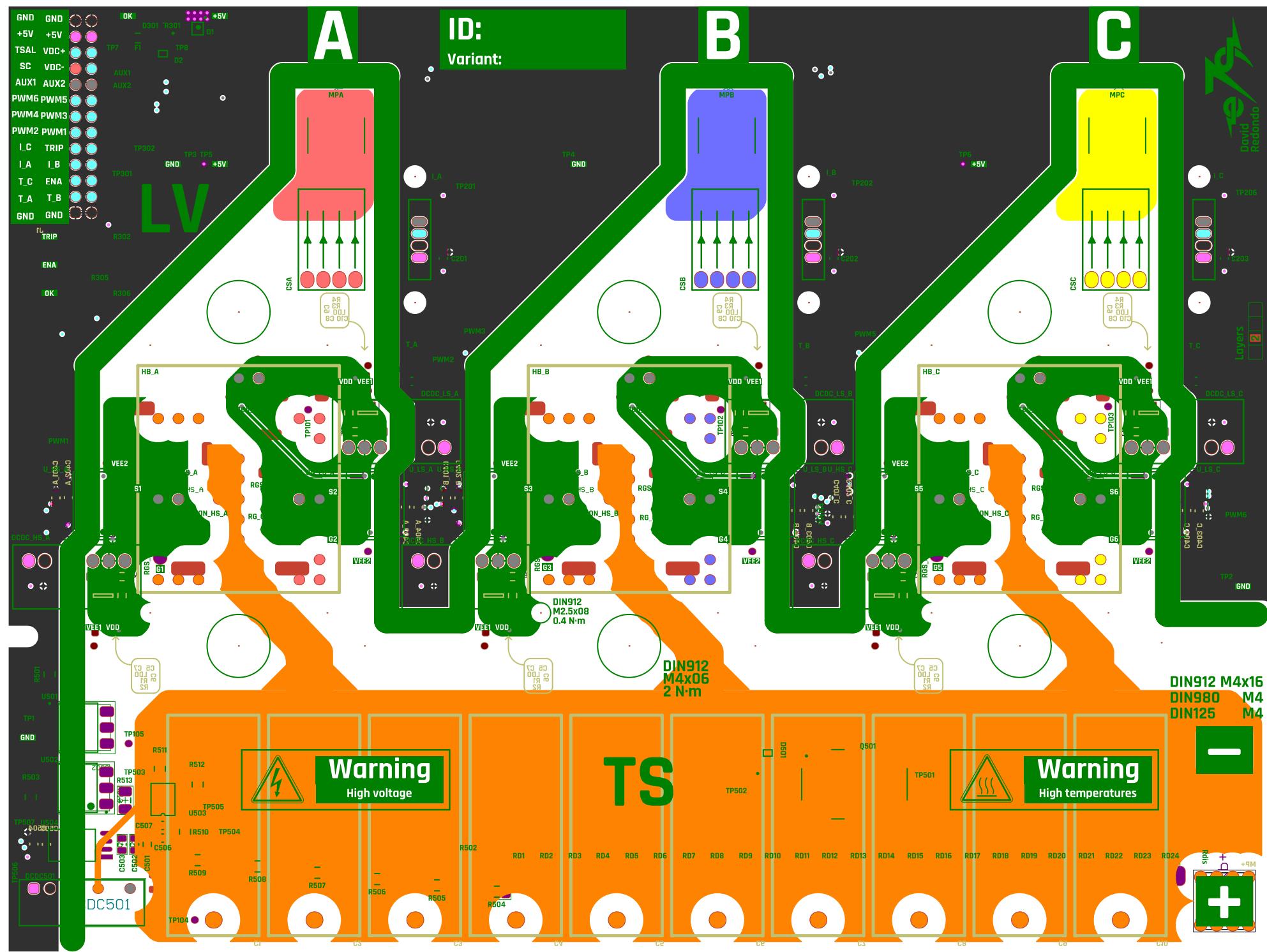
U504
 $(VDC_{sns+} - VDC_{sns-}) = 1/3 \cdot VDC_{div}$
 $= 1/3 \cdot ((TS+ - TS-)) = 1/3 \cdot (4.7 \text{ k}\Omega / (4.7 \text{ k}\Omega + 6 \cdot 68 \text{ k}\Omega)) = 1/3 \cdot 0.011388 \cdot (TS+ - TS-)$
 $(VDC_{sns+} - VDC_{sns-}) = 1/3 \cdot 0.011388 \cdot 600 \text{ V} = 2.278 \text{ V}$

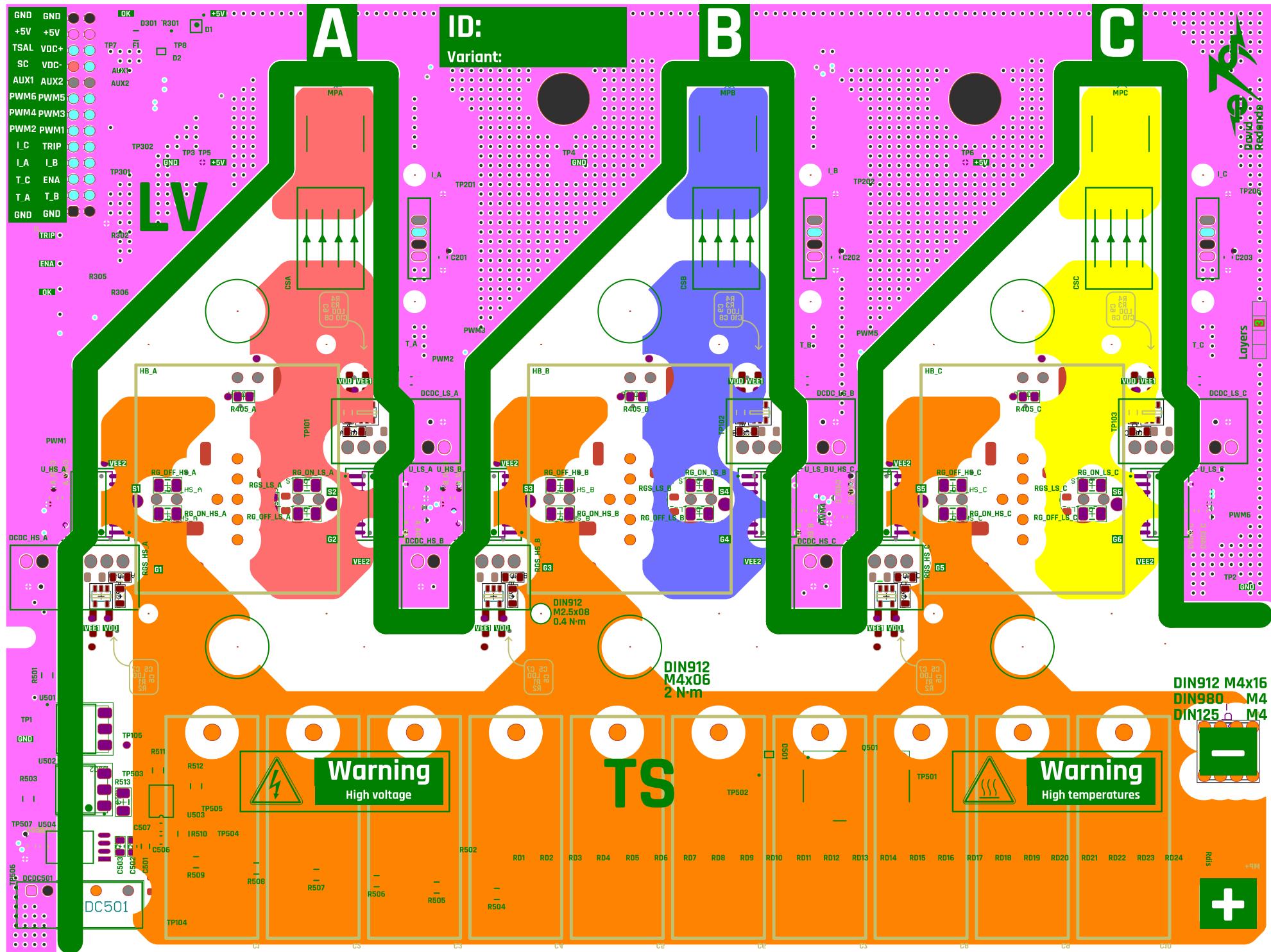
U501, U502
Isolation Voltage: AC For 1 Minute, R.H. = 40 ~ 60% Viso = 5000 Vrms
U504
Maximum transient isolation voltage: VTEST = VIOTM, t = 60 s (qualification test) VIOTM = 7071 Vpk

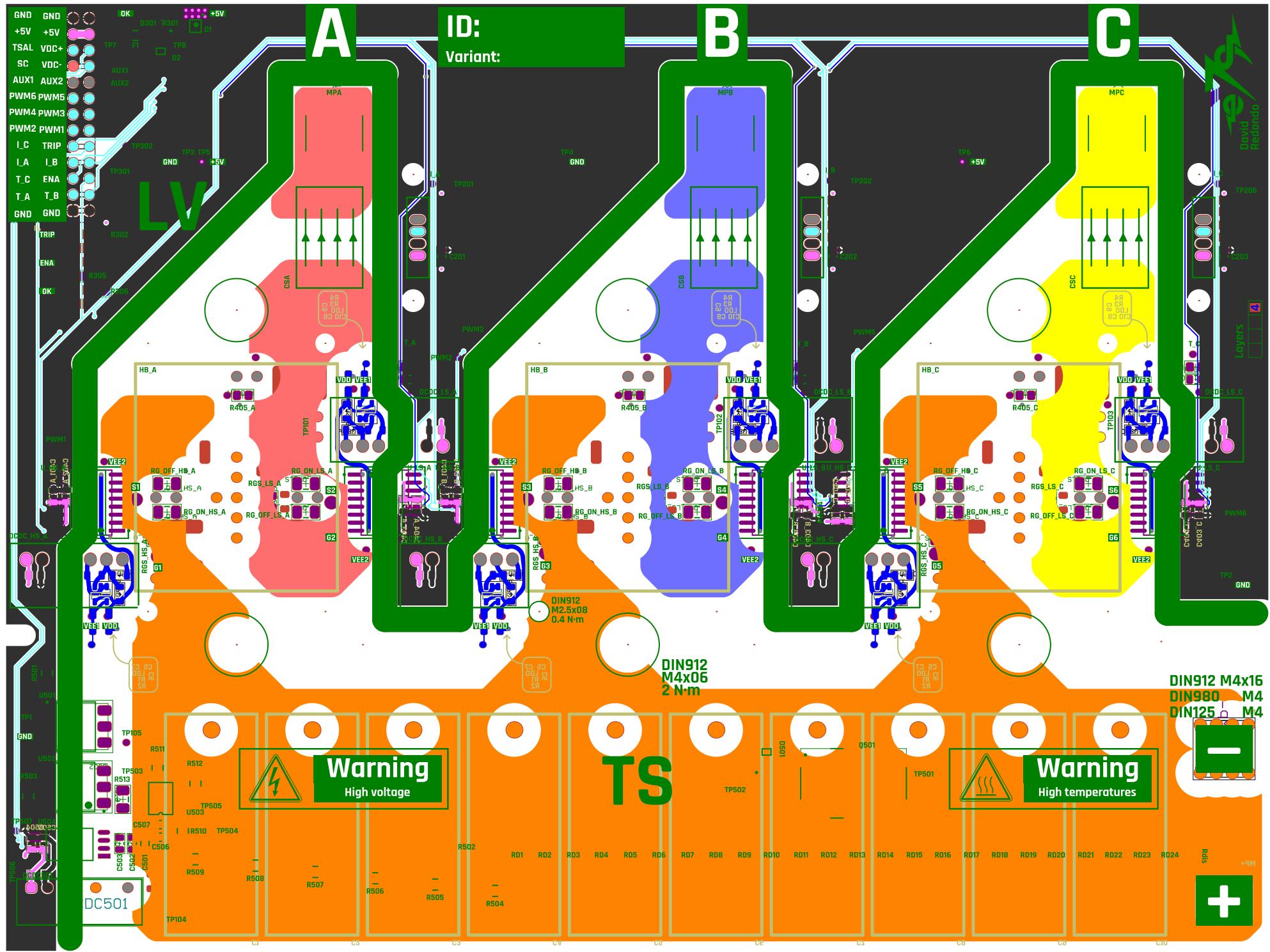
DCDC501
Isolation voltage input to output, tested 100% for 60s(2)
VISo = 3000 V

Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Power	Variant: Wolfspeed	
Size:	Page Contents: [5]DC.SchDoc	Version: 1.1	
Department:	Powertrain		
Author:	David Redondo	dredondovinolo@gmail.com	Sheet • of •
Checked by:			Date: 20/05/2024

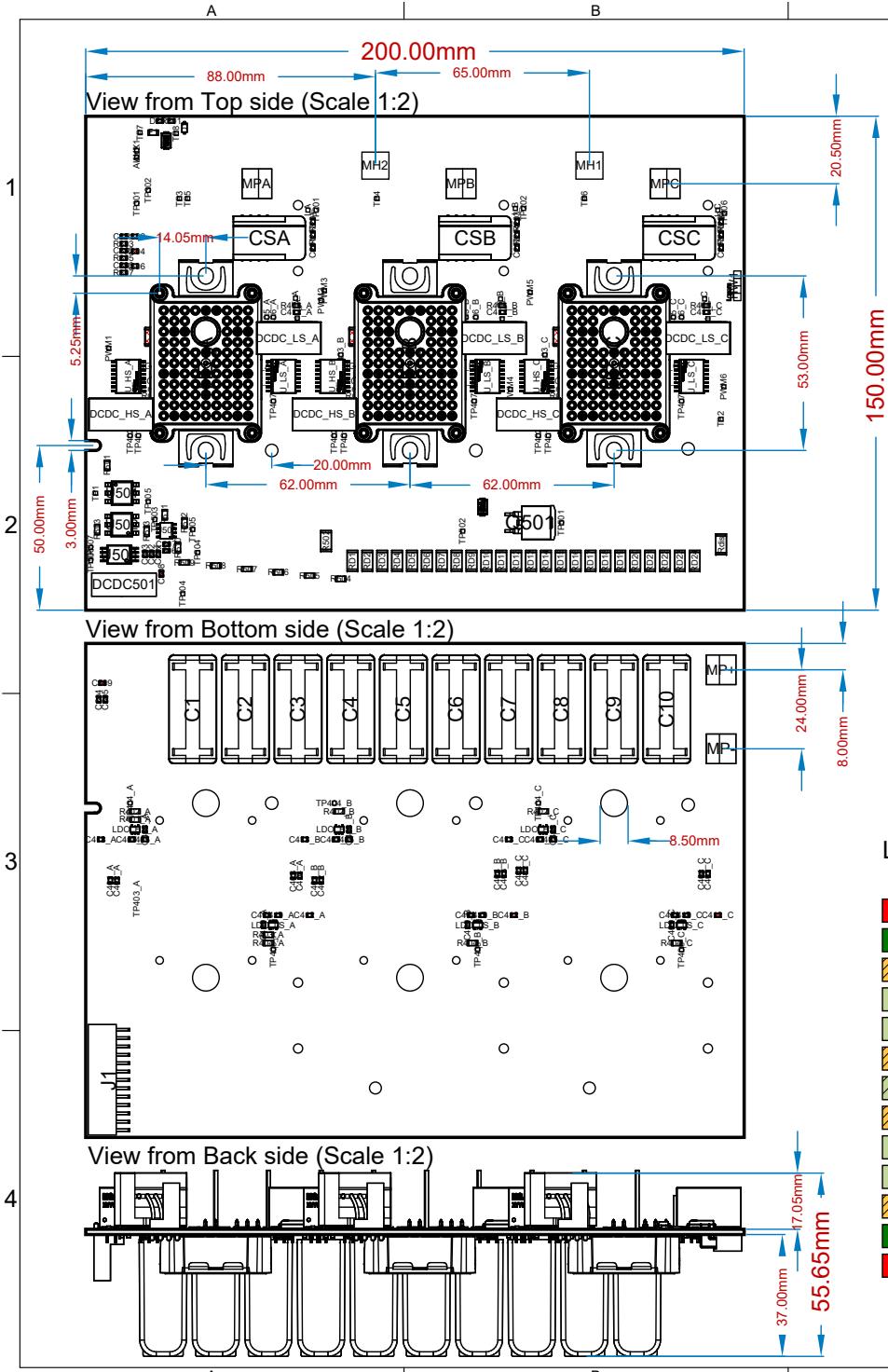








Inverter Power



Bill Of Materials

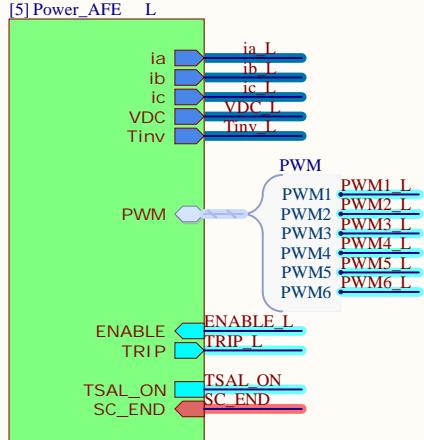
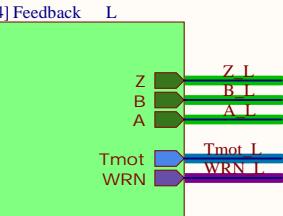
Designator	Name	Quantity
C405_A, C405_B, C405_C, C406_A, C406_B, C406_C,	10uF 850V	12
C408_A, C408_B, C408_C, C409_A, C409_B, C409_C	2220Y1K00104KZT	6
C1, C2, C3, C4, C5, C6, C7, C8, C9, C10	0437001.WRA	1
C1A, C2A, C3B, C2B, CC1, CC2	1779205141	1
F1	BZG05CSV1-E3-TR	1
DDC501	613026243121	1
J1	CA016M12FM3	3
D2	HB_A, HB_B, HB_C	1
R402_A, R402_B, R402_C, R404_A, R404_B, R404_C	CR1206-JW-303ELF	6
R401_A, R401_B, R401_C, R403_A, R403_B, R403_C	CR1206-JW-683ELF	6
R503	CRCW120610K0KFKEA	1
R511	CRCW120630K0KFKEA	1
HW1	LOGO CAPAS (4)	1
MP-, MP+, MPA, MPB, MPC	M4	5
D1	MBR0530	1
DCDC_HS_A, DDCDC_HS_B, DDCDC_HS_C, DCDC_LS_A,	MGJ6-series	6
DCDC_LS_B, DDCDC_LS_C	Mounting_Hole_M4	2
MH1, MH2	RCV2512470KFKEG	24
RD1, RD2, RD3, RD4, RD5, RD6, RD7, RD8, RD9, RD10,	RD11, RD12, RD13, RD14, RD15, RD16, RD17, RD18,	
RD19, RD20, RD21, RD22, RD23, RD24	RD406_A, R406_B, R406_C	
R301	CR0805-FX-1000ELF	3
R302, R305, R306, R405_A, R405_B, R405_C, RGS_HS_A,	CR0805-JW-102ELF	1
RGS_HS_B, RGS_HS_C, RGS_LS_A, RGS_LS_B,	CR0805-JW-103ELF	12
R504, R505, R506, R507, R508, R509	CR1206AFX-6802EAS	6
R501, R513	CR1206-FX-2201ELF	2
R510, R512	CRS1206-FX-4701ELF	2
CSA, CSB, CSC	LEM CKSR 50-NP	3
U503	LM311DR2G	1
C501	885012208058	1
RG_OFF_HS_A, RG_OFF_HS_B, RG_OFF_HS_C,	CRG1206F100R	12
RG_OFF_LS_A, RG_OFF_LS_B, RG_OFF_LS_C,	ISO224	1
RG_ON_HS_A, RG_ON_HS_B, RG_ON_HS_C,	TPS72301	6
RG_ON_LS_A, RG_ON_LS_B, RG_ON_LS_C	UCC21710	6
U504	SIHB150N60E-GE3	1
LDO_HS_A, LDO_HS_B, LDO_HS_C, LDO_LS_A,	R502, Rdis	2
LDO_LS_B, LDO_LS_C	R2M-2512FTK	2
U_HS_A, U_HS_B, U_HS_C, U_LS_A, U_LS_B, U_LS_C	U501, U502	2
Q501	4N35	2
R501, Rdis	D501	1
U502	D301	1
C201, C202, C203, C402_A, C402_B, C402_C, C404_A,	150080GS75000	1
C404_B, C404_C, C411_A, C411_B, C411_C, C502, C504,	885012207098	15
C506	885012207103	9
C401_A, C401_B, C401_C, C403_A, C403_B, C403_C,		
C503, C505, C507		

Copper thickness in hole 25-50um, watch out for 1.10mm holes
Chemical tin 1-15um

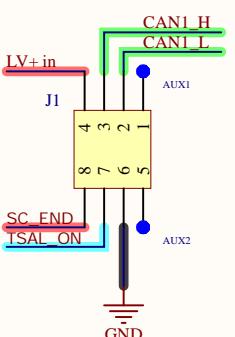
Layer Stack Legend

Material	Layer	Thickness	Dielectric Material	Type	Gerber
Top Overlay					GTO
Surface Material	Top Solder	0.01mm	Solder Resist	Solder Mask	GTS
CF-004	TOP	0.07mm		Signal	GTL
- Prepreg		0.10mm	PP-006	Dielectric	
- Preprep		0.10mm	PP-006	Dielectric	
Copper	GND	0.07mm		Signal	G1
- Preprep		0.90mm	FR-4	Dielectric	
Copper	PWR	0.07mm		Signal	G2
- Preprep		0.10mm	PP-006	Dielectric	
- Preprep		0.10mm	PP-006	Dielectric	
CF-004	BOT	0.07mm		Signal	GBL
Surface Material	Bottom Solder	0.01mm	Solder Resist	Solder Mask	GBS
Bottom Overlay				Legend	GBO

Total thickness: 1.60mm

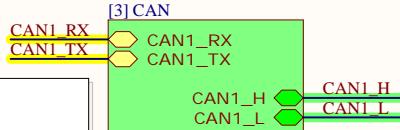
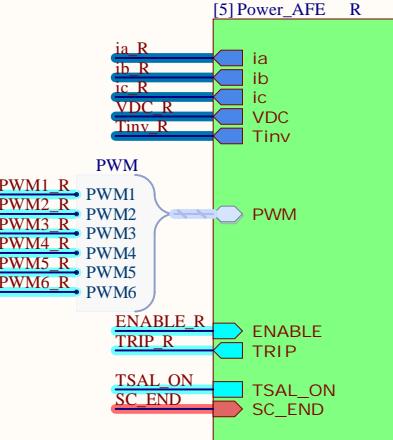
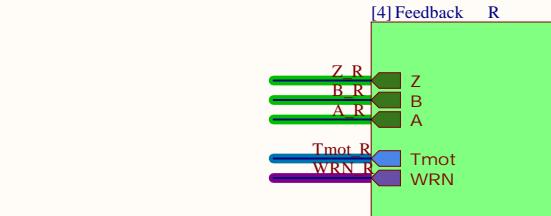
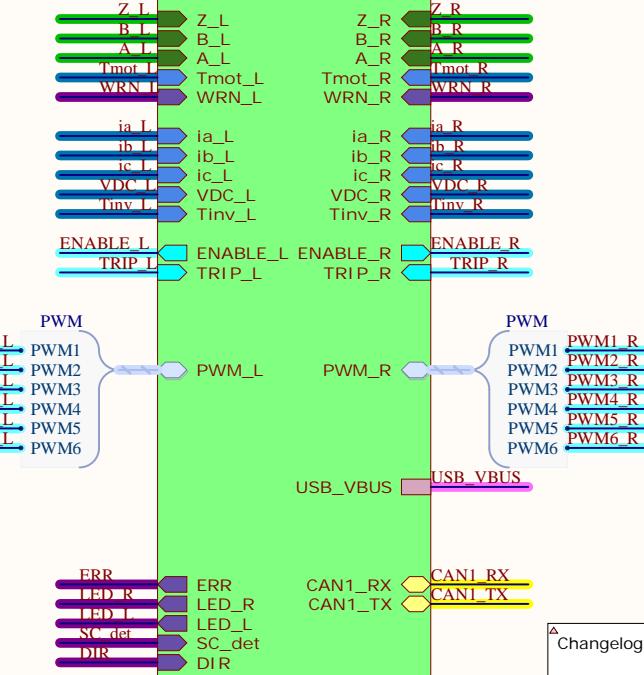
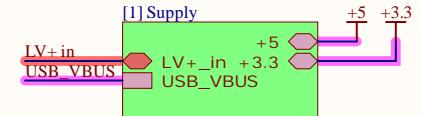


Car connector



- ① Cyan nets indicate external signals.
- ① Purple nets indicate internal 3.3V signals.
- ① Blue nets indicate analog signals read by the ADC.
- ① Red nets indicate 20-30V.
- ① Pink nets indicate treated supply.
- ① Light green nets indicate CAN.
- ① Yellow nets indicate serial communication.
- ① Dark green nets indicate input capture.

- Known issues:**
- DDCDC101 not tested yet.
 - L101 has a very low current rating. Choose new part with a rating of at least 3A.
 - Q101 and USB in general not tested yet.
 - U501 not working as expected.
 - U201 dot in silkscreen might be misleading. Check the orientation very carefully.
 - 3.3V OVP should be added to all non-5V-tolerant MCU pins.
 - Most if not all 15R resistors could very well be OR or solder bridges.
 - 10uF 50V 0805 caps are expensive and should be replaced with 10uF 50V 1206 or 10uF 50V 1210.



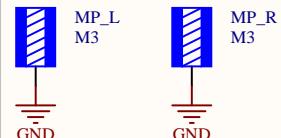
Changelog:

Version 1.0:

- Base version, sent to production 21-02-2024

Version 1.1:

- Pull-up and pull-down added for BOOT0 control
- Added I2C pull-ups
- LV+_sns deleted, VBAT connected to 3.3V
- Extras LED color and names changed
- Added layers physical logo
- Moved supply filter to +5V
- Some silkscreen

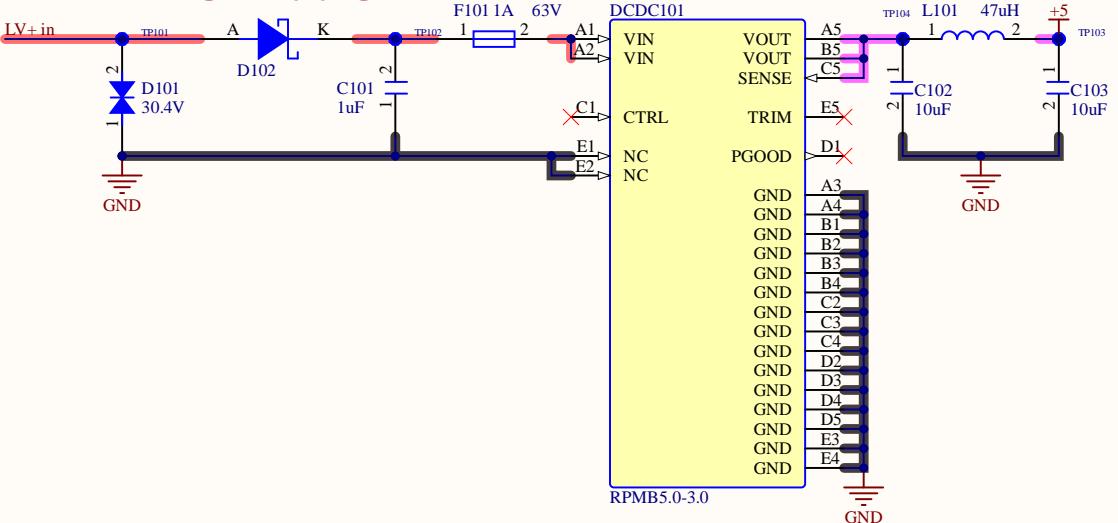


4 Layers

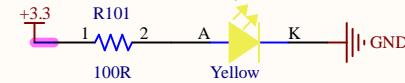
HW1

Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Control	Variant: [No Variations]	
Size:	Page Contents: Inverter_Control.SchDoc	Version: 1.1	
		Department: Powertrain	
Author:	David Redondo	dredondovinolo@gmail.com	Sheet 1 of 1
Checked by:		Date: 20/05/2024	

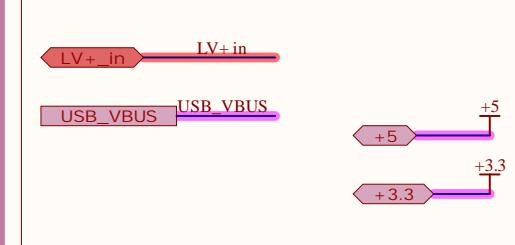
LV battery supply



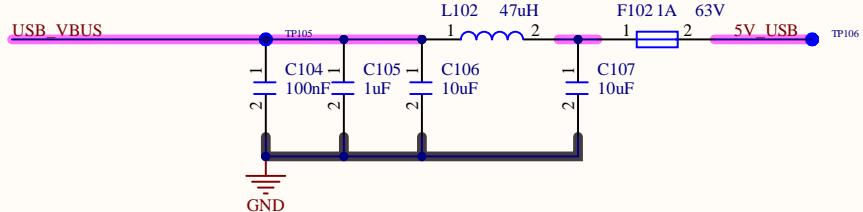
Supply OK



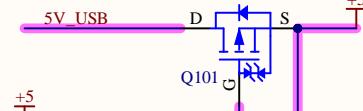
INPUTS/OUTPUTS



USB supply

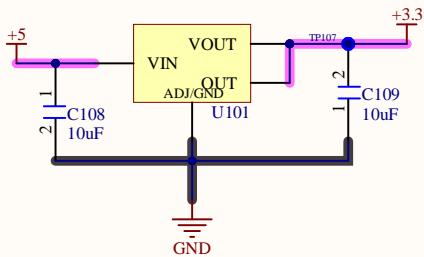


5 V selection

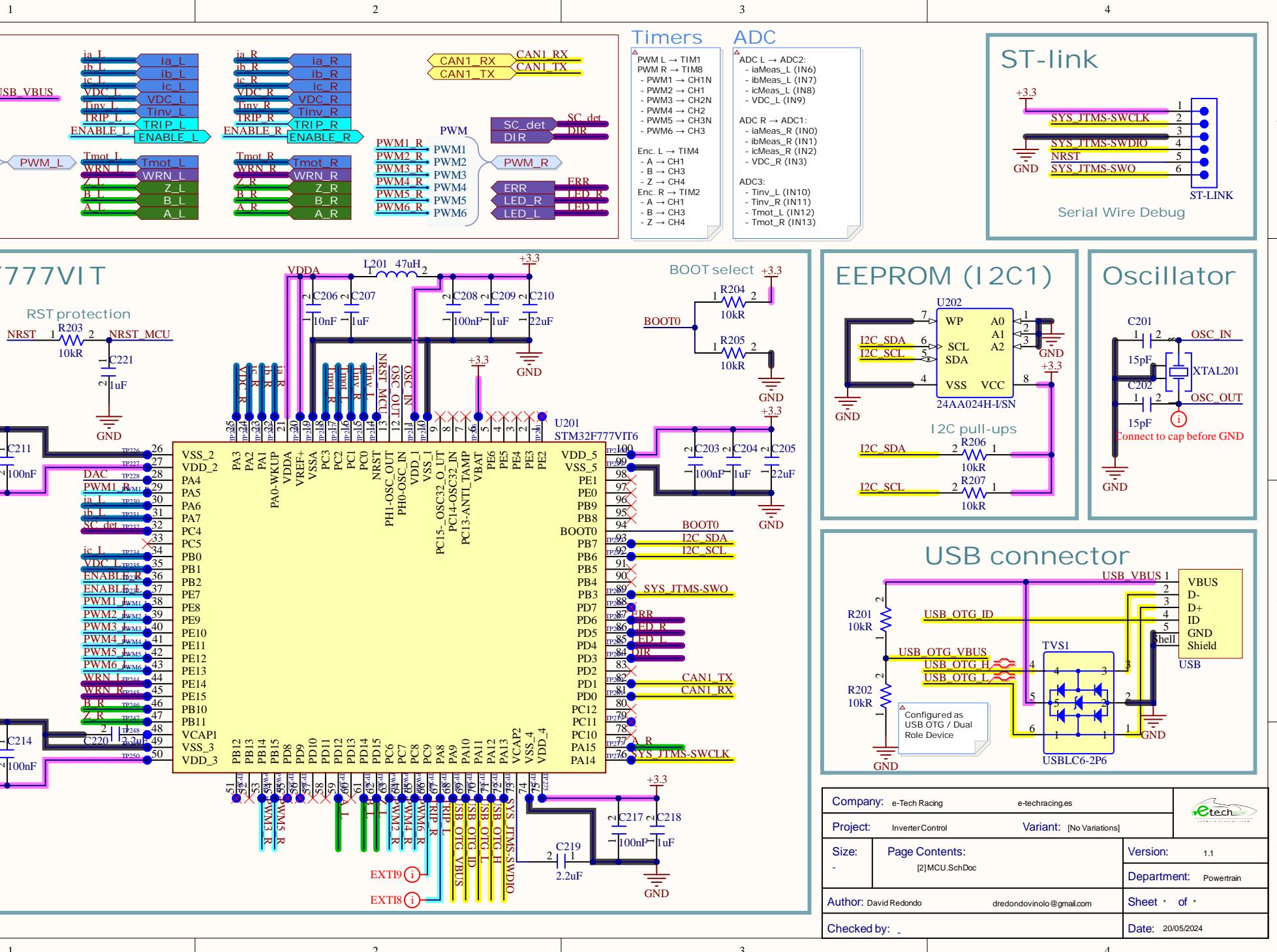


⚠️ PMOS disconnects USB power when the DCDC is supplying. Gate drivers should be disabled when the DCDC is not supplying via software (read LV+_in).

LDO

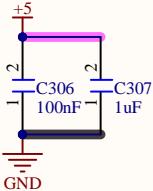


Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Control	Variant: [No Variations]	
Size:	Page Contents: [1]Supply.SchDoc	Version:	1.1
		Department:	Powertrain
Author:	David Redondo	dredondovinolo@gmail.com	Sheet • of •
Checked by:			Date: 20/05/2024

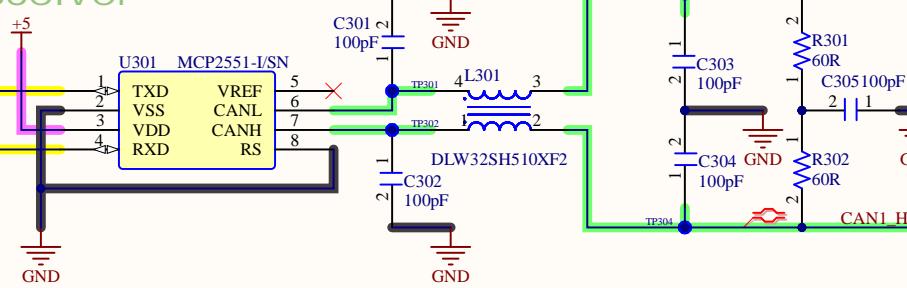


A

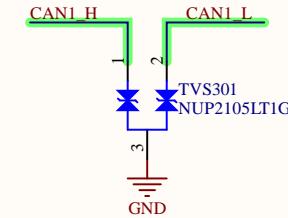
Decoupling



Transceiver



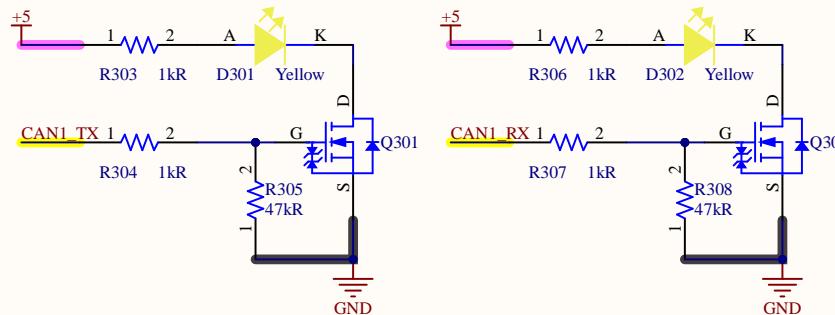
ESD



INPUTS/OUTPUTS

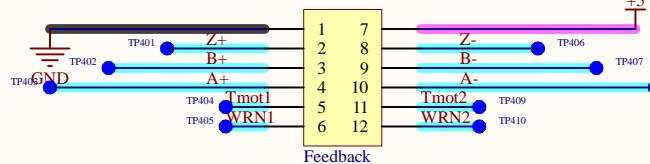
- <CAN1_RX> CAN1_RX
- <CAN1_TX> CAN1_TX
- <CAN1_H> CAN1_H
- <CAN1_L> CAN1_L

Status LEDs



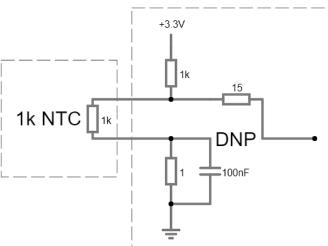
Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Control	Variant: [No Variations]	
Size:	Page Contents: [3]CAN.SchDoc	Version:	1.1
		Department:	Powertrain
Author:	David Redondo	dredondovinolo@gmail.com	Sheet • of •
Checked by:	•		Date: 20/05/2024

Feedback connector



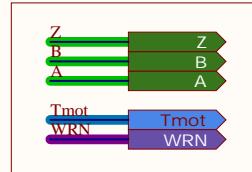
As the motors' temperature sensors are not specified, the user may modify the resistor combination to find a suitable input for the ADC, then load a custom lookup table to have an appropriate reading.

Example

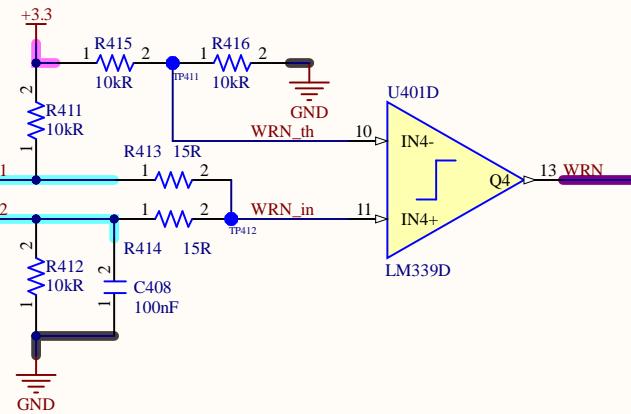


The WRN circuit can be used so that the MCU can detect a specified alarm. A resistive sensor can be used to detect any physical signal, such as overtemperature in any component (e.g. water outlet, gearbox, ...), underpressure of the cooling system, etc. Similar to the motor temperature sensors, the user may modify the resistor combination to have a suitable reading and adjust the voltage divider in order to set the threshold. Other types of sensors can be used, given a previous study and correct implementation.

INPUTS/OUTPUTS

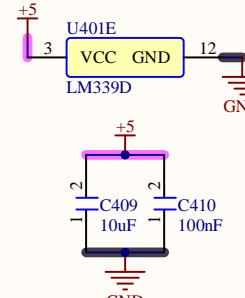


Auxiliary warning (WRN)



B-

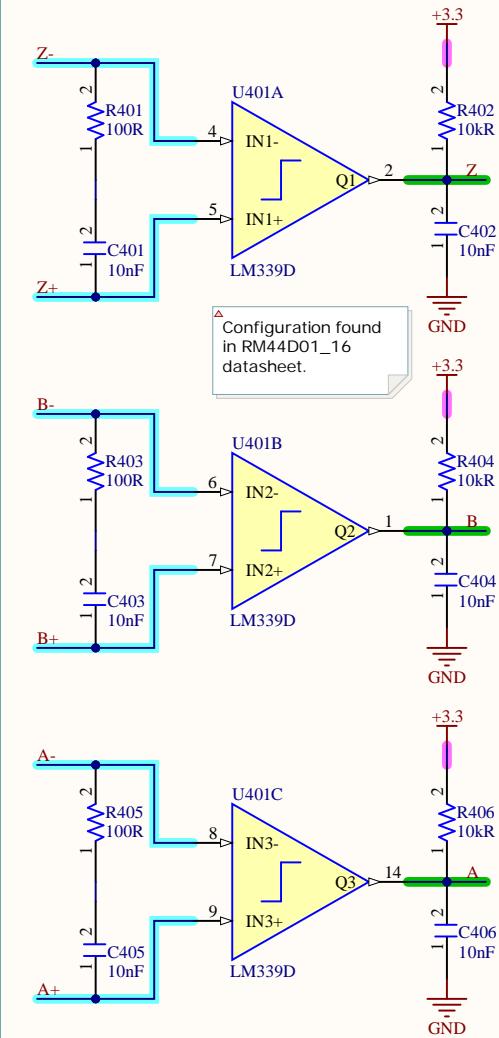
Comparator supply



Motor thermistor

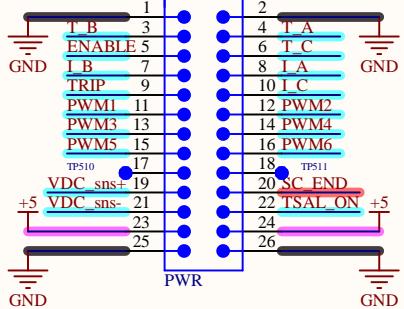
Only compatible with resistive temperature sensors.

Incremental encoder

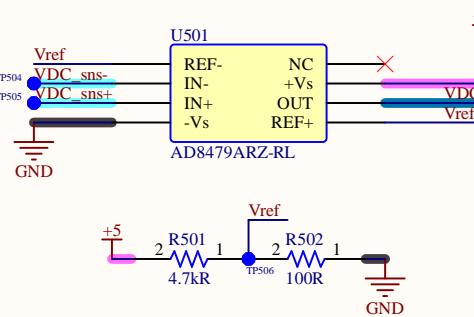


Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Control	Variant: [No Variations]	
Size: -	Page Contents: [4]Feedback.SchDoc	Version: 1.1	
		Department:	Powertrain
Author:	David Redondo	dredondovinolo@gmail.com	Sheet * of *
Checked by:	-		Date: 20/05/2024

Power PCB Connector



VDC sense AFE



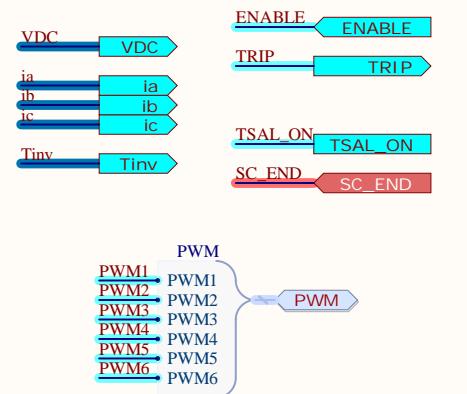
Difference amplifier

LT SPICE simulation for reference.

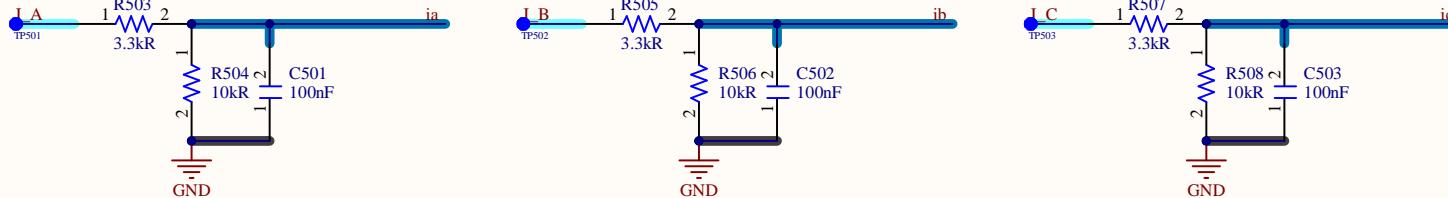
$$VDC = Vref + (VDC_{sns+} - VDC_{sns-}) = Vref + \frac{1}{3} \cdot 0.011388 \cdot (TS+ - TS-)$$

Maximum error due to using a voltage divider as a voltage reference is 1.79mV in ADC, which translates to 0.47V in (TS+ - TS-). Lowering the resistance values (while keeping proportionality) will reduce the error, but more current will be drawn.

INPUTS/OUTPUTS



Current sense

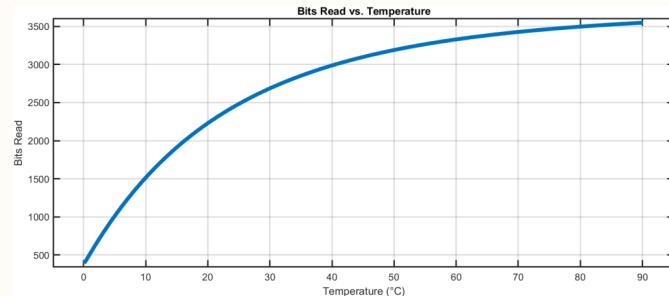


ENABLE is output directly from the MCU, it has been checked that UC21732 is able to detect it at 3.3V. Similarly, TRIP comes at 5V, and uses a 5V tolerant GPIO in the MCU.

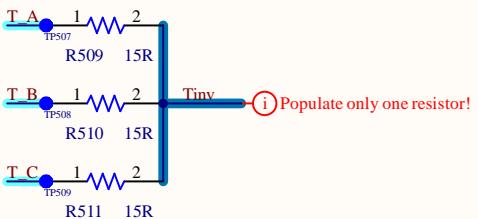
$VDC_offset = Vref \cdot 2^{12} \text{ bits} / (3.3V) = 0.02083 \cdot 2^{12} \text{ bits} / (3.3V) = 129 \text{ bits}$
 $VDC_gain = 1 / ((1/3 \cdot 0.011388 \text{ V/V}) \cdot (2^{12} \text{ bits} / 3.3 \text{ V})) = 0.212240269 \text{ V / bit}$
 $VDC_max = 0.212240269 \text{ V / bit} \cdot 2^{12} \text{ bits} = 869.34 \text{ V}$

$$\begin{aligned} ix_offset &= (10k/(3.3k+10k)) \cdot 2.5V \cdot 2^{12} \text{ bits} / (3.3V) = 0.02083 \cdot 2^{12} \text{ bits} / (3.3V) = 129 \text{ bits} \\ ix_gain &= (10k/(3.3k+10k)) / (12.5 \text{ mV/A} \cdot (2^{12} \text{ bits} / 3.3 \text{ V})) = 0.0484609962 \text{ A / bit} \\ ix_max(+/-) &= +/- 0.0484609962 \text{ A / bit} \cdot 2^{12} \text{ bits} / 2 = +/- 99 \text{ A} \end{aligned}$$

Inverters temperature should be calculated with a lookup table according to this graph. The lookup table and graph is generated with a MATLAB script which can be found in the simulations folder.



Temperature selection



Tiny is a pulsed signal that can read directly as a PWM input or be passed through an RC filter (in Inverter_Power) to convert it into an analog signal. This board intends to read it with the ADC. The reading itself is in the TS part of the power board and connected to the AIN pin of UCC21732.

Based on the sensed voltage, the duty cycle (D) of the UCC21732 isolated output signal is calculated using the following relationship: $D = -20 \cdot V_{AIN} + 100$

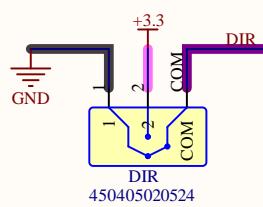
If filtered, the voltage at Tiny is calculated as: $V_{TINY} = VCC_GD \cdot D/100 = 5V \cdot (-20 \cdot V_{AIN} + 100)/100$

Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Control	Variant: [No Variations]	
Size:	Page Contents: [5]Power_AFE.SchDoc	Version: 1.1	
Department:	Powertrain		
Author:	David Redondo	dredondovinolo@gmail.com	Sheet 1 of 1
Checked by:			Date: 20/05/2024

A

A

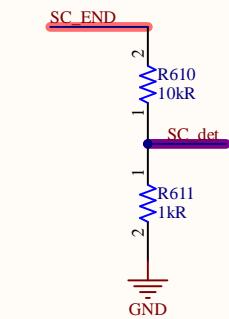
Reverse direction



B

B

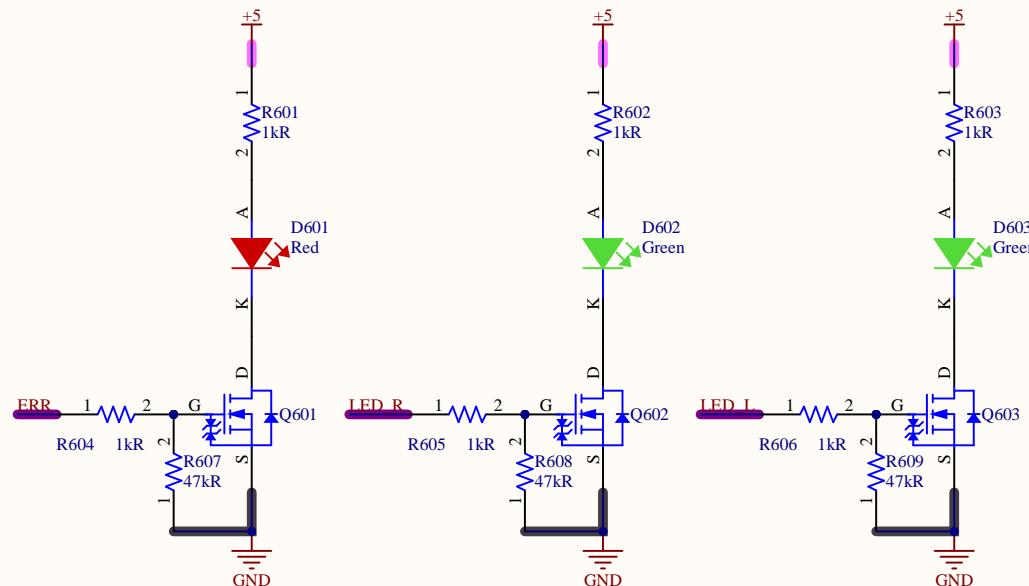
SC detection



C

C

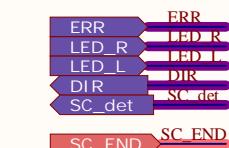
Status LEDs



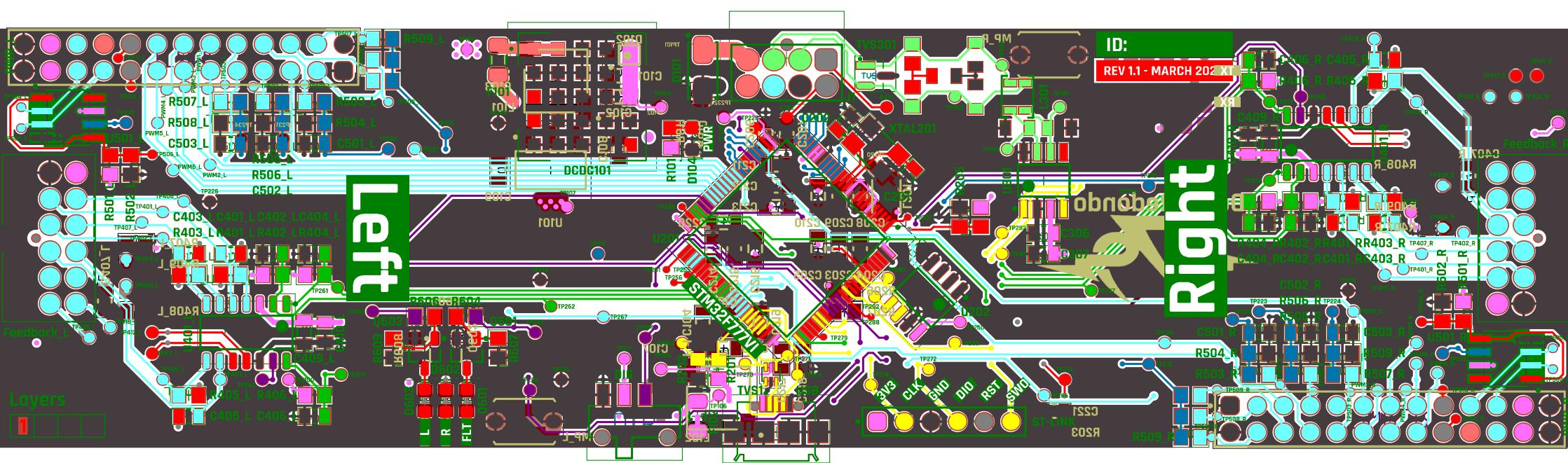
D

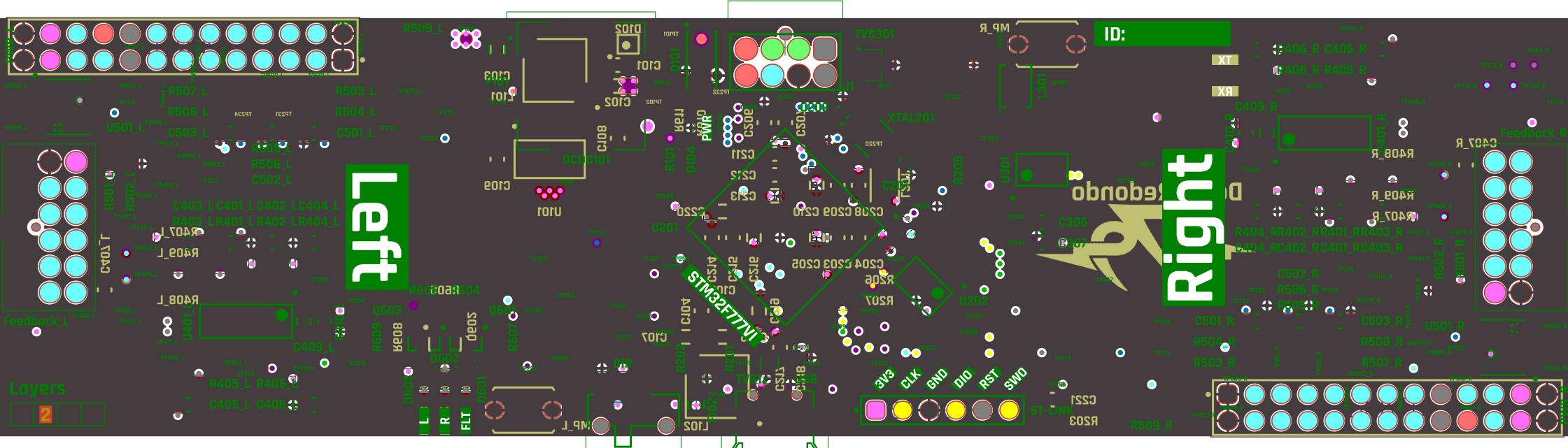
D

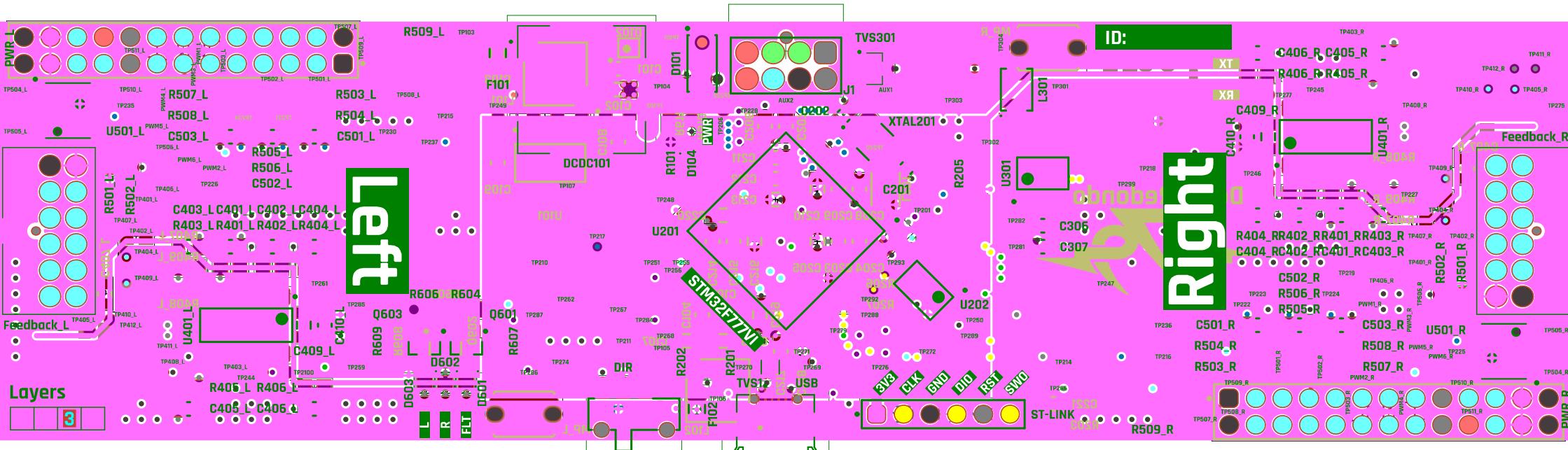
INPUTS/OUTPUTS

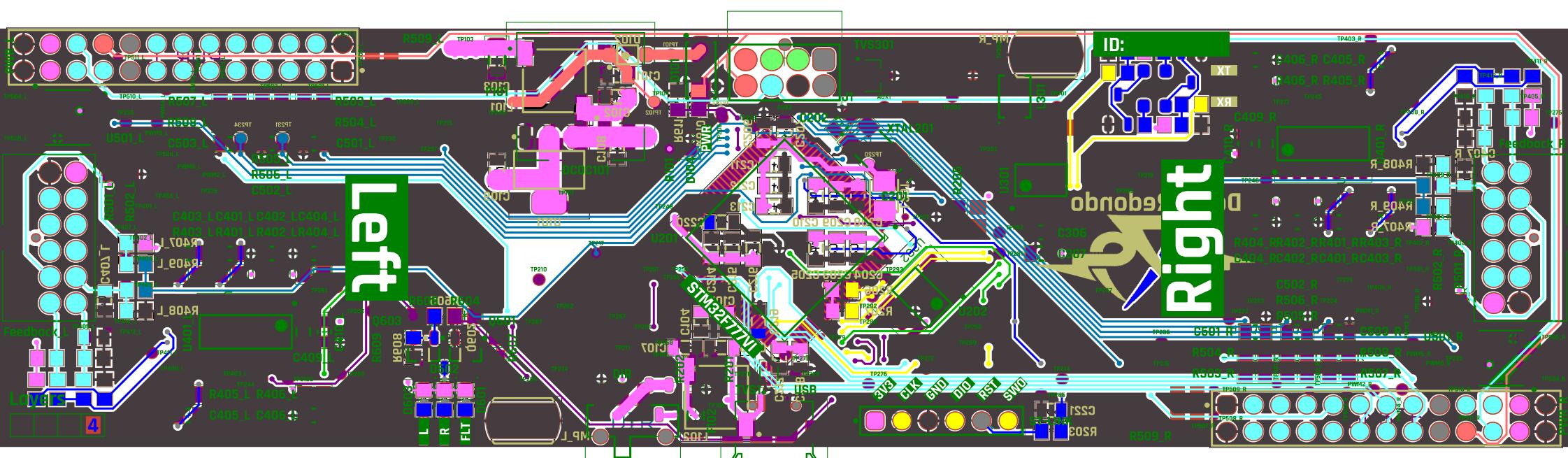


Company:	e-Tech Racing	e-techracing.es	
Project:	Inverter Control	Variant: [No Variations]	
Size:	Page Contents: [6] Extras.SchDoc	Version:	1.1
-		Department:	Powertrain
Author:	David Redondo	dredondovinolo@gmail.com	Sheet • of •
Checked by:	•		Date: 20/05/2024

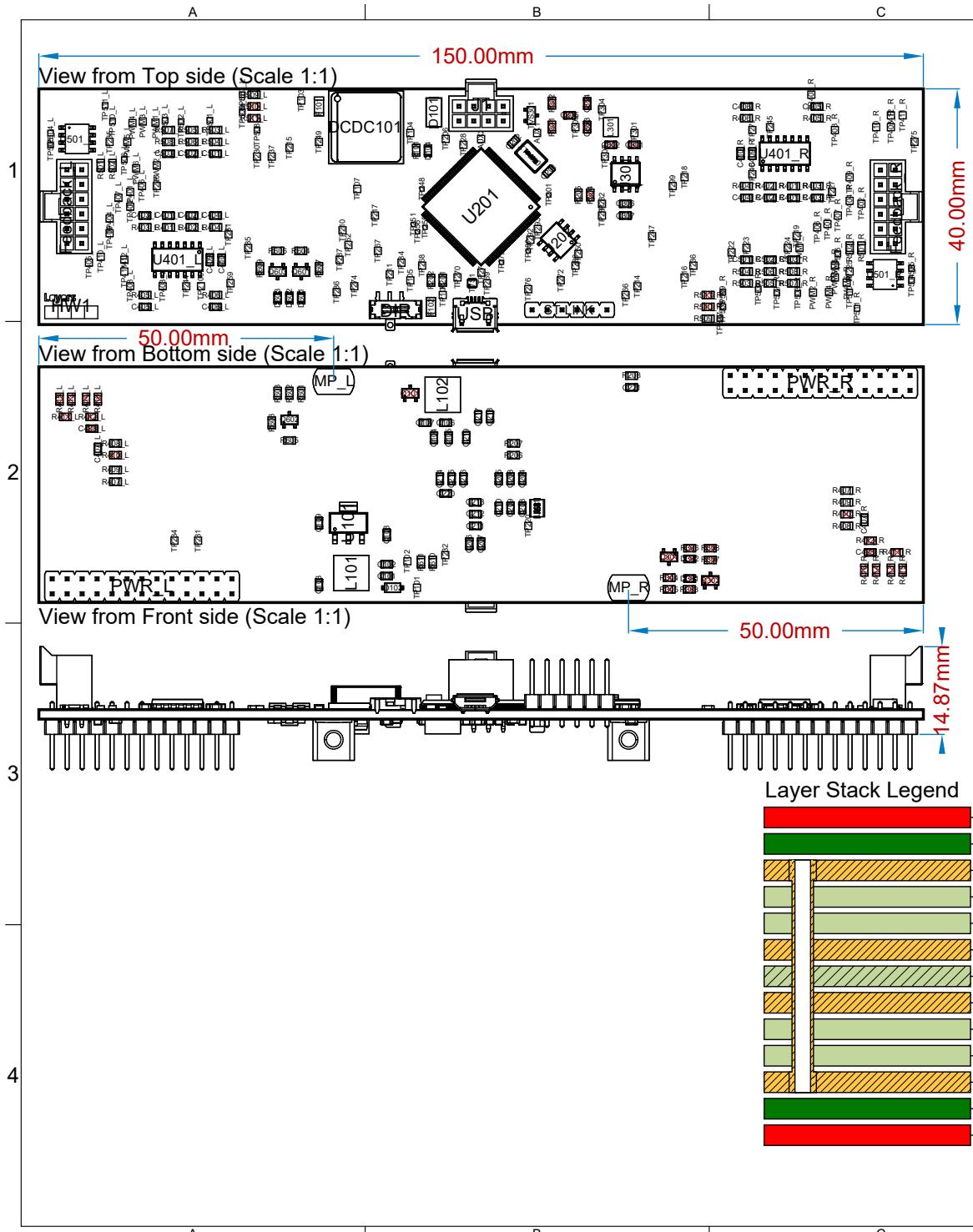








Inverter Control



Material	Layer	Thickness	Dielectric Material	Type	Gerber
Surface Material	Top Overlay				Legend GTO
CF-004	TOP	0.035mm	Solder Resist	Solder Mask GTS	GTL
Prepreg		0.100mm	PP-006	Dielectric	
Prepreg		0.100mm	PP-006	Dielectric	
Copper	GND	0.035mm		Signal G1	
		1.040mm	FR-4	Dielectric	
Copper	PWR	0.035mm		Signal G2	
Prepreg		0.100mm	PP-006	Dielectric	
Prepreg		0.100mm	PP-006	Dielectric	
CF-004	BOT	0.035mm		Signal GBL	
Surface Material	Bottom Solder	0.010mm	Solder Resist	Solder Mask GBS	
	Bottom Overlay			Legend GBO	

