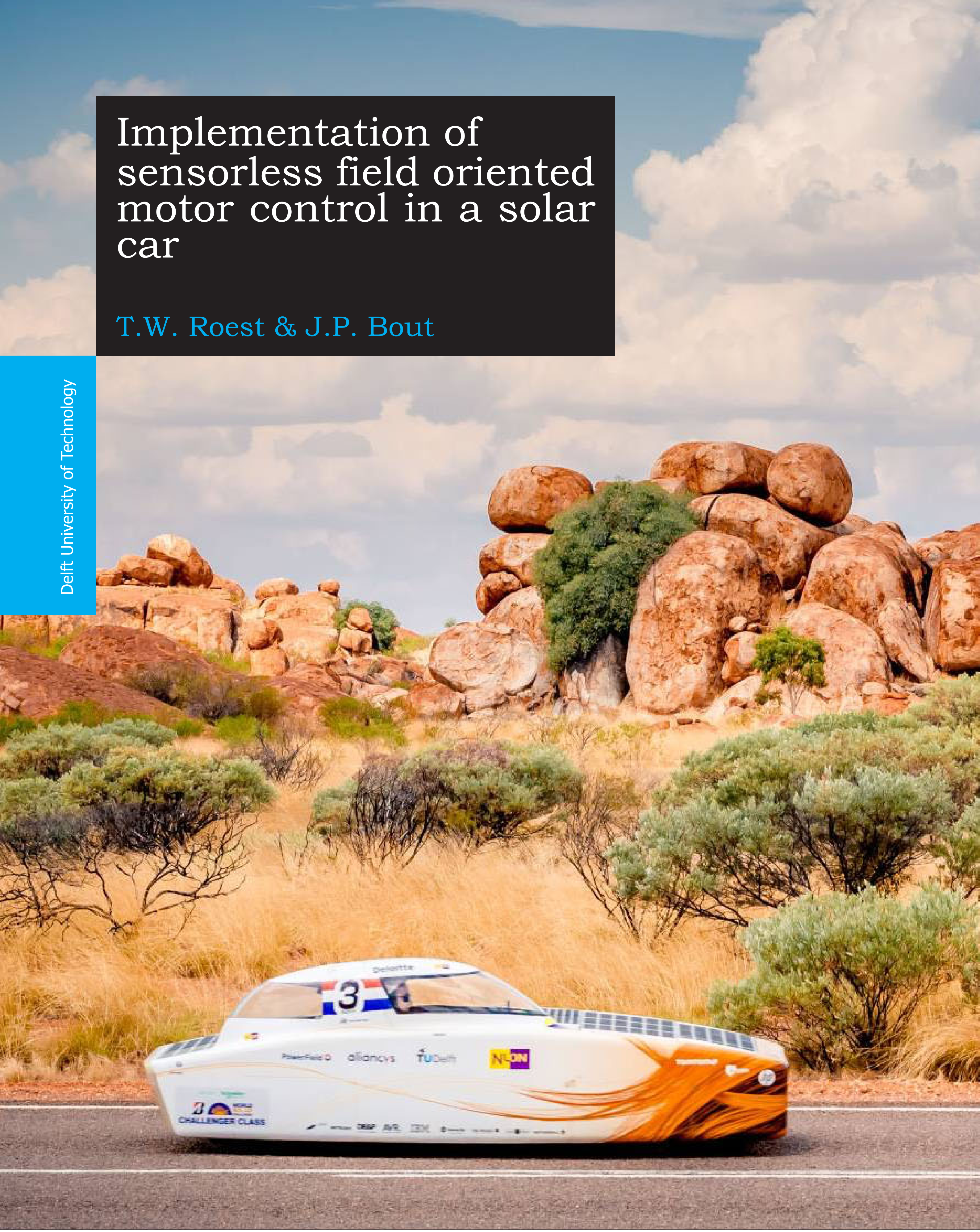
Implementation of sensorless field oriented motor control in a solar car

T.W. Roest & J.P. Bout

Delft University of Technology



Delft University of Technology

Challenge the future

Implementation of sensorless field oriented motor control in a solar car

by

T.W. Roest & J.P. Bout

in partial fulfilment of the requirements for the degree of

Bachelor of Science
in Electrical Engineering

at the Delft University of Technology,
to be defended publicly on Tuesday January 30, 2018 at 2:30 PM.

Supervisor:	dr. ir. P. van Duijsen	
Thesis committee:	prof. dr. ir. A. Smets,	TU Delft
	dr. ing. I. E. Lager,	TU Delft
	C. van Wezel, BSc,	Nuon Solar Team

This thesis is confidential and cannot be made public until January 30, 2023.

Delft University of Technology

Preface

Abstract

In this thesis the implementation of Field Oriented Control in the car of the NUON Solar Team will be discussed. The team decided to do research into the possibilities of a custom motor controller, and this thesis covers the beginning of that process. First a comparison between different control methods is made, and sensorless FOC is decided as the method to be used. Then the implementation of FOC is discussed. The Texas Instruments InstaSPIN™ technology has the preference for this implementation. This because it has all the elements of the control embedded in one microcontroller, along with a software estimator which eliminates the use of expensive sensors like a shaft encoder. Also the ease of use and the familiarity of TU Delft staff with this technology played a big part in this decision. With this technology, a simulation set-up was made as a proof of concept, which can also be used to teach the NUON Solar Team members the basics of motor control. After that the connections from the controller to the inverter are discussed, so it can be connected to other inverters in the future. In the controller, a CAN-interface is also implemented, which is essential for using it in the Nuna car. With the information presented in this thesis, the NUON Solar Team can continue the development of a controller. It is recommended to eventually leave the development kits used during this project, and integrate the MCU directly into an inverter. This will allow for well-tuned, efficient control of the motor with a small footprint.

Acknowledgements

The students would like to thank some people who have been especially helpful during this project. First of all thanks to dr. ir. Peter van Duijsen who was the supervisor of this project. His knowledge and enthusiasm were indispensable, and his help made this project a succes for us.

We would also like to thank the NUON Solar Team for assigning this project to us. It was a challenging and complex project. We have learned a lot and were able to incorporate several different aspects of the Electrical Engineering bachelor. Special thanks to ir. Bas van Wee and Casper van Wezel who were our daily supervisors from the NUON Solar Team.

Thanks to dr. ing. Ioan Lager for setting up the bachelor graduation project for only our group this quarter, and prof. dr. ir. Arno Smets, for being the chair of our thesis committee.

Our gratitude also goes out to dr. ing. Bart Roodenburg, who created a private workspace for us, and ir. Joris Koeners, who provided us with lab equipment and support when using the Texas Instruments development kits.

A great thanks to our families too, who have been an amazing support through our years of studying.

Last but not least, we would very much like to thank Elke Salzmann and Declan Buist, who worked on this project with us. You were inspiring and fun, and we really enjoyed working with you on this project!

T.W. Roest & J.P. Bout
Delft, January 2018

Contents

List of abbreviations

AC	Alternating Current
ADC	Analog-to-digital converter
BLDC	Brushless DC
CAN	Controller Area Network
DC	Direct Current
DTC	Direct Torque Control
EMF	Electromotive Force
EV	Electric Vehicle
FOC	Field Oriented Control
GPIO	General-purpose input/output
IDE	Integrated Development Environment
ISO	International Organization for Standardization
ISR	Interrupt service handler
MC	Motor Controller
MCU	Microcontroller unit
OCTW	Overcurrent & overtemperature warning
PCB	Printed circuit board
PI	Proportional Integral
PMSM	Permanent Magnet Synchronous Motor
PWM	Pulse Width Modulation
ROM	Read-Only-Memory
RTR	Remote Transmission Request
SPI	Serial Peripheral Interface
SVM	Space vector modulation

1 Introduction

In electric vehicle (EV) applications, the motor and its control can have a huge impact on the efficiency of the entire system. If there are too many losses, a lot of different problems can occur. The motor might overheat, the car might not be able to climb a hill, and the range of one battery charge might be significantly reduced. This means a good motor/controller combination is essential to ensure the best performance of any electric vehicle.

There is a specific class of EV's where efficiency is more important than in any other category: solar cars. The Nuon Solar Team from the Delft University of Technology has been competing in the Bridgestone World Solar Challenge in Australia for quite a few years, and in 2017 they became World Champion for the seventh time in the challenger class. To keep this title safe, they are always looking for new ways to improve their car: Nuna. Since 2014 they are also competing in the Sasol Solar Challenge in South Africa, and that brings a new set of challenges to the table as the conditions of the race are quite different.

As a part of the Bachelor Graduation project a group of students got an assignment from the Nuon Solar Team to research the possibility of a custom motor controller for the Nuna solar car. At the moment the motor/controller combinations used in the different Nuna vehicles are not quite as efficient as the team would like. In the next section an overview of the current problems is given.

1.1. The problem

A few years ago the Nuon Solar Team switched to a different motor for their car. The motor previously used was an axial flux permanent magnet synchronous motor (PMSM) manufactured by Marand. Right now they are using a radial flux brushless DC (BLDC) motor manufactured by Mitsuba. The controller previously used is the Tritium WaveSculptor22, and the Mitsuba motor comes with its own controller, the Mitsuba M1596C.

The Mitsuba motor works with both of these controllers. Both of the combinations are not optimal however. The Mitsuba motor controller is less efficient than the Tritium controller. Also this controller has analogue parameter control, which means it cannot interface with Nuna's Control Area Network (CAN) in a native way. Right now they have an extra converter in the car to convert the CAN signals to analogue signals and vice versa, which is far from optimal. The Tritium controller thus has an advantage over the Mitsuba controller in both efficiency and controllablity. However, that does not mean it's perfect. The Tritium controller switches to a different control method at a speed of around 25 km/h. This switching does not work well with the motor, so the car has to slow down again to allow this switching. This scenario is far from ideal, because at 25 km/h the car is almost always still accelerating. Also the Tritium controller is not capable of providing enough power to drive uphill, which is essential for the race in South Africa.

Both of the controllers are off-the-shelf products, which means they are essentially black box systems to the Nuon Solar Team. They only have a rough idea of how the controllers work, and are not able to modify them in a safe or efficient way.

1.2. The current solution

Currently the team uses different controllers for the races in Australia and South Africa.

- For the race in Australia they use the Tritium controller, as this has a higher efficiency than the Mitsuba controller. The fact that it cannot deliver as much power to the motor is not an issue here, because the race doesn't go through any mountainous areas. The issue with the switching during acceleration still remains however.
- For the race in South Africa the Mitsuba controller is used. This controller is needed because the Tritium controller cannot deliver enough power for the mountainous regions in South Africa. However, due to the lower efficiency, they are not able to drive according to their calculated energy consumption model. Especially uphill, the cruising speed is lower than the ideal speed because the motor might overheat otherwise.

1.3. The assignment

It is clear both of these controllers cannot deliver all the specifications needed for Nuna. That is why in this bachelor graduation project research is done on a better motor/controller combination than the ones currently existing. To build a complete controller would be far too much to ask from four students in a 10 week project, and that is why a focus is placed on a few subsystems. Two students made a simulation of the Marand and Mitsuba motors, to get the motors' specifications. The values gotten from these simulations are to be used as input for the controller. This thesis covers research into different control methods and an implementation of one of these methods. This means the inverter and motor are not considered in this thesis apart from some compatibility requirements. For the research about the motors in Nuna see [1]. The requirements for the control are as follows:

- **Efficiency** The whole system must be as efficient as possible.
- **Digital Interface** The controller must be able to interface with the CAN-bus in Nuna.
- **Monitoring** The voltages, currents, temperature and speed must be measured.
- **Flexibility** The controller must be able to handle different motor sizes, pole pairs etc.
- **User Interface** There has to be an interface where the configuration of the MC can be changed. For example the limits of the controller and error flags should be programmable.
- **Extra** The controller must be able to control both BLDC motors and PMSMs [1].

1.4. Deliverables

The requirements set in the previous section need to be translated into specific deliverables, so it is clear what the Nuon Solar Team can expect. This is also important for the project as it gives the students focus on what to work on. The following deliverables have been deducted from the requirements:

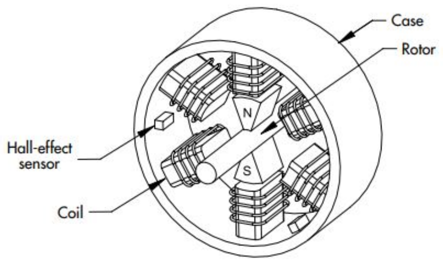
- A comprehensive study of control methods for BLDC motors and PMSMs has to be made. Here a comparison will be made to determine what control method can best be used for the Nuna car.
- After determining the best control method, a possible implementation solution needs to be found to make sure the chosen control method can be used practically.
- A small simulation set-up for testing and adjusting the control has to be made. This simulation can also be used to give the people in the Nuon Solar Team some basic motor control knowledge quickly.
- An extensive manual on how to make the simulation set-up is needed. This is paramount for a quick installation and will allow future contributors to this research to get started right away.
- A recommendation must also be given on how to implement the found solution.

2
Comparison of control methods

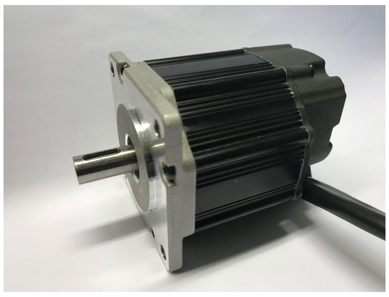
2.1. Introduction

First of all, the control method to be used by Nuna needs to be determined. Chapter 1 points out the control needs to be able to handle both BLDC machines and PMSMs. This chapter will compare different controlling techniques and make a decision on what technique should be used. For an introduction to motor types and how they work, see chapter 2 in [1].

Sensored motor control works with either Hall Effect sensors in the motor (BLDC control), or a rotary encoder placed on the shaft (PMSM control). Sensors and their wiring can, however, become quite expensive. Leaving the sensors out allows for a simpler construction and less wires from the motor. During recent years sensorless techniques have been developed thus far that efficiency is not an issue. This means that a sensorless technique is preferred, as it makes the construction simpler and cheaper. It will increase the complexity of the control algorithm, but due to the abundance of high quality microprocessors that is not a problem. Different sensorless techniques are available, and they all have their own advantages and disadvantages. A comparison is made between four popular methods.



(a) Hall Effect sensors in a motor ([2] Fig. 1).



(b) This motor has an encoder built in.

Figure 2.1: Sensored motor control techniques use Hall Effect sensors or encoders.

2.2. Sensorless control methods

Direct back-EMF

Direct back-EMF control is fairly simple compared to other sensorless techniques [3]. This works by detecting when the back-electromotive force of a phase crosses zero. Then after a set wait time the control switches to the next power stage [4]. It does, however, have a few drawbacks. It is sensitive to noise when detecting the zero crossing, and at low speeds, the low back-EMF will make it difficult to determine the actual zero crossing.

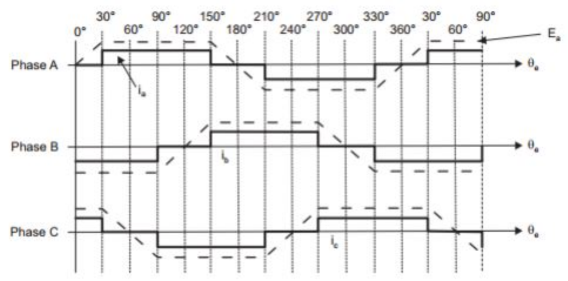


Figure 2.2: The phase back-EMFs and the switching behaviour with direct back-EMF control ([3] Fig. 3-2).

Indirect back-EMF

The problems that occur with direct back-EMF control can be partially solved by using indirect back-EMF control. Here the back-EMF is integrated about the zero crossing until a certain threshold is reached. This reduces the noise sensitivity. However, both back-EMF methods don't work optimal at low speeds because there is a low back-EMF from the motor [5].

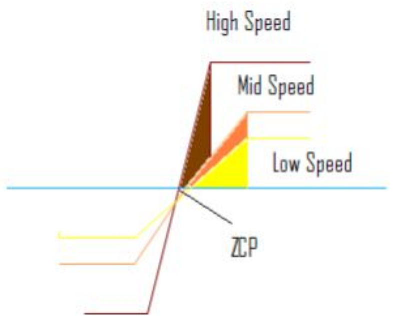


Figure 2.3: The integration of the back-EMF with indirect back-EMF control ([3] Fig. 3-4)

Direct Torque Control

Direct Torque Control (DTC) works with torque and flux reference vectors. Using a model estimation of the motor, the torque and flux in the motor are calculated with measurements of two phase currents, the intermediate circuit DC voltage and the state of the power switches [6]. Then the calculated torque and flux are compared to the reference and control is applied accordingly. DTC has quite large torque ripple [7]. This strains the construction of the car and often introduces audible noise.

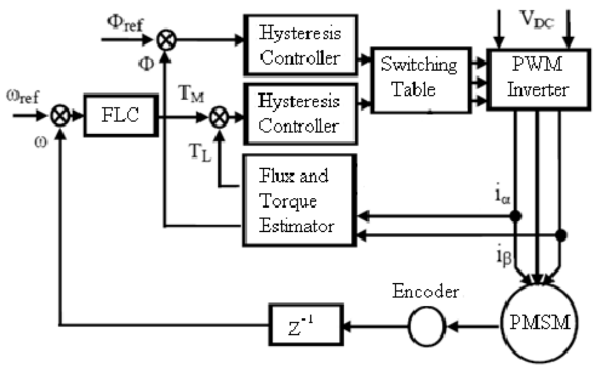


Figure 2.4: The block diagram of DTC. ω can also be estimated, so no encoder is needed ([8] Fig. 2).

Field Oriented Control

Field Oriented Control (FOC) is, like DTC, also vector-based. It transforms the torque and flux vectors so they are orthogonal to each other, and compares them to reference vectors. A more in-depth explanation of FOC will be given in chapter 3. While FOC is a little slower than DTC, it performs better on areas like torque-ripple and distortion [9]. However, FOC needs to know the position of the rotor, and so extra processing power is needed. This is not a big issue though, as microprocessors are fast, efficient and cheap.

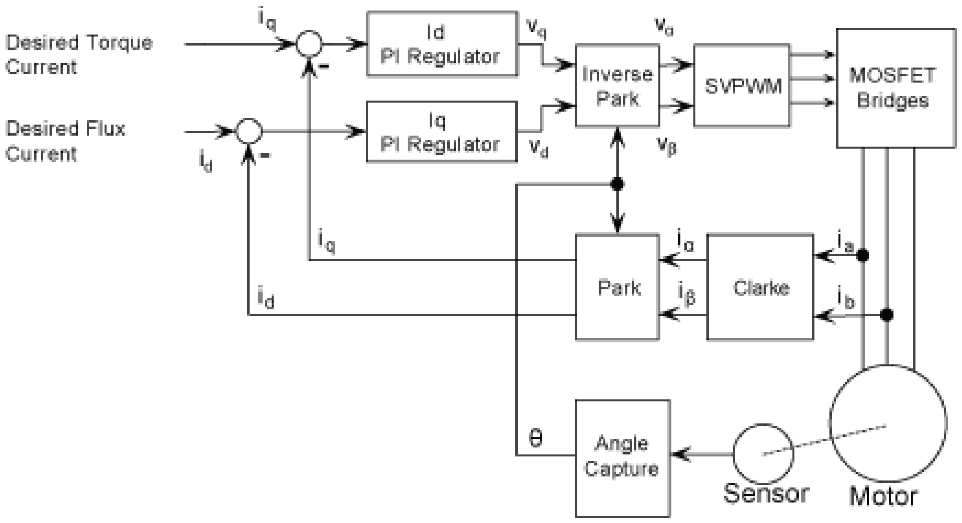


Figure 2.5: A basic implementation of Field Oriented Control with an encoder. The angle can also be estimated, removing the need for an encoder ([10]).

2.3. Results

In table 2.1, some of the main differences of the control methods have been summarized. There are several points of interest, some of which are more important than others.

Table 2.1: **Comparison of different sensorless control methods**

	Direct B-EMF	Indirect B-EMF	DTC	FOC
Computational simplicity	++	+	-	- -
Reaction time	-	-	++	+
Efficiency at high speed	+	+	+	+
Torque control at zero/low speed	- -	- -	-	+
Torque ripple	-	-	-	++

Looking back at the assignment presented in chapter 1, it is clear that torque control at zero/low speed is important. This will allow the Nuna vehicle to accelerate better when pulling away from a traffic light or to overtake other traffic more easily. From the comparison it became clear that FOC was the way to go. The separate torque control will allow the team to tune their car on the fly for different conditions. Also the torque ripple is very low (zero even when using a PMSM). This will strain the construction of the car less, and will provide a more comfortable ride for the driver.

2.4. FAST™ software encoder

Texas Instruments sells a unique solution for FOC, which implements the FAST™ software encoder. This technique is provided exclusively by Texas Instruments, and provides a more accurate control then any other solutions from other manufacturers. FAST™ is not a physical encoder, but rather an estimation algorithm implemented in the ROM of a microcontroller. Using the measured phase currents and voltages it estimates the flux, angle, speed and torque in the motor. Also it can identify some motor parameters like the inductances and stator resistance just by running a certain script. The stator resistance can even be recalculated while the motor is running. This allows the FOC to be constantly tuned to the motor parameters, which keeps the control as accurate as possible.

2.5. Implementation

For implementation of FOC Texas Instruments InstaSPIN™ technology will be used. Texas Instruments have different development boards and shields available for this technology. The technology works with the FAST™ software encoder, which makes it very accurate. The basics of the control have already been programmed, and with some minor adjustments it will work with different kinds of motor. This means if the Nuon Solar team decides to switch motors, they will just need to adjust a few values. The supervisor of this project, as well as other staff of the TU have used InstaSPIN™ technology before, so technical support can be found just down the hall. The FAST™ algorithm, the familiarity of TU Delft staff and the ease of use were the reasons this specific technology was chosen for the project. More decisions regarding the controller board are discussed in chapter 4.

3 Field Oriented Control

3.1. Introduction

Field oriented control (FOC) applies to brushless motors, not only AC but also DC, that operate in a sinusoidal mode. This means that all three wires are energized with currents that have a sinusoidal waveform. Each of the waveforms are 120 degrees apart from each other [10]. However in sinusoidal commutation, PI controllers have a limited bandwidth to keep a system stable. In motor control at higher speeds the time variant variables speed and frequency go up. This will result in a phase lag in the control loop and a gain error in the motor currents [11]. A visual implementation of field oriented control with an encoder is shown in fig. 2.5. Field oriented control uses the three phases of the inverter current output to change these into 2 parts: a torque generating current and a magnetic field generating current. These currents are time-invariant due to the transformations explained in section 3.2. After that the input user currents are calculated from the torque and speed in section 3.3. And finally the desired user currents are controlled using two PI controllers explained in section 3.4.

3.2. Coordinate System

A 3-phase synchronous machine is mostly represented in a two dimensional coordinate system with three axes, all 120 degrees apart from each other. Therefore none of the axes are orthogonal to each other. Field Oriented Control uses a rotating reference frame to control the components of the motor stator currents aligned with the rotor flux. By changing it to a rotating frame the currents, which are controlled, are time-invariant. Therefore the PI controllers work equally well at low and high speeds [11]. To change the 3-phase reference frame into a rotating reference frame there are two transformations needed: first the Clarke transformation to change from 3 phases into an two dimensional coordinate system with two orthogonal axes and secondly the Park transformation to make it rotational. See fig. 3.1 for a visual representation of the coordinate systems.

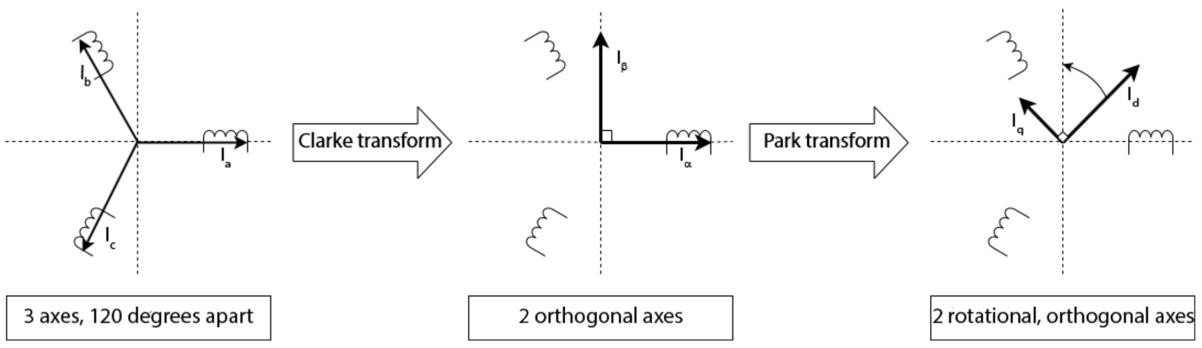


Figure 3.1: A visual representation of The Clarke and Park transformations

Clarke Transformation

The Clarke transform is expressed in mathematical form using the following equations[12]:

$$i_\alpha = \frac{2}{3} i_a - \frac{1}{3}(i_b - i_c) \tag{3.1}$$

$$i_\beta = \frac{2}{\sqrt{3}}(i_b - i_c) \tag{3.2}$$

or in matrix form:

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \sqrt{3} & -\sqrt{3} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{3.3}$$

with i$_\alpha$ and i$_\beta$ the components in the orthogonal coordinate system.
In the case that i$_\alpha$ is superposed with i$_a$ and i$_a$ + i$_b$ + i$_c$ equals zero, the Clarke transformation is expressed like:

$$i_\alpha = i_a, \tag{3.4}$$

$$i_\beta = \frac{1}{\sqrt{3}}(i_a + 2 i_b) \tag{3.5}$$

$$i_a + i_b + i_c = 0 \tag{3.6}$$

Park Transformation

To create a rotating reference frame for the Park transformation, the geometrical functions sine and cosine are needed. The direct and quadrature component in the Park transformation are represented by the following equations [12]:

$$i_d = i_\alpha \cos\theta + i_\beta \sin\theta \tag{3.7}$$

$$i_q = i_\beta \cos\theta - i_\alpha \sin\theta \tag{3.8}$$

or in matrix form:

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \tag{3.9}$$

where i$_d$ is the direct component, i$_q$ is the quadrature component and θ is the angle between the positive x-axis and the i$_d$ current vector. The direct component is used to control the flux input of the motor and the quadrature component is used to control the torque input of the motor.

Inverse Park Transformation

To get back from a rotating reference coordinate system to a non-rotating frame, the inverse park transformation is needed. The transformation is given in the following mathematical equations [12]:

$$v_\alpha = v_d \cos\theta - v_q \sin\theta \tag{3.10}$$

$$v_\beta = v_d \sin\theta + v_q \cos\theta \tag{3.11}$$

Where v$_d$ and v$_q$ are the direct, and respectively quadrature, voltage as the output of the two PI controllers. And v$_\alpha$ and v$_\beta$ are the orthogonal components of the three phase voltage system. These values are then used in the space vector modulation to drive the inverter.

3.3. Current control

The i$_{q,ref}$ and i$_{d,ref}$ or commanded torque and flux current can be calculated in multiple ways: The Q-axis current control or Maximum Torque per Amp (MTPA) control [13]. This depends on the state of the motor. In normal operation or during acceleration Q-axis current control is to be used. However, when driving above the rated speed field weakening is desired and MTPA control is preferred to be used.

Q-axis current control

First in the Q-axis current control, i$_{d,ref}$ is set to zero. Therefore the flux can't be controlled. However the torque can be controlled using the following formula [13]:

$$T = \frac{3P}{4}\psi_m i_{q,ref} - (L_q - L_d) i_{d,ref} i_{q,ref} \tag{3.12}$$

where T is the input torque, P is the total number of poles, ψ_m is the magnitude of the complex rotor flux vector, L$_q$ and L$_d$ are the motor inductances and i$_q$ and i$_d$ are the direct and quadrature reference currents. As i$_{d,ref}$ in Q-axis current control is zero and L$_q$ and L$_d$ are assumed equal, as the motors used by the Nuon Solar Team are both surface PMSM [1]. The formula can be rewritten into:

$$T = \frac{3P}{4}\psi_m i_{q,ref} \tag{3.13}$$

$$i_{q,ref} = \frac{4T}{3P\psi_m} \tag{3.14}$$

This can be further simplified using:

$$K_e = \frac{4}{3P\psi_m} \tag{3.15}$$

Where K$_e$ is the electrical motor constant. Now the commanded torque current becomes:

$$i_{q,ref} = \frac{T}{K_e} \tag{3.16}$$

To calculate the torque commanded current from the demanded speed, a Speed PI regulator is needed. This is shown in the block scheme of fig. 3.2. The gain factor K and time constant τ are dependent on the user case. Each motor needs different parameters as every motor is different.

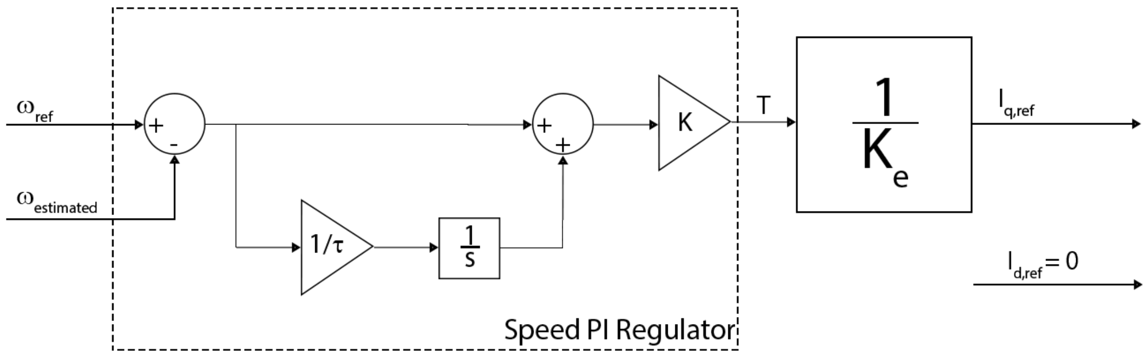


Figure 3.2: A Block scheme of the Speed control loop with the outputs Iq and Id.

MTPA control

To push the DC motor beyond its rated speed, field weakening can be used. The mechanical power is defined as torque times the speed. As the power is constant for rated speed, the speed cannot be increased due to the increasing back-emf, which is speed dependent. $i_{d,ref}$ will start to grow negatively, so the back-EMF is reduced. Now the electrical torque will reduce and the speed will increase. The speed increases until the back-EMF increases again and therefore there cannot be any more current drawn [14]. This is all done by using MTPA (Maximum Torque per Ampere) control. The following equations are used for $i_{d,ref}$ and $i_{q,ref}$ [13]:

$$i_{d,ref} = \frac{1}{4(\xi - 1)L_d}(\psi_m - \sqrt{{\psi_m}^2 + 8i_s^2(\xi - 1)^2{L_d}^2}) \tag{3.17}$$

$$i_{q,ref} = \sqrt{i_s{}^2 - i_d{}^2} \tag{3.18}$$

Where ξ is the saliency ratio L_q/L_d and i$_s$ is the input current. Now the torque can be maximized using the given magnitude of the current.

3.4. PI Control

After the MOSFET bridges the three currents are measured as shown in fig. 2.5. These currents are changed into the flux current (i$_d$) and into the torque current (i$_q$) using the Clarke and Park transformations, which are explained in section 3.2. After this the torque and flux currents are the input in two different PI Regulators to control the motor. An example of the i$_q$ PI Regulator is shown in fig. 3.3. In this PI controller the variables K and τ need to be determined by the user. These variables are different for every motor and depend on the motor parameters. The input of the i$_q$ PI controller is the i$_{q,ref}$ minus the estimated i$_q$. In fig. 3.3 1/s is the Laplace transform of the integral. Furthermore the

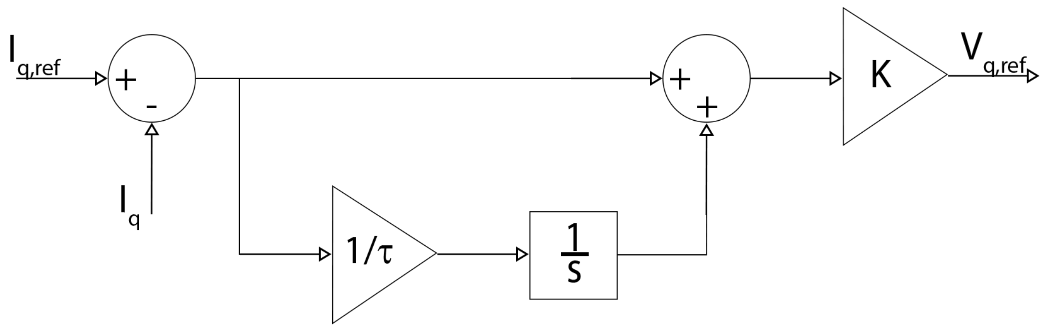


Figure 3.3: A Block scheme of the Iq PI regulator in fig. 2.5

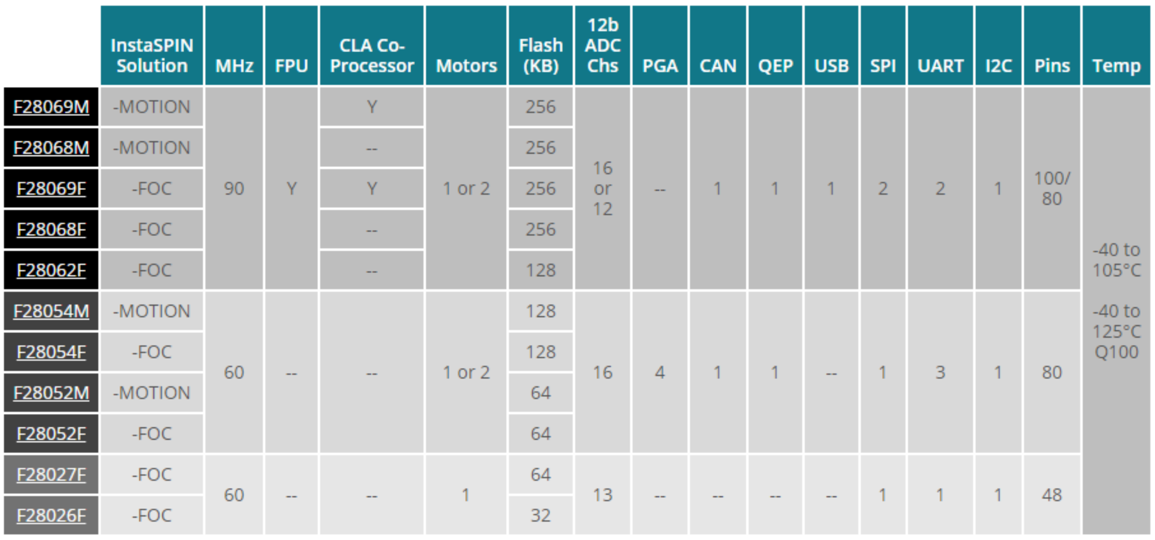
output v$_{q,ref}$ is than transformed back to a non-rotating frame (using the inverse park transformation, see section 3.2) to use it as an input for the SVPWM. The PI controller of i$_d$ looks the same, except that the input is i$_{d,ref}$ minus i$_d$, the output is v$_{d,ref}$ and the variables K and τ are different.

4

Controller Hardware

4.1. Controller choice

As discussed in chapter 2, the InstaSPIN™ technology by Texas Instruments is used for the controller. There are several microcontrollers available with this technology already implemented in the ROM. This will greatly simplify the development of the controller for the Nuna car. Thus, a decision has to be made on which one of these MCUs will be used. On the Texas Instruments website, fig. 4.1 can be found [15], in which a comparison of the specifications of the MCUs is laid out.



	InstaSPIN Solution	MHz	FPU	CLA Co-Processor	Motors	Flash (KB)	12b ADC Chs	PGA	CAN	QEP	USB	SPI	UART	I2C	Pins	Temp
F28069M	-MOTION	90	Y	Y	1 or 2	256	16 or 12	--	1	1	1	2	2	1	100/80	-40 to 105°C -40 to 125°C Q100
F28068M	-MOTION			--		256										
F28069F	-FOC			Y		256										
F28068F	-FOC			--		256										
F28062F	-FOC			--		128										
F28054M	-MOTION	60	--	--	1 or 2	128	16	4	1	1	--	1	3	1	80	
F28054F	-FOC					128										
F28052M	-MOTION					64										
F28052F	-FOC					64										
F28027F	-FOC	60	--	--	1	64	13	--	--	--	--	1	1	1	48	
F28026F	-FOC					32										

Figure 4.1: "TI's InstaSPIN™-enabled real-time controllers"

In addition to fig. 4.1, there are only a few MCUs available on the LaunchPad™ development platform. From the InstaSPIN™ enabled MCUs, only the F28027F and F28069M are available on a ready-to-go development board. This means the choice only depends on one particular requirement: CAN-interface capabilities. The F28069M is CAN compatible, and the F28027F is not. It is worth mentioning that fig. 4.1 shows the F28069M uses InstaSPIN-MOTION. This actually means it can use this as well as InstaSPIN-FOC and there is no conflict in control method. InstaSPIN-MOTION adds extra features that serve well in small household appliances [16], but are not needed for EV applications. InstaSPIN-FOC is what will be used. In conclusion the LAUNCHXL-F28069M is the only board that fits all of the requirements, so this will be used for the project.

Texas Instruments also have their own environment for programming MCUs, with a lot of different functions. For this application, the two most important tools are the Code Composer Studio IDE and

MotorWare™, which is the software and documentation package for developing InstaSPIN™ applications.

4.2. Process

In order to program a controller for the Nuna motor, several steps have to be taken. A small set-up with low-power BLDC motors will be built, which can be used to simulate races with actual data from the Nuon Solar Team. This will be useful for the students to understand the motor behaviour, and can be used by any future members of the Nuon Solar Team. In this way, work on the controller can continue after the project is done.

4.2.1. Components

The Launchpad development board has already been decided on in section 4.1. The next choice that has to be made is the inverter shield that will go on top of the LaunchPad. There are several BoosterPack™ shields available from Texas Instruments. For this project it is important to be able to connect the inverter to both the small motors used in the simulation as well as the motors used in Nuna. In this way the motor can be controlled up to a certain power. The final controller in Nuna will be connected to a high-power inverter, but it will be good to show the Nuon Solar Team some control features without the use of a high-power inverter. Of course the inverter board should still be able to deliver as much power as possible. This is why the decision had been made to use the BOOSTXL-DRV8323RH shield. This works up to 54V and with a continuous current of 15A [17].

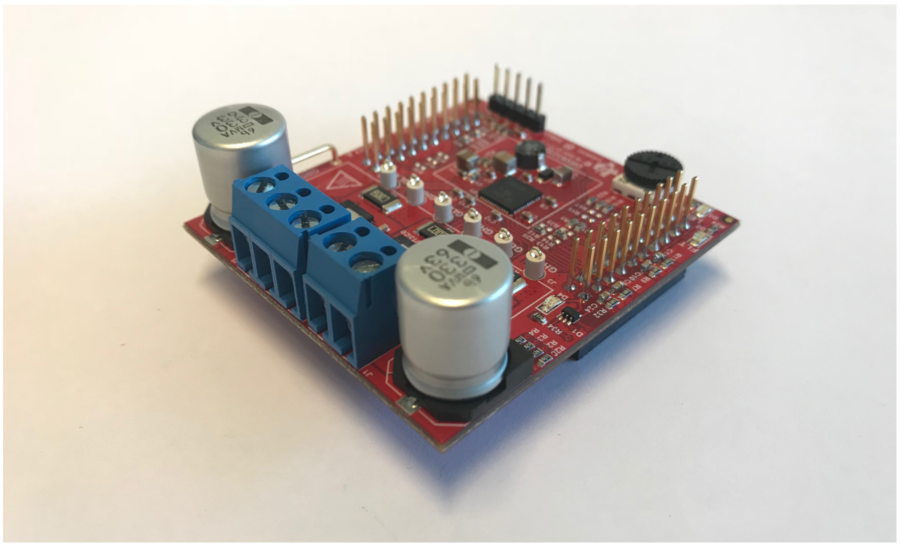


Figure 4.2: The BOOSTXL-DRV8323RH inverter

In order to have a small simulation set-up, some small motors are also needed. The 2MTR-DYNO evaluation module from Texas Instruments provides two low-power BLDC motors with a shaft coupler. This is exactly what is needed for the set-up. One motor will be tested as the actual motor while the other one acts as an active load. A schematic overview of the simulation set-up is given in fig. 4.3. A 30V 10A power supply was used, because that was available in the lab space. The communication between the PC's and the LaunchPads can be done either over USB or over CAN.

4.2.2. Development

After all the parts were ordered and had arrived, development could begin. However, there was a problem with the Launchpad/BoosterPack combination. The BOOSTXL-DRV8323RH turned out to not be directly compatible with the LAUNCHXL-F28069M. At first it seemed like a software problem, so all kinds of different software packages from Texas Instruments were downloaded to try and make it work. This did not work, and eventually an entry on the Texas Instruments E2E forum was found with [18] attached. A lab technician from the TU helped to solder some SMD capacitors to the BoosterPack, which were needed to make it work (see fig. 4.4). According to [18], only a few adjustments had to

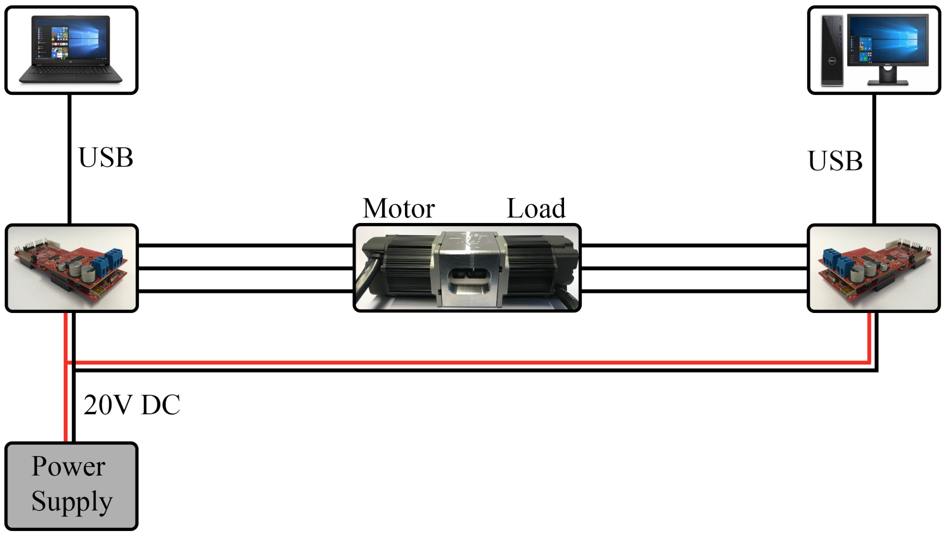


Figure 4.3: Schematic overview of the simulation set-up

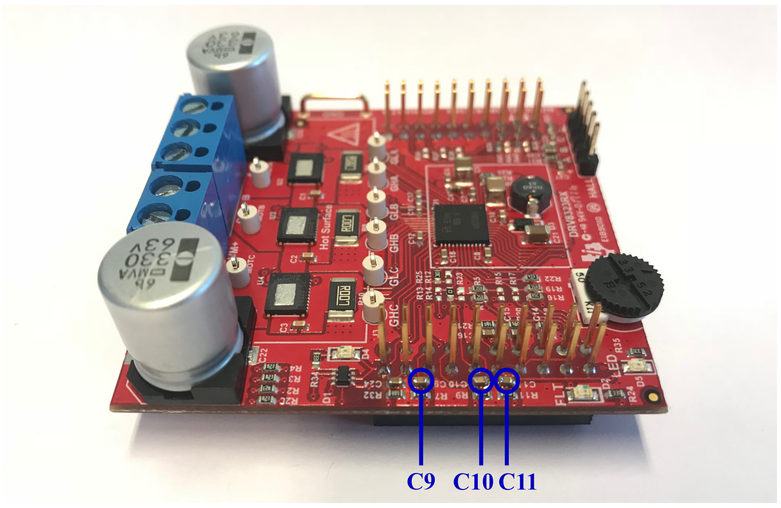


Figure 4.4: $0.1\mu F$ capacitors had to be soldered on C9, C10 and C11.

be made in Code Composer Studio to make the boards compatible. However, the hardware quick start guide found is for the DRV8323RS, which uses a Serial Peripheral Interface (SPI) for communication between the LaunchPad and the BoosterPack. The BoosterPacks ordered are DRV8323RH, which uses a hardware interface. Help was sought on the Texas Instruments E2E forum, but the answers gotten were too vague. To make the boards compatible several adjustments in the code had to be made regarding pins and register usage. Since neither of the students had prior experience with Code Composer Studio or any of the projects, this would be a near impossible task within the time scheduled for this project. So it was decided to order different BoosterPacks, that would work right out of the box with the LAUNCHXL-F28069M. This way CAN-capabilities can still be used. If the Nuon Solar Team desires to use the DRV8323RH in the future, extra programming will be needed, but it can be used.

The BoosterPack that was chosen was the BOOSTXL-DRV8301. This works up to 24V and with a continuous current of 10A. This is a significant drop in power compared to the DRV8323RH, but it can still be used for the simulation and driving the Nuna motor at low power. Since this BoosterPack is only used for the development and demos, it does not really matter how much power it delivers. In a later stage of development the Nuon Solar Team will still have to switch to a different inverter, which then will be controlled by the MCU that is used on the LAUNCHXL-F28069M. With the new BoosterPack, the development of the controller could finally begin.

To get familiar with Code Composer Studio and MotorWare™ the different tutorial labs available were followed. After this was done, it was clear all the tools to develop a controller for Nuna are there, and with some careful rearranging of code a proper controller could be developed from the labs. The installation of the software and setting up the simulation environment can be found in appendix A.

The most complete overview of the control for Nuna is given in Texas Instruments `proj_lab11a`. This is the lab that will be used for the motor controller the simulation set-up. The features of this lab are explained in section 5.2. For the torque controller in the simulation set-up, lab 4 will be used. How to use this lab will be briefly discussed in appendix A, and an in-depth explanation of this lab is given in [14].

4.2.3. Further development

As pointed out in section 4.1 and section 4.2.1, the LAUNCHXL-F28069M and the BOOSTXL-DRV8301 will only be used for the simulation, low-power demos & early prototyping on the Nuna motor. In a further stage of controller development, it is recommended to leave the current prototyping environment. If it is decided to design a custom inverter for Nuna, the LAUNCHXL is not the best solution, because it has a lot of connections that are unnecessary. The LAUNCHXL-F28069M is a platform build around the TMS320F28069M MCU for experimentation. When developing a custom solution for motor control this MCU can be integrated with the inverter. This will of course require a custom PCB, which is a lot of work. However, the best possible integration of the control can be guaranteed. A custom inverter will need extra measurement circuits to interface with the MCU so custom PCBs are already needed. Customizing the design will also allow for the controller/inverter combination to have the smallest footprint possible and have all the connections in a convenient place.

4.3. Pinout LAUNCHXL-F28069M

For the LAUNCHXL-F28069M to be connected to a different inverter than the BOOSTXL-DRV8301, it is needed to know where the different inputs and outputs are and what kind of signal type they use. To connect the controller to an inverter, the pins should be connected to the inverter as in fig. 4.5. Where these pins are located on the LAUNCHXL-F28069M is shown in fig. 4.6.

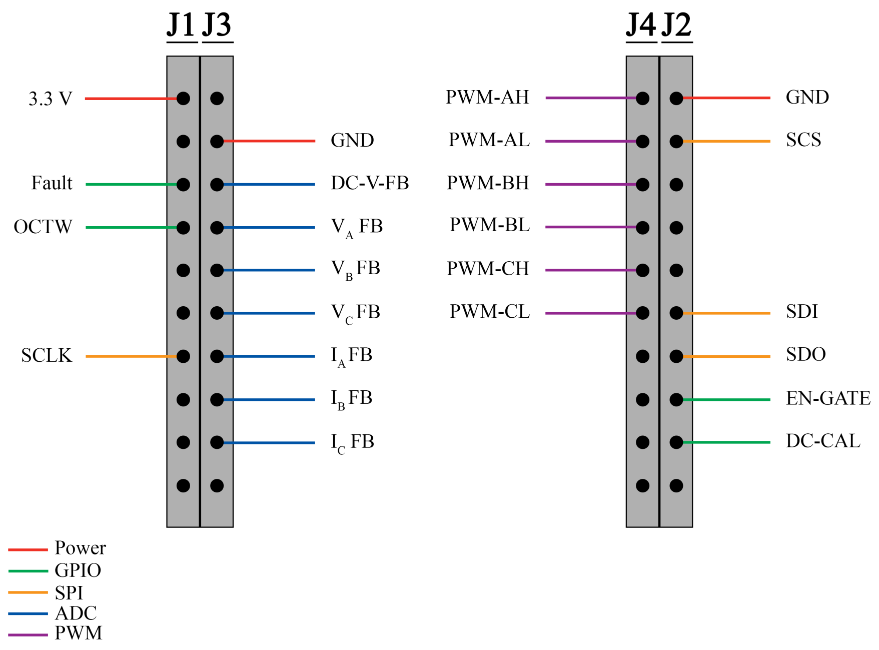


Figure 4.5: The pinout of the controller. Different signal types are represented by different colours.

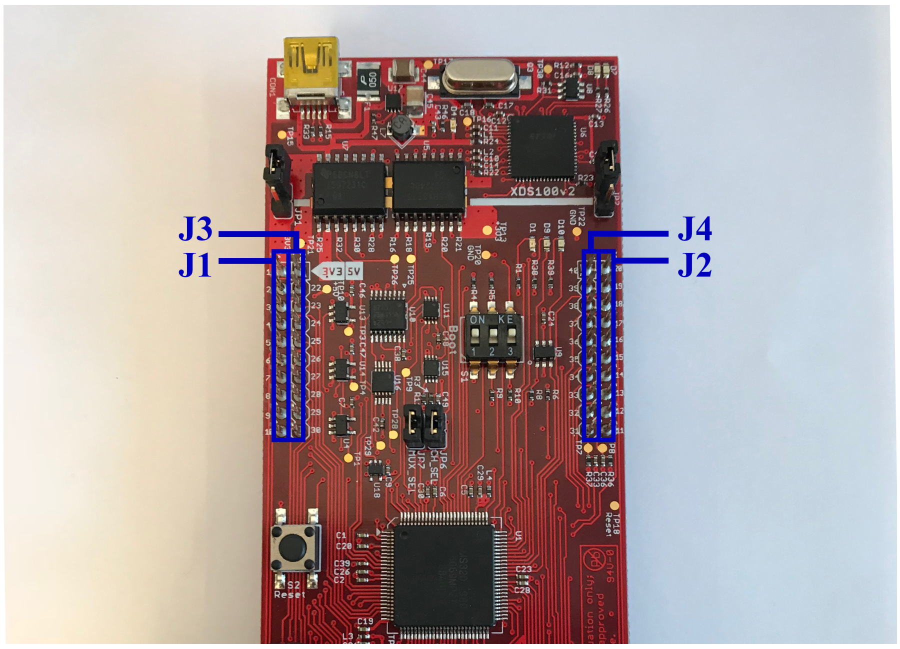


Figure 4.6: The location of the GPIO pins on the LAUNCHXL-F28069M.

Power signals

To power the controller, 3.3 V is needed. The CAN-bus from Nuna works on 12 V so a small buck-converter is recommended for getting the power from the CAN-bus. The BOOSTXL-DRV8301 also powers this pin through a buck-converter, but when using a larger inverter it is better to take the signal from the CAN-bus, as this has a lower voltage and a small converter can be used.

GPIO signals

These signals are specified for the feedback of the BOOSTXL-DRV8301. They will probably need to be redefined for use with a different inverter, as it presumably has a different communication standard. The signals are, however, quite useful for the functionality of the control and inverter, so they will be discussed here. This will give an idea of useful signals when setting up a different inverter with the LAUNCHXL-F28069M.

Fault (IN)

On the fault pin, the controller can see if a shutdown event has occured in the inverter [19]. This can happen due to events such as overcurrent, overtemperature, overvoltage or undervoltage.

OCTW (IN)

This pin lets the controller now when there has been an overcurrent or overtemperature event. Depending on the threshold and behaviour settings this may or may not lead to a shutdown event [19].

EN_GATE (OUT)

This pin is used to set the state of the gate driver, charge pump, current shunt amplifier, and internal regulator blocks of the BOOSTXL-DRV8301 [19]. This means energy can be saved with a low signal. It can also be used as a reset. With a short reset ($< 10\mu s$) only the gate driver will be reset, and with a long reset ($> 20\mu s$) all the blocks will be reset to a known state.

DC_CAL (OUT)

The DC_CAL pin is used for calibration to keep the DC offset and drift overtemperature low [19]. This signal will disconnect the load and short the input of the shunt resistor.

SPI signals

The Serial Peripheral Interface specification is used to set-up the BOOSTXL-DRV8301 [19]. In the SPI registers on the BOOSTXL-DRV8301 the thresholds and protection settings are stored. When using a different inverter, the signals are only relevant if the inverter also uses SPI for these settings.

ADC signals (IN)

Voltage measurement

Voltage feedback is needed for the FAST estimator to function properly. It takes a direct measurement from the motor phases, so some hardware settings for the voltage feedback need to be set [16]. The DRV-8301 also measures the voltage on the DC bus. The ADC input can only handle a maximum input of 3.3 V, so some kind of circuit is needed to keep the input in that range. Figure 4.7 shows a voltage

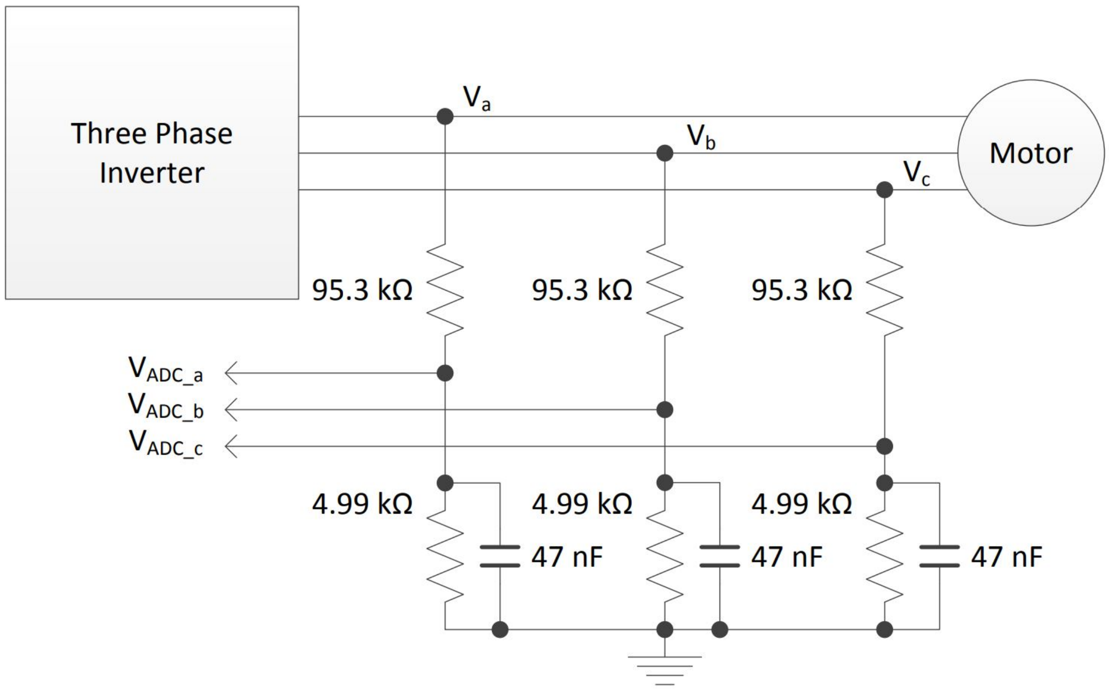


Figure 4.7: A voltage feedback circuit ([16] Fig. 5-7)

feedback circuit example. In this example the maximum phase voltage that can be sensed can be calculated as follows:

$$V_a^{max} = V_{ADC\_a}^{max} \times \frac{4.99k\Omega + 95.3k\Omega}{4.99k\Omega} = 66.3V \tag{4.1}$$

This value can be adjusted in the `user_j1.h` file:

```
//! \brief Defines the maximum voltage at the input to the AD converter
#define USER_ADC_FULL_SCALE_VOLTAGE_V (66.3)
```

For the BOOSTXL-DRV8301 this value is $V_{ADC\_a}^{max} = 26.314V$. The value of V_a^{max} should leave some headroom, so it should be about 20-30% higher than the rated voltage of the machine. The divider example from fig. 4.7 is ideal for a 48 V machine [16].

The switching nature of the inverter makes it hard to accurately measure the phase voltage. This is why the feedback signal needs to be filtered, which can be done using a capacitor. The filter needs to smooth out the signals, while still being able to let a high-speed voltage signal pass through. For a switching frequency of around 20 kHz a cutoff frequency of a few hundred Hz should be sufficient. In the case of the BOOSTXL-DRV8301, the voltage sensing circuit is as shown in fig. 4.8. To accurately detect the feedback voltage, the FAST estimator needs to now the pole of this filter. The frequency of

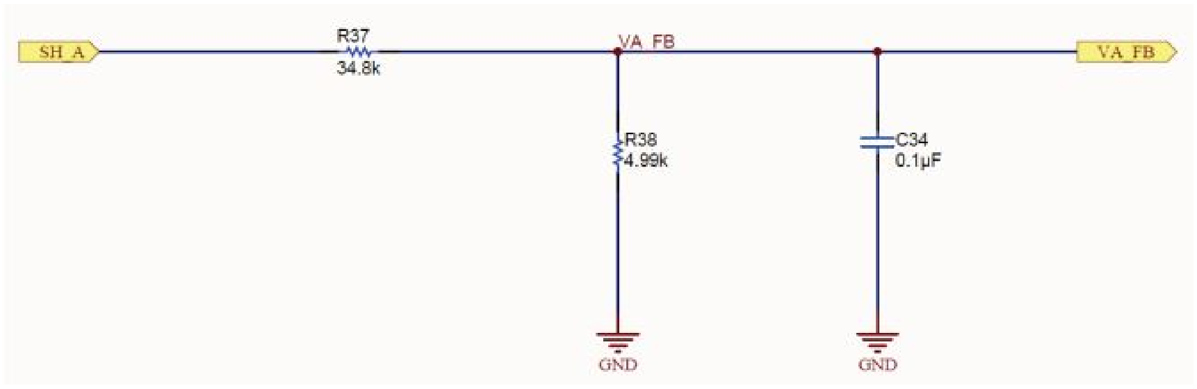


Figure 4.8: The voltage feedback circuit of the BOOSTXL-DRV8301 ([20] Fig. 6)

the pole can be calculated as follows (with the values of the BOOSTXL-DRV8301)[16]:

$$F_{filter\_pole} = \frac{1}{2\pi \times R_{parallel} \times C} = \frac{1}{2\pi \times \frac{34.8k\Omega \times 4.99k\Omega}{34.8k\Omega + 4.99k\Omega} \times 0.1\mu F} = 364.682Hz \tag{4.2}$$

The value of the filter pole's frequency for the FAST estimator can be changed in the `user_j1.h` file:

```
//! \brief POLES
// **************************************************************************
//! \brief Defines the analog voltage filter pole location, Hz
//! \brief Must match the hardware filter for Vph
#define USER_VOLTAGE_FILTER_POLE_Hz  (364.682)
```

Current measurement

The ADC's input range from 0 V to 3.3 V needs to be able to measure both positive and negative currents. To be able to tell the sign of the signal, a reference voltage of 1.65 V needs to be made. This can quite easily be done by making a buffered voltage divider, as shown in fig. 4.9 [16]. Using the 1.65 V as a reference voltage, the current can be sensed using a differential amplifier like in fig. 4.10.

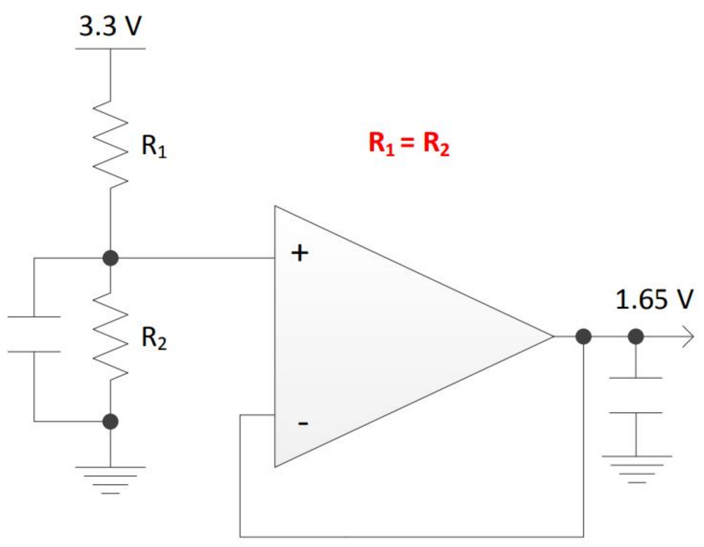


Figure 4.9: A voltage divider with a voltage follower to generate a 1.65 V reference signal ([16] Fig 5-2)

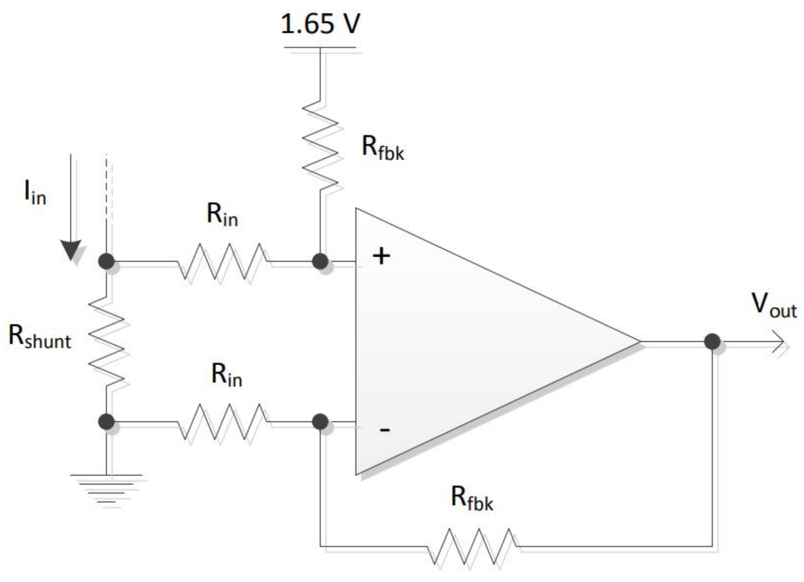


Figure 4.10: A differential amplifier circuit, which can be used for the current sensing ([16] Fig 5-3)

The transfer function of the circuit in fig. 4.10 is [16]:

$$V_{out} = 1.65 + I_{in} \times R_{shunt} \times \frac{R_{fbk}}{R_{in}} \tag{4.3}$$

With the values of the resistances, the maximum measurable current can be set. The maximum output voltage should be 3.3 V, and with the shunt resistor known, the other resistor values can be calculated. The maximum peak-to-peak current to be measured should be specified in the software. So if the phase current of the motor is ±10 A a value of 20 A should be specified. This can be adjusted in `user_j1.h`:

```
//! \brief Defines the maximum current at the AD converter
#define USER_ADC_FULL_SCALE_CURRENT_A        (33.0)
```

How well the current measurement represents the actual current depends heavily on the slew rate of the operational amplifier [16].

The op-amp of the differential amplifier can be configured to have either positive or negative current feedback. It is important to know which configuration is used, so that the MCU can have an accurate current measurement. In fig. 4.11 the two options are shown.

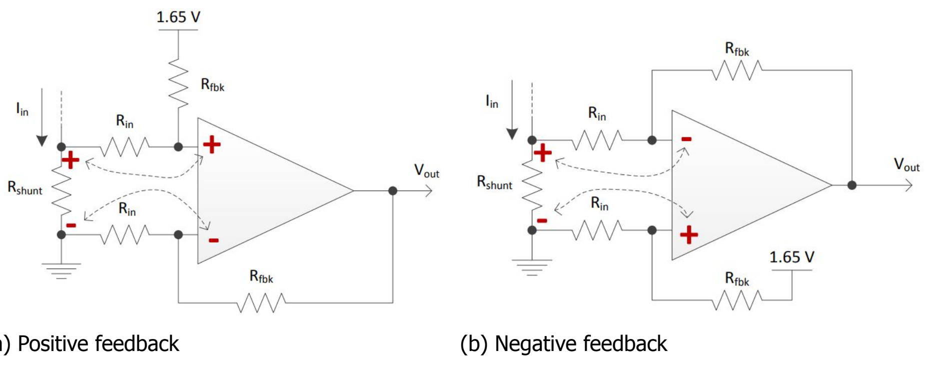


Figure 4.11: The two possible op-amp configurations for the differential amplifier ([16] Fig. 5-5 & 5-6)

The polarity of the feedback loop should be taken into account in the value of the current offset values. For a positive feedback the values of the offset should be negative, and for a negative feedback the values of the offset should be positive. The values can be set in `user_j1.h`:

```
#define    I_A_offset     (-0.8331743479)
#define    I_B_offset     (-0.8355930448)
#define    I_C_offset     (-0.8392037153)
```

(a) The current offset values for positive feedback

```
#define    I_A_offset     (0.8331743479)
#define    I_B_offset     (0.8355930448)
#define    I_C_offset     (0.8392037153)
```

(b) The current offset values for negative feedback

Not all the phases of an inverter necessarily have a shunt resistor to measure the current. The number of shunt resistors (2 or 3) can be specified in `user_j1.h`, so the Clarke transform can properly convert the three phase system to a two phase system. If three shunt resistors are available, they should all be used as that would give the most accurate result.

```
//! \brief Defines the number of current sensors used
//! \brief Defined by the hardware capability present
//! \brief May be (2) or (3)
#define USER_NUM_CURRENT_SENSORS              (3)
```

In fig. 4.12 a schematic overview of the measurement circuits together is given. This shows how to connect the voltage and current measurement circuits to the ADC pins on the LAUNCHXL-F28069M.

PWM signals (OUT)

The PWM signals coming out of the LAUNCHXL-F28069M use 0 V as low and 3.3 V as high. The frequency of the PWM signals can be changed to provide a suitable frequency for the motor provided. To keep the ripple low, motors with a lower inductance usually need a higher PWM frequency [16]. To make sure the control does not make audible noise, it is recommended the frequency is at least 20 kHz. The PWM frequency can be changed in `user_j1.h`:

```
//! \brief Defines the Pulse Width Modulation (PWM) frequency, kHz
#define USER_PWM_FREQ_kHz                  (20.0)
```

How the PWM signals are configured precisely is beyond the scope of this project, but it is explained in [21].

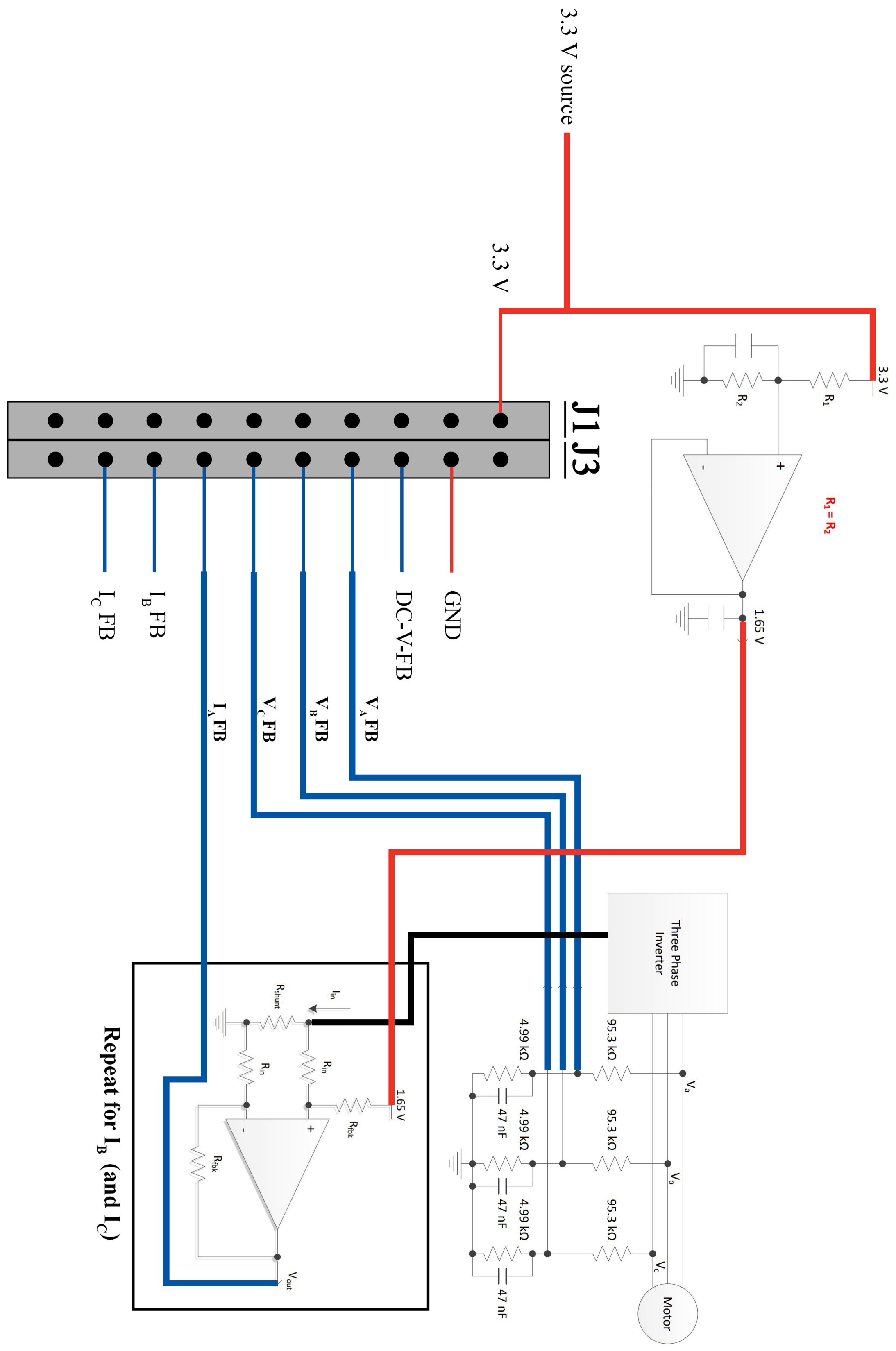


Figure 4.12: An overview of connecting the ADC pins

5 Controller Software

To understand how the FOC is implemented and how different motor specifications can be set in the controller it is essential to look at some sections of the C code used by the Texas Instruments labs. This chapter will focus on the code of lab 11a, where the motorcontroller itself is specified. For motor identification lab 2 can be used, and the torque load in the simulation will use lab 4. How these labs can be used is briefly discussed in appendix A. A more in depth explanation of the functions is given in [14]. In this project Code Composer Studio version 7.3.0, MotorWare™ version 1.01.00.18 and C2000Ware version 1.00.02.00 are used.

5.1. Adjustments for motor types

To use any kind of motor with the software provided, it must first be specified in the code what kind of motor it is. This can be done in the `user.h`, `user_j1.h`, and `user_j5.h` files. The `user_j1.h` and `user_j5.h` files are the same, with only one difference. The `user_j1.h` file is read when connecting the BoosterPack to the top slot of the LAUNCHXL-F28069M (closest to the USB port), and the `user_j5.h` file is for use of the bottom slot. Only `user_j1.h` will be used, as this project only uses the top slot. Which slot is used can be changed in the `user.h`:

```
// select whether to use the inverter on connector J1 or J5 of the LaunchPad
#define J1
```

The following section will show where to adjust the parameters for the motor in the code. Without these parameters, the control of the motor will not work properly. This is also where the values of the motor determined in [1] will eventually be entered. First, a motor must be defined. This is done in the `user_j1.h` file:

```
//! \brief USER MOTOR & ID SETTINGS
// **********************************************************************

//! \brief Define each motor with a unique name and ID number
// BLDC & SMPM motors
#define Estun_EMJ_04APB22           101
#define Anaheim_BLY172S             102
#define Tamagawa_A0100              103
#define Teknic_M2310PLN04K          104
#define Drone_A2212_1000KV 105
#define Drone_A2313_960KV 106

// IPM motors
// If user provides separate Ls-d, Ls-q
// else treat as SPM with user or identified average Ls
#define Belt_Drive_Washer_IPM       201
```

```
#define Anaheim_Salient              202

// ACIM motors
#define Marathon_5K33GN2A            301
```

There are already several motor types ready to use, but when using a new motor it will need to be specified. In the case of a BLDC or PMSM it can be defined here with the unique identifier 107. The simulation set-up uses the Teknic M2310PLN04K motors, which are already defined here, so that does not need to be added. Of all the motors defined, one must be selected. This can be done directly below the defining of the motors in `user_j1.h`:

```
//! \brief Uncomment the motor which should be included at compile
//! \brief These motor ID settings and motor parameters are then available
            to be used by the control system
//! \brief Once your ideal settings and parameters are identified update
            the motor section here so it is available in the binary code
//#define USER_MOTOR Estun_EMJ_04APB22
//#define USER_MOTOR Anaheim_BLY172S
//#define USER_MOTOR Tamagawa_A0100
//#define USER_MOTOR Drone_A2313_960KV
#define USER_MOTOR Teknic_M2310PLN04K
//#define USER_MOTOR Belt_Drive_Washer_IPM
//#define USER_MOTOR Marathon_5K33GN2A
//#define USER_MOTOR Anaheim_Salient
```

This will leave all the other motors with a grey layout in the code below. Only the selected motor will have a white background because it is used. The motor parameters show in the code as follows:

```
#elif (USER_MOTOR == Teknic_M2310PLN04K)
#define USER_MOTOR_TYPE                    MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS          (4)
#define USER_MOTOR_Rr                      (NULL)
#define USER_MOTOR_Rs                      (0.3918252)
#define USER_MOTOR_Ls_d                    (0.00023495)
#define USER_MOTOR_Ls_q                    (0.00023495)
#define USER_MOTOR_RATED_FLUX              (0.03955824)
#define USER_MOTOR_MAGNETIZING_CURRENT     (NULL)
#define USER_MOTOR_RES_EST_CURRENT         (1.0)
#define USER_MOTOR_IND_EST_CURRENT         (-0.5)
#define USER_MOTOR_MAX_CURRENT             (7.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz        (20.0)
```

If a new motor is introduced, with unidentified parameters, these values might need to be determined first. A new motor can be entered in the following way (assuming it is BLDC or PMSM) [14]:

- **USER_MOTOR_TYPE**
  Here the motor type is defined. In this case it should be MOTOR_Type_Pm.
- **USER_MOTOR_NUM_POLE_PAIRS**
  Here the number of pole pairs of the motor needs to be defined. In the case of the Teknic M2310PLN04K this number is 4. When adding the Mitsuba motor used by Nuna this number should be 16.
- **USER_MOTOR_Rr**
  The rotor resistance of the motor. Since this is irrelevant for BLDC machines and PMSMs this value should always be NULL.
- **USER_MOTOR_Rs**
  The stator resistance of the motor. A calculated value can be inserted here. If you want this value to be estimated by the software, NULL should be inserted here. Then after running an initialisation the estimated value can be copied here.

- **USER_MOTOR_Ls_d**
  The inductance of the motor can, like Rs, be entered or measured. In case an estimation needs to be done, enter NULL here.

- **USER_MOTOR_Ls_q**
  Since the motor specified is a BLDC (in the simulation set-up) or a surface PMSM (the Nuna motors), this value should approximately be the same as Ls_d. That means either NULL or the same calculated or estimated value.

- **USER_MOTOR_RATED_FLUX**
  The total flux linkage between the rotor and the stator. For now this value should be set to NULL, and after the initial estimation it should be updated.

- **USER_MOTOR_MAGNETIZING_CURRENT**
  This value is only used with induction motors, so it should be set to NULL.

- **USER_MOTOR_RES_EST_CURRENT**
  When in identification mode, the motor has to be started in open loop. This value is used to set the current peak during the initial startup. This value is used in the estimation of Rs. After the identification of the motor is complete this value is not used anymore. It is recommended to set this value at 10-20% of the rated current of the motor.

- **USER_MOTOR_IND_EST_CURRENT**
  This current can be set to the negative of USER_MOTOR_RES_EST_CURRENT. This is the value of the maximum current that will be used to estimate Ls.

- **USER_MOTOR_MAX_CURRENT**
  Should be set to the maximum nameplate current of the motor.

- **USER_MOTOR_FLUX_EST_FREQ_Hz**
  When identifying the motor, this will be used as the maximum commanded speed of the flux. For a PMSM motor, 20.0 Hz is a good starting point.

5.2. Functions in lab 11a

Lab 11a of the Texas Instruments manual shows most of the features that are needed for Nuna motor control. The most important functions are [14]:

- Keeping the controller as much out of ROM as possible. This will allow for modifications to the control to be implemented easily.
- Not using any motor identification. This should have already been done, with previous measurements or the values found in [1].
- Offset recalculation during standstill. This will allow for the control to be (re)calibrated when needed.
- Rs recalculation. This is also useful for (re)calibration.
- Rs Online. This will allow for the stator resistance to be recalculated while running the motor. This is very important, as this value will be used to calculate the temperature in the motor. This can be used to stop the motor from overheating.
- Speed and Id ramps. By not using steps to change the speed and Id reference overcurrents and overvoltages can be avoided.
- Space vector over-modulation. The lab will also allow for over-modulation to be used if desired. This thesis will not discuss over-modulation in detail.
- Field weakening is also available. This will allow the motor to spin beyond its rated speed by reducing the field in the motor.

- 1/Vbus compensation. This feature will adjust the output voltage to how much Vbus has dropped. This means the output voltage is always correctly compensated to match Vbus.

- CPU usage calculation. Here the percentage of the CPU used can be seen. This will allow for developers to check how extra functions impact the calculations, or show them they need to free up some space.

- The PID controllers can be independently tuned.

These features make for a great place to start developing Nuna's own controller, as a lot of control parameters can be adjusted beforehand and while running. How to use this lab is explained in appendix A. How these extra functions are implemented in the C code can be found in [14] under the lab 11a explanation.

