MEJORAS ELECTRÓNICAS Y PROGRAMACIÓN PARA LA PUESTA EN MARCHA DE UN ROBOT DELTA

Trabajo de Fin de Grado

Autor: Sergio Peña Aragón
Tutor: Santiago Torres Álvarez
13 DE JULIO DE 2023
Universidad de La Laguna
Grado en Ingeniería Electrónica Industrial

Agradecimientos

Durante el desarrollo de este Trabajo de Fin de Grado, he contado con el apoyo y la colaboración de diversas personas y entidades a las cuales deseo expresar mi más sincero agradecimiento.

En primer lugar, quiero agradecer a mi tutor, Santiago Torres Álvarez, por su orientación, guía y dedicación a lo largo de todo el proyecto. Su experiencia y conocimiento han sido fundamentales para superar los desafíos y lograr los objetivos establecidos. Además, quiero agradecerle por su paciencia, disponibilidad y valiosos consejos que me han ayudado a crecer profesionalmente.

También quiero mostrar mi gratitud a los miembros del tribunal que evaluaron este trabajo por su tiempo, comentarios constructivos y su contribución a mi formación académica.

Asimismo, quiero agradecer a mis compañeros y amigos que me han brindado su apoyo y estímulo durante este proceso. Sus palabras de aliento y sus perspectivas han sido una fuente constante de motivación.

No puedo dejar de mencionar a todas las personas que de una u otra manera contribuyeron en mi formación y en el desarrollo de este trabajo. Sus enseñanzas, contribuciones y colaboraciones han sido fundamentales para el éxito de este proyecto.

Adicionalmente, quiero agradecer a la Universidad de La Laguna por brindarme los recursos y las instalaciones necesarias para llevar a cabo este proyecto de investigación.

Por último, quiero expresar mi gratitud a mis familiares, en especial a mis padres Sergio Tomás Peña Martín y Zulma Aragón Mamani, por su amor, apoyo incondicional y por creer siempre en mí. Asimismo, quiero dedicar este trabajo a la memoria de mi querido padre. Aunque físicamente ya no se encuentra con nosotros, su amor, apoyo y sabiduría continúan inspirándome en cada paso que doy. Siempre estaré agradecido por su espíritu inquebrantable y por creer en mí. Este logro es también en

honor a su memoria y a todo lo que me enseñó sobre la importancia del esfuerzo, la perseverancia y la pasión por el conocimiento.

Sin la ayuda y el apoyo de todas estas personas, la realización de este Trabajo de Fin de Grado no habría sido posible. Les estoy profundamente agradecido por su confianza, generosidad y contribuciones, las cuales han dejado una huella significativa en mi formación académica y personal.

¡Muchas gracias a todos!

Te quiero Papa.

Índice general

Índice general 3
Índice de figuras 5
Índice de tablas 6
1 Resumen 7
2 Abstract 8
3 Introducción 9
3.1 Objetivos 9
3.2 Estructura de la memoria 10
4 Fundamentos teóricos 12
4.1 Estado del arte 12
4.2 Dispositivos de control 19
4.3 Modelo cinemático del Robot Delta 24
5 Implementación 27
5.1 Condiciones de partida del proyecto 27
5.2 Cinemática directa del Robot Delta 29
5.3 Cinemática inversa del robot delta 36
5.4 Configuración del firmware Marlin 40
5.4.1 Instalación firmware Marlin 40
5.4.2 Configuración del firmware 43
5.5 Configuración de la Orange Pi 46
5.5.1 Configuración básica 46
5.5.2 Configuración puertos UART 47
5.5.3 Configuración OctoPrint 48
5.6 Comunicación con la controladora Bigstreetech skr2 49
5.6.1 Comunicación Serial 49
5.6.2 Comunicación SPI 50
5.6.3 Comunicación I2C 51
5.6.4 Comunicación paralela 52
5.6.5 Comunicación USB 53
5.6.6 Pruebas realizadas para las comunicaciones con la placa BigTreeTech SKR 2 54
5.7 Generación del GCODE 56
5.7.1 Programas para la generación de GCODE 57
5.7.2 Elección del software de generación de GCODE 59

5.7.3 Configuración para la generación 59
6 Resultados 62
6.1 Análisis y resultados durante el proceso 62
6.2 Control del robot delta 63
6.3 Propuesta de mejoras 64
7 Presupuesto 66
8 Conclusiones 67
9 Conclusions 69
10 Bibliografía 71
10.1 Bibliografía de información 71
10.2 Bibliografía de imágenes 72
11 Anexo I: Código modificado del firmware Marlin 74
11.1 Configuration.h 74
11.2 Configuration_adv.h 77
11.3 Settings.cpp 77
11.4 Delta.h 78
11.5 Delta.cpp 79
11.6 Motion.cpp 81
12 Anexo II: Código para filtrar scripts con comandos gcode 82
13 Anexo III: Datasheets de los componentes 83

Índice de figuras

Figura 1: Humani Victus Instrumenta. Bibliografía de imágenes [1] 13
Figura 2: Robot delta X. Bibliografía de imágenes [2] 13
Figura 3: Robot delta lineal. Bibliografía de imágenes [3] 14
Figura 4: Robot delta rotatorio. Bibliografía de imágenes [4] 15
Figura 5: Aplicaciones robot delta. Bibliografía de imágenes [5][6][7] 16
Figura 6: Logo del lenguaje de programación Python. Bibliografía de imágenes [8] 18
Figura 7: Logo lenguaje de programación C++. Bibliografía de imágenes [9] 18
Figura 8: Logo lenguaje de programación Matlab. Bibliografía de imágenes [10] 19
Figura 9: Logo y aplicaciones del lenguaje de programación ROS. Bibliografía de imágenes [11] 19
Figura 10: Cara top de la placa bigtreetech skr2. Bibliografía de imágenes [12] 20
Figura 11: Cara bottom de la placa bigtreetech skr2. Bibliografía de imágenes [13] 21
Figura 12: Cara top de la placa Orange Pi zero 512MB. Bibliografía de imágenes [14] 22
Figura 13: Cara bottom de la placa Orange Pi zero 512MB. Bibliografía de imágenes [15] 22
Figura 14: Driver TMC2208 V3.0 con disipador y sin disipador. Bibliografía de imágenes [16] 23
Figura 15: Modelo cinemático del robot delta. Bibliografía de imágenes [17] 25
Figura 16: Estructura mecánica del robot delta 28
Figura 17: Electrónica del robot delta 29
Figura 18: Esquema del robot delta. Bibliografía de imágenes [18] 30
Figura 19: Esquema del robot delta con los ángulos theta. Bibliografía de imágenes [18] 30
Figura 20: Esquema del robot delta con las esferas. Bibliografía de imágenes [18] 31
Figura 21: Esquema base fija del robot delta. Bibliografía de imágenes [18] 32
Figura 22: Esquema del robot delta con esfera y círculo del brazo 1. Bibliografía de imágenes [18] ... 36
Figura 23: Esquema del robot en el plano YZ y base del efector final. Bibliografía de imágenes [18] .. 38
Figura 24: Esquema del robot delta con el cambio de sistema de coordenadas. Bibliografía de imágenes [18] 40
Figura 25: Pines para la conexión UART. Bibliografía de imágenes [19] 50
Figura 26: Pines para la conexión SPI. Bibliografía de imágenes [19] 51
Figura 27: Pines para la conexión I2C. Bibliografía de imágenes [19] 52
Figura 28: Pines para la conexión paralela. Bibliografía de imágenes [19] 53
Figura 29: Pines para la conexión USB. Bibliografía de imágenes [19] 54

Índice de tablas

Tabla 1: Componentes mecánicos 27
Tabla 2: Componentes electrónicos 28
Tabla 3: Extensiones para modificar el Firmware Marlin 41
Tabla 4: Componentes para la configuración de la Orange Pi zero 46

1 Resumen

El presente Trabajo de Fin de Grado (TFG) se centra en el diseño hardware y el desarrollo de la programación necesaria para poner en funcionamiento un robot Delta, que fue el resultado de un TFG previo. Además, se propone la implementación de mejoras tanto mecánicas como electrónicas con el objetivo de dotar al robot de movimientos precisos y permitir su participación en tareas conjuntas con actividades docentes en el laboratorio del Departamento de Ingeniería Informática y de Sistemas.

El enfoque principal de este TFG se encuentra en la programación del robot Delta, buscando optimizar su rendimiento y capacidades. Se abordan diversos aspectos de la programación, como el control de los motores y la implementación de los modelos cinemáticos tanto directo como inverso, con el objetivo de lograr movimientos precisos y fluidos.

Además de la programación, se realizan mejoras mecánicas en el robot Delta para optimizar su diseño y rendimiento. Se llevan a cabo mejoras en la electrónica, como la implementación de una placa Orange Pi zero, con el objetivo de proporcionar una comunicación de forma remota entre el robot Delta y el PC de control de la estructura.

El resultado de este trabajo permitirá poner en funcionamiento un robot Delta altamente capaz y versátil, con movimientos precisos. Estas capacidades tienen un gran potencial en entornos de laboratorio, donde se busca la automatización y la colaboración de sistemas robóticos en diferentes aplicaciones de la ingeniería de procesos y la robótica.

2 Abstract

This End-of-Degree Project (EDP) focuses on the hardware design and the development of the necessary programming to put into operation a Delta robot, which was the result of a previous EDP. In addition, the implementation of both mechanical and electronic improvements is proposed in order to provide the robot with precise movements and allow its participation in tasks in conjunction with teaching activities in the laboratory of the Department of Computer Science and Systems.

The main focus of this EDP is on the programming of the Delta robot, seeking to optimise its performance and capabilities. Various aspects of programming are addressed, such as motor control and the implementation of both direct and inverse kinematic models, with the aim of achieving precise and fluid movements.

In addition to programming, mechanical improvements are made to the Delta robot to optimise its design and performance. Improvements are made to the electronics, such as the implementation of an Orange Pi zero board, with the aim of providing remote communication between the Delta robot and the structure's control PC.

The result of this work will enable the operation of a highly capable and versatile Delta robot with precise movements. These capabilities have great potential in laboratory environments, where automation and collaboration of robotic systems is sought in different applications of process engineering and robotics.

3 Introducción

3.1 Objetivos

El objetivo técnico del presente Trabajo de Fin de Grado es la puesta en funcionamiento de un robot Delta mediante la implementación de un firmware personalizado, además de habilitar una conexión remota con ayuda de otra placa (Orange Pi zero), desde la cual se enviarán scripts con comandos *gcode* como comprobación de que el robot realice los movimientos pre-programados correctamente.

Lo primero que se llevará a cabo será realizar un estudio sobre el funcionamiento de la placa (BigTreeTech SKR 2) a programar, además del resto de componentes electrónicos que ya se habían implementado en el trabajo previo.

Para alcanzar dicho objetivo se requerirá el desarrollo de diferentes fases. En primer lugar, realizar la modificación del firmware en base a nuestro prototipo de robot, para lo cual una de las modificaciones importantes será la implementación de la cinemática directa e inversa para nuestro prototipo de robot Delta. Por otro lado, una vez se obtenga el firmware totalmente configurado, se continuará con la configuración de la placa Orange Pi zero, para poder establecer una conexión entre la BigTreeTech SKR 2 y el PC de control de forma inalámbrica. El último paso es la elección de un software adecuado para la generación de archivos en formato *gcode*.

Por otro lado, se persigue también el importante objetivo académico de completar satisfactoriamente la asignatura "Trabajo Fin de Grado" de 12 ECTS, que forma parte del plan de estudios del Grado en Ingeniería Electrónica Industrial y Automática.

Es importante destacar que, con el objetivo de optimizar los resultados de este TFG, se han aplicado tanto los conocimientos previamente adquiridos a lo largo de los cursos del Grado, como aquellos adquiridos y desarrollados específicamente para este proyecto. Esta capacidad de análisis y resolución de problemas ha sido fundamental para ampliar el conjunto de habilidades y conocimientos aprendidos en el transcurso del Grado.

3.2 Estructura de la memoria

La presente memoria se estructura de manera lógica y ordenada, abordando los aspectos fundamentales de este Trabajo de Fin de Grado sobre la puesta en funcionamiento de un robot delta. A continuación, se detalla la organización y contenido de cada sección:

- En la sección de "Introducción", se presentan los objetivos del proyecto, estableciendo la finalidad y los resultados esperados. Además, se proporciona una visión general de la estructura de la memoria y los apartados que se abordarán.

- La sección de "Fundamentos teóricos" se centra en sentar las bases teóricas necesarias para comprender el contexto en el que se desarrolla el proyecto. En primer lugar, se aborda el estado del arte del robot delta, explorando su diseño, características y aplicaciones típicas. A continuación, se analizan los dispositivos de control utilizados en el proyecto, como la placa Orange Pi zero, BigTreeTech SKR 2 y los drivers de motores *stepper* (paso a paso). Por último, se introduce el modelo cinemático del robot delta, que proporciona una descripción matemática de los movimientos y posición del robot.

- La sección de "Implementación" se enfoca en los aspectos prácticos del proyecto. Se describen las condiciones de partida del proyecto, estableciendo el marco de referencia y los requisitos iniciales. Luego, se detalla la configuración de la Orange Pi zero, resaltando los pasos y ajustes necesarios para su configuración y correcto funcionamiento. Se explora la comunicación con la controladora del robot y se aborda la cinemática inversa del robot delta, que permite determinar los ángulos de las articulaciones a partir de la posición deseada del efector final. Por último, se explora la generación de código en formato *gcode*, utilizado para controlar los movimientos del robot.

- En la sección de "Resultados", se presentan los hallazgos y logros obtenidos durante el desarrollo del proyecto. Se muestran los resultados experimentales, mediciones y pruebas realizadas, respaldados con imágenes, vídeos y datos relevantes.

- En el apartado de "Presupuesto", se presenta un desglose de los costos asociados al desarrollo del proyecto, incluyendo los componentes utilizados, herramientas, materiales y otros gastos asociados al proyecto.

- Las "Conclusiones" brindan un resumen de los resultados alcanzados, destacando los logros y limitaciones del proyecto. Se realiza una evaluación crítica del trabajo realizado y se plantean posibles líneas de investigación o mejoras futuras.

- La "Bibliografía" recopila las fuentes bibliográficas consultadas y citadas a lo largo de la memoria, garantizando la adecuada referencia y reconocimiento de las fuentes utilizadas.

- En los "Anexos", se incluyen materiales complementarios que amplían la información presentada en la memoria, como la guía de configuración de la Orange Pi zero, los códigos generados y los *datasheets* de los componentes nuevos utilizados en este TFG.

Con esta estructura, se busca proporcionar una visión clara y organizada de los contenidos tratados en la memoria, permitiendo una comprensión exhaustiva del trabajo realizado en el desarrollo de este TFG sobre la puesta en funcionamiento de un robot delta.

4 Fundamentos teóricos

En este apartado se introducirán algunos conceptos teóricos previos con el objetivo de facilitar la comprensión del contexto y desarrollo del proyecto que se ha llevado a cabo.

4.1 Estado del arte

Durante muchos siglos, ha existido una imagen arraigada en las culturas que asocia al robot como una máquina que se asemeja al ser humano. A lo largo de la historia, el deseo de fabricar máquinas capaces de realizar tareas de manera autónoma ha sido constante, y como resultado se han descrito numerosos dispositivos ingeniosos que son antecesores directos de los robots actuales [1].

El término robot fue utilizado por primera vez por Karel Capek (en su obra de teatro R.U.R (*Rossums Universal Robots*, en español "Los robots universales de Rossum"), publicada en 1920. La palabra robot viene del vocablo checo *robota*, que significa trabajo, en el sentido de la obligatoriedad, entendido como servidumbre, trabajo forzado o esclavitud, en referencia sobre todo a los llamados trabajadores alquilados que vivieron en el Imperio Austrohúngaro hasta 1848. Este concepto entronca con la terminología amo-esclavo de los robots actuales, cuando las unidades basan cada movimiento en una orden humana. En R.U.R se desarrolla el concepto de la fabricación en línea ejecutada por robots humanoides, tanto desde el punto de vista narrativo como filosófico. Años más tarde la novela fue adaptada al cine en la película Metrópolis y el término robot quedó fijado para ese significado. Aunque los robots de Capek eran humanos artificiales orgánicos, la palabra robot es casi siempre utilizada para referirse a humanos mecánicos. El término androide puede referirse a cualquiera de éstos, mientras que un *cyborg* (organismo cibernético u hombre biónico) es una criatura combinación de partes orgánicas y mecánicas, se puede apreciar en la Fig.1, obra que representa a un *cyborg* [1][2].

Figura 1: Humani Victus Instrumenta. Bibliografía de imágenes [1]

La investigación en el campo de los robots paralelos tiene sus inicios en 1942, cuando Pollard patentó un mecanismo para el pintado de automóviles. Posteriormente, en 1947, McCough propuso una plataforma de 6 grados de libertad que fue utilizada por Stewart en su simulador de vuelo. Estos manipuladores han encontrado aplicaciones industriales en una amplia variedad de áreas y han despertado un gran interés entre los investigadores a lo largo del tiempo. En los últimos años, se han propuesto diversas estructuras y mecanismos robóticos con múltiples morfologías, lo que ha permitido su aplicación en campos específicos como el empaquetamiento, ensamblaje y manipulación de objetos pequeños. Los robots paralelos presentan numerosas ventajas en comparación con los brazos seriales convencionales. Por lo general, ofrecen mayor rigidez y capacidad de movimiento de masas reducidas, lo que se traduce en una mayor precisión en las operaciones de manipulación [2].

Figura 2: Robot delta X. Bibliografía de imágenes [2]

Entre los robots paralelos, destaca el diseño original del robot delta (Fig.2), el cual fue desarrollado para satisfacer las necesidades de los sectores de producción y manufactura que requerían manipuladores altamente precisos, veloces y capaces de realizar tareas repetitivas. Este es un tipo de robot manipulador compuesto por tres brazos que convergen en un punto central. Cada brazo tiene una articulación rotatoria en la base y una conexión esférica en el extremo, lo que permite movimientos en el espacio tridimensional con gran agilidad y precisión. El diseño delta se basa en un mecanismo paralelo, lo que significa que los tres brazos trabajan en conjunto para controlar la posición y orientación del efector final. Estos robots pueden realizar hasta 300 movimientos por minuto [3].

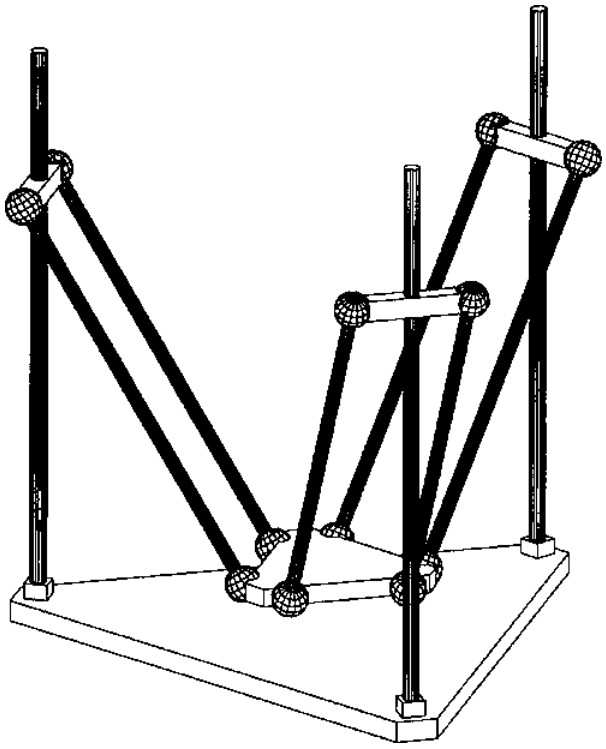


Figura 3: Robot delta lineal. Bibliografía de imágenes [3]

En este trabajo se va a trabajar con uno de los tipos de robot delta, el modelo **rotatorio** (Fig.4), que se caracteriza por una configuración mecánica que permite un movimiento rotatorio en lugar de un movimiento puramente traslacional (Fig.3). Este tipo de robot se compone de una estructura triangular fija en la base, tres brazos articulados y una plataforma móvil en el extremo de los brazos. A diferencia de los robots delta convencionales, que se centran en movimientos rápidos y precisos en el plano horizontal, los robots delta rotatorios permiten un rango adicional de movimiento rotatorio en el espacio tridimensional.

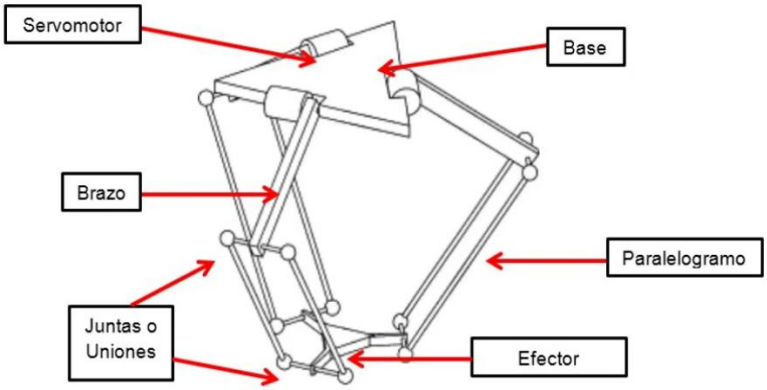


Figura 4: Robot delta rotatorio. Bibliografía de imágenes [4]

Una de las características distintivas del robot delta rotatorio es su capacidad para realizar movimientos rápidos y precisos en entornos de trabajo reducidos. La configuración mecánica triangular proporciona una excelente relación entre el área de trabajo y el espacio ocupado por el robot, lo que lo hace adecuado para aplicaciones en las que se requiere alta velocidad y precisión en espacios confinados. Además, la plataforma móvil en el extremo de los brazos permite un control preciso de la orientación y posición del objeto a manipular [3].

Entre las ventajas del robot delta rotatorio se encuentra su alta velocidad de desplazamiento y capacidad para realizar movimientos repetitivos rápidos. Esto lo hace especialmente adecuado para aplicaciones que implican tareas de ensamblaje, embalaje, *picking & place*, donde se requiere una alta productividad y precisión. Además, su diseño mecánico rígido y liviano le proporciona una excelente capacidad de carga y rigidez estructural [3].

Sin embargo, las desventajas del robot delta rotatorio incluyen su complejidad de programación y configuración. La cinemática inversa y la planificación de trayectorias pueden ser más complejas debido a la adición del movimiento rotatorio. Además, su diseño mecánico específico puede limitar su versatilidad en comparación con otros tipos de robots. También pueden surgir desafíos adicionales en términos de sincronización y coordinación entre los brazos y la plataforma móvil.

En cuanto a las aplicaciones, el robot delta rotatorio encuentra su utilidad en diversas industrias. En la industria manufacturera, se utiliza para la manipulación y ensamblaje de componentes electrónicos, dispositivos médicos y productos alimentarios (Fig.5), donde se requiere precisión y rapidez. En la industria del embalaje (Fig.5), se emplea para el llenado, sellado y etiquetado de productos en línea de producción. Además, en la investigación y desarrollo, el robot delta rotatorio se utiliza para investigaciones en robótica avanzada, prototipado rápido y pruebas de concepto[1]. Otro tipo de aplicación especial que se le está dando últimamente es su uso como una impresora 3D [4][5][6].

Figura 5: Aplicaciones robot delta. Bibliografía de imágenes [5][6][7]

En el campo de la programación de robots delta rotatorios, se han desarrollado diversas metodologías y técnicas para permitir un control preciso y eficiente de estos sistemas. Entre las principales técnicas utilizadas, se encuentra la cinemática directa e inversa, el control de trayectorias y la planificación de movimientos [7].

La cinemática directa se refiere al cálculo de la posición y orientación del efector final del robot a partir de las posiciones de sus articulaciones. En el caso de los robots delta rotatorios, la cinemática directa implica el uso de ecuaciones algebraicas para determinar la posición en el espacio del efector

[1] Pruebas de concepto: Son estudios de viabilidad que se realizan antes de involucrarse en un determinado proyecto o idea.

final en función de las coordenadas de las articulaciones. Esta información es fundamental para controlar la posición y el movimiento del robot en el espacio de trabajo [8].

La cinemática inversa, por otro lado, implica el cálculo de las posiciones de las articulaciones del robot a partir de una posición y orientación deseadas del efector final. Para los robots delta rotatorios, esto implica resolver un sistema de ecuaciones no lineales y complejas. Varios métodos numéricos, como el método de Newton-Raphson [8], se utilizan para encontrar las soluciones adecuadas. La cinemática inversa es esencial para lograr el control preciso del robot y permitir la interacción con el entorno de manera efectiva.

El control de trayectorias, o control dinámico, es otra técnica importante en la programación de robots delta rotatorios. Consiste en definir y seguir trayectorias de movimiento deseadas para el efector final del robot. Esto implica la generación de perfiles de velocidad y aceleración, así como el diseño de algoritmos de control que permitan al robot seguir la trayectoria de manera suave y precisa. Diversos enfoques, como los métodos basados en PID (Proporcional-Integral-Derivativo) y los algoritmos de control adaptativo, se aplican en el control de trayectorias de robots delta rotatorios [9].

La planificación de movimientos, o control cinemático, también desempeña un papel fundamental en la programación de estos robots. Implica la generación de secuencias de movimientos que permitan al robot alcanzar diferentes posiciones o ejecutar tareas específicas de manera eficiente. Esto implica el uso de algoritmos de planificación de movimientos, como el algoritmo RRT (R*apidly-Exploring Random Tree*), que permite encontrar caminos óptimos y seguros en el espacio de configuración del robot. La planificación de movimientos es esencial para optimizar el rendimiento del robot y minimizar el tiempo requerido para realizar tareas específicas [9].

En el campo de la robótica, existen varios lenguajes y entornos de programación que son ampliamente utilizados debido a sus ventajas y compatibilidad con los requerimientos específicos de la robótica. Algunos de los lenguajes y entornos más populares son los siguientes:

-Python: Es un lenguaje de programación de alto nivel que se ha vuelto muy popular en la comunidad de robótica. Python (Fig.6) es conocido por su simplicidad y legibilidad, lo que facilita el

desarrollo de algoritmos y la implementación de controladores para robots. Además, cuenta con una amplia variedad de bibliotecas y *frameworks*[2] específicos de robótica [10].

Figura 6: Logo del lenguaje de programación Python. Bibliografía de imágenes [8]

-C++: Es otro lenguaje ampliamente utilizado en robótica, especialmente cuando se requiere un rendimiento y una eficiencia computacional más altos. C++ (Fig.7) es conocido por su capacidad de proporcionar un control de bajo nivel y una ejecución rápida, lo que lo hace adecuado para aplicaciones que requieren tiempos de respuesta rápidos, como el control en tiempo real de robots delta rotatorios [11].

Figura 7: Logo lenguaje de programación C++. Bibliografía de imágenes [9]

-MATLAB: Es un entorno de programación y un lenguaje que se utiliza ampliamente en el campo de la robótica. Proporciona una amplia variedad de herramientas y funciones para realizar análisis, simulaciones y control de robots. MATLAB (Fig.8) es particularmente útil en tareas de modelado y simulación de sistemas robóticos, así como en el desarrollo de algoritmos de control para robots [12].

[2] Framework: Es un marco o esquema de trabajo generalmente utilizado por programadores para realizar el desarrollo de software.

Figura 8: Logo lenguaje de programación Matlab. Bibliografía de imágenes [10]

-ROS (*Robot Operating System*): Es un f*ramework* de código abierto diseñado específicamente para la programación y control de robots. Proporciona una estructura modular y herramientas para facilitar el desarrollo de aplicaciones robóticas, incluidos los robots delta rotatorios. ROS (Fig.9) permite la comunicación entre diferentes componentes de un sistema robótico, como sensores, actuadores y algoritmos de control, lo que facilita la integración y el desarrollo de sistemas complejos [13].

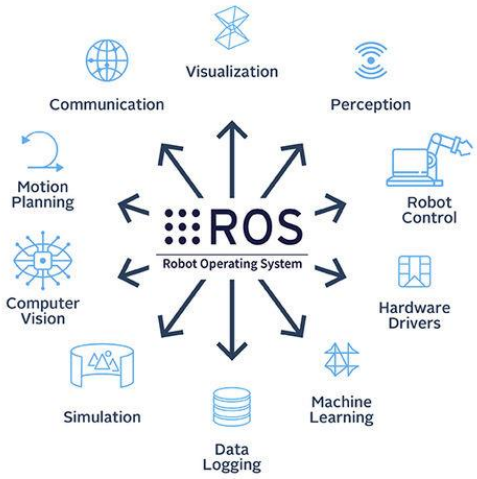


Figura 9: Logo y aplicaciones del lenguaje de programación ROS. Bibliografía de imágenes [11]

4.2 Dispositivos de control

En el contexto de los robots delta, es fundamental contar con dispositivos de control adecuados que permitan la interacción y supervisión de las operaciones del robot. Estos dispositivos desempeñan un papel esencial en el control de los movimientos, la comunicación con el sistema robótico y la supervisión de sus estados.

Uno de los dispositivos de control utilizados en la implementación y operación del robot delta es la placa BigTreeTech SKR 2 (Fig.10). Esta placa es ampliamente reconocida en la comunidad de la robótica debido a sus características y capacidades que la hacen adecuada para aplicaciones de control de robots orientados a la impresión 3D.

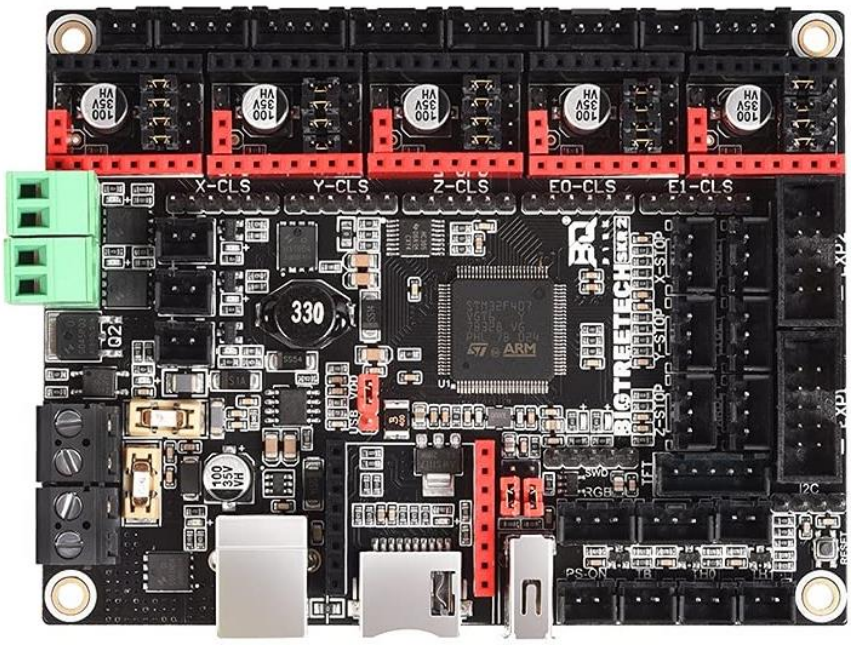


Figura 10: Cara top de la placa BigTreeTech SKR 2. Bibliografía de imágenes [12]

La placa BigTreeTech SKR 2 se basa en el microcontrolador de 32 bits STM32F407, que ofrece un alto rendimiento y una capacidad de procesamiento avanzada. Su arquitectura y diseño permiten una ejecución eficiente de algoritmos de control complejos, lo que resulta crucial en el control preciso y en tiempo real de las articulaciones y movimientos del robot delta [14].

Una de las características destacadas de la placa BigTreeTech SKR 2 es su amplia conectividad. Proporciona diversos puertos de entrada y salida, incluyendo puertos USB, UART, I2C y SPI, lo que facilita la conexión con otros componentes y dispositivos externos, como sensores, pantallas o sistemas de comunicación [14].

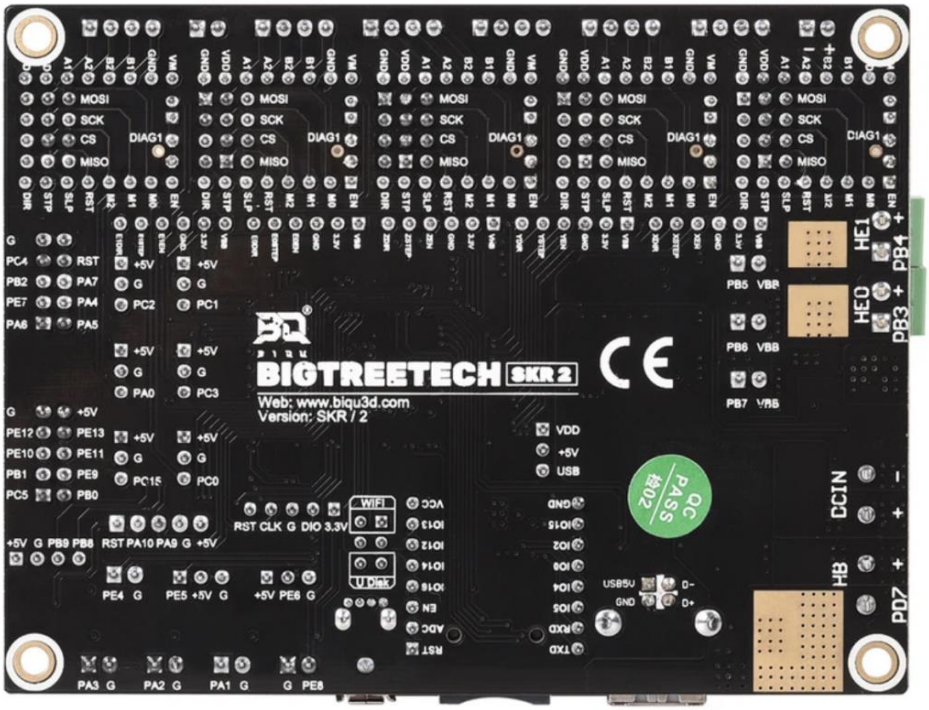


Figura 11: Cara bottom de la placa BigTreeTech SKR 2. Bibliografía de imágenes [13]

Además, la placa BigTreeTech SKR 2 (Fig.11) es compatible con diversos entornos y lenguajes de programación utilizados en el ámbito de la robótica, como Marlin y RepRap. Esto brinda flexibilidad al desarrollador, permitiéndole utilizar las herramientas y librerías adecuadas para la programación y el control del robot delta [14].

Otra ventaja de la placa SKR 2 es su capacidad de expansión. Dispone de ranuras para tarjetas microSD y puertos para módulos adicionales, como controladores de motores paso a paso o sensores específicos [14]. Estas capacidades de expansión permiten adaptar la placa a las necesidades específicas del robot delta y ampliar sus funcionalidades según los requisitos del proyecto. Toda esta información se encontrará más detallada en el manual de la BigTreeTech SKR 2 (Anexo III).

Otro dispositivo de control utilizado en este proyecto fue la placa Orange Pi Zero (Fig.12) la cual fue empleada como puente de comunicación entre la placa BigTreeTech SKR 2 y el PC de control, desempeñando así un papel crucial en el control y la programación del robot delta. Esta placa, basada en la tecnología de una Raspberry Pi [15], proporcionó una solución efectiva para enviar scripts y comandos desde el ordenador a través de Internet y transmitirlos a la BigTreeTech SKR 2.

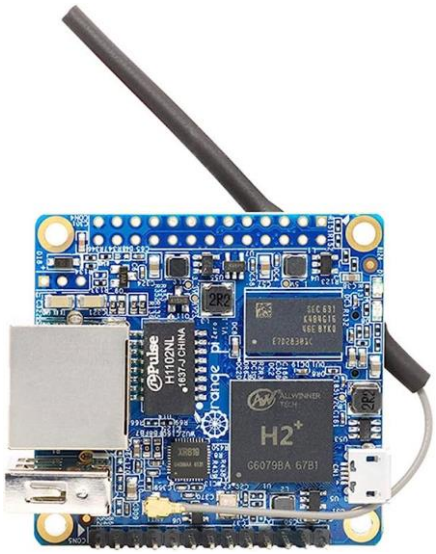


Figura 12: Cara top de la placa Orange Pi zero 512MB. Bibliografía de imágenes [14]

La Orange Pi zero (Fig.13), al contar con conectividad Ethernet y Wi-Fi integrada, permitió establecer una conexión sólida y confiable entre el PC de control y la placa de control del robot delta. A través de esta conexión, se pudieron enviar comandos de control, scripts y programas desde el ordenador a la placa Orange Pi zero, que a su vez los transmitía a la BigTreeTech SKR 2 para su ejecución en el robot.

Esta configuración brindó, a su vez, una forma conveniente de controlar el robot delta de forma remota, lo que permitió la supervisión y el ajuste de los movimientos del robot desde el PC de control. Además, la capacidad de enviar scripts y programas desde el ordenador a la Orange Pi zero facilitó la implementación de algoritmos de control personalizados y la programación de movimientos complejos del robot.

Figura 13: Cara bottom de la placa Orange Pi zero 512MB. Bibliografía de imágenes [15]

Cabe destacar que, la elección de la placa Orange Pi zero como dispositivo de comunicación se basó en su compatibilidad con los sistemas operativos y herramientas utilizadas en el proyecto, así como en su capacidad para establecer una conexión estable y transmitir datos de manera confiable. Su facilidad de configuración y su capacidad de integración con la BigTreeTech SKR 2 fueron factores determinantes en su selección.

Los *drivers* de motores *stepper* TMC2208 (Fig.14) desempeñaron un papel fundamental como dispositivos de control en el funcionamiento del robot delta. Estos *drivers* proporcionaron la capacidad de controlar y gestionar de manera eficiente los motores paso a paso (*stepper*) utilizados en el robot.

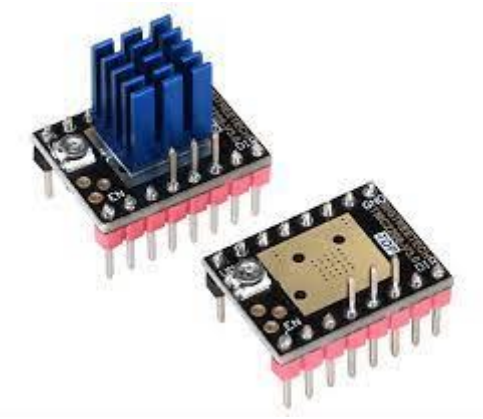


Figura 14: Driver TMC2208 V3.0 con disipador y sin disipador. Bibliografía de imágenes [16]

El TMC2208 es un *driver* de motor paso a paso altamente versátil y potente. Proporciona características avanzadas, como el control de corriente por detección de carga, lo que permite un rendimiento óptimo y una operación suave de los motores. Además, ofrece una interfaz de comunicación SPI (*Serial Peripheral Interface*) que permite una configuración y ajuste precisos de los parámetros del motor [16].

La elección de los *drivers* TMC2208 se basó en sus numerosas ventajas. Estos *drivers* ofrecen una reducción significativa del ruido y la vibración del motor, lo que resulta fundamental en la operación precisa y suave del robot delta. Además, su capacidad para detectar la carga y ajustar automáticamente la corriente del motor contribuye a una mayor eficiencia energética y evita posibles daños por sobrecalentamiento [16].

Además de sus características técnicas, los *drivers* TMC2208 son ampliamente utilizados en la comunidad de robótica debido a su facilidad de uso y su compatibilidad con los sistemas de control comunes. Su diseño compacto y su capacidad para manejar corrientes de motor de hasta 2 amperios los convierten en una opción popular para controlar los motores paso a paso del robot delta.

4.3 Modelo cinemático del Robot Delta

El modelo cinemático es una representación matemática que describe la relación entre las variables de entrada y las variables de salida en un sistema mecánico. En el caso del robot delta, el modelo cinemático permite determinar la posición y orientación del efector final (la herramienta o extremo del robot) en función de los ángulos de las articulaciones [17].

El robot delta se caracteriza por su estructura en forma de triángulo con tres brazos articulados conectados a una plataforma móvil. Para comprender su comportamiento cinemático, es necesario establecer una relación precisa entre los ángulos articulares de las articulaciones y la posición y orientación resultante del efector final.

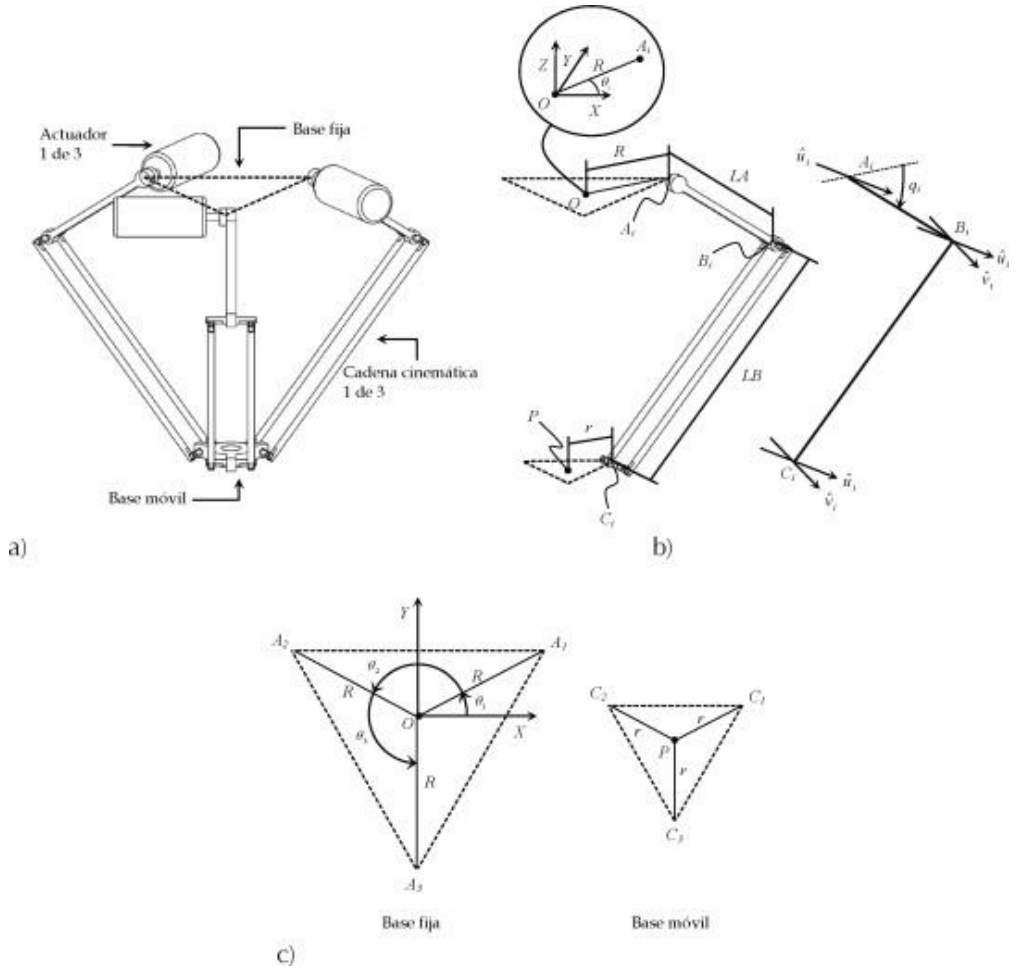


Figura 15: Modelo cinemático del robot delta. Bibliografía de imágenes [17]

El modelo cinemático del robot delta se basa en principios geométricos y trigonométricos para calcular las coordenadas cartesianas del efector final a partir de los ángulos articulares de las articulaciones (Fig.15). Este modelo permite resolver tanto el problema de la cinemática directa, donde se obtiene la posición y orientación del efector final a partir de los ángulos articulares, como el problema de la cinemática inversa, donde se determinan los ángulos articulares necesarios para alcanzar una posición y orientación deseada del efector final [17].

En el caso de la cinemática directa, se utilizan ecuaciones trigonométricas y la geometría del robot delta para calcular las coordenadas (x, y, z) del efector final a partir de los ángulos articulares (theta1, theta2, theta3). Por otro lado, en la cinemática inversa, se resuelven las ecuaciones para determinar los ángulos articulares requeridos (theta1, theta2, theta3) para alcanzar una posición y orientación específica del efector final [17].

Es importante destacar que el modelo cinemático del robot delta puede variar según las características y dimensiones específicas del robot en cuestión. Por lo tanto, es necesario adaptar el modelo a las dimensiones y restricciones del robot utilizado en el proyecto.

El conocimiento y comprensión del modelo cinemático del robot delta es fundamental para la programación y control del robot. Este modelo permite calcular las posiciones y trayectorias deseadas del efector final a partir de los ángulos articulares, y viceversa. Además, proporciona información clave para la planificación de movimientos, la detección de colisiones y la optimización de las tareas realizadas por el robot.

5 Implementación

5.1 Condiciones de partida del proyecto

Este proyecto parte de una base ya formada por otro Trabajo Fin de Grado anterior, desarrollado por Ricardo Melián Maury (Septiembre de 2022), el cuál trató sobre la construcción de un prototipo de robot delta. Este cuenta con gran parte de sus componentes realizados por impresión 3D la cual reduce notoriamente su costo además de aportarle gran levedad. El robot delta está compuesto por dos tipos de componentes, los mecánicos (Fig.16) y los electrónicos (Fig.17).

Tabla 1: Componentes mecánicos

Componentes	Cantidad
Anclaje de los motores	3
Anclaje de los bíceps	3
Base del efector final	1
Antebrazo	3
Parte derecha del bíceps	3
Parte izquierda del bíceps	3
Polea perteneciente al bíceps	3
Guarda de polea perteneciente al bíceps	3
Polea GT2 16 Dientes	3
Separador para bíceps	6
Separador para rótula	12
Soporte para el fin de carrera	3
Soporte de la fuente de alimentación	1
Soporte para la pantalla LCD	4
Soporte para placa controladora	4

Plancha cuadrada de madera de 50x50x16 cm	**2**
Listón de madera de 5,5x3,5x62 cm	**3**
Junta de rótula M3	**12**

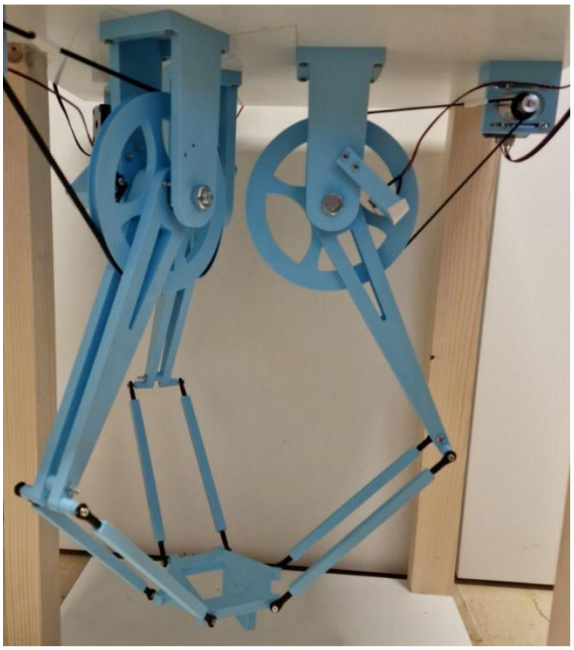


Figura 16: Estructura mecánica del robot delta

Tabla 2: Componentes electrónicos

Componentes	**Cantidad**
Placa BigTreeTech SKR2	**1**
Driver TMC2208	**5**
Pantalla LCD marca CREALITY	**1**
Motor NEMA 17	**3**
Fuente de alimentación 12V 30A	**1**
Cables de conexión	**10**

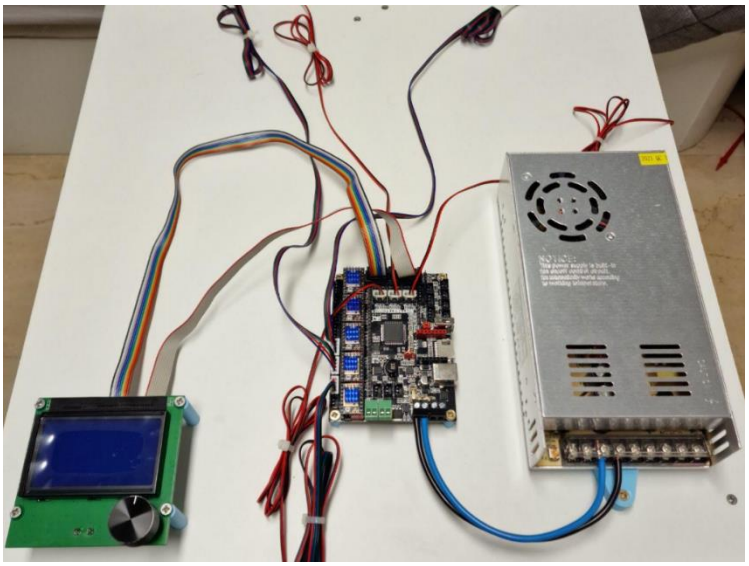


Figura 17: Electrónica del robot delta

En el TFG de Ricardo también se realizó un análisis de la cinemática del robot, pero se decidió no aplicarla y replantearla de nuevo ya que se identificó un error en el cálculo de la cinemática inversa y una limitación relacionada con la falta de un cálculo específico que considerará la relación entre los motores y las poleas que controlan los movimientos de los brazos del robot.

Al detectar esta limitación, se hizo evidente la importancia de abordar nuevamente el análisis de la cinemática directa e inversa en este TFG. Se reconoció la necesidad de desarrollar un cálculo más preciso y específico que tuviera en cuenta las características y configuración exacta del robot delta rotatorio en cuestión. Esto permitió obtener resultados más precisos y confiables en términos de posicionamiento y movimiento del efector final.

5.2 Cinemática directa del Robot Delta

En primer lugar, se determinó algunos parámetros clave de la geometría de nuestro robot. A continuación, se designó el lado del triángulo fijo como **f**, el lado del triángulo efector final como **e**, la longitud de la articulación superior (bíceps) como **rf**, y la longitud de la articulación paralelogramo (antebrazo) como **re**. Estos son los parámetros físicos que se determinan por el diseño de su robot. El sistema de referencia se elegirá con el origen en el centro de simetría del triángulo fijo, como se muestra a continuación en la Fig.18, por lo que la coordenada z del efector final siempre será negativa [18].

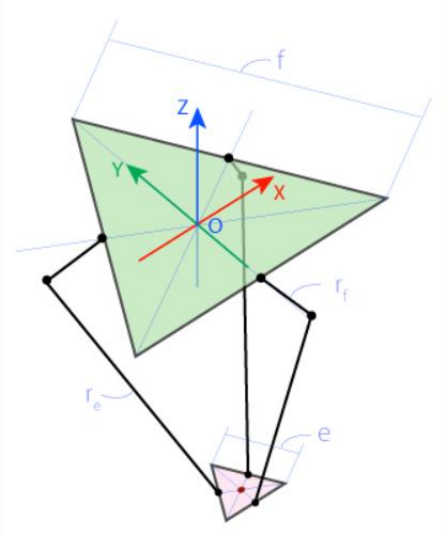


Figura 18: Esquema del robot delta. Bibliografía de imágenes [18]

Para la resolución de la cinemática directa y poder averiguar el punto (X0, Y0, Z0) se deben conocer los ángulos de unión theta1, theta2 y theta3. Con los ángulos theta (Fig.19) se puede calcular fácilmente los puntos J1, J2 y J3 [18].

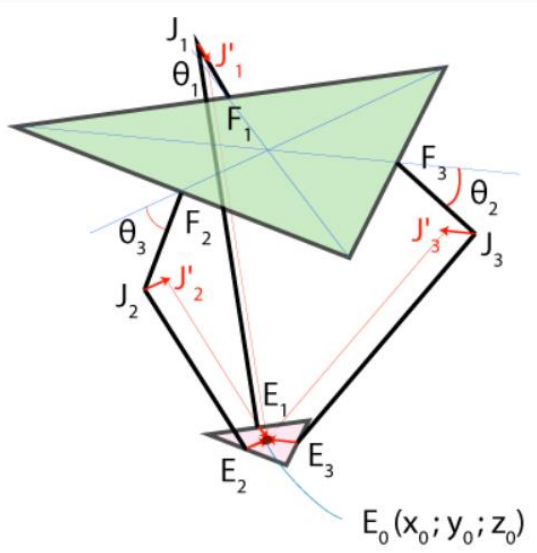


Figura 19: Esquema del robot delta con los ángulos theta. Bibliografía de imágenes [18]

Las uniones entre los puntos J y E pueden rotar libremente sobre los puntos J formando tres esferas de radio igual a la longitud del antebrazo. Posteriormente moveremos los centros de las esferas (Fig.20) desde los puntos J1, J2 y J3 a los puntos J'1, J'2 y J'3 utilizando los vectores de transición E1E0, E2E0 y E3E0 respectivamente. Después de esta transición las tres esferas se intersectarán en un punto (E0) [18].

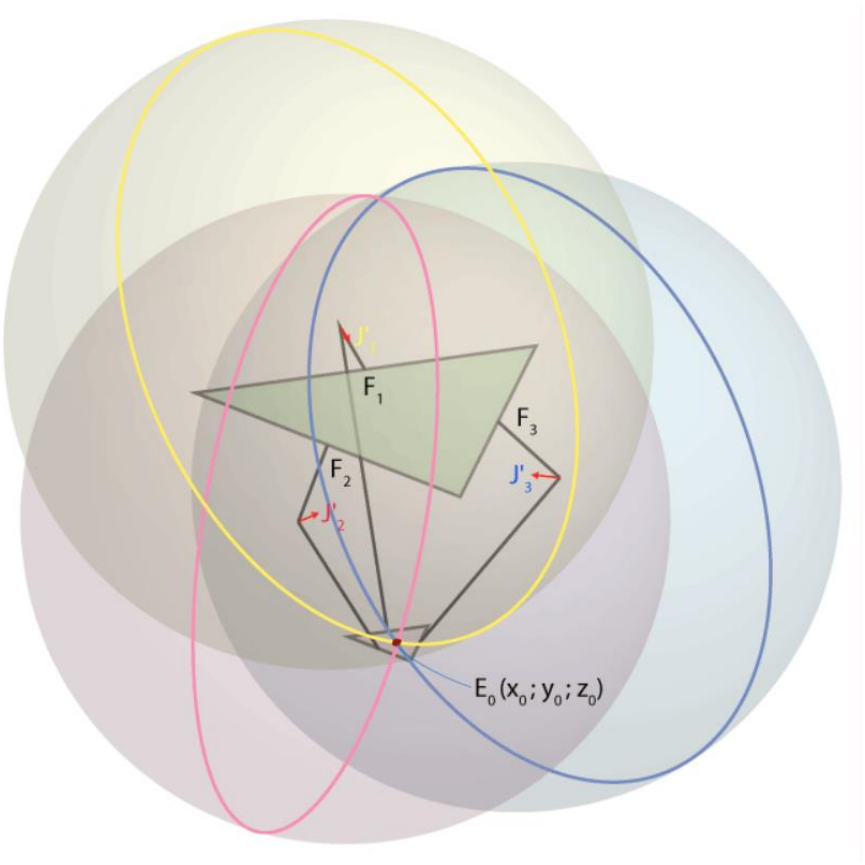


Figura 20: Esquema del robot delta con las esferas. Bibliografía de imágenes [18]

Por tanto, para encontrar las coordenadas (x0, y0, z0) del punto E0, necesitaremos resolver un conjunto de tres ecuaciones como $(x-xj)^2+(y-yj)^2+(z-zj)^2 = re^2$, donde las coordenadas de los centros de la esfera (xj, yj, zj) y el radio son conocidos [18].

Primero se deberá calcular las coordenadas de los puntos J'1, J'2 y J'3.

$$OF_1 = OF_2 = OF_3 = \frac{f}{2} \cdot \tan(30) = \frac{f}{2 \cdot \sqrt{3}}$$

$$J_1J'_1 = J_2J'_2 = J_3J'_3 = \frac{e}{2} \cdot \tan(30) = \frac{e}{2 \cdot \sqrt{3}}$$

$$F_1J_1 = r_f \cdot \cos(\theta_1)\,;\; F_2J_2 = r_f \cdot \cos(\theta_2)\,;\; F_3J_3 = r_f \cdot \cos(\theta_3)$$

Los puntos J'1, J'2 y J'3 (Fig.21) serán:

$$J'_1\left(0;\, \frac{-(f-e)}{2 \cdot \sqrt{3}} - r_f \cdot \cos(\theta_1);\; -r_f \cdot \sin(\theta_1)\right)$$

$$J'_2\left(\left[\frac{(f-e)}{2 \cdot \sqrt{3}} + r_f \cdot \cos(\theta_2)\right] \cdot \cos(30)\,;\, \left[\frac{(f-e)}{2 \cdot \sqrt{3}} + r_f \cdot \cos(\theta_2)\right] \cdot \sin(30)\,;\; -r_f \cdot \sin(\theta_2)\right)$$

$$J'_3\left(\left[\frac{(f-e)}{2\cdot\sqrt{3}}+r_f\cdot\cos(\theta_3)\right]\cdot\cos(30)\,;\,\left[\frac{(f-e)}{2\cdot\sqrt{3}}+r_f\cdot\cos(\theta_3)\right]\cdot\sin(30)\,;\,-r_f\cdot\sin(\theta_3)\right)$$

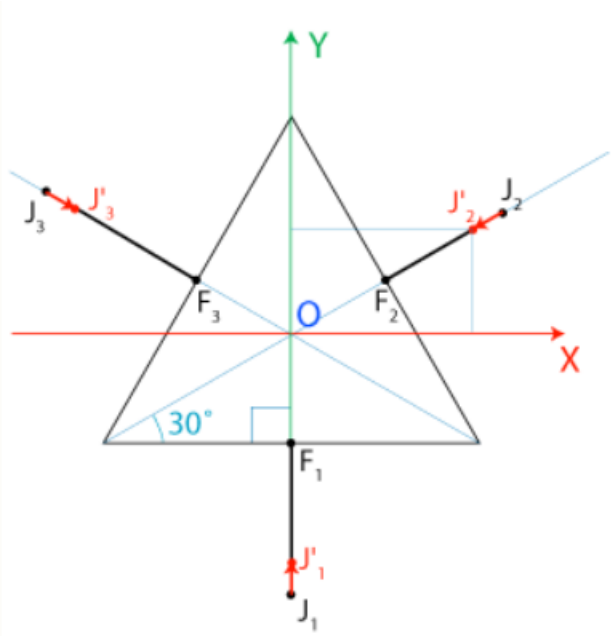


Figura 21: Esquema base fija del robot delta. Bibliografía de imágenes [18]

En las siguientes ecuaciones se designarán las coordenadas de los puntos J1, J2, J3 como (x1, y1, z1), (x2, y2, z2) y (x3, y3, z3). Teniendo en cuenta de que en este caso el punto x1 es igual a 0 [18]. Las ecuaciones de las tres esferas son las siguientes:

$$x^2+(y-y_1)^2+(z-z_1)^2={r_e}^2$$

$$(x-x_2)^2+(y-y_2)^2+(z-z_2)^2={r_e}^2$$

$$(x-x_3)^2+(y-y_3)^2+(z-z_3)^2={r_e}^2$$

Despejar los términos independientes:

(1) $x^2+y^2+z^2-2y_1y-2z_1z={r_e}^2-{y_1}^2-{z_1}^2$

(2) $x^2+y^2+z^2-2x_2x-2y_2y-2z_2z={r_e}^2-{x_2}^2-{y_2}^2-{z_2}^2$

(3) $x^2+y^2+z^2-2x_3x-2y_3y-2z_3z={r_e}^2-{x_3}^2-{y_3}^2-{z_3}^2$

Realizar un cambio de variable:

$$w_i={x_i}^2+{y_i}^2+{z_i}^2$$

Para hallar más ecuaciones se realizará la resta entre las ecuaciones (1), (2) y (3):

(4) = (1) - (2)

$$2x_2x - 2y_1y + 2y_2y - 2z_1z + 2z_2z = {x_2}^2 - {y_1}^2 + {y_2}^2 - {z_1}^2 + {z_2}^2$$

$$2 \cdot (x_2x + (-y_1 + y_2) \cdot y + (-z_1 + z_2) \cdot z) = w_2 - w_1$$

$$x_2x + (-y_1 + y_2) \cdot y + (-z_1 + z_2) \cdot z = \frac{w_2 - w_1}{2}$$

(5) = (1) - (3)

$$2x_3x - 2y_1y + 2y_3y - 2z_1z + 2z_3z = {x_3}^2 - {y_1}^2 + {y_3}^2 - {z_1}^2 + {z_3}^2$$

$$2 \cdot (x_3x + (-y_1 + y_3) \cdot y + (-z_1 + z_3) \cdot z) = w_3 - w_1$$

$$x_3x + (-y_1 + y_3) \cdot y + (-z_1 + z_3) \cdot z = \frac{w_3 - w_1}{2}$$

(6) = (2) – (3)

$$-2x_2x + 2x_3x - 2y_2y + 2y_3y - 2z_2z + 2z_3z = -{x_2}^2 + {x_3}^2 - {y_2}^2 + {y_3}^2 - {z_2}^2 + {z_3}^2$$

$$2 \cdot \big((-x_2 + x_3) + (-y_2 + y_3) \cdot y + (-z_2 + z_3) \cdot z\big) = w_3 - w_2$$

$$(-x_2 + x_3) + (-y_2 + y_3) \cdot y + (-z_2 + z_3) \cdot z = \frac{w_3 - w_2}{2}$$

Seguidamente se calcularán dos ecuaciones, una de ellas que solo dependan de la “x” y de la “z”, y la otra que solo dependan de la “y” y de la “z”. Para ello se realizarán operaciones entre las ecuaciones (4) y (5) [18].

(7) = (4) / (5); Habiendo despejado primeramente la “y” en ambas ecuaciones.

$$\frac{(-y_1 + y_2) \cdot y}{(-y_1 + y_3) \cdot y} = \frac{\frac{w_2 - w_1}{2} - x_2x - (-z_1 + z_2) \cdot z}{\frac{w_3 - w_1}{2} - x_3x - (-z_1 + z_3) \cdot z}$$

$$\left(\frac{w_3 - w_1}{2} - x_3x - (-z_1 + z_3) \cdot z\right) \cdot (-y_1 + y_2) = \left(\frac{w_2 - w_1}{2} - x_2x - (-z_1 + z_2) \cdot z\right) \cdot (-y_1 + y_3)$$

$$\frac{(w_3 - w_1) \cdot (-y_1 + y_2) - (w_2 - w_1) \cdot (-y_1 + y_3)}{2} =$$

$$= x \cdot \big(x_3 \cdot (-y_1 + y_2) - x_2 \cdot (-y_1 + y_3)\big) + z \cdot \big((-z_1 + z_3) \cdot (-y_1 + y_2) - (-z_1 + z_2) \cdot (-y_1 + y_3)\big)$$

A continuación, se realizará un cambio de variables para que la ecuación no se vea tan grande:

$$d = x_3 \cdot \left(-y_1 + y_2\right) - x_2 \cdot \left(-y_1 + y_3\right)$$

La ecuación quedaría:

$$\frac{(w_3 - w_1) \cdot (-y_1 + y_2) - (w_2 - w_1) \cdot (-y_1 + y_3)}{2} =$$

$$= x \cdot d + z \cdot \big((-z_1 + z_3) \cdot (-y_1 + y_2) - (-z_1 + z_2) \cdot (-y_1 + y_3)\big)$$

Despejar la x:

$$\frac{\frac{(w_3 - w_1) \cdot \left(-y_1 + y_2\right) - (w_2 - w_1) \cdot \left(-y_1 + y_3\right)}{2} - z \cdot \big((-z_1 + z_3) \cdot (-y_1 + y_2) - (-z_1 + z_2) \cdot (-y_1 + y_3)\big)}{d}$$

Realizar dos cambios de variables más:

$$b1 = \frac{(w_3 - w_1) \cdot \left(-y_1 + y_2\right) - (w_2 - w_1) \cdot \left(-y_1 + y_3\right)}{2 \cdot d}$$

$$a1 = \frac{-(-z_1 + z_3) \cdot \left(-y_1 + y_2\right) + (-z_1 + z_2) \cdot \left(-y_1 + y_3\right)}{d}$$

La ecuación (7) con los cambios de variables quedaría de la siguiente forma:

$$x = b1 + z \cdot a1$$

Nos falta otra ecuación más, para ello se realizará el mismo procedimiento, pero en vez de despejar la "y" ahora despejaremos la "x".

(8) = (4) / (5); Habiendo despejado la "x" en ambas ecuaciones

$$\frac{x_2 x}{x_3 x} = \frac{\frac{w_2 - w_1}{2} - (-y_1 + y_2) \cdot y - (-z_1 + z_2) \cdot z}{\frac{w_3 - w_1}{2} - (-y_1 + y_3) \cdot y - (-z_1 + z_3) \cdot z}$$

$$\left(\frac{w_3 - w_1}{2} - (-y_1 + y_3) \cdot y - (-z_1 + z_3) \cdot z\right) \cdot x_2 = \left(\frac{w_2 - w_1}{2} - (-y_1 + y_2) \cdot y - (-z_1 + z_2) \cdot z\right) \cdot x_3$$

$$\frac{(w_3 - w_1) \cdot x_2 - (w_2 - w_1) \cdot x_3}{2} =$$

$$= y \cdot \big(x_2 \cdot (-y_1 + y_3) - x_3 \cdot (-y_1 + y_3)\big) + z \cdot \big(x_2 \cdot (-z_1 + z_3) - x_3 \cdot (-z_1 + z_2)\big)$$

Aprovechamos el cambio de variable de d:

$$\frac{(w_3 - w_1) \cdot x_2 - (w_2 - w_1) \cdot x_3}{2} = y \cdot (-d) + z \cdot \big(x_2 \cdot (-z_1 + z_3) - x_3 \cdot (-z_1 + z_2)\big)$$

Ahora se despejará la y:

$$y = \frac{-\frac{(w_3 - w_1) \cdot x_2 - (w_2 - w_1) \cdot x_3}{2} + z \cdot \left(x_2 \cdot (-z_1 + z_3) - x_3 \cdot (-z_1 + z_2)\right)}{d}$$

Realizar dos cambios de variables más:

$$b2 = \frac{-(w_3 - w_1) \cdot x_2 + (w_2 - w_1) \cdot x_3}{2 \cdot d}$$

$$a2 = \frac{\left(x_2 \cdot (-z_1 + z_3) - x_3 \cdot (-z_1 + z_2)\right)}{d}$$

La ecuación (8) con los cambios de variables quedaría de la siguiente forma:

$$y = b2 + z \cdot a2$$

El último paso será sustituir las ecuaciones (7) y (8) en la (1):

$$(b1 + z \cdot a1)^2 + (b2 + z \cdot a2)^2 + z^2 - 2 \cdot y_1 \cdot (b2 + z \cdot a2) - 2 \cdot z_1 \cdot z =$$

$$= {r_e}^2 - {y_1}^2 - {z_1}^2$$

Desarrollando la ecuación y agrupándola en diferentes términos tenemos que:

$$(a1^2 + a2^2 + 1) \cdot z^2 + 2 \cdot (a1 \cdot b1 + a2 \cdot (b2 - y_1) - z_1) \cdot z =$$

$$= -(b1^2 + (b2 - y_1)^2 + {z_1}^2 - {r_e}^2)$$

Realizar un cambio de variable a los términos que acompañan a **z²**

$$a = a1^2 + a2^2 + 1$$

Realizar un cambio de variable a los términos que acompañan a **z**

$$b = 2 \cdot (a1 \cdot b1 + a2 \cdot (b2 - y_1) - z_1)$$

Realizar un cambio de variable a los términos independientes

$$c = b1^2 + (b2 - y_1)^2 + {z_1}^2 - {r_e}^2$$

Ahora solo faltaría resolver la ecuación cuadrática con la siguiente fórmula para hallar la coordenada z.

$$a \cdot z^2 + b \cdot z + c = 0$$

$$z = \frac{-b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a}$$

De la ecuación se escogerá el resultado más pequeño, por lo que elegimos el signo negativo que acompaña a la raíz cuadrada.