A.3. Documentación del *firmware*

e-Tech Racing's Inverter Firmware

v0

Generated by Doxygen 1.10.0

1 Topic Index	1
1.1 Topics	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Topic Documentation	7
4.1 Math Constants	7
4.1.1 Detailed Description	7
4.1.2 Macro Definition Documentation	7
4.1.2.1 DIV2	7
4.1.2.2 INV3	7
4.1.2.3 INV_DEG	8
4.1.2.4 IPI	8
4.1.2.5 IPI2	8
4.1.2.6 ISQ2	8
4.1.2.7 ISQ3	8
4.1.2.8 PI	8
4.1.2.9 PI2	8
4.1.2.10 SQ2	8
4.1.2.11 SQ3	9
4.2 - Integral Controllers	9
4.2.1 Detailed Description	9
4.2.2 Function Documentation	9
4.2.2.1 pi_aw_calc()	9
4.2.2.2 pi_calc()	10
4.2.2.3 pi_extsat_calc()	10
4.2.2.4 pi_init()	11
4.3 Clarke and Park Transformations	11
4.3.1 Detailed Description	12
4.3.2 Function Documentation	12
4.3.2.1 clarke3F_calc()	12
4.3.2.2 iclarke3F_calc()	12
4.3.2.3 irot_calc()	13
4.3.2.4 rot_calc()	13
4.4 Utility Functions	14
4.4.1 Detailed Description	14
4.4.2 Function Documentation	14
4.4.2.1 angle_calc()	14
4.4.2.2 svpwm_calc()	15

4.5 Signal Processing Functions	15
4.5.1 Detailed Description	16
4.5.2 Macro Definition Documentation	16
4.5.2.1 N_DATALOG	16
4.5.3 Function Documentation	16
4.5.3.1 avg_calc_10_samples()	16
4.5.3.2 datalog_calc()	17
4.5.3.3 filtreLP_calc()	17
4.5.3.4 filtreLP_init()	18
4.5.3.5 rampa_calc()	18
4.5.3.6 rampa_dual_calc()	19
4.5.3.7 RMS_calc()	19
4.6 Miscellaneous Functions	20
4.6.1 Detailed Description	20
4.6.2 Function Documentation	20
4.6.2.1 step_calc()	20
5 Data Structure Documentation	21
5.1 Analog Struct Reference	21
5.1.1 Detailed Description	21
5.1.2 Field Documentation	21
5.1.2.1 ia	21
5.1.2.2 ib	21
5.1.2.3 ic	22
5.1.2.4 vDC	22
5.2 angle_struct Struct Reference	22
5.2.1 Detailed Description	22
5.2.2 Field Documentation	22
5.2.2.1 angle	22
5.2.2.2 calc	22
5.2.2.3 freq	23
5.2.2.4 Ts	23
5.3 avg_struct_10 Struct Reference	23
5.3.1 Detailed Description	23
5.3.2 Field Documentation	23
5.3.2.1 in	23
5.3.2.2 out	23
5.4 CANMessageInfo Struct Reference	24
5.4.1 Field Documentation	24
5.4.1.1 DLC	24
5.4.1.2 getSig	24
5.4.1.3 ID	24

5.4.1.4 IDE	25
5.5 clarke3F_struct Struct Reference	25
5.5.1 Detailed Description	25
5.5.2 Field Documentation	25
5.5.2.1 a	25
5.5.2.2 b	25
5.5.2.3 calc	25
5.5.2.4 D	26
5.5.2.5 Q	26
5.6 datalog_struct Struct Reference	26
5.6.1 Field Documentation	26
5.6.1.1 calc	26
5.6.1.2 estat	26
5.6.1.3 i	27
5.6.1.4 j	27
5.6.1.5 log	27
5.6.1.6 prescaler	27
5.6.1.7 var	27
5.7 Duties Struct Reference	27
5.7.1 Detailed Description	28
5.7.2 Field Documentation	28
5.7.2.1 Da	28
5.7.2.2 Db	28
5.7.2.3 Dc	28
5.8 Encoder Struct Reference	28
5.8.1 Detailed Description	29
5.8.2 Field Documentation	29
5.8.2.1 A	29
5.8.2.2 B	29
5.8.2.3 cosTheta_e	29
5.8.2.4 directionMeas	29
5.8.2.5 sinTheta_e	29
5.8.2.6 theta_e	29
5.8.2.7 we	30
5.8.2.8 Z	30
5.9 Feedback Struct Reference	30
5.9.1 Detailed Description	30
5.9.2 Field Documentation	30
5.9.2.1 idMeas	30
5.9.2.2 iqMeas	30
5.9.2.3 speedMeas	31
5.9.2.4 torqueCalc	31

5.10 filtreLP_struct Struct Reference	31
5.10.1 Detailed Description	31
5.10.2 Field Documentation	31
5.10.2.1 alfa	31
5.10.2.2 calc	32
5.10.2.3 enable	32
5.10.2.4 fc	32
5.10.2.5 in	32
5.10.2.6 init	32
5.10.2.7 out	32
5.10.2.8 Ts	32
5.11 iclarke3F_struct Struct Reference	33
5.11.1 Detailed Description	33
5.11.2 Field Documentation	33
5.11.2.1 a	33
5.11.2.2 b	33
5.11.2.3 calc	33
5.11.2.4 D	33
5.11.2.5 Q	34
5.12 InverterStruct Struct Reference	34
5.12.1 Detailed Description	35
5.12.2 Field Documentation	35
5.12.2.1 analog	35
5.12.2.2 direction	35
5.12.2.3 duties	35
5.12.2.4 enable_pin	35
5.12.2.5 enable_port	35
5.12.2.6 encoder	35
5.12.2.7 feedback	36
5.12.2.8 hadc	36
5.12.2.9 htim	36
5.12.2.10 idLoop	36
5.12.2.11 iqLoop	36
5.12.2.12 led	36
5.12.2.13 motor	36
5.12.2.14 reference	36
5.12.2.15 speedLoop	37
5.12.2.16 state	37
5.12.2.17 templInverter	37
5.12.2.18 tempMotor	37
5.13 irot_struct Struct Reference	37
5.13.1 Detailed Description	37

5.13.2 Field Documentation	38
5.13.2.1 alpha	38
5.13.2.2 beta	38
5.13.2.3 calc	38
5.13.2.4 cosFi	38
5.13.2.5 d	38
5.13.2.6 q	38
5.13.2.7 sinFi	38
5.14 LED Struct Reference	39
5.14.1 Detailed Description	39
5.14.2 Field Documentation	39
5.14.2.1 mode	39
5.14.2.2 pin	39
5.14.2.3 port	39
5.15 MotorParameters Struct Reference	39
5.15.1 Detailed Description	40
5.15.2 Field Documentation	40
5.15.2.1 b	40
5.15.2.2 dTorque_max	40
5.15.2.3 iPhase_pk_max	40
5.15.2.4 J	40
5.15.2.5 lambda	41
5.15.2.6 Ld	41
5.15.2.7 Lq	41
5.15.2.8 pp	41
5.15.2.9 Rs	41
5.15.2.10 speed_max_RPM	41
5.15.2.11 torque_max	41
5.15.2.12 vDC_max	42
5.16 pi_aw_struct Struct Reference	42
5.16.1 Detailed Description	42
5.16.2 Field Documentation	42
5.16.2.1 calc	42
5.16.2.2 e	43
5.16.2.3 enable	43
5.16.2.4 Kaw	43
5.16.2.5 Ki	43
5.16.2.6 Kp	43
5.16.2.7 pi_consig	43
5.16.2.8 pi_fdb	43
5.16.2.9 pi_ffw	43
5.16.2.10 pi_int	44

5.16.2.11 pi_out	44
5.16.2.12 pi_out_max	44
5.16.2.13 pi_out_min	44
5.16.2.14 pi_out_postsat	44
5.16.2.15 pi_out_presat	44
5.16.2.16 Ts	44
5.17 pi_struct Struct Reference	45
5.17.1 Detailed Description	45
5.17.2 Field Documentation	45
5.17.2.1 calc	45
5.17.2.2 e	45
5.17.2.3 enable	45
5.17.2.4 init	46
5.17.2.5 K0	46
5.17.2.6 K1	46
5.17.2.7 Ki	46
5.17.2.8 Kp	46
5.17.2.9 pi_consig	46
5.17.2.10 pi_fdb	46
5.17.2.11 pi_ffw	46
5.17.2.12 pi_out	47
5.17.2.13 pi_out_max	47
5.17.2.14 pi_out_min	47
5.17.2.15 Ts	47
5.18 rampa_dual_struct Struct Reference	47
5.18.1 Detailed Description	47
5.18.2 Field Documentation	48
5.18.2.1 calc	48
5.18.2.2 Decr	48
5.18.2.3 enable	48
5.18.2.4 in	48
5.18.2.5 Incr	48
5.18.2.6 out	48
5.19 rampa_struct Struct Reference	48
5.19.1 Detailed Description	49
5.19.2 Field Documentation	49
5.19.2.1 calc	49
5.19.2.2 enable	49
5.19.2.3 in	49
5.19.2.4 Incr	49
5.19.2.5 out	49
5.20 Reference Struct Reference	50

5.20.1 Detailed Description	50
5.20.2 Field Documentation	50
5.20.2.1 idRef	50
5.20.2.2 iqRef	50
5.20.2.3 torqueRef	50
5.21 RMS_struct Struct Reference	50
5.21.1 Detailed Description	51
5.21.2 Field Documentation	51
5.21.2.1 Angle	51
5.21.2.2 Angle_ant	51
5.21.2.3 Freq	51
5.21.2.4 Measure	51
5.21.2.5 Out_RMS	51
5.21.2.6 Sq_Sum	52
5.21.2.7 T_exec	52
5.22 rot_struct Struct Reference	52
5.22.1 Detailed Description	52
5.22.2 Field Documentation	52
5.22.2.1 calc	52
5.22.2.2 cosFi	53
5.22.2.3 D	53
5.22.2.4 d	53
5.22.2.5 Q	53
5.22.2.6 q	53
5.22.2.7 sinFi	53
5.23 signal Struct Reference	53
5.23.1 Detailed Description	54
5.23.2 Field Documentation	54
5.23.2.1 factor	54
5.23.2.2 lengthBits	54
5.23.2.3 max	54
5.23.2.4 min	54
5.23.2.5 name	54
5.23.2.6 offset	54
5.23.2.7 unit	55
5.23.2.8 valueType	55
5.24 signal_positioned Struct Reference	55
5.24.1 Detailed Description	55
5.24.2 Field Documentation	55
5.24.2.1 attributes	55
5.24.2.2 position	56
5.25 step_struct Struct Reference	56

5.25.1 Detailed Description	56
5.25.2 Field Documentation	56
5.25.2.1 calc	56
5.25.2.2 Counter	56
5.25.2.3 enable	57
5.25.2.4 fs	57
5.25.2.5 In	57
5.25.2.6 Out	57
5.25.2.7 Pulses	57
5.25.2.8 Step	57
5.25.2.9 t_step	57
5.26 svpwm_struct Struct Reference	58
5.26.1 Detailed Description	58
5.26.2 Field Documentation	58
5.26.2.1 alpha	58
5.26.2.2 beta	58
5.26.2.3 calc	58
5.26.2.4 Da	58
5.26.2.5 Db	59
5.26.2.6 Dc	59
6 File Documentation	61
6.1 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN1db.h File Reference	61
6.1.1 Macro Definition Documentation	63
6.1.1.1 CAN_ID_EXT	63
6.1.1.2 CAN_ID_STD	63
6.1.1.3 DBC_linkMsgConverter	63
6.1.1.4 DBC_setup	63
6.1.2 Typedef Documentation	64
6.1.2.1 dbc_valueType	64
6.1.2.2 signal	64
6.1.2.3 signal_positioned	64
6.1.3 Enumeration Type Documentation	64
6.1.3.1 dbc_valueType	64
6.1.4 Function Documentation	64
6.1.4.1 __attribute__()	64
6.1.5 Variable Documentation	65
6.1.5.1 DCBus_RL	65
6.1.5.2 DCBus_RR	65
6.1.5.3 DLC	65
6.1.5.4 Enable_Inv_R	65
6.1.5.5 [struct]	65

6.1.5.6 getSigVal	65
6.1.5.7 lactual_RL	65
6.1.5.8 lactual_RR	65
6.1.5.9 lcnd_RL	65
6.1.5.10 lcnd_RR	65
6.1.5.11 ID	66
6.1.5.12 IDE	66
6.1.5.13 Iq_RL	66
6.1.5.14 Iq_RR	66
6.1.5.15 Power_RL	66
6.1.5.16 Power_RR	66
6.1.5.17 SpeedMotor_RL	66
6.1.5.18 SpeedMotor_RR	66
6.1.5.19 TempIGBT_RL	66
6.1.5.20 TempIGBT_RR	66
6.1.5.21 TempMotor_RL	67
6.1.5.22 TempMotor_RR	67
6.1.5.23 TorqueReal_RL	67
6.1.5.24 TorqueReal_RR	67
6.1.5.25 TotalPower	67
6.2 CAN1db.h	67
6.3 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN_e-Tech.h File Reference	74
6.3.1 Detailed Description	75
6.3.2 Function Documentation	76
6.3.2.1 handle_CAN()	76
6.3.2.2 send_CAN_message()	77
6.3.3 Variable Documentation	78
6.3.3.1 enableCAN	78
6.4 CAN_e-Tech.h	78
6.5 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CONTROL.h File Reference	78
6.5.1 Detailed Description	80
6.5.2 Function Documentation	80
6.5.2.1 calc_current_loop()	80
6.5.2.2 calc_duties()	81
6.6 CONTROL.h	81
6.7 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/FSM.h File Reference	82
6.7.1 Detailed Description	83
6.7.2 Function Documentation	83
6.7.2.1 eval_inv_FSM()	83
6.8 FSM.h	84
6.9 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/INVERTER.h File Reference	84
6.9.1 Detailed Description	86

6.9.2 Macro Definition Documentation	86
6.9.2.1 DT	86
6.9.2.2 TS	86
6.9.3 Enumeration Type Documentation	86
6.9.3.1 InverterState	86
6.9.4 Function Documentation	87
6.9.4.1 init_control_loops()	87
6.9.4.2 initialize_inverter()	88
6.9.5 Variable Documentation	89
6.9.5.1 inverter_left	89
6.9.5.2 inverter_right	89
6.10 INVERTER.h	89
6.11 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/main.h File Reference	90
6.11.1 Detailed Description	92
6.11.2 Macro Definition Documentation	92
6.11.2.1 A_L_GPIO_Port	92
6.11.2.2 A_L_Pin	92
6.11.2.3 A_R_GPIO_Port	93
6.11.2.4 A_R_Pin	93
6.11.2.5 B_L_GPIO_Port	93
6.11.2.6 B_L_Pin	93
6.11.2.7 B_R_GPIO_Port	93
6.11.2.8 B_R_Pin	93
6.11.2.9 DAC_GPIO_Port	93
6.11.2.10 DAC_Pin	93
6.11.2.11 DIR_GPIO_Port	93
6.11.2.12 DIR_Pin	93
6.11.2.13 ENABLE_L_GPIO_Port	94
6.11.2.14 ENABLE_L_Pin	94
6.11.2.15 ENABLE_R_GPIO_Port	94
6.11.2.16 ENABLE_R_Pin	94
6.11.2.17 ia_L_GPIO_Port	94
6.11.2.18 ia_L_Pin	94
6.11.2.19 ia_R_GPIO_Port	94
6.11.2.20 ia_R_Pin	94
6.11.2.21 ib_L_GPIO_Port	94
6.11.2.22 ib_L_Pin	94
6.11.2.23 ib_R_GPIO_Port	95
6.11.2.24 ib_R_Pin	95
6.11.2.25 ic_L_GPIO_Port	95
6.11.2.26 ic_L_Pin	95
6.11.2.27 ic_R_GPIO_Port	95

6.11.2.28 ic_R_Pin	95
6.11.2.29 LED_ERR_GPIO_Port	95
6.11.2.30 LED_ERR_Pin	95
6.11.2.31 LED_LEFT_GPIO_Port	95
6.11.2.32 LED_LEFT_Pin	95
6.11.2.33 LED_RIGHT_GPIO_Port	96
6.11.2.34 LED_RIGHT_Pin	96
6.11.2.35 PWM1_L_GPIO_Port	96
6.11.2.36 PWM1_L_Pin	96
6.11.2.37 PWM1_R_GPIO_Port	96
6.11.2.38 PWM1_R_Pin	96
6.11.2.39 PWM2_L_GPIO_Port	96
6.11.2.40 PWM2_L_Pin	96
6.11.2.41 PWM2_R_GPIO_Port	96
6.11.2.42 PWM2_R_Pin	96
6.11.2.43 PWM3_L_GPIO_Port	97
6.11.2.44 PWM3_L_Pin	97
6.11.2.45 PWM3_R_GPIO_Port	97
6.11.2.46 PWM3_R_Pin	97
6.11.2.47 PWM4_L_GPIO_Port	97
6.11.2.48 PWM4_L_Pin	97
6.11.2.49 PWM4_R_GPIO_Port	97
6.11.2.50 PWM4_R_Pin	97
6.11.2.51 PWM5_L_GPIO_Port	97
6.11.2.52 PWM5_L_Pin	97
6.11.2.53 PWM5_R_GPIO_Port	98
6.11.2.54 PWM5_R_Pin	98
6.11.2.55 PWM6_L_GPIO_Port	98
6.11.2.56 PWM6_L_Pin	98
6.11.2.57 PWM6_R_GPIO_Port	98
6.11.2.58 PWM6_R_Pin	98
6.11.2.59 SC_det_GPIO_Port	98
6.11.2.60 SC_det_Pin	98
6.11.2.61 Tinv_L_GPIO_Port	98
6.11.2.62 Tinv_L_Pin	98
6.11.2.63 Tinv_R_GPIO_Port	99
6.11.2.64 Tinv_R_Pin	99
6.11.2.65 Tmot_L_GPIO_Port	99
6.11.2.66 Tmot_L_Pin	99
6.11.2.67 Tmot_R_GPIO_Port	99
6.11.2.68 Tmot_R_Pin	99
6.11.2.69 TRIP_L_GPIO_Port	99

6.11.2.70 TRIP_L_Pin	99
6.11.2.71 TRIP_R_GPIO_Port	99
6.11.2.72 TRIP_R_Pin	99
6.11.2.73 VDC_L_GPIO_Port	100
6.11.2.74 VDC_L_Pin	100
6.11.2.75 VDC_R_GPIO_Port	100
6.11.2.76 VDC_R_Pin	100
6.11.2.77 WRN_L_GPIO_Port	100
6.11.2.78 WRN_L_Pin	100
6.11.2.79 WRN_R_GPIO_Port	100
6.11.2.80 WRN_R_Pin	100
6.11.2.81 Z_L_GPIO_Port	100
6.11.2.82 Z_L_Pin	100
6.11.2.83 Z_R_GPIO_Port	101
6.11.2.84 Z_R_Pin	101
6.11.3 Function Documentation	101
6.11.3.1 Error_Handler()	101
6.12 main.h	102
6.13 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MEASUREMENTS.h File Reference	103
6.13.1 Detailed Description	105
6.13.2 Macro Definition Documentation	105
6.13.2.1 CURRENT_OFFSET	105
6.13.2.2 CURRENT_SLOPE	105
6.13.2.3 VOLTAGE_OFFSET	106
6.13.2.4 VOLTAGE_SLOPE	106
6.13.3 Function Documentation	106
6.13.3.1 get_currents_voltage()	106
6.13.3.2 get_idiq()	107
6.13.3.3 get_linear()	108
6.13.3.4 get_temperature()	109
6.13.4 Variable Documentation	109
6.13.4.1 rawADC_left	109
6.13.4.2 rawADC_right	109
6.13.4.3 rawADC_temp	110
6.13.4.4 templInverterLUT	110
6.13.4.5 tempMotorLUT	110
6.14 MEASUREMENTS.h	110
6.15 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MOTOR.h File Reference	111
6.15.1 Detailed Description	112
6.15.2 Function Documentation	112
6.15.2.1 check_motor_parameters()	112
6.15.3 Variable Documentation	112

6.15.3.1 motor_left	112
6.15.3.2 motor_right	113
6.16 MOTOR.h	113
6.17 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PCB_IO.h File Reference	113
6.17.1 Detailed Description	115
6.17.2 Macro Definition Documentation	115
6.17.2.1 DIR_STATE	115
6.17.2.2 DISABLE	115
6.17.2.3 ENABLE	115
6.17.2.4 SC_DET_STATE	115
6.17.2.5 WRN_STATE	115
6.17.3 Enumeration Type Documentation	115
6.17.3.1 LEDMode	115
6.17.4 Function Documentation	116
6.17.4.1 handle_direction()	116
6.17.4.2 handle_LED()	116
6.17.5 Variable Documentation	117
6.17.5.1 led_left	117
6.17.5.2 led_right	117
6.17.5.3 ledError	117
6.18 PCB_IO.h	117
6.19 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PergaMOD.h File Reference	118
6.20 PergaMOD.h	120
6.21 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PWM.h File Reference	123
6.21.1 Detailed Description	125
6.21.2 Function Documentation	125
6.21.2.1 disable_PWM()	125
6.21.2.2 enable_PWM()	125
6.21.2.3 update_PWM()	125
6.22 PWM.h	126
6.23 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/REFERENCE.h File Reference	126
6.23.1 Detailed Description	128
6.23.2 Function Documentation	128
6.23.2.1 handle_torqueRef()	128
6.23.2.2 initialize_torque_ramp()	129
6.23.2.3 limit_torque_to_prevent_overspeed()	129
6.23.2.4 saturate_symmetric()	131
6.23.2.5 set_torque_direction()	131
6.24 REFERENCE.h	132
6.25 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/stm32f7xx_it.h File Reference	132
6.25.1 Detailed Description	134
6.25.2 Function Documentation	134

6.25.2.1 BusFault_Handler()	134
6.25.2.2 CAN1_RX0_IRQHandler()	134
6.25.2.3 DebugMon_Handler()	134
6.25.2.4 DMA2_Stream0_IRQHandler()	135
6.25.2.5 DMA2_Stream1_IRQHandler()	135
6.25.2.6 DMA2_Stream2_IRQHandler()	135
6.25.2.7 HardFault_Handler()	135
6.25.2.8 MemManage_Handler()	135
6.25.2.9 NMI_Handler()	135
6.25.2.10 PendSV_Handler()	135
6.25.2.11 SVC_Handler()	136
6.25.2.12 SysTick_Handler()	136
6.25.2.13 TIM1_UP_TIM10_IRQHandler()	136
6.25.2.14 TIM6_DAC_IRQHandler()	136
6.25.2.15 UsageFault_Handler()	137
6.26 stm32f7xx_it.h	137
6.27 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_1ms.h File Reference	138
6.27.1 Detailed Description	138
6.27.2 Function Documentation	139
6.27.2.1 tasks_1ms()	139
6.28 TASKS_1ms.h	139
6.29 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_20us.h File Reference	140
6.29.1 Detailed Description	140
6.29.2 Function Documentation	141
6.29.2.1 tasks_20us_left()	141
6.29.2.2 tasks_20us_right()	141
6.30 TASKS_20us.h	142
6.31 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/can.c File Reference	142
6.31.1 Detailed Description	142
6.31.2 Function Documentation	143
6.31.2.1 HAL_CAN_MspDeInit()	143
6.31.2.2 HAL_CAN_MspInit()	143
6.31.2.3 MX_CAN1_Init()	143
6.31.3 Variable Documentation	143
6.31.3.1 hcan1	143
6.32 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CAN_e-Tech.c File Reference	144
6.32.1 Detailed Description	144
6.32.2 Function Documentation	145
6.32.2.1 handle_CAN()	145
6.32.2.2 send_CAN_message()	145
6.32.3 Variable Documentation	146
6.32.3.1 enableCAN	146

6.33 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CONTROL.c File Reference	146
6.33.1 Detailed Description	147
6.33.2 Function Documentation	147
6.33.2.1 calc_current_loop()	147
6.33.2.2 calc_duties()	148
6.34 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/FSM.c File Reference	149
6.34.1 Detailed Description	150
6.34.2 Function Documentation	150
6.34.2.1 eval_inv_FSM()	150
6.35 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/INVERTER.c File Reference	150
6.35.1 Detailed Description	151
6.35.2 Function Documentation	152
6.35.2.1 init_control_loops()	152
6.35.2.2 initialize_inverter()	152
6.35.3 Variable Documentation	153
6.35.3.1 inverter_left	153
6.35.3.2 inverter_right	153
6.36 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/main.c File Reference	154
6.36.1 Detailed Description	154
6.36.2 Function Documentation	155
6.36.2.1 Error_Handler()	155
6.36.2.2 main()	155
6.36.2.3 SystemClock_Config()	156
6.37 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MEASUREMENTS.c File Reference	157
6.37.1 Detailed Description	167
6.37.2 Function Documentation	167
6.37.2.1 get_currents_voltage()	167
6.37.2.2 get_idiq()	168
6.37.2.3 get_linear()	169
6.37.2.4 get_temperature()	169
6.37.3 Variable Documentation	170
6.37.3.1 rawADC_left	170
6.37.3.2 rawADC_right	170
6.37.3.3 rawADC_temp	170
6.37.3.4 templInverterLUT	171
6.37.3.5 tempMotorLUT	176
6.38 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MOTOR.c File Reference	181
6.38.1 Detailed Description	182
6.38.2 Function Documentation	182
6.38.2.1 check_motor_parameters()	182
6.38.3 Variable Documentation	183
6.38.3.1 motor_left	183

6.38.3.2 motor_right	183
6.39 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PCB_IO.c File Reference	184
6.39.1 Detailed Description	184
6.39.2 Function Documentation	185
6.39.2.1 handle_direction()	185
6.39.2.2 handle_LED()	185
6.39.3 Variable Documentation	186
6.39.3.1 led_left	186
6.39.3.2 led_right	186
6.39.3.3 ledError	186
6.40 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PergaMOD.c File Reference	186
6.41 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PWM.c File Reference	187
6.41.1 Detailed Description	188
6.41.2 Function Documentation	188
6.41.2.1 disable_PWM()	188
6.41.2.2 enable_PWM()	189
6.41.2.3 update_PWM()	189
6.42 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/REFERENCE.c File Reference	189
6.42.1 Detailed Description	190
6.42.2 Function Documentation	191
6.42.2.1 handle_torqueRef()	191
6.42.2.2 initialize_torque_ramp()	192
6.42.2.3 limit_torque_to_prevent_overspeed()	192
6.42.2.4 saturate_symmetric()	193
6.42.2.5 set_torque_direction()	194
6.43 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/stm32f7xx_it.c File Reference	194
6.43.1 Detailed Description	196
6.43.2 Function Documentation	196
6.43.2.1 BusFault_Handler()	196
6.43.2.2 CAN1_RX0_IRQHandler()	196
6.43.2.3 DebugMon_Handler()	197
6.43.2.4 DMA2_Stream0_IRQHandler()	197
6.43.2.5 DMA2_Stream1_IRQHandler()	197
6.43.2.6 DMA2_Stream2_IRQHandler()	197
6.43.2.7 HardFault_Handler()	197
6.43.2.8 MemManage_Handler()	197
6.43.2.9 NMI_Handler()	197
6.43.2.10 PendSV_Handler()	198
6.43.2.11 SVC_Handler()	198
6.43.2.12 SysTick_Handler()	198
6.43.2.13 TIM1_UP_TIM10_IRQHandler()	198
6.43.2.14 TIM6_DAC_IRQHandler()	199

6.43.2.15 UsageFault_Handler()	199
6.43.3 Variable Documentation	199
6.43.3.1 hcan1	199
6.43.3.2 hdac	199
6.43.3.3 hdma_adc1	199
6.43.3.4 hdma_adc2	199
6.43.3.5 hdma_adc3	200
6.43.3.6 htim1	200
6.43.3.7 htim6	200
6.44 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS_1ms.c File Reference	200
6.44.1 Detailed Description	201
6.44.2 Function Documentation	201
6.44.2.1 tasks_1ms()	201
6.45 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS_20us.c File Reference	202
6.45.1 Detailed Description	203
6.45.2 Function Documentation	203
6.45.2.1 tasks_20us_left()	203
6.45.2.2 tasks_20us_right()	204
6.45.3 Variable Documentation	204
6.45.3.1 angle_left	204
6.45.3.2 elapsed_ticks	204
6.45.3.3 freqRamp_left	204
6.45.3.4 start_ticks	204
6.45.3.5 torqueRefln_left	204
6.45.3.6 vd_left	204
6.45.3.7 vDC_left	205
6.45.3.8 vq_left	205
6.46 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/tim.c File Reference	205
6.46.1 Detailed Description	206
6.46.2 Function Documentation	206
6.46.2.1 HAL_TIM_Base_MspDeInit()	206
6.46.2.2 HAL_TIM_Base_MspInit()	206
6.46.2.3 HAL_TIM_IC_MspDeInit()	206
6.46.2.4 HAL_TIM_IC_MspInit()	207
6.46.2.5 HAL_TIM_MspPostInit()	207
6.46.2.6 MX_TIM1_Init()	207
6.46.2.7 MX_TIM2_Init()	208
6.46.2.8 MX_TIM4_Init()	208
6.46.2.9 MX_TIM6_Init()	209
6.46.2.10 MX_TIM8_Init()	210
6.46.3 Variable Documentation	210
6.46.3.1 htim1	210

6.46.3.2 htim2	210
6.46.3.3 htim4	210
6.46.3.4 htim6	210
6.46.3.5 htim8	210

Index	211
--------------	------------

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

Math Constants	7
- Integral Controllers	9
Clarke and Park Transformations	11
Utility Functions	14
Signal Processing Functions	15
Miscellaneous Functions	20

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Analog	Structure for ADC measurements in units	21
angle_struct	Generates an angle based on a fixed frequency	22
avg_struct_10	Moving average filter for 10 samples	23
CANMessageInfo	24
clarke3F_struct	Clarke transformation for three-phase systems	25
datalog_struct	26
Duties	Structure to hold PWM configuration parameters	27
Encoder	Structure for encoder reading	28
Feedback	Structure for feedback values	30
filtreLP_struct	First-order low-pass filter	31
iclarke3F_struct	Inverse Clarke transformation for three-phase systems	33
InverterStruct	Inverter structure	34
irot_struct	Inverse rotation (counterclockwise)	37
LED	LED structure	39
MotorParameters	Structure to hold motor parameters	39
pi_aw_struct	PI Controller with internal saturation, anti-windup, and feedforward	42
pi_struct	PI Controller with external saturation and feedforward	45
rampa_dual_struct	Dual-ramp generator	47
rampa_struct	Single-ramp generator	48

Reference	
Structure for reference values	50
RMS_struct	
Root Mean Square (RMS) calculation	50
rot_struct	
Rotates the DQ axis in the opposite direction (clockwise)	52
signal	
signal_positioned	53
step_struct	
Step function generator	55
svpwm_struct	
Space Vector Pulse Width Modulation (SVPWM) implementation	58

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN1db.h	61
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN_e-Tech.h Header file for handling CAN communication with the car	74
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CONTROL.h Header file for control logic	78
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/FSM.h Header for the inverter Finite State Machine	82
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/INVERTER.h Header file for the inverter struct and extern variables	84
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/main.h : Header for <code>main.c</code> file. This file contains the common defines of the application	90
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MEASUREMENTS.h Header file for handling measurements	103
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MOTOR.h Header file for motor parameters	111
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PCB_IO.h Header file for handling GPIOs	113
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PergaMOD.h	118
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PWM.h Header file for controlling PWM output	123
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/REFERENCE.h Header file for torque reference handling	126
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/stm32f7xx_it.h This file contains the headers of the interrupt handlers	132
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_1ms.h Header file for functions related to tasks executed every 1ms	138
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_20us.h Header file for functions related to tasks executed every 20us in each PWM timer interruption .	140
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/can.c This file provides code for the configuration of the CAN instances	142
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CAN_e-Tech.c This file contains functions to handle CAN communication with the car	144
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CONTROL.c This file provides code for the control loop	146
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/FSM.c This file provides code for the inverter Finite State Machine	149

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ INVERTER.c	150
This file provides code for the inverter struct	
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ main.c	154
: Main program body	
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ MEASUREMENTS.c	157
This file provides functions for handling measurements	
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ MOTOR.c	181
Source file for motor parameters	
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ PCB_IO.c	184
This file provides functions for handling GPIOs	
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ PergaMOD.c	186
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ PWM.c	187
This file provides functions for controlling PWM output	
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ REFERENCE.c	189
Source file for torque reference handling	
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ stm32f7xx_it.c	194
Interrupt Service Routines	
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ TASKS_1ms.c	200
This file contains functions to execute tasks every 1ms	
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ TASKS_20us.c	202
This file contains functions executed every 20us in each PWM timer interruption	
C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/ tim.c	205
This file provides code for the configuration of the TIM instances	

Chapter 4

Topic Documentation

4.1 Math Constants

Macros

- #define **SQ2** 1.4142135624F
- #define **ISQ2** 0.7071067812F
- #define **SQ3** 1.7320508076F
- #define **ISQ3** 0.5773502692F
- #define **PI** 3.1415926536F
- #define **IPI** 0.3183098862F
- #define **PI2** 6.2831853072F
- #define **IPI2** 0.1591549431F
- #define **INV_DEG** 0.0027777778F
- #define **INV3** 0.3333333333F
- #define **DIV2** 0.5F

4.1.1 Detailed Description

4.1.2 Macro Definition Documentation

4.1.2.1 DIV2

```
#define DIV2 0.5F
```

1/2

4.1.2.2 INV3

```
#define INV3 0.3333333333F
```

Inverse of 3

4.1.2.3 INV_DEG

```
#define INV_DEG 0.002777778F
```

Inverse of 360

4.1.2.4 IPI

```
#define IPI 0.3183098862F
```

Inverse of Pi

4.1.2.5 IPI2

```
#define IPI2 0.1591549431F
```

Inverse of (2*Pi)

4.1.2.6 ISQ2

```
#define ISQ2 0.7071067812F
```

Inverse of square root of 2

4.1.2.7 ISQ3

```
#define ISQ3 0.5773502692F
```

Inverse of square root of 3

4.1.2.8 PI

```
#define PI 3.1415926536F
```

Pi

4.1.2.9 PI2

```
#define PI2 6.2831853072F
```

2*Pi

4.1.2.10 SQ2

```
#define SQ2 1.4142135624F
```

Square root of 2

4.1.2.11 SQ3

```
#define SQ3 1.7320508076F
```

Square root of 3

4.2 - Integral Controllers

Data Structures

- struct [pi_aw_struct](#)
PI Controller with internal saturation, anti-windup, and feedforward.
- struct [pi_struct](#)
PI Controller with external saturation and feedforward.

Functions

- void [pi_aw_calc](#) (volatile [pi_aw_struct](#) *v) __attribute__((section(".ccmram")))
Initializes the PI controller with anti-windup.
- void [pi_init](#) (volatile [pi_struct](#) *v) __attribute__((section(".ccmram")))
Initializes the PI controller.
- void [pi_calc](#) (volatile [pi_struct](#) *v) __attribute__((section(".ccmram")))
Calculates the output of the PI controller.
- void [pi_extsat_calc](#) (volatile [pi_struct](#) *v)
Calculates the output of the PI controller with external saturation.

4.2.1 Detailed Description

4.2.2 Function Documentation

4.2.2.1 [pi_aw_calc\(\)](#)

```
void pi_aw_calc (
    volatile pi\_aw\_struct * v )
```

Initializes the PI controller with anti-windup.

Parameters

v	Pointer to the PI controller structure.
---	---

Initializes the PI controller with anti-windup.

Parameters

v	Pointer to the PI control structure.
---	--------------------------------------

Note

This function computes the PI control action with anti-windup.

4.2.2.2 pi_calc()

```
void pi_calc (
    volatile pi_struct * v )
```

Calculates the output of the PI controller.

Parameters

<i>v</i>	Pointer to the PI controller structure.
----------	---

Calculates the output of the PI controller.

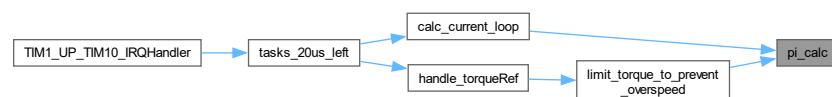
Parameters

<i>v</i>	Pointer to the PI structure.
----------	------------------------------

Note

This function computes the PI control action with feedforward and saturation.

Here is the caller graph for this function:

**4.2.2.3 pi_extsat_calc()**

```
void pi_extsat_calc (
    volatile pi_struct * v )
```

Calculates the output of the PI controller with external saturation.

Parameters

<i>v</i>	Pointer to the PI controller structure.
----------	---

Calculates the output of the PI controller with external saturation.

Parameters

<code>v</code>	Pointer to the PI structure.
----------------	------------------------------

Note

This function computes the PI control action without saturation for external saturation.

4.2.2.4 pi_init()

```
void pi_init (
    volatile pi_struct * v )
```

Initializes the PI controller.

Parameters

<code>v</code>	Pointer to the PI controller structure.
----------------	---

Initializes the PI controller.

Parameters

<code>v</code>	Pointer to the PI structure.
----------------	------------------------------

Note

This function initializes the constants used in the PI controller.

Here is the caller graph for this function:

**4.3 Clarke and Park Transformations****Data Structures**

- struct `clarke3F_struct`
Clarke transformation for three-phase systems.
- struct `iclarke3F_struct`
Inverse Clarke transformation for three-phase systems.
- struct `rot_struct`
Rotates the DQ axis in the opposite direction (clockwise).
- struct `irot_struct`
Inverse rotation (counterclockwise).

Functions

- void `clarke3F_calc` (volatile `clarke3F_struct` *v) `__attribute__((section(".ccmram")))`
Calculates the Clarke transformation.
- void `iclarke3F_calc` (volatile `iclarke3F_struct` *v)
Calculates the inverse Clarke transformation.
- void `rot_calc` (volatile `rot_struct` *v) `__attribute__((section(".ccmram")))`
Calculates the rotation in the opposite direction.
- void `irot_calc` (volatile `irot_struct` *v) `__attribute__((section(".ccmram")))`
Calculates the inverse rotation.

4.3.1 Detailed Description

4.3.2 Function Documentation

4.3.2.1 clarke3F_calc()

```
void clarke3F_calc (
    volatile clarke3F_struct * v )
```

Calculates the Clarke transformation.

Parameters

<code>v</code>	Pointer to the Clarke transformation structure.
----------------	---

Calculates the Clarke transformation.

Parameters

<code>v</code>	Pointer to the Clarke transformation structure.
----------------	---

Note

This function computes the Clarke transformation for three-phase signals.

4.3.2.2 iclarke3F_calc()

```
void iclarke3F_calc (
    volatile iclarke3F_struct * v )
```

Calculates the inverse Clarke transformation.

Parameters

<code>v</code>	Pointer to the inverse Clarke transformation structure.
----------------	---

Calculates the inverse Clarke transformation.

Parameters

v	Pointer to the inverse Clarke transformation structure.
---	---

Note

This function computes the inverse Clarke transformation for three-phase signals.

4.3.2.3 irot_calc()

```
void irot_calc (
    volatile irot_struct * v )
```

Calculates the inverse rotation.

Parameters

v	Pointer to the inverse rotation structure.
---	--

Calculates the inverse rotation.

Parameters

v	Pointer to the inverse rotation transformation structure.
---	---

Note

This function computes the inverse rotation transformation (counterclockwise).

4.3.2.4 rot_calc()

```
void rot_calc (
    volatile rot_struct * v )
```

Calculates the rotation in the opposite direction.

Parameters

v	Pointer to the rotation structure.
---	------------------------------------

Calculates the rotation in the opposite direction.

Parameters

v	Pointer to the rotation transformation structure.
---	---

Note

This function computes the rotation transformation (clockwise).

4.4 Utility Functions

Data Structures

- struct `angle_struct`
Generates an angle based on a fixed frequency.
- struct `svpwm_struct`
Space Vector Pulse Width Modulation (SVPWM) implementation.

Functions

- void `angle_calc` (volatile `angle_struct` **p*) `__attribute__((section(".ccmram")))`
Calculates the angle generation.
- void `svpwm_calc` (volatile `svpwm_struct` **v*) `__attribute__((section(".ccmram")))`
Calculates the SVPWM outputs.

4.4.1 Detailed Description

4.4.2 Function Documentation

4.4.2.1 `angle_calc()`

```
void angle_calc (
    volatile angle_struct * v )
```

Calculates the angle generation.

Parameters

<i>p</i>	Pointer to the angle structure.
----------	---------------------------------

Calculates the angle generation.

Parameters

<i>v</i>	Pointer to the angle generation structure.
----------	--

Note

This function generates the angle.

Here is the caller graph for this function:



4.4.2.2 svpwm_calc()

```
void svpwm_calc (
    volatile svpwm_struct * v )
```

Calculates the SVPWM outputs.

Parameters

v	Pointer to the SVPWM structure.
---	---------------------------------

Calculates the SVPWM outputs.

Parameters

v	Pointer to the SVPWM structure.
---	---------------------------------

Note

This function calculates the Space Vector Pulse Width Modulation (SVPWM).

Here is the caller graph for this function:



4.5 Signal Processing Functions

Data Structures

- struct rampa_struct

- struct `rampa_dual_struct`
 - Single-ramp generator.*
- struct `datalog_struct`
 - Dual-ramp generator.*
- struct `avg_struct_10`
 - Moving average filter for 10 samples.*
- struct `RMS_struct`
 - Root Mean Square (RMS) calculation.*
- struct `filtreLP_struct`
 - First-order low-pass filter.*

Macros

- `#define N_DATALOG 256`
 - Data logger for logging variables.*

Functions

- void `rampa_calc` (volatile `rampa_struct` *v) `__attribute__((section(".ccmram")))`
 - Calculates the output of the single ramp generator.*
- void `rampa_dual_calc` (volatile `rampa_dual_struct` *v) `__attribute__((section(".ccmram")))`
 - Calculates the output of the dual ramp generator.*
- void `datalog_calc` (volatile `datalog_struct` *dl)
 - Calculates the data log.*
- void `avg_calc_10_samples` (volatile `avg_struct_10` *v)
 - Calculates the moving average for 10 samples.*
- void `RMS_calc` (volatile `RMS_struct` *v) `__attribute__((section(".ccmram")))`
 - Calculates the RMS.*
- void `filtreLP_init` (volatile `filtreLP_struct` *v)
 - Initializes the first-order low-pass filter.*
- void `filtreLP_calc` (volatile `filtreLP_struct` *v) `__attribute__((section(".ccmram")))`
 - Calculates the output of the first-order low-pass filter.*

4.5.1 Detailed Description

4.5.2 Macro Definition Documentation

4.5.2.1 N_DATALOG

```
#define N_DATALOG 256
```

Data logger for logging variables.

Number of samples for the data log

4.5.3 Function Documentation

4.5.3.1 avg_calc_10_samples()

```
void avg_calc_10_samples (
    volatile avg_struct_10 * v )
```

Calculates the moving average for 10 samples.

Parameters

<i>v</i>	Pointer to the moving average structure.
----------	--

Calculates the moving average for 10 samples.

Parameters

<i>v</i>	Pointer to the structure containing the samples.
----------	--

Note

This function calculates the average of 10 samples.

4.5.3.2 **datalog_calc()**

```
void datalog_calc (
    volatile datalog\_struct * dl )
```

Calculates the data log.

Parameters

<i>dl</i>	Pointer to the data log structure.
<i>dl</i>	Pointer to the data log structure.

Note

This function calculates the data log.

4.5.3.3 **filtreLP_calc()**

```
void filtreLP_calc (
    volatile filtreLP\_struct * v )
```

Calculates the output of the first-order low-pass filter.

Parameters

<i>v</i>	Pointer to the filter structure.
----------	----------------------------------

Calculates the output of the first-order low-pass filter.

Parameters

<i>v</i>	Pointer to the first-order filter structure.
----------	--

Note

This function calculates the first-order filter.

4.5.3.4 `filtreLP_init()`

```
void filtreLP_init (
    volatile filtreLP_struct * v )
```

Initializes the first-order low-pass filter.

Parameters

<code>v</code>	Pointer to the filter structure.
----------------	----------------------------------

Initializes the first-order low-pass filter.

Parameters

<code>v</code>	Pointer to the first-order filter structure.
----------------	--

Note

This function initializes the first-order filter.

4.5.3.5 `rampa_calc()`

```
void rampa_calc (
    volatile rampa_struct * v )
```

Calculates the output of the single ramp generator.

Parameters

<code>v</code>	Pointer to the single ramp generator structure.
----------------	---

Calculates the output of the single ramp generator.

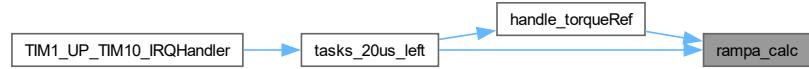
Parameters

<code>v</code>	Pointer to the ramp structure.
----------------	--------------------------------

Note

This function calculates the ramp.

Here is the caller graph for this function:



4.5.3.6 rampa_dual_calc()

```
void rampa_dual_calc (
    volatile rampa_dual_struct * v )
```

Calculates the output of the dual ramp generator.

Parameters

v	Pointer to the dual ramp generator structure.
---	---

Calculates the output of the dual ramp generator.

Parameters

v	Pointer to the dual ramp structure.
---	-------------------------------------

Note

This function calculates the dual ramp.

4.5.3.7 RMS_calc()

```
void RMS_calc (
    volatile RMS_struct * v )
```

Calculates the RMS.

Parameters

v	Pointer to the RMS structure.
---	-------------------------------

Calculates the RMS.

Parameters

v	Pointer to the RMS structure.
---	-------------------------------

Note

This function calculates the Root Mean Square (RMS).

4.6 Miscellaneous Functions

Data Structures

- struct `step_struct`

Step function generator.

Functions

- void `step_calc` (volatile `step_struct` *v) __attribute__((section(".ccmram"))))
Calculates the output of the step function generator.

4.6.1 Detailed Description

4.6.2 Function Documentation

4.6.2.1 `step_calc()`

```
void step_calc (
    volatile step_struct * v )
```

Calculates the output of the step function generator.

Parameters

v	Pointer to the step generator structure.
---	--

Calculates the output of the step function generator.

Parameters

v	Pointer to the step structure.
---	--------------------------------

Note

This function calculates the step function.

Chapter 5

Data Structure Documentation

5.1 Analog Struct Reference

Structure for ADC measurements in units.

```
#include <MEASUREMENTS.h>
```

Data Fields

- float `ia`
- float `ib`
- float `ic`
- float `vDC`

5.1.1 Detailed Description

Structure for ADC measurements in units.

5.1.2 Field Documentation

5.1.2.1 `ia`

```
float ia
```

Phase A current in A

5.1.2.2 `ib`

```
float ib
```

Phase B current in A

5.1.2.3 ic

```
float ic
```

Phase C current in A

5.1.2.4 vDC

```
float vDC
```

DC link voltage in V

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[MEASUREMENTS.h](#)

5.2 angle_struct Struct Reference

Generates an angle based on a fixed frequency.

```
#include <PergaMOD.h>
```

Data Fields

- float freq
- float Ts
- float angle
- void(* calc)()

5.2.1 Detailed Description

Generates an angle based on a fixed frequency.

5.2.2 Field Documentation

5.2.2.1 angle

```
float angle
```

Network angle

5.2.2.2 calc

```
void(* calc) ()
```

Pointer to the calculation function

5.2.2.3 freq

```
float freq
```

Frequency of the network

5.2.2.4 Ts

```
float Ts
```

Execution frequency of the angle integration

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[PergaMOD.h](#)

5.3 avg_struct_10 Struct Reference

Moving average filter for 10 samples.

```
#include <PergaMOD.h>
```

Data Fields

- float [out](#)
- float [in](#) [10]

5.3.1 Detailed Description

Moving average filter for 10 samples.

5.3.2 Field Documentation

5.3.2.1 in

```
float in[10]
```

Input and past samples

5.3.2.2 out

```
float out
```

Output variable

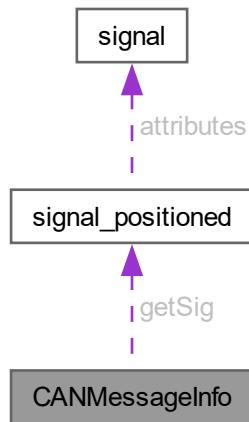
The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[PergaMOD.h](#)

5.4 CANMessageInfo Struct Reference

```
#include <CAN_e-Tech.h>
```

Collaboration diagram for CANMessageInfo:



Data Fields

- const uint32_t `ID`
- const uint8_t `IDE`
- const uint8_t `DLC`
- const `signal_positioned` * `getSig`

5.4.1 Field Documentation

5.4.1.1 DLC

```
const uint8_t DLC
```

5.4.1.2 getSig

```
const signal_positioned* getSig
```

5.4.1.3 ID

```
const uint32_t ID
```

5.4.1.4 IDE

```
const uint8_t IDE
```

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN_e-Tech.h

5.5 clarke3F_struct Struct Reference

Clarke transformation for three-phase systems.

```
#include <PergaMOD.h>
```

Data Fields

- float **a**
- float **b**
- float **D**
- float **Q**
- void(* **calc**)()

5.5.1 Detailed Description

Clarke transformation for three-phase systems.

5.5.2 Field Documentation

5.5.2.1 **a**

```
float a
```

Phase A input

5.5.2.2 **b**

```
float b
```

Phase B input

5.5.2.3 **calc**

```
void(* calc) ()
```

Pointer to the calculation function

5.5.2.4 D

float D

D-axis output

5.5.2.5 Q

float Q

Q-axis output

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[PergaMOD.h](#)

5.6 datalog_struct Struct Reference

```
#include <PergaMOD.h>
```

Data Fields

- `uint16_t i`
- `uint16_t j`
- `uint16_t estat`
- `uint16_t prescaler`
- `float * var`
- `void(* calc)()`
- `float log [N_DATALOG]`

5.6.1 Field Documentation

5.6.1.1 calc

```
void(* calc) ()
```

Pointer to the calculation function

5.6.1.2 estat

```
uint16_t estat
```

State: 0=stopped, 1=init, 2=running

5.6.1.3 i

```
uint16_t i
```

State variable

5.6.1.4 j

```
uint16_t j
```

State variable

5.6.1.5 log

```
float log[N_DATALOG]
```

Log array

5.6.1.6 prescaler

```
uint16_t prescaler
```

Prescaler

5.6.1.7 var

```
float* var
```

Pointer to variables

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[PergaMOD.h](#)

5.7 Duties Struct Reference

Structure to hold PWM configuration parameters.

```
#include <PWM.h>
```

Data Fields

- float [Da](#)
- float [Db](#)
- float [Dc](#)

5.7.1 Detailed Description

Structure to hold PWM configuration parameters.

5.7.2 Field Documentation

5.7.2.1 Da

```
float Da
```

Duty cycle for channel 1

5.7.2.2 Db

```
float Db
```

Duty cycle for channel 2

5.7.2.3 Dc

```
float Dc
```

Duty cycle for channel 3

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[PWM.h](#)

5.8 Encoder Struct Reference

Structure for encoder reading.

```
#include <MEASUREMENTS.h>
```

Data Fields

- `uint16_t A`
- `uint16_t B`
- `uint16_t Z`
- `float we`
- `float theta_e`
- `float sinTheta_e`
- `float cosTheta_e`
- `uint8_t directionMeas`

5.8.1 Detailed Description

Structure for encoder reading.

5.8.2 Field Documentation

5.8.2.1 A

```
uint16_t A
```

[Encoder](#) channel A value

5.8.2.2 B

```
uint16_t B
```

[Encoder](#) channel B value

5.8.2.3 cosTheta_e

```
float cosTheta_e
```

Electrical rotor position cosine

5.8.2.4 directionMeas

```
uint8_t directionMeas
```

Measured direction

5.8.2.5 sinTheta_e

```
float sinTheta_e
```

Electrical rotor position sine

5.8.2.6 theta_e

```
float theta_e
```

Electrical rotor position

5.8.2.7 we

```
float we
```

Electrical angular velocity

5.8.2.8 Z

```
uint16_t Z
```

Encoder channel Z value

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[MEASUREMENTS.h](#)

5.9 Feedback Struct Reference

Structure for feedback values.

```
#include <MEASUREMENTS.h>
```

Data Fields

- float [idMeas](#)
- float [iqMeas](#)
- float [torqueCalc](#)
- float [speedMeas](#)

5.9.1 Detailed Description

Structure for feedback values.

5.9.2 Field Documentation

5.9.2.1 idMeas

```
float idMeas
```

Measured d-axis current in A

5.9.2.2 iqMeas

```
float iqMeas
```

Measured q-axis current in A

5.9.2.3 speedMeas

```
float speedMeas
```

Measured speed in RPM

5.9.2.4 torqueCalc

```
float torqueCalc
```

Calculated torque in N·m

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[MEASUREMENTS.h](#)

5.10 **filtreLP_struct** Struct Reference

First-order low-pass filter.

```
#include <PergaMOD.h>
```

Data Fields

- float **in**
- float **out**
- float **alfa**
- float **Ts**
- float **fc**
- uint16_t **enable**
- void(* **init**)()
- void(* **calc**)()

5.10.1 Detailed Description

First-order low-pass filter.

5.10.2 Field Documentation

5.10.2.1 alfa

```
float alfa
```

Filter coefficient

5.10.2.2 calc

```
void(* calc) ()
```

Pointer to the calculation function

5.10.2.3 enable

```
uint16_t enable
```

Enable flag

5.10.2.4 fc

```
float fc
```

Cutoff frequency

5.10.2.5 in

```
float in
```

Input signal

5.10.2.6 init

```
void(* init) ()
```

Pointer to the initialization function

5.10.2.7 out

```
float out
```

Output signal

5.10.2.8 Ts

```
float Ts
```

Execution period

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PergaMOD.h

5.11 iclarke3F_struct Struct Reference

Inverse Clarke transformation for three-phase systems.

```
#include <PergaMOD.h>
```

Data Fields

- float **D**
- float **Q**
- float **a**
- float **b**
- void(* **calc**)()

5.11.1 Detailed Description

Inverse Clarke transformation for three-phase systems.

5.11.2 Field Documentation

5.11.2.1 **a**

```
float a
```

Phase A output

5.11.2.2 **b**

```
float b
```

Phase B output

5.11.2.3 **calc**

```
void(* calc) ()
```

Pointer to the calculation function

5.11.2.4 **D**

```
float D
```

D-axis input

5.11.2.5 Q

float Q

Q-axis input

The documentation for this struct was generated from the following file:

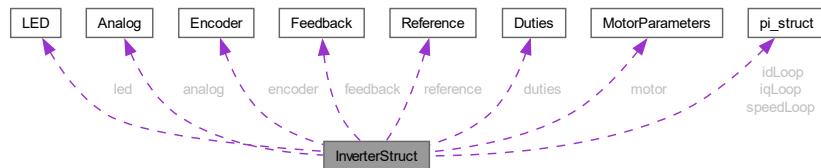
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PergaMOD.h

5.12 InverterStruct Struct Reference

Inverter structure.

```
#include <INVERTER.h>
```

Collaboration diagram for InverterStruct:



Data Fields

- `LED * led`
- `GPIO_TypeDef * enable_port`
- `uint16_t enable_pin`
- `TIM_HandleTypeDef * htim`
- `ADC_HandleTypeDef * hadc`
- `InverterState state`
- `Analog analog`
- `Encoder encoder`
- `Feedback feedback`
- `Reference reference`
- `Duties duties`
- `int8_t direction`
- `float templInverter`
- `float tempMotor`
- `MotorParameters * motor`
- `pi_struct idLoop`
- `pi_struct iqLoop`
- `pi_struct speedLoop`

5.12.1 Detailed Description

Inverter structure.

5.12.2 Field Documentation

5.12.2.1 analog

`Analog` `analog`

Structure for phase currents and DC voltage measurements

5.12.2.2 direction

`int8_t` `direction`

Motor direction: 1 CW, -1 CCW, 0 stopped

5.12.2.3 duties

`Duties` `duties`

Structure for duty cycles for phases A, B, and C

5.12.2.4 enable_pin

`uint16_t` `enable_pin`

Pin number for enabling/disabling the inverter

5.12.2.5 enable_port

`GPIO_TypeDef*` `enable_port`

Pointer to GPIO port for enabling/disabling the inverter

5.12.2.6 encoder

`Encoder` `encoder`

Structure for encoder input

5.12.2.7 feedback

`Feedback` feedback

Structure for measured currents and calculated mechanical torque and speed

5.12.2.8 hadc

`ADC_HandleTypeDef*` hadc

Handle of the ADC peripheral for current phase currents and DC voltage sensing

5.12.2.9 htim

`TIM_HandleTypeDef*` htim

Handle of the timer peripheral for PWM output

5.12.2.10 idLoop

`pi_struct` idLoop

PI controller for d-axis current

5.12.2.11 iqLoop

`pi_struct` iqLoop

PI controller for q-axis current

5.12.2.12 led

`LED*` led

Pointer to `LED` control structure

5.12.2.13 motor

`MotorParameters*` motor

Motor parameters struct

5.12.2.14 reference

`Reference` reference

Structure for referece currents and torque

5.12.2.15 speedLoop

```
pi_struct speedLoop
```

PI controller for motor speed

5.12.2.16 state

```
InverterState state
```

Current state of inverter operation

5.12.2.17 tempInverter

```
float tempInverter
```

Semiconductor temperature in degC

5.12.2.18 tempMotor

```
float tempMotor
```

Motor temperature in degC

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/INVERTER.h

5.13 irot_struct Struct Reference

Inverse rotation (counterclockwise).

```
#include <PergaMOD.h>
```

Data Fields

- float **d**
- float **q**
- float **sinFi**
- float **cosFi**
- float **alpha**
- float **beta**
- void(* **calc**)()

5.13.1 Detailed Description

Inverse rotation (counterclockwise).

5.13.2 Field Documentation

5.13.2.1 alpha

float alpha

alpha output

5.13.2.2 beta

float beta

beta output

5.13.2.3 calc

void(* calc) ()

Pointer to the calculation function

5.13.2.4 cosFi

float cosFi

Cosine of the angle of rotation

5.13.2.5 d

float d

D-axis input

5.13.2.6 q

float q

Q-axis input

5.13.2.7 sinFi

float sinFi

Sine of the angle of rotation

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PergaMOD.h

5.14 LED Struct Reference

[LED](#) structure.

```
#include <PCB_IO.h>
```

Data Fields

- `GPIO_TypeDef * port`
- `uint16_t pin`
- `LEDMode mode`

5.14.1 Detailed Description

[LED](#) structure.

5.14.2 Field Documentation

5.14.2.1 mode

`LEDMode mode`

Current [LED](#) mode

5.14.2.2 pin

`uint16_t pin`

Pin number for controlling the [LED](#)

5.14.2.3 port

`GPIO_TypeDef* port`

GPIO port for controlling the [LED](#)

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[PCB_IO.h](#)

5.15 MotorParameters Struct Reference

Structure to hold motor parameters.

```
#include <MOTOR.h>
```

Data Fields

- float `Ld`
- float `Lq`
- float `Rs`
- float `lambda`
- int `pp`
- float `J`
- float `b`
- float `torque_max`
- float `dTorque_max`
- float `speed_max_RPM`
- float `iPhase_pk_max`
- float `vDC_max`

5.15.1 Detailed Description

Structure to hold motor parameters.

5.15.2 Field Documentation

5.15.2.1 `b`

`float b`

Viscous friction in N·m·s

5.15.2.2 `dTorque_max`

`float dTorque_max`

Maximum torque increment in N·m/s

5.15.2.3 `iPhase_pk_max`

`float iPhase_pk_max`

Maximum phase current (peak value, or RMS*sqrt2)

5.15.2.4 `J`

`float J`

Rotational inertia in N·m·s²

5.15.2.5 lambda

```
float lambda
```

Magnet flux linkage measured $V_{pk_ph-n} \cdot s$ (phase-neutral peak voltage divided by electrical speed in rad/s)

5.15.2.6 Ld

```
float Ld
```

D-axis inductance in Henries

5.15.2.7 Lq

```
float Lq
```

Q-axis inductance in Henries

5.15.2.8 pp

```
int pp
```

Pole pairs (total number of poles divided by 2)

5.15.2.9 Rs

```
float Rs
```

Stator resistance in Ohms

5.15.2.10 speed_max_RPM

```
float speed_max_RPM
```

Maximum speed in RPM

5.15.2.11 torque_max

```
float torque_max
```

Maximum torque in N·m

5.15.2.12 vDC_max

```
float vDC_max
```

Maximum DC bus voltage in volts

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MOTOR.h

5.16 pi_aw_struct Struct Reference

PI Controller with internal saturation, anti-windup, and feedforward.

```
#include <PergaMOD.h>
```

Data Fields

- `uint16_t enable`
- `float Ts`
- `float Kp`
- `float Ki`
- `float Kaw`
- `float e [2]`
- `float pi_consig`
- `float pi_fdb`
- `float pi_out_max`
- `float pi_out_min`
- `float pi_out_presat`
- `float pi_out_postsat`
- `float pi_out`
- `float pi_int [2]`
- `float pi_ffw [2]`
- `void(* calc)()`

5.16.1 Detailed Description

PI Controller with internal saturation, anti-windup, and feedforward.

