DCMOTOR

System Document

C2000 Foundation Software

Table of Contents

1 System Overview

This document describes the "C" real control framework to demonstrate the DCMOTOR demo implemented using Code Composer Studio (CCS) version 2.2 (or above). The "C" framework is designed to run on TMS320C281x and TMS320C280x based controllers on CCS V2.2 (or above).

The framework uses the following modules viz.,

1. DATALOG
2. BDCPWM
3. QEP_NO_INDEX
4. RMPCNTL
5. PID_REG3
6. ESTIMATEDSPEED

In this system, the sensored drive of Brushed DC Motor (BDC) will be experimented. The user can quickly start evaluating the performance of sensored drive system by studying the speed and position controls.

The DCMOTOR demo has the following properties

C Frame work		
System Name	**Program memory usage 281x/280x**	**Data memory usage[1] 281x/280x**
DCMOTOR (IQ)	2864 words[2]/3098 words[2]	652 words/652 words

Development/Emulation	Code Composer Studio V.2.20 (or above) with Real Time debugging
Target Controller	Spectrum Digital – TMS320C281x or TMS320C280x board
Emulator	XDS510PP-PLUS (281x) / XDS510USB (280x)
PWM Frequency (281x)	40 kHz (BDCPWM, Timer 1-EVA)
(280x)	40 kHz (BDCPWM, EPWM1-2)
PWM Mode	Asymmetrical with no dead band (BDCPWM)
Interrupts (281x)	2 (Timer T1 underflow – Implements 40 kHz ISR execution rate, T2PISR)
(280x)	2 (EPWM1 Time Base CNT_zero – Implements 40 kHz ISR execution rate)
Peripheral Used (281x)	Timer T1/T2, PWM1-4
(280x)	EPWM1-2

[1] Excluding the Stack Size
[2] Excluding "IQmath" Look-up Tables

Figure 1 gives an overview of the hardware required for sensored control of brushed DC motor drives. Four PWM signals, generated by the DSP controller, are used to drive the H-bridge dc/dc converter. A pair of switches are operated depending the Rotation. For example, PWM1 and PWM4 are used to drive motor in one Rotational direction. And PWM2 and PWM3 are used to drive motor in another Rotational direction. The motor position can be controlled by using feedback from the QEP encoder. In this application, the motor is assumed to have an attached gearbox of a 256:1 ratio. That means that motor turns 256 revolutions to get 1 output revolution. The overall block diagram can be depicted in figure 2.