Una vez determinado el valor de Z se sustituye ese valor conocido en las ecuaciones (7) y (8) para hallar los valores X e Y respectivamente.

5.3 Cinemática inversa del robot delta

Para el análisis de la cinemática inversa debemos darnos cuenta de que, debido al diseño del robot, la articulación F1J1 sólo puede rotar en el plano YZ, formando un círculo con centro en el punto F1 y radio rf. A diferencia de F1, J1y E1 son las llamadas juntas universales, lo que significa que E1J1 puede rotar libremente en relación a E1, formando una esfera con centro en el punto E1 y radio re [18].

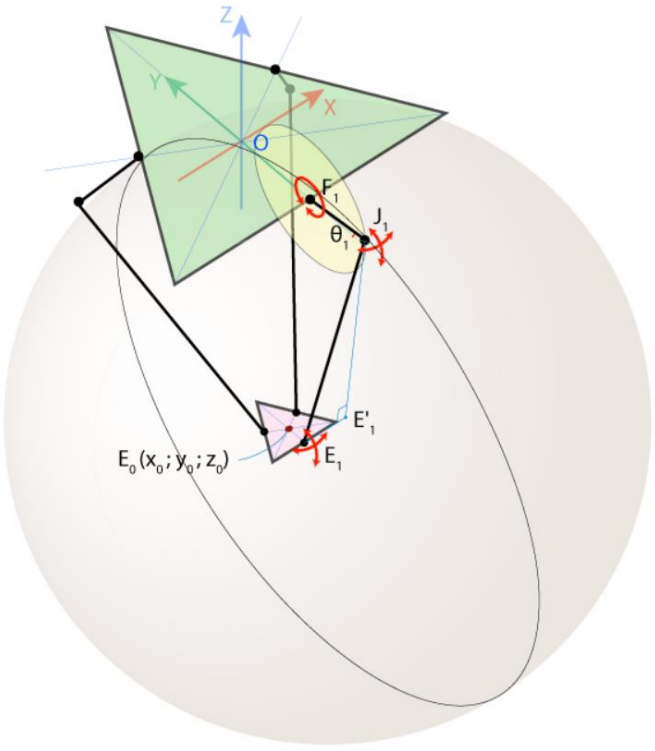


Figura 22: Esquema del robot delta con esfera y círculo del brazo 1. Bibliografía de imágenes [18]

La intersección de esta esfera con el plano YZ es una circunferencia con centro en el punto E'1 y radio E'1J1, donde E'1 es la proyección del punto E1 sobre el plano YZ. El punto J1 se puede encontrar ahora como intersección de dos círculos de radio conocido con centros en E'1 y F1 (debemos elegir sólo

un punto de intersección que tenga la menor coordenada en el eje Y). Y si sabemos J1, podemos calcular el ángulo theta1 [18].

En base a la buena elección del sistema de referencia se produce una simplicidad algebraica. Como la articulación F1J1 se mueve sólo en el plano YZ, se puede omitir completamente la coordenada X. Para aprovechar esta ventaja para los ángulos restantes theta2 y theta3, se debe utilizar la simetría del robot delta. En primer lugar, se gira el sistema de coordenadas en el plano XY alrededor del eje Z a través de un ángulo de 120 grados en sentido antihorario. Ahora se tiene un nuevo sistema de referencia X'Y'Z', y en este sistema podemos encontrar el ángulo theta2 utilizando el mismo algoritmo que se usó para encontrar theta1. El único cambio es que se tiene que determinar nuevas coordenadas x'0 e y'0 para el punto E0, lo que puede hacerse fácilmente utilizando la matriz de rotación correspondiente. Por último, para encontrar el ángulo theta3 se deberá que rotar el sistema de referencia en el sentido de las agujas del reloj la misma cantidad que para theta2, un ángulo de 120 grados [18].

Para theta2:

$$x' = x \cdot \cos(120) + y \cdot \sin(120)$$

$$y' = -x \cdot \sin(120) + y \cdot \cos(120)$$

Para theta3:

$$x' = x \cdot \cos(-120) + y \cdot \sin(-120)$$

$$y' = -x \cdot \sin(-120) + y \cdot \cos(-120)$$

A continuación, se detallarán los cálculos algebraicos que se realizaron para hallar los ángulos theta.

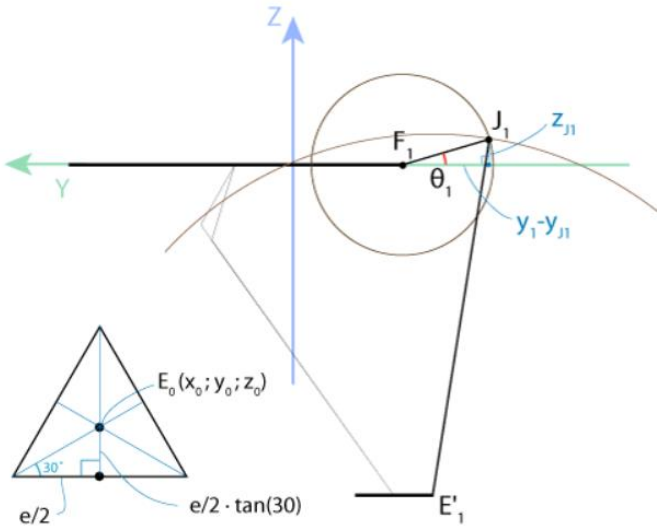


Figura 23: Esquema del robot en el plano YZ y base del efector final. Bibliografía de imágenes [18]

En base a la imagen sabemos que el punto E1' se encuentra en las coordenadas (x0, $y0 - \frac{e}{2 \cdot \sqrt{3}}$, z0), también sabemos que el punto F1 tiene las coordenadas (0, $\frac{-f}{2 \cdot \sqrt{3}}$, 0). Con esta información se hará uso de las ecuaciones cuadráticas para hallar los ángulos theta [18].

$$\left(y_{J1} - y_{F1}\right)^2 + {z_{J1}}^2 = {r_f}^2$$

$$\left(y_{J1} - y_{E'1}\right)^2 + \left(z_{J1} - z_{E'1}\right)^2 = {r_e}^2 - {x_0}^2$$

Se desarrollan los cuadrados:

(1) $${y_{J1}}^2 + {y_{F1}}^2 - 2 \cdot y_{J1} \cdot y_{F1} + {z_{J1}}^2 = {r_f}^2$$

(2) $${y_{J1}}^2 + {y_{E'1}}^2 - 2 \cdot y_{J1} \cdot y_{E'1} + {z_{J1}}^2 + {z_{E'1}}^2 - 2 \cdot z_{J1} \cdot z_{E'1} = {r_e}^2 - {x_0}^2$$

El siguiente paso será restar ambas ecuaciones ((1)-(2)) para eliminar los términos al cuadrado:

$${y_{F1}}^2 - {y_{E'1}}^2 + 2 \cdot y_{J1} \cdot (y_{E'1} - y_{F1}) + {z_{E'1}}^2 - 2 \cdot z_{J1} \cdot z_{E'1} = {r_e}^2 - {x_0}^2 - {r_f}^2$$

Despejar la z_{J1}:

$$z_{J1} = \frac{(y_{E'1} - y_{F1})}{z_{E'1}} \cdot y_{J1} + \frac{{y_{F1}}^2 - {y_{E'1}}^2 + {z_{E'1}}^2 - {r_e}^2 + {x_0}^2 + {r_f}^2}{2 \cdot z_{E'1}}$$

Ahora se realizará un cambio de variable para que la expresión no sea tan compleja.

$$b = \frac{(y_{E'1} - y_{F1})}{z_{E'1}}$$

$$a = \frac{{y_{F1}}^2 - {y_{E'1}}^2 + {z_{E'1}}^2 - {r_e}^2 + {x_0}^2 + {r_f}^2}{2 \cdot z_{E'1}}$$

La ecuación con las nuevas variables quedaría:

(3) $z_{J1} = b1 \cdot y_{J1} + a1$

A continuación, la se sustituirá en la ecuación (1).

$${y_{J1}}^2 + {y_{F1}}^2 - 2 \cdot y_{J1} \cdot y_{F1} + \left(b1 \cdot y_{J1} + a1\right)^2 = {r_f}^2$$

$${y_{J1}}^2 + {y_{F1}}^2 - 2 \cdot y_{J1} \cdot y_{F1} + b1^2 \cdot {y_{J1}}^2 + a1^2 + 2 \cdot a1 \cdot b1 \cdot y_{J1} = {r_f}^2$$

Hay que recordar que se saben todos los términos excepto y_{J1} por lo que nos encontramos con una ecuación cuadrática de la cual se resolverá agrupando los diferentes términos.

$$(1 + b1^2) \cdot {y_{J1}}^2 + (2 \cdot a1 \cdot b1 - 2 \cdot y_{F1}) \cdot y_{J1} + \left({y_{F1}}^2 + a1^2 - {r_f}^2\right) = 0$$

Realizar un cambio de variable para facilitar la resolución:

$$a = 1 + b1^2$$

$$b = 2 \cdot a1 \cdot b1 - 2 \cdot y_{F1}$$

$$c = {y_{F1}}^2 + a1^2 - {r_f}^2$$

Con el cambio de variable la ecuación quedaría de la siguiente forma:

$$a \cdot {y_{J1}}^2 + b \cdot y_{J1} + c = 0$$

Para resolverlo se empleará la siguiente fórmula:

$$y_{J1} = \frac{-b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a}$$

Una vez sabemos el valor de y_{J1}, se sustituye en la ecuación 3 y sacamos el valor z_{J1}. Sabiendo estos dos valores ya se puede calcular el valor de theta1 [18].

$$\theta_1 = \tan^{-1}\left(\frac{z_{J1}}{y_{F1} - y_{J1}}\right)$$

Como comenté al inicio de este apartado, para calcular los ángulos theta2 y theta3 se realiza el mismo procedimiento solo que habría que hacer una rotación en el sistema de coordenadas [18].

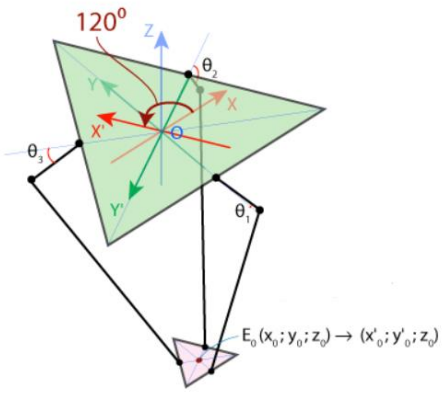


Figura 24: Esquema del robot delta con el cambio de sistema de coordenadas. Bibliografía de imágenes [18]

Para theta2:

$$x_0' = x_0 \cdot \cos(120) + y_0 \cdot \sin(120)$$

$$y_0' = -x_0 \cdot \sin(120) + y_0 \cdot \cos(120)$$

Para theta3:

$$x_0' = x_0 \cdot \cos(-120) + y_0 \cdot \sin(-120)$$

$$y_0' = -x_0 \cdot \sin(-120) + y_0 \cdot \cos(-120)$$

5.4 Configuración del firmware Marlin

5.4.1 Instalación firmware Marlin

Lo primero que se hará va a ser instalar el programa **Visual Studio Code**, el cual nos va a permitir cargar y modificar nuestro firmware. Para ello se seguirán los siguientes pasos:

- Visitar el sitio web oficial de Visual Studio Code en Visual Studio Code - Code Editing. Re-defined.
- En la página principal, se encontrará un botón grande y llamativo que dice "Descargar para [tu sistema operativo]". Hacer clic en dicho botón.
  - Si se está utilizando Windows, se descargará un archivo ejecutable ".exe".
  - Si se está utilizando macOS, se descargará un archivo con extensión ".dmg".
  - Si se está utilizando Linux, se descargará un archivo comprimido en formato ".tar.gz".

- Una vez descargado, abrir el archivo ejecutable o el archivo comprimido según corresponda.
    - En Windows, se ejecuta el archivo ".exe" y se sigue las instrucciones del instalador.
    - En macOS, se abre el archivo ".dmg" y se arrastra la aplicación Visual Studio Code al directorio "Aplicaciones" para instalarla.
    - En Linux, descomprimir el archivo ".tar.gz" en una ubicación deseada, como el directorio de inicio.
- Después de completar la instalación, iniciar Visual Studio Code.
    - En Windows y macOS, se podrá encontrar el acceso directo en el menú de inicio o en el Launchpad.
    - En Linux, navegar a la ubicación donde se descomprimió el archivo y ejecutar el comando "./code" en una terminal para iniciar la aplicación.
- Una vez que se abra Visual Studio Code, se podrá comenzar a utilizarlo. Sin embargo, es posible que se requiera personalizar la configuración o instalar extensiones adicionales para adaptarlo a las propias necesidades.
    - Acceder a la configuración y ajustes en el menú "Archivo" (Windows/Linux) o en el menú "Code" (macOS).
    - Para instalar extensiones, hacer clic en el icono de "Extensiones" en la barra lateral izquierda o utilizar el atajo de teclado "Ctrl+Shift+X" (Windows/Linux) o "Cmd+Shift+X" (macOS).

En nuestro caso para poder hacer la modificación del firmware correctamente hizo falta instalar varias extensiones que fueron las siguiente:

Tabla 3: Extensiones para modificar el Firmware Marlin

Extensiones	Descripción
Auto build Marlin	Agiliza el proceso de compilación y carga del firmware Marlin en impresoras 3D al automatizar tareas clave y simplificar el flujo de trabajo

C/C++	Pueden escribir, depurar y administrar su código C/C++ de manera más eficiente y efectiva
C/C++ Extension Pack	Proporciona un conjunto de herramientas esenciales y características avanzadas que mejoran la productividad del desarrollador y optimizan el flujo de trabajo en proyectos de C/C++.
C/C++ Themes	Proporciona una variedad de temas de color predefinidos y opciones de personalización para crear una experiencia visual agradable y cómoda durante el desarrollo.
CMake	Simplifica el desarrollo de proyectos que utilizan el sistema de compilación CMake. Proporciona características específicas, como resaltado de sintaxis, completado automático, generación y compilación de proyectos, y depuración
CMake Tools	Proporciona una interfaz intuitiva, comandos específicos de CMake, integración con la interfaz de usuario y soporte de depuración.
PlatformIO IDE	Proporciona soporte multiplataforma, gestión de librerías, programación y depuración, integración con PlataformaIO Core, y funcionalidades de control de versiones.
Better Comments (Opcional)	Mejora la legibilidad y expresividad de los comentarios en el código fuente, facilitando la identificación de tareas pendientes, problemas a resolver y notas importantes. Permite organizar los comentarios en secciones y personalizar su apariencia visual.
GitHub Copilot (Opcional)	Herramienta de inteligencia artificial que mejora la productividad y eficiencia en el proceso de programación. Proporciona sugerencias y autocompletado inteligente de código basado en modelos de IA entrenados en una amplia variedad de código fuente.

Live Share (Opcional)	Mejora la colaboración en el desarrollo de software al permitir la compartición de sesiones de código en tiempo real. Facilita la edición, revisión y depuración colaborativa, lo que acelera el proceso de desarrollo y mejora la comunicación entre los miembros del equipo.
Spanish Language Pack for Visual Studio Code (Opcional)	Permite trabajar con Visual Studio Code en español, lo que mejora la experiencia de uso para los usuarios que prefieren trabajar en su idioma nativo. Proporciona una interfaz intuitiva, ayuda contextual y documentación en español.

Lo siguiente será instalar el firmware Marlin, para lo que se ejecutaron los siguientes pasos:

- Descargar el firmware Marlin desde el repositorio oficial en GitHub. Se puede acceder al repositorio en el siguiente enlace: https://github.com/MarlinFirmware/Marlin.
- Hacer clic en el botón verde "Code" y luego seleccionar "Download ZIP" para descargar el archivo ZIP del firmware Marlin en el ordenador.
- Extraer el contenido del archivo ZIP en una ubicación deseada del sistema.
- Abrir Visual Studio Code, seleccionar la opción "Abrir carpeta..." en el menú "Archivo" y navegar hasta la ubicación donde se extrajo el firmware Marlin.
- Seleccionar la carpeta raíz del firmware Marlin y hacer clic en "Abrir" para cargar el proyecto en Visual Studio Code.
- Una vez que el proyecto se haya cargado, se podrá ver la estructura de archivos del firmware Marlin en el explorador lateral izquierdo de Visual Studio Code.

5.4.2 Configuración del firmware

La mayoría de las modificaciones se harán dentro de los archivos “Configuration.h” y “Configuration_adv.h”. Dentro del archivo “Configuraction.h” se harán las siguientes modificaciones:

- Escoger la placa que se está utilizando, en nuestro caso es una BigTreeTech SKR 2, que dentro del firmware se le llama “BOARD_BTT_SKR_V2_0_REV_B”. Si estamos utilizando

otra placa, el nombre de las placas compatibles con este firmware se encuentra en la siguiente ruta /Marlin/src/core/boards.h.

- Modificar la comunicación que se empleará y detallará en el apartado de "Comunicación con la controladora del robot". Habilitaremos el puerto de las comunicaciones por medio de los pines TFT.
- Habilitar la opción de robot delta, además de añadir algunos parámetros necesarios para el cálculo de la cinemática directa e inversa que se detallará más adelante como se implementó en el programa.
- Elegir los drivers que estamos usando para los motores, en nuestro caso son los TMC 2208.
- Habilitar los finales de carrera como máximos.
- Modificar los pasos por mm (*steps* / mm) o pasos por grados (*steps* / º), este valor se escogió a ojo comprobando si se cumplía la relación de forma manual.
- Cambiar la velocidad y aceleración a valores adecuados.
- Habilitar el "Jerk", necesario para la configuración de un robot delta, lo que hace es especificar el cambio de velocidad mínimo que requiere aceleración. Al cambiar la velocidad y la dirección, si la diferencia es menor que el valor establecido, puede ocurrir instantáneamente.
- Desactivar la opción de tener conectado una sonda para el Z mínimo.
- Modificar la dirección que debe tomar, cuando hace el home, a los finales de carrera.
- Modificar los valores limites que tiene el área de trabajo del robot.
- Asignar los valores de la posición que va a tener el robot cuando haga el home.
- Modificar la velocidad que va a tener el robot cuando haga el home.
- Del display habrá que modificar el idioma y habilitar el display que se usará.

Dentro del archivo "Configuration_adv.h" se realizarán las siguientes modificaciones:

- Modificar la distancia y la velocidad a la que se va a mover el robot una vez haya tocado el final de carrera.
- Modificar el valor de velocidad que tendrá el robot cuando lo movamos de forma manual.

- Habilitar que el display nos muestre información sobre el robot en el menú principal.
- Habilitar el espacio de trabajo en plano para las CNC.
- A cada *driver* se le modificará los pasos.

Para la implementación de la cinemática directa e inversa se modificaron los diferentes archivos:

- Archivo settings.cpp:
    - Añadir las variables constantes para los cálculos.
- Archivo delta.h:
    - Llamar a las variables constantes declaradas en el settings.
    - Declarar las funciones para el cálculo de la cinemática inversa.
    - Declarar la función para el cálculo de la cinemática directa.
- Archivo delta.cpp:
    - Llamar a las variables declaradas en el delta.h
    - Definir la función “delta_calcAngleYZ” que calcula un ángulo theta empleando el modelo matemático desarrollado en el apartado “Cinemática inversa del robot delta”.
    - Definir la función “inverse_kinematics” que llama tres veces a la función “delta_calcAngleYZ” para obtener los tres ángulos theta.
    - Definir la función “forward_kinematics”, la cual replica los cálculos matemáticos realizados en el apartado “Cinemática directa del robot delta”.
    - Modificar la forma que hacía el home, ya que por defecto se desplazaba en el eje z moviendo todos los motores, en cambio, con la nueva configuración hace el home moviendo cada motor de forma individual y yendo uno por uno.
- Archivo motion.cpp:
    - Modificar la forma en la que se le pasa los parámetros a la cinemática directa.

5.5 Configuración de la Orange Pi

Para la realización de la correcta configuración de la Orange Pi zero se necesitó una serie de componentes los cuales se indicarán en la Tabla 4.

Tabla 4: Componentes para la configuración de la Orange Pi zero

Componentes	Cantidad
Placa Orange Pi zero 512MB	**1**
Tarjeta microSD de 8GB	**1**
Convertidor SD a USB	**1**
Tira de 13 pines para soldar	**2**
Fuente de alimentación 5V 3A	**1**
Cable para alimentación micro USB	**1**
Cables de conexión	**4**
Cable ethernet RJ45	**1**

5.5.1 Configuración básica

Una vez adquiridos esos componentes se siguieron los siguientes pasos para la configuración básica:

1.) Descargar la imagen dentro de la página de Armbian.

2.) Formatear la tarjeta microSD.

3.) Instalar la imagen descargada dentro de la microSD.

4.) Encender la Orange Pi zero y averiguar la dirección IP que se la había asignado, para ello se hizo uso del programa Fing.

5.) Ahora con la IP disponible, se puede entrar a realizar la configuración gracias al programa Putty. Una vez dentro, pedirá un nombre de usuario. Como no se dispone de cuenta, se elige como usuario “root”.

6.) Lo siguiente que pedirá será la contraseña para ese usuario, que será “1234”.

7.) A continuación, se tiene que crear una nueva contraseña para el root.

8.) Elegir en “system command Shell” la opción de bash.

9.) Ahora se debe crear un nuevo usuario, para lo que pedirá un nombre de usuario y una contraseña para ese usuario.

10.) Se elige el idioma español seleccionando “es_ES.utf-8”.

11.) Ahora se escribe el comando de “sudo apt-get update” para comprobar que se tienen todas las actualizaciones. Como se acaba de instalar, dará aviso de su actualización, que se ejecuta mediante el comando “sudo apt-get upgrade”.

5.5.2 Configuración puertos UART

Lo siguiente será configurar los puertos UART. Para ello se modifica el archivo “armbienEnv.txt” siguiendo los siguientes pasos:

1.) Se abre el archivo "/boot/armbianEnv.txt" en un editor de texto. Este archivo contiene la configuración del sistema de arranque de Armbian en la placa.

2.) Se busca la sección del archivo que se refiere a las superposiciones (overlays). Por lo general, se encuentra una línea similar a "overlays =". Esta sección permite habilitar y configurar diferentes funciones y periféricos en la placa.

3.) Se modifica la línea de la sección overlays para habilitar el UART2. Se reemplaza el valor existente con "uart2". Por ejemplo, si la línea original es "overlays = ..." la nueva línea debe ser "overlays = uart2".

4.) A continuación, se busca la línea que se refiere a la configuración de la consola. Por lo general, se encuentra una línea similar a "console = ...".

5.) Se cambia la configuración de la consola para mostrar la salida en el puerto de visualización (display) en lugar de utilizar el puerto UART. Si la línea original es "console = both", cámbiala a "console = display".

6.) Se guardan los cambios realizados en el archivo "/boot/armbianEnv.txt".

7.) Se reinicia la placa Orange Pi Zero para que los cambios surtan efecto, ejecutando el comando "sudo reboot".

5.5.3 Configuración OctoPrint

Para realizar la instalación y configuración del OctoPrint se siguieron los siguientes pasos:

1.) Lo primero será instalar los paquetes de Python requeridos para OctoPrint, para ello se escribe el siguiente comando: “sudo apt install python3-pip python-dev-is-python3 python-setuptools python3-virtualenv git libyaml-dev build-essential virtualenv”.

2.) Ahora se debe asegurar de estar en la carpeta de inicio del usuario que se ha creado antes del siguiente paso.

3.) Se crea una carpeta llamada OctoPrint y se posiciona en esta.

4.) Dentro de la carpeta OctoPrint se crea un entorno virtual con el comando “virtualenv venv”.

5.) Ahora se activa ese entorno virtual. Esto se hace para que los comandos de Python y las bibliotecas que se instalen se tomen del entorno virtual en lugar del entorno global del sistema. Esto permite mantener un entorno de desarrollo limpio y controlar las dependencias específicas del proyecto.

6.) El siguiente comando que se emplea es “pip install pip –upgrade”. Se utiliza para actualizar la herramienta pip en el sistema. pip es el administrador de paquetes de Python que se utiliza para instalar, actualizar y administrar paquetes y dependencias de Python.

7.) Ahora con la herramienta pip se instala el octoprint con el siguiente comando “pip install octoprint”.

8.) A continuación, se otorgan ciertos permisos al usuario. Para ello se hace uso de los siguientes comandos: “sudo usermod -a -G tty NombreUsuario” y “sudo usermod -a -G dialout NombreUsuario”.

9.) Lo siguiente será añadir dos scripts para que el servidor se inicie nada más encender la placa. Se hará introduciendo los siguientes comandos: “wget https://github.com/foosel/OctoPrint/raw/master/scripts/octoprint.init && sudo mv octoprint.init /etc/init.d/octoprint” y “wget https://github.com/foosel/OctoPrint/raw/master/scripts/octoprint.default && sudo mv octoprint.default /etc/default/octoprint”.

10.) Después se ejecuta el siguiente comando “sudo chmod +x /etc/init.d/octoprint”, con el fin de cambiar los permisos del archivo octoprint en el directorio /etc/init.d/ para permitir su ejecución.

11.) Ahora se modifica la configuración de octoprint dentro de /etc/default/ mediante el siguiente comando “sudo nan:o /etc/default/octoprint”. Una vez dentro del archivo, se habilitan las siguientes líneas:

BASEDIR=/home/pi/.octoprint

CONGFILE=/home/pi/.octoprint/config.yaml

DAEMON=/home/pi/OctoPrint/venv/bin/octoprint

Ya habilitadas, se guarda el archivo y se sale del mismo.

12.) A continuación, se ejecuta el siguiente comando: “sudo update-rc.d octoprint defaults”, con el objetivo de configurar el servicio de OctoPrint, para que se inicie automáticamente durante el arranque del sistema y se detenga durante el apagado. Esto garantiza que el servicio esté disponible cada vez que se inicie el sistema.

13.) Por último, se enciende el servidor por primera vez con el siguiente comando: “/OctoPrint/venv/bin/octoprint serve”.

5.6 Comunicación con la controladora Bigstreetech skr2

Debido al hardware disponible en la placa, se exploraron varias opciones de comunicación para establecer una conexión efectiva. Entre las diferentes posibilidades, se consideró la comunicación serial a través de la configuración de los puertos UART. Esta elección se basó en las ventajas que ofrece este método de comunicación en el contexto de nuestro proyecto, a continuación, se detallarán cada una de ellas.

5.6.1 Comunicación Serial

Es un método de transferencia de datos en el que los bits de información se transmiten de manera secuencial, uno después del otro, a través de un solo canal de comunicación. Este canal puede ser un cable físico, como un cable RS-232 o un cable USB, o puede ser una conexión inalámbrica, como Bluetooth. En una comunicación serial, se utilizan dos líneas de señal principales:

- Línea de transmisión (Tx): Es la línea a través de la cual se envían los bits de datos. El dispositivo que transmite envía los bits uno por uno en secuencia.

- Línea de recepción (Rx): Es la línea a través de la cual se reciben los bits de datos. El dispositivo receptor está a la espera de recibir los bits enviados y los lee uno por uno en secuencia.

La placa SKR2 tiene puertos UART que permiten la comunicación serial. Puedes utilizar estos puertos para establecer una conexión serial con otros dispositivos, como un ordenador, una Orange Pi zero u otra placa controladora.

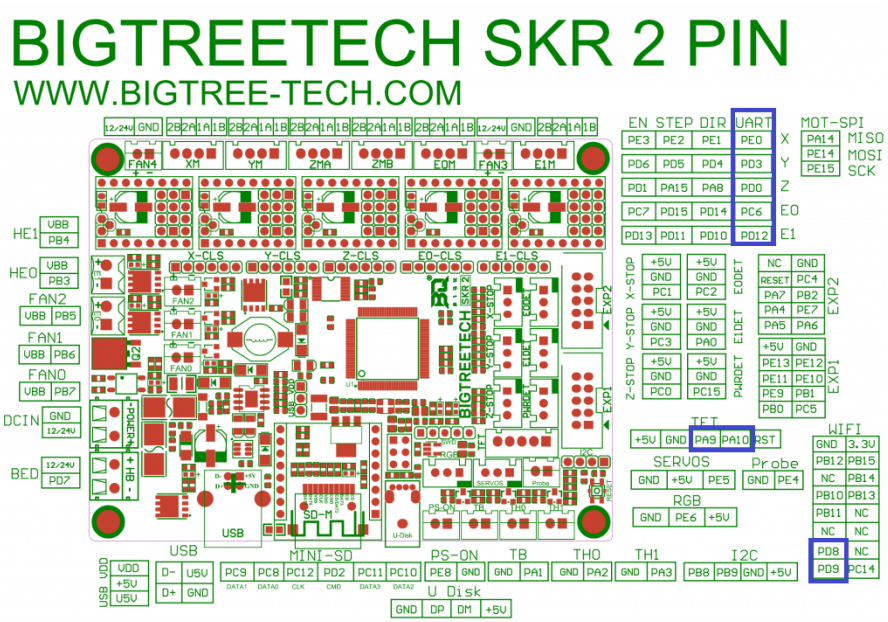


Figura 25: Pines para la conexión UART. Bibliografía de imágenes [19]

5.6.2 Comunicación SPI

Es un protocolo de comunicación síncrona utilizado para la transferencia de datos entre un dispositivo maestro y uno o varios dispositivos esclavos. Se utiliza comúnmente en aplicaciones de interconexión de dispositivos electrónicos, como microcontroladores, sensores, pantallas LCD, tarjetas de memoria, entre otros. En una comunicación SPI, se establece una conexión física y lógica entre el dispositivo maestro y los dispositivos esclavos a través de varias líneas de señal:

- SCLK (Serial Clock): Es la línea de reloj generada por el dispositivo maestro. Define la velocidad a la cual los datos serán transmitidos.
- MOSI (Master Output, Slave Input): Es la línea de datos de salida del dispositivo maestro y entrada de datos del dispositivo esclavo. El dispositivo maestro envía los datos a través de esta línea.

- MISO (Master Input, Slave Output): Es la línea de datos de entrada del dispositivo maestro y salida de datos del dispositivo esclavo. El dispositivo esclavo envía los datos al dispositivo maestro a través de esta línea.

- SS (Slave Select): Es la línea utilizada para seleccionar el dispositivo esclavo con el que se desea comunicar el dispositivo maestro. Permite establecer una conexión específica entre el maestro y el esclavo deseado.

La SKR2 cuenta con pines dedicados para la comunicación SPI (Interfaz Periférica en Serie). Puedes utilizar esta interfaz para conectar dispositivos compatibles con SPI, como pantallas LCD, sensores o módulos de expansión.

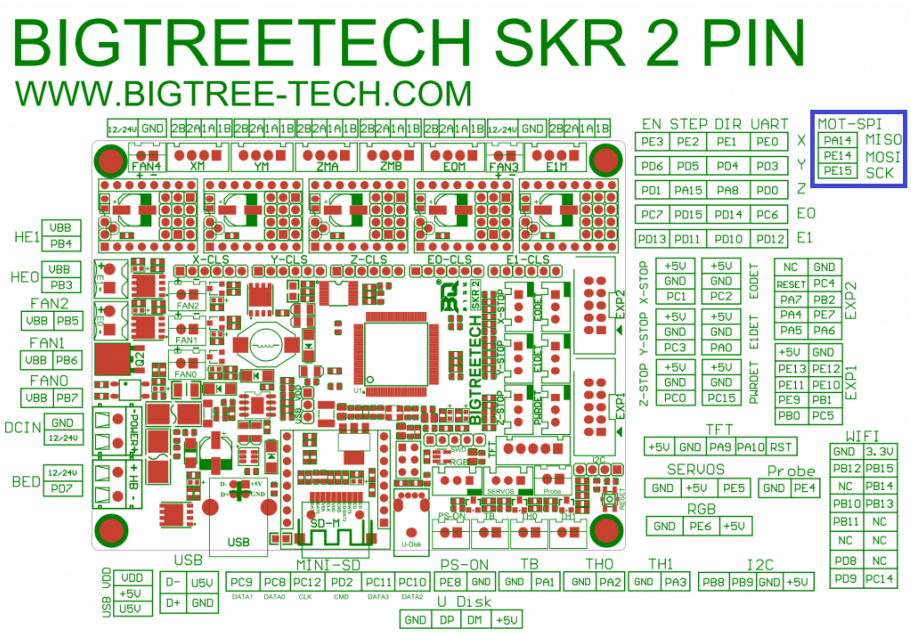


Figura 26: Pines para la conexión SPI. Bibliografía de imágenes [19]

5.6.3 Comunicación I2C

Es un protocolo de comunicación síncrona utilizado para la transferencia de datos entre dispositivos electrónicos. Fue desarrollado por Philips (ahora NXP *Semiconductors*) y se utiliza ampliamente en sistemas embebidos, periféricos y sensores. En una comunicación I2C, se establece una conexión entre un dispositivo maestro y uno o varios dispositivos esclavos a través de dos líneas de señal:

- SDA (Serial Data): Es la línea utilizada para la transmisión de datos entre el dispositivo maestro y los dispositivos esclavos. Tanto el maestro como los esclavos pueden enviar y recibir datos a través de esta línea.

- SCL (Serial Clock): Es la línea de reloj generada por el dispositivo maestro para sincronizar la transferencia de datos. El maestro controla la velocidad de la comunicación al establecer la frecuencia del reloj en esta línea.

En la comunicación I2C, los dispositivos esclavos son identificados mediante una dirección única de 7 bits. El dispositivo maestro envía la dirección del dispositivo esclavo con el que desea comunicarse antes de iniciar la transferencia de datos. La placa SKR2 tiene pines SDA (*Serial Data Line*) y SCL (*Serial Clock Line*) para la comunicación I2C (*Inter-Integrated Circuit*). Puedes utilizar esta interfaz para conectar varios dispositivos I2C, como sensores, pantallas o módulos de expansión.

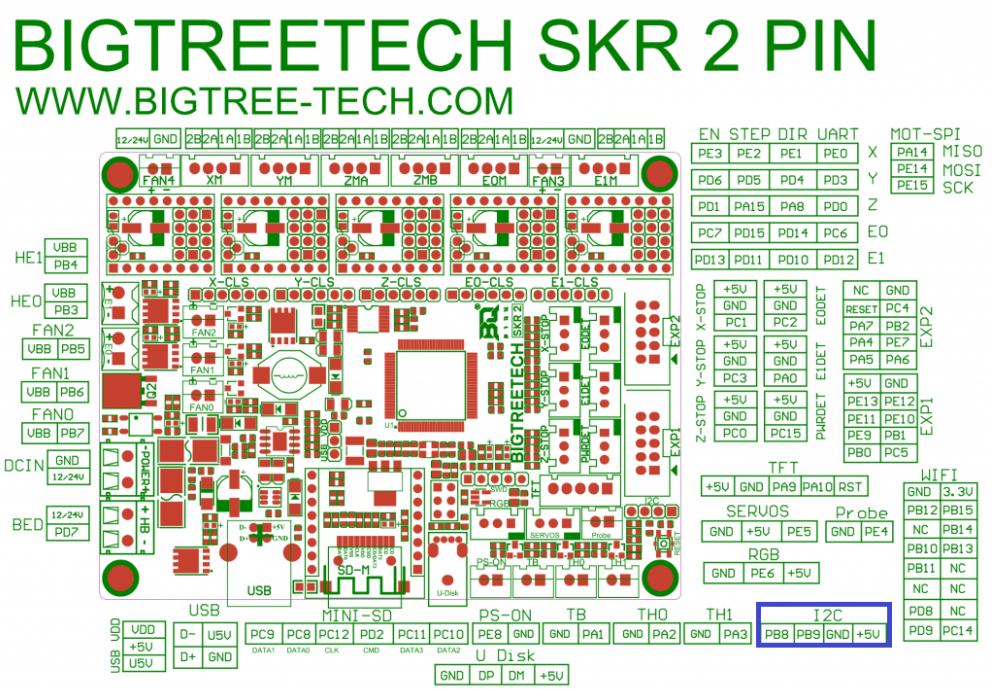


Figura 27: Pines para la conexión I2C. Bibliografía de imágenes [19]

5.6.4 Comunicación paralela

Es un método de transferencia de datos en el que varios bits se transmiten simultáneamente a través de múltiples líneas de datos en paralelo. Cada bit de información se transmite en una línea separada, lo que permite una transferencia de datos más rápida en comparación con la comunicación en serie.

En una comunicación paralela típica, los datos se dividen en palabras de varios bits y se envían a través de líneas de datos dedicadas. Por ejemplo, si se utiliza una comunicación paralela de 8 bits, cada palabra de datos se divide en 8 bits y se transmite a través de 8 líneas de datos simultáneamente. Además de las líneas de datos, también se utilizan líneas adicionales para señales de control, como señales de reloj, señales de habilitación y señales de dirección.

La comunicación paralela es comúnmente utilizada en aplicaciones que requieren una alta velocidad de transferencia de datos, como la conexión de periféricos a una placa base de computadora o la conexión de pantallas LCD. Sin embargo, a medida que las comunicaciones en serie, como USB y Ethernet, se han vuelto más comunes, la comunicación paralela ha sido reemplazada en muchos casos debido a su complejidad de cableado y limitaciones en la distancia de transmisión.

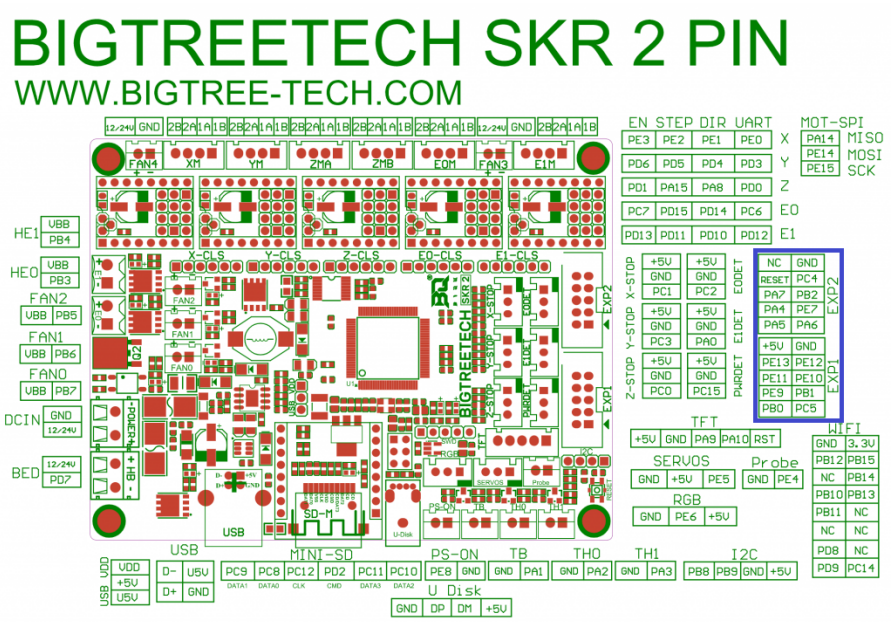


Figura 28: Pines para la conexión paralela. Bibliografía de imágenes [19]

5.6.5 Comunicación USB

Es un estándar de conexión que permite la transferencia de datos y la conexión de dispositivos electrónicos a través de un cable USB. Es ampliamente utilizado en computadoras, dispositivos móviles, periféricos y otros dispositivos electrónicos. En una comunicación USB, hay dos roles principales:

- Host (anfitrión): Es el dispositivo que controla la comunicación y proporciona la energía eléctrica. Normalmente, el ordenador o el dispositivo principal actúa como el host.
- Device (dispositivo): Es el dispositivo que se conecta al host y proporciona ciertas funcionalidades. Puede ser un teclado, un mouse, una impresora, un teléfono móvil u otros dispositivos electrónicos.

Este tipo de comunicación se basa en un modelo de transacción en el que el host y el dispositivo intercambian paquetes de datos. Estos paquetes pueden contener datos, comandos de control o

solicitudes de información. El cable USB consta de varios conductores que permiten diferentes tipos de comunicación:

- VCC: Es el conductor que suministra la energía eléctrica al dispositivo conectado.
- Data+ y Data-: Son los conductores utilizados para la transmisión de datos. La comunicación en USB es diferencial, lo que significa que los datos se transmiten en pares de señales complementarias.
- GND: Es el conductor de referencia de tierra.

Además de estos conductores básicos, un cable USB puede tener otros conductores adicionales, como los cables para la comunicación de audio, vídeo o la conexión de otros dispositivos periféricos. La placa SKR2 tiene puertos USB que pueden utilizarse para la comunicación con otros dispositivos. Se puede conectar a un ordenador para realizar la configuración, cargar firmware u otros fines de comunicación.

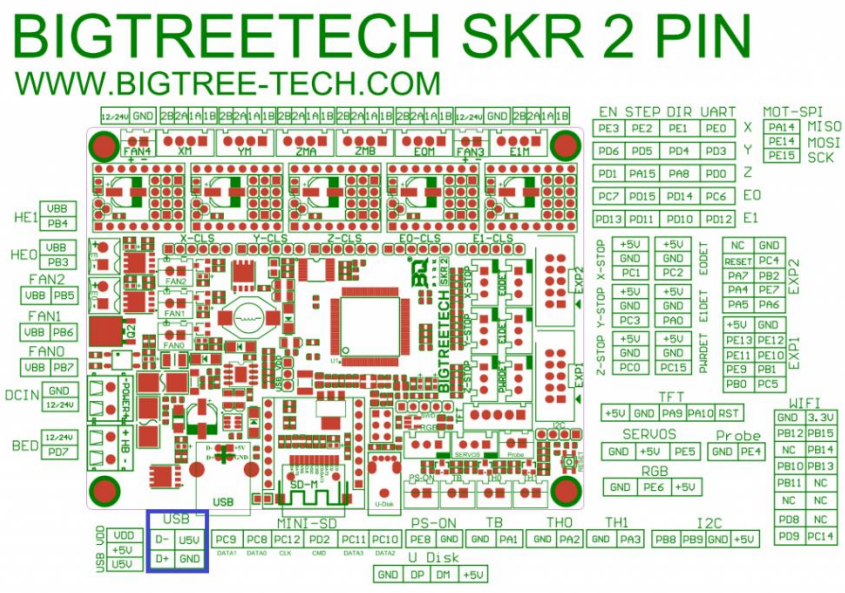


Figura 29: Pines para la conexión USB. Bibliografía de imágenes [19]

5.6.6 Pruebas realizadas para las comunicaciones con la placa BigTreeTech SKR 2

Al principio se empleó la **comunicación USB** como método de depuración para comprobar que la cinemática implementada era la correcta ya que por medio del programa Pronterface podíamos controlar los movimientos del robot. Para la configuración de esta comunicación USB se deben seguir estos pasos:

- **Conexión USB:** Asegúrarse de tener un cable USB adecuado para conectar la placa BigTreeTech SKR 2 al ordenador, en este caso el cable USB es de tipo AB (Macho/Macho). Conecta un extremo del cable al puerto USB de la placa SKR 2 y el otro extremo al puerto USB del ordenador.

- **Instalación del controlador:** En algunos casos, es posible que se necesite instalar un controlador para que el ordenador reconozca la placa BigTreeTech SKR 2. Consultar el sitio web del fabricante de la placa o buscar en línea el controlador correspondiente para el sistema operativo.

- **Descarga y ejecución de Pronterface:** Descargar el programa Pronterface desde el sitio web oficial (https://www.pronterface.com) e instalarlo en el ordenador. Pronterface es una interfaz gráfica de usuario (GUI) que permite controlar y enviar comandos a la placa BigTreeTech SKR 2.

- **Configuración de la conexión:** Abrir el programa Pronterface y buscar la opción de configuración de conexión. Normalmente, se encuentra en el menú "*Config*" o "*Settings*". Aquí, se deberá seleccionar el puerto COM correspondiente a la conexión USB de la placa BigTreeTech SKR 2. Si no se está seguro de qué puerto COM utilizar, se puede verificar en el administrador de dispositivos del sistema operativo.

- **Configuración de velocidad de comunicación:** Asegurarse de que la velocidad de comunicación (*baud rate*) en Pronterface coincida con la configuración de velocidad de la placa BigTreeTech SKR 2. La configuración que se le asignó dentro del firmware Marlin era de 115200 baudios.

- **Conexión y control de la placa:** Una vez que se ha configurado la conexión en Pronterface, se puede hacer clic en el botón "*Connect*" o "Conectar" para establecer la conexión con la placa BigTreeTech SKR 2. Si todo está configurado correctamente, se deberá ver información sobre la conexión en la interfaz.

- **Control de la placa:** Una vez conectada, se puede utilizar Pronterface para controlar la placa BigTreeTech SKR 2. A través de la interfaz, se podrá enviar comandos *gcode*, controlar los movimientos de los motores y monitorear el estado de la impresión en tiempo real.

Otro tipo de configuración que se empleó es la de **comunicación paralela**, con la que se hizo uso de una pantalla LCD para controlar la placa BigTreeTech SKR 2. Para la configuración física simplemente hay que asegurarse de tener un cable de pantalla LCD adecuado para conectar un extremo del cable a los pines EXP1 de la placa BigTreeTech SKR 2 y el otro extremo a los pines EXP3 de la pantalla LCD. En cuanto a cómo configurar lo que debe mostrar la pantalla ya se detalló en el apartado 5.4.2.

Por último, la comunicación que más se va a usar por comodidad es la **conexión UART**, ya que gracias a este tipo de conexión podemos conectar la placa BigTreeTech SKR 2con la placa Orange Pi zero, lo cual nos va a permitir controlar el robot de forma remota sin ningún cable de conexión entre las placas y el ordenador. Para la configuración física se seguirán los siguientes pasos:

- **Habilitar los pines GPIO:** Para poder usar los pines GPIO de la placa Orange Pi zero, es necesario soldar dos tiras de pines para las dos columnas de pines GPIO.
- **Conexión por cable:** Por medio de 4 cables conectaremos ambas placas de la siguiente forma: el pin de 5v de la Orange Pi zero con el pin TFT de 5v de la placa BigTreeTech SKR 2, después el pin de GND de la Orange Pi zero con el pin TFT de GND de la placa BigTreeTech SKR 2, lo siguiente será conectar el pin RX de la Orange Pi zero con el pin TFT de TX de la placa BigTreeTech SKR 2 y por último el pin TX de la Orange Pi zero con el pin TFT de RX de la placa BigTreeTech SKR 2.

Una vez las tengamos conectadas físicamente, debemos realizar la configuración dentro del firmware de cada placa que se detalló en el apartado 5.4.2 para la BigTreeTech SKR 2 y 5.5 para la Orange Pi zero.

5.7 Generación del GCODE

En el contexto de nuestro proyecto, donde se busca generar scripts con comandos *gcode* para la realización de pruebas y comprobaciones, se evaluaron diversas opciones para seleccionar la herramienta más adecuada. Entre las diferentes posibilidades, se decidió utilizar el software Inkscape, el cual será utilizado como una herramienta CNC para la generación de *gcode* y la verificación de resultados.

5.7.1 Programas para la generación de GCODE

Un posible programa para la generación de Gcode es el **PrusaSlicer**, el cual es un software de código abierto desarrollado por Prusa Research. Es una herramienta potente y versátil que permite convertir modelos 3D en formatos como STL, OBJ o AMF en instrucciones detalladas de impresión para la impresora.

El programa ofrece una interfaz intuitiva y fácil de usar, que permite a los usuarios configurar diversos parámetros de impresión para obtener resultados precisos y de alta calidad. Algunas de las características principales de **PrusaSlicer** incluyen:

- **Configuración de impresora:** ofrece una amplia gama de perfiles de impresora predefinidos para las impresoras Prusa y otros modelos populares. También permite la configuración manual de las características específicas de la impresora, como tamaño de la cama, tipo de extrusor, temperaturas, velocidades, etc.
- **Opciones de relleno y soporte:** El software permite personalizar la densidad del relleno de los objetos impresos y genera automáticamente estructuras de soporte para aquellos elementos que los requieran, lo que facilita la impresión de geometrías complicadas y mejora la calidad del resultado final.
- **Visualización y reparación de modelos:** ofrece una vista previa en 3D del modelo importado, lo que permite a los usuarios detectar posibles problemas y realizar reparaciones, como la eliminación de mallas no deseadas, la corrección de problemas de geometría o la optimización de la orientación del modelo.
- **Control de capas y ajustes de calidad:** El software permite ajustar la altura de capa, la velocidad de impresión, la calidad de acabado y otros parámetros para obtener resultados específicos según las necesidades del usuario.
- **Simulación de impresión:** incluye una función de simulación de impresión que muestra una representación visual de cómo se imprimirá el modelo en la impresora, lo que permite detectar posibles problemas antes de iniciar la impresión real.
- **Configuraciones avanzadas:** ofrece una amplia gama de opciones avanzadas para usuarios experimentados, como ajustes de algoritmo de relleno, compensación de encogimiento,

configuración de secuencias de cambio de herramienta (para impresoras multifilamento) y muchas otras configuraciones que permiten un mayor control sobre el proceso de impresión.

Otro de los programas posibles para usar para la generación de *gcode* es el **Inkscape**, el cual trata de un software de diseño gráfico de código abierto que se utiliza ampliamente para crear y editar gráficos vectoriales. Aunque no es una herramienta específicamente diseñada para generar archivos *gcode*, se puede utilizar en combinación con otros complementos para generar *gcode* para cortadoras láser y máquinas de control numérico por computadora (CNC).

Inkscape ofrece una amplia gama de herramientas y funciones para crear y editar gráficos vectoriales, como formas, líneas, texto, efectos y capas. Permite importar y exportar varios formatos de archivo, incluidos formatos vectoriales comunes como SVG (*Scalable Vector Graphics*).

Para generar *gcode* utilizando Inkscape, es necesario utilizar complementos específicos desarrollados por la comunidad. Estos complementos permiten la conversión de los elementos gráficos del diseño en instrucciones *gcode* comprensibles por las máquinas CNC. Algunos de los complementos populares utilizados con Inkscape para generar *gcode* son:

- **Gcodetools:** Es un complemento para Inkscape que permite convertir diseños vectoriales en *gcode*. Proporciona opciones para definir la velocidad de corte, la profundidad de corte y otras configuraciones específicas para la máquina CNC.
- **LaserGRBL:** Este complemento está diseñado específicamente para máquinas de corte láser y permite generar *gcode* compatible con el software de control LaserGRBL. Permite configurar la potencia del láser, la velocidad de corte y otros parámetros relevantes.
- **J Tech Photonics Laser Tool:** Este complemento está dirigido a máquinas de corte láser equipadas con láseres de la marca J Tech Photonics. Permite generar *gcode* con configuraciones específicas para estos láseres.

Es importante tener en cuenta que la generación de *gcode* con Inkscape requiere una configuración adecuada y la instalación de los complementos correctos según el tipo de máquina CNC o láser

que se utilice. Además, es necesario comprender los principios básicos de la operación de las máquinas CNC y los requisitos específicos de *gcode* para cada máquina.

5.7.2 Elección del software de generación de GCODE

El programa **Inkscape** se eligió en lugar de **PrusaSlicer** para generar *gcode* en este contexto específico debido a su funcionalidad y versatilidad en la creación y edición de gráficos vectoriales. Aunque PrusaSlicer es un software altamente especializado y potente para la preparación de modelos 3D y la generación de *gcode* específicamente para impresoras 3D FDM, no es el software más adecuado para la generación de *gcode* para máquinas CNC.

Inkscape, por otro lado, es ampliamente reconocido como un software de diseño vectorial y ofrece una amplia gama de herramientas y funciones para crear y editar gráficos vectoriales. Aunque no es su función principal, se pueden utilizar complementos específicos desarrollados por la comunidad para generar *gcode* para máquinas CNC y cortadoras láser.

La elección de Inkscape se basa en su capacidad para trabajar con gráficos vectoriales y permitir la creación de diseños complejos que se pueden exportar en formatos compatibles con las máquinas CNC. Inkscape también tiene una gran comunidad de usuarios y desarrolladores que han creado y compartido complementos para la generación de *gcode* específicamente adaptados a diferentes tipos de máquinas CNC.

5.7.3 Configuración para la generación

Para configurar el programa e instalarlo se siguieron los siguientes pasos:

- **Instalación del software:** Visitar la página web oficial de Inkscape (Draw Freely | Inkscape), después ir al apartado de descargar e instalar la versión correspondiente al sistema operativo (Windows, macOS o Linux).
- **Configuración inicial:** Al ejecutar el programa, debemos ajustar la plantilla de dibujo a nuestras medidas de la cama de impresión. Para ello debemos irnos a “archivo” y clickar en propiedades del documento. Se nos abrirá una ventana donde podemos modificar los parámetros de la plantilla.

- **Importar imagen:** Para generar Gcode debemos importar una imagen. Puedes hacerlo dentro de archivo dándole a “importar…”, ahí se nos abrirá una ventana donde podremos elegir la imagen que queremos imprimir. Después aparecerá otra ventana en la cual no se modificará ningún parámetro, simplemente le daremos a “ok”.

- **Vectorizar la imagen:** Seleccionar la imagen importada e ir al menú "Trayecto". Dentro de este menú, seleccionar "Vectorizar mapa de bits". Ajustar los parámetros para obtener la forma deseada y hacer clic en "Aplicar". Esto convertirá la imagen en un objeto vectorial que se puede utilizar para generar el *gcode*.

- **Ajustar la posición de la imagen:** Asegurarse de que la imagen vectorizada esté seleccionada y utilizar la barra de herramientas superior para verificar las coordenadas X e Y de la imagen. Ajustar la posición de la imagen arrastrándola para colocarla en el origen o en la posición deseada dentro de la plantilla.

- **Aplicar desvío dinámico:** Una vez seleccionada la imagen vectorizada, ir al menú "Trayecto" nuevamente y seleccionar "Objeto a trayecto". Luego, elegir "Desvío dinámico". Esta opción ayuda a optimizar los trazos y asegurar que el *gcode* resultante sea más limpio y eficiente. El propósito de esta operación es simplificar la representación del trazo y eliminar segmentos innecesarios.

- **Puntos de orientación:** Dirigirse al menú "Extensiones" y seleccionar "Gcodetools", luego elige "Puntos de orientación". No es necesario modificar ninguna configuración en este paso, simplemente hacer clic en "Aplicar".

- **Configuración de la cortadora:** Ahora, seleccionar la imagen vectorizada nuevamente e ir al menú "Trayecto". Seleccionar "Desvío dinámico" y luego elegir nuevamente "Extensiones" > "Gcodetools". En este punto, seleccionamos “biblioteca de herramientas”, ahí elegiremos el tipo de cortadora que se usará, en nuestro caso seleccionamos "Cilindro".

- **Parámetros de la cortadora:** Cuando hayamos realizado el paso anterior nos saldrá un cuadro verde con información de la cortadora, de ahí se modificarán los parámetros "Diameter" (diámetro) y "Depth Step" (paso de profundidad), para ello nos ayudaremos de la herramienta "crear y editar objetos de texto" que se encuentra a la izquierda del programa.

- **Generar el GCode:** Una vez que se haya configurado la cortadora correctamente, seleccionar la imagen vectorizada e ir nuevamente al menú "Desvío dinámico". Luego, dirígete al menú "Extensiones" > "Gcodetools" y elegir "Trayecto a GCode". Aparecerá una ventana

emergente en la que se puede modificar las preferencias para el *gcode*, como el nombre del archivo y la ubicación para guardar el *gcode*. Antes de hacer clic en "Aplicar", asegurarse de que se está en la pestaña "Trayecto a GCode" para evitar errores.

- **Modificación del Gcode:** La extensión en la que se generaba el archivo Gcode era ".ngc" la cual no reconocía el programa OctoPrint, por lo que se tuvo que pasar a una extensión ".gcode". Además, se perseguía filtrar los parámetros gcode que se encontraban dentro de dicho archivo, por lo que se creó un programa en lenguaje javascript, lo que nos permitió realizar ambas cosas, este se encuentra en el Anexo II.

6 Resultados

6.1 Análisis y resultados durante el proceso

Durante el desarrollo y puesta en funcionamiento del robot delta rotatorio, se han obtenido diversos resultados que han sido fundamentales para lograr el éxito del proyecto. En la fase inicial de comprobación de la electrónica, se identificaron problemas de conexión en los cables de los motores, lo que generaba movimientos incorrectos en ellos. Tras revisar la conexión y ajustar adecuadamente los cables, se logró el correcto funcionamiento de los tres motores, permitiendo movimientos independientes y coordinados.

Otro resultado significativo fue la detección de un error en la conexión de la pantalla LCD, que inicialmente no se encendía. Tras una minuciosa investigación, se descubrió el problema y se realizó la corrección necesaria, permitiendo que la pantalla se encendiera correctamente y mostrara la información relevante para el control del robot.

En cuanto a la cinemática del robot, se evidenció que la configuración por defecto no correspondía al tipo de robot delta rotatorio que se estaba implementando. Se procedió a implementar la cinemática adecuada y se utilizó el software PronterFace para verificar los resultados. Sin embargo, se observaron desviaciones en los movimientos en los ejes X e Y, lo que generaba trayectorias curvas en lugar de movimientos lineales. Además, al combinar movimientos en los ejes X, Y y Z, el robot realizaba movimientos diagonales en lugar de verticales. A pesar de revisar minuciosamente los cálculos de la cinemática y no encontrar errores, se sospecha que el problema puede estar relacionado con el peso ligero del robot, lo que podría afectar la estabilidad durante los movimientos.

Por último, se logró configurar de manera exitosa el software OctoPrint y comprobar la comunicación entre la placa Orange Pi zero y la BigTreeTech SKR 2. Se estableció una conexión adecuada y se enviaron comandos de prueba, los cuales fueron aceptados correctamente por el robot.

6.2 Control del robot delta

Durante el desarrollo del proyecto, se exploraron tres métodos para controlar el robot delta rotatorio, cada uno con sus propias ventajas y características. A continuación, se detallan las diferentes formas de control y su funcionamiento:

- **Control por USB:** Este método implica conectar el robot delta al PC de control mediante una conexión USB. Para ello, se utilizó el programa PronterFace, que ofrece una interfaz amigable y funcional. El proceso de conexión se realizó seleccionando el puerto USB adecuado y estableciendo el *Baudrate* correcto. Una vez conectado, se verificó el estado de la conexión en la pestaña de comandos, donde se esperaba la aparición del mensaje "*Printer is now online*". Desde PronterFace, se pudieron realizar acciones importantes como el *homing*, el movimiento individual de cada eje con la distancia deseada en milímetros, y el envío de scripts con comandos *gcode*.

- **Control mediante pantalla LCD:** Esta opción se refiere a la interacción directa con el robot delta a través de la pantalla LCD conectada a la configuración. Al encender la pantalla, si la configuración era la correcta, se mostró una interfaz con las coordenadas en los ejes X, Y, y Z. El control de la pantalla LCD se realizó mediante su rueda incorporada en el diseño, que permitía desplazarse hacia arriba o hacia abajo en las opciones y presionar para acceder a los diferentes menús.

- **Control mediante OctoPrint:** Para este método, se puso en marcha el servidor de OctoPrint a través de la Orange Pi zero, y luego se estableció la conexión con la BigTreeTech SKR 2 seleccionando el puerto correspondiente. Una vez conectado, se pudo controlar el robot de manera similar a través de PronterFace, incluyendo la función de *homing*, el movimiento de cada eje con distancias específicas y la capacidad de enviar scripts de comandos *gcode*.

Cada uno de estos métodos presentó ventajas y limitaciones particulares. El control por USB utilizando PronterFace resultó ser una opción conveniente para ajustar la configuración y realizar movimientos precisos. La pantalla LCD proporcionó una forma sencilla y rápida de controlar el robot directamente, lo que resultó útil para algunas tareas específicas. Por otro lado, OctoPrint permitió controlar el

robot de forma remota, lo que resultó beneficioso para la supervisión y el control en aplicaciones a distancia. A continuación, accediendo en el siguiente enlace se podrán ver dos vídeos con los resultados del trabajo.

"Control del Robot Delta por medio de la pantalla LCD"

"Control del Robot Delta por medio de OctoPrint"

6.3 Propuesta de mejoras

- **Mejora M1:** La adición de un brazo adicional en el eje del efector final en un robot Delta tiene varios propósitos y beneficios. Este brazo adicional, conocido como brazo central o brazo de compensación, se extiende desde el centro de la base fija hasta el centro de la base móvil, pasando por encima de los brazos principales del robot.

  La principal razón para añadir este brazo central es mejorar la estabilidad y la precisión del robot Delta. Al tener un brazo adicional en el eje del efector final, se reduce la carga que soportan los brazos principales durante el movimiento, lo que ayuda a minimizar la flexión y la deformación. Esto contribuye a una mayor rigidez estructural y reduce las posibles vibraciones y oscilaciones, lo que resulta en una mejor precisión en los movimientos del robot.

  Además, el brazo central también ayuda a equilibrar las cargas y fuerzas aplicadas al efector final. Al distribuir la carga de forma más uniforme entre los brazos principales y el brazo central, se reduce la tensión y el desgaste en los componentes mecánicos, lo que a su vez mejora la vida útil y la durabilidad del robot.

  Otro beneficio de añadir el brazo central es la mejora en la capacidad de carga del robot Delta. Al tener un brazo adicional para soportar el efector final, se aumenta la capacidad de carga útil del robot, lo que permite manipular objetos más pesados sin comprometer la estabilidad y la precisión.

- **Mejora M2:** A pesar de haber configurado el firmware Marlin en la placa BigTreeTech SKR 2, aún existen oportunidades para optimizar su rendimiento. Se podría investigar y explorar otras opciones de firmware disponibles en el mercado que se adapten mejor a las necesidades del robot delta rotatorio. Se recomienda analizar y comparar diferentes firmwares, como RepRap o Smoothieware, para evaluar si ofrecen mejoras en términos de funcionalidad, estabilidad o rendimiento. Esta exploración adicional podría permitir obtener un firmware más eficiente y mejor adaptado al robot delta.

- **Mejora M3:** Actualmente, se utiliza la placa Orange Pi zero para establecer la comunicación remota entre el ordenador y la placa BigTreeTech SKR 2. Sin embargo, se ha observado que la Orange Pi zero presenta limitaciones en su capacidad de procesamiento y funcionamiento. Como propuesta de mejora, se sugiere investigar y considerar otras opciones de placas o dispositivos que ofrezcan una comunicación remota más estable y eficiente. Al evaluar alternativas, es importante buscar dispositivos que cumplan con los requisitos de conectividad y potencia de procesamiento necesarios para controlar el robot delta rotatorio de manera efectiva.

- **Mejora M4:** En caso de mantener la utilización de la placa Orange Pi zero para la comunicación remota, es crucial abordar el problema del sobrecalentamiento que afecta su rendimiento. Se recomienda incorporar un disipador de calor o un sistema de refrigeración adecuado para mantener la temperatura de la placa dentro de límites aceptables durante su funcionamiento. Esto ayudará a prevenir interrupciones o fallos en la comunicación remota debido al sobrecalentamiento, asegurando así un control continuo y confiable del robot delta rotatorio.

Estas propuestas de mejora buscan optimizar diferentes aspectos del proyecto, desde el firmware utilizado hasta los componentes de comunicación remota y la gestión del calor. Implementar estas mejoras permitirá maximizar el rendimiento y la estabilidad del robot delta rotatorio, garantizando una operación más eficiente y confiable en futuras aplicaciones.

7 Presupuesto

CONCEPTO	PRECIO	UNIDADES	TOTAL
COSTES MATERIALES			**120,20 €**
COMPONENTES ELECTRÓNICOS			**118,95 €**
Orange Pi zero 512MB ARM	57,20 €	1	57,20 €
Extractor ElectroDH 71500	22,13 €	1	22,13 €
Tira de pines para soldar	0,12 €	10	1,20 €
Cable USB tipo AB	1,13 €	1	1,13 €
Cable pantalla LCD	3,95 €	1	3,95 €
Cable ethernet Cat6, 4.6m	5,90 €	1	5,90 €
Cable de puente Dupont	0,59 €	20	11,80 €
Convertidor de 8 canales 3.3V 5V TXS0108E	3,95 €	1	3,95 €
Finales de carrera SS-5GL	0,93 €	3	2,79 €
Tarjeta microSD 8GB	8,90 €	1	8,90 €
Convertidor microSD a USB	6,41 €	1	6,41 €
OTROS ELEMENTOS			**1,25 €**
Cinta de embalar	1,25 €	1	1,25 €
COSTES DE EJECUCIÓN			**6.015,76 €**
TRABAJO DE INGENIERÍA			**6.015,76 €**
Horas de taller de electrónica	3,94 €	4	15,76 €
Horas de trabajo ingeniero técnico industrial	20,00 €	300	6.000,00 €

CONCEPTO			TOTAL
COSTES MATERIALES			120,20 €
COSTES DE EJECUCIÓN			6.015,76 €
	TOTAL COSTES		**6.135,96 €**
	GASTOS GENERALES (G.G.)	**13%**	797,67 €
	BENEFICIO INDUSTRIAL (B.I.)	**6%**	368,16 €
	SUMA DE G.G. Y B.I.		1.165,83 €
	COSTE ESTIMADO		**7.301,79 €**
	I.G.I.C	**7%**	511,13 €
	COSTE TOTAL DEL PROYECTO		**7.812,92 €**
EL COSTE TOTAL DEL PROYECTO ES DE SIETE MIL OCHOCIENTOS DOCE EUROS CON NOVENTA Y DOS CÉNTIMOS			

8 Conclusiones

Se ha logrado abordar y ejecutar satisfactoriamente los objetivos iniciales del proyecto, lo cual se evidencia a través de los resultados obtenidos. La elección del firmware Marlin ha sido acertada, ya que ha permitido controlar y configurar de manera efectiva la placa BigTreeTech SKR 2. La verificación exhaustiva de la electrónica y la implementación de la cinemática específica han sido pasos fundamentales para asegurar el correcto funcionamiento del robot Delta rotatorio. La integración exitosa de la placa Orange Pi Zero ha facilitado la comunicación remota y el control del robot, lo que ha ampliado las capacidades y posibilidades de uso.

La puesta en funcionamiento del robot Delta ha sido un proceso que ha implicado diversas etapas y decisiones estratégicas. Inicialmente, se consideraron diferentes opciones de firmware para la placa BigTreeTech SKR 2, entre ellas el Reprap Firmware y el firmware Marlin. Después de analizar los distintos factores, se optó por el firmware Marlin por varias razones fundamentales.