5.16.2 Field Documentation

5.16.2.1 calc

```
void(* calc) ()
```

Pointer to the calculation function

5.16.2.2 e

```
float e[2]
```

Error at current and previous step

5.16.2.3 enable

```
uint16_t enable
```

Enable flag for the controller

5.16.2.4 Kaw

```
float Kaw
```

Anti-windup gain

5.16.2.5 Ki

```
float Ki
```

Integral gain

5.16.2.6 Kp

```
float Kp
```

Proportional gain

5.16.2.7 pi_consig

```
float pi_consig
```

Setpoint

5.16.2.8 pi_fdb

```
float pi_fdb
```

[Feedback](#)

5.16.2.9 pi_ffw

```
float pi_ffw[2]
```

Feedforward at current and previous step

5.16.2.10 pi_int

```
float pi_int[2]
```

Integrator Part

5.16.2.11 pi_out

```
float pi_out
```

Controller output

5.16.2.12 pi_out_max

```
float pi_out_max
```

Maximum output

5.16.2.13 pi_out_min

```
float pi_out_min
```

Minimum output

5.16.2.14 pi_out_postsat

```
float pi_out_postsat
```

Output after saturation

5.16.2.15 pi_out_presat

```
float pi_out_presat
```

Output before saturation

5.16.2.16 Ts

```
float Ts
```

Execution period

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PergaMOD.h

5.17 pi_struct Struct Reference

PI Controller with external saturation and feedforward.

```
#include <PergaMOD.h>
```

Data Fields

- `uint16_t enable`
- `float Ts`
- `float Kp`
- `float Ki`
- `float K0`
- `float K1`
- `float e [2]`
- `float pi_consig`
- `float pi_fdb`
- `float pi_out_max`
- `float pi_out_min`
- `float pi_out`
- `float pi_ffw [2]`
- `void(* init)()`
- `void(* calc)()`

5.17.1 Detailed Description

PI Controller with external saturation and feedforward.

5.17.2 Field Documentation

5.17.2.1 calc

```
void(* calc) ()
```

Pointer to the calculation function

5.17.2.2 e

```
float e[2]
```

Error at current and previous step

5.17.2.3 enable

```
uint16_t enable
```

Enable flag for the controller

5.17.2.4 init

```
void(* init) ()
```

Pointer to the initialization function

5.17.2.5 K0

```
float K0
```

$$K0 = Kp + (Ts*Ki)/2$$

5.17.2.6 K1

```
float K1
```

$$K0 = -Kp + (Ts*Ki)/2$$

5.17.2.7 Ki

```
float Ki
```

Integral gain

5.17.2.8 Kp

```
float Kp
```

Proportional gain

5.17.2.9 pi_consig

```
float pi_consig
```

Setpoint

5.17.2.10 pi_fdb

```
float pi_fdb
```

[Feedback](#)

5.17.2.11 pi_ffw

```
float pi_ffw[2]
```

Feedforward at current and previous step

5.17.2.12 pi_out

float pi_out

Controller output

5.17.2.13 pi_out_max

float pi_out_max

Maximum output

5.17.2.14 pi_out_min

float pi_out_min

Minimum output

5.17.2.15 Ts

float Ts

Execution period

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PergaMOD.h

5.18 rampa_dual_struct Struct Reference

Dual-ramp generator.

```
#include <PergaMOD.h>
```

Data Fields

- float `in`
- float `out`
- float `Incr`
- float `Decr`
- uint8_t `enable`
- void(* `calc`)()

5.18.1 Detailed Description

Dual-ramp generator.

5.18.2 Field Documentation

5.18.2.1 calc

```
void(* calc) ()
```

Pointer to the calculation function

5.18.2.2 Decr

```
float Decr
```

Decrement

5.18.2.3 enable

```
uint8_t enable
```

Enable flag

5.18.2.4 in

```
float in
```

Input signal

5.18.2.5 Incr

```
float Incr
```

Increment

5.18.2.6 out

```
float out
```

Output signal

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PergaMOD.h

5.19 rampa_struct Struct Reference

Single-ramp generator.

```
#include <PergaMOD.h>
```

Data Fields

- float `in`
- float `out`
- float `Incr`
- uint8_t `enable`
- void(* `calc`)()

5.19.1 Detailed Description

Single-ramp generator.

5.19.2 Field Documentation

5.19.2.1 calc

```
void(* calc) ()
```

Pointer to the calculation function

5.19.2.2 enable

```
uint8_t enable
```

Enable flag

5.19.2.3 in

```
float in
```

Input signal

5.19.2.4 Incr

```
float Incr
```

Increment

5.19.2.5 out

```
float out
```

Output signal

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[PergaMOD.h](#)

5.20 Reference Struct Reference

Structure for reference values.

```
#include <REFERENCE.h>
```

Data Fields

- float `idRef`
- float `iqRef`
- float `torqueRef`

5.20.1 Detailed Description

Structure for reference values.

5.20.2 Field Documentation

5.20.2.1 `idRef`

```
float idRef
```

`Reference` d-axis current in A

5.20.2.2 `iqRef`

```
float iqRef
```

`Reference` q-axis current in A

5.20.2.3 `torqueRef`

```
float torqueRef
```

`Reference` torque in N·m

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/`REFERENCE.h`

5.21 RMS_struct Struct Reference

Root Mean Square (RMS) calculation.

```
#include <PergaMOD.h>
```

Data Fields

- float [T_exec](#)
- float [Measure](#)
- float [Sq_Sum](#)
- float [Out_RMS](#)
- float [Freq](#)
- float [Angle](#)
- float [Angle_ant](#)

5.21.1 Detailed Description

Root Mean Square (RMS) calculation.

5.21.2 Field Documentation

5.21.2.1 Angle

float Angle

Angle

5.21.2.2 Angle_ant

float Angle_ant

Previous angle

5.21.2.3 Freq

float Freq

Output frequency of the PLL

5.21.2.4 Measure

float Measure

Signal to be RMSed

5.21.2.5 Out_RMS

float Out_RMS

RMSed signal

5.21.2.6 Sq_Sum

float Sq_Sum

Sum of squares

5.21.2.7 T_exec

float T_exec

Execution frequency of the function

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[PergaMOD.h](#)

5.22 rot_struct Struct Reference

Rotates the DQ axis in the opposite direction (clockwise).

```
#include <PergaMOD.h>
```

Data Fields

- float **D**
- float **Q**
- float **sinFi**
- float **cosFi**
- float **d**
- float **q**
- void(* **calc**)()

5.22.1 Detailed Description

Rotates the DQ axis in the opposite direction (clockwise).

5.22.2 Field Documentation

5.22.2.1 calc

```
void(* calc) ()
```

Pointer to the calculation function

5.22.2.2 cosFi

```
float cosFi
```

Cosine of the angle of rotation

5.22.2.3 D

```
float D
```

D-axis input

5.22.2.4 d

```
float d
```

Rotated D-axis

5.22.2.5 Q

```
float Q
```

Q-axis input

5.22.2.6 q

```
float q
```

Rotated Q-axis

5.22.2.7 sinFi

```
float sinFi
```

Sine of the angle of rotation

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[PergaMOD.h](#)

5.23 signal Struct Reference

```
#include <CAN1db.h>
```

Data Fields

- const char * `name`
- const char * `unit`
- const `dbc_valueType` `valueType`
- const `uint8_t` `lengthBits`
- const float `factor`
- const float `offset`
- const float `min`
- const float `max`

5.23.1 Detailed Description

Signals map CAN data to physical signals and provide units, conversion factors and data types. The same as used in CANdb++.

5.23.2 Field Documentation

5.23.2.1 `factor`

```
const float factor
```

5.23.2.2 `lengthBits`

```
const uint8_t lengthBits
```

5.23.2.3 `max`

```
const float max
```

5.23.2.4 `min`

```
const float min
```

5.23.2.5 `name`

```
const char* name
```

5.23.2.6 `offset`

```
const float offset
```

5.23.2.7 unit

```
const char* unit
```

5.23.2.8 valueType

```
const dbc_valueType valueType
```

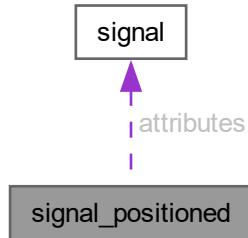
The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN1db.h

5.24 signal_positioned Struct Reference

```
#include <CAN1db.h>
```

Collaboration diagram for signal_positioned:



Data Fields

- const uint8_t `position`
- const `signal` * `attributes`

5.24.1 Detailed Description

Each message contains a struct of type `signal_positioned` to indicate the signal's position in the can frame and to allow access of the signals properties using a pointer.

5.24.2 Field Documentation

5.24.2.1 attributes

```
const signal* attributes
```

5.24.2.2 position

```
const uint8_t position
```

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN1db.h

5.25 step_struct Struct Reference

Step function generator.

```
#include <PergaMOD.h>
```

Data Fields

- float `fs`
- float `In`
- float `Out`
- float `Step`
- float `t_step`
- uint32_t `Pulses`
- uint32_t `Counter`
- uint16_t `enable`
- void(* `calc`)()

5.25.1 Detailed Description

Step function generator.

5.25.2 Field Documentation

5.25.2.1 calc

```
void(* calc) ()
```

Pointer to the calculation function

5.25.2.2 Counter

```
uint32_t Counter
```

Counter for pulses

5.25.2.3 enable

```
uint16_t enable
```

Enable flag

5.25.2.4 fs

```
float fs
```

Function execution frequency

5.25.2.5 In

```
float In
```

Input variable

5.25.2.6 Out

```
float Out
```

Output variable (with step when needed)

5.25.2.7 Pulses

```
uint32_t Pulses
```

Pulse counter for seconds

5.25.2.8 Step

```
float Step
```

Step amplitude

5.25.2.9 t_step

```
float t_step
```

Step duration in seconds

The documentation for this struct was generated from the following file:

- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PergaMOD.h

5.26 svpwm_struct Struct Reference

Space Vector Pulse Width Modulation (SVPWM) implementation.

```
#include <PergaMOD.h>
```

Data Fields

- float `alpha`
- float `beta`
- float `Da`
- float `Db`
- float `Dc`
- void(* `calc`)()

5.26.1 Detailed Description

Space Vector Pulse Width Modulation (SVPWM) implementation.

5.26.2 Field Documentation

5.26.2.1 `alpha`

```
float alpha
```

Input in per-unit for alpha phase (0deg)

5.26.2.2 `beta`

```
float beta
```

Input in per-unit for beta phase (90deg)

5.26.2.3 `calc`

```
void(* calc) ()
```

Pointer to the calculation function

5.26.2.4 `Da`

```
float Da
```

Output for phase A (0-1)

5.26.2.5 Db

float Db

Output for phase B (0-1)

5.26.2.6 Dc

float Dc

Output for phase C (0-1)

The documentation for this struct was generated from the following file:

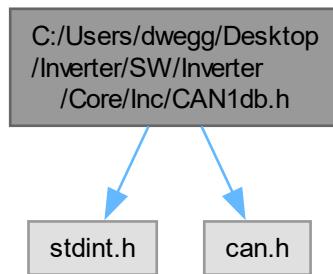
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/[PergaMOD.h](#)

Chapter 6

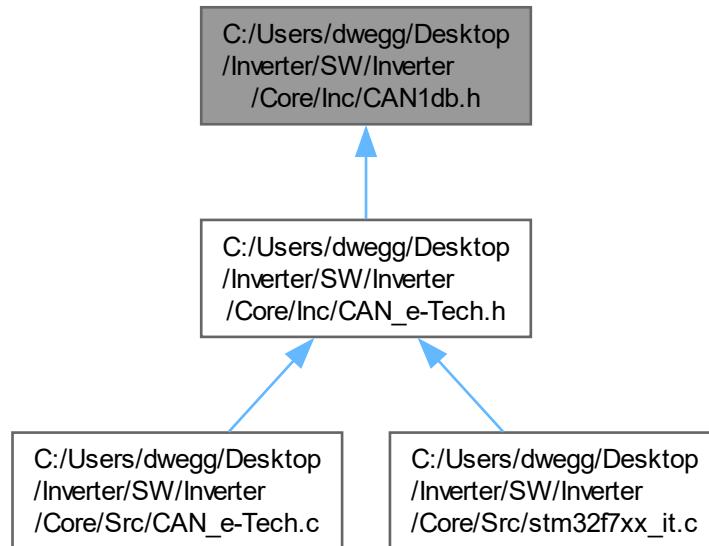
File Documentation

6.1 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN1db.h File Reference

```
#include "stdint.h"
#include "can.h"
Include dependency graph for CAN1db.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [signal](#)
- struct [signal_positioned](#)
- struct [DBC_INVERTERS_MSG_AP_ETAS_EnableInv](#)
- struct [DBC_INVERTERS_MSG_TBD_Inv_R_Power](#)
- struct [DBC_INVERTERS_MSG_TBD_Inv_R_Torque](#)
- struct [DBC_INVERTERS_MSG_AP_Inv_R_Error](#)
- struct [DBC_INVERTERS_MSG_AP_Inv_R_Speed](#)
- struct [DBC_INVERTERS_MSG_AP_Inv_R_DCBus](#)
- struct [DBC_INVERTERS_MSG_MEAS_Inv_R_Currents_R](#)
- struct [DBC_INVERTERS_MSG_MEAS_Inv_R_Currents_L](#)
- struct [DBC_INVERTERS_MSG_AP_Inv_R_Temperatures](#)

Macros

- #define [CAN_ID_STD](#) ((uint32_t)0x00000000U)
- #define [CAN_ID_EXT](#) ((uint32_t)0x00000004U)
- #define [DBC_linkMsgConverter](#)(dbc, arr) (dbc).[getSigVal](#) = (typeof((dbc).[getSigVal](#))) &(arr)
- #define [DBC_setup](#)(dbc, MSG)

Typedefs

- typedef enum [dbc_valueType](#) [dbc_valueType](#)
- typedef struct [signal](#) [signal](#)
- typedef struct [signal_positioned](#) [signal_positioned](#)

Enumerations

- enum `dbc_valueType` {
 `dbc_valueType_unsigned` = 0 , `dbc_valueType_signed` , `dbc_valueType_float` , `dbc_valueType_double` ,
 `dbc_valueType_void` }

Functions

- struct `__attribute__((packed)) DBC_INVERTERS_MSG_AP_ETAS_EnableInv_getSigVal`

Variables

- const `uint32_t ID`
- const `uint8_t IDE`
- const `uint8_t DLC`
- struct {
 `signal_positioned Enable_Inv_R`
} `getSig`
- * `getSigVal`

6.1.1 Macro Definition Documentation

6.1.1.1 CAN_ID_EXT

```
#define CAN_ID_EXT ((uint32_t)0x00000004U)
```

Extended Id as per STM32 HAL

6.1.1.2 CAN_ID_STD

```
#define CAN_ID_STD ((uint32_t)0x00000000U)
```

Standard Id

6.1.1.3 DBC_linkMsgConverter

```
#define DBC_linkMsgConverter(
    dbc,
    arr ) (dbc).getSigVal = (typeof((dbc).getSigVal)) &(arr)
```

6.1.1.4 DBC_setup

```
#define DBC_setup(
    dbc,
    MSG )
```

Value:

```
do { \
    (MSG).DLC = (dbc).DLC; \
    (MSG).IDE = (dbc).IDE; \
    (MSG).RTR = 0; \
    (MSG).ExtId = (MSG).IDE > 0 ? (dbc).ID : 0; \
    (MSG).StdId = (MSG).IDE > 0 ? 0 : (dbc).ID; \
} while (0)
```

6.1.2 Typedef Documentation

6.1.2.1 dbc_valueType

```
typedef enum dbc_valueType dbc_valueType
```

Data value types are defined here. Same list as is available in Vector CANdb++ for signal data types.

6.1.2.2 signal

```
typedef struct signal signal
```

Signals map CAN data to physical signals and provide units, conversion factors and data types. The same as used in CANdb++.

6.1.2.3 signal_positioned

```
typedef struct signal_positioned signal_positioned
```

Each message contains a struct of type [signal_positioned](#) to indicate the signal's position in the can frame and to allow access of the signals properties using a pointer.

6.1.3 Enumeration Type Documentation

6.1.3.1 dbc_valueType

```
enum dbc_valueType
```

Data value types are defined here. Same list as is available in Vector CANdb++ for signal data types.

Enumerator

dbc_valueType_unsigned	
dbc_valueType_signed	
dbc_valueType_float	
dbc_valueType_double	
dbc_valueType_void	

6.1.4 Function Documentation

6.1.4.1 __attribute__()

```
struct __attribute__ (
    (packed) )
```

6.1.5 Variable Documentation

6.1.5.1 DCBus_RL

```
signal_positioned DCBus_RL
```

6.1.5.2 DCBus_RR

```
signal_positioned DCBus_RR
```

6.1.5.3 DLC

```
const uint8_t DLC
```

6.1.5.4 Enable_Inv_R

```
signal_positioned Enable_Inv_R
```

6.1.5.5 [struct]

```
const struct { ... } getSig
```

6.1.5.6 getSigVal

```
* getSigVal
```

6.1.5.7 Iactual_RL

```
signal_positioned Iactual_RL
```

6.1.5.8 Iactual_RR

```
signal_positioned Iactual_RR
```

6.1.5.9 Icmd_RL

```
signal_positioned Icmd_RL
```

6.1.5.10 Icmd_RR

```
signal_positioned Icmd_RR
```

6.1.5.11 ID

```
const uint32_t ID
```

6.1.5.12 IDE

```
const uint8_t IDE
```

6.1.5.13 Iq_RL

```
signal_positioned Iq_RL
```

6.1.5.14 Iq_RR

```
signal_positioned Iq_RR
```

6.1.5.15 Power_RL

```
signal_positioned Power_RL
```

6.1.5.16 Power_RR

```
signal_positioned Power_RR
```

6.1.5.17 SpeedMotor_RL

```
signal_positioned SpeedMotor_RL
```

6.1.5.18 SpeedMotor_RR

```
signal_positioned SpeedMotor_RR
```

6.1.5.19 TempIGBT_RL

```
signal_positioned TempIGBT_RL
```

6.1.5.20 TempIGBT_RR

```
signal_positioned TempIGBT_RR
```

6.1.5.21 TempMotor_RL

```
signal_positioned TempMotor_RL
```

6.1.5.22 TempMotor_RR

```
signal_positioned TempMotor_RR
```

6.1.5.23 TorqueReal_RL

```
signal_positioned TorqueReal_RL
```

6.1.5.24 TorqueReal_RR

```
signal_positioned TorqueReal_RR
```

6.1.5.25 TotalPower

```
signal_positioned TotalPower
```

6.2 CAN1db.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * @formatter:off
00003  * Inverters.h
00004  *
00005  * Created on: mayo 12, 2024 22:01:14 :: Software Version: 0.5beta
00006  * Author: Valentin Felsner
00007  * Copyright by IMED Ltd :: http://www.imed.co.nz.
00008  *
00009  * This is a .dbc to .h C header converter.
00010  * It converts a Vector CANdb++ .dbc file to CAN message and signal structs to make them
00011  * more easily human readable during development and support auto-completion in eclipse-based
IDEs.
00012  *
00013  * This software is a beta version and provided "as is" - neither IMED Ltd nor the author
00014  * take any responsibility for the results. It may be used free of charge but it is not permitted
00015  * to redistribute the software nor to decompile and/or modify the original converter source
code.
00016  * Please send us an email to contact@imed.co.nz if you have any feature requests or bug reports,
thank you.
00017  * And if you or your business do find this software useful, come to New Zealand and join us for
a beer some time!
00018 */
00019
00020 #ifndef DBC_INVERTERS_H
00021 #define DBC_INVERTERS_H
00022
00023 #ifdef __cplusplus
00024 extern "C" {
00025 #endif /* __cplusplus */
00026
00027 /* Only define if not using STM Cube */
00028 #ifndef USE_HAL_DRIVER
00029     #ifndef CAN_ID_STD
00030         #define CAN_ID_STD      ((uint32_t)0x00000000U)
00031     #endif
00032     #ifndef CAN_ID_EXT
00033         #define CAN_ID_EXT      ((uint32_t)0x00000004U)
00034     #endif

```

```

00035 #endif
00036 #include "stdint.h"
00037 #include "can.h"
00038
00039 /*
00040 */
00041 * Helper function to link a converter struct pointer to a CAN message array (uint_8t[8])
00042 * This converter struct acts like a "getSigVal()" function.
00043 * Usage example:
00044 * DBC_linkMsgConverter(DBC_IMED_HELMBUS_MSG_BTN_STATUS, RxMessage);
00045 *
00046 * -> DBC_IMED_HELMBUS_MSG_BTN_STATUS is a generated CANdb message struct.
00047 * -> RxMessage is an uint8_t array[8] as received or sent by the can peripheral.
00048 *
00049 * Now, the message data can be accessed like this:
00050 * uint32_t uid = DBC_IMED_HELMBUS_MSG_BTN_STATUS.getSigVal->Board_UID;
00051 *
00052 * -> or whatever signals are defined in that message.
00053 *
00054 */
00055 ifndef DBC_linkMsgConverter
00056 define DBC_linkMsgConverter(dbc, arr) (dbc).getSigVal = (typeof((dbc).getSigVal)) &(arr)
00057 define DBC_setup(dbc, MSG) \
00058 do { \
00059     (MSG).DLC = (dbc).DLC; \
00060     (MSG).IDE = (dbc).IDE; \
00061     (MSG).RTR = 0; \
00062     (MSG).ExtId = (MSG).IDE > 0 ? (dbc).ID : 0; \
00063     (MSG).StdId = (MSG).IDE > 0 ? 0 : (dbc).ID; \
00064 } while (0)
00065
00066 /*
00067 */
00068 * Type definitions
00069 *
00070 */
00071
00072 typedef enum dbc_valueType {
00073     dbc_valueType_unsigned = 0, dbc_valueType_signed, dbc_valueType_float, dbc_valueType_double,
00074     dbc_valueType_void
00075 } dbc_valueType;
00076
00077 /*
00078 */
00079
00080 */
00081 * ValueTables
00082 *
00083 */
00084
00085
00086
00087 typedef struct signal {
00088     const char* name;
00089     const char* unit;
00090     const dbc_valueType valueType;
00091     const uint8_t lengthBits;
00092     const float factor, offset, min, max;
00093 } signal;
00094
00095 typedef struct signal_positioned {
00096     const uint8_t position;
00097     const signal* attributes;
00098 } signal_positioned;
00099
00100 #pragma GCC diagnostic push
00101 #pragma GCC diagnostic ignored "-Wpedantic"
00102
00103 /*
00104 */
00105
00106 */
00107 #endif
00108
00109 #pragma GCC diagnostic push
00110 #pragma GCC diagnostic ignored "-Wpedantic"
00111
00112
00113 /*
00114 */
00115 * Signals
00116 *
00117 */
00118
00119 static const signal DBC_INVERTERS_SIG_Enable_Inv_R = {
00120     .name = "Enable_Inv_R",
00121     .unit = "-",
00122     .valueType = dbc_valueType_unsigned,
00123     .lengthBits = 1,

```

```
00128     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 1.0f
00129 };
00130
00135 static const signal DBC_INVERTERS_SIG_TotalPower = {
00136     .name = "TotalPower",
00137     .unit = "kW",
00138     .valueType = dbc_valueType_unsigned,
00139     .lengthBits = 16,
00140     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00141 };
00142
00147 static const signal DBC_INVERTERS_SIG_Power_RR = {
00148     .name = "Power_RR",
00149     .unit = "kW",
00150     .valueType = dbc_valueType_unsigned,
00151     .lengthBits = 16,
00152     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00153 };
00154
00159 static const signal DBC_INVERTERS_SIG_Power_RL = {
00160     .name = "Power_RL",
00161     .unit = "kW",
00162     .valueType = dbc_valueType_unsigned,
00163     .lengthBits = 16,
00164     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00165 };
00166
00171 static const signal DBC_INVERTERS_SIG_TorqueReal_RR = {
00172     .name = "TorqueReal_RR",
00173     .unit = "Nm",
00174     .valueType = dbc_valueType_unsigned,
00175     .lengthBits = 16,
00176     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00177 };
00178
00183 static const signal DBC_INVERTERS_SIG_TorqueReal_RL = {
00184     .name = "TorqueReal_RL",
00185     .unit = "Nm",
00186     .valueType = dbc_valueType_unsigned,
00187     .lengthBits = 16,
00188     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00189 };
00190
00195 static const signal DBC_INVERTERS_SIG_SpeedMotor_RR = {
00196     .name = "SpeedMotor_RR",
00197     .unit = "rpm",
00198     .valueType = dbc_valueType_unsigned,
00199     .lengthBits = 16,
00200     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00201 };
00202
00207 static const signal DBC_INVERTERS_SIG_SpeedMotor_RL = {
00208     .name = "SpeedMotor_RL",
00209     .unit = "rpm",
00210     .valueType = dbc_valueType_unsigned,
00211     .lengthBits = 16,
00212     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00213 };
00214
00219 static const signal DBC_INVERTERS_SIG_DCBus_RR = {
00220     .name = "DCBus_RR",
00221     .unit = "V",
00222     .valueType = dbc_valueType_unsigned,
00223     .lengthBits = 16,
00224     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00225 };
00226
00231 static const signal DBC_INVERTERS_SIG_DCBus_RL = {
00232     .name = "DCBus_RL",
00233     .unit = "V",
00234     .valueType = dbc_valueType_unsigned,
00235     .lengthBits = 16,
00236     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00237 };
00238
00243 static const signal DBC_INVERTERS_SIG_Iq_RR = {
00244     .name = "Iq_RR",
00245     .unit = "A",
00246     .valueType = dbc_valueType_unsigned,
00247     .lengthBits = 16,
00248     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00249 };
00250
00255 static const signal DBC_INVERTERS_SIG_Icmd_RR = {
00256     .name = "Icmd_RR",
00257     .unit = "A",
00258     .valueType = dbc_valueType_unsigned,
```

```

00259     .lengthBits = 16,
00260     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00261 };
00262
00267 static const signal DBC_INVERTERS_SIG_Iactual_RR = {
00268     .name = "Iactual_RR",
00269     .unit = "A",
00270     .valueType = dbc_valueType_unsigned,
00271     .lengthBits = 16,
00272     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00273 };
00274
00279 static const signal DBC_INVERTERS_SIG_Iq_RL = {
00280     .name = "Iq_RL",
00281     .unit = "A",
00282     .valueType = dbc_valueType_unsigned,
00283     .lengthBits = 16,
00284     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00285 };
00286
00291 static const signal DBC_INVERTERS_SIG_Icmd_RL = {
00292     .name = "Icmd_RL",
00293     .unit = "A",
00294     .valueType = dbc_valueType_unsigned,
00295     .lengthBits = 16,
00296     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00297 };
00298
00303 static const signal DBC_INVERTERS_SIG_Iactual_RL = {
00304     .name = "Iactual_RL",
00305     .unit = "A",
00306     .valueType = dbc_valueType_unsigned,
00307     .lengthBits = 16,
00308     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00309 };
00310
00315 static const signal DBC_INVERTERS_SIG_TempMotor_RR = {
00316     .name = "TempMotor_RR",
00317     .unit = "°C",
00318     .valueType = dbc_valueType_unsigned,
00319     .lengthBits = 16,
00320     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00321 };
00322
00327 static const signal DBC_INVERTERS_SIG_TempMotor_RL = {
00328     .name = "TempMotor_RL",
00329     .unit = "°C",
00330     .valueType = dbc_valueType_unsigned,
00331     .lengthBits = 16,
00332     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00333 };
00334
00339 static const signal DBC_INVERTERS_SIG_TempIGBT_RR = {
00340     .name = "TempIGBT_RR",
00341     .unit = "°C",
00342     .valueType = dbc_valueType_unsigned,
00343     .lengthBits = 16,
00344     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00345 };
00346
00351 static const signal DBC_INVERTERS_SIG_TempIGBT_RL = {
00352     .name = "TempIGBT_RL",
00353     .unit = "°C",
00354     .valueType = dbc_valueType_unsigned,
00355     .lengthBits = 16,
00356     .factor = 1.0f, .offset = 0.0f, .min = 0.0f, .max = 65535.0f
00357 };
00358
00359
00360 /**
00361 *
-----*
00362 *          Messages
00363 *
-----*/
00364 */
00365
00370 static struct DBC_INVERTERS_MSG_AP_ETAS_EnableInv {
00371     const uint32_t ID;
00372     const uint8_t IDE;
00373     const uint8_t DLC;
00374     const struct {
00375         signal_positioned Enable_Inv_R;
00376     } getSig;
00377     struct __attribute__((packed)) DBC_INVERTERS_MSG_AP_ETAS_EnableInv_getSigVal {
00378         uint8_t Enable_Inv_R : 1;
00379     } *getSigVal;

```

```

00380 } DBC_INVERTERS_MSG_AP_ETAS_EnableInv __attribute__((unused)) = {
00381     .ID = 0xB1,
00382     .IDE = CAN_ID_STD,
00383     .DLC = 8,
00384     .getSig = {
00385         .Enable_Inv_R = {
00386             .position = 0,
00387             .attributes = &DBC_INVERTERS_SIG_Enable_Inv_R
00388         }
00389     },
00390     .getSigVal = 0
00391 };
00392
00393
00398 static struct DBC_INVERTERS_MSG_TBD_Inv_R_Power {
00399     const uint32_t ID;
00400     const uint8_t IDE;
00401     const uint8_t DLC;
00402     const struct {
00403         signal_positioned TotalPower;
00404         signal_positioned Power_RL;
00405         signal_positioned Power_RR;
00406     } getSig;
00407     struct __attribute__((packed)) DBC_INVERTERS_MSG_TBD_Inv_R_Power_getSigVal {
00408         uint16_t TotalPower : 16;
00409         uint16_t Power_RL : 16;
00410         uint16_t Power_RR : 16;
00411     } *getSigVal;
00412 } DBC_INVERTERS_MSG_TBD_Inv_R_Power __attribute__((unused)) = {
00413     .ID = 0xB4,
00414     .IDE = CAN_ID_STD,
00415     .DLC = 6,
00416     .getSig = {
00417         .TotalPower = {
00418             .position = 7,
00419             .attributes = &DBC_INVERTERS_SIG_TotalPower
00420         },
00421         .Power_RL = {
00422             .position = 23,
00423             .attributes = &DBC_INVERTERS_SIG_Power_RL
00424         },
00425         .Power_RR = {
00426             .position = 39,
00427             .attributes = &DBC_INVERTERS_SIG_Power_RR
00428         }
00429     },
00430     .getSigVal = 0
00431 };
00432
00433
00438 static struct DBC_INVERTERS_MSG_TBD_Inv_R_Torque {
00439     const uint32_t ID;
00440     const uint8_t IDE;
00441     const uint8_t DLC;
00442     const struct {
00443         signal_positioned TorqueReal_RL;
00444         signal_positioned TorqueReal_RR;
00445     } getSig;
00446     struct __attribute__((packed)) DBC_INVERTERS_MSG_TBD_Inv_R_Torque_getSigVal {
00447         uint16_t TorqueReal_RL : 16;
00448         uint16_t TorqueReal_RR : 16;
00449     } *getSigVal;
00450 } DBC_INVERTERS_MSG_TBD_Inv_R_Torque __attribute__((unused)) = {
00451     .ID = 0xC7,
00452     .IDE = CAN_ID_STD,
00453     .DLC = 4,
00454     .getSig = {
00455         .TorqueReal_RL = {
00456             .position = 7,
00457             .attributes = &DBC_INVERTERS_SIG_TorqueReal_RL
00458         },
00459         .TorqueReal_RR = {
00460             .position = 23,
00461             .attributes = &DBC_INVERTERS_SIG_TorqueReal_RR
00462         }
00463     },
00464     .getSigVal = 0
00465 };
00466
00467
00472 static struct DBC_INVERTERS_MSG_AP_Inv_R_Error {
00473     const uint32_t ID;
00474     const uint8_t IDE;
00475     const uint8_t DLC;
00476 } DBC_INVERTERS_MSG_AP_Inv_R_Error __attribute__((unused)) = {
00477     .ID = 0xB0,
00478     .IDE = CAN_ID_STD,

```

```

00479     .DLC = 8
00480 };
00481
00482
00483 static struct DBC_INVERTERS_MSG_AP_Inv_R_Speed {
00484     const uint32_t ID;
00485     const uint8_t IDE;
00486     const uint8_t DLC;
00487     const struct {
00488         signal_positioned SpeedMotor_RL;
00489         signal_positioned SpeedMotor_RR;
00490     } getSig;
00491     struct __attribute__((packed)) DBC_INVERTERS_MSG_AP_Inv_R_Speed_getSigVal {
00492         uint16_t SpeedMotor_RL : 16;
00493         uint16_t SpeedMotor_RR : 16;
00494     } *getSigVal;
00495 } DBC_INVERTERS_MSG_AP_Inv_R_Speed __attribute__((unused)) = {
00496     .ID = 0xB5,
00497     .IDE = CAN_ID_STD,
00498     .DLC = 4,
00499     .getSig = {
00500         .SpeedMotor_RL = {
00501             .position = 7,
00502             .attributes = &DBC_INVERTERS_SIG_SpeedMotor_RL
00503         },
00504         .SpeedMotor_RR = {
00505             .position = 23,
00506             .attributes = &DBC_INVERTERS_SIG_SpeedMotor_RR
00507         }
00508     },
00509     .getSigVal = 0
00510 };
00511
00512
00513
00514 };
00515
00516
00517 static struct DBC_INVERTERS_MSG_AP_Inv_R_DCBus {
00518     const uint32_t ID;
00519     const uint8_t IDE;
00520     const uint8_t DLC;
00521     const struct {
00522         signal_positioned DCBus_RL;
00523         signal_positioned DCBus_RR;
00524     } getSig;
00525     struct __attribute__((packed)) DBC_INVERTERS_MSG_AP_Inv_R_DCBus_getSigVal {
00526         uint16_t DCBus_RL : 16;
00527         uint16_t DCBus_RR : 16;
00528     } *getSigVal;
00529 } DBC_INVERTERS_MSG_AP_Inv_R_DCBus __attribute__((unused)) = {
00530     .ID = 0xB3,
00531     .IDE = CAN_ID_STD,
00532     .DLC = 4,
00533     .getSig = {
00534         .DCBus_RL = {
00535             .position = 7,
00536             .attributes = &DBC_INVERTERS_SIG_DCBus_RL
00537         },
00538         .DCBus_RR = {
00539             .position = 23,
00540             .attributes = &DBC_INVERTERS_SIG_DCBus_RR
00541         }
00542     },
00543     .getSigVal = 0
00544 };
00545
00546
00547
00548 };
00549
00550
00551 static struct DBC_INVERTERS_MSG_MEAS_Inv_R_Currents_R {
00552     const uint32_t ID;
00553     const uint8_t IDE;
00554     const uint8_t DLC;
00555     const struct {
00556         signal_positioned Iactual_RR;
00557         signal_positioned Icmd_RR;
00558         signal_positioned Iq_RR;
00559     } getSig;
00560     struct __attribute__((packed)) DBC_INVERTERS_MSG_MEAS_Inv_R_Currents_R_getSigVal {
00561         uint16_t Iactual_RR : 16;
00562         uint16_t Icmd_RR : 16;
00563         uint16_t Iq_RR : 16;
00564     } *getSigVal;
00565 } DBC_INVERTERS_MSG_MEAS_Inv_R_Currents_R __attribute__((unused)) = {
00566     .ID = 0xC9,
00567     .IDE = CAN_ID_STD,
00568     .DLC = 6,
00569     .getSig = {
00570         .Iactual_RR = {
00571             .position = 7,
00572             .attributes = &DBC_INVERTERS_SIG_Iactual_RR
00573         },
00574
00575
00576
00577

```

```

00578     .Icmd_RR = {
00579         .position = 23,
00580         .attributes = &DBC_INVERTERS_SIG_Icmd_RR
00581     },
00582     .Iq_RR = {
00583         .position = 39,
00584         .attributes = &DBC_INVERTERS_SIG_Iq_RR
00585     }
00586 },
00587     .getSigVal = 0
00588 };
00589
00590
00595 static struct DBC_INVERTERS_MSG_MEAS_Inv_R_Currents_L {
00596     const uint32_t ID;
00597     const uint8_t IDE;
00598     const uint8_t DLC;
00599     const struct {
00600         signal_positioned Iactual_RL;
00601         signal_positioned Icmd_RL;
00602         signal_positioned Iq_RL;
00603     } getSig;
00604     struct __attribute__((packed)) DBC_INVERTERS_MSG_MEAS_Inv_R_Currents_L_getSigVal {
00605         uint16_t Iactual_RL : 16;
00606         uint16_t Icmd_RL : 16;
00607         uint16_t Iq_RL : 16;
00608     } *getSigVal;
00609 } DBC_INVERTERS_MSG_MEAS_Inv_R_Currents_L __attribute__((unused)) = {
00610     .ID = 0xC8,
00611     .IDE = CAN_ID_STD,
00612     .DLC = 6,
00613     .getSig = {
00614         .Iactual_RL = {
00615             .position = 7,
00616             .attributes = &DBC_INVERTERS_SIG_Iactual_RL
00617         },
00618         .Icmd_RL = {
00619             .position = 23,
00620             .attributes = &DBC_INVERTERS_SIG_Icmd_RL
00621         },
00622         .Iq_RL = {
00623             .position = 39,
00624             .attributes = &DBC_INVERTERS_SIG_Iq_RL
00625         }
00626     },
00627     .getSigVal = 0
00628 };
00629
00630
00635 static struct DBC_INVERTERS_MSG_AP_Inv_R_Temperatures {
00636     const uint32_t ID;
00637     const uint8_t IDE;
00638     const uint8_t DLC;
00639     const struct {
00640         signal_positioned TempIGBT_RL;
00641         signal_positioned TempIGBT_RR;
00642         signal_positioned TempMotor_RL;
00643         signal_positioned TempMotor_RR;
00644     } getSig;
00645     struct __attribute__((packed)) DBC_INVERTERS_MSG_AP_Inv_R_Temperatures_getSigVal {
00646         uint16_t TempIGBT_RL : 16;
00647         uint16_t TempIGBT_RR : 16;
00648         uint16_t TempMotor_RL : 16;
00649         uint16_t TempMotor_RR : 16;
00650     } *getSigVal;
00651 } DBC_INVERTERS_MSG_AP_Inv_R_Temperatures __attribute__((unused)) = {
00652     .ID = 0xB2,
00653     .IDE = CAN_ID_STD,
00654     .DLC = 8,
00655     .getSig = {
00656         .TempIGBT_RL = {
00657             .position = 7,
00658             .attributes = &DBC_INVERTERS_SIG_TempIGBT_RL
00659         },
00660         .TempIGBT_RR = {
00661             .position = 23,
00662             .attributes = &DBC_INVERTERS_SIG_TempIGBT_RR
00663         },
00664         .TempMotor_RL = {
00665             .position = 39,
00666             .attributes = &DBC_INVERTERS_SIG_TempMotor_RL
00667         },
00668         .TempMotor_RR = {
00669             .position = 55,
00670             .attributes = &DBC_INVERTERS_SIG_TempMotor_RR
00671     },
00672 },
00673 }
```

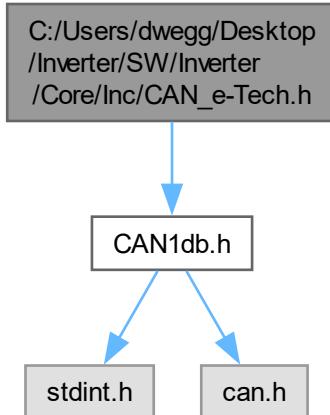
```
00673     .getSigVal = 0
00674 };
00675
00676
00677
00678
00679 /*
00680  * Nodes List:
00681  *   Inverters_Front
00682  *   Inverters_Rear
00683 */
00684
00685 /*
00686  * Functions
00687 */
00688
00689
00690 #pragma GCC diagnostic pop
00691
00692 #ifdef __cplusplus
00693 }
00694 #endif
00695
00696 #endif /* DBC_INVERTERS_H */
00697 /* @formatter:on */
```

6.3 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN_e-Tech.h File Reference

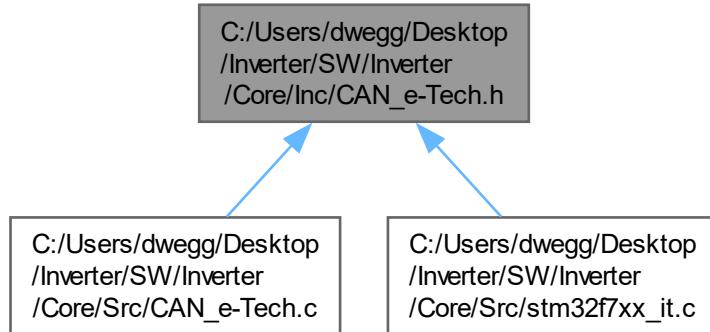
Header file for handling CAN communication with the car.

```
#include "CAN1db.h"
```

Include dependency graph for CAN_e-Tech.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [CANMessageInfo](#)

Functions

- void [handle_CAN](#) (CAN_HandleTypeDef *hcan)
Handle CAN messages.
- void [send_CAN_message](#) (CAN_HandleTypeDef *hcan, void *dbc_msg, const float *data)
Send a CAN message using [CAN1db.h](#) information.

Variables

- uint8_t [enableCAN](#)

6.3.1 Detailed Description

Header file for handling CAN communication with the car.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.3.2 Function Documentation

6.3.2.1 handle_CAN()

```
void handle_CAN (
    CAN_HandleTypeDef * hcan )
```

Handle CAN messages.

This function implements the logic to handle received CAN messages.

Parameters

<i>hcan</i>	Pointer to the CAN handle structure.
-------------	--------------------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



6.3.2.2 send_CAN_message()

```

void send_CAN_message (
    CAN_HandleTypeDef * hcan,
    void * dbc_msg,
    const float * data )

```

Send a CAN message using [CAN1db.h](#) information.

This function prepares and sends a CAN message using information from [CAN1db.h](#).

Parameters

<i>hcan</i>	Pointer to the CAN handle structure.
<i>dbc_msg</i>	Pointer to the structure containing CAN message information from CAN1db.h .
<i>data</i>	Pointer to the array of float data to be sent.

Here is the caller graph for this function:



6.3.3 Variable Documentation

6.3.3.1 enableCAN

```
uint8_t enableCAN [extern]
```

6.4 CAN_e-Tech.h

[Go to the documentation of this file.](#)

```

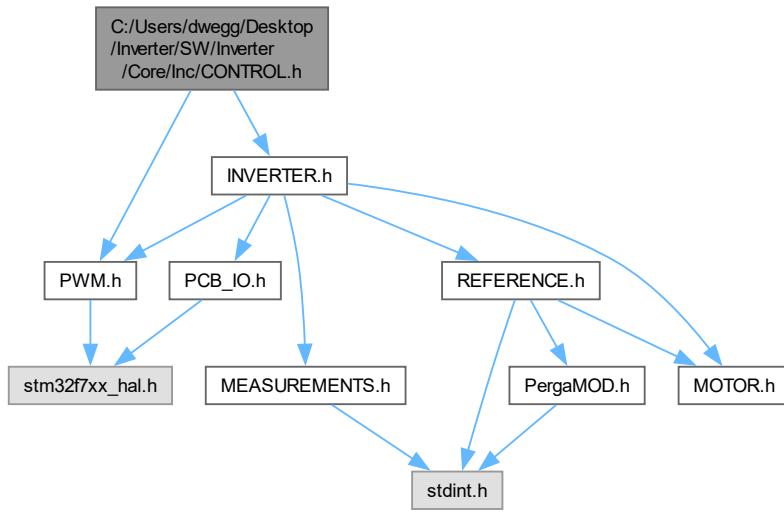
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 #ifndef CAN_E_TECH_H
00021 #define CAN_E_TECH_H
00022
00023 #include "CAN1db.h" // needs the CAN1db and its types
00024
00025 extern uint8_t enableCAN;
00026
00027 typedef struct {
00028     const uint32_t ID;
00029     const uint8_t IDE;
00030     const uint8_t DLC;
00031     const signal_positioned *getSig;
00032 } CANMessageInfo;
00033
00041 void handle_CAN(CAN_HandleTypeDef *hcan);
00042
00052 void send_CAN_message(CAN_HandleTypeDef *hcan, void *dbc_msg, const float *data);
00053
00054 #endif /* CAN_E_TECH_H */
  
```

6.5 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CONTROL.h File Reference

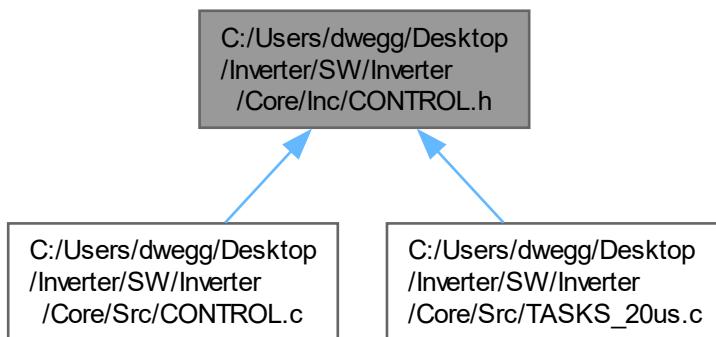
Header file for control logic.

```
#include "PWM.h"
#include "INVERTER.h"
```

Include dependency graph for CONTROL.h:



This graph shows which files directly or indirectly include this file:



Functions

- void `calc_current_loop` (volatile InverterStruct *inv)
Calculates the id-iq currents control actions.
- void `calc_duties` (float vd, float vq, float vDC, float sinTheta_e, float cosTheta_e, volatile Duties *duties)
function.

6.5.1 Detailed Description

Header file for control logic.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.5.2 Function Documentation

6.5.2.1 calc_current_loop()

```
void calc_current_loop (
    volatile InverterStruct * inv )
```

Calculates the id-iq currents control actions.

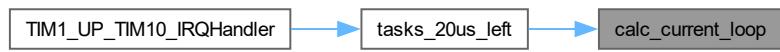
Parameters

<i>inv</i>	Pointer to the inverter structure.
------------	------------------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.2 calc_duties()

```
void calc_duties (
    float vd,
    float vq,
    float vDC,
    float sinTheta_e,
    float cosTheta_e,
    volatile Duties * duties )
```

function.

This function calculates the inverse Park transform and the duty cycles using SVPWM

Parameters

in	<i>vd</i>	Voltage in the d-axis.
in	<i>vq</i>	Voltage in the q-axis.
in	<i>vDC</i>	DC voltage.
in	<i>sinTheta_e</i>	Electrical angle sine (-1..1)
in	<i>cosTheta_e</i>	Electrical angle cosine (-1..1)
out	<i>duties</i>	Pointer to the duties structure.

Here is the call graph for this function:



Here is the caller graph for this function:



6.6 CONTROL.h

[Go to the documentation of this file.](#)

00001 /* USER CODE BEGIN Header */

```

00018 /* USER CODE END Header */
00019
00020 #ifndef CONTROL_H
00021 #define CONTROL_H
00022
00023 #include "PWM.h" // duties struct
00024 #include "INVERTER.h" // TS & Inverter struct
00025
00026 void calc_current_loop(volatile InverterStruct *inv);
00027
00028 void calc_duties(float vd, float vq, float vDC, float sinTheta_e, float cosTheta_e, volatile Duties
    *duties);
00029
00030 #endif /* CONTROL_H */

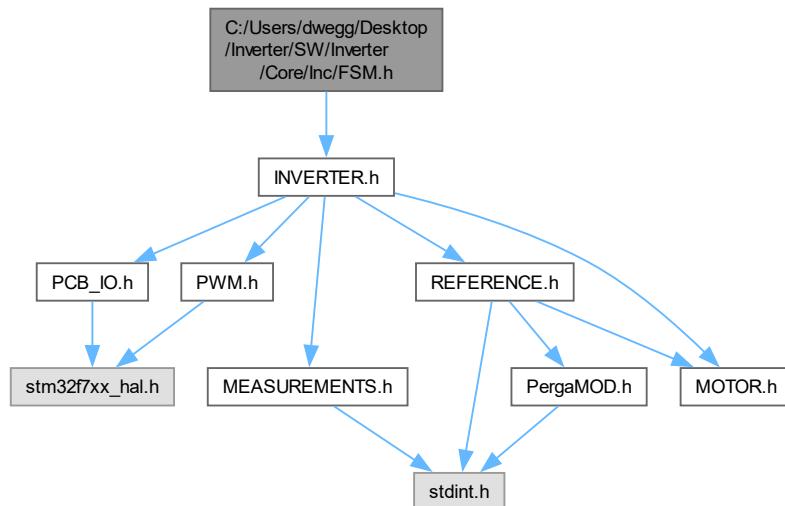
```

6.7 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/FSM.h File Reference

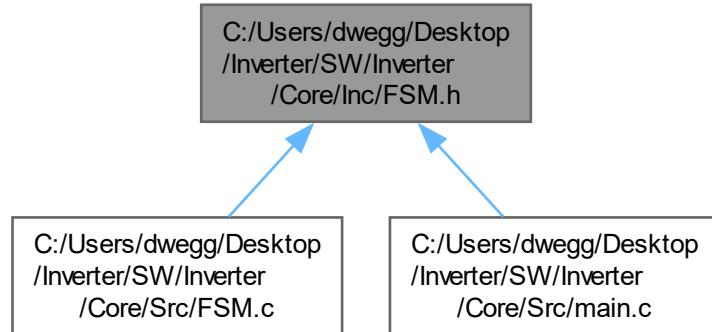
Header for the inverter Finite State Machine.

#include "INVERTER.h"

Include dependency graph for FSM.h:



This graph shows which files directly or indirectly include this file:



Functions

- void `eval_inv_FSM` (`volatile InverterStruct *inv`)
Run the Finite State Machine (FSM) for inverter operation control.

6.7.1 Detailed Description

Header for the inverter Finite State Machine.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.7.2 Function Documentation

6.7.2.1 `eval_inv_FSM()`

```
void eval_inv_FSM (
    volatile InverterStruct * inv )
```

Run the Finite State Machine (FSM) for inverter operation control.

Parameters

<i>inv</i>	Pointer to the inverter structure.
------------	------------------------------------

Run the Finite State Machine (FSM) for inverter operation control.

This function executes the finite state machine to control the inverter based on its current state.

Parameters

<i>inv</i>	Pointer to the inverter structure.
------------	------------------------------------

Here is the caller graph for this function:



6.8 FSM.h

[Go to the documentation of this file.](#)

```

00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 #ifndef FSM_H
00021 #define FSM_H
00022 #include "INVERTER.h" // inverter struct
00023
00024
00030 void eval_inv_FSM(volatile InverterStruct *inv);
00031
00032 #endif /* FSM_H */

```

6.9 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/INVERTER.h File Reference

Header file for the inverter struct and extern variables.

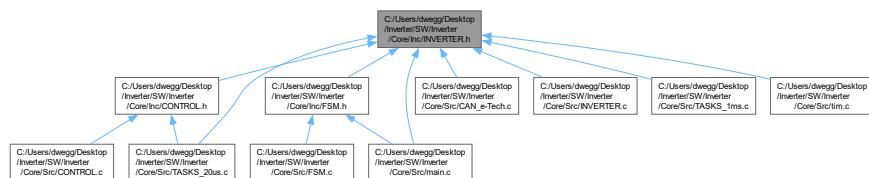
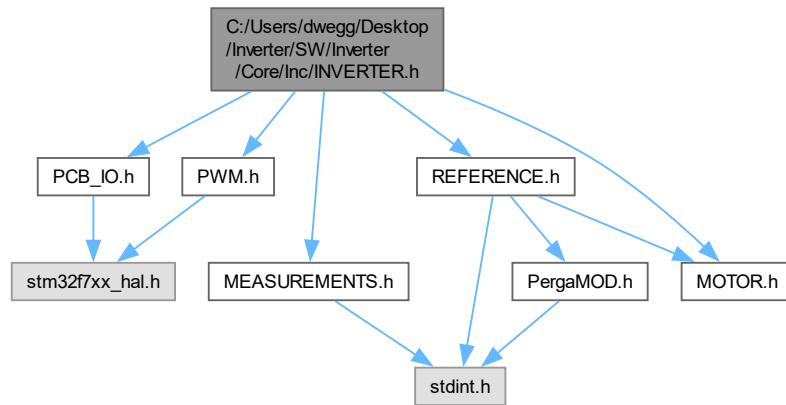
```

#include "PCB_IO.h"
#include "MEASUREMENTS.h"
#include "REFERENCE.h"
#include "MOTOR.h"

```

```
#include "PWM.h"
```

Include dependency graph for INVERTER.h:



Data Structures

- struct [InverterStruct](#)
Inverter structure.

Macros

- #define [TS](#) 0.00002
- #define [DT](#) 0.00000015

Enumerations

- enum [InverterState](#) { [INV_STATE_IDLE](#), [INV_STATE_STARTUP](#), [INV_STATE_RUNNING](#), [INV_STATE_FAULT](#) }
- Enumeration of inverter operation states.*

Functions

- void `initialize_inverter` (volatile `InverterStruct` *inv, `LED` *led, `GPIO_TypeDef` *enable_port, `uint16_t` enable_pin, `TIM_HandleTypeDef` *htim, `ADC_HandleTypeDef` *hadc, `MotorParameters` *motor)
Initialize the inverter.
- void `init_control_loops` (volatile `InverterStruct` *inv, `MotorParameters` *motor)
Initializes the id-iq current control PI controllers.

Variables

- volatile `InverterStruct` `inverter_left`
Left inverter structure.
- volatile `InverterStruct` `inverter_right`
Right inverter structure.

6.9.1 Detailed Description

Header file for the inverter struct and extern variables.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.9.2 Macro Definition Documentation

6.9.2.1 DT

```
#define DT 0.00000015
```

Dead time in seconds (150 ns), time in which both top and bottom transistors are open

6.9.2.2 TS

```
#define TS 0.000002
```

Switching time in seconds (20 us), inverse of the switching frequency of 50 kHz

6.9.3 Enumeration Type Documentation

6.9.3.1 InverterState

```
enum InverterState
```

Enumeration of inverter operation states.

Enumerator

INV_STATE_IDLE	Inverter idle state
INV_STATE_STARTUP	Inverter startup state
INV_STATE_RUNNING	Inverter running state
INV_STATE_FAULT	Inverter fault state

6.9.4 Function Documentation

6.9.4.1 init_control_loops()

```
void init_control_loops (
    volatile InverterStruct * inv,
    MotorParameters * motor )
```

Initializes the id-iq current control PI controllers.

Parameters

<i>inv</i>	Pointer to the inverter structure.
------------	------------------------------------

Initializes the id-iq current control PI controllers.

Parameters

<i>inv</i>	Pointer to the inverter structure.
------------	------------------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.4.2 initialize_inverter()

```
void initialize_inverter (
    volatile InverterStruct * inv,
    LED * led,
    GPIO_TypeDef * enable_port,
    uint16_t enable_pin,
    TIM_HandleTypeDef * htim,
    ADC_HandleTypeDef * hadc,
    MotorParameters * motor )
```

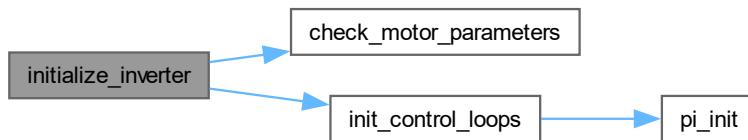
Initialize the inverter.

This function initializes the inverter structure with the specified [LED](#), GPIO port, and pin.

Parameters

out	<i>inv</i>	Pointer to the inverter structure.
in	<i>led</i>	Pointer to the LED structure.
in	<i>enable_port</i>	Pointer to the GPIO port for enabling/disabling the inverter.
in	<i>enable_pin</i>	Pin number for enabling/disabling the inverter.
in	<i>htim</i>	Timer peripheral for the PWM output.
in	<i>hadc</i>	ADC peripheral for the current phase current and DC voltage sensing.
in	<i>motor</i>	MotorParameters struct.

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.5 Variable Documentation

6.9.5.1 inverter_left

```
volatile InverterStruct inverter_left [extern]
```

Left inverter structure.

External declaration of the left inverter structure

External declaration of the left inverter structure.

6.9.5.2 inverter_right

```
volatile InverterStruct inverter_right [extern]
```

Right inverter structure.

External declaration of the right inverter structure

External declaration of the right inverter structure.

6.10 INVERTER.h

[Go to the documentation of this file.](#)

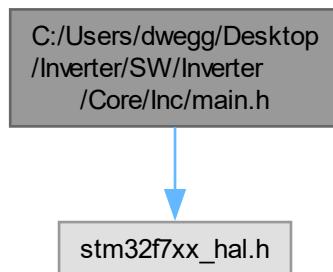
```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 #ifndef INVERTER_H
00021 #define INVERTER_H
00022
00023 #include "PCB_IO.h" // peripheral types
00024 #include "MEASUREMENTS.h" // a few structs
00025 #include "REFERENCE.h" // reference struct
00026 #include "MOTOR.h" // motor struct
00027 #include "PWM.h" // duties struct
00028
00029
00030
00031 #define TS 0.00002
00032 #define DT 0.00000015
00040 typedef enum {
00041     INV_STATE_IDLE,
00042     INV_STATE_STARTUP,
00043     INV_STATE_RUNNING,
00044     INV_STATE_FAULT
00045 } InverterState;
00046
00050 typedef struct {
00051     LED *led;
00052     GPIO_TypeDef *enable_port;
00053     uint16_t enable_pin;
00054     TIM_HandleTypeDef *htim;
00055     ADC_HandleTypeDef *hadc;
00056     InverterState state;
00057     Analog analog;
00058     Encoder encoder;
00059     Feedback feedback;
00060     Reference reference;
00061     Duties duties;
00062     int8_t direction;
00063     float tempInverter;
00064     float tempMotor;
00065     MotorParameters *motor;
00066     pi_struct idLoop;
00067     pi_struct iqLoop;
00068     pi_struct speedLoop;
00070 } InverterStruct;
00071
00072 extern volatile InverterStruct inverter_left;
00073 extern volatile InverterStruct inverter_right;
00088 void initialize_inverter(volatile InverterStruct *inv, LED *led, GPIO_TypeDef *enable_port, uint16_t
    enable_pin, TIM_HandleTypeDef *htim, ADC_HandleTypeDef *hadc, MotorParameters *motor);
00089
00090
00096 void init_control_loops(volatile InverterStruct *inv, MotorParameters *motor);
00097
00098 #endif /* INVERTER_H */
```

6.11 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/main.h File Reference

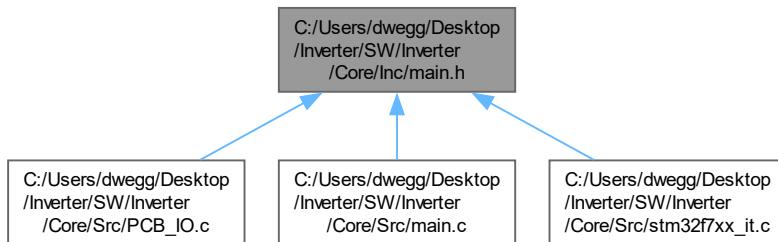
: Header for [main.c](#) file. This file contains the common defines of the application.

```
#include "stm32f7xx_hal.h"
```

Include dependency graph for main.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define Tinv_L_Pin GPIO_PIN_0`
- `#define Tinv_L_GPIO_Port GPIOC`
- `#define Tinv_R_Pin GPIO_PIN_1`
- `#define Tinv_R_GPIO_Port GPIOC`
- `#define Tmot_L_Pin GPIO_PIN_2`
- `#define Tmot_L_GPIO_Port GPIOC`
- `#define Tmot_R_Pin GPIO_PIN_3`
- `#define Tmot_R_GPIO_Port GPIOC`
- `#define ia_L_Pin GPIO_PIN_0`
- `#define ia_L_GPIO_Port GPIOA`
- `#define ib_L_Pin GPIO_PIN_1`

- #define ib_L_GPIO_Port GPIOA
- #define ic_L_Pin GPIO_PIN_2
- #define ic_L_GPIO_Port GPIOA
- #define VDC_L_Pin GPIO_PIN_3
- #define VDC_L_GPIO_Port GPIOA
- #define DAC_Pin GPIO_PIN_4
- #define DAC_GPIO_Port GPIOA
- #define PWM1_R_Pin GPIO_PIN_5
- #define PWM1_R_GPIO_Port GPIOA
- #define ia_R_Pin GPIO_PIN_6
- #define ia_R_GPIO_Port GPIOA
- #define ib_R_Pin GPIO_PIN_7
- #define ib_R_GPIO_Port GPIOA
- #define SC_det_Pin GPIO_PIN_4
- #define SC_det_GPIO_Port GPIOC
- #define ic_R_Pin GPIO_PIN_0
- #define ic_R_GPIO_Port GPIOB
- #define VDC_R_Pin GPIO_PIN_1
- #define VDC_R_GPIO_Port GPIOB
- #define ENABLE_R_Pin GPIO_PIN_2
- #define ENABLE_R_GPIO_Port GPIOB
- #define ENABLE_L_Pin GPIO_PIN_7
- #define ENABLE_L_GPIO_Port GPIOE
- #define PWM1_L_Pin GPIO_PIN_8
- #define PWM1_L_GPIO_Port GPIOE
- #define PWM2_L_Pin GPIO_PIN_9
- #define PWM2_L_GPIO_Port GPIOE
- #define PWM3_L_Pin GPIO_PIN_10
- #define PWM3_L_GPIO_Port GPIOE
- #define PWM4_L_Pin GPIO_PIN_11
- #define PWM4_L_GPIO_Port GPIOE
- #define PWM5_L_Pin GPIO_PIN_12
- #define PWM5_L_GPIO_Port GPIOE
- #define PWM6_L_Pin GPIO_PIN_13
- #define PWM6_L_GPIO_Port GPIOE
- #define WRN_L_Pin GPIO_PIN_14
- #define WRN_L_GPIO_Port GPIOE
- #define WRN_R_Pin GPIO_PIN_15
- #define WRN_R_GPIO_Port GPIOE
- #define B_R_Pin GPIO_PIN_10
- #define B_R_GPIO_Port GPIOB
- #define Z_R_Pin GPIO_PIN_11
- #define Z_R_GPIO_Port GPIOB
- #define PWM3_R_Pin GPIO_PIN_14
- #define PWM3_R_GPIO_Port GPIOB
- #define PWM5_R_Pin GPIO_PIN_15
- #define PWM5_R_GPIO_Port GPIOB
- #define A_L_Pin GPIO_PIN_12
- #define A_L_GPIO_Port GPIOD
- #define B_L_Pin GPIO_PIN_14
- #define B_L_GPIO_Port GPIOD
- #define Z_L_Pin GPIO_PIN_15
- #define Z_L_GPIO_Port GPIOD
- #define PWM2_R_Pin GPIO_PIN_6
- #define PWM2_R_GPIO_Port GPIOC

- #define PWM4_R_Pin GPIO_PIN_7
- #define PWM4_R_GPIO_Port GPIOC
- #define PWM6_R_Pin GPIO_PIN_8
- #define PWM6_R_GPIO_Port GPIOC
- #define TRIP_R_Pin GPIO_PIN_9
- #define TRIP_R_GPIO_Port GPIOC
- #define TRIP_L_Pin GPIO_PIN_8
- #define TRIP_L_GPIO_Port GPIOA
- #define A_R_Pin GPIO_PIN_15
- #define A_R_GPIO_Port GPIOA
- #define DIR_Pin GPIO_PIN_3
- #define DIR_GPIO_Port GPIOD
- #define LED_LEFT_Pin GPIO_PIN_4
- #define LED_LEFT_GPIO_Port GPIOD
- #define LED_RIGHT_Pin GPIO_PIN_5
- #define LED_RIGHT_GPIO_Port GPIOD
- #define LED_ERR_Pin GPIO_PIN_6
- #define LED_ERR_GPIO_Port GPIOD

Functions

- void Error_Handler (void)
This function is executed in case of error occurrence.

