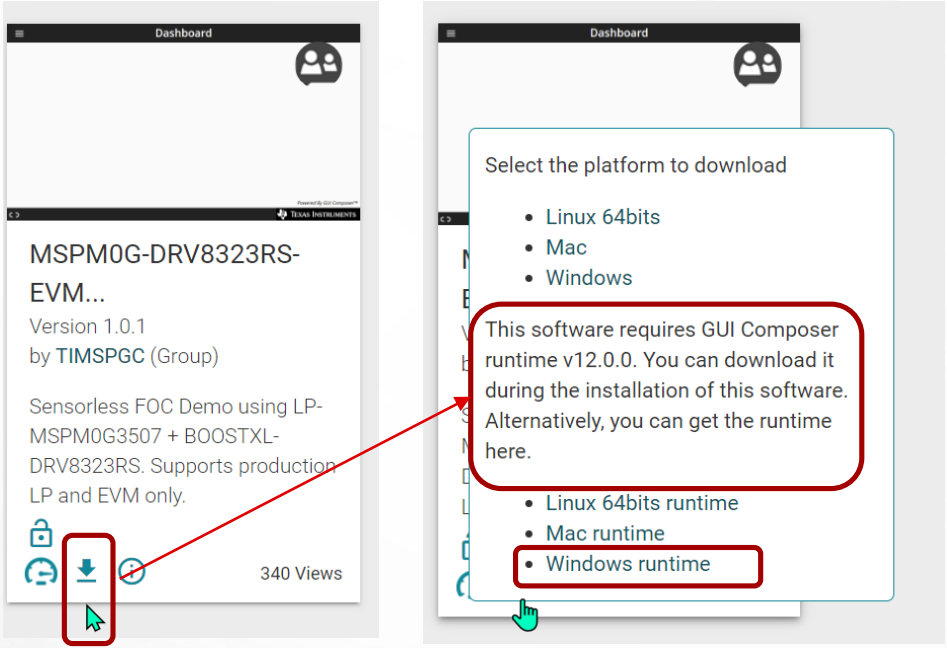
# MSPM0G-DRV8323RS GUI

# MSPM0G-DRV8323RS GUI

Access link:

https://dev.ti.com/gallery/view/TIMSPGC/MSPM0G-DRV8323RS-EVM-GUI/ver/1.0.1/

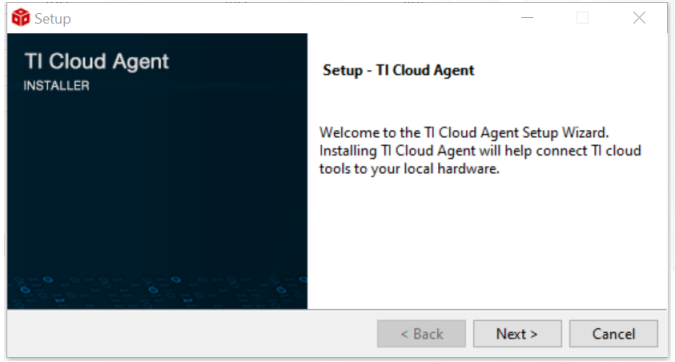



Install the software environment

# MSPM0G-DRV8323RS GUI

Access link:

https://dev.ti.com/gallery/view/TIMSPGC/MSPM0G-DRV8323RS-EVM-GUI/ver/1.0.1/

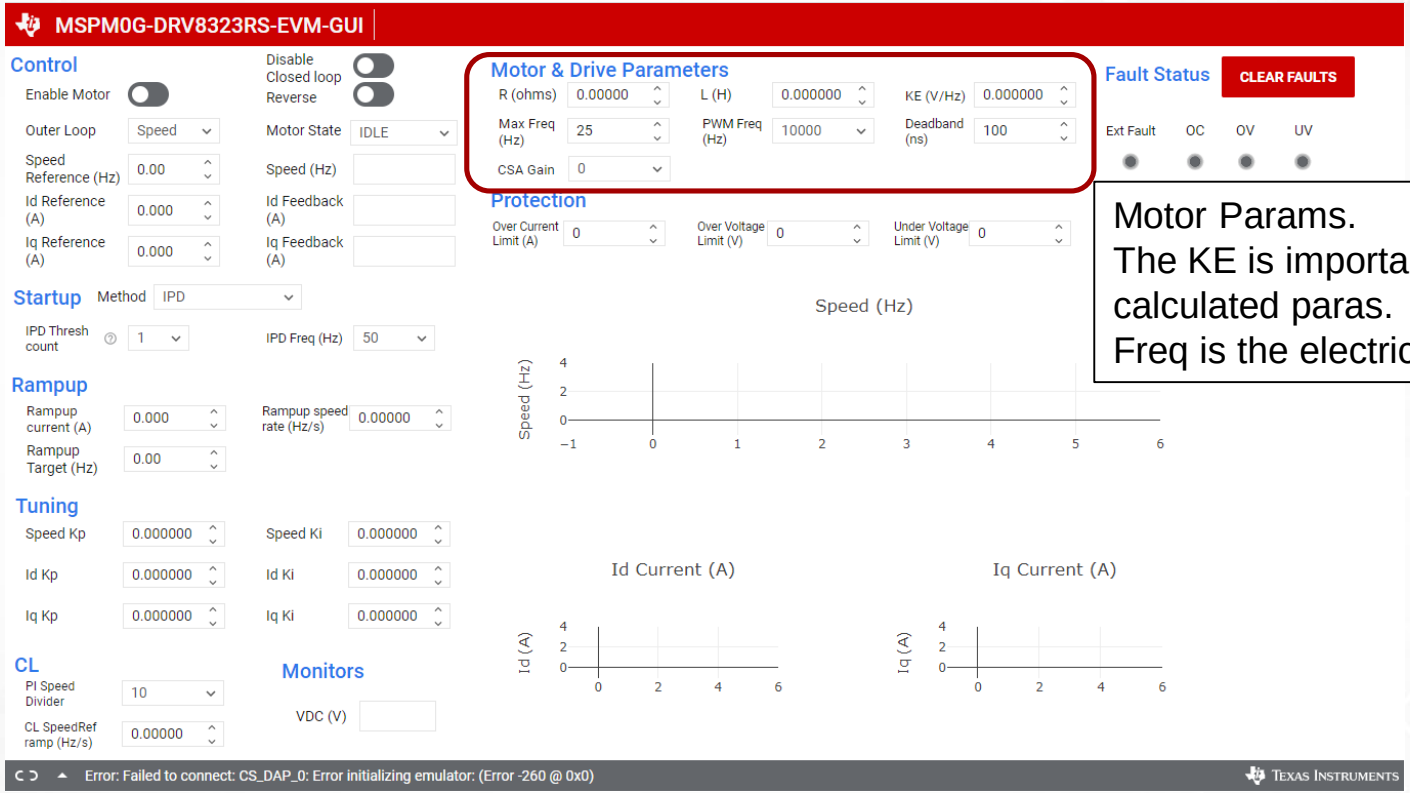

Might need install the TI Cloud Agent if this page shows the download request.



Texas Instruments

# MSPM0G-DRV8323RS GUI

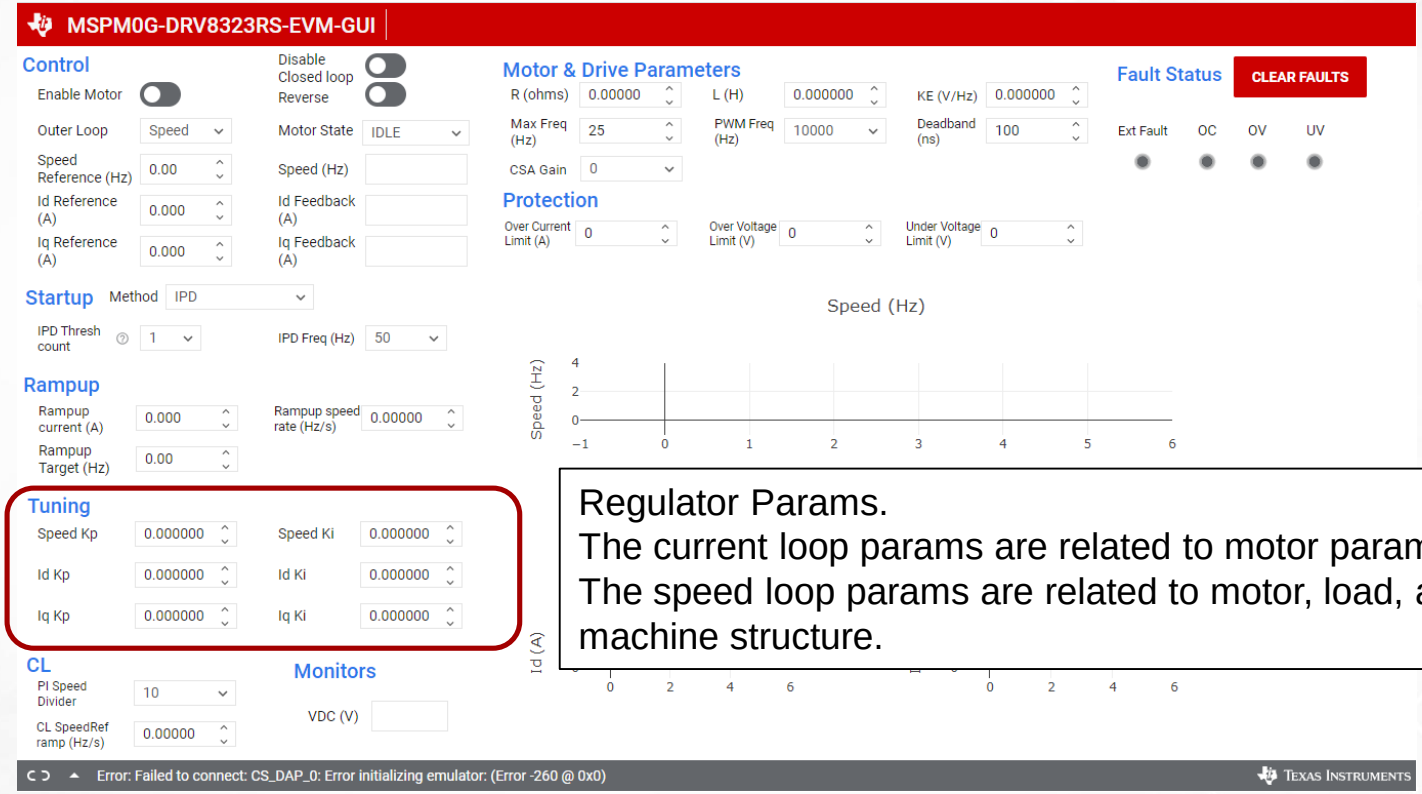
Introduction




Motor Params.
The KE is important for the ESMO calculated paras.
Freq is the electrical frequency.