5.2.1. FOC on the MCU

A block diagram of how FOC is implemented on the microprocessor is given in fig. 5.1 [16]. There is one big difference between the figure and lab 11a: the figure shows all of the control blocks inside a "CTRL" object that is called from the main loop, which is not used in lab 11a. Instead all the blocks seen here are directly implemented in the main interrupt service routine (ISR). This can be done because no motor identification is needed. The identification process is specified in the CTRL object, so if that is needed a previous lab should be used. Lab 11a provides a more open structure, because all parts of the control can be seen from the main ISR. This makes it easier to comprehend the control and modify specific parts.

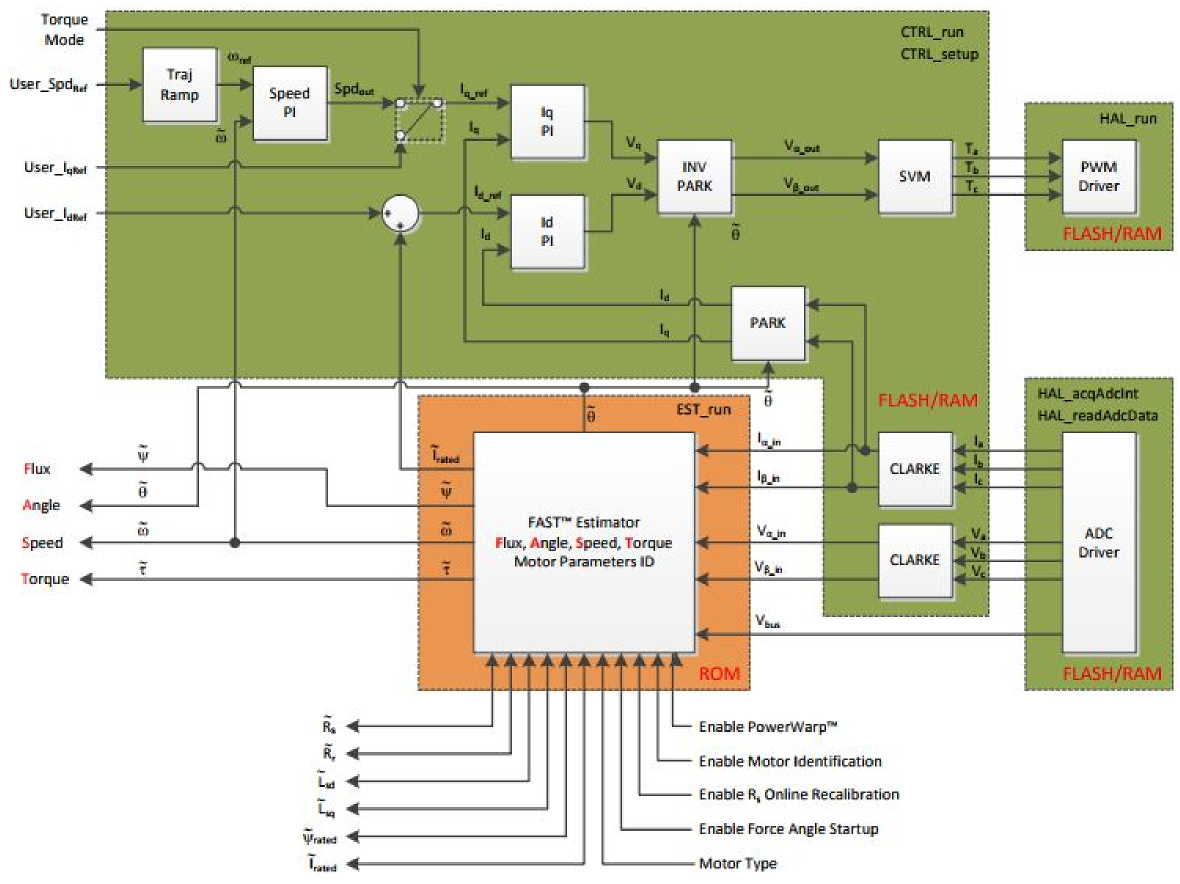


Figure 5.1: An overview of the way FOC is implemented with the TMS320F28069M MCU ([16] Fig. 1-3)

5.2.2. Field weakening in lab 11a

In lab 11a automatic field weakening is included. In fig. 5.2 the block diagram of this field weakening is shown. First, the output of the Id and Iq controller are used to calculate the output vector Vs. The output vector is then compared to VsRef, the maximum allowed output vector calculated from the limits set by the user. If Vs becomes larger than VsRef, field weakening will be applied [14]. As the motor's speed increases, Vs will become larger than VsRef, and at that point IdRef will start to grow negative. This will produce field weakening and keep Vs controlled to VsRef [14]. The changing of IdRef is done in the Field Weakening Controller. This controller uses a VsRef signal to determine the adjustment of IdRef, as shown in fig. 5.3. This field weakening method is an example, and other field weakening algorithms can be implemented in the motor controller [14].

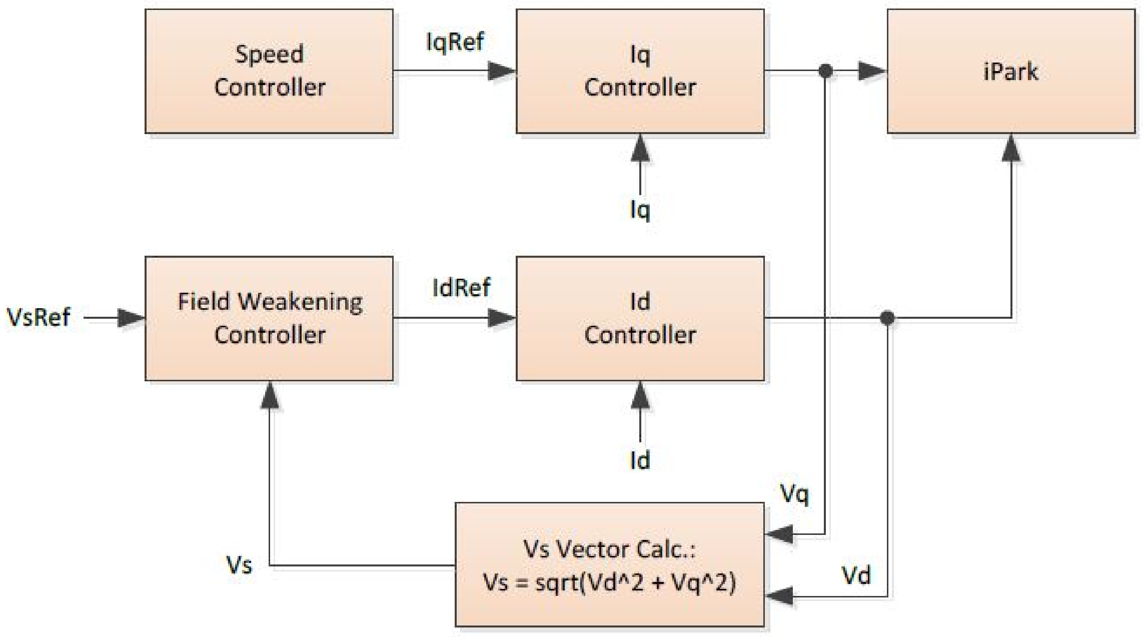


Figure 5.2: The block diagram of the field weakening implemented in lab 11a ([14])

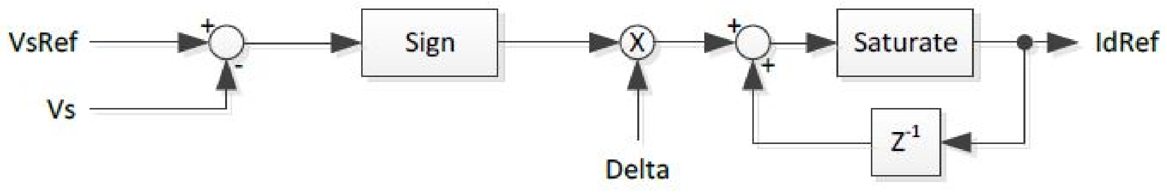


Figure 5.3: The field weakening controller ([14])

5.3. Lab 11a - Main ISR

The Code Composer Studio project for Lab 11a is very large. Here some of the basic functions of the project will be explained. Starting with the main interrupt service routine, this will give a proper overview of how the control works while running. This explanation can also be found in [14].

Forward FOC

Here data values are received from the ADC, the offset is applied to the currents and voltages, and the Clarke transforms on the currents and voltages are done.

```
// acknowledge the ADC interrupt
```

```
HAL_acqAdcInt(halHandle,ADC_IntNumber_1);
// convert the ADC data
HAL_readAdcDataWithOffsets(halHandle,&gAdcData);
// remove offsets
gAdcData.I.value[0] = gAdcData.I.value[0] - gOffsets_I_pu.value[0];
gAdcData.I.value[1] = gAdcData.I.value[1] - gOffsets_I_pu.value[1];
gAdcData.I.value[2] = gAdcData.I.value[2] - gOffsets_I_pu.value[2];
gAdcData.V.value[0] = gAdcData.V.value[0] - gOffsets_V_pu.value[0];
gAdcData.V.value[1] = gAdcData.V.value[1] - gOffsets_V_pu.value[1];
gAdcData.V.value[2] = gAdcData.V.value[2] - gOffsets_V_pu.value[2];
// run Clarke transform on current
CLARKE_run(clarkeHandle_I,&gAdcData.I,&Iab_pu);
// run Clarke transform on voltage
CLARKE_run(clarkeHandle_V,&gAdcData.V,&Vab_pu);
```

Estimator Run

This is the main function to ROM, to run the FAST™ estimator.

```
// run the estimator
EST_run(estHandle,
&Iab_pu,
&Vab_pu,
gAdcData.dcBus,
gMotorVars.SpeedRef_pu);
```

Extracting Estimated Variables

To actually control values with FOC, a few variables need to be extracted from the estimator. These are the following values:

- **F**lux - The value of Id.
- **A**ngle - The angle of the rotor.
- **S**peed - The speed at which the rotor is turning.
- **T**orque - The value of Iq.

```
// generate the motor electrical angle
angle_pu = EST_getAngle_pu(estHandle);
speed_pu = EST_getFm_pu(estHandle);
// get Idq from estimator to avoid sin and cos
EST_getIdq_pu(estHandle,&gIdq_pu);
```

Reading the Id and Iq values from the estimator is not necessary, but it saves execution cycles compared to getting phasor values from them and performing a Park transformation.

Speed Control

This is the first control block, which takes a reference from a global variable and an actual value from the estimator. The frequency at which the controller is activated can be set in `user_j1.h`.

```
// when appropriate, run the PID speed controller
if(pidCntSpeed++ >= USER_NUM_CTRL_TICKS_PER_SPEED_TICK)
{
// clear counter
pidCntSpeed = 0;
// run speed controller
PID_run_spd(pidHandle[0],
gMotorVars.SpeedRef_pu,
speed_pu,
&(gIdq_ref_pu.value[1]));
}
```

Id control

The controller of Id is executed next.

```
// get the reference value
refValue = gIdq_ref_pu.value[0];
// get the feedback value
fbackValue = gIdq_pu.value[0];
// run the Id PID controller
PID_run(pidHandle[1],refValue,fbackValue,&(gVdq_out_pu.value[0]));
```

Iq control

The controller for Iq is the last control block executed.

```
// get the Iq reference value
refValue = gIdq_ref_pu.value[1];
// get the feedback value
fbackValue = gIdq_pu.value[1];
// calculate Iq controller limits, and run Iq controller
outMax_pu = _IQsqrt(_IQ(USER_MAX_VS_MAG_PU * USER_MAX_VS_MAG_PU)
-_IQmpy(gVdq_out_pu.value[0],gVdq_out_pu.value[0]));
PID_setMinMax(pidHandle[2],-outMax_pu,outMax_pu);
PID_run(pidHandle[2],refValue,fbackValue,&(gVdq_out_pu.value[1]));
```

Using the value for available voltage from an output vector, as well as a limit set in `user_j1.h`, the output limit of the control is calculated before actually running the controller.

Inverse FOC

The outputs of the Id and Iq controllers go through an inverse Park transform and space vector modulation. For the inverse transform the most recent angle estimation is used in combination with a compensation factor. This compensation is based on the estimated speed and the frequency of the PWM module. This is needed because of the delay of the previous executions.

```
// compensate angle for PWM delay
angle_pu = angleDelayComp(speed_pu, angle_pu);
// compute the sin/cos phasor
phasor.value[0] = _IQcosPU(angle_pu);
241
phasor.value[1] = _IQsinPU(angle_pu);
// set the phasor in the inverse Park transform
IPARK_setPhasor(iparkHandle,&phasor);
// run the inverse Park module
IPARK_run(iparkHandle,&gVdq_out_pu,&Vab_pu);
```

The voltages out of the inverse Park transform need to be compensated for any drop in Vbus. The 1/Vbus calculation is read from the estimator, and the space vector modulation is executed with the compensated values.

```
// run the space Vector Generator (SVGEN) module
oneOverDcBus = EST_getOneOverDcBus_pu(estHandle);
Vab_pu.value[0] = _IQmpy(Vab_pu.value[0],oneOverDcBus);
Vab_pu.value[1] = _IQmpy(Vab_pu.value[1],oneOverDcBus);
SVGEN_run(svgenHandle,&Vab_pu,&(gPwmData.Tabc));
```

Writing PWM values

The final step of the interrupt service routine is writing the newly calculated PWM values to the PWM module through the hardware abstraction layer (HAL).

```
// write the PWM compare values
HAL_writePwmData(halHandle,&gPwmData);
```

At the end of this routine a full cycle of control like in fig. 5.1 has been executed.

6
CAN-interface

6.1. CAN Protocol

The Controller Area Network (CAN, or CAN Bus) is a method designed to let devices communicate with each other without using multiple wires. CAN is an International Organization for Standardization (ISO) defined serial communications bus [22]. It is designed to let control units and devices communicate with each other over two lines: a CAN-H (CAN High) and a CAN-L (CAN Low) wire. These are connected to all the parts of the network and finally ended with a CAN Data Bus terminal, like figure 6.1. This is often a 120 ohm resistor, preventing that data is reflected at the ends of the CAN line [23].

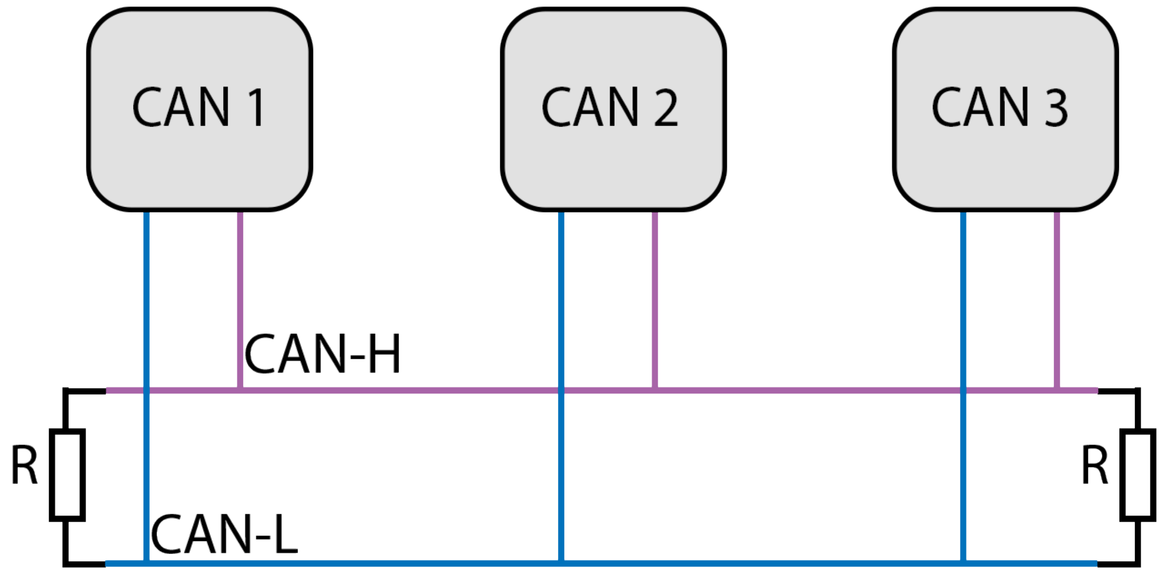


Figure 6.1: A simplified implementation of a CAN interface

CAN bus communication protocol is a carrier-sense, multiple-access protocol with collision detection and arbitration on message priority. [22] The sequence that is brought onto the lines is equal, however their amplitudes are opposite. Meaning a pulse on the CAN-H line goes from for example 7.5V to 4V, the pulse on the CAN-L goes from 0.5V to 4V and vice versa. This provides a better noise resistance and therefore less corrupted data. When the two CAN-bus lines (CAN-H and CAN-L) are 7.5 volt and respectively 0.5 volt, the voltage difference is at the maximum and the status of the bit is in dominant state with the value of 0. When both lines hit the 4 volt, the bit goes in recessive state and the value is 1 [23].

The transfer process, using figure 6.1, works as following:

1. CAN 1 provides the data to the CAN 1 controller.
2. The CAN 1 controller sends the data over the lines using electrical signals, if the line is free.
3. CAN 2 and 3 controllers receive the data.
4. Both the nodes check if they need the data and decide to accept or reject the data.

Their are two kinds of CAN protocols. The protocol for low speed CAN, ISO 11898-3, with speeds up to 125 kb/s (maximum distance of 500 meters). And ISO 11898-2, high speed CAN op to 1 Mb/s (maximum distance of 40 meters). These can be further divided into the length of the identifier (ID) [23]:

- Standard length of 11 bits.
- Extended length of 29 bits. Using a standard 11 bit identifier with an 18 bit extension.

The car of the Nuon Solar Team uses the ISO 11898-2 protocol, with preferably the Standard CAN identifier length.
If two or more devices try to send a message on the bus. The message, which has the highest priority or most dominant bits, will be sent first. The others are placed in the queue. Figure 6.2 shows how a CAN message is structured.

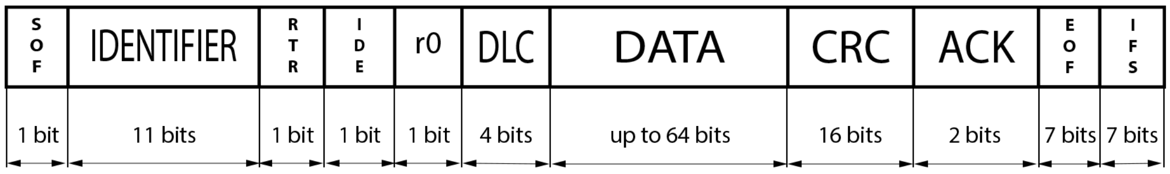


Figure 6.2: A standard CAN message structure: 11-Bit Identifier [22]

In short:

- SOF: Start Of Field (1 bit), indicates a message is send with a dominant bit.
- Identifier (11 bits), this is the priority of the message. How more dominant bits the identifier is, how more important the message is.
- RTR: Remote transmission request (1 bit). This bit is dominant if some other information is required.
- IDE: Identifier Extension (1 bit). If this bit is a zero the standard identifier is used. If this is a recessive 1 bit, the extended version is used.
- r0 (1 bit), reserved bit (for possible use by future standard amendment).
- DLC (4 bits), displays how much items were sent in the data field.
- Data (up to 64 bits), the actual information that was sent.
- CRC: Cyclic Redundancy Check (16 bits). This field is used to check if there were any faults in the message.
- ACK, Acknowledge Field (2 bits). All the receivers acknowledge in this field, they have received the data.
- EOF: End Of Field (7 bits). The end of the data protocol.

- IFS: Interframe Space (bit). The time required by the controller to move a correctly received frame to its proper position in a message buffer area.

The extended CAN message structure is the same as the Standard CAN message with the addition of:

- SRR: Substitute Remote Request (1 bit). Replaces the position of the RTR bit.
- IDE: Identifier Extension (1 bit), is a recessive 1.
- 18-bit extension identifier
- r1, an additional reserved bit. Following after the 18-bit identifier, the RTR bit and followed by the r0 bit. [22].

6.2. Requirements

To implement the microcontroller into the new Nuna, the following requirements for CAN were given:

- Minimum CAN bus supply voltage: 9 volt.
- Maximum CAN bus supply voltage: 15 volt.
- Nominal Can bus supply voltage: 12 volt.
- CAN bus data rate: 500 kbps.
- CAN bus isolation from high power DC bus: 1000 volt.
- CAN protocol ISO 11898-2 is used.
- *Standard Identifier: 11 bits.*

The requirement in italic is not mandatory, but preferable.

6.3. The enhanced Controller Area Network (eCAN)

The enhanced CAN module uses a serial communication protocol. It is ideal to use in noisy environments like the automotive and some other industrial fields. The enhanced CAN module has at least the following features [24]:

- Up till 32 mailboxes in extended mode.
- Supports data rates up to 1 Mbps.
- Fully compliant with ISO 11898-2 (version 2.0B).
- Configurable as receive or transmit mailbox, as well as standard or extended identifier.
- Composed of 0 to 8 bytes of data.

The eCAN module exists out of multiple parts. The most important are the mailboxes and the control and status registers. There are 32 mailboxes and here is the message defined that is received or needs to be transmitted. The control and status registers are used to control the mailboxes.

eCAN mailboxes

The mailboxes exist of 4 parts:

- **MSGID – Message Identifier (32 bits)**
  These bits set the identifier extension bit and acknowledge the identifier. The last 29 bits for the extended version and bit 28 till 18 for the standard identifier.
- **MSGCTRL – Message Control (32 bits)**
  These bits set the remote-transmission-request bit and define the data-length of the code.

- **CANMDL – Message Data Low (4 bytes)**
  The first 4 bytes of the actual data.
- **CANMDH – Message Data High (4 bytes)**
  The last 4 bytes of the actual data.

eCAN Control and Status Registers

In the Control and Status registers there are 26 registers each of 32 bits. The most important ones are:

- **CANME – Mailbox Enable**
  In this register every bit is used to enable or disable the individual mailboxes.
- **CANMD – Mailbox Direction**
  Every bit is used to set the direction of each individual mailbox. By writing a 1 the mailbox is configured to receive, set it to 0 the mailbox is a transmit mailbox.
- **CANTRS – Transmission Request Set**
  If one of the mailboxes is ready to transmit a message. This register is set and the transmission is started. The bit can't be cleared by itself. The CPU writes to the transmission-request-reset register when a transmission is send or aborted.
- **CANTA – Transmission Acknowledge**
  If the message was sent successfully, the bit of the transmission acknowledge is set. The number of the bit, that is set, is the number of the mailbox that has sent that message. To clear the bit a 1 must be written to the corresponding bit.
- **CANRMP – Received Message Pending**
  If a message is received in the mailbox, the corresponding bit of the received message pending register is set. This bit can only be reset by the CPU. If there is received a new incoming message, the old one is overwritten.
- **CANMC – Master Control**
  This register controls the settings of the CAN module. Some bits are protected from writing directly. The most important settings are:
  - Standard CAN Configuration bit. This bit defines if the eCan module is half or fully used.
  - Change-configuration request. When this bit is set the configuration register CANBTC can be changed.
  - The data byte order. This bit determines if the least significant or most significant byte is received or transmitted first.
  - The Self test mode bit. If the CAN module is in self test mode, it sends its own acknowledge bits so the code can be tested. To put it in normal CAN mode this bit needs to be zero.
- **CANBTC – Bit-Timing Configuration**
  The network-timing parameters are defined in this register. This register needs to be programmed before using the CAN module and to program this register the Change-configuration request bit in the Master Control register needs to be set.
- **CANES – Error and Status**
  The Error and Status register gives the actual status of the CAN module. It also displays the bus error flags and the status error flags.
- **CANTIOC – TX I/O Control**
  The CANTX pin should be configured for CAN use. Therefore in this register the transmit functionality is assigned to the CANTX pin.
- **CANRIOC – RX I/O Control**
  The CANRX pin should be configured for CAN use. Therefore in this register the receive functionality is assigned to the CANRX pin.

- **CANTSC – Time-Stamp Counter**
  In this register the counter at any given moment is displayed. It counts on the bit clock of the CAN bus. So for a 500 kilohertz clock the counter counts every 2 microseconds.

6.4. CAN code

The eCAN is not fully supported in MotorWare™. Later, a header file was added, which is a good introduction in the CAN support. However to use the CAN interface a C file needs to be added to the project. This is done using the steps of the SCI/UART tutorial [25]. Nevertheless, a user on the Texas Instruments E2E forum had the same problem and integrated the CAN module himself using the eCAN selftest of the F28069M development board [26]. How the code can be added to lab 11a can be found in appendix A.

The most important functions used in the CAN.c file in appendix D are:

- **ECAN_init**
  Allocate memory to the different registers.
- **ECAN_setBitrate**
  Change the bitrate of the CAN module. There can be chosen to set it on 1 MHz, 500 kHz, 250kHz, 125kHz, 100kHz, 83kHz, 50kHz or 20kHz. The default setting is 500 kHz.
- **ECAN_setBTCreg**
  Configure the rest of the Bit timing configuration register.
- **ECAN_clearMSGCTRL**
  Clear the Message Control of the mailboxes.
- **ECAN_clearMSGID**
  Clear the Message ID's of the mailboxes.
- **ECAN_clearMDL**
  Clear the Data Low of the mailboxes.
- **ECAN_clearMDH**
  Clear the Data High of the mailboxes.
- **ECAN_enableAllMailbox**
  Set the CANME register of all mailboxes to 1 to enable the mailboxes.
- **ECAN_setTXIO**
  Set the CANTXIO to 1 to enable the transmit functionality.
- **ECAN_setRXIO**
  Set the CANRXIO to 1 to enable the receive functionality.
- **ECAN_setSCCmode**
  Use the Standard CAN configuration mode. The CAN module uses only 16 mailboxes instead of 32.
- **ECAN_setECANmode**
  Use the full functionality of the CAN module. The module uses all 32 mailboxes.
- **ECAN_clearCANTA**
  Write to every CANTA register a 1 to clear all the transmission acknowledgements.
- **ECAN_clearCANRMP**
  Write to every CANRMP register a 1 to clear all the received message pending register bits.
- **ECAN_SelfTest**
  Define if the eCAN module has to use Self Test or use normal CAN mode.

- **ECAN_setMailboxDir**
  Set the directions of each mailbox. This is done by writing a 1 to let a mailbox receive or set a 0 to transmit.
- **ECAN_configMailbox**
  Configure the mailbox completely: Enable it, Set the Message ID, Specify number of bits that will be transmit or received. And finally define what mailbox direction is used.
- **ECAN_putDataMailbox**
  Write data to the mailbox MDL and MDH registers.
- **ECAN_configMasterReg**
  Configure the master control register.
- **ECAN_sendMSG**
  Send a message. First specifying the Data in the MDL and MDH register. Than set the CANTRS register to send the actual message and finally wait for an acknowledgement in the CANTA register.

7 Conclusion

In this thesis the different subjects presented in section 1.3 and section 1.4 have been discussed. A proper study has been done on different control methods and field oriented control was chosen as the most effective technique. FOC is effective at low speeds, and has a low torque ripple. Both of these characteristics make it a good solution for Nuna.

To get started with developing the FOC, Texas Instruments InstaSPIN™ technology was used. This provides an extensive developing environment, including ready to go development boards. The integrated FAST™ algorithm makes this solution unique and accurate. The ability to quickly and accurately estimate the flux, angle, speed and torque, as well as calculating motor parameters while running, was a big factor in the decision to use this technology. The project supervisor and other staff at TU Delft have experience with this developing environment already, which made the decision for this technology even easier. The LAUNCHXL-F28069M was used in combination with the BOOSTXL-DRV8301 Rev. B. The MCU on the LAUNCHXL already has FOC implemented in it, with a software encoder algorithm to keep the control fully sensorless.

Using these two boards together with a set of development motors, a simulation set-up was made. The set-up will allow people from the Nuon Solar Team to understand and interact with the controller quite easily. The Nuna motors can also be driven at low power with the development boards. How the control is implemented in the software is explained in this thesis. The connections on the LAUNCHXL-F298069M have been discussed as well. This means the Nuon Solar Team can connect the controller to a different inverter in a further stage of development. The CAN-interface capabilities have also been implemented, so the Nuon Solar Team can specify which values they want to input and output during driving.

For use of the simulation set-up, an elaborate manual was made. This will guide users through the installation of the required software from Texas Instruments and setting up the hardware. The Texas Instruments environment can be quite hard to grasp, because all kinds of different software packages have to be installed. With this manual it will become more clear on what is needed and what is not needed to set up the controller development.

Of the deliverables stated in section 1.4 all are done. The control method study can be found in chapter 2, the implementation choices and simulation set-up are explained in chapter 4 and chapter 5, the manual can be found in appendix A, and the recommendations for further research are in chapter 8.

8 Discussion & Recommendations

8.1. Discussion

While all the deliverables have been achieved, one big issue came up that delayed the project and had influence on the quality of the delivered result. The incompatibility of the BOOSTXL-DRV8323RH with the LAUNCHXL-F28069M was a major setback on the progress of the project. On the Texas Instruments website the LAUNCHXL-F28069M is listed on the page of the BOOSTXL-8323RH as a product that is compatible. However, when both the development boards arrived and were connected to each other, it became clear quite quickly that they were not properly configured to work together. The students did not understand why this was the case, because the boards were from the same manufacturer, the same product range and had the same technology on them.

Figuring out why the board did not work cost quite some time, and when the Hardware Quick Start Guide of the DRV8323RS was found, the problem seemed to have been solved. When the hardware adjustments (described in section 4.2.2) were made according to this guide, it became clear that software adjustments also had been made. At this point already almost two weeks had been spent on trying to get the boards to work together, as it was the first time either of the students worked with the Texas Instruments environment and could not fully comprehend what was going wrong. The decision was made to switch to the BOOSTXL-DRV8301 Rev. B, which worked properly the moment it was connected to the LAUNCHXL-F28069M.

A lot of time went into figuring out what was wrong with the first board, and that time could have been used more efficiently to explore more features of the control. This would not have been an issue if Texas Instruments had a clearer website. During this project, it became clear that the Texas Instruments environment is mostly built for people who know what they want and already have an idea of how to achieve that. Texas Instruments could benefit from making their software and support more beginner-friendly, as it is quite complicated and confusing when starting out.

In retrospective, it would have been wiser to switch to the BOOSTXL-DRV8301 sooner, because it only became clear what was wrong when the project had progressed quite a bit further. The understanding of the hardware and software came later because it took the students working with the BOOSTXL-DRV8301 to get how the boards interact and how the control works. The BOOSTXL-DRV8323RH can still be used with the LAUNCHXL-F28069M, but before it will work the communication with the boards has to be adjusted. The DRV8323RH works through a GPIO interface, while the DRV8301 works with SPI.

8.2. Recommendations

While some things did not go as planned, this project has still made a good start on the development of a motor controller. It is clear that the path started with this thesis can be a very fruitful one for the Nuon Solar Team. Some recommendations are done for future research based on this thesis.

- While the BOOSTXL-DRV8323RH can be made to work with the LAUNCHXL-F28069M, it is not recommended to put any more time in doing this. The BoosterPacks are just tools to get the simulation set-up running, running demos on the Nuna motors and doing early prototyping on the control. While the DRV8323RH can deliver more power than the DRV8301, it will still not be able to drive the motor properly. For the applications stated, the DRV8301 does the job well enough and the DRV8323RH provides very little benefits. In a further stage of development, another inverter will have to be used anyway, and until then the DRV8301 is sufficient.

- The CAN-interface has been integrated to work with lab 11a, and values can be read and sent while the motor is running. It is still necessary for the Nuon Solar Team to specify exactly which values should be sent over the CAN-bus and what mailboxes to use. With this added the control will be able to properly interface with the rest of the car.
- When looking at inverters in the future, it is probably wise to leave the LaunchPad development kit in use right now. An inverter can be connected to the GPIO pins, and it will work, but the solution is not ideal. The pins on the LAUNCHXL-F28069M are quite fragile and the durability is questionable at best. Instead it is a good idea to design a custom PCB for the TMS320F28069M MCU. This is the same MCU as on the LaunchPad, but with a custom PCB it can be much better integrated. The signals coming from the inverter and the phases need to be processed with tuned circuits and on a custom PCB these could all be integrated in an efficient and durable way. Connections can also be placed in a convenient way. If the Nuon Solar Team decides to also look into building a custom inverter, the measurements and control could even be integrated right into the inverter, making the whole combination a reliable block.

With the basis presented in this thesis and the recommendations named above, developing a motor controller for Nuna should prove a fruitful endeavor, with which they can keep their championship title for years to come!

A
InstaSpin™ manual

A.1. Introduction

This manual is to help people regarding the combination of the TI LAUNCHXL-F28069M in combination with the TI BOOSTXL-DRV8301 and two Teknic inc. Hudson BLDC motors. It helps by telling how the software needed has to be installed and in what order. Furthermore it explains how a FOC control (and CAN capabilities) can be implemented using Motorware.

A.2. Hardware

To build the simulation set-up the following equipment is needed:

- 2x Texas Instruments LAUNCHXL-F28069M

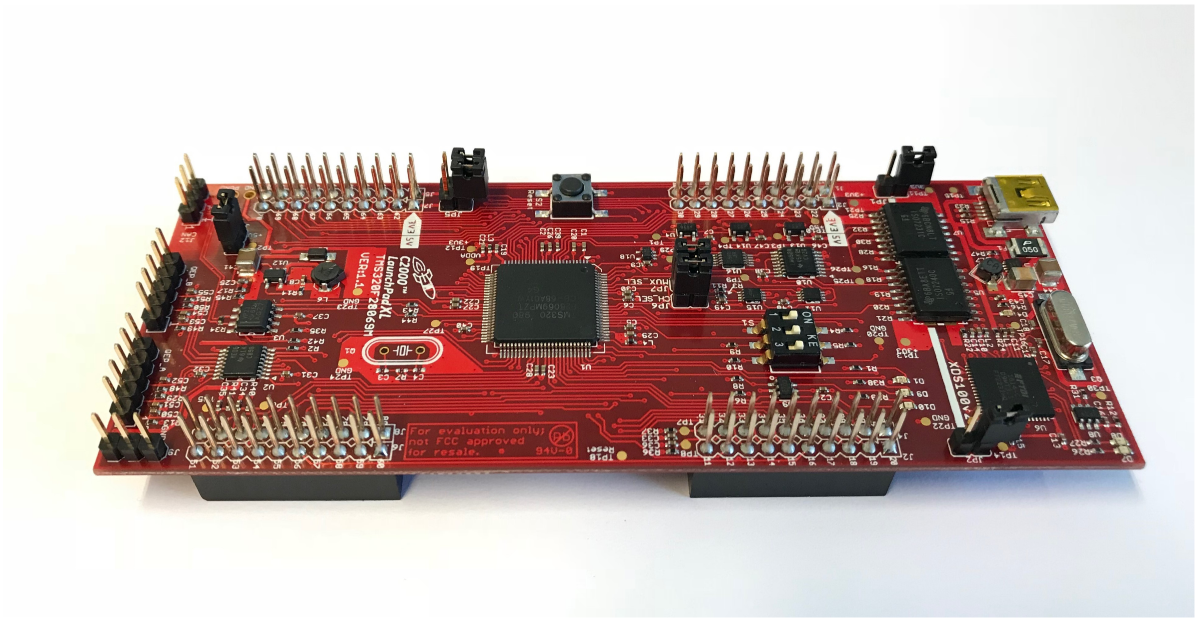


Figure A.1: TI LAUNCHXL-F28069M

- 2x Texas Instruments BOOSTXL-DRV8301 rev. B

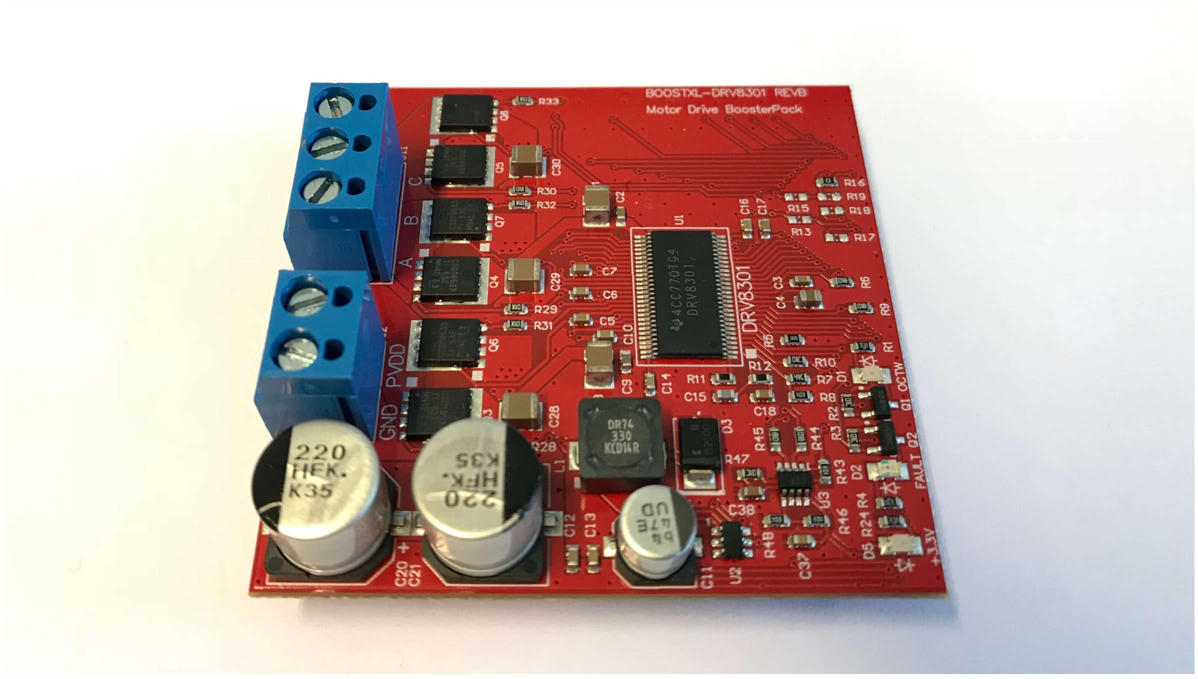


Figure A.2: TI BOOSTXL-DRV8301 Rev. B

- 1x Texas Instruments 2MTR-DYNO InstaSPIN-FOC Evaluation Module (this includes 2x Teknic Hudson M-2310, a shaft coupler and a frame mount for stability)

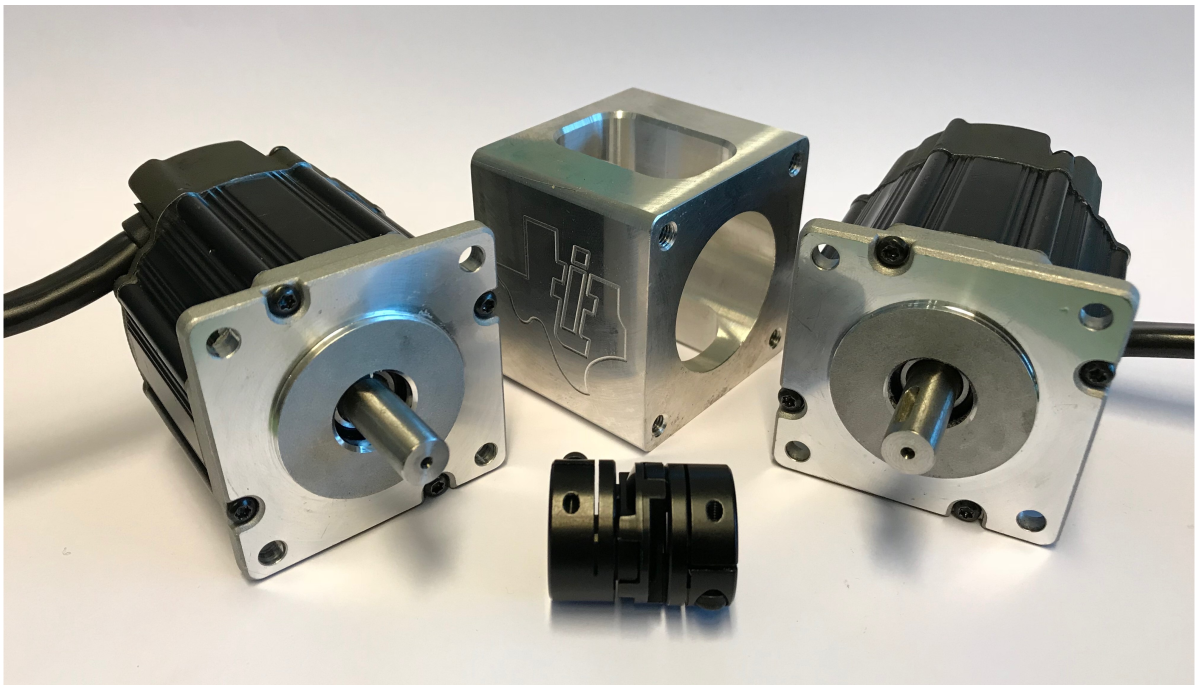


Figure A.3: Components of the 2MTR-DYNO kit

- 1x Power supply of 30 volt and 10 ampère
- Clamps to secure the set-up to the table
- 2x Personal computer running Microsoft Windows (in this manual Windows 10 Home is used)
- Cables to connect the BOOSTXL to the power supply

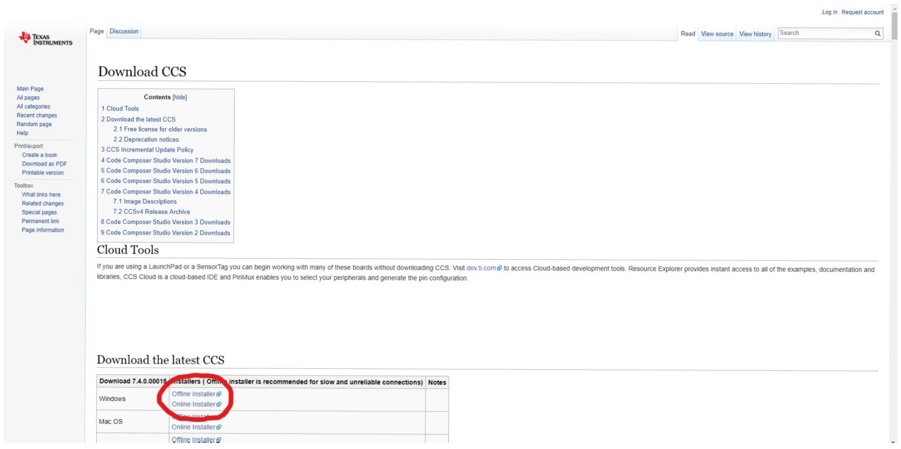
A.3. Software installation

The following software is needed to get the set-up running:

1. Code Composer Studio IDE
2. MotorWare™
3. C2000Ware

A.3.1. Code Composer Studio

1. Go to http://processors.wiki.ti.com/index.php/Download_CCS.
2. Download either the offline or online installer for Windows of the latest version of Code Composer Studio (in this manual the offline installer for Code Composer Studio 7.3.0 is used).



Log in Request account

Page Discussion Read View source View history Search

Main Page
All pages
All categories
Recent changes
Random page
Help
Print/export
Create a book
Download as PDF
Printable version
Toolbox
What links here
Related changes
Special pages
Permanent link
Page information

Download CCS

Contents [hide]
1 Cloud Tools
2 Download the latest CCS
2.1 Free license for older versions
2.2 Deprecation notices:
3 CCS Incremental Update Policy
4 Code Composer Studio Version 7 Downloads
5 Code Composer Studio Version 6 Downloads
6 Code Composer Studio Version 5 Downloads
7 Code Composer Studio Version 4 Downloads
7.1 Image Descriptions
7.2 CCSv4 Release Archive
8 Code Composer Studio Version 3 Downloads
9 Code Composer Studio Version 2 Downloads

Cloud Tools

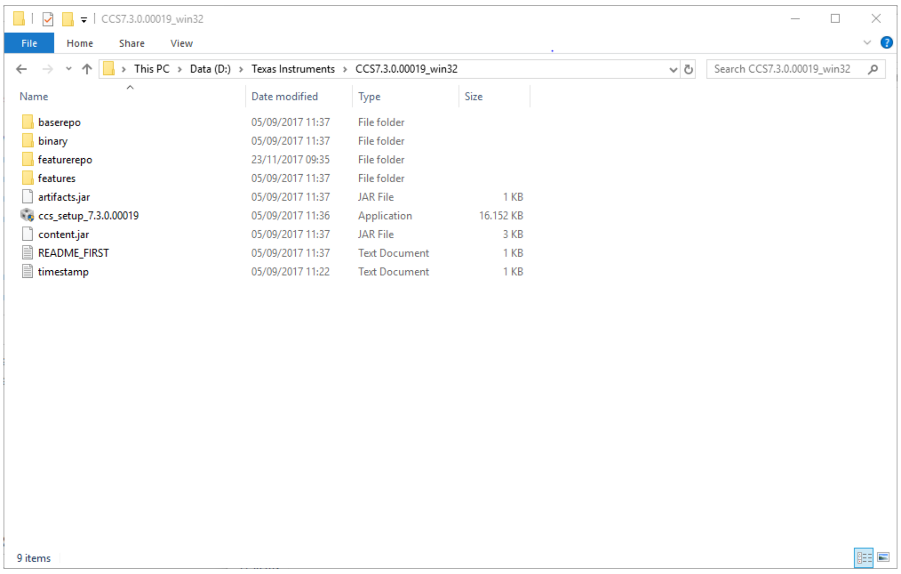
If you are using a LaunchPad or a SensorTag you can begin working with many of these boards without downloading CCS. Visit dev.ti.com to access Cloud-based development tools. Resource Explorer provides instant access to all of the examples, documentation and libraries, CCS Cloud is a cloud-based IDE and PinMux enables you to select your peripherals and generate the pin configuration.

Download the latest CCS

Download 7.4.0.00015	Installers (Offline installer is recommended for slow and unreliable connections)	Notes
Windows	Offline Installer Online Installer	
Mac OS	Offline Installer Online Installer	
	Offline Installer	

Figure A.4: step 2

3. Unzip the file and open css_setup_[version].exe



CCS7.3.0.00019_win32

File Home Share View

This PC › Data (D:) › Texas Instruments › CCS7.3.0.00019_win32

Search CCS7.3.0.00019_win32

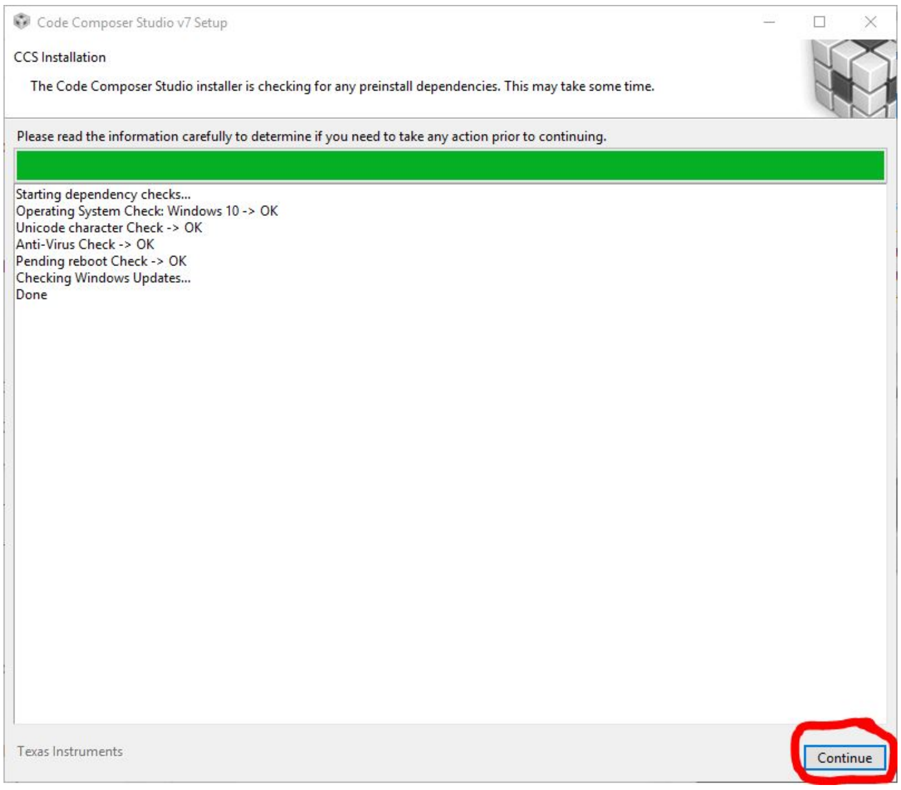
Name	Date modified	Type	Size
baserepo	05/09/2017 11:37	File folder	
binary	05/09/2017 11:37	File folder	
featurerepo	23/11/2017 09:35	File folder	
features	05/09/2017 11:37	File folder	
artifacts.jar	05/09/2017 11:37	JAR File	1 KB
ccs_setup_7.3.0.00019	05/09/2017 11:36	Application	16.152 KB
content.jar	05/09/2017 11:37	JAR File	3 KB
README_FIRST	05/09/2017 11:37	Text Document	1 KB
timestamp	05/09/2017 11:22	Text Document	1 KB

9 items

Figure A.5: step 3

4. A security warning will pop up. Click ”Yes”.

5. Click ”Continue”.



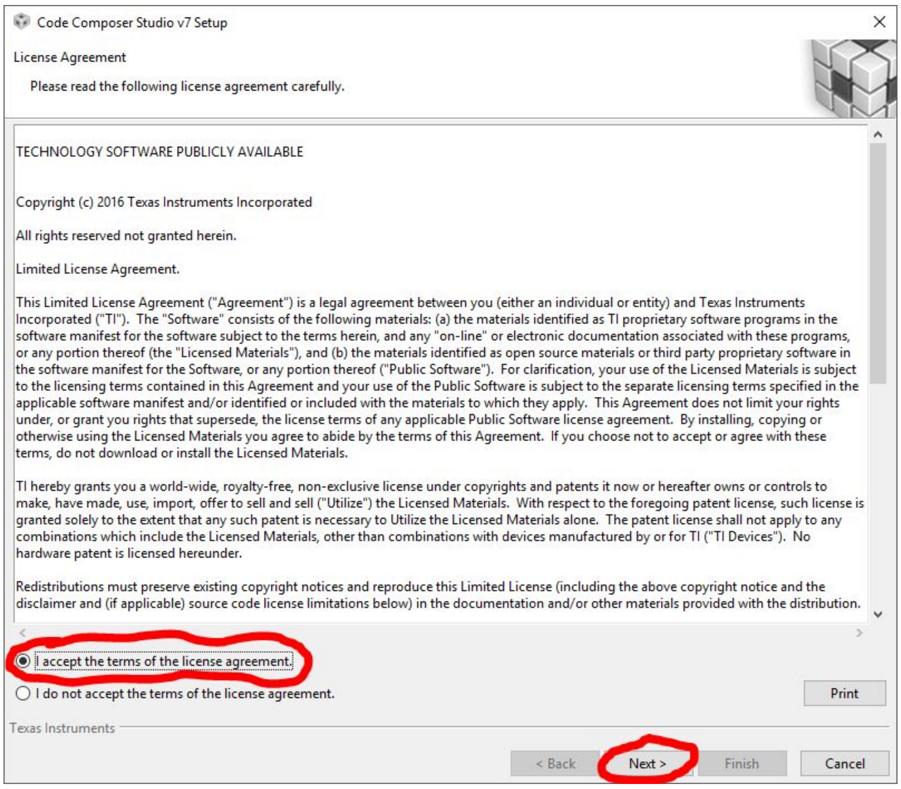
The Code Composer Studio installer is checking for any preinstall dependencies. This may take some time.

Please read the information carefully to determine if you need to take any action prior to continuing.

```
Starting dependency checks...
Operating System Check: Windows 10 -> OK
Unicode character Check -> OK
Anti-Virus Check -> OK
Pending reboot Check -> OK
Checking Windows Updates...
Done
```

Figure A.6: step 5

6. Accept the terms of the license agreement and click ”Next”.


Code Composer Studio v7 Setup

License Agreement

Please read the following license agreement carefully.

TECHNOLOGY SOFTWARE PUBLICLY AVAILABLE

Copyright (c) 2016 Texas Instruments Incorporated

All rights reserved not granted herein.

Limited License Agreement.

This Limited License Agreement ("Agreement") is a legal agreement between you (either an individual or entity) and Texas Instruments Incorporated ("TI"). The "Software" consists of the following materials: (a) the materials identified as TI proprietary software programs in the software manifest for the software subject to the terms herein, and any "on-line" or electronic documentation associated with these programs, or any portion thereof (the "Licensed Materials"), and (b) the materials identified as open source materials or third party proprietary software in the software manifest for the Software, or any portion thereof ("Public Software"). For clarification, your use of the Licensed Materials is subject to the licensing terms contained in this Agreement and your use of the Public Software is subject to the separate licensing terms specified in the applicable software manifest and/or identified or included with the materials to which they apply. This Agreement does not limit your rights under, or grant you rights that supersede, the license terms of any applicable Public Software license agreement. By installing, copying or otherwise using the Licensed Materials you agree to abide by the terms of this Agreement. If you choose not to accept or agree with these terms, do not download or install the Licensed Materials.

TI hereby grants you a world-wide, royalty-free, non-exclusive license under copyrights and patents it now or hereafter owns or controls to make, have made, use, import, offer to sell and sell ("Utilize") the Licensed Materials. With respect to the foregoing patent license, such license is granted solely to the extent that any such patent is necessary to Utilize the Licensed Materials alone. The patent license shall not apply to any combinations which include the Licensed Materials, other than combinations with devices manufactured by or for TI ("TI Devices"). No hardware patent is licensed hereunder.

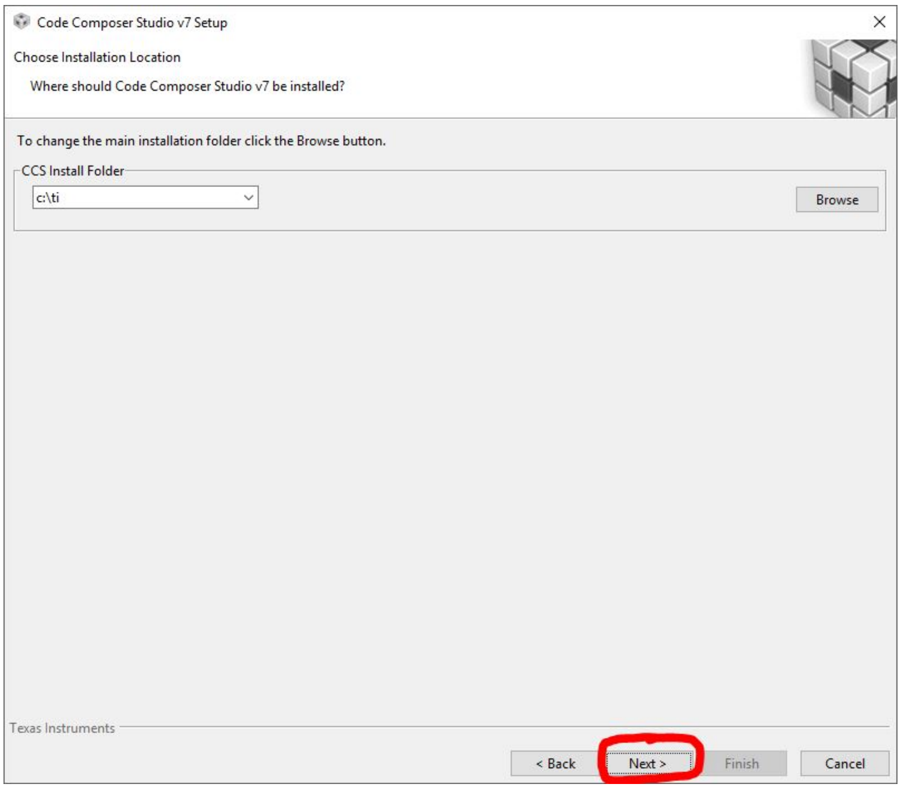
Redistributions must preserve existing copyright notices and reproduce this Limited License (including the above copyright notice and the disclaimer and (if applicable) source code license limitations below) in the documentation and/or other materials provided with the distribution.

[✓ I accept the terms of the license agreement.]

I do not accept the terms of the license agreement.

Figure A.7: step 6

7. Select a directory to install to and click ”Next”. (c:\ti is recommended).


Code Composer Studio v7 Setup

Choose Installation Location

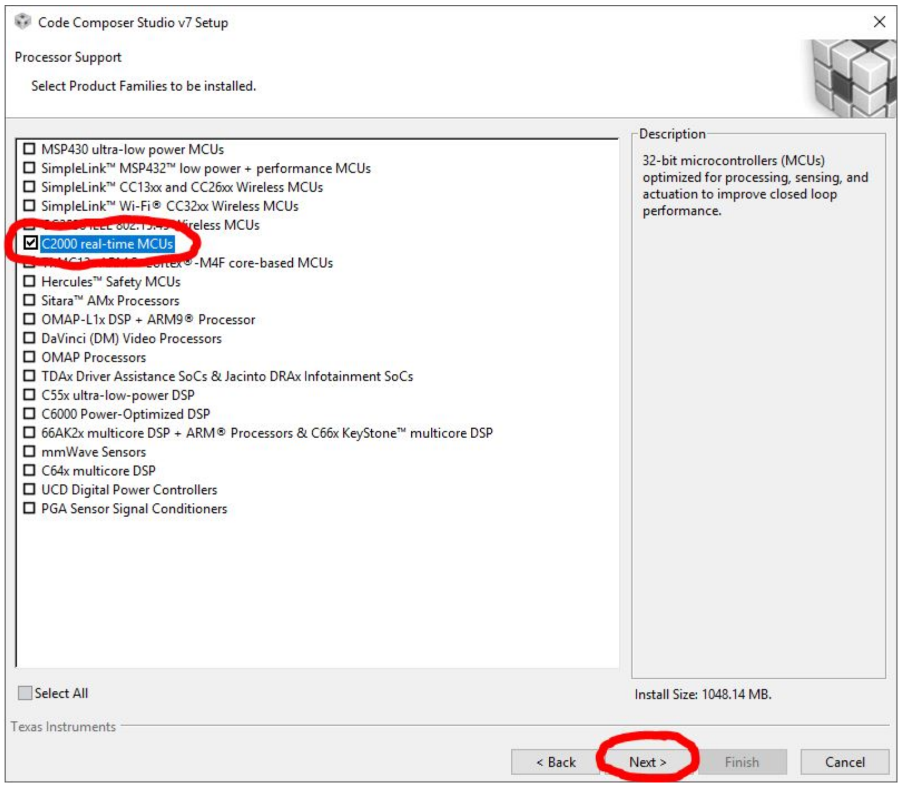
Where should Code Composer Studio v7 be installed?

To change the main installation folder click the Browse button.

CCS Install Folder: c:\ti

Figure A.8: step 7

8. Select the C2000 real-time MCUs family to be installed and click ”Next”.



Code Composer Studio v7 Setup

Processor Support

Select Product Families to be installed.

- ☐ MSP430 ultra-low power MCUs
- ☐ SimpleLink™ MSP432™ low power + performance MCUs
- ☐ SimpleLink™ CC13xx and CC26xx Wireless MCUs
- ☐ SimpleLink™ Wi-Fi® CC32xx Wireless MCUs
- ☐ CC2538 IEEE 802.15.4 Wireless MCUs
- ☑ C2000 real-time MCUs
- ☐ TM4C12x ARM® Cortex®-M4F core-based MCUs
- ☐ Hercules™ Safety MCUs
- ☐ Sitara™ AMx Processors
- ☐ OMAP-L1x DSP + ARM9® Processor
- ☐ DaVinci (DM) Video Processors
- ☐ OMAP Processors
- ☐ TDAx Driver Assistance SoCs & Jacinto DRAx Infotainment SoCs
- ☐ C55x ultra-low-power DSP
- ☐ C6000 Power-Optimized DSP
- ☐ 66AK2x multicore DSP + ARM® Processors & C66x KeyStone™ multicore DSP
- ☐ mmWave Sensors
- ☐ C64x multicore DSP
- ☐ UCD Digital Power Controllers
- ☐ PGA Sensor Signal Conditioners

Description

32-bit microcontrollers (MCUs) optimized for processing, sensing, and actuation to improve closed loop performance.

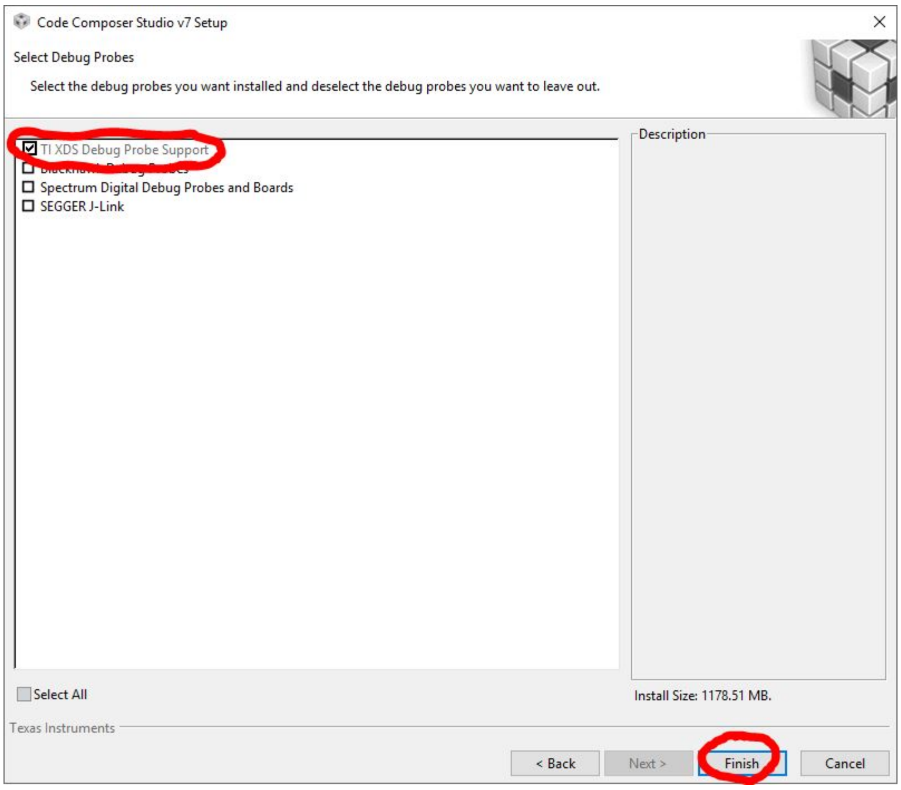
☐ Select All

Install Size: 1048.14 MB.

Texas Instruments

Figure A.9: step 8

9. Make sure the TI XDS Debug Probe Support is selected and click ”Finish”.


Code Composer Studio v7 Setup

Select Debug Probes

Select the debug probes you want installed and deselect the debug probes you want to leave out.

- [✓] TI XDS Debug Probe Support
- [ ] Blackhawk Debug Probes
- [ ] Spectrum Digital Debug Probes and Boards
- [ ] SEGGER J-Link

Description

- [ ] Select All

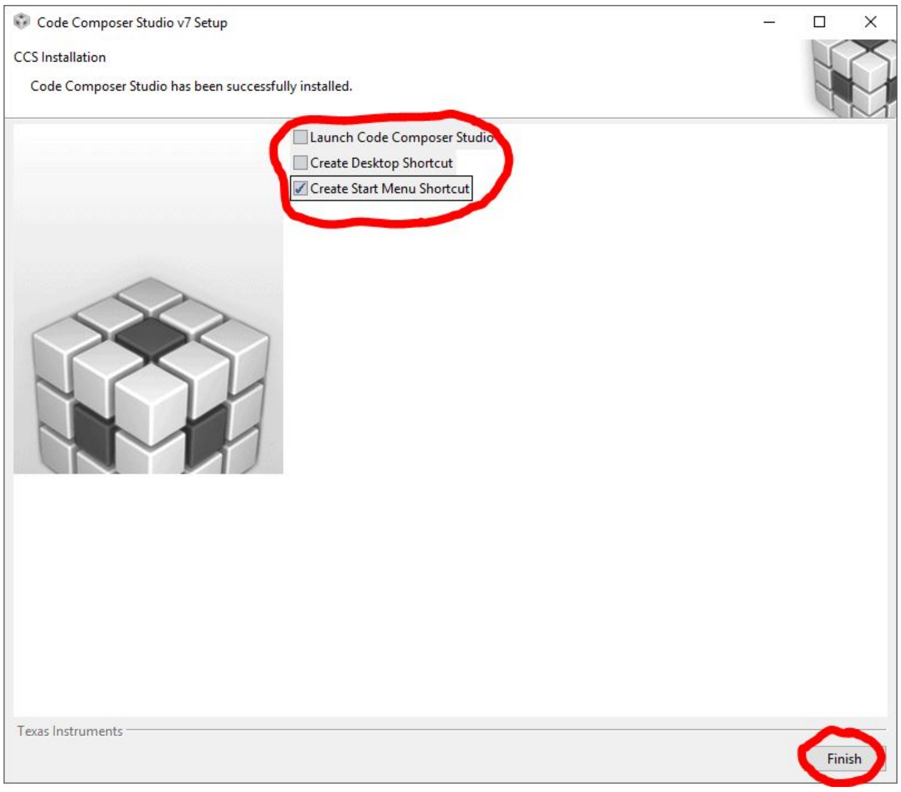
Install Size: 1178.51 MB.

Texas Instruments

Figure A.10: step 9

10. Wait for the installer to install Code Composer Studio and the packages (this will take a few minutes).

11. Deselect ”Launch Code Composer Studio” and any other options you don’t want.


Code Composer Studio v7 Setup

CCS Installation

Code Composer Studio has been successfully installed.

- [ ] Launch Code Composer Studio
- [ ] Create Desktop Shortcut
- [✓] Create Start Menu Shortcut

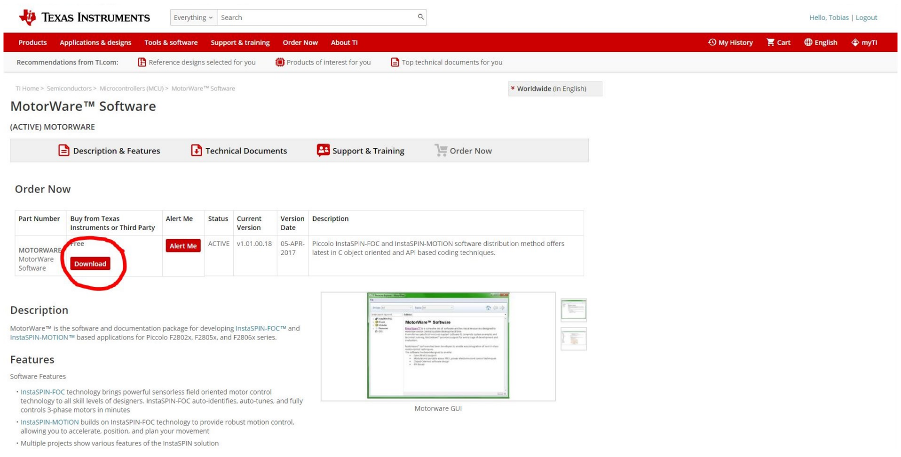
Texas Instruments

Figure A.11: step 11

A.3.2. MotorWare™

1. Go to http://www.ti.com/tool/MOTORWARE.

2. Click the download button.



Recommendations from TI.com: Reference designs selected for you · Products of interest for you · Top technical documents for you

TI Home > Semiconductors > Microcontrollers (MCU) > MotorWare™ Software

Worldwide (In English)

MotorWare™ Software

(ACTIVE) MOTORWARE

Order Now

Part Number	Buy from Texas Instruments or Third Party	Alert Me	Status	Current Version	Version Date	Description
MOTORWARE MotorWare Software	Free Download	Alert Me	ACTIVE	v1.01.00.18	05-APR-2017	Piccolo InstaSPIN-FOC and InstaSPIN-MOTION software distribution method offers latest in C object oriented and API based coding techniques.

Description

MotorWare™ is the software and documentation package for developing InstaSPIN-FOC™ and InstaSPIN-MOTION™ based applications for Piccolo F2802x, F2805x, and F2806x series.

Features

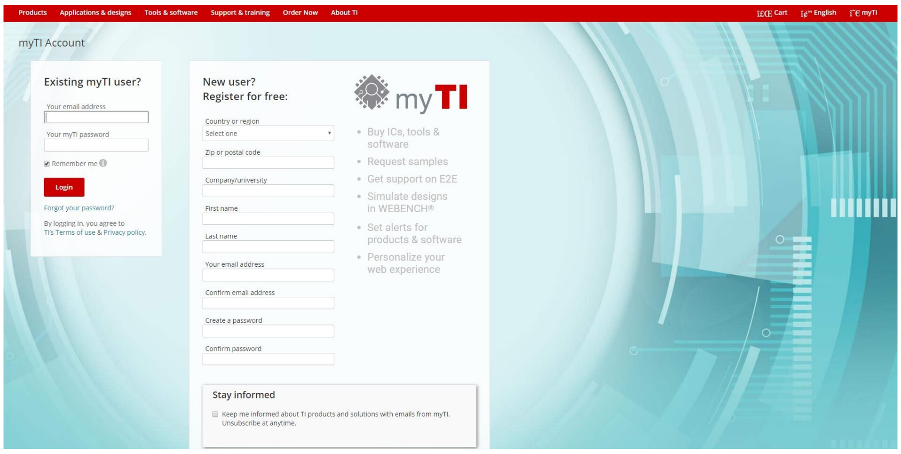
Software Features

- InstaSPIN-FOC technology brings powerful sensorless field oriented motor control technology to all skill levels of designers. InstaSPIN-FOC auto-identifies, auto-tunes, and fully controls 3-phase motors in minutes
- InstaSPIN-MOTION builds on InstaSPIN-FOC technology to provide robust motion control, allowing you to accelerate, position, and plan your movement
- Multiple projects show various features of the InstaSPIN solution

Motorware GUI

Figure A.12: step 2

3. Register a new MyTI account or log in if you already have an account.



myTI Account

Existing myTI user?

Your email address

Your myTI password

☑ Remember me

Forgot your password?

By logging in, you agree to TI's Terms of use & Privacy policy.

New user?
Register for free:

Country or region: Select one

Zip or postal code

Company/university

First name

Last name

Your email address

Confirm email address

Create a password

Confirm password

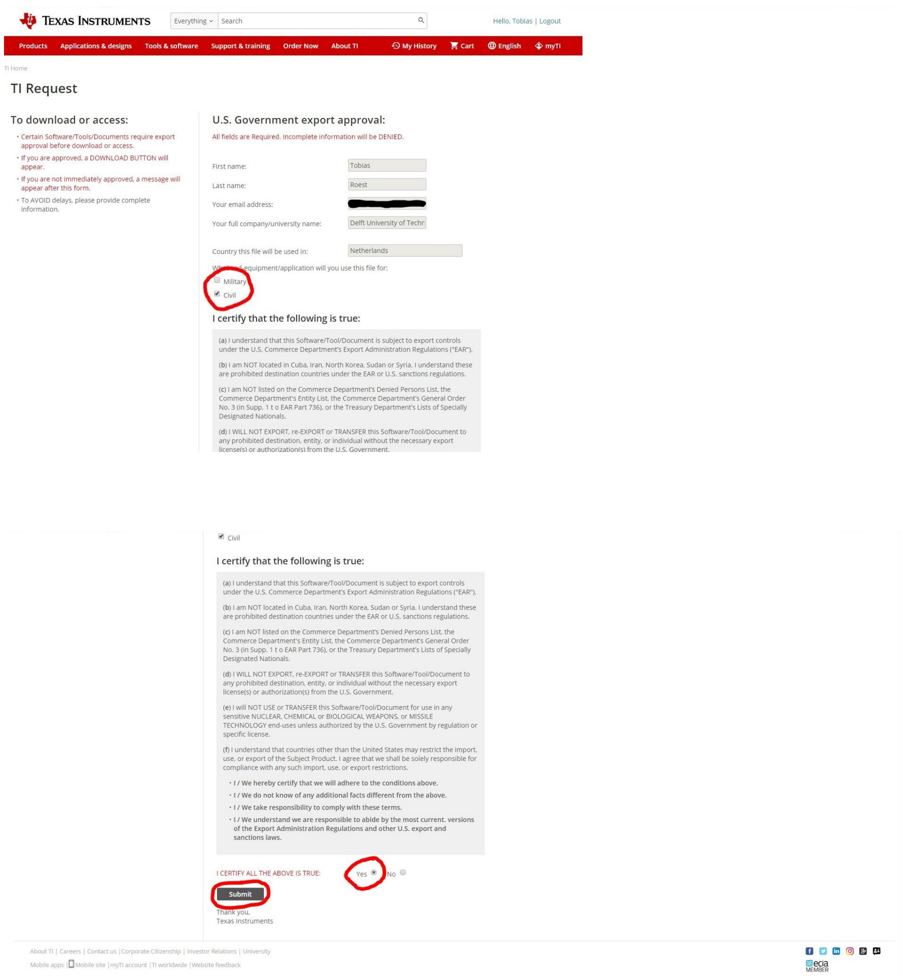
- Buy ICs, tools & software
- Request samples
- Get support on E2E
- Simulate designs in WEBENCH®
- Set alerts for products & software
- Personalize your web experience

Stay informed

☐ Keep me informed about TI products and solutions with emails from myTI. Unsubscribe at anytime.

Figure A.13: step 3

4. You will need to fill out a U.S. Government Export Approval form to be able to download MotorWare™. Select ”Civil” in the application category, and select ”Yes” at ”I CERTIFY ALL OF THE ABOVE IS TRUE:”. Then submit the form.



TI Request

To download or access:

- Certain Software/Tools/Documents require export approval before download or access.
- If you are approved, a DOWNLOAD BUTTON will appear.
- If you are not immediately approved, a message will appear after this form.
- To AVOID delays, please provide complete information.

U.S. Government export approval:

All fields are Required. Incomplete information will be DENIED.

First name:	Tobias
Last name:	Roest
Your email address:	[redacted]
Your full company/university name:	Delft University of Techn
Country this file will be used in:	Netherlands

What end-equipment/application will you use this file for:

☐ Military
[✓ Civil]

I certify that the following is true:

(a) I understand that this Software/Tool/Document is subject to export controls under the U.S. Commerce Department's Export Administration Regulations ("EAR").

(b) I am NOT located in Cuba, Iran, North Korea, Sudan or Syria. I understand these are prohibited destination countries under the EAR or U.S. sanctions regulations.

(c) I am NOT listed on the Commerce Department's Denied Persons List, the Commerce Department's Entity List, the Commerce Department's General Order No. 3 (in Supp. 1 t o EAR Part 736), or the Treasury Department's Lists of Specially Designated Nationals.

(d) I WILL NOT EXPORT, re-EXPORT or TRANSFER this Software/Tool/Document to any prohibited destination, entity, or individual without the necessary export license(s) or authorization(s) from the U.S. Government.

☑ Civil

I certify that the following is true:

(a) I understand that this Software/Tool/Document is subject to export controls under the U.S. Commerce Department's Export Administration Regulations ("EAR").

(b) I am NOT located in Cuba, Iran, North Korea, Sudan or Syria. I understand these are prohibited destination countries under the EAR or U.S. sanctions regulations.

(c) I am NOT listed on the Commerce Department's Denied Persons List, the Commerce Department's Entity List, the Commerce Department's General Order No. 3 (in Supp. 1 t o EAR Part 736), or the Treasury Department's Lists of Specially Designated Nationals.

(d) I WILL NOT EXPORT, re-EXPORT or TRANSFER this Software/Tool/Document to any prohibited destination, entity, or individual without the necessary export license(s) or authorization(s) from the U.S. Government.

(e) I will NOT USE or TRANSFER this Software/Tool/Document for use in any sensitive NUCLEAR, CHEMICAL or BIOLOGICAL WEAPONS, or MISSILE TECHNOLOGY end-uses unless authorized by the U.S. Government by regulation or specific license.

(f) I understand that countries other than the United States may restrict the import, use, or export of the Subject Product. I agree that we shall be solely responsible for compliance with any such import, use, or export restrictions.

- **I / We hereby certify that we will adhere to the conditions above.**
- **I / We do not know of any additional facts different from the above.**
- **I / We take responsibility to comply with these terms.**
- **I / We understand we are responsible to abide by the most current. versions of the Export Administration Regulations and other U.S. export and sanctions laws.**

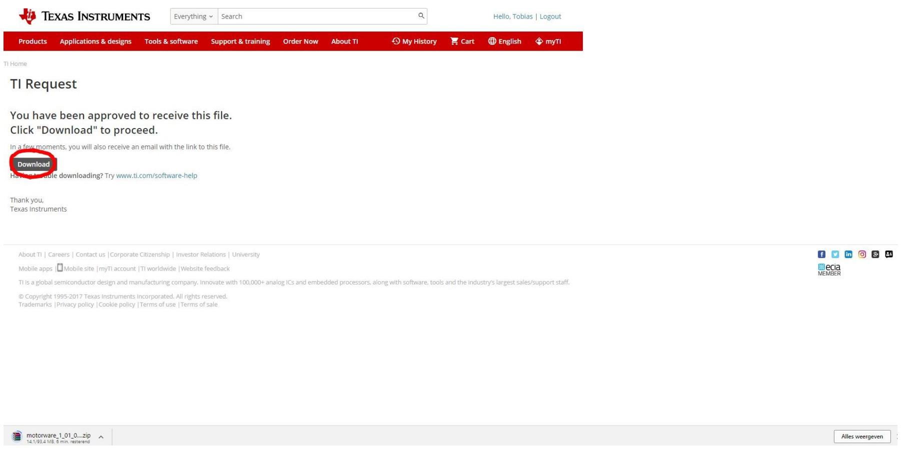
I CERTIFY ALL THE ABOVE IS TRUE: [✓ Yes] No

Thank you,
Texas Instruments

About TI | Careers | Contact us | Corporate Citizenship | Investor Relations | University
Mobile apps | Mobile site | myTI account | TI worldwide | Website feedback
ecia MEMBER

Figure A.14: step 4

5. Click ”Download”.



TI Request

You have been approved to receive this file.
Click "Download" to proceed.

In a few moments, you will also receive an email with the link to this file.