6.11.1 Detailed Description

: Header for [main.c](#) file. This file contains the common defines of the application.

Attention

Copyright (c) 2023 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

6.11.2 Macro Definition Documentation

6.11.2.1 A_L_GPIO_Port

```
#define A_L_GPIO_Port GPIOD
```

6.11.2.2 A_L_Pin

```
#define A_L_Pin GPIO_PIN_12
```

6.11.2.3 A_R_GPIO_Port

```
#define A_R_GPIO_Port GPIOA
```

6.11.2.4 A_R_Pin

```
#define A_R_Pin GPIO_PIN_15
```

6.11.2.5 B_L_GPIO_Port

```
#define B_L_GPIO_Port GPIOD
```

6.11.2.6 B_L_Pin

```
#define B_L_Pin GPIO_PIN_14
```

6.11.2.7 B_R_GPIO_Port

```
#define B_R_GPIO_Port GPIOB
```

6.11.2.8 B_R_Pin

```
#define B_R_Pin GPIO_PIN_10
```

6.11.2.9 DAC_GPIO_Port

```
#define DAC_GPIO_Port GPIOA
```

6.11.2.10 DAC_Pin

```
#define DAC_Pin GPIO_PIN_4
```

6.11.2.11 DIR_GPIO_Port

```
#define DIR_GPIO_Port GPIOD
```

6.11.2.12 DIR_Pin

```
#define DIR_Pin GPIO_PIN_3
```

6.11.2.13 ENABLE_L_GPIO_Port

```
#define ENABLE_L_GPIO_Port GPIOE
```

6.11.2.14 ENABLE_L_Pin

```
#define ENABLE_L_Pin GPIO_PIN_7
```

6.11.2.15 ENABLE_R_GPIO_Port

```
#define ENABLE_R_GPIO_Port GPIOB
```

6.11.2.16 ENABLE_R_Pin

```
#define ENABLE_R_Pin GPIO_PIN_2
```

6.11.2.17 ia_L_GPIO_Port

```
#define ia_L_GPIO_Port GPIOA
```

6.11.2.18 ia_L_Pin

```
#define ia_L_Pin GPIO_PIN_0
```

6.11.2.19 ia_R_GPIO_Port

```
#define ia_R_GPIO_Port GPIOA
```

6.11.2.20 ia_R_Pin

```
#define ia_R_Pin GPIO_PIN_6
```

6.11.2.21 ib_L_GPIO_Port

```
#define ib_L_GPIO_Port GPIOA
```

6.11.2.22 ib_L_Pin

```
#define ib_L_Pin GPIO_PIN_1
```

6.11.2.23 ib_R_GPIO_Port

```
#define ib_R_GPIO_Port GPIOA
```

6.11.2.24 ib_R_Pin

```
#define ib_R_Pin GPIO_PIN_7
```

6.11.2.25 ic_L_GPIO_Port

```
#define ic_L_GPIO_Port GPIOA
```

6.11.2.26 ic_L_Pin

```
#define ic_L_Pin GPIO_PIN_2
```

6.11.2.27 ic_R_GPIO_Port

```
#define ic_R_GPIO_Port GPIOB
```

6.11.2.28 ic_R_Pin

```
#define ic_R_Pin GPIO_PIN_0
```

6.11.2.29 LED_ERR_GPIO_Port

```
#define LED_ERR_GPIO_Port GPIOD
```

6.11.2.30 LED_ERR_Pin

```
#define LED_ERR_Pin GPIO_PIN_6
```

6.11.2.31 LED_LEFT_GPIO_Port

```
#define LED_LEFT_GPIO_Port GPIOD
```

6.11.2.32 LED_LEFT_Pin

```
#define LED_LEFT_Pin GPIO_PIN_4
```

6.11.2.33 LED_RIGHT_GPIO_Port

```
#define LED_RIGHT_GPIO_Port GPIOD
```

6.11.2.34 LED_RIGHT_Pin

```
#define LED_RIGHT_Pin GPIO_PIN_5
```

6.11.2.35 PWM1_L_GPIO_Port

```
#define PWM1_L_GPIO_Port GPIOE
```

6.11.2.36 PWM1_L_Pin

```
#define PWM1_L_Pin GPIO_PIN_8
```

6.11.2.37 PWM1_R_GPIO_Port

```
#define PWM1_R_GPIO_Port GPIOA
```

6.11.2.38 PWM1_R_Pin

```
#define PWM1_R_Pin GPIO_PIN_5
```

6.11.2.39 PWM2_L_GPIO_Port

```
#define PWM2_L_GPIO_Port GPIOE
```

6.11.2.40 PWM2_L_Pin

```
#define PWM2_L_Pin GPIO_PIN_9
```

6.11.2.41 PWM2_R_GPIO_Port

```
#define PWM2_R_GPIO_Port GPIOC
```

6.11.2.42 PWM2_R_Pin

```
#define PWM2_R_Pin GPIO_PIN_6
```

6.11.2.43 PWM3_L_GPIO_Port

```
#define PWM3_L_GPIO_Port GPIOE
```

6.11.2.44 PWM3_L_Pin

```
#define PWM3_L_Pin GPIO_PIN_10
```

6.11.2.45 PWM3_R_GPIO_Port

```
#define PWM3_R_GPIO_Port GPIOB
```

6.11.2.46 PWM3_R_Pin

```
#define PWM3_R_Pin GPIO_PIN_14
```

6.11.2.47 PWM4_L_GPIO_Port

```
#define PWM4_L_GPIO_Port GPIOE
```

6.11.2.48 PWM4_L_Pin

```
#define PWM4_L_Pin GPIO_PIN_11
```

6.11.2.49 PWM4_R_GPIO_Port

```
#define PWM4_R_GPIO_Port GPIOC
```

6.11.2.50 PWM4_R_Pin

```
#define PWM4_R_Pin GPIO_PIN_7
```

6.11.2.51 PWM5_L_GPIO_Port

```
#define PWM5_L_GPIO_Port GPIOE
```

6.11.2.52 PWM5_L_Pin

```
#define PWM5_L_Pin GPIO_PIN_12
```

6.11.2.53 PWM5_R_GPIO_Port

```
#define PWM5_R_GPIO_Port GPIOB
```

6.11.2.54 PWM5_R_Pin

```
#define PWM5_R_Pin GPIO_PIN_15
```

6.11.2.55 PWM6_L_GPIO_Port

```
#define PWM6_L_GPIO_Port GPIOE
```

6.11.2.56 PWM6_L_Pin

```
#define PWM6_L_Pin GPIO_PIN_13
```

6.11.2.57 PWM6_R_GPIO_Port

```
#define PWM6_R_GPIO_Port GPIOC
```

6.11.2.58 PWM6_R_Pin

```
#define PWM6_R_Pin GPIO_PIN_8
```

6.11.2.59 SC_det_GPIO_Port

```
#define SC_det_GPIO_Port GPIOC
```

6.11.2.60 SC_det_Pin

```
#define SC_det_Pin GPIO_PIN_4
```

6.11.2.61 Tinv_L_GPIO_Port

```
#define Tinv_L_GPIO_Port GPIOC
```

6.11.2.62 Tinv_L_Pin

```
#define Tinv_L_Pin GPIO_PIN_0
```

6.11.2.63 Tinv_R_GPIO_Port

```
#define Tinv_R_GPIO_Port GPIOC
```

6.11.2.64 Tinv_R_Pin

```
#define Tinv_R_Pin GPIO_PIN_1
```

6.11.2.65 Tmot_L_GPIO_Port

```
#define Tmot_L_GPIO_Port GPIOC
```

6.11.2.66 Tmot_L_Pin

```
#define Tmot_L_Pin GPIO_PIN_2
```

6.11.2.67 Tmot_R_GPIO_Port

```
#define Tmot_R_GPIO_Port GPIOC
```

6.11.2.68 Tmot_R_Pin

```
#define Tmot_R_Pin GPIO_PIN_3
```

6.11.2.69 TRIP_L_GPIO_Port

```
#define TRIP_L_GPIO_Port GPIOA
```

6.11.2.70 TRIP_L_Pin

```
#define TRIP_L_Pin GPIO_PIN_8
```

6.11.2.71 TRIP_R_GPIO_Port

```
#define TRIP_R_GPIO_Port GPIOC
```

6.11.2.72 TRIP_R_Pin

```
#define TRIP_R_Pin GPIO_PIN_9
```

6.11.2.73 VDC_L_GPIO_Port

```
#define VDC_L_GPIO_Port GPIOA
```

6.11.2.74 VDC_L_Pin

```
#define VDC_L_Pin GPIO_PIN_3
```

6.11.2.75 VDC_R_GPIO_Port

```
#define VDC_R_GPIO_Port GPIOB
```

6.11.2.76 VDC_R_Pin

```
#define VDC_R_Pin GPIO_PIN_1
```

6.11.2.77 WRN_L_GPIO_Port

```
#define WRN_L_GPIO_Port GPIOE
```

6.11.2.78 WRN_L_Pin

```
#define WRN_L_Pin GPIO_PIN_14
```

6.11.2.79 WRN_R_GPIO_Port

```
#define WRN_R_GPIO_Port GPIOE
```

6.11.2.80 WRN_R_Pin

```
#define WRN_R_Pin GPIO_PIN_15
```

6.11.2.81 Z_L_GPIO_Port

```
#define Z_L_GPIO_Port GPIOD
```

6.11.2.82 Z_L_Pin

```
#define Z_L_Pin GPIO_PIN_15
```

6.11.2.83 Z_R_GPIO_Port

```
#define Z_R_GPIO_Port GPIOB
```

6.11.2.84 Z_R_Pin

```
#define Z_R_Pin GPIO_PIN_11
```

6.11.3 Function Documentation

6.11.3.1 Error_Handler()

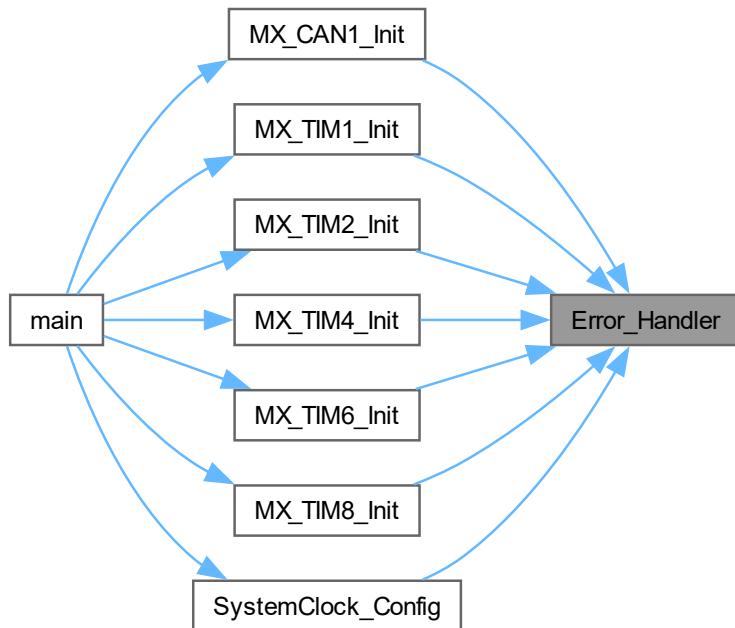
```
void Error_Handler ( void )
```

This function is executed in case of error occurrence.

Return values

None	
------	--

Here is the caller graph for this function:



6.12 main.h

[Go to the documentation of this file.](#)

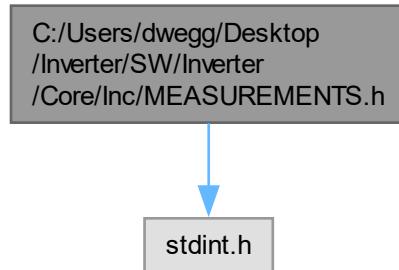
```
00001 /* USER CODE BEGIN Header */
00019 /* USER CODE END Header */
00020
00021 /* Define to prevent recursive inclusion -----*/
00022 #ifndef __MAIN_H
00023 #define __MAIN_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 /* Includes -----*/
00030 #include "stm32f7xx_hal.h"
00031
00032 /* Private includes -----*/
00033 /* USER CODE BEGIN Includes */
00034
00035 /* USER CODE END Includes */
00036
00037 /* Exported types -----*/
00038 /* USER CODE BEGIN ET */
00039
00040 /* USER CODE END ET */
00041
00042 /* Exported constants -----*/
00043 /* USER CODE BEGIN EC */
00044
00045 /* USER CODE END EC */
00046
00047 /* Exported macro -----*/
00048 /* USER CODE BEGIN EM */
00049
00050 /* USER CODE END EM */
00051
00052 /* Exported functions prototypes -----*/
00053 void Error_Handler(void);
00054
00055 /* USER CODE BEGIN EFP */
00056
00057 /* USER CODE END EFP */
00058
00059 /* Private defines -----*/
00060 #define Tinv_L_Pin GPIO_PIN_0
00061 #define Tinv_L_GPIO_Port GPIOC
00062 #define Tinv_R_Pin GPIO_PIN_1
00063 #define Tinv_R_GPIO_Port GPIOC
00064 #define Tmot_L_Pin GPIO_PIN_2
00065 #define Tmot_L_GPIO_Port GPIOC
00066 #define Tmot_R_Pin GPIO_PIN_3
00067 #define Tmot_R_GPIO_Port GPIOC
00068 #define ia_L_Pin GPIO_PIN_0
00069 #define ia_L_GPIO_Port GPIOA
00070 #define ib_L_Pin GPIO_PIN_1
00071 #define ib_L_GPIO_Port GPIOA
00072 #define ic_L_Pin GPIO_PIN_2
00073 #define ic_L_GPIO_Port GPIOA
00074 #define VDC_L_Pin GPIO_PIN_3
00075 #define VDC_L_GPIO_Port GPIOA
00076 #define DAC_Pin GPIO_PIN_4
00077 #define DAC_GPIO_Port GPIOA
00078 #define PWM1_R_Pin GPIO_PIN_5
00079 #define PWM1_R_GPIO_Port GPIOA
00080 #define ia_R_Pin GPIO_PIN_6
00081 #define ia_R_GPIO_Port GPIOA
00082 #define ib_R_Pin GPIO_PIN_7
00083 #define ib_R_GPIO_Port GPIOA
00084 #define SC_det_Pin GPIO_PIN_4
00085 #define SC_det_GPIO_Port GPIOC
00086 #define ic_R_Pin GPIO_PIN_0
00087 #define ic_R_GPIO_Port GPIOB
00088 #define VDC_R_Pin GPIO_PIN_1
00089 #define VDC_R_GPIO_Port GPIOB
00090 #define ENABLE_R_Pin GPIO_PIN_2
00091 #define ENABLE_R_GPIO_Port GPIOB
00092 #define ENABLE_L_Pin GPIO_PIN_7
00093 #define ENABLE_L_GPIO_Port GPIOE
00094 #define PWM1_L_Pin GPIO_PIN_8
00095 #define PWM1_L_GPIO_Port GPIOE
00096 #define PWM2_L_Pin GPIO_PIN_9
00097 #define PWM2_L_GPIO_Port GPIOE
00098 #define PWM3_L_Pin GPIO_PIN_10
00099 #define PWM3_L_GPIO_Port GPIOE
```

```
00100 #define PWM4_L_Pin GPIO_PIN_11
00101 #define PWM4_L_GPIO_Port GPIOE
00102 #define PWM5_L_Pin GPIO_PIN_12
00103 #define PWM5_L_GPIO_Port GPIOE
00104 #define PWM6_L_Pin GPIO_PIN_13
00105 #define PWM6_L_GPIO_Port GPIOE
00106 #define WRN_L_Pin GPIO_PIN_14
00107 #define WRN_L_GPIO_Port GPIOE
00108 #define WRN_R_Pin GPIO_PIN_15
00109 #define WRN_R_GPIO_Port GPIOE
00110 #define B_R_Pin GPIO_PIN_10
00111 #define B_R_GPIO_Port GPIOB
00112 #define Z_R_Pin GPIO_PIN_11
00113 #define Z_R_GPIO_Port GPIOB
00114 #define PWM3_R_Pin GPIO_PIN_14
00115 #define PWM3_R_GPIO_Port GPIOB
00116 #define PWM5_R_Pin GPIO_PIN_15
00117 #define PWM5_R_GPIO_Port GPIOB
00118 #define A_L_Pin GPIO_PIN_12
00119 #define A_L_GPIO_Port GPIOD
00120 #define B_L_Pin GPIO_PIN_14
00121 #define B_L_GPIO_Port GPIOD
00122 #define Z_L_Pin GPIO_PIN_15
00123 #define Z_L_GPIO_Port GPIOD
00124 #define PWM2_R_Pin GPIO_PIN_6
00125 #define PWM2_R_GPIO_Port GPIOC
00126 #define PWM4_R_Pin GPIO_PIN_7
00127 #define PWM4_R_GPIO_Port GPIOC
00128 #define PWM6_R_Pin GPIO_PIN_8
00129 #define PWM6_R_GPIO_Port GPIOC
00130 #define TRIP_R_Pin GPIO_PIN_9
00131 #define TRIP_R_GPIO_Port GPIOC
00132 #define TRIP_L_Pin GPIO_PIN_8
00133 #define TRIP_L_GPIO_Port GPIOA
00134 #define A_R_Pin GPIO_PIN_15
00135 #define A_R_GPIO_Port GPIOA
00136 #define DIR_Pin GPIO_PIN_3
00137 #define DIR_GPIO_Port GPIOD
00138 #define LED_LEFT_Pin GPIO_PIN_4
00139 #define LED_LEFT_GPIO_Port GPIOD
00140 #define LED_RIGHT_Pin GPIO_PIN_5
00141 #define LED_RIGHT_GPIO_Port GPIOD
00142 #define LED_ERR_Pin GPIO_PIN_6
00143 #define LED_ERR_GPIO_Port GPIOD
00144
00145 /* USER CODE BEGIN Private defines */
00146
00147 /* USER CODE END Private defines */
00148
00149 #ifdef __cplusplus
00150 }
00151 #endif
00152
00153#endif /* __MAIN_H */
```

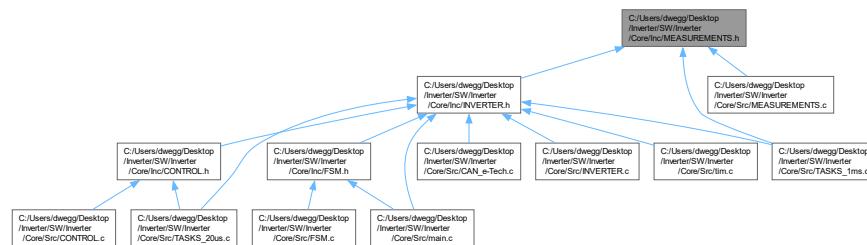
6.13 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/← MEASUREMENTS.h File Reference

Header file for handling measurements.

```
#include <stdint.h>
Include dependency graph for MEASUREMENTS.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [Encoder](#)
Structure for encoder reading.
- struct [Analog](#)
Structure for ADC measurements in units.
- struct [Feedback](#)
Structure for feedback values.

Macros

- #define [CURRENT_SLOPE](#) 54.4217687f
- #define [CURRENT_OFFSET](#) 1.70068027211f
- #define [VOLTAGE_SLOPE](#) 263.435f
- #define [VOLTAGE_OFFSET](#) 0.02083f

Functions

- `uint8_t get_currents_voltage (volatile uint32_t ADC_raw[], volatile Analog *analog, volatile Feedback *feedback, float sinTheta_e, float cosTheta_e)`
Get electrical ADC measurements.
- `float get_linear (uint32_t bits, float slope, float offset)`
Convert ADC reading to physical measurement with linear response.
- `void get_idiq (float ia, float ib, float ic, float sinTheta_e, float cosTheta_e, float *idMeas, float *iqMeas)`
Computes d-q currents from current measurements and electrical angle.
- `float get_temperature (uint32_t bits, const float tempLUT[])`
Retrieves temperature from a lookup table based on ADC bits.

Variables

- `const float templnverterLUT []`
- `const float tempMotorLUT []`
- `volatile uint32_t rawADC_left [4]`
Raw ADC data for the left inverter.
- `volatile uint32_t rawADC_right [4]`
Raw ADC data for the right inverter.
- `volatile uint32_t rawADC_temp [4]`
Raw ADC data for the temperatures.

6.13.1 Detailed Description

Header file for handling measurements.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.13.2 Macro Definition Documentation

6.13.2.1 CURRENT_OFFSET

```
#define CURRENT_OFFSET 1.70068027211f
```

[V] (10/(4.7+10))* 2.5 V

6.13.2.2 CURRENT_SLOPE

```
#define CURRENT_SLOPE 54.4217687f
```

[A/V] (10/(4.7+10)) * (1 / (12.5 mV / A))

6.13.2.3 VOLTAGE_OFFSET

```
#define VOLTAGE_OFFSET 0.02083f
```

[V] $(100/(4700+100) * 5 \text{ V})$

6.13.2.4 VOLTAGE_SLOPE

```
#define VOLTAGE_SLOPE 263.435f
```

[V/V] $1/(1/3 * 0.011388) \text{ V}$

6.13.3 Function Documentation

6.13.3.1 get_currents_voltage()

```
uint8_t get_currents_voltage (
    volatile uint32_t ADC_raw[],
    volatile Analog * analog,
    volatile Feedback * feedback,
    float sinTheta_e,
    float cosTheta_e )
```

Get electrical ADC measurements.

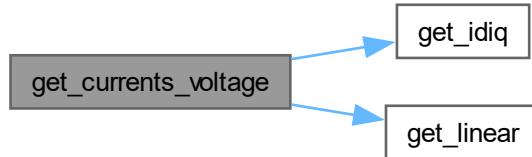
Parameters

in	<i>ADC_raw</i>	Pointer to the raw ADC values array.
out	<i>analog</i>	Pointer to the ADC struct to store the results.
out	<i>feedback</i>	Pointer to the Feedback struct to store id and iq.
in	<i>sinTheta_e</i>	Electrical angle sine (-1..1)
in	<i>cosTheta_e</i>	Electrical angle cosine (-1..1)

Return values

<i>OK</i>	0 if an error occurred, 1 if successful.
-----------	--

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.3.2 `get_idiq()`

```

void get_idiq (
    float ia,
    float ib,
    float ic,
    float sinTheta_e,
    float cosTheta_e,
    float * idMeas,
    float * iqMeas )
  
```

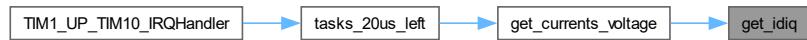
Computes d-q currents from current measurements and electrical angle.

This function computes the d-q currents from phase currents (ABC), theta_e, and stores the results in the provided pointers.

Parameters

in	<i>ia</i>	Phase A current in A.
in	<i>ib</i>	Phase B current in A.
in	<i>ic</i>	Phase C current in A.
in	<i>sinTheta_e</i>	Electrical angle sine (-1..1)
in	<i>cosTheta_e</i>	Electrical angle cosine (-1..1)
out	<i>idMeas</i>	Pointer to store the D-axis current.
out	<i>iqMeas</i>	Pointer to store the Q-axis current.

Here is the caller graph for this function:



6.13.3.3 `get_linear()`

```
float get_linear (
    uint32_t bits,
    float slope,
    float offset )
```

Convert ADC reading to physical measurement with linear response.

Parameters

in	<i>bits</i>	The ADC reading.
in	<i>slope</i>	The slope (volts per unit).
in	<i>offset</i>	The offset (volts at zero).

Return values

<i>measurement</i>	The physical measurement.
--------------------	---------------------------

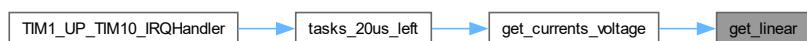
Parameters

in	<i>bits</i>	The ADC reading.
in	<i>slope</i>	The slope (units per volt).
in	<i>offset</i>	The offset (volts at zero).

Return values

<i>measurement</i>	The physical measurement.
--------------------	---------------------------

Here is the caller graph for this function:



6.13.3.4 get_temperature()

```
float get_temperature (
    uint32_t bits,
    const float tempLUT[ ] )
```

Retrieves temperature from a lookup table based on ADC bits.

This function retrieves temperature from a lookup table based on the ADC bits. The lookup table (LUT) must have a value for each possible ADC bit combination.

Parameters

in	<i>bits</i>	ADC reading converted to bits.
in	<i>tempLUT</i>	Lookup table containing temperature values.

Returns

Temperature corresponding to the provided ADC bits.

Here is the caller graph for this function:



6.13.4 Variable Documentation

6.13.4.1 rawADC_left

```
volatile uint32_t rawADC_left[4] [extern]
```

Raw ADC data for the left inverter.

External declaration of raw ADC data for the left inverter

External declaration of raw ADC data for the left inverter.

6.13.4.2 rawADC_right

```
volatile uint32_t rawADC_right[4] [extern]
```

Raw ADC data for the right inverter.

External declaration of raw ADC data for the right inverter

External declaration of raw ADC data for the right inverter.

6.13.4.3 rawADC_temp

```
volatile uint32_t rawADC_temp[4] [extern]
```

Raw ADC data for the temperatures.

External declaration of raw ADC data for the temperatures

External declaration of raw ADC data for the temperature readings.

6.13.4.4 tempInverterLUT

```
const float tempInverterLUT[] [extern]
```

6.13.4.5 tempMotorLUT

```
const float tempMotorLUT[] [extern]
```

6.14 MEASUREMENTS.h

[Go to the documentation of this file.](#)

```
00001 /* USER CODE BEGIN Header */
00017 /* USER CODE END Header */
00018
00019
00020 #ifndef MEASUREMENTS_H
00021 #define MEASUREMENTS_H
00022
00023 /* Define current and voltage gains/offsets */
00024 #define CURRENT_SLOPE 54.4217687f
00025 #define CURRENT_OFFSET 1.70068027211f
00026 #define VOLTAGE_SLOPE 263.435f
00027 #define VOLTAGE_OFFSET 0.02083f
00029 #include <stdint.h>
00030
00031 extern const float tempInverterLUT[];
00032 extern const float tempMotorLUT[];
00033
00034 extern volatile uint32_t rawADC_left[4];
00035 extern volatile uint32_t rawADC_right[4];
00036 extern volatile uint32_t rawADC_temp[4];
00042 typedef struct {
00043     uint16_t A;
00044     uint16_t B;
00045     uint16_t Z;
00046     float we;
00047     float theta_e;
00048     float sinTheta_e;
00049     float cosTheta_e;
00050     uint8_t directionMeas;
00051 } Encoder;
00052
00056 typedef struct {
00057     float ia;
00058     float ib;
00059     float ic;
00060     float vDC;
00061 } Analog;
00062
00066 typedef struct {
00067     float idMeas;
00068     float iqMeas;
00069     float torqueCalc;
00070     float speedMeas;
00071 } Feedback;
00072
```

```

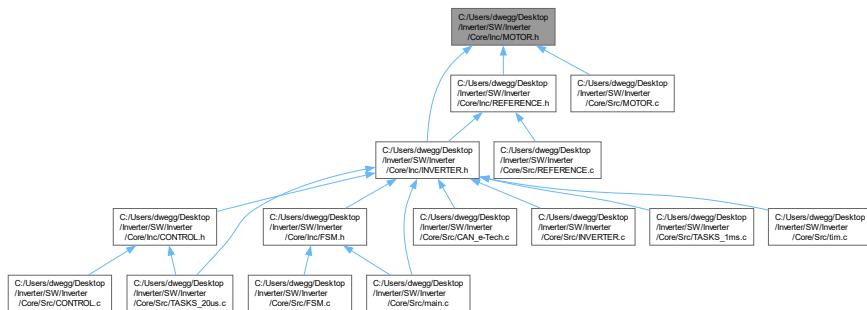
00082 uint8_t get_currents_voltage(volatile uint32_t ADC_raw[], volatile Analog* analog, volatile Feedback*
    feedback, float sinTheta_e, float cosTheta_e);
00083
00091 float get_linear(uint32_t bits, float slope, float offset);
00092
00107 void get_idiq(float ia, float ib, float ic, float sinTheta_e, float cosTheta_e, float *idMeas, float
    *iqMeas);
00108
00109
00120 float get_temperature(uint32_t bits, const float tempLUT[]);
00121 #endif /* MEASUREMENTS_H */

```

6.15 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MOTOR.h File Reference

Header file for motor parameters.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct **MotorParameters**

Structure to hold motor parameters.

Functions

- int **check_motor_parameters** (**MotorParameters** *motor, float Ts)

Perform a parameter check and correct possible errors.

Variables

- **MotorParameters motor_left**
Left motor parameters.
- **MotorParameters motor_right**
Right motor parameters.

6.15.1 Detailed Description

Header file for motor parameters.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.15.2 Function Documentation

6.15.2.1 check_motor_parameters()

```
int check_motor_parameters (
    MotorParameters * motor,
    float Ts )
```

Perform a parameter check and correct possible errors.

Parameters

in	<i>motor</i>	Pointer to the <code>MotorParameters</code> struct.
----	--------------	---

Return values

<i>OK</i>	0 if an error occurred, 1 if successful.
-----------	--

Here is the caller graph for this function:



6.15.3 Variable Documentation

6.15.3.1 motor_left

```
MotorParameters motor_left [extern]
```

Left motor parameters.

6.15.3.2 motor_right

```
MotorParameters motor_right [extern]
```

Right motor parameters.

6.16 MOTOR.h

[Go to the documentation of this file.](#)

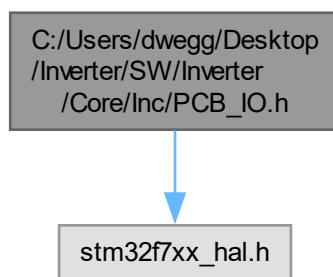
```
00001 /* USER CODE BEGIN Header */
00017 /* USER CODE END Header */
00018
00019 #ifndef MOTOR_H
00020 #define MOTOR_H
00021
00025 typedef struct {
00026     float Ld;
00027     float Lq;
00028     float Rs;
00029     float lambda;
00030     int pp;
00031     float J;
00032     float b;
00033     float torque_max;
00034     float dTorque_max;
00035     float speed_max_RPM;
00036     float iPhase_pk_max;
00037     float vDC_max;
00039 } MotorParameters;
00040
00041 extern MotorParameters motor_left;
00042 extern MotorParameters motor_right;
00043
00049 int check_motor_parameters(MotorParameters *motor, float Ts);
00050#endif /* MOTOR_H */
```

6.17 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PCB_IO.h File Reference

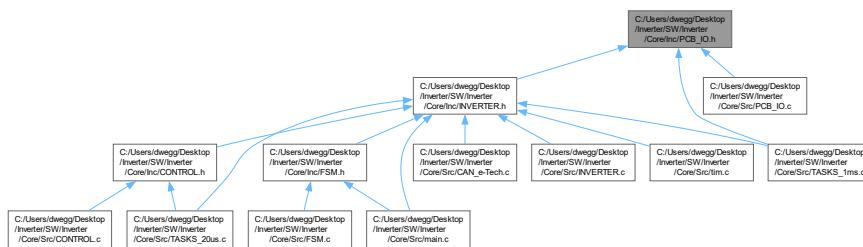
Header file for handling GPIOs.

```
#include "stm32f7xx_hal.h"
```

Include dependency graph for PCB_IO.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [LED](#)
LED structure.

Macros

- #define [SC_DET_STATE\(\)](#) (HAL_GPIO_ReadPin(SC_det_GPIO_Port, SC_det_Pin))
- #define [DIR_STATE\(\)](#) (HAL_GPIO_ReadPin(DIR_GPIO_Port, DIR_Pin))
- #define [WRN_STATE\(port, pin\)](#) (HAL_GPIO_ReadPin(port, pin))
- #define [ENABLE\(port, pin\)](#) do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_SET); } while(0)
- #define [DISABLE\(port, pin\)](#) do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_RESET); } while(0)

Enumerations

- enum [LEDMode](#) { [LED_MODE_BLINK_FAST](#) , [LED_MODE_BLINK_SLOW](#) , [LED_MODE_ON](#) , [LED_MODE_OFF](#) }

Functions

- void [handle_LED](#) ([LED](#) *led, uint32_t ms_counter)
LED handler function.
- void [handle_direction](#) (volatile int8_t *dir_left, volatile int8_t *dir_right)
Handles the direction of the motors.

Variables

- [LED led_left](#)
- [LED led_right](#)
- [LED ledError](#)

6.17.1 Detailed Description

Header file for handling GPIOs.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.17.2 Macro Definition Documentation

6.17.2.1 DIR_STATE

```
#define DIR_STATE( ) (HAL_GPIO_ReadPin(DIR_GPIO_Port, DIR_Pin))
```

6.17.2.2 DISABLE

```
#define DISABLE(
    port,
    pin ) do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_RESET); } while(0)
```

6.17.2.3 ENABLE

```
#define ENABLE(
    port,
    pin ) do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_SET); } while(0)
```

6.17.2.4 SC_DET_STATE

```
#define SC_DET_STATE( ) (HAL_GPIO_ReadPin(SC_det_GPIO_Port, SC_det_Pin))
```

6.17.2.5 WRN_STATE

```
#define WRN_STATE(
    port,
    pin ) (HAL_GPIO_ReadPin(port, pin))
```

6.17.3 Enumeration Type Documentation

6.17.3.1 LEDMode

```
enum LEDMode
```

Enumerator

LED_MODE_BLINK_FAST	Fast blink mode
LED_MODE_BLINK_SLOW	Slow blink mode
LED_MODE_ON	LED on mode
LED_MODE_OFF	LED off mode

6.17.4 Function Documentation

6.17.4.1 handle_direction()

```
void handle_direction (
    volatile int8_t * dir_left,
    volatile int8_t * dir_right )
```

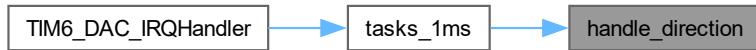
Handles the direction of the motors.

This function reads the state of the DIR switch and updates the directions of both the left and right motors. If one motor is set to rotate clockwise (CW), the other one is set to rotate counterclockwise (CCW), and vice versa.

Parameters

<i>dir_left</i>	Pointer to the direction parameter in the left inverter structure.
<i>dir_right</i>	Pointer to the direction parameter in the right inverter structure.

Here is the caller graph for this function:



6.17.4.2 handle_LED()

```
void handle_LED (
    LED * led,
    uint32_t ms_counter )
```

LED handler function.

This function handles the LED blinking modes based on the LED mode and current millisecond counter.

Parameters

<i>led</i>	Pointer to the LED structure.
<i>ms_counter</i>	Millisecond counter for timing.

This function handles the LED blinking modes based on the LED mode and current millisecond counter.

Parameters

<i>led</i>	Pointer to the LED structure.
<i>ms_counter</i>	Current millisecond counter.

Here is the caller graph for this function:



6.17.5 Variable Documentation

6.17.5.1 led_left

```
LED led_left [extern]
```

6.17.5.2 led_right

```
LED led_right [extern]
```

6.17.5.3 ledError

```
LED ledError [extern]
```

6.18 PCB_IO.h

[Go to the documentation of this file.](#)

```

00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020
00021 #ifndef PCB_IO_H
00022 #define PCB_IO_H
00023
00024 #include "stm32f7xx_hal.h"
00025
00026 // Read SC_det and DIR GPIOs
00027 #define SC_DET_STATE()          (HAL_GPIO_ReadPin(SC_det_GPIO_Port, SC_det_Pin))
00028 #define DIR_STATE()              (HAL_GPIO_ReadPin(DIR_GPIO_Port, DIR_Pin))
00029
00030 // Read WRN GPIOs
00031 #define WRN_STATE(port, pin)     (HAL_GPIO_ReadPin(port, pin))
00032
00033 // Control ENABLE GPIOs
00034 #define ENABLE(port, pin)        do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_SET); } while(0)
00035 #define DISABLE(port, pin)       do { HAL_GPIO_WritePin(port, pin, GPIO_PIN_RESET); } while(0)
00036
  
```

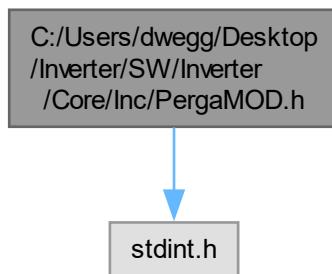
```

00037 // Define LED modes
00038 typedef enum {
00039     LED_MODE_BLINK_FAST,
00040     LED_MODE_BLINK_SLOW,
00041     LED_MODE_ON,
00042     LED_MODE_OFF
00043 } LEDMode;
00044
00045 typedef struct {
00046     GPIO_TypeDef *port;
00047     uint16_t pin;
00048     LEDMode mode;
00049 } LED;
00050
00051 // Declare LED variables as extern
00052 extern LED led_left;
00053 extern LED led_right;
00054 extern LED ledError;
00055
00056 // Function prototypes
00057 void handle_LED(LED *led, uint32_t ms_counter);
00058
00059 void handle_direction(volatile int8_t *dir_left, volatile int8_t *dir_right);
00060
00061 #endif /* PCB_IO_H */

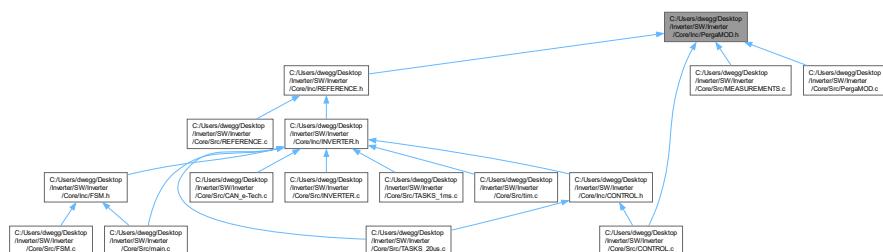
```

6.19 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PergaMOD.h File Reference

#include "stdint.h"
 Include dependency graph for PergaMOD.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [pi_aw_struct](#)
PI Controller with internal saturation, anti-windup, and feedforward.
- struct [pi_struct](#)
PI Controller with external saturation and feedforward.
- struct [clarke3F_struct](#)
Clarke transformation for three-phase systems.
- struct [iclare3F_struct](#)
Inverse Clarke transformation for three-phase systems.
- struct [rot_struct](#)
Rotates the DQ axis in the opposite direction (clockwise).
- struct [irot_struct](#)
Inverse rotation (counterclockwise).
- struct [angle_struct](#)
Generates an angle based on a fixed frequency.
- struct [svpwm_struct](#)
Space Vector Pulse Width Modulation (SVPWM) implementation.
- struct [rampa_struct](#)
Single-ramp generator.
- struct [rampa_dual_struct](#)
Dual-ramp generator.
- struct [datalog_struct](#)
- struct [avg_struct_10](#)
Moving average filter for 10 samples.
- struct [RMS_struct](#)
Root Mean Square (RMS) calculation.
- struct [filtreLP_struct](#)
First-order low-pass filter.
- struct [step_struct](#)
Step function generator.

Macros

- #define [SQ2](#) 1.4142135624F
- #define [ISQ2](#) 0.7071067812F
- #define [SQ3](#) 1.7320508076F
- #define [ISQ3](#) 0.5773502692F
- #define [PI](#) 3.1415926536F
- #define [IPI](#) 0.3183098862F
- #define [PI2](#) 6.2831853072F
- #define [IPI2](#) 0.1591549431F
- #define [INV_DEG](#) 0.0027777778F
- #define [INV3](#) 0.3333333333F
- #define [DIV2](#) 0.5F
- #define [N_DATALOG](#) 256
Data logger for logging variables.

Functions

- void `pi_aw_calc` (volatile `pi_aw_struct` *v) __attribute__((section(".ccmram")))

Initializes the PI controller with anti-windup.
- void `pi_init` (volatile `pi_struct` *v) __attribute__((section(".ccmram")))

Initializes the PI controller.
- void `pi_calc` (volatile `pi_struct` *v) __attribute__((section(".ccmram")))

Calculates the output of the PI controller.
- void `pi_extsat_calc` (volatile `pi_struct` *v)

Calculates the output of the PI controller with external saturation.
- void `clarke3F_calc` (volatile `clarke3F_struct` *v) __attribute__((section(".ccmram")))

Calculates the Clarke transformation.
- void `iclarke3F_calc` (volatile `iclarke3F_struct` *v)

Calculates the inverse Clarke transformation.
- void `rot_calc` (volatile `rot_struct` *v) __attribute__((section(".ccmram")))

Calculates the rotation in the opposite direction.
- void `irot_calc` (volatile `irot_struct` *v) __attribute__((section(".ccmram")))

Calculates the inverse rotation.
- void `angle_calc` (volatile `angle_struct` *p) __attribute__((section(".ccmram")))

Calculates the angle generation.
- void `svpwm_calc` (volatile `svpwm_struct` *v) __attribute__((section(".ccmram")))

Calculates the SVPWM outputs.
- void `rampa_calc` (volatile `rampa_struct` *v) __attribute__((section(".ccmram")))

Calculates the output of the single ramp generator.
- void `rampa_dual_calc` (volatile `rampa_dual_struct` *v) __attribute__((section(".ccmram")))

Calculates the output of the dual ramp generator.
- void `datalog_calc` (volatile `datalog_struct` *dl)

Calculates the data log.
- void `avg_calc_10_samples` (volatile `avg_struct_10` *v)

Calculates the moving average for 10 samples.
- void `RMS_calc` (volatile `RMS_struct` *v) __attribute__((section(".ccmram")))

Calculates the RMS.
- void `filtreLP_init` (volatile `filtreLP_struct` *v)

Initializes the first-order low-pass filter.
- void `filtreLP_calc` (volatile `filtreLP_struct` *v) __attribute__((section(".ccmram")))

Calculates the output of the first-order low-pass filter.
- void `step_calc` (volatile `step_struct` *v) __attribute__((section(".ccmram")))

Calculates the output of the step function generator.

6.20 PergaMOD.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * @file PergaMOD.h
00003  * @brief Library consolidating all the hardware-independent libraries of CITCEA.
00004  * This library is based on Alexandria created by Gabriel Gross and Quim Lopez Mestre in September
00005  * 2006.
00006  * Created by Gabriel Gross, Daniel Heredero, and Tomas Lledo in December 2015.
00007  * Translated to floats by Lucas Bouzon in March 2020.
00008  */
00009 */
0010
0011
0012
```



```

00013
00014
00015
00016
00017 */
00018
00019 #ifndef __PERGAMON_FLOAT_H__
00020 #define __PERGAMON_FLOAT_H__
00021
00022
00023 #include "stdint.h"
00024
00025 #define SQ2      1.4142135624F
00026 #define ISQ2    0.7071067812F
00027 #define SQ3      1.7320508076F
00028 #define ISQ3    0.5773502692F
00029 #define PI       3.1415926536F
00030 #define IPI     0.3183098862F
00031 #define PI2      6.2831853072F
00032 #define IPI2    0.1591549431F
00033 #define INV_DEG 0.0027777778F
00034 #define INV3     0.3333333333F
00035 #define DIV2     0.5F
00036
00037
00038
00039
00040
00041
00042
00043
00044
00045
00046
00047
00048
00049 typedef struct
00050 {
00051     uint16_t      enable;
00052     float        Ts;
00053     float        Kp;
00054     float        Ki;
00055     float        Kaw;
00056     float        e[2];
00057     float        pi_consig;
00058     float        pi_fdb;
00059     float        pi_out_max;
00060     float        pi_out_min;
00061     float        pi_out_presat;
00062     float        pi_out_postsat;
00063     float        pi_out;
00064     float        pi_int[2];
00065     float        pi_ffw[2];
00066     void(*calc)();
00067 } pi_aw_struct;
00068
00069 void pi_aw_calc(volatile pi_aw_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00070
00071
00072 typedef struct
00073 {
00074     uint16_t      enable;
00075     float        Ts;
00076     float        Kp;
00077     float        Ki;
00078     float        K0;
00079     float        K1;
00080     float        e[2];
00081     float        pi_consig;
00082     float        pi_fdb;
00083     float        pi_out_max;
00084     float        pi_out_min;
00085     float        pi_out;
00086     float        pi_ffw[2];
00087     void(*init)();
00088     void(*calc)();
00089 } pi_struct;
00090
00091 void pi_init(volatile pi_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00092
00093 void pi_calc(volatile pi_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00094
00095
00096 void pi_extsat_calc(volatile pi_struct *v);
00097
00098
00099 typedef struct
00100 {
00101     float        a;
00102     float        b;
00103     float        D;
00104     float        Q;
00105     void(*calc)();
00106 } clarke3F_struct;
00107
00108
00109 void clarke3F_calc(volatile clarke3F_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00110
00111
00112 typedef struct
00113 {
00114     float        D;
00115     float        Q;
00116     float        a;
00117     float        b;
00118 }
```

```

00149     void(*calc) ();
00150 } iclarke3F_struct;
00151
00156 void iclarke3F_calc(volatile iclarke3F_struct *v);
00157
00161 typedef struct
00162 {
00163     float      D;
00164     float      Q;
00165     float      sinFi;
00166     float      cosFi;
00167     float      d;
00168     float      q;
00169     void(*calc) ();
00170 } rot_struct;
00171
00176 void rot_calc(volatile rot_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00177
00181 typedef struct
00182 {
00183     float      d;
00184     float      q;
00185     float      sinFi;
00186     float      cosFi;
00187     float      alpha;
00188     float      beta;
00189     void(*calc) ();
00190 } irot_struct;
00191
00196 void irot_calc(volatile irot_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00197
00208 typedef struct {
00209     float      freq;
00210     float      Ts;
00211     float      angle;
00212     void(*calc) ();
00213 } angle_struct;
00214
00219 void angle_calc(volatile angle_struct *p) __attribute__( ( section ( ".ccmram" ) ) );
00220
00224 typedef struct
00225 {
00226     float      alpha;
00227     float      beta;
00228     float      Da;
00229     float      Db;
00230     float      Dc;
00231     void(*calc) ();
00232 } svpwm_struct;
00233
00238 void svpwm_calc(volatile svpwm_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00239
00249 typedef struct
00250 {
00251     float      in;
00252     float      out;
00253     float      Incr;
00254     uint8_t    enable;
00255     void(*calc) ();
00256 } rampa_struct;
00257
00261 typedef struct
00262 {
00263     float      in;
00264     float      out;
00265     float      Incr;
00266     float      Decr;
00267     uint8_t    enable;
00268     void(*calc) ();
00269 } rampa_dual_struct;
00270
00275 void rampa_calc(volatile rampa_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00276
00281 void rampa_dual_calc(volatile rampa_dual_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00282
00287 #define N_DATALOG 256
00289 typedef struct
00290 {
00291     uint16_t    i;
00292     uint16_t    j;
00293     uint16_t    estat;
00294     uint16_t    prescaler;
00295     float       *var;
00296     void(*calc) ();
00297     float       log[N_DATALOG];
00298 } datalog_struct;
00299

```

```

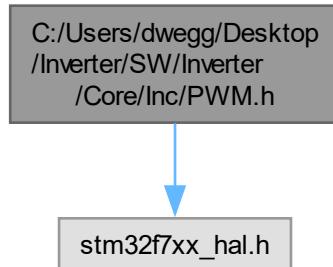
00304 void datalog_calc(volatile datalog_struct *dl);
00305
00309 typedef struct
00310 {
00311     float      out;
00312     float      in[10];
00313 }avg_struct_10;
00314
00319 void avg_calc_10_samples(volatile avg_struct_10 *v);
00320
00324 typedef struct {
00325     float      T_exec;
00326     float      Measure;
00327     float      Sq_Sum;
00328     float      Out_RMS;
00329     float      Freq;
00330     float      Angle;
00331     float      Angle_ant;
00332 } RMS_struct;
00333
00338 void RMS_calc(volatile RMS_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00339
00343 typedef struct
00344 {
00345     float      in;
00346     float      out;
00347     float      alfa;
00348     float      Ts;
00349     float      fc;
00350     uint16_t   enable;
00351     void(*init) ();
00352     void(*calc) ();
00353 } filtreLP_struct;
00354
00359 void filtreLP_init(volatile filtreLP_struct *v);
00360
00365 void filtreLP_calc(volatile filtreLP_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00366
00377 typedef struct
00378 {
00379     float      fs;
00380     float      In;
00381     float      Out;
00382     float      Step;
00383     float      t_step;
00384     uint32_t   Pulses;
00385     uint32_t   Counter;
00386     uint16_t   enable;
00387     void(*calc) ();
00388 } step_struct;
00389
00394 void step_calc(volatile step_struct *v) __attribute__( ( section ( ".ccmram" ) ) );
00395
00398 #endif // __PERGAMON_FLOAT_H__

```

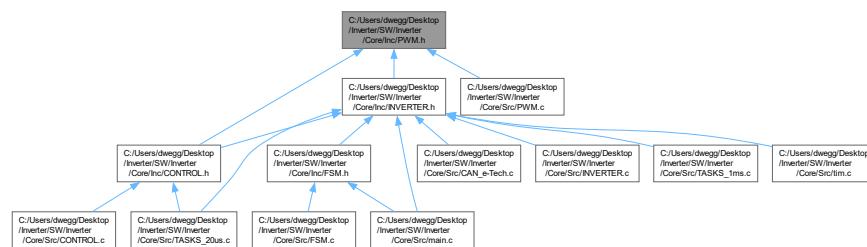
6.21 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PWM.h File Reference

Header file for controlling PWM output.

```
#include "stm32f7xx_hal.h"
Include dependency graph for PWM.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [Duties](#)

Structure to hold PWM configuration parameters.

Functions

- void [enable_PWM](#) (TIM_HandleTypeDef *htim)
Enable PWM output.
- void [disable_PWM](#) (TIM_HandleTypeDef *htim)
Disable PWM output.
- void [update_PWM](#) (TIM_HandleTypeDef *htim, [Duties](#) duties)
Set PWM duty cycles.

6.21.1 Detailed Description

Header file for controlling PWM output.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.21.2 Function Documentation

6.21.2.1 disable_PWM()

```
void disable_PWM (  
    TIM_HandleTypeDef * htim )
```

Disable PWM output.

This function disables PWM output for the specified timer.

Parameters

<i>htim</i>	Pointer to the TIM_HandleTypeDef structure.
-------------	---

6.21.2.2 enable_PWM()

```
void enable_PWM (  
    TIM_HandleTypeDef * htim )
```

Enable PWM output.

This function enables PWM output for the specified timer.

Parameters

<i>htim</i>	Pointer to the TIM_HandleTypeDef structure.
-------------	---

6.21.2.3 update_PWM()

```
void update_PWM (  
    TIM_HandleTypeDef * htim,  
    Duties duties )
```

Set PWM duty cycles.

This function sets the duty cycles for the PWM channels.

Parameters

<i>htim</i>	Pointer to the TIM_HandleTypeDef structure.
<i>duties</i>	Duties structure containing duty cycle values.

Here is the caller graph for this function:



6.22 PWM.h

[Go to the documentation of this file.](#)

```

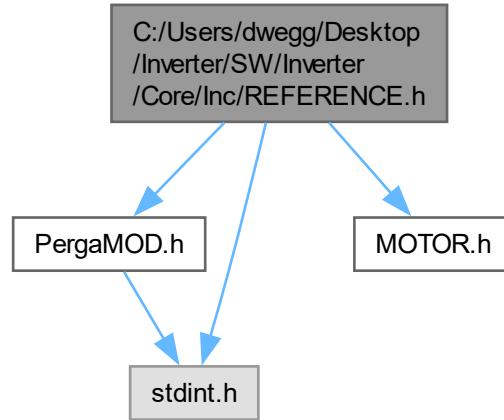
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 #ifndef PWM_H
00021 #define PWM_H
00022
00023 #include "stm32f7xx_hal.h"
00024
00028 typedef struct {
00029     float Da;
00030     float Db;
00031     float Dc;
00032 } Duties;
00033
00041 void enable_PWM(TIM_HandleTypeDef *htim);
00042
00050 void disable_PWM(TIM_HandleTypeDef *htim);
00051
00052
00061 void update_PWM(TIM_HandleTypeDef *htim, Duties duties);
00062
00063 #endif /* PWM_H */
  
```

6.23 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/REFERENCE.h File Reference

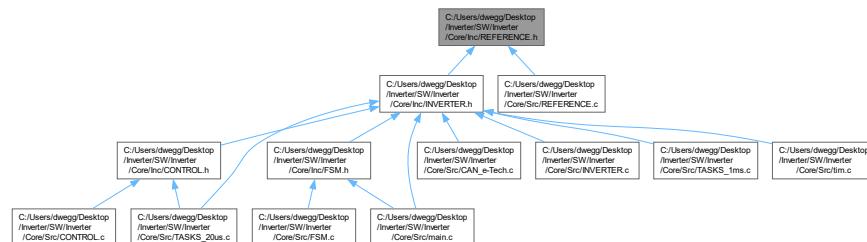
Header file for torque reference handling.

```
#include "PergaMOD.h"
#include "MOTOR.h"
```

```
#include <stdint.h>
Include dependency graph for REFERENCE.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `Reference`

Structure for reference values.

Functions

- float `handle_torqueRef` (float torqueRefIn, int8_t direction, float torqueMax, float speedMaxRPM, float speedMeas, volatile `pi_struct` *loopSpeed)
- Handles torque control based on the reference torque, direction, maximum torque, maximum speed, measured speed, maximum torque rate of change, speed control loop parameters, and sampling time.*
- void `initialize_torque_ramp` (float dTorqueMax, float Ts)
- Initializes torque ramp.*
- float `set_torque_direction` (float torqueRef, int8_t direction)
- Set torque direction based on inverter direction.*
- float `saturate_symmetric` (float ref, float max)

- Symmetrically saturate a reference value.*
- float `limit_torque_to_prevent_overspeed` (float `speedMax`, float `speedMeas`, float `torqueRefIn`, volatile `pi_struct *loopSpeed`)
Speed loop acts as a torque saturation, reducing torque in order to limit the maximum speed.

6.23.1 Detailed Description

Header file for torque reference handling.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.23.2 Function Documentation

6.23.2.1 handle_torqueRef()

```
float handle_torqueRef (
    float torqueRefIn,
    int8_t direction,
    float torqueMax,
    float speedMaxRPM,
    float speedMeas,
    volatile pi_struct * loopSpeed )
```

Handles torque control based on the reference torque, direction, maximum torque, maximum speed, measured speed, maximum torque rate of change, speed control loop parameters, and sampling time.

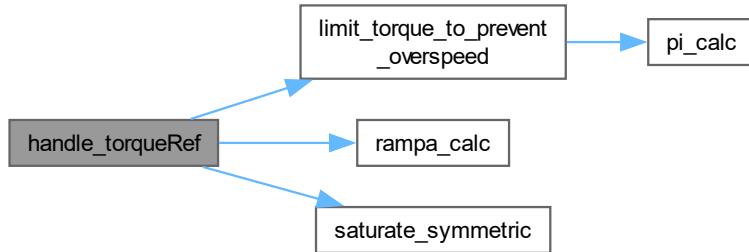
Parameters

<code>torqueRefIn</code>	Input reference torque.
<code>direction</code>	Direction of torque (1 for positive torque, -1 for negative torque).
<code>torqueMax</code>	Maximum allowable torque.
<code>speedMaxRPM</code>	Maximum allowable speed in RPM.
<code>speedMeas</code>	Measured speed.
<code>loopSpeed</code>	Speed control loop parameters.

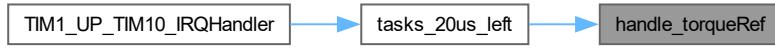
Returns

The output torque after handling direction, saturation, and rate limiting.

Here is the call graph for this function:



Here is the caller graph for this function:

**6.23.2.2 initialize_torque_ramp()**

```
void initialize_torque_ramp (
    float dTorqueMax,
    float Ts )
```

Initializes torque ramp.

Parameters

<code>dTorqueMax</code>	Maximum allowable rate of change of torque.
<code>Ts</code>	Sampling time.

6.23.2.3 limit_torque_to_prevent_overspeed()

```
float limit_torque_to_prevent_overspeed (
    float speedMaxRPM,
    float speedMeas,
```

```
float torqueRefIn,
volatile pi_struct * loopSpeed )
```

Speed loop acts as a torque saturation, reducing torque in order to limit the maximum speed.

Parameters

in	<i>speedMax</i>	The maximum speed value in RPM.
in	<i>speedMeas</i>	The measured speed value in RPM.
in	<i>torqueRefIn</i>	The torque reference value before this saturation.
in	<i>loopSpeed</i>	Pointer to the speed PI controller structure.

Returns

torqueRef_out The limited torque reference value after this saturation.

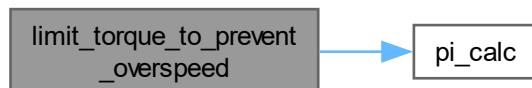
Parameters

in	<i>speedMaxRPM</i>	The maximum speed value in RPM.
in	<i>speedMeas</i>	The measured speed value in RPM.
in	<i>torqueRefIn</i>	The torque reference value before this saturation.
in	<i>loopSpeed</i>	Pointer to the speed PI controller structure.

Returns

torqueRefOut The limited torque reference value after this saturation.

Here is the call graph for this function:



Here is the caller graph for this function:



6.23.2.4 `saturate_symmetric()`

```
float saturate_symmetric (
    float ref,
    float max )
```

Symmetrically saturate a reference value.

This function symmetrically saturates a reference value based on the maximum allowed value. If the reference value exceeds the maximum allowed value, it is saturated to the maximum value. If the reference value is less than the negative of the maximum allowed value, it is saturated to the negative of the maximum value.

Parameters

in	<i>ref</i>	The reference value to saturate.
in	<i>max</i>	The maximum allowed value for saturation.

Returns

The saturated reference value.

Here is the caller graph for this function:



6.23.2.5 `set_torque_direction()`

```
float set_torque_direction (
    float torqueRefIn,
    int8_t direction )
```

Set torque direction based on inverter direction.

This function adjusts the torque reference based on the direction of the inverter. If the inverter is set to rotate counterclockwise (CCW), positive torque represents braking. If the inverter is set to rotate clockwise (CW), positive torque represents traction.

Parameters

in	<i>torqueRef</i>	The torque reference value to adjust.
in	<i>direction</i>	Pointer to the direction of the inverter (1 for CW, -1 for CCW).

Returns

The adjusted torque reference value.

This function adjusts the torque reference based on the desired direction. If the motor is set to rotate counterclockwise (CCW), positive torque represents traction, negative is braking. If the motor is set to rotate clockwise (CW), negative torque represents traction, positive is braking.

Parameters

in	<i>torqueRefIn</i>	The torque reference value to adjust.
in	<i>direction</i>	Pointer to the direction of the inverter (1 for CW, -1 for CCW).

Returns

torqueRefOut The adjusted torque reference value.