# MSPM0G-DRV8323RS GUI

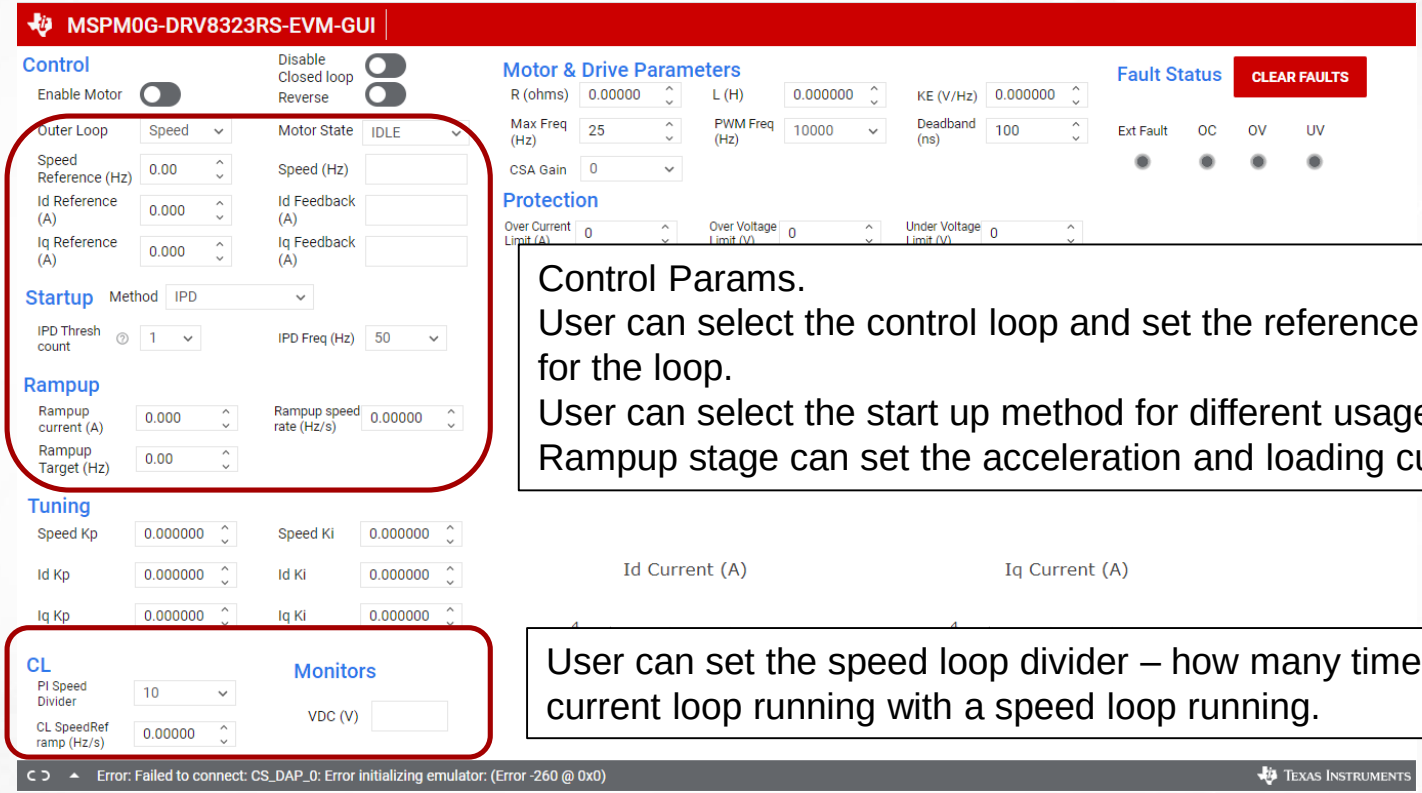
Introduction




Regulator Params.
The current loop params are related to motor params.
The speed loop params are related to motor, load, and machine structure.

# MSPM0G-DRV8323RS GUI

Introduction

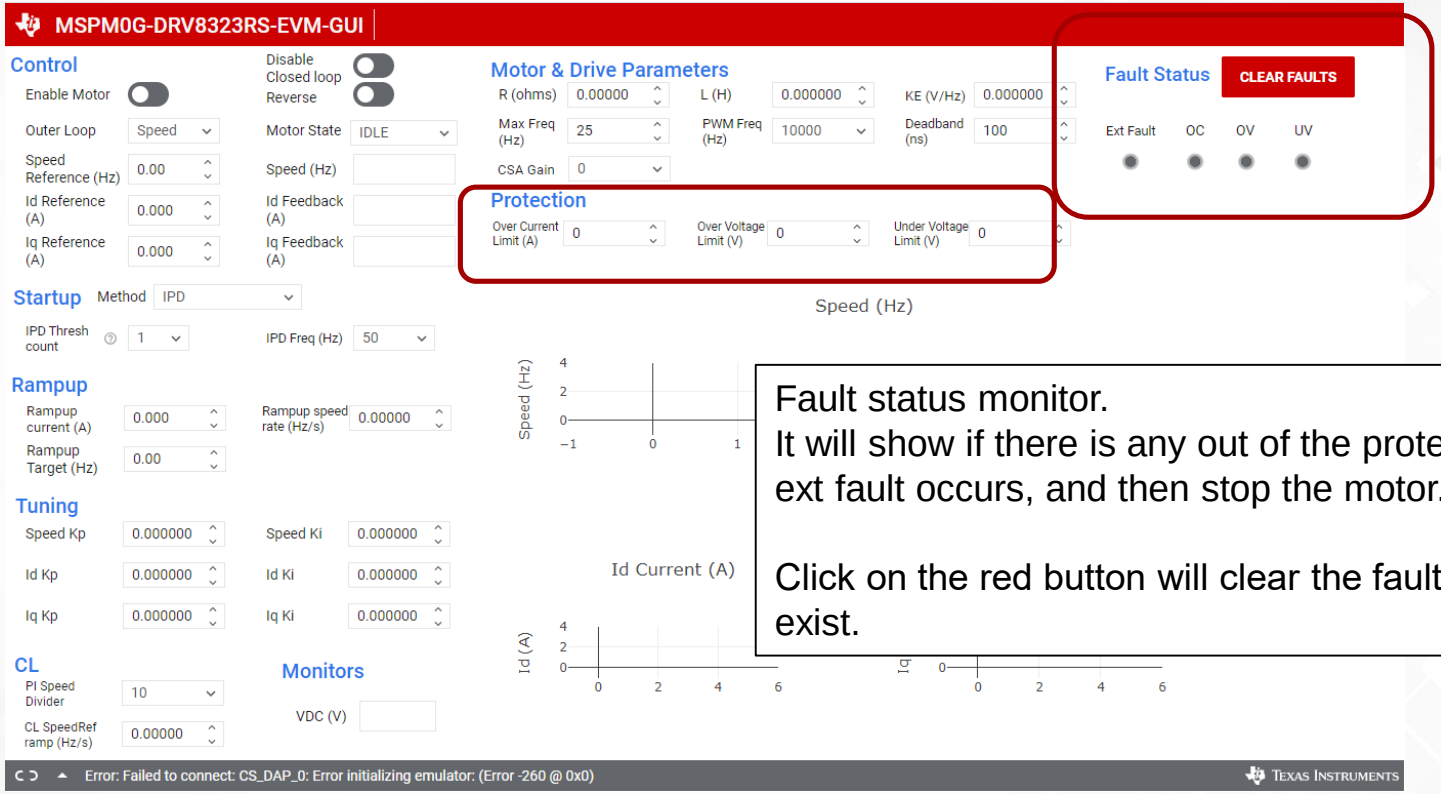



Control Params.
User can select the control loop and set the reference input for the loop.
User can select the start up method for different usage.
Rampup stage can set the acceleration and loading current.

User can set the speed loop divider – how many times current loop running with a speed loop running.

# MSPM0G-DRV8323RS GUI

Introduction

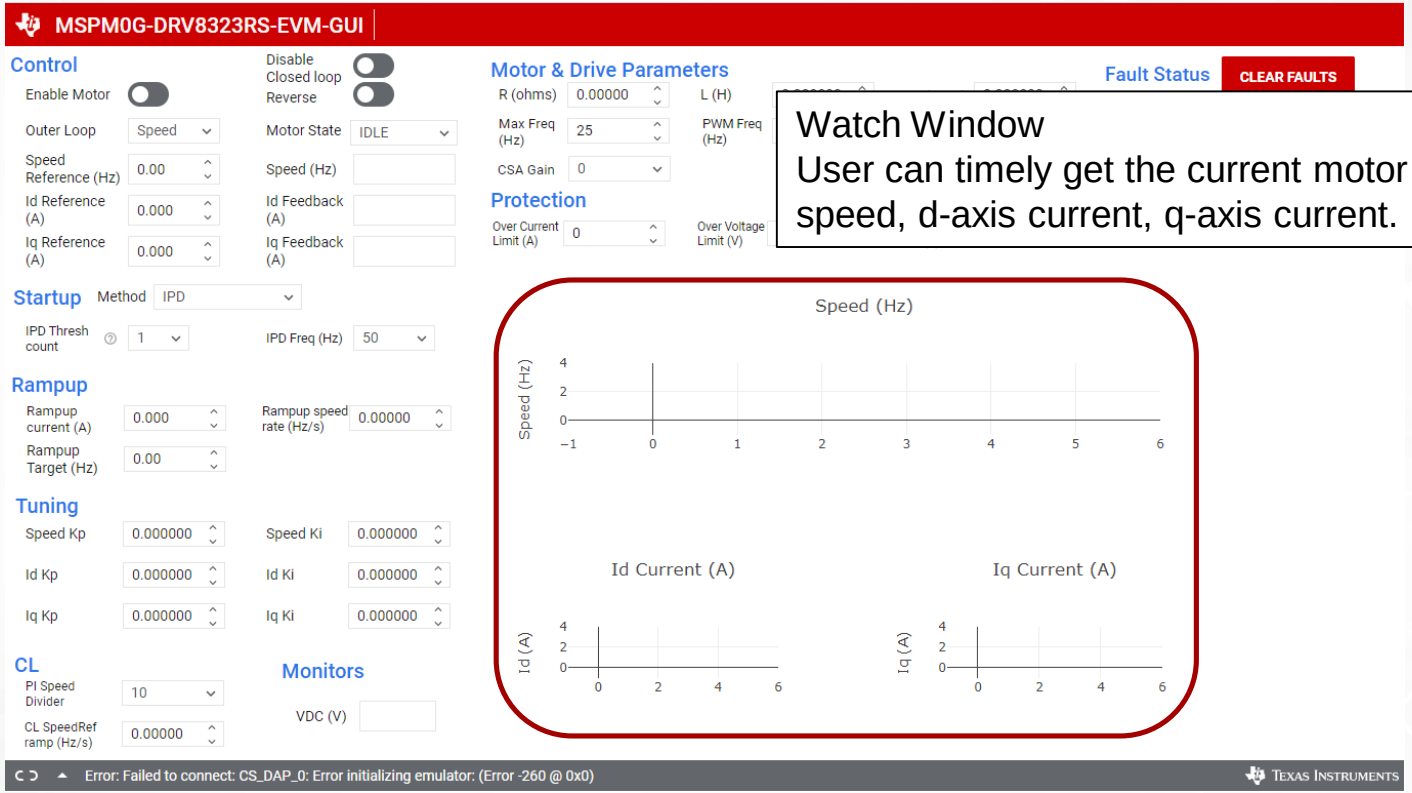



Fault status monitor.
It will show if there is any out of the protection limit or DRV ext fault occurs, and then stop the motor.

Click on the red button will clear the fault status if it doesn't exist.

# MSPM0G-DRV8323RS GUI

Introduction



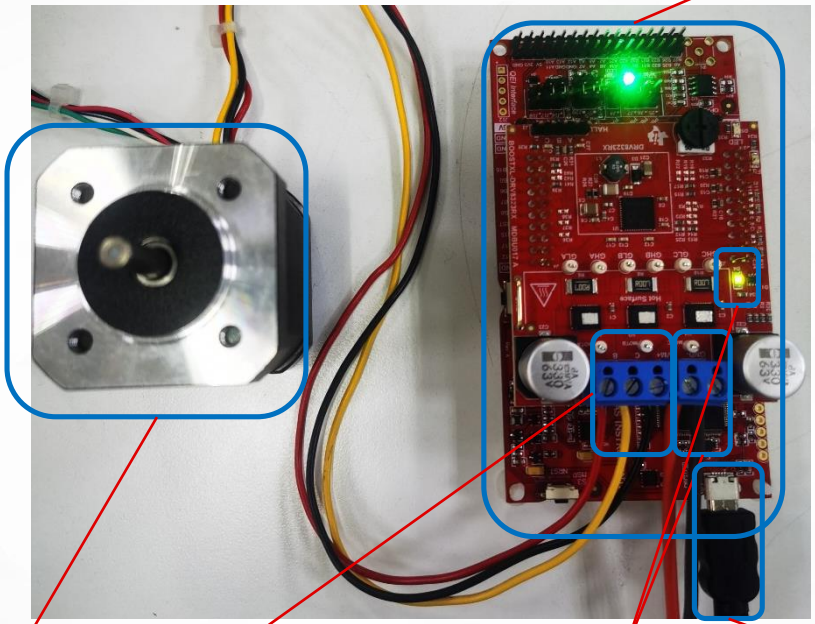

Watch Window

User can timely get the current motor status, including the speed, d-axis current, q-axis current.