En primer lugar, se llevó a cabo una exhaustiva verificación de la electrónica, asegurándose de que estuviera correctamente implementada y funcionando adecuadamente. Para ello, se configuraron los aspectos básicos en el firmware, como la capacidad de mover cada motor de forma independiente como si fueran ejes independientes. Asimismo, se realizó una prueba de funcionalidad de la pantalla LCD, verificando su correcto funcionamiento.

Una vez confirmada la integridad de la electrónica, se procedió a implementar la cinemática específica para un robot Delta. Inicialmente, se utilizó la configuración predeterminada en el firmware. Sin embargo, al realizar las pruebas de movimiento, se detectó que la cinemática Delta por defecto en el firmware era para un robot Delta lineal, no rotatorio como el modelo en cuestión. Por lo tanto, se tuvo que modificar el firmware para adaptar una nueva cinemática acorde a las características del robot Delta rotatorio.

Durante las pruebas de movimiento en los ejes, se constató que el movimiento individual de cada eje era correcto. Sin embargo, al realizar movimientos combinados en los ejes X y Z, se observó que el

desplazamiento vertical no se realizaba de forma perpendicular, sino que adoptaba una trayectoria diagonal. A pesar de los esfuerzos realizados, no fue posible identificar la causa precisa de este problema. Como método de verificación, se decidió utilizar la funcionalidad de una máquina CNC, generando *gcode* para dibujar en el plano XY sin realizar movimientos en el eje Z.

Además, se logró la integración exitosa de la placa Orange Pi zero, lo que permitió establecer una comunicación remota con la placa BigTreeTech SKR 2. Esta integración resultó fundamental para facilitar el control y la configuración del robot Delta de manera remota.

En conclusión, la puesta en funcionamiento del robot Delta ha implicado una serie de decisiones estratégicas y ajustes técnicos para garantizar su correcto desempeño. A través de la selección adecuada del firmware, la verificación exhaustiva de la electrónica, la implementación de la cinemática específica y la integración de la placa Orange Pi zero, se ha logrado superar los desafíos y avanzar hacia el objetivo final de un robot Delta plenamente funcional y controlable. Este proyecto ha brindado una valiosa experiencia en el campo de la robótica y la ingeniería, permitiendo adquirir conocimientos técnicos y habilidades prácticas y estratégicas en el ámbito de los robots Delta y su puesta en marcha.

9 Conclusions

The initial objectives of the project have been successfully addressed and implemented, as evidenced by the results obtained. The choice of the Marlin firmware has been the right one, as it has allowed to effectively control and configure the BigTreeTech SKR 2 board. The exhaustive verification of the electronics and the implementation of the specific kinematics have been fundamental steps to ensure the correct operation of the rotating Delta robot. The successful integration of the Orange Pi Zero board has facilitated remote communication and control of the robot, which has expanded the capabilities and possibilities of use.

The implementation of the Delta robot has been a process that has involved several stages and strategic decisions. Initially, different firmware options were considered for the BigTreeTech SKR 2 board, including Reprap Firmware and Marlin firmware. After analysing the various factors, the Marlin firmware was chosen for several key reasons.

First, a thorough verification of the electronics was carried out, making sure that they were correctly implemented and working properly. To do this, basic aspects were configured in the firmware, such as the ability to move each motor independently as if they were independent axes. In addition, a functionality test of the LCD screen was performed, verifying its correct operation.

Once the integrity of the electronics was confirmed, we proceeded to implement the specific kinematics for a Delta robot. Initially, the default configuration in the firmware was used. However, when performing the motion tests, it was detected that the default Delta kinematics in the firmware was for a linear Delta robot, not a rotating one like the model in question. Therefore, the firmware had to be modified to adapt a new kinematics according to the characteristics of the rotating Delta robot.

During the movement tests on the axes, the individual movement of each axis was found to be correct. However, when performing combined movements on the X and Z axes, it was observed that the vertical displacement was not perpendicular, but adopted a diagonal path. Despite the efforts made, it was not possible to identify the precise cause of this problem. As a verification method, it was decided to

use the functionality of a CNC machine, generating gcode for drawing in the XY plane without Z-axis movements.

In addition, the successful integration of the Orange Pi zero board was achieved, allowing remote communication with the BigTreeTech SKR 2 board. This integration was essential to facilitate remote control and configuration of the Delta robot.

In conclusion, the implementation of the Delta robot involved a series of strategic decisions and technical adjustments to ensure its correct performance. Through the appropriate selection of the firmware, the exhaustive verification of the electronics, the implementation of the specific kinematics and the integration of the Orange Pi zero board, it has been possible to overcome the challenges and move towards the final goal of a fully functional and controllable Delta robot. This project has provided valuable experience in the field of robotics and engineering, allowing the acquisition of technical knowledge and practical and strategic skills in the field of Delta robots and their implementation.

10 Bibliografía

10.1 Bibliografía de información

1. Historia de la robótica: de Arquitas de Tarento al robot Da Vinci (Parte I) (isciii.es)
2. T4397 (isciii.es)
3. Robots Delta… ¿qué, cómo, cuándo y por qué? - Urany®
4. Movimiento DELTA y su aplicación en la robótica – Máquinas de medición por coordenadas (maquinasdemedicionporcoordenadas.com)
5. KR DELTA - Robot higiénico | KUKA AG
6. Robot Delta - Wikipedia, la enciclopedia libre
7. Software para robotica | Programa 100% Gratuito ✔ | igus
8. Microsoft PowerPoint - T5 CINEMATICA OCW_Revision.pptx (ehu.eus)
9. v25n42a07.pdf (scielo.org.co)
10. Qué es Python, para qué sirve y cómo se usa (+ recursos para aprender) (hubspot.es)
11. C++: Qué es, Para qué sirve, Ventajas y Desventajas (workana.com)
12. MATLAB - El lenguaje del cálculo técnico (mathworks.com)
13. es/ROS/Introduccion - ROS Wiki
14. BigTreeTech SKR 2: Características y configuración de esta nueva y potente electrónica (3dwork.io)
15. La placa Orange Pi Zero, nueva competidora de Raspberry Pi Zero | TECNOLOGÍA | ComputerWorld
16. Driver TMC2208 V3.0 BIGTREETECH - driver stepper para impresora 3D (dhm-online.com)
17. https://revistas.itc.edu.co/index.php/letras/article/download/128/123/
18. https://hypertriangle.com/~alex/delta-robot-tutorial/

10.2 Bibliografía de imágenes

1. https://es.wikipedia.org/wiki/C%C3%ADborg
2. https://commons.wikimedia.org/wiki/File:Robot_Delta_X.jpg
3. https://www.researchgate.net/figure/A-DELTA-robot-with-linear-actuators_fig1_220061904
4. https://cl.urany.net/blog/robots-delta-qu%C3%A9-c%C3%B3mo-cu%C3%A1ndo-y-por-qu%C3%A9
5. https://urany.net/blog/robots-delta-qu%C3%A9-c%C3%B3mo-cu%C3%A1ndo-y-por-qu%C3%A9
6. https://www.researchgate.net/figure/Figura-27-El-robot-Delta-en-aplicaciones-quirurgicas_fig7_39425275
7. https://new.abb.com/news/es/detail/93548/abb-lanza-el-robot-delta-de-cinco-ejes-mas-rapido-de-su-clase-para-la-recogida-el-embalaje-y-la-reorientacion-de-productos-ligeros
8. https://www.xataka.com/aplicaciones/python-basic-nuestra-era-lenguaje-programacion-moda-ieee-spectrum-todos-quieren-conocerlo
9. https://www.genbeta.com/desarrollo/c-ha-sido-lenguaje-programacion-2022-tiobe-ocho-mejores-cursos-para-aprenderlo-gratis-0
10. https://techcommunity.microsoft.com/t5/azure-high-performance-computing/matlab-and-azure-a-match-made-in-performance-heaven/ba-p/3788737
11. https://www.directindustry.es/prod/automationware/product-192516-2394360.html
12. https://3dwork.io/skr-2-fallo-rev-a/
13. https://3dwork.io/guia-completa-skr2/
14. https://www.amazon.es/Orange-512MB-consejo-desarrollo-Frambuesa/dp/B085MH1FNG
15. https://www.amazon.es/Orange-512MB-consejo-desarrollo-Frambuesa/dp/B085MH1FNG
16. https://www.gams3d.com/producto/driver-bigtreetech-tmc-2208-v3/
17. https://www.sciencedirect.com/science/article/pii/S1405774315000074
18. https://hypertriangle.com/~alex/delta-robot-tutorial/

19. https://3dwork.io/guia-completa-skr2/

11 Anexo I: Código modificado del firmware Marlin

11.1 Configuration.h

```
// Choose the name from boards.h that matches your setup
#ifndef MOTHERBOARD
  #define MOTHERBOARD BOARD_BTT_SKR_V2_0_REV_B
#endif
```

```
#define SERIAL_PORT 1 //TODO: Estaba en 0

/**
 * Serial Port Baud Rate
 * This is the default communication speed for all serial ports.
 * Set the baud rate defaults for additional serial ports below.
 *
 * 250000 works in most cases, but you might try a lower speed if
 * you commonly experience drop-outs during host printing.
 * You may try up to 1000000 to speed up SD file transfer.
 *
 * :[2400, 9600, 19200, 38400, 57600, 115200, 250000, 500000, 1000000]
 */
#define BAUDRATE 115200 //TODO: Estaba en 250000

//#define BAUD_RATE_GCODE     // Enable G-code M575 to set the baud rate

/**
 * Select a secondary serial port on the board to use for communication with the host.
 * Currently Ethernet (-2) is only supported on Teensy 4.1 boards.
 * :[-2, -1, 0, 1, 2, 3, 4, 5, 6, 7]
 */
#define SERIAL_PORT_2 -1 //TODO: Estaba comentado y estaba en -1
```

```
#define X_DRIVER_TYPE  TMC2208
#define Y_DRIVER_TYPE  TMC2208
#define Z_DRIVER_TYPE  TMC2208
```

```
#define DEFAULT_AXIS_STEPS_PER_UNIT   { 203, 203, 203}
```

```
#define DEFAULT_MAX_FEEDRATE          { 500, 500, 500}
```

```
#define DEFAULT_MAX_ACCELERATION      { 1000, 1000, 1000}
```

```
#define DEFAULT_ACCELERATION          1000
#define DEFAULT_RETRACT_ACCELERATION  3000
#define DEFAULT_TRAVEL_ACCELERATION   1000
```

```
#define DELTA //TODO: Estaba descomentado
#if ENABLED(DELTA)

  #define DeltaROT
  //TODO: Datos para la nueva kinematic
  #define BaseEfector 205; //Efector final
  #define BaseFija 280; //Base fija
  #define Largo_Antebrazo 250; //Largo antebrazo
  #define Largo_bicep 220; //Largo biceps


  // Make delta curves from many straight lines (linear interpolation).
  // This is a trade-off between visible corners (not enough segments)
  // and processor overload (too many expensive sqrt calls).
  #define DEFAULT_SEGMENTS_PER_SECOND 200

  // After homing move down to a height where XY movement is unconstrained
  //#define DELTA_HOME_TO_SAFE_ZONE

  // Delta calibration menu
  // Add three-point calibration to the MarlinUI menu.
  // See http://minow.blogspot.com/index.html#4918805519571907051
  #define DELTA_CALIBRATION_MENU //TODO: Estaba comentado
```

```
#define DELTA_AUTO_CALIBRATION //TODO: Estaba comentado

#if ENABLED(DELTA_AUTO_CALIBRATION)
  // Default number of probe points : n*n (1 -> 7)
  #define DELTA_CALIBRATION_DEFAULT_POINTS 4
#endif

#if EITHER(DELTA_AUTO_CALIBRATION, DELTA_CALIBRATION_MENU)
  // Step size for paper-test probing
  #define PROBE_MANUALLY_STEP 0.05      // (mm)
#endif

// Print surface diameter/2 minus unreachable space (avoid collisions with vertical towers).
#define DELTA_PRINTABLE_RADIUS 125.0    // (mm) //TODO: Estaba en 140

// Maximum reachable area
#define DELTA_MAX_RADIUS       125.0    // (mm) //TODO: Estaba en 140

// Center-to-center distance of the holes in the diagonal push rods.
#define DELTA_DIAGONAL_ROD 250.0        // (mm)

// Distance between bed and nozzle Z home position
#define DELTA_HEIGHT 110.00             // (mm) Get this value from G33 auto calibrate //TODO: Estaba en 250

#define DELTA_ENDSTOP_ADJ { 0.0, 0.0, 0.0 } // Get these values from G33 auto calibrate
```

```
// Horizontal distance bridged by diagonal push rods when effector is centered.
#define DELTA_RADIUS 125.0              // (mm) Get this value from G33 auto calibrate //TODO: Estaba en 124

// Trim adjustments for individual towers
// tower angle corrections for X and Y tower / rotate XYZ so Z tower angle = 0
// measured in degrees anticlockwise looking from above the printer
#define DELTA_TOWER_ANGLE_TRIM { 0.0, 0.0, 0.0 } // Get these values from G33 auto calibrate
```

```
#define CLASSIC_JERK //TODO: Estaba comentado
#if ENABLED(CLASSIC_JERK)
  #define DEFAULT_XJERK 5.0 //TODO: Estaba en 10
  #define DEFAULT_YJERK 5.0 //TODO: Estaba en 10
  #define DEFAULT_ZJERK  5.0 //TODO: Estaba en 0.3
  //#define DEFAULT_IJERK  0.3
  //#define DEFAULT_JJERK  0.3
  //#define DEFAULT_KJERK  0.3
  //#define DEFAULT_UJERK  0.3
  //#define DEFAULT_VJERK  0.3
  //#define DEFAULT_WJERK  0.3

  //#define TRAVEL_EXTRA_XYJERK 0.0     // Additional jerk allowance for all travel moves

  //#define LIMITED_JERK_EDITING        // Limit edit via M205 or LCD to DEFAULT_aJERK * 2
  #if ENABLED(LIMITED_JERK_EDITING)
    #define MAX_JERK_EDIT_VALUES { 20, 20, 0.6, 10 } // ...or, set your own edit limits
  #endif
#endif

#define DEFAULT_EJERK    1.0  // May be used by Linear Advance //TODO: Estaba en 5
```

```
//#define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN
```

```
// Direction of endstops when homing; 1=MAX, -1=MIN
// :[-1,1]
#define X_HOME_DIR 1 //TODO: Estaba en -1
#define Y_HOME_DIR 1 //TODO: Estaba en -1
#define Z_HOME_DIR 1 //TODO: Estaba en -1
```

```
// The size of the printable area
#define X_BED_SIZE ((DELTA_PRINTABLE_RADIUS) * 2) //TODO: Estaba en 200, pero para la config del home es 125
#define Y_BED_SIZE ((DELTA_PRINTABLE_RADIUS) * 2) //TODO: Estaba en 200, pero para la config del home es 125

// Travel limits (linear=mm, rotational=°) after homing, corresponding to endstop positions.
#define X_MIN_POS -(DELTA_PRINTABLE_RADIUS) //TODO: Estaba en 0, pero para la config del home es -125
#define Y_MIN_POS -(DELTA_PRINTABLE_RADIUS) //TODO: Estaba en 0, pero para la config del home es -125
#define Z_MIN_POS 0 //TODO: Estaba en 0
#define X_MAX_POS DELTA_PRINTABLE_RADIUS //TODO: Estaba en X_BED_SIZE
#define Y_MAX_POS DELTA_PRINTABLE_RADIUS //TODO: Estaba en Y_BED_SIZE
#define Z_MAX_POS MANUAL_Z_HOME_POS //TODO: Estaba en 200, pero para la config del home es 110
```

```
#define BED_CENTER_AT_0_0 //TODO: Estaba comentado

// Manually set the home position. Leave these undefined for automatic settings.
// For DELTA this is the top-center of the Cartesian print volume.
//#define MANUAL_X_HOME_POS 100 //TODO: Estaba comentado y con valor 0
//#define MANUAL_Y_HOME_POS 100 //TODO: Estaba comentado y con valor 0
#define MANUAL_Z_HOME_POS DELTA_HEIGHT //TODO: Estaba comentado y con valor 0, pero para la config del home era 210
```

```
#define HOMING_FEEDRATE_MM_M { (10*100), (10*100), (10*100) }
```

```
#define LCD_LANGUAGE es
```

```
#define CR10_STOCKDISPLAY
```

11.2 Configuration_adv.h

```
#define HOMING_BUMP_MM       { 50, 50, 50 }
#define HOMING_BUMP_DIVISOR { 2, 2, 2 }
```

```
#if HAS_MANUAL_MOVE_MENU
  #define MANUAL_FEEDRATE { 50*60, 50*60, 50*60}
```

```
#define LCD_INFO_MENU //TODO: Estaba sin definir, es decir estaba comentado
#if ENABLED(LCD_INFO_MENU)
  #define LCD_PRINTER_INFO_IS_BOOTSCREEN // Show bootscreen(s) instead of Printer Info pages
#endif
```

```
#if EITHER(ARC_SUPPORT, BEZIER_CURVE_SUPPORT)
  #define CNC_WORKSPACE_PLANES      // Allow G2/G3/G5 to operate in XY, ZX, or YZ planes
#endif
```

```
#if AXIS_IS_TMC_CONFIG(X)
  #define X_CURRENT       800        // (mA) RMS current. Multiply by 1.414 for peak current.
  #define X_CURRENT_HOME  X_CURRENT  // (mA) RMS current for sensorless homing
  #define X_MICROSTEPS     64        // 0..256
  //TODO: La linea de arriba estaba a 16
  #define X_RSENSE          0.11     // Multiplied x1000 for TMC26X
  #define X_CHAIN_POS      -1        // -1..0: Not chained. 1: MCU MOSI connected. 2: Next in chain, ...
  //#define X_INTERPOLATE  true      // Enable to override 'INTERPOLATE' for the X axis
  //#define X_HOLD_MULTIPLIER 0.5    // Enable to override 'HOLD_MULTIPLIER' for the X axis
#endif
```

11.3 Settings.cpp

```
#if IS_KINEMATIC
  segments_per_second = DEFAULT_SEGMENTS_PER_SECOND;
  #if ENABLED(DELTA)
    const abc_float_t adj = DELTA_ENDSTOP_ADJ, dta = DELTA_TOWER_ANGLE_TRIM, ddr = DELTA_DIAGONAL_ROD_TRIM_TOWER;
    delta_height = DELTA_HEIGHT;
    delta_endstop_adj = adj;
    delta_radius = DELTA_RADIUS;
    delta_diagonal_rod = DELTA_DIAGONAL_ROD;
    delta_tower_angle_trim = dta;
    delta_diagonal_rod_trim = ddr;

    //TODO: Se añadio para el calculo de la cinematica
    BE = BaseEfector;
    BF = BaseFija;
    LargoA = Largo_Antebrazo;
    LargoB = Largo_bicep;

    sqrt3 = sqrt(3.0) ; //sqrt(3.0);
    pi = 3.141592653;
    sin120 = sqrt3/2.0;
    cos120 = -0.5;
    tan60 = sqrt3;
    sin30 = 0.5;
    tan30 = 1/sqrt3;
```

11.4 Delta.h

```
//TODO: Nueva configuración
extern float BE;
extern float BF;
extern float LargoA;
extern float LargoB;

//TODO: Constantes trigonométricas
extern float sqrt3; //sqrt(3.0);
extern float pi;
extern float sin120;
extern float cos120;
extern float tan60;
extern float sin30;
extern float tan30;
```

```
#if ENABLED(DeltaROT)
  #define DELTA_IK(V) delta.set(V.a, V.b, V.c)

#else
  #define DELTA_IK(V) delta.set(DELTA_Z(V, A_AXIS), DELTA_Z(V, B_AXIS), DELTA_Z(V, C_AXIS))
  // Macro to obtain the Z position of an individual tower
  #define DELTA_Z(V,T) V.z + SQRT(          \
    delta_diagonal_rod_2_tower[T] - HYPOT2( \
      delta_tower[T].x - V.x,              \
      delta_tower[T].y - V.y               \
    )                                      \
  )
#endif

//TODO: inverse kinematics
// helper functions, calculates angle theta1 (for YZ-pane)
int delta_calcAngleYZ(float x0, float y0, float z0, float &theta);

void inverse_kinematics(const xyz_pos_t &raw); //TODO: la funcion era inversa_kinematics(const xyz_pos_t &raw)
```

```
#if ENABLED(DeltaROT)
    //TODO: forward kinematics: (theta1, theta2, theta3) -> (x0, y0, z0)
  void forward_kinematics(float theta1, float theta2, float theta3);
```

11.5 Delta.cpp

```
//TODO: Nueva configuración
float BE;
float BF;
float LargoA;
float LargoB;

//TODO: Constantes trigonométricas
float sqrt3; //sqrt(3.0);
float pi;
float sin120;
float cos120;
float tan60;
float sin30;
float tan30;
```

```
int delta_calcAngleYZ(float x0, float y0, float z0, float &theta) {

float y1 = -0.5 * tan30 * BF; // BF/2 * tg 30
y0 -= 0.5 * tan30 * BE; // shift center to edge

// z = a + b*y
float a = (x0*x0 + y0*y0 + z0*z0 +LargoB*LargoB - LargoA*LargoA - y1*y1)/(2*z0);
float b = (y1-y0)/z0;

// discriminant
float d = -(a+b*y1)*(a+b*y1)+LargoB*(b*b*LargoB+LargoB);
if (d < 0) return -1; // non-existing point
float yj = (y1 - a*b - sqrt(d))/(b*b + 1); // choosing outer point
float zj = a + b*yj;
theta = 180.0*atan(-zj/(y1 - yj))/pi + ((yj>y1)?180.0:0.0);

return 0;
}
```

```
void inverse_kinematics(const xyz_pos_t &raw) {

  abc_pos_t angulos;

  float theta1;
  float theta2;
  float theta3;

  theta1 = theta2 = theta3 = 0;

  float x0, y0, z0;

  x0 = raw.x;
  y0 = raw.y;
  z0 = raw.z; //z - delta home position

  int status = delta_calcAngleYZ(x0, y0, z0, theta1);
  if (status == 0) status = delta_calcAngleYZ(x0*cos120 + y0*sin120, y0*cos120-x0*sin120, z0, theta2);
  if (status == 0) status = delta_calcAngleYZ(x0*cos120 - y0*sin120, y0*cos120+x0*sin120, z0, theta3);

  angulos.a = -theta1; //Se pone -
  angulos.b = -theta2;
  angulos.c = -theta3;

  DELTA_IK(angulos);
}
```

```
void forward_kinematics(float theta1, float theta2, float theta3) {

  float x0, y0, z0;

  float t = (BF-BE)*tan30/2;
  float dtr = pi/(float)180.0;

  theta1 *= dtr;
  theta2 *= dtr;
  theta3 *= dtr;

  float y1 = -(t + LargoB*cos(theta1));
  float z1 = -LargoB*sin(theta1);
  float y2 = (t + LargoB*cos(theta2))*sin30;
  float x2 = y2*tan60;
  float z2 = -LargoB*sin(theta2);
  float y3 = (t + LargoB*cos(theta3))*sin30;
  float x3 = -y3*tan60;
  float z3 = -LargoB*sin(theta3);
  float dnm = (y2-y1)*x3-(y3-y1)*x2;
  float w1 = y1*y1 + z1*z1;
  float w2 = x2*x2 + y2*y2 + z2*z2;
  float w3 = x3*x3 + y3*y3 + z3*z3;
```

```
// x = (a1*z + b1)/dnm
float a1 = (z2-z1)*(y3-y1)-(z3-z1)*(y2-y1);
float b1 = -((w2-w1)*(y3-y1)-(w3-w1)*(y2-y1))/2.0;

// y = (a2*z + b2)/dnm;
float a2 = -(z2-z1)*x3+(z3-z1)*x2;
float b2 = ((w2-w1)*x3 - (w3-w1)*x2)/2.0;

// a*z^2 + b*z + c = 0
float a = a1*a1 + a2*a2 + dnm*dnm;
float b = 2*(a1*b1 + a2*(b2-y1*dnm) - z1*dnm*dnm);
float c = (b2-y1*dnm)*(b2-y1*dnm) + b1*b1 + dnm*dnm*(z1*z1 - LargoA*LargoA);

// discriminant
float d = b*b - (float)4.0*a*c;
//if (d < 0) return -1; // non-existing point

#if(d >= 0)

  z0 = -(float)0.5*(b+sqrt(d))/a;
  x0 = (a1*z0 + b1)/dnm;
  y0 = (a2*z0 + b2)/dnm;

  //TODO: Revisar si los valores son los correctos para pasarlos cartes
  cartes.set(x0, y0, z0);
```

```
// At least one carriage has reached the top.
// Now re-home each carriage separately.
homeaxis(A_AXIS);
homeaxis(B_AXIS);
homeaxis(C_AXIS);

endstops.validate_homing_move();
```

11.6 Motion.cpp

```
void get_cartesian_from_steppers() {
  #if ENABLED(DELTA)
    forward_kinematics(planner.get_axis_position_degrees(A_AXIS), planner.get_axis_position_degrees(B_AXIS),
    planner.get_axis_position_degrees(C_AXIS)); //TODO: Estaba forward_kinematics(planner.get_axis_position_mm());
```

12 Anexo II: Código para filtrar scripts con comandos gcode

```
const fs = require('fs');

// Ruta del archivo de entrada
const filePath = './GcodesPrueba/Circulo.ngc';

// Ruta del archivo de salida
const outputPath = './GcodesPrueba/Circulo_filtrado.gcode';

// Leer el archivo
fs.readFile(filePath, 'utf8', (err, data) => {
  if (err) {
    console.error('Error al leer el archivo:', err);
    return;
  }

  // Separar el contenido en líneas
  const lines = data.split('\n');

  // Filtrar las líneas que empiezan por G00, G01, G02 o G03
  const filteredLines = lines.filter(line => line.startsWith('G00') || line.startsWith('G01') ||
  line.startsWith('G02') || line.startsWith('G03'))
    .map(line => {
      const match = line.match(/(X[-\d]+\.[-\d]+ Y[-\d]+\.[-\d]+)(?: Z[-\d]+\.[-\d]+ I[-\d]+\.[-\d]+ J[-\d]+\.[-\d]+)?/);
      return match ? `${line.substring(0, 3)} ${match[0]}` : null;
    })
    .filter(line => line !== null);
```

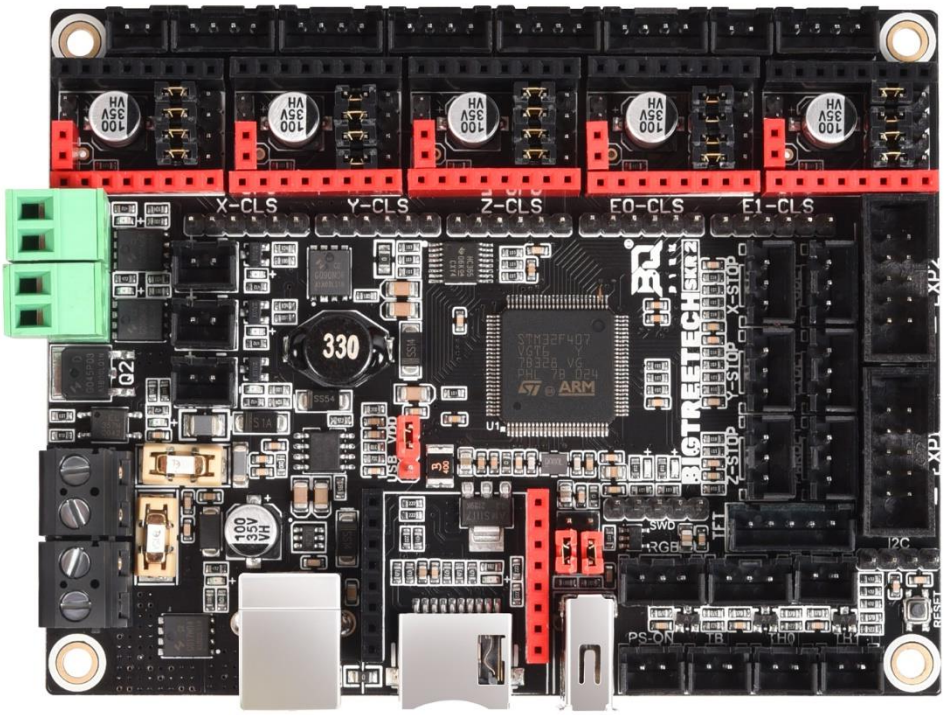
```
  // Añade saltos de línea
  const outputWithZ = filteredLines.join('\n');

  // Eliminar las coordenadas Z
  const outputData = outputWithZ.replace(/Z[\d.]+\s/g, '');

  // Guardar el contenido en el archivo de salida
  fs.writeFile(outputPath, outputData, 'utf8', err => {
    if (err) {
      console.error('Error al guardar el archivo de salida:', err);
      return;
    }
    console.log('El archivo de salida se ha guardado correctamente.');
  });
});
```

13 Anexo III: Datasheets de los componentes

Shenzhen BIGTREE technology co., LTD.
BIG TREE TECH

BIGTREETECH SKR 2

User Guide



I. Introduction to SKR Motherboard

The BIGTREETECH SKR 2 motherboard is a 32-bit 3D printer motherboard designed and manufactured by the 3D printing team of Shenzhen BIQU Technology Co., Ltd. It is a successor to the SKR1.4 and SKR1.4 Turbo series of motherboards and adds many features that we are sure the community will love.

1. Main board features:

1) Uses a 32-bit ARM Cortex-M4 series STM32F407VGT6 main control chip with a core frequency of *168MHz* that packs enough performance to run even the most demanding UI while ensuring stutter free printing.
2) Integrates a AOZ1284PI power regulator which supports 12-24V power input and a maximum output current of 4A – enough to power external components such as LEDs and even a raspberry pi;
3) Support all versions of the company's serial screen, SPI screen and LCD screen;
4) Upgrade the configuration firmware via SD card. This burning method is simple, convenient and efficient;
5) TMC Motor drivers can be used in SPI or UART mode by simply adjusting the onboard jumpers beneath each motor driver. Additionally you can connect or disconnect the TMC DIAG pin using an onboard jumper allowing you to use hard endstops or sensorless homing without the need to cut any pins.
6) Supports functions such as power loss print resume, filament runout detection, shutdown after printing, BLtouch and other ABL sensors, RGB lights, etc. Note that external modules will be required for many of these functions;
7) Use high-performance MOSFET to reduce heat generation;
8) The use of replaceable fuses makes the replacement process more convenient;
9) Includes a protection circuit on thermistor inputs which protects against short circuits between the heater cartridge and the bed heating element. This is a common mistake made by many users when replacing a nozzle or working on the hotend so we expect this feature to be a welcome one;
10) Includes an additional heater protection circuit which protects against runaway heating on the bed and hotend heaters. By default, the heaters are off so if the MCU or MOSFETs are damaged, the heaters will shut down instead of enter into thermal runaway;
11) Introduces a new “anti-reversal” stepper driver protection which is achieved through a combination of new hardware and Marlin firmware. This protects against motherboard or driver damage due to incorrect driver insertion;
12) ESP8266 WIFI module (ESP12S or ESP-07) interface for customers to use RRF firmware;
13) Onboard push-pull TF card slot (SDIO working mode) and U disk interface;

14) A filter circuit is added to the power input terminal to reduce ripple and noise interference;
15) The fuses now use a smaller package which leaves more space on the PCB for more features and makes it easier to swap them out;
16) The number of PWM controllable fan interfaces has been increased from one to three.

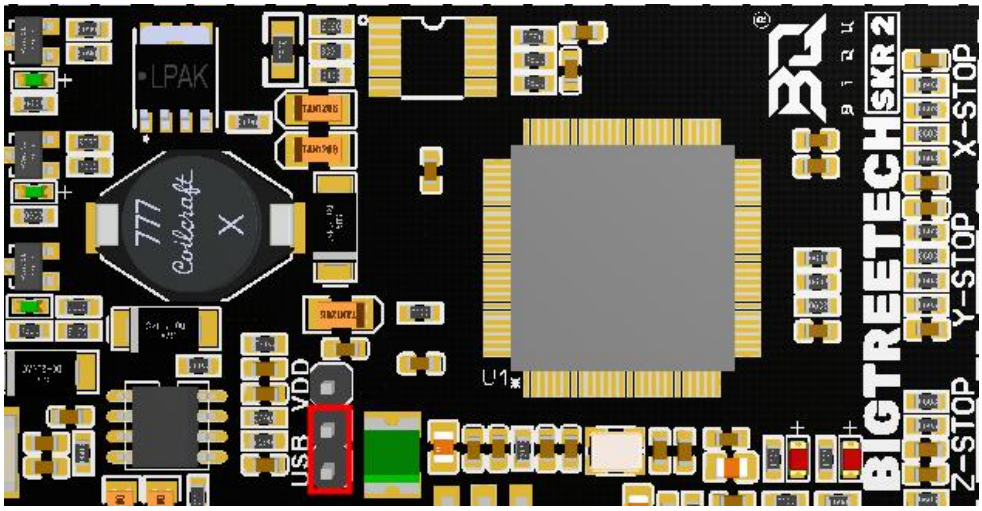
2. Main board parameters:

Appearance size: 110*85mm For details, please refer to: BIGTREETECH SKR 2-SIZE.pdf
Installation size: 102*76mm
Microprocessor: ARM Cortex-M4 CPU STM32F407VGT6
Input voltage: DC12V-DC24V
Logic voltage: DC 3.3V
Firmware support: Marlin, Reprap Firmware
WIFI interface: ESP-12S, ESP-07S
Fan interface: three CNC fans, two normally closed fans (non-controllable)
Expansion interface: I2C, Servos, Probe, PS-ON, PWR-DET, Fil-DET, RGB, etc.
Motor driver: support TMC5160, TMC2209, TMC2225, TMC2226, TMC2208, TMC2130, ST820, LV8729, DRV8825, A4988, etc., and can be externally connected to a motor drive
Drive working mode support: SPI, UART, STEP/DIR
Motor drive interface: X, Y, Z (double Z axis), E0, E1, five channels (each channel has a closed loop drive interface)
Temperature sensor interface: TH0, TH1, TB, 3 channels 100K NTC (thermal resistance)
Display: serial touch screen, SPI touch screen, LCD display
PC communication interface: square USB, easy to plug and unplug, communication baud rate 115200
Support file format: G-code
Support machine structure: XYZ, delta, kossel, Ultimaker, corexy
Recommended software: Cura, Simplify3D, pronterface, Repetier-host, Makerware

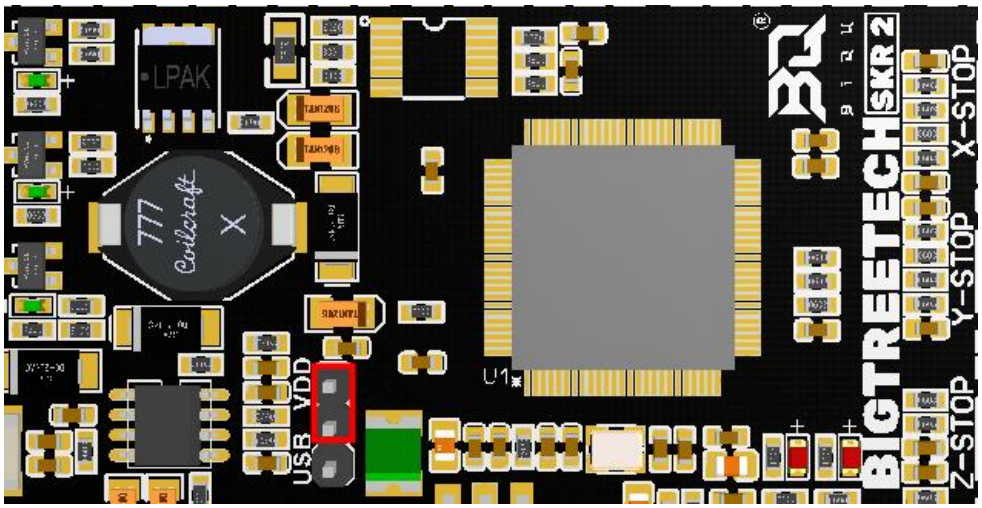
II. Power on the motherboard

After the SKR motherboard is powered on, the D6 LED will light up, indicating that the power supply is at nominal levels. The 5V SEL jumper, in the middle of the board, allows you to select whether you would like to use the 5V USB power input or the onboard 5V regulated supply. Configure the jumper as shown below:

1) When using USB to power the motherboard:

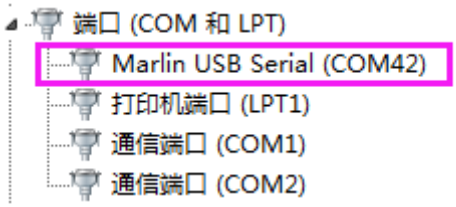


2) When using 12V-24V power supply:



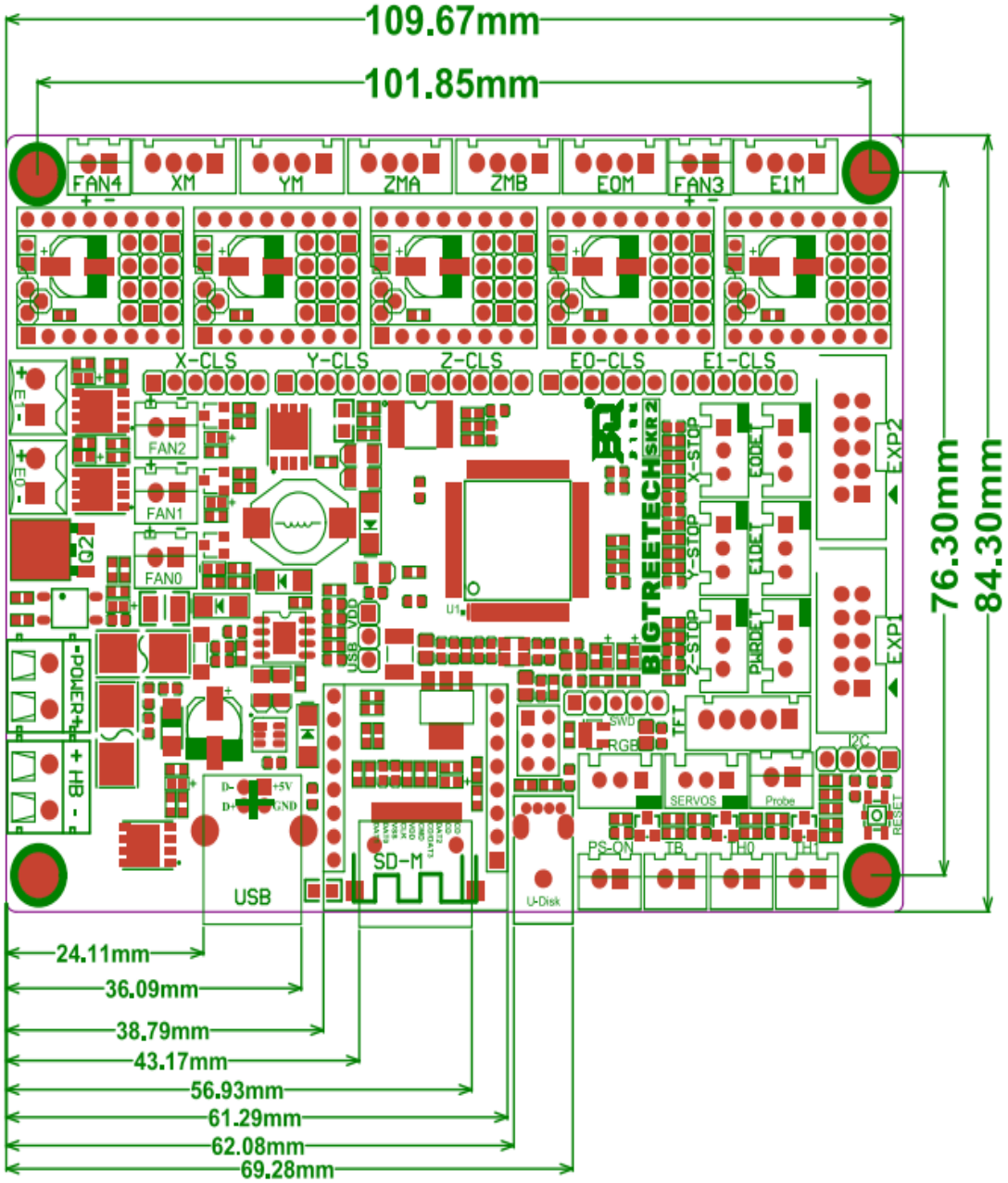
III. Communication between motherboard and computer

When using Marlin2.0 firmware the motherboard will enumerate as a virtual serial port in both macOS and windows. An example of the enumerated device within the windows device manager is shown below.

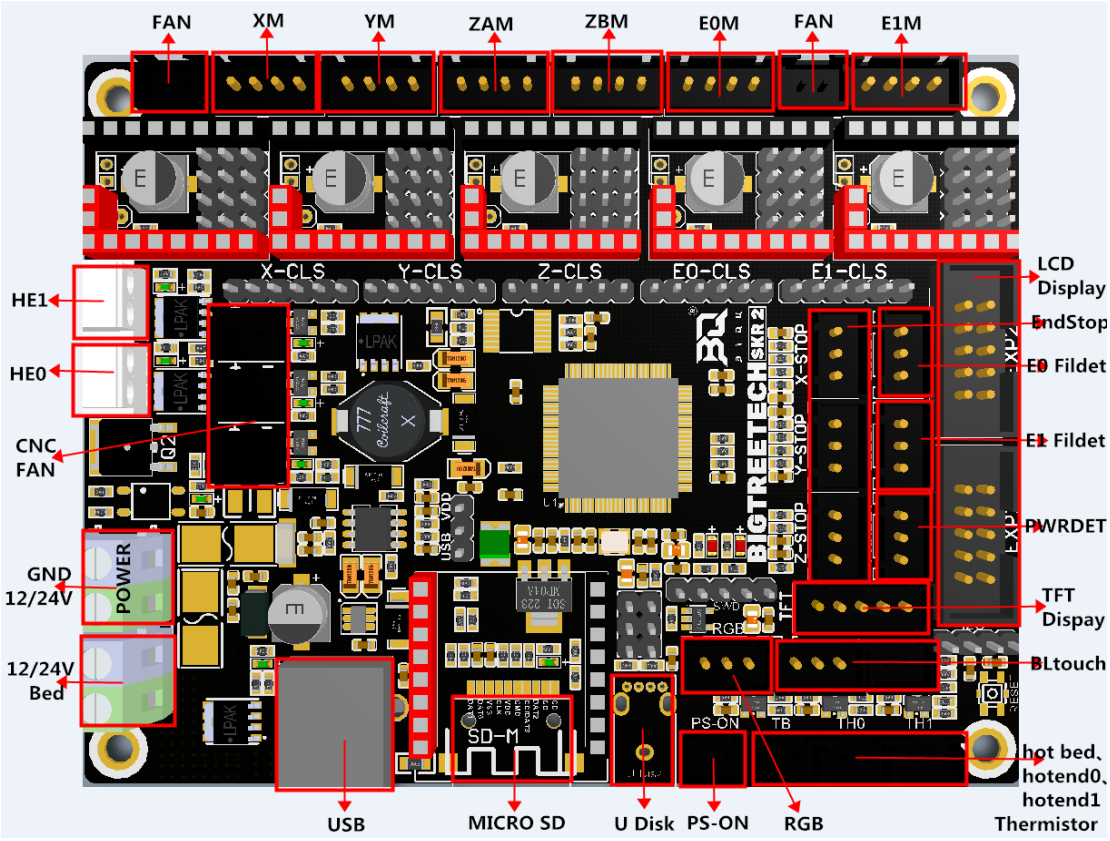


IV. Motherboard interface description

1. Motherboard size:



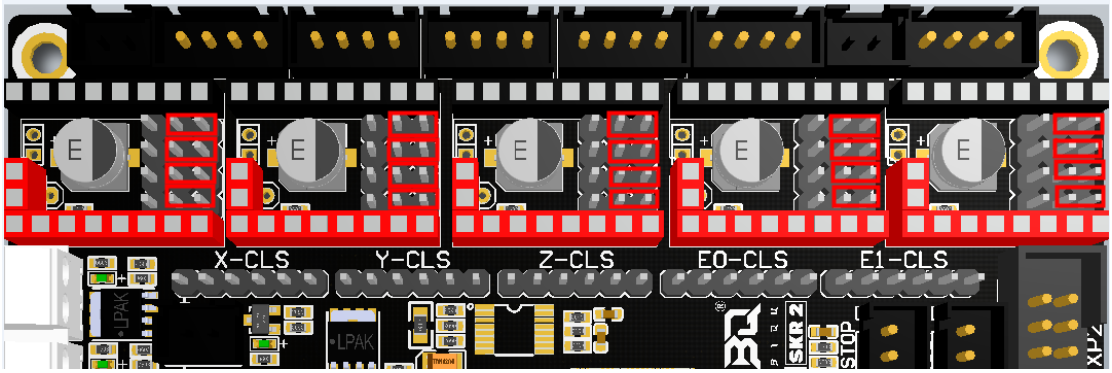
2. Motherboard wiring diagram:



3. Selection method of drive working mode:

① Normal STEP/DIR mode: (such as: A4988, DRV8825, LV8729, ST820, etc.) according to the drive subdivision table to select the shorting cap to short-circuit MS0-MS2.

Driver - STEP/DIR MODE



Note: If you use A4988 or DRV8825 driver, you must short-circuit RST and SLP with a jumper cap to work normally.

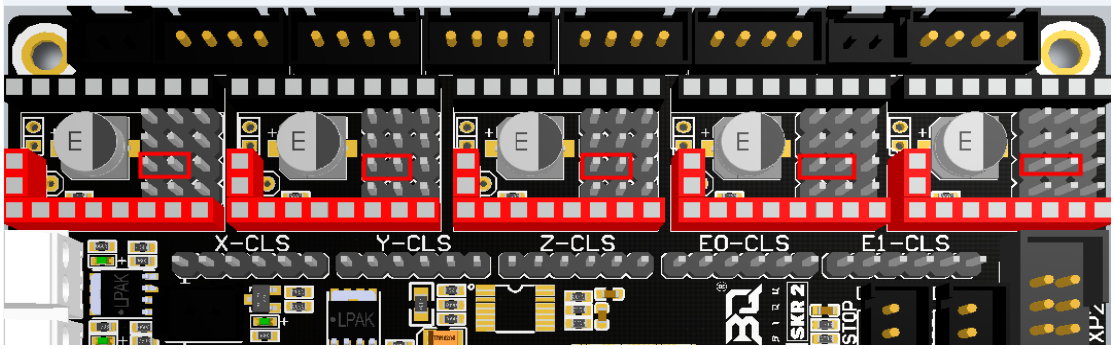
驱动芯片	MS1	MS2	MS3	细分	Excitation Mode
A4988 最大 16 细分 35V 2A	L	L	L	Full Step	2 Phase
	H	L	L	1/2	1-2 Phase
	L	H	L	1/4	W1-2 Phase
	H	H	L	1/8	2W1-2 Phase
	H	H	H	1/16	4W1-2 Phase
驱动电流计算公式 Rs=0.1Ω	Imax = Vref / (8 * Rs)				

驱动芯片	MD3	MD2	MD1	细分	Excitation Mode
LV8729 最大 128 细分 36V 1.8A	L	L	L	Full Step	2 Phase
	L	L	H	1/2	1-2 Phase
	L	H	L	1/4	W1-2 Phase
	L	H	H	1/8	2W1-2 Phase
	H	L	L	1/16	4W1-2 Phase
	H	L	H	1/32	8W1-2 Phase
	H	H	L	1/64	16W1-2 Phase
	H	H	H	1/128	32W1-2 Phase
驱动电流计算公式 Rs=0.22Ω	$I_{OUT} = (VREF / 5) / RF1$				

驱动芯片	MODE2	MODE1	MODE0	细分	Excitation Mode
DRV8825 最大 32 细分 8.2V-45V 2.5A at 24V T=25℃	L	L	L	Full Step	2 Phase
	L	L	H	1/2	1-2 Phase
	L	H	L	1/4	W1-2 Phase
	L	H	H	1/8	
	H	L	L	1/16	
	H	L	H	1/32	
	H	H	L	1/32	
	H	H	H	1/32	
驱动电流计算公式 Rs=0.1Ω	$I_{CHOP} = \frac{V_{REFX}}{5 \bullet R_{ISENSE}}$				

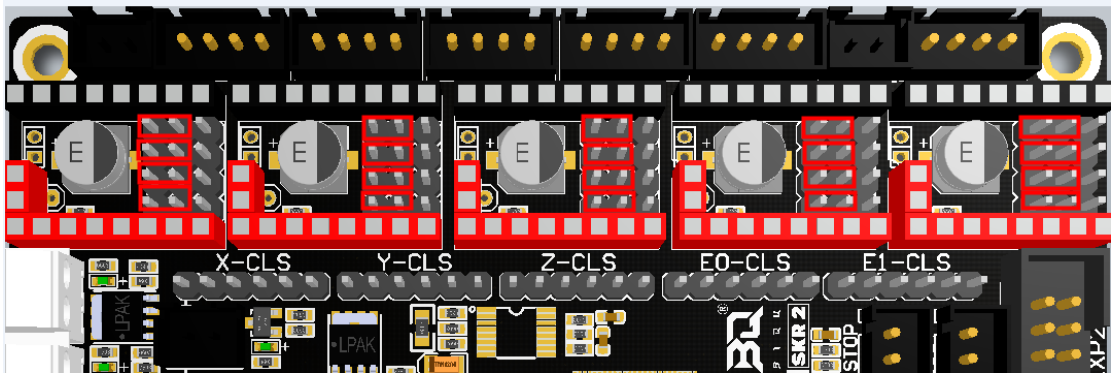
② UART mode driven by TMC: (such as: TMC2208, TMC2209, TMC2225, etc.) Each axis uses a shorting cap to short the position of the red box in the figure, and the subdivision and drive current are set by firmware.

Driver - UART MODE



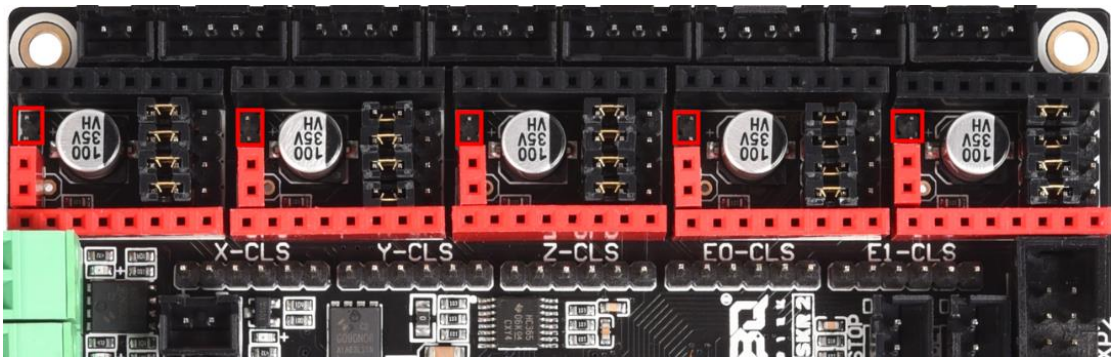
③ SPI mode of TMC drive: (such as: TMC2130, TMC5160, TMC5161, etc.) Use four shorting caps for each axis to short the position of the red box in the figure, and the subdivision and drive current can be set by firmware.

Driver - SPI MODE

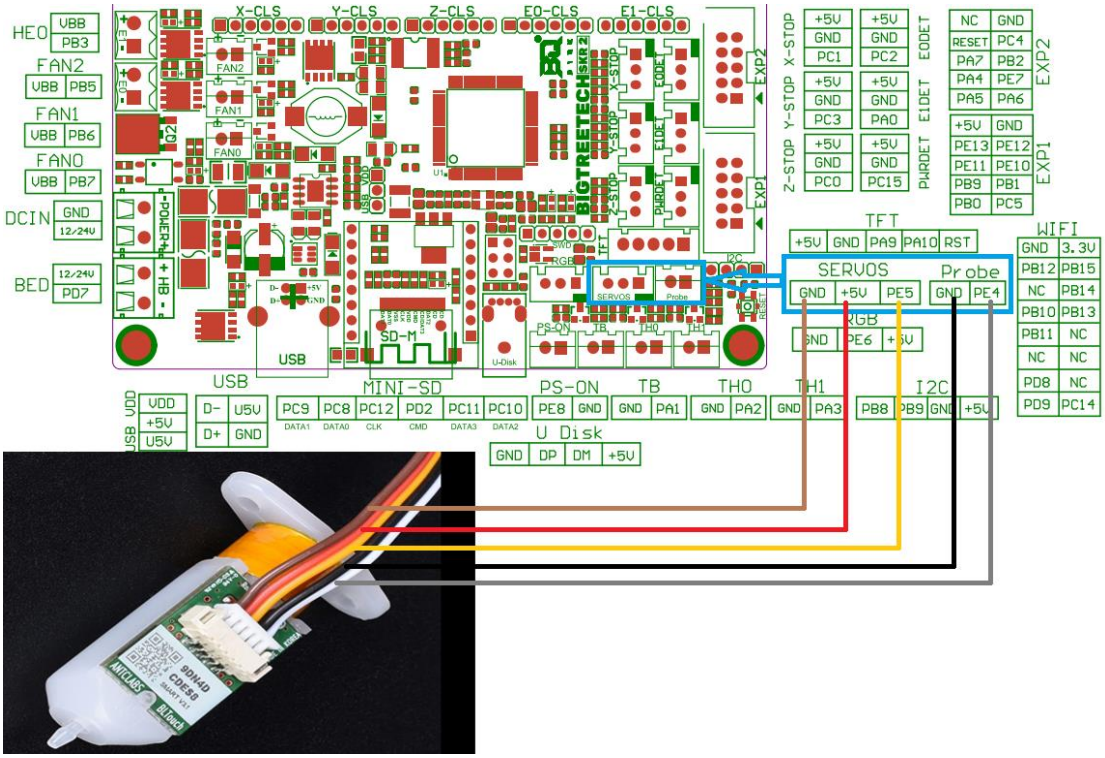


④DIAG Pin of TMC drive:

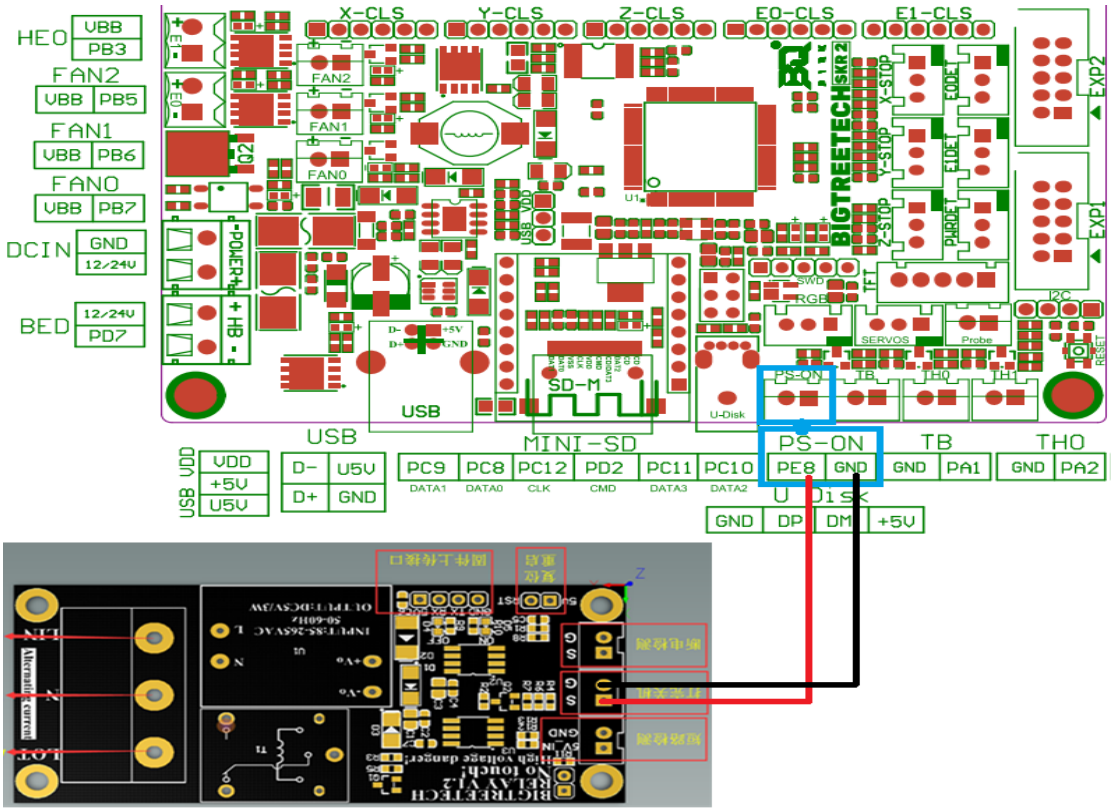
As shown in the picture, insert the short-circuit cap when using the sensorlesshoming function, and do not insert it when not using it. There is no need to cut the driving diag pin.



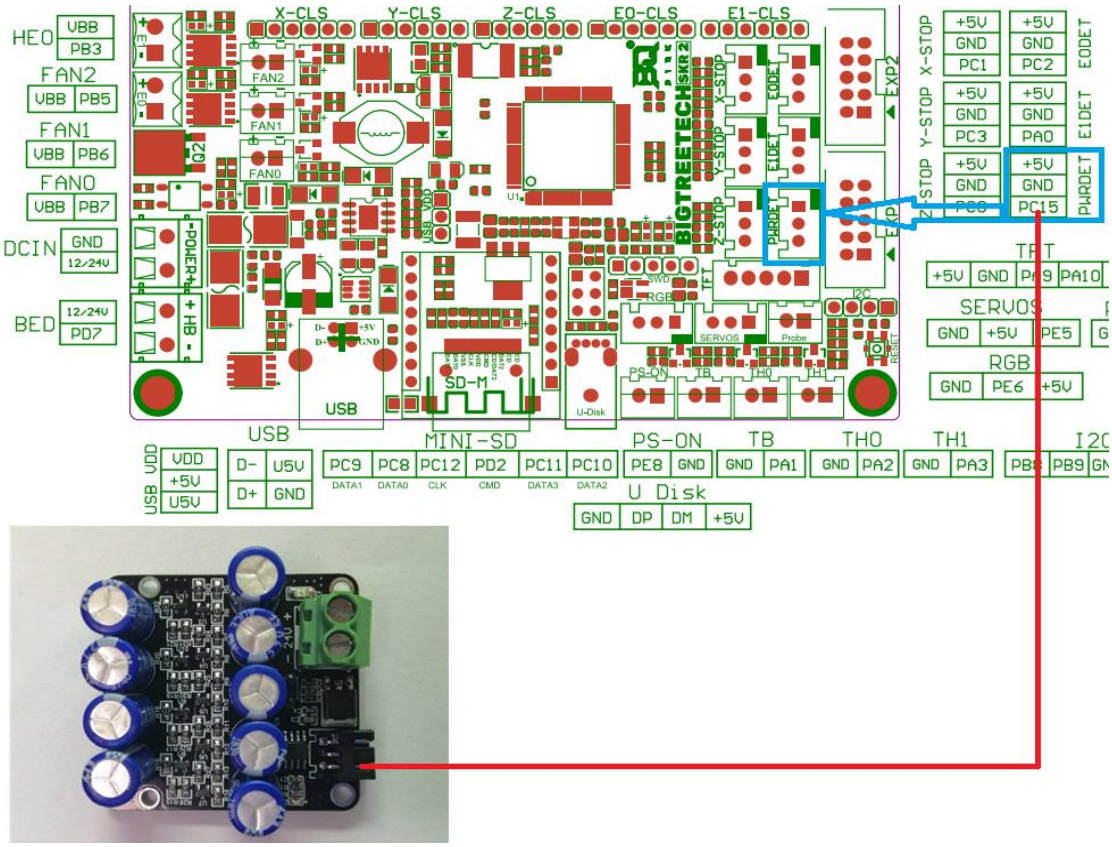
4. The connection between BIGTREETECH SKR 2 and BLtouch:



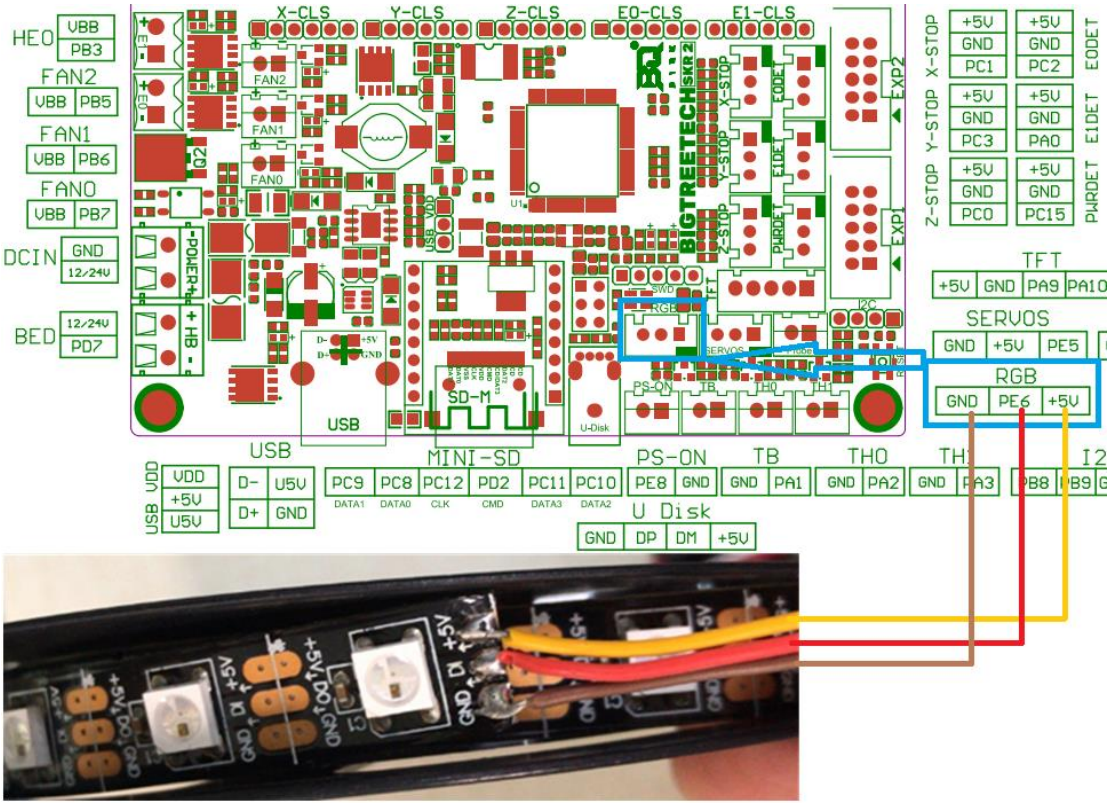
5. Connection between BIGTREETECH SKR 2 and shutdown (Relay V1.2):



6. The connection between BIGTREETECH SKR 2 and power failure (UPS 24V V1.0):



7. Connection between BIGTREETECH SKR 2 and RGB-LED drive:



8.Connecting the BIGTREETECH SKR 2 to a Raspberry Pi using TTL UART:

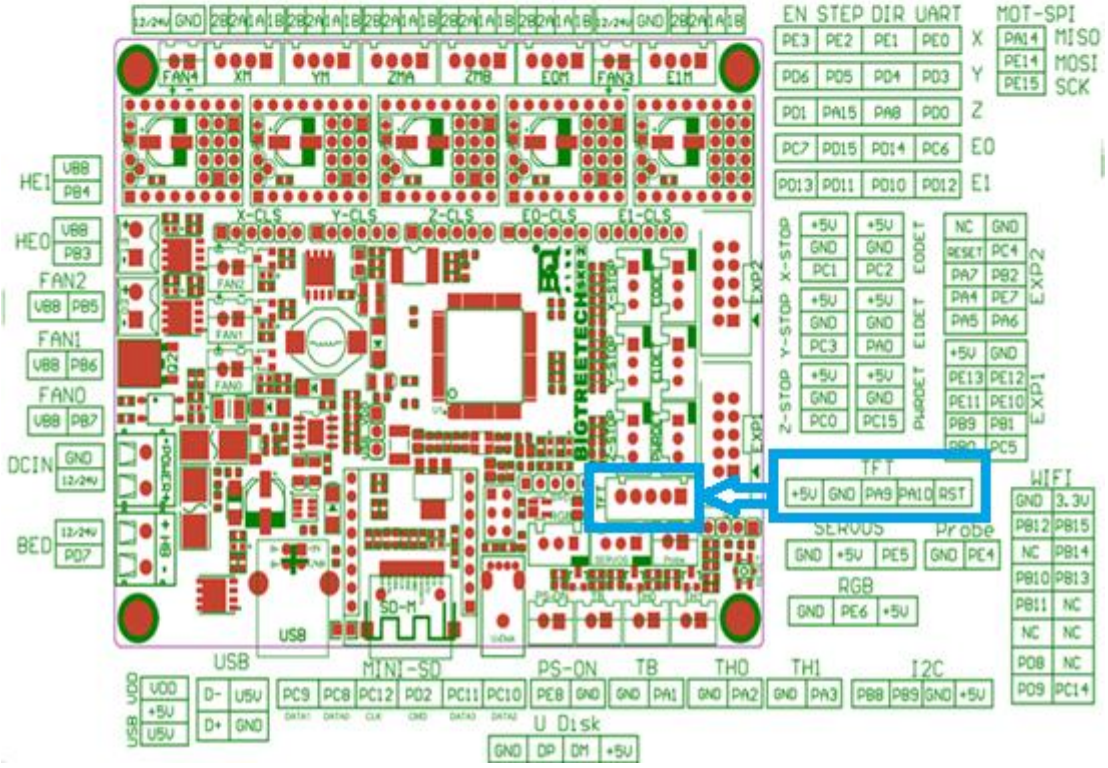
No need to connect V+

Rasberry Pi 3 and SKR V1.4 both with 3.3V logic.