Having trouble downloading? Try www.ti.com/software-help

Thank you,
Texas Instruments

About TI | Careers | Contact us | Corporate Citizenship | Investor Relations | University
Mobile apps | Mobile site | myTI account | TI worldwide | Website feedback

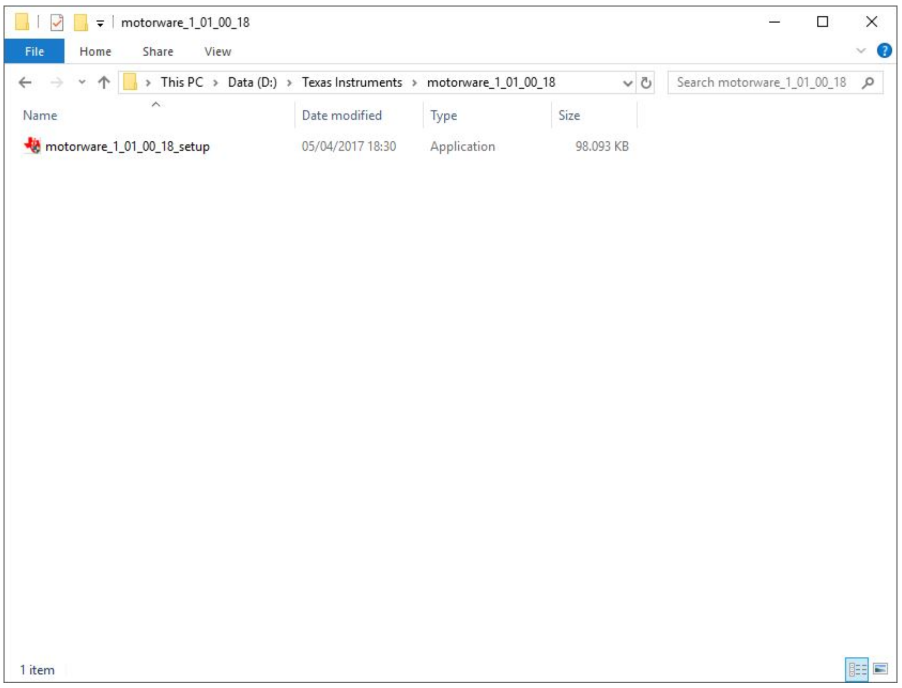
TI is a global semiconductor design and manufacturing company. Innovate with 100,000+ analog ICs and embedded processors, along with software, tools and the industry's largest sales/support staff.

© Copyright 1995-2017 Texas Instruments Incorporated. All rights reserved.

Trademarks | Privacy policy | Cookie policy | Terms of use | Terms of sale
ecia MEMBER
motorware_1_01_0....zip
14.1/93.4 MB, 6 min. resterend
Alles weergeven

Figure A.15: step 5

6. Unzip motorware_[version].zip and run motorware_[version]_setup.exe

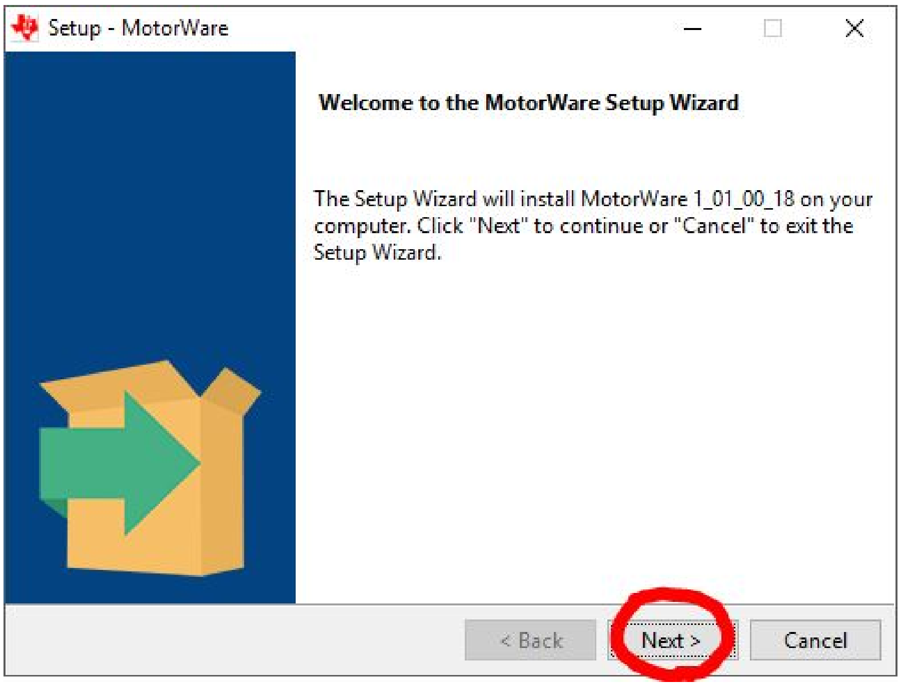

motorware_1_01_00_18
File
Home
Share
View
This PC > Data (D:) > Texas Instruments > motorware_1_01_00_18
Search motorware_1_01_00_18

Name	Date modified	Type	Size
motorware_1_01_00_18_setup	05/04/2017 18:30	Application	98.093 KB

Figure A.16: step 6

7. A security warning will pop up. Click ”Yes”.

8. Click ”Next”.

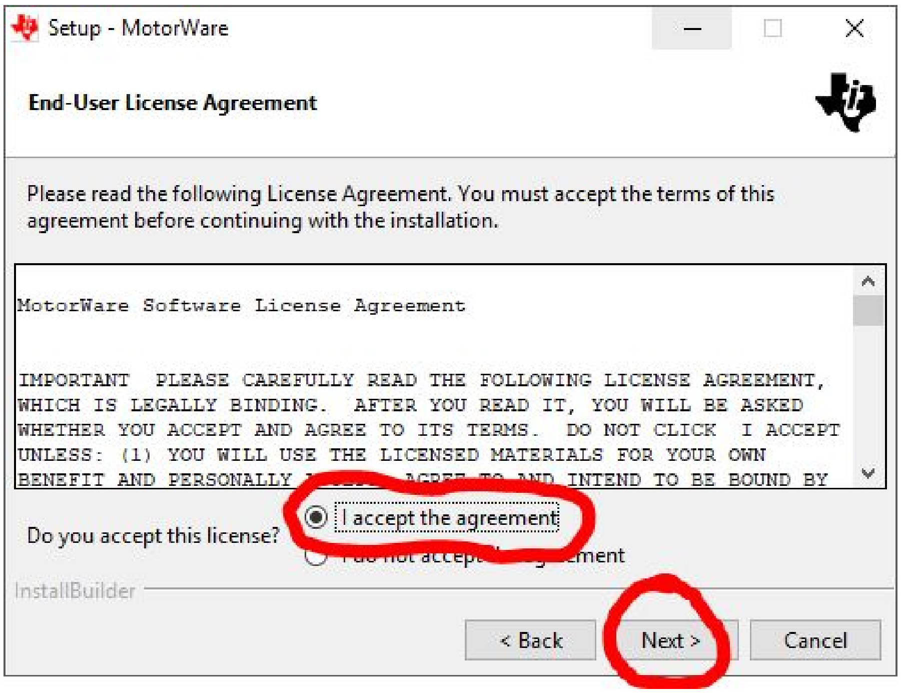


Welcome to the MotorWare Setup Wizard

The Setup Wizard will install MotorWare 1_01_00_18 on your computer. Click "Next" to continue or "Cancel" to exit the Setup Wizard.

Figure A.17: step 8

9. Accept the End-User License Agreement and click ”Next”.



End-User License Agreement

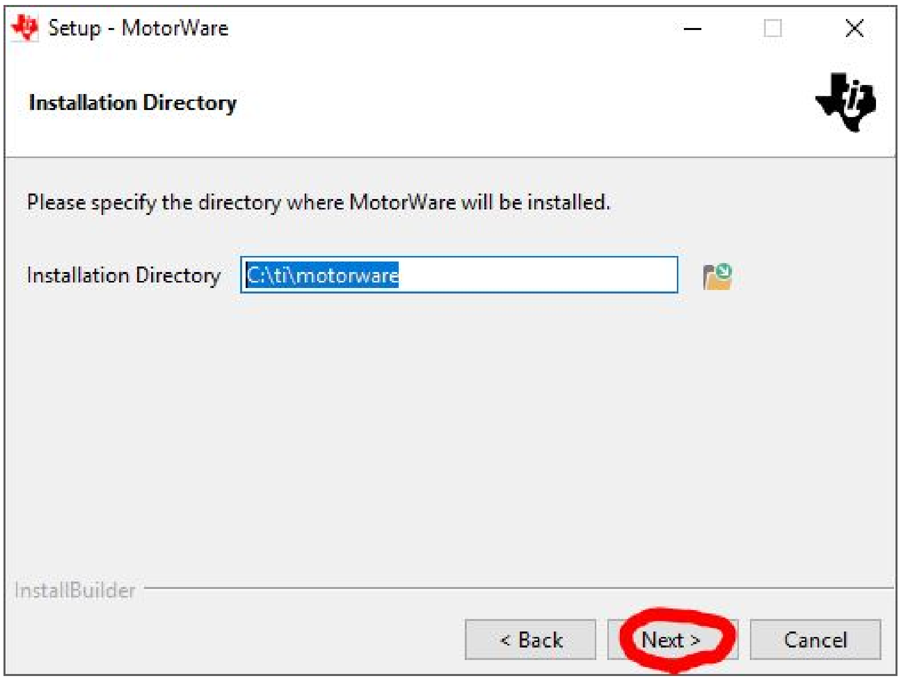
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

```
MotorWare Software License Agreement


IMPORTANT  PLEASE CAREFULLY READ THE FOLLOWING LICENSE AGREEMENT,
WHICH IS LEGALLY BINDING.  AFTER YOU READ IT, YOU WILL BE ASKED
WHETHER YOU ACCEPT AND AGREE TO ITS TERMS.  DO NOT CLICK  I ACCEPT
UNLESS: (1) YOU WILL USE THE LICENSED MATERIALS FOR YOUR OWN
BENEFIT AND PERSONALLY                AND INTEND TO BE BOUND BY
```

Do you accept this license? [✓ I accept the agreement] I do not accept the agreement

Figure A.18: step 9

10. Select a directory to install to and click ”Next”. (c:\ti\motorware is recommended). Make sure the directory is in the same folder as Code Composer Studio.

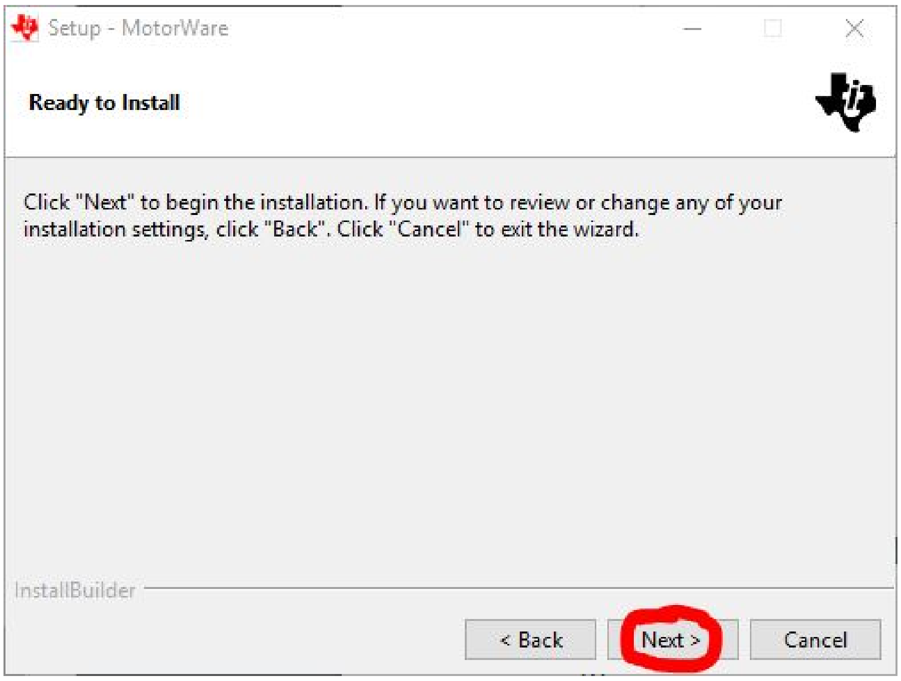


Please specify the directory where MotorWare will be installed.

Installation Directory: C:\ti\motorware

Figure A.19: step 10

11. Click "Next".

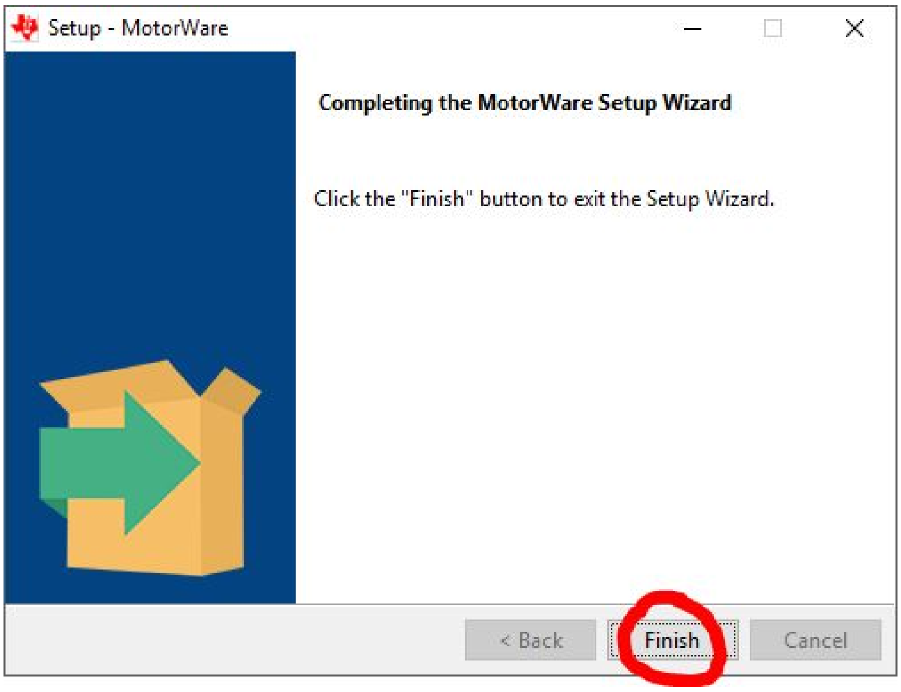


Click "Next" to begin the installation. If you want to review or change any of your installation settings, click "Back". Click "Cancel" to exit the wizard.

Figure A.20: step 11

12. Wait for the installer to install MotorWare™.

13. Click "Finish".



Completing the MotorWare Setup Wizard

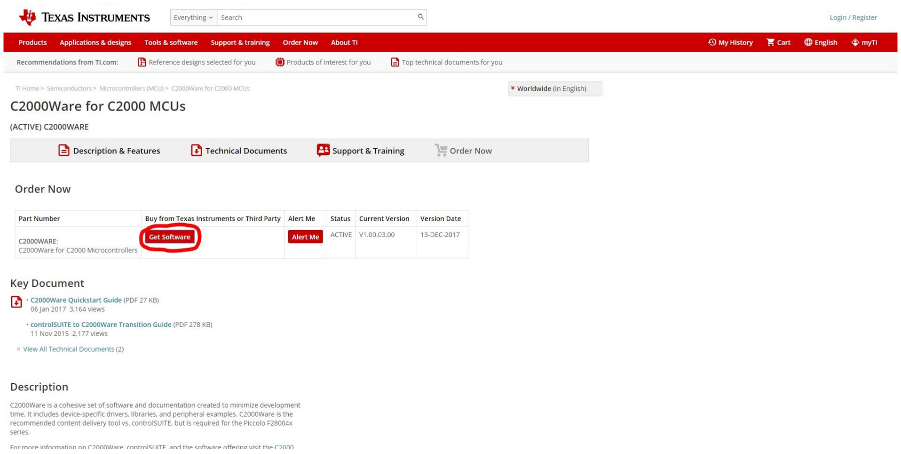
Click the "Finish" button to exit the Setup Wizard.

Figure A.21: step 13

A.3.3. C2000Ware

1. Go to http://www.ti.com/tool/C2000WARE.

2. Click the ”Get Software button.



C2000Ware for C2000 MCUs

(ACTIVE) C2000WARE

Order Now

Part Number	Buy from Texas Instruments or Third Party	Alert Me	Status	Current Version	Version Date
C2000WARE: C2000Ware for C2000 Microcontrollers	Get Software	Alert Me	ACTIVE	V1.00.03.00	13-DEC-2017

Key Document

- C2000Ware Quickstart Guide (PDF 27 KB)
  06 Jan 2017 3,164 views
- controlSUITE to C2000Ware Transition Guide (PDF 278 KB)
  11 Nov 2015 2,177 views

» View All Technical Documents (2)

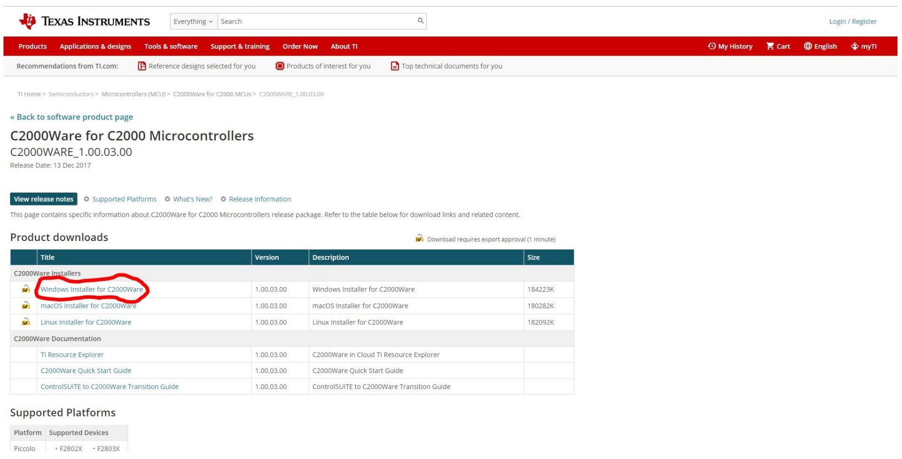
Description

C2000Ware is a cohesive set of software and documentation created to minimize development time. It includes device-specific drivers, libraries, and peripheral examples. C2000Ware is the recommended content delivery tool vs. controlSUITE, but is required for the Piccolo F28004x series.

For more information on C2000Ware, controlSUITE, and the software offering visit the C2000

Figure A.22: step 2

3. Click the link for the Windows installer.



TI Home > Semiconductors > Microcontrollers (MCU) > C2000Ware for C2000 MCUs > C2000WARE_1.00.03.00

« Back to software product page

C2000Ware for C2000 Microcontrollers
C2000WARE_1.00.03.00
Release Date: 13 Dec 2017

This page contains specific information about C2000Ware for C2000 Microcontrollers release package. Refer to the table below for download links and related content.

Product downloads

Download requires export approval (1 minute)

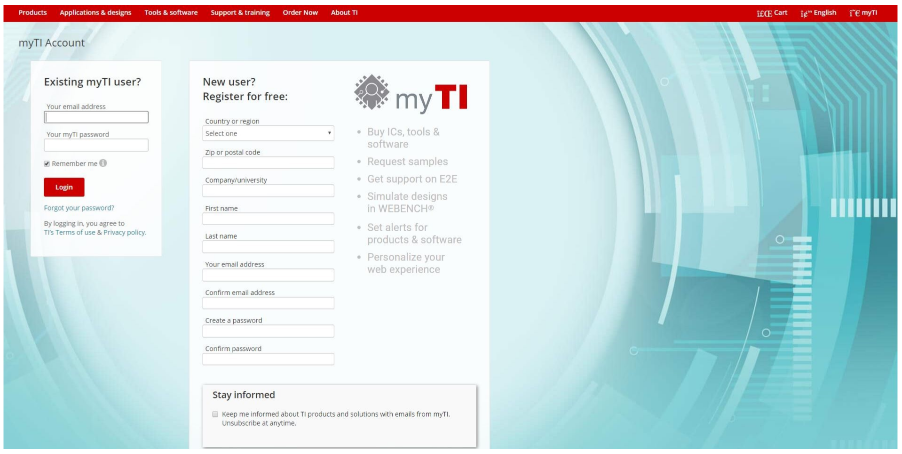
Title	Version	Description	Size
C2000Ware Installers			
Windows Installer for C2000Ware	1.00.03.00	Windows Installer for C2000Ware	184223K
macOS Installer for C2000Ware	1.00.03.00	macOS Installer for C2000Ware	180282K
Linux Installer for C2000Ware	1.00.03.00	Linux Installer for C2000Ware	182092K
C2000Ware Documentation			
TI Resource Explorer	1.00.03.00	C2000Ware in Cloud TI Resource Explorer	
C2000Ware Quick Start Guide	1.00.03.00	C2000Ware Quick Start Guide	
ControlSUITE to C2000Ware Transition Guide	1.00.03.00	ControlSUITE to C2000Ware Transition Guide	

Supported Platforms

Platform	Supported Devices
Piccolo	• F2802X • F2803X

Figure A.23: step 3

4. Register a new MyTI account or log in if you already have an account. (Account as probably been made before while installing MotorWare™).



myTI Account

Existing myTI user?

Your email address

Your myTI password

Remember me

Forgot your password?

By logging in, you agree to TI's Terms of use & Privacy policy.

New user?
Register for free:

Country or region: Select one

Zip or postal code

Company/university

First name

Last name

Your email address

Confirm email address

Create a password

Confirm password

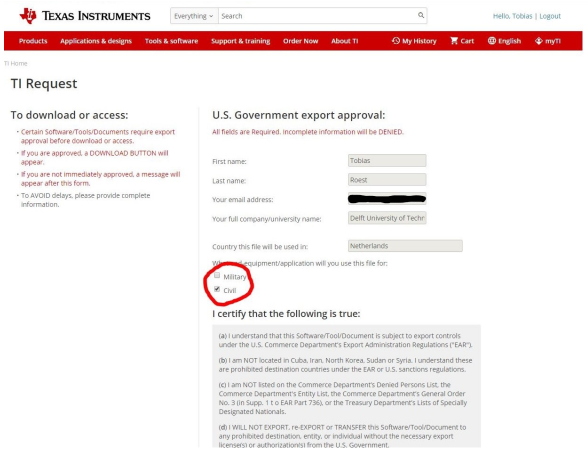
- Buy ICs, tools & software
- Request samples
- Get support on E2E
- Simulate designs in WEBENCH®
- Set alerts for products & software
- Personalize your web experience

Stay informed

Keep me informed about TI products and solutions with emails from myTI. Unsubscribe at anytime.

Figure A.24: step 4

5. You will need to fill out a U.S. Government Export Approval form to be able to download MotorWare™. Select "Civil" in the application category, and select "Yes" at "I CERTIFY ALL OF THE ABOVE IS TRUE:". Then submit the form.



TI Home

TI Request

To download or access:

- Certain Software/Tools/Documents require export approval before download or access.
- If you are approved, a DOWNLOAD BUTTON will appear.
- If you are not immediately approved, a message will appear after this form.
- To AVOID delays, please provide complete information.

U.S. Government export approval:

All fields are Required. Incomplete information will be DENIED.

First name:	Tobias
Last name:	Roest
Your email address:	[redacted]
Your full company/university name:	Delft University of Techn
Country this file will be used in:	Netherlands

What end-equipment/application will you use this file for:

Military [✓ Civil]

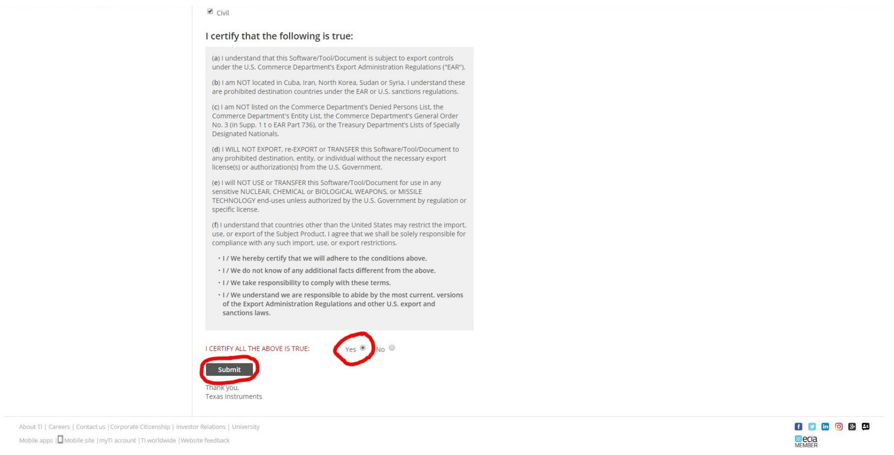
I certify that the following is true:

(a) I understand that this Software/Tool/Document is subject to export controls under the U.S. Commerce Department's Export Administration Regulations ("EAR").

(b) I am NOT located in Cuba, Iran, North Korea, Sudan or Syria. I understand these are prohibited destination countries under the EAR or U.S. sanctions regulations.

(c) I am NOT listed on the Commerce Department's Denied Persons List, the Commerce Department's Entity List, the Commerce Department's General Order No. 3 (in Supp. 1 t o EAR Part 736), or the Treasury Department's Lists of Specially Designated Nationals.

(d) I WILL NOT EXPORT, re-EXPORT or TRANSFER this Software/Tool/Document to any prohibited destination, entity, or individual without the necessary export license(s) or authorization(s) from the U.S. Government.



☑ Civil

I certify that the following is true:

(a) I understand that this Software/Tool/Document is subject to export controls under the U.S. Commerce Department's Export Administration Regulations ("EAR").

(b) I am NOT located in Cuba, Iran, North Korea, Sudan or Syria. I understand these are prohibited destination countries under the EAR or U.S. sanctions regulations.

(c) I am NOT listed on the Commerce Department's Denied Persons List, the Commerce Department's Entity List, the Commerce Department's General Order No. 3 (in Supp. 1 t o EAR Part 736), or the Treasury Department's Lists of Specially Designated Nationals.

(d) I WILL NOT EXPORT, re-EXPORT or TRANSFER this Software/Tool/Document to any prohibited destination, entity, or individual without the necessary export license(s) or authorization(s) from the U.S. Government.

(e) I will NOT USE or TRANSFER this Software/Tool/Document for use in any sensitive NUCLEAR, CHEMICAL or BIOLOGICAL WEAPONS, or MISSILE TECHNOLOGY end-uses unless authorized by the U.S. Government by regulation or specific license.

(f) I understand that countries other than the United States may restrict the import, use, or export of the Subject Product. I agree that we shall be solely responsible for compliance with any such import, use, or export restrictions.

- I / We hereby certify that we will adhere to the conditions above.
- I / We do not know of any additional facts different from the above.
- I / We take responsibility to comply with these terms.
- I / We understand we are responsible to abide by the most current. versions of the Export Administration Regulations and other U.S. export and sanctions laws.

I CERTIFY ALL THE ABOVE IS TRUE: [✓ Yes] No

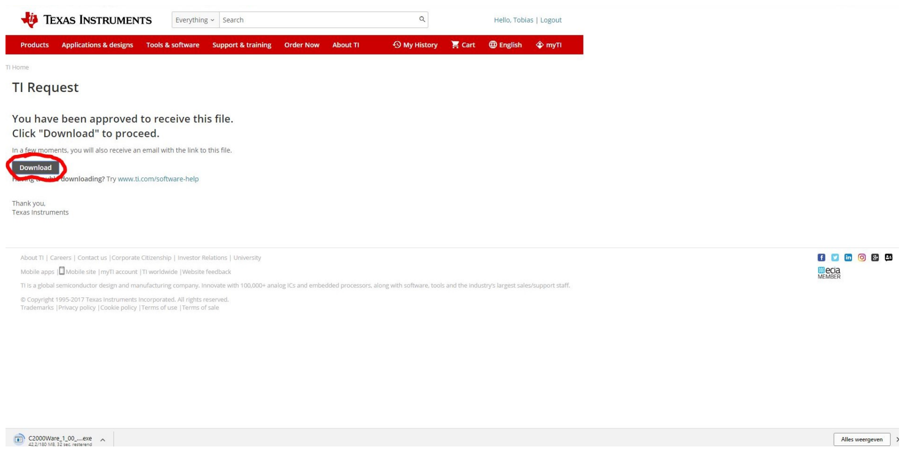
Submit

Thank you,
Texas Instruments

About TI | Careers | Contact us | Corporate Citizenship | Investor Relations | University
Mobile apps | Mobile site | myTI account | TI worldwide | Website feedback

Figure A.25: step 5

6. Click "Download".



TI Request

You have been approved to receive this file.
Click "Download" to proceed.

In a few moments, you will also receive an email with the link to this file.

Download

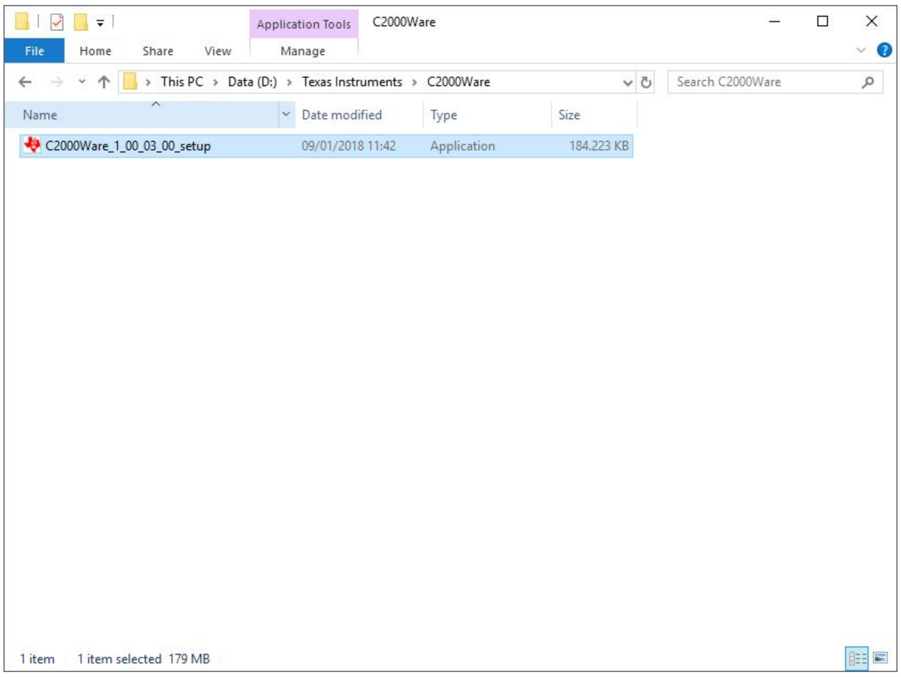
Having trouble downloading? Try www.ti.com/software-help

Thank you,
Texas Instruments

About TI | Careers | Contact us | Corporate Citizenship | Investor Relations | University
Mobile apps | Mobile site | myTI account | TI worldwide | Website feedback
TI is a global semiconductor design and manufacturing company. Innovate with 100,000+ analog ICs and embedded processors, along with software, tools and the industry's largest sales/support staff.
© Copyright 1995-2017 Texas Instruments Incorporated. All rights reserved.
Trademarks | Privacy policy | Cookie policy | Terms of use | Terms of sale

Figure A.26: step 6

7. Run C2000Ware_[version]_setup.exe

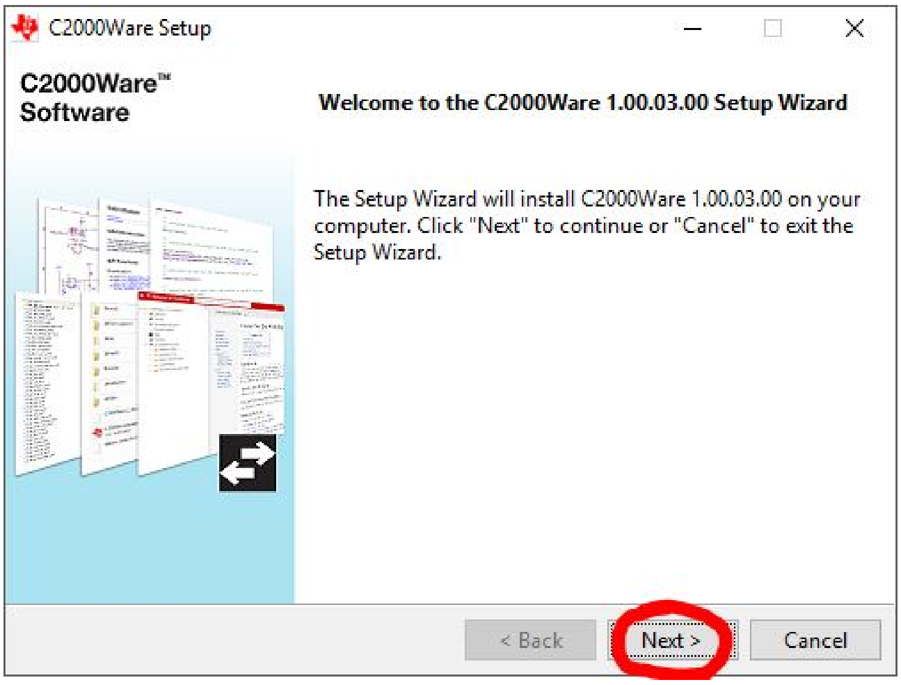


Name	Date modified	Type	Size
C2000Ware_1_00_03_00_setup	09/01/2018 11:42	Application	184.223 KB

1 item 1 item selected 179 MB

Figure A.27: step 7

8. Click "Next".

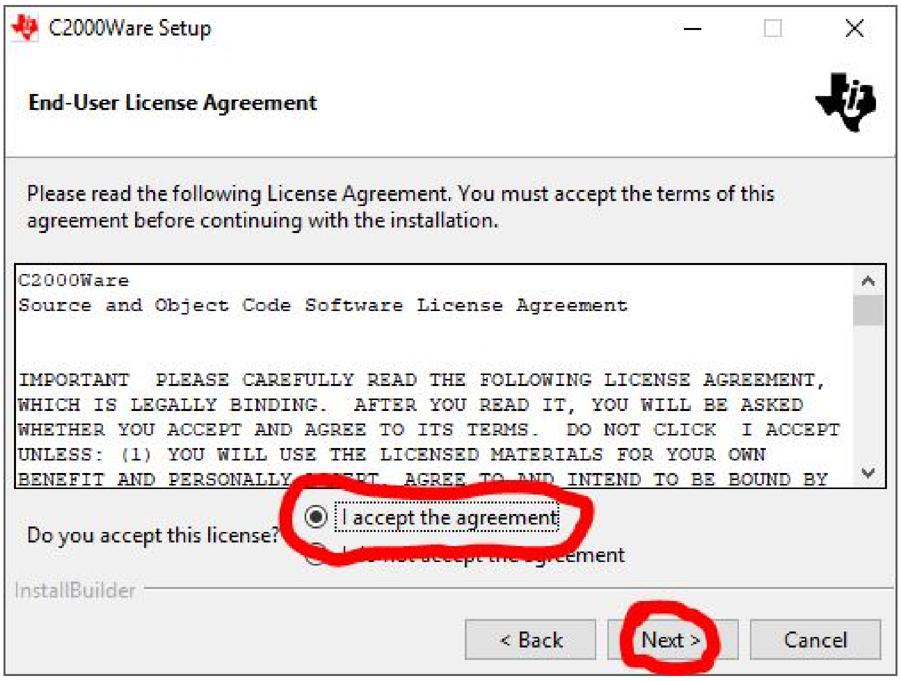


Welcome to the C2000Ware 1.00.03.00 Setup Wizard

The Setup Wizard will install C2000Ware 1.00.03.00 on your computer. Click "Next" to continue or "Cancel" to exit the Setup Wizard.

Figure A.28: step 8

9. Accept the End-User License Agreement and click "Next".



C2000Ware Setup

End-User License Agreement

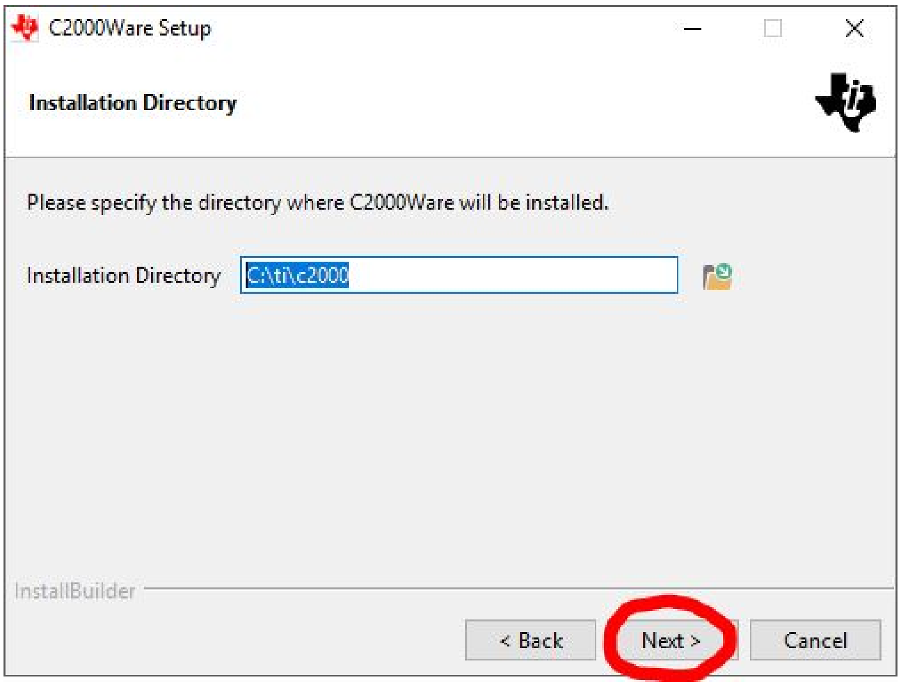
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

```
C2000Ware
Source and Object Code Software License Agreement


IMPORTANT  PLEASE CAREFULLY READ THE FOLLOWING LICENSE AGREEMENT,
WHICH IS LEGALLY BINDING.  AFTER YOU READ IT, YOU WILL BE ASKED
WHETHER YOU ACCEPT AND AGREE TO ITS TERMS.  DO NOT CLICK  I ACCEPT
UNLESS: (1) YOU WILL USE THE LICENSED MATERIALS FOR YOUR OWN
BENEFIT AND PERSONALLY ACCEPT, AGREE TO AND INTEND TO BE BOUND BY
```

Do you accept this license? [✓ I accept the agreement] I do not accept the agreement

Figure A.29: step 9

10. Select a directory to install to and click ”Next”. (c:\ti\c2000 is recommended). Make sure the directory is in the same folder as Code Composer Studio and MotorWare™.



C2000Ware Setup

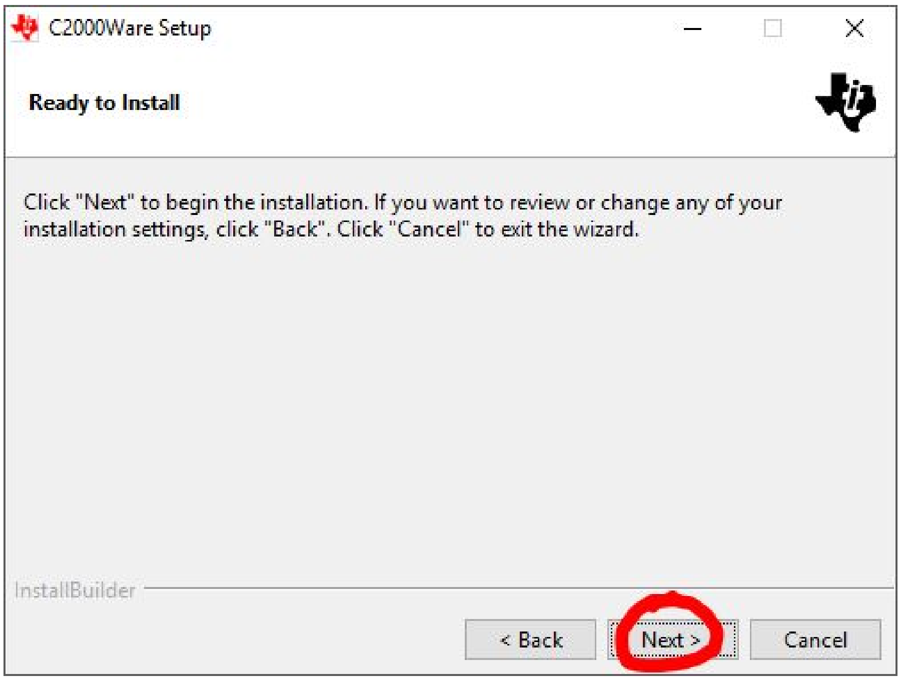
Installation Directory

Please specify the directory where C2000Ware will be installed.

Installation Directory C:\ti\c2000

Figure A.30: step 10

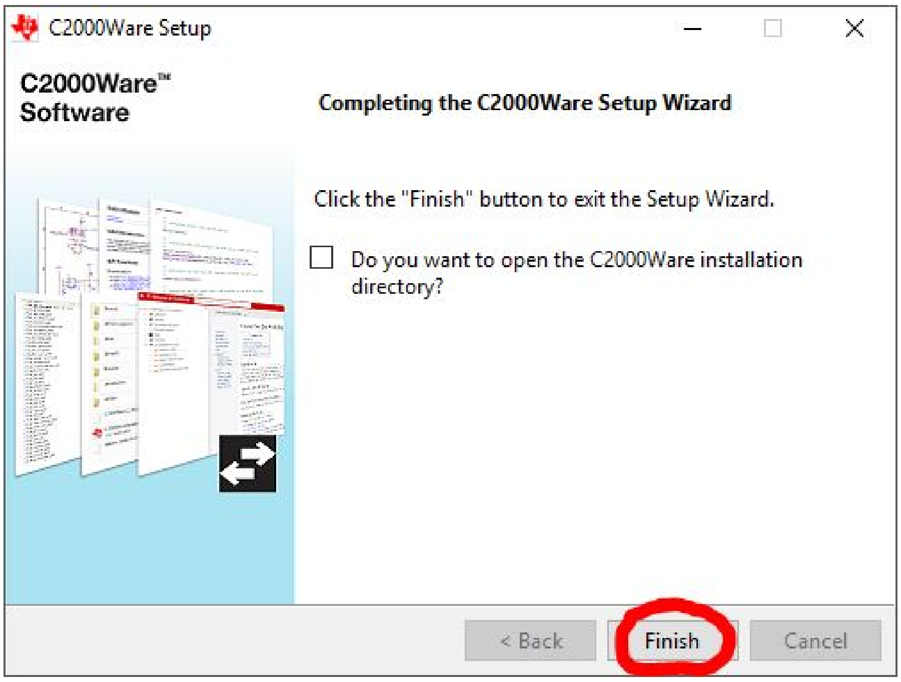
11. Click ”Next”.



Click "Next" to begin the installation. If you want to review or change any of your installation settings, click "Back". Click "Cancel" to exit the wizard.

Figure A.31: step 11

12. Wait for the installer to install C2000Ware.
13. If you don't want to view the C2000Ware directory, deselect the box. Click "Finish".



Click the "Finish" button to exit the Setup Wizard.

☐ Do you want to open the C2000Ware installation directory?

Figure A.32: step 13

A.4. Hardware configuration

1. Before connecting the LAUNCHXL-F28069M to anything, the settings of the board need to be changed. For this manual the board should be configured as follows [27]:

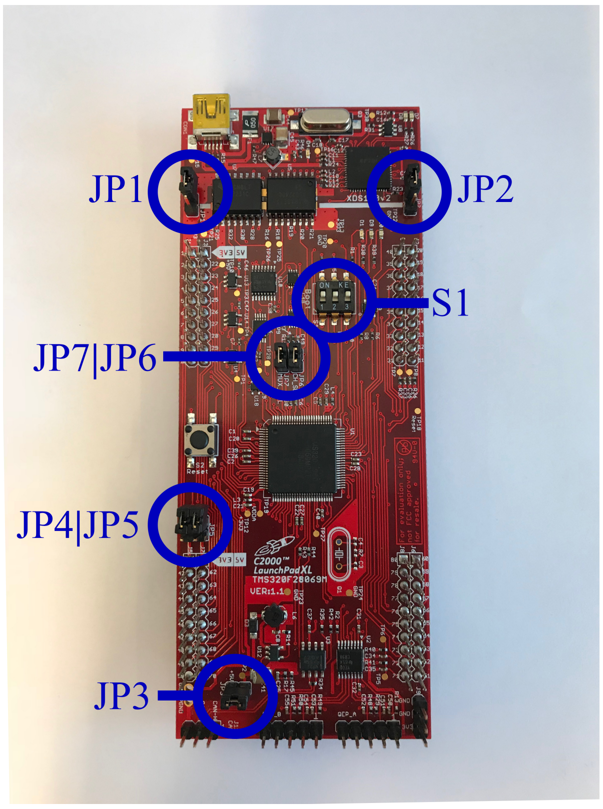


Figure A.33: The location of the jumpers and switches on the LAUNCHXL-F28069M

- Remove the jumpers from JP1 and JP2 to isolate the USB power from the rest of the board. The chip will be powered through the BoosterPack, but can still be programmed using USB.
- Set the switches on S1 to ON–ON–ON.
- Put the jumpers on JP3, JP6 and JP7.
- Jumpers 4 and 5:
  - Put the jumpers on JP4 and JP5, to use only the bottom BoosterPack headers J5-J8.
  - Remove the jumpers on JP4 and JP5, to use only the top BoosterPack headers J1-J4 or using both BoosterPacks J1-J4 and J5-J8.

  The BoosterPack will be connected to J1-J4 so remove the jumpers from JP4 and JP5.

2. Connect the BOOSTXL-DRV8301 to the LAUNCHXL-F28069M, with the connections on the same side as the USB-port.

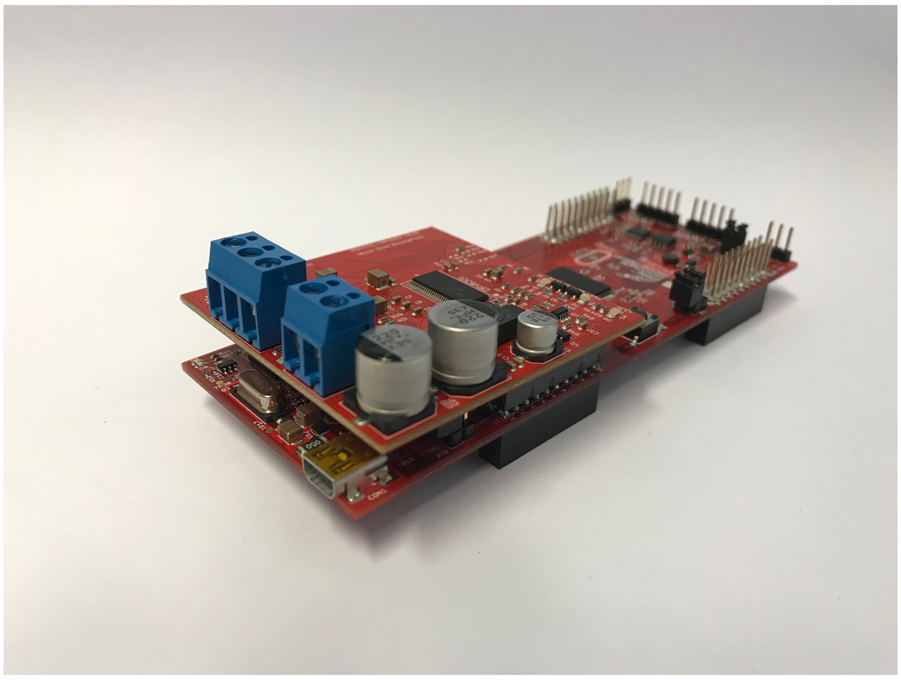


Figure A.34: The BOOSTXL-DRV8301 connected to the LAUNCHXL-F28069M

3. Connect the motor cable to the cable with smaller connections

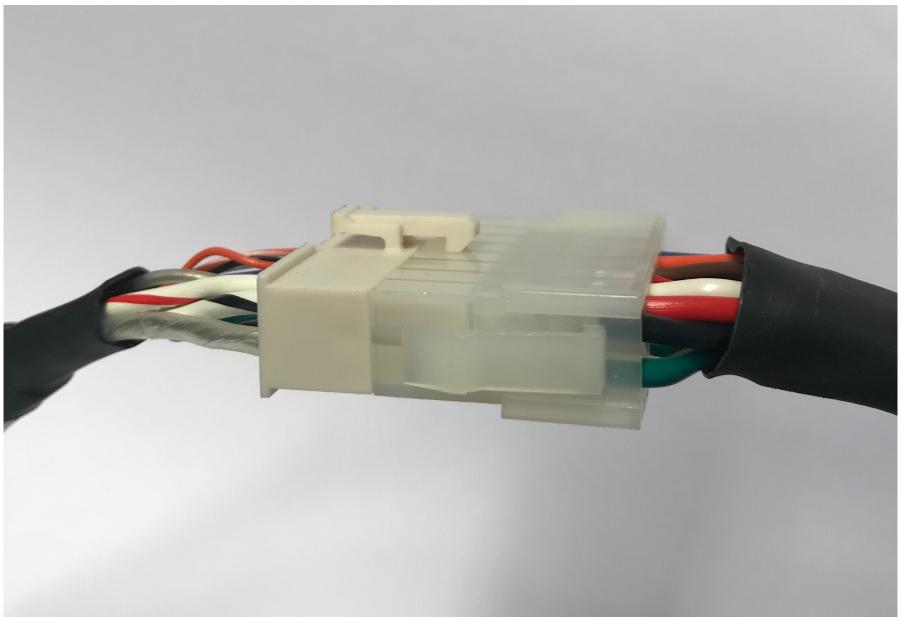


Figure A.35: The cable connection, the small hook clicks in place.

4. Connect the motor phase cables to the BOOSTXL-DRV8301. The green cable does not need to be connected, the other three can be connected in any way.

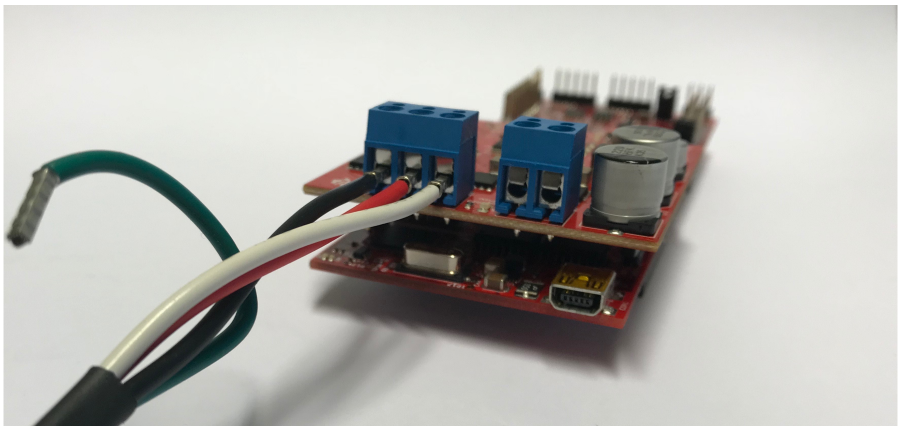


Figure A.36: The phase cables connected to the BOOSTXL-8301. The green cable should not be connected.

5. Connect the DC power cables to the BOOSTXL-8301.

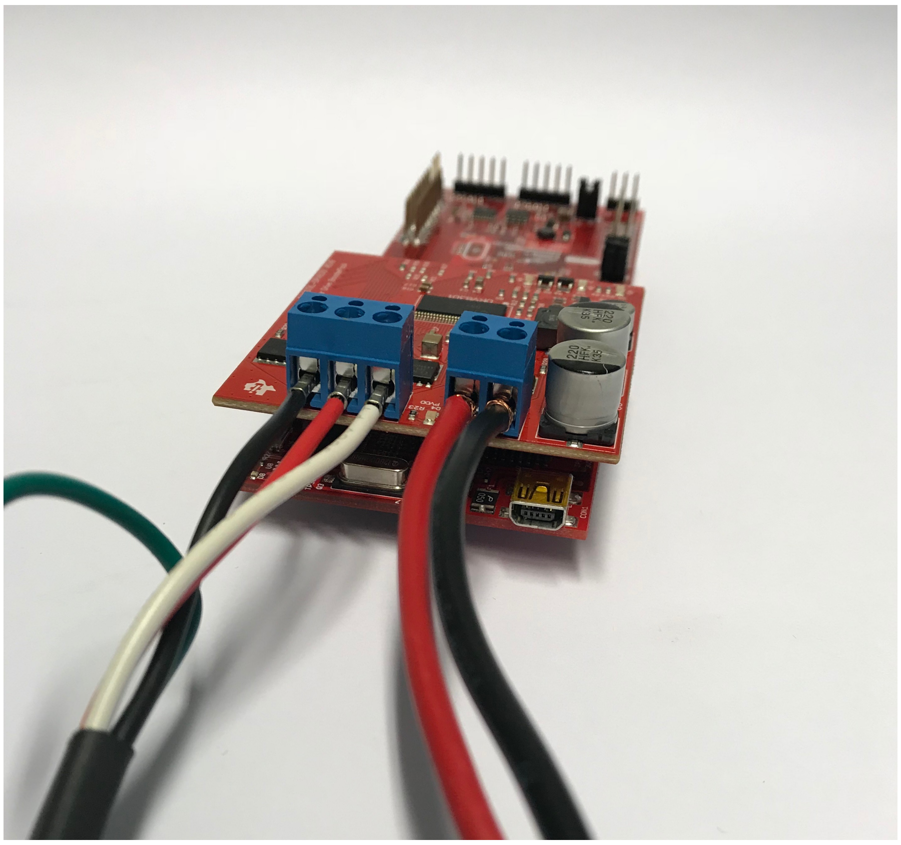


Figure A.37: The DC power cables connected to the BOOSTXL-8301.

6. Connect the DC cables to the power supply, and connect the LAUNCHXL-F28069M to your PC with the USB cable.

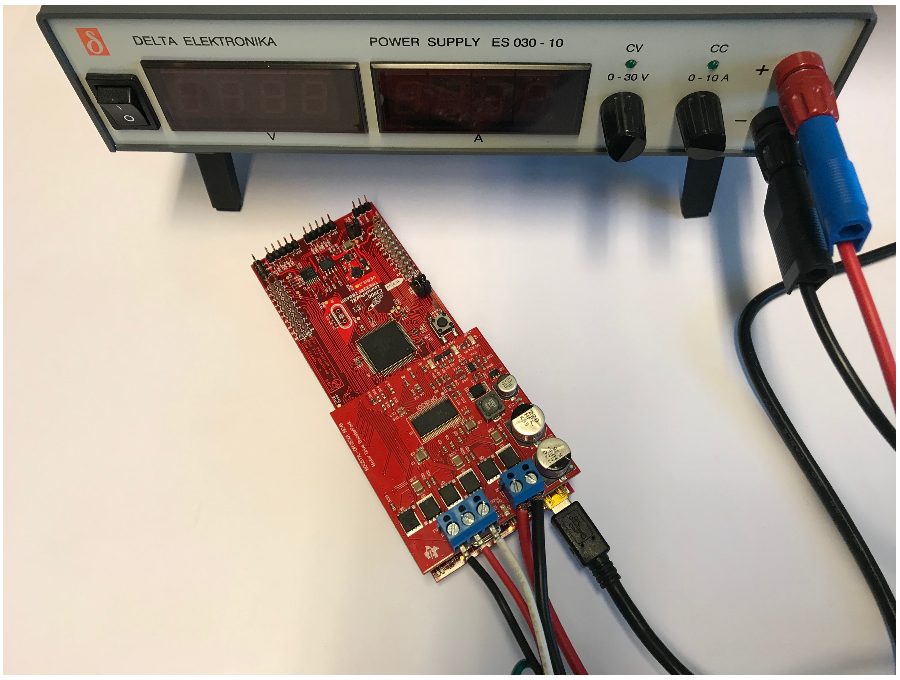


Figure A.38: The LaunchPad/BoosterPack combo with everything connected.

If you want to work with one motor first to get familiar with the hardware and software, continue with step 7 and then appendix A.5. If you wish to get started on the simulation set-up with multiple machines right away, go to step 8.

7. Fasten the Teknic Hudson M-2310 motor to the table using a clamp. Then continue with appendix A.5 and follow the InstaSPIN manual from there.

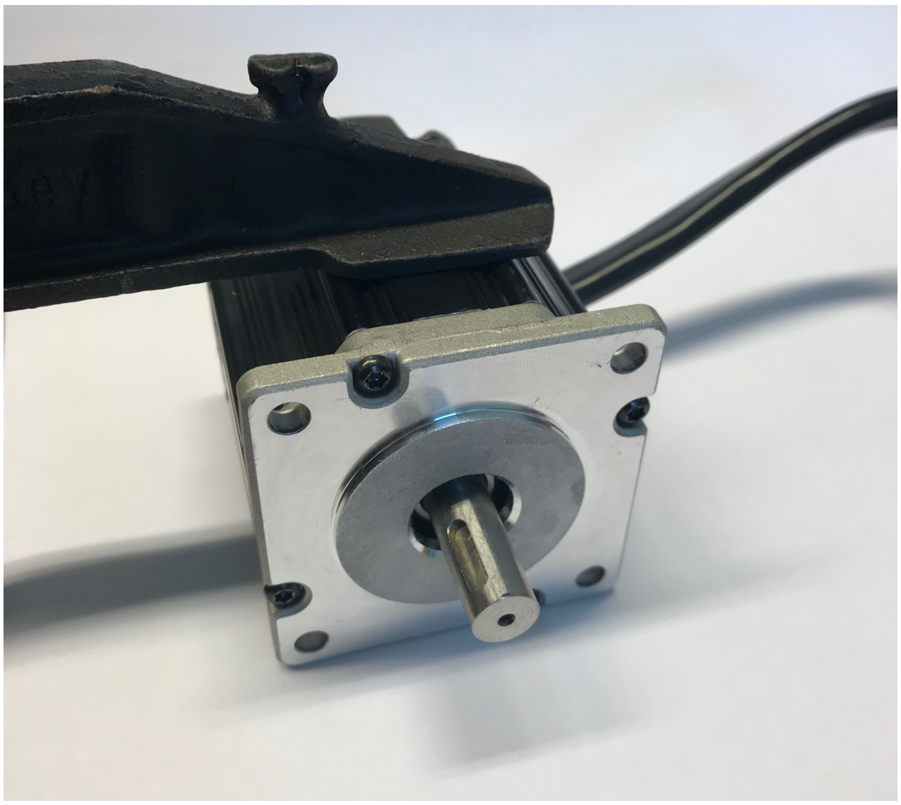


Figure A.39: Make sure the motor is clamped to the table firmly.

8. Repeat step 1-6, connecting the BoosterPack to the same power supply as the first one.

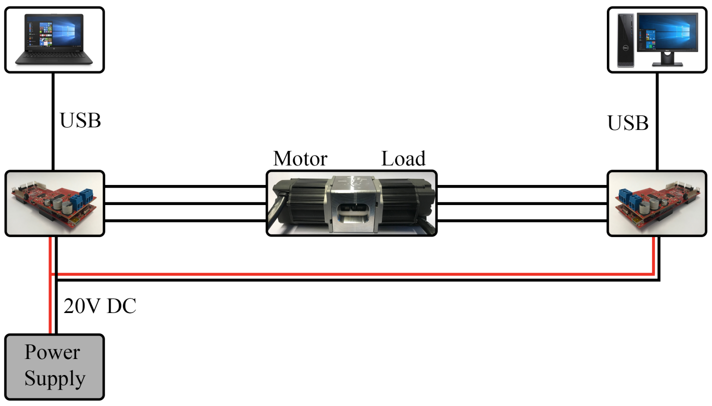


Figure A.40: A schematic representation of the simulation set-up which shows all the connections.

9. Find the Ruland Oldham coupling from the 2MTR-DYNO kit.

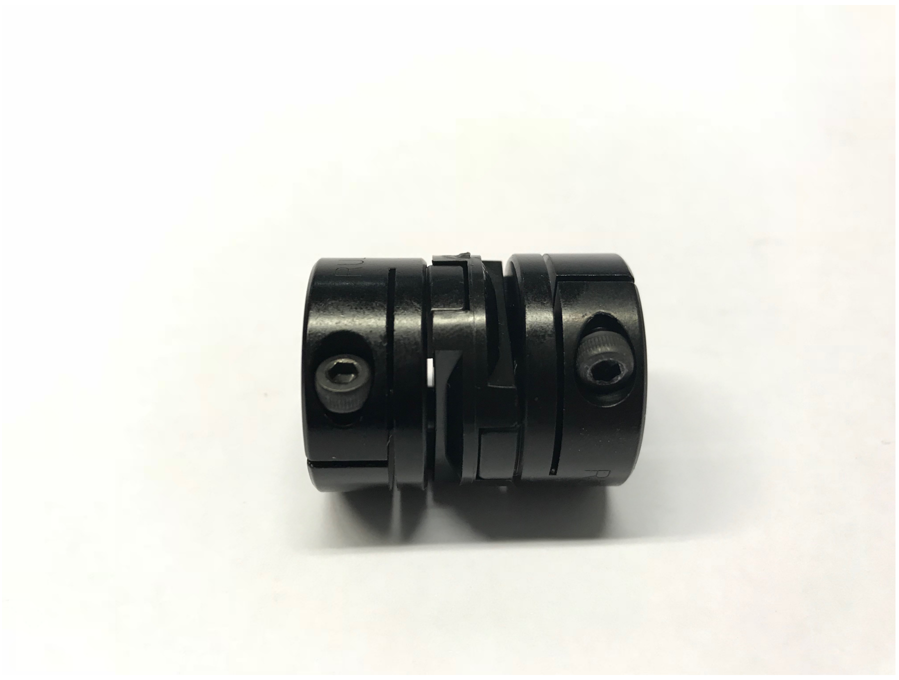


Figure A.41: The coupling used to couple the motors

10. Take it apart in two parts, one end, and one end with the middle part attached.

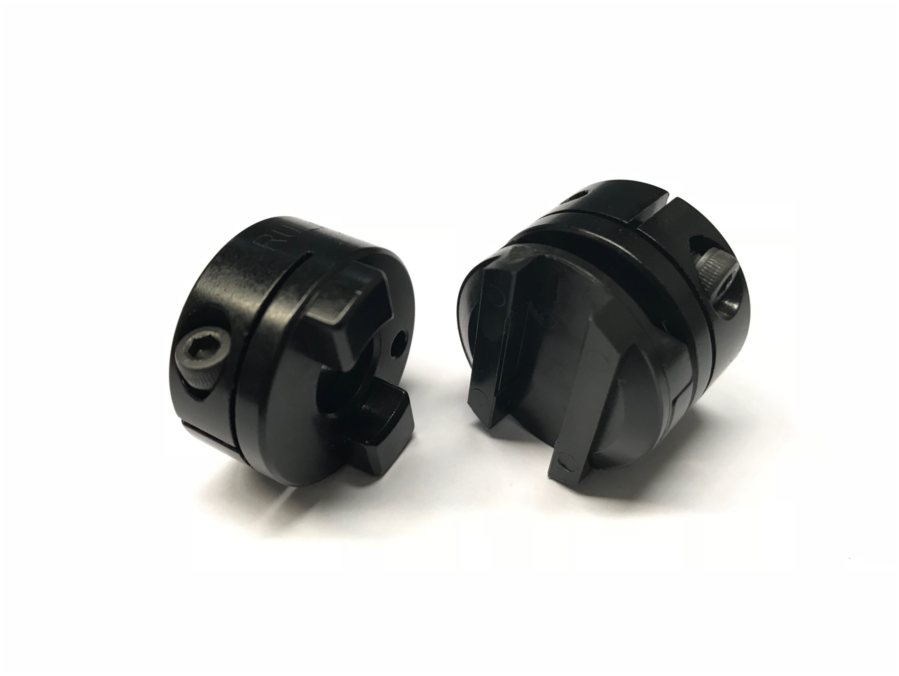


Figure A.42: The coupling in two pieces.

11. Make sure the motors are aligned 90 degrees from each other.

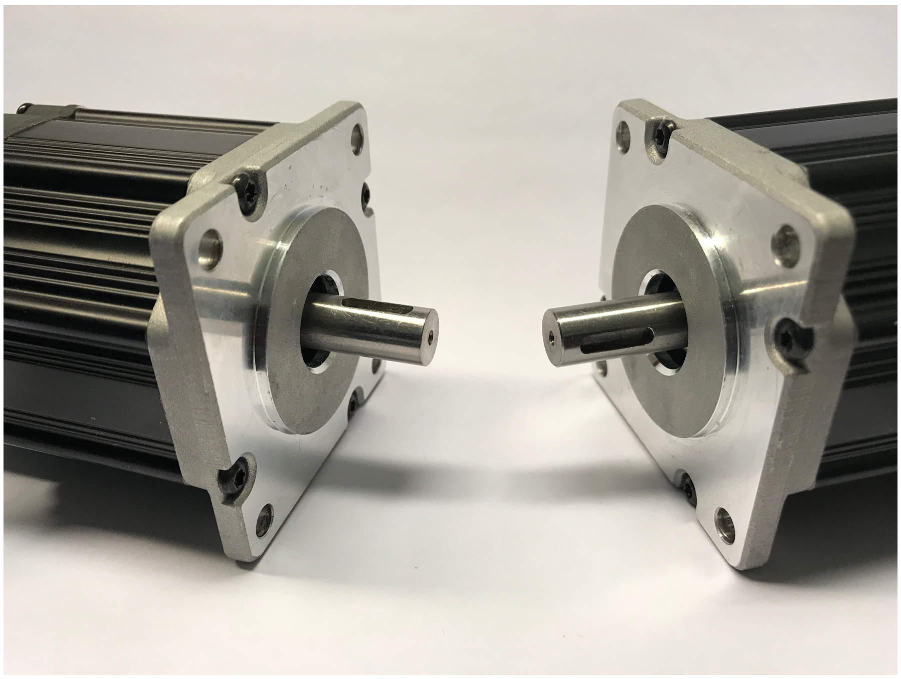


Figure A.43: The motor shafts should have a 90 degree angle.

12. Put the end of the coupling over the motor shafts. Do not shove them to far on the shaft, but rather leave them a bit over the edge.

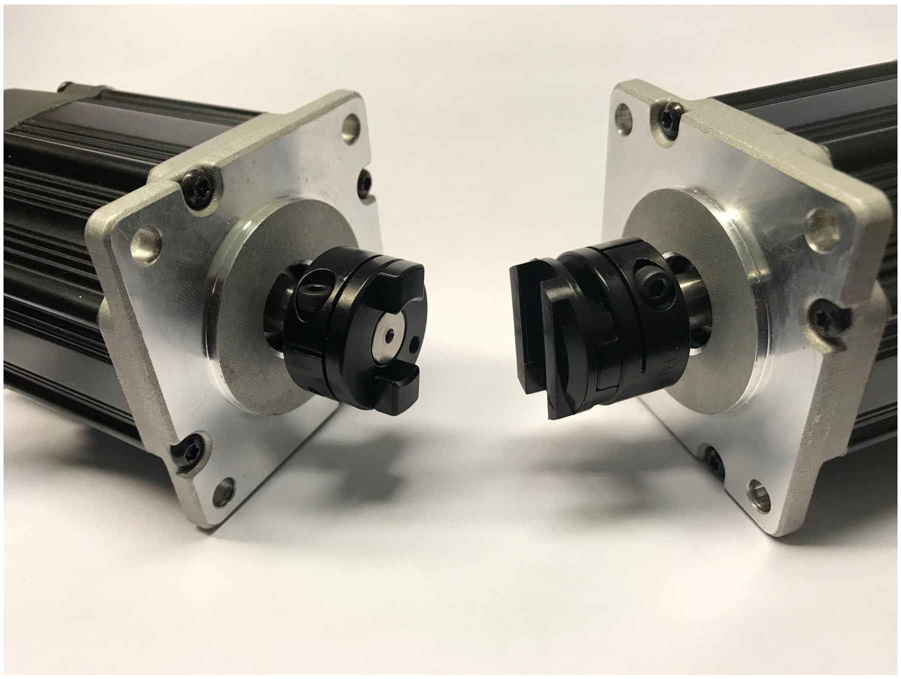


Figure A.44: The motors with the coupling parts over the shaft.

13. Place the two motors in the frame mount, make sure the coupling connects.

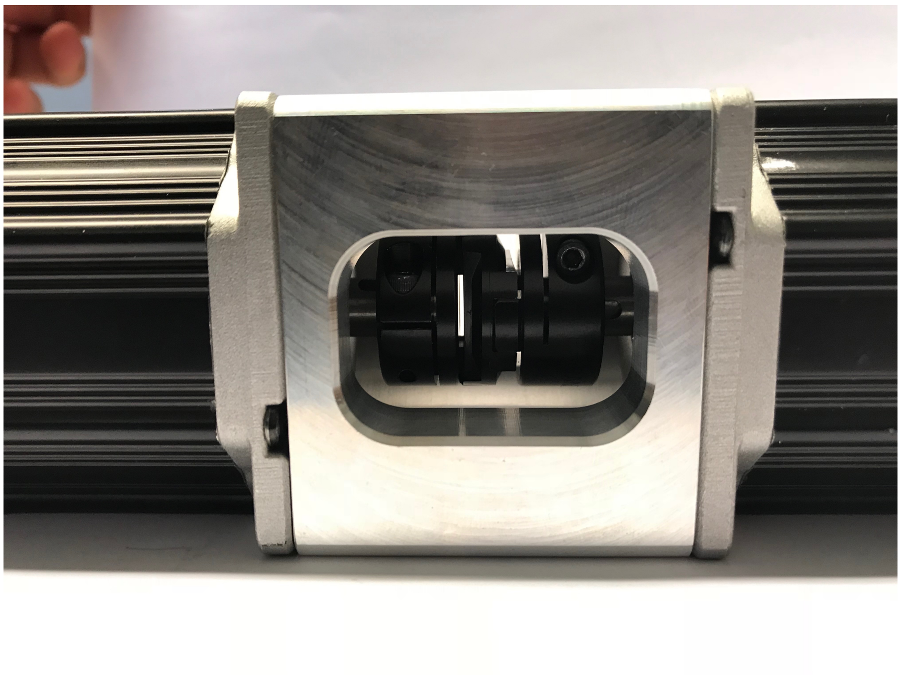


Figure A.45: The coupling is inside the frame mount.

14. Screw the motors to the frame mount with the eight screws.

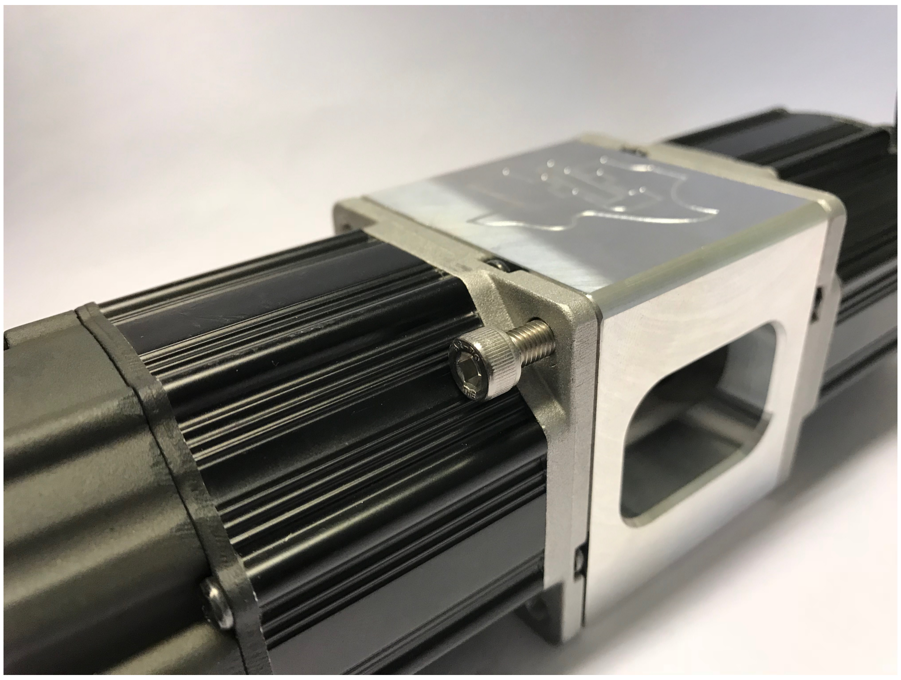


Figure A.46: Fasten the motors to the frame mount.

15. Make sure the coupling is pressed together well, and fasten the screws of the coupling through the hole in the frame mount.

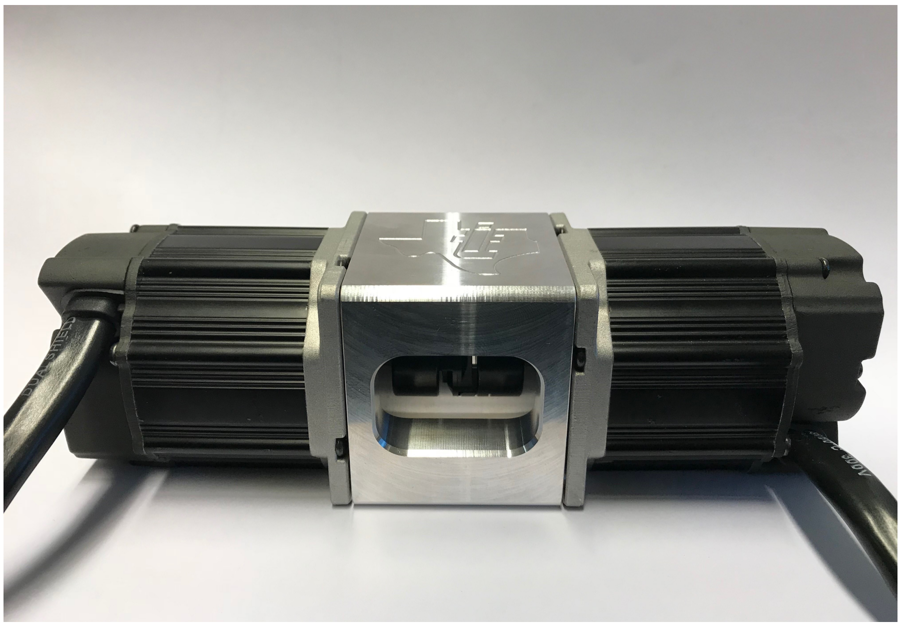


Figure A.47: The motors connected.

A.5. Getting started

A.5.1. Checking the drivers

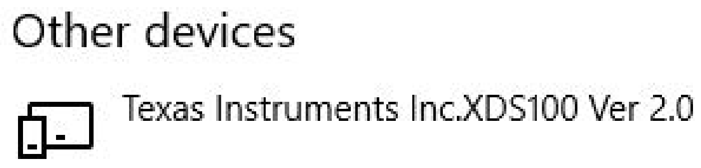
After all the software is installed and the hardware is set up, we can take a look at actually programming the LaunchPad. To make sure it works, check the drivers:

1. Make sure everything is connected and the DC power supply is on.

2. Click the settings icon in the Start menu.

3. Go to Devices.

4. If the drivers are correctly installed, a device will be shown under "Other Devices"



Other devices

Texas Instruments Inc.XDS100 Ver 2.0

Figure A.48: If this device shows up the driver has been installed correctly.

5. If this device doesn't show up, something has probably gone wrong during the installation of Code Composer Studio. Try re-installing (appendix A.3.1).

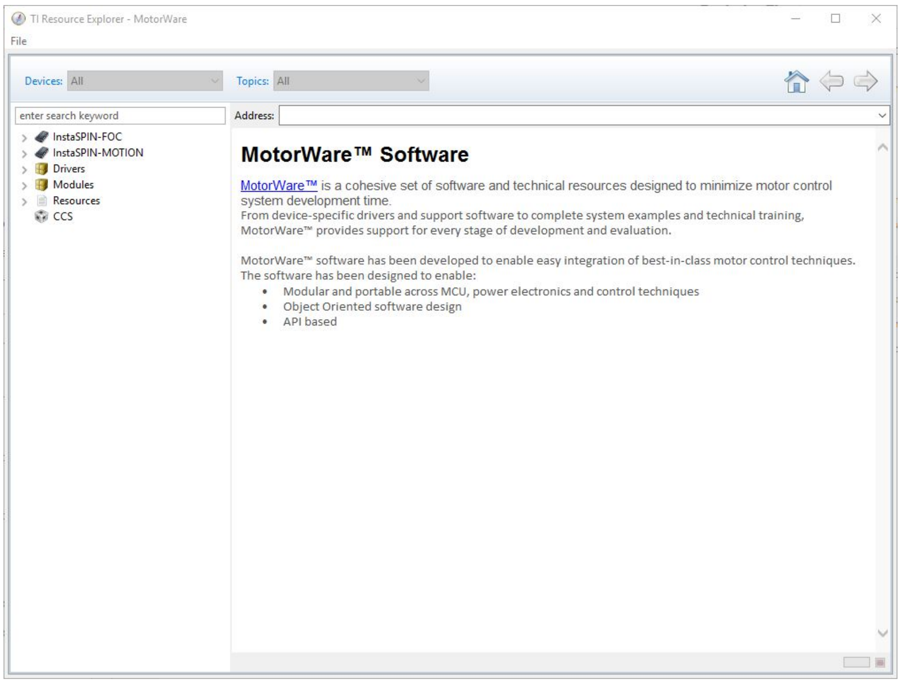
A.5.2. Starting with MotorWare™

This section will show where to find the lab manual in MotorWare™, as this can be quite difficult when there is no prior experience with it.

1. Run Code Composer Studio (In the Start menu it's in the "motorware" folder).

2. A security wrning will pop up. Click "Yes".

3. The MotorWare™ home screen will show.



MotorWare™ Software

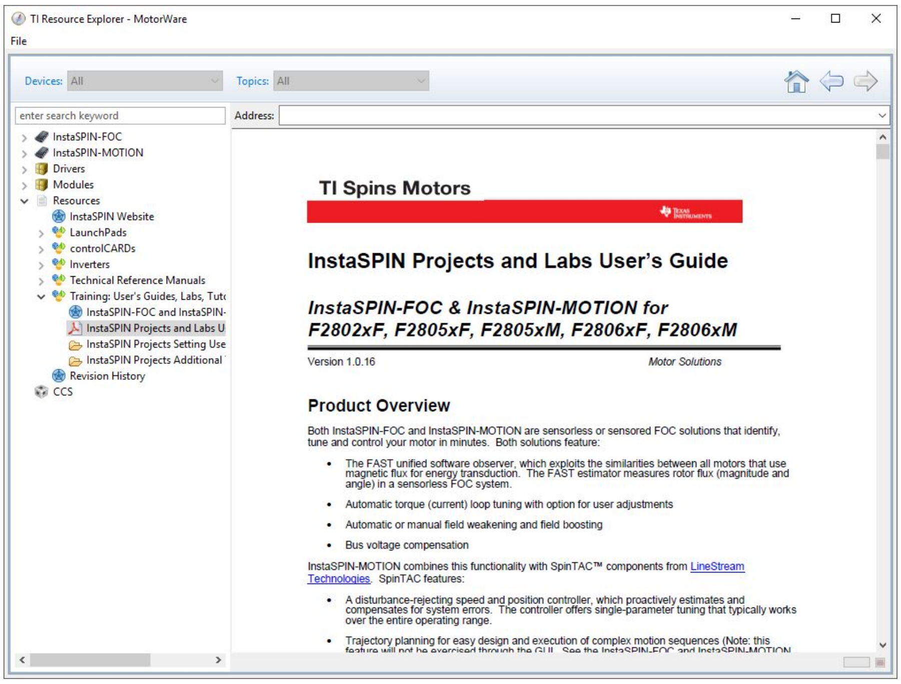
MotorWare™ is a cohesive set of software and technical resources designed to minimize motor control system development time.
From device-specific drivers and support software to complete system examples and technical training, MotorWare™ provides support for every stage of development and evaluation.