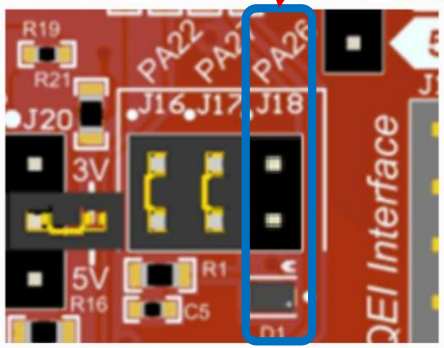
# Step by Step Operations - GUI

# Hardware Setup

MSPM0G3507 LaunchPad +
DRV8323RS EVM



Remove the PA26 jumper J18 in the LP-MSPM0G3507 as shown below




Motor

Motor Phase U/V/W

Power supply 4.5~34V

Turn on the power supply. The green VM LED on the BOOSTXL-DRV8323RS should turn on.

Connect USB to the PC for software download and debug

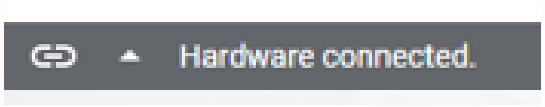
# Step2. Software setting

2-1. Connect the USB interface of LP-3507 with PC

Note:

Please power on firstly (bus dc power) and then load the program or connect board.

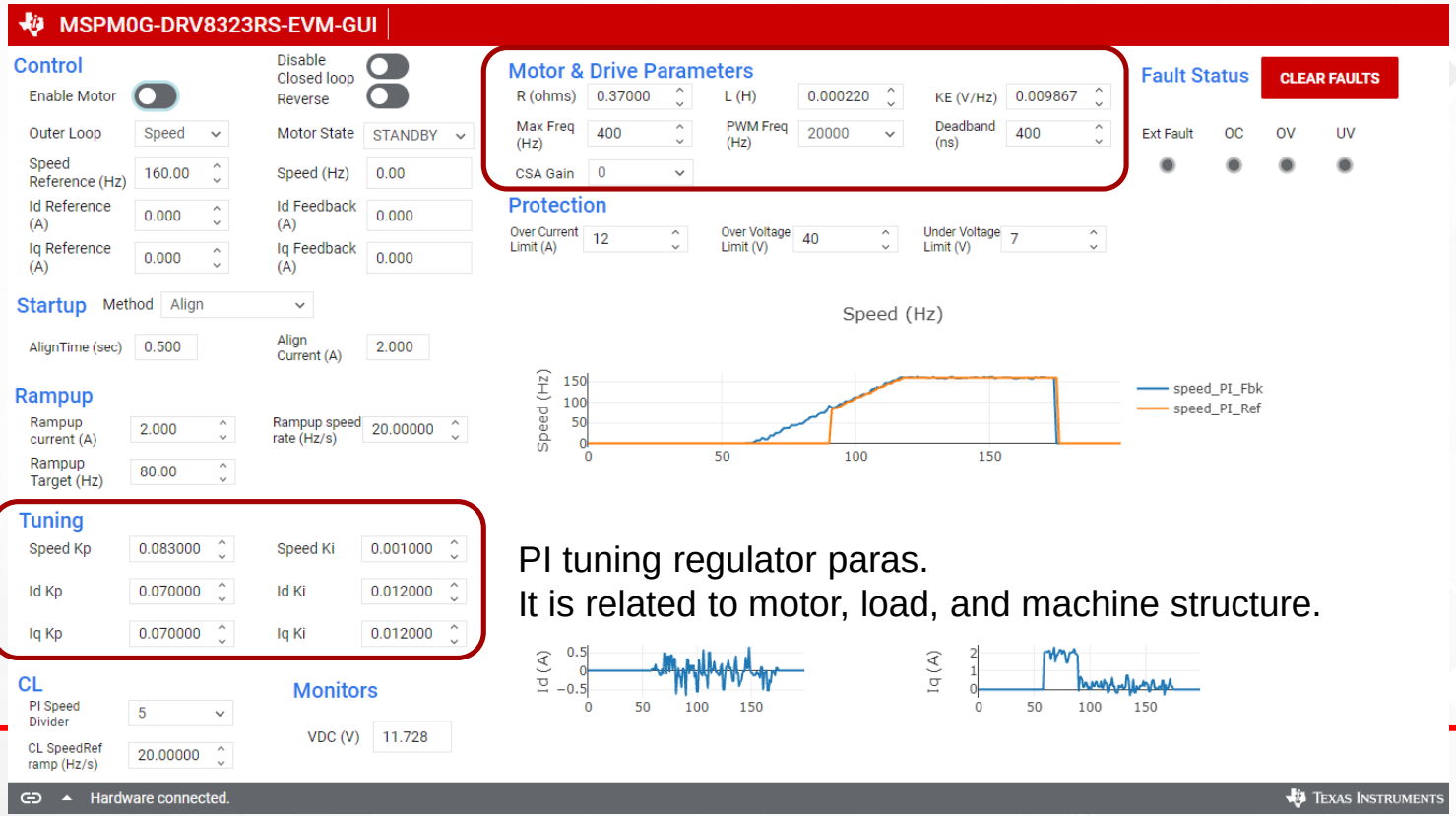
If connection error, please refresh the page and it will automatically reconnect.




Connection success

# Step2. Software setting

2-2. Set the correct parameters of the user motor and tuning params.

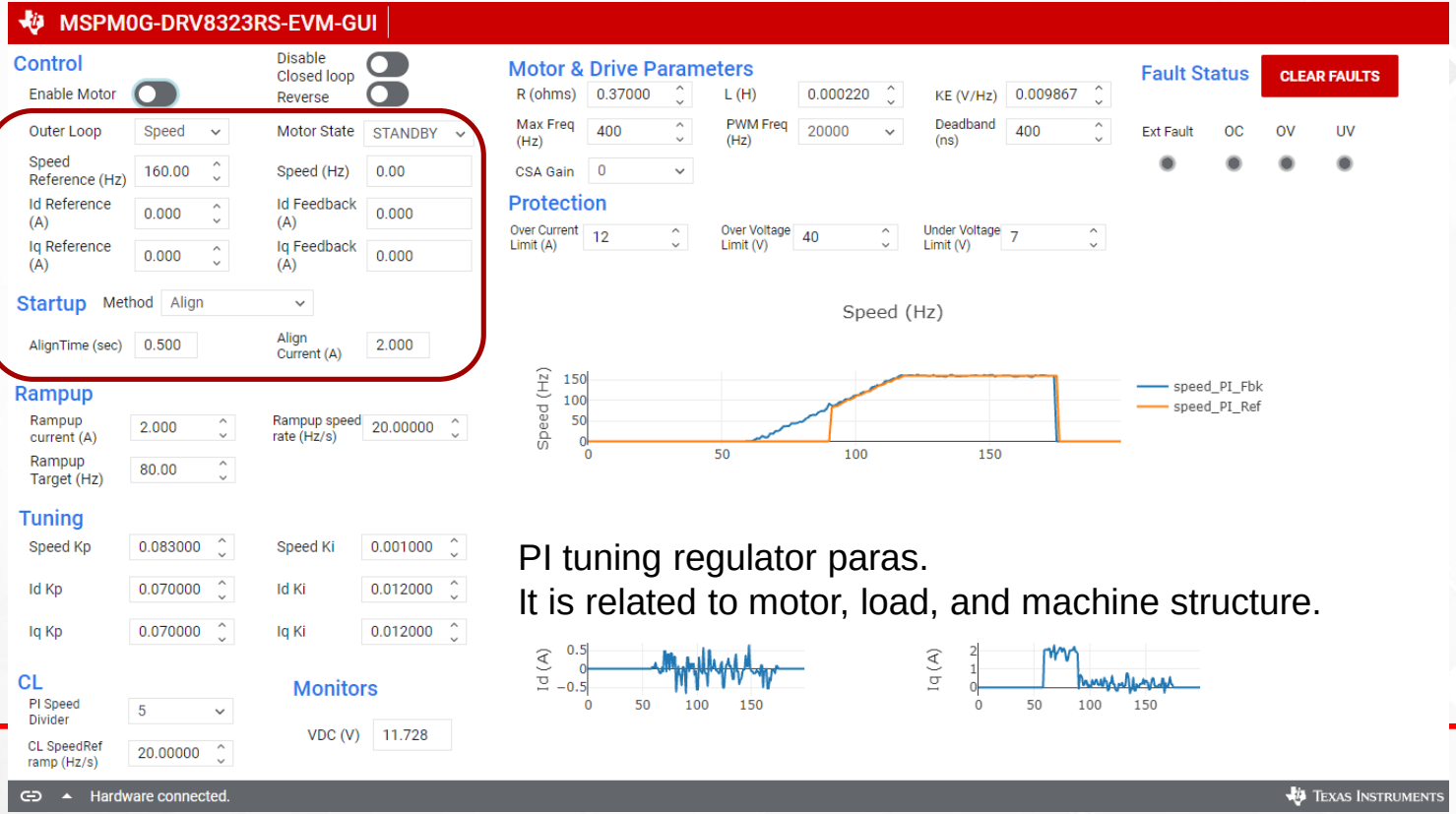



PI tuning regulator paras.
It is related to motor, load, and machine structure.

Texas Instruments

# Step2. Software setting

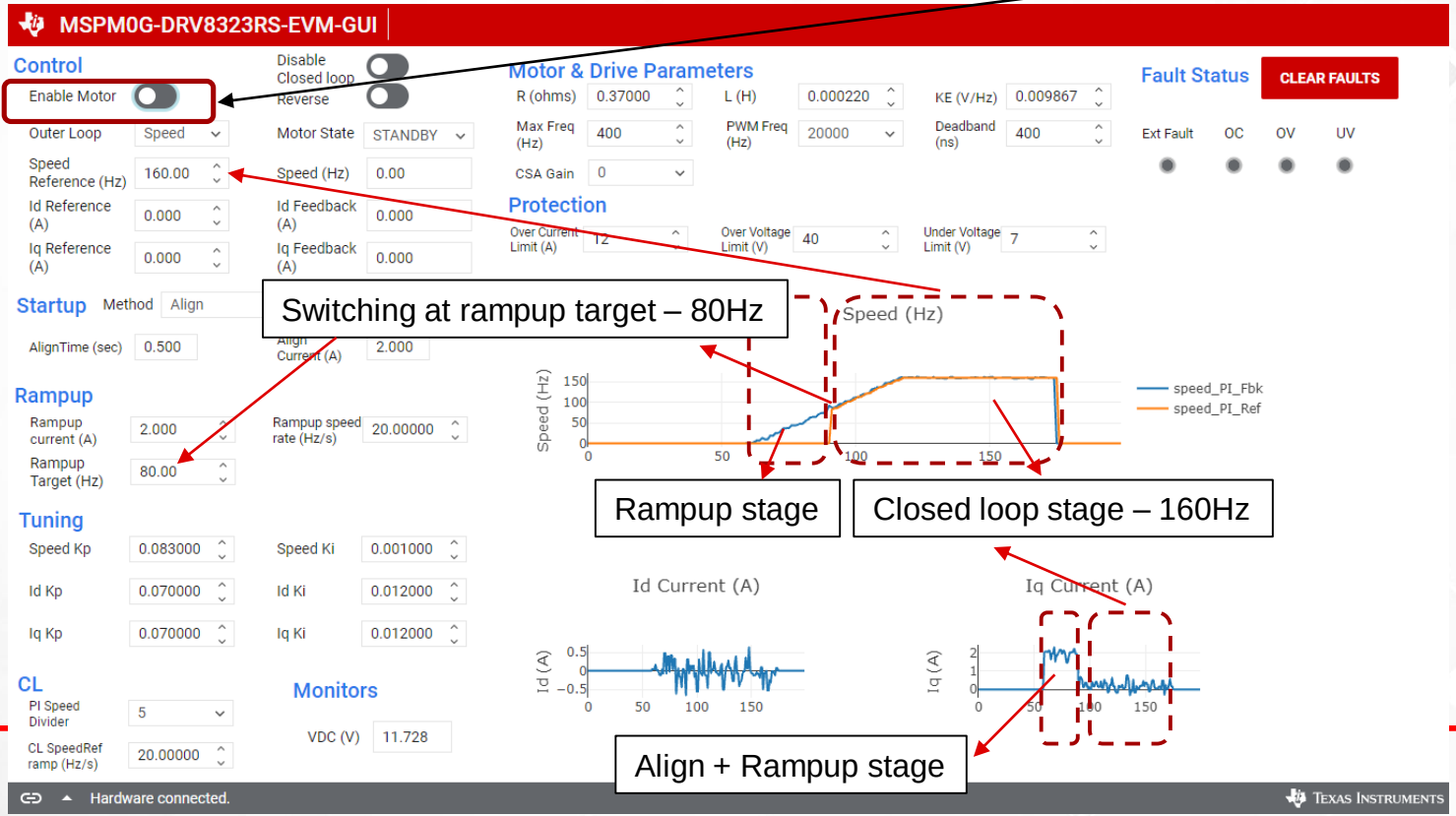
2-3. Set the appropriate control method and reference input.