6.24 REFERENCE.h

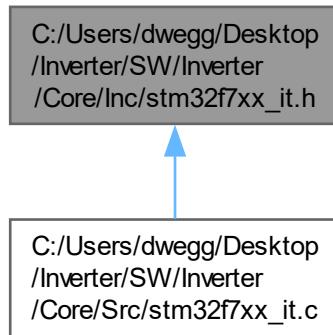
[Go to the documentation of this file.](#)

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 #ifndef REFERENCE_H
00021 #define REFERENCE_H
00022
00023 #include "PergaMOD.h" // ramp, pi struct
00024 #include "MOTOR.h" // motor struct
00025 #include <stdint.h>
00026
00030 typedef struct {
00031     float idRef;
00032     float iqRef;
00033     float torqueRef;
00034 } Reference;
00035
00036
00050 float handle_torqueRef(float torqueRefIn, int8_t direction, float torqueMax, float speedMaxRPM, float
    speedMeas, volatile pi_struct *loopSpeed);
00051
00058 void initialize_torque_ramp(float dTorqueMax, float Ts);
00059
00060
00072 float set_torque_direction(float torqueRef, int8_t direction);
00073
00085 float saturate_symmetric(float ref, float max);
00086
00095 float limit_torque_to_prevent_overspeed(float speedMax, float speedMeas, float torqueRefIn, volatile
    pi_struct *loopSpeed);
00096
00097 #endif /* REFERENCE_H */
```

6.25 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/stm32f7xx_it.h File Reference

This file contains the headers of the interrupt handlers.

This graph shows which files directly or indirectly include this file:



Functions

- void [NMI_Handler](#) (void)
This function handles Non maskable interrupt.
- void [HardFault_Handler](#) (void)
This function handles Hard fault interrupt.
- void [MemManage_Handler](#) (void)
This function handles Memory management fault.
- void [BusFault_Handler](#) (void)
This function handles Pre-fetch fault, memory access fault.
- void [UsageFault_Handler](#) (void)
This function handles Undefined instruction or illegal state.
- void [SVC_Handler](#) (void)
This function handles System service call via SWI instruction.
- void [DebugMon_Handler](#) (void)
This function handles Debug monitor.
- void [PendSV_Handler](#) (void)
This function handles Pendable request for system service.
- void [SysTick_Handler](#) (void)
This function handles System tick timer.
- void [CAN1_RX0_IRQHandler](#) (void)
This function handles CAN1 RX0 interrupts.
- void [TIM1_UP_TIM10_IRQHandler](#) (void)
This function handles TIM1 update interrupt and TIM10 global interrupt.
- void [TIM6_DAC_IRQHandler](#) (void)
This function handles TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts.
- void [DMA2_Stream0_IRQHandler](#) (void)
This function handles DMA2 stream0 global interrupt.
- void [DMA2_Stream1_IRQHandler](#) (void)
This function handles DMA2 stream1 global interrupt.
- void [DMA2_Stream2_IRQHandler](#) (void)
This function handles DMA2 stream2 global interrupt.

6.25.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

Copyright (c) 2023 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

6.25.2 Function Documentation

6.25.2.1 BusFault_Handler()

```
void BusFault_Handler (
    void )
```

This function handles Pre-fetch fault, memory access fault.

6.25.2.2 CAN1_RX0_IRQHandler()

```
void CAN1_RX0_IRQHandler (
    void )
```

This function handles CAN1 RX0 interrupts.

Here is the call graph for this function:



6.25.2.3 DebugMon_Handler()

```
void DebugMon_Handler (
    void )
```

This function handles Debug monitor.

6.25.2.4 DMA2_Stream0_IRQHandler()

```
void DMA2_Stream0_IRQHandler (
    void )
```

This function handles DMA2 stream0 global interrupt.

6.25.2.5 DMA2_Stream1_IRQHandler()

```
void DMA2_Stream1_IRQHandler (
    void )
```

This function handles DMA2 stream1 global interrupt.

6.25.2.6 DMA2_Stream2_IRQHandler()

```
void DMA2_Stream2_IRQHandler (
    void )
```

This function handles DMA2 stream2 global interrupt.

6.25.2.7 HardFault_Handler()

```
void HardFault_Handler (
    void )
```

This function handles Hard fault interrupt.

6.25.2.8 MemManage_Handler()

```
void MemManage_Handler (
    void )
```

This function handles Memory management fault.

6.25.2.9 NMI_Handler()

```
void NMI_Handler (
    void )
```

This function handles Non maskable interrupt.

6.25.2.10 PendSV_Handler()

```
void PendSV_Handler (
    void )
```

This function handles Pendable request for system service.

6.25.2.11 SVC_Handler()

```
void SVC_Handler (
    void )
```

This function handles System service call via SWI instruction.

6.25.2.12 SysTick_Handler()

```
void SysTick_Handler (
    void )
```

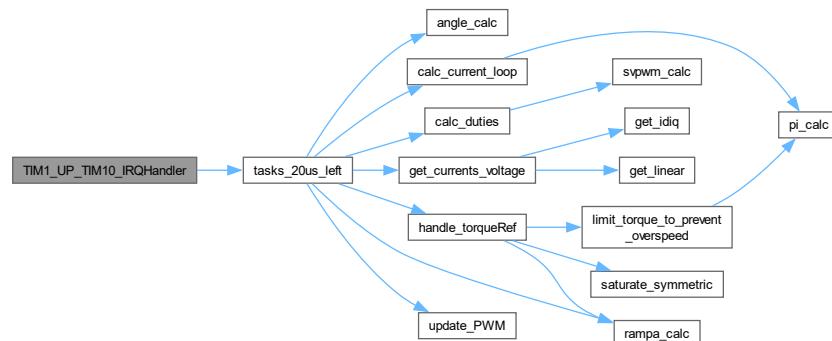
This function handles System tick timer.

6.25.2.13 TIM1_UP_TIM10_IRQHandler()

```
void TIM1_UP_TIM10_IRQHandler (
    void )
```

This function handles TIM1 update interrupt and TIM10 global interrupt.

Here is the call graph for this function:

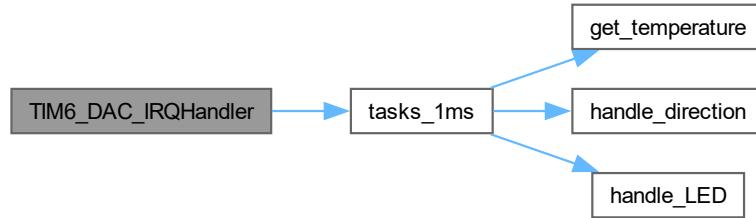


6.25.2.14 TIM6_DAC_IRQHandler()

```
void TIM6_DAC_IRQHandler (
    void )
```

This function handles TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts.

Here is the call graph for this function:



6.25.2.15 UsageFault_Handler()

```
void UsageFault_Handler (
    void )
```

This function handles Undefined instruction or illegal state.

6.26 stm32f7xx_it.h

[Go to the documentation of this file.](#)

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Define to prevent recursive inclusion -----*/
00021 #ifndef __STM32F7XX_IT_H
00022 #define __STM32F7XX_IT_H
00023
00024 #ifdef __cplusplus
00025     extern "C" {
00026 #endif
00027
00028 /* Private includes -----*/
00029 /* USER CODE BEGIN Includes */
00030
00031 /* USER CODE END Includes */
00032
00033 /* Exported types -----*/
00034 /* USER CODE BEGIN ET */
00035
00036 /* USER CODE END ET */
00037
00038 /* Exported constants -----*/
00039 /* USER CODE BEGIN EC */
00040
00041 /* USER CODE END EC */
00042
00043 /* Exported macro -----*/
00044 /* USER CODE BEGIN EM */
00045
00046 /* USER CODE END EM */
00047
00048 /* Exported functions prototypes -----*/
00049 void NMI_Handler(void);
00050 void HardFault_Handler(void);
00051 void MemManage_Handler(void);
00052 void BusFault_Handler(void);
00053 void UsageFault_Handler(void);
00054 void SVC_Handler(void);
00055 void DebugMon_Handler(void);
00056 void PendSV_Handler(void);
```

```

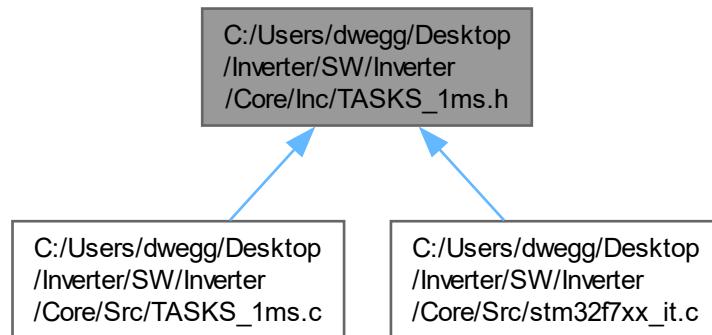
00057 void SysTick_Handler(void);
00058 void CAN1_RX0_IRQHandler(void);
00059 void TIM1_UP_TIM10_IRQHandler(void);
00060 void TIM6_DAC_IRQHandler(void);
00061 void DMA2_Stream0_IRQHandler(void);
00062 void DMA2_Stream1_IRQHandler(void);
00063 void DMA2_Stream2_IRQHandler(void);
00064 /* USER CODE BEGIN EFP */
00065
00066 /* USER CODE END EFP */
00067
00068 #ifdef __cplusplus
00069 }
00070 #endif
00071
00072 #endif /* __STM32F7xx_IT_H */

```

6.27 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_1ms.h File Reference

Header file for functions related to tasks executed every 1ms.

This graph shows which files directly or indirectly include this file:



Functions

- void `tasks_1ms` (void)
Function to be executed every 1ms.

6.27.1 Detailed Description

Header file for functions related to tasks executed every 1ms.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.27.2 Function Documentation

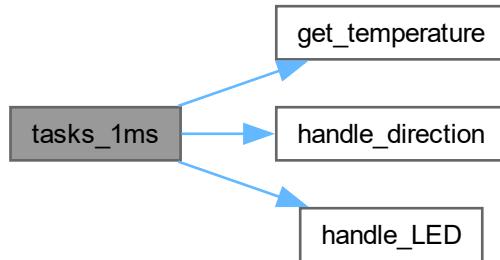
6.27.2.1 tasks_1ms()

```
void tasks_1ms (
    void )
```

Function to be executed every 1ms.

This function is called by the TIM6 IRQ handler every millisecond.

This function is called by the TIM6 IRQ handler every millisecond. It increments the millisecond counter and executes all the low priority tasks. Here is the call graph for this function:



Here is the caller graph for this function:



6.28 TASKS_1ms.h

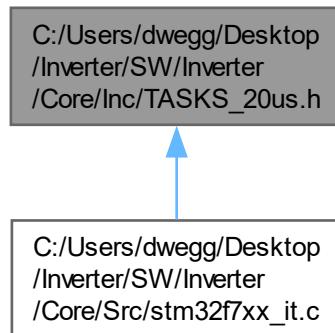
[Go to the documentation of this file.](#)

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020
00021 #ifndef TASKS_1MS_H
00022 #define TASKS_1MS_H
00023
00029 void tasks_1ms(void);
00030
00031 #endif /* TASKS_1MS_H */
```

6.29 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS_20us.h File Reference

Header file for functions related to tasks executed every 20us in each PWM timer interruption.

This graph shows which files directly or indirectly include this file:



Functions

- void [tasks_20us_left \(\)](#)
Function to be executed every TS.
- void [tasks_20us_right \(\)](#)
Function to be executed every TS.

6.29.1 Detailed Description

Header file for functions related to tasks executed every 20us in each PWM timer interruption.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.29.2 Function Documentation

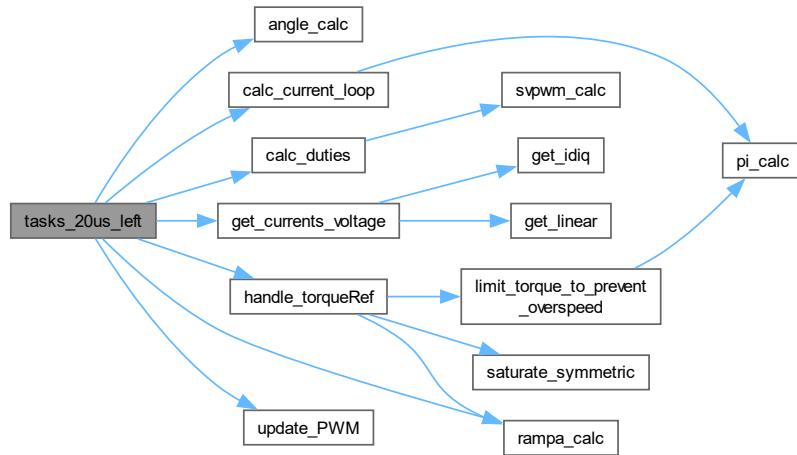
6.29.2.1 tasks_20us_left()

```
void tasks_20us_left (
    void )
```

Function to be executed every TS.

This function is called by the TIM1 trigger out handler every TS.

This function is called by the TIM1 trigger handler every TS. Here is the call graph for this function:



Here is the caller graph for this function:



6.29.2.2 tasks_20us_right()

```
void tasks_20us_right (
    void )
```

Function to be executed every TS.

This function is called by the TIM8 trigger out handler every TS.

This function is called by the TIM8 trigger handler every TS.

6.30 TASKS_20us.h

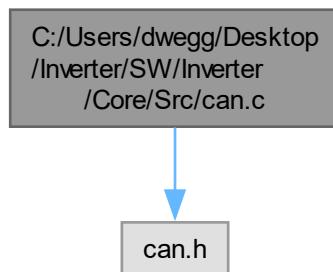
[Go to the documentation of this file.](#)

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00025 void tasks_20us_left();
00026
00032 void tasks_20us_right();
```

6.31 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/can.c File Reference

This file provides code for the configuration of the CAN instances.

```
#include "can.h"
Include dependency graph for can.c:
```



Functions

- void [MX_CAN1_Init](#) (void)
- void [HAL_CAN_MspInit](#) (CAN_HandleTypeDef *canHandle)
- void [HAL_CAN_MspDeInit](#) (CAN_HandleTypeDef *canHandle)

Variables

- CAN_HandleTypeDef [hcan1](#)

6.31.1 Detailed Description

This file provides code for the configuration of the CAN instances.

Attention

Copyright (c) 2024 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

6.31.2 Function Documentation

6.31.2.1 HAL_CAN_MspDeInit()

```
void HAL_CAN_MspDeInit (
    CAN_HandleTypeDef * canHandle )
```

CAN1 GPIO Configuration PD0 -----> CAN1_RX PD1 -----> CAN1_TX

6.31.2.2 HAL_CAN_MspInit()

```
void HAL_CAN_MspInit (
    CAN_HandleTypeDef * canHandle )
```

CAN1 GPIO Configuration PD0 -----> CAN1_RX PD1 -----> CAN1_TX

6.31.2.3 MX_CAN1_Init()

```
void MX_CAN1_Init (
    void )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.3 Variable Documentation

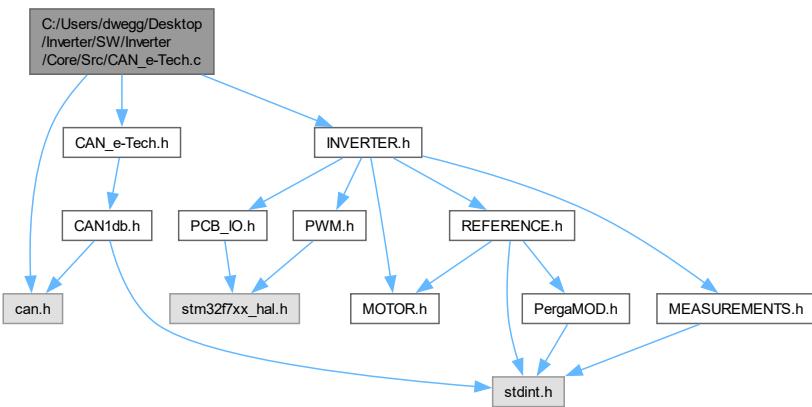
6.31.3.1 hcan1

```
CAN_HandleTypeDef hcan1
```

6.32 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CAN_e-Tech.c File Reference

This file contains functions to handle CAN communication with the car.

```
#include "can.h"
#include "INVERTER.h"
#include "CAN_e-Tech.h"
Include dependency graph for CAN_e-Tech.c:
```



Functions

- void `handle_CAN (CAN_HandleTypeDef *hcan)`
Handle CAN messages.
- void `send_CAN_message (CAN_HandleTypeDef *hcan, void *dbc_msg, const float *data)`
Send a CAN message using `CAN1db.h` information.

Variables

- `uint8_t enableCAN`

6.32.1 Detailed Description

This file contains functions to handle CAN communication with the car.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.32.2 Function Documentation

6.32.2.1 handle_CAN()

```
void handle_CAN (
    CAN_HandleTypeDef * hcan )
```

Handle CAN messages.

This function implements the logic to handle received CAN messages.

Parameters

<i>hcan</i>	Pointer to the CAN handle structure.
-------------	--------------------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



6.32.2.2 send_CAN_message()

```
void send_CAN_message (
    CAN_HandleTypeDef * hcan,
    void * dbc_msg,
    const float * data )
```

Send a CAN message using [CAN1db.h](#) information.

This function prepares and sends a CAN message using information from [CAN1db.h](#).

Parameters

<i>hcan</i>	Pointer to the CAN handle structure.
<i>dbc_msg</i>	Pointer to the structure containing CAN message information from CAN1db.h .
<i>data</i>	Pointer to the array of float data to be sent.

Here is the caller graph for this function:



6.32.3 Variable Documentation

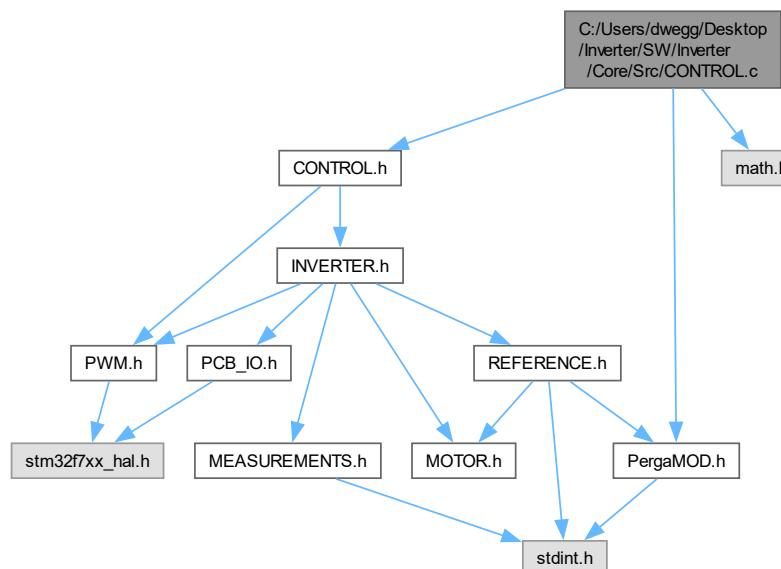
6.32.3.1 enableCAN

`uint8_t enableCAN`

6.33 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CONTROL.c File Reference

This file provides code for the control loop.

```
#include "CONTROL.h"
#include <math.h>
#include <PergaMOD.h>
Include dependency graph for CONTROL.c:
```



Functions

- void `calc_current_loop` (volatile `InverterStruct` *`inv`)
Calculates the id-iq currents control actions.
- void `calc_duties` (float `vd`, float `vq`, float `vDC`, float `sinTheta_e`, float `cosTheta_e`, volatile `Duties` *`duties`)
function.

6.33.1 Detailed Description

This file provides code for the control loop.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.33.2 Function Documentation

6.33.2.1 `calc_current_loop()`

```
void calc_current_loop (
    volatile InverterStruct * inv )
```

Calculates the id-iq currents control actions.

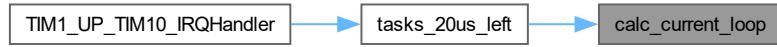
Parameters

<code>inv</code>	Pointer to the inverter structure.
------------------	------------------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



6.33.2.2 calc_duties()

```

void calc_duties (
    float vd,
    float vq,
    float vDC,
    float sinTheta_e,
    float cosTheta_e,
    volatile Duties * duties )
  
```

function.

This function calculates the inverse Park transform and the duty cycles using SVPWM

Parameters

in	<i>vd</i>	Voltage in the d-axis.
in	<i>vq</i>	Voltage in the q-axis.
in	<i>vDC</i>	DC voltage.
in	<i>sinTheta_e</i>	Electrical angle sine (-1..1)
in	<i>cosTheta_e</i>	Electrical angle cosine (-1..1)
out	<i>duties</i>	Pointer to the duties structure.

Here is the call graph for this function:



Here is the caller graph for this function:

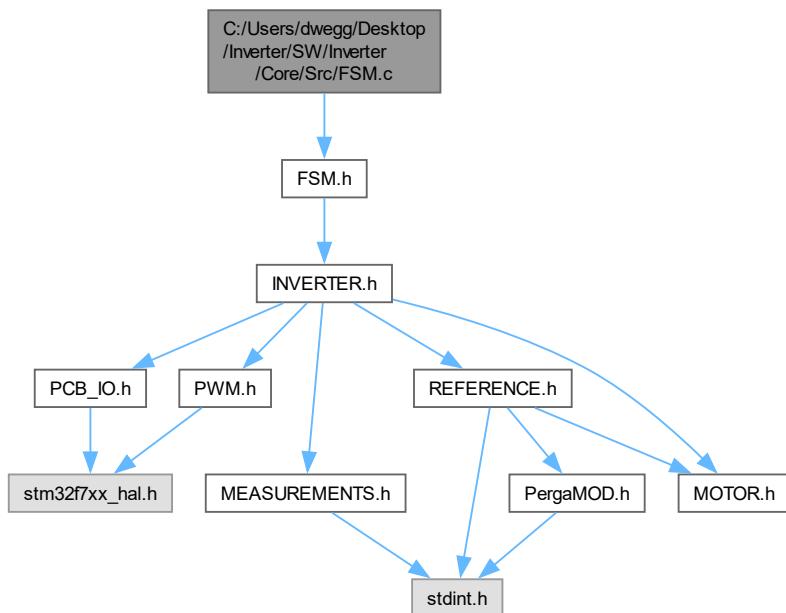


6.34 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/FSM.c File Reference

This file provides code for the inverter Finite State Machine.

```
#include "FSM.h"
```

Include dependency graph for FSM.c:



Functions

- void eval_inv_FSM (volatile InverterStruct *inv)

Execute the finite state machine for inverter.

6.34.1 Detailed Description

This file provides code for the inverter Finite State Machine.

Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.34.2 Function Documentation

6.34.2.1 eval_inv_FSM()

```
void eval_inv_FSM (
    volatile InverterStruct * inv )
```

Execute the finite state machine for inverter.

Run the Finite State Machine (FSM) for inverter operation control.

This function executes the finite state machine to control the inverter based on its current state.

Parameters

<i>inv</i>	Pointer to the inverter structure.
------------	------------------------------------

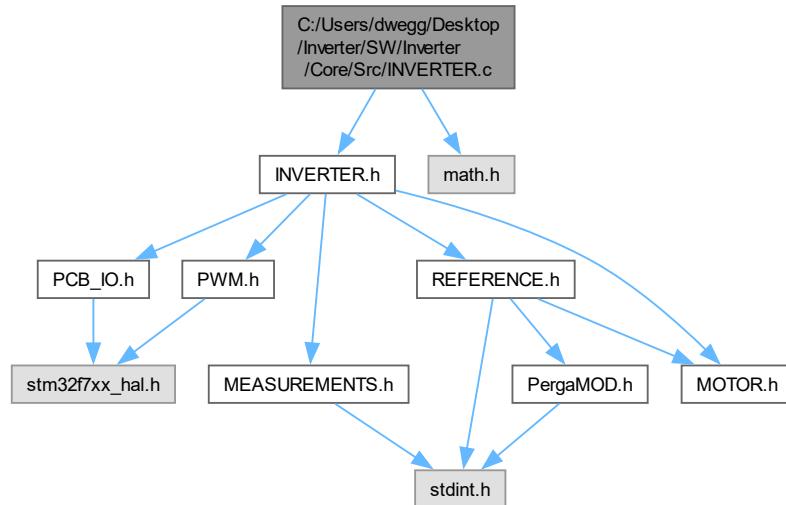
Here is the caller graph for this function:



6.35 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/INVERTER.c File Reference

This file provides code for the inverter struct.

```
#include "INVERTER.h"
#include <math.h>
Include dependency graph for INVERTER.c:
```



Functions

- void `initialize_inverter` (volatile InverterStruct *inv, LED *led, GPIO_TypeDef *enable_port, uint16_t enable_pin, TIM_HandleTypeDef *htim, ADC_HandleTypeDef *hadc, MotorParameters *motor)
Initialize the inverter.
- void `init_control_loops` (volatile InverterStruct *inv, MotorParameters *motor)
Initializes the PI controllers.

Variables

- volatile InverterStruct `inverter_left` = {0}
Left inverter structure.
- volatile InverterStruct `inverter_right` = {0}
Right inverter structure.

6.35.1 Detailed Description

This file provides code for the inverter struct.

Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

6.35.2 Function Documentation

6.35.2.1 init_control_loops()

```
void init_control_loops (
    volatile InverterStruct * inv,
    MotorParameters * motor )
```

Initializes the PI controllers.

Initializes the id-iq current control PI controllers.

Parameters

<i>inv</i>	Pointer to the inverter structure.
------------	------------------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



6.35.2.2 initialize_inverter()

```
void initialize_inverter (
    volatile InverterStruct * inv,
    LED * led,
    GPIO_TypeDef * enable_port,
    uint16_t enable_pin,
    TIM_HandleTypeDef * htim,
    ADC_HandleTypeDef * hadc,
    MotorParameters * motor )
```

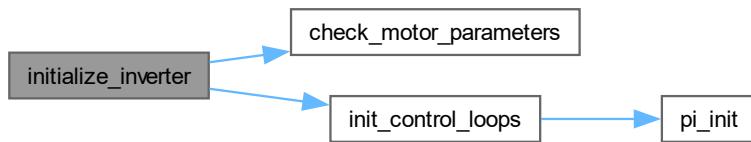
Initialize the inverter.

This function initializes the inverter structure with the specified [LED](#), GPIO port, and pin.

Parameters

<code>out</code>	<code>inv</code>	Pointer to the inverter structure.
<code>in</code>	<code>led</code>	Pointer to the <code>LED</code> structure.
<code>in</code>	<code>enable_port</code>	Pointer to the GPIO port for enabling/disabling the inverter.
<code>in</code>	<code>enable_pin</code>	Pin number for enabling/disabling the inverter.
<code>in</code>	<code>htim</code>	Timer peripheral for the PWM output.
<code>in</code>	<code>hadc</code>	ADC peripheral for the current phase current and DC voltage sensing.
<code>in</code>	<code>motor</code>	<code>MotorParameters</code> struct.

Here is the call graph for this function:



Here is the caller graph for this function:



6.35.3 Variable Documentation

6.35.3.1 inverter_left

```
volatile InverterStruct inverter_left = {0}
```

Left inverter structure.

External declaration of the left inverter structure.

6.35.3.2 inverter_right

```
volatile InverterStruct inverter_right = {0}
```

Right inverter structure.

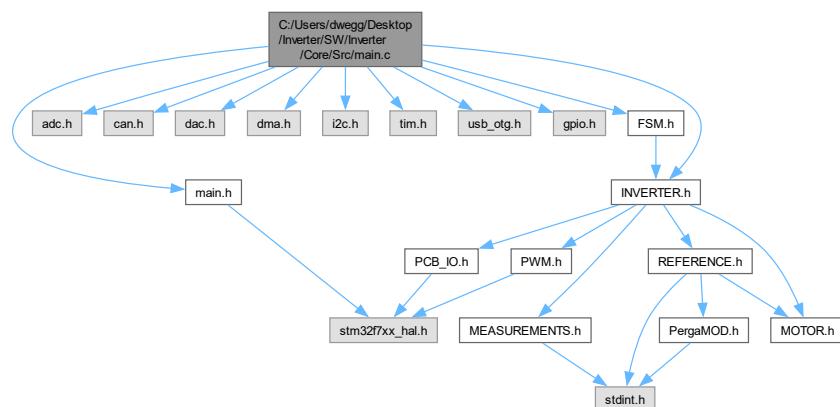
External declaration of the right inverter structure.

6.36 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/main.c File Reference

: Main program body

```
#include "main.h"
#include "adc.h"
#include "can.h"
#include "dac.h"
#include "dma.h"
#include "i2c.h"
#include "tim.h"
#include "usb_otg.h"
#include "gpio.h"
#include "FSM.h"
#include "INVERTER.h"
```

Include dependency graph for main.c:



Functions

- void [SystemClock_Config](#) (void)
System Clock Configuration.
- int [main](#) (void)
The application entry point.
- void [Error_Handler](#) (void)
This function is executed in case of error occurrence.

6.36.1 Detailed Description

: Main program body

Attention

Copyright (c) 2023 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

6.36.2 Function Documentation

6.36.2.1 Error_Handler()

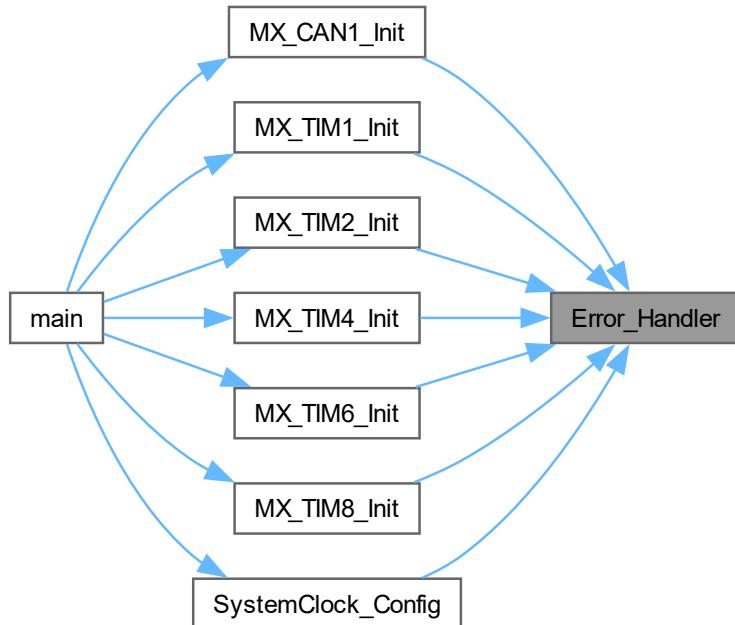
```
void Error_Handler (
    void )
```

This function is executed in case of error occurrence.

Return values

None	
------	--

Here is the caller graph for this function:



6.36.2.2 main()

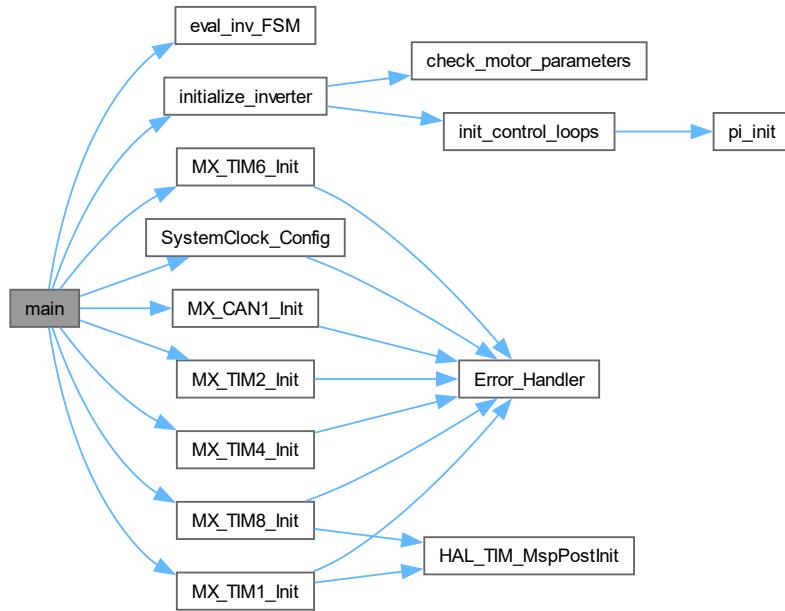
```
int main (
    void )
```

The application entry point.

Return values

int	
-----	--

Here is the call graph for this function:



6.36.2.3 SystemClock_Config()

```
void SystemClock_Config (
    void )
```

System Clock Configuration.

Return values

None	<input type="button" value=""/>
------	---------------------------------

Configure the main internal regulator output voltage

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Activate the Over-Drive mode

Initializes the CPU, AHB and APB buses clocks

Here is the call graph for this function:



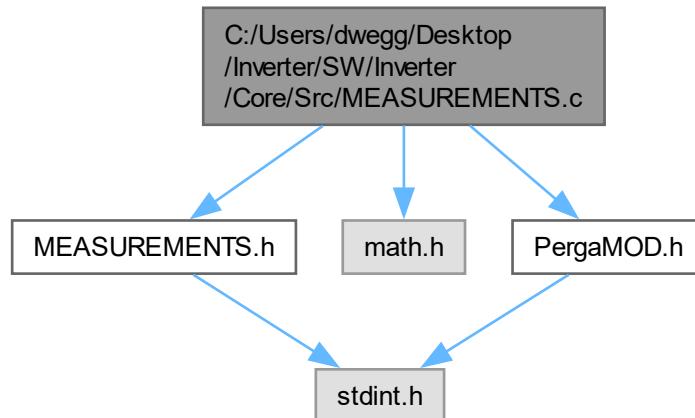
Here is the caller graph for this function:



6.37 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MEASUREMENTS.c File Reference

This file provides functions for handling measurements.

```
#include "MEASUREMENTS.h"
#include <math.h>
#include <PergaMOD.h>
Include dependency graph for MEASUREMENTS.c:
```



Functions

- `uint8_t get_currents_voltage (volatile uint32_t ADC_raw[], volatile Analog *analog, volatile Feedback *feedback, float sinTheta_e, float cosTheta_e)`
Get electrical ADC measurements.
- `float get_linear (uint32_t bits, float slope, float offset)`
Convert ADC reading to physical measurement with linear response.
- `void get_idiq (float ia, float ib, float ic, float sinTheta_e, float cosTheta_e, float *idMeas, float *iqMeas)`
Computes d-q currents from current measurements and electrical angle.
- `float get_temperature (uint32_t bits, const float tempLUT[])`
Retrieves temperature from a lookup table based on ADC bits.

Variables

5.24, 5.25, 5.26, 5.27, 5.28, 5.29, 5.30, 5.31, 5.32, 5.33, 5.34, 5.34, 5.35, 5.36, 5.37, 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, 5.43, 5.44, 5.45, 5.46, 5.47, 5.48, 5.49, 5.50, 5.51, 5.52, 5.53, 5.53, 5.54, 5.55, 5.56, 5.57, 5.58, 5.59, 5.60, 5.61, 5.62, 5.63, 5.64, 5.64, 5.65, 5.66, 5.67, 5.68, 5.69, 5.70, 5.71, 5.72, 5.73, 5.74, 5.75, 5.75, 5.76, 5.77, 5.78, 5.79, 5.80, 5.81, 5.82, 5.83, 5.84, 5.85, 5.86, 5.87, 5.88, 5.88, 5.89, 5.90, 5.91, 5.92, 5.93, 5.94, 5.95, 5.96, 5.97, 5.98, 5.99, 6.00, 6.01, 6.01, 6.02, 6.03, 6.04, 6.05, 6.06, 6.07, 6.08, 6.09, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.16, 6.17, 6.18, 6.19, 6.20, 6.21, 6.22, 6.23, 6.24, 6.25, 6.26, 6.27, 6.28, 6.29, 6.30, 6.31, 6.32, 6.32, 6.33, 6.34, 6.35, 6.36, 6.37, 6.38, 6.39, 6.40, 6.41, 6.42, 6.43, 6.44, 6.45, 6.46, 6.47, 6.48, 6.49, 6.50, 6.51, 6.51, 6.52, 6.53, 6.54, 6.55, 6.56, 6.57, 6.58, 6.59, 6.60, 6.61, 6.62, 6.63, 6.64, 6.65, 6.66, 6.67, 6.68, 6.69, 6.70, 6.71, 6.72, 6.73, 6.74, 6.75, 6.75, 6.76, 6.77, 6.78, 6.79, 6.80, 6.81, 6.82, 6.83, 6.84, 6.85, 6.86, 6.87, 6.88, 6.89, 6.90, 6.91, 6.92, 6.93, 6.94, 6.95, 6.96, 6.97, 6.98, 6.99, 7.00, 7.01, 7.02, 7.03, 7.04, 7.05, 7.06, 7.07, 7.08, 7.09, 7.10, 7.11, 7.12, 7.12, 7.13, 7.14, 7.15, 7.16, 7.17, 7.18, 7.19, 7.20, 7.21, 7.22, 7.23, 7.24, 7.25, 7.26, 7.27, 7.28, 7.29, 7.30, 7.31, 7.32, 7.33, 7.34, 7.35, 7.36, 7.37, 7.38, 7.39, 7.40, 7.41, 7.42, 7.43, 7.44, 7.45, 7.46, 7.47, 7.48, 7.49, 7.50, 7.51, 7.52, 7.53, 7.54, 7.55, 7.56, 7.57, 7.58, 7.59, 7.60, 7.61, 7.62, 7.63, 7.64, 7.65, 7.66, 7.67, 7.68, 7.69, 7.70, 7.71, 7.72, 7.73, 7.74, 7.75, 7.76, 7.77, 7.78, 7.79, 7.80, 7.81, 7.82, 7.83, 7.84, 7.85, 7.86, 7.87, 7.88, 7.89, 7.91, 7.92, 7.93, 7.94, 7.95, 7.96, 7.97, 7.98, 7.99, 8.00, 8.01, 8.02, 8.03, 8.04, 8.05, 8.06, 8.07, 8.08, 8.09, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 8.23, 8.24, 8.25, 8.26, 8.27, 8.29, 8.30, 8.31, 8.32, 8.33, 8.34, 8.35, 8.36, 8.37, 8.38, 8.39, 8.40, 8.41, 8.42, 8.43, 8.44, 8.45, 8.46, 8.47, 8.48, 8.49, 8.50, 8.51, 8.52, 8.54, 8.55, 8.56, 8.57, 8.58, 8.59, 8.60, 8.61, 8.62, 8.63, 8.64, 8.65, 8.66, 8.67, 8.68, 8.69, 8.70, 8.71, 8.72, 8.74, 8.75, 8.76, 8.77, 8.78, 8.79, 8.80, 8.81, 8.82, 8.83, 8.84, 8.85, 8.86, 8.87, 8.88, 8.89, 8.91, 8.92, 8.93, 8.94, 8.95, 8.96, 8.97, 8.98, 8.99, 9.00, 9.01, 9.02, 9.03, 9.04, 9.06, 9.07, 9.08, 9.09, 9.10, 9.11, 9.12, 9.13, 9.14, 9.15, 9.16, 9.17, 9.18, 9.20, 9.21, 9.22, 9.23, 9.24, 9.25, 9.26, 9.27, 9.28, 9.29, 9.30, 9.31, 9.33, 9.34, 9.35, 9.36, 9.37, 9.38, 9.39, 9.40, 9.41, 9.42, 9.43, 9.45, 9.46, 9.47, 9.48, 9.49, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55, 9.57, 9.58, 9.59, 9.60, 9.61, 9.62, 9.63, 9.64, 9.65, 9.66, 9.68, 9.69, 9.70, 9.71, 9.72, 9.73, 9.74, 9.75, 9.76, 9.78, 9.79, 9.80, 9.81, 9.82, 9.83, 9.84, 9.85, 9.86, 9.88, 9.89, 9.90, 9.91, 9.92, 9.93, 9.94, 9.95, 9.96, 9.98, 9.99, 10.00, 10.01, 10.02, 10.03, 10.04, 10.05, 10.07, 10.08, 10.09, 10.10, 10.11, 10.12, 10.13, 10.14, 10.16, 10.17, 10.18, 10.19, 10.20, 10.21, 10.22, 10.24, 10.25, 10.26, 10.27, 10.28, 10.29, 10.30, 10.31, 10.33, 10.34, 10.35, 10.36, 10.37, 10.38, 10.39, 10.41, 10.42, 10.43, 10.44, 10.45, 10.46, 10.47, 10.49, 10.50, 10.51, 10.52, 10.53, 10.54, 10.55, 10.57, 10.58, 10.59, 10.60, 10.61, 10.62, 10.64, 10.65, 10.66, 10.67, 10.68, 10.69, 10.70, 10.72, 10.73, 10.74, 10.75, 10.76, 10.77, 10.79, 10.80, 10.81, 10.82, 10.83, 10.84, 10.86, 10.87, 10.88, 10.89, 10.90, 10.91, 10.93, 10.94, 10.95, 10.96, 10.97, 10.98, 11.00, 11.01, 11.02, 11.03, 11.04, 11.05, 11.07, 11.08, 11.09, 11.10, 11.11, 11.13, 11.14, 11.15, 11.16, 11.17, 11.18, 11.20, 11.21, 11.22, 11.23, 11.24, 11.26, 11.27, 11.28, 11.29, 11.30, 11.32, 11.33, 11.34, 11.35, 11.36, 11.37, 11.39, 11.40, 11.41, 11.42, 11.43, 11.45, 11.46, 11.47, 11.48, 11.49, 11.51, 11.52, 11.53, 11.54, 11.55, 11.57, 11.58, 11.59, 11.60, 11.61, 11.63, 11.64, 11.65, 11.66, 11.68, 11.69, 11.70, 11.71, 11.72, 11.74, 11.75, 11.76, 11.77, 11.78, 11.80, 11.81, 11.82, 11.83, 11.85, 11.86, 11.87, 11.88, 11.89, 11.91, 11.92, 11.93, 11.94, 11.96, 11.97, 11.98, 11.99, 12.00, 12.02, 12.03, 12.04, 12.05, 12.07, 12.08, 12.09, 12.10, 12.11, 12.13, 12.14, 12.15, 12.16, 12.18, 12.19, 12.20, 12.21, 12.23, 12.24, 12.25, 12.26, 12.28, 12.29, 12.30, 12.31, 12.33, 12.34, 12.35, 12.36, 12.38, 12.39, 12.40, 12.41, 12.43, 12.44, 12.45, 12.46, 12.48, 12.49, 12.50, 12.51, 12.53, 12.54, 12.55, 12.56, 12.58, 12.59, 12.60, 12.61, 12.63, 12.64, 12.65, 12.66, 12.68, 12.69, 12.70, 12.72, 12.73, 12.74, 12.75, 12.77, 12.78, 12.79, 12.80, 12.82, 12.83, 12.84, 12.86, 12.87, 12.88, 12.89, 12.91, 12.92, 12.93, 12.94, 12.96, 12.97, 12.98, 13.00, 13.01, 13.02, 13.03, 13.05, 13.06, 13.07, 13.09, 13.10, 13.11, 13.12, 13.14, 13.15, 13.16, 13.18, 13.19, 13.20, 13.22, 13.23, 13.24, 13.25, 13.27, 13.28, 13.29, 13.31, 13.32, 13.33, 13.35, 13.36, 13.37, 13.38, 13.40, 13.41, 13.42, 13.44, 13.45, 13.46, 13.48, 13.49, 13.50, 13.52, 13.53, 13.54, 13.55, 13.57, 13.58, 13.59, 13.61, 13.62, 13.63, 13.65, 13.66, 13.67, 13.69, 13.70, 13.71, 13.73, 13.74, 13.75, 13.77, 13.78, 13.79, 13.81, 13.82, 13.83, 13.85, 13.86, 13.87, 13.89, 13.90, 13.91, 13.93, 13.94, 13.95, 13.97, 13.98, 13.99, 14.01, 14.02, 14.03, 14.05, 14.06, 14.07, 14.09, 14.10, 14.11, 14.13, 14.14, 14.16, 14.17, 14.18, 14.20, 14.21, 14.22, 14.24, 14.25, 14.26, 14.28, 14.29, 14.30, 14.32, 14.33, 14.35, 14.36, 14.37, 14.39, 14.40, 14.41, 14.43, 14.44, 14.45, 14.47, 14.48, 14.50, 14.51, 14.52, 14.54, 14.55, 14.56, 14.58, 14.59, 14.61, 14.62, 14.63, 14.65, 14.66, 14.67, 14.69, 14.70, 14.72, 14.73, 14.74, 14.76, 14.77, 14.79, 14.80, 14.81, 14.83, 14.84, 14.86, 14.87, 14.88, 14.90, 14.91, 14.93, 14.94, 14.95, 14.97, 14.98, 15.00, 15.01, 15.02, 15.04, 15.05, 15.07, 15.08, 15.09, 15.11, 15.12, 15.14, 15.15, 15.16, 15.18, 15.19, 15.21, 15.22, 15.24, 15.25, 15.26, 15.28, 15.29, 15.31, 15.32, 15.33, 15.35, 15.36, 15.38, 15.39, 15.41, 15.42, 15.43, 15.45, 15.46, 15.48, 15.49, 15.51, 15.52, 15.54, 15.55, 15.56, 15.58, 15.59, 15.61, 15.62, 15.64, 15.65, 15.66, 15.68, 15.69, 15.71, 15.72, 15.74, 15.75, 15.77, 15.78, 15.80, 15.81, 15.82, 15.84, 15.85, 15.87, 15.88, 15.90, 15.91, 15.93, 15.94, 15.96, 15.97, 15.99, 16.00, 16.01, 16.03, 16.04, 16.06, 16.07, 16.09, 16.10, 16.12, 16.13, 16.15, 16.16, 16.18, 16.19, 16.21, 16.22, 16.24, 16.25, 16.27, 16.28, 16.30, 16.31, 16.33, 16.34, 16.35, 16.37, 16.38, 16.40, 16.41, 16.43, 16.44, 16.46, 16.47, 16.49, 16.50, 16.52, 16.53, 16.55, 16.56, 16.58, 16.59, 16.61, 16.62, 16.64,

64, 16.66, 16.67, 16.69, 16.70, 16.72, 16.73, 16.75, 16.76, 16.78, 16.79, 16.81, 16.82, 16.84, 16.85, 16.87, 16.88, 16.90, 16.91, 16.93, 16.94, 16.96, 16.97, 16.99, 17.01, 17.02, 17.04, 17.05, 17.07, 17.08, 17.10, 17.11, 17.13, 17.14, 17.16, 17.17, 17.19, 17.21, 17.22, 17.24, 17.25, 17.27, 17.28, 17.30, 17.31, 17.33, 17.35, 17.36, 17.38, 17.39, 17.41, 17.42, 17.44, 17.45, 17.47, 17.49, 17.50, 17.52, 17.53, 17.55, 17.56, 17.58, 17.60, 17.61, 17.63, 17.64, 17.66, 17.67, 17.69, 17.71, 17.72, 17.74, 17.75, 17.77, 17.79, 17.80, 17.82, 17.83, 17.85, 17.86, 17.88, 17.90, 17.91, 17.93, 17.94, 17.96, 17.98, 17.99, 18.01, 18.02, 18.04, 18.06, 18.07, 18.09, 18.11, 18.12, 18.14, 18.15, 18.17, 18.19, 18.20, 18.22, 18.23, 18.25, 18.27, 18.28, 18.30, 18.32, 18.33, 18.35, 18.36, 18.38, 18.40, 18.41, 18.43, 18.45, 18.46, 18.48, 18.49, 18.51, 18.53, 18.54, 18.56, 18.58, 18.59, 18.61, 18.63, 18.64, 18.66, 18.68, 18.69, 18.71, 18.73, 18.74, 18.76, 18.77, 18.79, 18.81, 18.82, 18.84, 18.86, 18.87, 18.89, 18.91, 18.92, 18.94, 18.96, 18.97, 18.99, 19.01, 19.02, 19.04, 19.06, 19.08, 19.09, 19.11, 19.13, 19.14, 19.16, 19.18, 19.19, 19.21, 19.23, 19.24, 19.26, 19.28, 19.29, 19.31, 19.33, 19.35, 19.36, 19.38, 19.40, 19.41, 19.43, 19.45, 19.46, 19.48, 19.50, 19.52, 19.53, 19.55, 19.57, 19.58, 19.60, 19.62, 19.64, 19.65, 19.67, 19.69, 19.70, 19.72, 19.74, 19.76, 19.77, 19.79, 19.81, 19.83, 19.84, 19.86, 19.88, 19.90, 19.91, 19.93, 19.95, 19.97, 19.98, 20.00, 20.02, 20.04, 20.05, 20.07, 20.09, 20.11, 20.12, 20.14, 20.16, 20.18, 20.19, 20.21, 20.23, 20.25, 20.26, 20.28, 20.30, 20.32, 20.33, 20.35, 20.37, 20.39, 20.41, 20.42, 20.44, 20.46, 20.48, 20.49, 20.51, 20.53, 20.55, 20.57, 20.58, 20.60, 20.62, 20.64, 20.66, 20.67, 20.69, 20.71, 20.73, 20.75, 20.76, 20.78, 20.80, 20.82, 20.84, 20.85, 20.87, 20.89, 20.91, 20.93, 20.95, 20.96, 20.98, 21.00, 21.02, 21.04, 21.06, 21.07, 21.09, 21.11, 21.13, 21.15, 21.17, 21.18, 21.20, 21.22, 21.24, 21.26, 21.28, 21.29, 21.31, 21.33, 21.35, 21.37, 21.39, 21.41, 21.42, 21.44, 21.46, 21.48, 21.50, 21.52, 21.54, 21.55, 21.57, 21.59, 21.61, 21.63, 21.65, 21.67, 21.69, 21.70, 21.72, 21.74, 21.76, 21.78, 21.80, 21.82, 21.84, 21.86, 21.87, 21.89, 21.91, 21.93, 21.95, 21.97, 21.99, 22.01, 22.03, 22.05, 22.06, 22.08, 22.10, 22.12, 22.14, 22.16, 22.18, 22.20, 22.22, 22.24, 22.26, 22.28, 22.30, 22.31, 22.33, 22.35, 22.37, 22.39, 22.41, 22.43, 22.45, 22.47, 22.49, 22.51, 22.53, 22.55, 22.57, 22.59, 22.61, 22.63, 22.64, 22.66, 22.68, 22.70, 22.72, 22.74, 22.76, 22.78, 22.80, 22.82, 22.84, 22.86, 22.88, 22.90, 22.92, 22.94, 22.96, 22.98, 23.00, 23.02, 23.04, 23.06, 23.08, 23.10, 23.12, 23.14, 23.16, 23.18, 23.20, 23.22, 23.24, 23.26, 23.28, 23.30, 23.32, 23.34, 23.36, 23.38, 23.40, 23.42, 23.44, 23.46, 23.48, 23.50, 23.52, 23.54, 23.56, 23.58, 23.60, 23.62, 23.65, 23.67, 23.69, 23.71, 23.73, 23.75, 23.77, 23.79, 23.81, 23.83, 23.85, 23.87, 23.89, 23.91, 23.93, 23.95, 23.97, 24.00, 24.02, 24.04, 24.06, 24.08, 24.10, 24.12, 24.14, 24.16, 24.18, 24.20, 24.22, 24.25, 24.27, 24.29, 24.31, 24.33, 24.35, 24.37, 24.39, 24.41, 24.43, 24.46, 24.48, 24.50, 24.52, 24.54, 24.56, 24.58, 24.60, 24.63, 24.65, 24.67, 24.69, 24.71, 24.73, 24.75, 24.78, 24.80, 24.82, 24.84, 24.86, 24.88, 24.90, 24.93, 24.95, 24.97, 24.99, 25.01, 25.03, 25.06, 25.08, 25.10, 25.12, 25.14, 25.16, 25.19, 25.21, 25.23, 25.25, 25.27, 25.30, 25.32, 25.34, 25.36, 25.38, 25.41, 25.43, 25.45, 25.47, 25.49, 25.52, 25.54, 25.56, 25.58, 25.60, 25.63, 25.65, 25.67, 25.69, 25.72, 25.74, 25.76, 25.78, 25.81, 25.83, 25.85, 25.87, 25.89, 25.92, 25.94, 25.96, 25.98, 26.01, 26.03, 26.05, 26.08, 26.10, 26.12, 26.14, 26.17, 26.19, 26.21, 26.23, 26.26, 26.28, 26.30, 26.33, 26.35, 26.37, 26.39, 26.42, 26.44, 26.46, 26.49, 26.51, 26.53, 26.56, 26.58, 26.60, 26.63, 26.65, 26.67, 26.69, 26.72, 26.74, 26.76, 26.79, 26.81, 26.83, 26.86, 26.88, 26.90, 26.93, 26.95, 26.98, 27.00, 27.02, 27.05, 27.07, 27.09, 27.12, 27.14, 27.16, 27.19, 27.21, 27.24, 27.26, 27.28, 27.31, 27.33, 27.35, 27.38, 27.40, 27.43, 27.45, 27.47, 27.50, 27.52, 27.55, 27.57, 27.59, 27.62, 27.64, 27.67, 27.69, 27.72, 27.74, 27.76, 27.79, 27.81, 27.84, 27.86, 27.89, 27.91, 27.93, 27.96, 27.98, 28.01, 28.03, 28.06, 28.08, 28.11, 28.13, 28.16, 28.18, 28.21, 28.23, 28.26, 28.28, 28.30, 28.33, 28.35, 28.38, 28.40, 28.43, 28.45, 28.48, 28.50, 28.53, 28.55, 28.58, 28.60, 28.63, 28.66, 28.68, 28.71, 28.73, 28.76, 28.78, 28.81, 28.83, 28.86, 28.88, 28.91, 28.93, 28.96, 28.99, 29.01, 29.04, 29.06, 29.09, 29.11, 29.14, 29.17, 29.19, 29.22, 29.24, 29.27, 29.29, 29.32, 29.35, 29.37, 29.40, 29.42, 29.45, 29.48, 29.50, 29.53, 29.55, 29.58, 29.61, 29.63, 29.66, 29.69, 29.71, 29.74, 29.76, 29.79, 29.82, 29.84, 29.87, 29.90, 29.92, 29.95, 29.98, 30.00, 30.03, 30.06, 30.08, 30.11, 30.14, 30.16, 30.19, 30.22, 30.24, 30.27, 30.30, 30.33, 30.35, 30.38, 30.41, 30.43, 30.46, 30.49, 30.52, 30.54, 30.57, 30.60, 30.62, 30.65, 30.68, 30.71, 30.73, 30.76, 30.79, 30.82, 30.84, 30.87, 30.90, 30.93, 30.96, 30.98, 31.01, 31.04, 31.07, 31.09, 31.12, 31.15, 31.18, 31.21, 31.23, 31.26, 31.29, 31.32, 31.35, 31.37, 31.40, 31.43, 31.46, 31.49, 31.52, 31.54, 31.57, 31.60, 31.63, 31.66, 31.69, 31.72, 31.74, 31.77, 31.80, 31.83, 31.86, 31.89, 31.92, 31.95, 31.97, 32.00, 32.03, 32.06, 32.09, 32.12, 32.15, 32.18, 32.21, 32.24, 32.27, 32.29, 32.32, 32.35, 32.38, 32.41, 32.44, 32.47, 32.50, 32.53, 32.56, 32.59, 32.62, 32.65, 32.68, 32.71, 32.74, 32.77, 32.80, 32.83, 32.86, 32.89, 32.92, 32.95, 32.98, 33.01, 33.04, 33.07, 33.10, 33.13, 33.16, 33.19, 33.22, 33.25, 33.28, 33.31, 33.34, 33.37, 33.40, 33.43, 33.46, 33.49, 33.53, 33.56, 33.59, 33.62, 33.65, 33.68, 33.71, 33.74, 33.77, 33.80, 33.84, 33.87, 33.90, 33.93, 33.96, 33.99, 34.02, 34.05, 34.09, 34.12, 34.15, 34.18, 34.21, 34.24, 34.28, 34.31, 34.34, 34.37, 34.40, 34.43, 34.47, 34.50, 34.53, 34.56, 34.59, 34.63, 34.66, 34.69, 34.72, 34.76, 34.79, 34.82, 34.85, 34.89, 34.92, 34.95, 34.98, 35.02, 35.05, 35.08, 35.11, 35.15, 35.18, 35.21, 35.25, 35.28, 35.31, 35.35, 35.38, 35.41, 35.44, 35.48, 35.51, 35.54, 35.58, 35.61, 35.65, 35.68, 35.71, 35.75, 35.78, 35.81, 35.85, 35.88, 35.91, 35.95, 35.98, 36.02, 36.05, 36.08, 36.12, 36.15, 36.19, 36.22, 36.26, 36.29, 36.33, 36.36, 36.39, 36.43, 36.46, 36.50, 36.53, 36.57, 36.60, 36.64, 36.67, 36.71, 36.74, 36.78, 36.81, 36.85, 36.88, 36.89