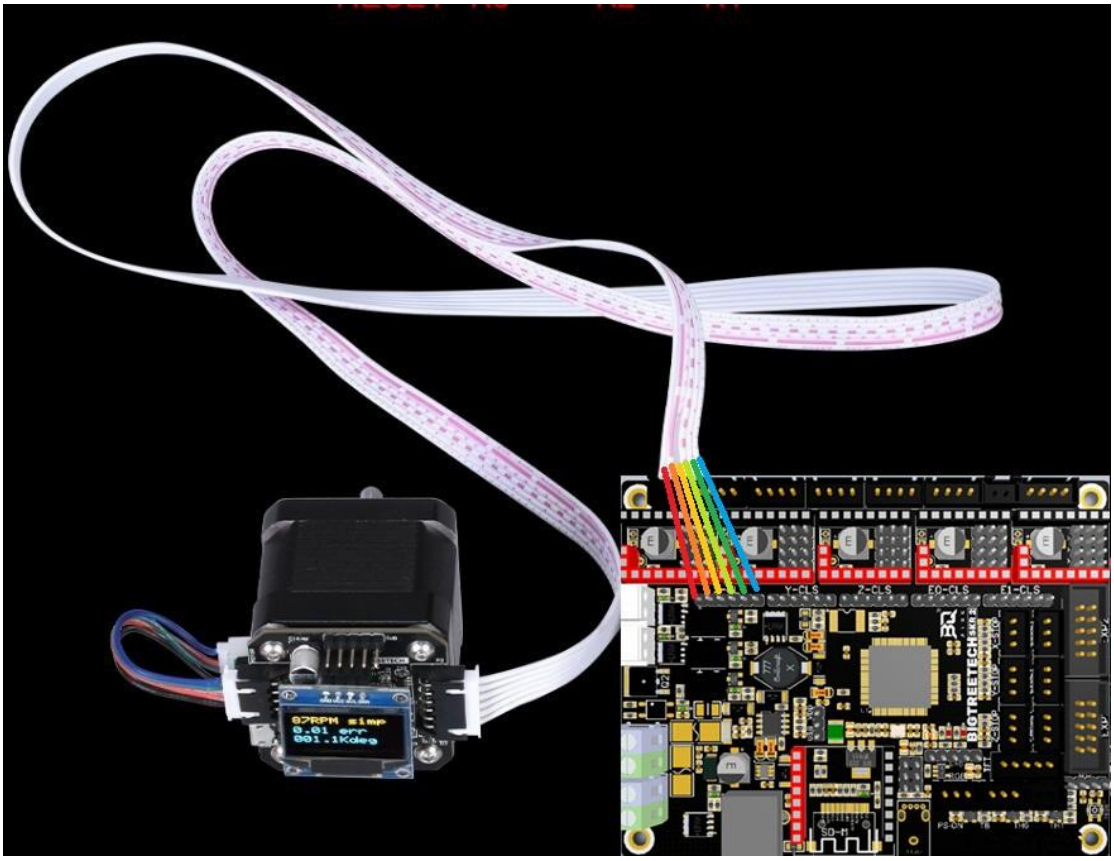
PI 3B+ @ GPIO connector TXD0 - RXD0 – GND (8-0-10)

		Pin no.			
DC Power	3.3V	1	2	5V	DC Power
SDA1, I²C	GPIO 2	3	4	5V	DC Power
SCL1, I²C	GPIO 3	5	6	GND	
GPIO_GCLK	GPIO 4	7	8	GPIO 14	TXD0
	GND	9	10	GPIO 15	RXD0
GPIO_GEN0	GPIO 17	11	12	GPIO 18	GPIO_GEN1
GPIO_GEN2	GPIO 27	13	14	GND	
GPIO_GEN3	GPIO 22	15	16	GPIO 23	GPIO_GEN4
DC Power	3.3V	17	18	GPIO 24	GPIO_GEN5
SPI_MOSI	GPIO 10	19	20	GND	
SPI_MISO	GPIO 9	21	22	GPIO 25	GPIO_GEN6
SPI_CLK	GPIO 11	23	24	GPIO 8	SPI_CE0_N
	GND	25	26	GPIO 7	SPI_CE1_N
I²C ID EEPROM	DNC	27	28	DNC	I²C ID EEPROM
	GPIO 5	29	30	GND	
	GPIO 6	31	32	GPIO 12	
	GPIO 13	33	34	GND	
	GPIO 19	35	36	GPIO 16	
	GPIO 26	37	38	GPIO 20	
	GND	39	40	GPIO 21	

BIGTREETECH SKR 2 @ TFT connector Tx-Rx-GND

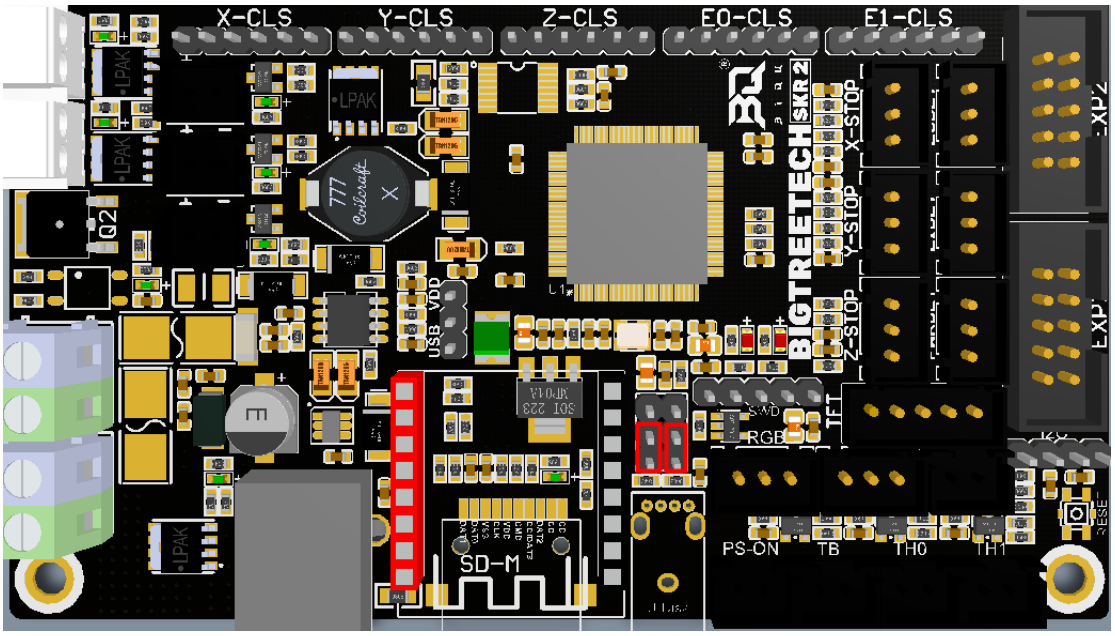


9. Connection between BIGTREETECH SKR 2 and closed-loop drive:

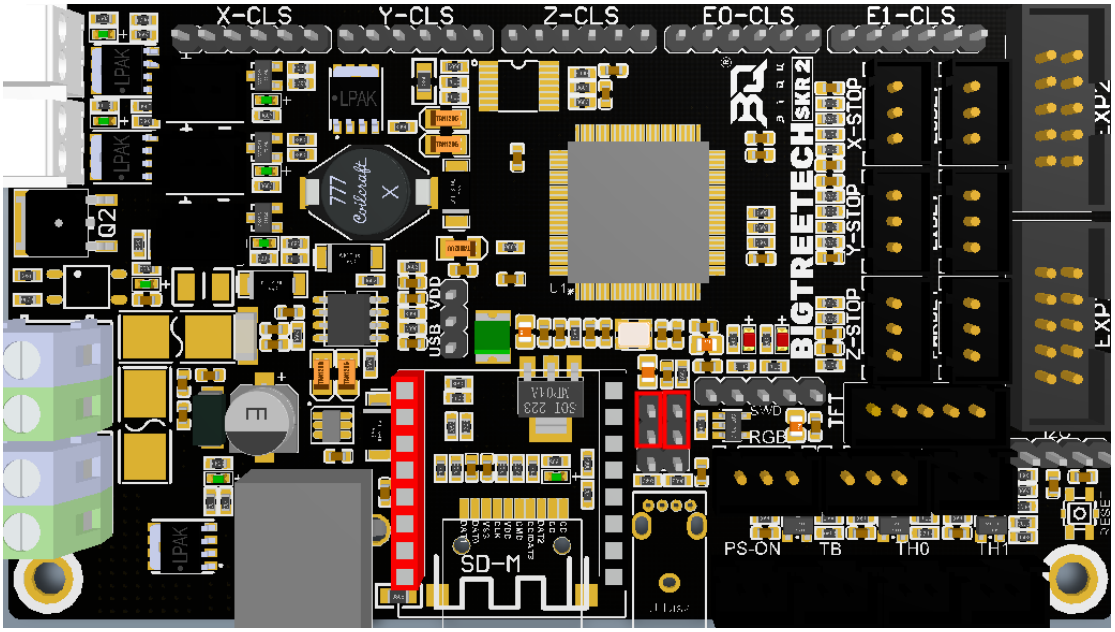


10. U disk function and RRF WIFI function selection:

①U disk function: When the U disk function is selected, the WIFI module only works in UART mode. At this time, RRF firmware cannot be used, only Marlin firmware can be used;



②RRF WIFI function: If selected (as per the jumper configuration below), the U disk function will not be available as WIFI will work through SPI mode, and RRF firmware can be used.



V. Motherboard firmware description

The motherboard comes pre-installed with firmware that is used for testing at the factory. The firmware is configured for use with an i3 style printer however you will likely find it useful to configure and install your own firmware.

Due to the abundance of printer types and firmware distributions in the market, BIGTREETECH cannot offer customized firmware for each application. You may, however, be able to find firmware for you application by trying one of the following:

Ask customer service or technical personnel to obtain;

Compile your own using a distribution of your choice.

Download from our original website:https://github.com/bigtreetech

If you decide to use Marlin then you will need to download the source code, use Visual Studio Code to configure the code to your application, find the firmware.bin file within the pio-build folder, copy it to the SD card and then reset the motherboard. There is an abundance of information on the internet which elaborates on these steps should you need more guidance.

If you require more information about the pin numbers when compiling your firmware, please refer to the BIGTREETECH SKR 2-PIN.pdf document.

VI. Matters needing attention:

1. The jumper that allows selection between USB power or on-board power must be inserted so as to select one of the two options otherwise no power will be supplied to the logic section of the motherboard and the power LED will not light up.

2. The power of the hot bed connected to the main board must be less than or equal to 10A. If you want to use a high-power hot bed, it is recommended to choose a hot bed with 24V power supply and use 24V to power the main board;

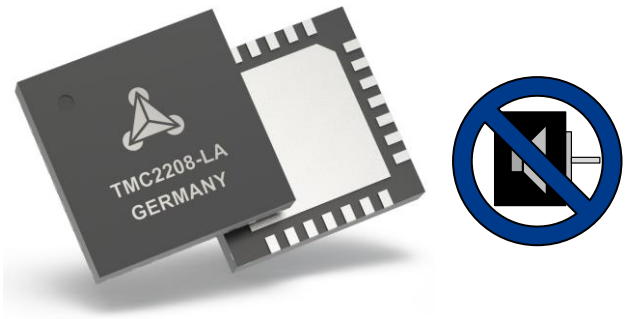
3. The jumpers that allow selection between SPI WiFi (for RRF) or U disk functionality should be inserted to select at least one option.;

4.The driver anti-reversal insertion function is a newly developed function from BIGTREETECH. It is currently only supported in Marlin firmware and therefore not available when using RRF firmware. Therefore, when you are not using Marlin firmware, please carefully check whether the driver is inserted correctly to avoid causing damage to the driver and/or the motherboard. Customers can also choose to install the Marlin firmware first and then install the RRF firmware after they are confident that the hardware installation is correct.

5. This motherboard uses a push-pull type SD slot which does not eject when pushing on the card after insertion. The installation stroke is far shorter than a push-push type SD slot and therefore customers should be gentle when inserting or removing a card. Returns for damage caused by mis-handling the SD card will not be entertained.
d

TMC2208/2 & TMC2224/0/5 family Datasheet

TMC2202, TMC2208, TMC2220, TMC2224, TMC2225 Step/Dir Drivers for Two-Phase Bipolar Stepper Motors up to 2A peak - stealthChop™ for Quiet Movement - UART Interface Option.



Applications

Compatible Design Upgrade
3D Printers
Printers, POS
Office and home automation
Textile, Sewing Machines
CCTV, Security
ATM, Cash recycler
HVAC

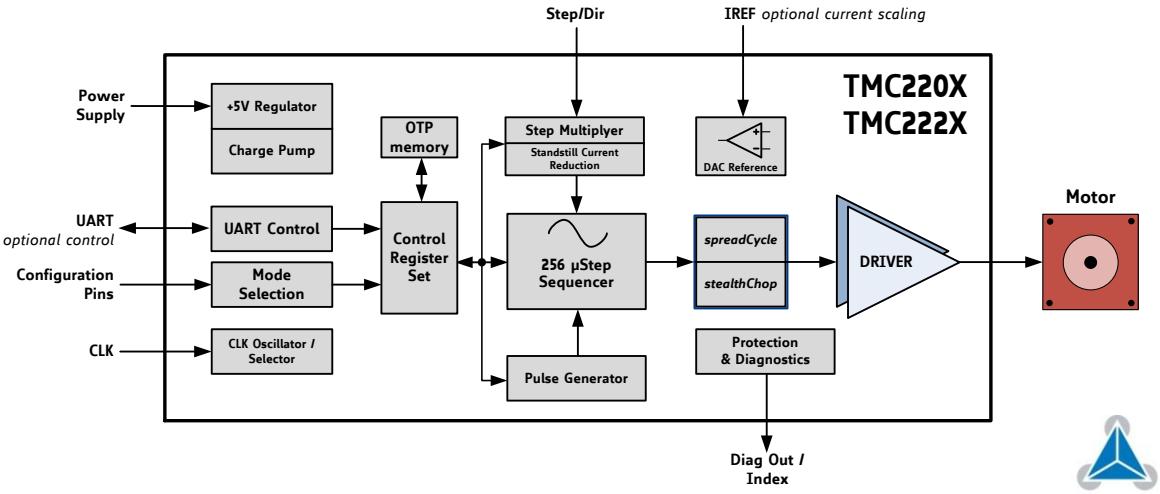
Features and Benefits

2-phase stepper motors up to 2A coil current (peak)
STEP/DIR Interface with 2, 4, 8, 16 or 32 microstep pin setting
Smooth Running 256 microsteps by **microPlyer™** interpolation
stealthChop2™ silent motor operation
spreadCycle™ highly dynamic motor control chopper
Low RDSon LS 280mΩ & HS 290mΩ (typ. at 25°C)
Voltage Range 4.75... 36V DC
Automatic Standby current reduction (option)
Internal Sense Resistor option (no sense resistors required)
Passive Braking and Freewheeling
Single Wire UART & OTP for advanced configuration options
Integrated Pulse Generator for standalone motion
Full Protection & Diagnostics
Choice of QFN, TQFP and HTSSOP packages for best fit

Description

The TMC2202, TMC2208, TMC2220, TMC2224 and TMC2225 are ultra-silent motor driver ICs for two phase stepper motors. Their pinning is compatible to a number of legacy drivers. TRINAMICs sophisticated stealthChop2 chopper ensures noiseless operation, maximum efficiency and best motor torque. Its fast current regulation and optional combination with spreadCycle allow for highly dynamic motion. Integrated power-MOSFETs handle motor current up to 1.4A RMS. Protection and diagnostic features support robust and reliable operation. A simple to use UART interface opens up more tuning and control options. Application specific tuning can be stored to OTP memory. Industries' most advanced STEP/DIR stepper motor driver family upgrades designs to noiseless and most precise operation for cost-effective and highly competitive solutions.

Block Diagram



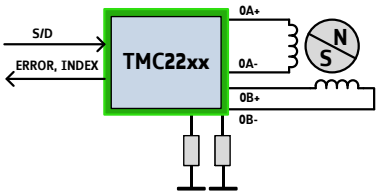
TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

TRINAMIC
MOTION CONTROL

APPLICATION EXAMPLES: SIMPLE SOLUTIONS – HIGHLY EFFECTIVE

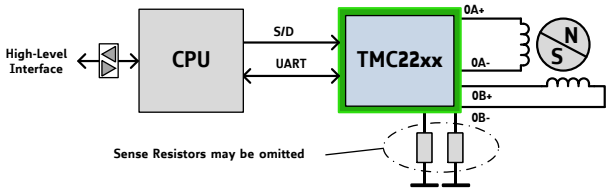
The TMC22xx family scores with power density, integrated power MOSFETs, smooth and quiet operation, and a congenial simplicity. The TMC22xx covers a wide spectrum of applications from battery systems to embedded applications with up to 2A motor current per coil. TRINAMICs unique chopper modes spreadCycle and stealthChop2 optimize drive performance. stealthChop reduces motor noise to the point of silence at low velocities. Standby current reduction keeps costs for power dissipation and cooling down. Extensive support enables rapid design cycles and fast time-to-market with competitive products.

STANDALONE REPLACEMENT FOR LEGACY STEPPER DRIVER



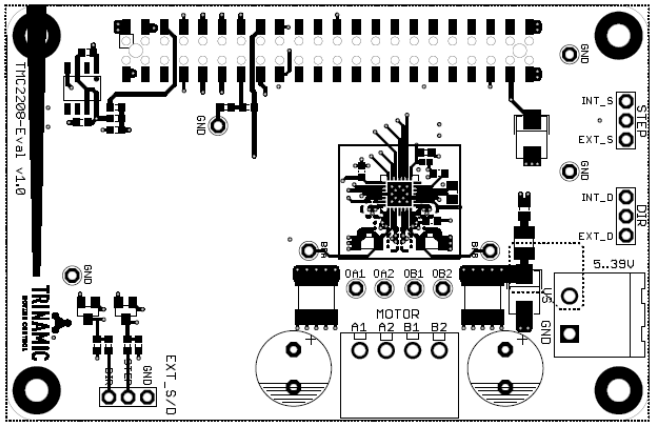
In this example, configuration is hard wired via pins. Software based motion control generates STEP and DIR (direction) signals, INDEX and ERROR signals report back status information.

UART INTERFACE FOR FULL DIAGNOSTICS AND CONTROL



A CPU operates the driver via step and direction signals. It accesses diagnostic information and configures the TMC22xx via the UART interface. The CPU manages motion control and the TMC22xx drives the motor and smoothens and optimizes drive performance.

TMC2208-EVAL EVALUATION BOARD



The TMC22xx-EVAL is part of TRINAMICs universal evaluation board system which provides a convenient handling of the hardware as well as a user-friendly software tool for evaluation. The TMC22xx evaluation board system consists of three parts: STARTRAMPE (base board), TMC2208-BRIDGE (connector board with several test points and stand-alone settings), and TMC22xx-EVAL.

ORDER CODES

Order code	Description	Size [mm²]
TMC2208-LA	stealthChop standalone driver; QFN28 (RoHS compliant)	5 x 5
TMC2224-LA	stealthChop standalone driver; QFN28 (RoHS compliant)	5 x 5
TMC2202-WA	stealthChop driver; wettable edge QFN32 (RoHS compliant)	5 x 5
TMC2220-TA	Option package: TQFP 48 – *please request for availability!*	9 x 9
TMC2225-SA	Option package: HTSSOP28 – *please request for availability!*	9.7 x 6.4
TMC2208-EVAL	Evaluation board for TMC2208 stepper motor driver	85 x 55
TMC2224-EVAL	Evaluation board for TMC2224 stepper motor driver	85 x 55
TMC22xx-Bridge	Connector and jumper board fitting to TMC22xx family	61 x 38
STARTRAMPE	Baseboard for TMC2208-EVAL and further evaluation boards	85 x 55

Table of Contents

1 PRINCIPLES OF OPERATION 4

1.1 KEY CONCEPTS 5
1.2 CONTROL INTERFACES 6
1.3 MOVING AND CONTROLLING THE MOTOR 6
1.4 STEALTHCHOP2 & SPREADCYCLE DRIVER 6
1.5 PRECISE CLOCK GENERATOR AND CLK INPUT 7
1.6 AUTOMATIC STANDSTILL POWER DOWN 7
1.7 INDEX OUTPUT 7

2 PIN ASSIGNMENTS 8

2.1 PACKAGE OUTLINE TMC2208 8
2.2 SIGNAL DESCRIPTIONS TMC2208 8
2.3 PACKAGE OUTLINE TMC2202 9
2.4 SIGNAL DESCRIPTIONS TMC2202 9
2.5 PACKAGE OUTLINE TMC2224 10
2.6 SIGNAL DESCRIPTIONS TMC2224 11
2.7 PACKAGE OUTLINE TMC2225 12
2.8 SIGNAL DESCRIPTIONS TMC2225 12
2.9 PACKAGE OUTLINE TMC2220 13
2.10 SIGNAL DESCRIPTIONS TMC2220 13

3 SAMPLE CIRCUITS 15

3.1 STANDARD APPLICATION CIRCUIT 15
3.2 INTERNAL RDSON SENSING 15
3.3 5V ONLY SUPPLY 16
3.4 CONFIGURATION PINS 17
3.5 HIGH MOTOR CURRENT 17
3.6 DRIVER PROTECTION AND EME CIRCUITRY 18

4 UART SINGLE WIRE INTERFACE 19

4.1 DATAGRAM STRUCTURE 19
4.2 CRC CALCULATION 21
4.3 UART SIGNALS 21
4.4 ADDRESSING MULTIPLE SLAVES 22

5 REGISTER MAP 23

5.1 GENERAL REGISTERS 24
5.2 VELOCITY DEPENDENT CONTROL 29
5.3 SEQUENCER REGISTERS 30
5.4 CHOPPER CONTROL REGISTERS 31

6 STEALTHCHOP™ 37

6.1 AUTOMATIC TUNING 37
6.2 STEALTHCHOP OPTIONS 39
6.3 STEALTHCHOP CURRENT REGULATOR 39
6.4 VELOCITY BASED SCALING 41
6.5 COMBINING STEALTHCHOP AND SPREADCYCLE 43
6.6 FLAGS IN STEALTHCHOP 44
6.7 FREEWHEELING AND PASSIVE BRAKING 45

7 SPREADCYCLE CHOPPER 47

7.1 SPREADCYCLE SETTINGS 48

8 SELECTING SENSE RESISTORS 51

9 MOTOR CURRENT CONTROL 52

9.1 ANALOG CURRENT SCALING VREF 53

10 INTERNAL SENSE RESISTORS 55

11 STEP/DIR INTERFACE 57

11.1 TIMING 57
11.2 CHANGING RESOLUTION 58
11.3 MICROPLYER STEP INTERPOLATOR AND STAND STILL DETECTION 59
11.4 INDEX OUTPUT 60

12 INTERNAL STEP PULSE GENERATOR 61

13 DRIVER DIAGNOSTIC FLAGS 62

13.1 TEMPERATURE MEASUREMENT 62
13.2 SHORT PROTECTION 62
13.3 OPEN LOAD DIAGNOSTICS 63
13.4 DIAGNOSTIC OUTPUT 63

14 QUICK CONFIGURATION GUIDE 64

15 EXTERNAL RESET 67

16 CLOCK OSCILLATOR AND INPUT 67

17 ABSOLUTE MAXIMUM RATINGS 68

18 ELECTRICAL CHARACTERISTICS 68

18.1 OPERATIONAL RANGE 68
18.2 DC AND TIMING CHARACTERISTICS 69
18.3 THERMAL CHARACTERISTICS 73

19 LAYOUT CONSIDERATIONS 74

19.1 EXPOSED DIE PAD 74
19.2 WIRING GND 74
19.3 SUPPLY FILTERING 74
19.4 LAYOUT EXAMPLE TMC2208 75

20 PACKAGE MECHANICAL DATA 76

20.1 DIMENSIONAL DRAWINGS QFN28 76
20.2 DIMENSIONAL DRAWINGS QFN32-WA 78
20.3 PACKAGE CODES 79

21 TABLE OF FIGURES 80

22 REVISION HISTORY 81

23 REFERENCES 81

1 Principles of Operation

The TMC22xx family of stepper drivers is intended as a drop-in upgrade for existing low cost stepper driver applications. Its silent drive technology stealthChop enables non-bugging motion control for home and office applications. A highly efficient power stage enables high current from a tiny package.

The TMC22xx requires just a few control pins on its tiny package. They allow selection of the most important setting: the desired microstep resolution. A choice of 2, 4, 8, 16 or 32 microsteps adapts the driver to the capabilities of the motion controller. Some package options also allow chopper mode selection by pin.

Even at low microstepping rate, the TMC22xx offers a number of unique enhancements over comparable products: TRINAMICs sophisticated stealthChop2 chopper plus the microstep enhancement microPlyer ensure noiseless operation, maximum efficiency and best motor torque. Its fast current regulation and optional combination with spreadCycle allow for highly dynamic motion. Protection and diagnostic features support robust and reliable operation. A simple-to-use 8 bit UART interface opens up more tuning and control options. Application specific tuning can be stored to on-chip OTP memory. Industries' most advanced step & direction stepper motor driver family upgrades designs to noiseless and most precise operation for cost-effective and highly competitive solutions.

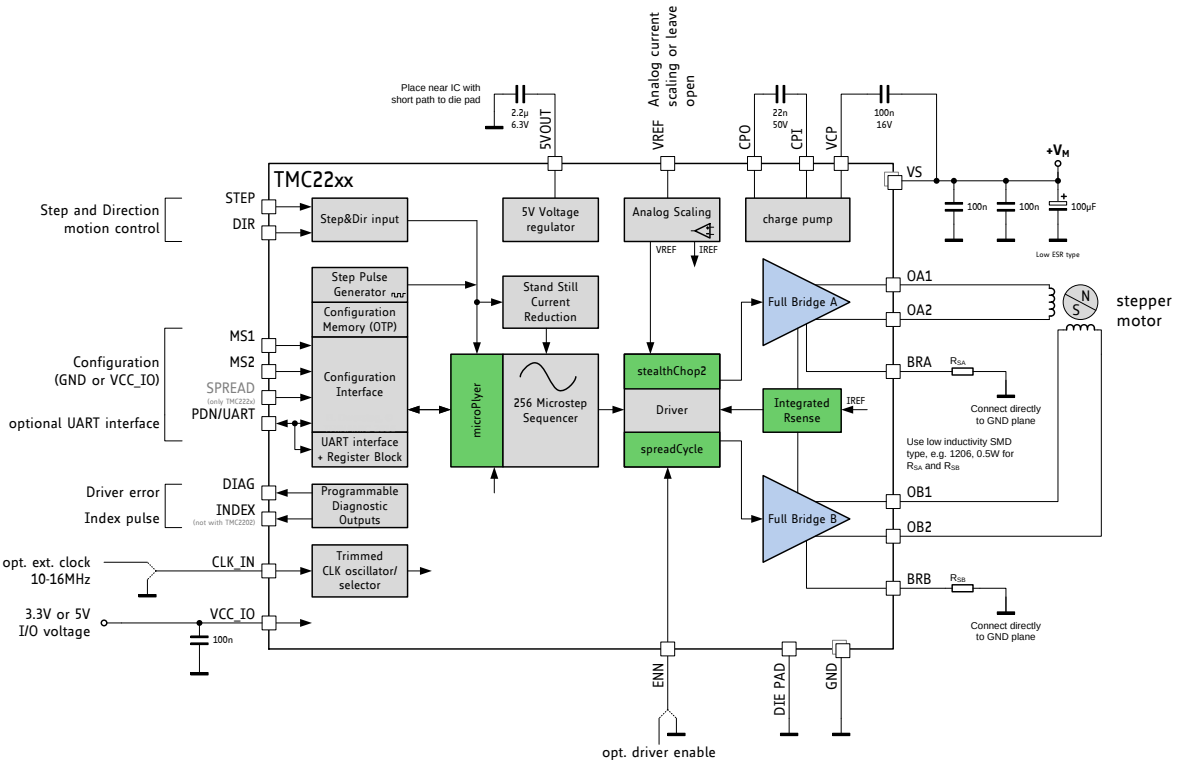


Figure 1.1 TMC22xx basic application block diagram

THREE MODES OF OPERATION:

OPTION 1: Standalone STEP/DIR Driver (Legacy Mode)

A CPU (µC) generates step & direction signals synchronized to additional motors and other components within the system. The TMC22xx operates the motor as commanded by the configuration pins and STEP/DIR signals. Motor run current either is fixed, or set by the CPU using the analog input VREF. The pin PDN_UART selects automatic standstill current reduction. Feedback from the driver to the CPU is granted by the INDEX and DIAG output signals. Enable or disable the motor using the ENN pin.

OPTION 2: Standalone STEP/DIR Driver with OTP pre-configuration

Additional options enabled by pre-programming OTP memory (label UART & OTP):

+ Tuning of the chopper to the application for application tailored performance
+ Cost reduction by switching the driver to internal sense resistor mode
+ Adapting the automatic power down level and timing for best application efficiency

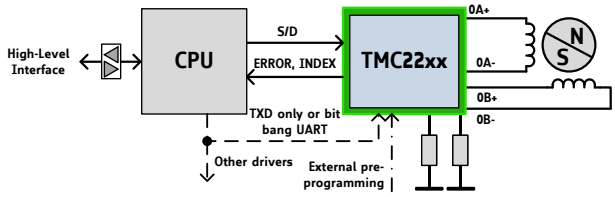


Figure 1.2 Stand-alone driver with pre-configuration

To enable the additional options, either one-time program the driver's OTP memory, or store configuration in the CPU and transfer it to the on-chip registers following each power-up. Operation uses the same signals as Option 1. Programming does not need to be done within the application - it can be executed during testing of the PCB! Alternatively, use bit-banging by CPU firmware to configure the driver. Multiple drivers can be programmed at the same time using a single TXD line.

OPTION 3: STEP/DIR Driver with Full Diagnostics and Control

Similar to Option 2, but pin PDN_UART is connected to the CPU UART interface.

Additional options (label UART):

+ Detailed diagnostics and thermal management
+ Passive braking and freewheeling for flexible, lowest power stop modes
+ More options for microstep resolution setting (fullstep to 256 microstep)
+ Software controlled motor current setting and more chopper options

This mode allows replacing all control lines like ENN, DIAG, INDEX, MS1, MS2, and analog current setting VREF by a single interface line. This way, only three signals are required for full control: STEP, DIR and PDN_UART. Even motion without external STEP pulses is provided by an internal programmable step pulse generator: Just set the desired motor velocity. However, no ramping is provided by the TMC22xx. Access to multiple driver ICs is possible using an analog multiplexer IC.

1.1 Key Concepts

The TMC22xx implements advanced features which are exclusive to TRINAMIC products. These features contribute toward greater precision, greater energy efficiency, higher reliability, smoother motion, and cooler operation in many stepper motor applications.

stealthChop2™ No-noise, high-precision chopper algorithm for inaudible motion and inaudible standstill of the motor. Allows faster motor acceleration and deceleration than stealthChop™ and extends stealthChop to low stand still motor currents.

spreadCycle™ High-precision cycle-by-cycle current control algorithm for highest dynamic movements.

microPlyer™ Microstep interpolator for obtaining full 256 microstep smoothness with lower resolution step inputs starting from fullstep

In addition to these performance enhancements, TRINAMIC motor drivers offer safeguards to detect and protect against shorted outputs, output open-circuit, overtemperature, and undervoltage conditions for enhancing safety and recovery from equipment malfunctions.

1.2 Control Interfaces

The TMC22xx supports both, discrete control lines for basic mode selection and a UART based single wire interface with CRC checking. The UART interface automatically becomes enabled when correct UART data is sent. When using UART, the pin selection may be disabled by control bits.

1.2.1 UART Interface

The single wire interface allows unidirectional operation (for parameter setting only), or bi-directional operation for full control and diagnostics. It can be driven by any standard microcontroller UART or even by bit banging in software. Baud rates from 9600 Baud to 500k Baud or even higher (when using an external clock) may be used. No baud rate configuration is required, as the TMC22xx automatically adapts to the masters' baud rate. The frame format is identical to the intelligent TRINAMIC controller & driver ICs TMC5130 and TMC5072. A CRC checksum allows data transmission over longer distance. For fixed initialization sequences, store the data including CRC into the µC, thus consuming only a few 100 bytes of code for a full initialization. CRC may be ignored during read access, if not desired. This makes CRC use an optional feature! The IC has a fixed address. Multiple drivers can be programmed in parallel by tying together all interface pins, in case no read access is required. An optional addressing can be provided by analog multiplexers, like 74HC4066.

From a software point of view the TMC22xx is a peripheral with a number of control and status registers. Most of them can either be written only or are read only. Some of the registers allow both, read and write access. In case read-modify-write access is desired for a write only register, a shadow register can be realized in master software.

1.3 Moving and Controlling the Motor

1.3.1 STEP/DIR Interface

The motor is controlled by a step and direction input. Active edges on the STEP input can be rising edges or both rising and falling edges as controlled by a special mode bit (DEDGE). Using both edges cuts the toggle rate of the STEP signal in half, which is useful for communication over slow interfaces such as optically isolated interfaces. The state sampled from the DIR input upon an active STEP edge determines whether to step forward or back. Each step can be a fullstep or a microstep, in which there are 2, 4, 8, 16, 32, 64, 128, or 256 microsteps per fullstep. A step impulse with a low state on DIR increases the microstep counter and a high decreases the counter by an amount controlled by the microstep resolution. An internal table translates the counter value into the sine and cosine values which control the motor current for microstepping.

1.3.2 Internal Step Pulse Generator

Some applications do not require a precisely co-ordinate motion – the motor just is required to move for a certain time and at a certain velocity. The TMC22xx comes with an internal pulse generator for these applications: Just provide the velocity via UART interface to move the motor. The velocity sign automatically controls the direction of the motion. However, the pulse generator does not integrate a ramping function. Motion at higher velocities will require ramping up and ramping down the velocity value via software.

STEP/DIR mode and internal pulse generator mode can be mixed in an application!

1.4 stealthChop2 & spreadCycle Driver

stealthChop is a voltage chopper based principle. It especially guarantees that the motor is absolutely quiet in standstill and in slow motion, except for noise generated by ball bearings. Unlike other voltage mode choppers, stealthChop2 does not require any configuration. It automatically learns the best settings during the first motion after power up and further optimizes the settings in subsequent motions. An initial homing sequence is sufficient for learning. Optionally, initial learning parameters can be stored to OTP. stealthChop2 allows high motor dynamics, by reacting at once to a change of motor velocity.

For highest velocity applications, spreadCycle is an option to stealthChop2. It can be enabled via input pin (TMC222x) or via UART and OTP. stealthChop2 and spreadCycle may even be used in a combined configuration for the best of both worlds: stealthChop2 for no-noise stand still, silent and smooth performance, spreadCycle at higher velocity for high dynamics and highest peak velocity at low vibration.

spreadCycle is an advanced cycle-by-cycle chopper mode. It offers smooth operation and good resonance dampening over a wide range of speed and load. The spreadCycle chopper scheme automatically integrates and tunes fast decay cycles to guarantee smooth zero crossing performance.

Benefits of using stealthChop2:

- Significantly improved microstepping with low cost motors
- Motor runs smooth and quiet
- Absolutely no standby noise
- Reduced mechanical resonance yields improved torque

1.5 Precise clock generator and CLK input

The TMC22xx provides a factory trimmed internal clock generator for precise chopper frequency and performance. However, an optional external clock input is available for cases, where quartz precision is desired, or where a lower or higher frequency is required. For safety, the clock input features timeout detection, and switches back to internal clock upon fail of the external source.

1.6 Automatic Standstill Power Down

An automatic current reduction drastically reduces application power dissipation and cooling requirements. Per default, the stand still current reduction is enabled by pulling PDN_UART input to GND. It reduces standstill power dissipation to less than 33% by going to slightly more than half of the run current.
Modify stand still current, delay time and decay via UART, or pre-programmed via internal OTP. Automatic freewheeling and passive motor braking are provided as an option for stand still. Passive braking reduces motor standstill power consumption to zero, while still providing effective dampening and braking!

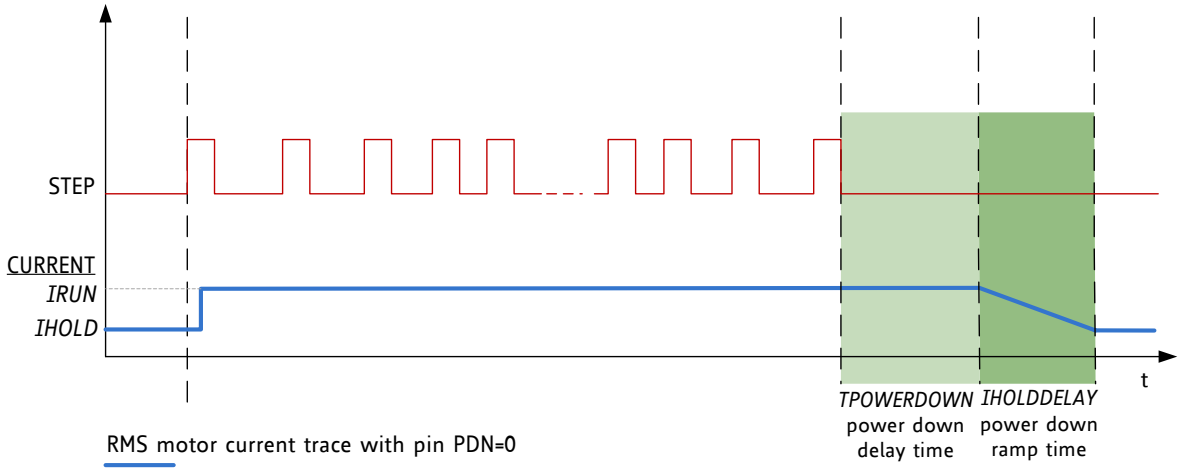


Figure 1.3 Automatic Motor Current Power Down

1.7 Index Output

The index output gives one pulse per electrical rotation, i.e. one pulse per each four fullsteps. It shows the internal sequencer microstep 0 position (*MSTEP* near 0). This is the power on position. In combination with a mechanical home switch, a more precise homing is enabled.

2 Pin Assignments

The TMC22xx family comes in a number of package variants in order to fit different footprints. Please check for availability.

2.1 Package Outline TMC2208

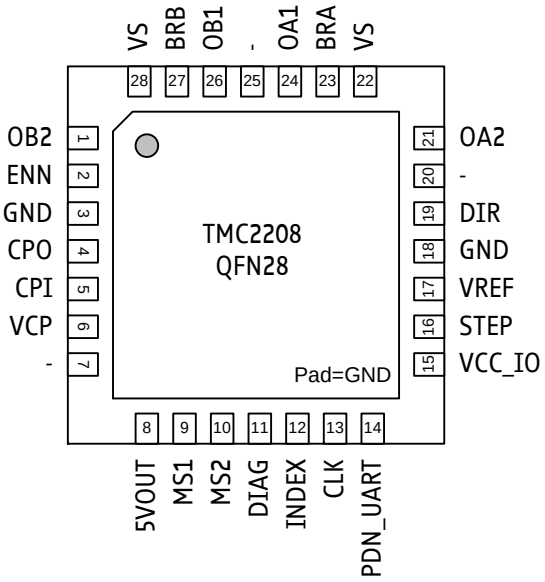


Figure 2.1 TMC2208 Pinning Top View – type: QFN28, 5x5mm², 0.5mm pitch

2.2 Signal Descriptions TMC2208

Pin	Number	Type	Function
OB2	1		Motor coil B output 2
ENN	2	DI	Enable not input. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level.
GND	3, 18		GND. Connect to GND plane near pin.
CPO	4		Charge pump capacitor output.
CPI	5		Charge pump capacitor input. Tie to CPO using 22nF 50V capacitor.
VCP	6		Charge pump voltage. Tie to VS using 100nF capacitor.
N.C.	7, 20, 25		Unused pin, leave open or connect to GND for compatibility to future versions.
5VOUT	8		Output of internal 5V regulator. Attach 2.2µF to 4.7µF ceramic capacitor to GND near to pin for best performance. Provide the shortest possible loop to the GND pad.
MS1	9	DI (pd)	Microstep resolution configuration (internal pull down resistors) MS2, MS1: 00: 1/8, 01: 1/2, 10: 1/4 11: 1/16
MS2	10	DI (pd)	
DIAG	11	DO	Diagnostic output. Hi level upon driver error. Reset by ENN=high.
INDEX	12	DO	Configurable index output. Provides index pulse.
CLK	13	DI	CLK input. Tie to GND using short wire for internal clock or supply external clock.
PDN_UART	14	DIO	Power down not control input (low = automatic standstill current reduction). Optional UART Input/Output. Power down function can be disabled in UART mode.
VCC_IO	15		3.3V to 5V IO supply voltage for all digital pins.

Pin	Number	Type	Function
STEP	16	DI	STEP input
VREF	17	AI	Analog reference voltage for current scaling or reference current for use of internal sense resistors (optional mode)
DIR	19	DI (pd)	DIR input (internal pull down resistor)
VS	22, 28		Motor supply voltage. Provide filtering capacity near pin with shortest possible loop to GND pad.
OA2	21		Motor coil A output 2
BRA	23		Sense resistor connection for coil A. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
OA1	24		Motor coil A output 1
OB1	26		Motor coil B output 1
BRB	27		Sense resistor connection for coil B. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
Exposed die pad	-		Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane. Serves as GND pin for power drivers and analogue circuitry.

2.3 Package Outline TMC2202

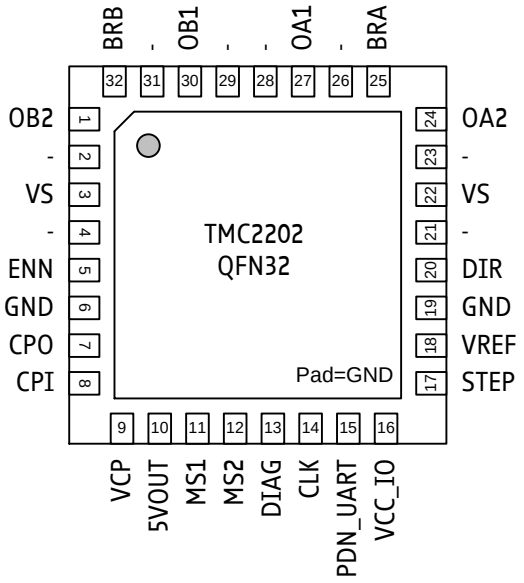


Figure 2.2 TMC2202 Pinning Top View – type: QFN32, 5x5mm², 0.5mm pitch

2.4 Signal Descriptions TMC2202

Pin	Number	Type	Function
OB2	1		Motor coil B output 2
N.C.	2, 4, 21, 23, 26, 28, 29, 31		Unused pin, leave open to provide for higher creeping voltage distances.
VS	3, 22		Motor supply voltage. Provide filtering capacity near pin with shortest possible loop to GND pad.
ENN	5	DI	Enable not input. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level.

Pin	Number	Type	Function
GND	6, 19		GND. Connect to GND plane near pin.
CPO	7		Charge pump capacitor output.
CPI	8		Charge pump capacitor input. Tie to CPO using 22nF 50V capacitor.
VCP	9		Charge pump voltage. Tie to VS using 100nF capacitor.
5VOUT	10		Output of internal 5V regulator. Attach 2.2µF to 4.7µF ceramic capacitor to GND near to pin for best performance. Provide the shortest possible loop to the GND pad.
MS1	11	DI (pd)	Microstep resolution configuration (internal pull down resistors)
MS2	12	DI (pd)	MS2, MS1: 00: 1/8, 01: 1/2, 10: 1/4 11: 1/16
DIAG	13	DO	Diagnostic output. Hi level upon driver error. Reset by ENN=high.
CLK	14	DI	CLK input. Tie to GND using short wire for internal clock or supply external clock.
PDN_UART	15	DIO	Power down not control input (low = automatic standstill current reduction). Optional UART Input/Output. Power down function can be disabled in UART mode.
VCC_IO	16		3.3V to 5V IO supply voltage for all digital pins.
STEP	17	DI	STEP input
VREF	18	AI	Analog reference voltage for current scaling or reference current for use of internal sense resistors (optional mode)
DIR	20	DI (pd)	DIR input (internal pull down resistor)
OA2	24		Motor coil A output 2
BRA	25		Sense resistor connection for coil A. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
OA1	27		Motor coil A output 1
OB1	30		Motor coil B output 1
BRB	32		Sense resistor connection for coil B. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
Exposed die pad	-		Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane. Serves as GND pin for power drivers and analogue circuitry.

2.5 Package Outline TMC2224

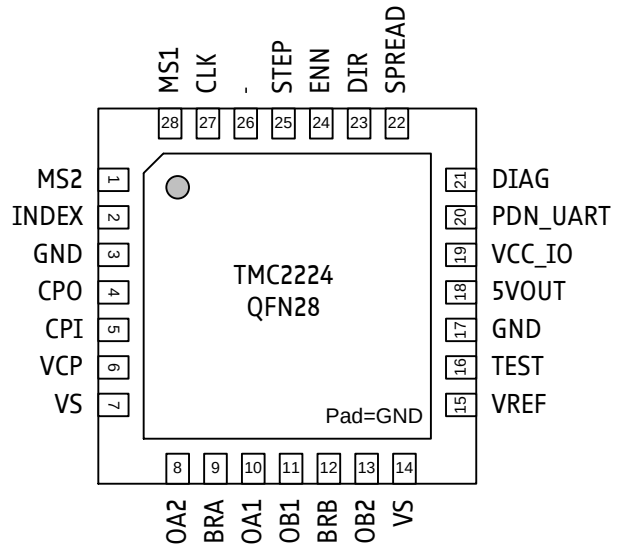


Figure 2.3 TMC2224 Pinning Top View – type: QFN28, 5x5mm², 0.5mm pitch

2.6 Signal Descriptions TMC2224

Pin	Number	Type	Function
MS1	28	DI (pd)	Microstep resolution configuration (internal pull down resistors) MS2, MS1: 00: 1/4, 01: 1/8, 10: 1/16, 11: 1/32
MS2	1	DI (pd)	
INDEX	2	DO	Configurable index output. Provides index pulse.
GND	3, 17		GND. Connect to GND plane near pin.
CPO	4		Charge pump capacitor output.
CPI	5		Charge pump capacitor input. Tie to CPO using 22nF 50V capacitor.
VCP	6		Charge pump voltage. Tie to VS using 100nF capacitor.
VS	7, 14		Motor supply voltage. Provide filtering capacity near pin with shortest possible loop to GND pad.
OA2	8		Motor coil A output 2
BRA	9		Sense resistor connection for coil A. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
OA1	10		Motor coil A output 1
OB1	11		Motor coil B output 1
BRB	12		Sense resistor connection for coil B. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
OB2	13		Motor coil B output 2
VREF	15	AI	Analog reference voltage for current scaling or reference current for use of internal sense resistors (optional mode)
TEST	16		Connect to GND. May alternatively be left open or connected to VREF.
5VOUT	18		Output of internal 5V regulator. Attach 2.2µF to 4.7µF ceramic capacitor to GND near to pin for best performance. Provide the shortest possible loop to the GND pad.
VCC_IO	19		3.3V to 5V IO supply voltage for all digital pins.
PDN_UART	20	DIO (pd)	Power down not control input (low = automatic standstill current reduction). (internal pull down resistor) Optional UART Input/Output. Power down function can be disabled in UART mode.
DIAG	21	DO	Diagnostic output. Hi level upon driver error. Reset by ENN=high.
SPREAD	22	DI (pd)	Chopper mode selection: Low=stealthChop, High=spreadCycle
DIR	23	DI (pd)	DIR input (internal pull down resistor)
ENN	24	DI	Enable not input. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level.
STEP	25	DI (pd)	STEP input (internal pull down resistor)
N.C.	26		Unused pin, leave open or connect to GND for compatibility to future versions.
CLK	27	DI	CLK input. Tie to GND using short wire for internal clock or supply external clock.
Exposed die pad	-		Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane. Serves as GND pin for power drivers and analogue circuitry.

2.7 Package Outline TMC2225

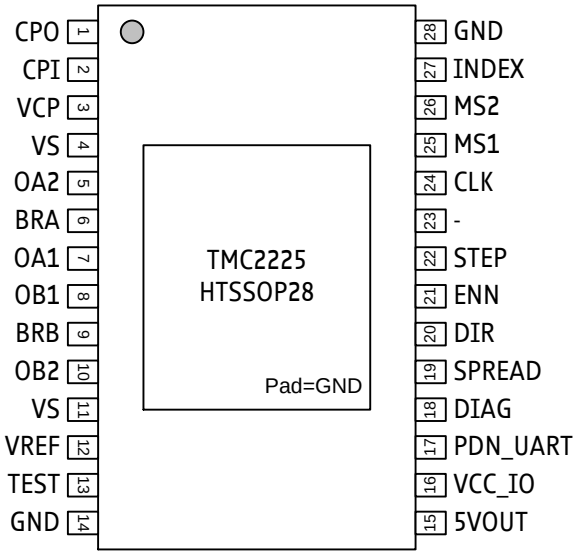


Figure 2.4 TMC2225 Pinning Top View – type: HTSSOP28, 9.7x6.4mm² over pins, 0.65mm pitch

2.8 Signal Descriptions TMC2225

Pin	Number	Type	Function
CPO	1		Charge pump capacitor output.
CPI	2		Charge pump capacitor input. Tie to CPO using 22nF 50V capacitor.
VCP	3		Charge pump voltage. Tie to VS using 100nF capacitor.
VS	4, 11		Motor supply voltage. Provide filtering capacity near pin with shortest possible loop to GND pad.
OA2	5		Motor coil A output 2
BRA	6		Sense resistor connection for coil A. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
OA1	7		Motor coil A output 1
OB1	8		Motor coil B output 1
BRB	9		Sense resistor connection for coil B. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
OB2	10		Motor coil B output 2
VREF	12	AI	Analog reference voltage for current scaling or reference current for use of internal sense resistors (optional mode)
TEST	13		Connect to GND. May alternatively be left open or connected to VREF.
GND	14, 28		GND. Connect to GND plane near pin.
5VOUT	15		Output of internal 5V regulator. Attach 2.2µF to 4.7µF ceramic capacitor to GND near to pin for best performance. Provide the shortest possible loop to the GND pad.
VCC_IO	16		3.3V to 5V IO supply voltage for all digital pins.
PDN_UART	17	DIO (pd)	Power down not control input (low = automatic standstill current reduction). (internal pull down resistor) Optional UART Input/Output. Power down function can be disabled in UART mode.
DIAG	18	DO	Diagnostic output. Hi level upon driver error. Reset by ENN=high.
SPREAD	19	DI (pd)	Chopper mode selection: Low=stealthChop, High=spreadCycle
DIR	20	DI (pd)	DIR input (internal pull down resistor)
ENN	21	DI	Enable not input. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level.

Pin	Number	Type	Function
STEP	22	DI (pd)	STEP input (internal pull down resistor)
N.C.	23		Unused pin, leave open or connect to GND for compatibility to future versions.
CLK	24	DI	CLK input. Tie to GND using short wire for internal clock or supply external clock.
MS1	25	DI (pd)	Microstep resolution configuration (internal pull down resistors)
MS2	26	DI (pd)	MS2, MS1: 00: 1/4, 01: 1/8, 10: 1/16, 11: 1/32
INDEX	27	DO	Configurable index output. Provides index pulse.
Exposed die pad	-		Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane. Serves as GND pin for power drivers and analogue circuitry.

2.9 Package Outline TMC2220

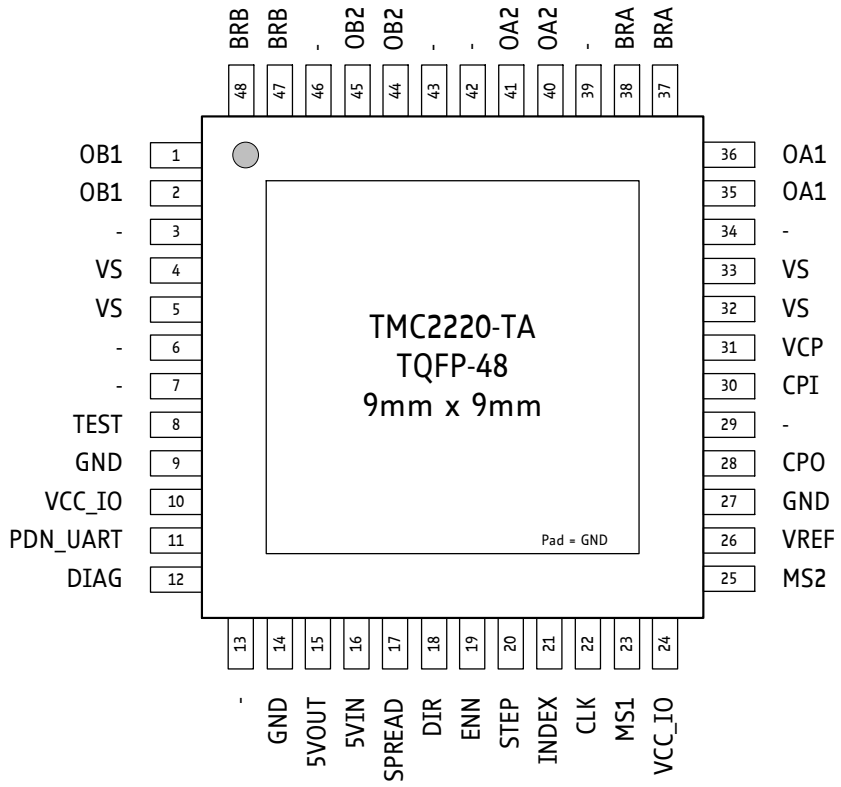


Figure 2.5 TMC2220 Pinning Top View – type: TQFP-EP 48, 9x9mm² over pins, 0.5mm pitch

2.10 Signal Descriptions TMC2220

Pin	Number	Type	Function
OB1	1, 2		Motor coil B output 1
VS	4, 5, 32, 33		Motor supply voltage. Provide filtering capacity near pin with shortest possible loop to GND pad.
TEST	8		Connect to GND. May alternatively be left open.
GND	9, 14, 27		GND. Connect to GND plane near pin.
VCC_IO	10, 24		3.3V to 5V IO supply voltage for all digital pins.

Pin	Number	Type	Function
PDN_UART	11	DIO (pd)	Power down not control input (low = automatic standstill current reduction). (internal pull down resistor) Optional UART Input/Output. Power down function can be disabled in UART mode.
5VOUT	15		Output of internal 5V regulator. Attach 2.2µF to 4.7µF ceramic capacitor to GND near to pin for best performance. Provide the shortest possible loop to the GND pad.
5VIN	16		Input of 5V supply. Directly connect to 5VOUT terminal.
DIAG	12	DO	Diagnostic output. Hi level upon driver error. Reset by ENN=high.
SPREAD	17	DI (pd)	Chopper mode selection: Low=stealthChop, High=spreadCycle
DIR	18	DI (pd)	DIR input (internal pull down resistor)
ENN	19	DI	Enable not input. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level.
STEP	20	DI (pd)	STEP input (internal pull down resistor)
INDEX	21	DO	Configurable index output. Provides index pulse.
CLK	22	DI	CLK input. Tie to GND using short wire for internal clock or supply external clock.
MS1	23	DI (pd)	Microstep resolution configuration (internal pull down resistors)
MS2	25	DI (pd)	MS2, MS1: 00: 1/4, 01: 1/8, 10: 1/16, 11: 1/32
VREF	26	AI	Analog reference voltage for current scaling or reference current for use of internal sense resistors (optional mode)
CPO	28		Charge pump capacitor output.
CPI	30		Charge pump capacitor input. Tie to CPO using 22nF 50V capacitor.
VCP	31		Charge pump voltage. Tie to VS using 100nF capacitor.
OA1	35, 36		Motor coil A output 1
BRA	37, 38		Sense resistor connection for coil A. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
OA2	40, 41		Motor coil A output 2
OB2	44, 45		Motor coil B output 2
BRB	47, 48		Sense resistor connection for coil B. Place sense resistor to GND near pin. Tie to GND when using internal sense resistor.
N.C.			Unused pin, leave open or connect to GND for compatibility to future versions.
Exposed die pad	-		Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane. Serves as GND pin for power drivers and analogue circuitry.

3 Sample Circuits

The sample circuits show the connection of external components in different operation and supply modes. The connection of the bus interface and further digital signals is left out for clarity.

3.1 Standard Application Circuit

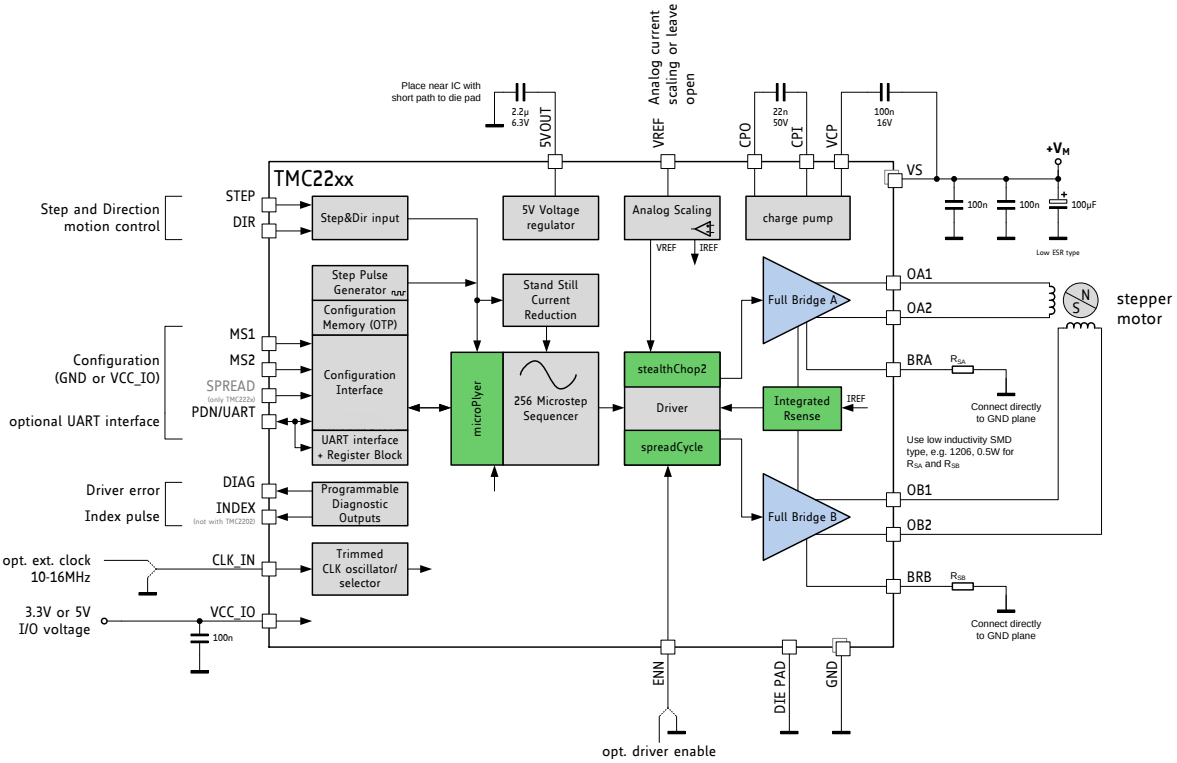


Figure 3.1 Standard application circuit

The standard application circuit uses a minimum set of additional components. Two sense resistors set the motor coil current. See chapter 8 to choose the right sense resistors. Use low ESR capacitors for filtering the power supply. The capacitors need to cope with the current ripple cause by chopper operation. A minimum capacity of 100µF near the driver is recommended for best performance. Current ripple in the supply capacitors also depends on the power supply internal resistance and cable length. VCC_IO can be supplied from 5VOUT, or from an external source, e.g. a 3.3V regulator.

Basic layout hints
Place sense resistors and all filter capacitors as close as possible to the related IC pins. Use a solid common GND for all GND connections, also for sense resistor GND. Connect 5VOUT filtering capacitor directly to 5VOUT and the die pad. See layout hints for more details. Low ESR electrolytic capacitors are recommended for VS filtering.

3.2 Internal RDSon Sensing

For cost critical or space limited applications, sense resistors can be omitted. For internal current sensing, a reference current set by a tiny external resistor programs the output current. For calculation of the reference resistor, refer chapter 9.1.

Attention
Be sure to switch the IC to RDSon mode, before enabling drivers: Set *otp_internalRsense* = 1.

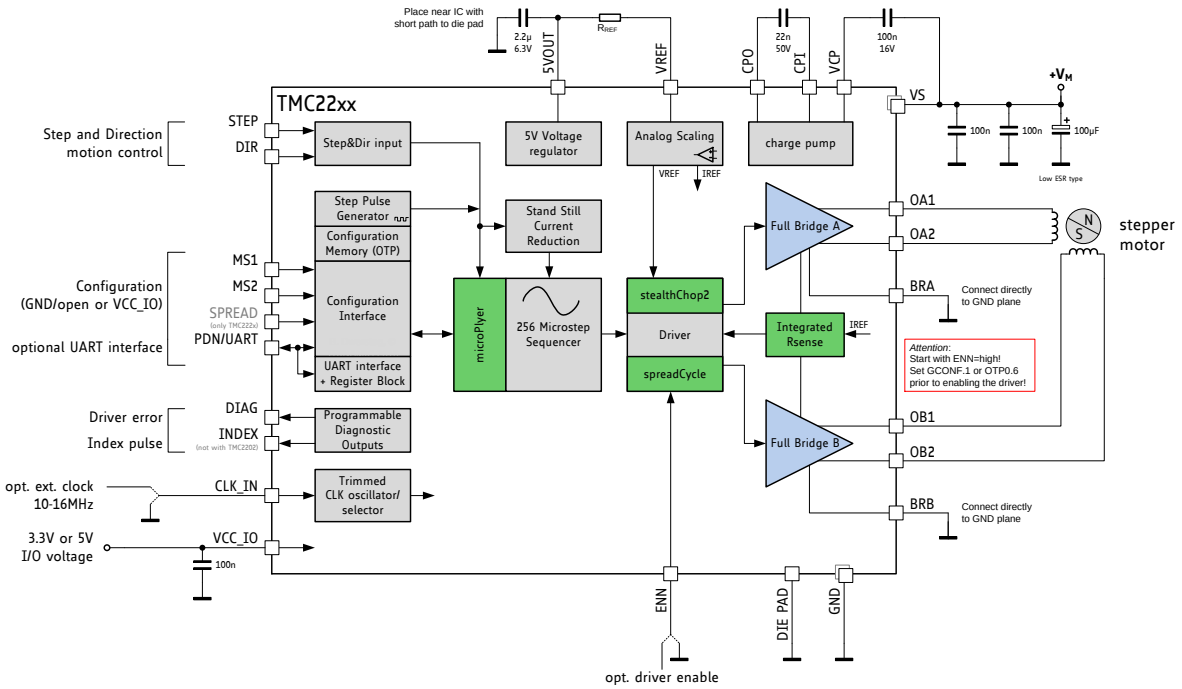


Figure 3.2 Application circuit using RDSon based sensing

3.3 5V Only Supply

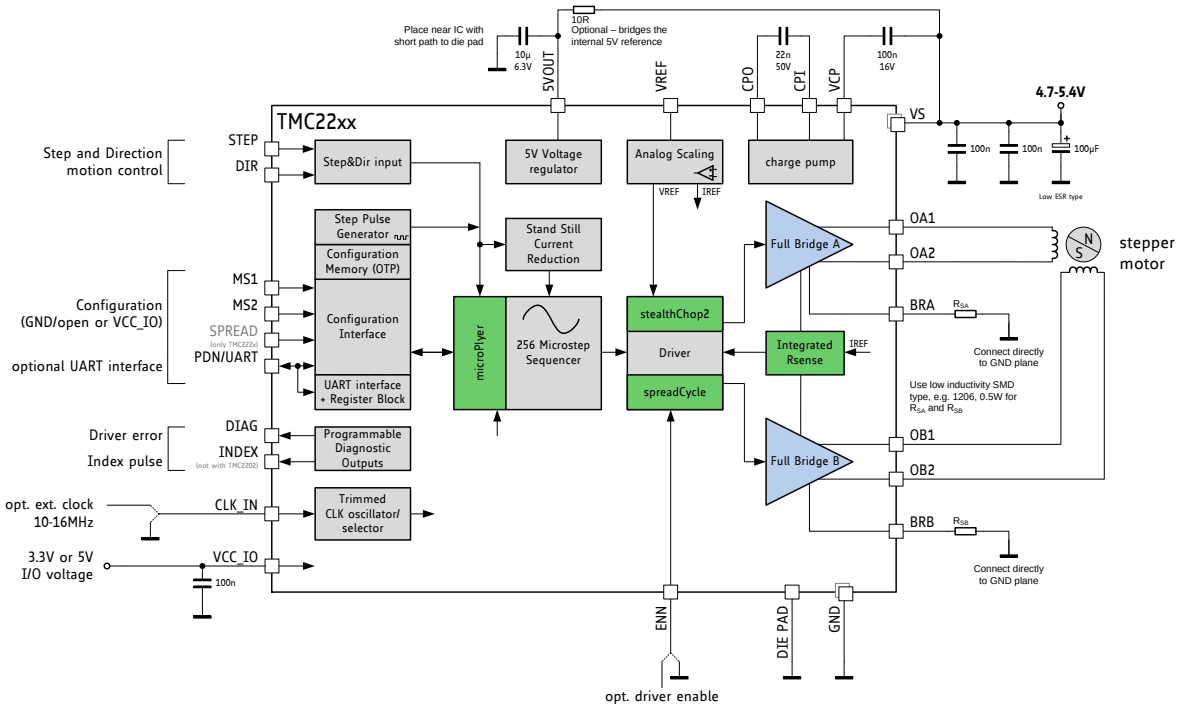


Figure 3.3 5V only operation

While the standard application circuit is limited to roughly 5.2V lower supply voltage, a 5V only application lets the IC run from a 5V +/-5% supply. In this application, linear regulator drop must be

minimized. Therefore, the internal 5V regulator is filtered with a higher capacitance. An optional resistor bridges the internal 5V regulator by connecting 5VOUT to the external power supply. This RC filter keeps chopper ripple away from 5VOUT. With this resistor, the external supply is the reference for the absolute motor current and must not exceed 5.5V.

3.4 Configuration Pins

The TMC22xx family members provide three or four configuration pins depending on the package option. These pins allow quick configuration for standalone operation. Several additional options can be set by OTP programming. In UART mode, the configuration pins can be disabled in order to set a different configuration via registers.

PDN_UART: Configuration of Standstill Power Down	
PDN_UART	**Current Setting**
GND	Enable automatic power down in standstill periods
VCC_IO	Disable
UART interface	When using the UART interface, the configuration pin should be disabled via *GCONF.pdn_disable* = 1. Program *IHOLD* as desired for standstill periods.

Options for TMC220x devices, only:

MS1/MS2: Configuration of Microstep Resolution for Step Input (TMC220x)		
MS2	**MS1**	**Microstep Setting**
GND	GND	8 microsteps
GND	VCC_IO	2 microsteps (half step)
VCC_IO	GND	4 microsteps (quarter step)
VCC_IO	VCC_IO	16 microsteps

Options for TMC222x devices, only:

SPREAD (only with TMC222x): Selection of Chopper Mode	
SPREAD	**Chopper Setting**
GND or Pin open / not available	stealthChop is selected. Automatic switching to spreadCycle in dependence of the step frequency can be programmed via OTP.
VCC_IO	spreadCycle operation.

MS1/MS2: Configuration of Microstep Resolution for Step Input (TMC222x)		
MS2	**MS1**	**Microstep Setting**
GND	GND	4 microsteps (quarter step)
GND	VCC_IO	8 microsteps
VCC_IO	GND	16 microsteps
VCC_IO	VCC_IO	32 microsteps

3.5 High Motor Current

When operating at a high motor current, the driver power dissipation due to MOSFET switch on-resistance significantly heats up the driver. This power dissipation will significantly heat up the PCB cooling infrastructure, if operated at an increased duty cycle. This in turn leads to a further increase of driver temperature. An increase of temperature by about 100°C increases MOSFET resistance by roughly 50%. This is a typical behavior of MOSFET switches. Therefore, under high duty cycle, high load conditions, thermal characteristics have to be carefully taken into account, especially when increased environment temperatures are to be supported. Refer the thermal characteristics and the layout hints for more information. As a thumb rule, thermal properties of the PCB design become

critical for the tiny QFN 5mm x 5mm package at or above 1A RMS motor current for increased periods of time. Keep in mind that resistive power dissipation raises with the square of the motor current. On the other hand, this means that a small reduction of motor current significantly saves heat dissipation and energy.