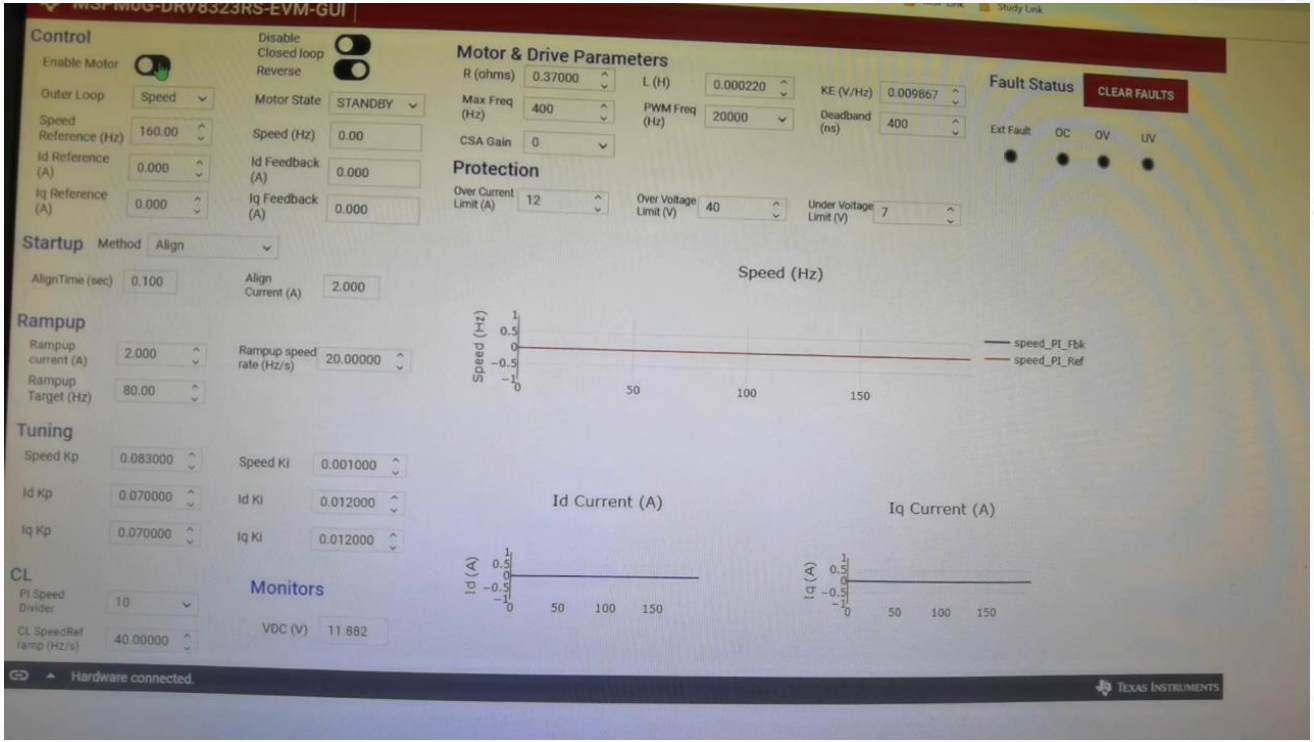

PI tuning regulator paras.
It is related to motor, load, and machine structure.

# Step3. Spin the motor

Here is the example select closed loop with speed loop, which means output of speed loop is the input of the current loop. Then click the **button** marked below to spin the motor



# Example of the GUI – Video

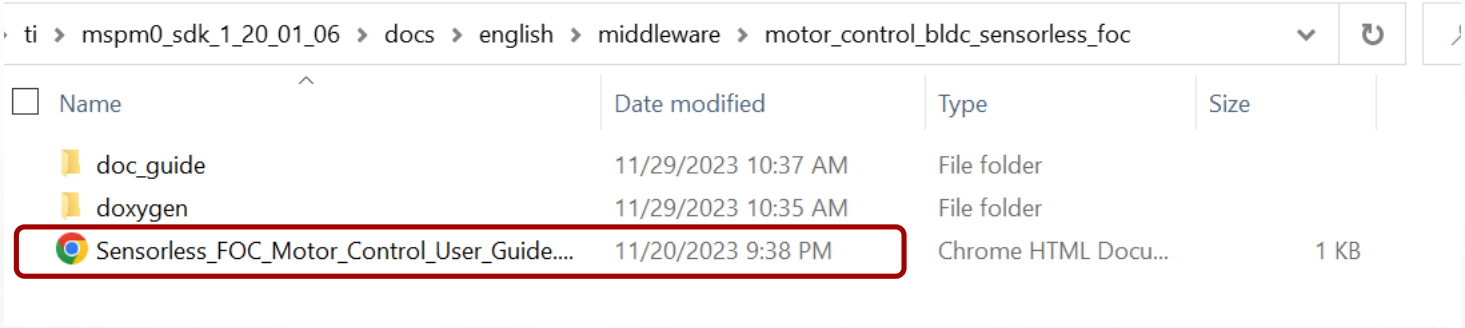


# MSPM0G-DRV8323RS CCS Project

# Environment required

| | |
|---|---|
| CCS latest version | MSPM0G3507 LaunchPad |
| SDK latest version | DRV8323RSEVM |

# Document for the foc sensorless solution



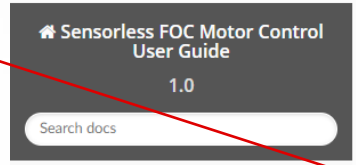
# MSPM0G-DRV8323RS GUI

Introduction for GUI: Download the SDK latest version and find the file path:

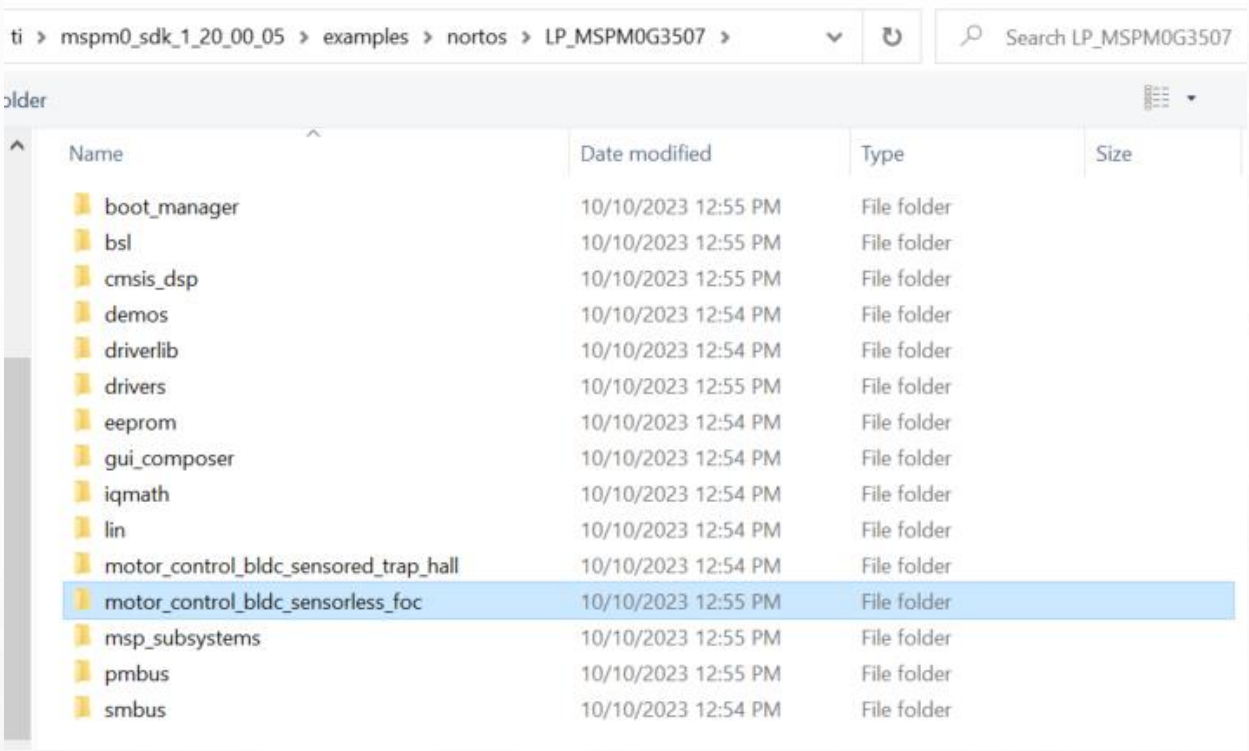
MSPM0 FOC software introduction (CCS Project)

MSPM0G-DRV8323RS GUI Introduction

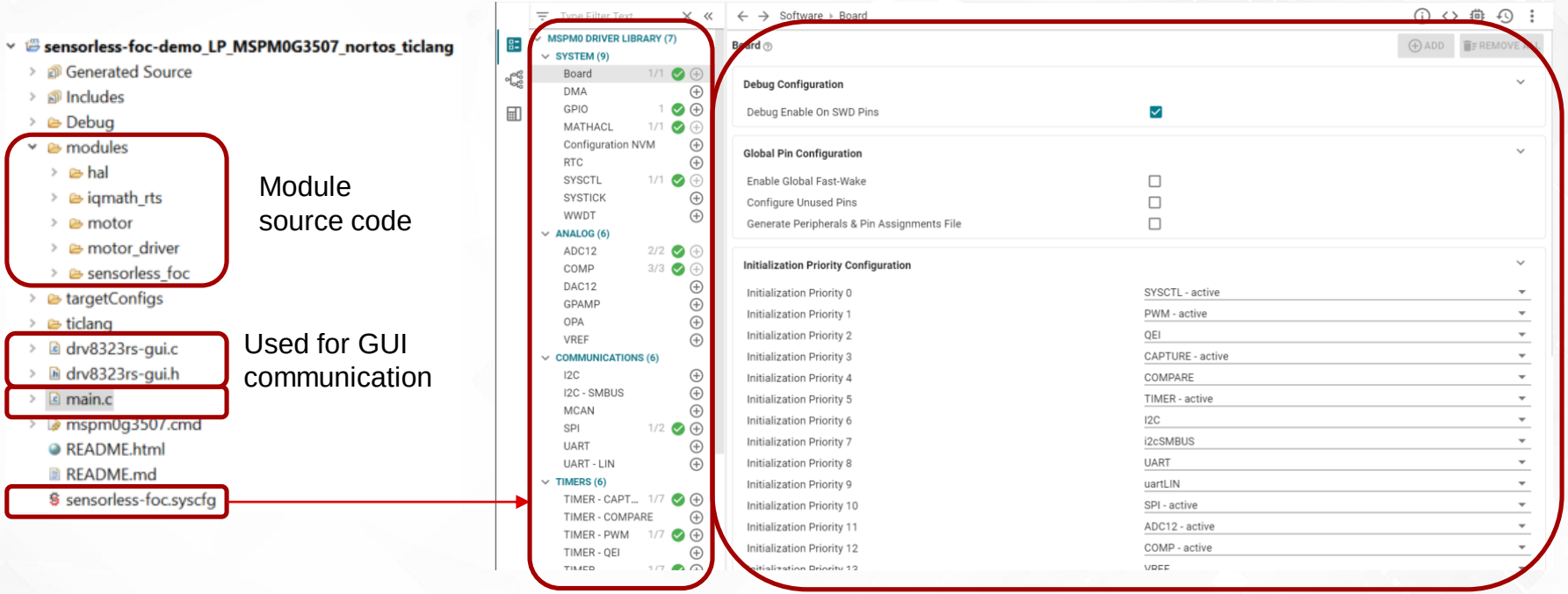
MSPM0G3507 LaunchPad and DRV8323RS EVM hardware connection introduction