MotorWare™ software has been developed to enable easy integration of best-in-class motor control techniques.
The software has been designed to enable:

- Modular and portable across MCU, power electronics and control techniques
- Object Oriented software design
- API based

Figure A.49: The MotorWare™ home screen

4. The manual explaining the InstaSPIN™ labs can be found under "Resources" -> "Training: User's Guide, Labs, Tutorials" -> "InstaSPIN Projects and Labs User's Guide".


TI Resource Explorer - MotorWare
File
Devices: All
Topics: All
enter search keyword
Address:
InstaSPIN-FOC
InstaSPIN-MOTION
Drivers
Modules
Resources
InstaSPIN Website
LaunchPads
controlCARDs
Inverters
Technical Reference Manuals
Training: User's Guides, Labs, Tuto
InstaSPIN-FOC and InstaSPIN-
InstaSPIN Projects and Labs U
InstaSPIN Projects Setting Use
InstaSPIN Projects Additional
Revision History
CCS

TI Spins Motors

InstaSPIN Projects and Labs User's Guide

InstaSPIN-FOC & InstaSPIN-MOTION for F2802xF, F2805xF, F2805xM, F2806xF, F2806xM

Version 1.0.16 *Motor Solutions*

Product Overview

Both InstaSPIN-FOC and InstaSPIN-MOTION are sensorless or sensored FOC solutions that identify, tune and control your motor in minutes. Both solutions feature:

- The FAST unified software observer, which exploits the similarities between all motors that use magnetic flux for energy transduction. The FAST estimator measures rotor flux (magnitude and angle) in a sensorless FOC system.
- Automatic torque (current) loop tuning with option for user adjustments
- Automatic or manual field weakening and field boosting
- Bus voltage compensation

InstaSPIN-MOTION combines this functionality with SpinTAC™ components from LineStream Technologies. SpinTAC features:

- A disturbance-rejecting speed and position controller, which proactively estimates and compensates for system errors. The controller offers single-parameter tuning that typically works over the entire operating range.
- Trajectory planning for easy design and execution of complex motion sequences (Note: this

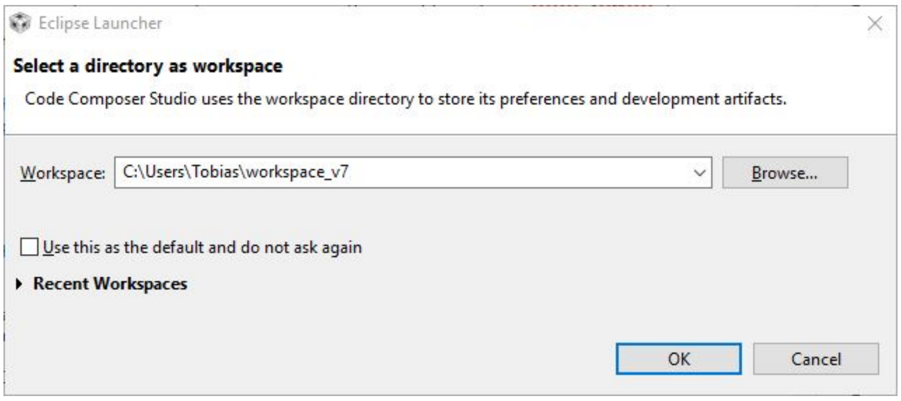
Figure A.50: Here the lab manual for InstaSPIN™ can be found.

A.5.3. Starting with Code Composer Studio

Now we can start using Code Composer Studio to program our MCU and control the motor.

1. Run Code Composer Studio (In the Start menu it's in the "Texas Instruments" folder).

2. A window will appear asking for the workspace directory.


Eclipse Launcher

Select a directory as workspace

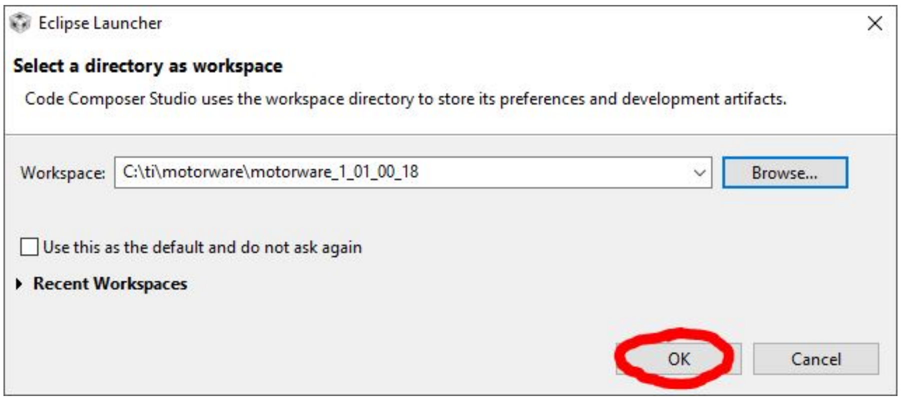
Code Composer Studio uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\Tobias\workspace_v7

☐ Use this as the default and do not ask again

Figure A.51: This window shows after you run Code Composer Studio.

3. Change the directory to the main MotorWare™ direction (if you followed the instructions from appendix A.3 the direction will be C:\ti\motorware\motorware_[version]). Then click ”OK”.


Eclipse Launcher

Select a directory as workspace

Code Composer Studio uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\ti\motorware\motorware_1_01_00_18

☐ Use this as the default and do not ask again

Figure A.52: Change the directory and click ”OK”.

4. The home screen of Code Composer Studio will show. Here there are several quick start options, such as introduction videos and a link to the wiki.

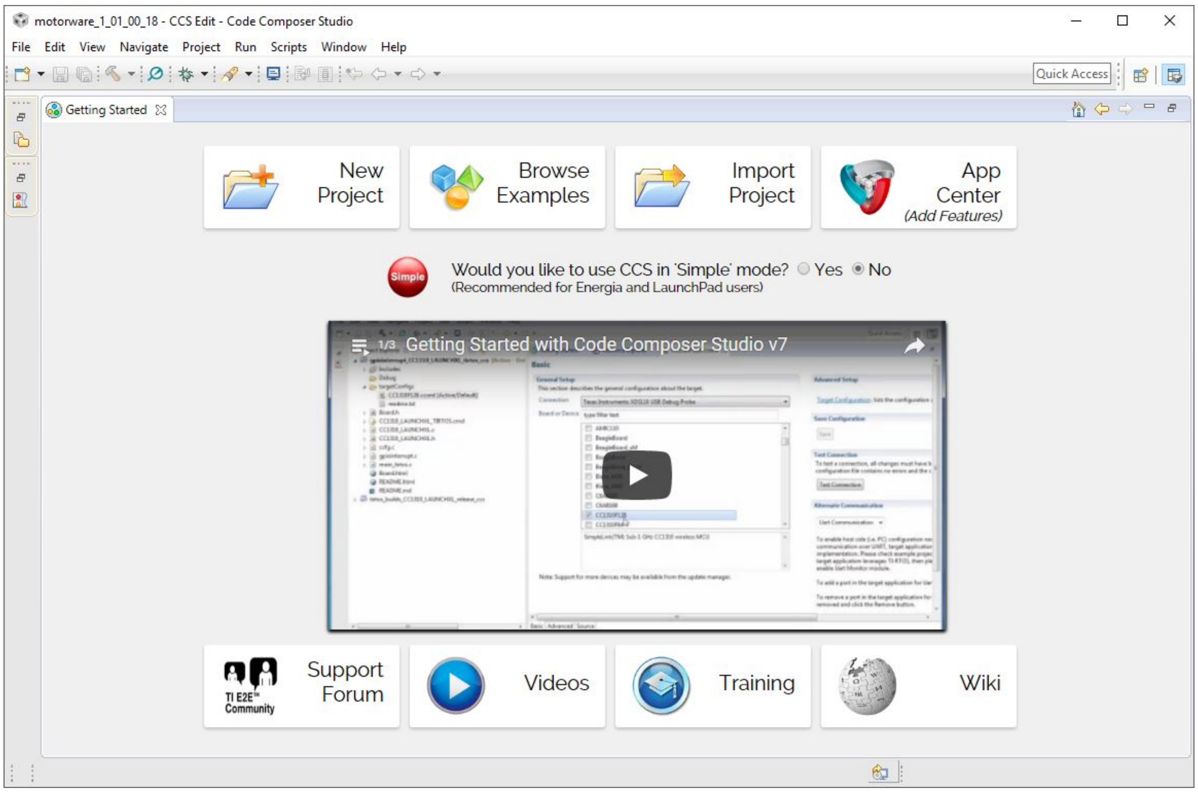

motorware_1_01_00_18 - CCS Edit - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
Getting Started
New Project
Browse Examples
Import Project
App Center (Add Features)
Would you like to use CCS in 'Simple' mode? Yes [✓ No]
(Recommended for Energia and LaunchPad users)
Getting Started with Code Composer Studio v7
Support Forum
Videos
Training
Wiki

Figure A.53: The screen Code Composer Studio shows when you first run it.

5. To get started on the InstaSPIN labs, click on "Import CCS projects..." in the Project menu.

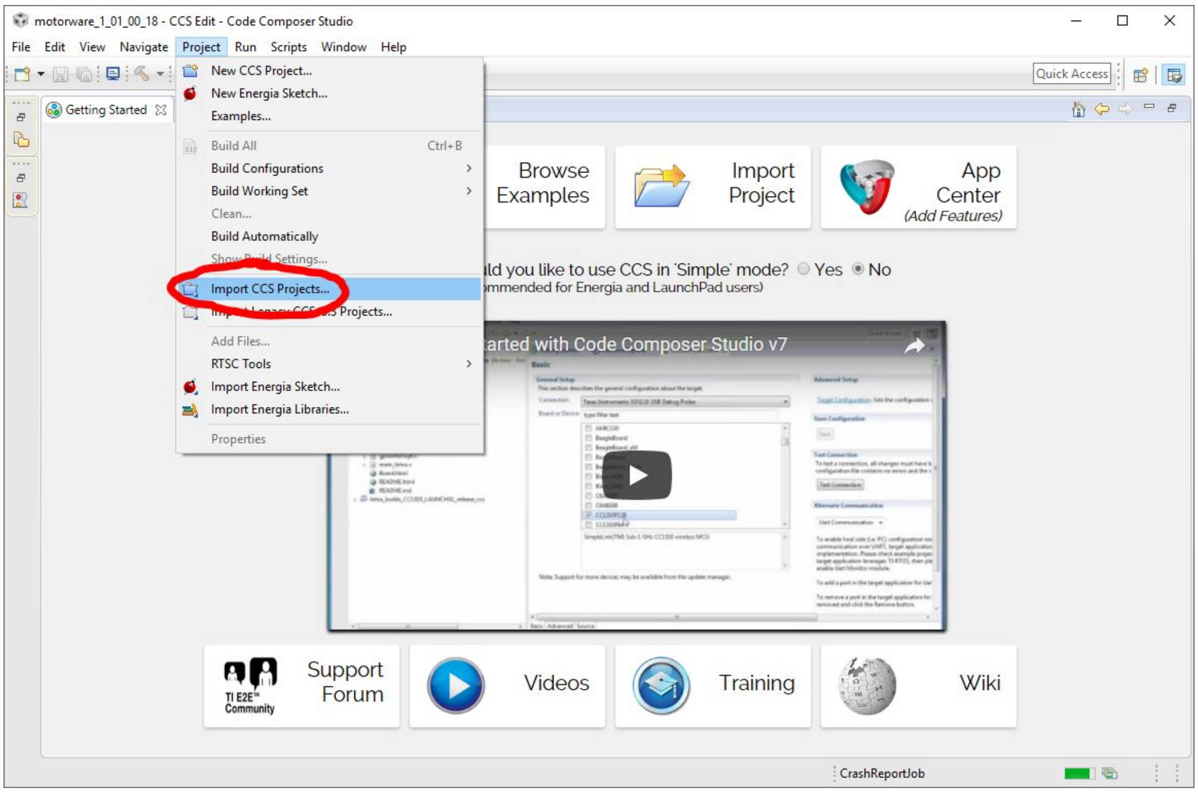


Figure A.54: Click Import CCS projects... in the Project menu.

6. In the window that pops up, click browse.

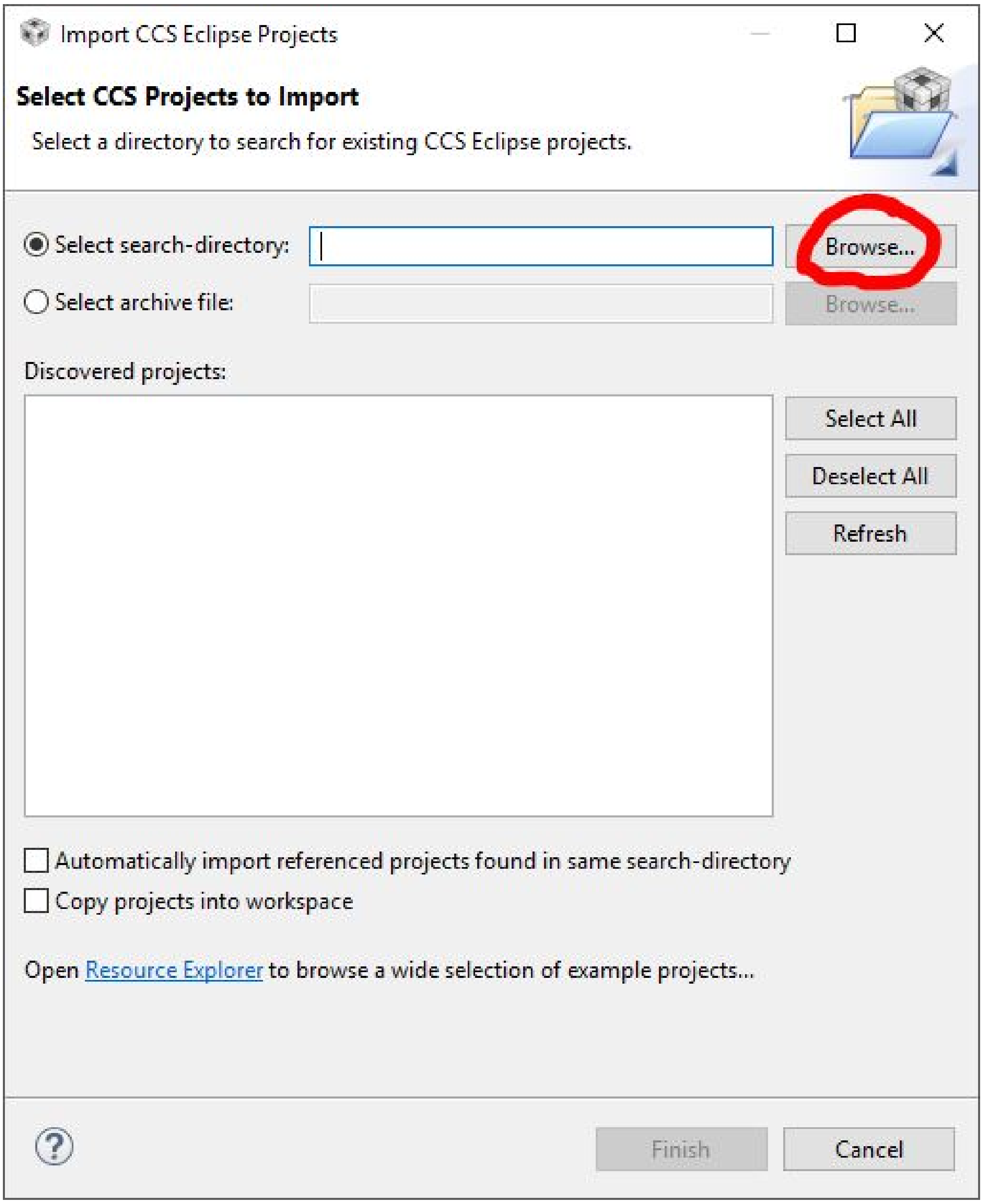
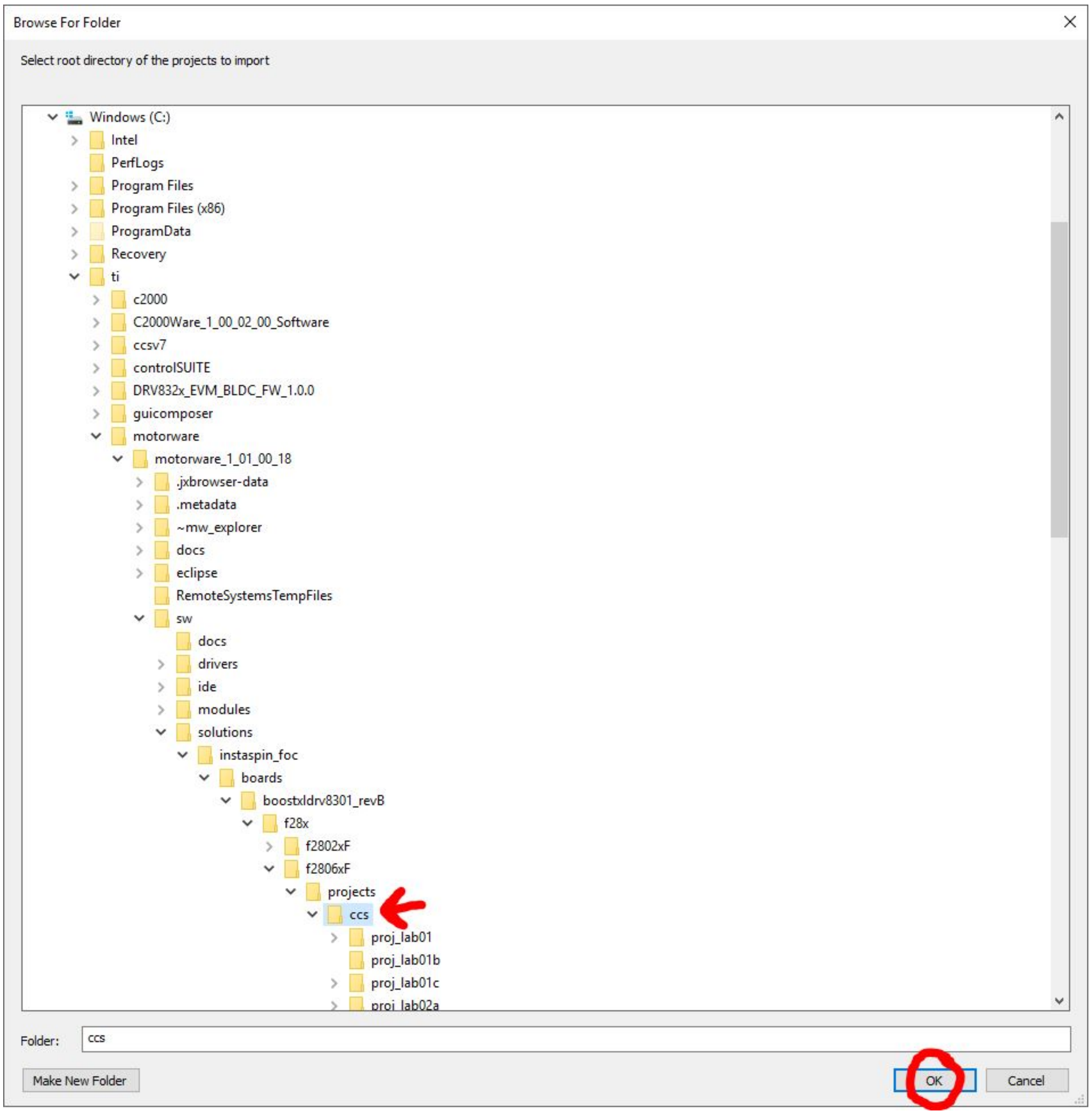

Import CCS Eclipse Projects
Select CCS Projects to Import
Select a directory to search for existing CCS Eclipse projects.
Select search-directory:
Browse...
Select archive file:
Browse...
Discovered projects:
Select All
Deselect All
Refresh
Automatically import referenced projects found in same search-directory
Copy projects into workspace
Open Resource Explorer to browse a wide selection of example projects...
Finish
Cancel

Figure A.55: Browse for a search-directory

7. The location for the labs is C:\ti\motorware\motorware_[version]\sw\solution\instaspin_foc\boards \boostxldrv8301_revB\f28x\f2806xF\project\ccs. Select it and click ”OK”.


Browse For Folder

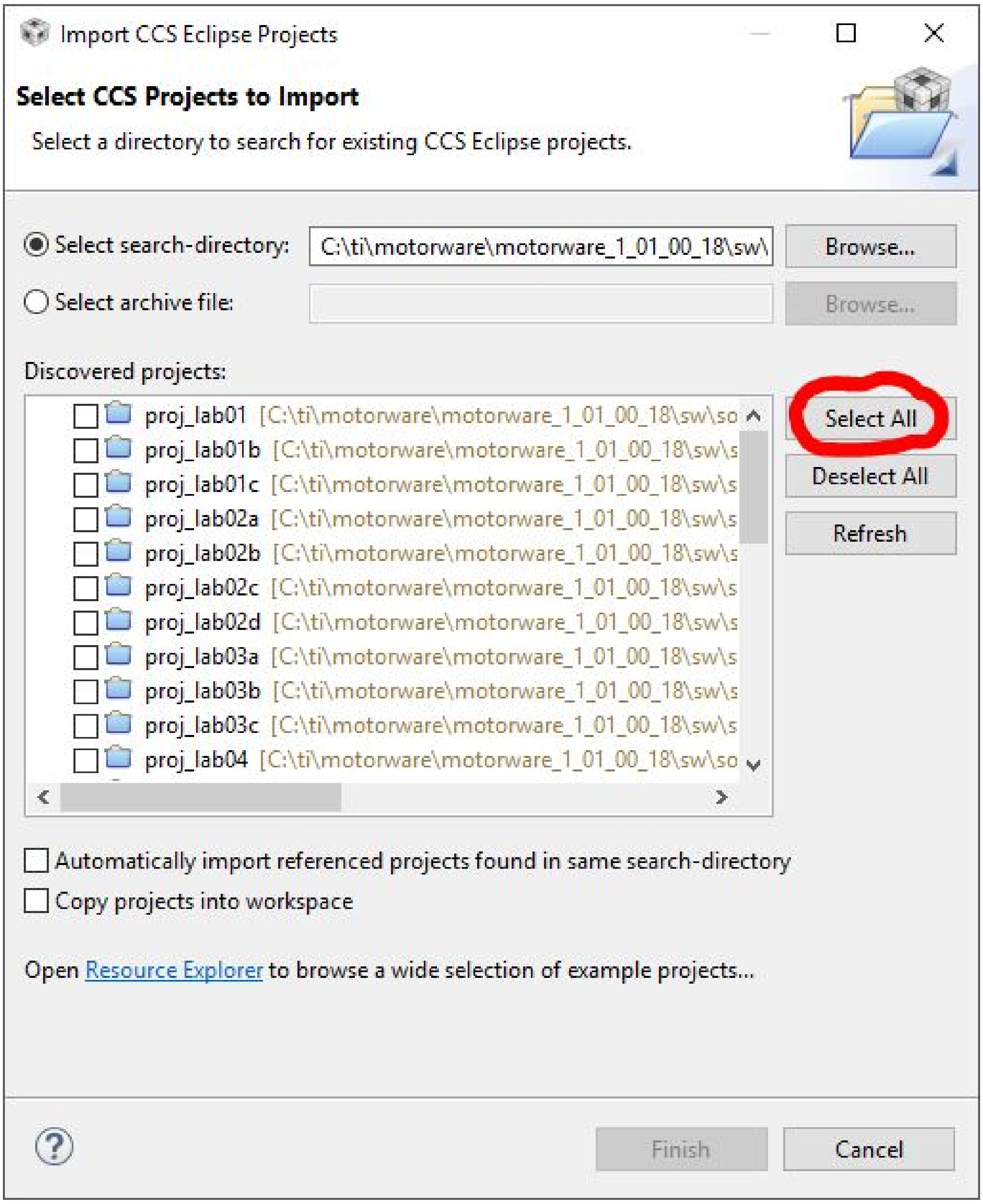
Select root directory of the projects to import

- Windows (C:)
  - Intel
  - PerfLogs
  - Program Files
  - Program Files (x86)
  - ProgramData
  - Recovery
  - ti
    - c2000
    - C2000Ware_1_00_02_00_Software
    - ccsv7
    - controlSUITE
    - DRV832x_EVM_BLDC_FW_1.0.0
    - guicomposer
    - motorware
      - motorware_1_01_00_18
        - .jxbrowser-data
        - .metadata
        - ~mw_explorer
        - docs
        - eclipse
        - RemoteSystemsTempFiles
        - sw
          - docs
          - drivers
          - ide
          - modules
          - solutions
            - instaspin_foc
              - boards
                - boostxldrv8301_revB
                  - f28x
                    - f2802xF
                    - f2806xF
                      - projects
                        - ccs
                          - proj_lab01
                          - proj_lab01b
                          - proj_lab01c
                          - proj_lab02a

Folder: ccs

Figure A.56: Find the ccs folder, select it and click ”OK”

8. The projects will show up in the previous window. Click ”Select All”.


Import CCS Eclipse Projects

Select CCS Projects to Import

Select a directory to search for existing CCS Eclipse projects.

(●) Select search-directory: C:\ti\motorware\motorware_1_01_00_18\sw\ [Browse...]

( ) Select archive file: [Browse...]

Discovered projects:

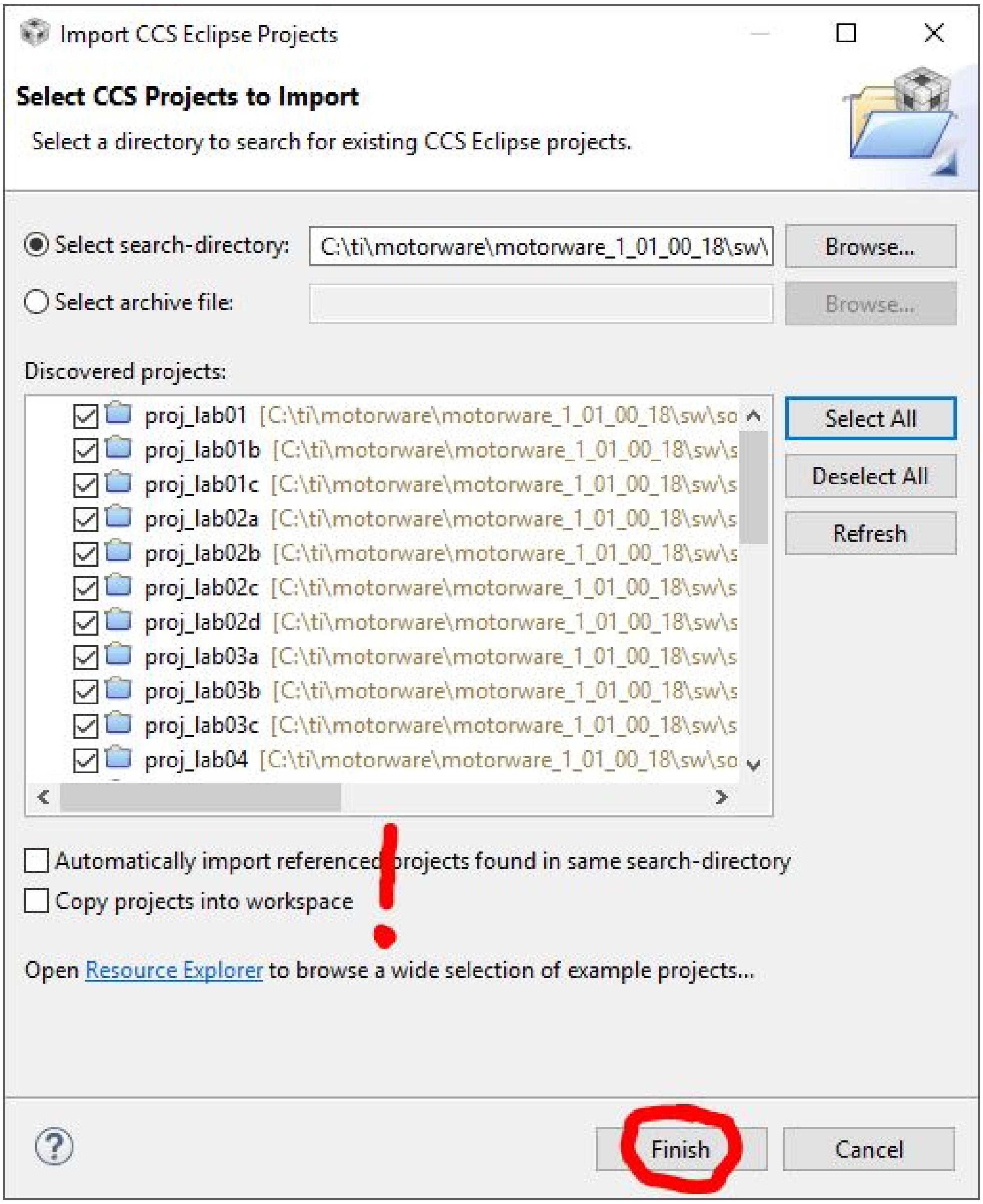
- [ ] proj_lab01 [C:\ti\motorware\motorware_1_01_00_18\sw\so
- [ ] proj_lab01b [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [ ] proj_lab01c [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [ ] proj_lab02a [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [ ] proj_lab02b [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [ ] proj_lab02c [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [ ] proj_lab02d [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [ ] proj_lab03a [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [ ] proj_lab03b [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [ ] proj_lab03c [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [ ] proj_lab04 [C:\ti\motorware\motorware_1_01_00_18\sw\so

- [ ] Automatically import referenced projects found in same search-directory
- [ ] Copy projects into workspace

Open Resource Explorer to browse a wide selection of example projects...

Figure A.57: The projects are found. Click ”Select All”.

9. Click ”Finish” to load the projects into Code Composer Studio. Make sure ”Copy projects into workspace” is NOT selected.


Import CCS Eclipse Projects
Select CCS Projects to Import
Select a directory to search for existing CCS Eclipse projects.
Select search-directory:
Browse...
Select archive file:
Browse...
Discovered projects:
Select All
Deselect All
Refresh
Automatically import referenced projects found in same search-directory
Copy projects into workspace
Open Resource Explorer to browse a wide selection of example projects...
Finish
Cancel

Select search-directory: C:\ti\motorware\motorware_1_01_00_18\sw\

Discovered projects:

- [✓] proj_lab01 [C:\ti\motorware\motorware_1_01_00_18\sw\so
- [✓] proj_lab01b [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [✓] proj_lab01c [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [✓] proj_lab02a [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [✓] proj_lab02b [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [✓] proj_lab02c [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [✓] proj_lab02d [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [✓] proj_lab03a [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [✓] proj_lab03b [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [✓] proj_lab03c [C:\ti\motorware\motorware_1_01_00_18\sw\s
- [✓] proj_lab04 [C:\ti\motorware\motorware_1_01_00_18\sw\so

Figure A.58: Click ”Finish to load the projects into Code Composer Studio.

10. All the projects will be loaded, and the home window will look like this.

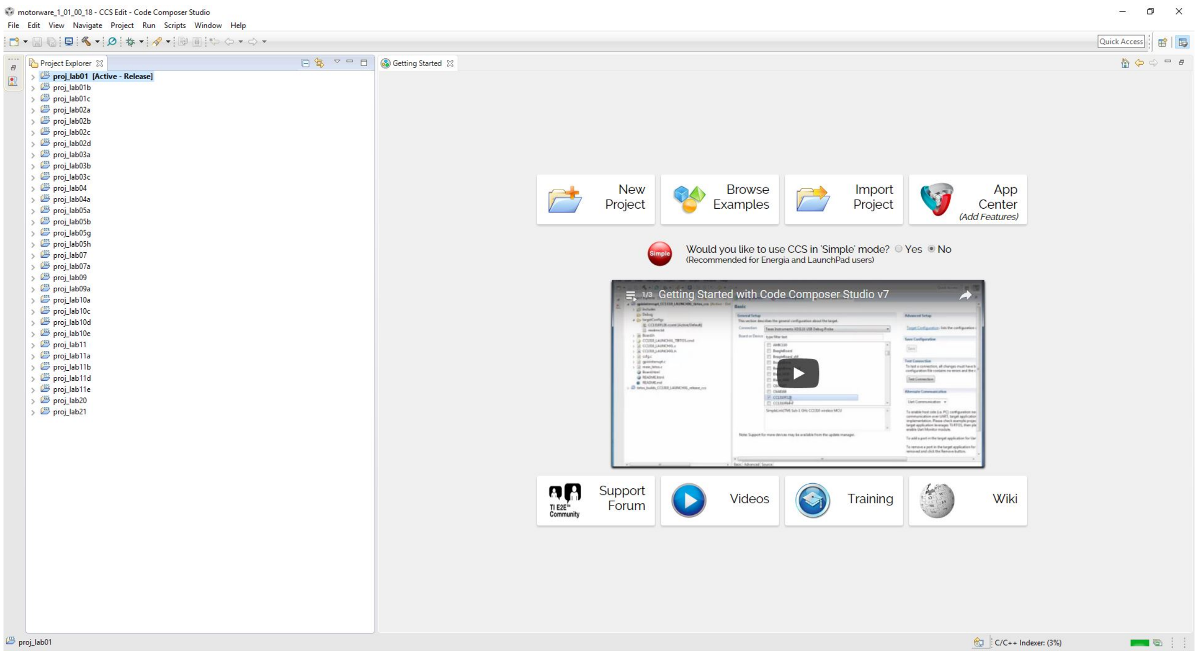


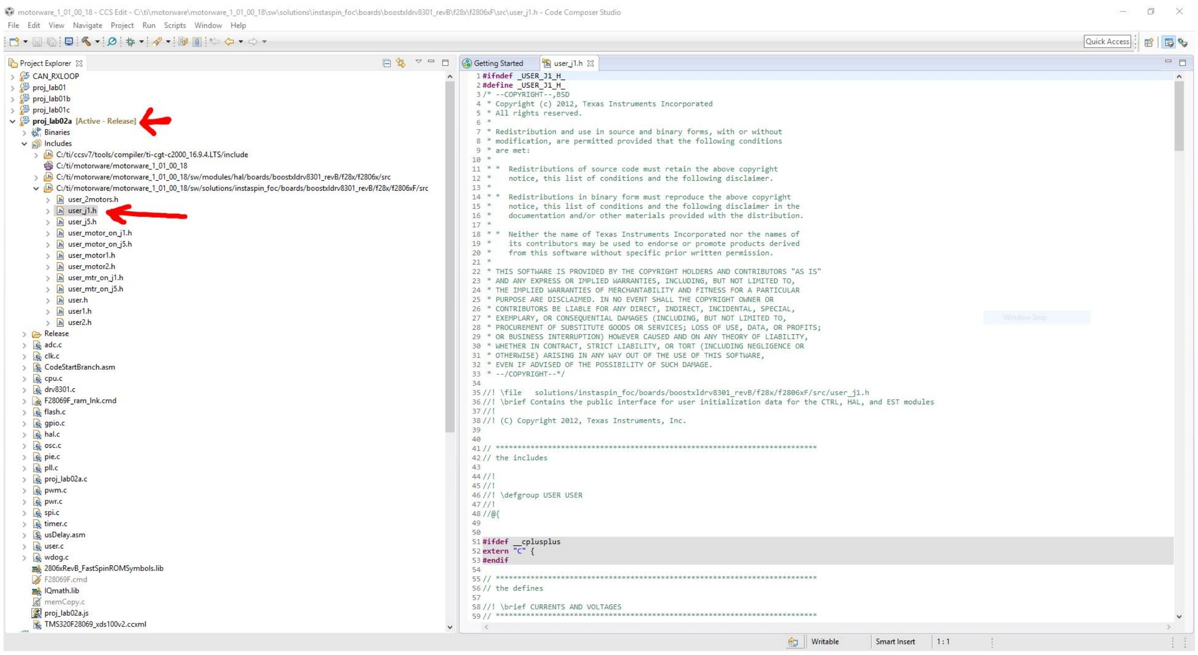
Figure A.59: The CCS home window with all the projects from the InstaSPIN labs loaded.

At this point the InstaSPIN manual found in step A.5.2 ([14]) can be followed to go through all the labs. If you are only interested in using the simulation set-up as quick as possible, keep following this manual. If anything is still unclear about any of the labs, check [14].

A.6. Using the InstaSPIN labs

A.6.1. Using lab 2 for estimating motor parameters

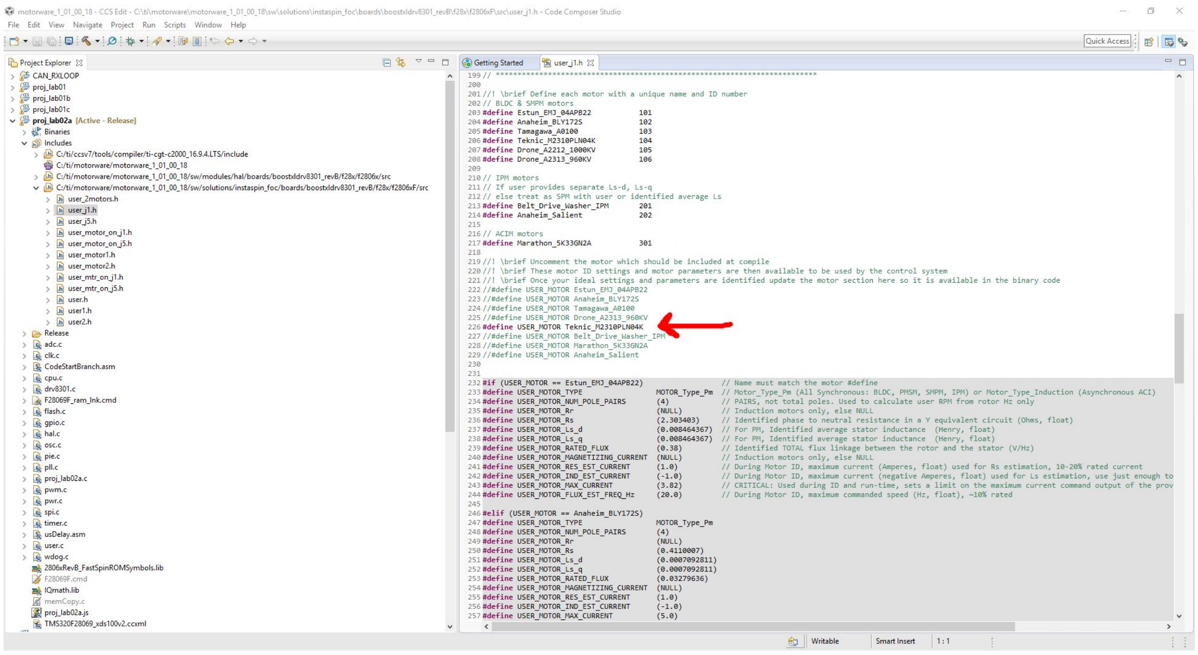
1. Doubleclick on `proj\_lab02a`, then find the `user\_j1.h` file, located in includes -> ”`C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\boards\boostxldrv8301_revB\f28x\f2806xF\src`” and double click on that. This will open the file in a windows to the right.



```
#ifndef _USER_J1_H_
#define _USER_J1_H_
/* --COPYRIGHT--,BSD
 * Copyright (c) 2012, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/

//! \file   solutions/instaspin_foc/boards/boostxldrv8301_revB/f28x/f2806xF/src/user_j1.h
//! \brief Contains the public interface for user initialization data for the CTRL, HAL, and EST modules
//!
//! (C) Copyright 2012, Texas Instruments, Inc.


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

//!
//!
//! \defgroup USER USER
//!
//@{


#ifdef __cplusplus
extern "C" {
#endif

// **************************************************************************
// the defines

//! \brief CURRENTS AND VOLTAGES
// **************************************************************************
```

Figure A.60: Double click on proj_lab2a and user_j1.h

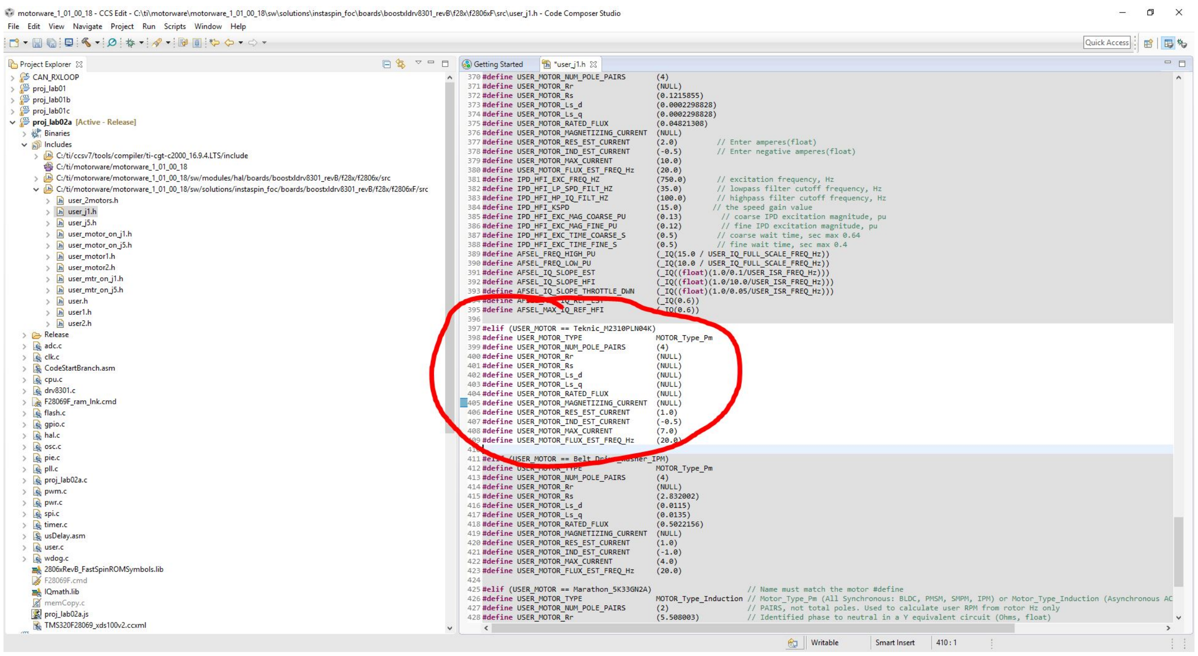
2. Find the motor selection section (around line 219) and comment everything except `#define USER_MOTOR Teknic_M2310PLN04K`



```
199 // **************************************************************************
200
201 //! \brief Define each motor with a unique name and ID number
202 // BLDC & SMPM motors
203 #define Estun_EMJ_04APB22          101
204 #define Anaheim_BLY172S            102
205 #define Tamagawa_A0100             103
206 #define Teknic_M2310PLN04K         104
207 #define Drone_A2212_1000KV         105
208 #define Drone_A2313_960KV          106
209
210 // IPM motors
211 // If user provides separate Ls-d, Ls-q
212 // else treat as SPM with user or identified average Ls
213 #define Belt_Drive_Washer_IPM      201
214 #define Anaheim_Salient            202
215
216 // ACIM motors
217 #define Marathon_5K33GN2A          301
218
219 //! \brief Uncomment the motor which should be included at compile
220 //! \brief These motor ID settings and motor parameters are then available to be used by the control system
221 //! \brief Once your ideal settings and parameters are identified update the motor section here so it is available in the binary code
222 //#define USER_MOTOR Estun_EMJ_04APB22
223 //#define USER_MOTOR Anaheim_BLY172S
224 //#define USER_MOTOR Tamagawa_A0100
225 //#define USER_MOTOR Drone_A2313_960KV
226 #define USER_MOTOR Teknic_M2310PLN04K
227 //#define USER_MOTOR Belt_Drive_Washer_IPM
228 //#define USER_MOTOR Marathon_5K33GN2A
229 //#define USER_MOTOR Anaheim_Salient
230
231
232 #if (USER_MOTOR == Estun_EMJ_04APB22)                  // Name must match the motor #define
233 #define USER_MOTOR_TYPE                 MOTOR_Type_Pm  // Motor_Type_Pm (All Synchronous: BLDC, PMSM, SMPM, IPM) or Motor_Type_Induction (Asynchronous ACI)
234 #define USER_MOTOR_NUM_POLE_PAIRS       (4)            // PAIRS, not total poles. Used to calculate user RPM from rotor Hz only
235 #define USER_MOTOR_Rr                   (NULL)         // Induction motors only, else NULL
236 #define USER_MOTOR_Rs                   (2.303403)     // Identified phase to neutral resistance in a Y equivalent circuit (Ohms, float)
237 #define USER_MOTOR_Ls_d                 (0.008464367)  // For PM, Identified average stator inductance  (Henry, float)
238 #define USER_MOTOR_Ls_q                 (0.008464367)  // For PM, Identified average stator inductance  (Henry, float)
239 #define USER_MOTOR_RATED_FLUX           (0.38)         // Identified TOTAL flux linkage between the rotor and the stator (V/Hz)
240 #define USER_MOTOR_MAGNETIZING_CURRENT  (NULL)         // Induction motors only, else NULL
241 #define USER_MOTOR_RES_EST_CURRENT      (1.0)          // During Motor ID, maximum current (Amperes, float) used for Rs estimation, 10-20% rated current
242 #define USER_MOTOR_IND_EST_CURRENT      (-1.0)         // During Motor ID, maximum current (negative Amperes, float) used for Ls estimation, use just enough to
243 #define USER_MOTOR_MAX_CURRENT          (3.82)         // CRITICAL: Used during ID and run-time, sets a limit on the maximum current command output of the prov
244 #define USER_MOTOR_FLUX_EST_FREQ_Hz     (20.0)         // During Motor ID, maximum commanded speed (Hz, float), ~10% rated
245
246 #elif (USER_MOTOR == Anaheim_BLY172S)
247 #define USER_MOTOR_TYPE                 MOTOR_Type_Pm
248 #define USER_MOTOR_NUM_POLE_PAIRS       (4)
249 #define USER_MOTOR_Rr                   (NULL)
250 #define USER_MOTOR_Rs                   (0.4110007)
251 #define USER_MOTOR_Ls_d                 (0.0007092811)
252 #define USER_MOTOR_Ls_q                 (0.0007092811)
253 #define USER_MOTOR_RATED_FLUX           (0.03279636)
254 #define USER_MOTOR_MAGNETIZING_CURRENT  (NULL)
255 #define USER_MOTOR_RES_EST_CURRENT      (1.0)
256 #define USER_MOTOR_IND_EST_CURRENT      (-1.0)
257 #define USER_MOTOR_MAX_CURRENT          (5.0)
```

Figure A.61: Uncomment the Teknic motor and comment the other ones

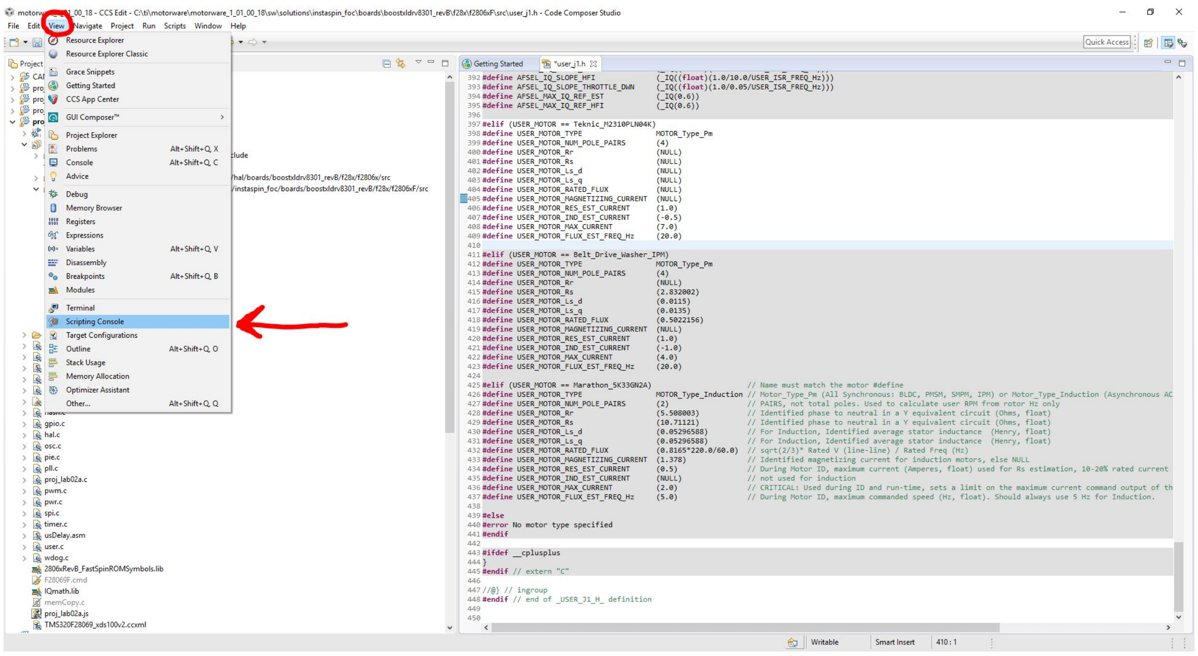
3. In the sections where the motors are defined, make sure the Teknic motor is defined as in fig. A.62.


motorware_1_01_00_18 - CCS Edit - C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\boards\boostxldrv8301_revB\f28x\f2806xF\src\user_j1.h - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
Project Explorer
CAN_RXLOOP
proj_lab01
proj_lab01b
proj_lab01c
proj_lab02a [Active - Release]
Binaries
Includes
C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/include
C:/ti/motorware/motorware_1_01_00_18
C:/ti/motorware/motorware_1_01_00_18/sw/modules/hal/boards/boostxldrv8301_revB/f28x/f2806x/src
C:/ti/motorware/motorware_1_01_00_18/sw/solutions/instaspin_foc/boards/boostxldrv8301_revB/f28x/f2806xF/src
user_2motors.h
user_j1.h
user_j5.h
user_motor_on_j1.h
user_motor_on_j5.h
user_motor1.h
user_motor2.h
user_mtr_on_j1.h
user_mtr_on_j5.h
user.h
user1.h
user2.h
Release
adc.c
clk.c
CodeStartBranch.asm
cpu.c
drv8301.c
F28069F_ram_lnk.cmd
flash.c
gpio.c
hal.c
osc.c
pie.c
pll.c
proj_lab02a.c
pwm.c
pwr.c
spi.c
timer.c
usDelay.asm
user.c
wdog.c
2806xRevB_FastSpinROMSymbols.lib
F28069F.cmd
IQmath.lib
memCopy.c
proj_lab02a.js
TMS320F28069_xds100v2.ccxml
Getting Started
*user_j1.h
Writable
Smart Insert
410 : 1

```
#define USER_MOTOR_NUM_POLE_PAIRS       (4)
#define USER_MOTOR_Rr                   (NULL)
#define USER_MOTOR_Rs                   (0.1215855)
#define USER_MOTOR_Ls_d                 (0.0002298828)
#define USER_MOTOR_Ls_q                 (0.0002298828)
#define USER_MOTOR_RATED_FLUX           (0.04821308)
#define USER_MOTOR_MAGNETIZING_CURRENT  (NULL)
#define USER_MOTOR_RES_EST_CURRENT      (2.0)         // Enter amperes(float)
#define USER_MOTOR_IND_EST_CURRENT      (-0.5)        // Enter negative amperes(float)
#define USER_MOTOR_MAX_CURRENT          (10.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz     (20.0)
#define IPD_HFI_EXC_FREQ_HZ             (750.0)       // excitation frequency, Hz
#define IPD_HFI_LP_SPD_FILT_HZ          (35.0)        // lowpass filter cutoff frequency, Hz
#define IPD_HFI_HP_IQ_FILT_HZ           (100.0)       // highpass filter cutoff frequency, Hz
#define IPD_HFI_KSPD                    (15.0)       // the speed gain value
#define IPD_HFI_EXC_MAG_COARSE_PU       (0.13)         // coarse IPD excitation magnitude, pu
#define IPD_HFI_EXC_MAG_FINE_PU         (0.12)         // fine IPD excitation magnitude, pu
#define IPD_HFI_EXC_TIME_COARSE_S       (0.5)         // coarse wait time, sec max 0.64
#define IPD_HFI_EXC_TIME_FINE_S         (0.5)         // fine wait time, sec max 0.4
#define AFSEL_FREQ_HIGH_PU              (_IQ(15.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_FREQ_LOW_PU               (_IQ(10.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_IQ_SLOPE_EST              (_IQ((float)(1.0/0.1/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_HFI              (_IQ((float)(1.0/10.0/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_THROTTLE_DWN     (_IQ((float)(1.0/0.05/USER_ISR_FREQ_Hz)))
#define AFSEL_MAX_IQ_REF_EST            (_IQ(0.6))
#define AFSEL_MAX_IQ_REF_HFI            (_IQ(0.6))

#elif (USER_MOTOR == Teknic_M2310PLN04K)
#define USER_MOTOR_TYPE                 MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS       (4)
#define USER_MOTOR_Rr                   (NULL)
#define USER_MOTOR_Rs                   (NULL)
#define USER_MOTOR_Ls_d                 (NULL)
#define USER_MOTOR_Ls_q                 (NULL)
#define USER_MOTOR_RATED_FLUX           (NULL)
#define USER_MOTOR_MAGNETIZING_CURRENT  (NULL)
#define USER_MOTOR_RES_EST_CURRENT      (1.0)
#define USER_MOTOR_IND_EST_CURRENT      (-0.5)
#define USER_MOTOR_MAX_CURRENT          (7.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz     (20.0)

#elif (USER_MOTOR == Belt_Drive_Washer_IPM)
#define USER_MOTOR_TYPE                 MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS       (4)
#define USER_MOTOR_Rr                   (NULL)
#define USER_MOTOR_Rs                   (2.832002)
#define USER_MOTOR_Ls_d                 (0.0115)
#define USER_MOTOR_Ls_q                 (0.0135)
#define USER_MOTOR_RATED_FLUX           (0.5022156)
#define USER_MOTOR_MAGNETIZING_CURRENT  (NULL)
#define USER_MOTOR_RES_EST_CURRENT      (1.0)
#define USER_MOTOR_IND_EST_CURRENT      (-1.0)
#define USER_MOTOR_MAX_CURRENT          (4.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz     (20.0)

#elif (USER_MOTOR == Marathon_5K33GN2A)                     // Name must match the motor #define
#define USER_MOTOR_TYPE                 MOTOR_Type_Induction // Motor_Type_Pm (All Synchronous: BLDC, PMSM, SMPM, IPM) or Motor_Type_Induction (Asynchronous AC
#define USER_MOTOR_NUM_POLE_PAIRS       (2)                  // PAIRS, not total poles. Used to calculate user RPM from rotor Hz only
#define USER_MOTOR_Rr                   (5.508003)           // Identified phase to neutral in a Y equivalent circuit (Ohms, float)
```

Figure A.62: Click ”Finish to load the projects into Code Composer Studio.

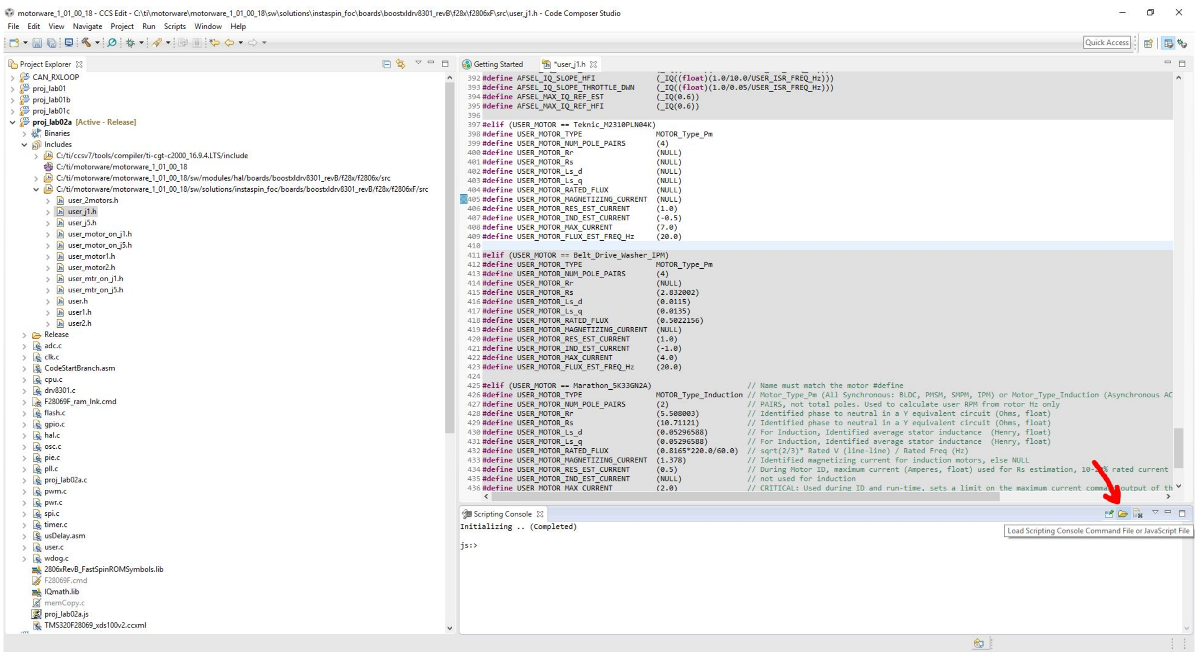
4. Open the Scripting Console from the view menu



```
#define AFSEL_IQ_SLOPE_HFI              (_IQ((float)(1.0/10.0/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_THROTTLE_DWN     (_IQ((float)(1.0/0.05/USER_ISR_FREQ_Hz)))
#define AFSEL_MAX_IQ_REF_EST            (_IQ(0.6))
#define AFSEL_MAX_IQ_REF_HFI            (_IQ(0.6))

#elif (USER_MOTOR == Teknic_M2310PLN04K)
#define USER_MOTOR_TYPE                 MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS       (4)
#define USER_MOTOR_Rr                   (NULL)
#define USER_MOTOR_Rs                   (NULL)
#define USER_MOTOR_Ls_d                 (NULL)
#define USER_MOTOR_Ls_q                 (NULL)
#define USER_MOTOR_RATED_FLUX           (NULL)
#define USER_MOTOR_MAGNETIZING_CURRENT  (NULL)
#define USER_MOTOR_RES_EST_CURRENT      (1.0)
#define USER_MOTOR_IND_EST_CURRENT      (-0.5)
#define USER_MOTOR_MAX_CURRENT          (7.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz     (20.0)

#elif (USER_MOTOR == Belt_Drive_Washer_IPM)
#define USER_MOTOR_TYPE                 MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS       (4)
#define USER_MOTOR_Rr                   (NULL)
#define USER_MOTOR_Rs                   (2.832002)
#define USER_MOTOR_Ls_d                 (0.0115)
#define USER_MOTOR_Ls_q                 (0.0135)
#define USER_MOTOR_RATED_FLUX           (0.5022156)
#define USER_MOTOR_MAGNETIZING_CURRENT  (NULL)
#define USER_MOTOR_RES_EST_CURRENT      (1.0)
#define USER_MOTOR_IND_EST_CURRENT      (-1.0)
#define USER_MOTOR_MAX_CURRENT          (4.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz     (20.0)

#elif (USER_MOTOR == Marathon_5K33GN2A)                     // Name must match the motor #define
#define USER_MOTOR_TYPE                 MOTOR_Type_Induction // Motor_Type_Pm (All Synchronous: BLDC, PMSM, SMPM, IPM) or Motor_Type_Induction (Asynchronous AC
#define USER_MOTOR_NUM_POLE_PAIRS       (2)                  // PAIRS, not total poles. Used to calculate user RPM from rotor Hz only
#define USER_MOTOR_Rr                   (5.508003)           // Identified phase to neutral in a Y equivalent circuit (Ohms, float)
#define USER_MOTOR_Rs                   (10.71121)           // Identified phase to neutral in a Y equivalent circuit (Ohms, float)
#define USER_MOTOR_Ls_d                 (0.05296588)         // For Induction, Identified average stator inductance  (Henry, float)
#define USER_MOTOR_Ls_q                 (0.05296588)         // For Induction, Identified average stator inductance  (Henry, float)
#define USER_MOTOR_RATED_FLUX           (0.8165*220.0/60.0)  // sqrt(2/3)* Rated V (line-line) / Rated Freq (Hz)
#define USER_MOTOR_MAGNETIZING_CURRENT  (1.378)              // Identified magnetizing current for induction motors, else NULL
#define USER_MOTOR_RES_EST_CURRENT      (0.5)                // During Motor ID, maximum current (Amperes, float) used for Rs estimation, 10-20% rated current
#define USER_MOTOR_IND_EST_CURRENT      (NULL)               // not used for induction
#define USER_MOTOR_MAX_CURRENT          (2.0)                // CRITICAL: Used during ID and run-time, sets a limit on the maximum current command output of th
#define USER_MOTOR_FLUX_EST_FREQ_Hz     (5.0)                // During Motor ID, maximum commanded speed (Hz, float). Should always use 5 Hz for Induction.

#else
#error No motor type specified
#endif

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

//@} // ingroup
#endif // end of _USER_J1_H_ definition


```

Figure A.63: Step 4

5. From the scripting console, click on the folder icon to open a file.


motorware_1_01_00_18 - CCS Edit - C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\boards\boostxldrv8301_revB\f28x\f2806xF\src\user_j1.h - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
Project Explorer
CAN_RXLOOP
proj_lab01
proj_lab01b
proj_lab01c
proj_lab02a [Active - Release]
Binaries
Includes
C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/include
C:/ti/motorware/motorware_1_01_00_18
C:/ti/motorware/motorware_1_01_00_18/sw/modules/hal/boards/boostxldrv8301_revB/f28x/f2806x/src
C:/ti/motorware/motorware_1_01_00_18/sw/solutions/instaspin_foc/boards/boostxldrv8301_revB/f28x/f2806xF/src
user_2motors.h
user_j1.h
user_j5.h
user_motor_on_j1.h
user_motor_on_j5.h
user_motor1.h
user_motor2.h
user_mtr_on_j1.h
user_mtr_on_j5.h
user.h
user1.h
user2.h
Release
adc.c
clk.c
CodeStartBranch.asm
cpu.c
drv8301.c
F28069F_ram_lnk.cmd
flash.c
gpio.c
hal.c
osc.c
pie.c
pll.c
proj_lab02a.c
pwm.c
pwr.c
spi.c
timer.c
usDelay.asm
user.c
wdog.c
2806xRevB_FastSpinROMSymbols.lib
F28069F.cmd
IQmath.lib
memCopy.c
proj_lab02a.js
TMS320F28069_xds100v2.ccxml
Getting Started
*user_j1.h
Scripting Console
Load Scripting Console Command File or JavaScript File

```
#define AFSEL_IQ_SLOPE_HFI              (_IQ((float)(1.0/10.0/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_THROTTLE_DWN     (_IQ((float)(1.0/0.05/USER_ISR_FREQ_Hz)))
#define AFSEL_MAX_IQ_REF_EST            (_IQ(0.6))
#define AFSEL_MAX_IQ_REF_HFI            (_IQ(0.6))

#elif (USER_MOTOR == Teknic_M2310PLN04K)
#define USER_MOTOR_TYPE                 MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS       (4)
#define USER_MOTOR_Rr                   (NULL)
#define USER_MOTOR_Rs                   (NULL)
#define USER_MOTOR_Ls_d                 (NULL)
#define USER_MOTOR_Ls_q                 (NULL)
#define USER_MOTOR_RATED_FLUX           (NULL)
#define USER_MOTOR_MAGNETIZING_CURRENT  (NULL)
#define USER_MOTOR_RES_EST_CURRENT      (1.0)
#define USER_MOTOR_IND_EST_CURRENT      (-0.5)
#define USER_MOTOR_MAX_CURRENT          (7.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz     (20.0)

#elif (USER_MOTOR == Belt_Drive_Washer_IPM)
#define USER_MOTOR_TYPE                 MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS       (4)
#define USER_MOTOR_Rr                   (NULL)
#define USER_MOTOR_Rs                   (2.832002)
#define USER_MOTOR_Ls_d                 (0.0115)
#define USER_MOTOR_Ls_q                 (0.0135)
#define USER_MOTOR_RATED_FLUX           (0.5022156)
#define USER_MOTOR_MAGNETIZING_CURRENT  (NULL)
#define USER_MOTOR_RES_EST_CURRENT      (1.0)
#define USER_MOTOR_IND_EST_CURRENT      (-1.0)
#define USER_MOTOR_MAX_CURRENT          (4.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz     (20.0)

#elif (USER_MOTOR == Marathon_5K33GN2A)                      // Name must match the motor #define
#define USER_MOTOR_TYPE                 MOTOR_Type_Induction // Motor_Type_Pm (All Synchronous: BLDC, PMSM, SMPM, IPM) or Motor_Type_Induction (Asynchronous AC
#define USER_MOTOR_NUM_POLE_PAIRS       (2)                  // PAIRS, not total poles. Used to calculate user RPM from rotor Hz only
#define USER_MOTOR_Rr                   (5.508003)           // Identified phase to neutral in a Y equivalent circuit (Ohms, float)
#define USER_MOTOR_Rs                   (10.71121)           // Identified phase to neutral in a Y equivalent circuit (Ohms, float)
#define USER_MOTOR_Ls_d                 (0.05296588)         // For Induction, Identified average stator inductance  (Henry, float)
#define USER_MOTOR_Ls_q                 (0.05296588)         // For Induction, Identified average stator inductance  (Henry, float)
#define USER_MOTOR_RATED_FLUX           (0.8165*220.0/60.0)  // sqrt(2/3)* Rated V (line-line) / Rated Freq (Hz)
#define USER_MOTOR_MAGNETIZING_CURRENT  (1.378)              // Identified magnetizing current for induction motors, else NULL
#define USER_MOTOR_RES_EST_CURRENT      (0.5)                // During Motor ID, maximum current (Amperes, float) used for Rs estimation, 10-20% rated current
#define USER_MOTOR_IND_EST_CURRENT      (NULL)               // not used for induction
#define USER_MOTOR_MAX_CURRENT          (2.0)                // CRITICAL: Used during ID and run-time, sets a limit on the maximum current comma[illegible] output of th
```

```
Initializing .. (Completed)

js:>
```

Figure A.64: Step 5

6. Go to the file location `C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src`

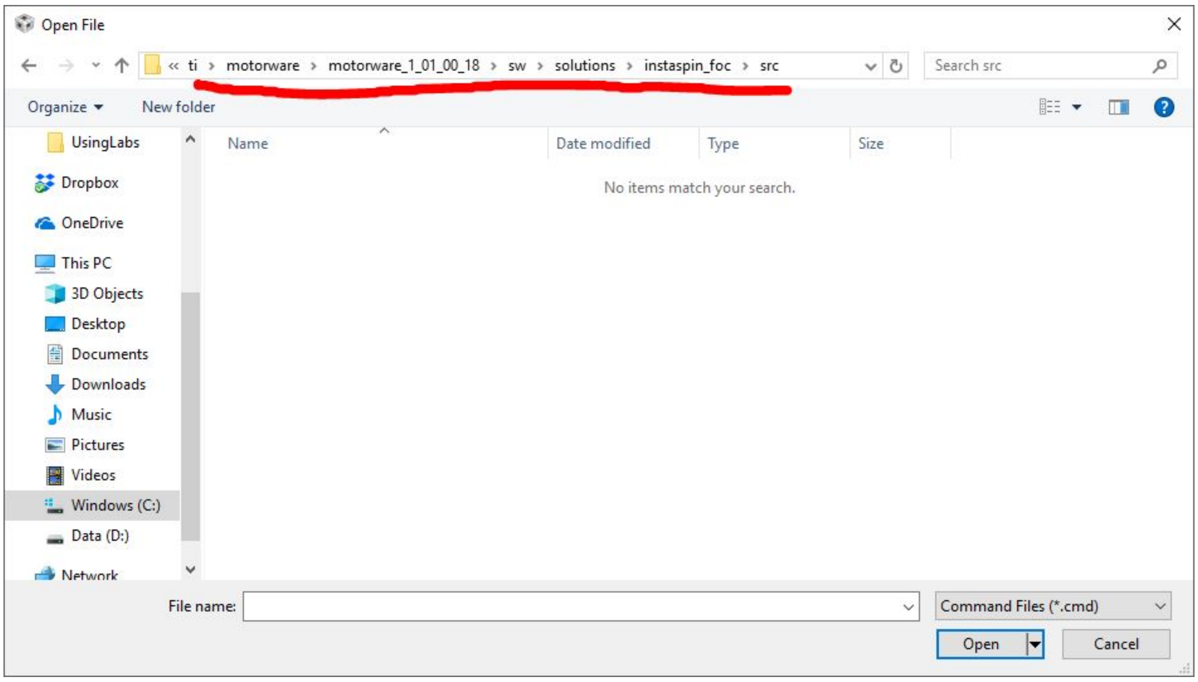
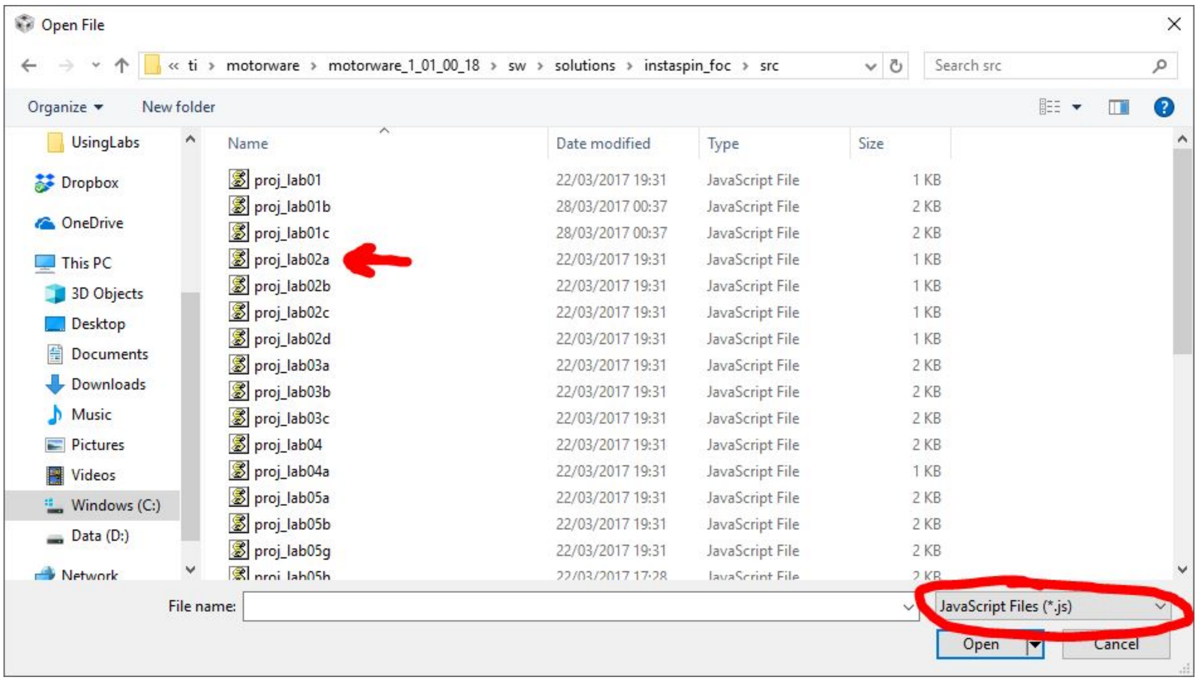


Figure A.65: Step 6

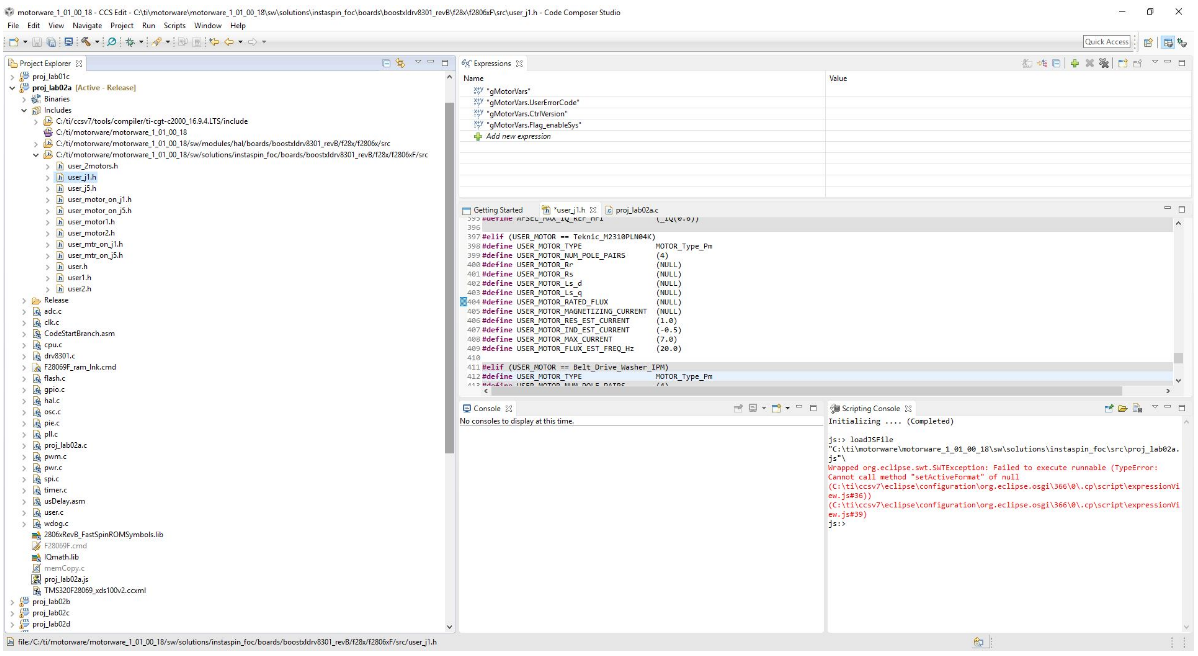
7. Check for JavaScript files, and open `proj_lab02a`.



