MCK2812HV Evaluation Board Software Debug Instruction

1. Test Program Using CC2000 or CCS3.1

In order to test our MCK2812HV Evaluation Board, it is required to install CCS2000 2.2+ CCS environment. Since this program is developed from USB2.0 emulator, it is recommended that users choose USB emulator at their convenience. If you are using other type of emulator, please refer to relative instructions to install driver accordingly.

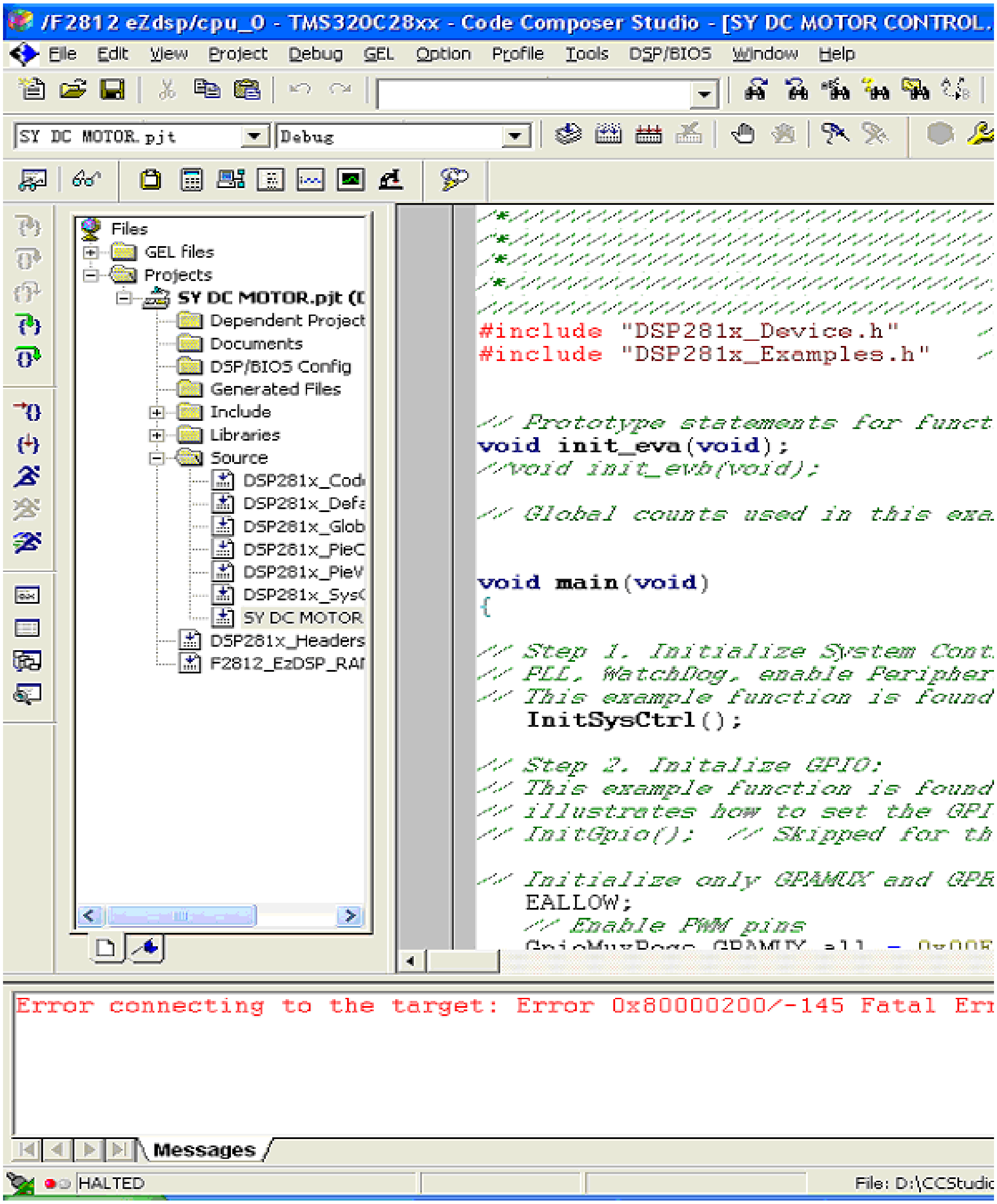
MCK2812HV evaluation board provides two testing programs: CCS2000 and CCS3.1. Because RTS28XX.lib is not compatible in CCS2000, there are two programs for your choice.