Pay special attention to good thermal properties of your PCB layout, when going for 1A RMS current or more.

An effect which might be perceived at medium motor velocities and motor sine wave peak currents above roughly 1.4A peak is a slight sine distortion of the current wave when using spreadCycle. It results from an increasing negative impact of parasitic internal diode conduction, which in turn negatively influences the duration of the fast decay cycle of the spreadCycle chopper. This is, because the current measurement does not see the full coil current during this phase of the sine wave, because an increasing part of the current flows directly from the power MOSFETs' drain to GND and does not flow through the sense resistor. This effect with most motors does not negatively influence the smoothness of operation, as it does not impact the critical current zero transition. The effect does not occur with stealthChop.

3.6 Driver Protection and EME Circuitry

Some applications have to cope with ESD events caused by motor operation or external influence. Despite ESD circuitry within the driver chips, ESD events occurring during operation can cause a reset or even a destruction of the motor driver, depending on their energy. Especially plastic housings and belt drive systems tend to cause ESD events of several kV. It is best practice to avoid ESD events by attaching all conductive parts, especially the motors themselves to PCB ground, or to apply electrically conductive plastic parts. In addition, the driver can be protected up to a certain degree against ESD events or live plugging / pulling the motor, which also causes high voltages and high currents into the motor connector terminals. A simple scheme uses capacitors at the driver outputs to reduce the dV/dt caused by ESD events. Larger capacitors will bring more benefit concerning ESD suppression, but cause additional current flow in each chopper cycle, and thus increase driver power dissipation, especially at high supply voltages. The values shown are example values – they may be varied between 100pF and 1nF. The capacitors also dampen high frequency noise injected from digital parts of the application PCB circuitry and thus reduce electromagnetic emission. A more elaborate scheme uses LC filters to de-couple the driver outputs from the motor connector. Varistors in between of the coil terminals eliminate coil overvoltage caused by live plugging. Optionally protect all outputs by a varistor to GND against ESD voltage.

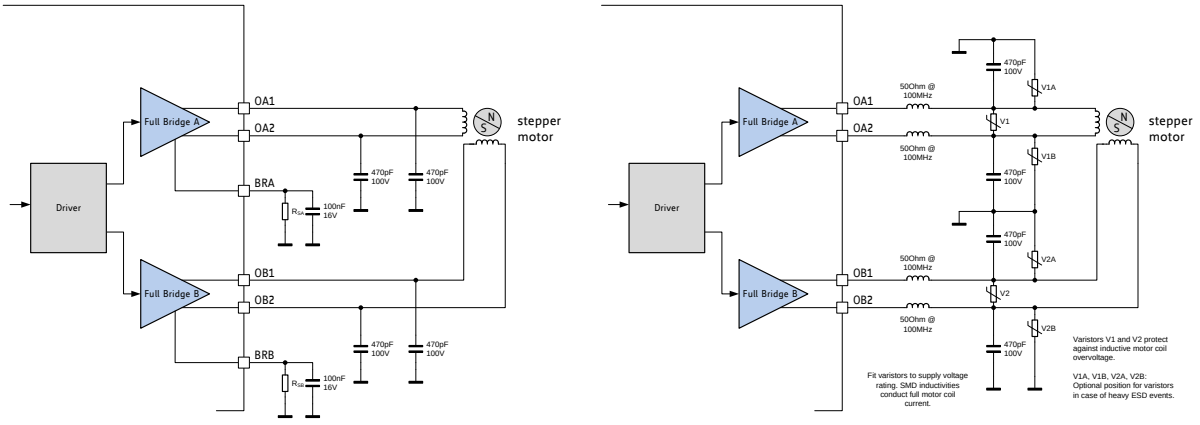


Figure 3.4 Simple ESD enhancement and more elaborate motor output protection

4 UART Single Wire Interface

The UART single wire interface allows control of the TMC22xx with any microcontroller UART. It shares transmit and receive line like an RS485 based interface. Data transmission is secured using a cyclic redundancy check, so that increased interface distances (e.g. over cables between two PCBs) can be bridged without danger of wrong or missed commands even in the event of electro-magnetic disturbance. The automatic baud rate detection makes this interface easy to use.

4.1 Datagram Structure

4.1.1 Write Access

UART WRITE ACCESS DATAGRAM STRUCTURE																			
each byte is LSB...MSB, highest byte transmitted first																			
0 ... 63																			
sync + reserved								**8 bit slave address**			**RW + 7 bit register addr.**			**32 bit data**			**CRC**		
0...7								8...15			16...23			24...55			56...63		
1	0	1	0	Reserved (don't cares but included in CRC)				*SLAVEADDR*=0			register address		1	data bytes 3, 2, 1, 0 (high to low byte)			CRC		
0	1	2	3	4	5	6	7	8	...	15	16	...	23	24	...	55	56	...	63

A sync nibble precedes each transmission to and from the TMC22xx and is embedded into the first transmitted byte, followed by an addressing byte (0 for TMC22xx). Each transmission allows a synchronization of the internal baud rate divider to the master clock. The actual baud rate is adapted and variations of the internal clock frequency are compensated. Thus, the baud rate can be freely chosen within the valid range. Each transmitted byte starts with a start bit (logic 0, low level on SWIOP) and ends with a stop bit (logic 1, high level on SWIOP). The bit time is calculated by measuring the time from the beginning of start bit (1 to 0 transition) to the end of the sync frame (1 to 0 transition from bit 2 to bit 3). All data is transmitted bytewise. The 32 bit data words are transmitted with the highest byte first.

A minimum baud rate of 9000 baud is permissible, assuming 20 MHz clock (worst case for low baud rate). Maximum baud rate is f_{CLK}/16 due to the required stability of the baud clock.

The slave address *SLAVEADDR* is always 0 for the TMC22xx.

The communication becomes reset if a pause time of longer than 63 bit times between the start bits of two successive bytes occurs. This timing is based on the last correctly received datagram. In this case, the transmission needs to be restarted after a failure recovery time of minimum 12 bit times of bus idle time. This scheme allows the master to reset communication in case of transmission errors. Any pulse on an idle data line below 16 clock cycles will be treated as a glitch and leads to a timeout of 12 bit times, for which the data line must be idle. Other errors like wrong CRC are also treated the same way. This allows a safe re-synchronization of the transmission after any error conditions. Remark, that due to this mechanism an abrupt reduction of the baud rate to less than 15 percent of the previous value is not possible.

Each accepted write datagram becomes acknowledged by the receiver by incrementing an internal cyclic datagram counter (8 bit). Reading out the datagram counter allows the master to check the success of an initialization sequence or single write accesses. Read accesses do not modify the counter.

The UART line must be logic high during idle state. Therefore, the power down function cannot be assigned by the pin PDN_UART in between of transmissions. In an application using the UART interface, set the desired power down function by register access and set *pdn_disable* in GCONF to disable the pin function.

4.1.2 Read Access

UART READ ACCESS REQUEST DATAGRAM STRUCTURE																					
each byte is LSB...MSB, highest byte transmitted first																					
sync + reserved								**8 bit slave address**				**RW + 7 bit register address**				**CRC**					
0...7								8...15				16...23				24...31					
1	0	1	0	Reserved (don't cares but included in CRC)				*SLAVEADDR*=0				register address			0	CRC					
0	1	2	3	4	5	6	7	8	...		15	16	...		23	24	...		31		

The read access request datagram structure is identical to the write access datagram structure, but uses a lower number of user bits. Its function is the addressing of the slave and the transmission of the desired register address for the read access. The TMC22xx responds with the same baud rate as the master uses for the read request.

In order to ensure a clean bus transition from the master to the slave, the TMC22xx does not immediately send the reply to a read access, but it uses a programmable delay time after which the first reply byte becomes sent following a read request. This delay time can be set in multiples of eight bit times using *SENDDELAY* time setting (default=8 bit times) according to the needs of the master.

UART READ ACCESS REPLY DATAGRAM STRUCTURE																					
each byte is LSB...MSB, highest byte transmitted first																					
0 63																					
sync + reserved								**8 bit master address**				**RW + 7 bit register addr.**				**32 bit data**				**CRC**	
0...7								8...15				16...23				24...55				56...63	
1	0	1	0	reserved (0)				0xFF				register address			0	data bytes 3, 2, 1, 0 (high to low byte)				CRC	
0	1	2	3	4	5	6	7	8	...		15	16	...		23	24	...		55	56	...	63

The read response is sent to the master using address code %11111111. The transmitter becomes switched inactive four bit times after the last bit is sent.

Address %11111111 is reserved for read access replies going to the master.

4.2 CRC Calculation

An 8 bit CRC polynomial is used for checking both read and write access. It allows detection of up to eight single bit errors. The CRC8-ATM polynomial with an initial value of zero is applied LSB to MSB, including the sync- and addressing byte. The sync nibble is assumed to always be correct. The TMC22xx responds only to correctly transmitted datagrams containing its own slave address. It increases its datagram counter for each correctly received write access datagram.

$$CRC = x^8 + x^2 + x^1 + x^0$$

SERIAL CALCULATION EXAMPLE

```
CRC = (CRC << 1) OR (CRC.7 XOR CRC.1 XOR CRC.0 XOR [new incoming bit])
```

C-CODE EXAMPLE FOR CRC CALCULATION

```
void swuart_calcCRC(UCHAR* datagram, UCHAR datagramLength)
{
  int i,j;
  UCHAR* crc = datagram + (datagramLength-1); // CRC located in last byte of message
  UCHAR currentByte;

  *crc = 0;
  for (i=0; i<(datagramLength-1); i++) {      // Execute for all bytes of a message
    currentByte = datagram[i];                // Retrieve a byte to be sent from Array
    for (j=0; j<8; j++) {
      if ((*crc >> 7) ^ (currentByte&0x01))   // update CRC based result of XOR operation
      {
        *crc = (*crc << 1) ^ 0x07;
      }
      else
      {
        *crc = (*crc << 1);
      }
      currentByte = currentByte >> 1;
    } // for CRC bit
  } // for message byte
}
```

4.3 UART Signals

The UART interface on the TMC22xx uses a single bi-direction pin:

UART INTERFACE SIGNAL	
PDN_UART	Non-inverted data input and output. I/O with Schmitt Trigger and VCC_IO level.

The IC checks PDN_UART for correctly received datagrams with its own address continuously. It adapts to the baud rate based on the sync nibble, as described before. In case of a read access, it switches on its output drivers and sends its response using the same baud rate. The output becomes switched off four bit times after transfer of the last stop bit.

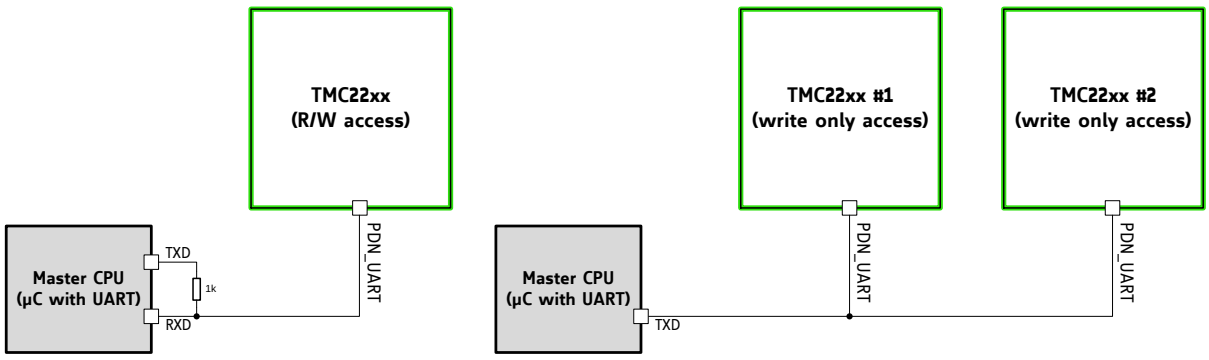


Figure 4.1 Attaching the TMC22xx to a microcontroller UART

4.4 Addressing Multiple Slaves

WRITE ONLY ACCESS

If read access is not used, and all slaves are to be programmed with the same initialization values, no addressing is required. All slaves can be programmed in parallel like a single device (Figure 4.1.).

ADDRESSING MULTIPLE SLAVES

As the TMC22xx uses a fixed UART address, in principle only one IC can be accessed per UART interface channel. Adding analog switches allows separated access to individual ICs. This scheme is similar to an SPI bus with individual slave select lines (Figure 4.2).

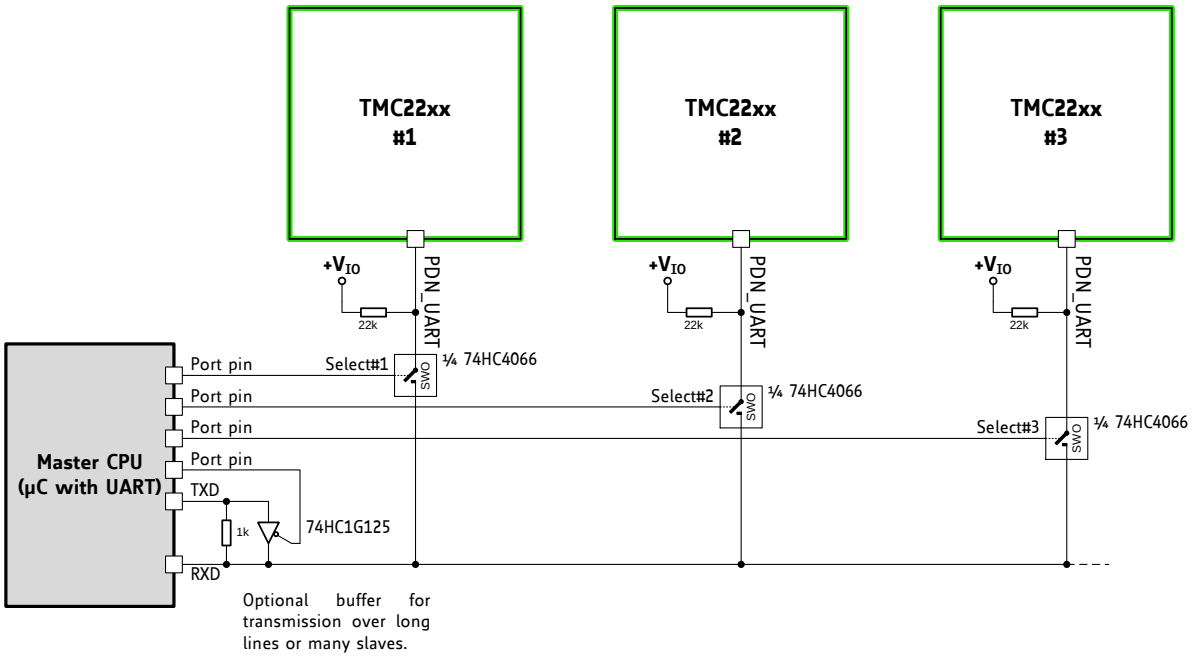


Figure 4.2 Addressing multiple TMC22xx via single wire interface using analog switches

PROCEED AS FOLLOWS TO CONTROL MULTIPLE SLAVES:

- Set the UART to 8 bits, no parity. Select a baud rate safely within the valid range. At 250kBaud, a write access transmission requires 320µs (=8 Bytes * (8+2) bits * 4µs).
- Before starting an access, activate the select pin going to the analog switch by setting it high. All other slaves select lines shall be off, unless a broadcast is desired.
- When using the optional buffer, set TMC22xx transmission send delay to an appropriate value allowing the µC to switch off the buffer before receiving reply data.
- To start a transmission, activate the TXD line buffer by setting the control pin low.
- When sending a read access request, switch off the buffer after transmission of the last stop bit is finished.
- Take into account, that all transmitted data also is received by the RXD input.

5 Register Map

This chapter gives an overview of the complete register set. Some of the registers bundling a number of single bits are detailed in extra tables. The functional practical application of the settings is detailed in dedicated chapters.

Note
- *Reset default*: All registers become reset to 0 upon power up, unless otherwise noted.
- Add 0x80 to the address **Addr** for write accesses!

NOTATION OF HEXADECIMAL AND BINARY NUMBERS	
0x	precedes a hexadecimal number, e.g. 0x04
%	precedes a multi-bit binary number, e.g. %100

NOTATION OF R/W FIELD	
R	Read only
W	Write only
R/W	Read- and writable register
R+C	Clear upon read

OVERVIEW REGISTER MAPPING

REGISTER	DESCRIPTION
General Configuration Registers	These registers contain - global configuration - global status flags - OTP read access and programming - interface configuration
Velocity Dependent Driver Feature Control Register Set	This register set offers registers for - driver current control, stand still reduction - setting thresholds for different chopper modes - internal pulse generator control
Chopper Register Set	This register set offers registers for - optimization of stealthChop2 and spreadCycle and read out of internal values - passive braking and freewheeling options - driver diagnostics - driver enable / disable

5.1 General Registers

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
RW	0x00	10	*GCONF*	**Bit**	***GCONF* – Global configuration flags**
				0	*I_scale_analog* (*Reset default=1*) 0: Use internal reference derived from 5VOUT 1: Use voltage supplied to VREF as current reference
				1	*internal_Rsense* (*Reset default: OTP*) 0: Operation with external sense resistors 1: Internal sense resistors. Use current supplied into VREF as reference for internal sense resistor. VREF pin internally is driven to GND in this mode.
				2	*en_spreadCycle* (*Reset default: OTP*) 0: stealthChop PWM mode enabled (depending on velocity thresholds). Initially switch from off to on state while in stand still, only. 1: spreadCycle mode enabled A high level on the pin SPREAD (TMC222x, only) inverts this flag to switch between both chopper modes.
				3	*shaft* 1: Inverse motor direction
				4	*index_otpw* 0: INDEX shows the first microstep position of sequencer 1: INDEX pin outputs overtemperature prewarning flag (*otpw*) instead
				5	*index_step* 0: INDEX output as selected by *index_otpw* 1: INDEX output shows step pulses from internal pulse generator (toggle upon each step)
				6	*pdn_disable* 0: PDN_UART controls standstill current reduction 1: PDN_UART input function disabled. Set this bit, when using the UART interface!
				7	*mstep_reg_select* 0: Microstep resolution selected by pins MS1, MS2 1: Microstep resolution selected by MSTEP register
				8	*multistep_filt* (*Reset default=*1) 0: No filtering of STEP pulses 1: Software pulse generator optimization enabled when fullstep frequency > 750Hz (roughly). TSTEP shows filtered step time values when active.
				9	*test_mode* 0: Normal operation 1: Enable analog test output on pin ENN (pull down resistor off), ENN treated as enabled. *IHOLD*[1..0] selects the function of DCO: 0…2: T120, DAC, VDDH *Attention: Not for user, set to 0 for normal operation!*

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
R+ WC	0x01	3	*GSTAT*	**Bit**	***GSTAT* – Global status flags** (Re-Write with '1' bit to clear respective flags)
				0	*reset* 1: Indicates that the IC has been reset since the last read access to *GSTAT*. All registers have been cleared to reset values.
				1	*drv_err* 1: Indicates, that the driver has been shut down due to overtemperature or short circuit detection since the last read access. Read DRV_STATUS for details. The flag can only be cleared when all error conditions are cleared.
				2	*uv_cp* 1: Indicates an undervoltage on the charge pump. The driver is disabled in this case. This flag is not latched and thus does not need to be cleared.
R	0x02	8	*IFCNT*		Interface transmission counter. This register becomes incremented with each successful UART interface write access. Read out to check the serial transmission for lost data. Read accesses do not change the content. The counter wraps around from 255 to 0.
W	0x03	4	*SLAVECONF*	**Bit**	***SLAVECONF***
				11..8	SENDDELAY for read access (time until reply is sent): 0, 1: 8 bit times 2, 3: 3*8 bit times 4, 5: 5*8 bit times 6, 7: 7*8 bit times 8, 9: 9*8 bit times 10, 11: 11*8 bit times 12, 13: 13*8 bit times 14, 15: 15*8 bit times
W	0x04	16	*OTP_PROG*	**Bit**	***OTP_PROGRAM* – OTP programming** Write access programs OTP memory (one bit at a time), Read access refreshes read data from OTP after a write
				2..0	*OTPBIT* Selection of OTP bit to be programmed to the selected byte location (n=0..7: programs bit n to a logic 1)
				5..4	*OTPBYTE* Selection of OTP programming location (0, 1 or 2)
				15..8	*OTPMAGIC* Set to *0xbd* to enable programming. A programming time of minimum 10ms per bit is recommended (check by reading *OTP_READ*).
R	0x05	24	*OTP_READ*	**Bit**	***OTP_READ*** (Access to OTP memory result and update) *See separate table!*
				7..0	*OTP0* byte 0 read data
				15..8	*OTP1* byte 1 read data
				23..16	*OTP2* byte 2 read data
R	0x06	10 + 8	*IOIN*	**Bit**	***INPUT*** (Reads the state of all input pins available)
				0	ENN (TMC220x)
				1	PDN_UART (TMC222x)
				2	MS1 (TMC220x), SPREAD (TMC222x)
				3	MS2 (TMC220x), DIR (TMC222x)

GENERAL CONFIGURATION REGISTERS (0x00…0x0F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
				4	DIAG (TMC220x), ENN (TMC222x)
				5	STEP (TMC222x)
				6	PDN_UART (TMC220x), MS1 (TMC222x)
				7	STEP (TMC220x), MS2 (TMC222x)
				8	SEL_A: Driver type 1: TMC220x 0: TMC222x
				9	DIR (TMC220x)
				31..24	*VERSION*: 0x20=first version of the IC Identical numbers mean full digital compatibility.
RW	0x07	5+2	*FACTORY_CONF*	4..0	*FCLKTRIM* (*Reset default: OTP*) 0…31: Lowest to highest clock frequency. Check at charge pump output. The frequency span is not guaranteed, but it is tested, that tuning to 12MHz internal clock is possible. The devices come preset to 12MHz clock frequency by OTP programming.
				9..8	*OTTRIM* (Default: OTP) %00: OT=143°C, OTPW=120°C %01: OT=150°C, OTPW=120°C %10: OT=150°C, OTPW=143°C %11: OT=157°C, OTPW=143°C

5.1.1 *OTP_READ* – OTP configuration memory

The OTP memory holds power up defaults for certain registers. All OTP memory bits are cleared to 0 by default. Programming only can set bits, clearing bits is not possible. Factory tuning of the clock frequency affects *otp0.0* to *otp0.4*. The state of these bits therefore may differ between individual ICs.

0x05: *OTP_READ* – OTP MEMORY MAP			
Bit	**Name**	**Function**	**Comment**
23	*otp2.7*	*otp_en_spreadCycle*	This flag determines if the driver defaults to spreadCycle or to stealthChop.
			0 Default: stealthChop (*GCONF.en_spreadCycle*=0) OTP 1.0 to 1.7 and 2.0 used for stealthChop spreadCycle settings: *HEND*=0; *HSTART*=5; *TOFF*=3
			1 Default: spreadCycle (*GCONF.en_spreadCycle*=1) OTP 1.0 to 1.7 and 2.0 used for spreadCycle stealthChop settings: *PWM_GRAD*=0; *TPWM_THRS*=0; *PWM_OFS*=36; *pwm_autograd*=1
22 21	*otp2.6* *otp2.5*	*OTP_IHOLD*	Reset default for standstill current *IHOLD* (used only if current reduction enabled, e.g. pin PDN_UART low). %00: *IHOLD*= 16 (53% of *IRUN*) %01: *IHOLD*= 2 (9% of *IRUN*) %10: *IHOLD*= 8 (28% of *IRUN*) %11: *IHOLD*= 24 (78% of *IRUN*) (Reset default for run current *IRUN*=31)
20 19	*otp2.4* *otp2.3*	*OTP_IHOLDDELAY*	Reset default for *IHOLDDELAY* %00: *IHOLDDELAY*= 1 %01: *IHOLDDELAY*= 2 %10: *IHOLDDELAY*= 4 %11: *IHOLDDELAY*= 8
18	*otp2.2*	*otp_PWM_FREQ*	Reset default for *PWM_FREQ*: 0: *PWM_FREQ*=%01=2/683 1: *PWM_FREQ*=%10=2/512
17	*otp2.1*	*otp_PWM_REG*	Reset default for *PWM_REG*: 0: *PWM_REG*=%1000: max. 4 increments / cycle 1: *PWM_REG*=%0010: max. 1 increment / cycle
16	*otp2.0*	*otp_PWM_OFS*	Depending on *otp_en_spreadCycle* 0: 0: *PWM_OFS*=36 1: *PWM_OFS*=00 (no feed forward scaling); *pwm_autograd*=0
		OTP_CHOPCONF8	1 Reset default for *CHOPCONF*.8 (*hend1*)
15 14 13	*otp1.7* *otp1.6* *otp1.5*	*OTP_TPWMTHRS*	Depending on *otp_en_spreadCycle* 0 Reset default for *TPWM_THRS* as defined by (0..7): 0: *TPWM_THRS*= 0 1: *TPWM_THRS*= 200 2: *TPWM_THRS*= 300 3: *TPWM_THRS*= 400 4: *TPWM_THRS*= 500 5: *TPWM_THRS*= 800 6: *TPWM_THRS*= 1200 7: *TPWM_THRS*= 4000
		OTP_CHOPCONF7…5	1 Reset default for *CHOPCONF*.5 to *CHOPCONF*.7 (*hstrt1*, *hstrt2* and *hend0*)
12	*otp1.4*	*otp_pwm_autograd*	Depending on *otp_en_spreadCycle* 0: 0: *pwm_autograd*=1 1: *pwm_autograd*=0

0x05: *OTP_READ* – OTP MEMORY MAP			
Bit	**Name**	**Function**	**Comment**
		OTP_CHOPCONF4	1 Reset default for CHOPCONF.4 (*hstrt0*); (*pwm_autograd*=1)
11	*otp1.3*	*OTP_PWM_GRAD*	Depending on *otp_en_spreadCycle*
10	*otp1.2*		0 Reset default for *PWM_GRAD* as defined by (0..15): 0: *PWM_GRAD*= 14 1: *PWM_GRAD*= 16 2: *PWM_GRAD*= 18 3: *PWM_GRAD*= 21 4: *PWM_GRAD*= 24 5: *PWM_GRAD*= 27 6: *PWM_GRAD*= 31 7: *PWM_GRAD*= 35 8: *PWM_GRAD*= 40 9: *PWM_GRAD*= 46 10: *PWM_GRAD*= 52 11: *PWM_GRAD*= 59 12: *PWM_GRAD*= 67 13: *PWM_GRAD*= 77 14: *PWM_GRAD*= 88 15: *PWM_GRAD*= 100
9	*otp1.1*		
8	*otp1.0*		
		OTP_CHOPCONF3…0	1 Reset default for *CHOPCONF*.0 to *CHOPCONF*.3 (*TOFF*)
7	*otp0.7*	*otp_TBL*	Reset default for *TBL*: 0: *TBL*=%10 1: *TBL*=%01
6	*otp0.6*	*otp_internalRsense*	Reset default for *GCONF.internal_Rsense* 0: External sense resistors 1: Internal sense resistors
5	*otp0.5*	*otp_OTTRIM*	Reset default for *OTTRIM*: 0: *OTTRIM*= %00 (143°C) 1: *OTTRIM*= %01 (150°C) (internal power stage temperature about 10°C above the sensor temperature limit)
4	*otp0.4*	*OTP_FCLKTRIM*	Reset default for *FCLKTRIM* 0: lowest frequency setting 31: highest frequency setting *Attention: This value is pre-programmed by factory clock trimming to the default clock frequency of 12MHz and differs between individual ICs! It should not be altered.*
3	*otp0.3*		
2	*otp0.2*		
1	*otp0.1*		
0	*otp0.0*		

5.2 Velocity Dependent Control

VELOCITY DEPENDENT DRIVER FEATURE CONTROL REGISTER SET (0X10…0X1F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
W	0x10	5 + 5 + 4	*IHOLD_IRUN*	**Bit**	***IHOLD_IRUN* – Driver current control**
				4..0	*IHOLD* (*Reset default: OTP*) Standstill current (0=1/32 … 31=32/32) In combination with stealthChop mode, setting *IHOLD*=0 allows to choose freewheeling or coil short circuit (passive braking) for motor stand still.
				12..8	*IRUN* (*Reset default=31*) Motor run current (0=1/32 … 31=32/32) *Hint:* Choose sense resistors in a way, that normal *IRUN* is 16 to 31 for best microstep performance.
				19..16	*IHOLDDELAY* (*Reset default: OTP*) Controls the number of clock cycles for motor power down after standstill is detected (*stst*=1) and *TPOWERDOWN* has expired. The smooth transition avoids a motor jerk upon power down. 0: instant power down 1..15: Delay per current reduction step in multiple of 2^18 clocks
W	0x11	8	*TPOWER DOWN*	*TPOWERDOWN* (*Reset default=20*) Sets the delay time from stand still (*stst*) detection to motor current power down. Time range is about 0 to 5.6 seconds. *0…((2^8)-1)* * 2^18 t_{CLK} *Attention: A minimum setting of 2 is required to allow automatic tuning of stealthChop PWM_OFFS_AUTO.*	
R	0x12	20	*TSTEP*	Actual measured time between two 1/256 microsteps derived from the step input frequency in units of 1/fCLK. Measured value is (2^20)-1 in case of overflow or stand still. The *TSTEP* related threshold uses a hysteresis of 1/16 of the compare value to compensate for jitter in the clock or the step frequency: (*Txxx**15/16)-1 is the lower compare value for each *TSTEP* based comparison. This means, that the lower switching velocity equals the calculated setting, but the upper switching velocity is higher as defined by the hysteresis setting.	
W	0x13	20	*TPWMTHRS*	Sets the upper velocity for stealthChop voltage PWM mode. *TSTEP* ≥ *TPWMTHRS* - stealthChop PWM mode is enabled, if configured When the velocity exceeds the limit set by *TPWMTHRS*, the driver switches to spreadCycle. 0: Disabled	
W	0x22	24	*VACTUAL*	*VACTUAL* allows moving the motor by UART control. It gives the motor velocity in +-(2^23)-1 [µsteps / t] 0: Normal operation. Driver reacts to STEP input. */*=0: Motor moves with the velocity given by *VACTUAL*. Step pulses can be monitored via INDEX output. The motor direction is controlled by the sign of *VACTUAL*.	

5.3 Sequencer Registers

The sequencer registers have a pure informative character and are read-only. They help for special cases like storing the last motor position before power off in battery powered applications.

MICROSTEPPING CONTROL REGISTER SET (0X60…0X6B)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	**Range [Unit]**
R	0x6A	10	*MSCNT*	Microstep counter. Indicates actual position in the microstep table for *CUR_A*. *CUR_B* uses an offset of 256 into the table. Reading out *MSCNT* allows determination of the motor position within the electrical wave.	0…1023
R	0x6B	9 + 9	*MSCURACT*	bit 8… 0: *CUR_A* (signed): Actual microstep current for motor phase A as read from the internal sine wave table (not scaled by current setting) bit 24… 16: *CUR_B* (signed): Actual microstep current for motor phase B as read from the internal sine wave table (not scaled by current setting)	+/-0…255

5.4 Chopper Control Registers

DRIVER REGISTER SET (0X6C…0X7F)						
R/W	**Addr**	**n**	***Register***	**Description */ bit names***		**Range [Unit]**
RW	0x6C	32	***CHOPCONF***	Chopper and driver configuration *See separate table!*		*Reset default= 0x10000053*
R	0x6F	32	***DRV_ STATUS***	Driver status flags and current level read back *See separate table!*		
RW	0x70	22	***PWMCONF***	stealthChop PWM chopper configuration *See separate table!*		*Reset default= 0xC10D0024*
R	0x71	9+8	*PWM_SCALE*	Results of stealthChop amplitude regulator. These values can be used to monitor automatic PWM amplitude scaling (255=max. voltage).		
				bit 7… 0	*PWM_SCALE_SUM*: Actual PWM duty cycle. This value is used for scaling the values *CUR_A* and *CUR_B* read from the sine wave table.	0…255
				bit 24… 16	*PWM_SCALE_AUTO*: 9 Bit signed offset added to the calculated PWM duty cycle. This is the result of the automatic amplitude regulation based on current measurement.	signed -255…+255
R	0x72	8+8	*PWM_AUTO*	These automatically generated values can be read out in order to determine a default */* power up setting for *PWM_GRAD* and *PWM_OFS*.		
				bit 7… 0	*PWM_OFS_AUTO*: Automatically determined offset value	0…255
				bit 23… 16	*PWM_GRAD_AUTO*: Automatically determined gradient value	0…255

5.4.1 *CHOPCONF* – Chopper Configuration

0x6C: *CHOPCONF* – CHOPPER CONFIGURATION			
Bit	**Name**	**Function**	**Comment**
31	*diss2vs*	Low side short protection disable	0: Short protection low side is on 1: Short protection low side is disabled
30	*diss2g*	short to GND protection disable	0: Short to GND protection is on 1: Short to GND protection is disabled
29	*dedge*	enable double edge step pulses	1: Enable step impulse at each step edge to reduce step frequency requirement. This mode is not compatible with the step filtering function (*multistep_filt*)
28	*intpol*	interpolation to 256 microsteps	1: The actual microstep resolution (*MRES*) becomes extrapolated to 256 microsteps for smoothest motor operation. (Default: 1)
27	*mres3*	*MRES* micro step resolution	%0000: Native 256 microstep setting.
26	*mres2*		
25	*mres1*		%0001 … %1000: 128, 64, 32, 16, 8, 4, 2, FULLSTEP Reduced microstep resolution. The resolution gives the number of microstep entries per sine quarter wave. When choosing a lower microstep resolution, the driver automatically uses microstep positions which result in a symmetrical wave. Number of microsteps per step pulse = 2^*MRES* (Selection by pins unless disabled by *GCONF.mstep_reg_select*)
24	*mres0*		
23	-	reserved	set to 0
22			
21			
20			
19			
18			
17	*vsense*	sense resistor voltage based current scaling	0: Low sensitivity, high sense resistor voltage 1: High sensitivity, low sense resistor voltage
16	*tbl1*	*TBL* blank time select	%00 … %11: Set comparator blank time to 16, 24, 32 or 40 clocks *Hint*: %00 or %01 is recommended for most applications (Default: OTP)
15	*tbl0*		
14	-	reserved	set to 0
13			
12			
11			
10	*hend3*	*HEND* hysteresis low value *OFFSET* sine wave offset	%0000 … %1111: Hysteresis is -3, -2, -1, 0, 1, …, 12 (1/512 of this setting adds to current setting) This is the hysteresis value which becomes used for the hysteresis chopper. (Default: OTP, resp. 5 in stealthChop mode)
9	*hend2*		
8	*hend1*		
7	*hend0*		
6	*hstrt2*	*HSTRT* hysteresis start value added to *HEND*	%000 … %111: Add 1, 2, …, 8 to hysteresis low value *HEND* (1/512 of this setting adds to current setting) *Attention: Effective HEND+HSTRT ≤ 16.* *Hint:* Hysteresis decrement is done each 16 clocks
5	*hstrt1*		
4	*hstrt0*		

0x6C: *CHOPCONF* – CHOPPER CONFIGURATION			
Bit	**Name**	**Function**	**Comment**
			(Default: OTP, resp. 0 in stealthChop mode)
3	*toff3*	*TOFF* off time and driver enable	Off time setting controls duration of slow decay phase N_{CLK}= 12 + 32**TOFF* %0000: Driver disable, all bridges off %0001: 1 – use only with *TBL* ≥ 2 %0010 … %1111: 2 … 15 (Default: OTP, resp. 3 in stealthChop mode)
2	*toff2*		
1	*toff1*		
0	*toff0*		

5.4.2 *PWMCONF* – Voltage PWM Mode stealthChop

0x70: *PWMCONF* – VOLTAGE MODE PWM STEALTHCHOP			
Bit	**Name**	**Function**	**Comment**
31 30 29 28	*PWM_LIM*	PWM automatic scale amplitude limit when switching on	Limit for *PWM_SCALE_AUTO* when switching back from spreadCycle to stealthChop. This value defines the upper limit for bits 7 to 4 of the automatic current control when switching back. It can be set to reduce the current jerk during mode change back to stealthChop. It does not limit *PWM_GRAD* or *PWM_GRAD_AUTO* offset. (Default = 12)
27 26 25 24	*PWM_REG*	Regulation loop gradient	User defined maximum PWM amplitude change per half wave when using *pwm_autoscale*=1. (1…15): 1: 0.5 increments (slowest regulation) 2: 1 increment (default with *OTP*2.1=1) 3: 1.5 increments 4: 2 increments … 8: 4 increments (default with *OTP*2.1=0) … 15: 7.5 increments (fastest regulation)
23	-	reserved	set to 0
22	-	reserved	set to 0
21 20	*freewheel1* *freewheel0*	Allows different standstill modes	Stand still option when motor current setting is zero (*I_HOLD*=0). %00: Normal operation %01: Freewheeling %10: Coil shorted using LS drivers %11: Coil shorted using HS drivers
19	*pwm_ autograd*	PWM automatic gradient adaptation	0: Fixed value for *PWM_GRAD* (*PWM_GRAD_AUTO* = *PWM_GRAD*) 1: Automatic tuning (only with *pwm_autoscale*=1) *PWM_GRAD_AUTO* is initialized with *PWM_GRAD* and becomes optimized automatically during motion. Preconditions 1. *PWM_OFS_AUTO* has been automatically initialized. This requires standstill at IRUN for >130ms in order to a) detect standstill b) wait > 128 chopper cycles at *IRUN* and c) regulate *PWM_OFS_AUTO* so that -1 < *PWM_SCALE_AUTO* < 1 2. Motor running and 1.5 * *PWM_OFS_AUTO* < *PWM_SCALE_SUM* < 4* *PWM_OFS_AUTO* and *PWM_SCALE_SUM* < 255. Time required for tuning *PWM_GRAD_AUTO* About 8 fullsteps per change of +/-1.
18	*pwm_ autoscale*	PWM automatic amplitude scaling	0: User defined feed forward PWM amplitude. The current settings *IRUN* and *IHOLD* have no influence! The resulting PWM amplitude (limited to 0…255) is: *PWM_OFS * ((CS_ACTUAL+1) / 32)* *+ PWM_GRAD * 256 / TSTEP* 1: Enable automatic current control *(Reset default)*
17	*pwm_freq1*	PWM frequency	%00: $f_{PWM}=2/1024\ f_{CLK}$

0x70: *PWMCONF* – VOLTAGE MODE PWM STEALTHCHOP			
Bit	**Name**	**Function**	**Comment**
16	*pwm_freq0*	selection	%01: $f_{PWM}=2/683\ f_{CLK}$ %10: $f_{PWM}=2/512\ f_{CLK}$ %11: $f_{PWM}=2/410\ f_{CLK}$
15 14 13 12 11 10 9 8	*PWM_ GRAD*	User defined amplitude gradient	Velocity dependent gradient for PWM amplitude: *PWM_GRAD* * 256 / *TSTEP* This value is added to *PWM_AMPL* to compensate for the velocity-dependent motor back-EMF. With automatic scaling (*pwm_autoscale*=1) the value is used for first initialization, only. Set *PWM_GRAD* to the application specific value (it can be read out from *PWM_GRAD_AUTO*) to speed up the automatic tuning process. An approximate value can be stored to OTP by programming *OTP_PWM_GRAD*.
7 6 5 4 3 2 1 0	*PWM_ OFS*	User defined amplitude (offset)	User defined PWM amplitude offset (0-255) related to full motor current (*CS_ACTUAL*=31) in stand still. (*Reset default=36*) When using automatic scaling (*pwm_autoscale*=1) the value is used for initialization, only. The autoscale function starts with *PWM_SCALE_AUTO=PWM_OFS* and finds the required offset to yield the target current automatically. *PWM_OFS* = 0 will disable scaling down motor current below a motor specific lower measurement threshold. This setting should only be used under certain conditions, i.e. when the power supply voltage can vary up and down by a factor of two or more. It prevents the motor going out of regulation, but it also prevents power down below the regulation limit. *PWM_OFS* > 0 allows automatic scaling to low PWM duty cycles even below the lower regulation threshold. This allows low (standstill) current settings based on the actual (hold) current scale (register *IHOLD_IRUN*).

5.4.3 DRV_STATUS – Driver Status Flags

0x6F: DRV_STATUS – DRIVER STATUS FLAGS AND CURRENT LEVEL READ BACK			
Bit	**Name**	**Function**	**Comment**
31	*stst*	standstill indicator	This flag indicates motor stand still in each operation mode. This occurs 2^20 clocks after the last step pulse.
30	*stealth*	stealthChop indicator	1: Driver operates in stealthChop mode 0: Driver operates in spreadCycle mode
29 28 27 26 25 24	-	reserved	Ignore these bits.
23 22 21	-	reserved	Ignore these bits.
20 19 18 17 16	*CS_ ACTUAL*	actual motor current / smart energy current	Actual current control scaling, for monitoring the function of the automatic current scaling.
15 14 13 12	-	reserved	Ignore these bits.
11	*t157*	157°C comparator	1: Temperature threshold is exceeded
10	*t150*	150°C comparator	1: Temperature threshold is exceeded
9	*t143*	143°C comparator	1: Temperature threshold is exceeded
8	*t120*	120°C comparator	1: Temperature threshold is exceeded
7	*olb*	open load indicator phase B	1: Open load detected on phase A or B. *Hint:* This is just an informative flag. The driver takes no action upon it. False detection may occur in fast motion and standstill. Check during slow motion, only.
6	*ola*	open load indicator phase A	
5	*s2vsb*	low side short indicator phase B	1: Short on low-side MOSFET detected on phase A or B. The driver becomes disabled. The flags stay active, until the driver is disabled by software (TOFF=0) or by the ENN input. Flags are separate for both chopper modes.
4	*s2vsa*	low side short indicator phase A	
3	*s2gb*	short to ground indicator phase B	1: Short to GND detected on phase A or B. The driver becomes disabled. The flags stay active, until the driver is disabled by software (TOFF=0) or by the ENN input. Flags are separate for both chopper modes.
2	*s2ga*	short to ground indicator phase A	
1	*ot*	overtemperature flag	1: The selected overtemperature limit has been reached. Drivers become disabled until *otpw* is also cleared due to cooling down of the IC. The overtemperature flag is common for both bridges.
0	*otpw*	overtemperature pre-warning flag	1: The selected overtemperature pre-warning threshold is exceeded. The overtemperature pre-warning flag is common for both bridges.

6 stealthChop™

stealthChop is an extremely quiet mode of operation for stepper motors. It is based on a voltage mode PWM. In case of standstill and at low velocities, the motor is absolutely noiseless. Thus, stealthChop operated stepper motor applications are very suitable for indoor or home use. The motor operates absolutely free of vibration at low velocities. With stealthChop, the motor current is applied by driving a certain effective voltage into the coil, using a voltage mode PWM. With the enhanced stealthChop2, the driver automatically adapts to the application for best performance. No more configurations are required. Optional configuration allows for tuning the setting in special cases, or for storing initial values for the automatic adaptation algorithm. For high velocity drives spreadCycle should be considered in combination with stealthChop.

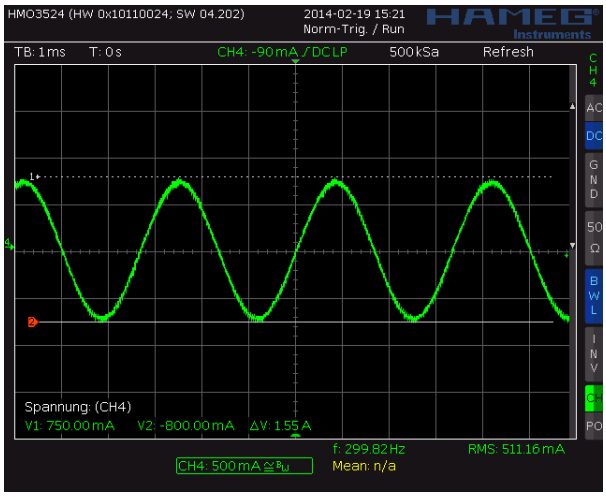


Figure 6.1 Motor coil sine wave current with stealthChop (measured with current probe)

6.1 Automatic Tuning

stealthChop2 integrates an automatic tuning procedure (AT), which adapts the most important operating parameters to the motor automatically. This way, stealthChop2 allows high motor dynamics and supports powering down the motor to very low currents. Just two steps have to be respected by the motion controller for best results: Start with the motor in standstill, but powered with nominal run current (AT#1). Move the motor at a medium velocity, e.g. as part of a homing procedure (AT#2). Figure 6.2 shows the tuning procedure.
Border conditions in for AT#1 and AT#2 are shown in the following table:

AUTOMATIC TUNING TIMING AND BORDER CONDITIONS

Step	Parameter	Conditions	Duration
AT#1	*PWM_OFS_AUTO*	- Motor in standstill and actual current scale (*CS*) is identical to run current (*IRUN*). - If standstill reduction is enabled (pin PDN_UART=0), an initial step pulse switches the drive back to run current. - Pins VS and VREF at operating level.	≤ 2^20+2*2^18 t_{CLK}, ≤ 130ms (with internal clock)
AT#2	*PWM_GRAD_AUTO*	- Motor must move at a velocity, where a significant amount of back EMF is generated and where the full run current can be reached. Conditions: - 1.5 * *PWM_OFS_AUTO* < *PWM_SCALE_SUM* < 4 * *PWM_OFS_AUTO* - *PWM_SCALE_SUM* < 255. *Hint: A typical range is 60-300 RPM. Determine best conditions with the evaluation board.*	8 fullsteps are required for a change of +/-1. For a typical motor with *PWM_GRAD_AUTO* optimum at 64 or less, up to 400 fullsteps are required when starting from OTP default 14.

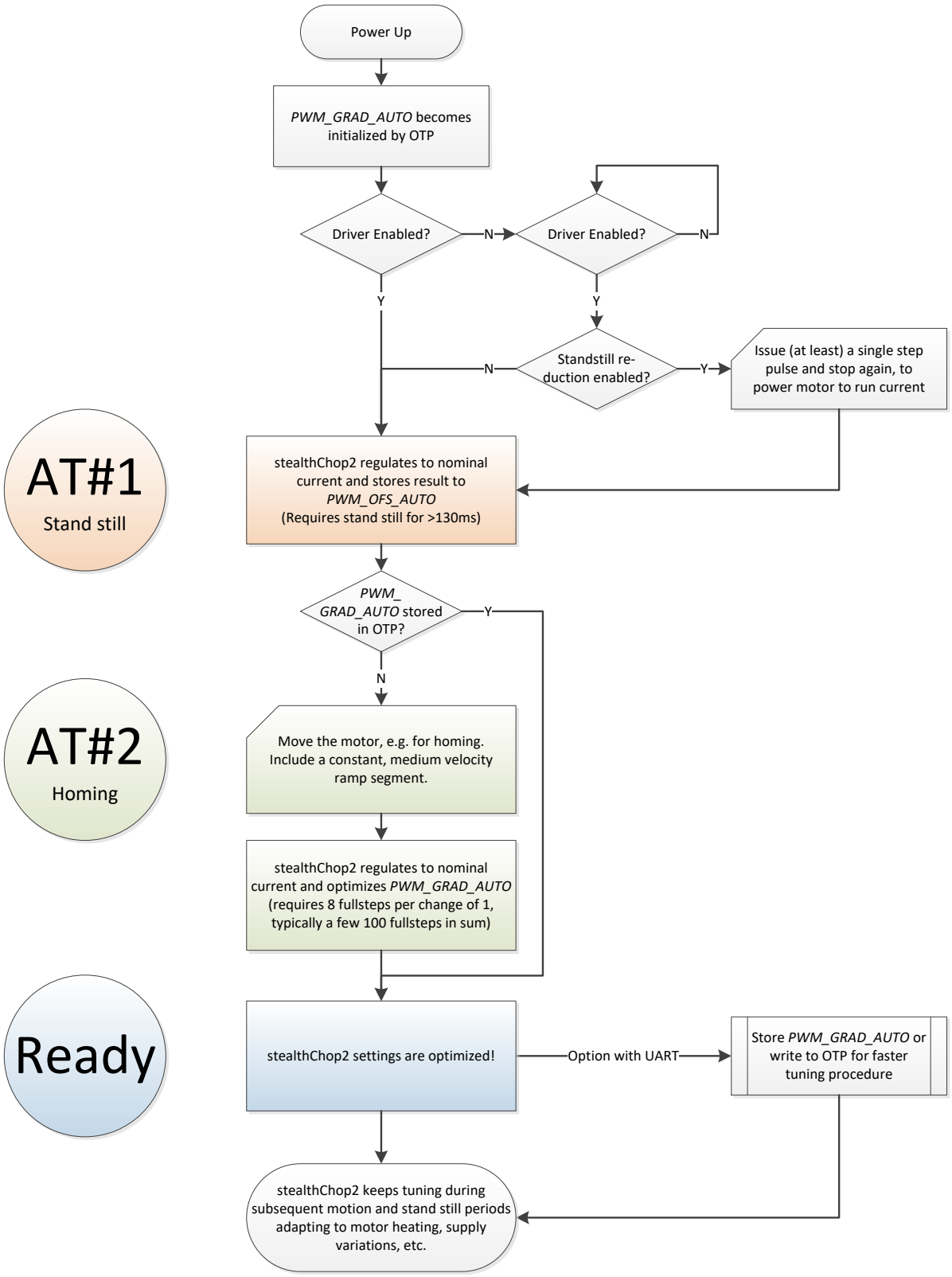


Figure 6.2 stealthChop2 automatic tuning procedure

Attention:
Modifying VREF or the supply voltage VS invalidates the result of the automatic tuning process. Motor current regulation cannot compensate significant changes until next AT#1 phase. Automatic tuning adapts to changed conditions whenever AT#1 and AT#2 conditions are fulfilled in the later operation.

6.2 stealthChop Options

UART

In order to match the motor current to a certain level, the effective PWM voltage becomes scaled depending on the actual motor velocity. Several additional factors influence the required voltage level to drive the motor at the target current: The motor resistance, its back EMF (i.e. directly proportional to its velocity) as well as the actual level of the supply voltage. Two modes of PWM regulation are provided: The automatic tuning mode (AT) using current feedback (*pwm_autoscale* = 1, *pwm_autograd* = 1) and a feed forward velocity controlled mode (*pwm_autoscale* = 0). The feed forward velocity controlled mode will not react to a change of the supply voltage or to events like a motor stall, but it provides very stable amplitude. It does not use nor require any means of current measurement. This is perfect when motor type and supply voltage are well known. Therefore we recommend the automatic mode, unless current regulation is not satisfying in the given operating conditions.

It is recommended to operate in automatic tuning mode.

Non-automatic mode (*pwm_autoscale*=0) should be taken into account only with well-known motor and operating conditions. In this case, programming via the UART interface is required. The operating parameters *PWM_GRAD* and *PWM_OFS* can be determined in automatic tuning mode initially.

Hint: In non-automatic mode the power supply current directly reflects mechanical load on the motor.

The stealthChop PWM frequency can be chosen in four steps in order to adapt the frequency divider to the frequency of the clock source. A setting in the range of 20-50kHz is good for most applications. It balances low current ripple and good higher velocity performance vs. dynamic power dissipation.

CHOICE OF PWM FREQUENCY FOR STEALTHCHOP				
Clock frequency f_{CLK}	**PWM_FREQ=%00** $f_{PWM}=2/1024\ f_{CLK}$	**PWM_FREQ=%01** $f_{PWM}=2/683\ f_{CLK}$ (default)	**PWM_FREQ=%10** $f_{PWM}=2/512\ f_{CLK}$ (OTP option)	**PWM_FREQ=%11** $f_{PWM}=2/410\ f_{CLK}$
18MHz	35.2kHz	52.7kHz	70.3kHz	87.8kHz
16MHz	31.3kHz	46.9kHz	62.5kHz	78.0kHz
12MHz (internal)	23.4kHz	35.1kHz	46.9kHz	58.5kHz
10MHz	19.5kHz	29.3kHz	39.1kHz	48.8kHz
8MHz	15.6kHz	23.4kHz	31.2kHz	39.0kHz

Table 6.1 Choice of PWM frequency – green / light green: recommended

6.3 stealthChop Current Regulator

In stealthChop voltage PWM mode, the autoscaling function (*pwm_autoscale* = 1, *pwm_auto_grad* = 1) regulates the motor current to the desired current setting. Automatic scaling is used as part of the automatic tuning process (AT), and for subsequent tracking of changes within the motor parameters. The driver measures the motor current during the chopper on time and uses a proportional regulator to regulate *PWM_SCALE_AUTO* in order match the motor current to the target current. *PWM_REG* is the proportionality coefficient for this regulator. Basically, the proportionality coefficient should be as small as possible in order to get a stable and soft regulation behavior, but it must be large enough to allow the driver to quickly react to changes caused by variation of the motor target current (e.g. change of VREF). During initial tuning step AT#2, *PWM_REG* also compensates for the change of motor velocity. Therefore, a high acceleration during AT#2 will require a higher setting of *PWM_REG*. With careful selection of homing velocity and acceleration, a minimum setting of the regulation gradient often is sufficient (*PWM_REG*=1). *PWM_REG* setting should be optimized for the fastest required acceleration and deceleration ramp (compare Figure 6.3 and Figure 6.4). The quality of the setting *PWM_REG* in phase AT#2 and the finished automatic tuning procedure (or non-automatic settings for *PWM_OFS* and *PWM_GRAD*) can be examined when monitoring motor current during an acceleration phase Figure 6.5.

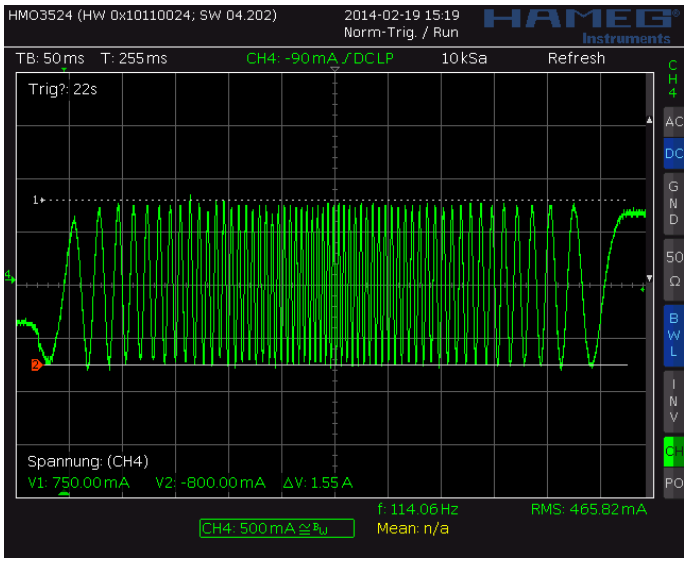


Figure 6.3 Scope shot: good setting for PWM_REG

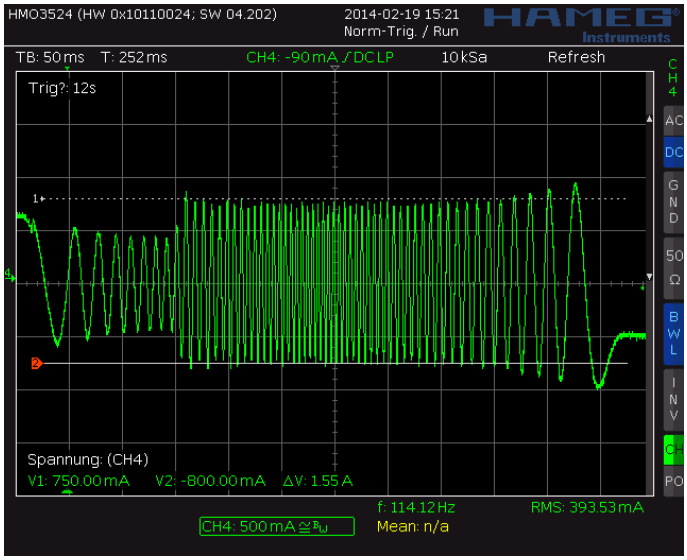


Figure 6.4 Scope shot: too small setting for PWM_REG during AT#2

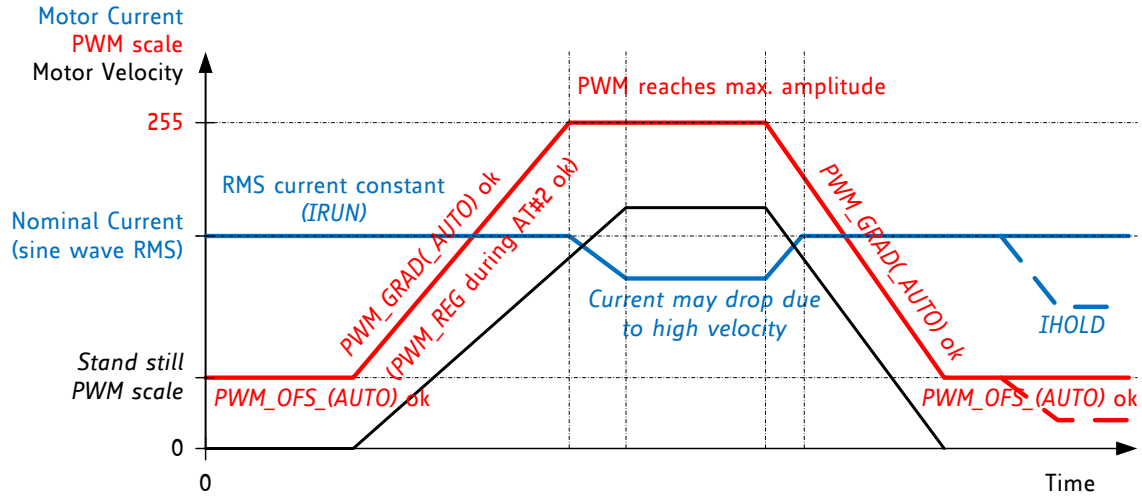


Figure 6.5 Successfully determined PWM_GRAD(_AUTO) and PWM_OFS(_AUTO)

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 14.

6.3.1 Lower Current Limit

The stealthChop current regulator imposes a lower limit for motor current regulation. As the coil current can be measured in the shunt resistor during chopper on phase only, a minimum chopper duty cycle allowing coil current regulation is given by the blank time as set by *TBL* and by the chopper frequency setting. Therefore, the motor specific minimum coil current in stealthChop autoscaling mode rises with the supply voltage and with the chopper frequency. A lower blanking time allows a lower current limit. It is important for the correct determination of *PWM_OFS_AUTO*, that in AT#1 the run current set by the sense resistor, VREF and *IRUN* is well within the regulation range. Lower currents (e.g. for standstill power down) are automatically realized based on *PWM_OFS_AUTO* and *PWM_GRAD_AUTO* respectively based on *PWM_OFS* and *PWM_GRAD* with non-automatic current scaling. The freewheeling option allows going to zero motor current.

Lower motor coil current limit for stealthChop2 automatic tuning:

$$I_{Lower\ Limit} = t_{BLANK} * f_{PWM} * \frac{V_M}{R_{COIL}}$$

With V_M the motor supply voltage and R_{COIL} the motor coil resistance.
$I_{Lower\ Limit}$ can be treated as a thumb value for the minimum nominal *IRUN* motor current setting.

EXAMPLE:

A motor has a coil resistance of 5Ω, the supply voltage is 24V. With *TBL*=%01 and *PWM_FREQ*=%00, t_{BLANK} is 24 clock cycles, f_{PWM} is 2/(1024 clock cycles):

$$I_{Lower\ Limit} = 24\ t_{CLK} * \frac{2}{1024\ t_{CLK}} * \frac{24V}{5\Omega} = \frac{24}{512} * \frac{24V}{5\Omega} = 225mA$$

This means, the motor target current for automatic tuning must be 225mA or more, taking into account all relevant settings. This lower current limit also applies for modification of the motor current via the analog input VREF.

Attention
For automatic tuning, a lower coil current limit applies. The motor current in automatic tuning phase AT#1 must exceed this lower limit. $I_{LOWER\ LIMIT}$ can be calculated or measured using a current probe. Setting the motor run-current or hold-current below the lower current limit during operation by modifying *IRUN* and *IHOLD* is possible after successful automatic tuning.

The lower current limit also limits the capability of the driver to respond to changes of VREF.

6.4 Velocity Based Scaling

Velocity based scaling scales the stealthChop amplitude based on the time between each two steps, i.e. based on *TSTEP*, measured in clock cycles. This concept basically does not require a current measurement, because no regulation loop is necessary. A pure velocity based scaling is available via UART programming, only, when setting *pwm_autoscale* = 0. The basic idea is to have a linear approximation of the voltage required to drive the target current into the motor. The stepper motor has a certain coil resistance and thus needs a certain voltage amplitude to yield a target current based on the basic formula I=U/R. With R being the coil resistance, U the supply voltage scaled by the PWM value, the current I results. The initial value for PWM_AMPL can be calculated:

$$PWM\_AMPL = \frac{374 * R_{COIL} * I_{COIL}}{V_M}$$

With V_M the motor supply voltage and I_{COIL} the target RMS current

The effective PWM voltage U_{PWM} (1/SQRT(2) x peak value) results considering the 8 bit resolution and 248 sine wave peak for the actual PWM amplitude shown as *PWM_SCALE*:

$$U_{PWM} = V_M * \frac{PWM\_SCALE}{256} * \frac{248}{256} * \frac{1}{\sqrt{2}} = V_M * \frac{PWM\_SCALE}{374}$$

With rising motor velocity, the motor generates an increasing back EMF voltage. The back EMF voltage is proportional to the motor velocity. It reduces the PWM voltage effective at the coil resistance and thus current decreases. The TMC22xx provides a second velocity dependent factor (*PWM_GRAD*) to compensate for this. The overall effective PWM amplitude (*PWM_SCALE_SUM*) in this mode automatically is calculated in dependence of the microstep frequency as:

$$PWM\_SCALE\_SUM = PWM\_OFS + PWM\_GRAD * 256 * \frac{f_{STEP}}{f_{CLK}}$$

With f_{STEP} being the microstep frequency for 256 microstep resolution equivalent and f_{CLK} the clock frequency supplied to the driver or the actual internal frequency

As a first approximation, the back EMF subtracts from the supply voltage and thus the effective current amplitude decreases. This way, a first approximation for *PWM_GRAD* setting can be calculated:

$$PWM\_GRAD = C_{BEMF} \left[\frac{V}{\frac{rad}{s}}\right] * 2\pi * \frac{f_{CLK} * 1.46}{V_M * MSPR}$$

C_{BEMF} is the back EMF constant of the motor in Volts per radian/second.
MSPR is the number of microsteps per rotation, e.g. 51200 = 256µsteps multiplied by 200 fullsteps for a 1.8° motor.

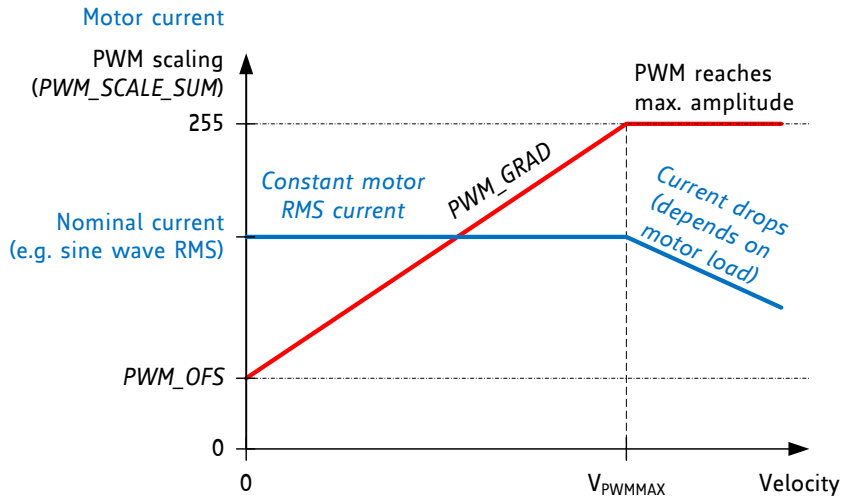


Figure 6.6 Velocity based PWM scaling (pwm_autoscale=0)

Hint
The values for *PWM_OFS* and *PWM_GRAD* can easily be optimized by tracing the motor current with a current probe on the oscilloscope. Alternatively, automatic tuning determines these values and they can be read out from *PWM_OFS_AUTO* and *PWM_GRAD_AUTO*.

UNDERSTANDING THE BACK EMF CONSTANT OF A MOTOR

The back EMF constant is the voltage a motor generates when turned with a certain velocity. Often motor datasheets do not specify this value, as it can be deducted from motor torque and coil current rating. Within SI units, the numeric value of the back EMF constant C_{BEMF} has the same numeric value as the numeric value of the torque constant. For example, a motor with a torque constant of 1 Nm/A would have a C_{BEMF} of 1V/rad/s. Turning such a motor with 1 rps (1 rps = 1 revolution per second = 6.28 rad/s) generates a back EMF voltage of 6.28V. Thus, the back EMF constant can be calculated as:

$$C_{BEMF}\left[\frac{V}{rad/s}\right] = \frac{HoldingTorque[Nm]}{2 * I_{COILNOM}[A]}$$

$I_{COILNOM}$ is the motor's rated phase current for the specified holding torque
HoldingTorque is the motor specific holding torque, i.e. the torque reached at $I_{COILNOM}$ on both coils. The torque unit is [Nm] where 1Nm = 100Ncm = 1000mNm.
The voltage is valid as RMS voltage per coil, thus the nominal current is multiplied by 2 in this formula, since the nominal current assumes a full step position, with two coils operating.

6.5 Combining stealthChop and spreadCycle

For applications requiring high velocity motion, spreadCycle may bring more stable operation in the upper velocity range. To combine no-noise operation with highest dynamic performance, the TMC22xx allows combining stealthChop and spreadCycle based on a velocity threshold (Figure 6.7). A velocity threshold (*TPWMTHRS*) can be preprogrammed to OTP to support this mode even in standalone operation. With this, stealthChop is only active at low velocities.