Name	Date modified	Type	Size
proj_lab01	22/03/2017 19:31	JavaScript File	1 KB
proj_lab01b	28/03/2017 00:37	JavaScript File	2 KB
proj_lab01c	28/03/2017 00:37	JavaScript File	2 KB
proj_lab02a	22/03/2017 19:31	JavaScript File	1 KB
proj_lab02b	22/03/2017 19:31	JavaScript File	1 KB
proj_lab02c	22/03/2017 19:31	JavaScript File	1 KB
proj_lab02d	22/03/2017 19:31	JavaScript File	1 KB
proj_lab03a	22/03/2017 19:31	JavaScript File	2 KB
proj_lab03b	22/03/2017 19:31	JavaScript File	2 KB
proj_lab03c	22/03/2017 19:31	JavaScript File	2 KB
proj_lab04	22/03/2017 19:31	JavaScript File	2 KB
proj_lab04a	22/03/2017 19:31	JavaScript File	1 KB
proj_lab05a	22/03/2017 19:31	JavaScript File	2 KB
proj_lab05b	22/03/2017 19:31	JavaScript File	2 KB
proj_lab05g	22/03/2017 19:31	JavaScript File	2 KB
proj_lab05h	22/03/2017 17:28	JavaScript File	2 KB

Figure A.66: Step 7

8. The watch window will open in the top right. Here you will be able to monitor and change values while running the motor.


motorware_1_01_00_18 - CCS Edit - C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\boards\boostxldrv8301_revB\f28x\f2806xF\src\user_j1.h - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
Project Explorer
proj_lab01c
proj_lab02a [Active - Release]
Binaries
Includes
C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/include
C:/ti/motorware/motorware_1_01_00_18
C:/ti/motorware/motorware_1_01_00_18/sw/modules/hal/boards/boostxldrv8301_revB/f28x/f2806x/src
C:/ti/motorware/motorware_1_01_00_18/sw/solutions/instaspin_foc/boards/boostxldrv8301_revB/f28x/f2806xF/src
user_2motors.h
user_j1.h
user_j5.h
user_motor_on_j1.h
user_motor_on_j5.h
user_motor1.h
user_motor2.h
user_mtr_on_j1.h
user_mtr_on_j5.h
user.h
user1.h
user2.h
Release
adc.c
clk.c
CodeStartBranch.asm
cpu.c
drv8301.c
F28069F_ram_lnk.cmd
flash.c
gpio.c
hal.c
osc.c
pie.c
pll.c
proj_lab02a.c
pwm.c
pwr.c
spi.c
timer.c
usDelay.asm
user.c
wdog.c
2806xRevB_FastSpinROMSymbols.lib
F28069F.cmd
IQmath.lib
memCopy.c
proj_lab02a.js
TMS320F28069_xds100v2.ccxml
proj_lab02b
proj_lab02c
proj_lab02d
Expressions
Getting Started
*user_j1.h
proj_lab02a.c
Console
Scripting Console
file:/C:/ti/motorware/motorware_1_01_00_18/sw/solutions/instaspin_foc/boards/boostxldrv8301_revB/f28x/f2806xF/src/user_j1.h

Name	Value
"gMotorVars"	
"gMotorVars.UserErrorCode"	
"gMotorVars.CtrlVersion"	
"gMotorVars.Flag_enableSys"	
Add new expression	

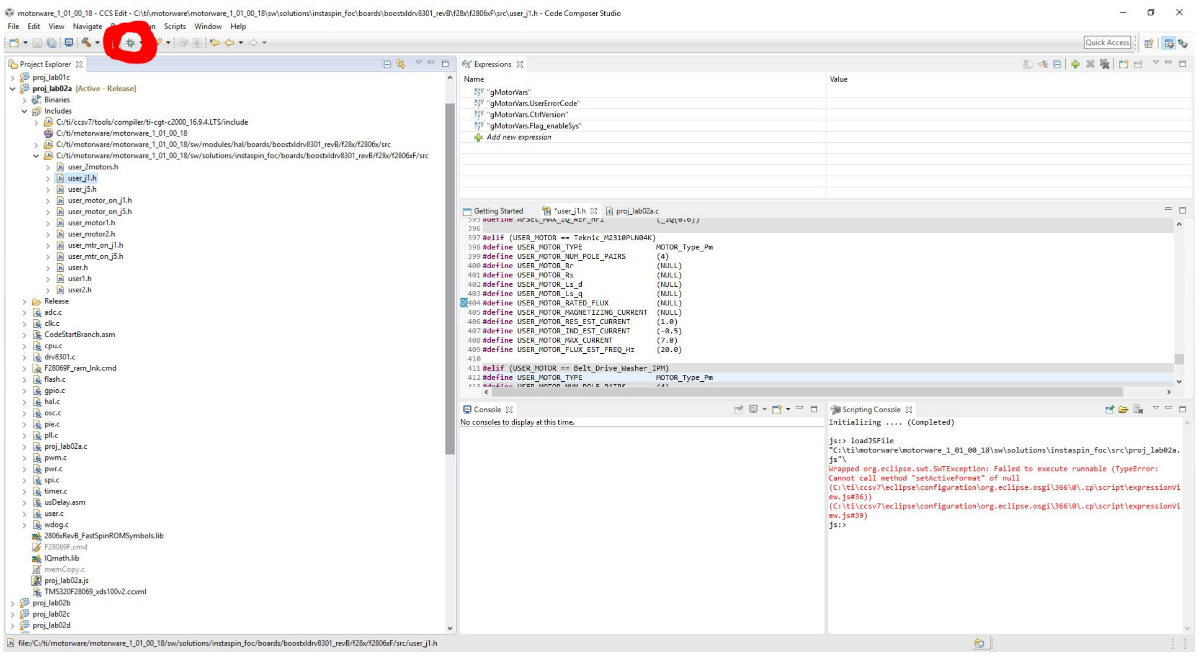
```
#elif (USER_MOTOR == Teknic_M2310PLN04K)
#define USER_MOTOR_TYPE                   MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS         (4)
#define USER_MOTOR_Rr                     (NULL)
#define USER_MOTOR_Rs                     (NULL)
#define USER_MOTOR_Ls_d                   (NULL)
#define USER_MOTOR_Ls_q                   (NULL)
#define USER_MOTOR_RATED_FLUX             (NULL)
#define USER_MOTOR_MAGNETIZING_CURRENT    (NULL)
#define USER_MOTOR_RES_EST_CURRENT        (1.0)
#define USER_MOTOR_IND_EST_CURRENT        (-0.5)
#define USER_MOTOR_MAX_CURRENT            (7.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz       (20.0)

#elif (USER_MOTOR == Belt_Drive_Washer_IPM)
#define USER_MOTOR_TYPE                   MOTOR_Type_Pm
```

```
No consoles to display at this time.
```

```
Initializing .... (Completed)

js:> loadJSFile
"C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab02a.
js"\
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError:
Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionVi
ew.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionVi
ew.js#39)
js:>
```

Figure A.67: Step 8

9. Click the debug button



motorware_1_01_00_18 - CCS Edit - C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\boards\boostxldrv8301_revB\f28x\f2806xF\src\user_j1.h - Code Composer Studio

Project Explorer

- proj_lab01c
- proj_lab02a [Active - Release]
  - Binaries
  - Includes
    - C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/include
    - C:/ti/motorware/motorware_1_01_00_18
    - C:/ti/motorware/motorware_1_01_00_18/sw/modules/hal/boards/boostxldrv8301_revB/f28x/f2806x/src
    - C:/ti/motorware/motorware_1_01_00_18/sw/solutions/instaspin_foc/boards/boostxldrv8301_revB/f28x/f2806xF/src
      - user_2motors.h
      - user_j1.h
      - user_j5.h
      - user_motor_on_j1.h
      - user_motor_on_j5.h
      - user_motor1.h
      - user_motor2.h
      - user_mtr_on_j1.h
      - user_mtr_on_j5.h
      - user.h
      - user1.h
      - user2.h
  - Release
  - adc.c
  - clk.c
  - CodeStartBranch.asm
  - cpu.c
  - drv8301.c
  - F28069F_ram_lnk.cmd
  - flash.c
  - gpio.c
  - hal.c
  - osc.c
  - pie.c
  - pll.c
  - proj_lab02a.c
  - pwm.c
  - pwr.c
  - spi.c
  - timer.c
  - usDelay.asm
  - user.c
  - wdog.c
  - 2806xRevB_FastSpinROMSymbols.lib
  - F28069F.cmd
  - IQmath.lib
  - memCopy.c
  - proj_lab02a.js
  - TMS320F28069_xds100v2.ccxml
- proj_lab02b
- proj_lab02c
- proj_lab02d

file:/C:/ti/motorware/motorware_1_01_00_18/sw/solutions/instaspin_foc/boards/boostxldrv8301_revB/f28x/f2806xF/src/user_j1.h

Expressions

Name	Value
"gMotorVars"	
"gMotorVars.UserErrorCode"	
"gMotorVars.CtrlVersion"	
"gMotorVars.Flag_enableSys"	
Add new expression	

Getting Started | *user_j1.h | proj_lab02a.c

```
#elif (USER_MOTOR == Teknic_M2310PLN04K)
#define USER_MOTOR_TYPE                 MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS       (4)
#define USER_MOTOR_Rr                   (NULL)
#define USER_MOTOR_Rs                   (NULL)
#define USER_MOTOR_Ls_d                 (NULL)
#define USER_MOTOR_Ls_q                 (NULL)
#define USER_MOTOR_RATED_FLUX           (NULL)
#define USER_MOTOR_MAGNETIZING_CURRENT  (NULL)
#define USER_MOTOR_RES_EST_CURRENT      (1.0)
#define USER_MOTOR_IND_EST_CURRENT      (-0.5)
#define USER_MOTOR_MAX_CURRENT          (7.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz     (20.0)

#elif (USER_MOTOR == Belt_Drive_Washer_IPM)
#define USER_MOTOR_TYPE                 MOTOR_Type_Pm
```

Console

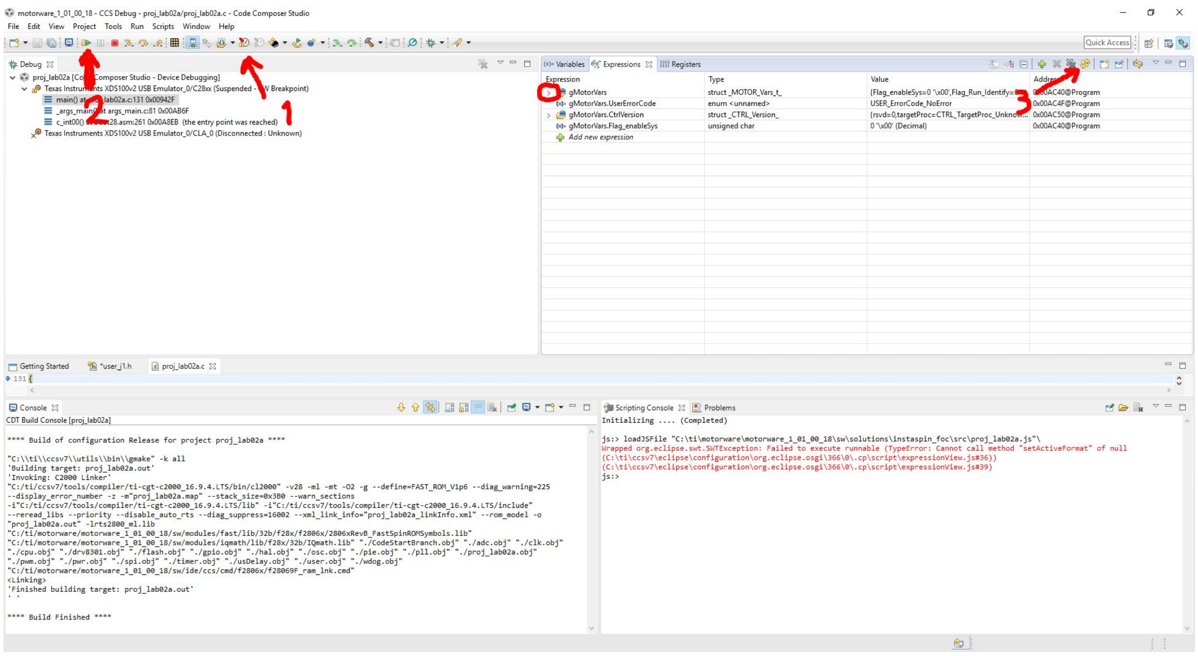
No consoles to display at this time.

Scripting Console

```
Initializing .... (Completed)

js:> loadJSFile
"C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab02a.
js"\
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError:
Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionVi
ew.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionVi
ew.js#39)
js:>
```

Figure A.68: The debug button is located in the top to the left

10. Enable real time (1), run the program (2) and allow continuous refresh in the watch window (3). Expand the gMotorVars folder in the watch window.



motorware_1_01_00_18 - CCS Debug - proj_lab02a/proj_lab02a.c - Code Composer Studio

Debug

- proj_lab02a [Code Composer Studio - Device Debugging]
  - Texas Instruments XDS100v2 USB Emulator_0/C28xx (Suspended - SW Breakpoint)
    - main() at proj_lab02a.c:131 0x00942F
    - _args_main() at args_main.c:81 0x00AB6F
    - c_int00() at boot28.asm:261 0x00A8EB (the entry point was reached)
  - Texas Instruments XDS100v2 USB Emulator_0/CLA_0 (Disconnected : Unknown)

Variables | Expressions | Registers

Expression	Type	Value	Address
gMotorVars	struct _MOTOR_Vars_t_	{Flag_enableSys=0 '\x00',Flag_Run_Identify=0...	0x00AC40@Program
gMotorVars.UserErrorCode	enum <unnamed>	USER_ErrorCode_NoError	0x00AC4F@Program
gMotorVars.CtrlVersion	struct _CTRL_Version_	{rsvd=0,targetProc=CTRL_TargetProc_Unknow...	0x00AC50@Program
gMotorVars.Flag_enableSys	unsigned char	0 '\x00' (Decimal)	0x00AC40@Program
Add new expression			

Getting Started | *user_j1.h | proj_lab02a.c

```
{
```

Console

CDT Build Console [proj_lab02a]

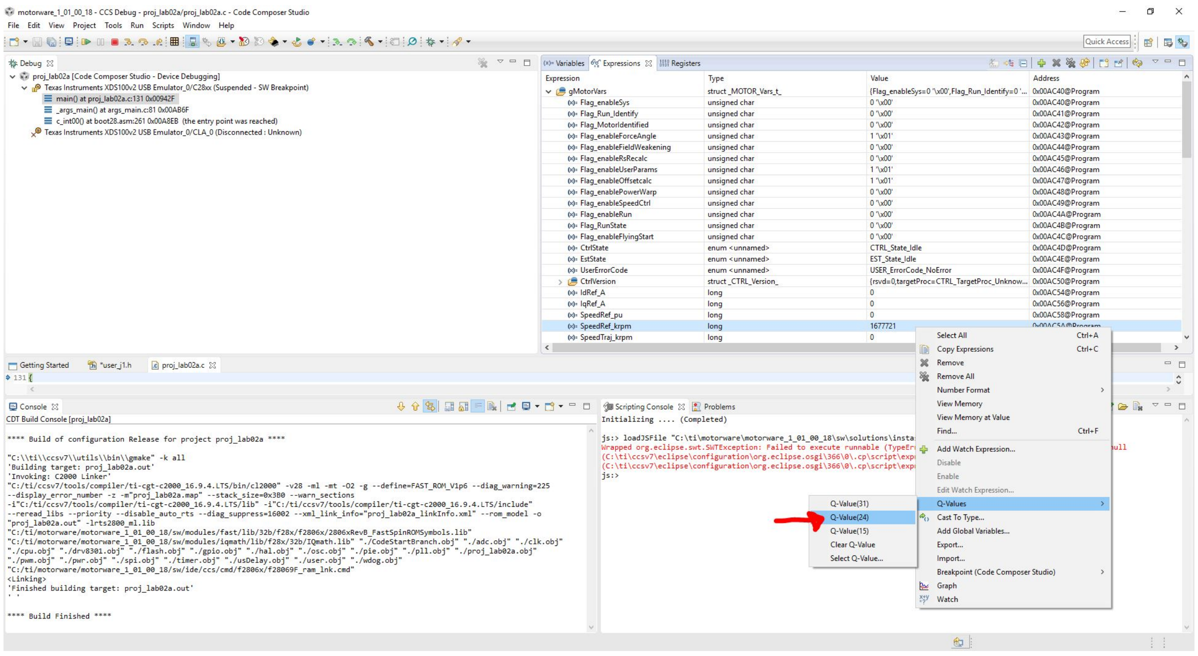
```
**** Build of configuration Release for project proj_lab02a ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
'Building target: proj_lab02a.out'
'Invoking: C2000 Linker'
"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/bin/cl2000" -v28 -ml -mt -O2 -g --define=FAST_ROM_V1p6 --diag_warning=225
--display_error_number -z -m"proj_lab02a.map" --stack_size=0x3B0 --warn_sections
-i"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/lib" -i"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/include"
--reread_libs --priority --disable_auto_rts --diag_suppress=16002 --xml_link_info="proj_lab02a_linkInfo.xml" --rom_model -o
"proj_lab02a.out" -lrts2800_ml.lib
"C:/ti/motorware/motorware_1_01_00_18/sw/modules/fast/lib/32b/f28x/f2806x/2806xRevB_FastSpinROMSymbols.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/modules/iqmath/lib/f28x/32b/IQmath.lib" "./CodeStartBranch.obj" "./adc.obj" "./clk.obj"
"./cpu.obj" "./drv8301.obj" "./flash.obj" "./gpio.obj" "./hal.obj" "./osc.obj" "./pie.obj" "./pll.obj" "./proj_lab02a.obj"
"./pwm.obj" "./pwr.obj" "./spi.obj" "./timer.obj" "./usDelay.obj" "./user.obj" "./wdog.obj"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram_lnk.cmd"
<Linking>
'Finished building target: proj_lab02a.out'
' '

**** Build Finished ****
```

Scripting Console | Problems

```
Initializing .... (Completed)

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab02a.js"\
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:>
```

Figure A.69: Step 10

11. Check if the value show what you expect. For example, SpeedRef_krpm should show a decimal number. Right-click the value and change the Q-value to 24.


motorware_1_01_00_18 - CCS Debug - proj_lab02a/proj_lab02a.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
proj_lab02a [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v2 USB Emulator_0/C28xx (Suspended - SW Breakpoint)
main() at proj_lab02a.c:131 0x00942F
_args_main() at args_main.c:81 0x00AB6F
c_int00() at boot28.asm:261 0x00A8EB (the entry point was reached)
Texas Instruments XDS100v2 USB Emulator_0/CLA_0 (Disconnected : Unknown)
Variables
Expressions
Registers
Select All Ctrl+A
Copy Expressions Ctrl+C
Remove
Remove All
Number Format
View Memory
View Memory at Value
Find... Ctrl+F
Add Watch Expression...
Disable
Enable
Edit Watch Expression...
Q-Values
Cast To Type...
Add Global Variables...
Export...
Import...
Breakpoint (Code Composer Studio)
Graph
Watch
Q-Value(31)
Q-Value(24)
Q-Value(15)
Clear Q-Value
Select Q-Value...
Getting Started
*user_j1.h
proj_lab02a.c
131 {
Console
Scripting Console
Problems

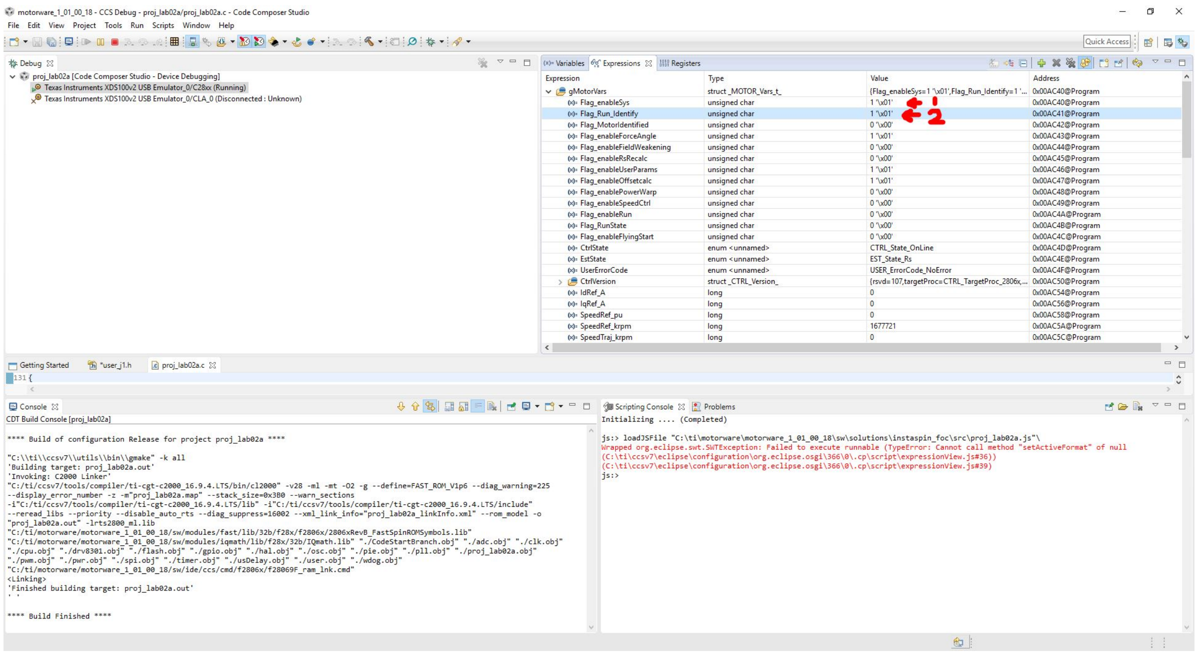
Expression	Type	Value	Address
gMotorVars	struct _MOTOR_Vars_t_	{Flag_enableSys=0 '\x00',Flag_Run_Identify=0 '...	0x00AC40@Program
Flag_enableSys	unsigned char	0 '\x00'	0x00AC40@Program
Flag_Run_Identify	unsigned char	0 '\x00'	0x00AC41@Program
Flag_MotorIdentified	unsigned char	0 '\x00'	0x00AC42@Program
Flag_enableForceAngle	unsigned char	1 '\x01'	0x00AC43@Program
Flag_enableFieldWeakening	unsigned char	0 '\x00'	0x00AC44@Program
Flag_enableRsRecalc	unsigned char	0 '\x00'	0x00AC45@Program
Flag_enableUserParams	unsigned char	1 '\x01'	0x00AC46@Program
Flag_enableOffsetcalc	unsigned char	1 '\x01'	0x00AC47@Program
Flag_enablePowerWarp	unsigned char	0 '\x00'	0x00AC48@Program
Flag_enableSpeedCtrl	unsigned char	0 '\x00'	0x00AC49@Program
Flag_enableRun	unsigned char	0 '\x00'	0x00AC4A@Program
Flag_RunState	unsigned char	0 '\x00'	0x00AC4B@Program
Flag_enableFlyingStart	unsigned char	0 '\x00'	0x00AC4C@Program
CtrlState	enum <unnamed>	CTRL_State_Idle	0x00AC4D@Program
EstState	enum <unnamed>	EST_State_Idle	0x00AC4E@Program
UserErrorCode	enum <unnamed>	USER_ErrorCode_NoError	0x00AC4F@Program
CtrlVersion	struct _CTRL_Version_	{rsvd=0,targetProc=CTRL_TargetProc_Unknow...	0x00AC50@Program
IdRef_A	long	0	0x00AC54@Program
IqRef_A	long	0	0x00AC56@Program
SpeedRef_pu	long	0	0x00AC58@Program
SpeedRef_krpm	long	1677721	0x00AC5A@Program
SpeedTraj_krpm	long	0	

```
CDT Build Console [proj_lab02a]

**** Build of configuration Release for project proj_lab02a ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
'Building target: proj_lab02a.out'
'Invoking: C2000 Linker'
"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/bin/cl2000" -v28 -ml -mt -O2 -g --define=FAST_ROM_V1p6 --diag_warning=225
--display_error_number -z -m"proj_lab02a.map" --stack_size=0x3B0 --warn_sections
-i"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/lib" -i"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/include"
--reread_libs --priority --disable_auto_rts --diag_suppress=16002 --xml_link_info="proj_lab02a_linkInfo.xml" --rom_model -o
"proj_lab02a.out" -lrts2800_ml.lib
"C:/ti/motorware/motorware_1_01_00_18/sw/modules/fast/lib/32b/f28x/f2806x/2806xRevB_FastSpinROMSymbols.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/modules/iqmath/lib/f28x/32b/IQmath.lib" "./CodeStartBranch.obj" "./adc.obj" "./clk.obj"
"./cpu.obj" "./drv8301.obj" "./flash.obj" "./gpio.obj" "./hal.obj" "./osc.obj" "./pie.obj" "./pll.obj" "./proj_lab02a.obj"
"./pwm.obj" "./pwr.obj" "./spi.obj" "./timer.obj" "./usDelay.obj" "./user.obj" "./wdog.obj"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram_lnk.cmd"
<Linking>
'Finished building target: proj_lab02a.out'
' '

**** Build Finished ****
```

```
Initializing .... (Completed)

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\insta
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeEr                                    ull
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\exp
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\exp
js:>
```

Figure A.70: Step 11

12. First set Flag_enableSys to 1, then set Flag_Run_Identify to 1. This will run the motor identification. After the identification is done, the motor will start spinning normally.


motorware_1_01_00_18 - CCS Debug - proj_lab02a/proj_lab02a.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
proj_lab02a [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v2 USB Emulator_0/C28xx (Running)
Texas Instruments XDS100v2 USB Emulator_0/CLA_0 (Disconnected : Unknown)
Variables
Expressions
Registers
1
2
Getting Started
*user_j1.h
proj_lab02a.c
131 {
Console
Scripting Console
Problems

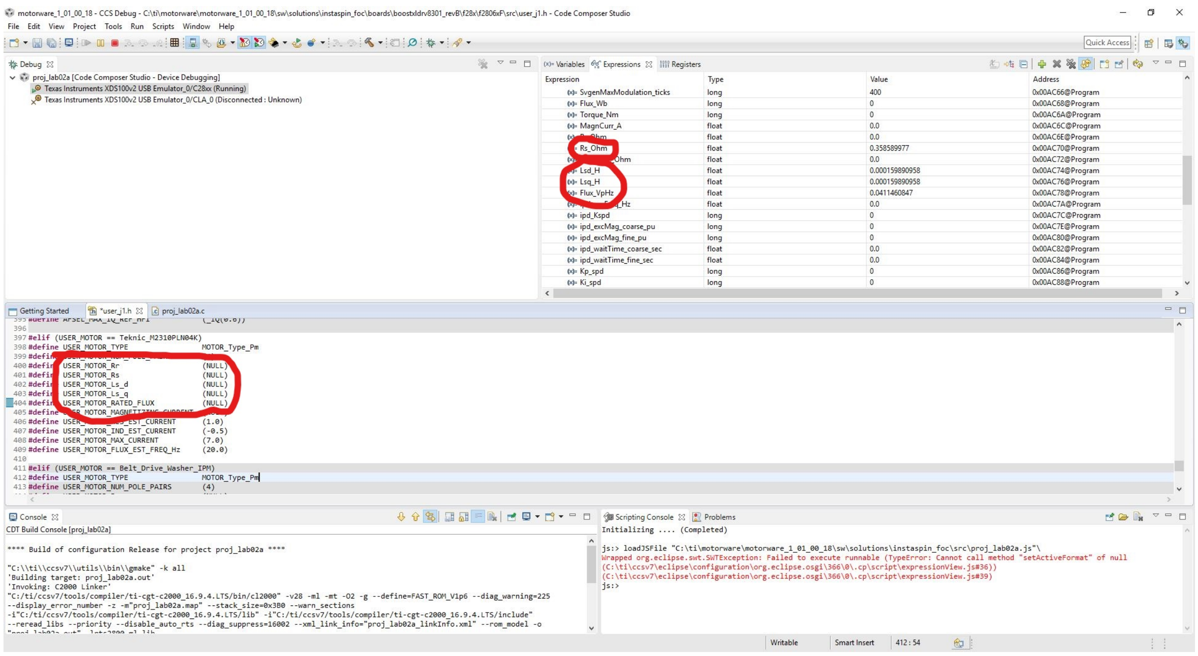
Expression	Type	Value	Address
gMotorVars	struct _MOTOR_Vars_t_	{Flag_enableSys=1 '\x01',Flag_Run_Identify=1 '...	0x00AC40@Program
Flag_enableSys	unsigned char	1 '\x01'	0x00AC40@Program
Flag_Run_Identify	unsigned char	1 '\x01'	0x00AC41@Program
Flag_MotorIdentified	unsigned char	0 '\x00'	0x00AC42@Program
Flag_enableForceAngle	unsigned char	1 '\x01'	0x00AC43@Program
Flag_enableFieldWeakening	unsigned char	0 '\x00'	0x00AC44@Program
Flag_enableRsRecalc	unsigned char	0 '\x00'	0x00AC45@Program
Flag_enableUserParams	unsigned char	1 '\x01'	0x00AC46@Program
Flag_enableOffsetcalc	unsigned char	1 '\x01'	0x00AC47@Program
Flag_enablePowerWarp	unsigned char	0 '\x00'	0x00AC48@Program
Flag_enableSpeedCtrl	unsigned char	0 '\x00'	0x00AC49@Program
Flag_enableRun	unsigned char	0 '\x00'	0x00AC4A@Program
Flag_RunState	unsigned char	0 '\x00'	0x00AC4B@Program
Flag_enableFlyingStart	unsigned char	0 '\x00'	0x00AC4C@Program
CtrlState	enum <unnamed>	CTRL_State_OnLine	0x00AC4D@Program
EstState	enum <unnamed>	EST_State_Rs	0x00AC4E@Program
UserErrorCode	enum <unnamed>	USER_ErrorCode_NoError	0x00AC4F@Program
CtrlVersion	struct _CTRL_Version_	{rsvd=107,targetProc=CTRL_TargetProc_2806x,...	0x00AC50@Program
IdRef_A	long	0	0x00AC54@Program
IqRef_A	long	0	0x00AC56@Program
SpeedRef_pu	long	0	0x00AC58@Program
SpeedRef_krpm	long	1677721	0x00AC5A@Program
SpeedTraj_krpm	long	0	0x00AC5C@Program

```
CDT Build Console [proj_lab02a]

**** Build of configuration Release for project proj_lab02a ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
'Building target: proj_lab02a.out'
'Invoking: C2000 Linker'
"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/bin/cl2000" -v28 -ml -mt -O2 -g --define=FAST_ROM_V1p6 --diag_warning=225
--display_error_number -z -m"proj_lab02a.map" --stack_size=0x3B0 --warn_sections
-i"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/lib" -i"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/include"
--reread_libs --priority --disable_auto_rts --diag_suppress=16002 --xml_link_info="proj_lab02a_linkInfo.xml" --rom_model -o
"proj_lab02a.out" -lrts2800_ml.lib
"C:/ti/motorware/motorware_1_01_00_18/sw/modules/fast/lib/32b/f28x/f2806x/2806xRevB_FastSpinROMSymbols.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/modules/iqmath/lib/f28x/32b/IQmath.lib" "./CodeStartBranch.obj" "./adc.obj" "./clk.obj"
"./cpu.obj" "./drv8301.obj" "./flash.obj" "./gpio.obj" "./hal.obj" "./osc.obj" "./pie.obj" "./pll.obj" "./proj_lab02a.obj"
"./pwm.obj" "./pwr.obj" "./spi.obj" "./timer.obj" "./usDelay.obj" "./user.obj" "./wdog.obj"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram_lnk.cmd"
<Linking>
'Finished building target: proj_lab02a.out'
' '

**** Build Finished ****
```

```
Initializing .... (Completed)

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab02a.js"\
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:>
```

Figure A.71: Step 12

13. After the motor identification is done, copy the following values in the watch window to the `user_j1.h` file: Rs, Ls_d, Ls_q and Flux_VpHz (the last one in RATED_FLUX).



motorware_1_01_00_18 - CCS Debug - C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\boards\boostxldrv8301_revB\f28x\f2806xF\src\user_j1.h - Code Composer Studio

proj_lab02a [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v2 USB Emulator_0/C28xx (Running)
Texas Instruments XDS100v2 USB Emulator_0/CLA_0 (Disconnected : Unknown)

Expression	Type	Value	Address
SvgenMaxModulation_ticks	long	400	0x00AC66@Program
Flux_Wb	long	0	0x00AC68@Program
Torque_Nm	long	0	0x00AC6A@Program
MagnCurr_A	float	0.0	0x00AC6C@Program
Rr_Ohm	float	0.0	0x00AC6E@Program
Rs_Ohm	float	0.358589977	0x00AC70@Program
[illegible]_Ohm	float	0.0	0x00AC72@Program
Lsd_H	float	0.000159890958	0x00AC74@Program
Lsq_H	float	0.000159890958	0x00AC76@Program
Flux_VpHz	float	0.0411460847	0x00AC78@Program
ipd_excFreq_Hz	float	0.0	0x00AC7A@Program
ipd_Kspd	long	0	0x00AC7C@Program
ipd_excMag_coarse_pu	long	0	0x00AC7E@Program
ipd_excMag_fine_pu	long	0	0x00AC80@Program
ipd_waitTime_coarse_sec	float	0.0	0x00AC82@Program
ipd_waitTime_fine_sec	float	0.0	0x00AC84@Program
Kp_spd	long	0	0x00AC86@Program
Ki_spd	long	0	0x00AC88@Program

```
#elif (USER_MOTOR == Teknic_M2310PLN04K)
#define USER_MOTOR_TYPE                   MOTOR_Type_Pm
#defi
#defi  USER_MOTOR_Rr                      (NULL)
#defi  USER_MOTOR_Rs                      (NULL)
#defi  USER_MOTOR_Ls_d                    (NULL)
#defi  USER_MOTOR_Ls_q                    (NULL)
#defi  USER_MOTOR_RATED_FLUX              (NULL)
#define    R_MOTOR_MAGNETIZING_CURRENT
#define USER        _EST_CURRENT          (1.0)
#define USER_MOTOR_IND_EST_CURRENT        (-0.5)
#define USER_MOTOR_MAX_CURRENT            (7.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz       (20.0)

#elif (USER_MOTOR == Belt_Drive_Washer_IPM)
#define USER_MOTOR_TYPE                   MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS         (4)
```

CDT Build Console [proj_lab02a]

```
**** Build of configuration Release for project proj_lab02a ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
'Building target: proj_lab02a.out'
'Invoking: C2000 Linker'
"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/bin/cl2000" -v28 -ml -mt -O2 -g --define=FAST_ROM_V1p6 --diag_warning=225
--display_error_number -z -m"proj_lab02a.map" --stack_size=0x3B0 --warn_sections
-i"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/lib" -i"C:/ti/ccsv7/tools/compiler/ti-cgt-c2000_16.9.4.LTS/include"
--reread_libs --priority --disable_auto_rts --diag_suppress=16002 --xml_link_info="proj_lab02a_linkInfo.xml" --rom_model -o
```

```
Initializing .... (Completed)

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab02a.js"\
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:>
```

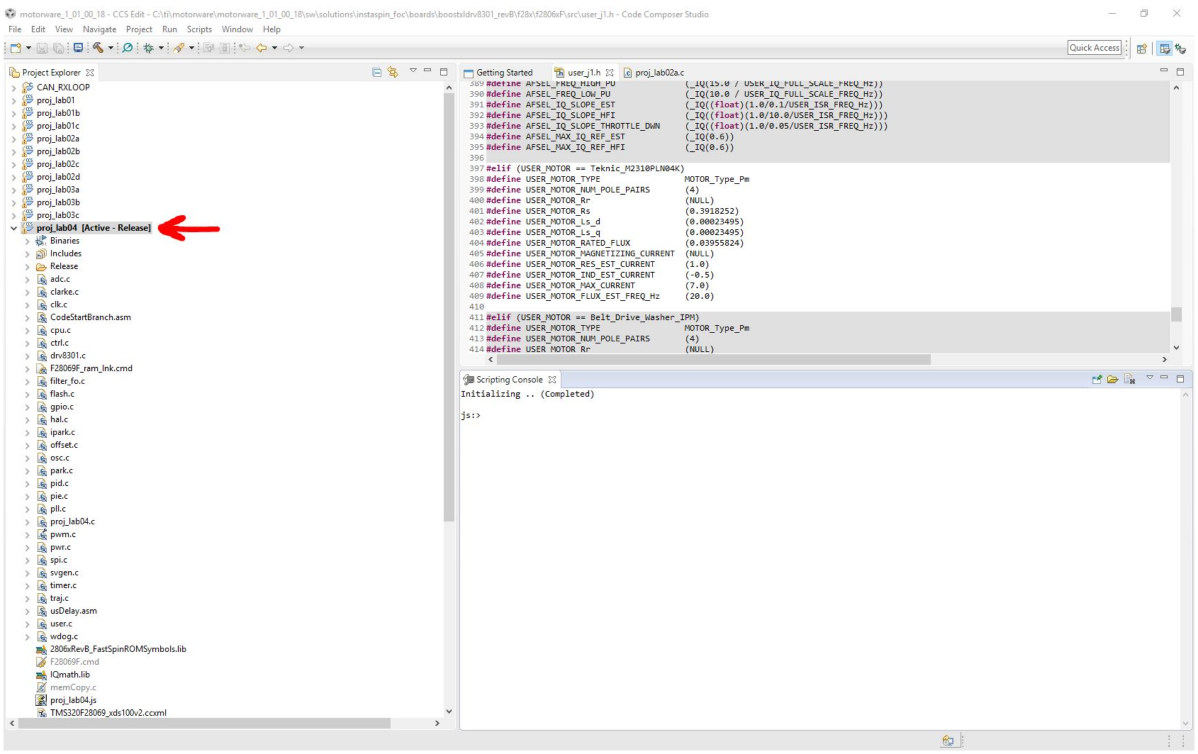
Figure A.72: Step 13

14. When the motor identification is done and the values are copied to user_j1.h the program can be stopped and the next part of this manual can be followed.

A.6.2. Using lab 4 for the torque load

Running lab 4 works essentially the same as running lab 2, so only a few differences will be explained here. It is assumed the motor parameters determined lab 2a are stored in user_j1.h.

1. Double click on proj_lab04. This will set lab 4 as the active project.



motorware_1_01_00_18 - CCS Edit - C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\boards\boostxldrv8301_revB\f28x\f2806xF\src\user_j1.h - Code Composer Studio

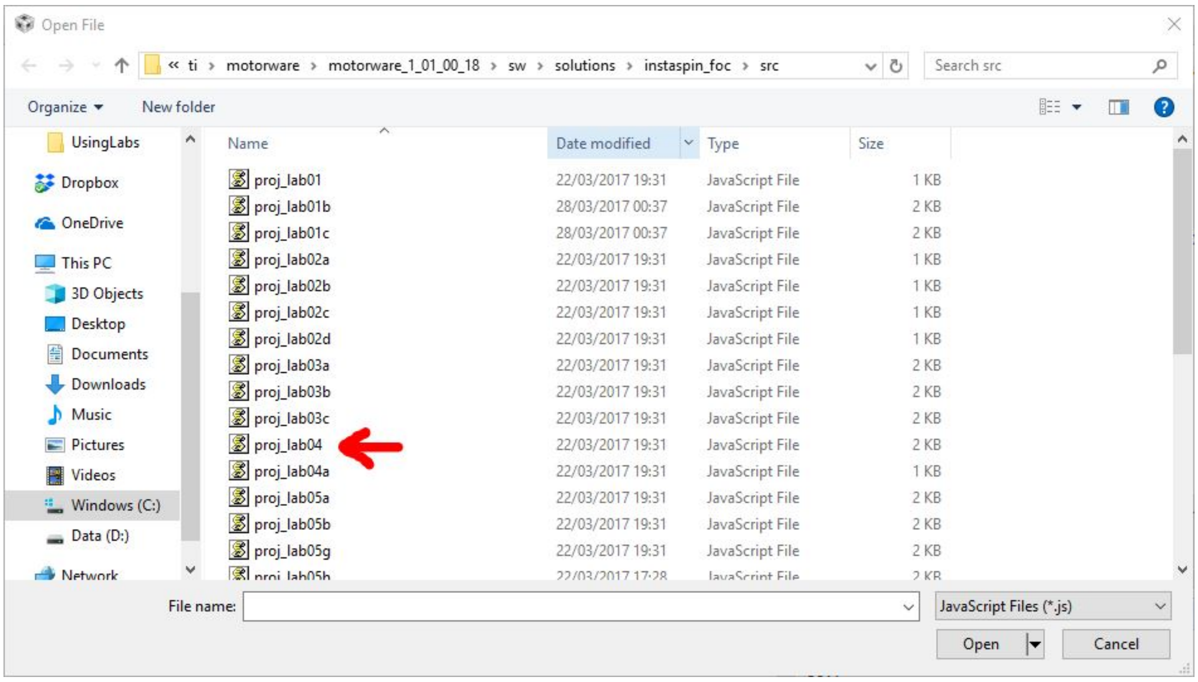
```
CAN_RXLOOP
proj_lab01
proj_lab01b
proj_lab01c
proj_lab02a
proj_lab02b
proj_lab02c
proj_lab02d
proj_lab03a
proj_lab03b
proj_lab03c
proj_lab04 [Active - Release]
  Binaries
  Includes
  Release
  adc.c
  clarke.c
  clk.c
  CodeStartBranch.asm
  cpu.c
  ctrl.c
  drv8301.c
  F28069F_ram_lnk.cmd
  filter_fo.c
  flash.c
  gpio.c
  hal.c
  ipark.c
  offset.c
  osc.c
  park.c
  pid.c
  pie.c
  pll.c
  proj_lab04.c
  pwm.c
  pwr.c
  spi.c
  svgen.c
  timer.c
  traj.c
  usDelay.asm
  user.c
  wdog.c
  2806xRevB_FastSpinROMSymbols.lib
  F28069F.cmd
  IQmath.lib
  memCopy.c
  proj_lab04.js
  TMS320F28069_xds100v2.ccxml
```

```c
#define AFSEL_FREQ_HIGH_PU              (_IQ(15.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_FREQ_LOW_PU               (_IQ(10.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_IQ_SLOPE_EST              (_IQ((float)(1.0/0.1/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_HFI              (_IQ((float)(1.0/10.0/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_THROTTLE_DWN     (_IQ((float)(1.0/0.05/USER_ISR_FREQ_Hz)))
#define AFSEL_MAX_IQ_REF_EST            (_IQ(0.6))
#define AFSEL_MAX_IQ_REF_HFI            (_IQ(0.6))

#elif (USER_MOTOR == Teknic_M2310PLN04K)
#define USER_MOTOR_TYPE                 MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS       (4)
#define USER_MOTOR_Rr                   (NULL)
#define USER_MOTOR_Rs                   (0.3918252)
#define USER_MOTOR_Ls_d                 (0.00023495)
#define USER_MOTOR_Ls_q                 (0.00023495)
#define USER_MOTOR_RATED_FLUX           (0.03955824)
#define USER_MOTOR_MAGNETIZING_CURRENT  (NULL)
#define USER_MOTOR_RES_EST_CURRENT      (1.0)
#define USER_MOTOR_IND_EST_CURRENT      (-0.5)
#define USER_MOTOR_MAX_CURRENT          (7.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz     (20.0)

#elif (USER_MOTOR == Belt_Drive_Washer_IPM)
#define USER_MOTOR_TYPE                 MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS       (4)
#define USER_MOTOR_Rr                   (NULL)
```

```
Initializing .. (Completed)

js:>
```

Figure A.73: Step 1

2. Open a different JavaScript file from the scripting console: proj_lab04.



Open File

« ti › motorware › motorware_1_01_00_18 › sw › solutions › instaspin_foc › src

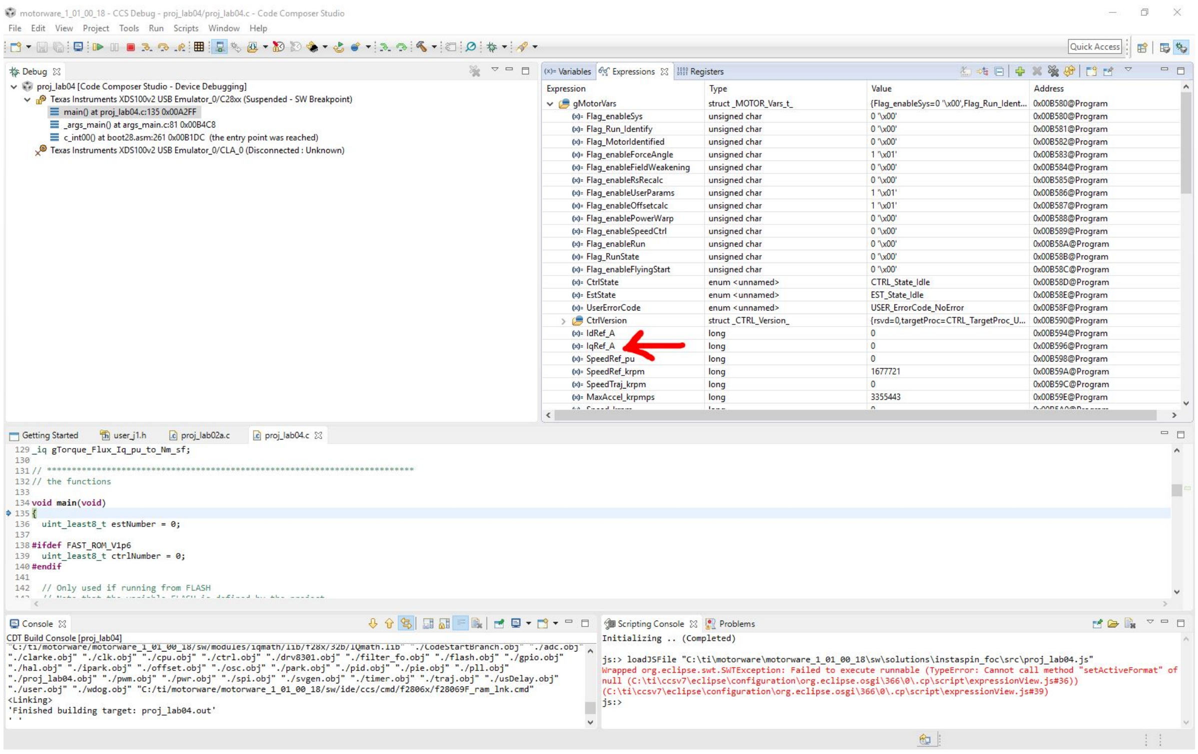
Name	Date modified	Type	Size
proj_lab01	22/03/2017 19:31	JavaScript File	1 KB
proj_lab01b	28/03/2017 00:37	JavaScript File	2 KB
proj_lab01c	28/03/2017 00:37	JavaScript File	2 KB
proj_lab02a	22/03/2017 19:31	JavaScript File	1 KB
proj_lab02b	22/03/2017 19:31	JavaScript File	1 KB
proj_lab02c	22/03/2017 19:31	JavaScript File	1 KB
proj_lab02d	22/03/2017 19:31	JavaScript File	1 KB
proj_lab03a	22/03/2017 19:31	JavaScript File	2 KB
proj_lab03b	22/03/2017 19:31	JavaScript File	2 KB
proj_lab03c	22/03/2017 19:31	JavaScript File	2 KB
proj_lab04	22/03/2017 19:31	JavaScript File	2 KB
proj_lab04a	22/03/2017 19:31	JavaScript File	1 KB
proj_lab05a	22/03/2017 19:31	JavaScript File	2 KB
proj_lab05b	22/03/2017 19:31	JavaScript File	2 KB
proj_lab05g	22/03/2017 19:31	JavaScript File	2 KB
proj_lab05h	22/03/2017 17:28	JavaScript File	2 KB

File name:

JavaScript Files (*.js)

Figure A.74: Open a different config file.

3. The torque of the motor can be changed in the expressions window using IqRef_A (make sure the variable is set to Q-value(24)).



```
motorware_1_01_00_18 - CCS Debug - proj_lab04/proj_lab04.c - Code Composer Studio

Debug
proj_lab04 [Code Composer Studio - Device Debugging]
  Texas Instruments XDS100v2 USB Emulator_0/C28xx (Suspended - SW Breakpoint)
    main() at proj_lab04.c:135 0x00A2FF
    _args_main() at args_main.c:81 0x00B4C8
    c_int00() at boot28.asm:261 0x00B1DC  (the entry point was reached)
  Texas Instruments XDS100v2 USB Emulator_0/CLA_0 (Disconnected : Unknown)
```

Expression	Type	Value	Address
gMotorVars	struct _MOTOR_Vars_t_	{Flag_enableSys=0 '\x00',Flag_Run_Ident...	0x00B580@Program
Flag_enableSys	unsigned char	0 '\x00'	0x00B580@Program
Flag_Run_Identify	unsigned char	0 '\x00'	0x00B581@Program
Flag_MotorIdentified	unsigned char	0 '\x00'	0x00B582@Program
Flag_enableForceAngle	unsigned char	1 '\x01'	0x00B583@Program
Flag_enableFieldWeakening	unsigned char	0 '\x00'	0x00B584@Program
Flag_enableRsRecalc	unsigned char	0 '\x00'	0x00B585@Program
Flag_enableUserParams	unsigned char	1 '\x01'	0x00B586@Program
Flag_enableOffsetcalc	unsigned char	1 '\x01'	0x00B587@Program
Flag_enablePowerWarp	unsigned char	0 '\x00'	0x00B588@Program
Flag_enableSpeedCtrl	unsigned char	0 '\x00'	0x00B589@Program
Flag_enableRun	unsigned char	0 '\x00'	0x00B58A@Program
Flag_RunState	unsigned char	0 '\x00'	0x00B58B@Program
Flag_enableFlyingStart	unsigned char	0 '\x00'	0x00B58C@Program
CtrlState	enum <unnamed>	CTRL_State_Idle	0x00B58D@Program
EstState	enum <unnamed>	EST_State_Idle	0x00B58E@Program
UserErrorCode	enum <unnamed>	USER_ErrorCode_NoError	0x00B58F@Program
CtrlVersion	struct _CTRL_Version_	{rsvd=0,targetProc=CTRL_TargetProc_U...	0x00B590@Program
IdRef_A	long	0	0x00B594@Program
IqRef_A	long	0	0x00B596@Program
SpeedRef_pu	long	0	0x00B598@Program
SpeedRef_krpm	long	1677721	0x00B59A@Program
SpeedTraj_krpm	long	0	0x00B59C@Program
MaxAccel_krpmps	long	3355443	0x00B59E@Program

```c
_iq gTorque_Flux_Iq_pu_to_Nm_sf;

// **************************************************************************
// the functions

void main(void)
{
  uint_least8_t estNumber = 0;

#ifdef FAST_ROM_V1p6
  uint_least8_t ctrlNumber = 0;
#endif

  // Only used if running from FLASH
```

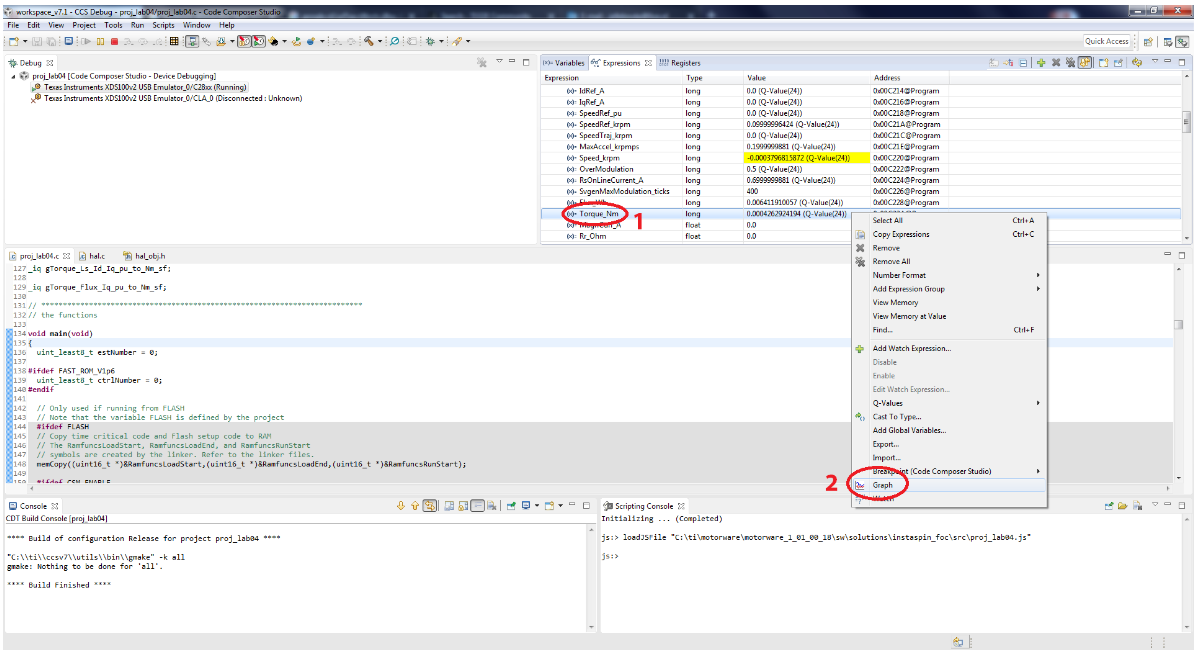
```
CDT Build Console [proj_lab04]
"C:/ti/motorware/motorware_1_01_00_18/sw/modules/iqmath/lib/f28x/32b/IQmath.lib" "./CodeStartBranch.obj" "./adc.obj"
"./clarke.obj" "./clk.obj" "./cpu.obj" "./ctrl.obj" "./drv8301.obj" "./filter_fo.obj" "./flash.obj" "./gpio.obj"
"./hal.obj" "./ipark.obj" "./offset.obj" "./osc.obj" "./park.obj" "./pid.obj" "./pie.obj" "./pll.obj"
"./proj_lab04.obj" "./pwm.obj" "./pwr.obj" "./spi.obj" "./svgen.obj" "./timer.obj" "./traj.obj" "./usDelay.obj"
"./user.obj" "./wdog.obj" "C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram_lnk.cmd"
<Linking>
'Finished building target: proj_lab04.out'
```

```
Scripting Console
Initializing .. (Completed)

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab04.js"
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of
null (C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:>
```

Figure A.75: Here the torque load can be changed.

4. Make sure the motor is identified and the values are stored in `user_j1.h`. If this is not the case, go back to appendix A.6.1.

5. When the motor is identified, right click on Torque_Nm in the expressions window and click on graph. This opens the graph window to see the torque.



Expression	Type	Value	Address
IdRef_A	long	0.0 (Q-Value(24))	0x00C214@Program
IqRef_A	long	0.0 (Q-Value(24))	0x00C216@Program
SpeedRef_pu	long	0.0 (Q-Value(24))	0x00C218@Program
SpeedRef_krpm	long	0.09999996424 (Q-Value(24))	0x00C21A@Program
SpeedTraj_krpm	long	0.0 (Q-Value(24))	0x00C21C@Program
MaxAccel_krpmps	long	0.1999999881 (Q-Value(24))	0x00C21E@Program
Speed_krpm	long	-0.0003796815872 (Q-Value(24))	0x00C220@Program
OverModulation	long	0.5 (Q-Value(24))	0x00C222@Program
RsOnLineCurrent_A	long	0.6999999881 (Q-Value(24))	0x00C224@Program
SvgenMaxModulation_ticks	long	400	0x00C226@Program
Flux_Wb	long	0.006411910057 (Q-Value(24))	0x00C228@Program
Torque_Nm	long	0.0004262924194 (Q-Value(24))	
MagnCurr_A	float	0.0	
Rr_Ohm	float	0.0	

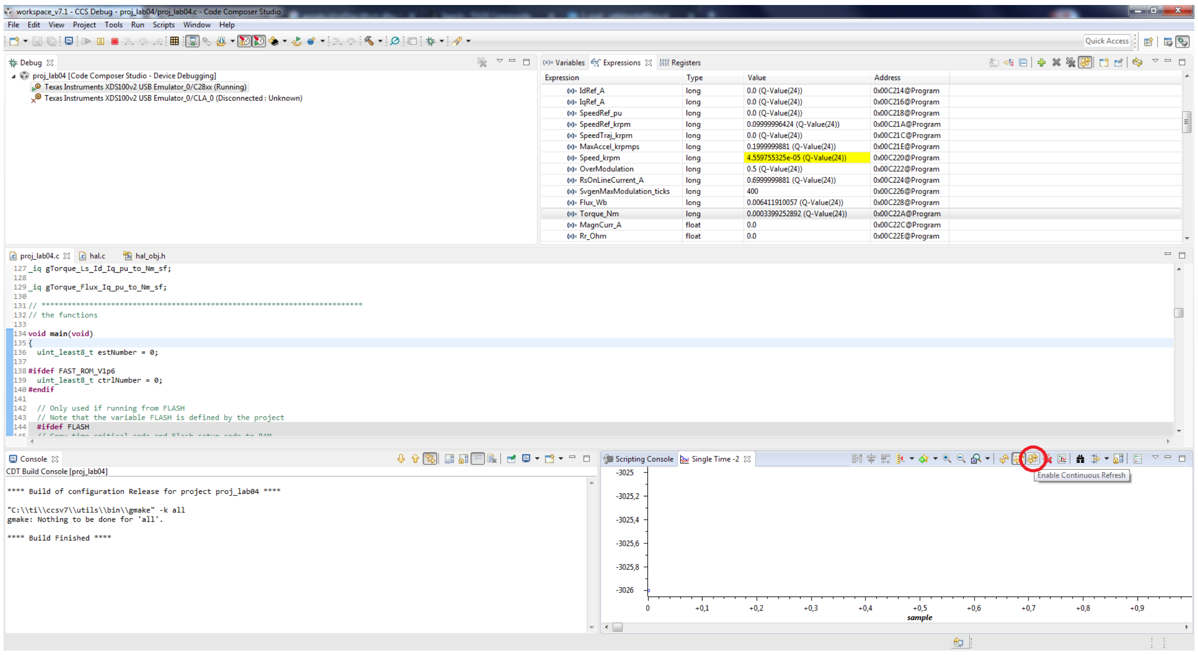
```c
_iq gTorque_Ls_Id_Iq_pu_to_Nm_sf;

_iq gTorque_Flux_Iq_pu_to_Nm_sf;

// **************************************************************************
// the functions

void main(void)
{
  uint_least8_t estNumber = 0;

#ifdef FAST_ROM_V1p6
  uint_least8_t ctrlNumber = 0;
#endif

  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
  // Copy time critical code and Flash setup code to RAM
  // The RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
  // symbols are created by the linker. Refer to the linker files.
  memCopy((uint16_t *)&RamfuncsLoadStart,(uint16_t *)&RamfuncsLoadEnd,(uint16_t *)&RamfuncsRunStart);
```

```
CDT Build Console [proj_lab04]

**** Build of configuration Release for project proj_lab04 ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
gmake: Nothing to be done for 'all'.

**** Build Finished ****
```

```
Scripting Console
Initializing ... (Completed)

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab04.js"

js:>
```

Figure A.76: Click here to show Torque_Nm in a graph.

6. Enable continuous refreshing in the graph window.


workspace_v7.1 - CCS Debug - proj_lab04/proj_lab04.c - Code Composer Studio

proj_lab04 [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v2 USB Emulator_0/C28xx (Running)
Texas Instruments XDS100v2 USB Emulator_0/CLA_0 (Disconnected : Unknown)

Expression	Type	Value	Address
IdRef_A	long	0.0 (Q-Value(24))	0x00C214@Program
IqRef_A	long	0.0 (Q-Value(24))	0x00C216@Program
SpeedRef_pu	long	0.0 (Q-Value(24))	0x00C218@Program
SpeedRef_krpm	long	0.09999996424 (Q-Value(24))	0x00C21A@Program
SpeedTraj_krpm	long	0.0 (Q-Value(24))	0x00C21C@Program
MaxAccel_krpmps	long	0.1999999881 (Q-Value(24))	0x00C21E@Program
Speed_krpm	long	4.559755325e-05 (Q-Value(24))	0x00C220@Program
OverModulation	long	0.5 (Q-Value(24))	0x00C222@Program
RsOnLineCurrent_A	long	0.6999999881 (Q-Value(24))	0x00C224@Program
SvgenMaxModulation_ticks	long	400	0x00C226@Program
Flux_Wb	long	0.006411910057 (Q-Value(24))	0x00C228@Program
Torque_Nm	long	0.0003399252892 (Q-Value(24))	0x00C22A@Program
MagnCurr_A	float	0.0	0x00C22C@Program
Rr_Ohm	float	0.0	0x00C22E@Program

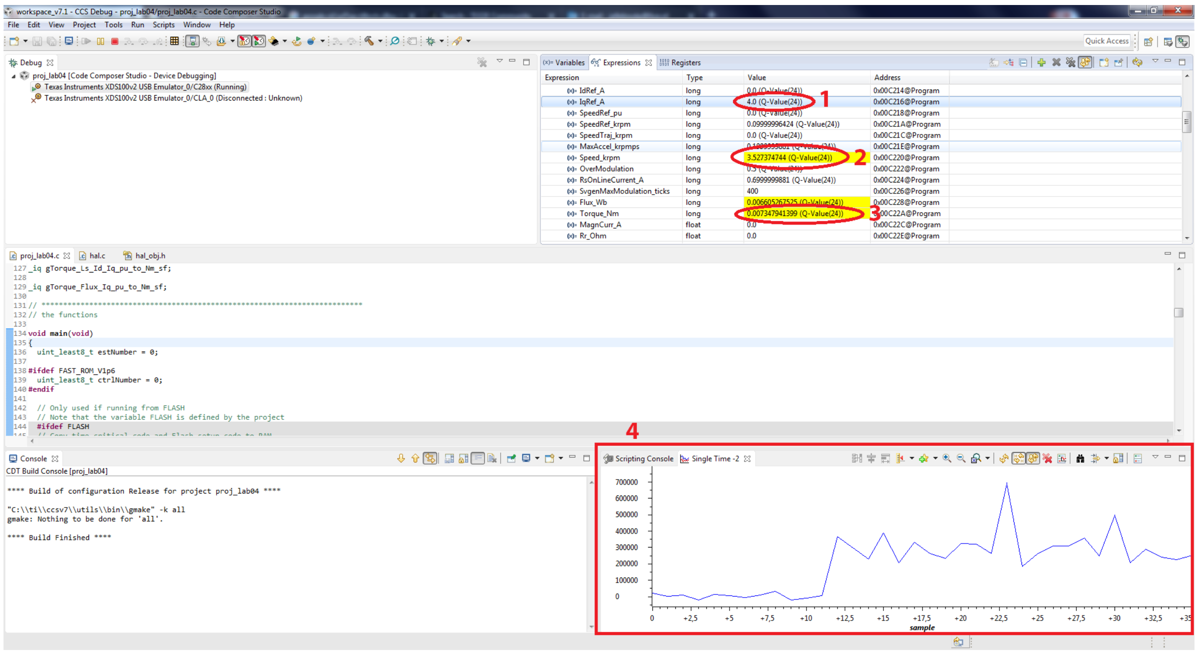
```
_iq gTorque_Ls_Id_Iq_pu_to_Nm_sf;

_iq gTorque_Flux_Iq_pu_to_Nm_sf;

// **************************************************************************
// the functions

void main(void)
{
  uint_least8_t estNumber = 0;

#ifdef FAST_ROM_V1p6
  uint_least8_t ctrlNumber = 0;
#endif

  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
```

CDT Build Console [proj_lab04]

```
**** Build of configuration Release for project proj_lab04 ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
gmake: Nothing to be done for 'all'.

**** Build Finished ****
```

Figure A.77: Enable continuous refresh

7. When changing IqRef_A (1) to a different value, the speed change can be seen (2), Torque_Nm will change (3) and this change can be seen in the graph window (4)


workspace_v7.1 - CCS Debug - proj_lab04/proj_lab04.c - Code Composer Studio

proj_lab04 [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v2 USB Emulator_0/C28xx (Running)
Texas Instruments XDS100v2 USB Emulator_0/CLA_0 (Disconnected : Unknown)

Expression	Type	Value	Address
IdRef_A	long	0.0 (Q-Value(24))	0x00C214@Program
IqRef_A	long	4.0 (Q-Value(24))	0x00C216@Program
SpeedRef_pu	long	0.0 (Q-Value(24))	0x00C218@Program
SpeedRef_krpm	long	0.09999996424 (Q-Value(24))	0x00C21A@Program
SpeedTraj_krpm	long	0.0 (Q-Value(24))	0x00C21C@Program
MaxAccel_krpmps	long	0.1999999881 (Q-Value(24))	0x00C21E@Program
Speed_krpm	long	3.527374744 (Q-Value(24))	0x00C220@Program
OverModulation	long	0.5 (Q-Value(24))	0x00C222@Program
RsOnLineCurrent_A	long	0.6999999881 (Q-Value(24))	0x00C224@Program
SvgenMaxModulation_ticks	long	400	0x00C226@Program
Flux_Wb	long	0.006605267525 (Q-Value(24))	0x00C228@Program
Torque_Nm	long	0.007347941399 (Q-Value(24))	0x00C22A@Program
MagnCurr_A	float	0.0	0x00C22C@Program
Rr_Ohm	float	0.0	0x00C22E@Program

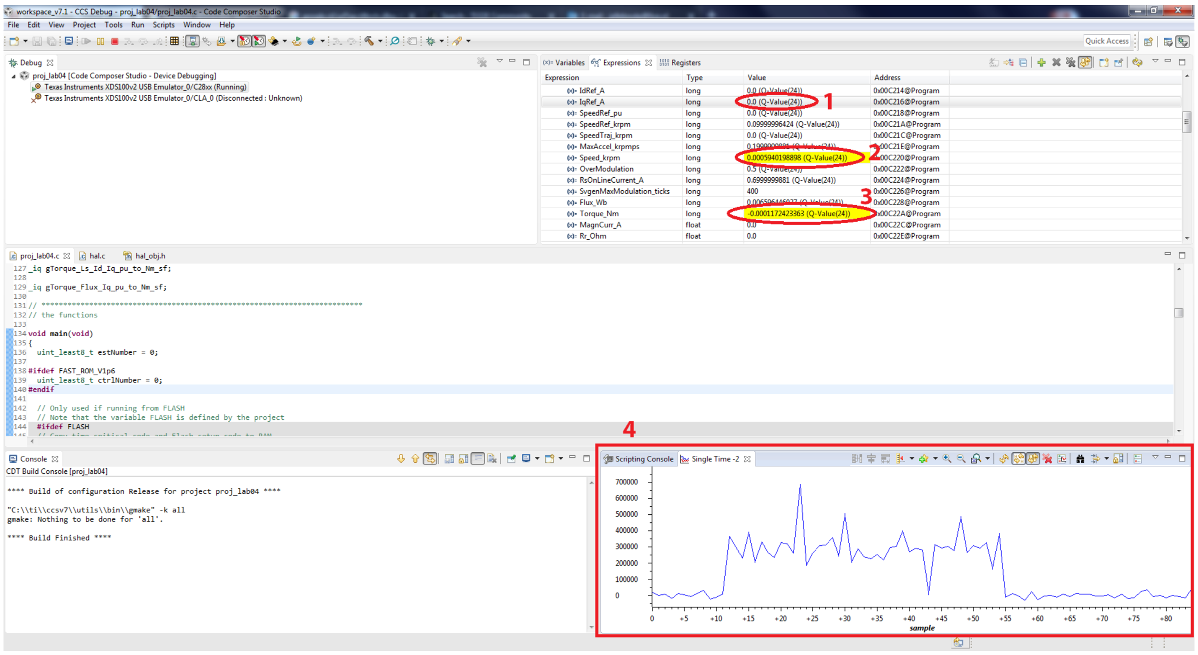
```
_iq gTorque_Ls_Id_Iq_pu_to_Nm_sf;

_iq gTorque_Flux_Iq_pu_to_Nm_sf;

// **************************************************************************
// the functions

void main(void)
{
  uint_least8_t estNumber = 0;

#ifdef FAST_ROM_V1p6
  uint_least8_t ctrlNumber = 0;
#endif

  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
```

CDT Build Console [proj_lab04]

```
**** Build of configuration Release for project proj_lab04 ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
gmake: Nothing to be done for 'all'.

**** Build Finished ****
```

Figure A.78: Changing IqRef_A (done here at sample +11) will influence the torque and speed of the motor. The change in torque is seen in the graph window.

8. Changing the IqRef_A back to zero, the motor will stop spinning as the torque will also become zero again.


workspace_v7.1 - CCS Debug - proj_lab04/proj_lab04.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
proj_lab04 [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v2 USB Emulator_0/C28xx (Running)
Texas Instruments XDS100v2 USB Emulator_0/CLA_0 (Disconnected : Unknown)
Variables
Expressions
Registers
1
2
3
4
Scripting Console
Single Time - 2
700000
600000
500000
400000
300000
200000
100000
0
0
+5
+10
+15
+20
+25
+30
+35
+40
+45
+50
+55
+60
+65
+70
+75
+80
sample

Expression	Type	Value	Address
IdRef_A	long	0.0 (Q-Value(24))	0x00C214@Program
IqRef_A	long	0.0 (Q-Value(24))	0x00C216@Program
SpeedRef_pu	long	0.0 (Q-Value(24))	0x00C218@Program
SpeedRef_krpm	long	0.09999996424 (Q-Value(24))	0x00C21A@Program
SpeedTraj_krpm	long	0.0 (Q-Value(24))	0x00C21C@Program
MaxAccel_krpmps	long	0.1999999881 (Q-Value(24))	0x00C21E@Program
Speed_krpm	long	0.0005940198898 (Q-Value(24))	0x00C220@Program
OverModulation	long	0.5 (Q-Value(24))	0x00C222@Program
RsOnLineCurrent_A	long	0.6999999881 (Q-Value(24))	0x00C224@Program
SvgenMaxModulation_ticks	long	400	0x00C226@Program
Flux_Wb	long	0.006596446027 (Q-Value(24))	0x00C228@Program
Torque_Nm	long	-0.0001172423363 (Q-Value(24))	0x00C22A@Program
MagnCurr_A	float	0.0	0x00C22C@Program
Rr_Ohm	float	0.0	0x00C22E@Program

proj_lab04.c hal.c hal_obj.h

```
_iq gTorque_Ls_Id_Iq_pu_to_Nm_sf;

_iq gTorque_Flux_Iq_pu_to_Nm_sf;

// **************************************************************************
// the functions

void main(void)
{
  uint_least8_t estNumber = 0;

#ifdef FAST_ROM_V1p6
  uint_least8_t ctrlNumber = 0;
#endif

  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
```

Console
CDT Build Console [proj_lab04]

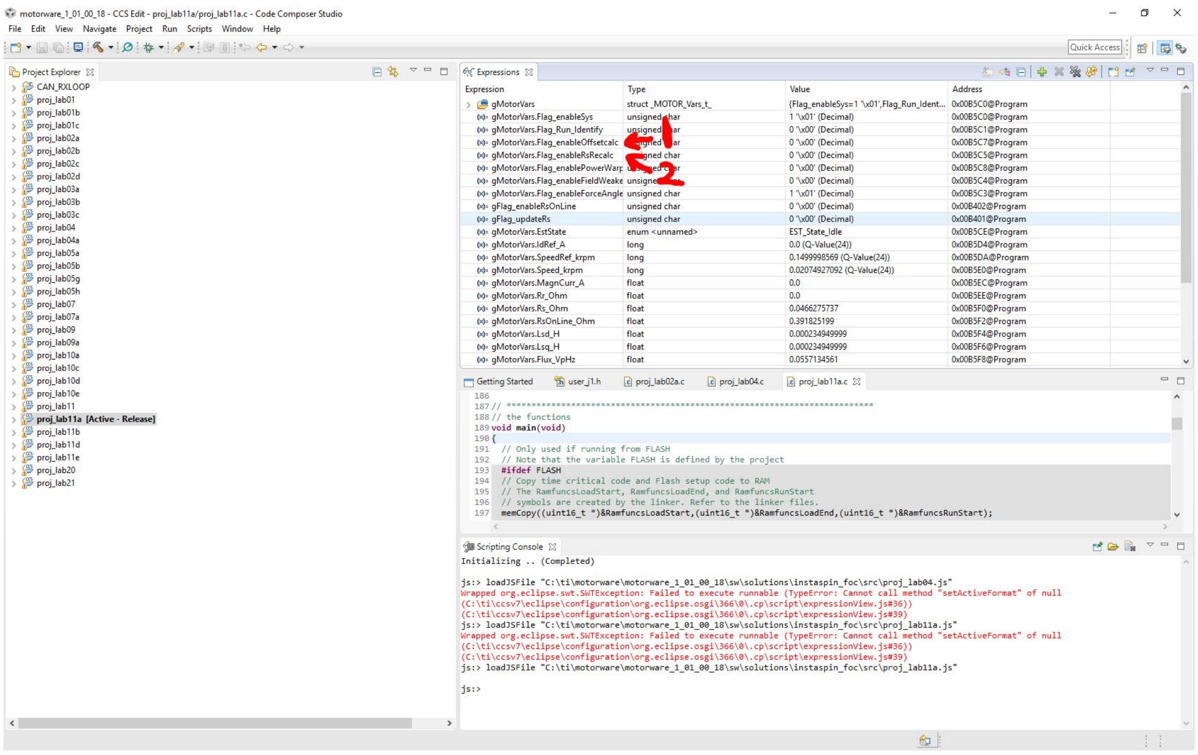
```
**** Build of configuration Release for project proj_lab04 ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
gmake: Nothing to be done for 'all'.

**** Build Finished ****
```

Figure A.79: IqRef_A is set back to zero at sample +54

A.6.3. Using lab 11a for the motor controller

1. Running lab 11a works the same as lab 2 and 4, with again selecting the appropriate project (`proj_lab11a`) and the appropriate JavaScript file (`proj_11a.js`).

2. Before running the motor, first set the enableOffsetcalc flag to 1. This will recalculate the voltage and current offsets for the measurements. After that is done, set the enableRsRecalc flag to one. This will update the stator resistance value for a more accurate control.



Expression	Type	Value	Address
gMotorVars	struct _MOTOR_Vars_t_	{Flag_enableSys=1 '\x01',Flag_Run_Ident...	0x00B5C0@Program
gMotorVars.Flag_enableSys	unsigned char	1 '\x01' (Decimal)	0x00B5C0@Program
gMotorVars.Flag_Run_Identify	unsigned char	0 '\x00' (Decimal)	0x00B5C1@Program
gMotorVars.Flag_enableOffsetcalc	unsigned char	0 '\x00' (Decimal)	0x00B5C7@Program
gMotorVars.Flag_enableRsRecalc	unsigned char	0 '\x00' (Decimal)	0x00B5C5@Program
gMotorVars.Flag_enablePowerWarp	unsigned char	0 '\x00' (Decimal)	0x00B5C8@Program
gMotorVars.Flag_enableFieldWeake	unsigned char	0 '\x00' (Decimal)	0x00B5C4@Program
gMotorVars.Flag_enableForceAngle	unsigned char	1 '\x01' (Decimal)	0x00B5C3@Program
gFlag_enableRsOnLine	unsigned char	0 '\x00' (Decimal)	0x00B402@Program
gFlag_updateRs	unsigned char	0 '\x00' (Decimal)	0x00B401@Program
gMotorVars.EstState	enum <unnamed>	EST_State_Idle	0x00B5CE@Program
gMotorVars.IdRef_A	long	0.0 (Q-Value(24))	0x00B5D4@Program
gMotorVars.SpeedRef_krpm	long	0.1499998569 (Q-Value(24))	0x00B5DA@Program
gMotorVars.Speed_krpm	long	0.02074927092 (Q-Value(24))	0x00B5E0@Program
gMotorVars.MagnCurr_A	float	0.0	0x00B5EC@Program
gMotorVars.Rr_Ohm	float	0.0	0x00B5EE@Program
gMotorVars.Rs_Ohm	float	0.0466275737	0x00B5F0@Program
gMotorVars.RsOnLine_Ohm	float	0.391825199	0x00B5F2@Program
gMotorVars.Lsd_H	float	0.000234949999	0x00B5F4@Program
gMotorVars.Lsq_H	float	0.000234949999	0x00B5F6@Program
gMotorVars.Flux_VpHz	float	0.0557134561	0x00B5F8@Program

Getting Started user_j1.h proj_lab02a.c proj_lab04.c proj_lab11a.c

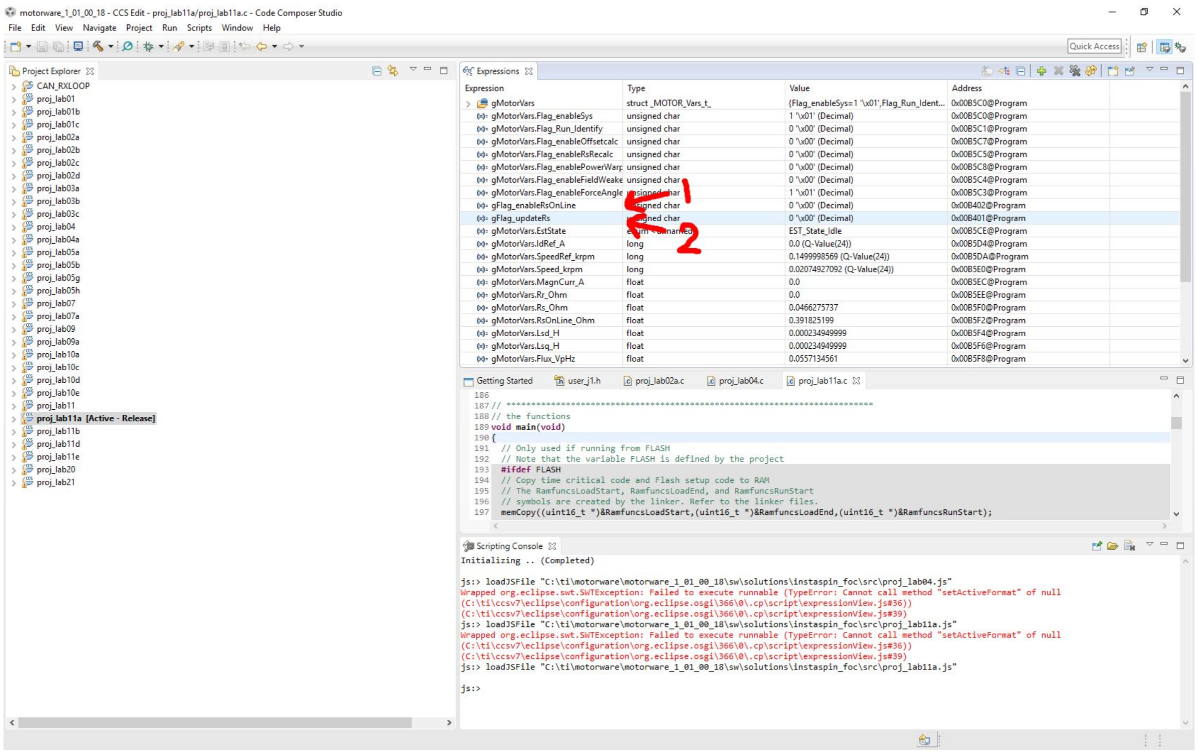
```

// **************************************************************************
// the functions
void main(void)
{
  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
  // Copy time critical code and Flash setup code to RAM
  // The RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
  // symbols are created by the linker. Refer to the linker files.
  memCopy((uint16_t *)&RamfuncsLoadStart,(uint16_t *)&RamfuncsLoadEnd,(uint16_t *)&RamfuncsRunStart);
```

Scripting Console

```
Initializing .. (Completed)

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab04.js"
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab11a.js"
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab11a.js"

js:>
```

Figure A.80: Step 1

3. While running the motor, the enableRsOnLine flag can be set to 1 to monitor the value of Rs. This is useful for calculating the temperature of the motor. With the updateRs flag, the most recent value of Rs will constantly be updated in the estimator, so the control will be more accurate.