Power part of electric motor driver board is the most brittle. In order to prevent unnecessary damages, please pay attention to the following:

- Quarantine power from evaluation board. DO not connect extra leads to power and control;
- If testing program is manually stopped, and user needs to restart the program, please make sure to choose RESET CPU command in DEBUG menu. After restart command, run the program again to prevent unexpected errors.
- When using our new electric motor program, please make sure that you MUST set timer to TxCON storage.
- When debugging, try to run without the motor, to make sure that the program runs well. Then it is safe to run the program with motor is on.
- After CCS is started, check connection with emulator first. If an electric motor is required, also check electric motor connection before power is supplied, and open CCS program after it is indicated that power is ready.

2. Testing Procedure 1 Brush DC motor revolution adjustment program

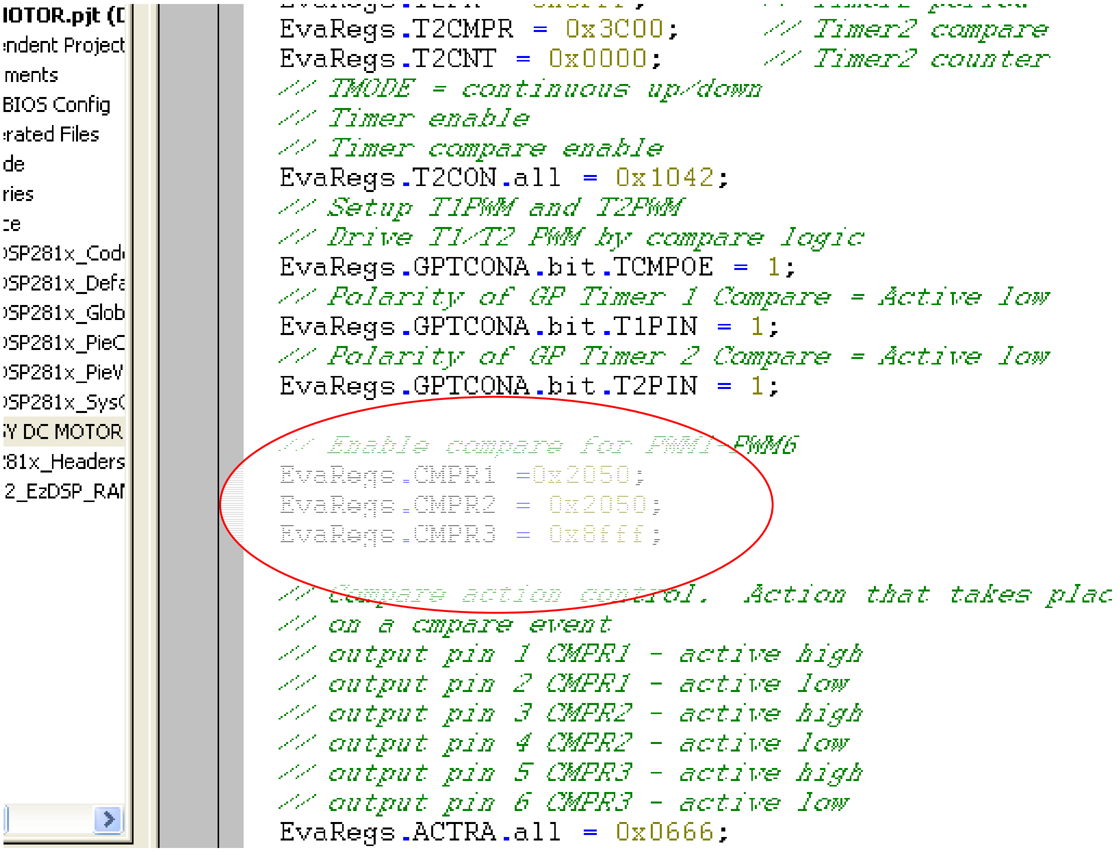
Copy SY DC MOTOR folder to X:\TI\MYPROJECTS, and open SYDC MOTOR.pjt in CCS@000/CCS3.1 environment.



```
//*///////////////////////////////////////////////////
//*///////////////////////////////////////////////////
//*///////////////////////////////////////////////////
//*///////////////////////////////////////////////////
/////////////////////////////////////////////////////
#include "DSP281x_Device.h"      //
#include "DSP281x_Examples.h"    //


// Prototype statements for funct
void init_eva(void);
//void init_evb(void);

// Global counts used in this exa


void main(void)
{

// Step 1. Initialize System Cont
// PLL, WatchDog, enable Peripher
// This example function is found
   InitSysCtrl();

// Step 2. Initalize GPIO:
// This example function is found
// illustrates how to set the GPI
// InitGpio();  // Skipped for th

// Initialize only GPAMUX and GPB
   EALLOW;
   // Enable PWM pins
   GpioMuxRegs.GPAMUX.all = 0x00F
```

```
Error connecting to the target: Error 0x80000200/-145 Fatal Err
```