- ```

• const float tempMotorLUT [] = {-2.45, -2.44, -2.44, -2.43, -2.42, -2.42, -2.41, -2.41, -2.40, -2.39, -2.39, -2.38,
-2.37, -2.37, -2.36, -2.36, -2.35, -2.34, -2.34, -2.33, -2.32, -2.32, -2.31, -2.31, -2.30, -2.29, -2.29, -2.28, -2.27,
-2.27, -2.26, -2.26, -2.25, -2.24, -2.24, -2.23, -2.22, -2.22, -2.21, -2.20, -2.20, -2.19, -2.19, -2.18, -2.17, -2.17,
-2.16, -2.15, -2.15, -2.14, -2.14, -2.13, -2.12, -2.12, -2.11, -2.10, -2.10, -2.09, -2.08, -2.08, -2.07, -2.07, -2.06,
-2.05, -2.05, -2.04, -2.03, -2.03, -2.02, -2.01, -2.01, -2.00, -2.00, -1.99, -1.98, -1.98, -1.97, -1.96, -1.96, -1.95,
-1.94, -1.94, -1.93, -1.93, -1.92, -1.91, -1.91, -1.90, -1.89, -1.89, -1.88, -1.87, -1.87, -1.86, -1.86, -1.85, -1.84,
-1.84, -1.83, -1.82, -1.82, -1.81, -1.80, -1.80, -1.79, -1.78, -1.78, -1.77, -1.77, -1.76, -1.75, -1.75, -1.75, -1.74, -1.73,
-1.73, -1.72, -1.71, -1.71, -1.70, -1.69, -1.69, -1.68, -1.67, -1.67, -1.66, -1.66, -1.65, -1.64, -1.64, -1.63, -1.62,
-1.62, -1.61, -1.60, -1.60, -1.59, -1.58, -1.58, -1.57, -1.56, -1.56, -1.55, -1.54, -1.54, -1.53, -1.53, -1.52, -1.51,
-1.51, -1.50, -1.49, -1.49, -1.48, -1.47, -1.47, -1.46, -1.45, -1.45, -1.44, -1.43, -1.43, -1.42, -1.41, -1.41, -1.40,
-1.39, -1.39, -1.38, -1.37, -1.37, -1.36, -1.36, -1.35, -1.34, -1.34, -1.34, -1.33, -1.32, -1.32, -1.32, -1.31, -1.30, -1.30, -1.29,
-1.28, -1.28, -1.27, -1.26, -1.26, -1.25, -1.24, -1.24, -1.23, -1.22, -1.22, -1.22, -1.21, -1.21, -1.20, -1.20, -1.20, -1.19, -1.18, -1.18,
-1.17, -1.16, -1.16, -1.15, -1.14, -1.14, -1.13, -1.12, -1.12, -1.12, -1.11, -1.11, -1.10, -1.10, -1.09, -1.08, -1.08, -1.08, -1.07, -1.06,
-1.06, -1.05, -1.04, -1.04, -1.03, -1.02, -1.02, -1.01, -1.00, -1.00, -0.99, -0.98, -0.98, -0.97, -0.96, -0.96, -0.95,
-0.94, -0.94, -0.93, -0.92, -0.92, -0.91, -0.90, -0.90, -0.89, -0.88, -0.88, -0.87, -0.86, -0.86, -0.85, -0.84, -0.84,
-0.83, -0.82, -0.82, -0.81, -0.80, -0.80, -0.79, -0.78, -0.78, -0.77, -0.76, -0.76, -0.75, -0.74, -0.73, -0.73, -0.72,
-0.71, -0.71, -0.70, -0.69, -0.69, -0.68, -0.67, -0.67, -0.66, -0.65, -0.65, -0.64, -0.63, -0.63, -0.62, -0.61, -0.61,
-0.60, -0.59, -0.59, -0.58, -0.57, -0.56, -0.56, -0.55, -0.54, -0.54, -0.54, -0.53, -0.52, -0.52, -0.51, -0.50, -0.50, -0.49,
-0.48, -0.48, -0.47, -0.46, -0.46, -0.45, -0.44, -0.43, -0.43, -0.42, -0.41, -0.41, -0.40, -0.39, -0.39, -0.38, -0.37,
-0.37, -0.36, -0.35, -0.35, -0.34, -0.33, -0.32, -0.32, -0.31, -0.31, -0.30, -0.30, -0.29, -0.28, -0.28, -0.27, -0.26, -0.26,
-0.25, -0.24, -0.23, -0.23, -0.22, -0.21, -0.21, -0.20, -0.19, -0.19, -0.18, -0.17, -0.17, -0.16, -0.15, -0.14, -0.14,
-0.13, -0.12, -0.12, -0.11, -0.10, -0.10, -0.09, -0.08, -0.07, -0.07, -0.06, -0.05, -0.05, -0.04, -0.03, -0.03, -0.02,
-0.01, -0.00, -0.00, -0.01, -0.02, -0.02, -0.03, -0.04, -0.04, -0.05, -0.06, -0.07, -0.07, -0.08, -0.09, -0.09, -0.09, -0.10, -0.11,
-0.12, -0.12, -0.13, -0.14, -0.15, -0.16, -0.16, -0.17, -0.18, -0.19, -0.19, -0.19, -0.20, -0.21, -0.21, -0.21, -0.22, -0.23, -0.24, -0.24,
-0.25, -0.26, -0.26, -0.27, -0.28, -0.29, -0.29, -0.30, -0.31, -0.31, -0.32, -0.33, -0.34, -0.34, -0.34, -0.35, -0.36, -0.36, -0.37, -0.38,
-0.39, -0.39, -0.40, -0.41, -0.41, -0.41, -0.42, -0.43, -0.43, -0.44, -0.44, -0.44, -0.45, -0.46, -0.46, -0.46, -0.47, -0.48, -0.49, -0.49, -0.50, -0.51, -0.51,
-0.52, -0.53, -0.54, -0.54, -0.55, -0.56, -0.56, -0.57, -0.58, -0.59, -0.59, -0.60, -0.61, -0.61, -0.62, -0.63, -0.64, -0.64, -0.65,
-0.66, -0.67, -0.67, -0.68, -0.69, -0.69, -0.70, -0.71, -0.72, -0.72, -0.73, -0.74, -0.74, -0.75, -0.75, -0.76, -0.76, -0.77, -0.77, -0.78, -0.79,
-0.80, -0.80, -0.81, -0.82, -0.83, -0.83, -0.84, -0.85, -0.85, -0.86, -0.86, -0.87, -0.87, -0.88, -0.88, -0.89, -0.89, -0.90, -0.90, -0.91, -0.91, -0.92, -0.93,
-0.94, -0.94, -0.95, -0.95, -0.96, -0.96, -0.96, -0.97, -0.97, -0.98, -0.98, -0.99, -0.99, -1.00, -1.01, -1.02, -1.02, -1.03, -1.03, -1.04, -1.04, -1.05, -1.05, -1.06, -1.06,
-1.07, -1.08, -1.08, -1.09, -1.10, -1.10, -1.11, -1.12, -1.13, -1.13, -1.14, -1.14, -1.15, -1.16, -1.16, -1.17, -1.17, -1.18, -1.18, -1.19, -1.19, -1.19, -1.20, -1.21, -1.21,
-1.22, -1.22, -1.23, -1.24, -1.25, -1.25, -1.26, -1.27, -1.28, -1.28, -1.29, -1.30, -1.31, -1.31, -1.32, -1.33, -1.33, -1.34, -1.34, -1.35,
-1.36, -1.37, -1.37, -1.38, -1.39, -1.40, -1.40, -1.41, -1.42, -1.43, -1.43, -1.44, -1.44, -1.45, -1.46, -1.46, -1.47, -1.47, -1.48, -1.48, -1.49, -1.49, -1.49}

```

1.50, 1.51, 1.52, 1.52, 1.53, 1.54, 1.55, 1.55, 1.56, 1.57, 1.58, 1.58, 1.59, 1.60, 1.61, 1.61, 1.61, 1.62, 1.63, 1.64, 1.64, 1.65, 1.66, 1.67, 1.67, 1.68, 1.69, 1.70, 1.71, 1.71, 1.72, 1.73, 1.74, 1.74, 1.75, 1.75, 1.76, 1.76, 1.77, 1.77, 1.78, 1.79, 1.80, 1.80, 1.81, 1.82, 1.83, 1.84, 1.84, 1.85, 1.86, 1.87, 1.87, 1.88, 1.89, 1.90, 1.90, 1.91, 1.92, 1.93, 1.93, 1.94, 1.95, 1.96, 1.97, 1.97, 1.98, 1.99, 2.00, 2.00, 2.01, 2.02, 2.03, 2.04, 2.04, 2.05, 2.06, 2.07, 2.07, 2.08, 2.09, 2.10, 2.10, 2.11, 2.12, 2.13, 2.14, 2.14, 2.15, 2.16, 2.17, 2.17, 2.18, 2.19, 2.20, 2.21, 2.21, 2.22, 2.23, 2.24, 2.25, 2.25, 2.26, 2.27, 2.28, 2.28, 2.29, 2.30, 2.31, 2.32, 2.32, 2.33, 2.34, 2.35, 2.35, 2.36, 2.37, 2.38, 2.39, 2.39, 2.40, 2.41, 2.42, 2.43, 2.43, 2.44, 2.45, 2.45, 2.46, 2.46, 2.47, 2.47, 2.48, 2.49, 2.50, 2.50, 2.51, 2.52, 2.53, 2.54, 2.54, 2.55, 2.56, 2.57, 2.58, 2.58, 2.59, 2.60, 2.61, 2.62, 2.62, 2.63, 2.64, 2.65, 2.66, 2.66, 2.67, 2.68, 2.69, 2.70, 2.70, 2.71, 2.72, 2.73, 2.74, 2.74, 2.75, 2.76, 2.77, 2.78, 2.78, 2.79, 2.80, 2.81, 2.82, 2.82, 2.83, 2.84, 2.85, 2.86, 2.86, 2.87, 2.88, 2.89, 2.90, 2.90, 2.91, 2.92, 2.93, 2.94, 2.94, 2.95, 2.96, 2.97, 2.98, 2.98, 2.99, 3.00, 3.01, 3.02, 3.02, 3.03, 3.04, 3.05, 3.06, 3.07, 3.07, 3.08, 3.09, 3.10, 3.11, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 3.16, 3.17, 3.18, 3.19, 3.20, 3.20, 3.21, 3.22, 3.23, 3.24, 3.24, 3.25, 3.26, 3.26, 3.27, 3.28, 3.29, 3.29, 3.30, 3.31, 3.32, 3.33, 3.34, 3.34, 3.35, 3.36, 3.37, 3.38, 3.38, 3.39, 3.40, 3.41, 3.42, 3.43, 3.43, 3.44, 3.45, 3.46, 3.47, 3.48, 3.48, 3.49, 3.50, 3.51, 3.52, 3.53, 3.53, 3.54, 3.55, 3.56, 3.57, 3.58, 3.58, 3.59, 3.60, 3.61, 3.62, 3.63, 3.63, 3.64, 3.65, 3.66, 3.67, 3.68, 3.68, 3.69, 3.70, 3.71, 3.72, 3.73, 3.73, 3.74, 3.75, 3.76, 3.77, 3.78, 3.78, 3.79, 3.80, 3.81, 3.82, 3.83, 3.83, 3.84, 3.85, 3.86, 3.87, 3.88, 3.89, 3.89, 3.90, 3.91, 3.92, 3.93, 3.94, 3.94, 3.95, 3.96, 3.97, 3.98, 3.99, 4.00, 4.00, 4.01, 4.02, 4.03, 4.04, 4.05, 4.05, 4.05, 4.06, 4.07, 4.08, 4.09, 4.10, 4.11, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.23, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29, 4.29, 4.30, 4.31, 4.32, 4.33, 4.34, 4.35, 4.35, 4.36, 4.37, 4.38, 4.39, 4.40, 4.41, 4.42, 4.42, 4.43, 4.44, 4.45, 4.46, 4.47, 4.48, 4.48, 4.49, 4.50, 4.51, 4.52, 4.53, 4.54, 4.55, 4.55, 4.55, 4.56, 4.57, 4.58, 4.59, 4.60, 4.61, 4.62, 4.62, 4.63, 4.64, 4.65, 4.66, 4.67, 4.68, 4.69, 4.69, 4.70, 4.71, 4.72, 4.73, 4.74, 4.75, 4.76, 4.76, 4.77, 4.78, 4.79, 4.80, 4.81, 4.82, 4.83, 4.83, 4.84, 4.85, 4.86, 4.87, 4.88, 4.89, 4.90, 4.91, 4.91, 4.92, 4.93, 4.94, 4.95, 4.96, 4.97, 4.98, 4.99, 4.99, 5.00, 5.01, 5.02, 5.03, 5.04, 5.05, 5.05, 5.06, 5.07, 5.07, 5.08, 5.09, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, 5.24, 5.25, 5.26, 5.27, 5.28, 5.29, 5.30, 5.31, 5.32, 5.33, 5.34, 5.34, 5.35, 5.36, 5.37, 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, 5.43, 5.44, 5.45, 5.46, 5.47, 5.48, 5.49, 5.50, 5.51, 5.52, 5.53, 5.53, 5.54, 5.55, 5.56, 5.57, 5.58, 5.59, 5.60, 5.61, 5.62, 5.63, 5.64, 5.64, 5.65, 5.66, 5.67, 5.68, 5.69, 5.70, 5.71, 5.72, 5.73, 5.74, 5.75, 5.75, 5.76, 5.77, 5.78, 5.79, 5.80, 5.81, 5.82, 5.83, 5.84, 5.85, 5.86, 5.87, 5.88, 5.88, 5.89, 5.90, 5.91, 5.92, 5.93, 5.94, 5.95, 5.96, 5.97, 5.98, 5.98, 5.99, 6.00, 6.01, 6.01, 6.02, 6.03, 6.04, 6.05, 6.06, 6.07, 6.08, 6.09, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.16, 6.17, 6.18, 6.19, 6.20, 6.21, 6.22, 6.23, 6.24, 6.25, 6.26, 6.27, 6.28, 6.29, 6.30, 6.31, 6.32, 6.32, 6.33, 6.34, 6.35, 6.36, 6.37, 6.38, 6.39, 6.40, 6.41, 6.42, 6.43, 6.44, 6.45, 6.46, 6.47, 6.48, 6.49, 6.50, 6.51, 6.51, 6.52, 6.53, 6.54, 6.55, 6.56, 6.57, 6.58, 6.59, 6.60, 6.61, 6.62, 6.63, 6.64, 6.65, 6.66, 6.67, 6.68, 6.69, 6.70, 6.71, 6.72, 6.73, 6.74, 6.75, 6.75, 6.76, 6.77, 6.78, 6.79, 6.80, 6.81, 6.82, 6.83, 6.84, 6.85, 6.86, 6.87, 6.88, 6.89, 6.90, 6.91, 6.92, 6.93, 6.94, 6.95, 6.96, 6.97, 6.98, 6.99, 7.00, 7.01, 7.02, 7.03, 7.04, 7.05, 7.06, 7.07, 7.08, 7.09, 7.10, 7.11, 7.12, 7.12, 7.13, 7.14, 7.15, 7.16, 7.17, 7.18, 7.19, 7.20, 7.21, 7.22, 7.23, 7.24, 7.25, 7.26, 7.27, 7.28, 7.29, 7.30, 7.31, 7.32, 7.33, 7.34, 7.35, 7.36, 7.37, 7.38, 7.39, 7.40, 7.41, 7.42, 7.43, 7.44, 7.45, 7.46, 7.47, 7.48, 7.49, 7.50, 7.51, 7.52, 7.53, 7.54, 7.55, 7.56, 7.57, 7.58, 7.59, 7.60, 7.61, 7.62, 7.63, 7.64, 7.65, 7.66, 7.67, 7.68, 7.69, 7.70, 7.71, 7.72, 7.73, 7.74, 7.75, 7.76, 7.77, 7.78, 7.79, 7.80, 7.81, 7.82, 7.83, 7.84, 7.85, 7.86, 7.87, 7.88, 7.89, 7.91, 7.92, 7.93, 7.94, 7.95, 7.96, 7.97, 7.98, 7.99, 8.00, 8.01, 8.02, 8.03, 8.04, 8.05, 8.06, 8.07, 8.08, 8.09, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 8.23, 8.24, 8.25, 8.26, 8.27, 8.29, 8.30, 8.31, 8.32, 8.33, 8.34, 8.35, 8.36, 8.37, 8.38, 8.39, 8.40, 8.41, 8.42, 8.43, 8.44, 8.45, 8.46, 8.47, 8.48, 8.49, 8.50, 8.51, 8.52, 8.54, 8.55, 8.56, 8.57, 8.58, 8.59, 8.60, 8.61, 8.62, 8.63, 8.64, 8.65, 8.66, 8.67, 8.68, 8.69, 8.70, 8.71, 8.72, 8.74, 8.75, 8.76, 8.77, 8.78, 8.79, 8.80, 8.81, 8.82, 8.83, 8.84, 8.85, 8.86, 8.87, 8.88, 8.89, 8.91, 8.92, 8.93, 8.94, 8.95, 8.96, 8.97, 8.98, 8.99, 9.00, 9.01, 9.02, 9.03, 9.04, 9.06, 9.07, 9.08, 9.09, 9.10, 9.11, 9.12, 9.13, 9.14, 9.15, 9.16, 9.17, 9.18, 9.20, 9.21, 9.22, 9.23, 9.24, 9.25, 9.26, 9.27, 9.28, 9.29, 9.30, 9.31, 9.33, 9.34, 9.35, 9.36, 9.37, 9.38, 9.39, 9.40, 9.41, 9.42, 9.43, 9.45, 9.46, 9.47, 9.48, 9.49, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55, 9.57, 9.58, 9.59, 9.60, 9.61, 9.62, 9.63, 9.64, 9.65, 9.66, 9.68, 9.69, 9.70, 9.71, 9.72, 9.73, 9.74, 9.75, 9.76, 9.78, 9.79, 9.80, 9.81, 9.82, 9.83, 9.84, 9.85, 9.86, 9.88, 9.89, 9.90, 9.91, 9.92, 9.93, 9.94, 9.95, 9.96, 9.98, 9.99, 10.00, 10.01, 10.02, 10.03, 10.04, 10.05, 10.07, 10.08, 10.09, 10.10, 10.11, 10.12, 10.13, 10.14, 10.16, 10.17, 10.18, 10.19, 10.20, 10.21, 10.22, 10.24, 10.25, 10.26, 10.27, 10.28, 10.29, 10.30, 10.31, 10.33, 10.34, 10.35, 10.36, 10.37, 10.38, 10.39, 10.41, 10.42, 10.43, 10.44, 10.45, 10.46, 10.47, 10.49, 10.50, 10.51, 10.52, 10.53, 10.54, 10.55, 10.57, 10.58, 10.59, 10.60, 10.61, 10.62, 10.64, 10.65, 10.66, 10.67, 10.68, 10.69, 10.70, 10.72, 10.73, 10.74, 10.75, 10.76, 10.77, 10.79, 10.80, 10.81, 10.82, 10.83, 10.84, 10.86, 10.87, 10.88, 10.89, 10.90, 10.91, 10.93, 10.94, 10.95, 10.96, 10.97, 10.98, 11.00, 11.01, 11.02, 11.03, 11.04, 11.05, 11.07, 11.08, 11.09, 11.10, 11.11, 11.13, 11.14, 11.15, 11.16, 11.17, 11.18, 11.20, 11.21, 11.22, 11.23, 11.24, 11.26, 11.27, 11.28, 11.29, 11.30, 11.32, 11.33, 11.34, 11.35, 11.36, 11.37, 11.39, 11.40, 11.41, 11.42, 11.43, 11.45, 11.46, 11.47, 11.48, 11.49, 11.51, 11.52, 11.53, 11.54, 11.55, 11.57, 11.58, 11.59, 11.60, 11.61, 11.63, 11.64

64, 11.65, 11.66, 11.68, 11.69, 11.70, 11.71, 11.72, 11.74, 11.75, 11.76, 11.77, 11.78, 11.80, 11.81, 11.82, 11.83, 11.85, 11.86, 11.87, 11.88, 11.89, 11.91, 11.92, 11.93, 11.94, 11.96, 11.97, 11.98, 11.99, 12.00, 12.02, 12.03, 12.04, 12.05, 12.07, 12.08, 12.09, 12.10, 12.11, 12.13, 12.14, 12.15, 12.16, 12.18, 12.19, 12.20, 12.21, 12.23, 12.24, 12.25, 12.26, 12.28, 12.29, 12.30, 12.31, 12.33, 12.34, 12.35, 12.36, 12.38, 12.39, 12.40, 12.41, 12.43, 12.44, 12.45, 12.46, 12.48, 12.49, 12.50, 12.51, 12.53, 12.54, 12.55, 12.56, 12.58, 12.59, 12.60, 12.61, 12.63, 12.64, 12.65, 12.66, 12.68, 12.69, 12.70, 12.72, 12.73, 12.74, 12.75, 12.77, 12.78, 12.79, 12.80, 12.82, 12.83, 12.84, 12.86, 12.87, 12.88, 12.89, 12.91, 12.92, 12.93, 12.94, 12.96, 12.97, 12.98, 13.00, 13.01, 13.02, 13.03, 13.05, 13.06, 13.07, 13.09, 13.10, 13.11, 13.12, 13.14, 13.15, 13.16, 13.18, 13.19, 13.20, 13.22, 13.23, 13.24, 13.25, 13.27, 13.28, 13.29, 13.31, 13.32, 13.33, 13.35, 13.36, 13.37, 13.38, 13.40, 13.41, 13.42, 13.44, 13.45, 13.46, 13.48, 13.49, 13.50, 13.52, 13.53, 13.54, 13.55, 13.57, 13.58, 13.59, 13.61, 13.62, 13.63, 13.65, 13.66, 13.67, 13.69, 13.70, 13.71, 13.73, 13.74, 13.75, 13.77, 13.78, 13.79, 13.81, 13.82, 13.83, 13.85, 13.86, 13.87, 13.89, 13.90, 13.91, 13.93, 13.94, 13.95, 13.97, 13.98, 13.99, 14.01, 14.02, 14.03, 14.05, 14.06, 14.07, 14.09, 14.10, 14.11, 14.13, 14.14, 14.16, 14.17, 14.18, 14.20, 14.21, 14.22, 14.24, 14.25, 14.26, 14.28, 14.29, 14.30, 14.32, 14.33, 14.35, 14.36, 14.37, 14.39, 14.40, 14.41, 14.43, 14.44, 14.45, 14.47, 14.48, 14.50, 14.51, 14.52, 14.54, 14.55, 14.56, 14.58, 14.59, 14.61, 14.62, 14.63, 14.65, 14.66, 14.67, 14.69, 14.70, 14.72, 14.73, 14.74, 14.76, 14.77, 14.79, 14.80, 14.81, 14.83, 14.84, 14.86, 14.87, 14.88, 14.90, 14.91, 14.93, 14.94, 14.95, 14.97, 14.98, 15.00, 15.01, 15.02, 15.04, 15.05, 15.07, 15.08, 15.09, 15.11, 15.12, 15.14, 15.15, 15.16, 15.18, 15.19, 15.21, 15.22, 15.24, 15.25, 15.26, 15.28, 15.29, 15.31, 15.32, 15.33, 15.35, 15.36, 15.38, 15.39, 15.41, 15.42, 15.43, 15.45, 15.46, 15.48, 15.49, 15.51, 15.52, 15.54, 15.55, 15.56, 15.58, 15.59, 15.61, 15.62, 15.64, 15.65, 15.66, 15.68, 15.69, 15.71, 15.72, 15.74, 15.75, 15.77, 15.78, 15.80, 15.81, 15.82, 15.84, 15.85, 15.87, 15.88, 15.90, 15.91, 15.93, 15.94, 15.96, 15.97, 15.99, 16.00, 16.01, 16.03, 16.04, 16.06, 16.07, 16.09, 16.10, 16.12, 16.13, 16.15, 16.16, 16.18, 16.19, 16.21, 16.22, 16.24, 16.25, 16.27, 16.28, 16.30, 16.31, 16.33, 16.34, 16.35, 16.37, 16.38, 16.40, 16.41, 16.43, 16.44, 16.46, 16.47, 16.49, 16.50, 16.52, 16.53, 16.55, 16.56, 16.58, 16.59, 16.61, 16.62, 16.64, 16.66, 16.67, 16.69, 16.70, 16.72, 16.73, 16.75, 16.76, 16.78, 16.79, 16.81, 16.82, 16.84, 16.85, 16.87, 16.88, 16.90, 16.91, 16.93, 16.94, 16.96, 16.97, 16.99, 17.01, 17.02, 17.04, 17.05, 17.07, 17.08, 17.10, 17.11, 17.13, 17.14, 17.16, 17.17, 17.19, 17.21, 17.22, 17.24, 17.25, 17.27, 17.28, 17.30, 17.31, 17.33, 17.35, 17.36, 17.38, 17.39, 17.41, 17.42, 17.44, 17.45, 17.47, 17.49, 17.50, 17.52, 17.53, 17.55, 17.56, 17.58, 17.60, 17.61, 17.63, 17.64, 17.66, 17.67, 17.69, 17.71, 17.72, 17.74, 17.75, 17.77, 17.79, 17.80, 17.82, 17.83, 17.85, 17.86, 17.88, 17.90, 17.91, 17.93, 17.94, 17.96, 17.98, 17.99, 18.01, 18.02, 18.04, 18.06, 18.07, 18.09, 18.11, 18.12, 18.14, 18.15, 18.17, 18.19, 18.20, 18.22, 18.23, 18.25, 18.27, 18.28, 18.30, 18.32, 18.33, 18.35, 18.36, 18.38, 18.40, 18.41, 18.43, 18.45, 18.46, 18.48, 18.49, 18.51, 18.53, 18.54, 18.56, 18.58, 18.59, 18.61, 18.63, 18.64, 18.66, 18.68, 18.69, 18.71, 18.73, 18.74, 18.76, 18.77, 18.79, 18.81, 18.82, 18.84, 18.86, 18.87, 18.89, 18.91, 18.92, 18.94, 18.96, 18.97, 18.99, 19.01, 19.02, 19.04, 19.06, 19.08, 19.09, 19.11, 19.13, 19.14, 19.16, 19.18, 19.19, 19.21, 19.23, 19.24, 19.26, 19.28, 19.29, 19.31, 19.33, 19.35, 19.36, 19.38, 19.40, 19.41, 19.43, 19.45, 19.46, 19.48, 19.50, 19.52, 19.53, 19.55, 19.57, 19.58, 19.60, 19.62, 19.64, 19.65, 19.67, 19.69, 19.70, 19.72, 19.74, 19.76, 19.77, 19.79, 19.81, 19.83, 19.84, 19.86, 19.88, 19.90, 19.91, 19.93, 19.95, 19.97, 19.98, 20.00, 20.02, 20.04, 20.05, 20.07, 20.09, 20.11, 20.12, 20.14, 20.16, 20.18, 20.19, 20.21, 20.23, 20.25, 20.26, 20.28, 20.30, 20.32, 20.33, 20.35, 20.37, 20.39, 20.41, 20.42, 20.44, 20.46, 20.48, 20.49, 20.51, 20.53, 20.55, 20.57, 20.58, 20.60, 20.62, 20.64, 20.66, 20.67, 20.69, 20.71, 20.73, 20.75, 20.76, 20.78, 20.80, 20.82, 20.84, 20.85, 20.87, 20.89, 20.91, 20.93, 20.95, 20.96, 20.98, 21.00, 21.02, 21.04, 21.06, 21.07, 21.09, 21.11, 21.13, 21.15, 21.17, 21.18, 21.20, 21.22, 21.24, 21.26, 21.28, 21.29, 21.31, 21.33, 21.35, 21.37, 21.39, 21.41, 21.42, 21.44, 21.46, 21.48, 21.50, 21.52, 21.54, 21.55, 21.57, 21.59, 21.61, 21.63, 21.65, 21.67, 21.69, 21.70, 21.72, 21.74, 21.76, 21.78, 21.80, 21.82, 21.84, 21.86, 21.87, 21.89, 21.91, 21.93, 21.95, 21.97, 21.99, 22.01, 22.03, 22.05, 22.06, 22.08, 22.10, 22.12, 22.14, 22.16, 22.18, 22.20, 22.22, 22.24, 22.26, 22.28, 22.30, 22.31, 22.33, 22.35, 22.37, 22.39, 22.41, 22.43, 22.45, 22.47, 22.49, 22.51, 22.53, 22.55, 22.57, 22.59, 22.61, 22.63, 22.64, 22.66, 22.68, 22.70, 22.72, 22.74, 22.76, 22.78, 22.80, 22.82, 22.84, 22.86, 22.88, 22.90, 22.92, 22.94, 22.96, 22.98, 23.00, 23.02, 23.04, 23.06, 23.08, 23.10, 23.12, 23.14, 23.16, 23.18, 23.20, 23.22, 23.24, 23.26, 23.28, 23.30, 23.32, 23.34, 23.36, 23.38, 23.40, 23.42, 23.44, 23.46, 23.48, 23.50, 23.52, 23.54, 23.56, 23.58, 23.60, 23.62, 23.65, 23.67, 23.69, 23.71, 23.73, 23.75, 23.77, 23.79, 23.81, 23.83, 23.85, 23.87, 23.89, 23.91, 23.93, 23.95, 23.97, 24.00, 24.02, 24.04, 24.06, 24.08, 24.10, 24.12, 24.14, 24.16, 24.18, 24.20, 24.22, 24.25, 24.27, 24.29, 24.31, 24.33, 24.35, 24.37, 24.39, 24.41, 24.43, 24.46, 24.48, 24.50, 24.52, 24.54, 24.56, 24.58, 24.60, 24.63, 24.65, 24.67, 24.69, 24.71, 24.73, 24.75, 24.78, 24.80, 24.82, 24.84, 24.86, 24.88, 24.90, 24.93, 24.95, 24.97, 24.99, 25.01, 25.03, 25.06, 25.08, 25.10, 25.12, 25.14, 25.16, 25.19, 25.21, 25.23, 25.25, 25.27, 25.30, 25.32, 25.34, 25.36, 25.38, 25.41, 25.43, 25.45, 25.47, 25.49, 25.52, 25.54, 25.56, 25.58, 25.60, 25.63, 25.65, 25.67, 25.69, 25.72, 25.74, 25.76, 25.78, 25.81, 25.83, 25.85, 25.87, 25.89, 25.92, 25.94, 25.96, 25.98, 26.01, 26.03, 26.05, 26.08, 26.10, 26.12, 26.14, 26.17, 26.19, 26.21, 26.23, 26.26, 26.28, 26.30, 26.33, 26.35, 26.37, 26.39, 26.42

42, 26.44, 26.46, 26.49, 26.51, 26.53, 26.56, 26.58, 26.60, 26.63, 26.65, 26.67, 26.69, 26.72, 26.74, 26.76, 26.79, 26.81, 26.83, 26.86, 26.88, 26.90, 26.93, 26.95, 26.98, 27.00, 27.02, 27.05, 27.07, 27.09, 27.12, 27.14, 27.16, 27.19, 27.21, 27.24, 27.26, 27.28, 27.31, 27.33, 27.35, 27.38, 27.40, 27.43, 27.45, 27.47, 27.50, 27.52, 27.55, 27.57, 27.59, 27.62, 27.64, 27.67, 27.69, 27.72, 27.74, 27.76, 27.79, 27.81, 27.84, 27.86, 27.89, 27.91, 27.93, 27.96, 27.98, 28.01, 28.03, 28.06, 28.08, 28.11, 28.13, 28.16, 28.18, 28.21, 28.23, 28.26, 28.28, 28.30, 28.33, 28.35, 28.38, 28.40, 28.43, 28.45, 28.48, 28.50, 28.53, 28.55, 28.58, 28.60, 28.63, 28.66, 28.68, 28.71, 28.73, 28.76, 28.78, 28.81, 28.83, 28.86, 28.88, 28.91, 28.93, 28.96, 28.99, 29.01, 29.04, 29.06, 29.09, 29.11, 29.14, 29.17, 29.19, 29.22, 29.24, 29.27, 29.29, 29.32, 29.35, 29.37, 29.40, 29.42, 29.45, 29.48, 29.50, 29.53, 29.55, 29.58, 29.61, 29.63, 29.66, 29.69, 29.71, 29.74, 29.76, 29.79, 29.82, 29.84, 29.87, 29.90, 29.92, 29.95, 29.98, 30.00, 30.03, 30.06, 30.08, 30.11, 30.14, 30.16, 30.19, 30.22, 30.24, 30.27, 30.30, 30.33, 30.35, 30.38, 30.41, 30.43, 30.46, 30.49, 30.52, 30.54, 30.57, 30.60, 30.62, 30.65, 30.68, 30.71, 30.73, 30.76, 30.79, 30.82, 30.84, 30.87, 30.90, 30.93, 30.96, 30.98, 31.01, 31.04, 31.07, 31.09, 31.12, 31.15, 31.18, 31.21, 31.23, 31.26, 31.29, 31.32, 31.35, 31.37, 31.40, 31.43, 31.46, 31.49, 31.52, 31.54, 31.57, 31.60, 31.63, 31.66, 31.69, 31.72, 31.74, 31.77, 31.80, 31.83, 31.86, 31.89, 31.92, 31.95, 31.97, 32.00, 32.03, 32.06, 32.09, 32.12, 32.15, 32.18, 32.21, 32.24, 32.27, 32.29, 32.32, 32.35, 32.38, 32.41, 32.44, 32.47, 32.50, 32.53, 32.56, 32.59, 32.62, 32.65, 32.68, 32.71, 32.74, 32.77, 32.80, 32.83, 32.86, 32.89, 32.92, 32.95, 32.98, 33.01, 33.04, 33.07, 33.10, 33.13, 33.16, 33.19, 33.22, 33.25, 33.28, 33.31, 33.34, 33.37, 33.40, 33.43, 33.46, 33.49, 33.53, 33.56, 33.59, 33.62, 33.65, 33.68, 33.71, 33.74, 33.77, 33.80, 33.84, 33.87, 33.90, 33.93, 33.96, 33.99, 34.02, 34.05, 34.09, 34.12, 34.15, 34.18, 34.21, 34.24, 34.28, 34.31, 34.34, 34.37, 34.40, 34.43, 34.47, 34.50, 34.53, 34.56, 34.59, 34.63, 34.66, 34.69, 34.72, 34.76, 34.79, 34.82, 34.85, 34.89, 34.92, 34.95, 34.98, 35.02, 35.05, 35.08, 35.11, 35.15, 35.18, 35.21, 35.25, 35.28, 35.31, 35.35, 35.38, 35.41, 35.44, 35.48, 35.51, 35.54, 35.58, 35.61, 35.65, 35.68, 35.71, 35.75, 35.78, 35.81, 35.85, 35.88, 35.91, 35.95, 35.98, 36.02, 36.05, 36.08, 36.12, 36.15, 36.19, 36.22, 36.26, 36.29, 36.33, 36.36, 36.39, 36.43, 36.46, 36.50, 36.53, 36.57, 36.60, 36.64, 36.67, 36.71, 36.74, 36.78, 36.81, 36.85, 36.88, 36.92, 36.95, 36.99, 37.02, 37.06, 37.09, 37.13, 37.17, 37.20, 37.24, 37.27, 37.31, 37.34, 37.38, 37.42, 37.45, 37.49, 37.52, 37.56, 37.60, 37.63, 37.67, 37.71, 37.74, 37.78, 37.82, 37.85, 37.89, 37.93, 37.96, 38.00, 38.04, 38.07, 38.11, 38.15, 38.18, 38.22, 38.26, 38.30, 38.33, 38.37, 38.41, 38.44, 38.48, 38.52, 38.56, 38.60, 38.63, 38.67, 38.71, 38.75, 38.78, 38.82, 38.86, 38.90, 38.94, 38.97, 39.01, 39.05, 39.09, 39.13, 39.17, 39.21, 39.24, 39.28, 39.32, 39.36, 39.40, 39.44, 39.48, 39.52, 39.56, 39.59, 39.63, 39.67, 39.71, 39.75, 39.79, 39.83, 39.87, 39.91, 39.95, 39.99, 40.03, 40.07, 40.11, 40.15, 40.19, 40.23, 40.27, 40.31, 40.35, 40.39, 40.43, 40.47, 40.51, 40.55, 40.59, 40.64, 40.68, 40.72, 40.76, 40.80, 40.84, 40.88, 40.92, 40.96, 41.01, 41.05, 41.09, 41.13, 41.17, 41.21, 41.26, 41.30, 41.34, 41.38, 41.42, 41.47, 41.51, 41.55, 41.59, 41.64, 41.68, 41.72, 41.76, 41.81, 41.85, 41.89, 41.93, 41.98, 42.02, 42.06, 42.11, 42.15, 42.19, 42.24, 42.28, 42.32, 42.37, 42.41, 42.46, 42.50, 42.54, 42.59, 42.63, 42.68, 42.72, 42.76, 42.81, 42.85, 42.90, 42.94, 42.99, 43.03, 43.08, 43.12, 43.17, 43.21, 43.26, 43.30, 43.35, 43.39, 43.44, 43.48, 43.53, 43.58, 43.62, 43.67, 43.71, 43.76, 43.81, 43.85, 43.90, 43.94, 43.99, 44.04, 44.08, 44.13, 44.18, 44.23, 44.27, 44.32, 44.37, 44.41, 44.46, 44.51, 44.56, 44.60, 44.65, 44.70, 44.75, 44.80, 44.84, 44.89, 44.94, 44.99, 45.04, 45.08, 45.13, 45.18, 45.23, 45.28, 45.33, 45.38, 45.43, 45.48, 45.53, 45.57, 45.62, 45.67, 45.72, 45.77, 45.82, 45.87, 45.92, 45.97, 46.02, 46.07, 46.12, 46.17, 46.23, 46.28, 46.33, 46.38, 46.43, 46.48, 46.53, 46.58, 46.63, 46.69, 46.74, 46.79, 46.84, 46.89, 46.95, 47.00, 47.05, 47.10, 47.15, 47.21, 47.26, 47.31, 47.37, 47.42, 47.47, 47.52, 47.58, 47.63, 47.69, 47.74, 47.79, 47.85, 47.90, 47.95, 48.01, 48.06, 48.12, 48.17, 48.23, 48.28, 48.34, 48.39, 48.45, 48.50, 48.56, 48.61, 48.67, 48.72, 48.78, 48.84, 48.89, 48.95, 49.00, 49.06, 49.12, 49.17, 49.23, 49.29, 49.34, 49.40, 49.46, 49.52, 49.57, 49.63, 49.69, 49.75, 49.80, 49.86, 49.92, 49.98, 50.04, 50.10, 50.16, 50.21, 50.27, 50.33, 50.39, 50.45, 50.51, 50.57, 50.63, 50.69, 50.75, 50.81, 50.87, 50.93, 50.99, 51.05, 51.11, 51.17, 51.24, 51.30, 51.36, 51.42, 51.48, 51.54, 51.61, 51.67, 51.73, 51.79, 51.86, 51.92, 51.98, 52.04, 52.11, 52.17, 52.23, 52.30, 52.36, 52.43, 52.49, 52.55, 52.62, 52.68, 52.75, 52.81, 52.88, 52.94, 53.01, 53.07, 53.14, 53.21, 53.27, 53.34, 53.40, 53.47, 53.54, 53.60, 53.67, 53.74, 53.80, 53.87, 53.94, 54.01, 54.08, 54.14, 54.21, 54.28, 54.35, 54.42, 54.49, 54.56, 54.63, 54.70, 54.76, 54.83, 54.90, 54.98, 55.05, 55.12, 55.19, 55.26, 55.33, 55.40, 55.47, 55.54, 55.62, 55.69, 55.76, 55.83, 55.91, 55.98, 56.05, 56.12, 56.20, 56.27, 56.35, 56.42, 56.49, 56.57, 56.64, 56.72, 56.79, 56.87, 56.94, 57.02, 57.10, 57.17, 57.25, 57.32, 57.40, 57.48, 57.56, 57.63, 57.71, 57.79, 57.87, 57.94, 58.02, 58.10, 58.18, 58.26, 58.34, 58.42, 58.50, 58.58, 58.66, 58.74, 58.82, 58.90, 58.98, 59.06, 59.15, 59.23, 59.31, 59.39, 59.48, 59.56, 59.64, 59.72, 59.81, 59.89, 59.98, 60.06, 60.15, 60.23, 60.32, 60.40, 60.49, 60.57, 60.66, 60.75, 60.83, 60.92, 61.01, 61.10, 61.18, 61.27, 61.36, 61.45, 61.54, 61.63, 61.72, 61.81, 61.90, 61.99, 62.08, 62.17, 62.26, 62.35, 62.44, 62.54, 62.63, 62.72, 62.82, 62.91, 63.00, 63.10, 63.19, 63.29, 63.38, 63.48, 63.57, 63.67, 63.76, 63.86, 63.96, 64.06, 64.15, 64.25, 64.35, 64.45, 64.55, 64.65, 64.75, 64.85, 64.95, 65.05, 65.15, 65.25, 65.35, 65.46, 65.56, 65.66, 65.76, 65.87, 65.97, 66.08, 66.18, 66.29, 66.39, 66.50, 66.61, 66.71, 66.82, 66.93, 67.03, 67.14, 67.25, 67.36, 67.47, 67.58, 67.69, 67.80, 67.91, 68.03, 68.14, 68.25, 68.36, 68.48, 68.59, 68.71, 68.82, 68.94, 69.05, 69.17, 69.29, 69.30

- volatile uint32\_t rawADC\_left [4] = {0}

### *Raw ADC data for the left inverter.*

- volatile uint32\_t rawADC\_right [4] = {0}

*Raw ADC data for the right inverter.*

- volatile uint32\_t rawADC\_temp [4] = {0}

### *Raw ADC data for the temperatures.*

### 6.37.1 Detailed Description

This file provides functions for handling measurements.

#### Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

### 6.37.2 Function Documentation

#### 6.37.2.1 get\_currents\_voltage()

```
uint8_t get_currents_voltage (
 volatile uint32_t ADC_raw[],
 volatile Analog * analog,
 volatile Feedback * feedback,
 float sinTheta_e,
 float cosTheta_e)
```

Get electrical ADC measurements.

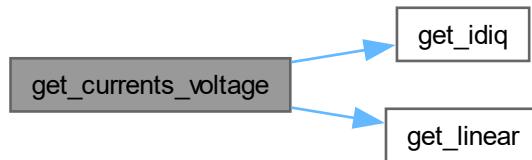
##### Parameters

|     |                   |                                                           |
|-----|-------------------|-----------------------------------------------------------|
| in  | <i>ADC_raw</i>    | Pointer to the raw ADC values array.                      |
| out | <i>analog</i>     | Pointer to the ADC struct to store the results.           |
| out | <i>feedback</i>   | Pointer to the <i>Feedback</i> struct to store id and iq. |
| in  | <i>sinTheta_e</i> | Electrical angle sine (-1..1)                             |
| in  | <i>cosTheta_e</i> | Electrical angle cosine (-1..1)                           |

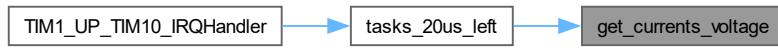
##### Return values

|           |                                          |
|-----------|------------------------------------------|
| <i>OK</i> | 0 if an error occurred, 1 if successful. |
|-----------|------------------------------------------|

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.37.2.2 `get_idiq()`

```

void get_idiq (
 float ia,
 float ib,
 float ic,
 float sinTheta_e,
 float cosTheta_e,
 float * idMeas,
 float * iqMeas)

```

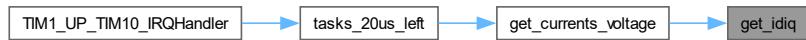
Computes d-q currents from current measurements and electrical angle.

This function computes the d-q currents from phase currents (ABC), theta<sub>e</sub>, and stores the results in the provided pointers.

#### Parameters

|     |                   |                                      |
|-----|-------------------|--------------------------------------|
| in  | <i>ia</i>         | Phase A current in A.                |
| in  | <i>ib</i>         | Phase B current in A.                |
| in  | <i>ic</i>         | Phase C current in A.                |
| in  | <i>sinTheta_e</i> | Electrical angle sine (-1..1)        |
| in  | <i>cosTheta_e</i> | Electrical angle cosine (-1..1)      |
| out | <i>idMeas</i>     | Pointer to store the D-axis current. |
| out | <i>iqMeas</i>     | Pointer to store the Q-axis current. |

Here is the caller graph for this function:



### 6.37.2.3 get\_linear()

```

float get_linear (
 uint32_t bits,
 float slope,
 float offset)

```

Convert ADC reading to physical measurement with linear response.

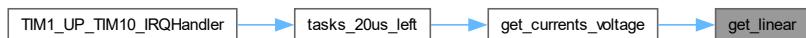
#### Parameters

|    |               |                             |
|----|---------------|-----------------------------|
| in | <i>bits</i>   | The ADC reading.            |
| in | <i>slope</i>  | The slope (units per volt). |
| in | <i>offset</i> | The offset (volts at zero). |

#### Return values

|                    |                           |
|--------------------|---------------------------|
| <i>measurement</i> | The physical measurement. |
|--------------------|---------------------------|

Here is the caller graph for this function:



### 6.37.2.4 get\_temperature()

```

float get_temperature (
 uint32_t bits,
 const float tempLUT[])

```

Retrieves temperature from a lookup table based on ADC bits.

This function retrieves temperature from a lookup table based on the ADC bits. The lookup table (LUT) must have a value for each possible ADC bit combination.

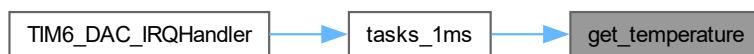
**Parameters**

|    |                |                                             |
|----|----------------|---------------------------------------------|
| in | <i>bits</i>    | ADC reading converted to bits.              |
| in | <i>tempLUT</i> | Lookup table containing temperature values. |

**Returns**

Temperature corresponding to the provided ADC bits.

Here is the caller graph for this function:



### 6.37.3 Variable Documentation

#### 6.37.3.1 rawADC\_left

```
volatile uint32_t rawADC_left[4] = {0}
```

Raw ADC data for the left inverter.

External declaration of raw ADC data for the left inverter.

#### 6.37.3.2 rawADC\_right

```
volatile uint32_t rawADC_right[4] = {0}
```

Raw ADC data for the right inverter.

External declaration of raw ADC data for the right inverter.

#### 6.37.3.3 rawADC\_temp