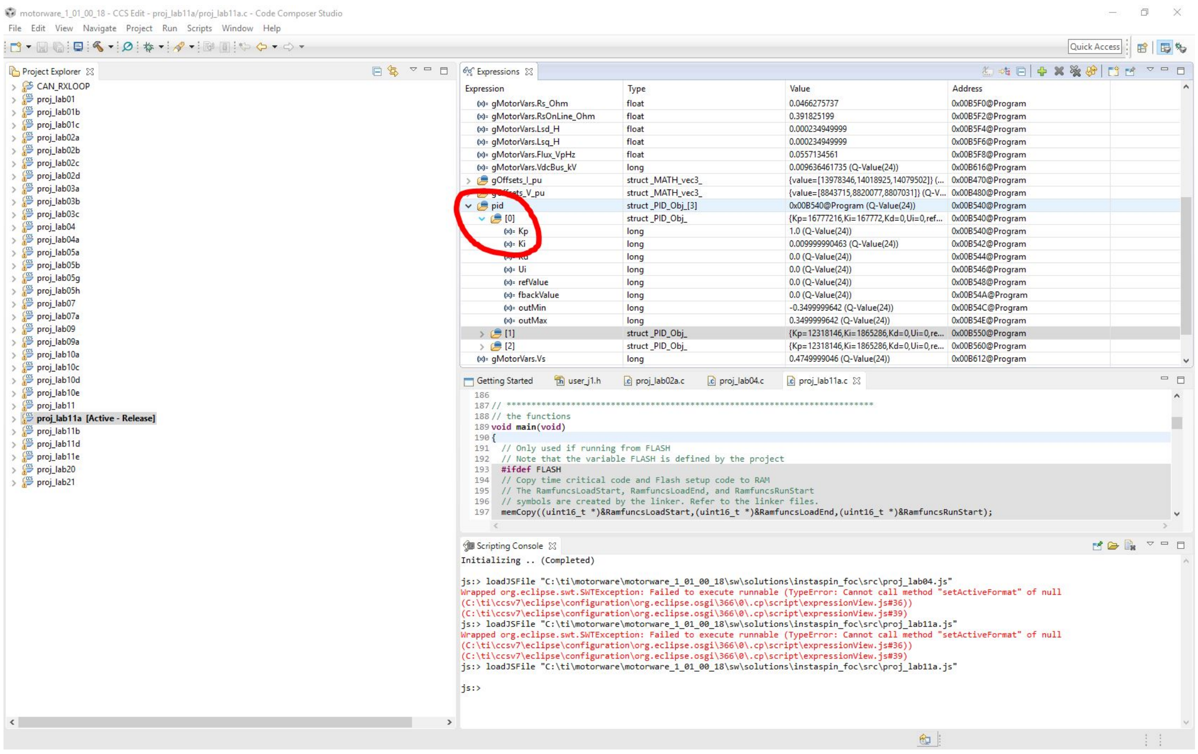
Expression	Type	Value	Address
gMotorVars	struct _MOTOR_Vars_t_	{Flag_enableSys=1 '\x01',Flag_Run_Ident...	0x00B5C0@Program
gMotorVars.Flag_enableSys	unsigned char	1 '\x01' (Decimal)	0x00B5C0@Program
gMotorVars.Flag_Run_Identify	unsigned char	0 '\x00' (Decimal)	0x00B5C1@Program
gMotorVars.Flag_enableOffsetcalc	unsigned char	0 '\x00' (Decimal)	0x00B5C7@Program
gMotorVars.Flag_enableRsRecalc	unsigned char	0 '\x00' (Decimal)	0x00B5C5@Program
gMotorVars.Flag_enablePowerWarp	unsigned char	0 '\x00' (Decimal)	0x00B5C8@Program
gMotorVars.Flag_enableFieldWeake	unsigned char	0 '\x00' (Decimal)	0x00B5C4@Program
gMotorVars.Flag_enableForceAngle	unsigned char	1 '\x01' (Decimal)	0x00B5C3@Program
gFlag_enableRsOnLine	unsigned char	0 '\x00' (Decimal)	0x00B402@Program
gFlag_updateRs	unsigned char	0 '\x00' (Decimal)	0x00B401@Program
gMotorVars.EstState	enum <unnamed>	EST_State_Idle	0x00B5CE@Program
gMotorVars.IdRef_A	long	0.0 (Q-Value(24))	0x00B5D4@Program
gMotorVars.SpeedRef_krpm	long	0.1499998569 (Q-Value(24))	0x00B5DA@Program
gMotorVars.Speed_krpm	long	0.02074927092 (Q-Value(24))	0x00B5E0@Program
gMotorVars.MagnCurr_A	float	0.0	0x00B5EC@Program
gMotorVars.Rr_Ohm	float	0.0	0x00B5EE@Program
gMotorVars.Rs_Ohm	float	0.0466275737	0x00B5F0@Program
gMotorVars.RsOnLine_Ohm	float	0.391825199	0x00B5F2@Program
gMotorVars.Lsd_H	float	0.000234949999	0x00B5F4@Program
gMotorVars.Lsq_H	float	0.000234949999	0x00B5F6@Program
gMotorVars.Flux_VpHz	float	0.0557134561	0x00B5F8@Program

```

// ***************************************************************************
// the functions
void main(void)
{
  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
  // Copy time critical code and Flash setup code to RAM
  // The RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
  // symbols are created by the linker. Refer to the linker files.
  memCopy((uint16_t *)&RamfuncsLoadStart,(uint16_t *)&RamfuncsLoadEnd,(uint16_t *)&RamfuncsRunStart);
```

```
Initializing .. (Completed)

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab04.js"
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab11a.js"
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab11a.js"

js:>
```

Figure A.81: Enable the online recalculation of the stator resistance.

4. From the watch window, all PID controllers can also be tuned. The controllers are: the speed controller, the Iq controller and the Id controller. To tune the controllers change the Kp and Ki values. (for more information on tuning the controllers, see lab 5a and 5b in [14].


motorware_1_01_00_18 - CCS Edit - proj_lab11a/proj_lab11a.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
Project Explorer
Expressions
Getting Started
user_j1.h
proj_lab02a.c
proj_lab04.c
proj_lab11a.c
Scripting Console

```
CAN_RXLOOP
proj_lab01
proj_lab01b
proj_lab01c
proj_lab02a
proj_lab02b
proj_lab02c
proj_lab02d
proj_lab03a
proj_lab03b
proj_lab03c
proj_lab04
proj_lab04a
proj_lab05a
proj_lab05b
proj_lab05g
proj_lab05h
proj_lab07
proj_lab07a
proj_lab09
proj_lab09a
proj_lab10a
proj_lab10c
proj_lab10d
proj_lab10e
proj_lab11
proj_lab11a [Active - Release]
proj_lab11b
proj_lab11d
proj_lab11e
proj_lab20
proj_lab21
```

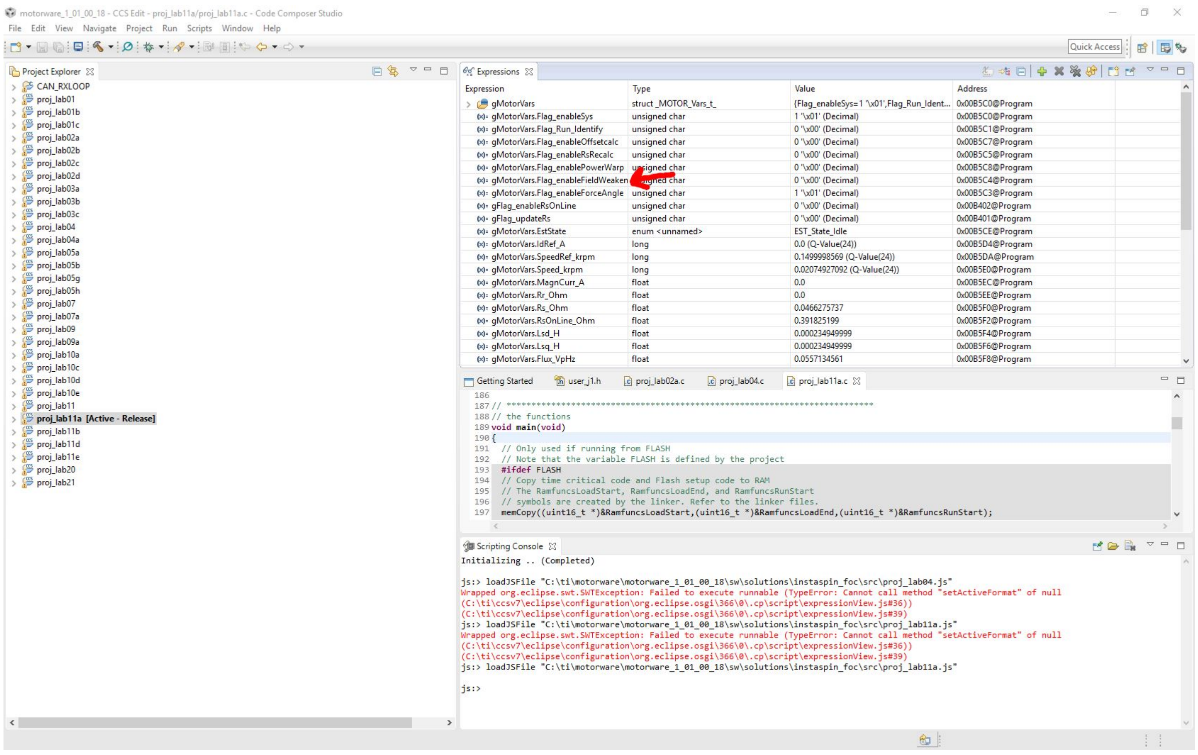
Expression	Type	Value	Address
gMotorVars.Rs_Ohm	float	0.0466275737	0x00B5F0@Program
gMotorVars.RsOnLine_Ohm	float	0.391825199	0x00B5F2@Program
gMotorVars.Lsd_H	float	0.000234949999	0x00B5F4@Program
gMotorVars.Lsq_H	float	0.000234949999	0x00B5F6@Program
gMotorVars.Flux_VpHz	float	0.0557134561	0x00B5F8@Program
gMotorVars.VdcBus_kV	long	0.009636461735 (Q-Value(24))	0x00B616@Program
gOffsets_I_pu	struct _MATH_vec3_	{value=[13978346,14018925,14079502]} (...	0x00B470@Program
gOffsets_V_pu	struct _MATH_vec3_	{value=[8843715,8820077,8807031]} (Q-V...	0x00B480@Program
pid	struct _PID_Obj_[3]	0x00B540@Program (Q-Value(24))	0x00B540@Program
[0]	struct _PID_Obj_	{Kp=16777216,Ki=167772,Kd=0,Ui=0,ref...	0x00B540@Program
Kp	long	1.0 (Q-Value(24))	0x00B540@Program
Ki	long	0.009999990463 (Q-Value(24))	0x00B542@Program
Kd	long	0.0 (Q-Value(24))	0x00B544@Program
Ui	long	0.0 (Q-Value(24))	0x00B546@Program
refValue	long	0.0 (Q-Value(24))	0x00B548@Program
fbackValue	long	0.0 (Q-Value(24))	0x00B54A@Program
outMin	long	-0.3499999642 (Q-Value(24))	0x00B54C@Program
outMax	long	0.3499999642 (Q-Value(24))	0x00B54E@Program
[1]	struct _PID_Obj_	{Kp=12318146,Ki=1865286,Kd=0,Ui=0,re...	0x00B550@Program
[2]	struct _PID_Obj_	{Kp=12318146,Ki=1865286,Kd=0,Ui=0,re...	0x00B560@Program
gMotorVars.Vs	long	0.4749999046 (Q-Value(24))	0x00B612@Program

```c

// **************************************************************************
// the functions
void main(void)
{
  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
  // Copy time critical code and Flash setup code to RAM
  // The RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
  // symbols are created by the linker. Refer to the linker files.
  memCopy((uint16_t *)&RamfuncsLoadStart,(uint16_t *)&RamfuncsLoadEnd,(uint16_t *)&RamfuncsRunStart);
```

```
Initializing .. (Completed)

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab04.js"
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab11a.js"
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab11a.js"

js:>
```

Figure A.82: The controllers can be independently tuned.

5. Field weakening can also be enabled in this lab. To do that, change the value of the enableField-Weakening flag to 1 in the watch window.


motorware_1_01_00_18 - CCS Edit - proj_lab11a/proj_lab11a.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
Project Explorer
Expressions
Getting Started
user_j1.h
proj_lab02a.c
proj_lab04.c
proj_lab11a.c
Scripting Console

```
CAN_RXLOOP
proj_lab01
proj_lab01b
proj_lab01c
proj_lab02a
proj_lab02b
proj_lab02c
proj_lab02d
proj_lab03a
proj_lab03b
proj_lab03c
proj_lab04
proj_lab04a
proj_lab05a
proj_lab05b
proj_lab05g
proj_lab05h
proj_lab07
proj_lab07a
proj_lab09
proj_lab09a
proj_lab10a
proj_lab10c
proj_lab10d
proj_lab10e
proj_lab11
proj_lab11a [Active - Release]
proj_lab11b
proj_lab11d
proj_lab11e
proj_lab20
proj_lab21
```

Expression	Type	Value	Address
gMotorVars	struct _MOTOR_Vars_t_	{Flag_enableSys=1 '\x01',Flag_Run_Ident...	0x00B5C0@Program
gMotorVars.Flag_enableSys	unsigned char	1 '\x01' (Decimal)	0x00B5C0@Program
gMotorVars.Flag_Run_Identify	unsigned char	0 '\x00' (Decimal)	0x00B5C1@Program
gMotorVars.Flag_enableOffsetcalc	unsigned char	0 '\x00' (Decimal)	0x00B5C7@Program
gMotorVars.Flag_enableRsRecalc	unsigned char	0 '\x00' (Decimal)	0x00B5C5@Program
gMotorVars.Flag_enablePowerWarp	unsigned char	0 '\x00' (Decimal)	0x00B5C8@Program
gMotorVars.Flag_enableFieldWeaken	unsigned char	0 '\x00' (Decimal)	0x00B5C4@Program
gMotorVars.Flag_enableForceAngle	unsigned char	1 '\x01' (Decimal)	0x00B5C3@Program
gFlag_enableRsOnLine	unsigned char	0 '\x00' (Decimal)	0x00B402@Program
gFlag_updateRs	unsigned char	0 '\x00' (Decimal)	0x00B401@Program
gMotorVars.EstState	enum <unnamed>	EST_State_Idle	0x00B5CE@Program
gMotorVars.IdRef_A	long	0.0 (Q-Value(24))	0x00B5D4@Program
gMotorVars.SpeedRef_krpm	long	0.1499998569 (Q-Value(24))	0x00B5DA@Program
gMotorVars.Speed_krpm	long	0.02074927092 (Q-Value(24))	0x00B5E0@Program
gMotorVars.MagnCurr_A	float	0.0	0x00B5EC@Program
gMotorVars.Rr_Ohm	float	0.0	0x00B5EE@Program
gMotorVars.Rs_Ohm	float	0.0466275737	0x00B5F0@Program
gMotorVars.RsOnLine_Ohm	float	0.391825199	0x00B5F2@Program
gMotorVars.Lsd_H	float	0.000234949999	0x00B5F4@Program
gMotorVars.Lsq_H	float	0.000234949999	0x00B5F6@Program
gMotorVars.Flux_VpHz	float	0.0557134561	0x00B5F8@Program

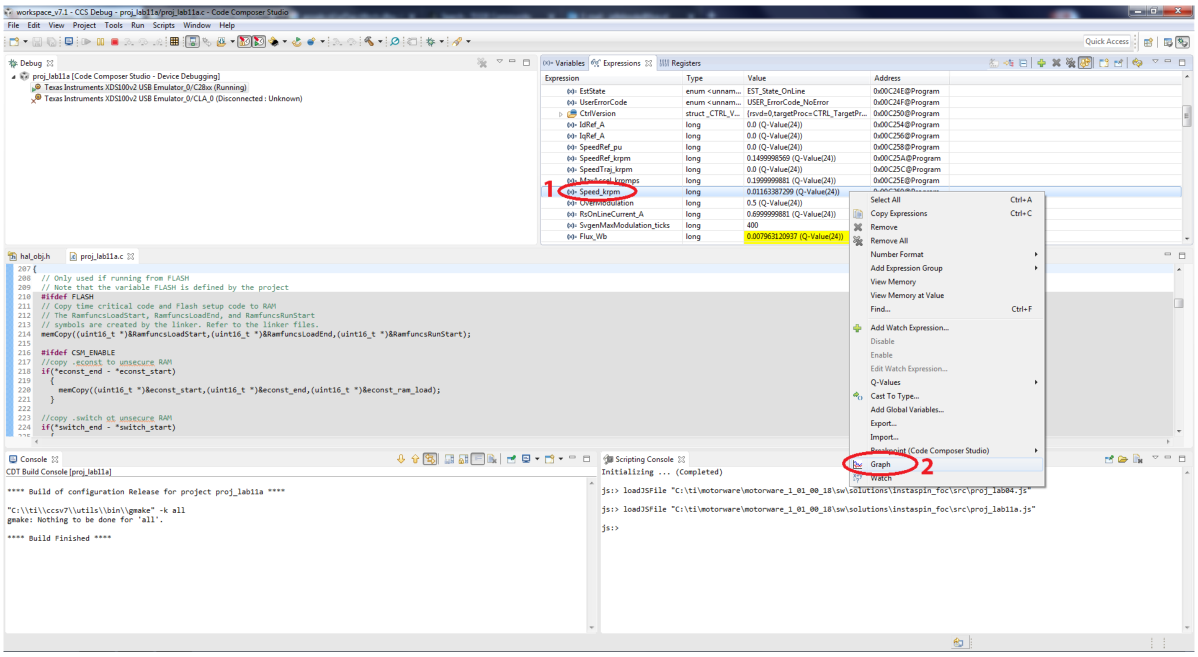
```c

// **************************************************************************
// the functions
void main(void)
{
  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
  // Copy time critical code and Flash setup code to RAM
  // The RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
  // symbols are created by the linker. Refer to the linker files.
  memCopy((uint16_t *)&RamfuncsLoadStart,(uint16_t *)&RamfuncsLoadEnd,(uint16_t *)&RamfuncsRunStart);
```

```
Initializing .. (Completed)

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab04.js"
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab11a.js"
Wrapped org.eclipse.swt.SWTException: Failed to execute runnable (TypeError: Cannot call method "setActiveFormat" of null
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#36))
(C:\ti\ccsv7\eclipse\configuration\org.eclipse.osgi\366\0\.cp\script\expressionView.js#39)
js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab11a.js"

js:>
```

Figure A.83: Field weakening can be enabled by setting this flag to 1.

6. To open the graph to observe the actual speed of the motor. Right-click on the variable (in this

case: Speed_krpm) and click on graph.



Expression	Type	Value	Address
EstState	enum <unnam...	EST_State_OnLine	0x00C24E@Program
UserErrorCode	enum <unnam...	USER_ErrorCode_NoError	0x00C24F@Program
CtrlVersion	struct _CTRL_V...	{rsvd=0,targetProc=CTRL_TargetPr...	0x00C250@Program
IdRef_A	long	0.0 (Q-Value(24))	0x00C254@Program
IqRef_A	long	0.0 (Q-Value(24))	0x00C256@Program
SpeedRef_pu	long	0.0 (Q-Value(24))	0x00C258@Program
SpeedRef_krpm	long	0.1499998569 (Q-Value(24))	0x00C25A@Program
SpeedTraj_krpm	long	0.0 (Q-Value(24))	0x00C25C@Program
MaxAccel_krpmps	long	0.1999999881 (Q-Value(24))	0x00C25E@Program
Speed_krpm	long	0.01163387299 (Q-Value(24))	
OverModulation	long	0.5 (Q-Value(24))	
RsOnLineCurrent_A	long	0.6999999881 (Q-Value(24))	
SvgenMaxModulation_ticks	long	400	
Flux_Wb	long	0.007963120937 (Q-Value(24))	

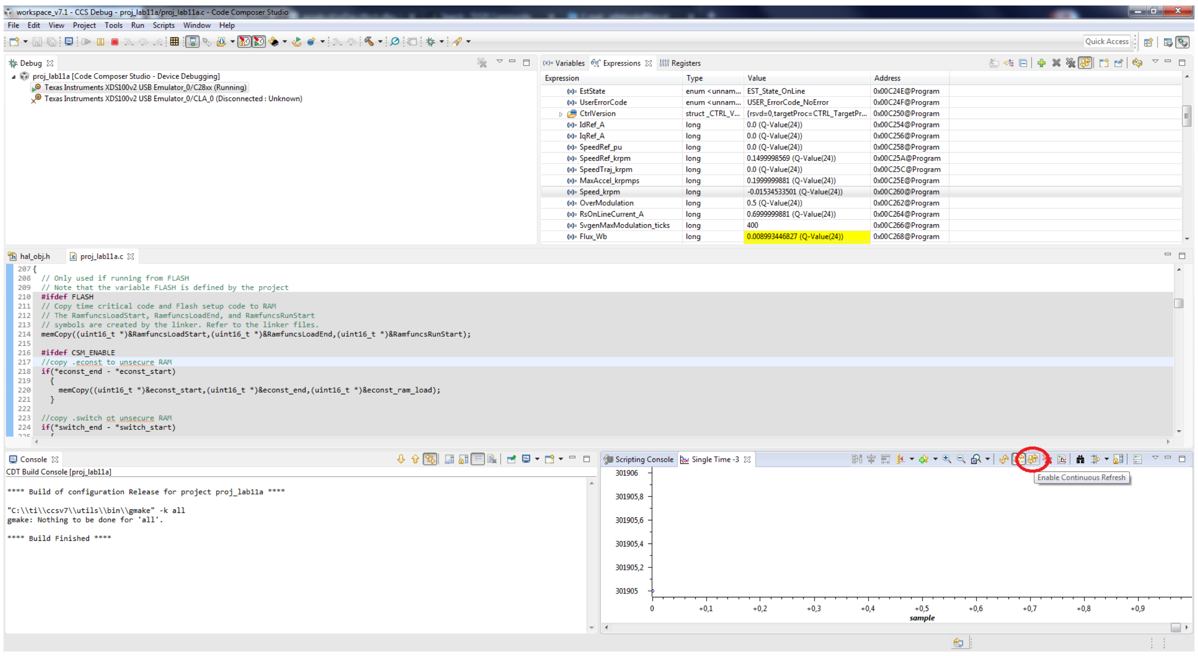
```
{
  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
  // Copy time critical code and Flash setup code to RAM
  // The RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
  // symbols are created by the linker. Refer to the linker files.
  memCopy((uint16_t *)&RamfuncsLoadStart,(uint16_t *)&RamfuncsLoadEnd,(uint16_t *)&RamfuncsRunStart);

  #ifdef CSM_ENABLE
  //copy .econst to unsecure RAM
  if(*econst_end - *econst_start)
    {
      memCopy((uint16_t *)&econst_start,(uint16_t *)&econst_end,(uint16_t *)&econst_ram_load);
    }

  //copy .switch ot unsecure RAM
  if(*switch_end - *switch_start)
```

```
CDT Build Console [proj_lab11a]

**** Build of configuration Release for project proj_lab11a ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
gmake: Nothing to be done for 'all'.

**** Build Finished ****
```

```
Initializing ... (Completed)

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab04.js"

js:> loadJSFile "C:\ti\motorware\motorware_1_01_00_18\sw\solutions\instaspin_foc\src\proj_lab11a.js"

js:>
```

Figure A.84: To open the graph of the actual speed, right click on speed_kprm and click on graph.

7. Enable the continuous refreshing, otherwise refreshing the graph needs to be done by hand.


workspace_v7.1 - CCS Debug - proj_lab11a/proj_lab11a.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
Quick Access
Debug
proj_lab11a [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v2 USB Emulator_0/C28xx (Running)
Texas Instruments XDS100v2 USB Emulator_0/CLA_0 (Disconnected : Unknown)
Variables
Expressions
Registers
hal_obj.h
proj_lab11a.c
Console
Scripting Console
Single Time - 3
Enable Continuous Refresh
301906
301905,8
301905,6
301905,4
301905,2
301905
0
+0,1
+0,2
+0,3
+0,4
+0,5
+0,6
+0,7
+0,8
+0,9
sample

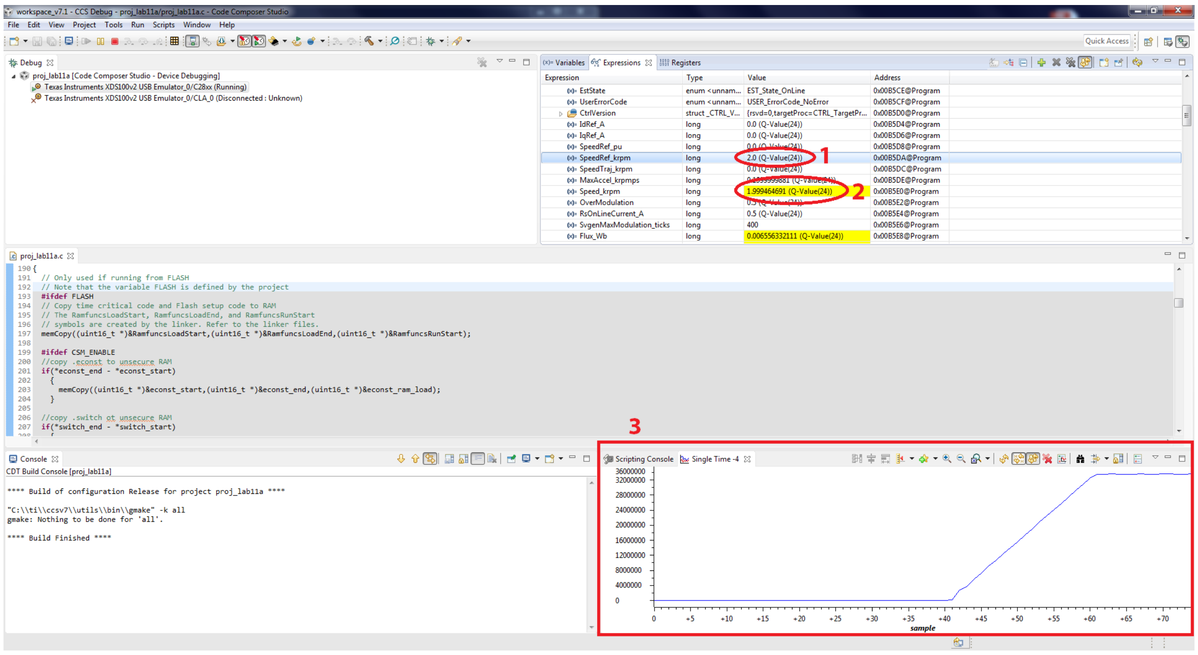
Expression	Type	Value	Address
EstState	enum <unnam...	EST_State_OnLine	0x00C24E@Program
UserErrorCode	enum <unnam...	USER_ErrorCode_NoError	0x00C24F@Program
CtrlVersion	struct _CTRL_V...	{rsvd=0,targetProc=CTRL_TargetPr...	0x00C250@Program
IdRef_A	long	0.0 (Q-Value(24))	0x00C254@Program
IqRef_A	long	0.0 (Q-Value(24))	0x00C256@Program
SpeedRef_pu	long	0.0 (Q-Value(24))	0x00C258@Program
SpeedRef_krpm	long	0.1499998569 (Q-Value(24))	0x00C25A@Program
SpeedTraj_krpm	long	0.0 (Q-Value(24))	0x00C25C@Program
MaxAccel_krpmps	long	0.1999999881 (Q-Value(24))	0x00C25E@Program
Speed_krpm	long	-0.01534533501 (Q-Value(24))	0x00C260@Program
OverModulation	long	0.5 (Q-Value(24))	0x00C262@Program
RsOnLineCurrent_A	long	0.6999999881 (Q-Value(24))	0x00C264@Program
SvgenMaxModulation_ticks	long	400	0x00C266@Program
Flux_Wb	long	0.008993446827 (Q-Value(24))	0x00C268@Program

```
{
  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
  // Copy time critical code and Flash setup code to RAM
  // The RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
  // symbols are created by the linker. Refer to the linker files.
  memCopy((uint16_t *)&RamfuncsLoadStart,(uint16_t *)&RamfuncsLoadEnd,(uint16_t *)&RamfuncsRunStart);

  #ifdef CSM_ENABLE
  //copy .econst to unsecure RAM
  if(*econst_end - *econst_start)
    {
      memCopy((uint16_t *)&econst_start,(uint16_t *)&econst_end,(uint16_t *)&econst_ram_load);
    }

  //copy .switch ot unsecure RAM
  if(*switch_end - *switch_start)
```

```
CDT Build Console [proj_lab11a]

**** Build of configuration Release for project proj_lab11a ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
gmake: Nothing to be done for 'all'.

**** Build Finished ****
```

Figure A.85: Enable continuous refreshing.

8. While changing the reference speed at SpeedRef_krpm. The actual speed will change by the maximum acceleration set. This can be observed in the graphs of fig. A.86 and fig. A.87.

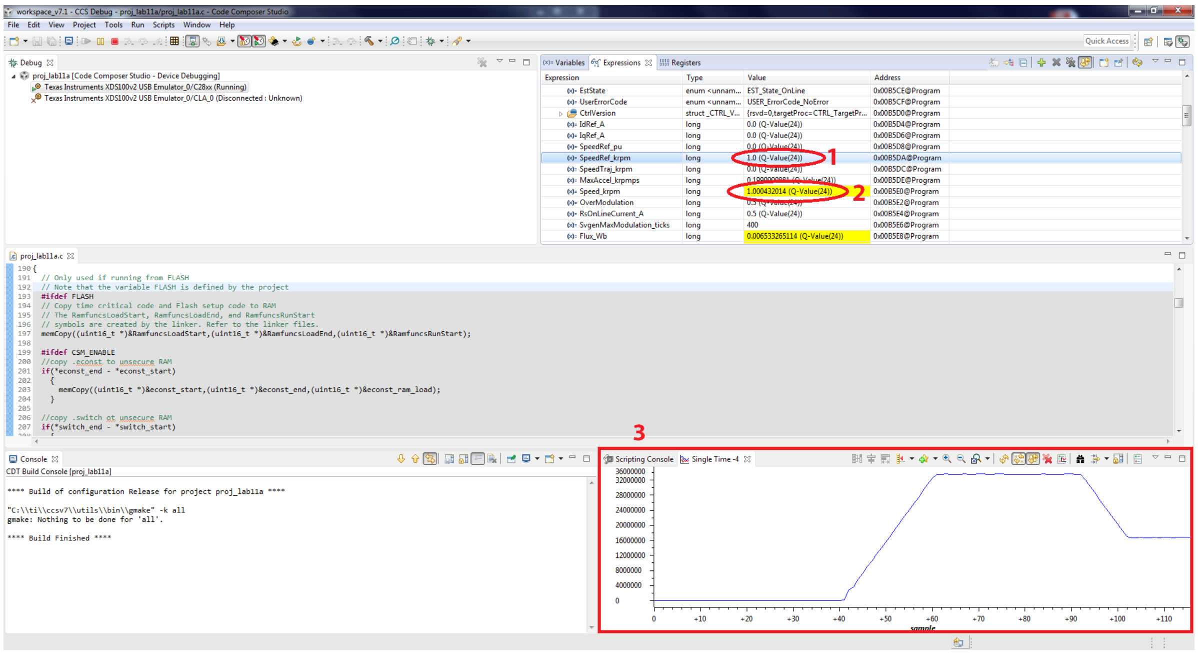


Expression	Type	Value	Address
EstState	enum <unnam...	EST_State_OnLine	0x00B5CE@Program
UserErrorCode	enum <unnam...	USER_ErrorCode_NoError	0x00B5CF@Program
CtrlVersion	struct _CTRL_V...	{rsvd=0,targetProc=CTRL_TargetPr...	0x00B5D0@Program
IdRef_A	long	0.0 (Q-Value(24))	0x00B5D4@Program
IqRef_A	long	0.0 (Q-Value(24))	0x00B5D6@Program
SpeedRef_pu	long	0.0 (Q-Value(24))	0x00B5D8@Program
SpeedRef_krpm	long	2.0 (Q-Value(24))	0x00B5DA@Program
SpeedTraj_krpm	long	0.0 (Q-Value(24))	0x00B5DC@Program
MaxAccel_krpmps	long	0.1999999881 (Q-Value(24))	0x00B5DE@Program
Speed_krpm	long	1.999464691 (Q-Value(24))	0x00B5E0@Program
OverModulation	long	0.5 (Q-Value(24))	0x00B5E2@Program
RsOnLineCurrent_A	long	0.5 (Q-Value(24))	0x00B5E4@Program
SvgenMaxModulation_ticks	long	400	0x00B5E6@Program
Flux_Wb	long	0.006556332111 (Q-Value(24))	0x00B5E8@Program

```
{
  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
  // Copy time critical code and Flash setup code to RAM
  // The RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
  // symbols are created by the linker. Refer to the linker files.
  memCopy((uint16_t *)&RamfuncsLoadStart,(uint16_t *)&RamfuncsLoadEnd,(uint16_t *)&RamfuncsRunStart);

  #ifdef CSM_ENABLE
  //copy .econst to unsecure RAM
  if(*econst_end - *econst_start)
    {
      memCopy((uint16_t *)&econst_start,(uint16_t *)&econst_end,(uint16_t *)&econst_ram_load);
    }

  //copy .switch ot unsecure RAM
  if(*switch_end - *switch_start)
```

```
CDT Build Console [proj_lab11a]

**** Build of configuration Release for project proj_lab11a ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
gmake: Nothing to be done for 'all'.

**** Build Finished ****
```

Figure A.86: 1. Change the speed, 2. The actual speed changes and 3. Graph shows how the speed changed.



Expression	Type	Value	Address
EstState	enum <unnam...	EST_State_OnLine	0x00B5CE@Program
UserErrorCode	enum <unnam...	USER_ErrorCode_NoError	0x00B5CF@Program
CtrlVersion	struct _CTRL_V...	{rsvd=0,targetProc=CTRL_TargetPr...	0x00B5D0@Program
IdRef_A	long	0.0 (Q-Value(24))	0x00B5D4@Program
IqRef_A	long	0.0 (Q-Value(24))	0x00B5D6@Program
SpeedRef_pu	long	0.0 (Q-Value(24))	0x00B5D8@Program
SpeedRef_krpm	long	1.0 (Q-Value(24))	0x00B5DA@Program
SpeedTraj_krpm	long	0.0 (Q-Value(24))	0x00B5DC@Program
MaxAccel_krpmps	long	0.1999999881 (Q-Value(24))	0x00B5DE@Program
Speed_krpm	long	1.000432014 (Q-Value(24))	0x00B5E0@Program
OverModulation	long	0.5 (Q-Value(24))	0x00B5E2@Program
RsOnLineCurrent_A	long	0.5 (Q-Value(24))	0x00B5E4@Program
SvgenMaxModulation_ticks	long	400	0x00B5E6@Program
Flux_Wb	long	0.006533265114 (Q-Value(24))	0x00B5E8@Program

Figure A.87: Another example like fig. A.86.

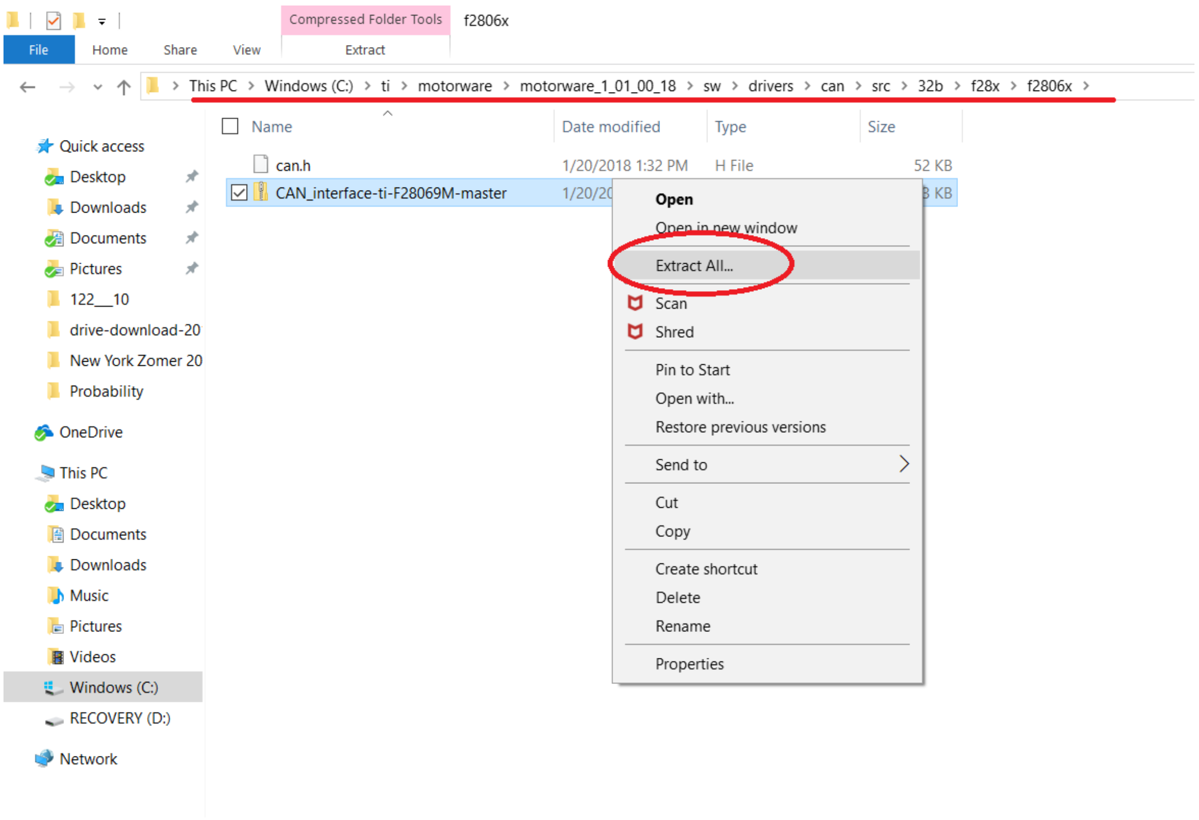
More information on the use of this lab can be found in [14].

A.7. Adding the eCAN module

For integrating the CAN module of the Launchpad first some files need to be downloaded from the archive. This is all done with help from the work of Andrew Buckin [26].

1. Download the zip file: "CAN_interface-ti-F28069M-master" from the archive.
2. Save it in the folder "`C:\ti\motorware\motorware_1_01_00_18\sw\solutions\`

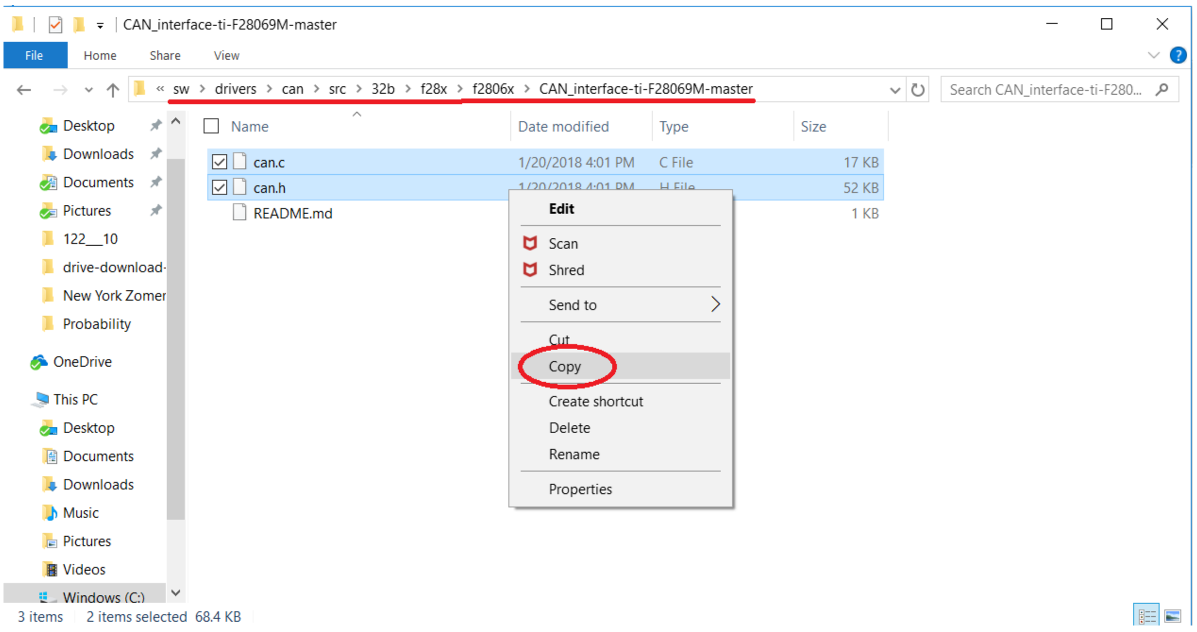
`instaspin_foc\boards\boostxldrv8301_revB\f28x\f2806xF\src”` And extract all the files.



Name	Date modified	Type	Size
can.h	1/20/2018 1:32 PM	H File	52 KB
CAN_interface-ti-F28069M-master	1/20/20		3 KB

Figure A.88: Step 2: Extract all the files.

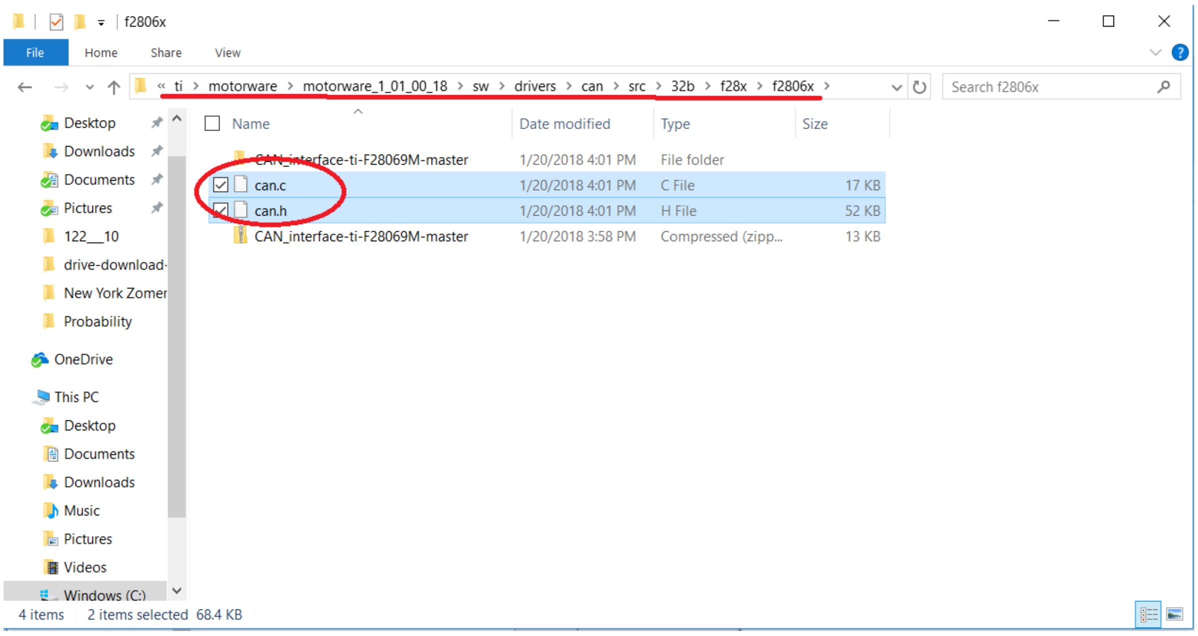
3. Copy the can.c and can.h file from `”C:\ti\motorware\motorware_1_01_00_18\sw\drivers\can\src\32b\f28x\f2806x\CAN_interface-ti-F28069M-master”`



Name	Date modified	Type	Size
can.c	1/20/2018 4:01 PM	C File	17 KB
can.h	1/20/2018 4:01 PM	H File	52 KB
README.md			1 KB

Figure A.89: Step 3: Copy the files from the underlined folder.

4. Paste the can.c and can.h files in ”`C:\ti\motorware\motorware_1_01_00_18\sw\drivers\can\src\32b\f28x\f2806x`”



Name	Date modified	Type	Size
CAN_interface-ti-F28069M-master	1/20/2018 4:01 PM	File folder	
can.c	1/20/2018 4:01 PM	C File	17 KB
can.h	1/20/2018 4:01 PM	H File	52 KB
CAN_interface-ti-F28069M-master	1/20/2018 3:58 PM	Compressed (zipp...	13 KB

Figure A.90: Step 4: Paste the files from the underlined folder.

5. In Code Composer Studio open a project according to the instructions of appendix A.5.3.

6. Right-click on the project where the CAN interface needs to be integrated and click ”Add Files...”

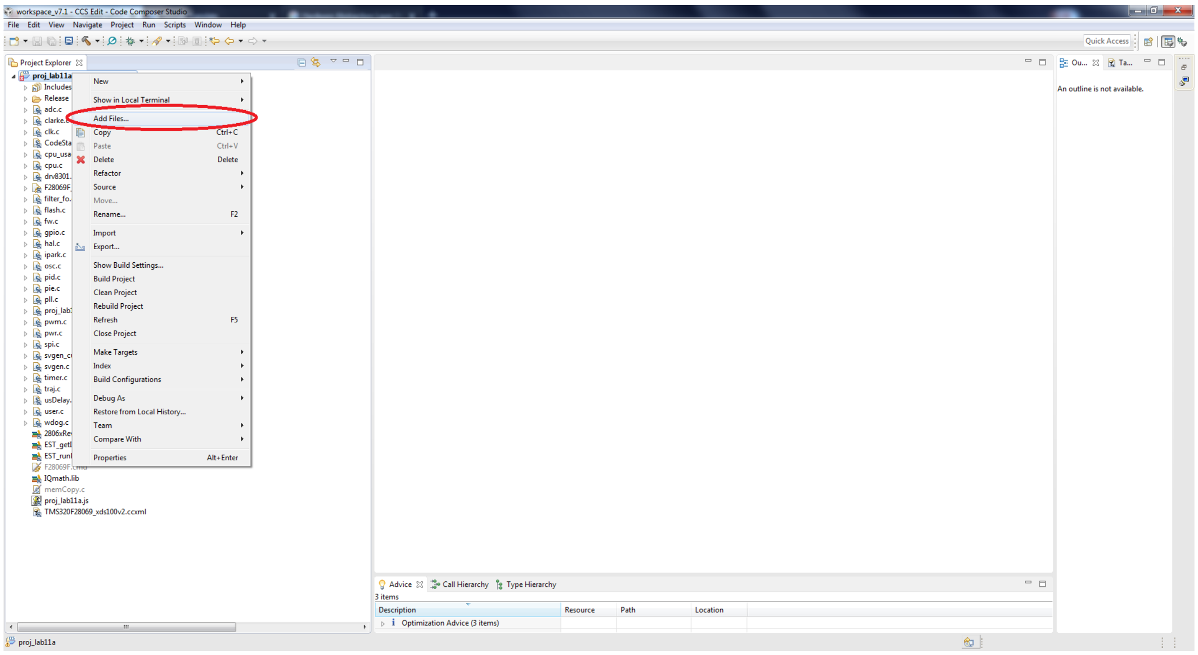


Figure A.91: Step 6: Right-click on the project and click on ”Add Files...”.

7. Go to ”`C:\ti\motorware\motorware_1_01_00_18\sw\drivers\can\src\32b\f28x\f2806x`” and select can.c.

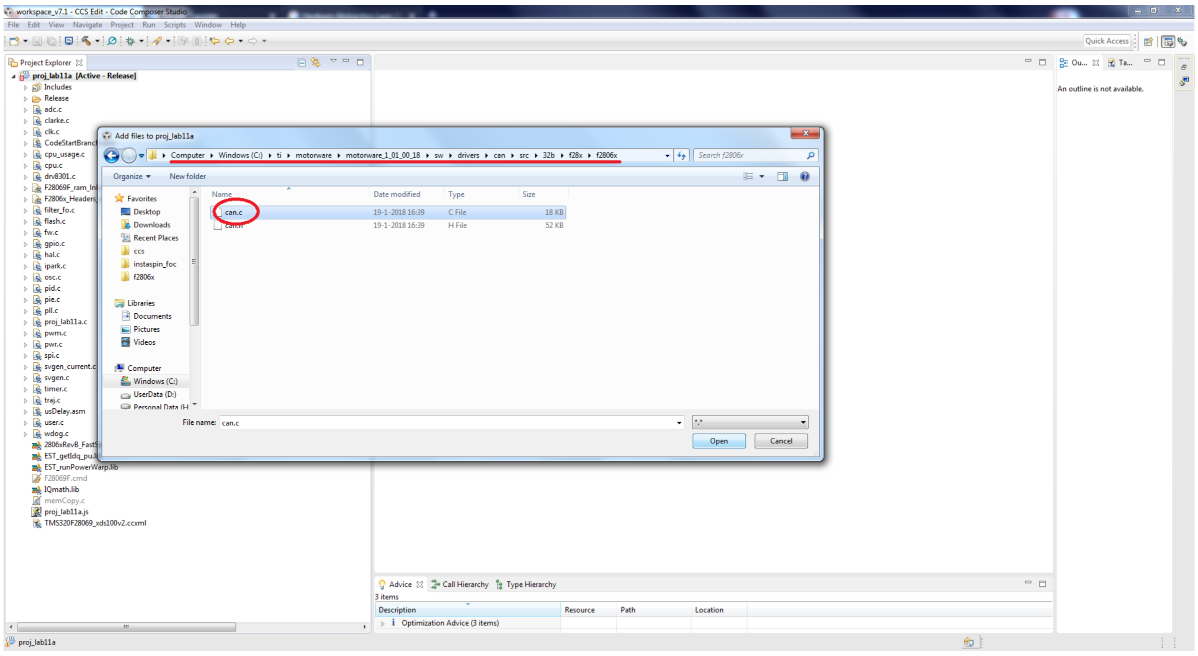

workspace_v7.1 - CCS Edit - Code Composer Studio
Add files to proj_lab11a
Computer ▸ Windows (C:) ▸ ti ▸ motorware ▸ motorware_1_01_00_18 ▸ sw ▸ drivers ▸ can ▸ src ▸ 32b ▸ f28x ▸ f2806x
can.c
can.h
File name: can.c
Open
Cancel

Figure A.92: Step 7: Go to the underlined folder and select can.c.

8. Select link files and set the ”Create link locations relative to:” drop-down menu to ”MW_INSTALL_DIR”. Than click OK.

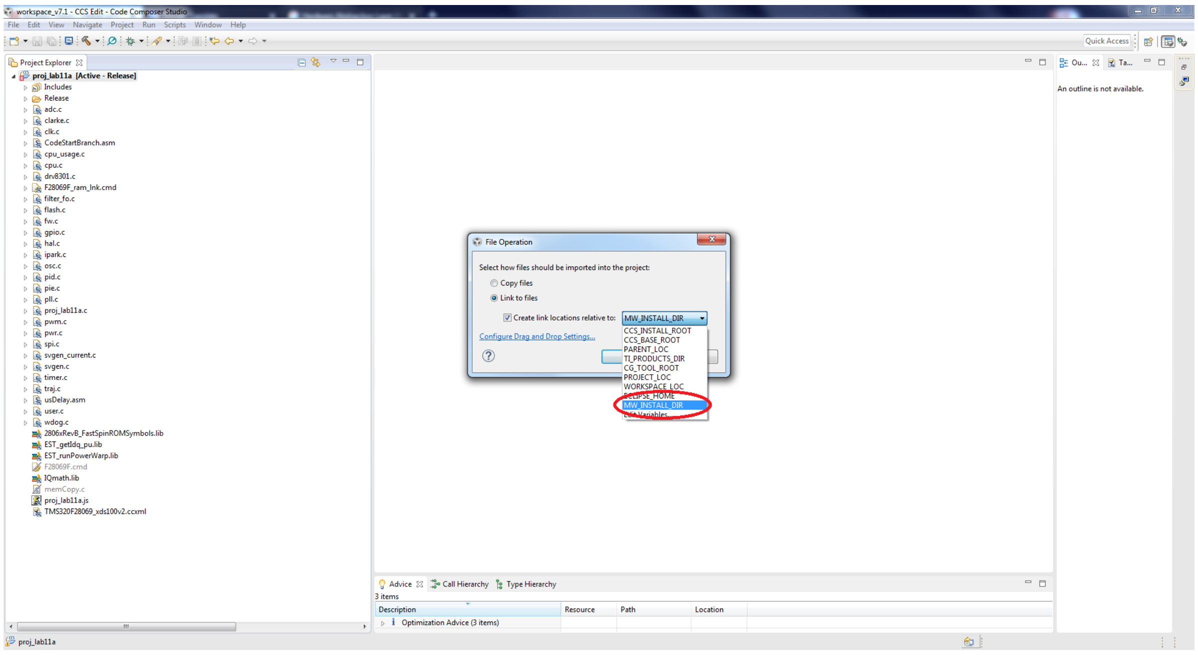

workspace_v7.1 - CCS Edit - Code Composer Studio
File Operation
Select how files should be imported into the project:
Copy files
Link to files
Create link locations relative to: MW_INSTALL_DIR
Configure Drag and Drop Settings...
CCS_INSTALL_ROOT
CCS_BASE_ROOT
PARENT_LOC
TI_PRODUCTS_DIR
CG_TOOL_ROOT
PROJECT_LOC
WORKSPACE_LOC
ECLIPSE_HOME
MW_INSTALL_DIR
Edit Variables...

Figure A.93: Step 8: Select Link files and select MW_INSTALL_DIR and click Ok.

9. Right-click on the project where the CAN interface needs to be integrated and click ”Add Files...”

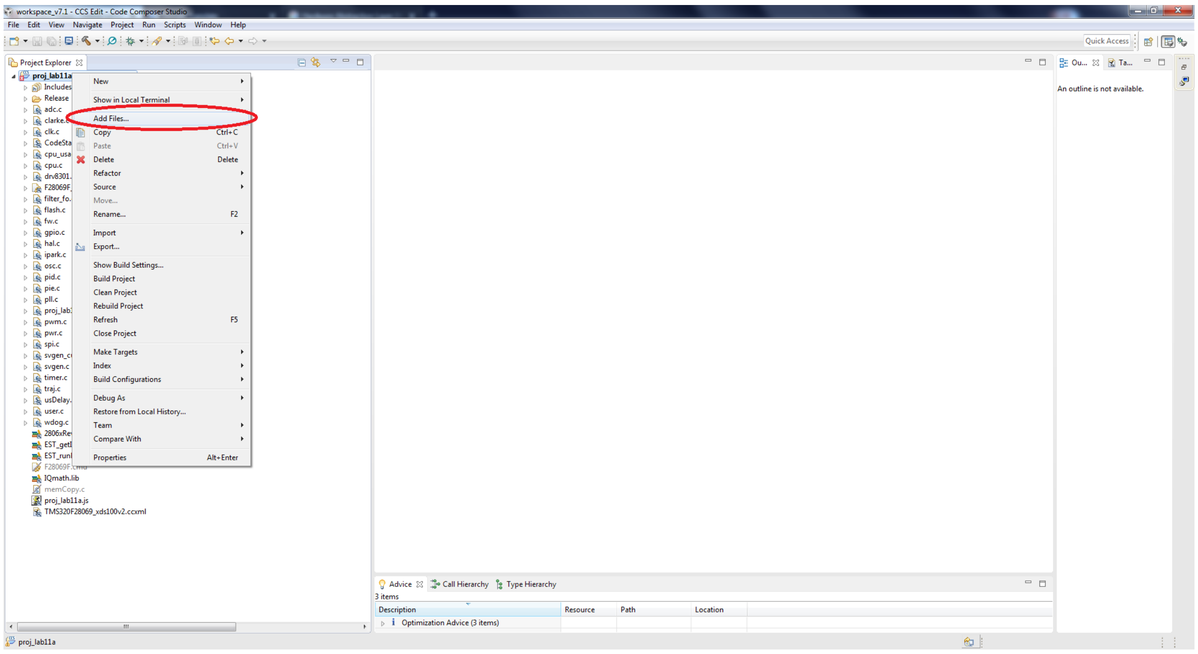
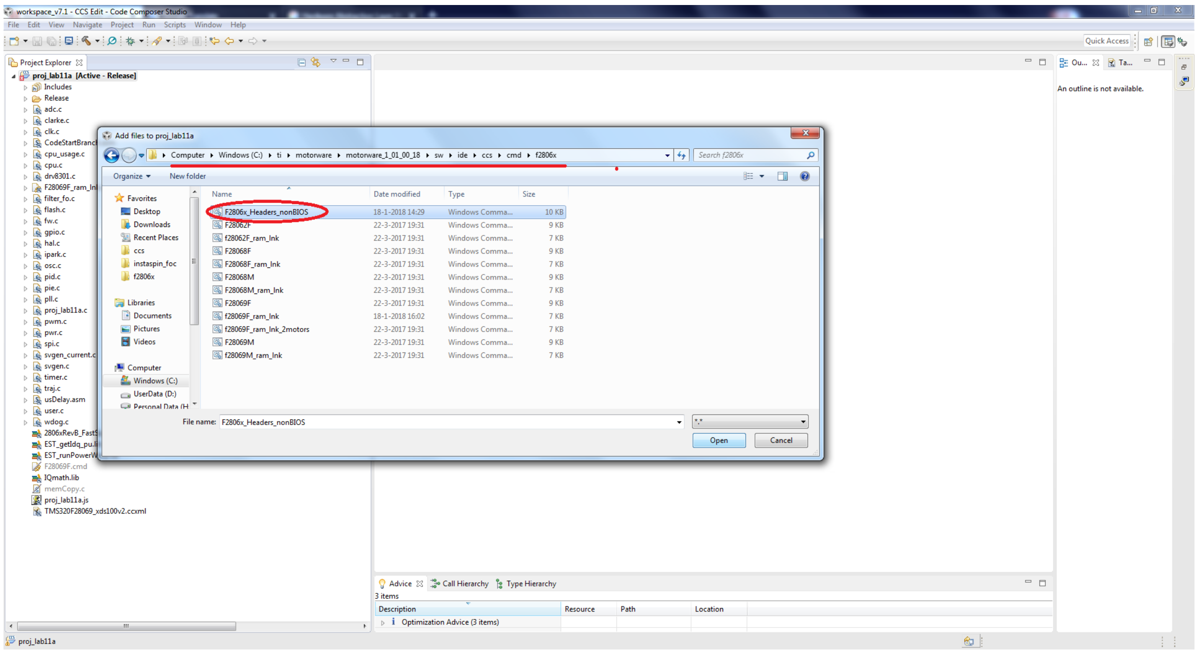


Figure A.94: Step 9: Right-click on the project and click on "Add Files...".

10. Go to "`C:\ti\motorware\motorware_1_01_00_18\sw\ide\ccs\cmd\f2806x`" and select F2806x_Headers_nonBIOS.

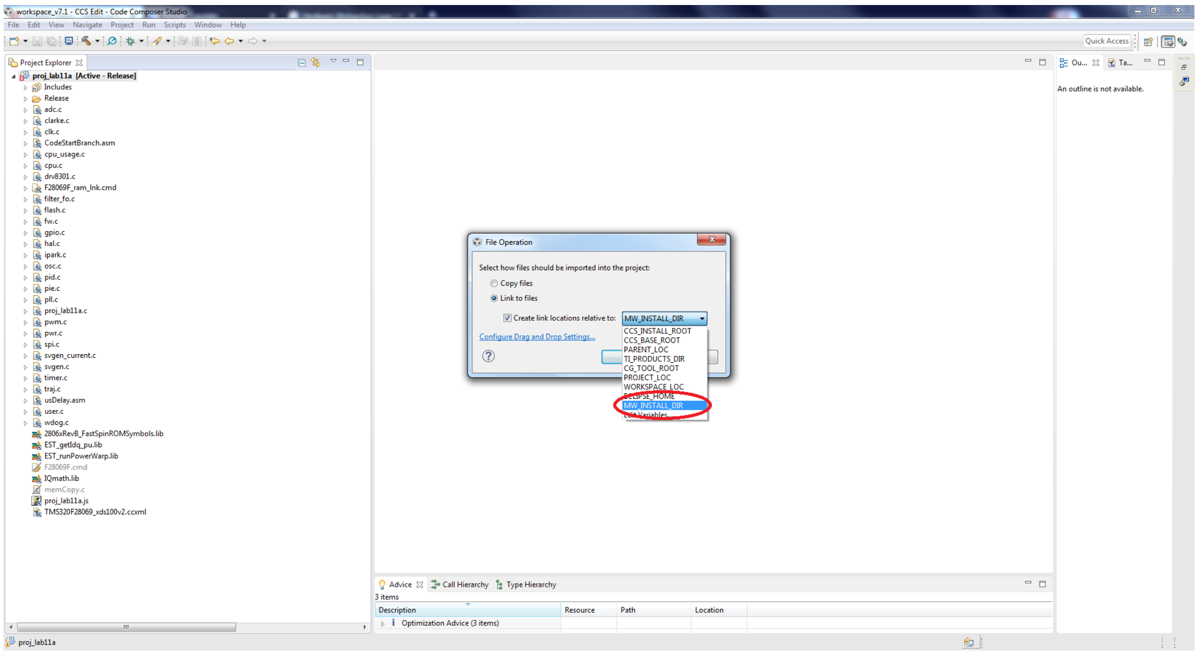

Add files to proj_lab11a
Computer ▸ Windows (C:) ▸ ti ▸ motorware ▸ motorware_1_01_00_18 ▸ sw ▸ ide ▸ ccs ▸ cmd ▸ f2806x
Search f2806x
Organize
New folder
Open
Cancel

Name	Date modified	Type	Size
F2806x_Headers_nonBIOS	18-1-2018 14:29	Windows Comma...	10 KB
F28062F	22-3-2017 19:31	Windows Comma...	9 KB
f28062F_ram_lnk	22-3-2017 19:31	Windows Comma...	7 KB
F28068F	22-3-2017 19:31	Windows Comma...	9 KB
F28068F_ram_lnk	22-3-2017 19:31	Windows Comma...	7 KB
F28068M	22-3-2017 19:31	Windows Comma...	9 KB
F28068M_ram_lnk	22-3-2017 19:31	Windows Comma...	7 KB
F28069F	22-3-2017 19:31	Windows Comma...	9 KB
f28069F_ram_lnk	18-1-2018 16:02	Windows Comma...	7 KB
f28069F_ram_lnk_2motors	22-3-2017 19:31	Windows Comma...	7 KB
F28069M	22-3-2017 19:31	Windows Comma...	9 KB
f28069M_ram_lnk	22-3-2017 19:31	Windows Comma...	7 KB

File name: F2806x_Headers_nonBIOS

Figure A.95: Step 10: Go to the underlined folder and select F2806x_Headers_nonBIOS.

11. Select link files and set the "Create link locations relative to:" drop-down menu to "MW_INSTALL_DIR". Than click OK.


File Operation

Select how files should be imported into the project:

( ) Copy files
(●) Link to files

[✓] Create link locations relative to: MW_INSTALL_DIR

Configure Drag and Drop Settings...

Figure A.96: Step 11: Select Link files and select MW_INSTALL_DIR and click Ok.

After adding the files, the codes of hal.c, hal.h and hal_obj.h need to be modified to integrate and initialize the CAN module.

12. Open hal.c, select hal_obj.h. Then right-click and click on ”Open Declaration” or press F3.

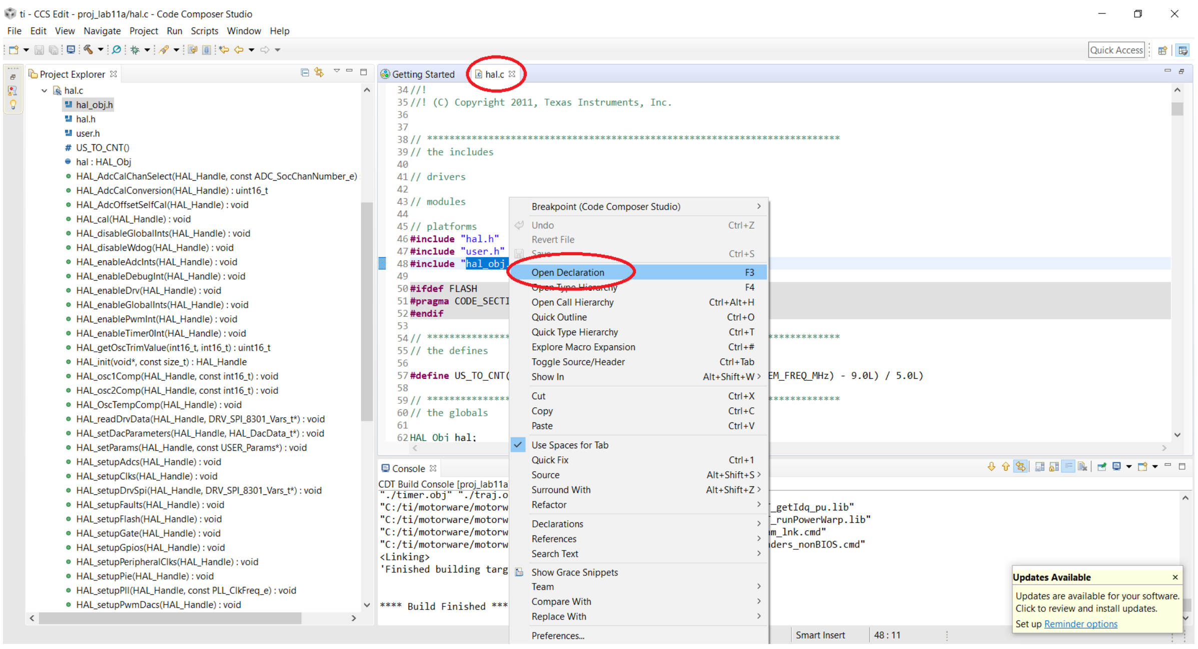
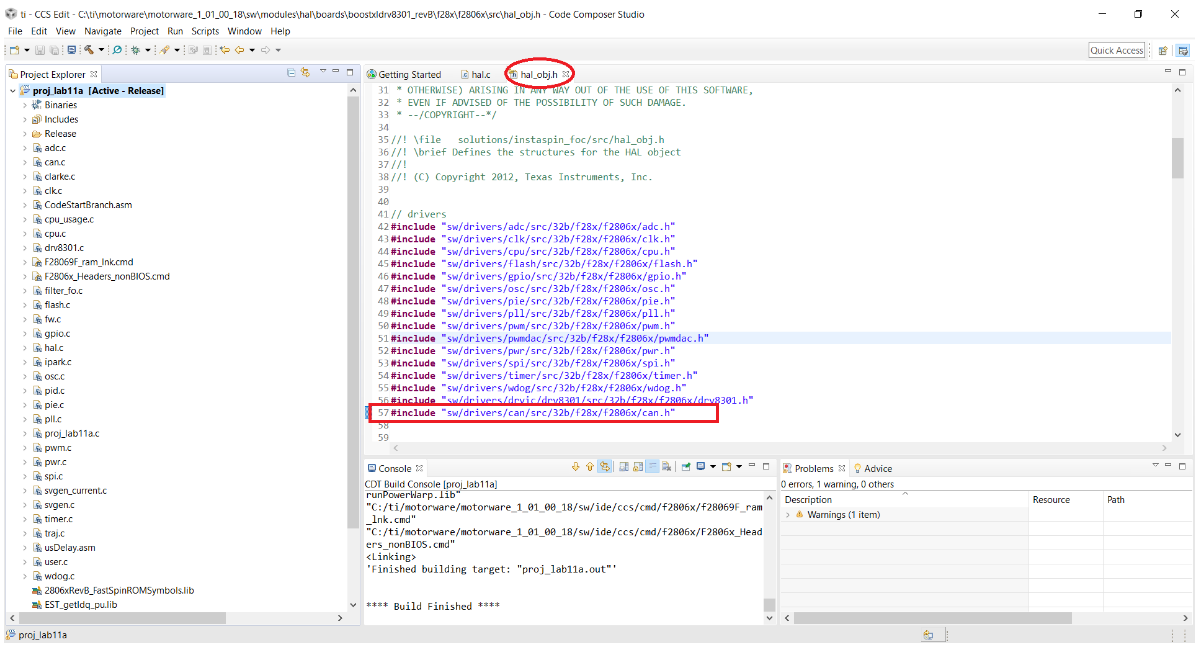

```
//!
//! (C) Copyright 2011, Texas Instruments, Inc.


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

// drivers

// modules

// platforms
#include "hal.h"
#include "user.h"
#include "hal_obj

#ifdef FLASH
#pragma CODE_SECTI
#endif

// ***************
// the defines

#define US_TO_CNT(
```

Figure A.97: Step 12: Open the file hal_obj.h.

13. Then hal_obj.h opens. Add the following line of code beneath the rest of the drivers:

```
#include ”sw/drivers/can/src/32b/f28x/f2806x/can.h”
```



ti - CCS Edit - C:\ti\motorware\motorware_1_01_00_18\sw\modules\hal\boards\boostxldrv8301_revB\f28x\f2806x\src\hal_obj.h - Code Composer Studio

```
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/

//! \file   solutions/instaspin_foc/src/hal_obj.h
//! \brief Defines the structures for the HAL object
//!
//! (C) Copyright 2012, Texas Instruments, Inc.


// drivers
#include "sw/drivers/adc/src/32b/f28x/f2806x/adc.h"
#include "sw/drivers/clk/src/32b/f28x/f2806x/clk.h"
#include "sw/drivers/cpu/src/32b/f28x/f2806x/cpu.h"
#include "sw/drivers/flash/src/32b/f28x/f2806x/flash.h"
#include "sw/drivers/gpio/src/32b/f28x/f2806x/gpio.h"
#include "sw/drivers/osc/src/32b/f28x/f2806x/osc.h"
#include "sw/drivers/pie/src/32b/f28x/f2806x/pie.h"
#include "sw/drivers/pll/src/32b/f28x/f2806x/pll.h"
#include "sw/drivers/pwm/src/32b/f28x/f2806x/pwm.h"
#include "sw/drivers/pwmdac/src/32b/f28x/f2806x/pwmdac.h"
#include "sw/drivers/pwr/src/32b/f28x/f2806x/pwr.h"
#include "sw/drivers/spi/src/32b/f28x/f2806x/spi.h"
#include "sw/drivers/timer/src/32b/f28x/f2806x/timer.h"
#include "sw/drivers/wdog/src/32b/f28x/f2806x/wdog.h"
#include "sw/drivers/drvic/drv8301/src/32b/f28x/f2806x/drv8301.h"
#include "sw/drivers/can/src/32b/f28x/f2806x/can.h"
```

CDT Build Console [proj_lab11a]

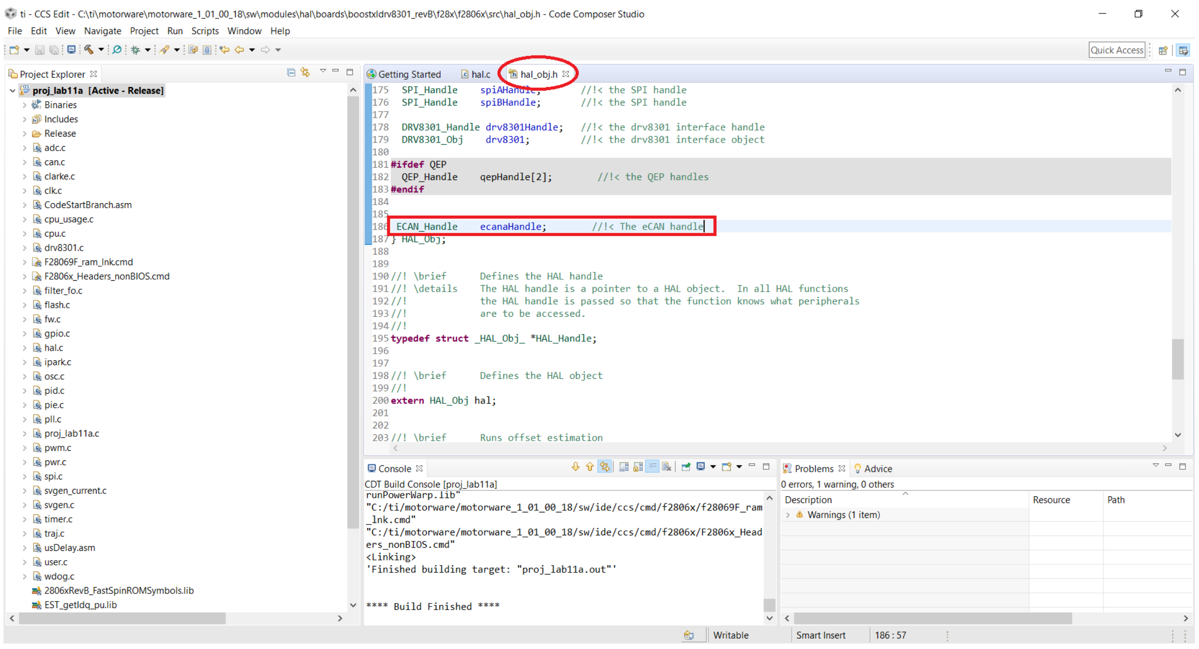
```
runPowerWarp.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram
_lnk.cmd"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/F2806x_Head
ers_nonBIOS.cmd"
<Linking>
'Finished building target: "proj_lab11a.out"'

**** Build Finished ****
```

Problems: 0 errors, 1 warning, 0 others — Warnings (1 item)

Figure A.98: Step 13: Add the line of code to hal_obj.h.

14. Add the following line of code at the end of the HAL_Obj function:

```
ECAN_Handle     ecanaHandle;         //!< The eCAN handle
```



ti - CCS Edit - C:\ti\motorware\motorware_1_01_00_18\sw\modules\hal\boards\boostxldrv8301_revB\f28x\f2806x\src\hal_obj.h - Code Composer Studio

```
  SPI_Handle    spiAHandle;       //!< the SPI handle
  SPI_Handle    spiBHandle;       //!< the SPI handle

  DRV8301_Handle drv8301Handle;   //!< the drv8301 interface handle
  DRV8301_Obj    drv8301;         //!< the drv8301 interface object

#ifdef QEP
  QEP_Handle    qepHandle[2];        //!< the QEP handles
#endif


 ECAN_Handle    ecanaHandle;        //!< The eCAN handle
} HAL_Obj;


//! \brief      Defines the HAL handle
//! \details    The HAL handle is a pointer to a HAL object.  In all HAL functions
//!             the HAL handle is passed so that the function knows what peripherals
//!             are to be accessed.
//!
typedef struct _HAL_Obj_ *HAL_Handle;


//! \brief      Defines the HAL object
//!
extern HAL_Obj hal;


//! \brief      Runs offset estimation
```

CDT Build Console [proj_lab11a]

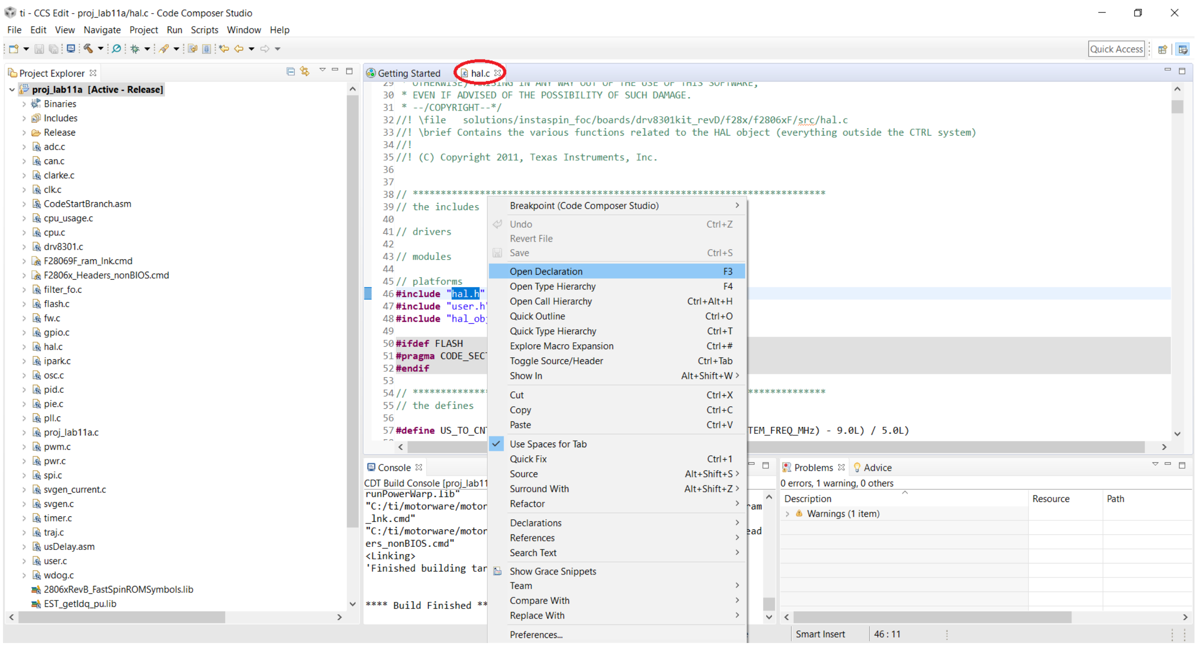
```
runPowerWarp.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram
_lnk.cmd"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/F2806x_Head
ers_nonBIOS.cmd"
<Linking>
'Finished building target: "proj_lab11a.out"'

**** Build Finished ****
```

Problems: 0 errors, 1 warning, 0 others — Warnings (1 item)

Writable   Smart Insert   186 : 57

Figure A.99: Step 14: Add the line of code to hal_obj.h.

15. Close hal_obj.h, open the hal.c code and select hal.h. Right-click and click on ”Open Declaration” or press F3.

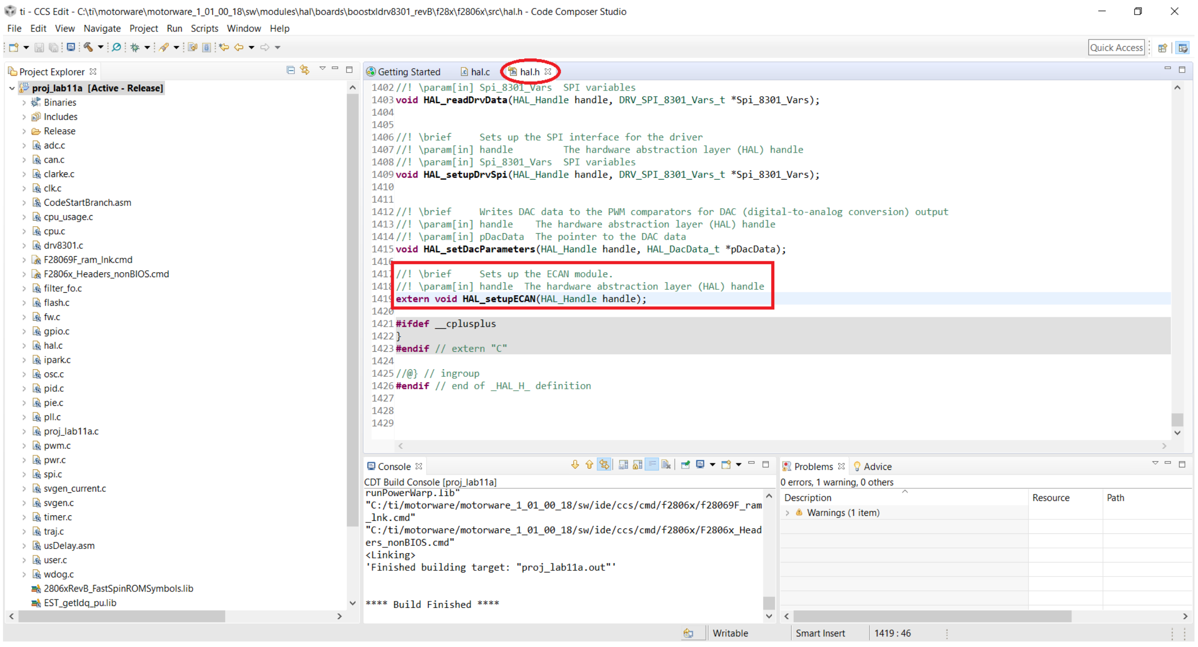


```
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/
//! \file   solutions/instaspin_foc/boards/drv8301kit_revD/f28x/f2806xF/src/hal.c
//! \brief Contains the various functions related to the HAL object (everything outside the CTRL system)
//!
//! (C) Copyright 2011, Texas Instruments, Inc.


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

// drivers

// modules

// platforms
#include "hal.h"
#include "user.h
#include "hal_ob

#ifdef FLASH
#pragma CODE_SEC
#endif

// *************
// the defines

#define US_TO_CN                                        TEM_FREQ_MHz) - 9.0L) / 5.0L)
```

Figure A.100: Step 15: Open the file hal.h.