# CCS Project Path



# CCS Project Overall




Sysconfig for peripheral setting
(automatically generate the code)

# Software structure

Main.c

system initialization

main loop

Update FOC control parameters
Handle the GUI command

Polling Task

1ms timer interrupt

FOC loop (Handle the state machine)

100us PWM interrupt

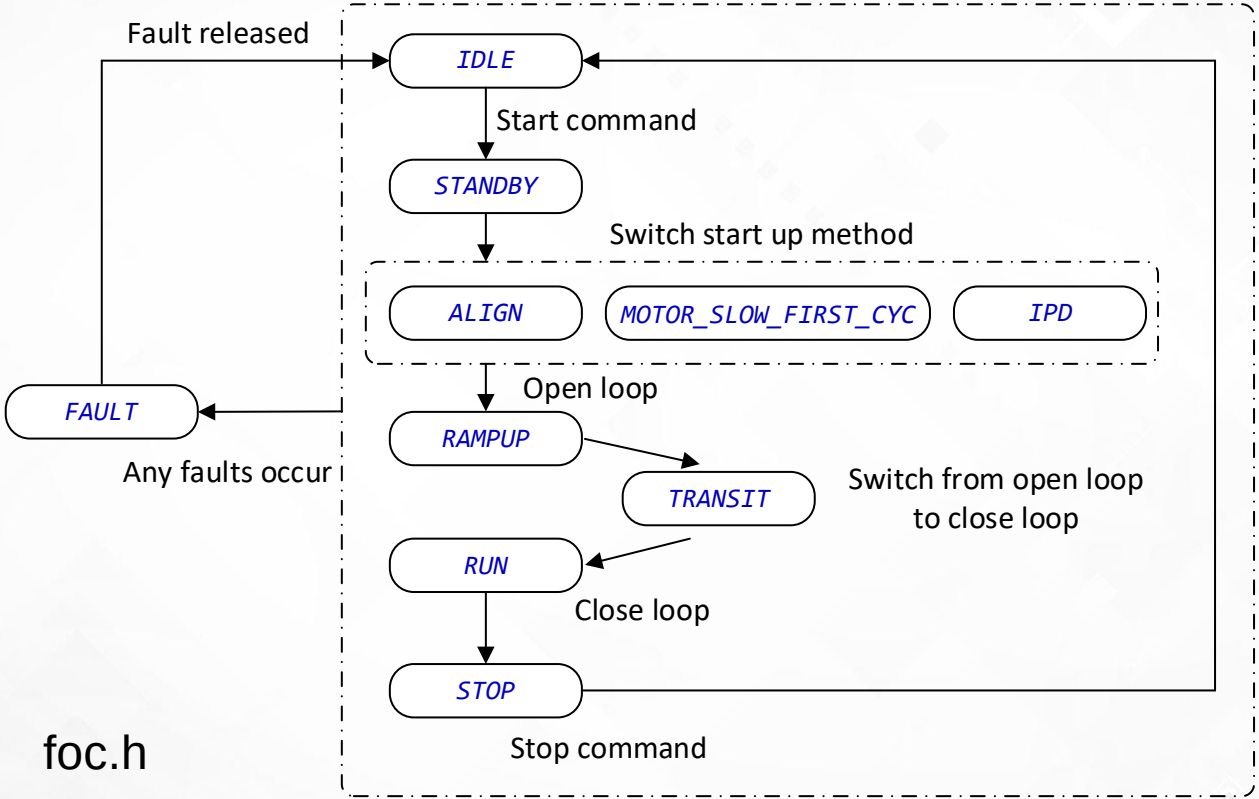
FOC run (Handle the foc algorithm)

ADC sampling interrupt

Update the adc sampling value

Interrupt Task

# State Machine




foc.h

FOC sensorless flow:

Three stage start with multi-state machines.

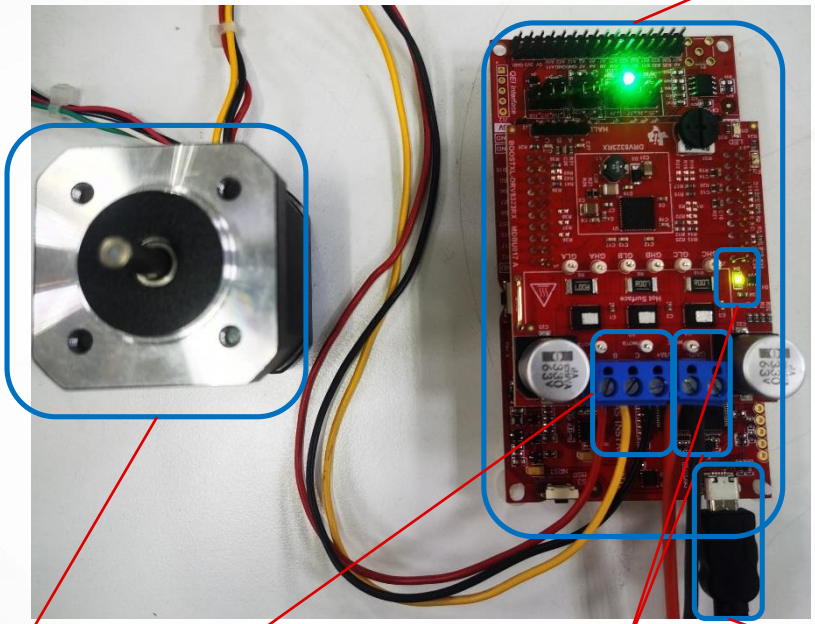
Based on SDK 1.20.01 version

# Step by Step Operations - CCS Project

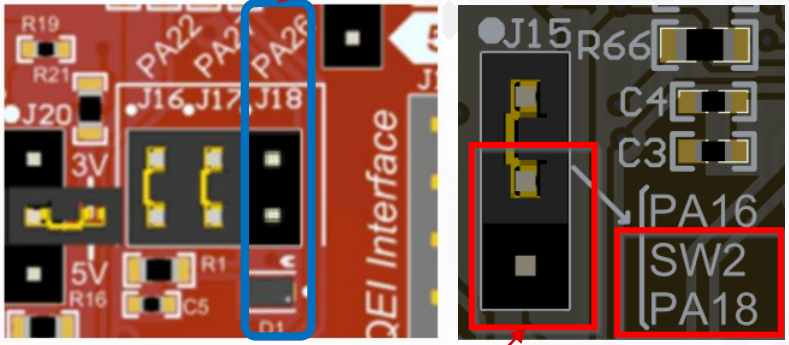
Note:
The instructions are included in ***Sensorless FOC Motor Control User Guide***

# Hardware Setup

MSPM0G3507 LaunchPad +
DRV8323RS EVM



Remove the PA26 jumper J18 in the LP-MSPM0G3507 as shown below




Connect PA18 with SW2 for ISENC In Jumper J15

Motor

Motor Phase U/V/W

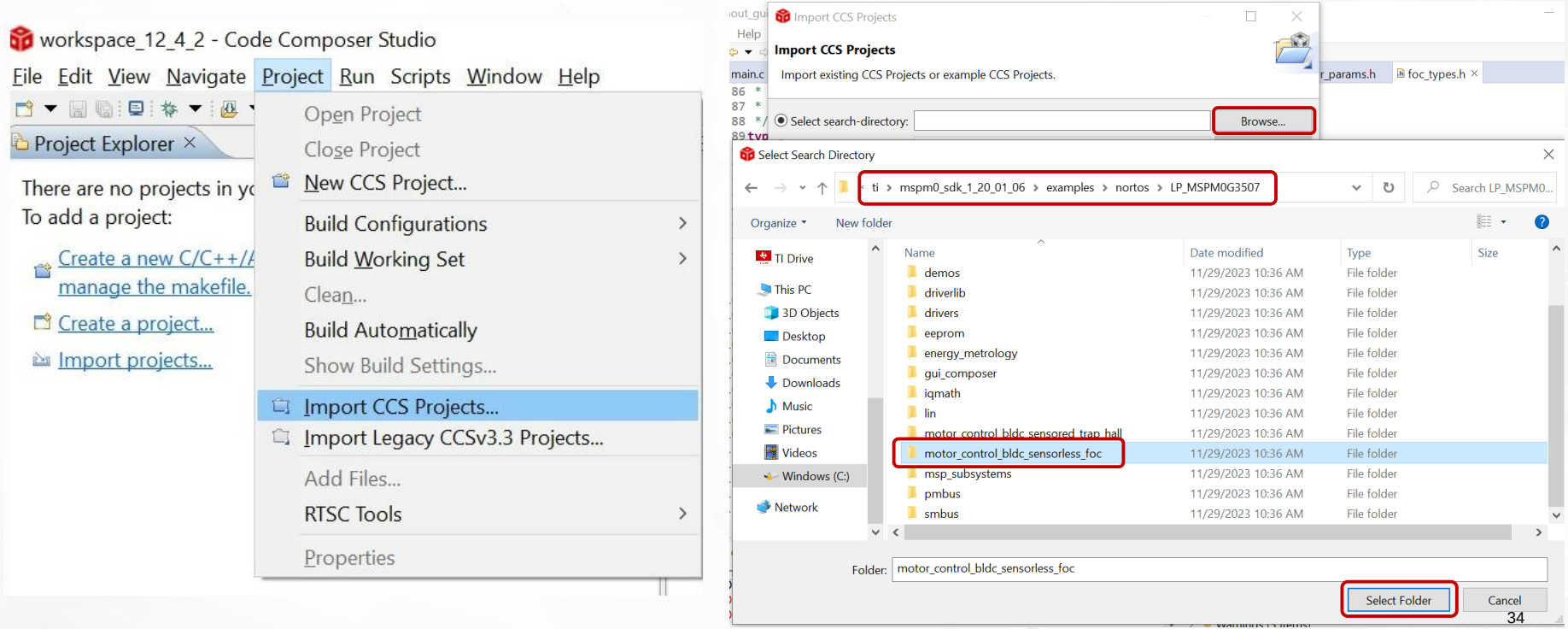
Power supply 4.5~34V

Turn on the power supply. The green VM LED on the BOOSTXL-DRV8323RS should turn on.