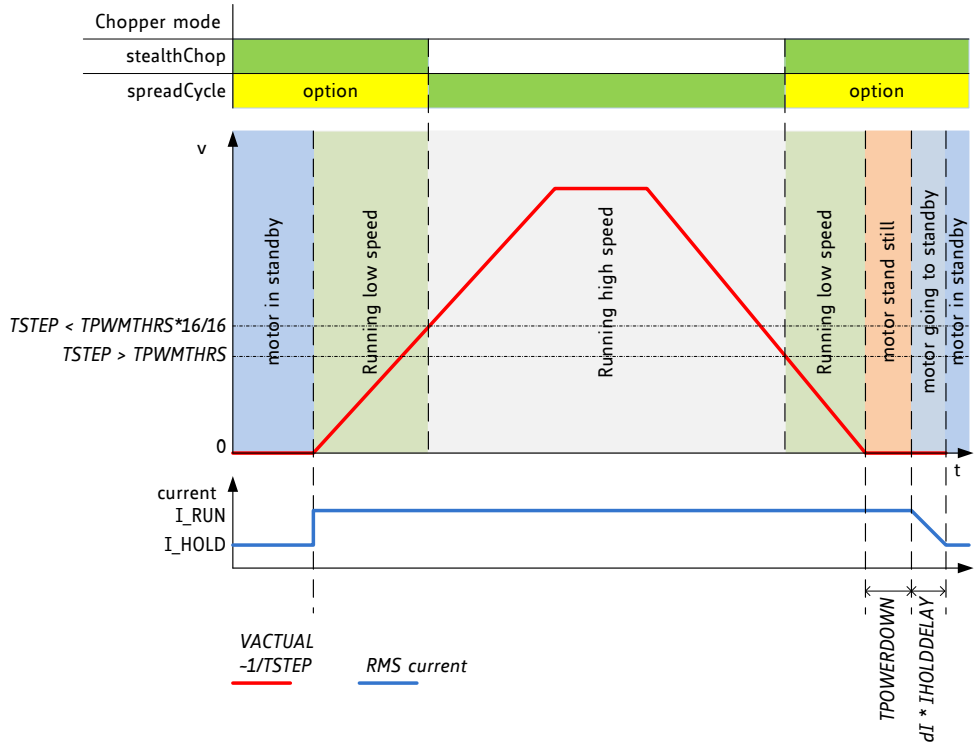


Figure 6.7 TPWMTHRS for optional switching to spreadCycle

As a first step, both chopper principles should be parameterized and optimized individually (spreadCycle settings may be programmed to OTP memory). In a next step, a transfer velocity has to be fixed. For example, stealthChop operation is used for precise low speed positioning, while spreadCycle shall be used for highly dynamic motion. *TPWMTHRS* determines the transition velocity. Read out *TSTEP* when moving at the desired velocity and program the resulting value to *TPWMTHRS*. Use a low transfer velocity to avoid a jerk at the switching point.

A jerk occurs when switching at higher velocities, because the back-EMF of the motor (which rises with the velocity) causes a phase shift of up to 90° between motor voltage and motor current. So when switching at higher velocities between voltage PWM and current PWM mode, this jerk will occur with increased intensity. A high jerk may even produce a temporary overcurrent condition (depending on the motor coil resistance). At low velocities (e.g. 1 to a few 10 RPM), it can be completely neglected for most motors. Therefore, consider the switching jerk when choosing *TPWMTHRS*. Set *TPWMTHRS* zero if you want to work with stealthChop only.

When enabling the stealthChop mode the first time using automatic current regulation, the motor must be at stand still in order to allow a proper current regulation. When the drive switches to stealthChop at a higher velocity, stealthChop logic stores the last current regulation setting until the motor returns to a lower velocity again. This way, the regulation has a known starting point when returning to a lower velocity, where stealthChop becomes re-enabled. Therefore, neither the velocity threshold nor the supply voltage must be considerably changed during the phase while the chopper is switched to a different mode, because otherwise the motor might lose steps or the instantaneous current might be too high or too low.

A motor stall or a sudden change in the motor velocity may lead to the driver detecting a short circuit or to a state of automatic current regulation, from which it cannot recover. Clear the error flags and restart the motor from zero velocity to recover from this situation.

Hint
Start the motor from standstill when switching on stealthChop the first time and keep it stopped for at least 128 chopper periods to allow stealthChop to do initial standstill current control.

6.6 Flags in stealthChop

As stealthChop uses voltage mode driving, status flags based on current measurement respond slower, respectively the driver reacts delayed to sudden changes of back EMF, like on a motor stall.

A motor stall can lead to an overcurrent condition. Depending on the previous motor velocity, and on the coil resistance of the motor, it may trigger the overcurrent detection. With low velocities, where the back EMF is just a fraction of the supply voltage, there is no danger of triggering the short detection.

6.6.1 Open Load Flags

In stealthChop mode, status information is different from the cycle-by-cycle regulated spreadCycle mode. OLA and OLB show if the current regulation sees that the nominal current can be reached on both coils.

- A flickering OLA or OLB can result from asymmetries in the sense resistors or in the motor coils.
- An interrupted motor coil leads to a continuously active open load flag for the coil.
- One or both flags are active, if the current regulation did not succeed in scaling up to the full target current within the last few fullsteps (because no motor is attached or a high velocity exceeds the PWM limit).

If desired, do an on-demand open load test using the spreadCycle chopper, as it delivers the safest result. With stealthChop, *PWM_SCALE_SUM* can be checked to detect the correct coil resistance.

6.6.2 PWM_SCALE_SUM Informs about the Motor State

Information about the motor state is available with automatic scaling by reading out *PWM_SCALE_SUM*. As this parameter reflects the actual voltage required to drive the target current into the motor, it depends on several factors: motor load, coil resistance, supply voltage, and current setting. Therefore, an evaluation of the *PWM_SCALE_SUM* value allows checking the motor operation point. When reaching the limit (255), the current regulator cannot sustain the full motor current, e.g. due to a drop in supply volage.

6.7 Freewheeling and Passive Braking

stealthChop provides different options for motor standstill. These options can be enabled by setting the standstill current *IHOLD* to zero and choosing the desired option using the *FREEWHEEL* setting. The desired option becomes enabled after a time period specified by *TPOWERDOWN* and *IHOLD_DELAY*. Current regulation becomes frozen once the motor target current is at zero current in order to ensure a quick startup. With the freewheeling options, both freewheeling and passive braking can be realized. Passive braking is an effective eddy current motor braking, which consumes a minimum of energy, because no active current is driven into the coils. However, passive braking will allow slow turning of the motor when a continuous torque is applied.

Hint
Operate the motor within your application when exploring stealthChop. Motor performance often is better with a mechanical load, because it prevents the motor from stalling due mechanical oscillations which can occur without load.

PARAMETERS RELATED TO STEALTHCHOP			
Parameter	**Description**	**Setting**	**Comment**
en_spread_cycle	General disable for use of stealthChop (register *GCONF*). The input SPREAD is XORed to this flag.	1	Do not use stealthChop
		0	stealthChop enabled
TPWMTHRS	Specifies the upper velocity for operation in stealthChop. Entry the *TSTEP* reading (time between two microsteps) when operating at the desired threshold velocity.	0 … 1048575	stealthChop is disabled if *TSTEP* falls *TPWMTHRS*
PWM_LIM	Limiting value for limiting the current jerk when switching from spreadCycle to stealthChop. Reduce the value to yield a lower current jerk.	0 … 15	Upper four bits of 8 bit amplitude limit (Default=12)
pwm_autoscale	Enable automatic current scaling using current measurement or use forward controlled velocity based mode.	0	Forward controlled mode
		1	Automatic scaling with current regulator
pwm_autograd	Enable automatic tuning of *PWM_GRAD_AUTO*	0	disable, use *PWM_GRAD* from register instead
		1	enable
PWM_FREQ	PWM frequency selection. Use the lowest setting giving good results. The frequency measured at each of the chopper outputs is half of the effective chopper frequency f_{PWM}.	0	$f_{PWM}=2/1024\ f_{CLK}$
		1	$f_{PWM}=2/683\ f_{CLK}$
		2	$f_{PWM}=2/512\ f_{CLK}$
		3	$f_{PWM}=2/410\ f_{CLK}$
PWM_REG	User defined PWM amplitude (gradient) for velocity based scaling or regulation loop gradient when *pwm_autoscale*=1.	1 … 15	Results in 0.5 to 7.5 steps for *PWM_SCALE_AUTO* regulator per fullstep
PWM_OFS	User defined PWM amplitude (offset) for velocity based scaling and initialization value for automatic tuning of *PWM_OFFS_AUTO*.	0 … 255	*PWM_OFS*=0 disables linear current scaling based on current setting
PWM_GRAD	User defined PWM amplitude (gradient) for velocity based scaling and initialization value for automatic tuning of *PWM_GRAD_AUTO*.	0 … 255	Reset value can be pre-programmed by OTP
FREEWHEEL	Stand still option when motor current setting is zero (*I_HOLD*=0). Only available with stealthChop enabled. The freewheeling option makes the motor easy movable, while both coil short options realize a passive brake.	0	Normal operation
		1	Freewheeling
		2	Coil short via LS drivers
		3	Coil short cia HS drivers
PWM_SCALE_AUTO	Read back of the actual stealthChop voltage PWM scaling correction as determined by the current regulator.	-255 … 255	(read only) Scaling value becomes frozen when operating in spreadCycle
PWM_SCALE_AUTO PWM_OFS_AUTO	Allow monitoring of the automatic tuning and determination of initial values for *PWM_OFS* and *PWM_GRAD*.	0 … 255	(read only)
TOFF	General enable for the motor driver, the actual value does not influence stealthChop	0	Driver off
		1 … 15	Driver enabled
TBL	Comparator *blank time*. This time needs to safely cover the switching event and the duration of the ringing on the sense resistor. Choose a setting of 1 or 2 for typical applications. For higher capacitive loads, 3 may be required. Lower settings allow stealthChop to regulate down to lower coil current values.	0	16 t_{CLK}
		1	24 t_{CLK}
		2	32 t_{CLK}
		3	40 t_{CLK}

7 spreadCycle Chopper

While stealthChop is a voltage mode PWM controlled chopper, spreadCycle is a cycle-by-cycle current control. Therefore, it can react extremely fast to changes in motor velocity or motor load. spreadCycle will give better performance in medium to high velocity range for motors and applications which tend to resonance. The currents through both motor coils are controlled using choppers. The choppers work independently of each other. In Figure 7.1 the different chopper phases are shown.

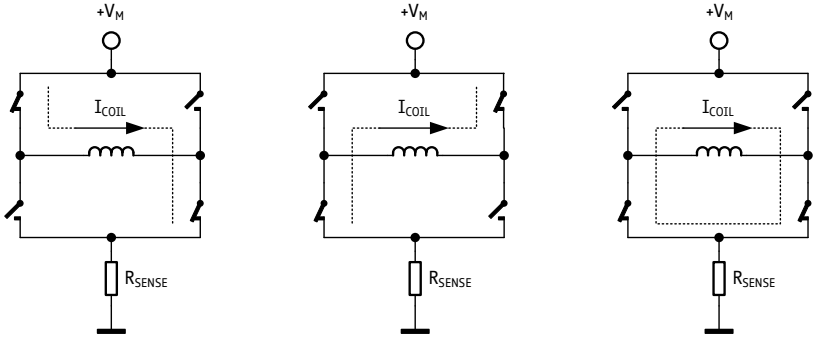


Figure 7.1 Chopper phases

Although the current could be regulated using only on phases and fast decay phases, insertion of the slow decay phase is important to reduce electrical losses and current ripple in the motor. The duration of the slow decay phase is specified in a control parameter and sets an upper limit on the chopper frequency. The current comparator can measure coil current during phases when the current flows through the sense resistor, but not during the slow decay phase, so the slow decay phase is terminated by a timer. The on phase is terminated by the comparator when the current through the coil reaches the target current. The fast decay phase may be terminated by either the comparator or another timer.

When the coil current is switched, spikes at the sense resistors occur due to charging and discharging parasitic capacitances. During this time, typically one or two microseconds, the current cannot be measured. Blanking is the time when the input to the comparator is masked to block these spikes.

The spreadCycle chopper mode cycles through four phases: on, slow decay, fast decay, and a second slow decay.

The chopper frequency is an important parameter for a chopped motor driver. A too low frequency might generate audible noise. A higher frequency reduces current ripple in the motor, but with a too high frequency magnetic losses may rise. Also power dissipation in the driver rises with increasing frequency due to the increased influence of switching slopes causing dynamic dissipation. Therefore, a compromise needs to be found. Most motors are optimally working in a frequency range of 16 kHz to 30 kHz. The chopper frequency is influenced by a number of parameter settings as well as by the motor inductivity and supply voltage.

Hint
A chopper frequency in the range of 16 kHz to 30 kHz gives a good result for most motors when using spreadCycle. A higher frequency leads to increased switching losses.

7.1 spreadCycle Settings

The spreadCycle (patented) chopper algorithm is a precise and simple to use chopper mode which automatically determines the optimum length for the fast-decay phase. The spreadCycle will provide superior microstepping quality even with default settings. Several parameters are available to optimize the chopper to the application.

Each chopper cycle is comprised of an on phase, a slow decay phase, a fast decay phase and a second slow decay phase (see Figure 7.3). The two slow decay phases and the two blank times per chopper cycle put an upper limit to the chopper frequency. The slow decay phases typically make up for about 30%-70% of the chopper cycle in standstill and are important for low motor and driver power dissipation.

Calculation of a starting value for the slow decay time *TOFF*:

EXAMPLE:

Target Chopper frequency: 25kHz.
Assumption: Two slow decay cycles make up for 50% of overall chopper cycle time

$$t_{OFF} = \frac{1}{25kHz} * \frac{50}{100} * \frac{1}{2} = 10\mu s$$

For the *TOFF* setting this means:

$$TOFF = (t_{OFF} * f_{CLK} - 12)/32$$

With 12 MHz clock this gives a setting of TOFF=3.4, i.e. 3 or 4.
With 16 MHz clock this gives a setting of TOFF=4.6, i.e. 4 or 5.

The hysteresis start setting forces the driver to introduce a minimum amount of current ripple into the motor coils. The current ripple must be higher than the current ripple which is caused by resistive losses in the motor in order to give best microstepping results. This will allow the chopper to precisely regulate the current both for rising and for falling target current. The time required to introduce the current ripple into the motor coil also reduces the chopper frequency. Therefore, a higher hysteresis setting will lead to a lower chopper frequency. The motor inductance limits the ability of the chopper to follow a changing motor current. Further the duration of the on phase and the fast decay must be longer than the blanking time, because the current comparator is disabled during blanking.

It is easiest to find the best setting by starting from a low hysteresis setting (e.g. *HSTRT*=0, *HEND*=0) and increasing *HSTRT*, until the motor runs smoothly at low velocity settings. This can best be checked when measuring the motor current either with a current probe or by probing the sense resistor voltages (see Figure 7.2). Checking the sine wave shape near zero transition will show a small ledge between both half waves in case the hysteresis setting is too small. At medium velocities (i.e. 100 to 400 fullsteps per second), a too low hysteresis setting will lead to increased humming and vibration of the motor.

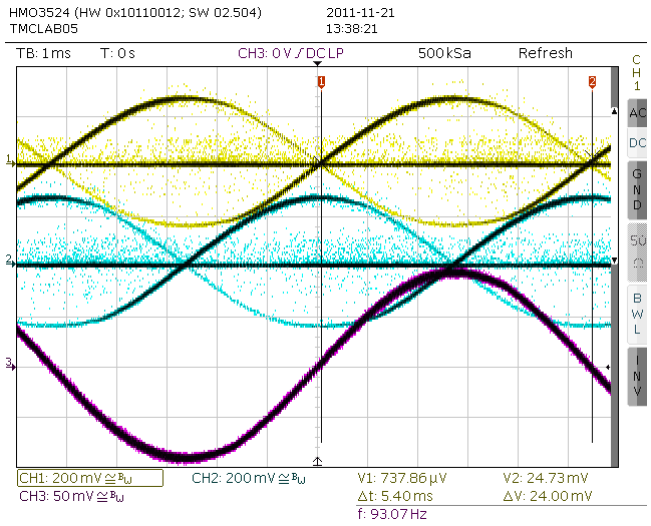


Figure 7.2 No ledges in current wave with sufficient hysteresis (magenta: current A, yellow & blue: sense resistor voltages A and B)

A too high hysteresis setting will lead to reduced chopper frequency and increased chopper noise but will not yield any benefit for the wave shape.

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 14.
For detail procedure see Application Note AN001 - *Parameterization of spreadCycle*

As experiments show, the setting is quite independent of the motor, because higher current motors typically also have a lower coil resistance. Therefore choosing a low to medium default value for the hysteresis (for example, effective hysteresis = 4) normally fits most applications. The setting can be optimized by experimenting with the motor: A too low setting will result in reduced microstep accuracy, while a too high setting will lead to more chopper noise and motor power dissipation. When measuring the sense resistor voltage in motor standstill at a medium coil current with an oscilloscope, a too low setting shows a fast decay phase not longer than the blanking time. When the fast decay time becomes slightly longer than the blanking time, the setting is optimum. You can reduce the off-time setting, if this is hard to reach.

The hysteresis principle could in some cases lead to the chopper frequency becoming too low, e.g. when the coil resistance is high when compared to the supply voltage. This is avoided by splitting the hysteresis setting into a start setting (*HSTRT+HEND*) and an end setting (*HEND*). An automatic hysteresis decrementer (HDEC) interpolates between both settings, by decrementing the hysteresis value stepwise each 16 system clocks. At the beginning of each chopper cycle, the hysteresis begins with a value which is the sum of the start and the end values (*HSTRT+HEND*), and decrements during the cycle, until either the chopper cycle ends or the hysteresis end value (*HEND*) is reached. This way, the chopper frequency is stabilized at high amplitudes and low supply voltage situations, if the frequency gets too low. This avoids the frequency reaching the audible range.

Hint
Highest motor velocities sometimes benefit from setting *TOFF* to 1 or 2 and a short *TBL* of 1 or 0.

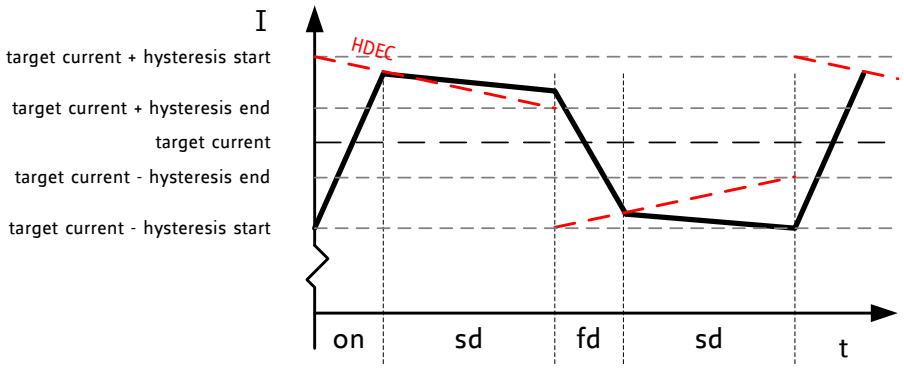


Figure 7.3 spreadCycle chopper scheme showing coil current during a chopper cycle

These parameters control spreadCycle mode:

Parameter	Description	Setting	Comment
TOFF	Sets the slow decay time (*off time*). This setting also limits the maximum chopper frequency. For operation with stealthChop, this parameter is not used, but it is required to enable the motor. In case of operation with stealthChop only, any setting is OK. Setting this parameter to zero completely disables all driver transistors and the motor can free-wheel.	0	chopper off
		1…15	off time setting N_{CLK}= 12 + 32*TOFF (1 will work with minimum blank time of 24 clocks)
TBL	Comparator *blank time*. This time needs to safely cover the switching event and the duration of the ringing on the sense resistor. For most applications, a setting of 1 or 2 is good. For highly capacitive loads, a setting of 2 or 3 will be required.	0	16 t_{CLK}
		1	24 t_{CLK}
		2	32 t_{CLK}
		3	40 t_{CLK}
HSTRT	*Hysteresis start* setting. This value is an offset from the hysteresis end value *HEND*.	0…7	*HSTRT*=1…8 This value adds to HEND.
HEND	*Hysteresis end* setting. Sets the hysteresis end value after a number of decrements. The sum *HSTRT+HEND* must be ≤16. At a current setting of max. 30 (amplitude reduced to 240), the sum is not limited.	0…2	-3…-1: negative HEND
		3	0: zero HEND
		4…15	1…12: positive HEND

Even at HSTRT=0 and HEND=0, the TMC22xx sets a minimum hysteresis via analog circuitry.

EXAMPLE:

A hysteresis of 4 has been chosen. You might decide to not use hysteresis decrement. In this case set:

HEND=6 (sets an effective end value of 6-3=3)
HSTRT=0 (sets minimum hysteresis, i.e. 1: 3+1=4)

In order to take advantage of the variable hysteresis, we can set most of the value to the HSTRT, i.e. 4, and the remaining 1 to hysteresis end. The resulting configuration register values are as follows:

HEND=0 (sets an effective end value of -3)
HSTRT=6 (sets an effective start value of hysteresis end +7: 7-3=4)

8 Selecting Sense Resistors

Set the desired maximum motor current by selecting an appropriate value for the sense resistor. The following table shows the RMS current values which can be reached using standard resistors and motor types fitting without additional motor current scaling.

CHOICE OF R_{SENSE} AND RESULTING MAX. MOTOR CURRENT		
R_{SENSE} [Ω]	**RMS current [A]** VREF=2.5V (or open), *IRUN*=31, *vsense*=0 (standard)	**Fitting motor type** (examples)
1.00	0.23	
0.82	0.27	
0.75	0.30	300mA motor
0.68	0.33	400mA motor
0.50	0.44	
470m	0.47	500mA motor
390m	0.56	600mA motor
330m	0.66	700mA motor
270m	0.79	800mA motor
220m	0.96	1A motor
180m	1.15	1.2A motor
150m	1.35	
120m	1.64*)	
100m	1.92*)	1.5A motor

*) Value exceeds upper current rating, scaling down required, e.g. by reduced VREF.

Sense resistors should be carefully selected. The full motor current flows through the sense resistors. Due to chopper operation the sense resistors see pulsed current from the MOSFET bridges. Therefore, a low-inductance type such as film or composition resistors is required to prevent voltage spikes causing ringing on the sense voltage inputs leading to unstable measurement results. Also, a low-inductance, low-resistance PCB layout is essential. Any common GND path for the two sense resistors must be avoided, because this would lead to coupling between the two current sense signals. A massive ground plane is best. Please also refer to layout considerations in chapter 19.

The sense resistor needs to be able to conduct the peak motor coil current in motor standstill conditions, unless standby power is reduced. Under normal conditions, the sense resistor conducts less than the coil RMS current, because no current flows through the sense resistor during the slow decay phases. A 0.5W type is sufficient for most applications up to 1.2A RMS current.

Attention
Be sure to use a symmetrical sense resistor layout and short and straight sense resistor traces of identical length. Well matching sense resistors ensure best performance.
A compact layout with massive ground plane is best to avoid parasitic resistance effects.

9 Motor Current Control

The basic motor current is set by the resistance of the sense resistors. Several possibilities allow scaling down motor current, e.g. to adapt for different motors, or to reduce motor current in standstill or low load situations.

METHODS FOR SCALING MOTOR CURRENT			
Method	**Parameters**	**Range**	**Primary Use**
Pin VREF voltage (chapter 9.1)	VREF input scales *IRUN* and *IHOLD*. Can be disabled by *GCONF.i_scale_analog*	2.5V: 100% … 0.5V: 20% >2.5V or open: 100% <0.5V: not recommended	- Fine tuning of motor current to fit the motor type - Manual tuning via poti - Delayed or soft power-up - Standstill current reduction (preferred only with spreadCycle)
Pin ENN	Disable / enable driver stage	0: Motor enable 1: Motor disable	- Disable motor to allow freewheeling
Pin PDN_UART	Disable / enable standstill current reduction to *IHOLD*	0: Standstill current reduction enabled. 1: Disable	- Enable current reduction to reduce heat up in stand still
OTP memory	*OTP_IHOLD*, *OTP_IHOLDDELAY*	9% to 78% standby current. Reduction in about 300ms to 2.5s	- Program current reduction to fit application for highest efficiency and lowest heat up
OTP memory	*otp_internalRsense*	0: Use sense resistors 1: Internal resistors	- Save two sense resistors on BOM, set current by single inexpensive 0603 resistor.
UART interface	*IHOLD_IRUN* *TPOWERDOWN* *OTP*	*IRUN*, *IHOLD*: 1/32 to 32/32 of full scale current.	- Fine programming of run and hold (stand still) current - Change *IRUN* for situation specific motor current - Set OTP options
UART interface	*CHOPCONF.vsense* flag	0: Normal, most robust 1: Reduced voltage level	- Set *vsense* for half power dissipation in sense resistor to use smaller 0.25W resistors.

Select the sense resistor to deliver enough current for the motor at full current scale (VREF=2.5V). This is the default current scaling (*IRUN* = 31).

STANDALONE MODE RMS RUN CURRENT CALCULATION:

$$I_{RMS} = \frac{325mV}{R_{SENSE} + 20m\Omega} * \frac{1}{\sqrt{2}} * \frac{V_{VREF}}{2.5V}$$

IRUN and *IHOLD* allow for scaling of the actual current scale (*CS*) from 1/32 to 32/32 when using UART interface, or via automatic standstill current reduction:

RMS CURRENT CALCULATION WITH UART CONTROL OPTIONS OR HOLD CURRENT SETTING:

$$I_{RMS} = \frac{CS + 1}{32} * \frac{V_{FS}}{R_{SENSE} + 20m\Omega} * \frac{1}{\sqrt{2}}$$

CS is the current scale setting as set by the *IHOLD* and *IRUN*.
V_{FS} is the full scale voltage as determined by *vsense* control bit (please refer to electrical characteristics, V_{SRTL} and V_{SRTH}). Default is 325mV.

With analog scaling of V_{FS} (*I_scale_analog*=1, default), the resulting voltage V_{FS}' is calculated by:

$$V'_{FS} = V_{FS} * \frac{V_{VREF}}{2.5V}$$

with V_{VREF} the voltage on pin VREF in the range 0V to $V_{5VOUT}/2$

Hint
For best precision of current setting, measure and fine tune the current in the application.

PARAMETERS FOR MOTOR CURRENT CONTROL			
Parameter	**Description**	**Setting**	**Comment**
IRUN	Current scale when motor is running. Scales coil current values as taken from the internal sine wave table. For high precision motor operation, work with a current scaling factor in the range 16 to 31, because scaling down the current values reduces the effective microstep resolution by making microsteps coarser.	0 … 31	scaling factor 1/32, 2/32, … 32/32 *IRUN* is full scale (setting 31) in standalone mode.
IHOLD	Identical to *IRUN*, but for motor in stand still.		
IHOLD DELAY	Allows smooth current reduction from run current to hold current. *IHOLDDELAY* controls the number of clock cycles for motor power down after *TPOWERDOWN* in increments of 2^18 clocks: 0=instant power down, 1..15: Current reduction delay per current step in multiple of 2^18 clocks. *Example:* When using *IRUN*=31 and *IHOLD*=16, 15 current steps are required for hold current reduction. A *IHOLDDELAY* setting of 4 thus results in a power down time of 4*15*2^18 clock cycles, i.e. roughly one second at 16MHz clock frequency.	0	instant *IHOLD*
		1 … 15	$1*2^{18}$ … $15*2^{18}$ clocks per current decrement
TPOWER DOWN	Sets the delay time from stand still (*stst*) detection to motor current power down. Time range is about 0 to 5.6 seconds.	0 … 255	*0…((2^8)-1)* * 2^18 t_{CLK} A minimum setting of 2 is required to allow automatic tuning of *PWM_OFFS_AUTO*
vsense	Allows control of the sense resistor *voltage range* for full scale current. The low voltage range allows a reduction of sense resistor power dissipation.	0	V_{FS} = 0.32 V
		1	V_{FS} = 0.18 V

9.1 Analog Current Scaling VREF

When a high flexibility of the output current scaling is desired, the analog input of the driver can be used for current control, rather than choosing a different set of sense resistors or scaling down the run current via the interface using *IRUN* or *IHOLD* parameters. This way, a simple voltage divider adapts a board to different motors.

VREF SCALES THE MOTOR CURRENT

The TMC22xx provides an internal reference voltage for current control, directly derived from the 5VOUT supply output. Alternatively, an external reference voltage can be used. This reference voltage becomes scaled down for the chopper comparators. The chopper comparators compare the voltages on BRA and BRB to the scaled reference voltage for current regulation. When *I_scale_analog* in *GCONF* is enabled (default), the external voltage on VREF is amplified and filtered and becomes used as reference voltage. A voltage of 2.5V (or any voltage between 2.5V and 5V) gives the same current

scaling as the internal reference voltage. A voltage between 0V and 2.5V linearly scales the current between 0 and the current scaling defined by the sense resistor setting. It is not advised to work with reference voltages below about 0.5V to 1V for full scale, because relative analog noise caused by digital circuitry and power supply ripple has an increased impact on the chopper precision at low VREF voltages. For best precision, choose the sense resistors in a way that the desired maximum current is reached with VREF in the range 2V to 2.4V. Be sure to optimize the chopper settings for the normal run current of the motor.

DRIVING VREF

The easiest way to provide a voltage to VREF is to use a voltage divider from a stable supply voltage or a microcontroller's DAC output. A PWM signal also allows current control. The PWM becomes transformed to an analog voltage using an additional R/C low-pass at the VREF pin. The PWM duty cycle controls the analog voltage. Choose the R and C values to form a low pass with a corner frequency of several milliseconds while using PWM frequencies well above 10 kHz. VREF additionally provides an internal low-pass filter with 3.5kHz bandwidth.

Hint
Using a low reference voltage (e.g. below 1V), for adaptation of a high current driver to a low current motor will lead to reduced analog performance. Adapt the sense resistors to fit the desired motor current for the best result.

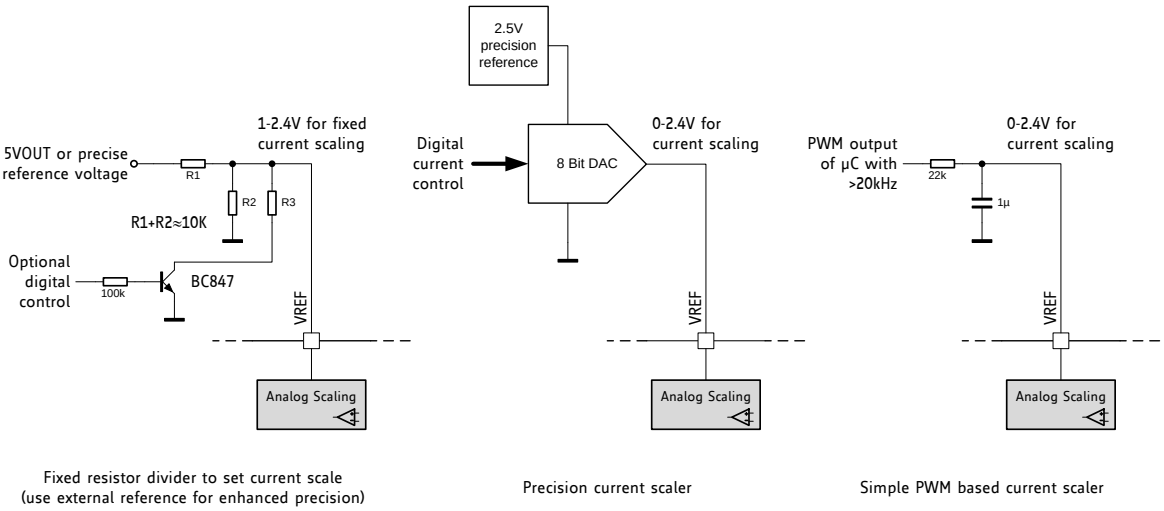


Figure 9.1 Scaling the motor current using the analog input

10 Internal Sense Resistors

The TMC22xx provides the option to eliminate external sense resistors. In this mode the external sense resistors become omitted (shorted) and the internal on-resistance of the power MOSFETs is used for current measurement (see chapter 3.2). As MOSFETs are both, temperature dependent and subject to production stray, a tiny external resistor connected from +5VOUT to VREF provides a precise absolute current reference. This resistor converts the 5V voltage into a reference current. Be sure to directly attach BRA and BRB pins to GND in this mode near the IC package. The mode is enabled by setting *internal_Rsense* in *GCONF* (OTP option).

COMPARING INTERNAL SENSE RESISTORS VS. SENSE RESISTORS		
Item	**Internal Sense Resistors**	**External Sense Resistors**
Ease of use	Need to set OTP parameter first	(+) Default
Cost	(+) Save cost for sense resistors	
Current precision	Slightly reduced	(+) Good
Current Range Recommended	200mA RMS to 1.2A RMS	50mA to 1.4A RMS
Recommended chopper	stealthChop, spreadCycle shows slightly reduced performance at >1A	stealthChop or spreadCycle

While the RDSon based measurements bring benefits concerning cost and size of the driver, it gives slightly less precise coil current regulation when compared to external sense resistors. The internal sense resistors have a certain temperature dependence, which is automatically compensated by the driver IC. However, for high current motors, a temperature gradient between the ICs internal sense resistors and the compensation circuit will lead to an initial current overshoot of some 10% during driver IC heat up. While this phenomenon shows for roughly a second, it might even be beneficial to enable increased torque during initial motor acceleration.

PRINCIPLE OF OPERATION

A reference current into the VREF pin is used as reference for the motor current. In order to realize a certain current, a single resistor (R_{REF}) can be connected between 5VOUT and VREF (pls. refer the table for the choice of the resistor). VREF input resistance is about 1kOhm. The resulting current into VREF is amplified 3000 times. Thus, a current of 0.5mA yields a motor current of 1.5A peak. For calculation of the reference resistor, the internal resistance of VREF needs to be considered additionally.

CHOICE OF R_{REF} FOR OPERATION WITHOUT SENSE RESISTORS		
R_{REF} [Ω]	**Peak current [A]** (*CS*=31, *vsense*=0)	**RMS current [A]** (*CS*=31, *vsense*=0)
6k8	1.92	1.35
7k5	1.76	1.24
8k2	1.63	1.15
9k1	1.49	1.05
10k	1.36	0.96
12k	1.15	0.81
15k	0.94	0.66
18k	0.79	0.55
22k	0.65	0.45
27k	0.60	0.42
33k	0.54	0.38

vsense=1 allows a lower peak current setting of about 55% of the value yielded with *vsense*=0 (as specified by V_{SRTH} / V_{SRTL}).

In RDSon measurement mode, connect the BRA and BRB pins to GND using the shortest possible path (i.e. lowest possible PCB resistance). RDSon based measurement gives best results when combined with stealthChop. When using spreadCycle with RDSon based current measurement, slightly asymmetric current measurement for positive currents (on phase) and negative currents (fast decay phase) may result in chopper noise. This especially occurs at high die temperature and increased motor current.

Note
The absolute current levels achieved with RDSon based current sensing may depend on PCB layout exactly like with external sense resistors, because trace resistance on BR pins will add to the effective sense resistance. Therefore we recommend to measure and calibrate the current setting within the application.

Thumb rule
RDSon based current sensing works best for motors with up to 1A RMS current. The best results are yielded with stealthChop operation in combination with RDSon based current sensing.
For most precise current control and for best results with spreadCycle, it is recommended to use external 1% sense resistors rather than RDSon based current control.

11 STEP/DIR Interface

The STEP and DIR inputs provide a simple, standard interface compatible with many existing motion controllers. The microPlyer step pulse interpolator brings the smooth motor operation of high-resolution microstepping to applications originally designed for coarser stepping.

11.1 Timing

Figure 11.1 shows the timing parameters for the STEP and DIR signals, and the table below gives their specifications. Only rising edges are active. STEP and DIR are sampled and synchronized to the system clock. An internal analog filter removes glitches on the signals, such as those caused by long PCB traces. If the signal source is far from the chip, and especially if the signals are carried on cables, the signals should be filtered or differentially transmitted.

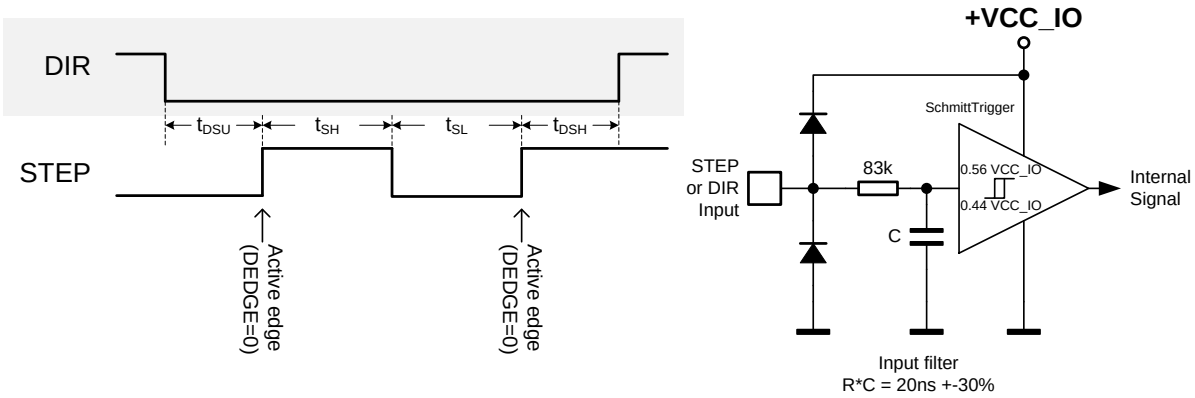


Figure 11.1 STEP and DIR timing, Input pin filter

STEP and DIR interface timing	**AC-Characteristics** clock period is t_{CLK}					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
step frequency (at maximum microstep resolution)	f_{STEP}				½ f_{CLK}	
fullstep frequency	f_{FS}				f_{CLK}/512	
STEP input minimum low time	t_{SL}		max(t_{FILTSD}, t_{CLK}+20)	100		ns
STEP input minimum high time	t_{SH}		max(t_{FILTSD}, t_{CLK}+20)	100		ns
DIR to STEP setup time	t_{DSU}		20			ns
DIR after STEP hold time	t_{DSH}		20			ns
STEP and DIR spike filtering time *)	t_{FILTSD}	rising and falling edge	13	20	30	ns
STEP and DIR sampling relative to rising CLK input	$t_{SDCLKHI}$	before rising edge of CLK input		t_{FILTSD}		ns

*) These values are valid with full input logic level swing, only. Asymmetric logic levels will increase filtering delay t_{FILTSD}, due to an internal input RC filter.

11.2 Changing Resolution

The TMC22xx includes an internal microstep table with 1024 sine wave entries to generate sinusoidal motor coil currents. These 1024 entries correspond to one electrical revolution or four fullsteps. The microstep resolution setting determines the step width taken within the table. Depending on the DIR input, the microstep counter is increased (DIR=0) or decreased (DIR=1) with each STEP pulse by the step width. The microstep resolution determines the increment respectively the decrement. At maximum resolution, the sequencer advances one step for each step pulse. At half resolution, it advances two steps. Increment is up to 256 steps for fullstepping. The sequencer has special provision to allow seamless switching between different microstep rates at any time. When switching to a lower microstep resolution, it calculates the nearest step within the target resolution and reads the current vector at that position. This behavior especially is important for low resolutions like fullstep and halfstep, because any failure in the step sequence would lead to asymmetrical run when comparing a motor running clockwise and counterclockwise.

EXAMPLES:

Fullstep: Cycles through table positions: 128, 384, 640 and 896 (45°, 135°, 225° and 315° electrical position, both coils on at identical current). The coil current in each position corresponds to the RMS-Value (0.71 * amplitude). Step size is 256 (90° electrical)

Half step: The first table position is 64 (22.5° electrical), Step size is 128 (45° steps)

Quarter step: The first table position is 32 (90°/8=11.25° electrical), Step size is 64 (22.5° steps)

This way equidistant steps result and they are identical in both rotation directions. Some older drivers also use zero current (table entry 0, 0°) as well as full current (90°) within the step tables. This kind of stepping is avoided because it provides less torque and has a worse power dissipation in driver and motor.

Step position	table position	current coil A	current coil B
Half step 0	64	38.3%	92.4%
Full step 0	128	70.7%	70.7%
Half step 1	192	92.4%	38.3%
Half step 2	320	92.4%	-38.3%
Full step 1	384	70.7%	-70.7%
Half step 3	448	38.3%	-92.4%
Half step 4	576	-38.3%	-92.4%
Full step 2	640	-70.7%	-70.7%
Half step 5	704	-92.4%	-38.3%
Half step 6	832	-92.4%	38.3%
Full step 3	896	-70.7%	70.7%
Half step 7	960	-38.3%	92.4%

See chapter 3.4 for resolution settings available in stand-alone mode.

11.3 microPlyer Step Interpolator and Stand Still Detection

For each active edge on STEP, microPlyer produces microsteps at 256x resolution, as shown in Figure 11.2. It interpolates the time in between of two step impulses at the step input based on the last step interval. This way, from 2 microsteps (128 microstep to 256 microstep interpolation) up to 256 microsteps (full step input to 256 microsteps) are driven for a single step pulse.

The step rate for the interpolated 2 to 256 microsteps is determined by measuring the time interval of the previous step period and dividing it into up to 256 equal parts. The maximum time between two microsteps corresponds to 2^{20} (roughly one million system clock cycles), for an even distribution of 256 microsteps. At 12 MHz system clock frequency, this results in a minimum step input frequency of roughly 12 Hz for microPlyer operation. A lower step rate causes a standstill event to be detected. At that frequency, microsteps occur at a rate of (system clock frequency)/2^{16} ~ 256 Hz. When a stand still is detected, the driver automatically begins standby current reduction if selected by pin PDN.

Attention
microPlyer only works perfectly with a stable STEP frequency.

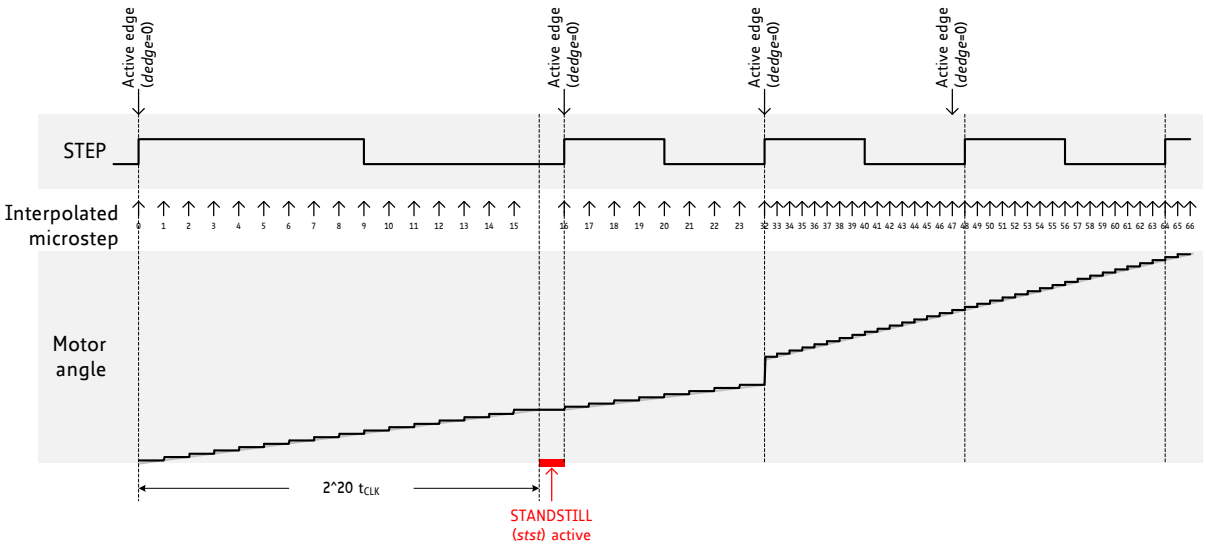


Figure 11.2 microPlyer microstep interpolation with rising STEP frequency (Example: 16 to 256)

In Figure 11.2, the first STEP cycle is long enough to set the *stst* bit standstill. Detection of standstill will enable the standby current reduction. This bit is cleared on the next STEP active edge. Then, the external STEP frequency increases. After one cycle at the higher rate microPlyer adapts the interpolated microstep rate to the higher frequency. During the last cycle at the slower rate, microPlyer did not generate all 16 microsteps, so there is a small jump in motor angle between the first and second cycles at the higher rate.

11.4 Index Output

An active INDEX output signals that the sine curve of motor coil A is at its positive zero transition. This correlates to the zero point of the microstep sequence. Usually, the cosine curve of coil B is at its maximum at the same time. Thus the index signal is active once within each electrical period, and corresponds to a defined position of the motor within a sequence of four fullsteps. The INDEX output this way allows the detection of a certain microstep pattern, and thus helps to detect a position with more precision than a stop switch can do.

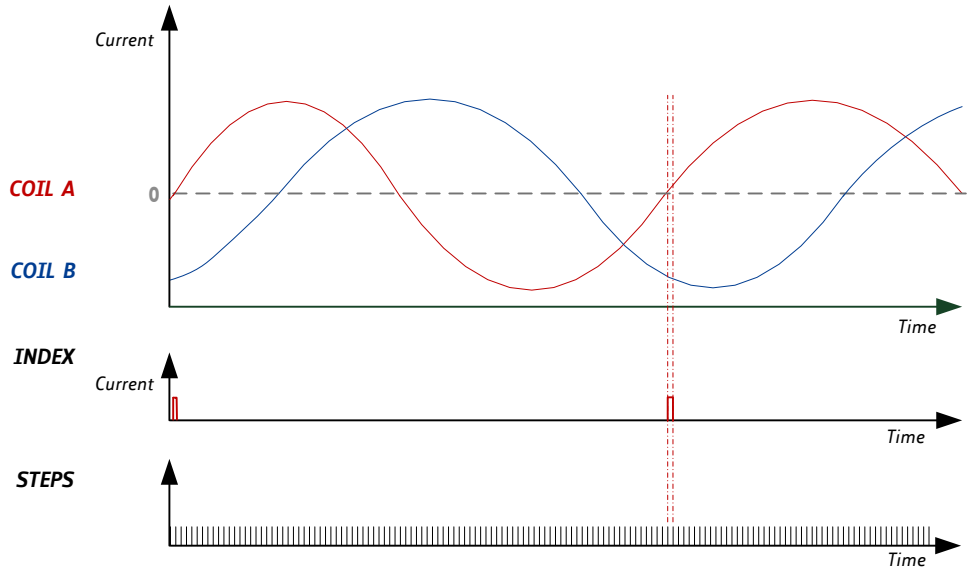


Figure 11.3 Index signal at positive zero transition of the coil A sine curve

Hint
The index output allows precise detection of the microstep position within one electrical wave, i.e. within a range of four fullsteps. With this, homing accuracy and reproducibility can be enhanced to microstep accuracy, even when using an inexpensive home switch.

12 Internal Step Pulse Generator

The TMC22xx family integrates a high resolution step pulse generator, allowing motor motion via the UART interface. However, no velocity ramping is provided. Ramping is not required, if the target motion velocity is smaller than the start & stop frequency of the motor. For higher velocities, ramp up the frequency in small steps to accelerate the motor, and ramp down again to decelerate the motor. Figure 12.1 shows an example motion profile ramping up the motion velocity in discrete steps. Choose the ramp velocity steps considerably smaller than the maximum start velocity of the motor, because motor torque drops at higher velocity, and motor load at higher velocity typically increases.

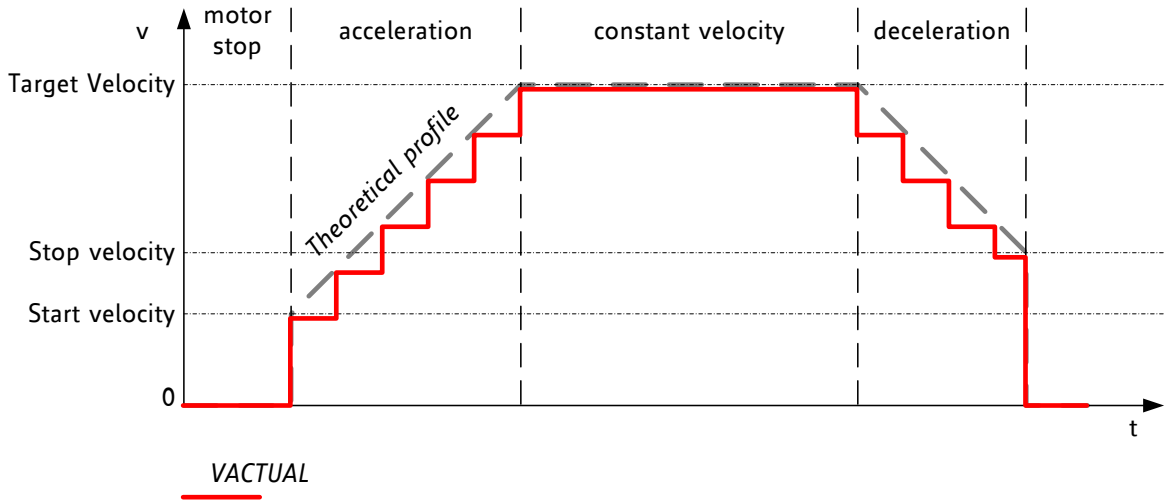


Figure 12.1 Software generated motion profile

PARAMETER VS. UNITS		
Parameter / Symbol	**Unit**	**calculation / description / comment**
f_{CLK}[Hz]	[Hz]	clock frequency of the TMC22xx in [Hz]
µstep velocity v[Hz]	µsteps / s	v[Hz] = *VACTUAL*[22xx] * (f_{CLK}[Hz]/2 / 2^23) With internal oscillator: v[Hz] = *VACTUAL*[22xx] * 0.715Hz
USC microstep count	counts	microstep resolution in number of microsteps (i.e. the number of microsteps between two fullsteps – normally 256)
rotations per second v[rps]	rotations / s	v[rps] = v[Hz] / USC / FSC FSC: motor fullsteps per rotation, e.g. 200
TSTEP, *TPWMTHRS*	-	*TSTEP* = f_{CLK} / f_{STEP} The time reference for velocity threshold is referred to the actual microstep frequency of the step input respectively velocity v[Hz].
VACTUAL	Two's complement signed internal velocity	*VACTUAL*[22xx] = (f_{CLK}[Hz]/2 / 2^23) / v[Hz] With internal oscillator: *VACTUAL*[22xx] = 0.715Hz / v[Hz]

Hint
To monitor internal step pulse execution, program the INDEX output to provide step pulses (*GCONF.index_step*). It toggles upon each step. Use a timer input on your CPU to count pulses. Alternatively, regularly poll *MSCNT* to grasp steps done in the previous polling interval. It wraps around from 1023 to 0.

13 Driver Diagnostic Flags

The TMC22xx drivers supply a complete set of diagnostic and protection capabilities, like short to GND protection, short to VS protection and undervoltage detection. A detection of an open load condition allows testing if a motor coil connection is interrupted. See the *DRV_STATUS* table for details.

13.1 Temperature Measurement

The driver integrates a four level temperature sensor (pre-warning and thermal shutdown) for diagnostics and for protection of the IC against excess heat. The thresholds can be adapted by UART or OTP programming. Heat is mainly generated by the motor driver stages, and, at increased voltage, by the internal voltage regulator. Most critical situations, where the driver MOSFETs could be overheated, are avoided by the short to GND protection. For many applications, the overtemperature pre-warning will indicate an abnormal operation situation and can be used to initiate user warning or power reduction measures like motor current reduction. The thermal shutdown is just an emergency measure and temperature rising to the shutdown level should be prevented by design.

TEMPERATURE THRESHOLDS	
Overtemperature Setting	**Comment**
143°C (OTPW: 120°C)	Default setting. This setting is safest to switch off the driver stage before the IC can be destroyed by overheating. On a large PCB, the power MOSFETs reach roughly 150°C peak temperature when the temperature detector is triggered with this setting. This is a trip typical point for overtemperature shut down. The overtemperature pre-warning threshold of 120°C gives lots of headroom to react to high driver temperature, e.g. by reducing motor current.
150°C (OTPW: 120°C or 143°C)	Optional setting (OTP or UART). For small PCBs with high thermal resistance between PCB and environment, this setting provides some additional headroom. The small PCB shows less temperature difference between the MOSFETs and the sensor.
157°C (OTPW: 143°C)	Optional setting (UART). For applications, where a stop of the motor cannot be tolerated, this setting provides highest headroom, e.g. at high environment temperature ratings. Using the 143°C overtemperature pre-warning to reduce motor current ensures that the motor is not switched off by the thermal threshold.

Attention
Overtemperature protection cannot in all cases avoid thermal destruction of the IC. In case the rated motor current is exceed, e.g. by operating a motor in stealthChop with wrong parameters, or with automatic tuning parameters not fitting the operating conditions, excess heat generation can quickly heat up the driver before the overtemperature sensor can react. This is due to a delay in heat conduction over the IC die.

After triggering the overtemperature sensor (*ot* flag), the driver remains switched off until the system temperature falls below the pre-warning level (*otpw*) to avoid continuous heating to the shutdown level.

13.2 Short Protection

The TMC22xx power stages are protected against a short circuit condition by an additional measurement of the current flowing through each of the power stage MOSFETs. This is important, as most short circuit conditions result from a motor cable insulation defect, e.g. when touching the conducting parts connected to the system ground. The short detection is protected against spurious triggering, e.g. by ESD discharges, by retrying three times before switching off the motor.

Once a short condition is safely detected, the corresponding driver bridge (A or B) becomes switched off, and the *s2ga* or *s2gb* flag, respectively *s2vsa* or *s2vsb* becomes set. In order to restart the motor, disable and re-enable the driver. Note, that short protection cannot protect the system and the power stages for all possible short events, as a short event is rather undefined and a complex network of external components may be involved. Therefore, short circuits should basically be avoided.

13.3 Open Load Diagnostics

UART

Interrupted cables are a common cause for systems failing, e.g. when connectors are not firmly plugged. The TMC22xx detects open load conditions by checking, if it can reach the desired motor coil current. This way, also undervoltage conditions, high motor velocity settings or short and overtemperature conditions may cause triggering of the open load flag, and inform the user, that motor torque may suffer. In motor stand still, open load cannot always be measured, as the coils might eventually have zero current.

Open load detection is provided for system debugging.
In order to safely detect an interrupted coil connection, read out the open load flags at low or nominal motor velocity operation, only. If possible, use spreadCycle for testing, as it provides the most accurate test. However, the *ola* and *olb* flags have just informative character and do not cause any action of the driver.

13.4 Diagnostic Output

The diagnostic output DIAG and the index output INDEX provide important status information. An active DIAG output shows that the driver cannot work normally. The INDEX output signals the microstep counter zero position, to allow referencing (homing) a drive to a certain current pattern. The function set of the INDEX output can be modified by UART. Figure 13.1 shows the available signals and control bits.

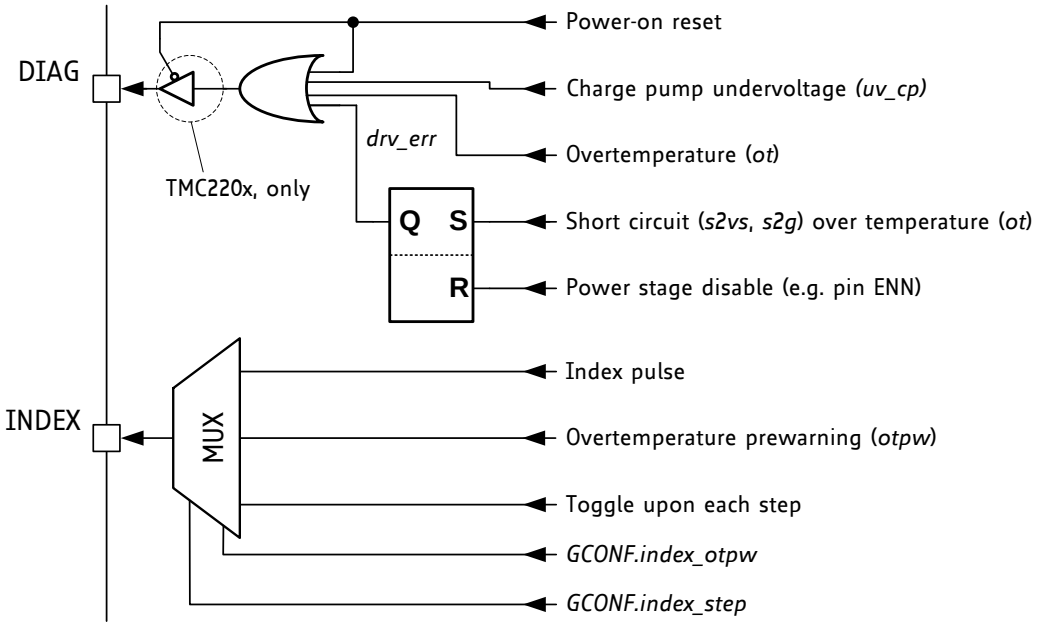


Figure 13.1 DIAG and INDEX outputs

14 Quick Configuration Guide

This guide is meant as a practical tool to come to a first configuration. Do a minimum set of measurements and decisions for tuning the driver to determine UART settings or OTP parameters. The flow-charts concentrate on the basic function set to make a motor run smoothly. Once the motor runs, you may decide to explore additional features, e.g. freewheeling in more detail. A current probe on one motor coil is a good aid to find the best settings, but it is not a must.

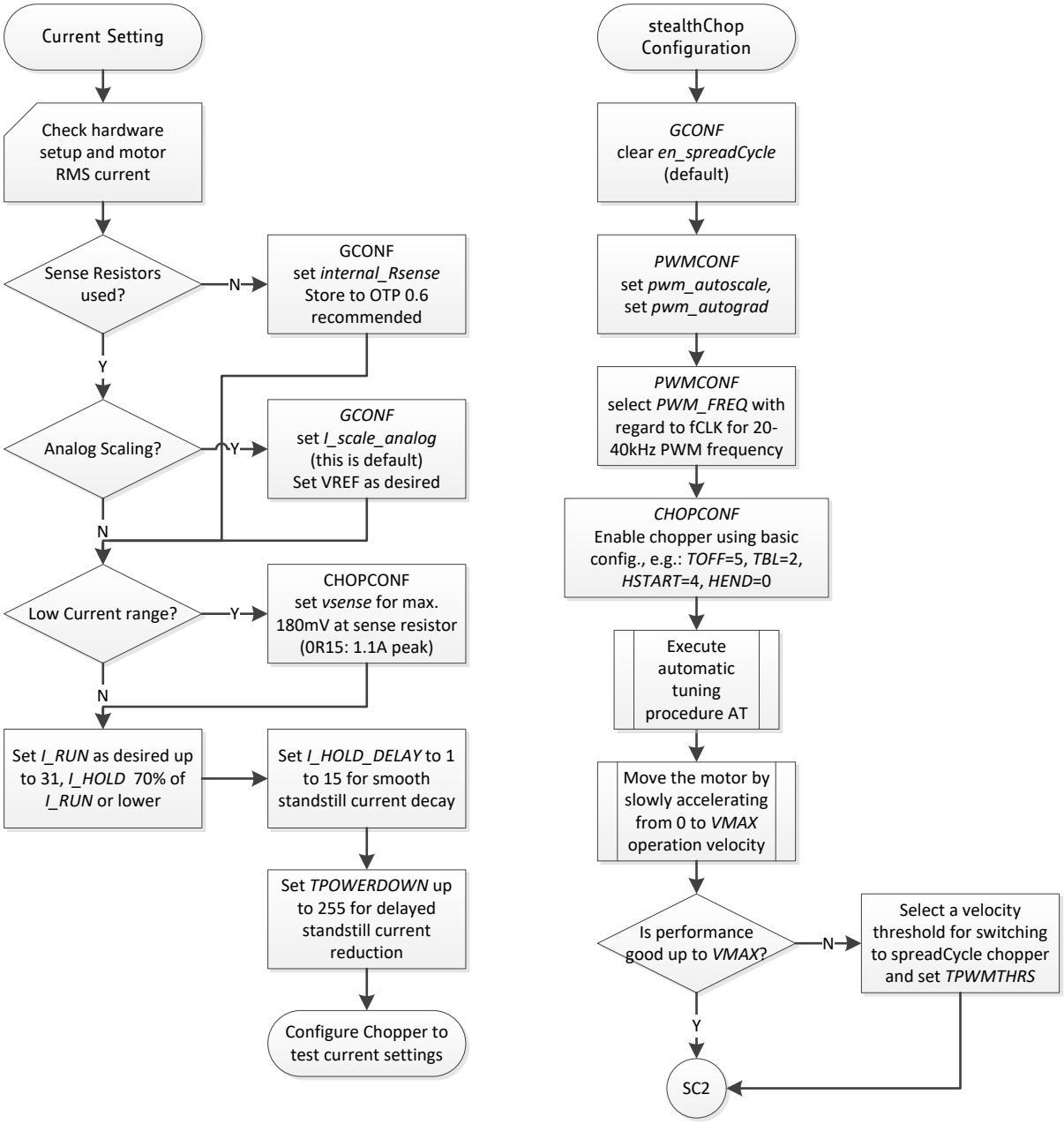


Figure 14.1 Current Setting and first steps with stealthChop

Hint
Use the evaluation board to explore settings and to generate the required configuration datagrams.

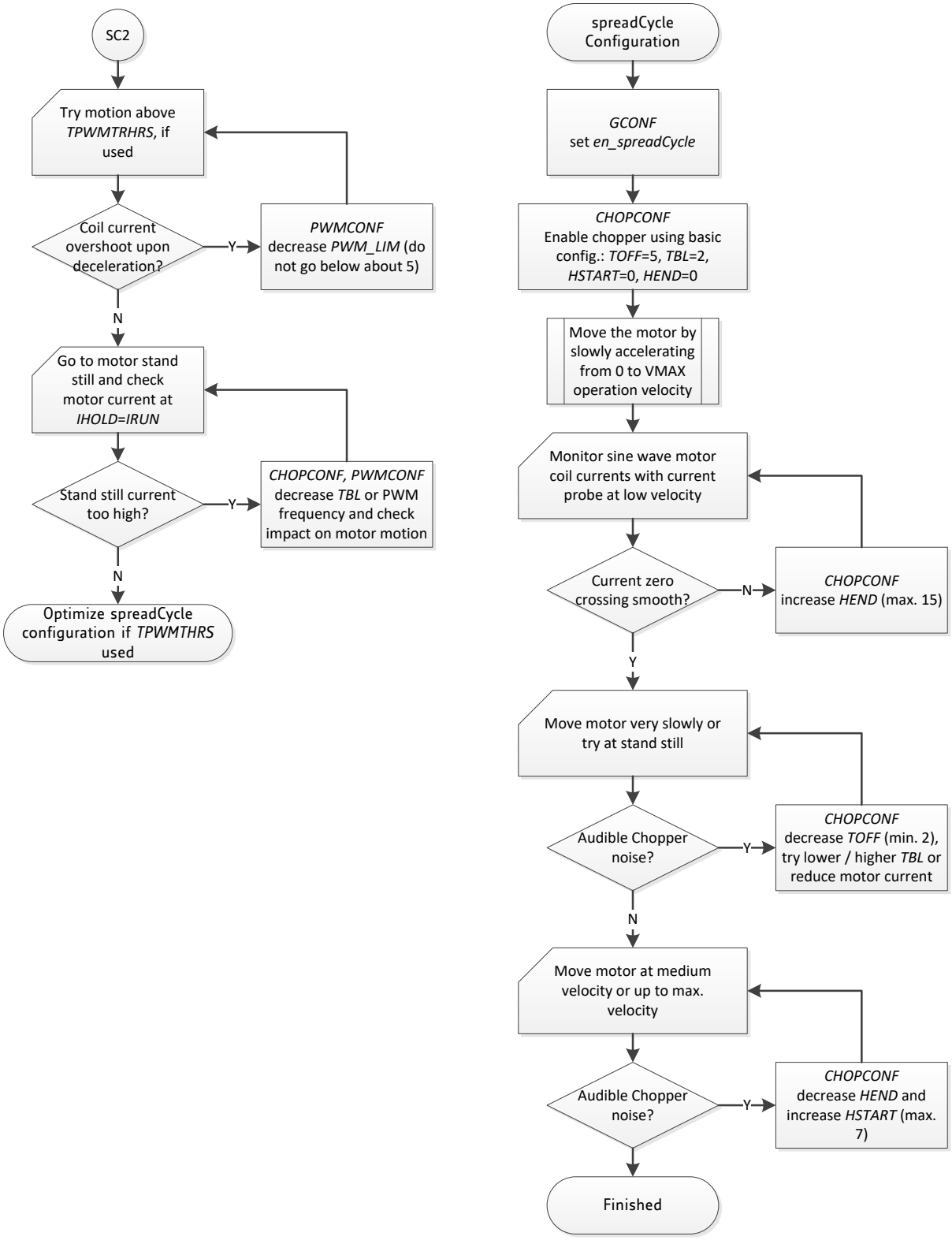


Figure 14.2 Tuning stealthChop and spreadCycle

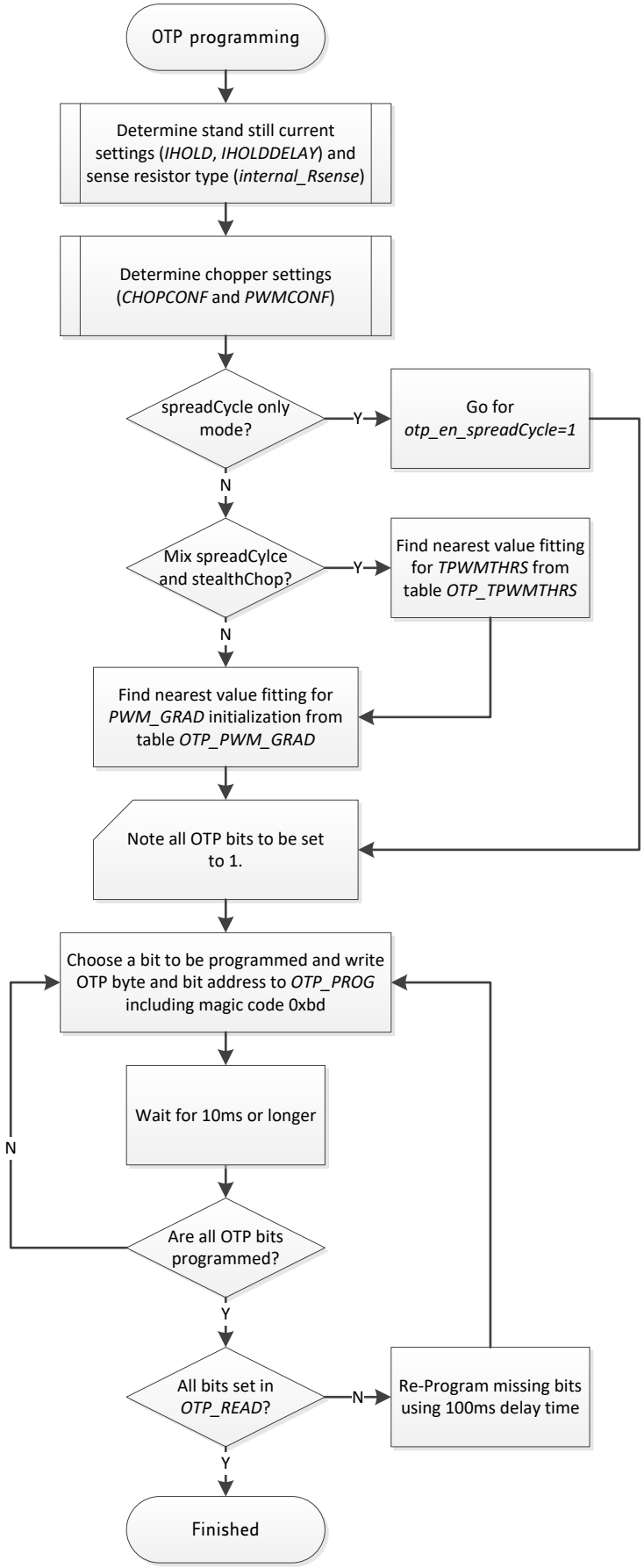


Figure 14.3 OTP programming

15 External Reset

The chip is loaded with default values during power on via its internal power-on reset. Some of the registers are initialized from the OTP at power up. In order to reset the chip to power on defaults, any of the supply voltages monitored by internal reset circuitry (VS, +5VOUT or VCC_IO) must be cycled. As +5VOUT is the output of the internal voltage regulator, it cannot be cycled via an external source except by cycling VS. It is easiest and safest to cycle VCC_IO in order to completely reset the chip. Also, current consumed from VCC_IO is low and therefore it has simple driving requirements. Due to the input protection diodes not allowing the digital inputs to rise above VCC_IO level, all inputs must be driven low during this reset operation. When this is not possible, an input protection resistor may be used to limit current flowing into the related inputs.