16. Go to the end of the headerfile hal.h and add the following lines of code:

```
//! \brief      Sets up the ECAN module.
//! \param[in] handle  The hardware abstraction layer (HAL) handle
extern void HAL_setupECAN(HAL_Handle handle);
```


ti - CCS Edit - C:\ti\motorware\motorware_1_01_00_18\sw\modules\hal\boards\boostxldrv8301_revB\f28x\f2806x\src\hal.h - Code Composer Studio
Project Explorer
proj_lab11a [Active - Release]
Getting Started
hal.c
hal.h
Console
Problems
Advice
0 errors, 1 warning, 0 others
Description
Resource
Path
Warnings (1 item)
Writable
Smart Insert
1419 : 46

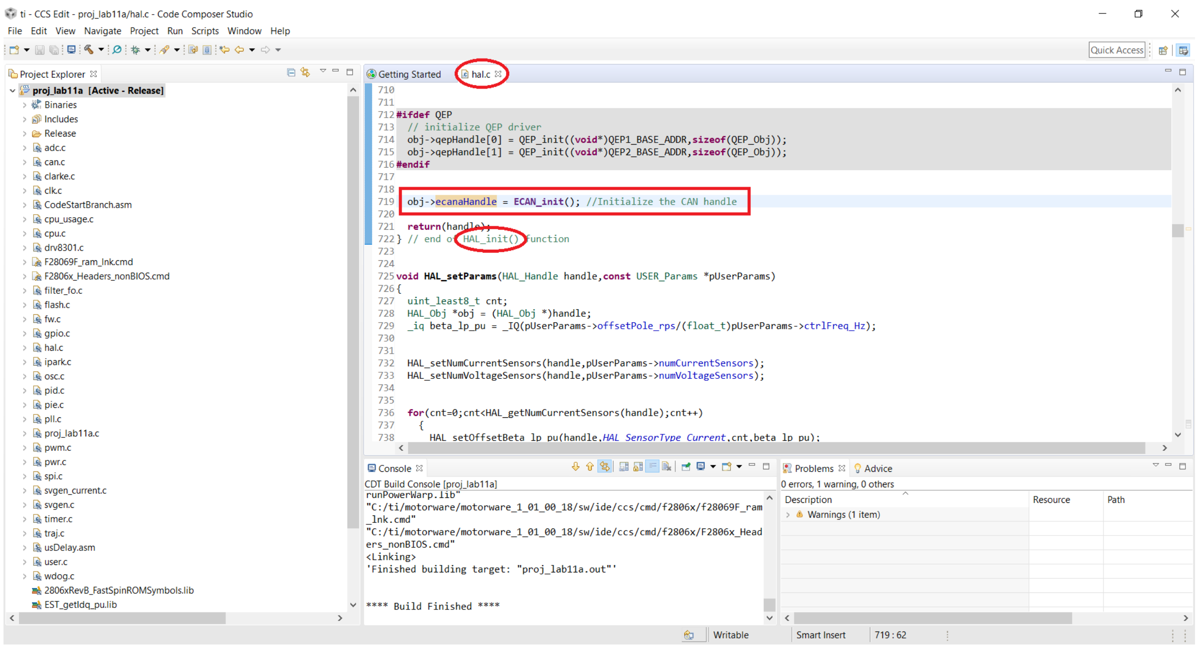
```
//! \param[in] Spi_8301_Vars  SPI variables
void HAL_readDrvData(HAL_Handle handle, DRV_SPI_8301_Vars_t *Spi_8301_Vars);


//! \brief     Sets up the SPI interface for the driver
//! \param[in] handle         The hardware abstraction layer (HAL) handle
//! \param[in] Spi_8301_Vars  SPI variables
void HAL_setupDrvSpi(HAL_Handle handle, DRV_SPI_8301_Vars_t *Spi_8301_Vars);


//! \brief     Writes DAC data to the PWM comparators for DAC (digital-to-analog conversion) output
//! \param[in] handle    The hardware abstraction layer (HAL) handle
//! \param[in] pDacData  The pointer to the DAC data
void HAL_setDacParameters(HAL_Handle handle, HAL_DacData_t *pDacData);

//! \brief     Sets up the ECAN module.
//! \param[in] handle  The hardware abstraction layer (HAL) handle
extern void HAL_setupECAN(HAL_Handle handle);

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

//@} // ingroup
#endif // end of _HAL_H_ definition
```

```
CDT Build Console [proj_lab11a]
runPowerWarp.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram
_lnk.cmd"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/F2806x_Head
ers_nonBIOS.cmd"
<Linking>
'Finished building target: "proj_lab11a.out"'

**** Build Finished ****
```

Figure A.101: Step 16: Add the lines of code to hal.h.

17. Close hal.h and open hal.c. At the end of the HAL_init() function, add the following line of code:

```
obj->ecanaHandle = ECAN_init(); //Initialize the CAN handle
```



```


#ifdef QEP
  // initialize QEP driver
  obj->qepHandle[0] = QEP_init((void*)QEP1_BASE_ADDR,sizeof(QEP_Obj));
  obj->qepHandle[1] = QEP_init((void*)QEP2_BASE_ADDR,sizeof(QEP_Obj));
#endif


  obj->ecanaHandle = ECAN_init(); //Initialize the CAN handle

  return(handle);
} // end of HAL_init() function


void HAL_setParams(HAL_Handle handle,const USER_Params *pUserParams)
{
  uint_least8_t cnt;
  HAL_Obj *obj = (HAL_Obj *)handle;
  _iq beta_lp_pu = _IQ(pUserParams->offsetPole_rps/(float_t)pUserParams->ctrlFreq_Hz);


  HAL_setNumCurrentSensors(handle,pUserParams->numCurrentSensors);
  HAL_setNumVoltageSensors(handle,pUserParams->numVoltageSensors);


  for(cnt=0;cnt<HAL_getNumCurrentSensors(handle);cnt++)
    {
      HAL_setOffsetBeta_lp_pu(handle,HAL_SensorType_Current,cnt,beta_lp_pu);
```

```
CDT Build Console [proj_lab11a]
runPowerWarp.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram
_lnk.cmd"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/F2806x_Head
ers_nonBIOS.cmd"
<Linking>
'Finished building target: "proj_lab11a.out"'


**** Build Finished ****
```

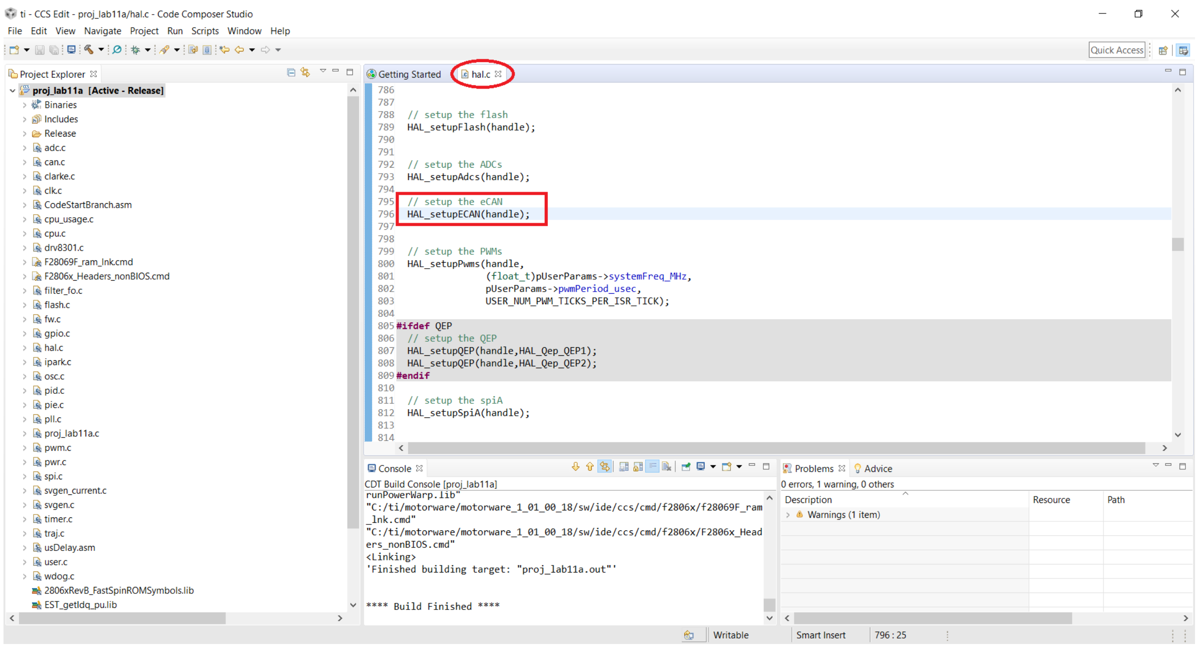
0 errors, 1 warning, 0 others

Warnings (1 item)

Figure A.102: Step 17: Add the line of code to hal.c.

18. Then call the function HAL_setupCAN by adding the following lines of code:

```
// setup the eCAN
HAL_setupECAN(handle);
```



```


  // setup the flash
  HAL_setupFlash(handle);


  // setup the ADCs
  HAL_setupAdcs(handle);

  // setup the eCAN
  HAL_setupECAN(handle);


  // setup the PWMs
  HAL_setupPwms(handle,
                (float_t)pUserParams->systemFreq_MHz,
                pUserParams->pwmPeriod_usec,
                USER_NUM_PWM_TICKS_PER_ISR_TICK);

#ifdef QEP
  // setup the QEP
  HAL_setupQEP(handle,HAL_Qep_QEP1);
  HAL_setupQEP(handle,HAL_Qep_QEP2);
#endif

  // setup the spiA
  HAL_setupSpiA(handle);


```

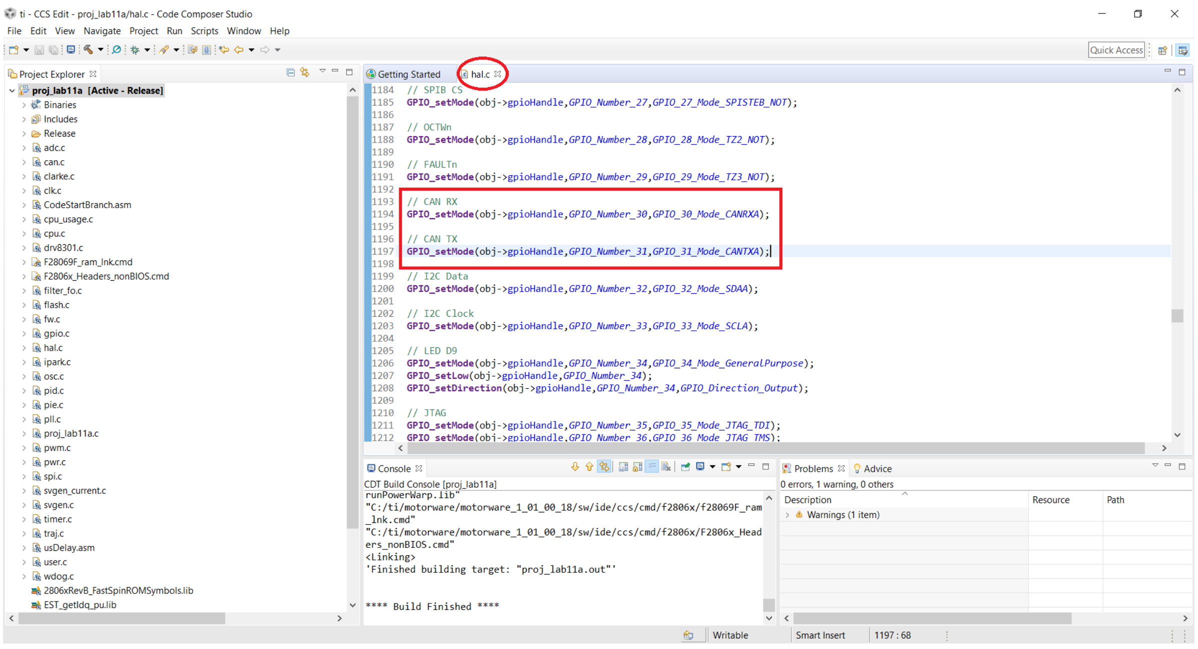
```
CDT Build Console [proj_lab11a]
runPowerWarp.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram
_lnk.cmd"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/F2806x_Head
ers_nonBIOS.cmd"
<Linking>
'Finished building target: "proj_lab11a.out"'


**** Build Finished ****
```

0 errors, 1 warning, 0 others

Warnings (1 item)

Figure A.103: Step 18: Add the lines of code to hal.c.

19. Check if the GPIO's are declared right. GPIO30 is for receiving CAN messages (CANRXA) and GPIO31 is for transmitting CAN messages (CANTXA).



```
// SPIB CS
GPIO_setMode(obj->gpioHandle,GPIO_Number_27,GPIO_27_Mode_SPISTEB_NOT);

// OCTWn
GPIO_setMode(obj->gpioHandle,GPIO_Number_28,GPIO_28_Mode_TZ2_NOT);

// FAULTn
GPIO_setMode(obj->gpioHandle,GPIO_Number_29,GPIO_29_Mode_TZ3_NOT);

// CAN RX
GPIO_setMode(obj->gpioHandle,GPIO_Number_30,GPIO_30_Mode_CANRXA);

// CAN TX
GPIO_setMode(obj->gpioHandle,GPIO_Number_31,GPIO_31_Mode_CANTXA);

// I2C Data
GPIO_setMode(obj->gpioHandle,GPIO_Number_32,GPIO_32_Mode_SDAA);

// I2C Clock
GPIO_setMode(obj->gpioHandle,GPIO_Number_33,GPIO_33_Mode_SCLA);

// LED D9
GPIO_setMode(obj->gpioHandle,GPIO_Number_34,GPIO_34_Mode_GeneralPurpose);
GPIO_setLow(obj->gpioHandle,GPIO_Number_34);
GPIO_setDirection(obj->gpioHandle,GPIO_Number_34,GPIO_Direction_Output);

// JTAG
GPIO_setMode(obj->gpioHandle,GPIO_Number_35,GPIO_35_Mode_JTAG_TDI);
GPIO_setMode(obj->gpioHandle,GPIO_Number_36,GPIO_36_Mode_JTAG_TMS);
```

CDT Build Console [proj_lab11a]

```
runPowerWarp.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram
_lnk.cmd"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/F2806x_Head
ers_nonBIOS.cmd"
<Linking>
'Finished building target: "proj_lab11a.out"'

**** Build Finished ****
```

0 errors, 1 warning, 0 others

Description	Resource	Path
Warnings (1 item)		

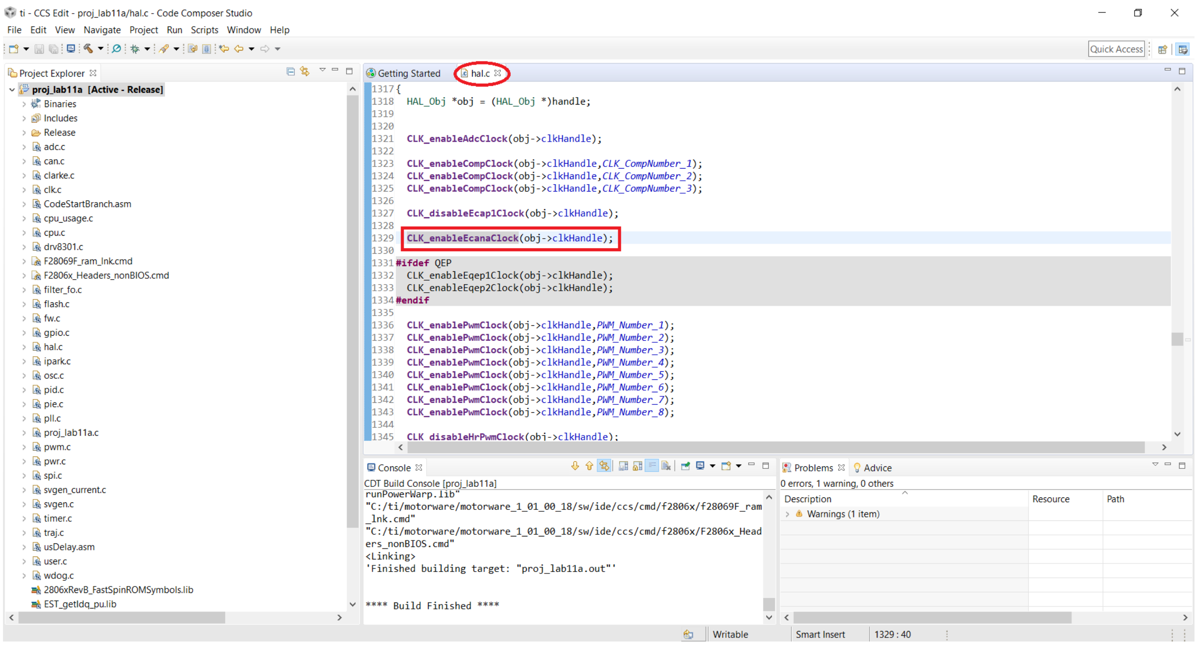
Figure A.104: Step 19: Check the GPIO 30 and 31 functions in hal.c.

20. Check if the eCAN clock is enabled in hal.c. If not change

    ```
    CLK_disableEcanaClock(obj->clkHandle);
    ```

    in

    ```
    CLK_enableEcanaClock(obj->clkHandle);
    ```



```
{
  HAL_Obj *obj = (HAL_Obj *)handle;


  CLK_enableAdcClock(obj->clkHandle);

  CLK_enableCompClock(obj->clkHandle,CLK_CompNumber_1);
  CLK_enableCompClock(obj->clkHandle,CLK_CompNumber_2);
  CLK_enableCompClock(obj->clkHandle,CLK_CompNumber_3);

  CLK_disableEcap1Clock(obj->clkHandle);

  CLK_enableEcanaClock(obj->clkHandle);

#ifdef QEP
  CLK_enableEqep1Clock(obj->clkHandle);
  CLK_enableEqep2Clock(obj->clkHandle);
#endif

  CLK_enablePwmClock(obj->clkHandle,PWM_Number_1);
  CLK_enablePwmClock(obj->clkHandle,PWM_Number_2);
  CLK_enablePwmClock(obj->clkHandle,PWM_Number_3);
  CLK_enablePwmClock(obj->clkHandle,PWM_Number_4);
  CLK_enablePwmClock(obj->clkHandle,PWM_Number_5);
  CLK_enablePwmClock(obj->clkHandle,PWM_Number_6);
  CLK_enablePwmClock(obj->clkHandle,PWM_Number_7);
  CLK_enablePwmClock(obj->clkHandle,PWM_Number_8);

  CLK_disableHrPwmClock(obj->clkHandle);
```

CDT Build Console [proj_lab11a]

```
runPowerWarp.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram
_lnk.cmd"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/F2806x_Head
ers_nonBIOS.cmd"
<Linking>
'Finished building target: "proj_lab11a.out"'

**** Build Finished ****
```

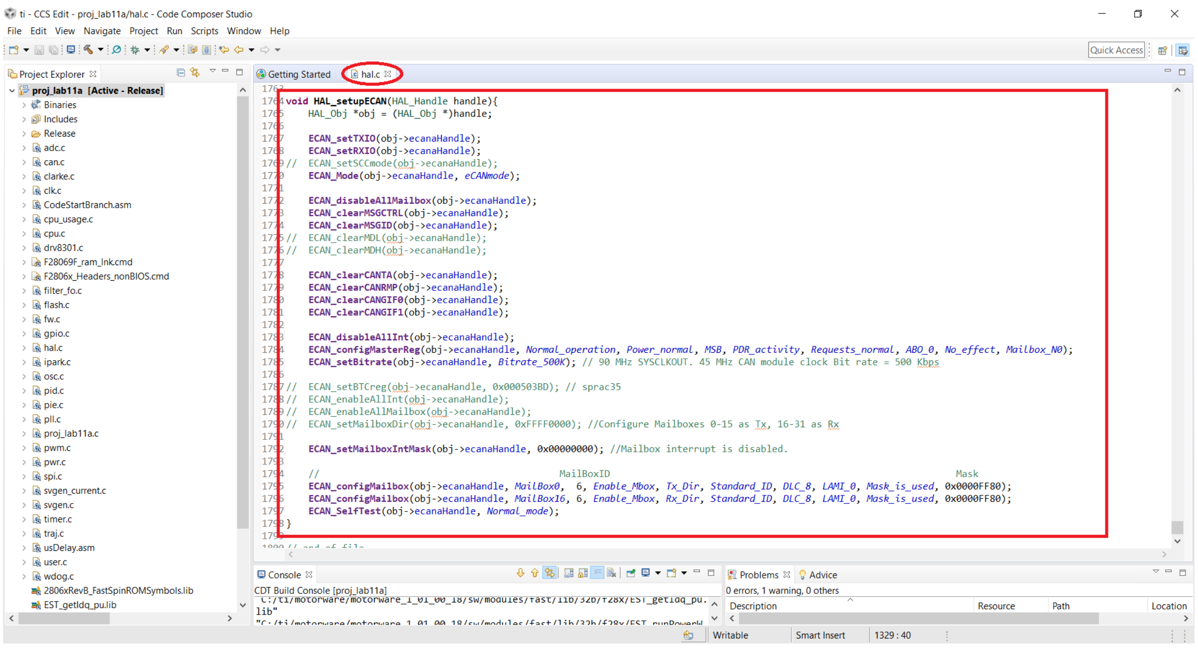
0 errors, 1 warning, 0 others

Description	Resource	Path
Warnings (1 item)		

Figure A.105: Step 20: Check if the eCAN clock is enabled in hal.c.

21. Add the following function at the end of hal.c:

```
void HAL_setupECAN(HAL_Handle handle){
    HAL_Obj *obj = (HAL_Obj *)handle;

    ECAN_setTXIO(obj->ecanaHandle);
    ECAN_setRXIO(obj->ecanaHandle);
//  ECAN_setSCCmode(obj->ecanaHandle);
    ECAN_Mode(obj->ecanaHandle, eCANmode);

    ECAN_disableAllMailbox(obj->ecanaHandle);
    ECAN_clearMSGCTRL(obj->ecanaHandle);
    ECAN_clearMSGID(obj->ecanaHandle);
//  ECAN_clearMDL(obj->ecanaHandle);
//  ECAN_clearMDH(obj->ecanaHandle);

    ECAN_clearCANTA(obj->ecanaHandle);
    ECAN_clearCANRMP(obj->ecanaHandle);
    ECAN_clearCANGIF0(obj->ecanaHandle);
    ECAN_clearCANGIF1(obj->ecanaHandle);

    ECAN_disableAllInt(obj->ecanaHandle);
    ECAN_configMasterReg(obj->ecanaHandle, Normal_operation,
    Power_normal, MSB, PDR_activity, Requests_normal,
    ABO_0, No_effect, Mailbox_N0);
    ECAN_setBitrate(obj->ecanaHandle, Bitrate_500K);
    // 90 MHz SYSCLKOUT. 45 MHz CAN module clock Bit rate = 500 Kbps

//  ECAN_setBTCreg(obj->ecanaHandle, 0x000503BD); // sprac35
//  ECAN_enableAllInt(obj->ecanaHandle);
//  ECAN_enableAllMailbox(obj->ecanaHandle);
//  ECAN_setMailboxDir(obj->ecanaHandle, 0xFFFF0000);
    //Configure Mailboxes 0-15 as Tx, 16-31 as Rx

    ECAN_setMailboxIntMask(obj->ecanaHandle, 0x00000000);
                            //Mailbox interrupt is disabled.

    //MailBoxID                         Mask
    ECAN_configMailbox(obj->ecanaHandle, MailBox0,  6, Enable_Mbox,
                        Tx_Dir, Standard_ID, DLC_8, LAMI_0,
                        Mask_is_used, 0x0000FF80);
    ECAN_configMailbox(obj->ecanaHandle, MailBox16, 6, Enable_Mbox,
                        Rx_Dir, Standard_ID, DLC_8, LAMI_0,
                        Mask_is_used, 0x0000FF80);
    ECAN_SelfTest(obj->ecanaHandle, Normal_mode);
}
```



```
void HAL_setupECAN(HAL_Handle handle){
    HAL_Obj *obj = (HAL_Obj *)handle;

    ECAN_setTXIO(obj->ecanaHandle);
    ECAN_setRXIO(obj->ecanaHandle);
//  ECAN_setSCCmode(obj->ecanaHandle);
    ECAN_Mode(obj->ecanaHandle, eCANmode);

    ECAN_disableAllMailbox(obj->ecanaHandle);
    ECAN_clearMSGCTRL(obj->ecanaHandle);
    ECAN_clearMSGID(obj->ecanaHandle);
//  ECAN_clearMDL(obj->ecanaHandle);
//  ECAN_clearMDH(obj->ecanaHandle);

    ECAN_clearCANTA(obj->ecanaHandle);
    ECAN_clearCANRMP(obj->ecanaHandle);
    ECAN_clearCANGIF0(obj->ecanaHandle);
    ECAN_clearCANGIF1(obj->ecanaHandle);

    ECAN_disableAllInt(obj->ecanaHandle);
    ECAN_configMasterReg(obj->ecanaHandle, Normal_operation, Power_normal, MSB, PDR_activity, Requests_normal, ABO_0, No_effect, Mailbox_N0);
    ECAN_setBitrate(obj->ecanaHandle, Bitrate_500K); // 90 MHz SYSCLKOUT. 45 MHz CAN module clock Bit rate = 500 Kbps

//  ECAN_setBTCreg(obj->ecanaHandle, 0x000503BD); // sprac35
//  ECAN_enableAllInt(obj->ecanaHandle);
//  ECAN_enableAllMailbox(obj->ecanaHandle);
//  ECAN_setMailboxDir(obj->ecanaHandle, 0xFFFF0000); //Configure Mailboxes 0-15 as Tx, 16-31 as Rx

    ECAN_setMailboxIntMask(obj->ecanaHandle, 0x00000000); //Mailbox interrupt is disabled.

    //                                        MailBoxID                                                        Mask
    ECAN_configMailbox(obj->ecanaHandle, MailBox0,  6, Enable_Mbox, Tx_Dir, Standard_ID, DLC_8, LAMI_0, Mask_is_used, 0x0000FF80);
    ECAN_configMailbox(obj->ecanaHandle, MailBox16, 6, Enable_Mbox, Rx_Dir, Standard_ID, DLC_8, LAMI_0, Mask_is_used, 0x0000FF80);
    ECAN_SelfTest(obj->ecanaHandle, Normal_mode);
}
```

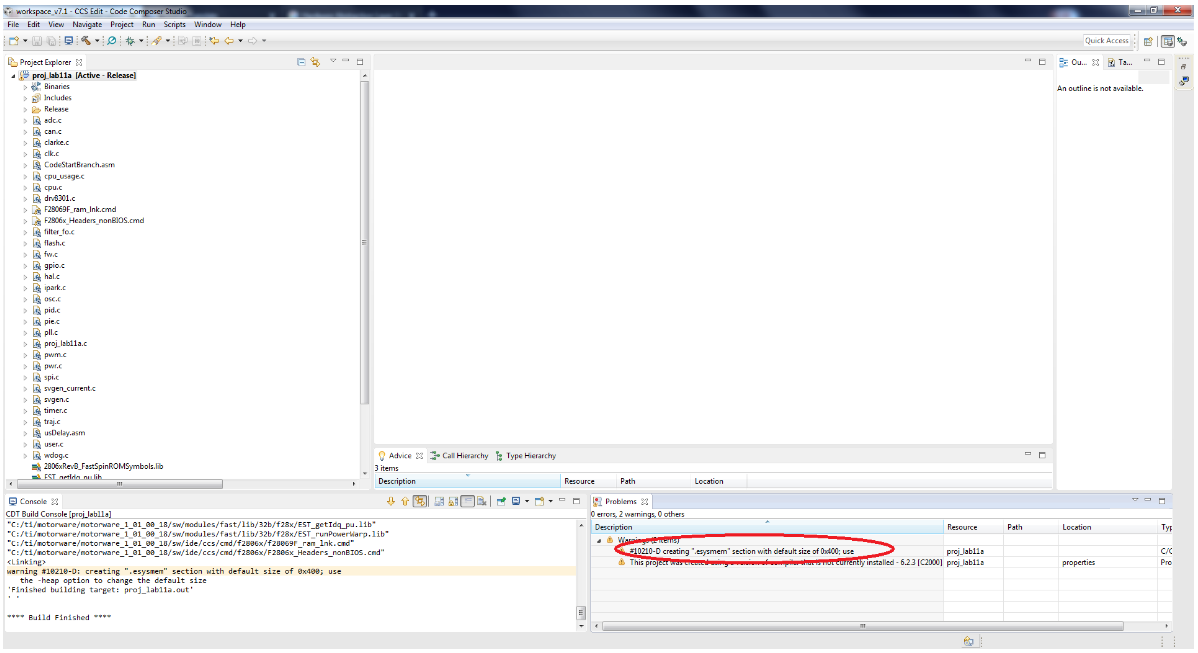
Figure A.106: Step 21: Add the following lines of code to hal.c.

Now the CAN interface is fully integrated in the project and ready to use. However the mailboxes still need to be configured in the proj_labx.c code, where x is the lab in which the CAN module is integrated.

A.7.1. .esysmem warning after building

It is possible that after building the project the following warning appears:

```
warning #10210-D: creating ".esysmem" section with default size of 0x400;
use the -heap option to change the default size
```



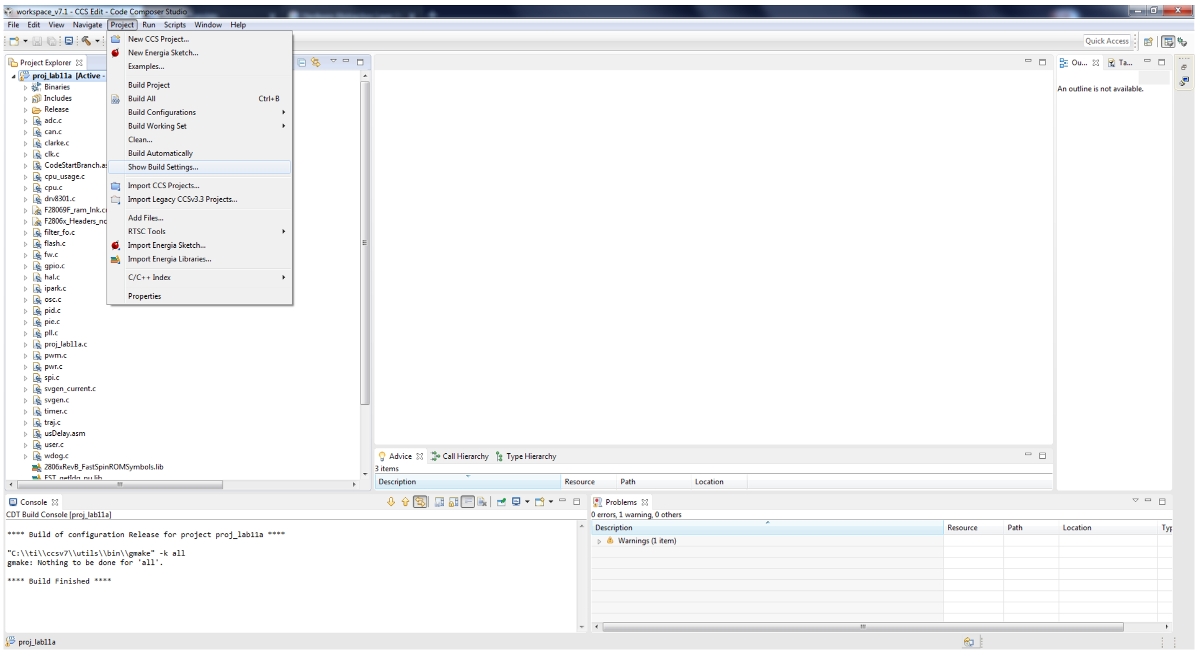
```
CDT Build Console [proj_lab11a]
"C:/ti/motorware/motorware_1_01_00_18/sw/modules/fast/lib/32b/f28x/EST_getIdq_pu.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/modules/fast/lib/32b/f28x/EST_runPowerWarp.lib"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/f28069F_ram_lnk.cmd"
"C:/ti/motorware/motorware_1_01_00_18/sw/ide/ccs/cmd/f2806x/F2806x_Headers_nonBIOS.cmd"
<Linking>
warning #10210-D: creating ".esysmem" section with default size of 0x400; use
   the -heap option to change the default size
'Finished building target: proj_lab11a.out'
' '

**** Build Finished ****
```

```
Problems
0 errors, 2 warnings, 0 others
Description                                                                          Resource     Path  Location    Type
Warnings (2 items)
  #10210-D creating ".esysmem" section with default size of 0x400; use               proj_lab11a                    C/C
  This project was created using a version of compiler that is not currently installed - 6.2.3 [C2000]  proj_lab11a  properties  Pro
```

Figure A.107: The .esysmem warning.

This can be resolved by doing the following:

1. Go to Project and click on "Show Build Settings...".

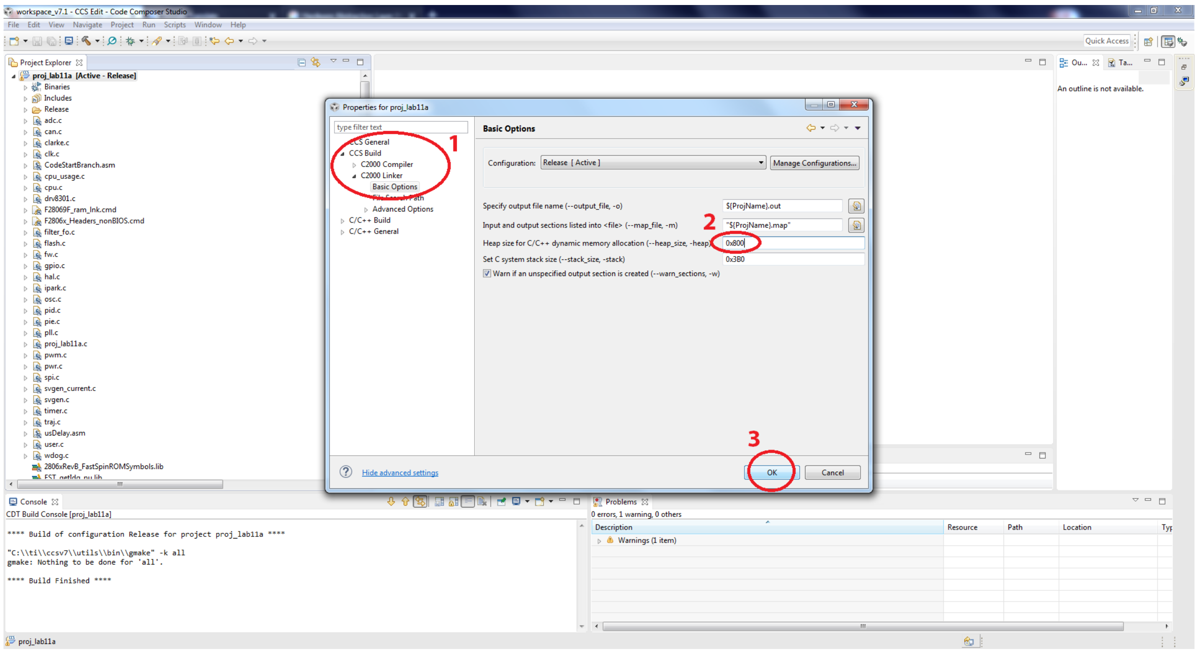


CDT Build Console [proj_lab11a]

```
**** Build of configuration Release for project proj_lab11a ****

"C:\\ti\\ccsv7\\utils\\bin\\gmake" -k all
gmake: Nothing to be done for 'all'.

**** Build Finished ****
```

Figure A.108: Step 1: Go to build settings.

2. In the build settings, open the "C2000 Linker" drop-down menu. In this menu select "Basic Options". At that moment the Heap Size is left empty. Fill in a hexadecimal number of at least 0x400, but preferably 0x800 or larger. Than click OK and the issue is resolved.



Configuration: Release [Active]

Specify output file name (--output_file, -o): ${ProjName}.out
Input and output sections listed into <file> (--map_file, -m): "${ProjName}.map"
Heap size for C/C++ dynamic memory allocation (--heap_size, -heap): 0x800
Set C system stack size (--stack_size, -stack): 0x3B0
[✓] Warn if an unspecified output section is created (--warn_sections, -w)

Figure A.109: Step 2: Change the Heap Size in the build settings.

B

user.h & user_j1.h code

B.1. user.h

```
#ifndef _USER_H_
#define _USER_H_
/* --COPYRIGHT--,BSD
 * Copyright (c) 2012, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/

//! \file   solutions/instaspin_foc/boards/boostxldrv8301_revB/f28x/f2806xF/src/user.h
//! \brief  Contains the public interface for user initialization data for the CTRL, HAL, and EST modules
//!
//! (C) Copyright 2012, Texas Instruments, Inc.


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

// modules
#include "sw/modules/types/src/types.h"
#include "sw/modules/motor/src/32b/motor.h"
#include "sw/modules/est/src/32b/est.h"
#include "sw/modules/est/src/est_states.h"
#include "sw/modules/est/src/est_Flux_states.h"
#include "sw/modules/est/src/est_Ls_states.h"
#include "sw/modules/est/src/est_Rs_states.h"
#include "sw/modules/ctrl/src/32b/ctrl_obj.h"


// select whether to use the inverter on connector J1 or J5 of the LaunchPad
#define J1
```

```
// platforms
#include "sw/modules/fast/src/32b/userParams.h"
#ifdef J5
#include "user_j5.h"
#else
#include "user_j1.h"
#endif

//!
//!
//! \defgroup USER USER
//!
//@{


#ifdef __cplusplus
extern "C" {
#endif

// **************************************************************************
// the defines


//! \brief CURRENTS AND VOLTAGES
// **************************************************************************
//! \brief Defines the voltage scale factor for the system
//! \brief Compile time calculation for scale factor (ratio) used throughout the system
#define USER_VOLTAGE_SF               ((float_t)((USER_ADC_FULL_SCALE_VOLTAGE_V)/(
     USER_IQ_FULL_SCALE_VOLTAGE_V)))

//! \brief Defines the current scale factor for the system
//! \brief Compile time calculation for scale factor (ratio) used throughout the system
#define USER_CURRENT_SF               ((float_t)((USER_ADC_FULL_SCALE_CURRENT_A)/(
     USER_IQ_FULL_SCALE_CURRENT_A)))


//! \brief CLOCKS & TIMERS
// **************************************************************************
//! \brief Defines the system clock frequency, MHz
#define USER_SYSTEM_FREQ_MHz             (90.0)

//! \brief Defines the address of controller handle
//!
#define USER_CTRL_HANDLE_ADDRESS    (0x13C40)
#define USER_CTRL_HANDLE_ADDRESS_1   (0x13D36)

//! \brief Defines the address of estimator handle
//!
#define USER_EST_HANDLE_ADDRESS     (0x13840)
#define USER_EST_HANDLE_ADDRESS_1    (0x13A3E)

//! \brief Defines the direct voltage (Vd) scale factor
//!
#define USER_VD_SF                  (0.95)

//! \brief Defines the Pulse Width Modulation (PWM) period, usec
//! \brief Compile time calculation
#define USER_PWM_PERIOD_usec        (1000.0/USER_PWM_FREQ_kHz)

//! \brief Defines the Interrupt Service Routine (ISR) frequency, Hz
//!
#define USER_ISR_FREQ_Hz            ((float_t)USER_PWM_FREQ_kHz * 1000.0 / (float_t)
     USER_NUM_PWM_TICKS_PER_ISR_TICK)

//! \brief Defines the Interrupt Service Routine (ISR) period, usec
//!
#define USER_ISR_PERIOD_usec        (USER_PWM_PERIOD_usec * (float_t)USER_NUM_PWM_TICKS_PER_ISR_TICK)


//! \brief DECIMATION
// **************************************************************************
//! \brief Defines the controller frequency, Hz
//! \brief Compile time calculation
#define USER_CTRL_FREQ_Hz           (uint_least32_t)(USER_ISR_FREQ_Hz/USER_NUM_ISR_TICKS_PER_CTRL_TICK)

//! \brief Defines the estimator frequency, Hz
//! \brief Compile time calculation
#define USER_EST_FREQ_Hz            (uint_least32_t)(USER_CTRL_FREQ_Hz/USER_NUM_CTRL_TICKS_PER_EST_TICK)

//! \brief Defines the trajectory frequency, Hz
//! \brief Compile time calculation
#define USER_TRAJ_FREQ_Hz           (uint_least32_t)(USER_CTRL_FREQ_Hz/USER_NUM_CTRL_TICKS_PER_TRAJ_TICK)
```

```
//! \brief Defines the controller execution period, usec
//! \brief Compile time calculation
#define USER_CTRL_PERIOD_usec        (USER_ISR_PERIOD_usec * USER_NUM_ISR_TICKS_PER_CTRL_TICK)

//! \brief Defines the controller execution period, sec
//! \brief Compile time calculation
#define USER_CTRL_PERIOD_sec         ((float_t)USER_CTRL_PERIOD_usec/(float_t)1000000.0)


//! \brief LIMITS
// **************************************************************************
//! \brief Defines the maximum current slope for Id trajectory during PowerWarp
//! \brief For Induction motors only, controls how fast Id input can change under PowerWarp control
#define USER_MAX_CURRENT_SLOPE_POWERWARP    (0.3*USER_MOTOR_RES_EST_CURRENT/USER_IQ_FULL_SCALE_CURRENT_A/
     USER_TRAJ_FREQ_Hz)  // 0.3*RES_EST_CURRENT / IQ_FULL_SCALE_CURRENT / TRAJ_FREQ Typical to produce 1-
     sec rampup/down

//! \brief Defines the starting maximum acceleration AND deceleration for the speed profiles, Hz/s
//! \brief Updated in run-time through user functions
//! \brief Inverter, motor, inertia, and load will limit actual acceleration capability
#define USER_MAX_ACCEL_Hzps                 (20.0)       // 20.0 Default

//! \brief Defines maximum acceleration for the estimation speed profiles, Hz/s
//! \brief Only used during Motor ID (commission)
#define USER_MAX_ACCEL_EST_Hzps            (5.0)         // 5.0 Default, don't change

//! \brief Defines the maximum current slope for Id trajectory during estimation
#define USER_MAX_CURRENT_SLOPE             (USER_MOTOR_RES_EST_CURRENT/USER_IQ_FULL_SCALE_CURRENT_A/
     USER_TRAJ_FREQ_Hz)      // USER_MOTOR_RES_EST_CURRENT/USER_IQ_FULL_SCALE_CURRENT_A/USER_TRAJ_FREQ_Hz
     Default, don't change

//! \brief Defines the fraction of IdRated to use during rated flux estimation
//!
#define USER_IDRATED_FRACTION_FOR_RATED_FLUX (1.0)       // 1.0 Default, don't change

//! \brief Defines the fraction of IdRated to use during inductance estimation
//!
#define USER_IDRATED_FRACTION_FOR_L_IDENT     (1.0)       // 1.0 Default, don't change

//! \brief Defines the IdRated delta to use during estimation
//!
#define USER_IDRATED_DELTA                  (0.00002)

//! \brief Defines the fraction of SpeedMax to use during inductance estimation
//!
#define USER_SPEEDMAX_FRACTION_FOR_L_IDENT   (1.0)       // 1.0 Default, don't change

//! \brief Defines flux fraction to use during inductance identification
//!
#define USER_FLUX_FRACTION              (1.0)             // 1.0 Default, don't change

//! \brief Defines the PowerWarp gain for computing Id reference
//! \brief Induction motors only
#define USER_POWERWARP_GAIN                   (1.0)          // 1.0 Default, don't change


//! \brief POLES
// **************************************************************************
//! \brief Defines the analog voltage filter pole location, rad/s
//! \brief Compile time calculation from Hz to rad/s
#define USER_VOLTAGE_FILTER_POLE_rps   (2.0 * MATH_PI * USER_VOLTAGE_FILTER_POLE_Hz)

//! \brief Defines the software pole location for the voltage and current offset estimation, rad/s
//! \brief Should not be changed from default of (20.0)
#define USER_OFFSET_POLE_rps             (20.0)   // 20.0 Default, do not change

//! \brief Defines the software pole location for the flux estimation, rad/s
//! \brief Should not be changed from default of (100.0)
#define USER_FLUX_POLE_rps               (100.0)   // 100.0 Default, do not change

//! \brief Defines the software pole location for the direction filter, rad/s
#define USER_DIRECTION_POLE_rps              (6.0)    // 6.0 Default, do not change

//! \brief Defines the software pole location for the speed control filter, rad/s
#define USER_SPEED_POLE_rps            (100.0)   // 100.0 Default, do not change

//! \brief Defines the software pole location for the DC bus filter, rad/s
#define USER_DCBUS_POLE_rps            (100.0)   // 100.0 Default, do not change

//! \brief Defines the convergence factor for the estimator
//! \brief Do not change from default for FAST
#define   USER_EST_KAPPAQ                (1.5)    // 1.5 Default, do not change

// **************************************************************************
// end the defines

//! \brief USER MOTOR & ID SETTINGS
// ****************************************************************************
// this section defined in user_j1.h or user_j5.h

#ifndef USER_MOTOR
#error Motor is not defined in user.h
#endif

#ifndef USER_MOTOR_TYPE
#error The motor type is not defined in user.h
#endif

#ifndef USER_MOTOR_NUM_POLE_PAIRS
#error Number of motor pole pairs is not defined in user.h
#endif

#ifndef USER_MOTOR_Rr
#error The rotor resistance is not defined in user.h
#endif

#ifndef USER_MOTOR_Rs
#error The stator resistance is not defined in user.h
#endif

#ifndef USER_MOTOR_Ls_d
#error The direct stator inductance is not defined in user.h
#endif

#ifndef USER_MOTOR_Ls_q
#error The quadrature stator inductance is not defined in user.h
#endif

#ifndef USER_MOTOR_RATED_FLUX
#error The rated flux of motor is not defined in user.h
#endif

#ifndef USER_MOTOR_MAGNETIZING_CURRENT
#error The magnetizing current is not defined in user.h
#endif

#ifndef USER_MOTOR_RES_EST_CURRENT
#error The resistance estimation current is not defined in user.h
#endif

#ifndef USER_MOTOR_IND_EST_CURRENT
#error The inductance estimation current is not defined in user.h
#endif

#ifndef USER_MOTOR_MAX_CURRENT
#error The maximum current is not defined in user.h
#endif

#ifndef USER_MOTOR_FLUX_EST_FREQ_Hz
#error The flux estimation frequency is not defined in user.h
#endif


// ****************************************************************************
// the functions


//! \brief      Sets the user parameter values
//! \param[in]  pUserParams  The pointer to the user param structure
extern void USER_setParams(USER_Params *pUserParams);


//! \brief      Checks for errors in the user parameter values
//! \param[in]  pUserParams  The pointer to the user param structure
extern void USER_checkForErrors(USER_Params *pUserParams);


//! \brief      Gets the error code in the user parameters
//! \param[in]  pUserParams  The pointer to the user param structure
//! \return     The error code
extern USER_ErrorCode_e USER_getErrorCode(USER_Params *pUserParams);


//! \brief      Sets the error code in the user parameters
//! \param[in]  pUserParams  The pointer to the user param structure
//! \param[in]  errorCode    The error code
extern void USER_setErrorCode(USER_Params *pUserParams,const USER_ErrorCode_e errorCode);
```

```
//! \brief      Recalculates Inductances with the correct Q Format
//! \param[in]  handle       The controller (CTRL) handle
extern void USER_softwareUpdate1p6(CTRL_Handle handle);


//! \brief      Updates Id and Iq PI gains
//! \param[in]  handle       The controller (CTRL) handle
extern void USER_calcPIgains(CTRL_Handle handle);


//! \brief      Computes the scale factor needed to convert from torque created by Ld, Lq, Id and Iq, from
      per unit to Nm
//! \return     The scale factor to convert torque from (Ld - Lq) * Id * Iq from per unit to Nm, in IQ24
     format
extern _iq USER_computeTorque_Ls_Id_Iq_pu_to_Nm_sf(void);


//! \brief      Computes the scale factor needed to convert from torque created by flux and Iq, from per
     unit to Nm
//! \return     The scale factor to convert torque from Flux * Iq from per unit to Nm, in IQ24 format
extern _iq USER_computeTorque_Flux_Iq_pu_to_Nm_sf(void);


//! \brief      Computes the scale factor needed to convert from per unit to Wb
//! \return     The scale factor to convert from flux per unit to flux in Wb, in IQ24 format
extern _iq USER_computeFlux_pu_to_Wb_sf(void);


//! \brief      Computes the scale factor needed to convert from per unit to V/Hz
//! \return     The scale factor to convert from flux per unit to flux in V/Hz, in IQ24 format
extern _iq USER_computeFlux_pu_to_VpHz_sf(void);


//! \brief      Computes Flux in Wb or V/Hz depending on the scale factor sent as parameter
//! \param[in]  handle       The controller (CTRL) handle
//! \param[in]  sf           The scale factor to convert flux from per unit to Wb or V/Hz
//! \return     The flux in Wb or V/Hz depending on the scale factor sent as parameter, in IQ24 format
extern _iq USER_computeFlux(CTRL_Handle handle, const _iq sf);


//! \brief      Computes Torque in Nm
//! \param[in]  handle          The controller (CTRL) handle
//! \param[in]  torque_Flux_sf  The scale factor to convert torque from (Ld - Lq) * Id * Iq from per unit
     to Nm
//! \param[in]  torque_Ls_sf    The scale factor to convert torque from Flux * Iq from per unit to Nm
//! \return     The torque in Nm, in IQ24 format
extern _iq USER_computeTorque_Nm(CTRL_Handle handle, const _iq torque_Flux_sf, const _iq torque_Ls_sf);


//! \brief      Computes Torque in lbin
//! \param[in]  handle          The controller (CTRL) handle
//! \param[in]  torque_Flux_sf  The scale factor to convert torque from (Ld - Lq) * Id * Iq from per unit
     to lbin
//! \param[in]  torque_Ls_sf    The scale factor to convert torque from Flux * Iq from per unit to lbin
//! \return     The torque in lbin, in IQ24 format
extern _iq USER_computeTorque_lbin(CTRL_Handle handle, const _iq torque_Flux_sf, const _iq torque_Ls_sf);


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

//@} // ingroup
#endif // end of _USER_H_ definition
```

B.2. user_j1.h

```
#ifndef _USER_J1_H_
#define _USER_J1_H_
/* --COPYRIGHT--,BSD
 * Copyright (c) 2012, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/

//! \file   solutions/instaspin_foc/boards/boostxldrv8301_revB/f28x/f2806xF/src/user_j1.h
//! \brief Contains the public interface for user initialization data for the CTRL, HAL, and EST modules
//!
//! (C) Copyright 2012, Texas Instruments, Inc.


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

//!
//!
//! \defgroup USER USER
//!
//@{


#ifdef __cplusplus
extern "C" {
#endif

// **************************************************************************
// the defines

//! \brief CURRENTS AND VOLTAGES
// **************************************************************************
//! \brief Defines the full scale frequency for IQ variable, Hz
//! \brief All frequencies are converted into (pu) based on the ratio to this value
//! \brief this value MUST be larger than the maximum speed that you are expecting from the motor
#define USER_IQ_FULL_SCALE_FREQ_Hz        (800.0)   // 800 Example with buffer for 8-pole 6 KRPM motor to
      be run to 10 KRPM with field weakening; Hz =(RPM * Poles) / 120

//! \brief Defines full scale value for the IQ30 variable of Voltage inside the system
//! \brief All voltages are converted into (pu) based on the ratio to this value
//! \brief WARNING: this value MUST meet the following condition: USER_IQ_FULL_SCALE_VOLTAGE_V > 0.5 *
      USER_MOTOR_MAX_CURRENT * USER_MOTOR_Ls_d * USER_VOLTAGE_FILTER_POLE_rps,
//! \brief WARNING: otherwise the value can saturate and roll-over, causing an inaccurate value
//! \brief WARNING: this value is OFTEN greater than the maximum measured ADC value, especially with high
      Bemf motors operating at higher than rated speeds
//! \brief WARNING: if you know the value of your Bemf constant, and you know you are operating at a
      multiple speed due to field weakening, be sure to set this value higher than the expected Bemf
      voltage
//! \brief It is recommended to start with a value ~3x greater than the USER_ADC_FULL_SCALE_VOLTAGE_V and
      increase to 4-5x if scenarios where a Bemf calculation may exceed these limits
//! \brief This value is also used to calculate the minimum flux value: USER_IQ_FULL_SCALE_VOLTAGE_V/
      USER_EST_FREQ_Hz/0.7
#define USER_IQ_FULL_SCALE_VOLTAGE_V      (24.0)   // 24.0 Example for boostxldrv8301_revB typical usage
      and the Anaheim motor
```

```
//! \brief Defines the maximum voltage at the input to the AD converter
//! \brief The value that will be represented by the maximum ADC input (3.3V) and conversion (0FFFh)
//! \brief Hardware dependent, this should be based on the voltage sensing and scaling to the ADC input
#define USER_ADC_FULL_SCALE_VOLTAGE_V        (26.314)       // 26.314 boostxldrv8301_revB voltage scaling

//! \brief Defines the full scale current for the IQ variables, A
//! \brief All currents are converted into (pu) based on the ratio to this value
//! \brief WARNING: this value MUST be larger than the maximum current readings that you are expecting
     from the motor or the reading will roll over to 0, creating a control issue
#define USER_IQ_FULL_SCALE_CURRENT_A          (20.0) // 20.0 Example for boostxldrv8301_revB typical usage

//! \brief Defines the maximum current at the AD converter
//! \brief The value that will be represented by the maximum ADC input (3.3V) and conversion (0FFFh)
//! \brief Hardware dependent, this should be based on the current sensing and scaling to the ADC input
#define USER_ADC_FULL_SCALE_CURRENT_A         (33.0)  // 33.0 boostxldrv8301_revB current scaling

//! \brief Defines the number of current sensors used
//! \brief Defined by the hardware capability present
//! \brief May be (2) or (3)
#define USER_NUM_CURRENT_SENSORS             (3)    // 3 Preferred setting for best performance across full
     speed range, allows for 100% duty cycle

//! \brief Defines the number of voltage (phase) sensors
//! \brief Must be (3)
#define USER_NUM_VOLTAGE_SENSORS             (3) // 3 Required

//! \brief ADC current offsets for A, B, and C phases
//! \brief One-time hardware dependent, though the calibration can be done at run-time as well
//! \brief After initial board calibration these values should be updated for your specific hardware so
     they are available after compile in the binary to be loaded to the controller
#define   I_A_offset    (0.8331743479)
#define   I_B_offset    (0.8355930448)
#define   I_C_offset    (0.8392037153)

//! \brief ADC voltage offsets for A, B, and C phases
//! \brief One-time hardware dependent, though the calibration can be done at run-time as well
//! \brief After initial board calibration these values should be updated for your specific hardware so
     they are available after compile in the binary to be loaded to the controller
#define   V_A_offset    (0.5271264911)
#define   V_B_offset    (0.5257175565)
#define   V_C_offset    (0.5249399543)


//! \brief CLOCKS & TIMERS
// **************************************************************************
//! \brief Defines the Pulse Width Modulation (PWM) frequency, kHz
//! \brief PWM frequency can be set directly here up to 30 KHz safely (60 KHz MAX in some cases)
//! \brief For higher PWM frequencies (60 KHz+ typical for low inductance, high current ripple motors) it
     is recommended to use the ePWM hardware
//! \brief and adjustable ADC SOC to decimate the ADC conversion done interrupt to the control system, or
     to use the software Que example.
//! \brief Otherwise you risk missing interrupts and disrupting the timing of the control state machine
#define USER_PWM_FREQ_kHz                 (45.0) //30.0 Example, 8.0 - 30.0 KHz typical; 45-80 KHz may be
     required for very low inductance, high speed motors

//! \brief Defines the maximum Voltage vector (Vs) magnitude allowed.  This value sets the maximum
     magnitude for the output of the
//! \brief Id and Iq PI current controllers.  The Id and Iq current controller outputs are Vd and Vq.
//! \brief The relationship between Vs, Vd, and Vq is:  Vs = sqrt(Vd^2 + Vq^2).  In this FOC controller,
     the
//! \brief Vd value is set equal to USER_MAX_VS_MAG*USER_VD_MAG_FACTOR.  Vq = sqrt(USER_MAX_VS_MAG^2 - Vd
     ^2).
//! \brief Set USER_MAX_VS_MAG = 0.5 for a pure sinewave with a peak at SQRT(3)/2 = 86.6% duty cycle.  No
     current reconstruction is needed for this scenario.
//! \brief Set USER_MAX_VS_MAG = 1/SQRT(3) = 0.5774 for a pure sinewave with a peak at 100% duty cycle.
     Current reconstruction will be needed for this scenario (Lab10a-x).
//! \brief Set USER_MAX_VS_MAG = 2/3 = 0.6666 to create a trapezoidal voltage waveform.  Current
     reconstruction will be needed for this scenario (Lab10a-x).
//! \brief For space vector over-modulation, see lab 10 for details on system requirements that will allow
      the SVM generator to go all the way to trapezoidal.
#define USER_MAX_VS_MAG_PU        (0.5)     // Set to 0.5 if a current reconstruction technique is not used
     .  Look at the module svgen_current in lab10a-x for more info.


//! \brief DECIMATION
// **************************************************************************
//! \brief Defines the number of pwm clock ticks per isr clock tick
//!        Note: Valid values are 1, 2 or 3 only
#define USER_NUM_PWM_TICKS_PER_ISR_TICK        (3)

//! \brief Defines the number of isr ticks (hardware) per controller clock tick (software)
//! \brief Controller clock tick (CTRL) is the main clock used for all timing in the software
//! \brief Typically the PWM Frequency triggers (can be decimated by the ePWM hardware for less overhead)
```

```
    an ADC SOC
//! \brief ADC SOC triggers an ADC Conversion Done
//! \brief ADC Conversion Done triggers ISR
//! \brief This relates the hardware ISR rate to the software controller rate
//! \brief Typcially want to consider some form of decimation (ePWM hardware, CURRENT or EST) over 16KHz
    ISR to insure interrupt completes and leaves time for background tasks
#define USER_NUM_ISR_TICKS_PER_CTRL_TICK       (1)      // 2 Example, controller clock rate (CTRL) runs at
     PWM / 2; ex 30 KHz PWM, 15 KHz control

//! \brief Defines the number of controller clock ticks per current controller clock tick
//! \brief Relationship of controller clock rate to current controller (FOC) rate
#define USER_NUM_CTRL_TICKS_PER_CURRENT_TICK   (1)      // 1 Typical, Forward FOC current controller (Iq/
    Id/IPARK/SVPWM) runs at same rate as CTRL.

//! \brief Defines the number of controller clock ticks per estimator clock tick
//! \brief Relationship of controller clock rate to estimator (FAST) rate
//! \brief Depends on needed dynamic performance, FAST provides very good results as low as 1 KHz while
    more dynamic or high speed applications may require up to 15 KHz
#define USER_NUM_CTRL_TICKS_PER_EST_TICK       (1)      // 1 Typical, FAST estimator runs at same rate as
    CTRL;

//! \brief Defines the number of controller clock ticks per speed controller clock tick
//! \brief Relationship of controller clock rate to speed loop rate
#define USER_NUM_CTRL_TICKS_PER_SPEED_TICK  (15)   // 15 Typical to match PWM, ex: 15KHz PWM, controller,
    and current loop, 1KHz speed loop

//! \brief Defines the number of controller clock ticks per trajectory clock tick
//! \brief Relationship of controller clock rate to trajectory loop rate
//! \brief Typically the same as the speed rate
#define USER_NUM_CTRL_TICKS_PER_TRAJ_TICK   (15)   // 15 Typical to match PWM, ex: 10KHz controller &
    current loop, 1KHz speed loop, 1 KHz Trajectory


//! \brief LIMITS
// **************************************************************************
//! \brief Defines the maximum negative current to be applied in Id reference
//! \brief Used in field weakening only, this is a safety setting (e.g. to protect against demagnetization
    )
//! \brief User must also be aware that overall current magnitude [sqrt(Id^2 + Iq^2)] should be kept below
     any machine design specifications
#define USER_MAX_NEGATIVE_ID_REF_CURRENT_A     (-0.5 * USER_MOTOR_MAX_CURRENT)   // -0.5 *
    USER_MOTOR_MAX_CURRENT Example, adjust to meet safety needs of your motor

//! \brief Defines the R/L estimation frequency, Hz
//! \brief User higher values for low inductance motors and lower values for higher inductance
//! \brief motors.  The values can range from 100 to 300 Hz.
#define USER_R_OVER_L_EST_FREQ_Hz (300)               // 300 Default

//! \brief Defines the low speed limit for the flux integrator, pu
//! \brief This is the speed range (CW/CCW) at which the ForceAngle object is active, but only if Enabled
//! \brief Outside of this speed - or if Disabled - the ForcAngle will NEVER be active and the angle is
    provided by FAST only
#define USER_ZEROSPEEDLIMIT   (0.5 / USER_IQ_FULL_SCALE_FREQ_Hz)     // 0.002 pu, 1-5 Hz typical; Hz =
    USER_ZEROSPEEDLIMIT * USER_IQ_FULL_SCALE_FREQ_Hz

//! \brief Defines the force angle frequency, Hz
//! \brief Frequency of stator vector rotation used by the ForceAngle object
//! \brief Can be positive or negative
#define USER_FORCE_ANGLE_FREQ_Hz   (2.0 * USER_ZEROSPEEDLIMIT * USER_IQ_FULL_SCALE_FREQ_Hz)      // 1.0
    Typical force angle start-up speed


//! \brief POLES
// **************************************************************************
//! \brief Defines the analog voltage filter pole location, Hz
//! \brief Must match the hardware filter for Vph
#define USER_VOLTAGE_FILTER_POLE_Hz  (364.682)   // 364.682, value for boostxldrv8301_revB hardware


//! \brief USER MOTOR & ID SETTINGS
// **************************************************************************

//! \brief Define each motor with a unique name and ID number
// BLDC & SMPM motors
#define Estun_EMJ_04APB22           101
#define Anaheim_BLY172S             102
#define Tamagawa_A0100              103
#define Teknic_M2310PLN04K          104
#define Drone_A2212_1000KV                      105
#define Drone_A2313_960KV                       106

// IPM motors
// If user provides separate Ls-d, Ls-q
// else treat as SPM with user or identified average Ls
```

```
#define Belt_Drive_Washer_IPM          201
#define Anaheim_Salient                202

// ACIM motors
#define Marathon_5K33GN2A              301

//! \brief Uncomment the motor which should be included at compile
//! \brief These motor ID settings and motor parameters are then available to be used by the control
       system
//! \brief Once your ideal settings and parameters are identified update the motor section here so it is
       available in the binary code
//#define USER_MOTOR Estun_EMJ_04APB22
#define USER_MOTOR Anaheim_BLY172S
//#define USER_MOTOR Tamagawa_A0100
//#define USER_MOTOR Drone_A2313_960KV
//#define USER_MOTOR Teknic_M2310PLN04K
//#define USER_MOTOR Belt_Drive_Washer_IPM
//#define USER_MOTOR Marathon_5K33GN2A
//#define USER_MOTOR Anaheim_Salient


#if (USER_MOTOR == Estun_EMJ_04APB22)                          // Name must match the motor #define
#define USER_MOTOR_TYPE                    MOTOR_Type_Pm       // Motor_Type_Pm (All Synchronous: BLDC, PMSM, SMPM
       , IPM) or Motor_Type_Induction (Asynchronous ACI)
#define USER_MOTOR_NUM_POLE_PAIRS          (4)                 // PAIRS, not total poles. Used to calculate user
       RPM from rotor Hz only
#define USER_MOTOR_Rr                      (NULL)              // Induction motors only, else NULL
#define USER_MOTOR_Rs                      (2.303403)          // Identified phase to neutral resistance in a Y
       equivalent circuit (Ohms, float)
#define USER_MOTOR_Ls_d                    (0.008464367)       // For PM, Identified average stator inductance (
       Henry, float)
#define USER_MOTOR_Ls_q                    (0.008464367)       // For PM, Identified average stator inductance (
       Henry, float)
#define USER_MOTOR_RATED_FLUX              (0.38)              // Identified TOTAL flux linkage between the rotor
       and the stator (V/Hz)
#define USER_MOTOR_MAGNETIZING_CURRENT     (NULL)              // Induction motors only, else NULL
#define USER_MOTOR_RES_EST_CURRENT         (1.0)               // During Motor ID, maximum current (Amperes, float
       ) used for Rs estimation, 10-20% rated current
#define USER_MOTOR_IND_EST_CURRENT         (-1.0)              // During Motor ID, maximum current (negative
       Amperes, float) used for Ls estimation, use just enough to enable rotation
#define USER_MOTOR_MAX_CURRENT             (3.82)              // CRITICAL: Used during ID and run-time, sets a
       limit on the maximum current command output of the provided Speed PI Controller to the Iq controller
#define USER_MOTOR_FLUX_EST_FREQ_Hz        (20.0)              // During Motor ID, maximum commanded speed (Hz,
       float), ~10% rated

#elif (USER_MOTOR == Anaheim_BLY172S)
#define USER_MOTOR_TYPE                    MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS          (4)
#define USER_MOTOR_Rr                      (NULL)
#define USER_MOTOR_Rs                      (0.4110007)
#define USER_MOTOR_Ls_d                    (0.0007092811)
#define USER_MOTOR_Ls_q                    (0.0007092811)
#define USER_MOTOR_RATED_FLUX              (0.03279636)
#define USER_MOTOR_MAGNETIZING_CURRENT     (NULL)
#define USER_MOTOR_RES_EST_CURRENT         (1.0)
#define USER_MOTOR_IND_EST_CURRENT         (-1.0)
#define USER_MOTOR_MAX_CURRENT             (5.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz        (20.0)
#define IPD_HFI_EXC_FREQ_HZ                (750.0)        // excitation frequency, Hz
#define IPD_HFI_LP_SPD_FILT_HZ             (35.0)         // lowpass filter cutoff frequency, Hz
#define IPD_HFI_HP_IQ_FILT_HZ              (100.0)        // highpass filter cutoff frequency, Hz
#define IPD_HFI_KSPD                       (15.0)        // the speed gain value
#define IPD_HFI_EXC_MAG_COARSE_PU          (0.13)           // coarse IPD excitation magnitude, pu
#define IPD_HFI_EXC_MAG_FINE_PU            (0.12)           // fine IPD excitation magnitude, pu
#define IPD_HFI_EXC_TIME_COARSE_S          (0.5)          // coarse wait time, sec max 0.64
#define IPD_HFI_EXC_TIME_FINE_S            (0.5)          // fine wait time, sec max 0.4
#define AFSEL_FREQ_HIGH_PU                 (_IQ(15.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_FREQ_LOW_PU                  (_IQ(10.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_IQ_SLOPE_EST                 (_IQ((float)(1.0/0.1/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_HFI                 (_IQ((float)(1.0/10.0/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_THROTTLE_DWN        (_IQ((float)(1.0/0.05/USER_ISR_FREQ_Hz)))
#define AFSEL_MAX_IQ_REF_EST               (_IQ(0.6))
#define AFSEL_MAX_IQ_REF_HFI               (_IQ(0.6))

#define USER_MOTOR_FREQ_LOW                                 (10.0)                   // Hz - suggested to set
       to 10% of rated motor frequency
#define USER_MOTOR_FREQ_HIGH                        (100.0)                   // Hz - suggested to set to 100%
       of rated motor frequency
#define USER_MOTOR_FREQ_MAX                                 (120.0)                  // Hz - suggested to set
       to 120% of rated motor frequency
#define USER_MOTOR_VOLT_MIN                                 (3.0)                    // Volt - suggested to set
        to 15% of rated motor voltage
#define USER_MOTOR_VOLT_MAX                                 (18.0)                   // Volt - suggested to set
        to 100% of rated motor voltage
```

```
#elif (USER_MOTOR == Tamagawa_A0100)
#define USER_MOTOR_TYPE                  MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS        (4)
#define USER_MOTOR_Rr                    (NULL)
#define USER_MOTOR_Rs                    (0.262230158)
#define USER_MOTOR_Ls_d                  (0.000459346687)
#define USER_MOTOR_Ls_q                  (0.000459346687)
#define USER_MOTOR_RATED_FLUX            (0.0484918393)
#define USER_MOTOR_MAGNETIZING_CURRENT   (NULL)
#define USER_MOTOR_RES_EST_CURRENT       (1.0)
#define USER_MOTOR_IND_EST_CURRENT       (-1.0)
#define USER_MOTOR_MAX_CURRENT           (5.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz      (20.0)
#define IPD_HFI_EXC_FREQ_HZ              (750.0)       // excitation frequency, Hz
#define IPD_HFI_LP_SPD_FILT_HZ           (35.0)        // lowpass filter cutoff frequency, Hz
#define IPD_HFI_HP_IQ_FILT_HZ            (100.0)       // highpass filter cutoff frequency, Hz
#define IPD_HFI_KSPD                     (15.0)       // the speed gain value
#define IPD_HFI_EXC_MAG_COARSE_PU        (0.13)         // coarse IPD excitation magnitude, pu
#define IPD_HFI_EXC_MAG_FINE_PU          (0.12)         // fine IPD excitation magnitude, pu
#define IPD_HFI_EXC_TIME_COARSE_S        (0.5)         // coarse wait time, sec max 0.64
#define IPD_HFI_EXC_TIME_FINE_S          (0.5)         // fine wait time, sec max 0.4
#define AFSEL_FREQ_HIGH_PU               (_IQ(15.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_FREQ_LOW_PU                (_IQ(10.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_IQ_SLOPE_EST               (_IQ((float)(1.0/0.1/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_HFI               (_IQ((float)(1.0/10.0/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_THROTTLE_DWN      (_IQ((float)(1.0/0.05/USER_ISR_FREQ_Hz)))
#define AFSEL_MAX_IQ_REF_EST             (_IQ(0.6))
#define AFSEL_MAX_IQ_REF_HFI             (_IQ(0.6))

#elif (USER_MOTOR == Drone_A2212_1000KV)
#define USER_MOTOR_TYPE                  MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS        (4)
#define USER_MOTOR_Rr                    (NULL)
#define USER_MOTOR_Rs                    (0.4110007)
#define USER_MOTOR_Ls_d                  (0.0007092811)
#define USER_MOTOR_Ls_q                  (0.0007092811)
#define USER_MOTOR_RATED_FLUX            (0.03279636)
#define USER_MOTOR_MAGNETIZING_CURRENT   (NULL)
#define USER_MOTOR_RES_EST_CURRENT       (1.0)
#define USER_MOTOR_IND_EST_CURRENT       (-1.0)
#define USER_MOTOR_MAX_CURRENT           (5.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz      (20.0)
#define IPD_HFI_EXC_FREQ_HZ              (750.0)       // excitation frequency, Hz
#define IPD_HFI_LP_SPD_FILT_HZ           (35.0)        // lowpass filter cutoff frequency, Hz
#define IPD_HFI_HP_IQ_FILT_HZ            (100.0)       // highpass filter cutoff frequency, Hz
#define IPD_HFI_KSPD                     (15.0)       // the speed gain value
#define IPD_HFI_EXC_MAG_COARSE_PU        (0.13)         // coarse IPD excitation magnitude, pu
#define IPD_HFI_EXC_MAG_FINE_PU          (0.12)         // fine IPD excitation magnitude, pu
#define IPD_HFI_EXC_TIME_COARSE_S        (0.5)         // coarse wait time, sec max 0.64
#define IPD_HFI_EXC_TIME_FINE_S          (0.5)         // fine wait time, sec max 0.4
#define AFSEL_FREQ_HIGH_PU               (_IQ(15.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_FREQ_LOW_PU                (_IQ(10.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_IQ_SLOPE_EST               (_IQ((float)(1.0/0.1/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_HFI               (_IQ((float)(1.0/10.0/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_THROTTLE_DWN      (_IQ((float)(1.0/0.05/USER_ISR_FREQ_Hz)))
#define AFSEL_MAX_IQ_REF_EST             (_IQ(0.6))
#define AFSEL_MAX_IQ_REF_HFI             (_IQ(0.6))

#elif (USER_MOTOR == Drone_A2313_960KV)
#define USER_MOTOR_TYPE                  MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS        (4)
#define USER_MOTOR_Rr                    (NULL)
#define USER_MOTOR_Rs                    (0.4110007)
#define USER_MOTOR_Ls_d                  (0.0007092811)
#define USER_MOTOR_Ls_q                  (0.0007092811)
#define USER_MOTOR_RATED_FLUX            (0.03279636)
#define USER_MOTOR_MAGNETIZING_CURRENT   (NULL)
#define USER_MOTOR_RES_EST_CURRENT       (1.0)
#define USER_MOTOR_IND_EST_CURRENT       (-1.0)
#define USER_MOTOR_MAX_CURRENT           (5.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz      (20.0)
#define IPD_HFI_EXC_FREQ_HZ              (750.0)       // excitation frequency, Hz
#define IPD_HFI_LP_SPD_FILT_HZ           (35.0)        // lowpass filter cutoff frequency, Hz
#define IPD_HFI_HP_IQ_FILT_HZ            (100.0)       // highpass filter cutoff frequency, Hz
#define IPD_HFI_KSPD                     (15.0)       // the speed gain value
#define IPD_HFI_EXC_MAG_COARSE_PU        (0.13)         // coarse IPD excitation magnitude, pu
#define IPD_HFI_EXC_MAG_FINE_PU          (0.12)         // fine IPD excitation magnitude, pu
#define IPD_HFI_EXC_TIME_COARSE_S        (0.5)         // coarse wait time, sec max 0.64
#define IPD_HFI_EXC_TIME_FINE_S          (0.5)         // fine wait time, sec max 0.4
#define AFSEL_FREQ_HIGH_PU               (_IQ(15.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_FREQ_LOW_PU                (_IQ(10.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_IQ_SLOPE_EST               (_IQ((float)(1.0/0.1/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_HFI               (_IQ((float)(1.0/10.0/USER_ISR_FREQ_Hz)))
```

```
#define AFSEL_IQ_SLOPE_THROTTLE_DWN      (_IQ((float)(1.0/0.05/USER_ISR_FREQ_Hz)))
#define AFSEL_MAX_IQ_REF_EST             (_IQ(0.6))
#define AFSEL_MAX_IQ_REF_HFI             (_IQ(0.6))

#elif (USER_MOTOR == Anaheim_Salient)                    // When using IPD_HFI, set decimation to 1 and PWM
      to 15.0 KHz
#define USER_MOTOR_TYPE                  MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS        (4)
#define USER_MOTOR_Rr                    (NULL)
#define USER_MOTOR_Rs                    (0.1215855)
#define USER_MOTOR_Ls_d                  (0.0002298828)
#define USER_MOTOR_Ls_q                  (0.0002298828)
#define USER_MOTOR_RATED_FLUX            (0.04821308)
#define USER_MOTOR_MAGNETIZING_CURRENT   (NULL)
#define USER_MOTOR_RES_EST_CURRENT       (2.0)          // Enter amperes(float)
#define USER_MOTOR_IND_EST_CURRENT       (-0.5)         // Enter negative amperes(float)
#define USER_MOTOR_MAX_CURRENT           (10.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz      (20.0)
#define IPD_HFI_EXC_FREQ_HZ              (750.0)        // excitation frequency, Hz
#define IPD_HFI_LP_SPD_FILT_HZ           (35.0)         // lowpass filter cutoff frequency, Hz
#define IPD_HFI_HP_IQ_FILT_HZ            (100.0)        // highpass filter cutoff frequency, Hz
#define IPD_HFI_KSPD                     (15.0)        // the speed gain value
#define IPD_HFI_EXC_MAG_COARSE_PU        (0.13)           // coarse IPD excitation magnitude, pu
#define IPD_HFI_EXC_MAG_FINE_PU          (0.12)           // fine IPD excitation magnitude, pu
#define IPD_HFI_EXC_TIME_COARSE_S        (0.5)          // coarse wait time, sec max 0.64
#define IPD_HFI_EXC_TIME_FINE_S          (0.5)          // fine wait time, sec max 0.4
#define AFSEL_FREQ_HIGH_PU               (_IQ(15.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_FREQ_LOW_PU                (_IQ(10.0 / USER_IQ_FULL_SCALE_FREQ_Hz))
#define AFSEL_IQ_SLOPE_EST               (_IQ((float)(1.0/0.1/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_HFI               (_IQ((float)(1.0/10.0/USER_ISR_FREQ_Hz)))
#define AFSEL_IQ_SLOPE_THROTTLE_DWN      (_IQ((float)(1.0/0.05/USER_ISR_FREQ_Hz)))
#define AFSEL_MAX_IQ_REF_EST             (_IQ(0.6))
#define AFSEL_MAX_IQ_REF_HFI             (_IQ(0.6))

#elif (USER_MOTOR == Teknic_M2310PLN04K)
#define USER_MOTOR_TYPE                  MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS        (4)
#define USER_MOTOR_Rr                    (NULL)
#define USER_MOTOR_Rs                    (0.3918252)
#define USER_MOTOR_Ls_d                  (0.00023495)
#define USER_MOTOR_Ls_q                  (0.00023495)
#define USER_MOTOR_RATED_FLUX            (0.03955824)
#define USER_MOTOR_MAGNETIZING_CURRENT   (NULL)
#define USER_MOTOR_RES_EST_CURRENT       (1.0)
#define USER_MOTOR_IND_EST_CURRENT       (-0.5)
#define USER_MOTOR_MAX_CURRENT           (7.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz      (20.0)

#elif (USER_MOTOR == Belt_Drive_Washer_IPM)
#define USER_MOTOR_TYPE                  MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS        (4)
#define USER_MOTOR_Rr                    (NULL)
#define USER_MOTOR_Rs                    (2.832002)
#define USER_MOTOR_Ls_d                  (0.0115)
#define USER_MOTOR_Ls_q                  (0.0135)
#define USER_MOTOR_RATED_FLUX            (0.5022156)
#define USER_MOTOR_MAGNETIZING_CURRENT   (NULL)
#define USER_MOTOR_RES_EST_CURRENT       (1.0)
#define USER_MOTOR_IND_EST_CURRENT       (-1.0)
#define USER_MOTOR_MAX_CURRENT           (4.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz      (20.0)

#elif (USER_MOTOR == Marathon_5K33GN2A)                         // Name must match the motor #define
#define USER_MOTOR_TYPE                  MOTOR_Type_Induction  // Motor_Type_Pm (All Synchronous: BLDC, PMSM
     , SMPM, IPM) or Motor_Type_Induction (Asynchronous ACI)
#define USER_MOTOR_NUM_POLE_PAIRS        (2)                   // PAIRS, not total poles. Used to calculate
     user RPM from rotor Hz only
#define USER_MOTOR_Rr                    (5.508003)            // Identified phase to neutral in a Y
     equivalent circuit (Ohms, float)
#define USER_MOTOR_Rs                    (10.71121)            // Identified phase to neutral in a Y
     equivalent circuit (Ohms, float)
#define USER_MOTOR_Ls_d                  (0.05296588)          // For Induction, Identified average stator
     inductance  (Henry, float)
#define USER_MOTOR_Ls_q                  (0.05296588)          // For Induction, Identified average stator
     inductance  (Henry, float)
#define USER_MOTOR_RATED_FLUX            (0.8165*220.0/60.0)  // sqrt(2/3)* Rated V (line-line) / Rated
     Freq (Hz)
#define USER_MOTOR_MAGNETIZING_CURRENT   (1.378)               // Identified magnetizing current for
     induction motors, else NULL
#define USER_MOTOR_RES_EST_CURRENT       (0.5)                 // During Motor ID, maximum current (Amperes,
      float) used for Rs estimation, 10-20% rated current
#define USER_MOTOR_IND_EST_CURRENT       (NULL)                // not used for induction
#define USER_MOTOR_MAX_CURRENT           (2.0)                 // CRITICAL: Used during ID and run-time,
     sets a limit on the maximum current command output of the provided Speed PI Controller to the Iq
     controller
#define USER_MOTOR_FLUX_EST_FREQ_Hz      (5.0)                  // During Motor ID, maximum commanded speed (
     Hz, float). Should always use 5 Hz for Induction.

#else
#error No motor type specified
#endif

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

//@} // ingroup
#endif // end of _USER_J1_H_ definition
```