Connect USB to the PC for software download and debug

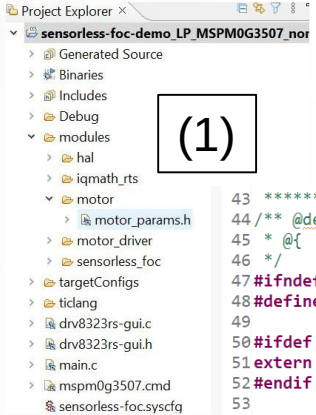
# CCS Project setup

1. Import CCS Projects as shown below



# CCS Project setup

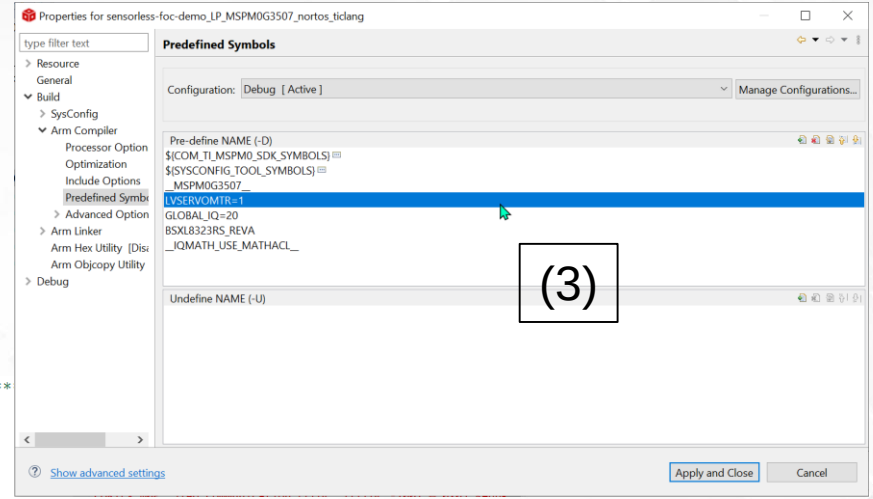
## 2. Setting the Motor Parameters




(1)

```
43 ************************************************************
44/** @defgroup MOTOR_MODULE Motor_Module
45 * @{
46 */
47#ifndef MOTOR_PARAMS_H
48#define MOTOR_PARAMS_H
49
50#ifdef __cplusplus
51extern "C" {
52#endif
53
54#if (LVSERVOMTR)
55/* Define the electrical motor parameters (Teknic 2310 Servomotor) */
56/* @brief Stator resistance (ohm) */
57#define MOTOR_PARA_RS                (0.38)
58/* @brief Stator inductance (H) */
59#define MOTOR_PARA_LS                (0.000169)
60/* @brief Number of poles */
61#define MOTOR_PARA_POLES             (8)
62/* @brief Back Emf constant in V/Hz */
63#define MOTOR_PARA_KE                (0.069)
64/* @brief Maximum Frequency */
65#define MOTOR_PARA_MAX_FREQ          (400)
66
```

(2)



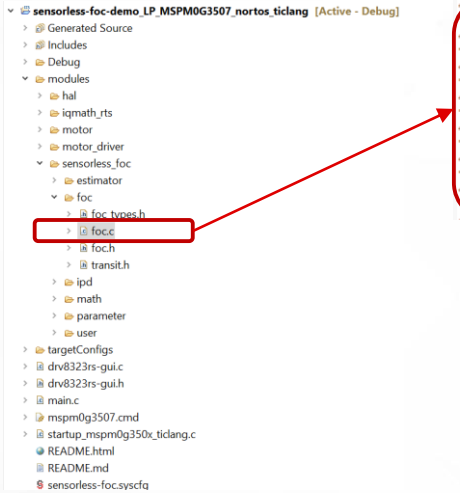

(3)

The motor parameter can be found in the motor_params.h file. These are the motor parameters required for Sensorless FOC:

- Stator resistance, Rs (ohm)
- Stator inductance, Ls (H)
- Number of motor poles (optional)
- Back Emf constant in V/Hz
- Maximum Frequency of the motor (Hz)

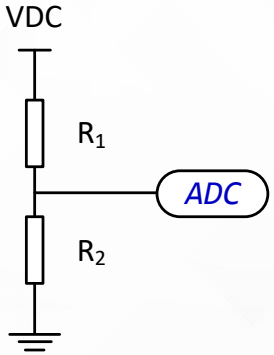
# CCS Project setup

3. Setting the circuit parameters (if use DRV8323RS EVM, then skip this step)

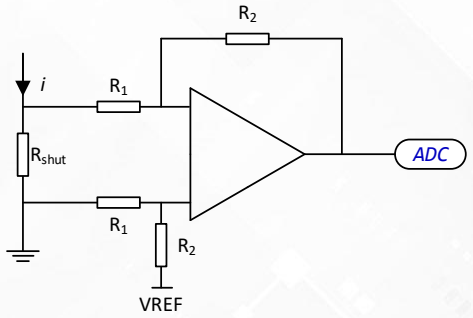



```
/** @brief Computing voltage scale factor */
#define USER_DEFAULT_VOLTAGE_SF                 (HAL_ADC_DEFAULT_REF_VOLTAGE * \
                                                (USER_DEFAULT_FOC_VOLT_RATIO))

/** @brief Computing current scale factor */
#define USER_DEFAULT_CURRENT_SF                 (HAL_ADC_DEFAULT_REF_VOLTAGE/ \
                        (2 * USER_DEFAULT_DRV_RSHUNT * USER_DEFAULT_BASE_CSA_GAIN))
```



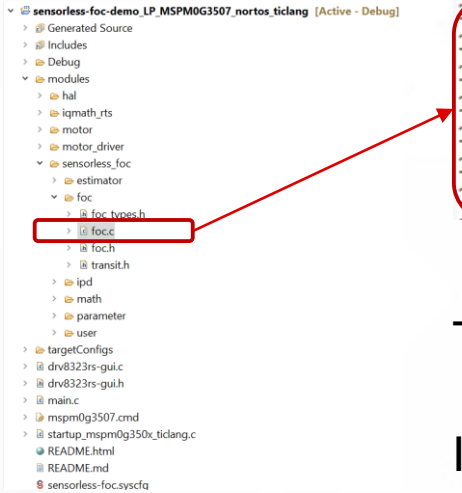

Voltage scale circuit




Current scale circuit

# CCS Project setup

3. Setting the circuit parameters



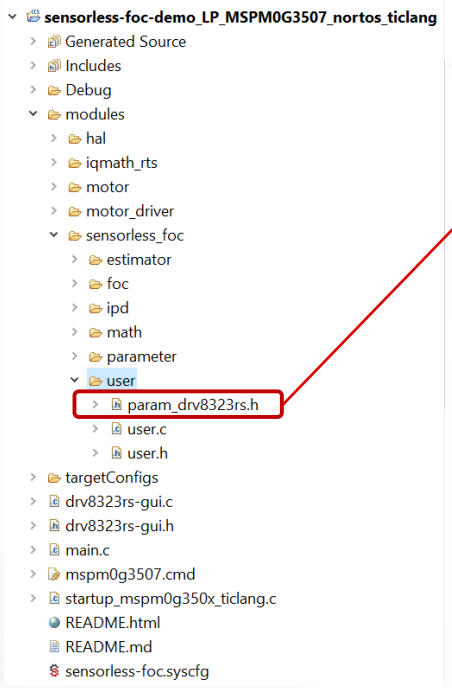

```
126 /** @brief Computing voltage scale factor */
127 #define USER_DEFAULT_VOLTAGE_SF                 (HAL_ADC_DEFAULT_REF_VOLTAGE * \
128                                                 (USER_DEFAULT_FOC_VOLT_RATIO))
129
130 /** @brief Computing current scale factor */
131 #define USER_DEFAULT_CURRENT_SF                 (HAL_ADC_DEFAULT_REF_VOLTAGE/ \
132                                                 (2 * USER_DEFAULT_DRV_RSHUNT * USER_DEFAULT_BASE_CSA_GAIN))
```

The reason that there needs “2” for current scale:

In the current calculation formula, it has divided by “2”, so here gives a compensation.

# CCS Project setup

## 3. Setting the circuit parameters

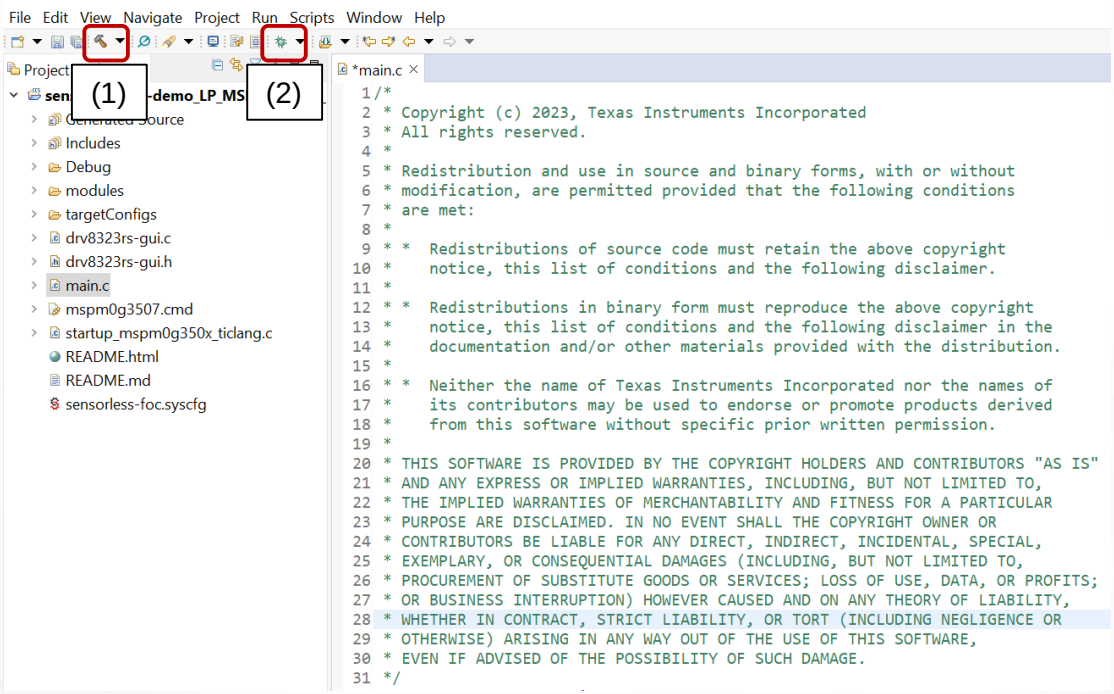



Basically, the foc control useful parameters is included in param_drv8323rs.h, include the voltage/current scale circuit macro definition params; setting params in each motor control state, …etc.

Please refers to "***MSPM0 Sensorless FOC Software User's Guide***" for details instruction, or refers to the note in param_drv8323rs.h.

# CCS Project run

## 1. Enter debug mode to run the demo




Connect the hardware and turn on the power supply.

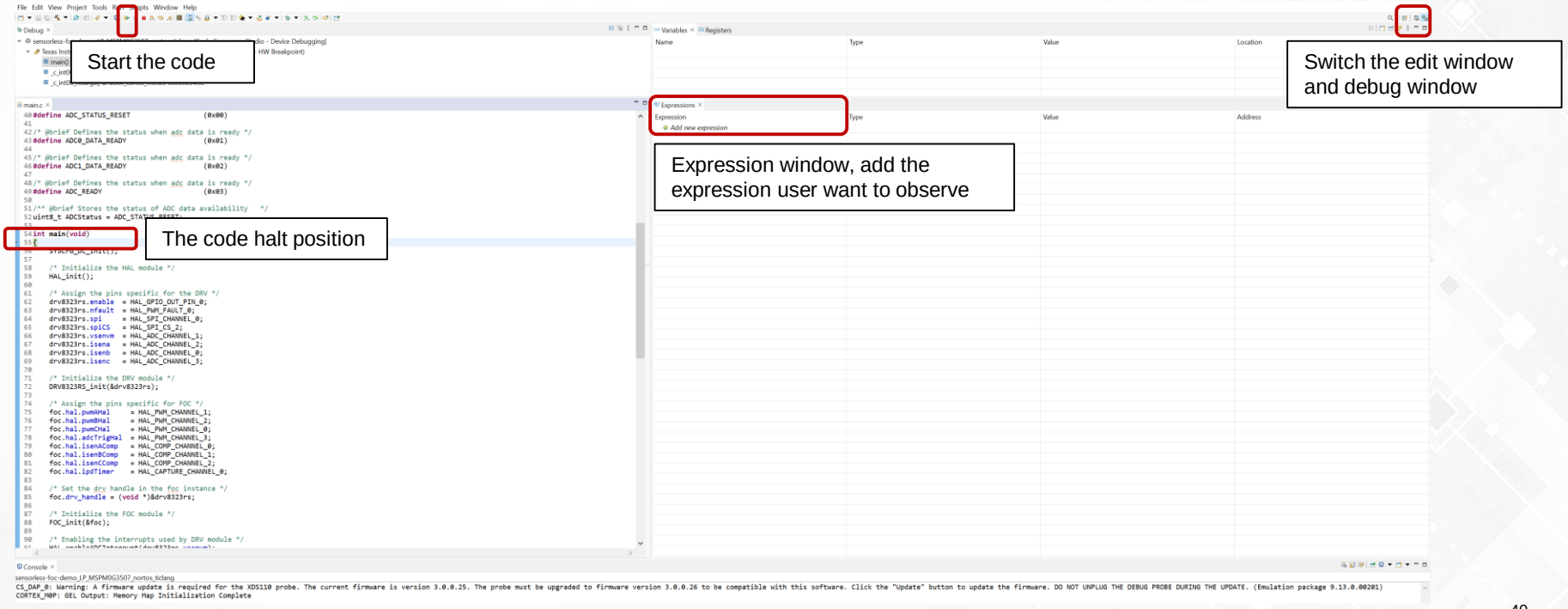
Click on build button.