16 Clock Oscillator and Input

The clock is the timing reference for all functions: the chopper frequency, the blank time, the standstill power down timing, and the internal step pulse generator etc. The on-chip clock oscillator is calibrated in order to provide timing precise enough for most operation cases.

USING THE INTERNAL CLOCK

Directly tie the CLK input to GND near to the IC if the internal clock oscillator is to be used. The internal clock frequency is factory-trimmed to 12MHz by OTP programming.

USING AN EXTERNAL CLOCK

When an external clock is available, any frequency of 8 to 18MHz can be used to clock the TMC22xx. The duty cycle of the clock signal is uncritical, as long as minimum high or low input time for the pin is satisfied (refer to electrical characteristics). Make sure, that the clock source supplies clean CMOS output logic levels and steep slopes when using a high clock frequency. The external clock input is enabled with the first positive polarity seen on the CLK input. Modifying the clock frequency is an easy way to adapt the stealthChop chopper frequency or to synchronize multiple drivers. Working with a very low clock frequency down to 4 MHz can help reducing power consumption and electromagnetic emissions, but it will sacrifice some performance.

Use an external clock source to synchronize multiple drivers, or to get precise motor operation with the internal pulse generator for motion. The external clock frequency selection also allows modifying the power down timing and the chopper frequency.

Hint
Switching off the external clock frequency would stop the chopper and could lead to an overcurrent condition. Therefore, TMC22xx has an internal timeout of 32 internal clocks. In case the external clock fails, it switches back to internal clock.

17 Absolute Maximum Ratings

The maximum ratings may not be exceeded under any circumstances. Operating the circuit at or near more than one maximum rating at a time for extended periods shall be avoided by application design.

Parameter	Symbol	Min	Max	Unit
Supply voltage operating with inductive load	V_{VS}	-0.5	39	V
Supply and bridge voltage max. *)	V_{VMAX}		40	V
I/O supply voltage	V_{VIO}	-0.5	5.5	V
digital supply voltage (when using external supply)	V_{5VOUT}	-0.5	5.5	V
Logic input voltage	V_I	-0.5	V_{VIO}+0.5	V
VREF input voltage (Do not exceed both, VCC_IO and 5VOUT by more than 10%, as this enables a test mode)	V_{VREF}	-0.5	6	V
Maximum current to / from digital pins and analog low voltage I/Os	I_{IO}		+/-10	mA
5V regulator output current (internal plus external load)	I_{5VOUT}		25	mA
5V regulator continuous power dissipation (V_{VM}-5V) * I_{5VOUT}	P_{5VOUT}		0.5	W
Power bridge repetitive output current	I_{Ox}		2.5	A
Junction temperature	T_J	-50	150	°C
Storage temperature	T_{STG}	-55	150	°C
ESD-Protection for interface pins (Human body model, HBM)	V_{ESDAP}		4	kV
ESD-Protection for handling (Human body model, HBM)	V_{ESD}		1	kV

*) Stray inductivity of GND and VS connections will lead to ringing of the supply voltage when driving an inductive load. This ringing results from the fast switching slopes of the driver outputs in combination with reverse recovery of the body diodes of the output driver MOSFETs. Even small trace inductivities as well as stray inductivity of sense resistors can easily generate a few volts of ringing leading to temporary voltage overshoot. This should be considered when working near the maximum voltage.

18 Electrical Characteristics

18.1 Operational Range

Parameter	Symbol	Min	Max	Unit
Junction temperature	T_J	-40	125	°C
Supply voltage (using internal +5V regulator)	V_{VS}	5.5	36	V
Supply voltage (using internal +5V regulator) for OTP programming	V_{VS}	6	36	V
Supply voltage (internal +5V regulator bridged: V_{5VOUT}=V_{VS})	V_{VS}	4.7	5.4	V
I/O supply voltage	V_{VIO}	3.00	5.25	V
VCC voltage when using optional external source (supplies digital logic and charge pump)	V_{VCC}	4.6	5.25	V
RMS motor coil current per coil (value for design guideline)	I_{RMS}		1.2	A
RMS motor coil current per coil with duty cycle limit, e.g. 1s on, 1s standby (value for design guideline)	I_{RMS}		1.4	A
Peak output current per motor coil output (sine wave peak) using external or internal current sensing	I_{Ox}		2.0	A

18.2 DC and Timing Characteristics

DC characteristics contain the spread of values guaranteed within the specified supply voltage range unless otherwise specified. Typical values represent the average value of all parts measured at +25°C. Temperature variation also causes stray to some values. A device with typical values will not leave Min/Max range within the full temperature range.

Power supply current	**DC-Characteristics** V_{VS} = V_{VSA} = 24.0V					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Total supply current, driver disabled I_{VS}	I_S	f_{CLK}=12MHz		7	10	mA
Total supply current, operating, I_{VS}	I_S	f_{CLK}=12MHz, 35kHz chopper, no load		7.5		mA
Supply current, driver disabled, dependency on CLK frequency	I_{VS}	f_{CLK} variable, additional to I_{VS0}		0.3		mA/MHz
Internal current consumption from 5V supply on VCC pin	I_{VCC}	f_{CLK}=12MHz, 35kHz chopper		4.5		mA
IO supply current (typ. at 5V)	I_{VIO}	no load on outputs, inputs at V_{IO} or GND Excludes pullup / pull-down resistors		15	30	µA

Motor driver section	**DC- and Timing-Characteristics** V_{VS} = 24.0V					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
RDS_{ON} lowside MOSFET	R_{ONL}	measure at 100mA, 25°C, static state		0.28	0.38	Ω
RDS_{ON} highside MOSFET	R_{ONH}	measure at 100mA, 25°C, static state		0.29	0.39	Ω
slope, MOSFET turning on	t_{SLPON}	measured at 700mA load current (resistive load)	40	80	160	ns
slope, MOSFET turning off	t_{SLPOFF}	measured at 700mA load current (resistive load)	40	80	160	ns
Current sourcing, driver off	I_{OIDLE}	O_{XX} pulled to GND	120	180	250	µA

Charge pump	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Charge pump output voltage	V_{VCP}-V_{VS}	operating, typical f_{chop}<40kHz	4.0	V_{VCC} - 0.3	V_{VCC}	V
Charge pump voltage threshold for undervoltage detection	V_{VCP}-V_{VS}	using internal 5V regulator voltage	3.3	3.6	4.0	V
Charge pump frequency	f_{CP}			1/16 f_{CLKOSC}		

Linear regulator	**DC-Characteristics** V_{VS} = V_{VSA} = 24.0V					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Output voltage	V_{5VOUT}	I_{5VOUT} = 0mA T_J = 25°C	4.80	5.0	5.25	V
Output resistance	R_{5VOUT}	Static load		1		Ω
Deviation of output voltage over the full temperature range	$V_{5VOUT(DEV)}$	I_{5VOUT} = 5mA T_J = full range		+/-30	+/-100	mV
Deviation of output voltage over the full supply voltage range	$V_{5VOUT(DEV)}$	I_{5VOUT} = 5mA V_{VS} = variable		+/-15	+/-30	mV / 10V

Clock oscillator and input	**Timing-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Clock oscillator frequency (factory calibrated)	f_{CLKOSC}	t_J=-50°C		11.7		MHz
	f_{CLKOSC}	t_J= 25°C	11.5	12.0	12.5	MHz
	f_{CLKOSC}	t_J=150°C		12.1		MHz
External clock frequency (operating)	f_{CLK}		4	10-16	18	MHz
External clock high / low level time	t_{CLK}	CLK driven to 0.1 V_{VIO} / 0.9 V_{VIO}	10			ns
External clock timeout detection in cycles of internal f_{CLKOSC}	$x_{timeout}$	External clock stuck at low or high	32		48	cycles f_{CLKOSC}

Detector levels	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
V_{VS} undervoltage threshold for RESET	V_{UV_VS}	V_{VS} rising	3.5	4.2	4.6	V
V_{5VOUT} undervoltage threshold for RESET	V_{UV_5VOUT}	V_{5VOUT} rising		3.5		V
V_{VCC_IO} undervoltage threshold for RESET	V_{UV_VIO}	V_{VCC_IO} rising (delay typ. 10µs)	2.1	2.55	3.0	V
V_{VCC_IO} undervoltage detector hysteresis	$V_{UV\_VIOHYST}$			0.3		V
Short to GND detector threshold (V_{VS} - V_{Ox})	V_{OS2G}		2	2.5	3	V
Short to VS detector threshold (V_{Ox})	V_{OS2G}		1.6	2	2.3	V
Short detector delay (high side / low side switch on to short detected)	t_{S2G}	High side output clamped to V_{SP}-3V	0.8	1.3	2	µs
Overtemperature prewarning 120°C	t_{OTPW}	Temperature rising	100	120	140	°C
Overtemperature shutdown or prewarning 143°C (appr. 153°C IC peak temp.)	t_{OT143}	Temperature rising	128	143	163	°C
Overtemperature shutdown 150°C (appr. 160°C IC peak temp.)	t_{OT150}	Temperature rising	135	150	170	°C
Overtemperature shutdown 157°C (appr. 167°C IC peak temp.)	t_{OT157}	Temperature rising	142	157	177	°C
Difference between temperature detector and power stage temperature, slow heat up	t_{OTDIFF}	Power stage causing high temperature, normal 4 Layer PCB		10		°C

Sense resistor voltage levels	**DC-Characteristics** **f_{CLK}=16MHz**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Sense input peak threshold voltage (low sensitivity)	V_{SRTL}	*vsense*=0 *csactual*=31 *CUR_A/B*=248 Hyst.=0; I_{BRxy}=0		325		mV
Sense input peak threshold voltage (high sensitivity)	V_{SRTH}	*vsense*=1 *csactual*=31 *CUR_A/B*=248 Hyst.=0; I_{BRxy}=0		180		mV
Sense input tolerance / motor current full scale tolerance -using internal reference	I_{COIL}	*I_scale_analog*=0, *vsense*=0	-5		+5	%
Sense input tolerance / motor current full scale tolerance -using external reference voltage	I_{COIL}	*I_scale_analog*=1, V_{AIN}=2V, *vsense*=0	-2		+2	%
Internal resistance from pin BRxy to internal sense comparator (additional to sense resistor)	R_{BRxy}			30		mΩ

Digital pins	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Input voltage low level	V_{INLO}		-0.3		0.3 V_{VIO}	V
Input voltage high level	V_{INHI}		0.7 V_{VIO}		V_{VIO}+0.3	V
Input Schmitt trigger hysteresis	V_{INHYST}			0.12 V_{VIO}		V
Output voltage low level	V_{OUTLO}	I_{OUTLO} = 2mA			0.2	V
Output voltage high level	V_{OUTHI}	I_{OUTHI} = -2mA	V_{VIO}-0.2			V
Input leakage current	I_{ILEAK}		-10		10	µA
Pullup / pull-down resistors	R_{PU}/R_{PD}		132	166	200	kΩ
Digital pin capacitance	C			3.5		pF

AIN/IREF input	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
AIN_IREF input resistance to 2.5V (=5VOUT/2)	R_{AIN}	Measured to GND (*internalRsense*=0)	260	330	400	kΩ
AIN_IREF input voltage range for linear current scaling	V_{AIN}	Measured to GND (*IscaleAnalog*=1)	0	0.5-2.4	V_{5VOUT}/2	V
AIN_IREF open input voltage level	V_{AINO}	Open circuit voltage (*internalRsense*=0)		V_{5VOUT}/2		V
AIN_IREF input resistance to GND for reference current input	R_{IREF}	Measured to GND (*internalRsense*=1)	0.5	0.7	1.0	kΩ
AIN_IREF current amplification for reference current to coil current at maximum setting	$I_{REFAMPL}$	I_{IREF} = 0.25mA		3000		Times
Motor current full scale tolerance -using RDSon measurement (value for design guideline to calculate reproduction of certain motor current & torque)	I_{COIL}	*Internal_Rsense*=1, *vsense*=0, I_{IREF} = 0.25mA (after reaching thermal balance)	-10		+10	%

18.3 Thermal Characteristics

The following table shall give an idea on the thermal resistance of the package. The thermal resistance for a four layer board will provide a good idea on a typical application. Actual thermal characteristics will depend on the PCB layout, PCB type and PCB size. The thermal resistance will benefit from thicker CU (inner) layers for spreading heat horizontally within the PCB. Also, air flow will reduce thermal resistance.

A thermal resistance of 30K/W for a typical board means, that the package is capable of continuously dissipating 3.3W at an ambient temperature of 25°C with the die temperature staying below 125°C. Note, that a thermally optimized layout is required.

Parameter	Symbol	Conditions	Typ	Unit
Typical power dissipation	P_D	stealthChop or spreadCycle, 1A RMS in two phase motor, sinewave, 40 or 20kHz chopper, 24V, 110°C peak surface of package (motor QSH4218-035-10-027)	2.5	W
Typical power dissipation	P_D	stealthChop or spreadCycle, 0.7A RMS in two phase motor, sinewave, 40 or 20kHz chopper, 24V, 68°C peak surface of package (motor QSH4218-035-10-027)	1.36	W
Thermal resistance junction to ambient on a multilayer board QFN28	R_{TMJA}	Dual signal and two internal power plane board (2s2p) as defined in JEDEC EIA JESD51-5 and JESD51-7 (FR4, 35µm CU, 70mm x 133mm, d=1.5mm)	30	K/W
TQFP48-EP			24	
Thermal resistance junction to case	R_{TJC}	Junction to heat slug of package	6	K/W

Table 18.1 Thermal characteristics TMC22xx

Note
A spread-sheet for calculating TMC22xx power dissipation is available on www.trinamic.com.

19 Layout Considerations

19.1 Exposed Die Pad

The TMC22xx uses its die attach pad to dissipate heat from the drivers and the linear regulator to the board. For best electrical and thermal performance, use a reasonable amount of solid, thermally conducting vias between the die attach pad and the ground plane. The printed circuit board should have a solid ground plane spreading heat into the board and providing for a stable GND reference.

19.2 Wiring GND

All signals of the TMC22xx are referenced to their respective GND. Directly connect all GND pins under the device to a common ground area (GND and die attach pad). The GND plane right below the die attach pad should be treated as a virtual star point. For thermal reasons, the PCB top layer shall be connected to a large PCB GND plane spreading heat within the PCB.

Attention
Especially the sense resistors are susceptible to GND differences and GND ripple voltage, as the microstep current steps make up for voltages down to 0.5 mV. No current other than the sense resistor current should flow on their connections to GND and to the TMC22xx. Optimally place them close to the IC, with one or more vias to the GND plane for each sense resistor. The two sense resistors for one coil should not share a common ground connection trace or vias, as also PCB traces have a certain resistance.

19.3 Supply Filtering

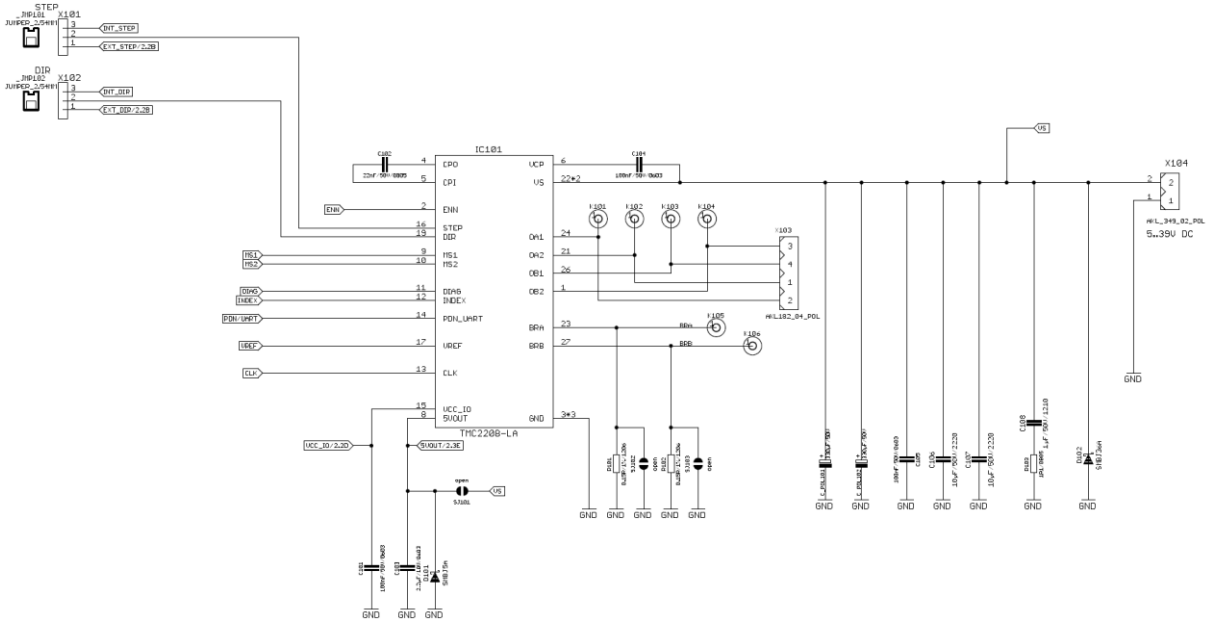
The 5VOUT output voltage ceramic filtering capacitor (2.2 to 4.7 µF recommended) should be placed as close as possible to the 5VOUT pin, with its GND return going directly to the die pad or the nearest GND pin. This ground connection shall not be shared with other loads or additional vias to the GND plane. Use as short and as thick connections as possible.

The motor supply pins VS should be decoupled with an electrolytic capacitor (47 µF or larger is recommended) and a ceramic capacitor, placed close to the device.

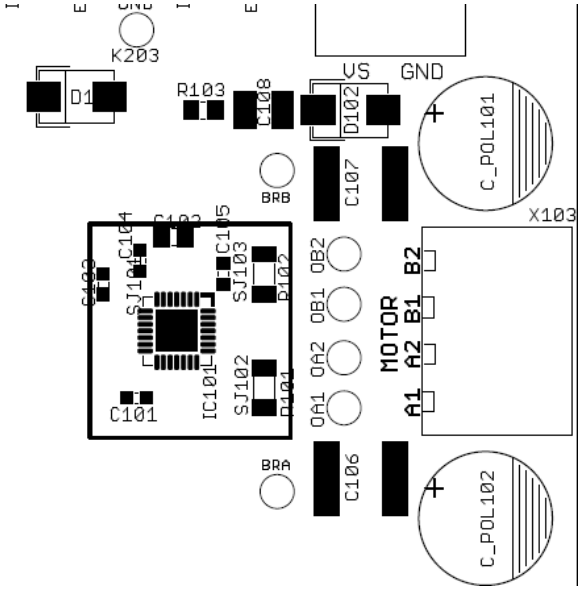
Take into account that the switching motor coil outputs have a high dV/dt. Thus capacitive stray into high resistive signals can occur, if the motor traces are near other traces over longer distances.

19.4 Layout Example TMC2208

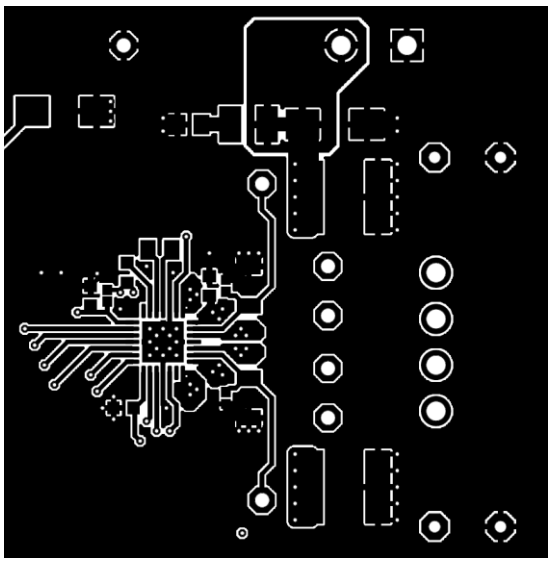
Schematic



Placement (Excerpt)



Top Layout (Excerpt, showing die pad vias)



The complete schematics and layout data for all TMC22xx evaluation boards are available on the TRINAMIC website.

20 Package Mechanical Data

20.1 Dimensional Drawings QFN28

Attention: Drawings not to scale.

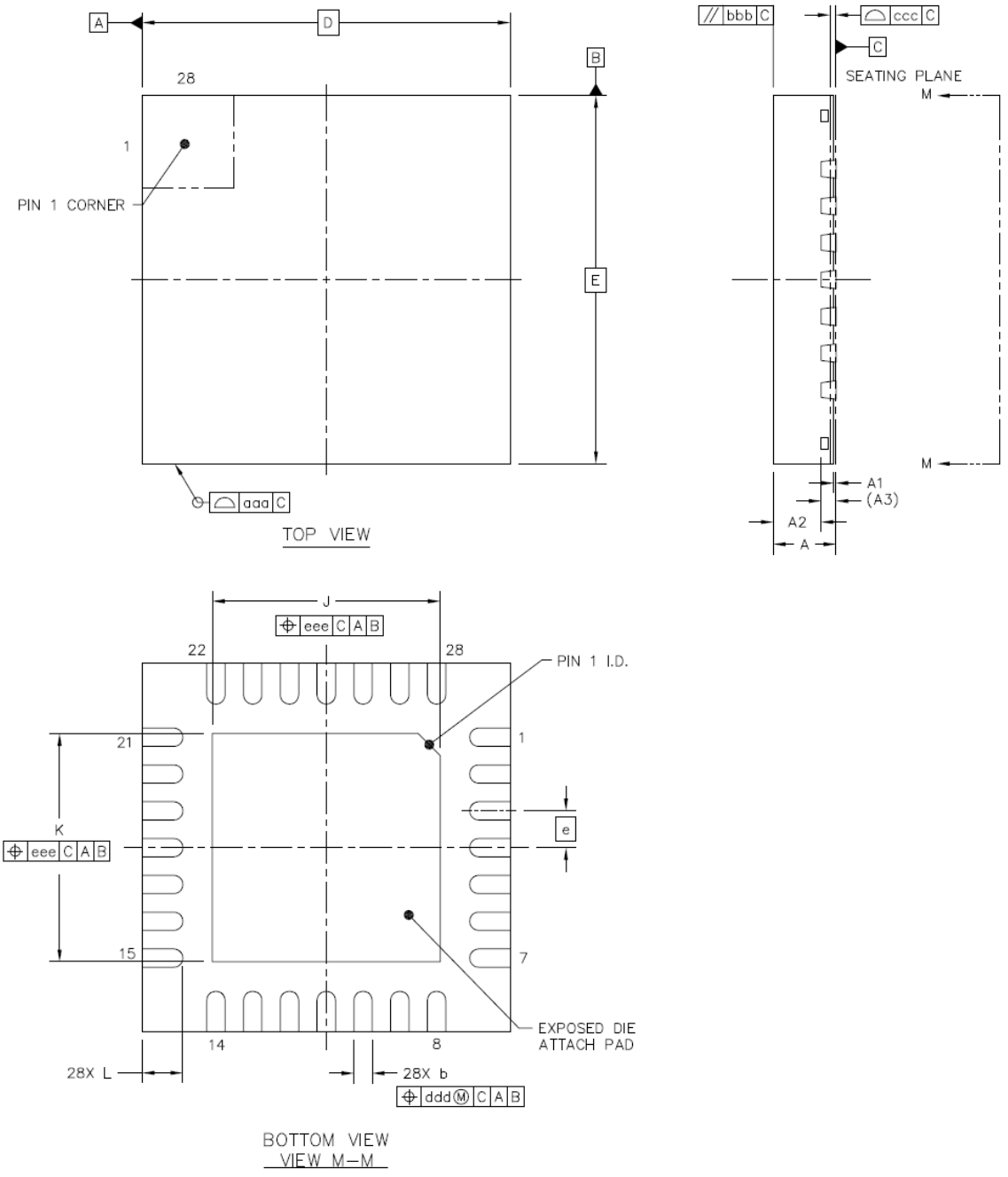


Figure 20.1 Dimensional drawings QFN28

Parameter [mm]	Ref	Min	Nom	Max
total thickness	A	0.8	0.85	0.9
stand off	A1	0	0.035	0.05
mold thickness	A2		0.65	
lead frame thickness	A3		0.203	
body size X	D		5.0	
body size Y	E		5.0	
lead pitch	e		0.5	
exposed die pad size X	J	3	3.1	3.2
exposed die pad size Y	K	3	3.1	3.2
lead length	L	0.5	0.55	0.6
package edge tolerance	aaa			0.1
mold flatness	bbb			0.1
coplanarity	ccc			0.08
lead offset	ddd			0.1
exposed pad offset	eee			0.1

20.2 Dimensional Drawings QFN32-WA

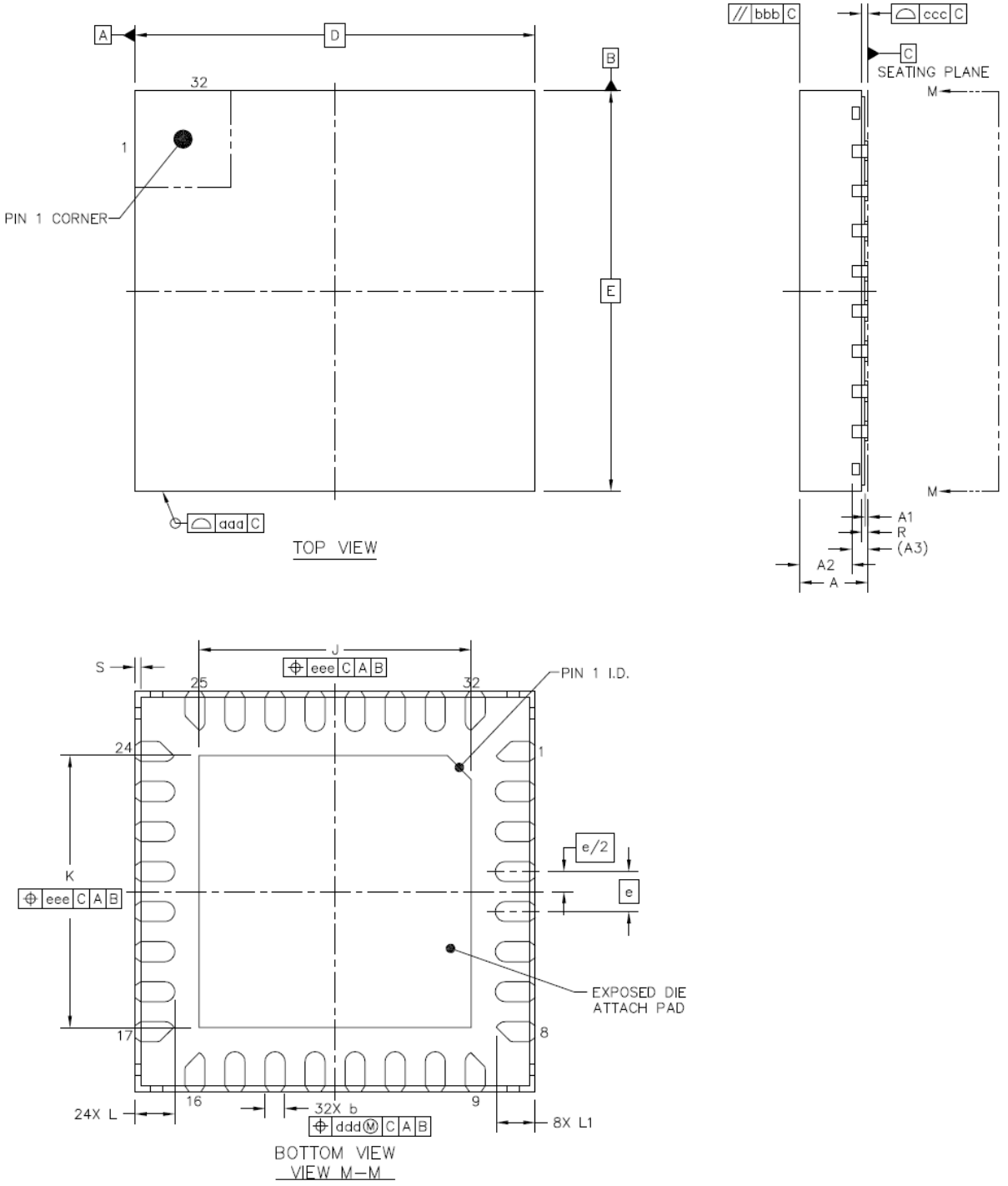


Figure 20.2 Dimensional drawings QFN32-WA

Parameter [mm]	Ref	Min	Nom	Max
total thickness	A	0.8	0.85	0.9
stand off	A1	0	0.035	0.05
mold thickness	A2		0.65	
lead frame thickness	A3		0.203	
lead width	b	0.2	0.25	0.3
body size X	D		5.0	
body size Y	E		5.0	
lead pitch	e		0.5	
exposed die pad size X	J	3.3	3.4	3.5
exposed die pad size Y	K	3.3	3.4	3.5
lead length	L	0.45	0.5	0.55
lead length	L1	0.4	0.5	0.55
package edge tolerance	aaa		0.1	
mold flatness	bbb		0.1	
coplanarity	ccc		0.08	
lead offset	ddd		0.1	
exposed pad offset	eee		0.1	
half-cut width	R	0.075		
half-cut depth	S			0.075

20.3 Package Codes

Type	Package	Temperature range	Code & marking
TMC2208-LA	QFN28 (RoHS)	-40°C ... +125°C	TMC2208-LA
TMC2224-LA	QFN28 (RoHS)	-40°C ... +125°C	TMC2224-LA
TMC2202-WA	QFN32 (RoHS)	-40°C ... +125°C	TMC2202-WA

21 Table of Figures

FIGURE 1.1 TMC22XX BASIC APPLICATION BLOCK DIAGRAM 4
FIGURE 1.2 STAND-ALONE DRIVER WITH PRE-CONFIGURATION 5
FIGURE 1.3 AUTOMATIC MOTOR CURRENT POWER DOWN 7
FIGURE 2.1 TMC2208 PINNING TOP VIEW – TYPE: QFN28, 5X5MM², 0.5MM PITCH 8
FIGURE 2.2 TMC2202 PINNING TOP VIEW – TYPE: QFN32, 5X5MM², 0.5MM PITCH 9
FIGURE 2.3 TMC2224 PINNING TOP VIEW – TYPE: QFN28, 5X5MM², 0.5MM PITCH 10
FIGURE 2.4 TMC2225 PINNING TOP VIEW – TYPE: HTSSOP28, 9.7X6.4MM² OVER PINS, 0.65MM PITCH 12
FIGURE 2.5 TMC2220 PINNING TOP VIEW – TYPE: TQFP-EP 48, 9X9MM² OVER PINS, 0.5MM PITCH 13
FIGURE 3.1 STANDARD APPLICATION CIRCUIT 15
FIGURE 3.2 APPLICATION CIRCUIT USING RDSON BASED SENSING 16
FIGURE 3.3 5V ONLY OPERATION 16
FIGURE 3.4 SIMPLE ESD ENHANCEMENT AND MORE ELABORATE MOTOR OUTPUT PROTECTION 18
FIGURE 4.1 ATTACHING THE TMC22XX TO A MICROCONTROLLER UART 21
FIGURE 4.2 ADDRESSING MULTIPLE TMC22XX VIA SINGLE WIRE INTERFACE USING ANALOG SWITCHES 22
FIGURE 6.1 MOTOR COIL SINE WAVE CURRENT WITH STEALTHCHOP (MEASURED WITH CURRENT PROBE) 37
FIGURE 6.2 STEALTHCHOP2 AUTOMATIC TUNING PROCEDURE 38
FIGURE 6.3 SCOPE SHOT: GOOD SETTING FOR PWM_REG 40
FIGURE 6.4 SCOPE SHOT: TOO SMALL SETTING FOR PWM_REG DURING AT#2 40
FIGURE 6.5 SUCCESSFULLY DETERMINED PWM_GRAD(_AUTO) AND PWM_OFS(_AUTO) 40
FIGURE 6.6 VELOCITY BASED PWM SCALING (PWM_AUTOSCALE=0) 42
FIGURE 6.7 TPWMTHRS FOR OPTIONAL SWITCHING TO SPREADCYCLE 43
FIGURE 7.1 CHOPPER PHASES 47
FIGURE 7.2 NO LEDGES IN CURRENT WAVE WITH SUFFICIENT HYSTERESIS (MAGENTA: CURRENT A, YELLOW & BLUE: SENSE RESISTOR VOLTAGES A AND B) 49
FIGURE 7.3 SPREADCYCLE CHOPPER SCHEME SHOWING COIL CURRENT DURING A CHOPPER CYCLE 50
FIGURE 9.1 SCALING THE MOTOR CURRENT USING THE ANALOG INPUT 54
FIGURE 11.1 STEP AND DIR TIMING, INPUT PIN FILTER 57
FIGURE 11.2 MICROPLYER MICROSTEP INTERPOLATION WITH RISING STEP FREQUENCY (EXAMPLE: 16 TO 256) 59
FIGURE 11.3 INDEX SIGNAL AT POSITIVE ZERO TRANSITION OF THE COIL A SINE CURVE 60
FIGURE 12.1 SOFTWARE GENERATED MOTION PROFILE 61
FIGURE 13.1 DIAG AND INDEX OUTPUTS 63
FIGURE 14.1 CURRENT SETTING AND FIRST STEPS WITH STEALTHCHOP 64
FIGURE 14.2 TUNING STEALTHCHOP AND SPREADCYCLE 65
FIGURE 14.3 OTP PROGRAMMING 66
FIGURE 20.1 DIMENSIONAL DRAWINGS QFN28 76
FIGURE 20.2 DIMENSIONAL DRAWINGS QFN32-WA 78

22 Revision History

Version	Date	Author BD= Bernhard Dwersteg	Description
V0.25	2016-09-02	BD	First complete datasheet
V0.28	2016-09-21	BD	Some detail wording, request for option packages, added CRC procedure
V1.00	2016-11-01	BD	Adapted min/max DC and Timing Characteristics (I_S, V_{CP_UV}, V_{UV_VS}, 25°C for clock calibration frequency) to fit test limits
V1.01	2016-11-16	BD	Corrected wording in DRV_STATUS
V1.02	2017-05-16	BD	Added QFN-32

Table 22.1 Document Revisions

23 References

[TMC22xx-EVAL] TMC22xx Evaluation board: Manuals, software and PCB data available on www.trinamic.com

[AN001] Trinamic Application Note 001 - Parameterization of spreadCycle™, www.trinamic.com

Calculation sheet TMC22xx_Calculations.xlsx www.trinamic.com

Orange Pi Zero
User Manual

Contents

I. Orange Pi Introduction....1
1. What is Orange Pi Zero?....1
2. What can I do with Orange Pi Zero?....1
3. Whom is it for?....1
4. Hardware specification of Orange Pi Zero....1
5. GPIO Specifications....4
6. Specification of CSI Camera Connector....4
II. Using Method Introduction....6
1. Step 1: Prepare Accessories Needed....6
2. Step 2: Prepare a TF Card....7
3. Step 3: Boot your Orange Pi....13
4. Step 4: Turn off your Orange Pi Correctly....15
5. Other configuration....15
6. Universal Software Configuration....18
III. Linux Kernel Source Code Compilation....28
1. Download Linux Source Code....28
2. Compile Project Source Code....29
3. Update the Kernel Image File and Replace Library....31
IV. Android Kernel Source Code Compilation....33
1. Install JDK....33
2. Install Platform Supported Software....34
3. Download Android Source Package....34
4. Install Compiler Tool Chain....34
5. Compile Lichee Source Code....35
6. Compile Command of Android Code....35
V. Use Project Configuration Files....38
1. sys_config.fex Introduction....38
2. Examples....38
VI. OrangePi Driver development....41
1. Device Driver and Application Programming....41
2. Compile device driver....44
3. Cross compiler Application Program....46

4. Running Driver and Application.....48

VII. Using Debug tools on OrangePi.....50

1. Operation Steps on Windows.....50

2. Operation Steps on Linux.....54

I. Orange Pi Introduction

1. What is Orange Pi Zero?

It's an open-source single-board computer. It can run Android 4.4, Ubuntu, Debian, Rasberry Pi Image. It uses the AllWinner H2 SoC, and has 256MB/512MB DDR3 SDRAM(256MB version is Standard version).

2. What can I do with Orange Pi Zero?

You can use it to build…

- A computer
- A wireless server
- Games
- Music and sounds
- HD video
- A speaker
- Android
- Scratch
- .....

Pretty much anything else, because Orange Pi Zero is open source

3. Whom is it for?

Orange Pi Zero is for anyone who wants to create with technology– not just consuming. It's a simple, fun, useful tool and you can use it to take control of the world around you.

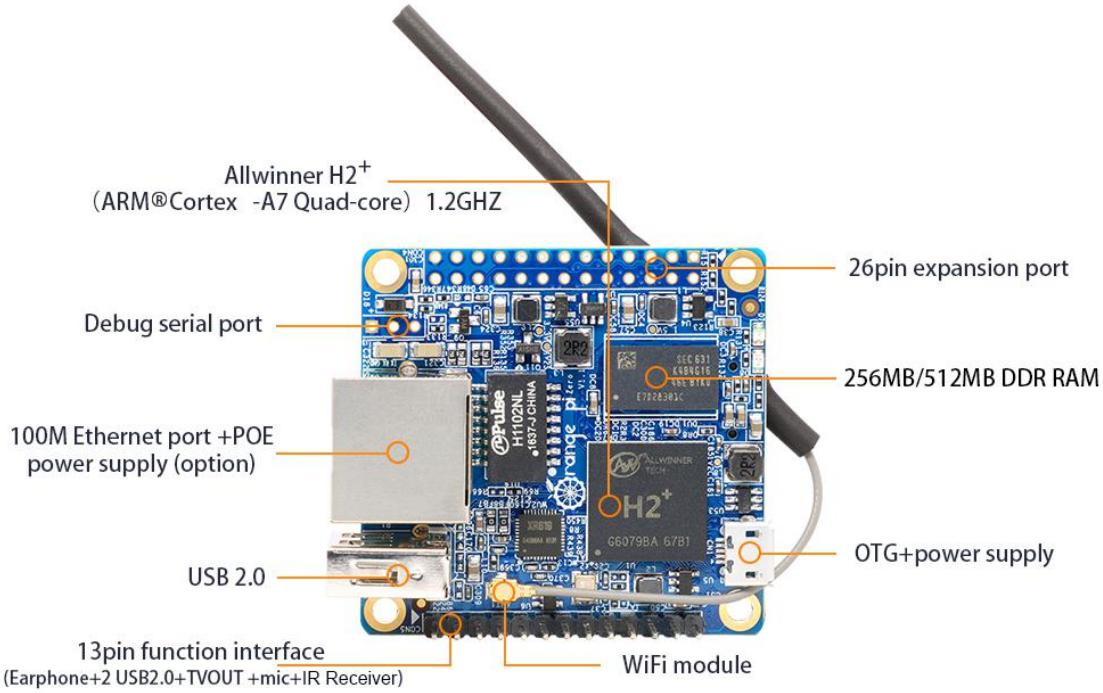
4. Hardware specification of Orange Pi Zero

Hardware specification	
CPU	H2 Quad-core Cortex-A7 H.265/HEVC 4K
GPU	Mali400MP2 GPU @600MHz, Supports OpenGL ES 2.0
Memory (SDRAM)	256MB/512MB DDR3 (shared with GPU)(256MB is Standard version)
Onboard Storage	TF card (Max. 32GB) / NOR flash(2MB)

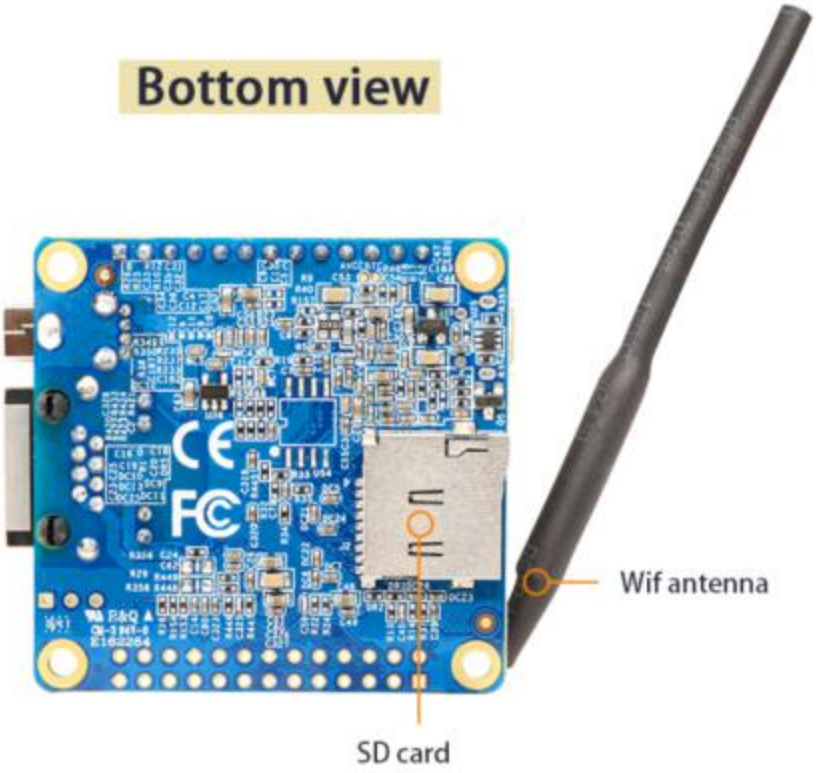
Wifi Antenne	Yes
Onboard Network	100M/10M Ethernet RJ45 (Integrated POE power supply)
Onboard WIFI	XR819, IEEE 802.11 b/g/n
Audio Input	No, need to use an expansion board
Video Outputs	Supports external board via 13pins
Power Source	USB OTG can supply power(Integrated POE power supply)
USB 2.0 Ports	One USB 2.0 HOST, one USB 2.0 OTG
Buttons	Power button
Low-level peripherals	26 Pins Header, compatible with Raspberry Pi B+ 13 Pins Header, with 2x USB, IR pin, AUDIO(MIC, AV)
GPIO(1x3) pin	UART,ground.
LED	Power led & Status led
Supported OS	Android Lubuntu, Debian, Rasberry Pi Image
Interface definition	
Product size	45mm × 48mm
Weight	30g
Orange Pi™ is a trademark of the Shenzhen Xunlong Software CO., Limited	

Interface instructions

Top view

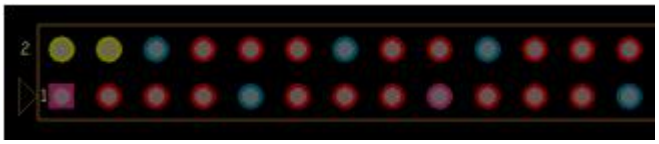


Bottom view



5. GPIO Specifications

A 26-pin GPIO interface on the Orange Pi Zero is the same as Model A and Model B of Raspberry Pi. The picture below is GPIO pin define of Orange Pi Zero.

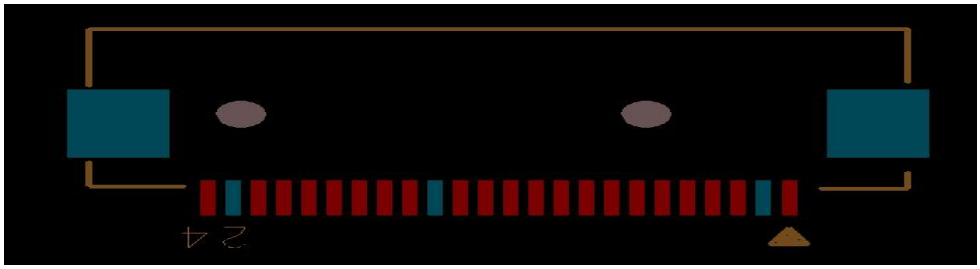


OrangePi_zero(H2)		
CON3-P01	VCC-3V3	
CON3-P02	VCC-5V	
CON3-P03	TWI0-SDA	PA12
CON3-P04	VCC-5V	
CON3-P05	TWI0-SCK	PA11
CON3-P06	GND	
CON3-P07	PWM1	PA6
CON3-P08	UART2_TX	PA0
CON3-P09	GND	
CON3-P10	UART2_RX	PA1
CON3-P11	S-TWI-SCK	PL0
CON3-P12	PD11	PD11
CON3-P13	S-TWI-SDA	PL1
CON3-P14	GND	
CON3-P15	UART2_CTS	PA3
CON3-P16	TWI1-SDA	PA19
CON3-P17	VCC3V3-EXT	
CON3-P18	TWI1-SCK	PA18
CON3-P19	SPI1_MOSI	PA15
CON3-P20	GND	
CON3-P21	SPI1_MISO	PA16
CON3-P22	UART2_RTS	PA2
CON3-P23	SPI1_CLK	PA14
CON3-P24	SPI1_CS	PA13
CON3-P25	GND	
CON3-P26	PD14	PD14

6. Specification of CSI Camera Connector

The CSI Camera Connector is a 24-pin FPC connector which can connect external camera module with proper signal pin mappings. The pin of

CIS connector can be defined as follows. The connector marked with "CON 1" on the Orange Pi Zero is camera connector.



Orange Pi Zero-CSI

CON1-P01	NC	
CON1-P02	GND	
CON1-P03	TWI2-SDA	PE13
CON1-P04	VCC-CSI	
CON1-P05	TWI2-SCK	PE12
CON1-P06	CSI-RESET#	PE15
CON1-P07	CSI-VSYNC	PE3
CON1-P08	CSI-STBY-EN	PE15
CON1-P09	CSI-HSYNC	PE2
CON1-P10	VDD1V8-CSI	
CON1-P11	VCC-CSI	
CON1-P12	CSI-D7	PE11
CON1-P13	CSI-MCLK	PE1
CON1-P14	CSI-D6	PE10
CON1-P15	GND	
CON1-P16	CSI-D5	PE9
CON1-P17	CSI-PCLK	PE0
CON1-P18	CSI-D4	PE8
CON1-P19	CSI-D0	PE4
CON1-P20	CSI-D3	PE7
CON1-P21	CSI-D1	PE5
CON1-P22	CSI-D2	PE6
CON1-P23	GND	
CON1-P24	AFVCC-CSI	

II. Using Method Introduction

Follow these steps, you can configure and run your Orange Pi in a very short period of time. Boot your Orange Pi need to complete the following steps.

1. Step 1: Prepare Accessories Needed

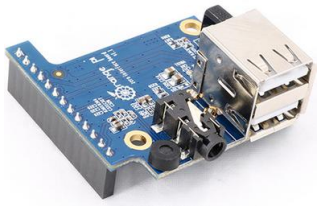
You need at least some accessories like the following if it is your first time to use the Orange Pi(we would suggest you using the Expansion board at the same time).

No.	Items	Requirements and Instructions
1	TF card	8GB min.; class 10. Branded TF cards would be reference which are much more reliable.
2	AV video cable	A standard AV video cable can be used to connect stimulated monitor if a HDMI monitor is unavailable.
3	Keyboard and mouse	Any keyboard and mouse with USB port is applicable; Keyboard and mouse are high-power, so a USB concentrator is required.
4	Ethernet cable/USB WiFi(Optional)	Network is optional, It makes more convenient to mount and upgrade software in your Orange Pi PC.
5	DC power adapter	5V/2V min. high qualified power adapter, OTG can used a power supply.
6	Audio cable (Optional)	You can select an audio cable with 3.5mm jack to feel stereo audio.

AV video cable

TF card

DC power adapter



Expansion Board

2. Step 2: Prepare a TF Card

In order to use Orange Pi normally, you must install the operating system into TF card first.

1) Write Linux into TF Card Based on Windows Platform

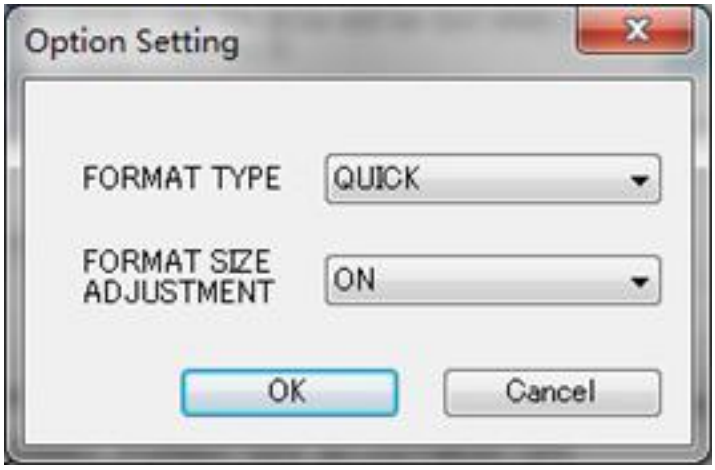
a. Inserting the TF card into the computer, the capacity of the card must be bigger than the operating system, usually requires 8GB or bigger.

b. Formatting the TF card.

i Download tools for formatting TF card, such as TF Formatter, it could be downloaded from:
https://www.sdcard.org/downloads/formatter_4/eula_windows/

ii Unzip the downloaded files, and run *setup.exe*

iii In the *options settings* select the "*form*at" button for quick formatting. "*Format size adjustment*" select "(ON)"



iv Make sure the inserted TF card disk are in accordance with the chosen disk.

v Click the "*Format*" button.

c. Download the operating system image file from the download page, the page address is as following:

http://www.orangepi.org/downloadresources

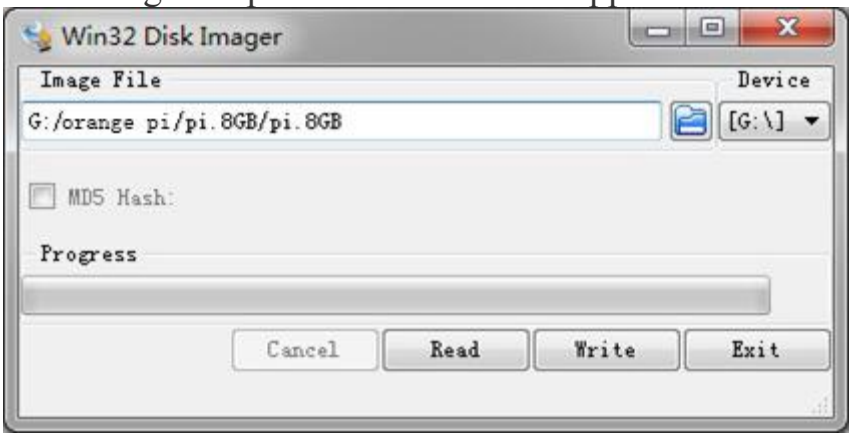
d. Unzip the downloaded file (in addition to the Android system, this method can be used to burn to write, the Android system need another burn, the following will introduce)

e. Right click to download the file, select "*Unzip file*" to write image to TF card

i Download tools to write image,such as *Win32 Diskimager,* here is the download page:

http://sourceforge.net/projects/win32diskimager/files/Archive/

ii Select the image file path that has been unzipped.



iii Click "*Write*" button and wait for the image to write.

iv After the image is written, click "*Exit*" button.

2) Write Linux into TF card based on Linux platform?

a. Inserting the TF card into the computer, the capacity of the card must be larger than the operating system image, usually requires 4GB or greater capacity.

b. Formatting the TF card.

i Run ***fdisk –l*** order to make sure TF disk.

ii Run ***umount /dev/sdxx*** to uninstall all partitions of TF Card.

iii Run ***sudo fdisk /dev/sdx*** order. Use ***o*** command to delete all partitions of TF Card, and then us ***n*** order to add a new partition, finally use ***w*** command to save and exit.

iv Run ***sudo mkfs.vfat /dev/sdx1*** command to format the TF card partition set up last step to FAT32 form(according to your TF card disk to replace***x***). Or you could skip this step since command in Linux will format TF card automatic.

c. Download the OS image from download page

http://www.orangepi.org/`downloadresources

d. Unzip and right click the downloaded file, select " *Unzip file*"

e. Write image to TF card

i Run **sudo fdisk –l** order to make sure the TF card disk

ii make sure the image file **hash key** is the same as download page mention(optional). It will output ***sha1sum [path]/[imagename]***, which should be same as the image paye *"SHA-1"*

iii Run ***umount /dev/sdxx*** order to uninstall all partitions in TF Card

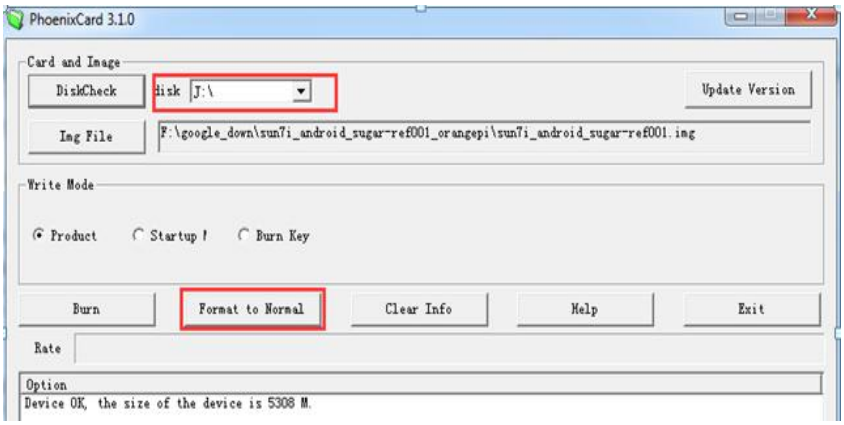
iv Run ***sudo dd bs=4M if=[path]/[imagename] of=/dev/sdx*** to write down image file. Wait for the image to write. If it cannot work at 4M, then replace a 1M which takes more time. You can run ***sudo pkill –USR1 –n –x dd*** order to monitoring procedure.

3) Use PhoenixCard tool to write Android image into TF card

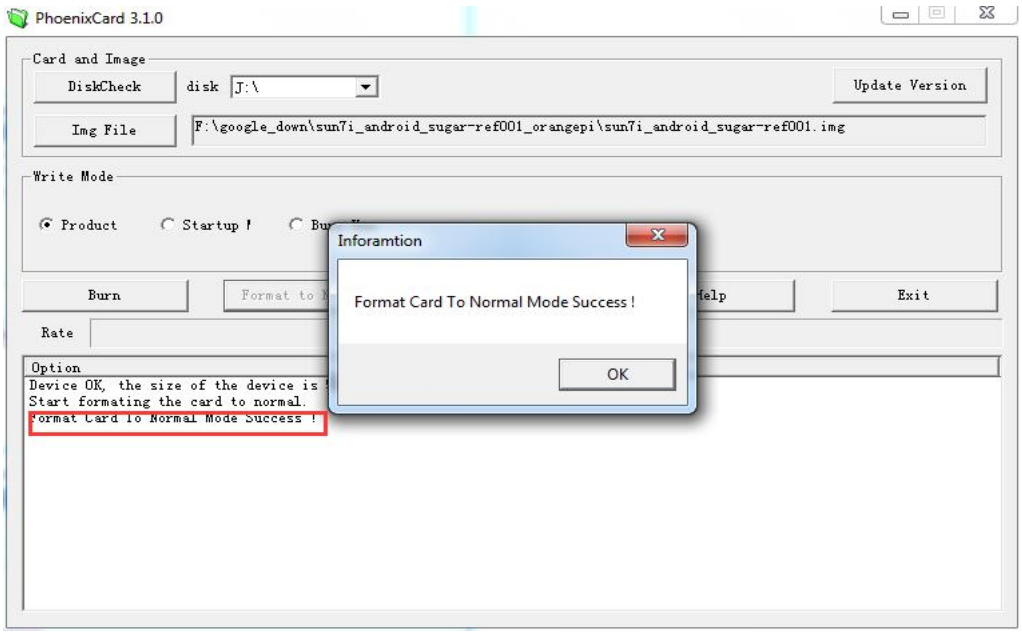
It is impossible for Android image to be written into TF card by using *dd* command under Linux or using *Win32 Diskimager* under Windows. Here

PhoenixCard tool is applicable for Android image writing.

a. Download the Android OS image and **PhoenixCard** tool.
   Download **PhoenixCard** from here:
   https://drive.google.com/file/d/0B_VynIqhAcB7NTg2UkRDdHRWX2s/edit?usp=sharing
   Download Android OS image from here:
   http://www.orangepi.org/downloadresources/
b. Format the TF card

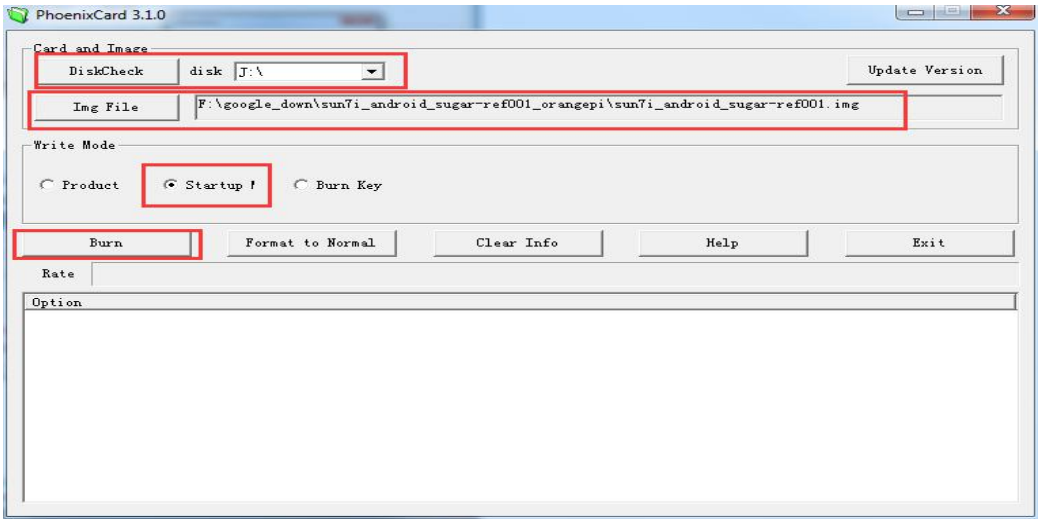


c. Please make sure the inserted TF card is in accordance with the chosen TF card, click "*restore*" button for TF card formatting.

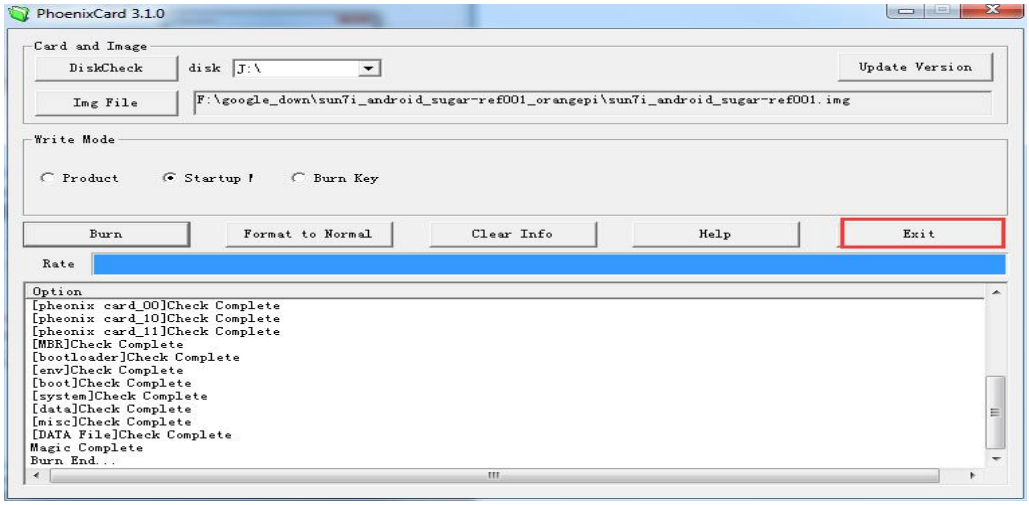


d. Click "*OK*" button after successfully formatted the TF card to normal.

e. Burn the Android OS image into your TF card. Please pay attention to the following with red marks.



f. Click "*Burn*" button for writing to TF card and wait for it finish



g. Click "*Exit*" button after burn Android image to TF card successfully.

4) Write Armbian Image into TF Card

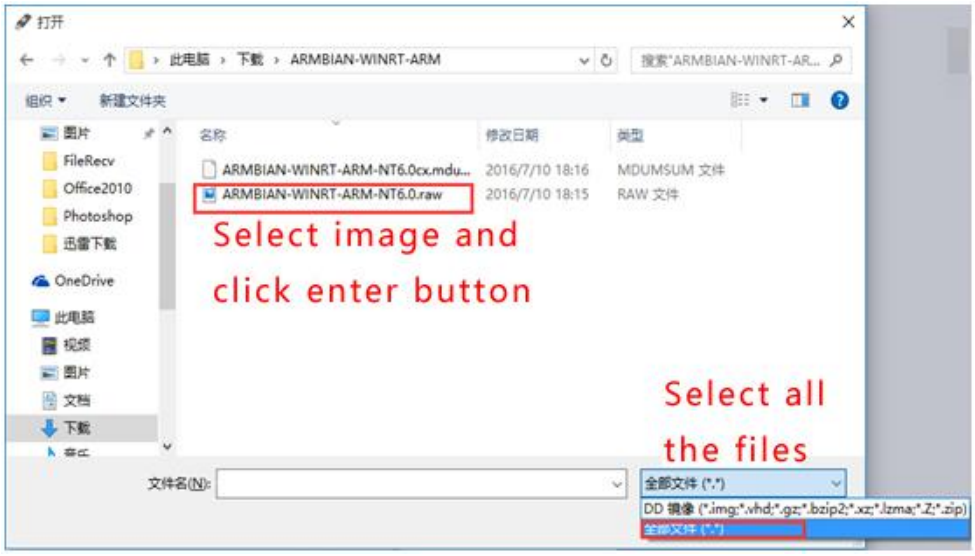
a. Insert TF card into computer, please note that the TF card capacity must bigger than the operating system image, usually need to be 8GB or bigger.

b. Download the OS image file from the download page:
http://www.armbian.com/download/

c. Write the image into TF card.

i Download image writing tool such as *Rufus*, the download page: https://rufus.akeo.ie/

ii Select the image file path that has been unzipped



iii Click "*start*" button and wait for the image to write.

iv After the image is written, click "*close*" button

3. Step 3: Boot your Orange Pi

1) Hardware Connection Sketch Map

Orange Pi Zero runs on Android 4.4 system

Orange Pi Zero runs on Debian system

Orange Pi Zero runs on Ubuntu system

2) Details of Booting Steps

a. Insert the TF card with written image in to the TF card slot.
b. You could use HDMI cable to connect your Orange Pi to HDMI TV or monitor.

You could also use AV interface and audio interface to connect output

video and audio to analog TV or display.

c. There is 13pin on board which you could connect to expansion board. For expansion board, 2USB ports, mic and IR receiver are available.

d. It is the network module on board, which you can access Orange Pi to the wired network.

e. You could connect to a power adapter on mic USB OTG with a power adapter up to or bigger than 5V/2A. Avoid using smaller power GSM mobile phone charger, it is not able to output 2A even if it marked "5V/2A ".

The Orange Pi will boot in a few minutes If the above steps are successful. There will be graphical interface in he monitor. It may take a long time to start the first time, please wait patiently. The next time will boot very fast.

4. Step 4: Turn off your Orange Pi Correctly

- You can use the shutdown button on the interface to safety close the Orange Pi.
- You can also close the system by entering commands in the shell:

sudo halt
or
sudo shutdown –h

It will be safety to turn off the Orange Pi. If directly use the power button to shut down the system may damage the file system on TF Card. After the system is closed, the power can be cut off by more than 5 seconds' press.

5. Other configuration

1) Connect to the wired network

- Method 1:

a. Enter the following in the command line:
$ ifconfig
To check whether there is (wlan*)
b. If no, load the corresponding module according to the wlan model

$ insmod xradio_wlan.ko
For example: For xr819 is xradio_wlan.ko

c. Enter command ifconfig, you should see wlan0(hypothesis it is wlan0)
d. Configure wired network, first you need to know ssid and psk(account and password), enter corresponding wlan*, ssid, psk

```
$ sudo nano /etc/network/interfaces        (add the following contents)
auto wlan0
iface wlan0 inet dhcp
wpa-ssid xxxx
wpa-psk xxxx
```

e. Reboot the computer and the wired network will work.

```
$ sudo reboot
```

- Method 2:

a. Build wifi hotspot configuration file of wpa_supplication.conf for on /etc/network/ directory and add the following:

```
network={
        ssid="wifi hot spot name"
        psk="wifi hot spot password"
        priority=1
}
```

b. Connect wifi, here is the command:

```
ifconfig wlan0 up
sudo wpa_supplicant -i wlan0 -c /etc/network/wpa_supplication.conf &
dhcpcd wlan0 &
```

c. Test the condition of wifi connection

Use iwconfig command, you will find the related information of wlan0, use ping command to test.

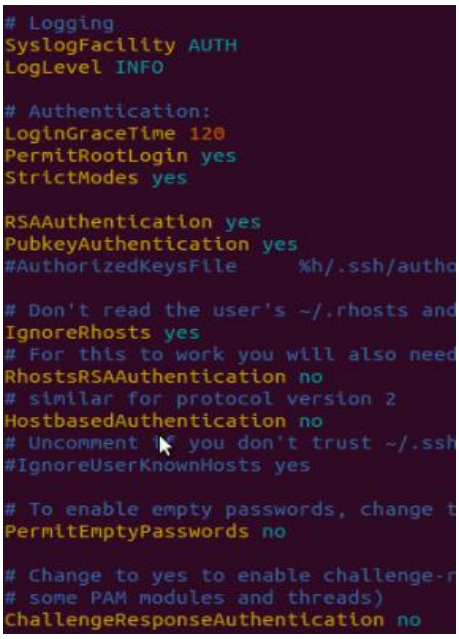
2) Login via vnc and ssh

If there is no condition for connecting HDMI, you could enter the system via vnc or ssh remote login.