```
volatile uint32_t rawADC_temp[4] = {0}
```

Raw ADC data for the temperatures.

External declaration of raw ADC data for the temperature readings.

#### 6.37.3.4 templInverterLUT

19, 3.20, 3.20, 3.21, 3.22, 3.23, 3.24, 3.24, 3.25, 3.26, 3.27, 3.28, 3.29, 3.29, 3.30, 3.31, 3.32, 3.33, 3.34, 3.34, 3.35, 3.36, 3.37, 3.38, 3.38, 3.39, 3.40, 3.41, 3.42, 3.43, 3.43, 3.43, 3.44, 3.45, 3.46, 3.47, 3.48, 3.48, 3.49, 3.50, 3.51, 3.52, 3.53, 3.53, 3.54, 3.55, 3.55, 3.56, 3.57, 3.58, 3.58, 3.59, 3.60, 3.61, 3.62, 3.63, 3.63, 3.64, 3.65, 3.66, 3.67, 3.68, 3.68, 3.69, 3.69, 3.70, 3.71, 3.72, 3.73, 3.73, 3.74, 3.75, 3.76, 3.77, 3.78, 3.78, 3.79, 3.80, 3.81, 3.82, 3.83, 3.83, 3.84, 3.85, 3.86, 3.87, 3.88, 3.89, 3.89, 3.90, 3.91, 3.92, 3.93, 3.94, 3.94, 3.95, 3.95, 3.96, 3.97, 3.98, 3.99, 4.00, 4.00, 4.01, 4.02, 4.03, 4.04, 4.05, 4.05, 4.06, 4.07, 4.08, 4.09, 4.10, 4.11, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.22, 4.23, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29, 4.29, 4.30, 4.31, 4.32, 4.33, 4.34, 4.35, 4.35, 4.36, 4.37, 4.38, 4.39, 4.40, 4.41, 4.42, 4.42, 4.43, 4.44, 4.45, 4.46, 4.47, 4.48, 4.48, 4.48, 4.49, 4.50, 4.51, 4.52, 4.53, 4.54, 4.55, 4.55, 4.56, 4.57, 4.58, 4.59, 4.60, 4.61, 4.62, 4.62, 4.63, 4.64, 4.65, 4.66, 4.67, 4.68, 4.69, 4.69, 4.70, 4.71, 4.72, 4.73, 4.74, 4.74, 4.75, 4.75, 4.76, 4.76, 4.77, 4.77, 4.78, 4.79, 4.80, 4.81, 4.82, 4.83, 4.83, 4.84, 4.85, 4.86, 4.86, 4.87, 4.88, 4.89, 4.89, 4.90, 4.91, 4.91, 4.92, 4.93, 4.94, 4.95, 4.96, 4.97, 4.98, 4.99, 4.99, 5.00, 5.01, 5.02, 5.03, 5.03, 5.04, 5.05, 5.06, 5.07, 5.07, 5.08, 5.09, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, 5.24, 5.25, 5.26, 5.27, 5.28, 5.29, 5.30, 5.31, 5.31, 5.32, 5.33, 5.34, 5.34, 5.35, 5.36, 5.37, 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, 5.43, 5.44, 5.45, 5.46, 5.47, 5.48, 5.49, 5.50, 5.51, 5.52, 5.53, 5.53, 5.54, 5.55, 5.56, 5.57, 5.58, 5.59, 5.59, 5.60, 5.61, 5.62, 5.63, 5.64, 5.64, 5.65, 5.66, 5.67, 5.68, 5.69, 5.70, 5.71, 5.72, 5.73, 5.74, 5.75, 5.75, 5.76, 5.77, 5.78, 5.79, 5.80, 5.81, 5.82, 5.83, 5.84, 5.85, 5.86, 5.87, 5.88, 5.88, 5.88, 5.89, 5.89, 5.90, 5.91, 5.92, 5.93, 5.94, 5.95, 5.96, 5.97, 5.98, 5.99, 5.99, 6.00, 6.01, 6.01, 6.02, 6.03, 6.04, 6.05, 6.06, 6.07, 6.08, 6.09, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.16, 6.16, 6.17, 6.18, 6.19, 6.20, 6.21, 6.22, 6.23, 6.24, 6.25, 6.26, 6.27, 6.28, 6.29, 6.30, 6.31, 6.32, 6.32, 6.33, 6.34, 6.35, 6.36, 6.37, 6.38, 6.39, 6.40, 6.41, 6.42, 6.43, 6.44, 6.45, 6.46, 6.46, 6.47, 6.48, 6.49, 6.50, 6.51, 6.51, 6.52, 6.53, 6.54, 6.55, 6.56, 6.57, 6.58, 6.59, 6.60, 6.61, 6.62, 6.63, 6.64, 6.65, 6.66, 6.67, 6.68, 6.69, 6.70, 6.71, 6.72, 6.73, 6.74, 6.75, 6.75, 6.76, 6.77, 6.78, 6.79, 6.80, 6.81, 6.82, 6.83, 6.84, 6.85, 6.86, 6.87, 6.88, 6.89, 6.89, 6.90, 6.91, 6.92, 6.93, 6.94, 6.95, 6.96, 6.97, 6.98, 6.99, 7.00, 7.01, 7.02, 7.03, 7.04, 7.05, 7.06, 7.06, 7.07, 7.08, 7.09, 7.10, 7.11, 7.12, 7.12, 7.13, 7.14, 7.15, 7.16, 7.17, 7.18, 7.19, 7.20, 7.21, 7.22, 7.23, 7.24, 7.25, 7.26, 7.27, 7.28, 7.29, 7.30, 7.31, 7.32, 7.33, 7.34, 7.35, 7.36, 7.36, 7.37, 7.38, 7.39, 7.40, 7.41, 7.42, 7.43, 7.44, 7.45, 7.46, 7.47, 7.48, 7.49, 7.50, 7.51, 7.52, 7.53, 7.54, 7.55, 7.56, 7.57, 7.58, 7.59, 7.60, 7.61, 7.62, 7.63, 7.64, 7.65, 7.66, 7.67, 7.67, 7.68, 7.69, 7.70, 7.71, 7.72, 7.73, 7.74, 7.75, 7.76, 7.77, 7.78, 7.79, 7.80, 7.81, 7.82, 7.83, 7.84, 7.85, 7.86, 7.87, 7.88, 7.89, 7.91, 7.92, 7.93, 7.94, 7.95, 7.96, 7.97, 7.98, 7.99, 8.00, 8.01, 8.02, 8.03, 8.04, 8.05, 8.06, 8.07, 8.08, 8.09, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16, 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 8.23, 8.24, 8.25, 8.26, 8.27, 8.29, 8.30, 8.31, 8.31, 8.32, 8.33, 8.34, 8.35, 8.36, 8.37, 8.38, 8.39, 8.40, 8.41, 8.42, 8.43, 8.44, 8.45, 8.46, 8.47, 8.48, 8.49, 8.50, 8.51, 8.52, 8.54, 8.55, 8.56, 8.57, 8.58, 8.59, 8.60, 8.61, 8.62, 8.63, 8.63, 8.64, 8.65, 8.66, 8.67, 8.68, 8.69, 8.70, 8.71, 8.72, 8.74, 8.75, 8.76, 8.77, 8.78, 8.79, 8.80, 8.81, 8.82, 8.83, 8.84, 8.85, 8.86, 8.87, 8.88, 8.89, 8.91, 8.92, 8.93, 8.94, 8.95, 8.96, 8.96, 8.97, 8.98, 8.99, 9.00, 9.01, 9.02, 9.03, 9.04, 9.06, 9.07, 9.08, 9.09, 9.10, 9.11, 9.12, 9.13, 9.14, 9.15, 9.16, 9.17, 9.18, 9.20, 9.21, 9.22, 9.23, 9.24, 9.25, 9.26, 9.27, 9.28, 9.29, 9.29, 9.30, 9.31, 9.33, 9.34, 9.35, 9.36, 9.37, 9.38, 9.39, 9.40, 9.41, 9.42, 9.43, 9.45, 9.46, 9.47, 9.48, 9.49, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55, 9.57, 9.58, 9.59, 9.60, 9.61, 9.62, 9.63, 9.63, 9.64, 9.65, 9.66, 9.68, 9.69, 9.70, 9.71, 9.72, 9.73, 9.74, 9.75, 9.76, 9.78, 9.79, 9.80, 9.81, 9.82, 9.83, 9.84, 9.85, 9.86, 9.88, 9.89, 9.90, 9.91, 9.92, 9.93, 9.94, 9.95, 9.96, 9.98, 9.98, 9.99, 10.00, 10.01, 10.02, 10.03, 10.04, 10.05, 10.07, 10.08, 10.09, 10.10, 10.11, 10.12, 10.12, 10.13, 10.14, 10.16, 10.17, 10.18, 10.19, 10.20, 10.21, 10.22, 10.24, 10.25, 10.26, 10.27, 10.27, 10.28, 10.29, 10.30, 10.31, 10.33, 10.34, 10.35, 10.36, 10.37, 10.38, 10.39, 10.41, 10.42, 10.42, 10.43, 10.44, 10.45, 10.46, 10.47, 10.49, 10.50, 10.51, 10.52, 10.53, 10.54, 10.55, 10.57, 10.57, 10.58, 10.60, 10.61, 10.62, 10.64, 10.65, 10.66, 10.67, 10.68, 10.69, 10.70, 10.72, 10.72, 10.73, 10.74, 10.75, 10.76, 10.77, 10.79, 10.80, 10.81, 10.82, 10.83, 10.84, 10.86, 10.87, 10.87, 10.88, 10.89, 10.90, 10.91, 10.93, 10.94, 10.95, 10.96, 10.97, 10.98, 11.00, 11.01, 11.02, 11.02, 11.03, 11.04, 11.05, 11.07, 11.08, 11.09, 11.10, 11.11, 11.13, 11.14, 11.15, 11.16, 11.17, 11.17, 11.18, 11.20, 11.21, 11.22, 11.23, 11.24, 11.26, 11.27, 11.28, 11.29, 11.30, 11.32, 11.33, 11.33, 11.34, 11.35, 11.36, 11.37, 11.39, 11.40, 11.41, 11.42, 11.43, 11.45, 11.46, 11.47, 11.48, 11.48, 11.49, 11.51, 11.52, 11.53, 11.54, 11.55, 11.57, 11.58, 11.59, 11.60, 11.61, 11.63, 11.64, 11.64, 11.65, 11.66, 11.68, 11.69, 11.70, 11.71, 11.72, 11.74, 11.75, 11.76, 11.77, 11.78, 11.80, 11.80, 11.81

81, 11.82, 11.83, 11.85, 11.86, 11.87, 11.88, 11.89, 11.91, 11.92, 11.93, 11.94, 11.96, 11.←  
97, 11.98, 11.99, 12.00, 12.02, 12.03, 12.04, 12.05, 12.07, 12.08, 12.09, 12.10, 12.11, 12.←  
13, 12.14, 12.15, 12.16, 12.18, 12.19, 12.20, 12.21, 12.23, 12.24, 12.25, 12.26, 12.28, 12.←  
29, 12.30, 12.31, 12.33, 12.34, 12.35, 12.36, 12.38, 12.39, 12.40, 12.41, 12.43, 12.44, 12.←  
45, 12.46, 12.48, 12.49, 12.50, 12.51, 12.53, 12.54, 12.55, 12.56, 12.58, 12.59, 12.60, 12.←  
61, 12.63, 12.64, 12.65, 12.66, 12.68, 12.69, 12.70, 12.72, 12.73, 12.74, 12.75, 12.77, 12.←  
78, 12.79, 12.80, 12.82, 12.83, 12.84, 12.86, 12.87, 12.88, 12.89, 12.91, 12.92, 12.93, 12.←  
94, 12.96, 12.97, 12.98, 13.00, 13.01, 13.02, 13.03, 13.05, 13.06, 13.07, 13.09, 13.10, 13.←  
11, 13.12, 13.14, 13.15, 13.16, 13.18, 13.19, 13.20, 13.22, 13.23, 13.24, 13.25, 13.27, 13.←  
28, 13.29, 13.31, 13.32, 13.33, 13.35, 13.36, 13.37, 13.38, 13.40, 13.41, 13.42, 13.44, 13.←  
45, 13.46, 13.48, 13.49, 13.50, 13.52, 13.53, 13.54, 13.55, 13.57, 13.58, 13.59, 13.61, 13.←  
62, 13.63, 13.65, 13.66, 13.67, 13.69, 13.70, 13.71, 13.73, 13.74, 13.75, 13.77, 13.78, 13.←  
79, 13.81, 13.82, 13.83, 13.85, 13.86, 13.87, 13.89, 13.90, 13.91, 13.93, 13.94, 13.95, 13.←  
97, 13.98, 13.99, 14.01, 14.02, 14.03, 14.05, 14.06, 14.07, 14.09, 14.10, 14.11, 14.13, 14.←  
14, 14.16, 14.17, 14.18, 14.20, 14.21, 14.22, 14.24, 14.25, 14.26, 14.28, 14.29, 14.30, 14.←  
32, 14.33, 14.35, 14.36, 14.37, 14.39, 14.40, 14.41, 14.43, 14.44, 14.45, 14.47, 14.48, 14.←  
50, 14.51, 14.52, 14.54, 14.55, 14.56, 14.58, 14.59, 14.61, 14.62, 14.63, 14.65, 14.66, 14.←  
67, 14.69, 14.70, 14.72, 14.73, 14.74, 14.76, 14.77, 14.79, 14.80, 14.81, 14.83, 14.84, 14.←  
86, 14.87, 14.88, 14.90, 14.91, 14.93, 14.94, 14.95, 14.97, 14.98, 15.00, 15.01, 15.02, 15.←  
04, 15.05, 15.07, 15.08, 15.09, 15.11, 15.12, 15.14, 15.15, 15.16, 15.18, 15.19, 15.21, 15.←  
22, 15.24, 15.25, 15.26, 15.28, 15.29, 15.31, 15.32, 15.33, 15.35, 15.36, 15.38, 15.39, 15.←  
41, 15.42, 15.43, 15.45, 15.46, 15.48, 15.49, 15.51, 15.52, 15.54, 15.55, 15.56, 15.58, 15.←  
59, 15.61, 15.62, 15.64, 15.65, 15.66, 15.68, 15.69, 15.71, 15.72, 15.74, 15.75, 15.77, 15.←  
78, 15.80, 15.81, 15.82, 15.84, 15.85, 15.87, 15.88, 15.90, 15.91, 15.93, 15.94, 15.96, 15.←  
97, 15.99, 16.00, 16.01, 16.03, 16.04, 16.06, 16.07, 16.09, 16.10, 16.12, 16.13, 16.15, 16.←  
16, 16.18, 16.19, 16.21, 16.22, 16.24, 16.25, 16.27, 16.28, 16.30, 16.31, 16.33, 16.34, 16.←  
35, 16.37, 16.38, 16.40, 16.41, 16.43, 16.44, 16.46, 16.47, 16.49, 16.50, 16.52, 16.53, 16.←  
55, 16.56, 16.58, 16.59, 16.61, 16.62, 16.64, 16.66, 16.67, 16.69, 16.70, 16.72, 16.73, 16.←  
75, 16.76, 16.78, 16.79, 16.81, 16.82, 16.84, 16.85, 16.87, 16.88, 16.90, 16.91, 16.93, 16.←  
94, 16.96, 16.97, 16.99, 17.01, 17.02, 17.04, 17.05, 17.07, 17.08, 17.10, 17.11, 17.13, 17.←  
14, 17.16, 17.17, 17.19, 17.21, 17.22, 17.24, 17.25, 17.27, 17.28, 17.30, 17.31, 17.33, 17.←  
35, 17.36, 17.38, 17.39, 17.41, 17.42, 17.44, 17.45, 17.47, 17.49, 17.50, 17.52, 17.53, 17.←  
55, 17.56, 17.58, 17.60, 17.61, 17.63, 17.64, 17.66, 17.67, 17.69, 17.71, 17.72, 17.74, 17.←  
75, 17.77, 17.79, 17.80, 17.82, 17.83, 17.85, 17.86, 17.88, 17.90, 17.91, 17.93, 17.94, 17.←  
96, 17.98, 17.99, 18.01, 18.02, 18.04, 18.06, 18.07, 18.09, 18.11, 18.12, 18.14, 18.15, 18.←  
17, 18.19, 18.20, 18.22, 18.23, 18.25, 18.27, 18.28, 18.30, 18.32, 18.33, 18.35, 18.36, 18.←  
38, 18.40, 18.41, 18.43, 18.45, 18.46, 18.48, 18.49, 18.51, 18.53, 18.54, 18.56, 18.58, 18.←  
59, 18.61, 18.63, 18.64, 18.66, 18.68, 18.69, 18.71, 18.73, 18.74, 18.76, 18.77, 18.79, 18.←  
81, 18.82, 18.84, 18.86, 18.87, 18.89, 18.91, 18.92, 18.94, 18.96, 18.97, 18.99, 19.01, 19.←  
02, 19.04, 19.06, 19.08, 19.09, 19.11, 19.13, 19.14, 19.16, 19.18, 19.19, 19.21, 19.23, 19.←  
24, 19.26, 19.28, 19.29, 19.31, 19.33, 19.35, 19.36, 19.38, 19.40, 19.41, 19.43, 19.45, 19.←  
46, 19.48, 19.50, 19.52, 19.53, 19.55, 19.57, 19.58, 19.60, 19.62, 19.64, 19.65, 19.67, 19.←  
69, 19.70, 19.72, 19.74, 19.76, 19.77, 19.79, 19.81, 19.83, 19.84, 19.86, 19.88, 19.90, 19.←  
91, 19.93, 19.95, 19.97, 19.98, 20.00, 20.02, 20.04, 20.05, 20.07, 20.09, 20.11, 20.12, 20.←  
14, 20.16, 20.18, 20.19, 20.21, 20.23, 20.25, 20.26, 20.28, 20.30, 20.32, 20.33, 20.35, 20.←  
37, 20.39, 20.41, 20.42, 20.44, 20.46, 20.48, 20.49, 20.51, 20.53, 20.55, 20.57, 20.58, 20.←  
60, 20.62, 20.64, 20.66, 20.67, 20.69, 20.71, 20.73, 20.75, 20.76, 20.78, 20.80, 20.82, 20.←  
84, 20.85, 20.87, 20.89, 20.91, 20.93, 20.95, 20.96, 20.98, 21.00, 21.02, 21.04, 21.06, 21.←  
07, 21.09, 21.11, 21.13, 21.15, 21.17, 21.18, 21.20, 21.22, 21.24, 21.26, 21.28, 21.29, 21.←  
31, 21.33, 21.35, 21.37, 21.39, 21.41, 21.42, 21.44, 21.46, 21.48, 21.50, 21.52, 21.54, 21.←  
55, 21.57, 21.59, 21.61, 21.63, 21.65, 21.67, 21.69, 21.70, 21.72, 21.74, 21.76, 21.78, 21.←  
80, 21.82, 21.84, 21.86, 21.87, 21.89, 21.91, 21.93, 21.95, 21.97, 21.99, 22.01, 22.03, 22.←  
05, 22.06, 22.08, 22.10, 22.12, 22.14, 22.16, 22.18, 22.20, 22.22, 22.24, 22.26, 22.28, 22.←  
30, 22.31, 22.33, 22.35, 22.37, 22.39, 22.41, 22.43, 22.45, 22.47, 22.49, 22.51, 22.53, 22.←  
55, 22.57, 22.59, 22.61, 22.63, 22.64, 22.66, 22.68, 22.70, 22.72, 22.74, 22.76, 22.78, 22.←  
80, 22.82, 22.84, 22.86, 22.88, 22.90, 22.92, 22.94, 22.96, 22.98, 23.00, 23.02, 23.04, 23.←  
06, 23.08, 23.10, 23.12, 23.14, 23.16, 23.18, 23.20, 23.22, 23.24, 23.26, 23.28, 23.30, 23.←  
32, 23.34, 23.36, 23.38, 23.40, 23.42, 23.44, 23.46, 23.48, 23.50, 23.52, 23.54, 23.56, 23.←

58, 23.60, 23.62, 23.65, 23.67, 23.69, 23.71, 23.73, 23.75, 23.77, 23.79, 23.81, 23.83, 23.←  
85, 23.87, 23.89, 23.91, 23.93, 23.95, 23.97, 24.00, 24.02, 24.04, 24.06, 24.08, 24.10, 24.←  
12, 24.14, 24.16, 24.18, 24.20, 24.22, 24.25, 24.27, 24.29, 24.31, 24.33, 24.35, 24.37, 24.←  
39, 24.41, 24.43, 24.46, 24.48, 24.50, 24.52, 24.54, 24.56, 24.58, 24.60, 24.63, 24.65, 24.←  
67, 24.69, 24.71, 24.73, 24.75, 24.78, 24.80, 24.82, 24.84, 24.86, 24.88, 24.90, 24.93, 24.←  
95, 24.97, 24.99, 25.01, 25.03, 25.06, 25.08, 25.10, 25.12, 25.14, 25.16, 25.19, 25.21, 25.←  
23, 25.25, 25.27, 25.30, 25.32, 25.34, 25.36, 25.38, 25.41, 25.43, 25.45, 25.47, 25.49, 25.←  
52, 25.54, 25.56, 25.58, 25.60, 25.63, 25.65, 25.67, 25.69, 25.72, 25.74, 25.76, 25.78, 25.←  
81, 25.83, 25.85, 25.87, 25.89, 25.92, 25.94, 25.96, 25.98, 26.01, 26.03, 26.05, 26.08, 26.←  
10, 26.12, 26.14, 26.17, 26.19, 26.21, 26.23, 26.26, 26.28, 26.30, 26.33, 26.35, 26.37, 26.←  
39, 26.42, 26.44, 26.46, 26.49, 26.51, 26.53, 26.56, 26.58, 26.60, 26.63, 26.65, 26.67, 26.←  
69, 26.72, 26.74, 26.76, 26.79, 26.81, 26.83, 26.86, 26.88, 26.90, 26.93, 26.95, 26.98, 27.←  
00, 27.02, 27.05, 27.07, 27.09, 27.12, 27.14, 27.16, 27.19, 27.21, 27.24, 27.26, 27.28, 27.←  
31, 27.33, 27.35, 27.38, 27.40, 27.43, 27.45, 27.47, 27.50, 27.52, 27.55, 27.57, 27.59, 27.←  
62, 27.64, 27.67, 27.69, 27.72, 27.74, 27.76, 27.79, 27.81, 27.84, 27.86, 27.89, 27.91, 27.←  
93, 27.96, 27.98, 28.01, 28.03, 28.06, 28.08, 28.11, 28.13, 28.16, 28.18, 28.21, 28.23, 28.←  
26, 28.28, 28.30, 28.33, 28.35, 28.38, 28.40, 28.43, 28.45, 28.48, 28.50, 28.53, 28.55, 28.←  
58, 28.60, 28.63, 28.66, 28.68, 28.71, 28.73, 28.76, 28.78, 28.81, 28.83, 28.86, 28.88, 28.←  
91, 28.93, 28.96, 28.99, 29.01, 29.04, 29.06, 29.09, 29.11, 29.14, 29.17, 29.19, 29.22, 29.←  
24, 29.27, 29.29, 29.32, 29.35, 29.37, 29.40, 29.42, 29.45, 29.48, 29.50, 29.53, 29.55, 29.←  
58, 29.61, 29.63, 29.66, 29.69, 29.71, 29.74, 29.76, 29.79, 29.82, 29.84, 29.87, 29.90, 29.←  
92, 29.95, 29.98, 30.00, 30.03, 30.06, 30.08, 30.11, 30.14, 30.16, 30.19, 30.22, 30.24, 30.←  
27, 30.30, 30.33, 30.35, 30.38, 30.41, 30.43, 30.46, 30.49, 30.52, 30.54, 30.57, 30.60, 30.←  
62, 30.65, 30.68, 30.71, 30.73, 30.76, 30.79, 30.82, 30.84, 30.87, 30.90, 30.93, 30.96, 30.←  
98, 31.01, 31.04, 31.07, 31.09, 31.12, 31.15, 31.18, 31.21, 31.23, 31.26, 31.29, 31.32, 31.←  
35, 31.37, 31.40, 31.43, 31.46, 31.49, 31.52, 31.54, 31.57, 31.60, 31.63, 31.66, 31.69, 31.←  
72, 31.74, 31.77, 31.80, 31.83, 31.86, 31.89, 31.92, 31.95, 31.97, 32.00, 32.03, 32.06, 32.←  
09, 32.12, 32.15, 32.18, 32.21, 32.24, 32.27, 32.29, 32.32, 32.35, 32.38, 32.41, 32.44, 32.←  
47, 32.50, 32.53, 32.56, 32.59, 32.62, 32.65, 32.68, 32.71, 32.74, 32.77, 32.80, 32.83, 32.←  
86, 32.89, 32.92, 32.95, 32.98, 33.01, 33.04, 33.07, 33.10, 33.13, 33.16, 33.19, 33.22, 33.←  
25, 33.28, 33.31, 33.34, 33.37, 33.40, 33.43, 33.46, 33.49, 33.53, 33.56, 33.59, 33.62, 33.←  
65, 33.68, 33.71, 33.74, 33.77, 33.80, 33.84, 33.87, 33.90, 33.93, 33.96, 33.99, 34.02, 34.←  
05, 34.09, 34.12, 34.15, 34.18, 34.21, 34.24, 34.28, 34.31, 34.34, 34.37, 34.40, 34.43, 34.←  
47, 34.50, 34.53, 34.56, 34.59, 34.63, 34.66, 34.69, 34.72, 34.76, 34.79, 34.82, 34.85, 34.←  
89, 34.92, 34.95, 34.98, 35.02, 35.05, 35.08, 35.11, 35.15, 35.18, 35.21, 35.25, 35.28, 35.←  
31, 35.35, 35.38, 35.41, 35.44, 35.48, 35.51, 35.54, 35.58, 35.61, 35.65, 35.68, 35.71, 35.←  
75, 35.78, 35.81, 35.85, 35.88, 35.91, 35.95, 35.98, 36.02, 36.05, 36.08, 36.12, 36.15, 36.←  
19, 36.22, 36.26, 36.29, 36.33, 36.36, 36.39, 36.43, 36.46, 36.50, 36.53, 36.57, 36.60, 36.←  
64, 36.67, 36.71, 36.74, 36.78, 36.81, 36.85, 36.88, 36.92, 36.95, 36.99, 37.02, 37.06, 37.←  
09, 37.13, 37.17, 37.20, 37.24, 37.27, 37.31, 37.34, 37.38, 37.42, 37.45, 37.49, 37.52, 37.←  
56, 37.60, 37.63, 37.67, 37.71, 37.74, 37.78, 37.82, 37.85, 37.89, 37.93, 37.96, 38.00, 38.←  
04, 38.07, 38.11, 38.15, 38.18, 38.22, 38.26, 38.30, 38.33, 38.37, 38.41, 38.44, 38.48, 38.←  
52, 38.56, 38.60, 38.63, 38.67, 38.71, 38.75, 38.78, 38.82, 38.86, 38.90, 38.94, 38.97, 39.←  
01, 39.05, 39.09, 39.13, 39.17, 39.21, 39.24, 39.28, 39.32, 39.36, 39.40, 39.44, 39.48, 39.←  
52, 39.56, 39.59, 39.63, 39.67, 39.71, 39.75, 39.79, 39.83, 39.87, 39.91, 39.95, 39.99, 40.←  
03, 40.07, 40.11, 40.15, 40.19, 40.23, 40.27, 40.31, 40.35, 40.39, 40.43, 40.47, 40.51, 40.←  
55, 40.59, 40.64, 40.68, 40.72, 40.76, 40.80, 40.84, 40.88, 40.92, 40.96, 41.01, 41.05, 41.←  
09, 41.13, 41.17, 41.21, 41.26, 41.30, 41.34, 41.38, 41.42, 41.47, 41.51, 41.55, 41.59, 41.←  
64, 41.68, 41.72, 41.76, 41.81, 41.85, 41.89, 41.93, 41.98, 42.02, 42.06, 42.11, 42.15, 42.←  
19, 42.24, 42.28, 42.32, 42.37, 42.41, 42.46, 42.50, 42.54, 42.59, 42.63, 42.68, 42.72, 42.←  
76, 42.81, 42.85, 42.90, 42.94, 42.99, 43.03, 43.08, 43.12, 43.17, 43.21, 43.26, 43.30, 43.←  
35, 43.39, 43.44, 43.48, 43.53, 43.58, 43.62, 43.67, 43.71, 43.76, 43.81, 43.85, 43.90, 43.←  
94, 43.99, 44.04, 44.08, 44.13, 44.18, 44.23, 44.27, 44.32, 44.37, 44.41, 44.46, 44.51, 44.←  
56, 44.60, 44.65, 44.70, 44.75, 44.80, 44.84, 44.89, 44.94, 44.99, 45.04, 45.08, 45.13, 45.←  
18, 45.23, 45.28, 45.33, 45.38, 45.43, 45.48, 45.53, 45.57, 45.62, 45.67, 45.72, 45.77, 45.←  
82, 45.87, 45.92, 45.97, 46.02, 46.07, 46.12, 46.17, 46.23, 46.28, 46.33, 46.38, 46.43, 46.←  
48, 46.53, 46.58, 46.63, 46.69, 46.74, 46.79, 46.84, 46.89, 46.95, 47.00, 47.05, 47.10, 47.←  
15, 47.21, 47.26, 47.31, 47.37, 47.42, 47.47, 47.52, 47.58, 47.63, 47.69, 47.74, 47.79, 47.←



### 6.37.3.5 tempMotorLUT

0.69, 0.69, 0.70, 0.71, 0.72, 0.72, 0.73, 0.74, 0.75, 0.75, 0.76, 0.77, 0.77, 0.78, 0.79, 0.79, 0.79,  
0.80, 0.80, 0.81, 0.82, 0.83, 0.83, 0.84, 0.85, 0.85, 0.86, 0.87, 0.88, 0.88, 0.89, 0.89, 0.90, 0.91,  
0.91, 0.92, 0.93, 0.94, 0.94, 0.95, 0.96, 0.96, 0.97, 0.98, 0.99, 0.99, 1.00, 1.01, 1.02, 1.02,  
1.02, 1.03, 1.04, 1.05, 1.05, 1.06, 1.07, 1.08, 1.08, 1.09, 1.10, 1.10, 1.11, 1.12, 1.13, 1.13,  
1.14, 1.15, 1.16, 1.16, 1.17, 1.18, 1.19, 1.19, 1.20, 1.21, 1.22, 1.22, 1.23, 1.24, 1.25, 1.25,  
1.25, 1.26, 1.27, 1.28, 1.28, 1.29, 1.30, 1.31, 1.31, 1.32, 1.33, 1.34, 1.34, 1.35, 1.36, 1.37,  
1.37, 1.38, 1.39, 1.40, 1.40, 1.41, 1.42, 1.43, 1.43, 1.44, 1.45, 1.46, 1.46, 1.47, 1.47, 1.48, 1.48,  
1.49, 1.49, 1.50, 1.51, 1.52, 1.52, 1.53, 1.54, 1.55, 1.55, 1.56, 1.57, 1.58, 1.58, 1.59, 1.60,  
1.61, 1.61, 1.62, 1.63, 1.64, 1.64, 1.65, 1.66, 1.67, 1.67, 1.68, 1.69, 1.70, 1.71, 1.71, 1.71,  
1.72, 1.73, 1.74, 1.74, 1.75, 1.76, 1.77, 1.77, 1.78, 1.79, 1.80, 1.80, 1.81, 1.82, 1.83, 1.84,  
1.84, 1.85, 1.86, 1.87, 1.87, 1.88, 1.89, 1.90, 1.90, 1.91, 1.92, 1.93, 1.93, 1.94, 1.95, 1.95,  
1.96, 1.97, 1.97, 1.98, 1.99, 2.00, 2.00, 2.01, 2.02, 2.03, 2.04, 2.04, 2.05, 2.06, 2.07, 2.07,  
2.08, 2.09, 2.10, 2.10, 2.11, 2.12, 2.13, 2.14, 2.14, 2.15, 2.16, 2.17, 2.17, 2.18, 2.19, 2.19,  
2.20, 2.21, 2.21, 2.22, 2.23, 2.24, 2.25, 2.25, 2.26, 2.27, 2.28, 2.28, 2.29, 2.30, 2.31, 2.32,  
2.32, 2.33, 2.34, 2.35, 2.35, 2.36, 2.37, 2.38, 2.39, 2.39, 2.40, 2.41, 2.42, 2.43, 2.43, 2.43,  
2.44, 2.45, 2.46, 2.46, 2.47, 2.48, 2.49, 2.50, 2.50, 2.51, 2.52, 2.53, 2.54, 2.54, 2.55, 2.56,  
2.57, 2.58, 2.58, 2.59, 2.60, 2.61, 2.62, 2.62, 2.63, 2.64, 2.65, 2.66, 2.66, 2.67, 2.68, 2.68,  
2.69, 2.70, 2.70, 2.71, 2.72, 2.73, 2.74, 2.74, 2.75, 2.76, 2.77, 2.78, 2.78, 2.79, 2.80, 2.81,  
2.82, 2.82, 2.83, 2.84, 2.85, 2.86, 2.86, 2.87, 2.88, 2.89, 2.90, 2.90, 2.91, 2.92, 2.93, 2.93,  
2.94, 2.94, 2.95, 2.96, 2.97, 2.98, 2.98, 2.99, 3.00, 3.01, 3.02, 3.02, 3.03, 3.04, 3.05, 3.06,  
3.07, 3.07, 3.08, 3.09, 3.10, 3.11, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 3.16, 3.17, 3.18, 3.18,  
3.19, 3.20, 3.20, 3.21, 3.22, 3.23, 3.24, 3.24, 3.25, 3.26, 3.27, 3.28, 3.29, 3.29, 3.30, 3.31,  
3.32, 3.33, 3.34, 3.34, 3.35, 3.36, 3.37, 3.38, 3.38, 3.39, 3.40, 3.41, 3.42, 3.43, 3.43, 3.43,  
3.44, 3.45, 3.46, 3.47, 3.48, 3.48, 3.49, 3.50, 3.51, 3.52, 3.53, 3.53, 3.54, 3.55, 3.56, 3.57,  
3.58, 3.58, 3.59, 3.60, 3.61, 3.62, 3.63, 3.63, 3.64, 3.65, 3.66, 3.67, 3.68, 3.68, 3.69, 3.69,  
3.70, 3.71, 3.72, 3.73, 3.73, 3.74, 3.75, 3.76, 3.77, 3.78, 3.78, 3.79, 3.80, 3.81, 3.82, 3.83,  
3.83, 3.84, 3.85, 3.86, 3.87, 3.88, 3.89, 3.89, 3.90, 3.91, 3.92, 3.93, 3.94, 3.94, 3.95, 3.95,  
3.96, 3.97, 3.98, 3.99, 4.00, 4.00, 4.01, 4.02, 4.03, 4.04, 4.04, 4.05, 4.05, 4.05, 4.06, 4.07, 4.08, 4.09,  
4.10, 4.11, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, 4.22,  
4.23, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29, 4.29, 4.30, 4.31, 4.32, 4.33, 4.34, 4.35, 4.35,  
4.36, 4.37, 4.38, 4.39, 4.40, 4.41, 4.42, 4.42, 4.43, 4.44, 4.45, 4.46, 4.47, 4.48, 4.48, 4.48,  
4.49, 4.50, 4.51, 4.52, 4.53, 4.54, 4.55, 4.55, 4.56, 4.57, 4.58, 4.59, 4.60, 4.61, 4.62, 4.62,  
4.63, 4.64, 4.65, 4.66, 4.67, 4.68, 4.69, 4.69, 4.70, 4.71, 4.72, 4.73, 4.74, 4.75, 4.76, 4.76,  
4.76, 4.77, 4.78, 4.79, 4.80, 4.81, 4.82, 4.83, 4.83, 4.84, 4.85, 4.85, 4.86, 4.87, 4.88, 4.89, 4.90,  
4.91, 4.91, 4.92, 4.93, 4.94, 4.95, 4.96, 4.97, 4.98, 4.99, 4.99, 5.00, 5.01, 5.02, 5.03, 5.03,  
5.04, 5.05, 5.06, 5.07, 5.07, 5.08, 5.09, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.16, 5.17,  
5.18, 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, 5.24, 5.25, 5.26, 5.27, 5.28, 5.29, 5.30, 5.31, 5.31,  
5.32, 5.33, 5.34, 5.34, 5.35, 5.36, 5.37, 5.38, 5.39, 5.40, 5.41, 5.42, 5.43, 5.43, 5.44, 5.45,  
5.46, 5.47, 5.48, 5.49, 5.50, 5.51, 5.52, 5.53, 5.53, 5.54, 5.55, 5.56, 5.57, 5.58, 5.59, 5.59,  
5.60, 5.61, 5.62, 5.63, 5.64, 5.64, 5.65, 5.66, 5.67, 5.68, 5.69, 5.70, 5.71, 5.72, 5.73, 5.74,  
5.75, 5.75, 5.76, 5.77, 5.78, 5.79, 5.80, 5.81, 5.82, 5.83, 5.84, 5.84, 5.85, 5.86, 5.87, 5.88, 5.88,  
5.88, 5.89, 5.90, 5.91, 5.92, 5.93, 5.94, 5.95, 5.96, 5.97, 5.98, 5.99, 5.99, 6.00, 6.01, 6.01, 6.02,  
6.03, 6.04, 6.05, 6.06, 6.07, 6.08, 6.09, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, 6.16, 6.16,  
6.17, 6.18, 6.19, 6.20, 6.21, 6.22, 6.23, 6.24, 6.25, 6.26, 6.27, 6.28, 6.29, 6.30, 6.31, 6.32,  
6.32, 6.33, 6.34, 6.35, 6.36, 6.37, 6.38, 6.39, 6.40, 6.41, 6.42, 6.43, 6.44, 6.45, 6.46, 6.46,  
6.47, 6.48, 6.49, 6.50, 6.51, 6.51, 6.52, 6.53, 6.54, 6.55, 6.56, 6.57, 6.58, 6.59, 6.60, 6.61,  
6.62, 6.63, 6.64, 6.65, 6.66, 6.67, 6.68, 6.69, 6.70, 6.71, 6.72, 6.73, 6.74, 6.75, 6.75, 6.75,  
6.76, 6.77, 6.78, 6.79, 6.80, 6.81, 6.82, 6.83, 6.84, 6.85, 6.86, 6.87, 6.88, 6.89, 6.90, 6.91,  
6.92, 6.93, 6.94, 6.95, 6.96, 6.97, 6.98, 6.99, 7.00, 7.01, 7.02, 7.03, 7.04, 7.05, 7.06, 7.06,  
7.07, 7.08, 7.09, 7.10, 7.11, 7.12, 7.12, 7.13, 7.14, 7.15, 7.16, 7.17, 7.18, 7.19, 7.20, 7.21,  
7.22, 7.23, 7.24, 7.25, 7.26, 7.27, 7.28, 7.29, 7.30, 7.31, 7.32, 7.33, 7.34, 7.35, 7.36, 7.36,  
7.37, 7.38, 7.39, 7.40, 7.41, 7.42, 7.43, 7.44, 7.45, 7.46, 7.47, 7.48, 7.49, 7.50, 7.51, 7.52,  
7.53, 7.54, 7.55, 7.56, 7.57, 7.58, 7.59, 7.60, 7.61, 7.62, 7.63, 7.64, 7.65, 7.66, 7.67, 7.67,  
7.68, 7.69, 7.70, 7.71, 7.72, 7.73, 7.74, 7.75, 7.76, 7.77, 7.78, 7.79, 7.80, 7.81, 7.82, 7.83,  
7.84, 7.85, 7.86, 7.87, 7.88, 7.89, 7.91, 7.92, 7.93, 7.94, 7.95, 7.96, 7.97, 7.98, 7.99, 8.00,  
8.01, 8.02, 8.03, 8.04, 8.05, 8.06, 8.07, 8.08, 8.09, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15,  
8.16, 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 8.23, 8.24, 8.25, 8.26, 8.27, 8.28, 8.29, 8.30, 8.31, 8.31,  
8.32, 8.33, 8.34, 8.35, 8.36, 8.37, 8.38, 8.39, 8.40, 8.41, 8.42, 8.43, 8.44, 8.45, 8.46, 8.47,

8.48, 8.49, 8.50, 8.51, 8.52, 8.54, 8.55, 8.56, 8.57, 8.58, 8.59, 8.60, 8.61, 8.62, 8.63, 8.←  
64, 8.65, 8.66, 8.67, 8.68, 8.69, 8.70, 8.71, 8.72, 8.74, 8.75, 8.76, 8.77, 8.78, 8.79, 8.80,  
8.81, 8.82, 8.83, 8.84, 8.85, 8.86, 8.87, 8.88, 8.89, 8.91, 8.92, 8.93, 8.94, 8.95, 8.96, 8.←  
97, 8.98, 8.99, 9.00, 9.01, 9.02, 9.03, 9.04, 9.06, 9.07, 9.08, 9.09, 9.10, 9.11, 9.12, 9.13,  
9.14, 9.15, 9.16, 9.17, 9.18, 9.20, 9.21, 9.22, 9.23, 9.24, 9.25, 9.26, 9.27, 9.28, 9.29, 9.←  
30, 9.31, 9.33, 9.34, 9.35, 9.36, 9.37, 9.38, 9.39, 9.40, 9.41, 9.42, 9.43, 9.45, 9.46, 9.47,  
9.48, 9.49, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55, 9.57, 9.58, 9.59, 9.60, 9.61, 9.62, 9.63, 9.←  
64, 9.65, 9.66, 9.68, 9.69, 9.70, 9.71, 9.72, 9.73, 9.74, 9.75, 9.76, 9.78, 9.79, 9.80, 9.81,  
9.82, 9.83, 9.84, 9.85, 9.86, 9.88, 9.89, 9.90, 9.91, 9.92, 9.93, 9.94, 9.95, 9.96, 9.98, 9.←  
99, 10.00, 10.01, 10.02, 10.03, 10.04, 10.05, 10.07, 10.08, 10.09, 10.10, 10.11, 10.12, 10.←  
13, 10.14, 10.16, 10.17, 10.18, 10.19, 10.20, 10.21, 10.22, 10.24, 10.25, 10.26, 10.27, 10.←  
28, 10.29, 10.30, 10.31, 10.33, 10.34, 10.35, 10.36, 10.37, 10.38, 10.39, 10.41, 10.42, 10.←  
43, 10.44, 10.45, 10.46, 10.47, 10.49, 10.50, 10.51, 10.52, 10.53, 10.54, 10.55, 10.57, 10.←  
58, 10.59, 10.60, 10.61, 10.62, 10.64, 10.65, 10.66, 10.67, 10.68, 10.69, 10.70, 10.72, 10.←  
73, 10.74, 10.75, 10.76, 10.77, 10.79, 10.80, 10.81, 10.82, 10.83, 10.84, 10.86, 10.87, 10.←  
88, 10.89, 10.90, 10.91, 10.93, 10.94, 10.95, 10.96, 10.97, 10.98, 11.00, 11.01, 11.02, 11.←  
03, 11.04, 11.05, 11.07, 11.08, 11.09, 11.10, 11.11, 11.13, 11.14, 11.15, 11.16, 11.17, 11.←  
18, 11.20, 11.21, 11.22, 11.23, 11.24, 11.26, 11.27, 11.28, 11.29, 11.30, 11.32, 11.33, 11.←  
34, 11.35, 11.36, 11.37, 11.39, 11.40, 11.41, 11.42, 11.43, 11.45, 11.46, 11.47, 11.48, 11.←  
49, 11.51, 11.52, 11.53, 11.54, 11.55, 11.57, 11.58, 11.59, 11.60, 11.61, 11.63, 11.64, 11.←  
65, 11.66, 11.68, 11.69, 11.70, 11.71, 11.72, 11.74, 11.75, 11.76, 11.77, 11.78, 11.80, 11.←  
81, 11.82, 11.83, 11.85, 11.86, 11.87, 11.88, 11.89, 11.91, 11.92, 11.93, 11.94, 11.96, 11.←  
97, 11.98, 11.99, 12.00, 12.02, 12.03, 12.04, 12.05, 12.07, 12.08, 12.09, 12.10, 12.11, 12.←  
13, 12.14, 12.15, 12.16, 12.18, 12.19, 12.20, 12.21, 12.23, 12.24, 12.25, 12.26, 12.28, 12.←  
29, 12.30, 12.31, 12.33, 12.34, 12.35, 12.36, 12.38, 12.39, 12.40, 12.41, 12.43, 12.44, 12.←  
45, 12.46, 12.48, 12.49, 12.50, 12.51, 12.53, 12.54, 12.55, 12.56, 12.58, 12.59, 12.60, 12.←  
61, 12.63, 12.64, 12.65, 12.66, 12.68, 12.69, 12.70, 12.72, 12.73, 12.74, 12.75, 12.77, 12.←  
78, 12.79, 12.80, 12.82, 12.83, 12.84, 12.86, 12.87, 12.88, 12.89, 12.91, 12.92, 12.93, 12.←  
94, 12.96, 12.97, 12.98, 13.00, 13.01, 13.02, 13.03, 13.05, 13.06, 13.07, 13.09, 13.10, 13.←  
11, 13.12, 13.14, 13.15, 13.16, 13.18, 13.19, 13.20, 13.22, 13.23, 13.24, 13.25, 13.27, 13.←  
28, 13.29, 13.31, 13.32, 13.33, 13.35, 13.36, 13.37, 13.38, 13.40, 13.41, 13.42, 13.44, 13.←  
45, 13.46, 13.48, 13.49, 13.50, 13.52, 13.53, 13.54, 13.55, 13.57, 13.58, 13.59, 13.61, 13.←  
62, 13.63, 13.65, 13.66, 13.67, 13.69, 13.70, 13.71, 13.73, 13.74, 13.75, 13.77, 13.78, 13.←  
79, 13.81, 13.82, 13.83, 13.85, 13.86, 13.87, 13.89, 13.90, 13.91, 13.93, 13.94, 13.95, 13.←  
97, 13.98, 13.99, 14.01, 14.02, 14.03, 14.05, 14.06, 14.07, 14.09, 14.10, 14.11, 14.13, 14.←  
14, 14.16, 14.17, 14.18, 14.20, 14.21, 14.22, 14.24, 14.25, 14.26, 14.28, 14.29, 14.30, 14.←  
32, 14.33, 14.35, 14.36, 14.37, 14.39, 14.40, 14.41, 14.43, 14.44, 14.45, 14.47, 14.48, 14.←  
50, 14.51, 14.52, 14.54, 14.55, 14.56, 14.58, 14.59, 14.61, 14.62, 14.63, 14.65, 14.66, 14.←  
67, 14.69, 14.70, 14.72, 14.73, 14.74, 14.76, 14.77, 14.79, 14.80, 14.81, 14.83, 14.84, 14.←  
86, 14.87, 14.88, 14.90, 14.91, 14.93, 14.94, 14.95, 14.97, 14.98, 15.00, 15.01, 15.02, 15.←  
04, 15.05, 15.07, 15.08, 15.09, 15.11, 15.12, 15.14, 15.15, 15.16, 15.18, 15.19, 15.21, 15.←  
22, 15.24, 15.25, 15.26, 15.28, 15.29, 15.31, 15.32, 15.33, 15.35, 15.36, 15.38, 15.39, 15.←  
41, 15.42, 15.43, 15.45, 15.46, 15.48, 15.49, 15.51, 15.52, 15.54, 15.55, 15.56, 15.58, 15.←  
59, 15.61, 15.62, 15.64, 15.65, 15.66, 15.68, 15.69, 15.71, 15.72, 15.74, 15.75, 15.77, 15.←  
78, 15.80, 15.81, 15.82, 15.84, 15.85, 15.87, 15.88, 15.90, 15.91, 15.93, 15.94, 15.96, 15.←  
97, 15.99, 16.00, 16.01, 16.03, 16.04, 16.06, 16.07, 16.09, 16.10, 16.12, 16.13, 16.15, 16.←  
16, 16.18, 16.19, 16.21, 16.22, 16.24, 16.25, 16.27, 16.28, 16.30, 16.31, 16.33, 16.34, 16.←  
35, 16.37, 16.38, 16.40, 16.41, 16.43, 16.44, 16.46, 16.47, 16.49, 16.50, 16.52, 16.53, 16.←  
55, 16.56, 16.58, 16.59, 16.61, 16.62, 16.64, 16.66, 16.67, 16.69, 16.70, 16.72, 16.73, 16.←  
75, 16.76, 16.78, 16.79, 16.81, 16.82, 16.84, 16.85, 16.87, 16.88, 16.90, 16.91, 16.93, 16.←  
94, 16.96, 16.97, 16.99, 17.01, 17.02, 17.04, 17.05, 17.07, 17.08, 17.10, 17.11, 17.13, 17.←  
14, 17.16, 17.17, 17.19, 17.21, 17.22, 17.24, 17.25, 17.27, 17.28, 17.30, 17.31, 17.33, 17.←  
35, 17.36, 17.38, 17.39, 17.41, 17.42, 17.44, 17.45, 17.47, 17.49, 17.50, 17.52, 17.53, 17.←  
55, 17.56, 17.58, 17.60, 17.61, 17.63, 17.64, 17.66, 17.67, 17.69, 17.71, 17.72, 17.74, 17.←  
75, 17.77, 17.79, 17.80, 17.82, 17.83, 17.85, 17.86, 17.88, 17.90, 17.91, 17.93, 17.94, 17.←  
96, 17.98, 17.99, 18.01, 18.02, 18.04, 18.06, 18.07, 18.09, 18.11, 18.12, 18.14, 18.15, 18.←  
17, 18.19, 18.20, 18.22, 18.23, 18.25, 18.27, 18.28, 18.30, 18.32, 18.33, 18.35, 18.36, 18.←  
38, 18.40, 18.41, 18.43, 18.45, 18.46, 18.48, 18.49, 18.51, 18.53, 18.54, 18.56, 18.58, 18.←

59, 18.61, 18.63, 18.64, 18.66, 18.68, 18.69, 18.71, 18.73, 18.74, 18.76, 18.77, 18.79, 18.←  
81, 18.82, 18.84, 18.86, 18.87, 18.89, 18.91, 18.92, 18.94, 18.96, 18.97, 18.99, 19.01, 19.←  
02, 19.04, 19.06, 19.08, 19.09, 19.11, 19.13, 19.14, 19.16, 19.18, 19.19, 19.21, 19.23, 19.←  
24, 19.26, 19.28, 19.29, 19.31, 19.33, 19.35, 19.36, 19.38, 19.40, 19.41, 19.43, 19.45, 19.←  
46, 19.48, 19.50, 19.52, 19.53, 19.55, 19.57, 19.58, 19.60, 19.62, 19.64, 19.65, 19.67, 19.←  
69, 19.70, 19.72, 19.74, 19.76, 19.77, 19.79, 19.81, 19.83, 19.84, 19.86, 19.88, 19.90, 19.←  
91, 19.93, 19.95, 19.97, 19.98, 20.00, 20.02, 20.04, 20.05, 20.07, 20.09, 20.11, 20.12, 20.←  
14, 20.16, 20.18, 20.19, 20.21, 20.23, 20.25, 20.26, 20.28, 20.30, 20.32, 20.33, 20.35, 20.←  
37, 20.39, 20.41, 20.42, 20.44, 20.46, 20.48, 20.49, 20.51, 20.53, 20.55, 20.57, 20.58, 20.←  
60, 20.62, 20.64, 20.66, 20.67, 20.69, 20.71, 20.73, 20.75, 20.76, 20.78, 20.80, 20.82, 20.←  
84, 20.85, 20.87, 20.89, 20.91, 20.93, 20.95, 20.96, 20.98, 21.00, 21.02, 21.04, 21.06, 21.←  
07, 21.09, 21.11, 21.13, 21.15, 21.17, 21.18, 21.20, 21.22, 21.24, 21.26, 21.28, 21.29, 21.←  
31, 21.33, 21.35, 21.37, 21.39, 21.41, 21.42, 21.44, 21.46, 21.48, 21.50, 21.52, 21.54, 21.←  
55, 21.57, 21.59, 21.61, 21.63, 21.65, 21.67, 21.69, 21.70, 21.72, 21.74, 21.76, 21.78, 21.←  
80, 21.82, 21.84, 21.86, 21.87, 21.89, 21.91, 21.93, 21.95, 21.97, 21.99, 22.01, 22.03, 22.←  
05, 22.06, 22.08, 22.10, 22.12, 22.14, 22.16, 22.18, 22.20, 22.22, 22.24, 22.26, 22.28, 22.←  
30, 22.31, 22.33, 22.35, 22.37, 22.39, 22.41, 22.43, 22.45, 22.47, 22.49, 22.51, 22.53, 22.←  
55, 22.57, 22.59, 22.61, 22.63, 22.64, 22.66, 22.68, 22.70, 22.72, 22.74, 22.76, 22.78, 22.←  
80, 22.82, 22.84, 22.86, 22.88, 22.90, 22.92, 22.94, 22.96, 22.98, 23.00, 23.02, 23.04, 23.←  
06, 23.08, 23.10, 23.12, 23.14, 23.16, 23.18, 23.20, 23.22, 23.24, 23.26, 23.28, 23.30, 23.←  
32, 23.34, 23.36, 23.38, 23.40, 23.42, 23.44, 23.46, 23.48, 23.50, 23.52, 23.54, 23.56, 23.←  
58, 23.60, 23.62, 23.65, 23.67, 23.69, 23.71, 23.73, 23.75, 23.77, 23.79, 23.81, 23.83, 23.←  
85, 23.87, 23.89, 23.91, 23.93, 23.95, 23.97, 24.00, 24.02, 24.04, 24.06, 24.08, 24.10, 24.←  
12, 24.14, 24.16, 24.18, 24.20, 24.22, 24.25, 24.27, 24.29, 24.31, 24.33, 24.35, 24.37, 24.←  
39, 24.41, 24.43, 24.46, 24.48, 24.50, 24.52, 24.54, 24.56, 24.58, 24.60, 24.63, 24.65, 24.←  
67, 24.69, 24.71, 24.73, 24.75, 24.78, 24.80, 24.82, 24.84, 24.86, 24.88, 24.90, 24.93, 24.←  
95, 24.97, 24.99, 25.01, 25.03, 25.06, 25.08, 25.10, 25.12, 25.14, 25.16, 25.19, 25.21, 25.←  
23, 25.25, 25.27, 25.30, 25.32, 25.34, 25.36, 25.38, 25.41, 25.43, 25.45, 25.47, 25.49, 25.←  
52, 25.54, 25.56, 25.58, 25.60, 25.63, 25.65, 25.67, 25.69, 25.72, 25.74, 25.76, 25.78, 25.←  
81, 25.83, 25.85, 25.87, 25.89, 25.92, 25.94, 25.96, 25.98, 26.01, 26.03, 26.05, 26.08, 26.←  
10, 26.12, 26.14, 26.17, 26.19, 26.21, 26.23, 26.26, 26.28, 26.30, 26.33, 26.35, 26.37, 26.←  
39, 26.42, 26.44, 26.46, 26.49, 26.51, 26.53, 26.56, 26.58, 26.60, 26.63, 26.65, 26.67, 26.←  
69, 26.72, 26.74, 26.76, 26.79, 26.81, 26.83, 26.86, 26.88, 26.90, 26.93, 26.95, 26.98, 27.←  
00, 27.02, 27.05, 27.07, 27.09, 27.12, 27.14, 27.16, 27.19, 27.21, 27.24, 27.26, 27.28, 27.←  
31, 27.33, 27.35, 27.38, 27.40, 27.43, 27.45, 27.47, 27.50, 27.52, 27.55, 27.57, 27.59, 27.←  
62, 27.64, 27.67, 27.69, 27.72, 27.74, 27.76, 27.79, 27.81, 27.84, 27.86, 27.89, 27.91, 27.←  
93, 27.96, 27.98, 28.01, 28.03, 28.06, 28.08, 28.11, 28.13, 28.16, 28.18, 28.21, 28.23, 28.←  
26, 28.28, 28.30, 28.33, 28.35, 28.38, 28.40, 28.43, 28.45, 28.48, 28.50, 28.53, 28.55, 28.←  
58, 28.60, 28.63, 28.66, 28.68, 28.71, 28.73, 28.76, 28.78, 28.81, 28.83, 28.86, 28.88, 28.←  
91, 28.93, 28.96, 28.99, 29.01, 29.04, 29.06, 29.09, 29.11, 29.14, 29.17, 29.19, 29.22, 29.←  
24, 29.27, 29.29, 29.32, 29.35, 29.37, 29.40, 29.42, 29.45, 29.48, 29.50, 29.53, 29.55, 29.←  
58, 29.61, 29.63, 29.66, 29.69, 29.71, 29.74, 29.76, 29.79, 29.82, 29.84, 29.87, 29.90, 29.←  
92, 29.95, 29.98, 30.00, 30.03, 30.06, 30.08, 30.11, 30.14, 30.16, 30.19, 30.22, 30.24, 30.←  
27, 30.30, 30.33, 30.35, 30.38, 30.41, 30.43, 30.46, 30.49, 30.52, 30.54, 30.57, 30.60, 30.←  
62, 30.65, 30.68, 30.71, 30.73, 30.76, 30.79, 30.82, 30.84, 30.87, 30.90, 30.93, 30.96, 30.←  
98, 31.01, 31.04, 31.07, 31.09, 31.12, 31.15, 31.18, 31.21, 31.23, 31.26, 31.29, 31.32, 31.←  
35, 31.37, 31.40, 31.43, 31.46, 31.49, 31.52, 31.54, 31.57, 31.60, 31.63, 31.66, 31.69, 31.←  
72, 31.74, 31.77, 31.80, 31.83, 31.86, 31.89, 31.92, 31.95, 31.97, 32.00, 32.03, 32.06, 32.←  
09, 32.12, 32.15, 32.18, 32.21, 32.24, 32.27, 32.29, 32.32, 32.35, 32.38, 32.41, 32.44, 32.←  
47, 32.50, 32.53, 32.56, 32.59, 32.62, 32.65, 32.68, 32.71, 32.74, 32.77, 32.80, 32.83, 32.←  
86, 32.89, 32.92, 32.95, 32.98, 33.01, 33.04, 33.07, 33.10, 33.13, 33.16, 33.19, 33.22, 33.←  
25, 33.28, 33.31, 33.34, 33.37, 33.40, 33.43, 33.46, 33.49, 33.53, 33.56, 33.59, 33.62, 33.←  
65, 33.68, 33.71, 33.74, 33.77, 33.80, 33.84, 33.87, 33.90, 33.93, 33.96, 33.99, 34.02, 34.←  
05, 34.09, 34.12, 34.15, 34.18, 34.21, 34.24, 34.28, 34.31, 34.34, 34.37, 34.40, 34.43, 34.←  
47, 34.50, 34.53, 34.56, 34.59, 34.63, 34.66, 34.69, 34.72, 34.76, 34.79, 34.82, 34.85, 34.←  
89, 34.92, 34.95, 34.98, 35.02, 35.05, 35.08, 35.11, 35.15, 35.18, 35.21, 35.25, 35.28, 35.←  
31, 35.35, 35.38, 35.41, 35.44, 35.48, 35.51, 35.54, 35.58, 35.61, 35.65, 35.68, 35.71, 35.←  
75, 35.78, 35.81, 35.85, 35.88, 35.91, 35.95, 35.98, 36.02, 36.05, 36.08, 36.12, 36.15, 36.←

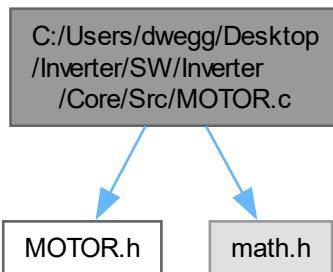
19, 36.22, 36.26, 36.29, 36.33, 36.36, 36.39, 36.43, 36.46, 36.50, 36.53, 36.57, 36.60, 36.←  
64, 36.67, 36.71, 36.74, 36.78, 36.81, 36.85, 36.88, 36.92, 36.95, 36.99, 37.02, 37.06, 37.←  
09, 37.13, 37.17, 37.20, 37.24, 37.27, 37.31, 37.34, 37.38, 37.42, 37.45, 37.49, 37.52, 37.←  
56, 37.60, 37.63, 37.67, 37.71, 37.74, 37.78, 37.82, 37.85, 37.89, 37.93, 37.96, 38.00, 38.←  
04, 38.07, 38.11, 38.15, 38.18, 38.22, 38.26, 38.30, 38.33, 38.37, 38.41, 38.44, 38.48, 38.←  
52, 38.56, 38.60, 38.63, 38.67, 38.71, 38.75, 38.78, 38.82, 38.86, 38.90, 38.94, 38.97, 39.←  
01, 39.05, 39.09, 39.13, 39.17, 39.21, 39.24, 39.28, 39.32, 39.36, 39.40, 39.44, 39.48, 39.←  
52, 39.56, 39.59, 39.63, 39.67, 39.71, 39.75, 39.79, 39.83, 39.87, 39.91, 39.95, 39.99, 40.←  
03, 40.07, 40.11, 40.15, 40.19, 40.23, 40.27, 40.31, 40.35, 40.39, 40.43, 40.47, 40.51, 40.←  
55, 40.59, 40.64, 40.68, 40.72, 40.76, 40.80, 40.84, 40.88, 40.92, 40.96, 41.01, 41.05, 41.←  
09, 41.13, 41.17, 41.21, 41.26, 41.30, 41.34, 41.38, 41.42, 41.47, 41.51, 41.55, 41.59, 41.←  
64, 41.68, 41.72, 41.76, 41.81, 41.85, 41.89, 41.93, 41.98, 42.02, 42.06, 42.11, 42.15, 42.←  
19, 42.24, 42.28, 42.32, 42.37, 42.41, 42.46, 42.50, 42.54, 42.59, 42.63, 42.68, 42.72, 42.←  
76, 42.81, 42.85, 42.90, 42.94, 42.99, 43.03, 43.08, 43.12, 43.17, 43.21, 43.26, 43.30, 43.←  
35, 43.39, 43.44, 43.48, 43.53, 43.58, 43.62, 43.67, 43.71, 43.76, 43.81, 43.85, 43.90, 43.←  
94, 43.99, 44.04, 44.08, 44.13, 44.18, 44.23, 44.27, 44.32, 44.37, 44.41, 44.46, 44.51, 44.←  
56, 44.60, 44.65, 44.70, 44.75, 44.80, 44.84, 44.89, 44.94, 44.99, 45.04, 45.08, 45.13, 45.←  
18, 45.23, 45.28, 45.33, 45.38, 45.43, 45.48, 45.53, 45.57, 45.62, 45.67, 45.72, 45.77, 45.←  
82, 45.87, 45.92, 45.97, 46.02, 46.07, 46.12, 46.17, 46.23, 46.28, 46.33, 46.38, 46.43, 46.←  
48, 46.53, 46.58, 46.63, 46.69, 46.74, 46.79, 46.84, 46.89, 46.95, 47.00, 47.05, 47.10, 47.←  
15, 47.21, 47.26, 47.31, 47.37, 47.42, 47.47, 47.52, 47.58, 47.63, 47.69, 47.74, 47.79, 47.←  
85, 47.90, 47.95, 48.01, 48.06, 48.12, 48.17, 48.23, 48.28, 48.34, 48.39, 48.45, 48.50, 48.←  
56, 48.61, 48.67, 48.72, 48.78, 48.84, 48.89, 48.95, 49.00, 49.06, 49.12, 49.17, 49.23, 49.←  
29, 49.34, 49.40, 49.46, 49.52, 49.57, 49.63, 49.69, 49.75, 49.80, 49.86, 49.92, 49.98, 50.←  
04, 50.10, 50.16, 50.21, 50.27, 50.33, 50.39, 50.45, 50.51, 50.57, 50.63, 50.69, 50.75, 50.←  
81, 50.87, 50.93, 50.99, 51.05, 51.11, 51.17, 51.24, 51.30, 51.36, 51.42, 51.48, 51.54, 51.←  
61, 51.67, 51.73, 51.79, 51.86, 51.92, 51.98, 52.04, 52.11, 52.17, 52.23, 52.30, 52.36, 52.←  
43, 52.49, 52.55, 52.62, 52.68, 52.75, 52.81, 52.88, 52.94, 53.01, 53.07, 53.14, 53.21, 53.←  
27, 53.34, 53.40, 53.47, 53.54, 53.60, 53.67, 53.74, 53.80, 53.87, 53.94, 54.01, 54.08, 54.←  
14, 54.21, 54.28, 54.35, 54.42, 54.49, 54.56, 54.63, 54.70, 54.76, 54.83, 54.90, 54.98, 55.←  
05, 55.12, 55.19, 55.26, 55.33, 55.40, 55.47, 55.54, 55.62, 55.69, 55.76, 55.83, 55.91, 55.←  
98, 56.05, 56.12, 56.20, 56.27, 56.35, 56.42, 56.49, 56.57, 56.64, 56.72, 56.79, 56.87, 56.←  
94, 57.02, 57.10, 57.17, 57.25, 57.32, 57.40, 57.48, 57.56, 57.63, 57.71, 57.79, 57.87, 57.←  
94, 58.02, 58.10, 58.18, 58.26, 58.34, 58.42, 58.50, 58.58, 58.66, 58.74, 58.82, 58.90, 58.←  
98, 59.06, 59.15, 59.23, 59.31, 59.39, 59.48, 59.56, 59.64, 59.72, 59.81, 59.89, 59.98, 60.←  
06, 60.15, 60.23, 60.32, 60.40, 60.49, 60.57, 60.66, 60.75, 60.83, 60.92, 61.01, 61.10, 61.←  
18, 61.27, 61.36, 61.45, 61.54, 61.63, 61.72, 61.81, 61.90, 61.99, 62.08, 62.17, 62.26, 62.←  
35, 62.44, 62.54, 62.63, 62.72, 62.82, 62.91, 63.00, 63.10, 63.19, 63.29, 63.38, 63.48, 63.←  
57, 63.67, 63.76, 63.86, 63.96, 64.06, 64.15, 64.25, 64.35, 64.45, 64.55, 64.65, 64.75, 64.←  
85, 64.95, 65.05, 65.15, 65.25, 65.35, 65.46, 65.56, 65.66, 65.76, 65.87, 65.97, 66.08, 66.←  
18, 66.29, 66.39, 66.50, 66.61, 66.71, 66.82, 66.93, 67.03, 67.14, 67.25, 67.36, 67.47, 67.←  
58, 67.69, 67.80, 67.91, 68.03, 68.14, 68.25, 68.36, 68.48, 68.59, 68.71, 68.82, 68.94, 69.←  
05, 69.17, 69.29, 69.40, 69.52, 69.64, 69.76, 69.88, 70.00, 70.12, 70.24, 70.36, 70.48, 70.←  
60, 70.73, 70.85, 70.97, 71.10, 71.22, 71.35, 71.47, 71.60, 71.73, 71.86, 71.98, 72.11, 72.←  
24, 72.37, 72.50, 72.63, 72.76, 72.90, 73.03, 73.16, 73.30, 73.43, 73.57, 73.70, 73.84, 73.←  
98, 74.11, 74.25, 74.39, 74.53, 74.67, 74.81, 74.95, 75.10, 75.24, 75.38, 75.53, 75.67, 75.←  
82, 75.97, 76.11, 76.26, 76.41, 76.56, 76.71, 76.86, 77.01, 77.17, 77.32, 77.47, 77.63, 77.←  
78, 77.94, 78.10, 78.26, 78.42, 78.58, 78.74, 78.90, 79.06, 79.23, 79.39, 79.55, 79.72, 79.←  
89, 80.06, 80.23, 80.40, 80.57, 80.74, 80.91, 81.08, 81.26, 81.44, 81.61, 81.79, 81.97, 82.←  
15, 82.33, 82.51, 82.70, 82.88, 83.07, 83.25, 83.44, 83.63, 83.82, 84.01, 84.20, 84.40, 84.←  
59, 84.79, 84.98, 85.18, 85.38, 85.58, 85.79, 85.99, 86.20, 86.40, 86.61, 86.82, 87.03, 87.←  
24, 87.46, 87.67, 87.89, 88.11, 88.33, 88.55, 88.77, 89.00, 89.22, 89.45, 89.68, 89.91, 90.←  
14, 90.38, 90.62, 90.85, 91.09, 91.34, 91.58, 91.82, 92.07, 92.32, 92.57, 92.83, 93.08, 93.←  
34, 93.60, 93.86, 94.13, 94.39, 94.66, 94.93, 95.20, 95.48, 95.76, 96.04, 96.32, 96.61, 96.89,  
97.18, 97.48, 97.77, 98.07, 98.37, 98.68, 98.98, 99.29, 99.61, 99.92, 100.24, 100.56, 100.89,  
101.22, 101.55, 101.88, 102.22, 102.56, 102.91, 103.26, 103.61, 103.97, 104.33, 104.70, 105.←  
07, 105.44, 105.82, 106.20, 106.58, 106.98, 107.37, 107.77, 108.18, 108.59, 109.00, 109.42,  
109.85, 110.28, 110.71, 111.16, 111.60, 112.06, 112.52, 112.99, 113.46, 113.94, 114.43, 114.←

## 6.38 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MOTOR.c File Reference

## Source file for motor parameters.

```
#include "MOTOR.h"
#include <math.h>
```

Include dependency graph for MOTOR.c:



## Functions

- int `check_motor_parameters` (`MotorParameters` \*`motor`, float `Ts`)  
*Perform a parameter check and correct possible errors.*

## Variables

- `MotorParameters motor_left`  
*Left motor parameters.*
- `MotorParameters motor_right`  
*Right motor parameters.*

### 6.38.1 Detailed Description

Source file for motor parameters.

#### Attention

Copyright (c) 2024 David Redondo (@dweggg on GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

### 6.38.2 Function Documentation

#### 6.38.2.1 `check_motor_parameters()`

```
int check_motor_parameters (
 MotorParameters * motor,
 float Ts)
```

Perform a parameter check and correct possible errors.

**Parameters**

|    |              |                                                        |
|----|--------------|--------------------------------------------------------|
| in | <i>motor</i> | Pointer to the <a href="#">MotorParameters</a> struct. |
|----|--------------|--------------------------------------------------------|

**Return values**

|    |                                          |
|----|------------------------------------------|
| OK | 0 if an error occurred, 1 if successful. |
|----|------------------------------------------|

Here is the caller graph for this function:



### 6.38.3 Variable Documentation

#### 6.38.3.1 motor\_left

[MotorParameters](#) *motor\_left*

**Initial value:**

```
= {
 .Ld = 0.00291F,
 .Lq = 0.00295F,
 .Rs = 1.95F,
 .lambda = 0.13391F,
 .pp = 4,
 .J = 0.00093F,
 .b = 0.632653F,
 .torque_max = 10.0F,
 .dTorque_max = 1.0F,
 .speed_max_RPM = 8500.0F,
 .iPhase_pk_max = 60.0F,
 .vDC_max = 450.0F
}
```

Left motor parameters.

#### 6.38.3.2 motor\_right

[MotorParameters](#) *motor\_right*

**Initial value:**

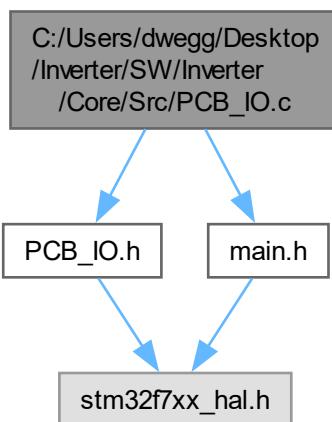
```
= {
 .Ld = 0.00291F,
 .Lq = 0.00295F,
 .Rs = 1.95F,
 .lambda = 0.13391F,
 .pp = 4,
 .J = 0.00093F,
 .b = 0.632653F,
 .torque_max = 10.0F,
 .dTorque_max = 1.0F,
 .speed_max_RPM = 8500.0F,
 .iPhase_pk_max = 60.0F,
 .vDC_max = 450.0F
}
```

Right motor parameters.

## 6.39 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PCB\_IO.c File Reference

This file provides functions for handling GPIOs.

```
#include "PCB_IO.h"
#include "main.h"
Include dependency graph for PCB_IO.c:
```



### Functions

- void `handle_LED (LED *led, uint32_t ms_counter)`  
*LED handler function.*
- void `handle_direction (volatile int8_t *dir_left, volatile int8_t *dir_right)`  
*Handles the direction of the motors.*

### Variables

- `LED led_left = { .port = LED_LEFT_GPIO_Port, .pin = LED_LEFT_Pin, .mode = LED_MODE_OFF }`
- `LED led_right = { .port = LED_RIGHT_GPIO_Port, .pin = LED_RIGHT_Pin, .mode = LED_MODE_OFF }`
- `LED ledError = { .port = LED_ERR_GPIO_Port, .pin = LED_ERR_Pin, .mode = LED_MODE_OFF }`

### 6.39.1 Detailed Description

This file provides functions for handling GPIOs.

#### Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

## 6.39.2 Function Documentation

### 6.39.2.1 handle\_direction()

```
void handle_direction (
 volatile int8_t * dir_left,
 volatile int8_t * dir_right)
```

Handles the direction of the motors.

This function reads the state of the DIR switch and updates the directions of both the left and right motors. If one motor is set to rotate clockwise (CW), the other one is set to rotate counterclockwise (CCW), and vice versa.

#### Parameters

|                  |                                                                     |
|------------------|---------------------------------------------------------------------|
| <i>dir_left</i>  | Pointer to the direction parameter in the left inverter structure.  |
| <i>dir_right</i> | Pointer to the direction parameter in the right inverter structure. |

Here is the caller graph for this function:



### 6.39.2.2 handle\_LED()

```
void handle_LED (
 LED * led,
 uint32_t ms_counter)
```

[LED](#) handler function.

This function handles the [LED](#) blinking modes based on the [LED](#) mode and current millisecond counter.

#### Parameters

|                   |                                               |
|-------------------|-----------------------------------------------|
| <i>led</i>        | Pointer to the <a href="#">LED</a> structure. |
| <i>ms_counter</i> | Current millisecond counter.                  |

Here is the caller graph for this function:



### 6.39.3 Variable Documentation

#### 6.39.3.1 led\_left

```
LED led_left = { .port = LED_LEFT_GPIO_Port, .pin = LED_LEFT_Pin, .mode = LED_MODE_OFF }
```

#### 6.39.3.2 led\_right

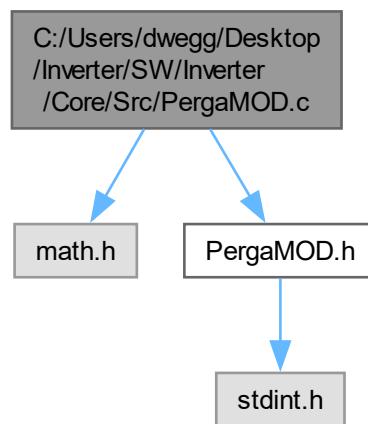
```
LED led_right = { .port = LED_RIGHT_GPIO_Port, .pin = LED_RIGHT_Pin, .mode = LED_MODE_OFF }
```

#### 6.39.3.3 ledError

```
LED ledError = { .port = LED_ERR_GPIO_Port, .pin = LED_ERR_Pin, .mode = LED_MODE_OFF }
```

## 6.40 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PergaMOD.c File Reference

```
#include <math.h>
#include <PergaMOD.h>
Include dependency graph for PergaMOD.c:
```



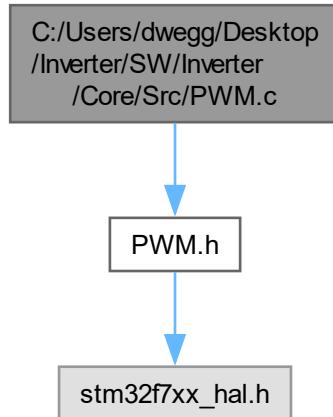
## Functions

- void `pi_aw_calc` (volatile `pi_aw_struct` \*v)  
*Calculates the Proportional-Integral (PI) control with anti-windup.*
- void `pi_init` (volatile `pi_struct` \*v)  
*Initializes the constants of the Proportional-Integral (PI) controller.*
- void `pi_calc` (volatile `pi_struct` \*v)  
*Calculates the Proportional-Integral (PI) control action with feedforward and saturation.*
- void `pi_extsat_calc` (volatile `pi_struct` \*v)  
*Calculates the Proportional-Integral (PI) control action without saturation for external saturation.*
- void `clarke3F_calc` (volatile `clarke3F_struct` \*v)  
*Calculates the Clarke transformation for three-phase signals.*
- void `iclarke3F_calc` (volatile `iclarke3F_struct` \*v)  
*Calculates the inverse Clarke transformation for three-phase signals.*
- void `rot_calc` (volatile `rot_struct` \*v)  
*Calculates the rotation transformation (clockwise).*
- void `irot_calc` (volatile `irot_struct` \*v)  
*Calculates the inverse rotation transformation (counterclockwise).*
- void `angle_calc` (volatile `angle_struct` \*v)  
*Generates the angle.*
- void `svpwm_calc` (volatile `svpwm_struct` \*v)  
*Calculates the Space Vector Pulse Width Modulation (SVPWM).*
- void `rampa_calc` (volatile `rampa_struct` \*v)  
*Calculates the ramp.*
- void `rampa_dual_calc` (volatile `rampa_dual_struct` \*v)  
*Calculates the dual ramp.*
- void `datalog_calc` (volatile `datalog_struct` \*dl)  
*Calculates the data log.*
- void `filtreLP_init` (volatile `filtreLP_struct` \*v)  
*Initializes the first-order filter.*
- void `filtreLP_calc` (volatile `filtreLP_struct` \*v)  
*Calculates the first-order filter.*
- void `avg_calc_10_samples` (volatile `avg_struct_10` \*v)  
*Calculates the average of 10 samples.*
- void `RMS_calc` (volatile `RMS_struct` \*v)  
*Calculates the Root Mean Square (RMS).*
- void `step_calc` (volatile `step_struct` \*v)  
*Calculates the step function.*

## 6.41 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PWM.c File Reference

This file provides functions for controlling PWM output.

```
#include "PWM.h"
Include dependency graph for PWM.c:
```



## Functions

- void `enable_PWM` (TIM\_HandleTypeDef \*htim)  
*Enable PWM output.*
- void `disable_PWM` (TIM\_HandleTypeDef \*htim)  
*Disable PWM output.*
- void `update_PWM` (TIM\_HandleTypeDef \*htim, **Duties** duties)  
*Set PWM duty cycles.*

### 6.41.1 Detailed Description

This file provides functions for controlling PWM output.

#### Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

### 6.41.2 Function Documentation

#### 6.41.2.1 disable\_PWM()

```
void disable_PWM (
 TIM_HandleTypeDef * htim)
```

Disable PWM output.

This function disables PWM output for the specified timer.

**Parameters**

|             |                                             |
|-------------|---------------------------------------------|
| <i>htim</i> | Pointer to the TIM_HandleTypeDef structure. |
|-------------|---------------------------------------------|

**6.41.2.2 enable\_PWM()**

```
void enable_PWM (
 TIM_HandleTypeDef * htim)
```

Enable PWM output.

This function enables PWM output for the specified timer.

**Parameters**

|             |                                             |
|-------------|---------------------------------------------|
| <i>htim</i> | Pointer to the TIM_HandleTypeDef structure. |
|-------------|---------------------------------------------|

**6.41.2.3 update\_PWM()**

```
void update_PWM (
 TIM_HandleTypeDef * htim,
 Duties duties)
```

Set PWM duty cycles.

This function sets the duty cycles for the PWM channels.

**Parameters**

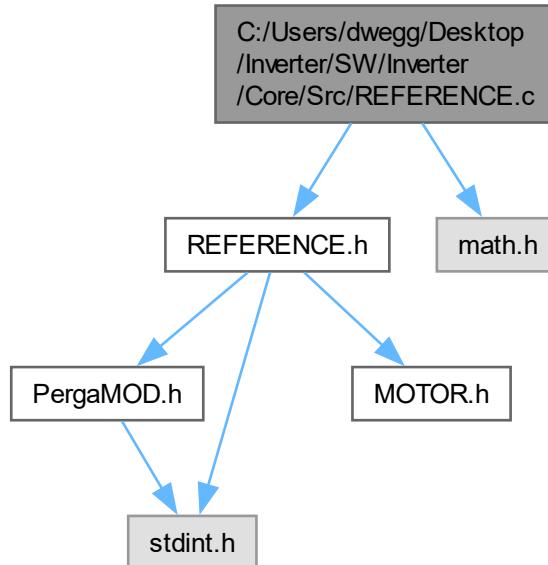
|               |                                                |
|---------------|------------------------------------------------|
| <i>htim</i>   | Pointer to the TIM_HandleTypeDef structure.    |
| <i>duties</i> | Duties structure containing duty cycle values. |

Here is the caller graph for this function:

**6.42 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/REFERENCE.c File Reference**

Source file for torque reference handling.

```
#include "REFERENCE.h"
#include <math.h>
Include dependency graph for REFERENCE.c:
```



## Functions

- float `handle_torqueRef` (float torqueRefIn, int8\_t direction, float torqueMax, float speedMaxRPM, float speedMeas, volatile `pi_struct` \*loopSpeed)
 

*Handles torque control based on the reference torque, direction, maximum torque, maximum speed, measured speed, maximum torque rate of change, speed control loop parameters, and sampling time.*
- void `initialize_torque_ramp` (float dTorqueMax, float Ts)
 

*Initializes torque ramp.*
- float `set_torque_direction` (float torqueRefIn, int8\_t direction)
 

*Set torque direction based on inverter direction.*
- float `saturate_symmetric` (float ref, float max)
 

*Symmetrically saturate a reference value.*
- float `limit_torque_to_prevent_overspeed` (float speedMaxRPM, float speedMeas, float torqueRefIn, volatile `pi_struct` \*loopSpeed)
 

*Speed loop acts as a torque saturation, reducing torque in order to limit the maximum speed.*

### 6.42.1 Detailed Description

Source file for torque reference handling.

## Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

## 6.42.2 Function Documentation

### 6.42.2.1 handle\_torqueRef()

```
float handle_torqueRef (
 float torqueRefIn,
 int8_t direction,
 float torqueMax,
 float speedMaxRPM,
 float speedMeas,
 volatile pi_struct * loopSpeed)
```

Handles torque control based on the reference torque, direction, maximum torque, maximum speed, measured speed, maximum torque rate of change, speed control loop parameters, and sampling time.

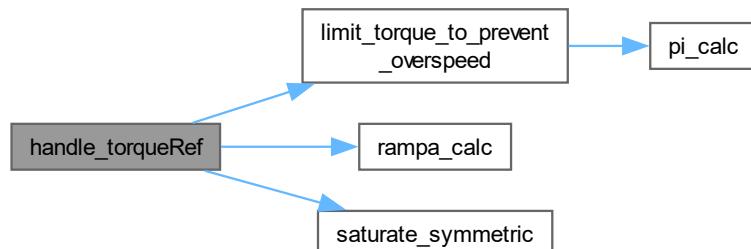
#### Parameters

|                    |                                                                      |
|--------------------|----------------------------------------------------------------------|
| <i>torqueRefIn</i> | Input reference torque.                                              |
| <i>direction</i>   | Direction of torque (1 for positive torque, -1 for negative torque). |
| <i>torqueMax</i>   | Maximum allowable torque.                                            |
| <i>speedMaxRPM</i> | Maximum allowable speed in RPM.                                      |
| <i>speedMeas</i>   | Measured speed.                                                      |
| <i>loopSpeed</i>   | Speed control loop parameters.                                       |

#### Returns

The output torque after handling direction, saturation, and rate limiting.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.42.2.2 initialize\_torque\_ramp()

```
void initialize_torque_ramp (
 float dTorqueMax,
 float Ts)
```

Initializes torque ramp.

##### Parameters

|                   |                                             |
|-------------------|---------------------------------------------|
| <i>dTorqueMax</i> | Maximum allowable rate of change of torque. |
| <i>Ts</i>         | Sampling time.                              |

#### 6.42.2.3 limit\_torque\_to\_prevent\_overspeed()

```
float limit_torque_to_prevent_overspeed (
 float speedMaxRPM,
 float speedMeas,
 float torqueRefIn,
 volatile pi_struct * loopSpeed)
```

Speed loop acts as a torque saturation, reducing torque in order to limit the maximum speed.

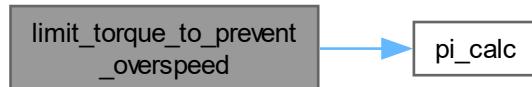
##### Parameters

|    |                    |                                                    |
|----|--------------------|----------------------------------------------------|
| in | <i>speedMaxRPM</i> | The maximum speed value in RPM.                    |
| in | <i>speedMeas</i>   | The measured speed value in RPM.                   |
| in | <i>torqueRefIn</i> | The torque reference value before this saturation. |
| in | <i>loopSpeed</i>   | Pointer to the speed PI controller structure.      |

**Returns**

`torqueRefOut` The limited torque reference value after this saturation.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.42.2.4 `saturate_symmetric()`

```
float saturate_symmetric (
 float ref,
 float max)
```

Symmetrically saturate a reference value.

This function symmetrically saturates a reference value based on the maximum allowed value. If the reference value exceeds the maximum allowed value, it is saturated to the maximum value. If the reference value is less than the negative of the maximum allowed value, it is saturated to the negative of the maximum value.

**Parameters**

|    |                  |                                           |
|----|------------------|-------------------------------------------|
| in | <code>ref</code> | The reference value to saturate.          |
| in | <code>max</code> | The maximum allowed value for saturation. |

**Returns**

The saturated reference value.

Here is the caller graph for this function:

**6.42.2.5 set\_torque\_direction()**

```
float set_torque_direction (
 float torqueRefIn,
 int8_t direction)
```

Set torque direction based on inverter direction.

This function adjusts the torque reference based on the desired direction. If the motor is set to rotate counterclockwise (CCW), positive torque represents traction, negative is braking. If the motor is set to rotate clockwise (CW), negative torque represents traction, positive is braking.

**Parameters**

|    |                    |                                                                  |
|----|--------------------|------------------------------------------------------------------|
| in | <i>torqueRefIn</i> | The torque reference value to adjust.                            |
| in | <i>direction</i>   | Pointer to the direction of the inverter (1 for CW, -1 for CCW). |

**Returns**

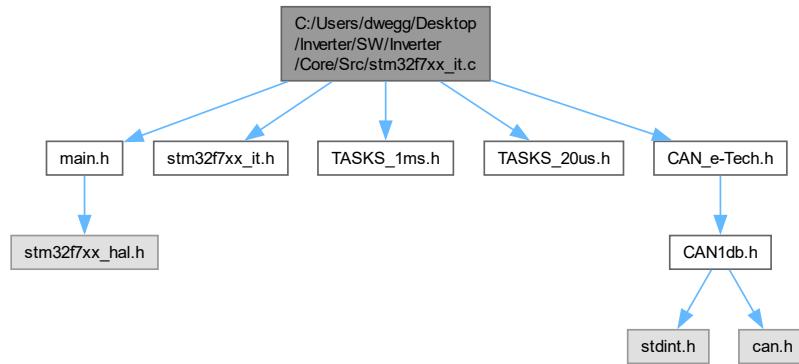
*torqueRefOut* The adjusted torque reference value.

**6.43 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/stm32f7xx\_it.c File Reference**

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f7xx_it.h"
#include "TASKS_1ms.h"
#include "TASKS_20us.h"
#include "CAN_e-Tech.h"
```

Include dependency graph for stm32f7xx\_it.c:



## Functions

- void [NMI\\_Handler](#) (void)
 

*This function handles Non maskable interrupt.*
- void [HardFault\\_Handler](#) (void)
 

*This function handles Hard fault interrupt.*
- void [MemManage\\_Handler](#) (void)
 

*This function handles Memory management fault.*
- void [BusFault\\_Handler](#) (void)
 

*This function handles Pre-fetch fault, memory access fault.*
- void [UsageFault\\_Handler](#) (void)
 

*This function handles Undefined instruction or illegal state.*
- void [SVC\\_Handler](#) (void)
 

*This function handles System service call via SWI instruction.*
- void [DebugMon\\_Handler](#) (void)
 

*This function handles Debug monitor.*
- void [PendSV\\_Handler](#) (void)
 

*This function handles Pendable request for system service.*
- void [SysTick\\_Handler](#) (void)
 

*This function handles System tick timer.*
- void [CAN1\\_RX0\\_IRQHandler](#) (void)
 

*This function handles CAN1 RX0 interrupts.*
- void [TIM1\\_UP\\_TIM10\\_IRQHandler](#) (void)
 

*This function handles TIM1 update interrupt and TIM10 global interrupt.*
- void [TIM6\\_DAC\\_IRQHandler](#) (void)
 

*This function handles TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts.*
- void [DMA2\\_Stream0\\_IRQHandler](#) (void)
 

*This function handles DMA2 stream0 global interrupt.*
- void [DMA2\\_Stream1\\_IRQHandler](#) (void)
 

*This function handles DMA2 stream1 global interrupt.*
- void [DMA2\\_Stream2\\_IRQHandler](#) (void)
 

*This function handles DMA2 stream2 global interrupt.*

## Variables

- DMA\_HandleTypeDef [hdma\\_adc1](#)
- DMA\_HandleTypeDef [hdma\\_adc2](#)
- DMA\_HandleTypeDef [hdma\\_adc3](#)
- CAN\_HandleTypeDef [hcan1](#)
- DAC\_HandleTypeDef [hdac](#)
- TIM\_HandleTypeDef [htim1](#)
- TIM\_HandleTypeDef [htim6](#)

### 6.43.1 Detailed Description

Interrupt Service Routines.

#### Attention

Copyright (c) 2023 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

### 6.43.2 Function Documentation

#### 6.43.2.1 BusFault\_Handler()

```
void BusFault_Handler (
 void)
```

This function handles Pre-fetch fault, memory access fault.

