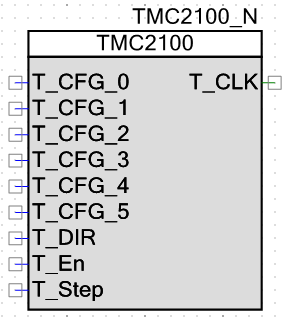
Trinamic TMC2100 Component for PSOC Creator



**TO DO:**

* Create a component customizer in order to enable/disable the chopper config pins, since these pins are usually set by the hardware, it would be useful to disable them in order to give the user more GPIOs. This may require modifying some of the API’s.
* Bring Component documentation up to Cypress’s standards.
* Verify that the clock structure translates on different architectures. PSOC 5LP works, need to verify psoc 4 and probably psoc 6 as well when it comes out.
* Verify the clock output option works……

**Purpose:**

The Trinamic TMC2100 is a standalone stepper motor driver for 2 phase stepper motors. This document details a custom component for PSOC Creator which allows the user to easily interface the TMC2100 with Cypress's PSOC line of microcontrollers.

**API Documentation:**

**Note:** These API’s follow the Cypress naming convention, so if you have a component called Stepper, then you would call the TMC2100\_stop API as Stepper\_TMC2100\_Stop().

* TMC2100\_Stop() – Stops the stepper’s rotation. Doesn’t release the stepper motor, so the motor will continue to draw a significant amount of current.
* TMC2100\_Release(int TMC\_Lock) – Releases the stepper motor. Should not be called while the motor is rotating. Can also be used to lock the motor by setting tmc\_lock to 0.
* TMC2100\_Stop\_and\_Release()- Calls the TMC2100\_Stop() and TMC2100\_Release() functions.
* TMC2100\_Config( int TMC\_CLK\_EN, int Current\_sensing, int Microsteps, int Interpolation, int Chopper\_mode)- Configures the TMC2100’s settings. The Microsteps parameter is particularly important since it affects several other api’s. The parameters are as follows:
  + TMC\_CLK\_EN- Allows the PSOC to drive the clock input of the TMC2100. Generally not recommended, since the internal oscillator on the TMC2100 is good enough for most applications.
  + Current\_sensing- Determines how the TMC2100 senses current. This can generally be set in hardware, so this setting is not always necessary.
  + Microsteps- Determines whether the TMC2100 makes full, half, quarter or microsteps.
  + Interpolation-Determines whether the TMC2100 uses interpolation or not.
  + Chopper\_mode- Determines whether the stepper used spreadCycle or stealthChop. For more information on these terms, see the TMC2100 datasheet.
* TMC2100\_Chopper\_Config(int Off\_time, int Chopper\_hysterisis, int Blank\_Time)- Configures the chopper settings. Since these can be set in hardware and are rarely changed, this function rarely has to be called. Note each of these variables can be set to either LOW, MEDIUM, or HIGH and these values are defined in the API.h file.
* TMC2100\_Initialize()- Calls the TMC2100\_Chopper\_Config() and TMC2100\_Config() functions.
* TMC2100\_Start(int user\_steps, int Standstill\_power\_down, int step\_period , int direction)- Starts the stepper’s rotation. The variables are as follows
  + User\_steps-Number of steps/halfsteps/quartersteps/usteps the user wants the stepper to move. Which setting, (full steps, halfsteps etc…) depends on what value the user set for Microsteps in the TMC2100\_Config function. When User\_steps = 0, the motor rotates continuously.
  + Standstill\_power\_down- Used to set the standstill power down functionality of the TMC2100 chip. See the TMC2100 datasheet for more information about this functionality.
  + Step\_period- Amount of time between steps, essentially sets the stepper’s speed.
  + Direction-Sets the direction of rotation.
* TMC2100\_Degrees\_to\_Steps( double degrees, int steps\_per\_revolution, int Microsteps) – Given how many full steps per revolution the motor has and whether the user is using full steps, halfsteps etc…., this function will take a degrees value and return an integer which the user can use for the user\_steps variable in the TMC2100\_Start() function.

**Example Code:**

/\* ========================================

\*

\* TMC2100 Component Example

\* Written by Matthew Bon for Digikey Electronics

\*

\* ========================================

\*/

#include "project.h"

#include "TMC2100\_1\_API.h"

double degrees = 90;

int stepper = 200;

int number\_steps = 0;

int main(void)

{

//int bob;

CyGlobalIntEnable; /\* Enable global interrupts. \*/

/\* Place your initialization/startup code here (e.g. MyInst\_Start()) \*/

TMC2100\_1\_TMC2100\_Config(false, Internal\_ref\_V, Halfstep, Interpolation\_off, spreadCycle);

number\_steps = TMC2100\_1\_TMC2100\_Degrees\_to\_Steps(degrees, stepper, Halfstep);

TMC2100\_1\_TMC2100\_Start(number\_steps, false, 60000, Low);

while (TMC2100\_1\_toggles != 0 ) //Wait for motor to complete motion

{

}

TMC2100\_1\_TMC2100\_Start(number\_steps, false, 30000, High);

while (TMC2100\_1\_toggles != 0 ) //Wait for motor to complete motion

{

}

TMC2100\_1\_TMC2100\_Release(UNLOCK);

for(;;)

/\* Place your application code here. \*/

}

}

/\* [] END OF FILE \*/