C

proj_lab11a.c

```c
/* --COPYRIGHT--,BSD
 * Copyright (c) 2015, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * --/COPYRIGHT--*/
//! \file   solutions/instaspin_foc/src/proj_lab011a.c
//! \brief A Feature Rich Example without Controller Module
//!
//! (C) Copyright 2015, Texas Instruments, Inc.

//! \defgroup PROJ_LAB11A PROJ_LAB11A
//@{

//! \defgroup PROJ_LAB11A_OVERVIEW Project Overview
//!
//! A Feature Rich Example without Controller Module
//!

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

// system includes
#include <math.h>
#include "main.h"

#ifdef FLASH
#pragma CODE_SECTION(mainISR,"ramfuncs");
#pragma CODE_SECTION(runSetTrigger,"ramfuncs");
#pragma CODE_SECTION(runFieldWeakening,"ramfuncs");
#pragma CODE_SECTION(runCurrentReconstruction,"ramfuncs");
#pragma CODE_SECTION(angleDelayComp,"ramfuncs");
#endif
```

```
// Include header files used in the main function

// **************************************************************************
// the defines

// **************************************************************************
// the globals

CLARKE_Handle   clarkeHandle_I;               //!< the handle for the current Clarke transform
CLARKE_Obj      clarke_I;                     //!< the current Clarke transform object

CLARKE_Handle   clarkeHandle_V;               //!< the handle for the voltage Clarke transform
CLARKE_Obj      clarke_V;                     //!< the voltage Clarke transform object

CPU_USAGE_Handle  cpu_usageHandle;
CPU_USAGE_Obj     cpu_usage;

EST_Handle      estHandle;                    //!< the handle for the estimator

FW_Handle       fwHandle;
FW_Obj          fw;

PID_Obj         pid[3];                       //!< three handles for PID controllers 0 - Speed, 1 - Id, 2
     - Iq
PID_Handle      pidHandle[3];                 //!< three objects for PID controllers 0 - Speed, 1 - Id, 2
     - Iq
uint16_t        pidCntSpeed;                  //!< count variable to decimate the execution of the speed
     PID controller

IPARK_Handle    iparkHandle;                  //!< the handle for the inverse Park transform
IPARK_Obj       ipark;                        //!< the inverse Park transform object

FILTER_FO_Handle  filterHandle[6];            //!< the handles for the 3-current and 3-voltage filters for
      offset calculation
FILTER_FO_Obj     filter[6];                  //!< the 3-current and 3-voltage filters for offset
     calculation

SVGENCURRENT_Obj      svgencurrent;
SVGENCURRENT_Handle   svgencurrentHandle;

SVGEN_Handle    svgenHandle;                  //!< the handle for the space vector generator
SVGEN_Obj       svgen;                        //!< the space vector generator object

TRAJ_Handle     trajHandle_Id;                //!< the handle for the id reference trajectory
TRAJ_Obj        traj_Id;                      //!< the id reference trajectory object

TRAJ_Handle     trajHandle_spd;               //!< the handle for the speed reference trajectory
TRAJ_Obj        traj_spd;                     //!< the speed reference trajectory object

#ifdef CSM_ENABLE
#pragma DATA_SECTION(halHandle,"rom_accessed_data");
#endif
HAL_Handle      halHandle;                    //!< the handle for the hardware abstraction layer (HAL)

HAL_PwmData_t   gPwmData = {_IQ(0.0), _IQ(0.0), _IQ(0.0)};      //!< contains the three pwm values -1.0 -
      0%, 1.0 - 100%

HAL_AdcData_t   gAdcData;                     //!< contains three current values, three voltage values and
      one DC buss value

MATH_vec3       gOffsets_I_pu = {_IQ(0.0), _IQ(0.0), _IQ(0.0)};  //!< contains the offsets for the current
      feedback

MATH_vec3       gOffsets_V_pu = {_IQ(0.0), _IQ(0.0), _IQ(0.0)};  //!< contains the offsets for the voltage
      feedback

MATH_vec2       gIdq_ref_pu = {_IQ(0.0), _IQ(0.0)};             //!< contains the Id and Iq references

MATH_vec2       gVdq_out_pu = {_IQ(0.0), _IQ(0.0)};             //!< contains the output Vd and Vq from
     the current controllers

MATH_vec2       gIdq_pu = {_IQ(0.0), _IQ(0.0)};                 //!< contains the Id and Iq measured
     values

#ifdef CSM_ENABLE
#pragma DATA_SECTION(gUserParams,"rom_accessed_data");
#endif
USER_Params     gUserParams;

volatile MOTOR_Vars_t gMotorVars = MOTOR_Vars_INIT;   //!< the global motor variables that are defined in
     main.h and used for display in the debugger's watch window

#ifdef FLASH
// Used for running BackGround in flash, and ISR in RAM
```

```
extern uint16_t *RamfuncsLoadStart, *RamfuncsLoadEnd, *RamfuncsRunStart;

#ifdef CSM_ENABLE
extern uint16_t *econst_start, *econst_end, *econst_ram_load;
extern uint16_t *switch_start, *switch_end, *switch_ram_load;
#endif
#endif


MATH_vec3 gIavg = {_IQ(0.0), _IQ(0.0), _IQ(0.0)};
uint16_t gIavg_shift = 1;
MATH_vec3         gPwmData_prev = {_IQ(0.0), _IQ(0.0), _IQ(0.0)};

#ifdef DRV8301_SPI
// Watch window interface to the 8301 SPI
DRV_SPI_8301_Vars_t gDrvSpi8301Vars;
#endif

#ifdef DRV8305_SPI
// Watch window interface to the 8305 SPI
DRV_SPI_8305_Vars_t gDrvSpi8305Vars;
#endif

_iq gFlux_pu_to_Wb_sf;

_iq gFlux_pu_to_VpHz_sf;

_iq gTorque_Ls_Id_Iq_pu_to_Nm_sf;

_iq gTorque_Flux_Iq_pu_to_Nm_sf;

_iq gSpeed_krpm_to_pu_sf = _IQ((float_t)USER_MOTOR_NUM_POLE_PAIRS * 1000.0 / (USER_IQ_FULL_SCALE_FREQ_Hz *
    60.0));

_iq gSpeed_hz_to_krpm_sf = _IQ(60.0 / (float_t)USER_MOTOR_NUM_POLE_PAIRS / 1000.0);

_iq gIs_Max_squared_pu = _IQ((USER_MOTOR_MAX_CURRENT * USER_MOTOR_MAX_CURRENT) / (
    USER_IQ_FULL_SCALE_CURRENT_A * USER_IQ_FULL_SCALE_CURRENT_A));

float_t gCpuUsagePercentageMin = 0.0;
float_t gCpuUsagePercentageAvg = 0.0;
float_t gCpuUsagePercentageMax = 0.0;

uint32_t gOffsetCalcCount = 0;

uint16_t gTrjCnt = 0;

volatile bool gFlag_enableRsOnLine = false;

volatile bool gFlag_updateRs = false;

volatile _iq gRsOnLineFreq_Hz = _IQ(0.2);

volatile _iq gRsOnLineId_mag_A = _IQ(0.5);

volatile _iq gRsOnLinePole_Hz = _IQ(0.2);

// **************************************************************************
// the functions
void main(void)
{
  // Only used if running from FLASH
  // Note that the variable FLASH is defined by the project
  #ifdef FLASH
  // Copy time critical code and Flash setup code to RAM
  // The RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
  // symbols are created by the linker. Refer to the linker files.
  memCopy((uint16_t *)&RamfuncsLoadStart,(uint16_t *)&RamfuncsLoadEnd,(uint16_t *)&RamfuncsRunStart);

  #ifdef CSM_ENABLE
  //copy .econst to unsecure RAM
  if(*econst_end - *econst_start)
        {
          memCopy((uint16_t *)&econst_start,(uint16_t *)&econst_end,(uint16_t *)&econst_ram_load);
        }

  //copy .switch ot unsecure RAM
  if(*switch_end - *switch_start)
        {
          memCopy((uint16_t *)&switch_start,(uint16_t *)&switch_end,(uint16_t *)&switch_ram_load);
        }
  #endif
  #endif
  // initialize the hardware abstraction layer
  halHandle = HAL_init(&hal,sizeof(hal));

  // check for errors in user parameters
  USER_checkForErrors(&gUserParams);


  // store user parameter error in global variable
  gMotorVars.UserErrorCode = USER_getErrorCode(&gUserParams);


  // do not allow code execution if there is a user parameter error
  if(gMotorVars.UserErrorCode != USER_ErrorCode_NoError)
    {
      for(;;)
        {
          gMotorVars.Flag_enableSys = false;
        }
    }

  // initialize the Clarke modules
  clarkeHandle_I = CLARKE_init(&clarke_I,sizeof(clarke_I));
  clarkeHandle_V = CLARKE_init(&clarke_V,sizeof(clarke_V));

  // initialize the estimator
  estHandle = EST_init((void *)USER_EST_HANDLE_ADDRESS, 0x200);

  // initialize the user parameters
  USER_setParams(&gUserParams);

  // set the hardware abstraction layer parameters
  HAL_setParams(halHandle,&gUserParams);

#ifdef FAST_ROM_V1p6
  {
    CTRL_Handle ctrlHandle = CTRL_init((void *)USER_CTRL_HANDLE_ADDRESS, 0x200);
    CTRL_Obj *obj = (CTRL_Obj *)ctrlHandle;
    obj->estHandle = estHandle;

    // initialize the estimator through the controller
    CTRL_setParams(ctrlHandle,&gUserParams);
    CTRL_setUserMotorParams(ctrlHandle);
    CTRL_setupEstIdleState(ctrlHandle);
  }
#else
  {
    // initialize the estimator
    EST_setEstParams(estHandle,&gUserParams);
    EST_setupEstIdleState(estHandle);
  }
#endif

  // disable Rs recalculation by default
  gMotorVars.Flag_enableRsRecalc = true;
  EST_setFlag_enableRsRecalc(estHandle,false);

  // configure RsOnLine
  EST_setFlag_enableRsOnLine(estHandle,gFlag_enableRsOnLine);
  EST_setFlag_updateRs(estHandle,gFlag_updateRs);
  EST_setRsOnLineAngleDelta_pu(estHandle,_IQmpy(gRsOnLineFreq_Hz, _IQ(1.0/USER_ISR_FREQ_Hz)));
  EST_setRsOnLineId_mag_pu(estHandle,_IQmpy(gRsOnLineId_mag_A, _IQ(1.0/USER_IQ_FULL_SCALE_CURRENT_A)));

  // Calculate coefficients for all filters
  {
    _iq b0 = _IQmpy(gRsOnLinePole_Hz, _IQ(1.0/USER_ISR_FREQ_Hz));
    _iq a1 = b0 - _IQ(1.0);
    EST_setRsOnLineFilterParams(estHandle,EST_RsOnLineFilterType_Current,b0,a1,_IQ(0.0),b0,a1,_IQ(0.0));
    EST_setRsOnLineFilterParams(estHandle,EST_RsOnLineFilterType_Voltage,b0,a1,_IQ(0.0),b0,a1,_IQ(0.0));
  }

  // set the number of current sensors
  setupClarke_I(clarkeHandle_I,USER_NUM_CURRENT_SENSORS);

  // set the number of voltage sensors
  setupClarke_V(clarkeHandle_V,USER_NUM_VOLTAGE_SENSORS);

  // set the pre-determined current and voltage feeback offset values
  gOffsets_I_pu.value[0] = _IQ(I_A_offset);
  gOffsets_I_pu.value[1] = _IQ(I_B_offset);
  gOffsets_I_pu.value[2] = _IQ(I_C_offset);
  gOffsets_V_pu.value[0] = _IQ(V_A_offset);
  gOffsets_V_pu.value[1] = _IQ(V_B_offset);
  gOffsets_V_pu.value[2] = _IQ(V_C_offset);
```

```
// initialize the PID controllers
{
  _iq maxCurrent_pu = _IQ(USER_MOTOR_MAX_CURRENT / USER_IQ_FULL_SCALE_CURRENT_A);
  _iq maxVoltage_pu = _IQ(USER_MAX_VS_MAG_PU * USER_VD_SF);
  float_t fullScaleCurrent = USER_IQ_FULL_SCALE_CURRENT_A;
  float_t fullScaleVoltage = USER_IQ_FULL_SCALE_VOLTAGE_V;
  float_t IsrPeriod_sec = 1.0 / USER_ISR_FREQ_Hz;
  float_t Ls_d = USER_MOTOR_Ls_d;
  float_t Ls_q = USER_MOTOR_Ls_q;
  float_t Rs = USER_MOTOR_Rs;
  float_t RoverLs_d = Rs/Ls_d;
  float_t RoverLs_q = Rs/Ls_q;
  _iq Kp_Id = _IQ((0.25*Ls_d*fullScaleCurrent)/(IsrPeriod_sec*fullScaleVoltage));
  _iq Ki_Id = _IQ(RoverLs_d*IsrPeriod_sec);
  _iq Kp_Iq = _IQ((0.25*Ls_q*fullScaleCurrent)/(IsrPeriod_sec*fullScaleVoltage));
  _iq Ki_Iq = _IQ(RoverLs_q*IsrPeriod_sec);

  pidHandle[0] = PID_init(&pid[0],sizeof(pid[0]));
  pidHandle[1] = PID_init(&pid[1],sizeof(pid[1]));
  pidHandle[2] = PID_init(&pid[2],sizeof(pid[2]));

  PID_setGains(pidHandle[0],_IQ(1.0),_IQ(0.01),_IQ(0.0));
  PID_setMinMax(pidHandle[0],-maxCurrent_pu,maxCurrent_pu);
  PID_setUi(pidHandle[0],_IQ(0.0));
  pidCntSpeed = 0;

  PID_setGains(pidHandle[1],Kp_Id,Ki_Id,_IQ(0.0));
  PID_setMinMax(pidHandle[1],-maxVoltage_pu,maxVoltage_pu);
  PID_setUi(pidHandle[1],_IQ(0.0));

  PID_setGains(pidHandle[2],Kp_Iq,Ki_Iq,_IQ(0.0));
  PID_setMinMax(pidHandle[2],_IQ(0.0),_IQ(0.0));
  PID_setUi(pidHandle[2],_IQ(0.0));
}

// initialize the speed reference in kilo RPM where base speed is USER_IQ_FULL_SCALE_FREQ_Hz
gMotorVars.SpeedRef_krpm = _IQmpy(_IQ(10.0), gSpeed_hz_to_krpm_sf);

// initialize the inverse Park module
iparkHandle = IPARK_init(&ipark,sizeof(ipark));

// initialize and configure offsets using filters
{
  uint16_t cnt = 0;
  _iq b0 = _IQ(gUserParams.offsetPole_rps/(float_t)gUserParams.ctrlFreq_Hz);
  _iq a1 = (b0 - _IQ(1.0));
  _iq b1 = _IQ(0.0);

  for(cnt=0;cnt<6;cnt++)
    {
      filterHandle[cnt] = FILTER_FO_init(&filter[cnt],sizeof(filter[0]));
      FILTER_FO_setDenCoeffs(filterHandle[cnt],a1);
      FILTER_FO_setNumCoeffs(filterHandle[cnt],b0,b1);
      FILTER_FO_setInitialConditions(filterHandle[cnt],_IQ(0.0),_IQ(0.0));
    }

  gMotorVars.Flag_enableOffsetcalc = false;
}

// initialize the space vector generator module
svgenHandle = SVGEN_init(&svgen,sizeof(svgen));

// Initialize and setup the 100% SVM generator
svgencurrentHandle = SVGENCURRENT_init(&svgencurrent,sizeof(svgencurrent));

// setup svgen current
{
  float_t minWidth_microseconds = 2.0;
  uint16_t minWidth_counts = (uint16_t)(minWidth_microseconds * USER_SYSTEM_FREQ_MHz);
  float_t fdutyLimit = 0.5-(2.0*minWidth_microseconds*USER_PWM_FREQ_kHz*0.001);
  _iq dutyLimit = _IQ(fdutyLimit);

  SVGENCURRENT_setMinWidth(svgencurrentHandle, minWidth_counts);
  SVGENCURRENT_setIgnoreShunt(svgencurrentHandle, use_all);
  SVGENCURRENT_setMode(svgencurrentHandle,all_phase_measurable);
  SVGENCURRENT_setVlimit(svgencurrentHandle,dutyLimit);
}

// set overmodulation to maximum value
gMotorVars.OverModulation = _IQ(USER_MAX_VS_MAG_PU);
// initialize the speed reference trajectory
trajHandle_spd = TRAJ_init(&traj_spd,sizeof(traj_spd));

// configure the speed reference trajectory
```

```
  TRAJ_setTargetValue(trajHandle_spd,_IQ(0.0));
  TRAJ_setIntValue(trajHandle_spd,_IQ(0.0));
  TRAJ_setMinValue(trajHandle_spd,_IQ(-1.0));
  TRAJ_setMaxValue(trajHandle_spd,_IQ(1.0));
  TRAJ_setMaxDelta(trajHandle_spd,_IQ(USER_MAX_ACCEL_Hzps / USER_IQ_FULL_SCALE_FREQ_Hz / USER_ISR_FREQ_Hz)
      );

  // initialize the Id reference trajectory
  trajHandle_Id = TRAJ_init(&traj_Id,sizeof(traj_Id));

  if(USER_MOTOR_TYPE == MOTOR_Type_Pm)
    {
      // configure the Id reference trajectory
      TRAJ_setTargetValue(trajHandle_Id,_IQ(0.0));
      TRAJ_setIntValue(trajHandle_Id,_IQ(0.0));
      TRAJ_setMinValue(trajHandle_Id,_IQ(-USER_MOTOR_MAX_CURRENT / USER_IQ_FULL_SCALE_CURRENT_A));
      TRAJ_setMaxValue(trajHandle_Id,_IQ(USER_MOTOR_MAX_CURRENT / USER_IQ_FULL_SCALE_CURRENT_A));
      TRAJ_setMaxDelta(trajHandle_Id,_IQ(USER_MOTOR_RES_EST_CURRENT / USER_IQ_FULL_SCALE_CURRENT_A /
          USER_ISR_FREQ_Hz));

      // Initialize field weakening
      fwHandle = FW_init(&fw,sizeof(fw));
      FW_setFlag_enableFw(fwHandle, false); // Disable field weakening
      FW_clearCounter(fwHandle); // Clear field weakening counter
      FW_setNumIsrTicksPerFwTick(fwHandle, FW_NUM_ISR_TICKS_PER_CTRL_TICK); // Set the number of ISR per
          field weakening ticks
      FW_setDeltas(fwHandle, FW_INC_DELTA, FW_DEC_DELTA); // Set the deltas of field weakening
      FW_setOutput(fwHandle, _IQ(0.0)); // Set initial output of field weakening to zero
      FW_setMinMax(fwHandle,_IQ(USER_MAX_NEGATIVE_ID_REF_CURRENT_A/USER_IQ_FULL_SCALE_CURRENT_A),_IQ(0.0))
          ; // Set the field weakening controller limits
    }
  else
    {
      // configure the Id reference trajectory
      TRAJ_setTargetValue(trajHandle_Id,_IQ(0.0));
      TRAJ_setIntValue(trajHandle_Id,_IQ(0.0));
      TRAJ_setMinValue(trajHandle_Id,_IQ(0.0));
      TRAJ_setMaxValue(trajHandle_Id,_IQ(USER_MOTOR_MAGNETIZING_CURRENT / USER_IQ_FULL_SCALE_CURRENT_A));
      TRAJ_setMaxDelta(trajHandle_Id,_IQ(USER_MOTOR_MAGNETIZING_CURRENT / USER_IQ_FULL_SCALE_CURRENT_A /
          USER_ISR_FREQ_Hz));
    }

  // initialize the CPU usage module
  cpu_usageHandle = CPU_USAGE_init(&cpu_usage,sizeof(cpu_usage));
  CPU_USAGE_setParams(cpu_usageHandle,
                                          HAL_getTimerPeriod(halHandle,1),     // timer period, cnts
                     (uint32_t)USER_ISR_FREQ_Hz);                   // average over 1 second of ISRs

  // setup faults
  HAL_setupFaults(halHandle);

  // initialize the interrupt vector table
  HAL_initIntVectorTable(halHandle);

  // enable the ADC interrupts
  HAL_enableAdcInts(halHandle);

  // enable global interrupts
  HAL_enableGlobalInts(halHandle);

  // enable debug interrupts
  HAL_enableDebugInt(halHandle);

  // disable the PWM
  HAL_disablePwm(halHandle);

  // compute scaling factors for flux and torque calculations
  gFlux_pu_to_Wb_sf = USER_computeFlux_pu_to_Wb_sf();
  gFlux_pu_to_VpHz_sf = USER_computeFlux_pu_to_VpHz_sf();
  gTorque_Ls_Id_Iq_pu_to_Nm_sf = USER_computeTorque_Ls_Id_Iq_pu_to_Nm_sf();
  gTorque_Flux_Iq_pu_to_Nm_sf = USER_computeTorque_Flux_Iq_pu_to_Nm_sf();

  // enable the system by default
  gMotorVars.Flag_enableSys = true;

#ifdef DRV8301_SPI
  // turn on the DRV8301 if present
  HAL_enableDrv(halHandle);
  // initialize the DRV8301 interface
  HAL_setupDrvSpi(halHandle,&gDrvSpi8301Vars);
#endif

#ifdef DRV8305_SPI
  // turn on the DRV8305 if present
  HAL_enableDrv(halHandle);
  // initialize the DRV8305 interface
  HAL_setupDrvSpi(halHandle,&gDrvSpi8305Vars);
#endif

  // Begin the background loop
  for(;;)
  {
    // Waiting for enable system flag to be set
    while(!(gMotorVars.Flag_enableSys));

    // loop while the enable system flag is true
    while(gMotorVars.Flag_enableSys)
      {
        if(gMotorVars.Flag_Run_Identify)
          {
            // disable Rs recalculation
            EST_setFlag_enableRsRecalc(estHandle,false);

            // update estimator state
            EST_updateState(estHandle,0);

            #ifdef FAST_ROM_V1p6
              // call this function to fix 1p6
              softwareUpdate1p6(estHandle);
            #endif

            // enable the PWM
            HAL_enablePwm(halHandle);

            // set trajectory target for speed reference
            TRAJ_setTargetValue(trajHandle_spd,_IQmpy(gMotorVars.SpeedRef_krpm, gSpeed_krpm_to_pu_sf));

            if(USER_MOTOR_TYPE == MOTOR_Type_Pm)
              {
                // set trajectory target for Id reference
                TRAJ_setTargetValue(trajHandle_Id,gIdq_ref_pu.value[0]);
              }
            else
              {
                if(gMotorVars.Flag_enablePowerWarp)
                  {
                    _iq Id_target_pw_pu = EST_runPowerWarp(estHandle,TRAJ_getIntValue(trajHandle_Id),
                         gIdq_pu.value[1]);
                    TRAJ_setTargetValue(trajHandle_Id,Id_target_pw_pu);
                    TRAJ_setMinValue(trajHandle_Id,_IQ(USER_MOTOR_MAGNETIZING_CURRENT * 0.3 /
                         USER_IQ_FULL_SCALE_CURRENT_A));
                    TRAJ_setMaxDelta(trajHandle_Id,_IQ(USER_MOTOR_MAGNETIZING_CURRENT * 0.3 /
                         USER_IQ_FULL_SCALE_CURRENT_A / USER_ISR_FREQ_Hz));
                  }
                else
                  {
                    // set trajectory target for Id reference
                    TRAJ_setTargetValue(trajHandle_Id,_IQ(USER_MOTOR_MAGNETIZING_CURRENT /
                         USER_IQ_FULL_SCALE_CURRENT_A));
                    TRAJ_setMinValue(trajHandle_Id,_IQ(0.0));
                    TRAJ_setMaxDelta(trajHandle_Id,_IQ(USER_MOTOR_MAGNETIZING_CURRENT /
                         USER_IQ_FULL_SCALE_CURRENT_A / USER_ISR_FREQ_Hz));
                  }
              }
          }
        else if(gMotorVars.Flag_enableRsRecalc)
          {
            // set angle to zero
            EST_setAngle_pu(estHandle,_IQ(0.0));

            // enable or disable Rs recalculation
            EST_setFlag_enableRsRecalc(estHandle,true);

            // update estimator state
            EST_updateState(estHandle,0);

            #ifdef FAST_ROM_V1p6
              // call this function to fix 1p6
              softwareUpdate1p6(estHandle);
            #endif

            // enable the PWM
            HAL_enablePwm(halHandle);

            // set trajectory target for speed reference
            TRAJ_setTargetValue(trajHandle_spd,_IQ(0.0));

            // set trajectory target for Id reference
            TRAJ_setTargetValue(trajHandle_Id,_IQ(USER_MOTOR_RES_EST_CURRENT /
                 USER_IQ_FULL_SCALE_CURRENT_A));

            // if done with Rs recalculation, disable flag
            if(EST_getState(estHandle) == EST_State_OnLine) gMotorVars.Flag_enableRsRecalc = false;
          }
        else
          {
            // set estimator to Idle
            EST_setIdle(estHandle);

            // disable the PWM
            if(!gMotorVars.Flag_enableOffsetcalc) HAL_disablePwm(halHandle);

            // clear the speed reference trajectory
            TRAJ_setTargetValue(trajHandle_spd,_IQ(0.0));
            TRAJ_setIntValue(trajHandle_spd,_IQ(0.0));

            // clear the Id reference trajectory
            TRAJ_setTargetValue(trajHandle_Id,_IQ(0.0));
            TRAJ_setIntValue(trajHandle_Id,_IQ(0.0));

            // configure trajectory Id defaults depending on motor type
            if(USER_MOTOR_TYPE == MOTOR_Type_Pm)
              {
                TRAJ_setMinValue(trajHandle_Id,_IQ(-USER_MOTOR_MAX_CURRENT / USER_IQ_FULL_SCALE_CURRENT_A)
                     );
                TRAJ_setMaxDelta(trajHandle_Id,_IQ(USER_MOTOR_RES_EST_CURRENT /
                     USER_IQ_FULL_SCALE_CURRENT_A / USER_ISR_FREQ_Hz));
              }
            else
              {
                TRAJ_setMinValue(trajHandle_Id,_IQ(0.0));
                TRAJ_setMaxDelta(trajHandle_Id,_IQ(USER_MOTOR_MAGNETIZING_CURRENT /
                     USER_IQ_FULL_SCALE_CURRENT_A / USER_ISR_FREQ_Hz));
              }

            // clear integral outputs
            PID_setUi(pidHandle[0],_IQ(0.0));
            PID_setUi(pidHandle[1],_IQ(0.0));
            PID_setUi(pidHandle[2],_IQ(0.0));

            // clear Id and Iq references
            gIdq_ref_pu.value[0] = _IQ(0.0);
            gIdq_ref_pu.value[1] = _IQ(0.0);

            // disable RsOnLine flags
            gFlag_enableRsOnLine = false;
            gFlag_updateRs = false;

            // disable PowerWarp flag
            gMotorVars.Flag_enablePowerWarp = false;
          }

        // update the global variables
        updateGlobalVariables(estHandle);

        // set field weakening enable flag depending on user's input
        FW_setFlag_enableFw(fwHandle,gMotorVars.Flag_enableFieldWeakening);

                // set the speed acceleration
                TRAJ_setMaxDelta(trajHandle_spd,_IQmpy(MAX_ACCEL_KRPMPS_SF,gMotorVars.MaxAccel_krpmps));

        // update CPU usage
        updateCPUusage();

        updateRsOnLine(estHandle);

        // enable/disable the forced angle
        EST_setFlag_enableForceAngle(estHandle,gMotorVars.Flag_enableForceAngle);

#ifdef DRV8301_SPI
        HAL_writeDrvData(halHandle,&gDrvSpi8301Vars);

        HAL_readDrvData(halHandle,&gDrvSpi8301Vars);
#endif
#ifdef DRV8305_SPI
        HAL_writeDrvData(halHandle,&gDrvSpi8305Vars);

        HAL_readDrvData(halHandle,&gDrvSpi8305Vars);
#endif

      } // end of while(gFlag_enableSys) loop
```

```
    // disable the PWM
    HAL_disablePwm(halHandle);

    gMotorVars.Flag_Run_Identify = false;
  } // end of for(;;) loop
} // end of main() function


//! \brief     The main ISR that implements the motor control.
interrupt void mainISR(void)
{
  _iq angle_pu = _IQ(0.0);
  _iq speed_pu = _IQ(0.0);
  _iq oneOverDcBus;
  MATH_vec2 Iab_pu;
  MATH_vec2 Vab_pu;
  MATH_vec2 phasor;
  uint32_t timer1Cnt;

  // read the timer 1 value and update the CPU usage module
  timer1Cnt = HAL_readTimerCnt(halHandle,1);
  CPU_USAGE_updateCnts(cpu_usageHandle,timer1Cnt);

  // acknowledge the ADC interrupt
  HAL_acqAdcInt(halHandle,ADC_IntNumber_1);

  // convert the ADC data
  HAL_readAdcDataWithOffsets(halHandle,&gAdcData);

  // remove offsets
  gAdcData.I.value[0] = gAdcData.I.value[0] - gOffsets_I_pu.value[0];
  gAdcData.I.value[1] = gAdcData.I.value[1] - gOffsets_I_pu.value[1];
  gAdcData.I.value[2] = gAdcData.I.value[2] - gOffsets_I_pu.value[2];
  gAdcData.V.value[0] = gAdcData.V.value[0] - gOffsets_V_pu.value[0];
  gAdcData.V.value[1] = gAdcData.V.value[1] - gOffsets_V_pu.value[1];
  gAdcData.V.value[2] = gAdcData.V.value[2] - gOffsets_V_pu.value[2];

  // run the current reconstruction algorithm
  runCurrentReconstruction();

  // run Clarke transform on current
  CLARKE_run(clarkeHandle_I,&gAdcData.I,&Iab_pu);

  // run Clarke transform on voltage
  CLARKE_run(clarkeHandle_V,&gAdcData.V,&Vab_pu);

  // run a trajectory for Id reference, so the reference changes with a ramp instead of a step
  TRAJ_run(trajHandle_Id);

  // run the estimator
  EST_run(estHandle,
          &Iab_pu,
          &Vab_pu,
          gAdcData.dcBus,
          TRAJ_getIntValue(trajHandle_spd));

  // generate the motor electrical angle
  angle_pu = EST_getAngle_pu(estHandle);
  speed_pu = EST_getFm_pu(estHandle);

  // get Idq from estimator to avoid sin and cos
  EST_getIdq_pu(estHandle,&gIdq_pu);

  // run the appropriate controller
  if((gMotorVars.Flag_Run_Identify) || (gMotorVars.Flag_enableRsRecalc))
    {
      _iq refValue;
      _iq fbackValue;
      _iq outMax_pu;

      // when appropriate, run the PID speed controller
      if((pidCntSpeed++ >= USER_NUM_CTRL_TICKS_PER_SPEED_TICK) && (!gMotorVars.Flag_enableRsRecalc))
        {
          // calculate Id reference squared
          _iq Id_ref_squared_pu = _IQmpy(PID_getRefValue(pidHandle[1]),PID_getRefValue(pidHandle[1]));

          // Take into consideration that Iq^2+Id^2 = Is^2
          _iq Iq_Max_pu = _IQsqrt(gIs_Max_squared_pu - Id_ref_squared_pu);

          // clear counter
          pidCntSpeed = 0;

          // Set new min and max for the speed controller output
          PID_setMinMax(pidHandle[0], -Iq_Max_pu, Iq_Max_pu);
          // run speed controller
          PID_run_spd(pidHandle[0],TRAJ_getIntValue(trajHandle_spd),speed_pu,&(gIdq_ref_pu.value[1]));
        }

      // get the reference value from the trajectory module
      refValue = TRAJ_getIntValue(trajHandle_Id) + EST_getRsOnLineId_pu(estHandle);

      // get the feedback value
      fbackValue = gIdq_pu.value[0];

      // run the Id PID controller
      PID_run(pidHandle[1],refValue,fbackValue,&(gVdq_out_pu.value[0]));

      // set Iq reference to zero when doing Rs recalculation
      if(gMotorVars.Flag_enableRsRecalc) gIdq_ref_pu.value[1] = _IQ(0.0);

      // get the Iq reference value
      refValue = gIdq_ref_pu.value[1];

      // get the feedback value
      fbackValue = gIdq_pu.value[1];

      // calculate Iq controller limits, and run Iq controller
      _iq max_vs = _IQmpy(gMotorVars.OverModulation,EST_getDcBus_pu(estHandle));
      outMax_pu = _IQsqrt(_IQmpy(max_vs,max_vs) - _IQmpy(gVdq_out_pu.value[0],gVdq_out_pu.value[0]));
      PID_setMinMax(pidHandle[2],-outMax_pu,outMax_pu);
      PID_run(pidHandle[2],refValue,fbackValue,&(gVdq_out_pu.value[1]));

      // compensate angle for PWM delay
      angle_pu = angleDelayComp(speed_pu, angle_pu);

      // compute the sin/cos phasor
      phasor.value[0] = _IQcosPU(angle_pu);
      phasor.value[1] = _IQsinPU(angle_pu);

      // set the phasor in the inverse Park transform
      IPARK_setPhasor(iparkHandle,&phasor);

      // run the inverse Park module
      IPARK_run(iparkHandle,&gVdq_out_pu,&Vab_pu);

      // run the space Vector Generator (SVGEN) module
      oneOverDcBus = EST_getOneOverDcBus_pu(estHandle);
      Vab_pu.value[0] = _IQmpy(Vab_pu.value[0],oneOverDcBus);
      Vab_pu.value[1] = _IQmpy(Vab_pu.value[1],oneOverDcBus);
      SVGEN_run(svgenHandle,&Vab_pu,&(gPwmData.Tabc));

      // run the PWM compensation and current ignore algorithm
      SVGENCURRENT_compPwmData(svgencurrentHandle,&(gPwmData.Tabc),&gPwmData_prev);

      gTrjCnt++;
    }
  else if(gMotorVars.Flag_enableOffsetcalc == true)
    {
      runOffsetsCalculation();
    }
  else
    {
      // disable the PWM
      HAL_disablePwm(halHandle);

      // Set the PWMs to 50% duty cycle
      gPwmData.Tabc.value[0] = _IQ(0.0);
      gPwmData.Tabc.value[1] = _IQ(0.0);
      gPwmData.Tabc.value[2] = _IQ(0.0);
    }

  // write the PWM compare values
  HAL_writePwmData(halHandle,&gPwmData);

  if(gTrjCnt >= gUserParams.numCtrlTicksPerTrajTick)
  {
          // clear counter
          gTrjCnt = 0;

          // run a trajectory for speed reference, so the reference changes with a ramp instead of a step
          TRAJ_run(trajHandle_spd);
  }

  // run function to set next trigger
  if(!gMotorVars.Flag_enableRsRecalc) runSetTrigger();

  // run field weakening
  if(USER_MOTOR_TYPE == MOTOR_Type_Pm) runFieldWeakening();

  // read the timer 1 value and update the CPU usage module
  timer1Cnt = HAL_readTimerCnt(halHandle,1);
  CPU_USAGE_updateCnts(cpu_usageHandle,timer1Cnt);

  // run the CPU usage module
  CPU_USAGE_run(cpu_usageHandle);

  return;
} // end of mainISR() function


_iq angleDelayComp(const _iq fm_pu, const _iq angleUncomp_pu)
{
  _iq angleDelta_pu = _IQmpy(fm_pu,_IQ(USER_IQ_FULL_SCALE_FREQ_Hz/(USER_PWM_FREQ_kHz*1000.0)));
  _iq angleCompFactor = _IQ(1.0 + (float_t)USER_NUM_PWM_TICKS_PER_ISR_TICK * 0.5);
  _iq angleDeltaComp_pu = _IQmpy(angleDelta_pu, angleCompFactor);
  uint32_t angleMask = ((uint32_t)0xFFFFFFFF >> (32 - GLOBAL_Q));
  _iq angleComp_pu;
  _iq angleTmp_pu;

  // increment the angle
  angleTmp_pu = angleUncomp_pu + angleDeltaComp_pu;

  // mask the angle for wrap around
  // note: must account for the sign of the angle
  angleComp_pu = _IQabs(angleTmp_pu) & angleMask;

  // account for sign
  if(angleTmp_pu < _IQ(0.0))
    {
      angleComp_pu = -angleComp_pu;
    }

  return(angleComp_pu);
} // end of angleDelayComp() function


void runCurrentReconstruction(void)
{
  SVGENCURRENT_MeasureShunt_e measurableShuntThisCycle = SVGENCURRENT_getMode(svgencurrentHandle);

  // run the current reconstruction algorithm
  SVGENCURRENT_RunRegenCurrent(svgencurrentHandle, (MATH_vec3 *)(gAdcData.I.value));

  gIavg.value[0] += (gAdcData.I.value[0] - gIavg.value[0])>>gIavg_shift;
  gIavg.value[1] += (gAdcData.I.value[1] - gIavg.value[1])>>gIavg_shift;
  gIavg.value[2] += (gAdcData.I.value[2] - gIavg.value[2])>>gIavg_shift;

  if(measurableShuntThisCycle > two_phase_measurable)
  {
          gAdcData.I.value[0] = gIavg.value[0];
          gAdcData.I.value[1] = gIavg.value[1];
          gAdcData.I.value[2] = gIavg.value[2];
  }

  return;
} // end of runCurrentReconstruction() function

void runSetTrigger(void)
{
  SVGENCURRENT_IgnoreShunt_e ignoreShuntNextCycle = SVGENCURRENT_getIgnoreShunt(svgencurrentHandle);
  SVGENCURRENT_VmidShunt_e midVolShunt = SVGENCURRENT_getVmid(svgencurrentHandle);

  // Set trigger point in the middle of the low side pulse
  HAL_setTrigger(halHandle,ignoreShuntNextCycle,midVolShunt);

  return;
} // end of runSetTrigger() function

void runFieldWeakening(void)
{
  if(FW_getFlag_enableFw(fwHandle) == true)
    {
      FW_incCounter(fwHandle);

      if(FW_getCounter(fwHandle) > FW_getNumIsrTicksPerFwTick(fwHandle))
        {
          _iq refValue;
          _iq fbackValue;

          FW_clearCounter(fwHandle);

          refValue = gMotorVars.VsRef;

          fbackValue =_IQmpy(gMotorVars.Vs,EST_getOneOverDcBus_pu(estHandle));

          FW_run(fwHandle, refValue, fbackValue, &(gIdq_ref_pu.value[0]));

          gMotorVars.IdRef_A = _IQmpy(gIdq_ref_pu.value[0], _IQ(USER_IQ_FULL_SCALE_CURRENT_A));
        }
    }
  else
    {
      gIdq_ref_pu.value[0] = _IQmpy(gMotorVars.IdRef_A, _IQ(1.0/USER_IQ_FULL_SCALE_CURRENT_A));
    }

  return;
} // end of runFieldWeakening() function


void runOffsetsCalculation(void)
{
  uint16_t cnt;

  // enable the PWM
  HAL_enablePwm(halHandle);

  for(cnt=0;cnt<3;cnt++)
    {
      // Set the PWMs to 50% duty cycle
      gPwmData.Tabc.value[cnt] = _IQ(0.0);

      // reset offsets used
      gOffsets_I_pu.value[cnt] = _IQ(0.0);
      gOffsets_V_pu.value[cnt] = _IQ(0.0);

      // run offset estimation
      FILTER_FO_run(filterHandle[cnt],gAdcData.I.value[cnt]);
      FILTER_FO_run(filterHandle[cnt+3],gAdcData.V.value[cnt]);
    }

  if(gOffsetCalcCount++ >= gUserParams.ctrlWaitTime[CTRL_State_OffLine])
    {
      gMotorVars.Flag_enableOffsetcalc = false;
      gOffsetCalcCount = 0;

      for(cnt=0;cnt<3;cnt++)
        {
          // get calculated offsets from filter
          gOffsets_I_pu.value[cnt] = FILTER_FO_get_y1(filterHandle[cnt]);
          gOffsets_V_pu.value[cnt] = FILTER_FO_get_y1(filterHandle[cnt+3]);

          // clear filters
          FILTER_FO_setInitialConditions(filterHandle[cnt],_IQ(0.0),_IQ(0.0));
          FILTER_FO_setInitialConditions(filterHandle[cnt+3],_IQ(0.0),_IQ(0.0));
        }
    }

  return;
} // end of runOffsetsCalculation() function


void softwareUpdate1p6(EST_Handle handle)
{
  float_t fullScaleInductance = USER_IQ_FULL_SCALE_VOLTAGE_V/(USER_IQ_FULL_SCALE_CURRENT_A*
       USER_VOLTAGE_FILTER_POLE_rps);
  float_t Ls_coarse_max = _IQ30toF(EST_getLs_coarse_max_pu(handle));
  int_least8_t lShift = ceil(log(USER_MOTOR_Ls_d/(Ls_coarse_max*fullScaleInductance))/log(2.0));
  uint_least8_t Ls_qFmt = 30 - lShift;
  float_t L_max = fullScaleInductance * pow(2.0,lShift);
  _iq Ls_d_pu = _IQ30(USER_MOTOR_Ls_d / L_max);
  _iq Ls_q_pu = _IQ30(USER_MOTOR_Ls_q / L_max);


  // store the results
  EST_setLs_d_pu(handle,Ls_d_pu);
  EST_setLs_q_pu(handle,Ls_q_pu);
  EST_setLs_qFmt(handle,Ls_qFmt);

  return;
} // end of softwareUpdate1p6() function

//! \brief     Setup the Clarke transform for either 2 or 3 sensors.
//! \param[in] handle             The clarke (CLARKE) handle
//! \param[in] numCurrentSensors  The number of current sensors
void setupClarke_I(CLARKE_Handle handle,const uint_least8_t numCurrentSensors)
```

```
{
  _iq alpha_sf,beta_sf;

  // initialize the Clarke transform module for current
  if(numCurrentSensors == 3)
    {
      alpha_sf = _IQ(MATH_ONE_OVER_THREE);
      beta_sf = _IQ(MATH_ONE_OVER_SQRT_THREE);
    }
  else if(numCurrentSensors == 2)
    {
      alpha_sf = _IQ(1.0);
      beta_sf = _IQ(MATH_ONE_OVER_SQRT_THREE);
    }
  else
    {
      alpha_sf = _IQ(0.0);
      beta_sf = _IQ(0.0);
    }

  // set the parameters
  CLARKE_setScaleFactors(handle,alpha_sf,beta_sf);
  CLARKE_setNumSensors(handle,numCurrentSensors);

  return;
} // end of setupClarke_I() function


//! \brief      Setup the Clarke transform for either 2 or 3 sensors.
//! \param[in] handle             The clarke (CLARKE) handle
//! \param[in] numVoltageSensors  The number of voltage sensors
void setupClarke_V(CLARKE_Handle handle,const uint_least8_t numVoltageSensors)
{
  _iq alpha_sf,beta_sf;

  // initialize the Clarke transform module for voltage
  if(numVoltageSensors == 3)
    {
      alpha_sf = _IQ(MATH_ONE_OVER_THREE);
      beta_sf = _IQ(MATH_ONE_OVER_SQRT_THREE);
    }
 else
    {
      alpha_sf = _IQ(0.0);
      beta_sf = _IQ(0.0);
    }

  // set the parameters
  CLARKE_setScaleFactors(handle,alpha_sf,beta_sf);
  CLARKE_setNumSensors(handle,numVoltageSensors);

  return;
} // end of setupClarke_V() function


//! \brief      Update the global variables (gMotorVars).
//! \param[in] handle  The estimator (EST) handle
void updateGlobalVariables(EST_Handle handle)
{
  // get the speed estimate
  gMotorVars.Speed_krpm = EST_getSpeed_krpm(handle);

  // get the torque estimate
  {
    _iq Flux_pu = EST_getFlux_pu(handle);
    _iq Id_pu = PID_getFbackValue(pidHandle[1]);
    _iq Iq_pu = PID_getFbackValue(pidHandle[2]);
    _iq Ld_minus_Lq_pu = _IQ30toIQ(EST_getLs_d_pu(handle)-EST_getLs_q_pu(handle));
    _iq Torque_Flux_Iq_Nm = _IQmpy(_IQmpy(Flux_pu,Iq_pu),gTorque_Flux_Iq_pu_to_Nm_sf);
    _iq Torque_Ls_Id_Iq_Nm = _IQmpy(_IQmpy(_IQmpy(Ld_minus_Lq_pu,Id_pu),Iq_pu),
         gTorque_Ls_Id_Iq_pu_to_Nm_sf);
    _iq Torque_Nm = Torque_Flux_Iq_Nm + Torque_Ls_Id_Iq_Nm;

    gMotorVars.Torque_Nm = Torque_Nm;
  }

  // get the magnetizing current
  gMotorVars.MagnCurr_A = EST_getIdRated(handle);

  // get the rotor resistance
  gMotorVars.Rr_Ohm = EST_getRr_Ohm(handle);

  // get the stator resistance
  gMotorVars.Rs_Ohm = EST_getRs_Ohm(handle);
  // get the online stator resistance
  gMotorVars.RsOnLine_Ohm = EST_getRsOnLine_Ohm(handle);

  // get the stator inductance in the direct coordinate direction
  gMotorVars.Lsd_H = EST_getLs_d_H(handle);

  // get the stator inductance in the quadrature coordinate direction
  gMotorVars.Lsq_H = EST_getLs_q_H(handle);

  // get the flux in V/Hz in floating point
  gMotorVars.Flux_VpHz = EST_getFlux_VpHz(handle);

  // get the flux in Wb in fixed point
  gMotorVars.Flux_Wb = _IQmpy(EST_getFlux_pu(handle),gFlux_pu_to_Wb_sf);

  // get the estimator state
  gMotorVars.EstState = EST_getState(handle);

  // Get the DC buss voltage
  gMotorVars.VdcBus_kV = _IQmpy(gAdcData.dcBus,_IQ(USER_IQ_FULL_SCALE_VOLTAGE_V/1000.0));

  // read Vd and Vq vectors per units
  gMotorVars.Vd = gVdq_out_pu.value[0];
  gMotorVars.Vq = gVdq_out_pu.value[1];

  // calculate vector Vs in per units
  gMotorVars.Vs = _IQsqrt(_IQmpy(gMotorVars.Vd, gMotorVars.Vd) + _IQmpy(gMotorVars.Vq, gMotorVars.Vq));

  // read Id and Iq vectors in amps
  gMotorVars.Id_A = _IQmpy(gIdq_pu.value[0], _IQ(USER_IQ_FULL_SCALE_CURRENT_A));
  gMotorVars.Iq_A = _IQmpy(gIdq_pu.value[1], _IQ(USER_IQ_FULL_SCALE_CURRENT_A));

  // calculate vector Is in amps
  gMotorVars.Is_A = _IQsqrt(_IQmpy(gMotorVars.Id_A, gMotorVars.Id_A) + _IQmpy(gMotorVars.Iq_A, gMotorVars.
       Iq_A));

  return;
} // end of updateGlobalVariables() function


void updateRsOnLine(EST_Handle handle)
{
  // execute Rs OnLine code
  if(gMotorVars.Flag_Run_Identify == true)
    {
      if((EST_getState(handle) == EST_State_OnLine) && (gFlag_enableRsOnLine))
        {
          float_t RsError_Ohm = gMotorVars.RsOnLine_Ohm − gMotorVars.Rs_Ohm;

          EST_setFlag_enableRsOnLine(handle,true);
          EST_setRsOnLineId_mag_pu(handle,_IQmpy(gRsOnLineId_mag_A,_IQ(1.0/USER_IQ_FULL_SCALE_CURRENT_A)))
               ;
          EST_setRsOnLineAngleDelta_pu(handle,_IQmpy(gRsOnLineFreq_Hz, _IQ(1.0/USER_ISR_FREQ_Hz)));

          if(abs(RsError_Ohm) < (gMotorVars.Rs_Ohm ∗ 0.05))
            {
              EST_setFlag_updateRs(handle,true);
            }
        }
      else
        {
          EST_setRsOnLineId_mag_pu(handle,_IQ(0.0));
          EST_setRsOnLineId_pu(handle,_IQ(0.0));
          EST_setRsOnLine_pu(handle, EST_getRs_pu(handle));
          EST_setFlag_enableRsOnLine(handle,false);
          EST_setFlag_updateRs(handle,false);
          EST_setRsOnLine_qFmt(handle,EST_getRs_qFmt(handle));
        }
    }

  return;
} // end of updateRsOnLine() function
void updateCPUusage(void)
{
  uint32_t minDeltaCntObserved = CPU_USAGE_getMinDeltaCntObserved(cpu_usageHandle);
  uint32_t avgDeltaCntObserved = CPU_USAGE_getAvgDeltaCntObserved(cpu_usageHandle);
  uint32_t maxDeltaCntObserved = CPU_USAGE_getMaxDeltaCntObserved(cpu_usageHandle);
  uint16_t pwmPeriod = HAL_readPwmPeriod(halHandle,PWM_Number_1);
  float_t  cpu_usage_den = (float_t)pwmPeriod ∗ (float_t)USER_NUM_PWM_TICKS_PER_ISR_TICK ∗ 2.0;

  // calculate the minimum cpu usage percentage
  gCpuUsagePercentageMin = (float_t)minDeltaCntObserved / cpu_usage_den ∗ 100.0;
  // calculate the average cpu usage percentage
  gCpuUsagePercentageAvg = (float_t)avgDeltaCntObserved / cpu_usage_den * 100.0;

  // calculate the maximum cpu usage percentage
  gCpuUsagePercentageMax = (float_t)maxDeltaCntObserved / cpu_usage_den * 100.0;

  return;
} // end of updateCPUusage() function


//@} //defgroup
// end of file
```

D

can.c code

```
/*
 * can.c
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * See LICENCE file for LGPL details.
 *
 * Contains the various functions related to the eCAN
 * Target Device:  TMS320F2802x
 * Copyright (C): 2017  Andrew Buckin
 *  Created on: April 20, 2017
 *       Author: Andrew Buckin
 *
 */
#include <stdlib.h>
#include "can.h"

#define NELEMS(x) (sizeof(x)/sizeof((x)[0]))
//! \brief Define to allow protected register writes (legacy)
//!
#define  EALLOW asm(" EALLOW")

//! \brief Define to allow protected register writes
//!
#define  ENABLE_PROTECTED_REGISTER_WRITE_MODE  asm(" EALLOW")

//! \brief Define to disable protected register writes (legacy)
//!
#define  EDIS   asm(" EDIS")

//! \brief Define to disable protected register writes
//!
#define  DISABLE_PROTECTED_REGISTER_WRITE_MODE asm(" EDIS")

ECAN_Handle ECAN_init(){

        ECAN_Handle handle = (ECAN_Handle)calloc(1, sizeof(ECAN_Obj));

        if(handle == NULL){
                return (ECAN_Handle)NULL;
        }

        handle->ECanaRegs = (ECAN_REGS_t*)ECANA_REGS_ADDR;
        handle->ECanaMboxes = (ECAN_MBOXES_t*)ECANA_MBOX_ADDR;
        handle->ECanaLAMRegs = (LAM_REGS_t*)ECANA_LAM_ADDR;
        handle->ECanaMOTORegs = (MOTO_REGS_t*)ECANA_MOTO_ADDR;
        handle->ECanaMOTSRegs = (MOTS_REGS_t*)ECANA_MOTS_ADDR;

        return handle;
}
```

```
void ECAN_setBitrate(ECAN_Handle handle, ECAN_Bitrate_e bitrate){
        struct ECAN_REGS ECanaShadow;

        EALLOW; // Allow access to protected bits

        ECanaShadow.CANMC.all = handle->ECanaRegs->CANMC.all;
        ECanaShadow.CANMC.bit.CCR = 1 ;            // Set CCR = 1
        handle->ECanaRegs->CANMC.all = ECanaShadow.CANMC.all;

        // Wait until the CPU has been granted permission to change the configuration registers
        do
        {
          ECanaShadow.CANES.all = handle->ECanaRegs->CANES.all;
        } while(ECanaShadow.CANES.bit.CCE != 1 );       // Wait for CCE bit to be set..

        ECanaShadow.CANBTC.all = 0;
        // http://www.bittiming.can-wiki.info/   BT=15
        // 90 MHz SYSCLKOUT. (45 MHz CAN module clock)
        switch(bitrate) {
           case Bitrate_1M:   ECanaShadow.CANBTC.all = 0x00020059; break;
           case Bitrate_500K: ECanaShadow.CANBTC.all = 0x00050059; break;
           case Bitrate_250K: ECanaShadow.CANBTC.all = 0x000b0059; break;
           case Bitrate_125K: ECanaShadow.CANBTC.all = 0x00170059; break;
           case Bitrate_100K: ECanaShadow.CANBTC.all = 0x001d0059; break;
           case Bitrate_83K:  ECanaShadow.CANBTC.all = 0x00230059; break;
           case Bitrate_50K:  ECanaShadow.CANBTC.all = 0x003b0059; break;
           case Bitrate_20K:  ECanaShadow.CANBTC.all = 0x00950059; break;
           default:                        ECanaShadow.CANBTC.all = 0x000503BD; break; //500K 100ns
                sprac35
                //ECanaShadow.CANBTC.bit.BRPREG = 5;
                //ECanaShadow.CANBTC.bit.SJWREG = 3;
                //ECanaShadow.CANBTC.bit.SAM = 1;
                //ECanaShadow.CANBTC.bit.TSEG1REG = 7;
                //ECanaShadow.CANBTC.bit.TSEG2REG = 5;
        }

        handle->ECanaRegs->CANBTC.all = ECanaShadow.CANBTC.all;

        ECanaShadow.CANMC.all = handle->ECanaRegs->CANMC.all;
        ECanaShadow.CANMC.bit.CCR = 0 ;            // Set CCR = 0
        handle->ECanaRegs->CANMC.all = ECanaShadow.CANMC.all;

        // Wait until the CPU no longer has permission to change the configuration registers
        do
        {
          ECanaShadow.CANES.all = handle->ECanaRegs->CANES.all;
        } while(ECanaShadow.CANES.bit.CCE != 0 );       // Wait for CCE bit to be  cleared..


        EDIS; // Disable access to protected bits
}

void ECAN_setBTCreg(ECAN_Handle handle, long BTC_ALL){
        struct ECAN_REGS ECanaShadow;

        EALLOW; // Allow access to protected bits

        ECanaShadow.CANMC.all = handle->ECanaRegs->CANMC.all;
        ECanaShadow.CANMC.bit.CCR = 1 ;            // Set CCR = 1
        handle->ECanaRegs->CANMC.all = ECanaShadow.CANMC.all;

        // Wait until the CPU has been granted permission to change the configuration registers
        do
        {
          ECanaShadow.CANES.all = handle->ECanaRegs->CANES.all;
        } while(ECanaShadow.CANES.bit.CCE != 1 );       // Wait for CCE bit to be set..

        ECanaShadow.CANBTC.all = BTC_ALL;

        handle->ECanaRegs->CANBTC.all = ECanaShadow.CANBTC.all;

        ECanaShadow.CANMC.all = handle->ECanaRegs->CANMC.all;
        ECanaShadow.CANMC.bit.CCR = 0 ;            // Set CCR = 0
        handle->ECanaRegs->CANMC.all = ECanaShadow.CANMC.all;

        // Wait until the CPU no longer has permission to change the configuration registers
        do
        {
          ECanaShadow.CANES.all = handle->ECanaRegs->CANES.all;
        } while(ECanaShadow.CANES.bit.CCE != 0 );       // Wait for CCE bit to be  cleared..


        EDIS; // Disable access to protected bits
}
```

```
void ECAN_enableAllInt(ECAN_Handle handle){
        struct ECAN_REGS ECanaShadow;

        EALLOW; // Allow access to protected bits

        //Enable interrupts.
        ECanaShadow.CANGIM.all = 0;
        ECanaShadow.CANGIM.bit.AAIM = 1; /∗ Abort acknowledge ∗/
        ECanaShadow.CANGIM.bit.WDIM = 1; /∗ Write denied ∗/
        ECanaShadow.CANGIM.bit.WUIM = 1; /∗ Wake up ∗/
        ECanaShadow.CANGIM.bit.BOIM = 1; /∗ Bus−off ∗/
        ECanaShadow.CANGIM.bit.EPIM = 1; /∗ Error−passive ∗/
        ECanaShadow.CANGIM.bit.WLIM = 1; /∗ Warning level ∗/
        ECanaShadow.CANGIM.bit.GIL = 1;  /∗ Global Interrup Level. ∗/ //the GIL (GIM.2) bit can be set to
             have the global interrupts on another level than the mailbox interrupts
        ECanaShadow.CANGIM.bit.I0EN = 1; /∗ Interrupt 0 enable ∗/
        ECanaShadow.CANGIM.bit.I1EN = 1; /∗ Interrupt 1 enable ∗/
        ECanaShadow.CANGIM.bit.RMLIM = 1; /∗ Received Messae Lost ∗/

        handle−>ECanaRegs−>CANGIM.all = ECanaShadow.CANGIM.all;

        EDIS; // Disable access to protected bits
}

void ECAN_disableAllInt(ECAN_Handle handle){
        struct ECAN_REGS ECanaShadow;

        EALLOW; // Allow access to protected bits

        //Disable interrupts.
        ECanaShadow.CANGIM.all = 0;
        handle−>ECanaRegs−>CANGIM.all = ECanaShadow.CANGIM.all;

        EDIS; // Disable access to protected bits
}

void ECAN_clearMSGCTRL(ECAN_Handle handle){
        volatile struct MBOX ∗mbox = (&(handle−>ECanaMboxes−>MBOX0));
//      EALLOW; // Allow access to protected bits

    int i = 0;
    for(i = 0; i < 32; i++)
    {
        mbox−>MSGCTRL.all = 0x00000000;
        //mbox−>MSGCTRL.all = i; //Test
        ∗mbox++;
    }

//      EDIS; // Disable access to protected bits
}

void ECAN_clearMSGID(ECAN_Handle handle){
        volatile struct MBOX ∗mbox = (&(handle−>ECanaMboxes−>MBOX0));
//      EALLOW; // Allow access to protected bits
        /∗ Disable all Mailboxes  ∗/
        handle−>ECanaRegs−>CANME.all = 0x00000000;        // Required before writing the MSGIDs

    int i = 0;
    for(i = 0; i < 32; i++)
    {
        mbox−>MSGID.all = 0x00000000;
        //mbox−>MSGID.all = i; //Test
        ∗mbox++;
    }
    //handle−>ECanaRegs−>CANME.all = 0xFFFFFFFF;
//      EDIS; // Disable access to protected bits
}

void ECAN_clearMDL(ECAN_Handle handle){
        volatile struct MBOX ∗mbox = (&(handle−>ECanaMboxes−>MBOX0));
//      EALLOW; // Allow access to protected bits

    int i = 0;
    for(i = 0; i < 32; i++)
    {
        mbox−>MDL.all = 0x00000000;
        //mbox−>MDL.all = i; //Test
        ∗mbox++;
    }

//      EDIS; // Disable access to protected bits
}
```

```
void ECAN_clearMDH(ECAN_Handle handle){
        volatile struct MBOX *mbox = (&(handle->ECanaMboxes->MBOX0));
//      EALLOW; // Allow access to protected bits

    int i = 0;
    for(i = 0; i < 32; i++)
    {
        mbox->MDH.all = 0x00000000;
        //mbox->MDH.all = i; //Test
        *mbox++;
    }

//      EDIS; // Disable access to protected bits
}

void ECAN_enableAllMailbox(ECAN_Handle handle){
//      EALLOW; // Allow access to protected bits
        /* enable all Mailboxes  */
        handle->ECanaRegs->CANME.all = 0xFFFFFFFF;
//      EDIS; // Disable access to protected bits
}

void ECAN_disableAllMailbox(ECAN_Handle handle){
//      EALLOW; // Allow access to protected bits
        /* disable all Mailboxes  */
        handle->ECanaRegs->CANME.all = 0x00000000;
//      EDIS; // Disable access to protected bits
}

void ECAN_setTXIO(ECAN_Handle handle){
        struct ECAN_REGS ECanaShadow;
        EALLOW; // Allow access to protected bits
        ECanaShadow.CANTIOC.all = handle->ECanaRegs->CANTIOC.all;
        ECanaShadow.CANTIOC.bit.TXFUNC = 1;
        handle->ECanaRegs->CANTIOC.all = ECanaShadow.CANTIOC.all;
        EDIS; // Disable access to protected bits
}

void ECAN_setRXIO(ECAN_Handle handle){
        struct ECAN_REGS ECanaShadow;
        EALLOW; // Allow access to protected bits
        ECanaShadow.CANRIOC.all = handle->ECanaRegs->CANRIOC.all;
        ECanaShadow.CANRIOC.bit.RXFUNC = 1;
        handle->ECanaRegs->CANRIOC.all = ECanaShadow.CANRIOC.all;
        EDIS; // Disable access to protected bits
}

void ECAN_resetTXIO(ECAN_Handle handle){
        struct ECAN_REGS ECanaShadow;
        EALLOW; // Allow access to protected bits
        ECanaShadow.CANTIOC.all = handle->ECanaRegs->CANTIOC.all;
        ECanaShadow.CANTIOC.bit.TXFUNC = 0;
        handle->ECanaRegs->CANTIOC.all = ECanaShadow.CANTIOC.all;
        EDIS; // Disable access to protected bits
}

void ECAN_resetRXIO(ECAN_Handle handle){
        struct ECAN_REGS ECanaShadow;
        EALLOW; // Allow access to protected bits
        ECanaShadow.CANRIOC.all = handle->ECanaRegs->CANRIOC.all;
        ECanaShadow.CANRIOC.bit.RXFUNC = 0;
        handle->ECanaRegs->CANRIOC.all = ECanaShadow.CANRIOC.all;
        EDIS; // Disable access to protected bits
}

void ECAN_setSCCmode(ECAN_Handle handle){
        struct ECAN_REGS ECanaShadow;
        EALLOW; // Allow access to protected bits
        /* Configure eCAN for HECC mode - (reqd to access mailboxes 16 thru 31) SCC mode */
                                                                        // HECC mode also enables
                                                                            time-stamping feature
        ECanaShadow.CANMC.all = handle->ECanaRegs->CANMC.all;
        ECanaShadow.CANMC.bit.SCB = 1;
        handle->ECanaRegs->CANMC.all = ECanaShadow.CANMC.all;
        EDIS; // Disable access to protected bits
}

void ECAN_setECANmode(ECAN_Handle handle){
        struct ECAN_REGS ECanaShadow;
        EALLOW; // Allow access to protected bits
        ECanaShadow.CANMC.all = handle->ECanaRegs->CANMC.all;
        ECanaShadow.CANMC.bit.SCB = 0;
        handle->ECanaRegs->CANMC.all = ECanaShadow.CANMC.all;
        EDIS; // Disable access to protected bits
```

```
}

void ECAN_Mode(ECAN_Handle handle, SCB_Bit_e mode){
        struct ECAN_REGS ECanaShadow;
        EALLOW; // Allow access to protected bits
        ECanaShadow.CANMC.all = handle->ECanaRegs->CANMC.all;
        ECanaShadow.CANMC.bit.SCB = mode;
        handle->ECanaRegs->CANMC.all = ECanaShadow.CANMC.all;
        EDIS; // Disable access to protected bits
}


void ECAN_setSelfTest(ECAN_Handle handle){
        struct ECAN_REGS ECanaShadow;
        EALLOW; // Allow access to protected bits
        ECanaShadow.CANMC.all = handle->ECanaRegs->CANMC.all;
        ECanaShadow.CANMC.bit.STM = 1;     // Enable self-test mode.
    handle->ECanaRegs->CANMC.all = ECanaShadow.CANMC.all;
        EDIS; // Disable access to protected bits
}

void ECAN_resetSelfTest(ECAN_Handle handle){
        struct ECAN_REGS ECanaShadow;
        EALLOW; // Allow access to protected bits
        ECanaShadow.CANMC.all = handle->ECanaRegs->CANMC.all;
        ECanaShadow.CANMC.bit.STM = 0;     // Disable self-test mode.
    handle->ECanaRegs->CANMC.all = ECanaShadow.CANMC.all;
        EDIS; // Disable access to protected bits
}

void ECAN_SelfTest(ECAN_Handle handle,STM_Bit_e mode){
        struct ECAN_REGS ECanaShadow;
        EALLOW; // Allow access to protected bits
        ECanaShadow.CANMC.all = handle->ECanaRegs->CANMC.all;
        ECanaShadow.CANMC.bit.STM = mode;     // Disable self-test mode.
    handle->ECanaRegs->CANMC.all = ECanaShadow.CANMC.all;
        EDIS; // Disable access to protected bits
}

void ECAN_clearCANTA(ECAN_Handle handle){
//      EALLOW; // Allow access to protected bits
        /* Clear all Transmission-Acknowledge Register bits */
        handle->ECanaRegs->CANTA.all = 0xFFFFFFFF;
//      EDIS; // Disable access to protected bits
}
void ECAN_clearCANRMP(ECAN_Handle handle){
//      EALLOW; // Allow access to protected bits
        /* Clear all Received-Message-Pending Register bits */
        handle->ECanaRegs->CANRMP.all = 0xFFFFFFFF;
//      EDIS; // Disable access to protected bits
}
void ECAN_clearCANGIF0(ECAN_Handle handle){
//      EALLOW; // Allow access to protected bits
        /* Clear all Global Interrupt Flag 0 */
        handle->ECanaRegs->CANGIF0.all = 0xFFFFFFFF;
//      EDIS; // Disable access to protected bits
}
void ECAN_clearCANGIF1(ECAN_Handle handle){
//      EALLOW; // Allow access to protected bits
        /* Clear all Global Interrupt Flag 1 */
        handle->ECanaRegs->CANGIF1.all = 0xFFFFFFFF;
//      EDIS; // Disable access to protected bits
}

void ECAN_setMailboxDir(ECAN_Handle handle,long dir){
//      EALLOW; // Allow access to protected bits
        /* Mailbox-Direction Register  */
        handle->ECanaRegs->CANMD.all = dir;
//      EDIS; // Disable access to protected bits
}

void ECAN_setMailboxIntMask(ECAN_Handle handle,long mask){
        EALLOW; // Allow access to protected bits
        /* Mailbox Interrupt Mask Register  */
        handle->ECanaRegs->CANMIM.all = mask;
        EDIS; // Disable access to protected bits
}

void ECAN_configMailbox(ECAN_Handle handle, ECAN_MailBox_e MailBoxN, uint32_t msgid, Enable_Mbox_e
    enable_t, ECAN_MailDir_e dir_t,IDE_Bit_e IDE_t, DLC_Bit_e length, LAMI_Bit_e lami_bit, AME_Bit_e
    AME_t, uint32_t mask){
        struct ECAN_REGS ECanaShadow;
        volatile struct MBOX *mbox = (&(handle->ECanaMboxes->MBOX0)) + /*(uint32_t)*/MailBoxN;
        volatile union CANLAM_REG *lam = (&(handle->ECanaLAMRegs->LAM0)) + /*(uint32_t)*/MailBoxN;
        uint8_t enable = enable_t;
        uint8_t dir = dir_t;
//      EALLOW; // Allow access to protected bits

        ECanaShadow.CANME.all = handle->ECanaRegs->CANME.all;
        ECanaShadow.CANME.all = (~((~ECanaShadow.CANME.all) | (((uint32_t)1) << MailBoxN))) | (((uint32_t)
            Disable_Mbox) << MailBoxN);
        handle->ECanaRegs->CANME.all = ECanaShadow.CANME.all;

        lam->all = mask;
        lam->bit.LAMI = lami_bit;

        if (IDE_t == Extended_ID)
        {
                mbox->MSGID.all = msgid;
        }
        else
        {
                mbox->MSGID.bit.EXTMSGID_L = 0;
                mbox->MSGID.bit.EXTMSGID_H = 0;
                mbox->MSGID.bit.STDMSGID = (uint16_t)msgid;
        }

        mbox->MSGID.bit.IDE = IDE_t;
        mbox->MSGID.bit.AME = AME_t;
        mbox->MSGID.bit.AAM = Normal_transmit;

    // Specify that bits will be sent/received
    mbox->MSGCTRL.bit.DLC = length; //8;

    //Enable/disable mbox.
        enable = enable & 1; //Making sure it's only one bit.
        ECanaShadow.CANME.all = handle->ECanaRegs->CANME.all;
        ECanaShadow.CANME.all = (~((~ECanaShadow.CANME.all) | (((uint32_t)1) << MailBoxN))) | (((uint32_t)
            enable) << MailBoxN);
        handle->ECanaRegs->CANME.all = ECanaShadow.CANME.all;

        //Setting rx/tx mode.
        dir = dir & 1; //Making sure it's only one bit.
        ECanaShadow.CANMD.all = handle->ECanaRegs->CANMD.all;
        ECanaShadow.CANMD.all = (~((~ECanaShadow.CANMD.all) | (((uint32_t)1) << MailBoxN))) | (((uint32_t)
            dir) << MailBoxN);
        handle->ECanaRegs->CANMD.all = ECanaShadow.CANMD.all;
//      EDIS; // Disable access to protected bits
}

void ECAN_putDataMailbox(ECAN_Handle handle,ECAN_MailBox_e MailBoxN, long MDL_t, long MDH_t){
        volatile struct MBOX *mbox = (&(handle->ECanaMboxes->MBOX0)) + /*(uint32_t)*/MailBoxN;
//      EALLOW; // Allow access to protected bits
        mbox->MDL.all = MDL_t;
        mbox->MDH.all = MDH_t;
//      EDIS; // Disable access to protected bits
}

void ECAN_configMasterReg(ECAN_Handle handle, CCR_Bit_e CCR_t, PDR_Bit_e PDR_t, DBO_Bit_e DBO_t,
     WUBA_Bit_e WUBA_t, CDR_Bit_e CDR_t, ABO_Bit_e ABO_t, SRES_Bit_e SRES_t, MBNR_Bit_e MBNR_t){
        struct ECAN_REGS ECanaShadow;
        EALLOW; // Allow access to protected bits
        /* config Master Control Register  */
        ECanaShadow.CANMC.all = handle->ECanaRegs->CANMC.all;
        ECanaShadow.CANMC.bit.CCR = CCR_t;
        ECanaShadow.CANMC.bit.PDR = PDR_t;
        ECanaShadow.CANMC.bit.DBO = DBO_t;
        ECanaShadow.CANMC.bit.WUBA = WUBA_t;
        ECanaShadow.CANMC.bit.CDR = CDR_t;
        ECanaShadow.CANMC.bit.ABO = ABO_t;
        ECanaShadow.CANMC.bit.SRES = SRES_t;
        ECanaShadow.CANMC.bit.MBNR = MBNR_t;
        handle->ECanaRegs->CANMC.all = ECanaShadow.CANMC.all;
        EDIS; // Disable access to protected bits
}

int ECAN_sendMSG(ECAN_Handle handle,ECAN_MailBox_e MailBoxN, long MDL_t, long MDH_t){
        volatile struct MBOX *mbox = (&(handle->ECanaMboxes->MBOX0)) + MailBoxN;

        mbox->MDL.all = MDL_t;
        mbox->MDH.all = MDH_t;

    handle->ECanaRegs->CANTRS.all  |= 1 << MailBoxN;
    //Wait for transmit acknowledge
    while(!(handle->ECanaRegs->CANTA.all >> MailBoxN));
    //Wait TRS.n = 0
    while(handle->ECanaRegs->CANTRS.all >> MailBoxN);
    //Set TA.n
    handle->ECanaRegs->CANTA.all |= 1 << MailBoxN;
    //Wait until read TA.n is 0
    while(handle->ECanaRegs->CANTA.all >> MailBoxN);
    return 0;
}
```

Bibliography

[1] D. Buist and E. Salzmann, *Permanent Magnet Motor Simulation for the Nuon Solar Team*, Bsc thesis, Delft University of Technology (2018).

[2] Galil, *Brushless Sine Drives- Application Note* (Jan. 2011), available: sine-drive-setup.php.

[3] O. Mohammed, *A study of control systems for brushless DC motors*, Master's thesis, University of Toledo (2014), theses and Dissertations. 1702.

[4] *Appl. Note DRM117*, Freescale.

[5] G.-J. Su and J. McKeever, *Low cost sensorless control of brushless DC motors with improved speed range,* in *APEC. Seventeenth Annual IEEE Applied Power Electronics Conference and Exposition (Cat. No.02CH37335)* (IEEE, 2002).

[6] J. Schofield, *Direct torque control - DTC,* in *IEE Colloquium on Vector Control and Direct Torque Control of Induction Motors* (IEE, 1995).

[7] D. S. Nair, G. Jagadanand, and S. George, *Analysis of direct torque controlled BLDC motor with reduced torque ripple,* in *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)* (IEEE, 2015).

[8] B. Santhosh Kumar, *Direct torque control of inverter fed pmsm drive using svm, International Journal of Application or Innovation in Engineering & Management (IJAIEM),* **2**, 601 (2013).

[9] X. Garcia, B. Zigmund, A. Terlizzi, R. Pavlanin, and L. Salvatore, *Comparison between foc and dtc strategies for permanent magnet synchronous motors,* Advances in Electrical and Electronic Engineering **5** (2011).

[10] *Field Oriented Control (FOC) Made Ultra Simple,* `https://www.roboteq.com/index.php/applications/100-how-to/359-field-oriented-control-foc-made-ultra-simple` (), accessed: 2017-12-18.

[11] Copley Control Corp., *What is 'Field Oriented Control' and what good is it?* Available: Field-Oriented-Control.pdf Accessed: 2017-12-19.

[12] Texas Instruments, *Clarke & Park Transforms on the TMS320C2xx* (1997), available: bpra048.pdf.

[13] K. H. Nam, *AC Motor Control and Electric Vehicle Applications* (CRC Press, Boca Raton, FL, 2010).

[14] Texas Instruments, *InstaSPIN Projects and Labs User's Guide* (Feb. 2013 [Revised Mar. 2017]).

[15] *Texas Instruments. "TI's InstaSPIN™-enabled real-time controllers",* `http://www.ti.com/ww/en/mcu/instaspin/instaspin-foc_MCUs.shtml` (), accessed: 2017-12-20.

[16] Texas Instruments, *InstaSPIN-FOC™ and InstaSPIN-MOTION™ User's Guide* (Jan. 2013 [Revised Feb. 2017]), available: spruhj1g.pdf.

[17] Texas Instruments, *DRV832XX EVM User's Guide* (Feb. 2017 [Revised May 2017]), available: slvub01a.pdf.

[18] Texas Instruments, *BOOSTXL8323RS Hardware Quick Start Guide* (Oct. 2017), available: drv8323RS HQSG.

[19] Texas Instruments, *BOOSTXL-DRV8301 datasheet* (Aug. 2011 [Revised Jan. 2016]), available: slos719f.pdf.

[20] Texas Instruments, *BOOSTXL-DRV8301 Hardware User's Guide* (Oct. 2013), available: slvu974.pdf.

[21] Texas Instruments, *TMS320x2806x Piccolo Technical Reference Manual* (Jan. 2011 [Revised Apr. 2017]), available: spruh18g.pdf.

[22] Texas Instruments, *Introduction to the Controller Area Network (CAN)* (Aug. 2002 [Revised May 2016]), available: sloa101b.pdf.

[23] Kiril Mucevski, *Automotive CAN Bus System Explained,* `https://www.linkedin.com/pulse/automotive-can-bus-system-explained-kiril-mucevski/`, accessed: 2018-01-02.

[24] Texas Instruments, *TMS320x280x/2801x Enhanced Controller Area Network (eCAN)* (Jan. 2009), available: sprueu0.pdf.

[25] Texas Instruments, *Hardware Abstraction Layer (HAL) Module of MotorWare™*, available: motorware_hal_tutorial.pdf.

[26] *Texas Instruments Forum. "DRV8301-69M-KIT: CANBUS/Serial communication,* `https://e2e.ti.com/support/microcontrollers/c2000/f/902/p/587358/2158620?tisearch=e2e-sitesearch&keymatch=f28069m%20eCAN%20commmunication#pi316717=1` (), accessed: 2018-01-19.

[27] Texas Instruments, *Read Me First, InstaSPIN-FOC and InstaSPIN-MOTION LaunchPad and BoosterPack* (Oct. 2013 [Revised Aug. 2015]).