#### 6.43.2.2 CAN1\_RX0\_IRQHandler()

```
void CAN1_RX0_IRQHandler (
 void)
```

This function handles CAN1 RX0 interrupts.

Here is the call graph for this function:



#### 6.43.2.3 DebugMon\_Handler()

```
void DebugMon_Handler (
 void)
```

This function handles Debug monitor.

#### 6.43.2.4 DMA2\_Stream0\_IRQHandler()

```
void DMA2_Stream0_IRQHandler (
 void)
```

This function handles DMA2 stream0 global interrupt.

#### 6.43.2.5 DMA2\_Stream1\_IRQHandler()

```
void DMA2_Stream1_IRQHandler (
 void)
```

This function handles DMA2 stream1 global interrupt.

#### 6.43.2.6 DMA2\_Stream2\_IRQHandler()

```
void DMA2_Stream2_IRQHandler (
 void)
```

This function handles DMA2 stream2 global interrupt.

#### 6.43.2.7 HardFault\_Handler()

```
void HardFault_Handler (
 void)
```

This function handles Hard fault interrupt.

#### 6.43.2.8 MemManage\_Handler()

```
void MemManage_Handler (
 void)
```

This function handles Memory management fault.

#### 6.43.2.9 NMI\_Handler()

```
void NMI_Handler (
 void)
```

This function handles Non maskable interrupt.

#### 6.43.2.10 PendSV\_Handler()

```
void PendSV_Handler (
 void)
```

This function handles Pendable request for system service.

#### 6.43.2.11 SVC\_Handler()

```
void SVC_Handler (
 void)
```

This function handles System service call via SWI instruction.

#### 6.43.2.12 SysTick\_Handler()

```
void SysTick_Handler (
 void)
```

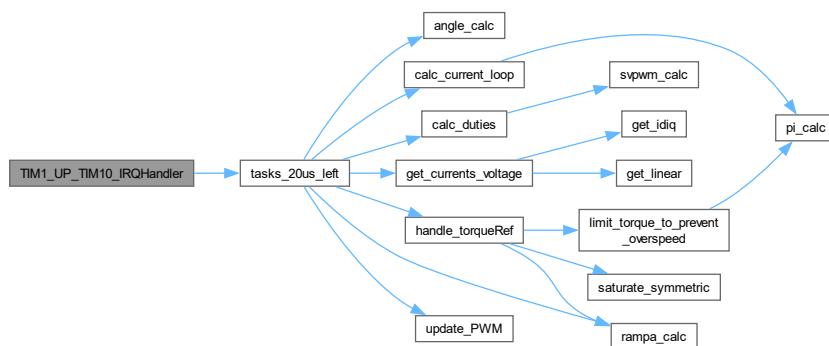
This function handles System tick timer.

#### 6.43.2.13 TIM1\_UP\_TIM10\_IRQHandler()

```
void TIM1_UP_TIM10_IRQHandler (
 void)
```

This function handles TIM1 update interrupt and TIM10 global interrupt.

Here is the call graph for this function:

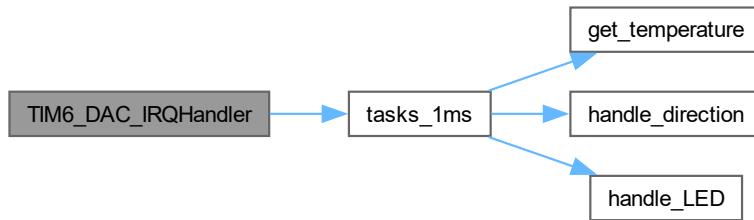


#### 6.43.2.14 TIM6\_DAC\_IRQHandler()

```
void TIM6_DAC_IRQHandler (
 void)
```

This function handles TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts.

Here is the call graph for this function:



#### 6.43.2.15 UsageFault\_Handler()

```
void UsageFault_Handler (
 void)
```

This function handles Undefined instruction or illegal state.

### 6.43.3 Variable Documentation

#### 6.43.3.1 hcan1

```
CAN_HandleTypeDef hcan1 [extern]
```

#### 6.43.3.2 hdac

```
DAC_HandleTypeDef hdac [extern]
```

#### 6.43.3.3 hdma\_adc1

```
DMA_HandleTypeDef hdma_adc1 [extern]
```

#### 6.43.3.4 hdma\_adc2

```
DMA_HandleTypeDef hdma_adc2 [extern]
```

### 6.43.3.5 hdma\_adc3

```
DMA_HandleTypeDef hdma_adc3 [extern]
```

### 6.43.3.6 htim1

```
TIM_HandleTypeDef htim1 [extern]
```

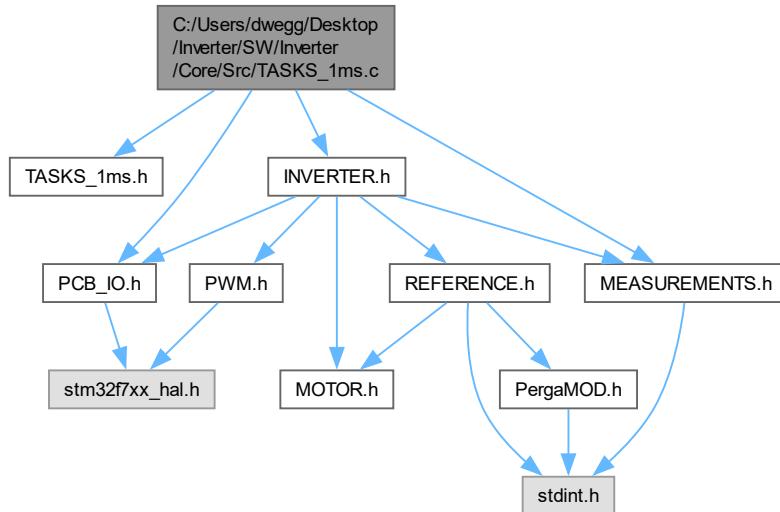
### 6.43.3.7 htim6

```
TIM_HandleTypeDef htim6 [extern]
```

## 6.44 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS\_1ms.c File Reference

This file contains functions to execute tasks every 1ms.

```
#include "TASKS_1ms.h"
#include "PCB_IO.h"
#include "INVERTER.h"
#include "MEASUREMENTS.h"
Include dependency graph for TASKS_1ms.c:
```



### Functions

- void `tasks_1ms` (void)

*Function to be executed every 1ms.*

### 6.44.1 Detailed Description

This file contains functions to execute tasks every 1ms.

#### Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

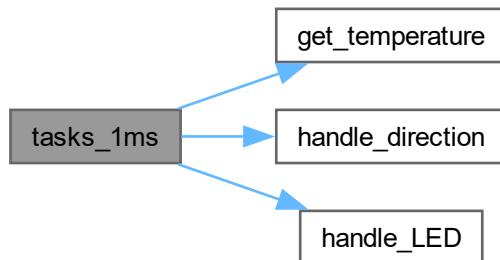
### 6.44.2 Function Documentation

#### 6.44.2.1 tasks\_1ms()

```
void tasks_1ms (
 void)
```

Function to be executed every 1ms.

This function is called by the TIM6 IRQ handler every millisecond. It increments the millisecond counter and executes all the low priority tasks. Here is the call graph for this function:



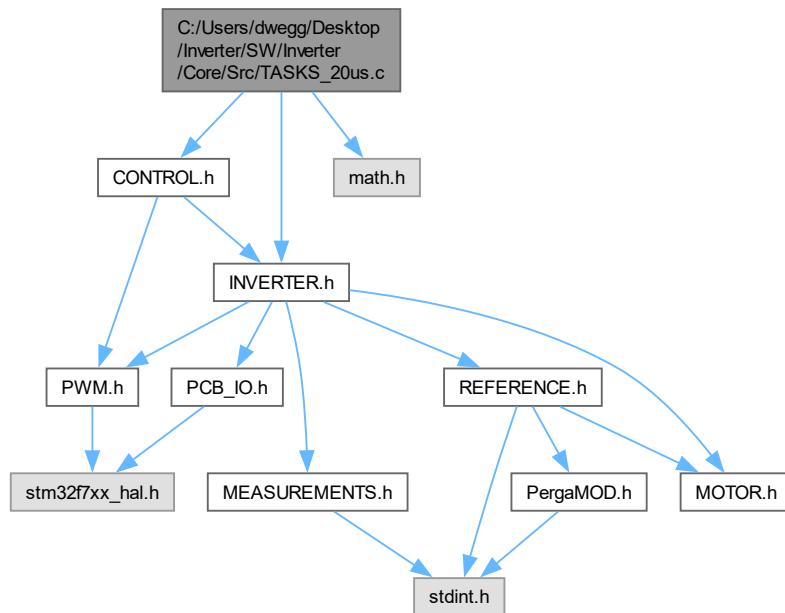
Here is the caller graph for this function:



## 6.45 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS\_20us.c File Reference

This file contains functions executed every 20us in each PWM timer interruption.

```
#include "CONTROL.h"
#include "INVERTER.h"
#include <math.h>
Include dependency graph for TASKS_20us.c:
```



### Functions

- void `tasks_20us_left` (void)  
*Function to be executed every TS.*
- void `tasks_20us_right` (void)  
*Function to be executed every TS.*

### Variables

- float `vd_left` = 0.0F
- float `vq_left` = 7.5F
- float `vDC_left` = 15.0F
- float `torqueRefIn_left` = 0.0F
- uint32\_t `start_ticks` = 0
- uint32\_t `elapsed_ticks` = 0
- `angle_struct angle_left`
- `rampa_struct freqRamp_left`

## 6.45.1 Detailed Description

This file contains functions executed every 20us in each PWM timer interruption.

### Attention

Copyright (c) 2024 David Redondo (@dweggg in GitHub). All rights reserved.

This software is licensed under the Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. For more information, see: <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>

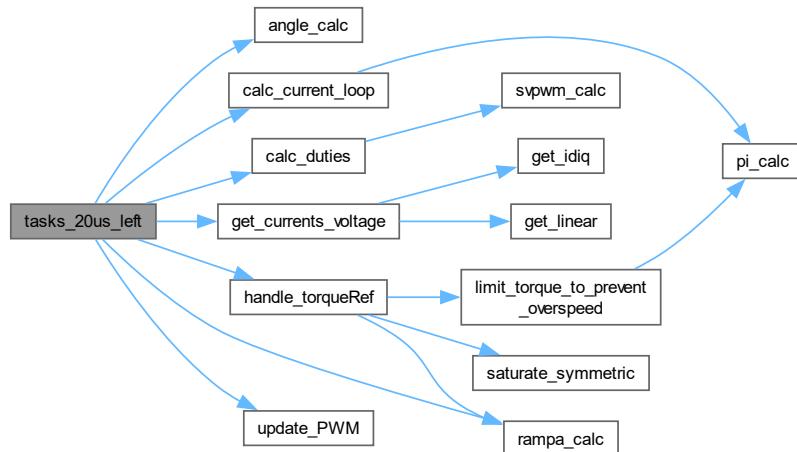
## 6.45.2 Function Documentation

### 6.45.2.1 tasks\_20us\_left()

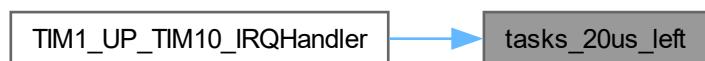
```
void tasks_20us_left (
 void)
```

Function to be executed every TS.

This function is called by the TIM1 trigger handler every TS. Here is the call graph for this function:



Here is the caller graph for this function:



### 6.45.2.2 tasks\_20us\_right()

```
void tasks_20us_right (
 void)
```

Function to be executed every TS.

This function is called by the TIM8 trigger handler every TS.

## 6.45.3 Variable Documentation

### 6.45.3.1 angle\_left

```
angle_struct angle_left
```

#### Initial value:

```
= {
 .freq = 0.0F,
 .Ts = TS,
}
```

### 6.45.3.2 elapsed\_ticks

```
uint32_t elapsed_ticks = 0
```

### 6.45.3.3 freqRamp\_left

```
rampa_struct freqRamp_left
```

#### Initial value:

```
= {
 .in = 5.0F,
 .Incr = TS,
}
```

### 6.45.3.4 start\_ticks

```
uint32_t start_ticks = 0
```

### 6.45.3.5 torqueRefIn\_left

```
float torqueRefIn_left = 0.0F
```

### 6.45.3.6 vd\_left

```
float vd_left = 0.0F
```

### 6.45.3.7 vDC\_left

```
float vDC_left = 15.0F
```

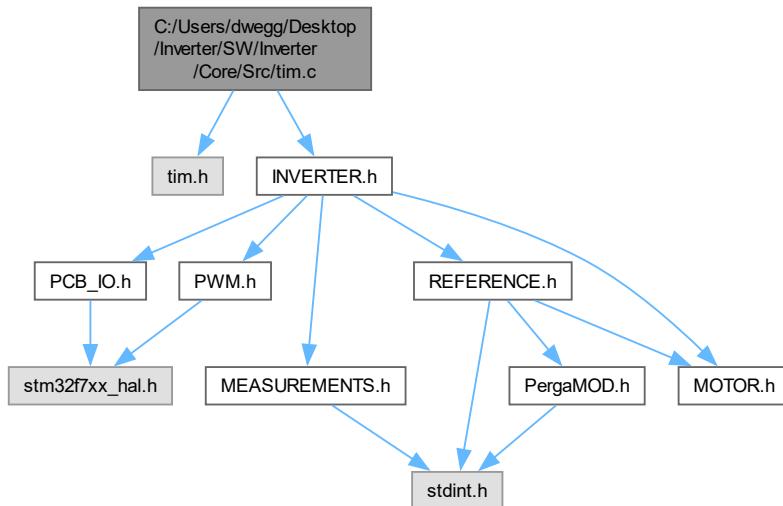
### 6.45.3.8 vq\_left

```
float vq_left = 7.5F
```

## 6.46 C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/tim.c File Reference

This file provides code for the configuration of the TIM instances.

```
#include "tim.h"
#include "INVERTER.h"
Include dependency graph for tim.c:
```



## Functions

- void [MX\\_TIM1\\_Init](#) (void)
- void [MX\\_TIM2\\_Init](#) (void)
- void [MX\\_TIM4\\_Init](#) (void)
- void [MX\\_TIM6\\_Init](#) (void)
- void [MX\\_TIM8\\_Init](#) (void)
- void [HAL\\_TIM\\_Base\\_MspInit](#) (TIM\_HandleTypeDef \*tim\_baseHandle)
- void [HAL\\_TIM\\_IC\\_MspInit](#) (TIM\_HandleTypeDef \*tim\_icHandle)
- void [HAL\\_TIM\\_MspPostInit](#) (TIM\_HandleTypeDef \*timHandle)
- void [HAL\\_TIM\\_Base\\_MspDeInit](#) (TIM\_HandleTypeDef \*tim\_baseHandle)
- void [HAL\\_TIM\\_IC\\_MspDeInit](#) (TIM\_HandleTypeDef \*tim\_icHandle)

## Variables

- TIM\_HandleTypeDef [htim1](#)
- TIM\_HandleTypeDef [htim2](#)
- TIM\_HandleTypeDef [htim4](#)
- TIM\_HandleTypeDef [htim6](#)
- TIM\_HandleTypeDef [htim8](#)

### 6.46.1 Detailed Description

This file provides code for the configuration of the TIM instances.

#### Attention

Copyright (c) 2024 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

### 6.46.2 Function Documentation

#### 6.46.2.1 HAL\_TIM\_Base\_MspDeInit()

```
void HAL_TIM_Base_MspDeInit (
 TIM_HandleTypeDef * tim_baseHandle)
```

Uncomment the line below to disable the "TIM6\_DAC\_IRQHandler" interrupt Be aware, disabling shared interrupt may affect other IPs

#### 6.46.2.2 HAL\_TIM\_Base\_MspInit()

```
void HAL_TIM_Base_MspInit (
 TIM_HandleTypeDef * tim_baseHandle)
```

#### 6.46.2.3 HAL\_TIM\_IC\_MspDeInit()

```
void HAL_TIM_IC_MspDeInit (
 TIM_HandleTypeDef * tim_icHandle)
```

TIM2 GPIO Configuration PB10 -----> TIM2\_CH3 PA15 -----> TIM2\_CH1

TIM4 GPIO Configuration PD12 -----> TIM4\_CH1 PD14 -----> TIM4\_CH3

#### 6.46.2.4 HAL\_TIM\_IC\_MspInit()

```
void HAL_TIM_IC_MspInit (
 TIM_HandleTypeDef * tim_icHandle)
```

TIM2 GPIO Configuration PB10 -----> TIM2\_CH3 PA15 -----> TIM2\_CH1

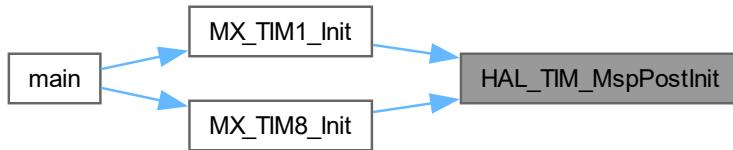
TIM4 GPIO Configuration PD12 -----> TIM4\_CH1 PD14 -----> TIM4\_CH3

#### 6.46.2.5 HAL\_TIM\_MspPostInit()

```
void HAL_TIM_MspPostInit (
 TIM_HandleTypeDef * timHandle)
```

TIM1 GPIO Configuration PE8 -----> TIM1\_CH1N PE9 -----> TIM1\_CH1 PE10 -----> TIM1\_CH2N PE11 -----> TIM1\_CH2 PE12 -----> TIM1\_CH3N PE13 -----> TIM1\_CH3

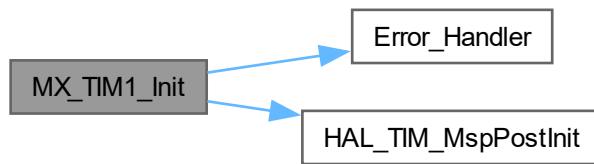
TIM8 GPIO Configuration PA5 -----> TIM8\_CH1N PB14 -----> TIM8\_CH2N PB15 -----> TIM8\_CH3N PC6 -----> TIM8\_CH1 PC7 -----> TIM8\_CH2 PC8 -----> TIM8\_CH3



#### 6.46.2.6 MX\_TIM1\_Init()

```
void MX_TIM1_Init (
 void)
```

Here is the call graph for this function:



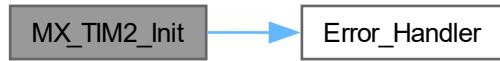
Here is the caller graph for this function:



#### 6.46.2.7 MX\_TIM2\_Init()

```
void MX_TIM2_Init (
 void)
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.46.2.8 MX\_TIM4\_Init()

```
void MX_TIM4_Init (
 void)
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.46.2.9 MX\_TIM6\_Init()

```
void MX_TIM6_Init (
 void)
```

Here is the call graph for this function:



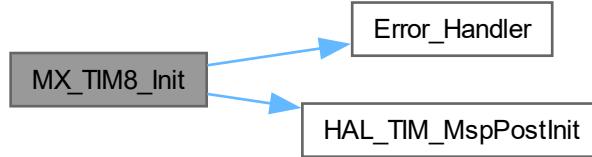
Here is the caller graph for this function:



#### 6.46.2.10 MX\_TIM8\_Init()

```
void MX_TIM8_Init (
 void)
```

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.46.3 Variable Documentation

#### 6.46.3.1 htim1

```
TIM_HandleTypeDef htim1
```

#### 6.46.3.2 htim2

```
TIM_HandleTypeDef htim2
```

#### 6.46.3.3 htim4

```
TIM_HandleTypeDef htim4
```

#### 6.46.3.4 htim6

```
TIM_HandleTypeDef htim6
```

#### 6.46.3.5 htim8

```
TIM_HandleTypeDef htim8
```

# Index

- Integral Controllers, 9
  - pi\_aw\_calc, 9
  - pi\_calc, 10
  - pi\_extsat\_calc, 10
  - pi\_init, 11
- \_\_attribute\_\_
  - CAN1db.h, 64
- A
  - Encoder, 29
- a
  - clarke3F\_struct, 25
  - iclarke3F\_struct, 33
- A\_L\_GPIO\_Port
  - main.h, 92
- A\_L\_Pin
  - main.h, 92
- A\_R\_GPIO\_Port
  - main.h, 92
- A\_R\_Pin
  - main.h, 93
- alfa
  - filtreLP\_struct, 31
- alpha
  - irot\_struct, 38
  - svpwm\_struct, 58
- Analog, 21
  - ia, 21
  - ib, 21
  - ic, 21
  - vDC, 22
- analog
  - InverterStruct, 35
- Angle
  - RMS\_struct, 51
- angle
  - angle\_struct, 22
- Angle\_ant
  - RMS\_struct, 51
- angle\_calc
  - Utility Functions, 14
- angle\_left
  - TASKS\_20us.c, 204
- angle\_struct, 22
  - angle, 22
  - calc, 22
  - freq, 22
  - Ts, 23
- attributes
  - signal\_positioned, 55
- avg\_calc\_10\_samples
  - Signal Processing Functions, 16
- avg\_struct\_10, 23
  - in, 23
  - out, 23
- B
  - Encoder, 29
- b
  - clarke3F\_struct, 25
  - iclarke3F\_struct, 33
  - MotorParameters, 40
- B\_L\_GPIO\_Port
  - main.h, 93
- B\_L\_Pin
  - main.h, 93
- B\_R\_GPIO\_Port
  - main.h, 93
- B\_R\_Pin
  - main.h, 93
- beta
  - irot\_struct, 38
  - svpwm\_struct, 58
- BusFault\_Handler
  - stm32f7xx\_it.c, 196
  - stm32f7xx\_it.h, 134
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN1db.h, 61, 67
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CAN\_e-Tech.h, 74, 78
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/CONTROL.h, 78, 81
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/FSM.h, 82, 84
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/INVERTER.h, 84, 89
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/main.h, 90, 102
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MEASUREMENTS, 103, 110
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/MOTOR.h, 111, 113
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PCB\_IO.h, 113, 117
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PergaMOD.h, 118, 120
- C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/PWM.h, 123, 126

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/REFERENCE.c, 148  
 126, 132  
 CONTROL.h, 80

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/stm32f7xx\_it.h, 132, 137  
 HAL\_CAN\_MspDeInit, 143

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS\_H.c, 143  
 138, 139  
 hcan1, 143

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Inc/TASKS\_MOTOR.c, 143  
 140, 142  
 CAN1\_RX0\_IRQHandler

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/can.c, 196  
 stm32f7xx\_it.h, 134

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CAN\_Idb.h  
 Tech.c, 144  
 \_\_attribute\_\_, 64

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/CONTROLD\_C\_ID\_EXT, 63  
 146  
 CAN\_ID\_STD, 63

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/FSM.c, DBC\_linkMsgConverter, 63  
 149  
 DBC\_setup, 63

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/INVERTER.c, 64  
 150  
 dbc\_valueType, 64  
 dbc\_valueType\_double, 64

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/main.c, dbc\_valueType\_float, 64  
 154  
 dbc\_valueType\_signed, 64

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MEASUREMENTS.c, 64  
 157  
 dbc\_valueType\_unsigned, 64  
 dbc\_valueType\_void, 64

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/MOTORDCBus\_RL, 65  
 181  
 DCBus\_RR, 65

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PCB\_IDT\_C, 65  
 184  
 Enable\_Inv\_R, 65

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PergaMsg.c, 65  
 186  
 getSig, 65  
 getSigVal, 65

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/PWM.c, 65  
 187  
 lactual\_RL, 65

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/REFERENCE.h, 65  
 189  
 lcnd\_RR, 65

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/stm32f7xx\_it.h, 65, 194  
 IDE, 66

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS\_Idq\_RL.c, 66  
 200  
 lq\_RR, 66

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/TASKS\_Power.c, 66  
 202  
 Power\_RR, 66

C:/Users/dwegg/Desktop/Inverter/SW/Inverter/Core/Src/tim.c, 64  
 205  
 signal, 64  
 signal\_positioned, 64  
 SpeedMotor\_RL, 66  
 SpeedMotor\_RR, 66  
 TempIGBT\_RL, 66  
 TempIGBT\_RR, 66  
 TempMotor\_RL, 66  
 TempMotor\_RR, 67  
 TorqueReal\_RL, 67  
 TorqueReal\_RR, 67  
 TotalPower, 67

calc  
 angle\_struct, 22  
 clarke3F\_struct, 25  
 datalog\_struct, 26  
 filtreLP\_struct, 31  
 iclarke3F\_struct, 33  
 irot\_struct, 38  
 pi\_aw\_struct, 42  
 pi\_struct, 45  
 rampa\_dual\_struct, 48  
 rampa\_struct, 49  
 rot\_struct, 52  
 step\_struct, 56  
 svpwm\_struct, 58

calc\_current\_loop  
 CONTROL.c, 147  
 CONTROL.h, 80

calc\_duties  
 CAN\_e-Tech.c  
 enableCAN, 146  
 handle\_CAN, 145  
 send\_CAN\_message, 145

CAN\_e-Tech.h  
 enableCAN, 78  
 handle\_CAN, 76  
 send\_CAN\_message, 77

CAN\_ID\_EXT

CAN1db.h, 63  
CAN\_ID\_STD  
  CAN1db.h, 63  
CANMessageInfo, 24  
  DLC, 24  
  getSig, 24  
  ID, 24  
  IDE, 24  
check\_motor\_parameters  
  MOTOR.c, 182  
  MOTOR.h, 112  
Clarke and Park Transformations, 11  
  clarke3F\_calc, 12  
  iclarke3F\_calc, 12  
  irot\_calc, 13  
  rot\_calc, 13  
clarke3F\_calc  
  Clarke and Park Transformations, 12  
clarke3F\_struct, 25  
  a, 25  
  b, 25  
  calc, 25  
  D, 25  
  Q, 26  
CONTROL.c  
  calc\_current\_loop, 147  
  calc\_duties, 148  
CONTROL.h  
  calc\_current\_loop, 80  
  calc\_duties, 80  
cosFi  
  irot\_struct, 38  
  rot\_struct, 52  
cosTheta\_e  
  Encoder, 29  
Counter  
  step\_struct, 56  
CURRENT\_OFFSET  
  MEASUREMENTS.h, 105  
CURRENT\_SLOPE  
  MEASUREMENTS.h, 105

D

  clarke3F\_struct, 25  
  iclarke3F\_struct, 33  
  rot\_struct, 53

d

  irot\_struct, 38  
  rot\_struct, 53

Da

  Duties, 28  
  svpwm\_struct, 58

DAC\_GPIO\_Port  
  main.h, 93

DAC\_Pin  
  main.h, 93

datalog\_calc  
  Signal Processing Functions, 17

datalog\_struct, 26

  calc, 26  
  estat, 26  
  i, 26  
  j, 27  
  log, 27  
  prescaler, 27  
  var, 27

Db

  Duties, 28  
  svpwm\_struct, 58

DBC\_linkMsgConverter  
  CAN1db.h, 63

DBC\_setup  
  CAN1db.h, 63

dbc\_valueType  
  CAN1db.h, 64

dbc\_valueType\_double  
  CAN1db.h, 64

dbc\_valueType\_float  
  CAN1db.h, 64

dbc\_valueType\_signed  
  CAN1db.h, 64

dbc\_valueType\_unsigned  
  CAN1db.h, 64

dbc\_valueType\_void  
  CAN1db.h, 64

Dc

  Duties, 28  
  svpwm\_struct, 59

DCBus\_RL  
  CAN1db.h, 65

DCBus\_RR  
  CAN1db.h, 65

DebugMon\_Handler  
  stm32f7xx\_it.c, 196  
  stm32f7xx\_it.h, 134

Decr  
  rampa\_dual\_struct, 48

DIR\_GPIO\_Port  
  main.h, 93

DIR\_Pin  
  main.h, 93

DIR\_STATE  
  PCB\_IO.h, 115

direction  
  InverterStruct, 35

directionMeas  
  Encoder, 29

DISABLE  
  PCB\_IO.h, 115

disable\_PWM  
  PWM.c, 188  
  PWM.h, 125

DIV2  
  Math Constants, 7

DLC

  CAN1db.h, 65  
  CANMessageInfo, 24

DMA2\_Stream0\_IRQHandler  
 stm32f7xx\_it.c, 197  
 stm32f7xx\_it.h, 134

DMA2\_Stream1\_IRQHandler  
 stm32f7xx\_it.c, 197  
 stm32f7xx\_it.h, 135

DMA2\_Stream2\_IRQHandler  
 stm32f7xx\_it.c, 197  
 stm32f7xx\_it.h, 135

DT  
 INVERTER.h, 86

dTorque\_max  
 MotorParameters, 40

Duties, 27  
 Da, 28  
 Db, 28  
 Dc, 28

duties  
 InverterStruct, 35

e  
 pi\_aw\_struct, 42  
 pi\_struct, 45

elapsed\_ticks  
 TASKS\_20us.c, 204

ENABLE  
 PCB\_IO.h, 115

enable  
 filtreLP\_struct, 32  
 pi\_aw\_struct, 43  
 pi\_struct, 45  
 rampa\_dual\_struct, 48  
 rampa\_struct, 49  
 step\_struct, 56

Enable\_Inv\_R  
 CAN1db.h, 65

ENABLE\_L\_GPIO\_Port  
 main.h, 93

ENABLE\_L\_Pin  
 main.h, 94

enable\_pin  
 InverterStruct, 35

enable\_port  
 InverterStruct, 35

enable\_PWM  
 PWM.c, 189  
 PWM.h, 125

ENABLE\_R\_GPIO\_Port  
 main.h, 94

ENABLE\_R\_Pin  
 main.h, 94

enableCAN  
 CAN\_e-Tech.c, 146  
 CAN\_e-Tech.h, 78

Encoder, 28  
 A, 29  
 B, 29  
 cosTheta\_e, 29  
 directionMeas, 29

sinTheta\_e, 29  
 theta\_e, 29  
 we, 29  
 Z, 30

encoder  
 InverterStruct, 35

Error\_Handler  
 main.c, 155  
 main.h, 101

estat  
 datalog\_struct, 26

eval\_inv\_FSM  
 FSM.c, 150  
 FSM.h, 83

factor  
 signal, 54

fc  
 filtreLP\_struct, 32

Feedback, 30  
 idMeas, 30  
 iqMeas, 30  
 speedMeas, 30  
 torqueCalc, 31

feedback  
 InverterStruct, 35

filtreLP\_calc  
 Signal Processing Functions, 17

filtreLP\_init  
 Signal Processing Functions, 18

filtreLP\_struct, 31  
 alfa, 31  
 calc, 31  
 enable, 32  
 fc, 32  
 in, 32  
 init, 32  
 out, 32  
 Ts, 32

Freq  
 RMS\_struct, 51

freq  
 angle\_struct, 22

freqRamp\_left  
 TASKS\_20us.c, 204

fs  
 step\_struct, 57

FSM.c  
 eval\_inv\_FSM, 150

FSM.h  
 eval\_inv\_FSM, 83

get\_currents\_voltage  
 MEASUREMENTS.c, 167  
 MEASUREMENTS.h, 106

get\_idiq  
 MEASUREMENTS.c, 168  
 MEASUREMENTS.h, 107

get\_linear

MEASUREMENTS.c, 169  
MEASUREMENTS.h, 108  
get\_temperature  
    MEASUREMENTS.c, 169  
    MEASUREMENTS.h, 108  
getSig  
    CAN1db.h, 65  
    CANMessageInfo, 24  
getSigVal  
    CAN1db.h, 65  
hadc  
    InverterStruct, 36  
HAL\_CAN\_MspDeInit  
    can.c, 143  
HAL\_CAN\_MspInit  
    can.c, 143  
HAL\_TIM\_Base\_MspDeInit  
    tim.c, 206  
HAL\_TIM\_Base\_MspInit  
    tim.c, 206  
HAL\_TIM\_IC\_MspDeInit  
    tim.c, 206  
HAL\_TIM\_IC\_MspInit  
    tim.c, 206  
HAL\_TIM\_MspPostInit  
    tim.c, 207  
handle\_CAN  
    CAN\_e-Tech.c, 145  
    CAN\_e-Tech.h, 76  
handle\_direction  
    PCB\_IO.c, 185  
    PCB\_IO.h, 116  
handle\_LED  
    PCB\_IO.c, 185  
    PCB\_IO.h, 116  
handle\_torqueRef  
    REFERENCE.c, 191  
    REFERENCE.h, 128  
HardFault\_Handler  
    stm32f7xx\_it.c, 197  
    stm32f7xx\_it.h, 135  
hcan1  
    can.c, 143  
    stm32f7xx\_it.c, 199  
hdac  
    stm32f7xx\_it.c, 199  
hdma\_adc1  
    stm32f7xx\_it.c, 199  
hdma\_adc2  
    stm32f7xx\_it.c, 199  
hdma\_adc3  
    stm32f7xx\_it.c, 199  
htim  
    InverterStruct, 36  
htim1  
    stm32f7xx\_it.c, 200  
    tim.c, 210  
htim2  
    tim.c, 210  
htim4  
    tim.c, 210  
htim6  
    stm32f7xx\_it.c, 200  
    tim.c, 210  
htim8  
    tim.c, 210  
i  
    datalog\_struct, 26  
ia  
    Analog, 21  
ia\_L\_GPIO\_Port  
    main.h, 94  
ia\_L\_Pin  
    main.h, 94  
ia\_R\_GPIO\_Port  
    main.h, 94  
ia\_R\_Pin  
    main.h, 94  
lactual\_RL  
    CAN1db.h, 65  
lactual\_RR  
    CAN1db.h, 65  
ib  
    Analog, 21  
ib\_L\_GPIO\_Port  
    main.h, 94  
ib\_L\_Pin  
    main.h, 94  
ib\_R\_GPIO\_Port  
    main.h, 94  
ib\_R\_Pin  
    main.h, 95  
ic  
    Analog, 21  
ic\_L\_GPIO\_Port  
    main.h, 95  
ic\_L\_Pin  
    main.h, 95  
ic\_R\_GPIO\_Port  
    main.h, 95  
ic\_R\_Pin  
    main.h, 95  
iclarke3F\_calc  
    Clarke and Park Transformations, 12  
iclarke3F\_struct, 33  
    a, 33  
    b, 33  
    calc, 33  
    D, 33  
    Q, 33  
lcmd\_RL  
    CAN1db.h, 65  
lcmd\_RR  
    CAN1db.h, 65  
ID  
    CAN1db.h, 65

CANMessageInfo, 24  
**IDE**  
 CAN1db.h, 66  
 CANMessageInfo, 24  
**idLoop**  
 InverterStruct, 36  
**idMeas**  
 Feedback, 30  
**idRef**  
 Reference, 50  
**In**  
 step\_struct, 57  
**in**  
 avg\_struct\_10, 23  
 filtreLP\_struct, 32  
 rampa\_dual\_struct, 48  
 rampa\_struct, 49  
**Incr**  
 rampa\_dual\_struct, 48  
 rampa\_struct, 49  
**init**  
 filtreLP\_struct, 32  
 pi\_struct, 45  
**init\_control\_loops**  
 INVERTER.c, 152  
 INVERTER.h, 87  
**initialize\_inverter**  
 INVERTER.c, 152  
 INVERTER.h, 87  
**initialize\_torque\_ramp**  
 REFERENCE.c, 192  
 REFERENCE.h, 129  
**INV3**  
 Math Constants, 7  
**INV\_DEG**  
 Math Constants, 7  
**INV\_STATE\_FAULT**  
 INVERTER.h, 87  
**INV\_STATE\_IDLE**  
 INVERTER.h, 87  
**INV\_STATE\_RUNNING**  
 INVERTER.h, 87  
**INV\_STATE\_STARTUP**  
 INVERTER.h, 87  
**INVERTER.c**  
 init\_control\_loops, 152  
 initialize\_inverter, 152  
 inverter\_left, 153  
 inverter\_right, 153  
**INVERTER.h**  
 DT, 86  
 init\_control\_loops, 87  
 initialize\_inverter, 87  
 INV\_STATE\_FAULT, 87  
 INV\_STATE\_IDLE, 87  
 INV\_STATE\_RUNNING, 87  
 INV\_STATE\_STARTUP, 87  
 inverter\_left, 89  
 inverter\_right, 89  
 InverterState, 86  
 TS, 86  
**inverter\_left**  
 INVERTER.c, 153  
 INVERTER.h, 89  
**inverter\_right**  
 INVERTER.c, 153  
 INVERTER.h, 89  
**InverterState**  
 INVERTER.h, 86  
**InverterStruct**, 34  
 analog, 35  
 direction, 35  
 duties, 35  
 enable\_pin, 35  
 enable\_port, 35  
 encoder, 35  
 feedback, 35  
 hadc, 36  
 htim, 36  
 idLoop, 36  
 iqLoop, 36  
 led, 36  
 motor, 36  
 reference, 36  
 speedLoop, 36  
 state, 37  
 tempInverter, 37  
 tempMotor, 37  
**iPhase\_pk\_max**  
 MotorParameters, 40  
**IPI**  
 Math Constants, 8  
**IPI2**  
 Math Constants, 8  
**Iq\_RL**  
 CAN1db.h, 66  
**Iq\_RR**  
 CAN1db.h, 66  
**iqLoop**  
 InverterStruct, 36  
**iqMeas**  
 Feedback, 30  
**iqRef**  
 Reference, 50  
**irot\_calc**  
 Clarke and Park Transformations, 13  
**irot\_struct**, 37  
 alpha, 38  
 beta, 38  
 calc, 38  
 cosFi, 38  
 d, 38  
 q, 38  
 sinFi, 38  
**ISQ2**  
 Math Constants, 8

ISQ3  
    Math Constants, 8

J  
    MotorParameters, 40

j  
    datalog\_struct, 27

K0  
    pi\_struct, 46

K1  
    pi\_struct, 46

Kaw  
    pi\_aw\_struct, 43

Ki  
    pi\_aw\_struct, 43  
    pi\_struct, 46

Kp  
    pi\_aw\_struct, 43  
    pi\_struct, 46

lambda  
    MotorParameters, 40

Ld  
    MotorParameters, 41

LED, 39  
    mode, 39  
    pin, 39  
    port, 39

led  
    InverterStruct, 36

LED\_ERR\_GPIO\_Port  
    main.h, 95

LED\_ERR\_Pin  
    main.h, 95

led\_left  
    PCB\_IO.c, 186  
    PCB\_IO.h, 117

LED\_LEFT\_GPIO\_Port  
    main.h, 95

LED\_LEFT\_Pin  
    main.h, 95

LED\_MODE\_BLINK\_FAST  
    PCB\_IO.h, 116

LED\_MODE\_BLINK\_SLOW  
    PCB\_IO.h, 116

LED\_MODE\_OFF  
    PCB\_IO.h, 116

LED\_MODE\_ON  
    PCB\_IO.h, 116

led\_right  
    PCB\_IO.c, 186  
    PCB\_IO.h, 117

LED\_RIGHT\_GPIO\_Port  
    main.h, 95

LED\_RIGHT\_Pin  
    main.h, 96

ledError  
    PCB\_IO.c, 186

        PCB\_IO.h, 117

        LEDMode  
            PCB\_IO.h, 115

        lengthBits  
            signal, 54

        limit\_torque\_to\_prevent\_overspeed  
            REFERENCE.c, 192  
            REFERENCE.h, 129

        log  
            datalog\_struct, 27

        Lq  
            MotorParameters, 41

        main  
            main.c, 155

        main.c  
            Error\_Handler, 155  
            main, 155  
            SystemClock\_Config, 156

        main.h  
            A\_L\_GPIO\_Port, 92  
            A\_L\_Pin, 92  
            A\_R\_GPIO\_Port, 92  
            A\_R\_Pin, 93  
            B\_L\_GPIO\_Port, 93  
            B\_L\_Pin, 93  
            B\_R\_GPIO\_Port, 93  
            B\_R\_Pin, 93  
            DAC\_GPIO\_Port, 93  
            DAC\_Pin, 93  
            DIR\_GPIO\_Port, 93  
            DIR\_Pin, 93  
            ENABLE\_L\_GPIO\_Port, 93  
            ENABLE\_L\_Pin, 94  
            ENABLE\_R\_GPIO\_Port, 94  
            ENABLE\_R\_Pin, 94  
            Error\_Handler, 101  
            ia\_L\_GPIO\_Port, 94  
            ia\_L\_Pin, 94  
            ia\_R\_GPIO\_Port, 94  
            ia\_R\_Pin, 94  
            ib\_L\_GPIO\_Port, 94  
            ib\_L\_Pin, 94  
            ib\_R\_GPIO\_Port, 94  
            ib\_R\_Pin, 95  
            ic\_L\_GPIO\_Port, 95  
            ic\_L\_Pin, 95  
            ic\_R\_GPIO\_Port, 95  
            ic\_R\_Pin, 95  
            LED\_ERR\_GPIO\_Port, 95  
            LED\_ERR\_Pin, 95  
            LED\_LEFT\_GPIO\_Port, 95  
            LED\_LEFT\_Pin, 95  
            LED\_RIGHT\_GPIO\_Port, 95  
            LED\_RIGHT\_Pin, 96  
            PWM1\_L\_GPIO\_Port, 96  
            PWM1\_L\_Pin, 96  
            PWM1\_R\_GPIO\_Port, 96  
            PWM1\_R\_Pin, 96

PWM2\_L\_GPIO\_Port, 96  
 PWM2\_L\_Pin, 96  
 PWM2\_R\_GPIO\_Port, 96  
 PWM2\_R\_Pin, 96  
 PWM3\_L\_GPIO\_Port, 96  
 PWM3\_L\_Pin, 97  
 PWM3\_R\_GPIO\_Port, 97  
 PWM3\_R\_Pin, 97  
 PWM4\_L\_GPIO\_Port, 97  
 PWM4\_L\_Pin, 97  
 PWM4\_R\_GPIO\_Port, 97  
 PWM4\_R\_Pin, 97  
 PWM5\_L\_GPIO\_Port, 97  
 PWM5\_L\_Pin, 97  
 PWM5\_R\_GPIO\_Port, 97  
 PWM5\_R\_Pin, 98  
 PWM6\_L\_GPIO\_Port, 98  
 PWM6\_L\_Pin, 98  
 PWM6\_R\_GPIO\_Port, 98  
 PWM6\_R\_Pin, 98  
 SC\_det\_GPIO\_Port, 98  
 SC\_det\_Pin, 98  
 Tinv\_L\_GPIO\_Port, 98  
 Tinv\_L\_Pin, 98  
 Tinv\_R\_GPIO\_Port, 98  
 Tinv\_R\_Pin, 99  
 Tmot\_L\_GPIO\_Port, 99  
 Tmot\_L\_Pin, 99  
 Tmot\_R\_GPIO\_Port, 99  
 Tmot\_R\_Pin, 99  
 TRIP\_L\_GPIO\_Port, 99  
 TRIP\_L\_Pin, 99  
 TRIP\_R\_GPIO\_Port, 99  
 TRIP\_R\_Pin, 99  
 VDC\_L\_GPIO\_Port, 99  
 VDC\_L\_Pin, 100  
 VDC\_R\_GPIO\_Port, 100  
 VDC\_R\_Pin, 100  
 WRN\_L\_GPIO\_Port, 100  
 WRN\_L\_Pin, 100  
 WRN\_R\_GPIO\_Port, 100  
 WRN\_R\_Pin, 100  
 Z\_L\_GPIO\_Port, 100  
 Z\_L\_Pin, 100  
 Z\_R\_GPIO\_Port, 100  
 Z\_R\_Pin, 101  
 Math Constants, 7  
 DIV2, 7  
 INV3, 7  
 INV\_DEG, 7  
 IPI, 8  
 IPI2, 8  
 ISQ2, 8  
 ISQ3, 8  
 PI, 8  
 PI2, 8  
 SQ2, 8  
 SQ3, 8

max  
 signal, 54  
 Measure  
 RMS\_struct, 51  
 MEASUREMENTS.c  
 get\_currents\_voltage, 167  
 get\_idiq, 168  
 get\_linear, 169  
 get\_temperature, 169  
 rawADC\_left, 170  
 rawADC\_right, 170  
 rawADC\_temp, 170  
 templInverterLUT, 170  
 tempMotorLUT, 176  
 MEASUREMENTS.h  
 CURRENT\_OFFSET, 105  
 CURRENT\_SLOPE, 105  
 get\_currents\_voltage, 106  
 get\_idiq, 107  
 get\_linear, 108  
 get\_temperature, 108  
 rawADC\_left, 109  
 rawADC\_right, 109  
 rawADC\_temp, 109  
 templInverterLUT, 110  
 tempMotorLUT, 110  
 VOLTAGE\_OFFSET, 105  
 VOLTAGE\_SLOPE, 106  
 MemManage\_Handler  
 stm32f7xx\_it.c, 197  
 stm32f7xx\_it.h, 135  
 min  
 signal, 54  
 Miscellaneous Functions, 20  
 step\_calc, 20  
 mode  
 LED, 39  
 motor  
 InverterStruct, 36  
 MOTOR.c  
 check\_motor\_parameters, 182  
 motor\_left, 183  
 motor\_right, 183  
 MOTOR.h  
 check\_motor\_parameters, 112  
 motor\_left, 112  
 motor\_right, 112  
 motor\_left  
 MOTOR.c, 183  
 MOTOR.h, 112  
 motor\_right  
 MOTOR.c, 183  
 MOTOR.h, 112  
 MotorParameters, 39  
 b, 40  
 dTorque\_max, 40  
 iPhase\_pk\_max, 40  
 J, 40

lambda, 40  
Ld, 41  
Lq, 41  
pp, 41  
Rs, 41  
speed\_max\_RPM, 41  
torque\_max, 41  
vDC\_max, 41  
MX\_CAN1\_Init  
can.c, 143  
MX\_TIM1\_Init  
tim.c, 207  
MX\_TIM2\_Init  
tim.c, 208  
MX\_TIM4\_Init  
tim.c, 208  
MX\_TIM6\_Init  
tim.c, 209  
MX\_TIM8\_Init  
tim.c, 209

N\_DATALOG  
Signal Processing Functions, 16

name  
signal, 54

NMI\_Handler  
stm32f7xx\_it.c, 197  
stm32f7xx\_it.h, 135

offset  
signal, 54

Out  
step\_struct, 57

out  
avg\_struct\_10, 23  
filtreLP\_struct, 32  
rampa\_dual\_struct, 48  
rampa\_struct, 49

Out\_RMS  
RMS\_struct, 51

PCB\_IO.c  
handle\_direction, 185  
handle\_LED, 185  
led\_left, 186  
led\_right, 186  
ledError, 186

PCB\_IO.h  
DIR\_STATE, 115  
DISABLE, 115  
ENABLE, 115  
handle\_direction, 116  
handle\_LED, 116  
led\_left, 117  
LED\_MODE\_BLINK\_FAST, 116  
LED\_MODE\_BLINK\_SLOW, 116  
LED\_MODE\_OFF, 116  
LED\_MODE\_ON, 116  
led\_right, 117

ledError, 117  
LEDMode, 115  
SC\_DET\_STATE, 115  
WRN\_STATE, 115

PendSV\_Handler  
stm32f7xx\_it.c, 197  
stm32f7xx\_it.h, 135

PI  
Math Constants, 8

PI2  
Math Constants, 8

pi\_aw\_calc  
- Integral Controllers, 9

pi\_aw\_struct, 42  
calc, 42  
e, 42  
enable, 43  
Kaw, 43  
Ki, 43  
Kp, 43  
pi\_consig, 43  
pi\_fdb, 43  
pi\_ffw, 43  
pi\_int, 43  
pi\_out, 44  
pi\_out\_max, 44  
pi\_out\_min, 44  
pi\_out\_postsat, 44  
pi\_out\_presat, 44  
Ts, 44

pi\_calc  
- Integral Controllers, 10

pi\_consig  
pi\_aw\_struct, 43  
pi\_struct, 46

pi\_extsat\_calc  
- Integral Controllers, 10

pi\_fdb  
pi\_aw\_struct, 43  
pi\_struct, 46

pi\_ffw  
pi\_aw\_struct, 43  
pi\_struct, 46

pi\_init  
- Integral Controllers, 11

pi\_int  
pi\_aw\_struct, 43

pi\_out  
pi\_aw\_struct, 44  
pi\_struct, 46

pi\_out\_max  
pi\_aw\_struct, 44  
pi\_struct, 47

pi\_out\_min  
pi\_aw\_struct, 44  
pi\_struct, 47

pi\_out\_postsat  
pi\_aw\_struct, 44

pi\_out\_presat  
     pi\_aw\_struct, 44  
 pi\_struct, 45  
     calc, 45  
     e, 45  
     enable, 45  
     init, 45  
     K0, 46  
     K1, 46  
     Ki, 46  
     Kp, 46  
     pi\_consig, 46  
     pi\_fdb, 46  
     pi\_ffw, 46  
     pi\_out, 46  
     pi\_out\_max, 47  
     pi\_out\_min, 47  
     Ts, 47  
 pin  
     LED, 39  
 port  
     LED, 39  
 position  
     signal\_positioned, 55  
 Power\_RL  
     CAN1db.h, 66  
 Power\_RR  
     CAN1db.h, 66  
 pp  
     MotorParameters, 41  
 prescaler  
     datalog\_struct, 27  
 Pulses  
     step\_struct, 57  
 PWM.c  
     disable\_PWM, 188  
     enable\_PWM, 189  
     update\_PWM, 189  
 PWM.h  
     disable\_PWM, 125  
     enable\_PWM, 125  
     update\_PWM, 125  
 PWM1\_L\_GPIO\_Port  
     main.h, 96  
 PWM1\_L\_Pin  
     main.h, 96  
 PWM1\_R\_GPIO\_Port  
     main.h, 96  
 PWM1\_R\_Pin  
     main.h, 96  
 PWM2\_L\_GPIO\_Port  
     main.h, 96  
 PWM2\_L\_Pin  
     main.h, 96  
 PWM2\_R\_GPIO\_Port  
     main.h, 96  
 PWM2\_R\_Pin  
     main.h, 96  
 PWM3\_L\_GPIO\_Port  
     main.h, 96  
 PWM3\_L\_Pin  
     main.h, 97  
 PWM3\_R\_GPIO\_Port  
     main.h, 97  
 PWM3\_R\_Pin  
     main.h, 97  
 PWM4\_L\_GPIO\_Port  
     main.h, 97  
 PWM4\_L\_Pin  
     main.h, 97  
 PWM4\_R\_GPIO\_Port  
     main.h, 97  
 PWM4\_R\_Pin  
     main.h, 97  
 PWM5\_L\_GPIO\_Port  
     main.h, 97  
 PWM5\_L\_Pin  
     main.h, 97  
 PWM5\_R\_GPIO\_Port  
     main.h, 97  
 PWM5\_R\_Pin  
     main.h, 98  
 PWM6\_L\_GPIO\_Port  
     main.h, 98  
 PWM6\_L\_Pin  
     main.h, 98  
 PWM6\_R\_GPIO\_Port  
     main.h, 98  
 PWM6\_R\_Pin  
     main.h, 98  
 Q  
     clarke3F\_struct, 26  
     iclarke3F\_struct, 33  
     rot\_struct, 53  
 q  
     irot\_struct, 38  
     rot\_struct, 53  
 rampa\_calc  
     Signal Processing Functions, 18  
 rampa\_dual\_calc  
     Signal Processing Functions, 19  
 rampa\_dual\_struct, 47  
     calc, 48  
     Decr, 48  
     enable, 48  
     in, 48  
     Incr, 48  
     out, 48  
 rampa\_struct, 48  
     calc, 49  
     enable, 49  
     in, 49  
     Incr, 49  
     out, 49  
 rawADC\_left

MEASUREMENTS.c, 170  
MEASUREMENTS.h, 109  
rawADC\_right  
    MEASUREMENTS.c, 170  
    MEASUREMENTS.h, 109  
rawADC\_temp  
    MEASUREMENTS.c, 170  
    MEASUREMENTS.h, 109  
Reference, 50  
    idRef, 50  
    iqRef, 50  
    torqueRef, 50  
reference  
    InverterStruct, 36  
REFERENCE.c  
    handle\_torqueRef, 191  
    initialize\_torque\_ramp, 192  
    limit\_torque\_to\_prevent\_overspeed, 192  
    saturate\_symmetric, 193  
    set\_torque\_direction, 194  
REFERENCE.h  
    handle\_torqueRef, 128  
    initialize\_torque\_ramp, 129  
    limit\_torque\_to\_prevent\_overspeed, 129  
    saturate\_symmetric, 130  
    set\_torque\_direction, 131  
RMS\_calc  
    Signal Processing Functions, 19  
RMS\_struct, 50  
    Angle, 51  
    Angle\_ant, 51  
    Freq, 51  
    Measure, 51  
    Out\_RMS, 51  
    Sq\_Sum, 51  
    T\_exec, 52  
rot\_calc  
    Clarke and Park Transformations, 13  
rot\_struct, 52  
    calc, 52  
    cosFi, 52  
    D, 53  
    d, 53  
    Q, 53  
    q, 53  
    sinFi, 53  
Rs  
    MotorParameters, 41  
saturate\_symmetric  
    REFERENCE.c, 193  
    REFERENCE.h, 130  
SC\_det\_GPIO\_Port  
    main.h, 98  
SC\_det\_Pin  
    main.h, 98  
SC\_DET\_STATE  
    PCB\_IO.h, 115  
send\_CAN\_message  
    CAN\_e-Tech.c, 145  
    CAN\_e-Tech.h, 77  
set\_torque\_direction  
    REFERENCE.c, 194  
    REFERENCE.h, 131  
signal, 53  
    CAN1db.h, 64  
    factor, 54  
    lengthBits, 54  
    max, 54  
    min, 54  
    name, 54  
    offset, 54  
    unit, 54  
    valueType, 55  
Signal Processing Functions, 15  
    avg\_calc\_10\_samples, 16  
    datalog\_calc, 17  
    filtreLP\_calc, 17  
    filtreLP\_init, 18  
    N\_DATALOG, 16  
    rampa\_calc, 18  
    rampa\_dual\_calc, 19  
    RMS\_calc, 19  
    signal\_positioned, 55  
    attributes, 55  
    CAN1db.h, 64  
    position, 55  
sinFi  
    irot\_struct, 38  
    rot\_struct, 53  
sinTheta\_e  
    Encoder, 29  
speed\_max\_RPM  
    MotorParameters, 41  
speedLoop  
    InverterStruct, 36  
speedMeas  
    Feedback, 30  
SpeedMotor\_RL  
    CAN1db.h, 66  
SpeedMotor\_RR  
    CAN1db.h, 66  
SQ2  
    Math Constants, 8  
SQ3  
    Math Constants, 8  
Sq\_Sum  
    RMS\_struct, 51  
start\_ticks  
    TASKS\_20us.c, 204  
state  
    InverterStruct, 37  
Step  
    step\_struct, 57  
step\_calc  
    Miscellaneous Functions, 20  
step\_struct, 56

calc, 56  
 Counter, 56  
 enable, 56  
 fs, 57  
 In, 57  
 Out, 57  
 Pulses, 57  
 Step, 57  
 t\_step, 57  
**stm32f7xx\_it.c**  
     BusFault\_Handler, 196  
     CAN1\_RX0\_IRQHandler, 196  
     DebugMon\_Handler, 196  
     DMA2\_Stream0\_IRQHandler, 197  
     DMA2\_Stream1\_IRQHandler, 197  
     DMA2\_Stream2\_IRQHandler, 197  
     HardFault\_Handler, 197  
     hcan1, 199  
     hdac, 199  
     hdma\_adc1, 199  
     hdma\_adc2, 199  
     hdma\_adc3, 199  
     htim1, 200  
     htim6, 200  
     MemManage\_Handler, 197  
     NMI\_Handler, 197  
     PendSV\_Handler, 197  
     SVC\_Handler, 198  
     SysTick\_Handler, 198  
     TIM1\_UP\_TIM10\_IRQHandler, 198  
     TIM6\_DAC\_IRQHandler, 198  
     UsageFault\_Handler, 199  
**stm32f7xx\_it.h**  
     BusFault\_Handler, 134  
     CAN1\_RX0\_IRQHandler, 134  
     DebugMon\_Handler, 134  
     DMA2\_Stream0\_IRQHandler, 134  
     DMA2\_Stream1\_IRQHandler, 135  
     DMA2\_Stream2\_IRQHandler, 135  
     HardFault\_Handler, 135  
     MemManage\_Handler, 135  
     NMI\_Handler, 135  
     PendSV\_Handler, 135  
     SVC\_Handler, 135  
     SysTick\_Handler, 136  
     TIM1\_UP\_TIM10\_IRQHandler, 136  
     TIM6\_DAC\_IRQHandler, 136  
     UsageFault\_Handler, 137  
**SVC\_Handler**  
     stm32f7xx\_it.c, 198  
     stm32f7xx\_it.h, 135  
**svpwm\_calc**  
     Utility Functions, 15  
**svpwm\_struct**, 58  
     alpha, 58  
     beta, 58  
     calc, 58  
     Da, 58  
**Db**, 58  
**Dc**, 59  
**SystemClock\_Config**  
     main.c, 156  
**SysTick\_Handler**  
     stm32f7xx\_it.c, 198  
     stm32f7xx\_it.h, 136  
**T\_exec**  
     RMS\_struct, 52  
**t\_step**  
     step\_struct, 57  
**tasks\_1ms**  
     TASKS\_1ms.c, 201  
     TASKS\_1ms.h, 139  
**TASKS\_1ms.c**  
     tasks\_1ms, 201  
**TASKS\_1ms.h**  
     tasks\_1ms, 139  
**TASKS\_20us.c**  
     angle\_left, 204  
     elapsed\_ticks, 204  
     freqRamp\_left, 204  
     start\_ticks, 204  
     tasks\_20us\_left, 203  
     tasks\_20us\_right, 203  
     torqueRefIn\_left, 204  
     vd\_left, 204  
     vDC\_left, 204  
     vq\_left, 205  
**TASKS\_20us.h**  
     tasks\_20us\_left, 141  
     tasks\_20us\_right, 141  
**tasks\_20us\_left**  
     TASKS\_20us.c, 203  
     TASKS\_20us.h, 141  
**tasks\_20us\_right**  
     TASKS\_20us.c, 203  
     TASKS\_20us.h, 141  
**TempIGBT\_RL**  
     CAN1db.h, 66  
**TempIGBT\_RR**  
     CAN1db.h, 66  
**tempInverter**  
     InverterStruct, 37  
**tempInverterLUT**  
     MEASUREMENTS.c, 170  
     MEASUREMENTS.h, 110  
**tempMotor**  
     InverterStruct, 37  
**TempMotor\_RL**  
     CAN1db.h, 66  
**TempMotor\_RR**  
     CAN1db.h, 67  
**tempMotorLUT**  
     MEASUREMENTS.c, 176  
     MEASUREMENTS.h, 110  
**theta\_e**  
     Encoder, 29

tim.c  
    HAL\_TIM\_Base\_MspDeInit, 206  
    HAL\_TIM\_Base\_MspInit, 206  
    HAL\_TIM\_IC\_MspDeInit, 206  
    HAL\_TIM\_IC\_MspInit, 206  
    HAL\_TIM\_MspPostInit, 207  
    htim1, 210  
    htim2, 210  
    htim4, 210  
    htim6, 210  
    htim8, 210  
    MX\_TIM1\_Init, 207  
    MX\_TIM2\_Init, 208  
    MX\_TIM4\_Init, 208  
    MX\_TIM6\_Init, 209  
    MX\_TIM8\_Init, 209  
TIM1\_UP\_TIM10\_IRQHandler  
    stm32f7xx\_it.c, 198  
    stm32f7xx\_it.h, 136  
TIM6\_DAC\_IRQHandler  
    stm32f7xx\_it.c, 198  
    stm32f7xx\_it.h, 136  
Tinv\_L\_GPIO\_Port  
    main.h, 98  
Tinv\_L\_Pin  
    main.h, 98  
Tinv\_R\_GPIO\_Port  
    main.h, 98  
Tinv\_R\_Pin  
    main.h, 99  
Tmot\_L\_GPIO\_Port  
    main.h, 99  
Tmot\_L\_Pin  
    main.h, 99  
Tmot\_R\_GPIO\_Port  
    main.h, 99  
Tmot\_R\_Pin  
    main.h, 99  
torque\_max  
    MotorParameters, 41  
torqueCalc  
    Feedback, 31  
TorqueReal\_RL  
    CAN1db.h, 67  
TorqueReal\_RR  
    CAN1db.h, 67  
torqueRef  
    Reference, 50  
torqueRefln\_left  
    TASKS\_20us.c, 204  
TotalPower  
    CAN1db.h, 67  
TRIP\_L\_GPIO\_Port  
    main.h, 99  
TRIP\_L\_Pin  
    main.h, 99  
TRIP\_R\_GPIO\_Port  
    main.h, 99  
TRIP\_R\_Pin  
    main.h, 99  
TS  
    INVERTER.h, 86  
Ts  
    angle\_struct, 23  
    filtreLP\_struct, 32  
    pi\_aw\_struct, 44  
    pi\_struct, 47  
unit  
    signal, 54  
update\_PWM  
    PWM.c, 189  
    PWM.h, 125  
UsageFault\_Handler  
    stm32f7xx\_it.c, 199  
    stm32f7xx\_it.h, 137  
Utility Functions, 14  
    angle\_calc, 14  
    svpwm\_calc, 15  
valueType  
    signal, 55  
var  
    datalog\_struct, 27  
vd\_left  
    TASKS\_20us.c, 204  
vDC  
    Analog, 22  
VDC\_L\_GPIO\_Port  
    main.h, 99  
VDC\_L\_Pin  
    main.h, 100  
vDC\_left  
    TASKS\_20us.c, 204  
vDC\_max  
    MotorParameters, 41  
VDC\_R\_GPIO\_Port  
    main.h, 100  
VDC\_R\_Pin  
    main.h, 100  
VOLTAGE\_OFFSET  
    MEASUREMENTS.h, 105  
VOLTAGE\_SLOPE  
    MEASUREMENTS.h, 106  
vq\_left  
    TASKS\_20us.c, 205  
we  
    Encoder, 29  
WRN\_L\_GPIO\_Port  
    main.h, 100  
WRN\_L\_Pin  
    main.h, 100  
WRN\_R\_GPIO\_Port  
    main.h, 100  
WRN\_R\_Pin  
    main.h, 100

WRN\_STATE  
PCB\_IO.h, [115](#)

Z

Encoder, [30](#)

Z\_L\_GPIO\_Port  
main.h, [100](#)

Z\_L\_Pin  
main.h, [100](#)

Z\_R\_GPIO\_Port  
main.h, [100](#)

Z\_R\_Pin  
main.h, [101](#)

---

## B. Bibliografía

---

- [1] FSG, 2024. URL [https://www.formulastudent.de/fileadmin/user\\_upload/all/2024/rules/FS-Rules\\_2024\\_v1.1.pdf](https://www.formulastudent.de/fileadmin/user_upload/all/2024/rules/FS-Rules_2024_v1.1.pdf).
- [2] Jason Sylvestre. Inverter dc link capacitor selection, Nov 2020. URL <https://www.specterengineering.com/blog/2019/9/7/dc-link-capacitor-selection-for-your-inverter>.