- Project should build with no errors.

Click on debug button.

# CCS Project run

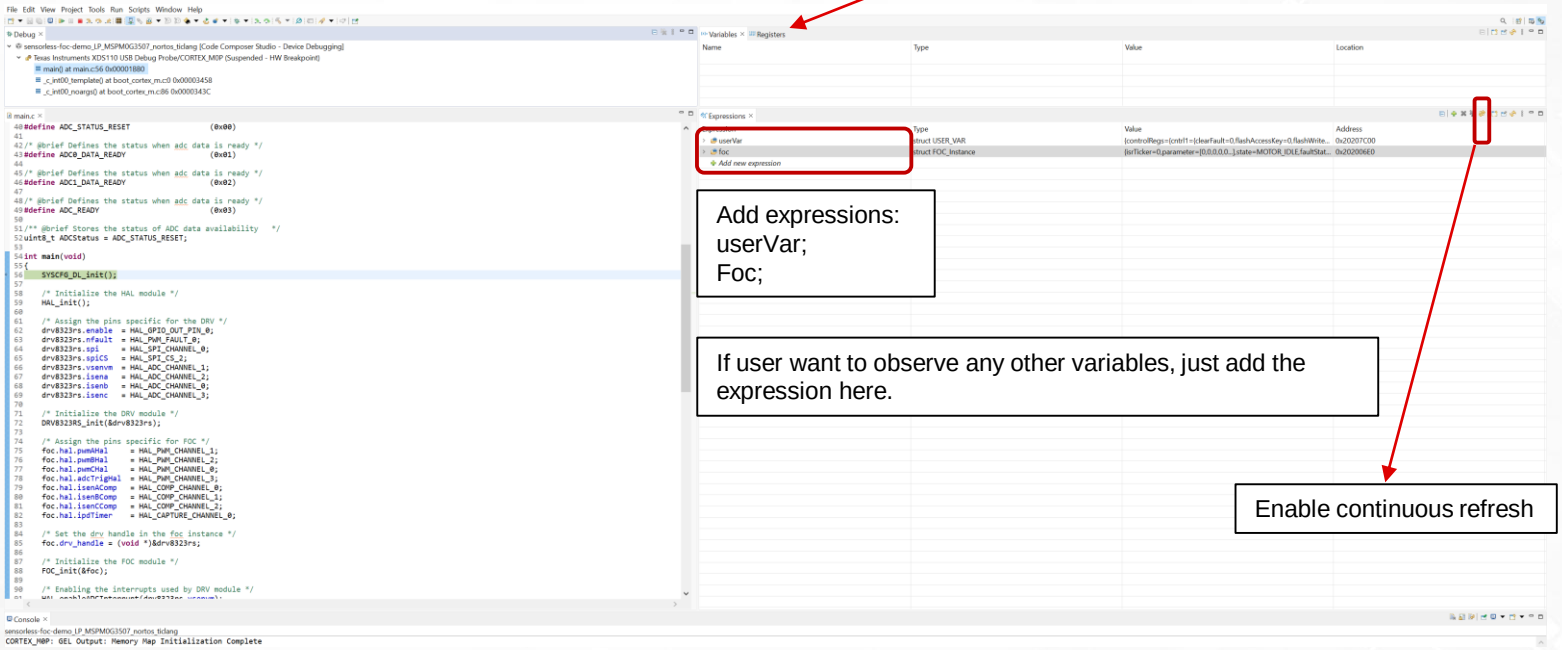
## 1. Enter debug mode to run the demo



# CCS Project run

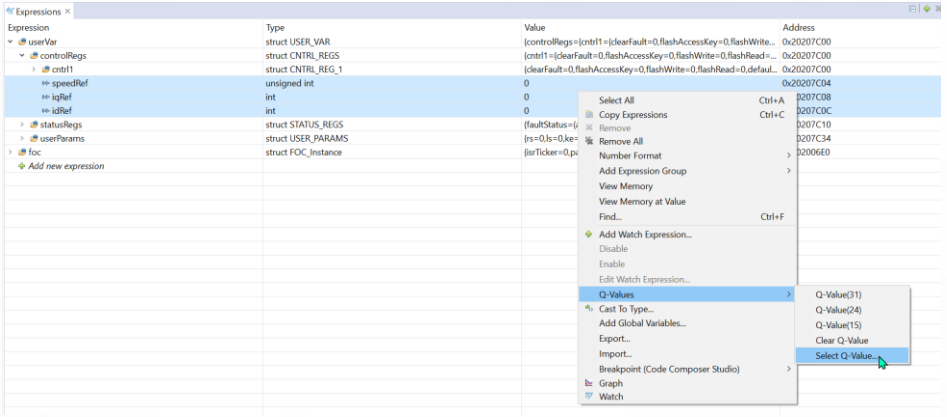
## 2. Add the expression for observing

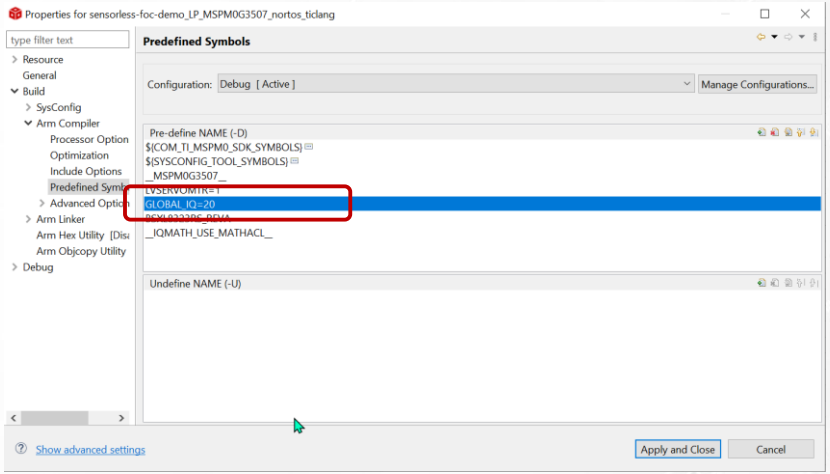
If user want to observe register real-time value, open the Register window



# CCS Project run

2. Add the expression for observing




Some of the variables are in IQ20 format (Global default IQ format).
In order to be displayed and updated properly during debug, you should switch them to IQ(20) format in the CCS expressions window by right clicking the expression, select Q-Values, then click “Select Q-Value”.

# CCS Project run

## 3. Running the project

- Start the code 

You should turn on the power firstly and then start, or the current offset detection might be wrong.
Below is an example of the watch window:

Expressions

| Expression | Type | Value | Address |
|---|---|---|---|
| userVar | struct USER_VAR | {controlRegs={cntrl1={clearFault=0,flashAc... | 0x20207C00 |
| foc | struct FOC_Instance | {isrTicker=1894372,parameter={398458,1... | 0x202006E0 |
| isrTicker | unsigned int | 1894690 | 0x202006E0 |
| parameter | unsigned int[34] | {398458,177,72351,400,0...} | 0x202006E4 |
| state | enum MOTOR | MOTOR_STANDBY | 0x2020076C |
| faultStatus | union FAULT_STATUS | {all=0,bits={extFaultIn=0,uvlo=0,ovlo=0,o... | 0x20200770 |
| cmd | struct COMMAND | {motorState=MOTOR_STATE_STOP,IqRef=... | 0x20200774 |
| control | struct CONTROL | {align={cnt=0,alignTime=500,alignCurr=2... | 0x20200790 |
| vdcAdc | unsigned int | 843 | 0x202007C8 |
| iaAdc | unsigned int | 2069 | 0x202007CC |
| ibAdc | unsigned int | 2050 | 0x202007D0 |
| icAdc | unsigned int | 2070 | 0x202007D4 |
| offsetCalib | unsigned char | 1 '\x01' | 0x202007D8 |
| offsetA | unsigned int | 2067 | 0x202007DC |
| offsetB | unsigned int | 2054 | 0x202007E0 |
| offsetC | unsigned int | 2062 | 0x202007E4 |
| vdc | int | 11.79782772 (Q-Value(20)) | 0x202007E8 |
| ia | int | 0.1381130219 (Q-Value(20)) | 0x202007EC |
| ib | int | -0.09207630157 (Q-Value(20)) | 0x202007F0 |
| overVoltageLimit | int | 40.0 (Q-Value(20)) | 0x202007F4 |
| underVoltageLimit | int | 7.0 (Q-Value(20)) | 0x202007F8 |
| overCurrentLimit | int | 12.0 (Q-Value(20)) | 0x202007FC |
| PISpdExecDivider | enum FOC_SPD_DIV | FOC_SPD_DIV_10 | 0x20200800 |
| PISpdExecCount | unsigned int | 0 | 0x20200804 |
| enablePWM | unsigned char | 0 '\x00' | 0x20200808 |
| enablePWMStatus | unsigned char | 0 '\x00' | 0x20200809 |
| csaDiv | unsigned char | 0 '\x00' | 0x2020080A |
| drv_handle | void * | 0x20200B0C | 0x2020080C |
| hal | struct FOC_HAL | {pwmAHal=HAL_PWM_CHANNEL_1,pwm... | 0x20200810 |

Motor status and Fault status

Actual current value converted by adc

Current offset for bipolar current sensing

Detected voltage / current value for motor control

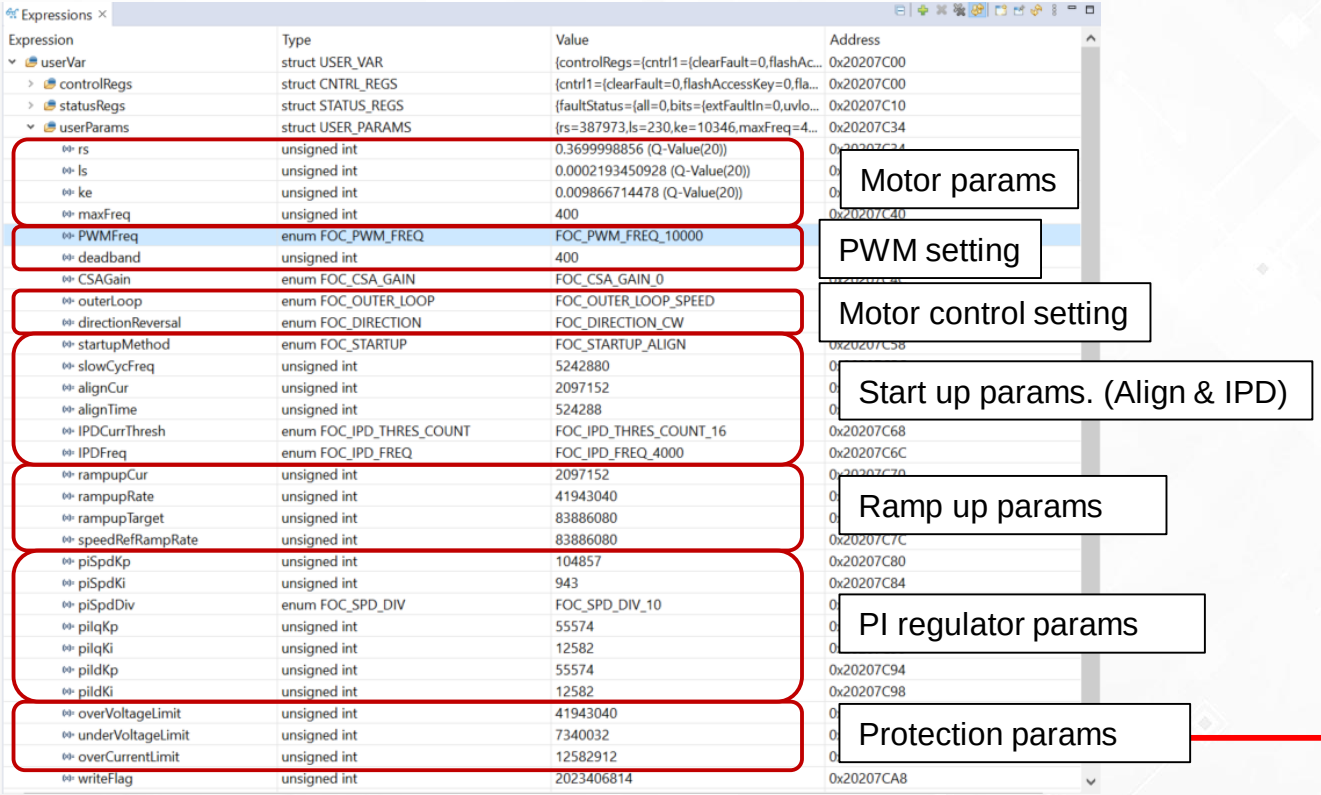
Protection parameters

**Note: FOC algorithm only use two phase currents.**

# CCS Project run

User can change the params during the debug mode, select *userVar.userParams*

## 4. Modify the parameters



# CCS Project run

User can change the params during the debug mode, select *userVar.userParams*

## 4. Modify the parameters

All the data listed in *userVar.userParams*, user can set it default value in the corresponding header files and macro definitions.