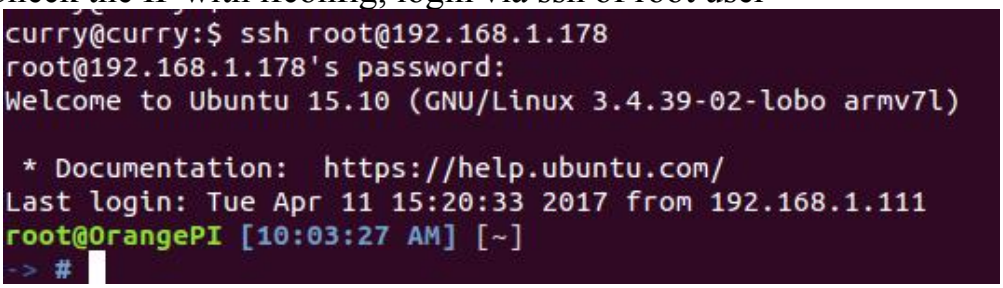
- Login via serial port and install ssh

```
apt-get install ssh
```

- Modify ssh configuration file /etc/ssh/sshd_config



- Check the IP with ifconfig, login via ssh of root user



3) HDMI or 3.5mm Sound Output(3.5mm sound output would require using an expansion board)

a. The sound was default to output via HDMI on image, it could check and change via alsamixer.

ls /etc/asound.conf

card indicates card number, device indicates device number.

aplay -l it could check the system to load the sound card number and details

cat /proc/asound/cards it also could check the sound card and details

It could be used after use alsamixer to change the sound card.

alsactl store -f /var/lib/alsa/asound.state used for saving modified parameters

b. It needs to modify configuration on file system for output on 3.5mm of /etc/asound.conf, modify card1 into card0, or use amixer to modify. The default one is configured, or you could use player on graphical interface to switch via sound channel selection.

c. How to use mic sound recording
arecord -d 5 -f cd -t wav 123.wav
After recording, use the following to play
aplay 123.wav

6. Universal Software Configuration

1) Default Account Changing

The default log in account is orangepi. In order to secure, it is recommended to modify the default orangepi accounts to your own account, for example Zhangsan. Steps are as follows:

a. Use root account to login Orange Pi(please note that do not login with the account of orangepi)

b. $ usermod -l zhangsan orangepi Change orangepi account into Zhangsan

```
@orangepi:~$ usermod -l zhangsan orangepi
```

c. $ groupmod -n zhangsan orangepi Change group

```
@orangepi:~$ groupmod -n zhangsan orangepi
```

d. $ mv /home/ornagepi /home/zhangsan Change directory of original orangepi

```
@orangepi:~$ mv /home/orangepi /home/zhangsan
```

e. $ usermod -d /home/orangepi orangepi Set this directory to orangepi user's home directory

```
@orangepi:~$ usermod -d /home/zhangsan zhangsan
```

f. $ cat /etc/passwd It should be shown as below:

```
pulse:x:112:121:PulseAudio daemon,,,:/var/run/pulse:/bin/false
zhangsan:x:1001:1001:orangepi,,,,:/home/zhangsan:/bin/bash
```

After the modification of the above iterms, it can be used the new account Zhangsan to land.

2) U Disk Automatic Mounted Configuration

a. sudo apt-get install usbmount

b. sudo vim /etc/udev/rules.d/automount.rules

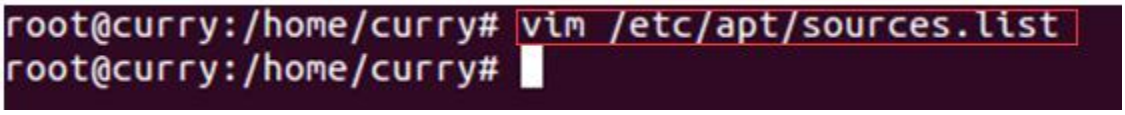
ACTION=="add",KERNEL=="sdb*", RUN+="/usr/bin/pmount --sync --umask 000 %k"
ACTION=="remove", KERNEL=="sdb*", RUN+="/usr/bin/pumount %k"
ACTION=="add",KERNEL=="sdc*", RUN+="/usr/bin/pmount --sync

--umask 000 %k"
ACTION=="remove", KERNEL=="sdc*", RUN+="/usr/bin/pumount %k"

c. udevadm control –reload-rules
It could refer to this:
http://unix.stackexchange.com/questions/134797/how-to-automatically-mount-an-usb-device-on-plugin-time-on-an-already-running-sy

3) System Source Configuration

Take Ubuntu as an example:
a. Open the source file
$ sudo vi /etc/apt/sources.list



b. Edit source file

Replace the source file with your favorite source. Take an example of Ubuntu 16.04 on Zhonkeda source:

deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial main multiverse restricted universe

deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-backports main multiverse restricted universe

deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-proposed main multiverse restricted universe

deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-security main multiverse restricted universe

deb http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-updates main multiverse restricted universe

deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial main multiverse restricted universe

deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-backports main multiverse restricted universe

deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-proposed main multiverse restricted universe

deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-security main multiverse restricted universe

deb-src http://mirrors.ustc.edu.cn/ubuntu-ports/ xenial-updates main multiverse restricted universe

Note: xenial is the version of the code name in this source, if the other version of Ubuntu needs to replace the corresponding version code which can be found on the internet.

4) Remote desktop installation

There are a lot of software, such as VNG, XRDP, X2GO, etc. For X2GO, it has more functions, and desktop color restore is very good which does not need too much configuration. And XRDP is much more safety than VNC.

a. $sudo apt-get install tightvncserver Install VNC

```
apt-get install tightvncserver
```

b. vncpassw Set the password: do not execute this command but executing vncserver directly. It will prompt you to enter the password twice, when prompted whether can be read only to select the *N*.

```
root@curry:/home/curry/tools/minidlna/minidlna-1.1.0# vncpasswd
Using password file /root/.vnc/passwd
VNC directory /root/.vnc does not exist, creating.
Password:
Verify:
```

c. Open one or more of desktops by vncserver or vncserver:1(vncserver:2)… you can also transfer more parameters through the full command as below:

vncserver :1 -geometry 1024x768 -depth 16 -pixelformat rgb565

(*Note:* If it prompted you that cannot find the file or other error when installing, please run sudo apt-get update to update the software source and try installing again.)

5) NAS and DLAN Configuration

a. NAS：

There are many files could be reference from Internet, for example: http://www.geekfan.net/5003/, it detailed descriptions on the operation and the mounted of U disk is very useful.

b. DLNA:

Mainly through the minidlna software to achieve the sharing of media resources within the LAN, such as sharing video, music, etc.. The installation steps are as follows:

i sudo apt-get minidlna

ii Execute the following command to modify the configuration file:

sudo nano /etc/minidlna.conf

Note: you can also use other text editor to modify.

iii Add the following:

media_dir=A,/nas, path: /DLNA/Music

media_dir=V,/nas, path: /DLNA/Video

media_dir=P,/nas, path: DLNA/Picture

db_dir=/nas, path: /DLNA/log

db_dir=/nas, path: /DLNA/db

ctrl +o and enter, ctrl +x to save and exit.

iv Established above folders respectively, noted that path consistency and assigned to read and write permissions. In order for convenient, it could be Chmod 755, such as sudo Chmod 755 /nas path /DLNA/Music

v Re-start minidlna to take effect the configuration: /etc/init.d/minidlna restart.

Transmit the corresponding file on the computer to the corresponding folder through samba.

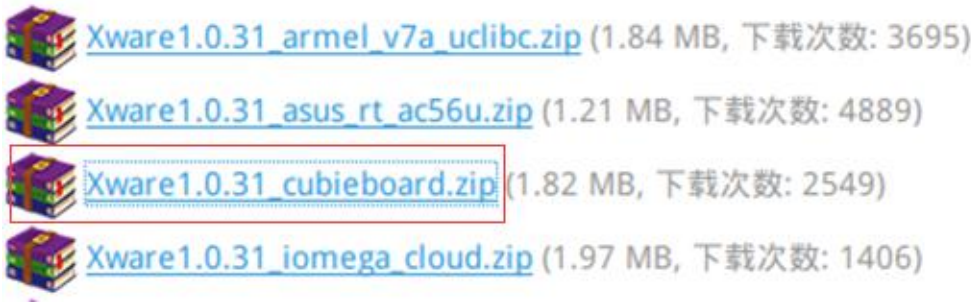
Note: It is recommended to download MoliPlayer on the mobile device. The effect is good and no blue light pressure on both Android and IOS.

6) Thunder remote download

a. Go to the Thunder routing forum to download the required installation package first. The link for stable version:

http://luyou.xunlei.com/thread-12545-1-1.html.

Download Xware1.0.31_cubieboard zip file.



Note: If you want to try the latest version, you can download the latest test version: http://luyou.xunlei.com/thread-15167-1-1.htm.

b. Enter the directory after uploaded the unzip file to OrangePi. It is recommended to rename the file to xunlei

c. Installation method of version 1.0.31:

i $ cd /xxx/xunlei The xxx is the directory of installation xunlei file

ii $ chmod 755 portal

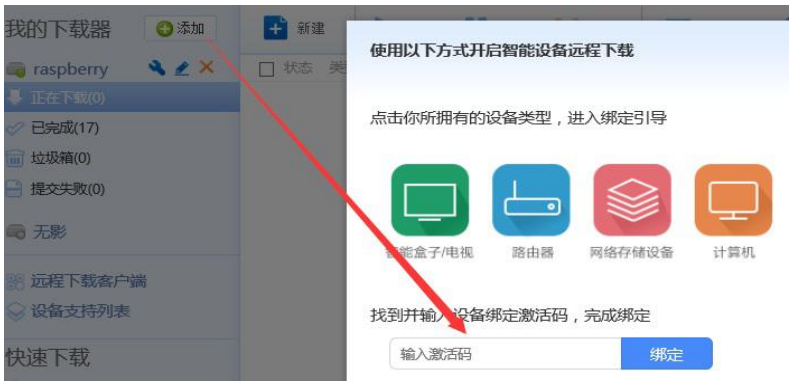
iii $ ./portal

```
root@curry:/home/curry/Downloads/xunlei# ls
EmbedThunderManager  ETMDaemon  portal  vod_httpserver
root@curry:/home/curry/Downloads/xunlei# chmod 755 portal
root@curry:/home/curry/Downloads/xunlei#
```

iv You will get an activation code after booting like the following:

```
YOUR CONTROL PORT IS: 9000

starting xunlei service...
etm path: /home/echo/xunlei
execv: /home/echo/xunlei/lib/ETMDaemon.

getting xunlei service info...
Connecting to 127.0.0.1:9000 (127.0.0.1:9000)
                                Here you will get
THE ACTIVE CODE IS:             an activation code

go to http://yuancheng.xunlei.com, bind your device with the active code.
finished.
```

v Copy this activation code to *http://yuancheng.xunlei.com* (Which required to log in with account of Thunder). Then click the tab on the top right corner to add, fill in the activation code to complete the binding according to the following figure.



vi Setting start up

$ sudo nano /etc/rc.loacl
add the following contents before exit 0
cd /xx/xunlei
./portal &
ctrl +o and enter, ctrl +x to save and exit.

d. Installation of version 3.0.32.253:

i $ cd /xxx/xunlei The xxx is the directory of installation file of xunlei

ii $ sudo nano thunder_mounts.cfg Modify the download path

iii chmod +x etm_monitor

iv Run ./etm_monitor, there will be an activation code page like version 1.0.32. And then binding on the Thunder remote page (above steps 4, 5). There might be one or two errors while running, ignore it (selection type of shell and generation of INI file).

v Setting start up

sudo nano /etc/rc.loacl add the following contents before exit 0

cd /xx/xunlei

./etm_monitor &

ctrl +o and enter, ctrl +x to save and exit.

It could be remote downloading on computer, mobile phone or tablet by login yuancheng.xunlei.com

7) Modify the size of ext4 file system

After made the written image into SD card for booting, enter into rootfs partition's expansion of file system. It could enhance the performance of SD card to avoid limited storage cause problem.

- Method 1

Extend rootfs file partition of TF card on PC:

Select the specified disk, right click and select the corresponding disk, select "change size" and adjust it into your desired size, click "re-size", close the dialog box and click "apply all operations", select the application to complete the expansion operation

- Method 2

Enter into the system and extend via shell

Before partition

```
root@Orangepi:~# df -lh
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p2  2.0G  565M  1.4G  30% /
devtmpfs        482M     0  482M   0% /dev
tmpfs           490M     0  490M   0% /dev/shm
tmpfs           490M   13M  478M   3% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           490M     0  490M   0% /sys/fs/cgroup
/dev/mmcblk0p1   50M   13M   38M  26% /boot
```

Enter into system and expend via resize_rootfs.sh

```
root@Orangepi:/usr/local/sbin# resize_rootfs.sh
+ DEVICE=/dev/mmcblk0
+ PART=2
+ resize
+ fdisk -l /dev/mmcblk0
+ grep /dev/mmcblk0p2
+ awk {print $2}
+ start=143360
+ echo 143360
143360
+ set +e
+ fdisk /dev/mmcblk0

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
```

Enter resize_rootfs.sh on command line, the system will expending automatically,

Reboot the system and use df -lh to check whether expending is successful

```
+ set -e
+ partx -u /dev/mmcblk0
+ resize2fs /dev/mmcblk0p2
resize2fs 1.42.13 (17-May-2015)
Filesystem at /dev/mmcblk0p2 is mounted on /; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/mmcblk0p2 is now 3871616 (4k) blocks long.

+ echo Done!
Done!
root@Orangepi:/usr/local/sbin# df -lh
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p2   15G  566M   14G   4% /
devtmpfs        482M     0  482M   0% /dev
tmpfs           490M     0  490M   0% /dev/shm
tmpfs           490M   13M  478M   3% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           490M     0  490M   0% /sys/fs/cgroup
/dev/mmcblk0p1   50M   13M   38M  26% /boot
```

a. Expand file system

i Boot to Linux, umount /dev/sdb1 and /dev/sdb2, if it prompts disk busy, then use fuser to clean the using disk(we will recommend using another Linux booting disk to lead the system).

ii Use fdisk /dev/sdb to adjust the partition size, after into it, enter p, and keep in mind about the initial position of needed extending size partition.

iii Enter d to delete the partition need to change the size(my file system is /dev/sdb2, which is the 2 partition).

iv Enter n to build a new partition, make sure the initial position is the

same as you deleted, and enter the number as you desire.

v Enter w to save the partition data.

vi Use the following command to check the file system(make sure it is a right file system)

e2fsck -f /dev/sdb2

vii Adjust the partition size

resize2fs /dev/sdb2

viii It could mount a disk partition, you could check whether it has changed.

b. Shrink file system

i Boot to Linux, umount /dev/sdb1 and /dev/sdb2, if it prompts disk busy, then use fuser to clean the using disk(we will recommend using another Linux booting disk to lead the system).

ii Use the following command to check the file system(make sure it is a right file system)

e2fsck -f /dev/sdb2

iii Modify the size of file system(Use resize2fs)

resize2fs /dev/sdb2 900M

The "s"after the number represents specifying the size of file system via the sectors(every sector calculated by 512 bite). You could also specify it into K(KB), M(MB), G(GB), etc.

iv Use fdisk /dev/sdb to adjust the partition size, after into it, enter p, and keep in mind about the initial position of needed extending size partition. You need to first delete the partition then build a new one because the fdisk could not modify the size dynamic(you need to calculate the size, it have to enough to contain the file system adjusted in last step).

v Enter d to delete the partition need to change the size(my file system is /dev/sdb2, which is the 2 partition).

vi Enter n to build a new partition, make sure the initial position is the same as you deleted, and enter the number as you desire. Besides, if it is boot-able partition you want to change, note that need to keep the bootable mark in case cannot boot.

The above illustration is using fdisk and resize2fs to modify partition and file system, you could also use gparted. Gparted has graphical interface and it could help you to re-size file system at the same time of re-sizing partition. Goarted is much easier to use and reduce the change to make mistake. For now our official Lubuntu and Raspbian could not use it.

8) eth0 and wlan0 static mac address setting

a. If the system do not use systemd, you could modify rc.local directory and add the following:

```
$ vim /etc/rc.local
MAC=00:e0:4c:a1:2b:d4
ifconfig wlan0 down
ifconfig wlan0 hw ether $MAC
ifconfig wlan0 up
dhclient &
```

After rebooting, you could use ifconfig to check whether mac address has changed.

b. If the system used systemd, you also need to add the following besides the above steps:

```
$ cd /etc/systemd/system/
```

$ vim change_mac_address.service (You could name the server, format just like the following)

```
[unit]
Description=Change OrangePi Wifi mac address

[Service]
ExecStart=/etc/rc.local
RemainAfterExit=yes

[Install]
sWantedBy=multi-user.target
```

```
$ systemctl enable change_mac_address.service
```

Modify mac address of eth0 is same as modifying wlan0's, just need to replace wlan0 into eth0.

9) Orange Pi Android root

There is defaulted with root permission on Android pre-installed, but lacking authorization management software. The following is how to add authorization management software.

You need to have UsbModeSwitch.apk and UPDATE-SuperSU-v2.46.zip, install kingroot and make sure OTG on Orange Pi could connect to PC.

a. Open adb debug mode

Use U disk or card reader to install UsbModeSwitch.apk into Orange Pi OS and open it, tick "enable usb device mode" and use debug cable to

connect OTG port and PC (make sure it is micro usb-cable in case other cables could not be recognized). Normally PC would search and install adb driver software automatically. If PC failed to install, you could install PC version's Peasecod to install the driver software.

b. After connected Orange PI and PC, open command mode of PC, enter related command of adb(you need to install adb debug command, which Peasecod has adb command). Here is the command:

adb remount
adb shell

windows(win+r) command line enter into command mode, then enter into kingroot directory and execute the following steps:

adb shell
root@rabbit-p1:/ # mkdir /tmp
root@rabbit-p1:/ # cd /system/bin
root@rabbit-p1:/ # mount -o remount, rw /system
root@rabbit-p1:/system/bin # ln -s busybox-smp unzip
Logout adb shell Mode
root@rabbit-p1:/exit (Or Ctrl + C)
Unzip UPDATE-SuperSU-v2.46.zip
You will obtain META-INF/com/google/android/update-binary and put it into specific catalog.
adb push /path/UPDATE-SuperSU-v2.46.zip /data/local/tmp path is file's path
adb push /path/ update-binary /data/local/tmp
adb shell
root@rabbit-p1:/ #cd /data/local/tmp
root@rabbit-p1:/ #sh update-binary 0 1
/data/local/tmp/UPDATE-SuperSU-v2.46.zip

.....
.....

After executed scripts, enter reboot command and reboot it, you could use the device authorization management software normally.

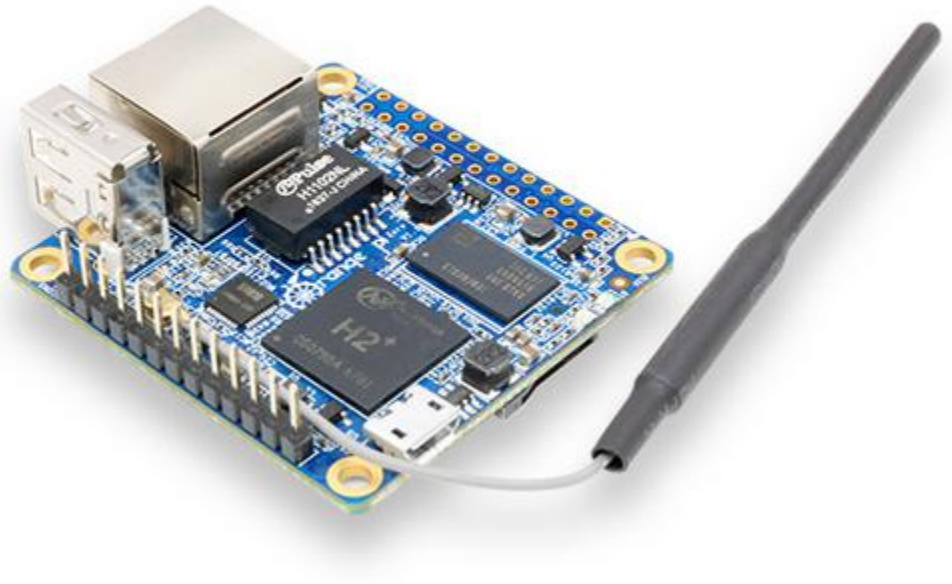
After rebooted, there might be no super administrator icon, you need to delete the desk configuration file and reboot the board.

III. Linux Kernel Source Code Compilation

In order to support the rapid development of the project, we are writing this sections for project configuration options to the binary file. When the system is running, it can get the information of the system running by reading the binary file, which can greatly simplify the time of project development.

This manual describes how to use the binary file to speed up the development of the project.

Hardware: Orange Pi development board*1, Card reader*1, TF card*1, power supply*1



Note: In the following sections, * indicates wild-cards, you need to fill in the actual values according to their file storage path.

1. Download Linux Source Code

You could download the source code from the official website:
http://www.orangepi.org/downloadresources/

Subsection and compress the file, then unzip it after finish downloaded:

buildroot: Project compilation script
brandy: gcc-linaro, boot and uboot source code and open source cross compiler tool
linux-3.4: Kernel source code
tools: Tools of project compilation
build.sh: compilation script

2. Compile Project Source Code

You need to compile the entire project while it is your first time to use the source code. You can use the following commands in the /lichee directory to complete the project:

- Enter into content of lichee, command

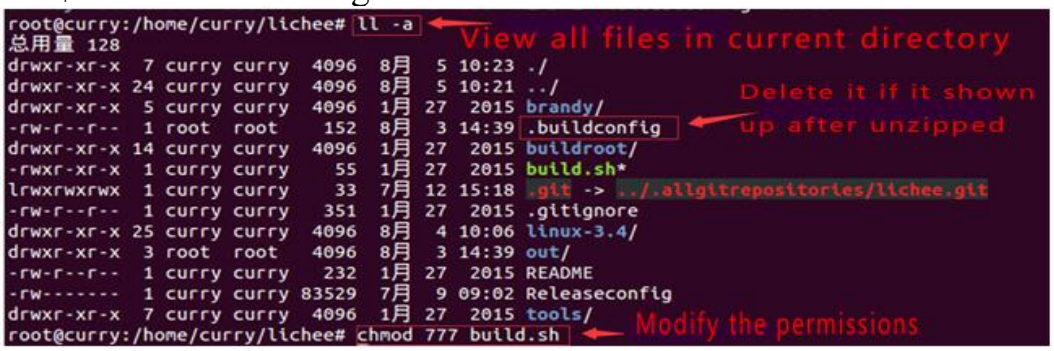
  $ ll -a

Check if there is an executable permission on build.sh, if not, modify the permissions

  $ chmod 755 build.sh

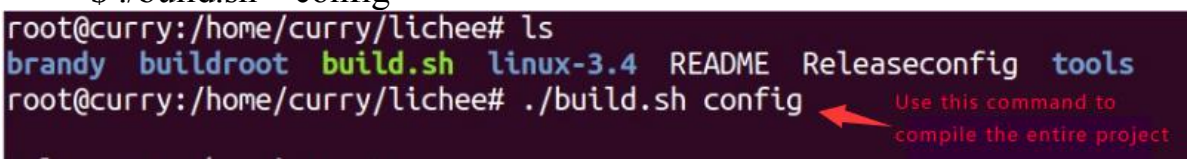
- If there is .buildconfig after commanded ll –a, delete it

  $ rm –rf .buildconfig



- Use the following command to compile the entire project

  $ ./build.sh config



At this point the system will prompt the choice of the chip, for OrangePi, select sun8iw7p1

At this point, the system will be prompted the choice of the board, for the OrangePi, select dragonboard，dolphin and dolphin-p2

```
curry@curry:$ sudo ./build.sh config

Welcome to mkscript setup progress
All available chips:
   0. sun8iw6p1
   1. sun8iw7p1
   2. sun8iw8p1
   3. sun9iw1p1
Choice: 1
All available platforms:
   0. android
   1. dragonboard
   2. linux
Choice: 1
All available business:
   0. dolphin
   1. secure
   2. karaok
Choice: 0
LICHEE_BUSINESS=dolphin
using kernel 'linux-3.4':
All available boards:
   0. dolphin-cmcc-wasu-p1
   1. dolphin-karaok
   2. dolphin-p1
   3. dolphin-p1-secure
   4. dolphin-p2
   5. dolphin-perf
Choice: 4
```

Appear this interface indicates waiting for the compiler.

```
INFO: ----------------------------------------
INFO: build lichee ...
INFO: chip: sun8iw7p1
INFO: platform: dragonboard
INFO: business:
INFO: kernel: linux-3.4
INFO: board: dolphin-p1
INFO: output: out/sun8iw7p1/dragonboard/dolphin-p1
INFO: ----------------------------------------
INFO: build buildroot ...
installing external toolchain
please wait for a few minutes ...
```

Wait fifteen minutes or so, compile complete.

```
make[1]:正在离开目录 `/home/curry/Downloads/lichee/buildroot/target/
generating rootfs...
blocks: 85M -> 112M
Creating filesystem with parameters:
    Size: 117440512
    Block size: 4096
    Blocks per group: 32768
    Inodes per group: 7168
    Inode size: 256
    Journal blocks: 1024
    Label:
    Blocks: 28672
    Block groups: 1
    Reserved block group size: 7
Created filesystem with 3653/7168 inodes and 23020/28672 blocks
e2fsck 1.42.9 (4-Feb-2014)
success in generating rootfs
Build at: 2016年 08月 03日 星期三 14:55:30 CST
INFO: build rootfs OK.
----------------------------------------
build sun8iw7p1 dragonboard  lichee OK
----------------------------------------
```

Indicates success

3. Update the Kernel Image File and Replace Library

- After compilation is finished, the following files will be generated in the directory：

libs： lichee/out/sun8iw7p1/android/common/lib/modules/3.4.39

Download image from official website：

http://www.orangepi.org/downloadresources/

- Write the image：

$ sudo dd bs=4M if=*.img of=/dev/sdb

- Pull out the card reader, and then insert it again.

Copy the kernel image file generated by the compiler to the first partition (boot partition)

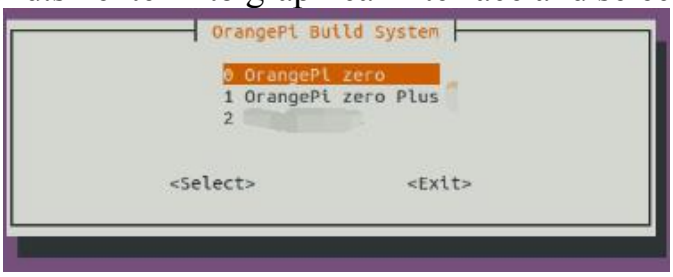
Copy the lib library which generated after compilation to the second partition (rootfs partition)

We would suggest using compilation system on github of official website.

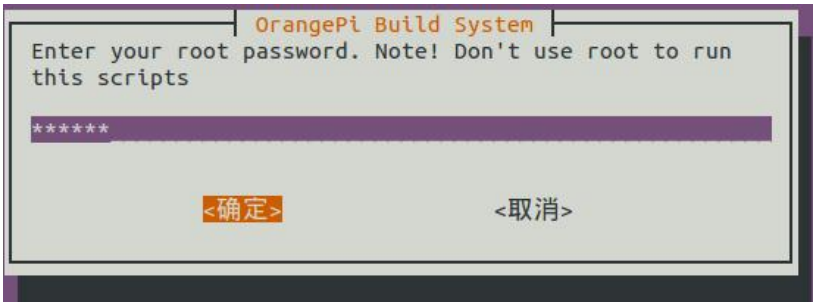
```
curry@curry:$ ls
build.sh  external  kernel  output  scripts  toolchain  uboot
```

build.sh Execute script into the graphical interface of compilation
extenal Inside are patch and some configuration kernel file
output File generated
script Script compiled
toolchain Cross compiler location
uboot uboot source code

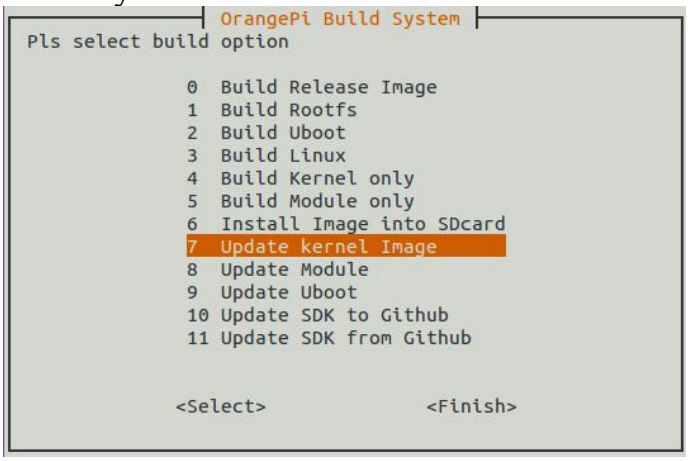
Execute./build.sh enter into graphical interface and select Zero



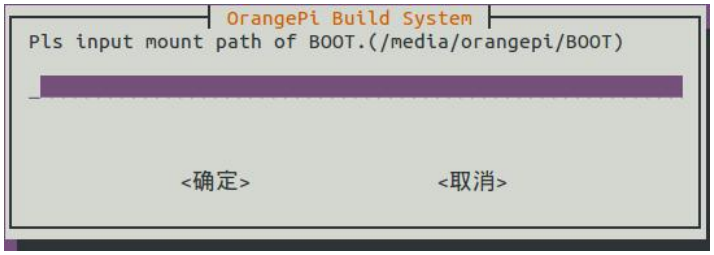
Enter password of root



Update Kernel directory and module



Select corresponding file directory and update uImage and modules



IV. Android Kernel Source Code Compilation

Hardware: Orange Pi development board*1, Card reader*1, TF card*1, power supply*1

Software

Linux host computer, which hard disk space at least 50G (to meet a fully compiled need)

Linux host computer needs:

Version 2.7.3 of Python;

Version 3.81-3.82 of GNU Make;

Version 1.7 or higher version of Git.

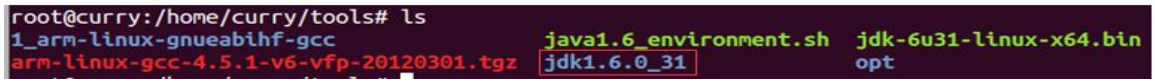
1. Install JDK

The following will illustrate jdk1.6 installation, it would be same for jdk1.7 installation.

- Download and install JDK, you will obtain jdk-6u31-linux-x64.bin

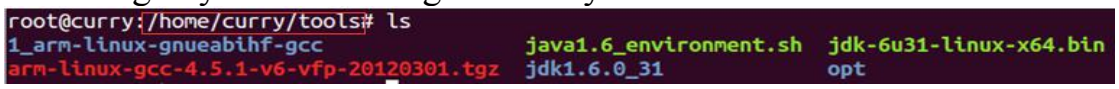
- Modify the permission of jdk-6u31-linux-x64.bin, which has no prior permission
- $./jdk-6u31-linux-x64.bin

It will generate a folder:

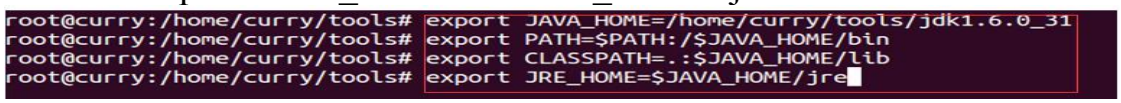


- Input at terminal

Note that JAVA_HOME is the name of the current directory, you need to fill in according to your own storage directory.




```
$ export  JAVA_HOME=*/jdk1.6.0_31
$ export  PATH=$PATH:/$JAVA_HOME/bin
$ export  CLASSPATH=.:$JAVA_HOME/lib
$ export  JRE_HOME=$JAVA_HOME/jre
```



- Command line input Jav and press tab to see whether it can auto completion (Java), which indicates it can successfully installed

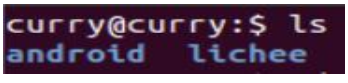
2. Install Platform Supported Software

```
$ sudo apt-get install git gnupg flex bison gperf build-essential \
  zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \
  libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \
  libgl1-mesa-dev g++-multilib mingw32 tofrodos \
  python-markdown libxml2-utils xsltproc zlib1g-dev:i386
$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1
  /usr/lib/i386-linux-gnu/libGL.so
```

3. Download Android Source Package

Download website: http://www.orangepi.org/downloadresources/

Then you will obtain the following directories:



4. Install Compiler Tool Chain

The compiler tool chain has been integrated in Android SDK. Tool chain is on: lichee/brandy/gcc-linaro/ of Android SDK(already exist)

```
brandy  buildroot  build.sh  linux-3.4  README  tools
root@curry:/home/curry/OrangePi/android/lichee# cd brandy/gcc-linaro/bin/
root@curry:/home/curry/OrangePi/android/lichee/brandy/gcc-linaro/bin# ls
arm-linux-gnueabi-addr2line     arm-linux-gnueabi-gprof
arm-linux-gnueabi-ar            arm-linux-gnueabi-ld
arm-linux-gnueabi-as            arm-linux-gnueabi-ld.bfd
arm-linux-gnueabi-c++           arm-linux-gnueabi-ldd
arm-linux-gnueabi-c++filt       arm-linux-gnueabi-ld.gold
arm-linux-gnueabi-cpp           arm-linux-gnueabi-nm
```

5. Compile Lichee Source Code

There are Android and Lichee after unzipped the package, enter the directory of Lichee:

$ cd lichee

$ ./build.sh lunch

Select sun8iw7p1

Print information of successful compilation

```
sun8iw7p1 compile Kernel successful

INFO: build kernel OK.
INFO: build rootfs ...
INFO: skip make rootfs for android
INFO: build rootfs OK.
----------------------------------------
build sun8iw7p1 android dolphin lichee OK
```

6. Compile Command of Android Code

Input the command:

$ cd android

$ source ./build/envsetup.sh

```
root@curry:/home/curry/OrangePi/android/android# source ./build/envsetup.sh
including device/generic/armv7-a-neon/vendorsetup.sh
including device/generic/x86/vendorsetup.sh
including device/generic/mips/vendorsetup.sh
including device/asus/tilapia/vendorsetup.sh
including device/asus/grouper/vendorsetup.sh
including device/asus/deb/vendorsetup.sh
including device/asus/flo/vendorsetup.sh
```

$ lunch dolphin_fvd_p1-eng # Select the scheme number

```
curry@curry:$ lunch

You're building on Linux

Lunch menu... pick a combo:
     1. rabbit_cmccwasu_p1-eng
     2. rabbit_gms_p1-eng
     3. rabbit_fvd_p1-eng
     4. rabbit_aosp_p1-eng
     5. rabbit_aosp_p1-user
     6. rabbit_fvd_p1-user
     7. rabbit_fvd_p1-userdebug
     8. rabbit_aosp_perf-eng
     9. jaws_optimus-eng
    10. cheetah_fvd_p1-eng
    11. cheetah_fvd_p1-user
    12. cheetah_cts_p1-eng
    13. cheetah_cts_p1-user
    14. cheetah_cmcc_p1-eng
    15. cheetah_cmcc_p1-user
    16. molly-eng
    17. jaws_tvd_p1-eng
    18. rabbit_32bit_fvd_p1-eng
    19. cheetah_perf-eng
    20. eagle_fvd_p1_normal-eng
    21. eagle_fvd_p1_secure-eng
    22. dolphin_fvd_p1-eng
    23. dolphin_fvd_p1-user

Which would you like? [aosp_arm-eng] 10
```

$ extract-bsp # Copy the kernel and the drive module

```
curry@curry:$ extract-bsp
/expansion/xspace/OrangePi/Android/H5/zeroplus/android/device/*/cheetah-p1/bImage copied!
/expansion/xspace/OrangePi/Android/H5/zeroplus/android/device/*/cheetah-p1/modules copied!
curry@curry:$ make
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=5.1
TARGET_PRODUCT=cheetah_fvd_p1
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
```

$ make The rear values of # is for the simultaneous compilation process, dependent on the host configuration

```
Creating filesystem with parameters:
    Size: 805306368
    Block size: 4096
    Blocks per group: 32768
    Inodes per group: 8192
    Inode size: 256
    Journal blocks: 3072
    Label:
    Blocks: 196608
    Block groups: 6
    Reserved block group size: 47
Created filesystem with 1393/49152 inodes and 79017/196608 blocks
+ '[' 0 -ne 0 ']'
Install system fs image: out/target/product/dolphin-fvd-p1/system.img
out/target/product/dolphin-fvd-p1/system.img+out/target/product/dolphin-fvd-p1/obj/PACKAGING
/recovery_patch_intermediates/recovery_from_boot.p maxsize=822163584 blocksize=4224 total=31
3479604 reserve=8308608
```

$ pack #Packaged into firmware

```
Dragon execute image.cfg SUCCESS !
----------image is at----------

/home/curry/OrangePi/android/lichee/tools/pack/sun8iw7p1_android_dolphin-p1_uart0.img

pack finish
```

$ cd */lichee/tools/pack/

```
root@curry:/home/curry# cd /home/curry/OrangePi/android/lichee/tools/pack/
root@curry:/home/curry/OrangePi/android/lichee/tools/pack# ls
chips  common  createkeys  out  pack  parser.sh  pctools  sun8iw7p1_android_dolphin-p1_uart0.img
root@curry:/home/curry/OrangePi/android/lichee/tools/pack#
```

V. Use Project Configuration Files

1. sys_config.fex Introduction

Configure hardware: sys_config.fe

The sys_config.fex is a binary configuration file that used by the SOC kernel driver or LiveSuit for a particular target board, including how to set up a variety of peripherals, ports, and I/O which based on the target version.

For OrangePi, the location of the project configuration document is：
lichee/tools/pack/chips/sun8iw7p1/configs/dolphin-p1/sys_config.fex

Copy the file to the directory of /lichee, use command:

```
$ cd ./lichee
$ cp ./tools/pack/chips/sun8iw7p1/configs/dolphin-p1/sys_config.fex ./
```

You could personalized configuration of sys_config.fex according to sysconfig1.fex_manul_linux_BSP_v0.4.pdf.
Direcotory of sysconfig1.fex_manul_linux_BSP_v0.4.pdf is /lichee/buildroot/docs.

2. Examples

1) Modify the output mode into tv

- tv-out out, the output type of tv0 is invalid, you need to set the output type of tv1 into pal.

  Modify defaulted enable display output configuration into tv

```
[tv0]
used                            = 1
tv_dac_used                     = 1
dac_src0                        = 0
dac_type0= 0
interface= 1
[tvout_para]
tvout_used= 1
tvout_channel_num= 1

[disp]
disp_init_enable= 1
disp_mode= 1
screen0_output_type= 2
screen0_output_mode= 11
```

screen1_output_type= 2
screen1_output_mode= 11
dev0_output_type = 4
dev0_output_mode = 4
dev0_screen_id = 0
dev0_do_hpd = 1
dev1_output_type = 2
dev1_output_mode = 11

Modify sys_confi and replace it when it generated OrangePiH5.dtb. If would be faster if use compilation system on githug. About compilation you could refer to the charter of Linux Compilation.

2) Loading tv.ko module automatically after booted

Enter /lib/ directory, enter command:
depmod -a
Add one more line on /etc/modules
tv
It would be tv out after booted

- Capacitance touch panel (capacitor tp)

Configuration Item	Configuration Meaning
ctp_used=xx	Whether turn on capacitance touch panel, if so set the value as 1, and vice verso 0.
ctp_name =xx	Indicates the control scheme used in the specified scheme, for now there are: "ft5x_ts" or "Goodix-TS".
ctp_twi_id=xx	Used for selecting i2c adapter, there are 0 and 2.
ctp_twi_addr =xx	Indicates the device address of i2c, it is related to the specific hardware.
ctp_screen_max_x＝xx	Maximum coordinates of the X axis of the touch panel
ctp_screen_max_y=xx	Maximum coordinates of the Y axis o the touch panel
ctp_revert_x_flag=xx	Whether needed to flip the X coordinates, if so then set 1, and vice verso 0.
ctp_revert_y_flag=xx	Whether needed to flip the Y coordinates, if so then set 1, and vice verso 0.
ctp_int_port=xx	GPIO configuration of the interrupt signal of

	capacitive touch panel
ctp_wakeup=xx	GPIO configuration of the wake-up signal of capacitive touch panel
ctp_io_port=xx	Capacitive screen IO signal, currently share with interrupt signal common pin

Configuration samples:

```
ctp_used        = 1
ctp_name        = "ft5x_ts"
ctp_twi_id      = 2
ctp_twi_addr    = 0x70
ctp_screen_max_x    = 800
ctp_screen_max_y    = 480
ctp_revert_x_flag   = 0
ctp_revert_y_flag   = 0
ctp_int_port        = port:PH21<6><default>
ctp_wakeup          = port:PB13<1><default><default><1>
ctp_io_port         = port:PH21<0><default>
```

Note: If you want to support the new capacitive touch IC, you need to combine the configuration of the BSP A10 layer, which should be based on the original capacitive touch IC code, to make the appropriate changes. Specifically, 1) ctp_twi_id should be consistent with the hardware connection in sys_config; 2) In the drive part of the code: the use of twi from the device name + address should be consistent with the ctp_name and ctp_twi_addr in sys_config configuration. At the same time, the other sub configuration in sysconfig should also be properly configured, these configurations should be corresponding processing in the program.

VI. OrangePi Driver development

In order to help developers become more familiar with OrangePi, this manual describes how to use simple device driver modules and applications on the development board.

Hardware: Orange Pi development board*1, Card reader*1, TF card*1, power supply*1

1. Device Driver and Application Programming

1) Application Program (app.c)

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>

int main(int argc, char *argv[])
{
	int cnt, fd;
	char buf[32] = {0};
	if(argc != 2)
	{
		printf("Usage : %s </dev/xxx>\r\n", argv[0]);
		return -1;
	}

	fd = open(argv[1], O_RDWR);
	if(fd < 0)
	{
		printf("APP Error : open device is Failed!\r\n");
		return -1;
	}
	read(fd, buf, sizeof(buf));
	printf("buf = %s\r\n", buf);
	close(fd);
	return 0;
}
```

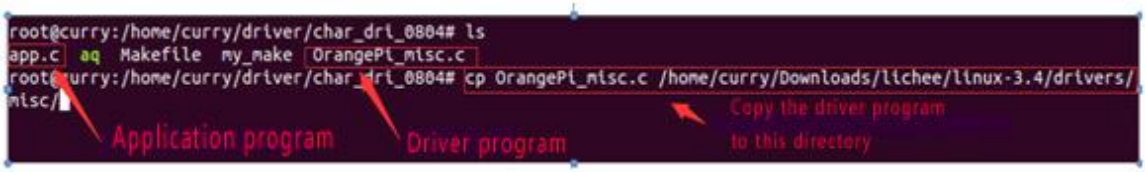
2) Driver Program (OrangePi_misc.c)

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/miscdevice.h>
#include <linux/init.h>
#include <asm-generic/uaccess.h>

static int orangepi_open(struct inode *inodp, struct file *filp)
{
	return 0;
}

static ssize_t orangepi_read(struct file *filp, char __user *buf, size_t
count, loff_t *offset)
{
	char str[] = "Hello World";
	copy_to_user(buf, str, count);
	return 0;
}

static struct file_operations tOrangePiFops = {
	.owner = THIS_MODULE,
	.open  = orangepi_open,
	.read  = orangepi_read,
};

static struct miscdevice OrangePi_Misc = {
	.minor = 255,
	.name  = "orangepimisc",
	.fops  = &tOrangePiFops,
};
```

```
static int __init OrangePi_misc_init(void)
{
	int ret;
	printk("func : %s, line : %d\r\n", __func__, __LINE__);

	ret = misc_register(&OrangePi_Misc);
	if(ret < 0){
		printk("Driver Error : misc_register is Failed!\r\n");
		return -1;
	}
	return 0;
}

static void __exit OrangePi_misc_exit(void)
{
	int ret;
	printk("func : %s, line : %d\r\n", __func__, __LINE__);
	ret = misc_deregister(&OrangePi_Misc);
	if(ret < 0){

		printk("Driver Error : misc_register is Failed\r\n");
	}
}

module_init(OrangePi_misc_init);
module_exit(OrangePi_misc_exit);
```

2. Compile device driver

Copy the OrangePi_misc.c to the */lichee/linux-3.4/driver/misc directory:



Enter to */lichee/linux-3.4/drivers/misc/, and modify makefile

Modify Makefile on currently file, shown as following:

```
43 obj-$(CONFIG_SPEAR13XX_PCIE_GADGET) += spear13xx_pcie_gadget.o
44 obj-$(CONFIG_VMWARE_BALLOON)    += vmw_balloon.o
45 obj-$(CONFIG_ARM_CHARLCD)   += arm-charlcd.o
46 obj-$(CONFIG_PCH_PHUB)      += pch_phub.o
47 obj-y               += ti-st/
48 obj-$(CONFIG_AB8500_PWM)    += ab8500-pwm.o
49 obj-y               += lis3lv02d/
50 obj-y               += carma/
51 obj-$(CONFIG_USB_SWITCH_FSA9480) += fsa9480.o
52 obj-$(CONFIG_ALTERA_STAPL)  +=altera-stapl/
53 obj-$(CONFIG_MAX8997_MUIC)  += max8997-muic.o
54 obj-$(CONFIG_WL127X_RFKILL) += wl127x-rfkill.o
55 obj-$(CONFIG_SENSORS_AK8975)    += akm8975.o
56 obj-$(CONFIG_SUNXI_VIBRATOR)    += sunxi-vibrator.o
57 obj-$(CONFIG_SUNXI_BROM_READ)   += sunxi_brom_read.o
58 obj-$(CONFIG_NET)     += rf_pm/
59 obj-$(CONFIG_ORANGEPI_MISC) += OrangePi_misc.o
```

Re-modify Makefile

There is Kconfig on the same sibling folders with Makefile. Each Kconfig respectively describes the the source directory file related kernel configuration menu. In the kernel configuration making menuconfig, it read from the Kconfig config menu and the user configuration saved to the config. In the kernel compile, the main Makefile by calling this.Config could know the user's configuration of the kernel.

Kconfig is corresponding to the kernel configuration menu. Add a new driver to the kernel source code, you can modify the Kconfig to increase the configuration menu for your drive, so you can choose whether the menuconfig driver was compiled or not.

```
config SUNXI_BROM_READ
    tristate "Read the BROM infomation"
    depends on ARCH_SUN8I
    default n
    ---help---
     This option can allow program access brom space by the file node.

config ORANGEPI_MISC
    tristate
    default n
```

Modify Kconfig

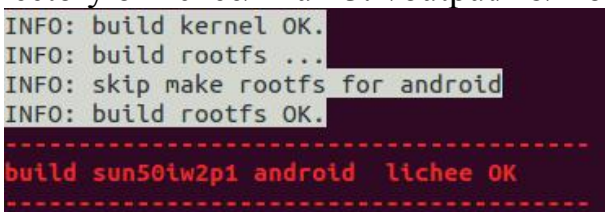
Back to the source code directory:

```
root@curry:/home/curry/Downloads/lichee# cd /home/curry/Downloads/lichee/
```

Back to the source code directory

$ ./build.sh

After compiled the kernel, there will be an orangepi_misc.ko file generated on the directory of lichee/linux-3.4/output/lib/modules/3.4.39



There is a .ko module which generated after compiled of OrangePi_misc.c on */lichee/linux-3.4/output/lib/modules/3.4.39/

Insert U disk (please note the SD card should have been written image) if the SD card system is mounted to the directory / dev / SDB, SD card will have two sub mount points, respectively are / dev / sdb1 and /dev/sdb2. Two partition of SD card will automatically mount to the PC /media/ directory, the first partition is the boot partition and the second partition is the rootfs partition.

Copy the OrangePi_misc.ko file to /media/*/lib/modules/3.4.39.

$ cp OrangePi_misc.ko /media/*/lib/modules/3.4.39

3. Cross compiler Application Program

Here will take arm-linux-gnueabihf-gcc as an example. Check whether there is the cross compiler, if not, then download and install it.

$ arm-linux-gnueabihf-gcc -v

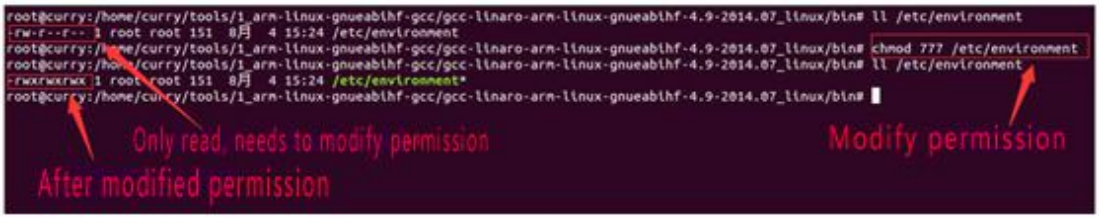
While compiling the application, you will fill that you need the cross compiler arm-linux-gnueabihf-gcc, download and install it.

Unzip the downloaded file and enter the the directory

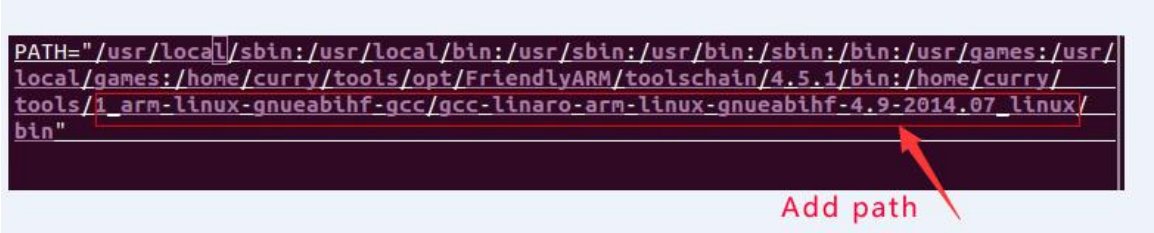
Check the information after entering bin directory

pwd hows the path and export it into the whole project

$ ll /etc/environment shows that the file can only read, need to modify permissions
$ chmod 755 /etc/environment
Modify permission



Add the path to the whole environment variable



Compile the application with cross compiler

$ arm-linux-gnueabihf-gcc app.c –o aq
There will be an ap application generated in the directory, copy it to the development board file system(on the rootfs directory of /home/orangepi/)
$ cp aq /media/*/home/orangepi/

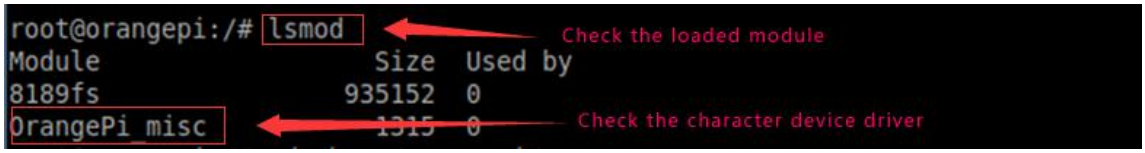
4. Running Driver and Application

Removed the SD card and inserted it into the development board and power on.

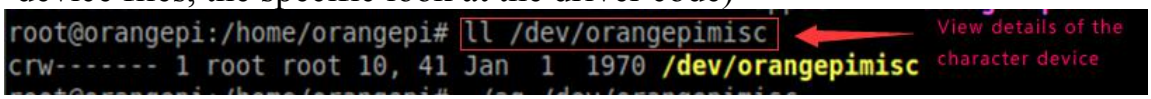
You need to switch to root users and load module driver module to the development board first.

$ insmod /lib/modules/orangepi.ko

$ lsmod To check whether it is loaded



$ ll /dev/orangepimisc(Miscellaneous equipment automatically generated device files, the specific look at the driver code)



Executive application (note the use of the application, the specific check at the code)

$ ./aq /dev/orangepimisc

VII. Using Debug tools on OrangePi

Hardware: Orange Pi development board*1, Card reader*1, TF card*1, power supply*1

TTL to USB cable

1. Operation Steps on Windows

In order to get more debugging information in the project development process of using OrangePi, OrangePi default support for serial information

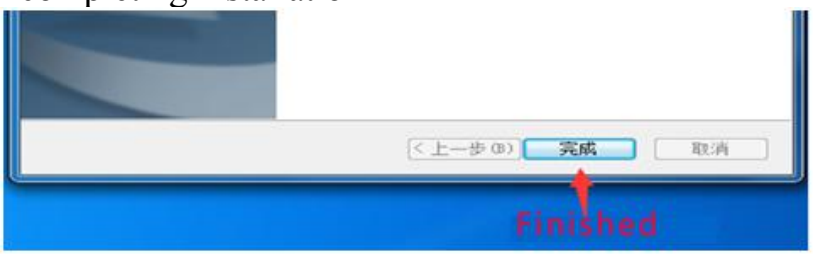
debugging. For developers, you can simply get the serial port debugging information with the materials mentioned above. The host computer using different serial debugging tools are similar, basically can reference with the following manual for deployment. There are a lot of debugging tools for Windows platform, the most commonly used tool is putty. This section takes putty as an example to explain the deployment.

1) Install USB driver on Windows

- Download and unzip the latest version of driver PL2303_Prolific_DriverInstaller_v130.zip

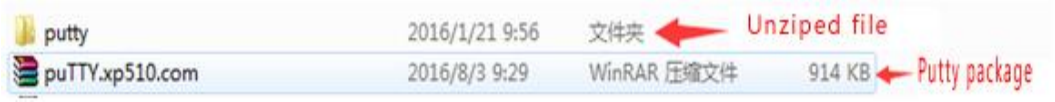
- Choose application installation as Administrator

- Wait for completing installation



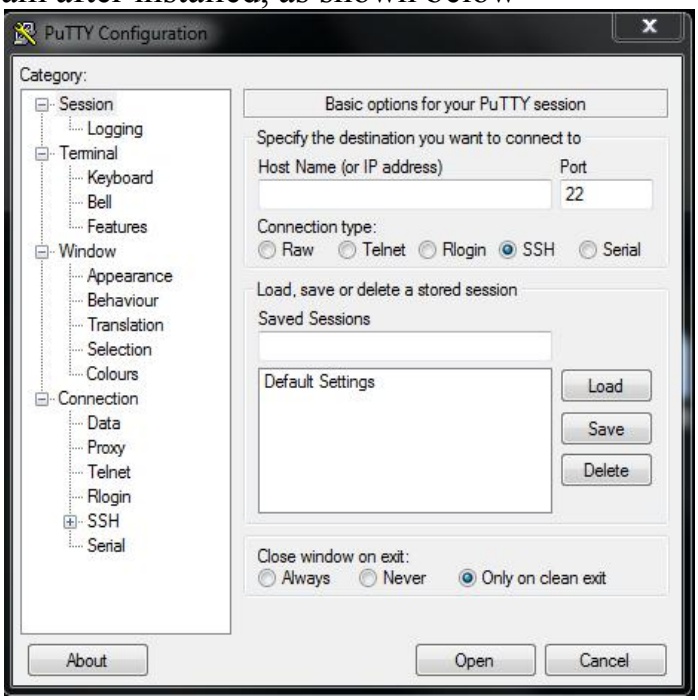
2) Install putty on Windows

- Download putty installation package



- Unzip and install

- Open program after installed, as shown below

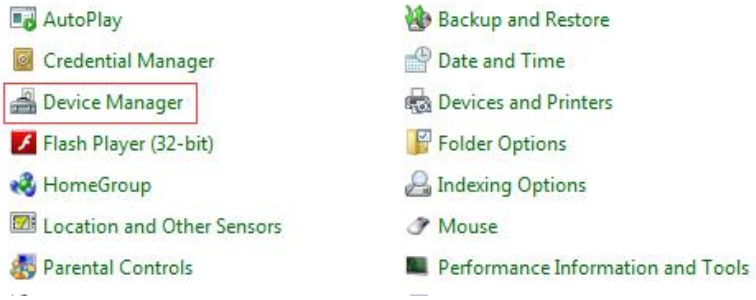


3) Connecting method

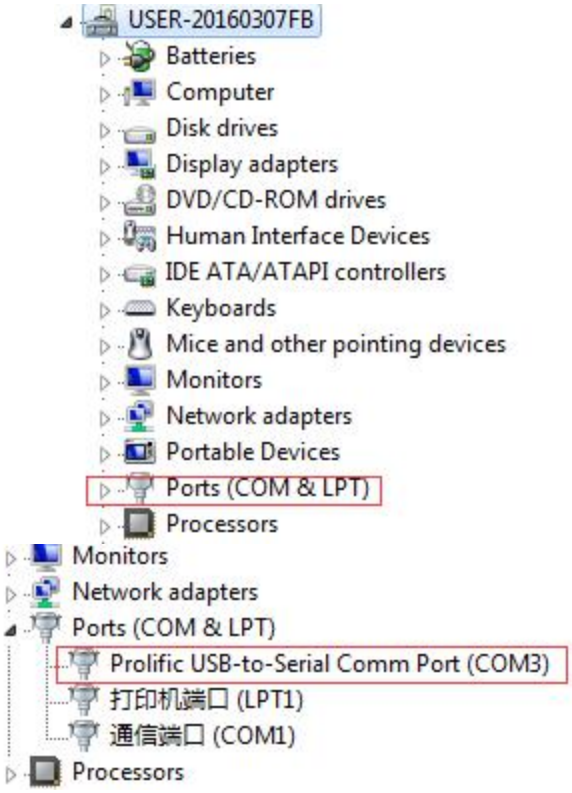
Use the TTL to the serial port cable, one end connected to OrangePi, the other end connected to PC

4) Equipment information acquisition

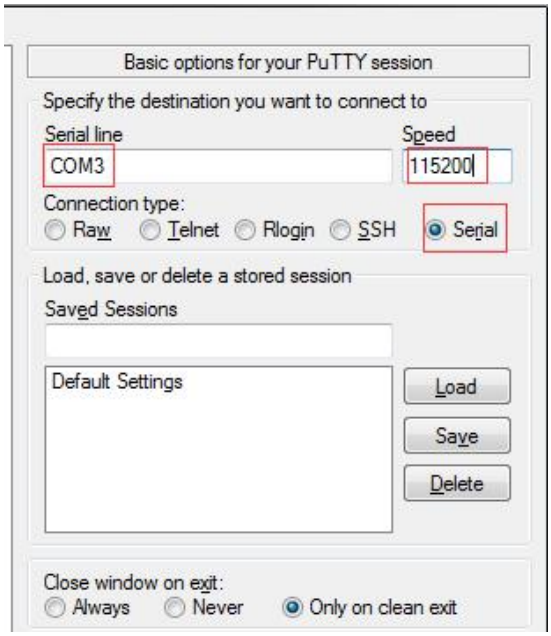
- *Start* menu select *control panel*



- Click on the *device manager* to check the *port numbe*r



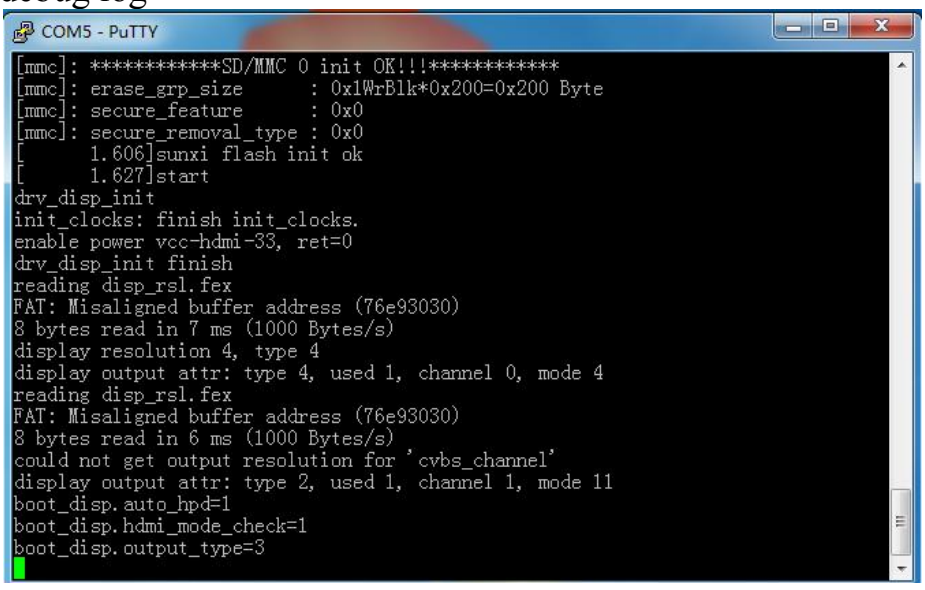
5) Putty Configuration



Serial port should set to the corresponding port number (COM5), the speed should set to 115200

6) Serial Debug Port

Power on and boot OrangePi, the serial port will automatic print debug log

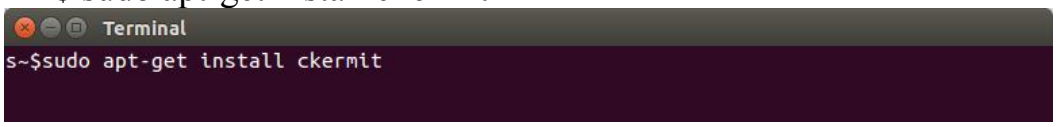


2. Operation Steps on Linux

There are Minicom and Kermit serial debugging tools for Linux, this section will take Kermit as an example to have an illustrate.

1) Install Kermit

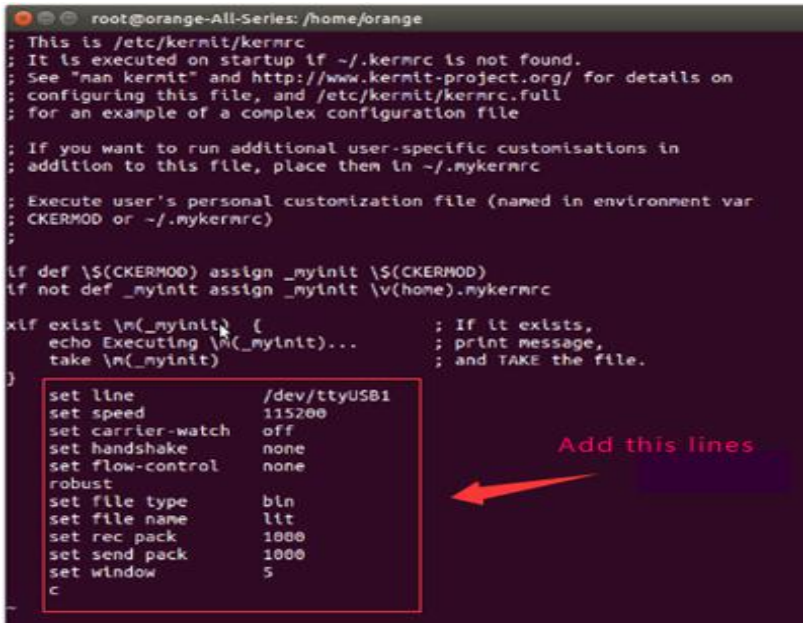
- Install the Kermit by execute command:
  $ sudo apt-get install ckermit



- Configurate Kermit
  $ sudo vi /etc/kermit/kermrc

- Add lines：

```
set line          /dev/ttyUSB1
set speed             115200
set carrier-watch   off
set handshake   none
set flow-control     none
robust
set file type      bin
set file name         lit
set rec pack          1000
set send pack        1000
set window            5
```



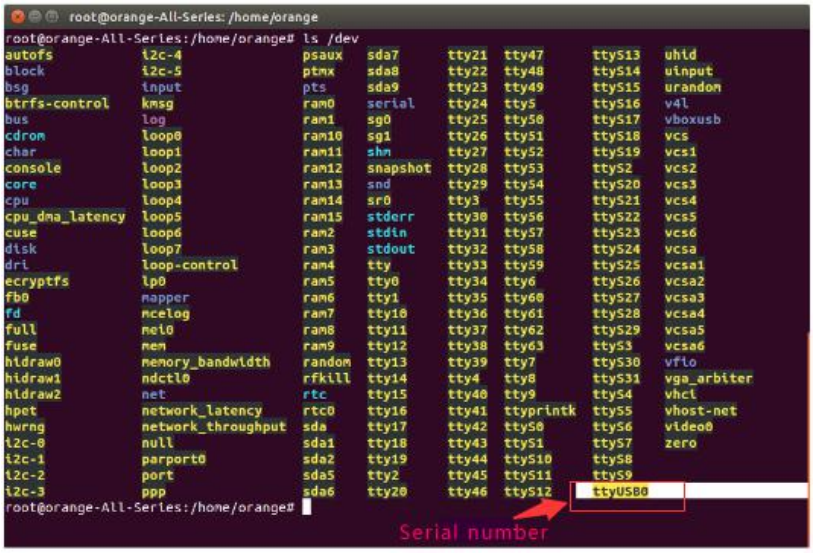
2) Connecting method

Use the TTL to the serial port cable, one end connected to OrangePi, the other end connected to PC

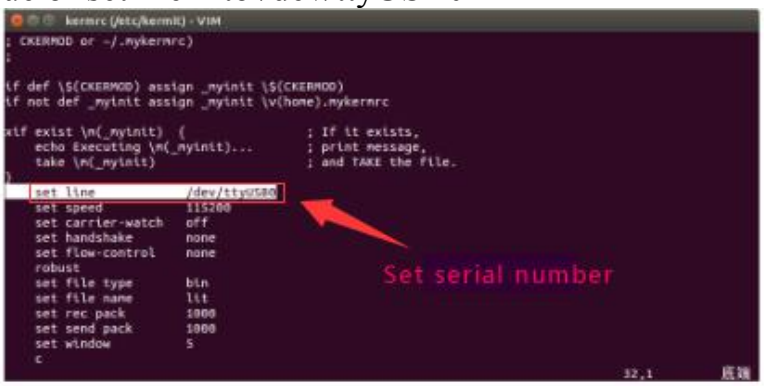
3) Equipment information acquisitio

Input command in the PC terminal to check the device number of TTL to the serial cable

$ ls /dev/

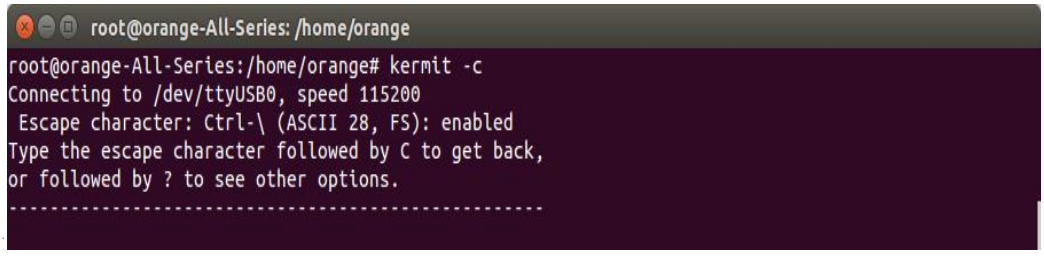


- It can be seen from the figure that TTL to the serial port cable is identified as ttyUSB0, configure the /ect/kermit/kermitc file, update the serial port information.
  $ sudo vi /etc/kermit/kermitc
- Set the value of setline into /dev/ttyUSB0



4) Start debug

- Input command in the host computer terminal, enter the Kermit mode:
  $ sudo kermit –c



- ower on and boot OrangePi, the serial port will automatic print debug log