Testing program 1 is a DC brush motor (with current less than 5A) demonstration program. After opening CC C2000, choose File→LOAD Program to load SY DC MOTOR.out to onboard memory. Pay attention to the jumper at MP/MC must be in MP status. In DEBUG menu, choose reset and then RUN the program. NOTE: whenever it is required to terminate running, reset DSP IMMEDIATELY (CCS→DEBUG→RESET), otherwise motor and MOS might be overheated to burn.



```
EvaRegs.T2CMPR = 0x3C00;      // Timer2 compare
EvaRegs.T2CNT = 0x0000;       // Timer2 counter
// TMODE = continuous up/down
// Timer enable
// Timer compare enable
EvaRegs.T2CON.all = 0x1042;
// Setup T1PWM and T2PWM
// Drive T1/T2 PWM by compare logic
EvaRegs.GPTCONA.bit.TCMPOE = 1;
// Polarity of GP Timer 1 Compare = Active low
EvaRegs.GPTCONA.bit.T1PIN = 1;
// Polarity of GP Timer 2 Compare = Active low
EvaRegs.GPTCONA.bit.T2PIN = 1;

// Enable compare for PWM1-PWM6
EvaRegs.CMPR1 =0x2050;
EvaRegs.CMPR2 = 0x2050;
EvaRegs.CMPR3 = 0x8fff;

// Compare action control.  Action that takes plac
// on a cmpare event
// output pin 1 CMPR1 - active high
// output pin 2 CMPR1 - active low
// output pin 3 CMPR2 - active high
// output pin 4 CMPR2 - active low
// output pin 5 CMPR3 - active high
// output pin 6 CMPR3 - active low
EvaRegs.ACTRA.all = 0x0666;
```

As circled above, these three sentences are to modify PWM, hence adjust DC brush motor's revolution speed and rotating direction.
EvaRegs.CMPRx modification range is from 0x100 to 0x8fff

EvaRegs.CMPR1 corresponds to J4 U phase.

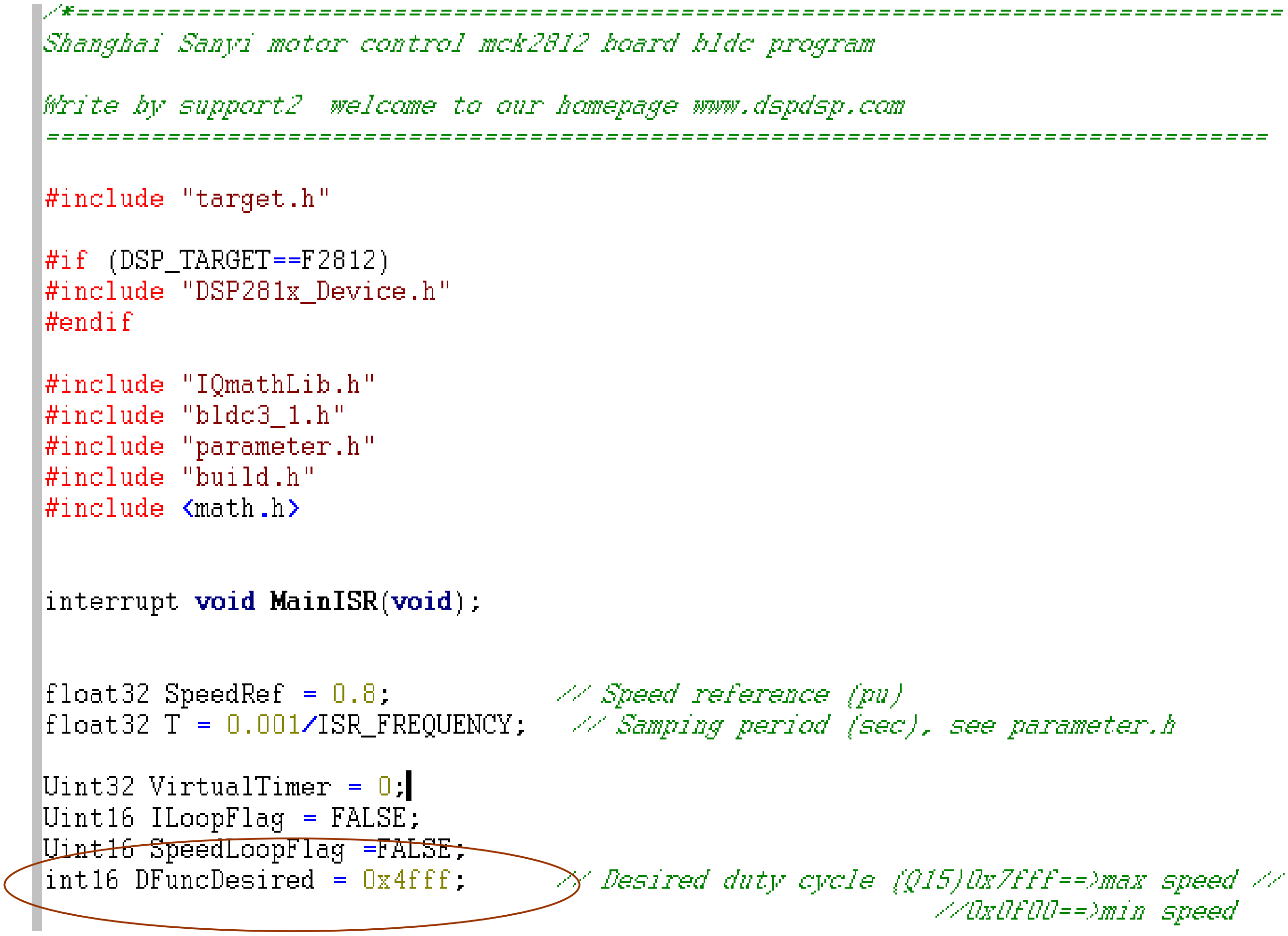
EvaRegs.CMPR2 corresponds to J4 V phase.