**CCS Project setup** describe some parameters definition.

If user can't find the definition you want to modify, you can use *userVar.userParams* to change during debug mode. It doesn't require reloading the program or restart the code, modifications will automatically take effect.

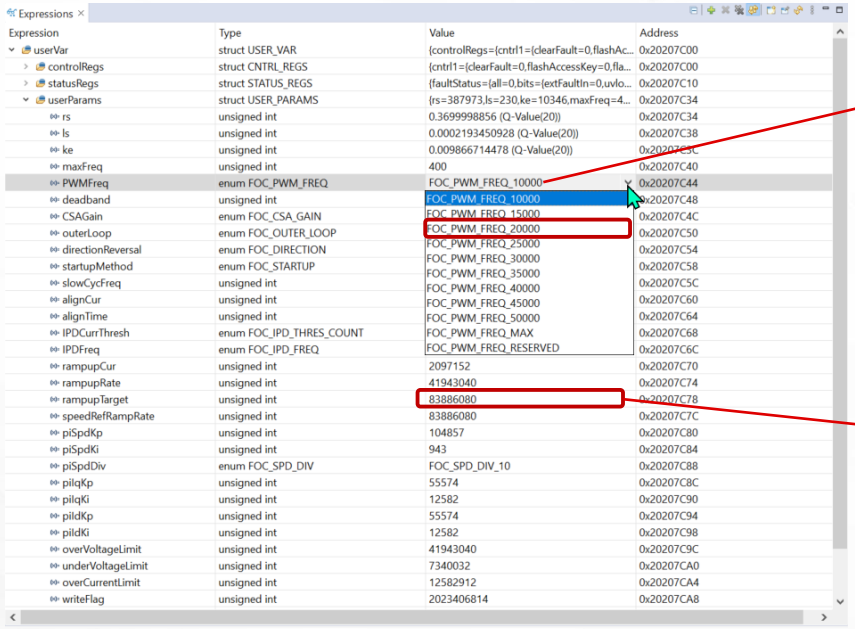
If user spin the motor through the GUI firstly, then he can use the same parameters in the GUI here.

# CCS Project run

User can change the params during the debug mode, select *userVar.userParams*

4. Modify the parameters

Examples:




An example to change the PWM frequency.
1. Double click on the value
2. Select the frequency you want

An example to change the rampup target speed
1. Select the correct IQ format for the data (IQ20)
2. Double click on the value
3. Input the value you want by keyboard

# CCS Project run

## 5. Spin the motor – open loop

User can control motor status by *userVar.controlRegs*

Expressions

| Expression | Type | Value | Address |
|---|---|---|---|
| userVar | struct USER_VAR | {controlRegs={cntrl1={clearFault=0,flashAc... | 0x20207C00 |
| controlRegs | struct CNTRL_REGS | {cntrl1={clearFault=0,flashAccessKey=0,fla... | 0x20207C00 |
| cntrl1 | struct CNTRL_REG_1 | {clearFault=0,flashAccessKey=0,flashWrite... | 0x20207C00 |
| clearFault | unsigned int : 1 | 0 | 0x20207C00 bit 0 |
| flashAccessKey | unsigned int : 4 | 0 | 0x20207C00 bit 1-4 |
| flashWrite | unsigned int : 1 | 0 | 0x20207C00 bit 5 |
| flashRead | unsigned int : 1 | 0 | 0x20207C00 bit 6 |
| defaultRead | unsigned int : 1 | 0 | 0x20207C00 bit 7 |
| disCL | unsigned int : 1 | 1 | 0x20207C00 bit 8 |
| enableMotor | unsigned int : 1 | 1 | 0x20207C00 bit 9 |
| speedRef | unsigned int | 0.0 (Q-Value(20)) | 0x20207C04 |
| iqRef | int | 0.0 (Q-Value(20)) | 0x20207C08 |
| idRef | int | 0.0 (Q-Value(20)) | 0x20207C0C |
| statusRegs | struct STATUS_REGS | {faultStatus={all=0,bits={extFaultIn=0,uvlo... | 0x20207C10 |
| faultStatus | union FAULT_STATUS | {all=0,bits={extFaultIn=0,uvlo=0,ovlo=0,o... | 0x20207C10 |
| motorState | enum MOTOR | MOTOR_RAMPUP | 0x20207C14 |
| vdc | int | 11.91018772 (Q-Value(20)) | 0x20207C18 |
| speedFbk | int | 79.88662148 (Q-Value(20)) | 0x20207C1C |
| speedRef | int | 0.0 (Q-Value(20)) | 0x20207C20 |
| iqCurr | int | 1.933246613 (Q-Value(20)) | 0x20207C24 |
| iqCurrRef | int | 2.0 (Q-Value(20)) | 0x20207C28 |
| idCurr | int | 0.1720647812 (Q-Value(20)) | 0x20207C2C |
| idCurrRef | int | 0.0 (Q-Value(20)) | 0x20207C30 |
| userParams | struct USER_PARAMS | {rs=387973,ls=230,ke=10346,maxFreq=4... | 0x20207C34 |
| foc | struct FOC_Instance | {isrTicker=29192175,parameter=[387973,... | 0x202006E0 |
| isrTicker | unsigned int | 29193181 | 0x202006E0 |
| parameter | unsigned int[34] | [387973,230,10346,400,2...] | 0x202006E4 |
| state | enum MOTOR | MOTOR_RAMPUP | 0x2020076C |
| faultStatus | union FAULT_STATUS | {all=0,bits={extFaultIn=0,uvlo=0,ovlo=0,o... | 0x20200770 |
| cmd | struct COMMAND | {motorState=MOTOR_STATE_START,IqRef... | 0x20200774 |
| control | struct CONTROL | {align={cnt=500,alignTime=500,alignCurr... | 0x20200790 |
| vdcAdc | unsigned int | 868 | 0x202007C8 |
| iaAdc | unsigned int | 2104 | 0x202007CC |
| ibAdc | unsigned int | 1964 | 0x202007D0 |

Set disCL = 1, and then Set enableMotor =1. The motor will start to spin and end in open loop at the speed of the ramp up target you set.

User can control open loop speed by *userVar.userParams* ;

or control by *foc.control.rampup* ;

| | | |
|---|---|---|
| foc | struct FOC_Instance | {isrTicker=33144873,parameter=[387973,... |
| isrTicker | unsigned int | 33145646 |
| parameter | unsigned int[34] | [387973,230,10346,400,2...] |
| state | enum MOTOR | MOTOR_STANDBY |
| faultStatus | union FAULT_STATUS | {all=0,bits={extFaultIn=0,uvlo=0,ovlo=0,o... |
| cmd | struct COMMAND | {motorState=MOTOR_STATE_STOP,IqRef=... |
| control | struct CONTROL | {align={cnt=0,alignTime=500,alignCurr=2... |
| align | struct ALIGN_Instance | {cnt=0,alignTime=500,alignCurr=2097152} |
| rampup | struct RAMPUP_Instance | {rampupCurr=2097152,rate=41920,target... |
| rampupCurr | int | 2.0 (Q-Value(20)) |
| rate | int | 0.03997802734 (Q-Value(20)) |
| target | int | 80.0 (Q-Value(20)) |
| rampOut | int | 0 |

# CCS Project run

## 5. Spin the motor – close loop

User can control motor status by *userVar.controlRegs*

Expressions

| Expression | Type | Value | Address |
|---|---|---|---|
| userVar | struct USER_VAR | {controlRegs={cntrl1={clearFault=0,flashAc... | 0x20207C00 |
| controlRegs | struct CNTRL_REGS | {cntrl1={clearFault=0,flashAccessKey=0,fla... | 0x20207C00 |
| cntrl1 | struct CNTRL_REG_1 | {clearFault=0,flashAccessKey=0,flashWrite... | 0x20207C00 |
| clearFault | unsigned int : 1 | 0 | 0x20207C00 bit 0 |
| flashAccessKey | unsigned int : 4 | 0 | 0x20207C00 bit 1-4 |
| flashWrite | unsigned int : 1 | 0 | 0x20207C00 bit 5 |
| flashRead | unsigned int : 1 | 0 | 0x20207C00 bit 6 |
| defaultRead | unsigned int : 1 | 0 | 0x20207C00 bit 7 |
| disCL | unsigned int : 1 | 0 | 0x20207C00 bit 8 |
| enableMotor | unsigned int : 1 | 1 | 0x20207C00 bit 9 |
| speedRef | unsigned int | 160.0 (Q-Value(20)) | 0x20207C04 |
| iqRef | int | 0.0 (Q-Value(20)) | 0x20207C08 |
| idRef | int | 0.0 (Q-Value(20)) | 0x20207C0C |
| statusRegs | struct STATUS_REGS | {faultStatus={all=0,bits={extFaultIn=0,uvlo... | 0x20207C10 |
| faultStatus | union FAULT_STATUS | {all=0,bits={extFaultIn=0,uvlo=0,ovlo=0,o... | 0x20207C10 |
| motorState | enum MOTOR | MOTOR_RUN | 0x20207C14 |
| vdc | int | 11.96636772 (Q-Value(20)) | 0x20207C18 |
| speedFbk | int | 159.7398071 (Q-Value(20)) | 0x20207C1C |
| speedRef | int | 159.9963379 (Q-Value(20)) | 0x20207C20 |
| iqCurr | int | 0.06970119476 (Q-Value(20)) | 0x20207C24 |
| iqCurrRef | int | 0.2961921692 (Q-Value(20)) | 0x20207C28 |
| idCurr | int | 0.03873348236 (Q-Value(20)) | 0x20207C2C |
| idCurrRef | int | 0.0 (Q-Value(20)) | 0x20207C30 |
| userParams | struct USER_PARAMS | {rs=387973,ls=230,ke=10346,maxFreq=4... | 0x20207C34 |
| foc | struct FOC_Instance | {isrTicker=34764217,parameter=[387973,... | 0x202006E0 |
| isrTicker | unsigned int | 34765187 | 0x202006E0 |
| parameter | unsigned int[34] | [387973,230,10346,400,2...] | 0x202006E4 |
| state | enum MOTOR | MOTOR_RUN | 0x2020076C |
| faultStatus | union FAULT_STATUS | {all=0,bits={extFaultIn=0,uvlo=0,ovlo=0,o... | 0x20200770 |
| cmd | struct COMMAND | {motorState=MOTOR_STATE_START,IqRef... | 0x20200774 |
| control | struct CONTROL | {align={cnt=500,alignTime=500,alignCurr... | 0x20200790 |
| vdcAdc | unsigned int | 846 | 0x202007C8 |
| iaAdc | unsigned int | 2060 | 0x202007CC |
| ibAdc | unsigned int | 2064 | 0x202007D0 |

Set disCL = 0, set speedRef as you want, and then Set enableMotor =1.
The motor will start to spin and end in close loop at the speed of the speedRef you set.

User can change close loop speed by changing the speedRef.