EvaRegs.CMPR3 corresponds to J4 W phase.

In order to run a DC brush motor, only two phases are required. Increasing the DC voltage will increase motor revolution till rating voltage. For example, U and V phases are needed. If U is power, and V is ground, just put EvaRegs.CMOR1=0x2050, EvaRegs.CMPR2=0x8fff. Changing values of CMPR1 and CMPR2 would reverse rotation.

3. Testing Program 2 – Brushless DC Motor Revolution Adjustment Program

Open BLDC.PRJ IN CCS3.1 environment. If it is in CCS2000, please open BLDC 2000.PRJ
Testing program 2 is designed for three-phase brushless DC motor (with current less than 5A) revolution adjustment. Its basic function is to adjust motor speed by modifying int16 DfuncDesired=0x4fff.
Before running the program, please make sure the connection of three-phase motor and Hole signal input(J1).
After opening CC C2000/CCS3.1, choose File→Load program, and load SY BLDC.out (CCS2000) or BLDC OPEN LOOP LIB EXCUTE.out (CCS3.1). MP/MC jumper needs to be MP.
In DEBUG menu, choose Reset before RUN this program.
NOTE1: There would be about 4 seconds waiting time delay between program initialization and motor starting. This is done to prevent burning of DSP.
NOTE2: whenever it is required to terminate running, reset DSP IMMEDIATELY (CCS→DEBUG→RESET), otherwise motor and MOS might be overheated to burn.



```
/*=========================================================================
Shanghai Sanyi motor control mck2812 board bldc program

Write by support2  welcome to our homepage www.dspdsp.com
=========================================================================

#include "target.h"

#if (DSP_TARGET==F2812)
#include "DSP281x_Device.h"
#endif

#include "IQmathLib.h"
#include "bldc3_1.h"
#include "parameter.h"
#include "build.h"
#include <math.h>


interrupt void MainISR(void);


float32 SpeedRef = 0.8;            // Speed reference (pu)
float32 T = 0.001/ISR_FREQUENCY;    // Samping period (sec), see parameter.h

Uint32 VirtualTimer = 0;
Uint16 ILoopFlag = FALSE;
Uint16 SpeedLoopFlag =FALSE;
int16 DFuncDesired = 0x4fff;       // Desired duty cycle (Q15)0x7fff==>max speed //
                                                         //0x0f00==>min speed
```

As circled above, 0x0000 corresponds lowest revolution, and 0x7fff corresponds to highest revolution. If not necessary, please do not change other parameters in this program.