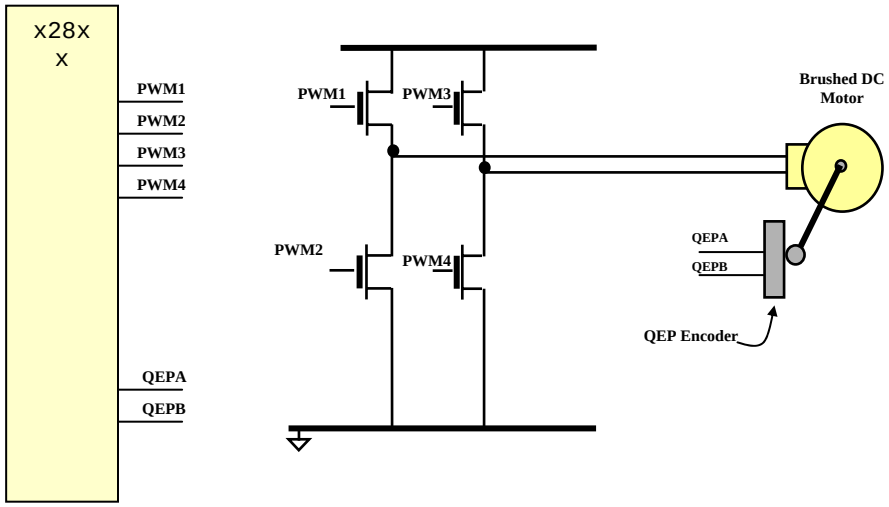


Figure 1: A Brushed DC motor drive implementation

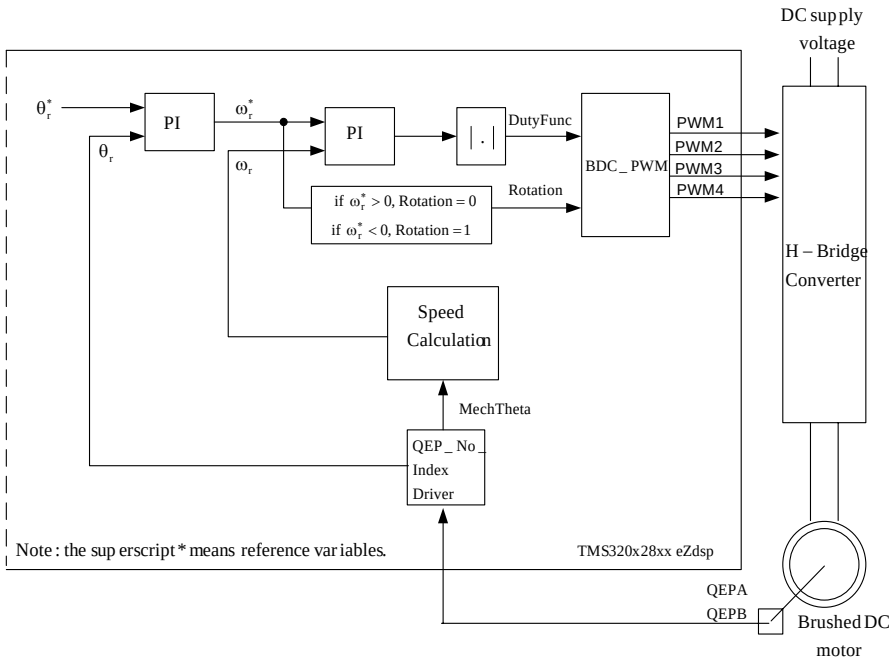


Figure 2: Overall block diagram of sensored position control of BDC motor

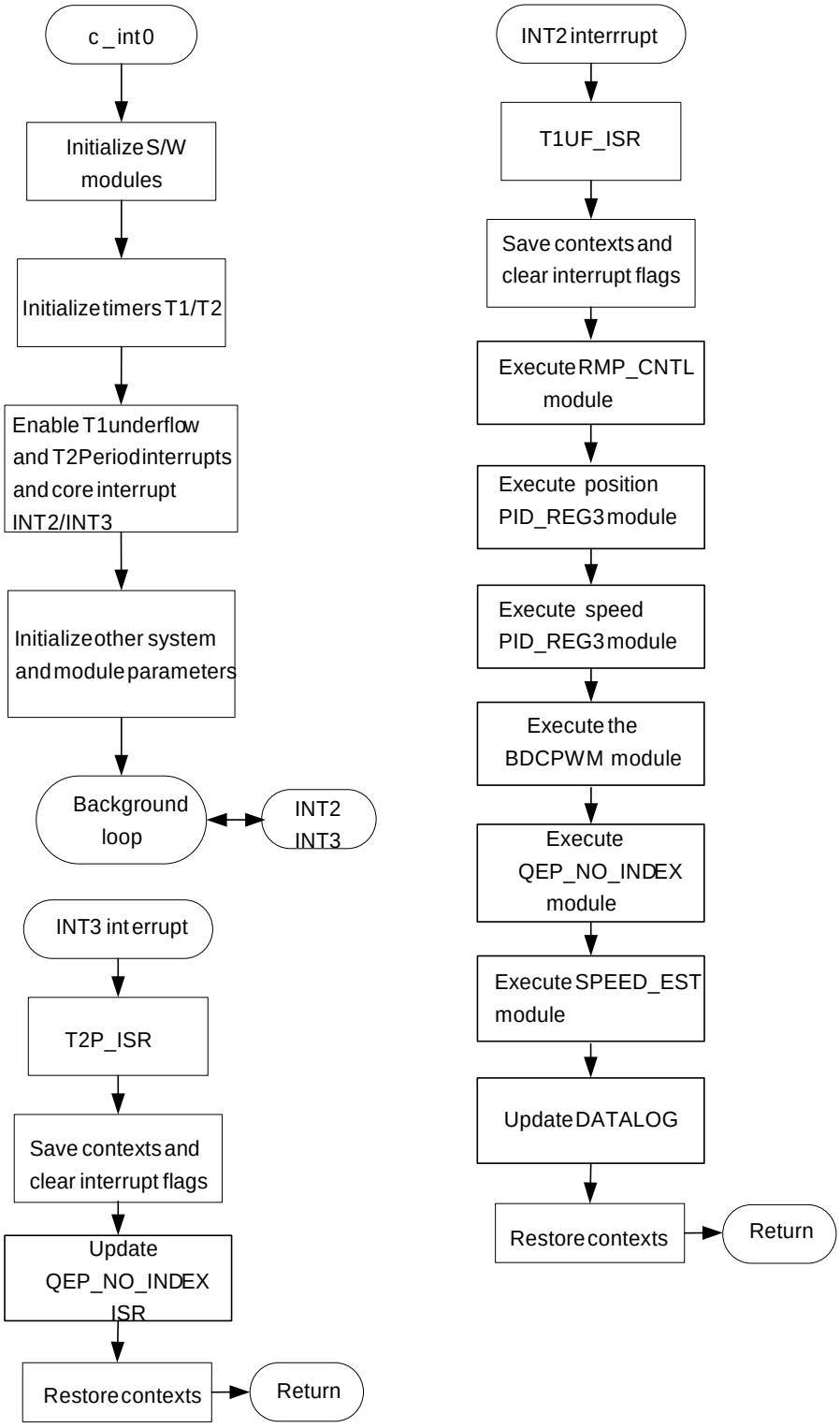


Figure 3a: Software flowchart (TMS320F281x series)

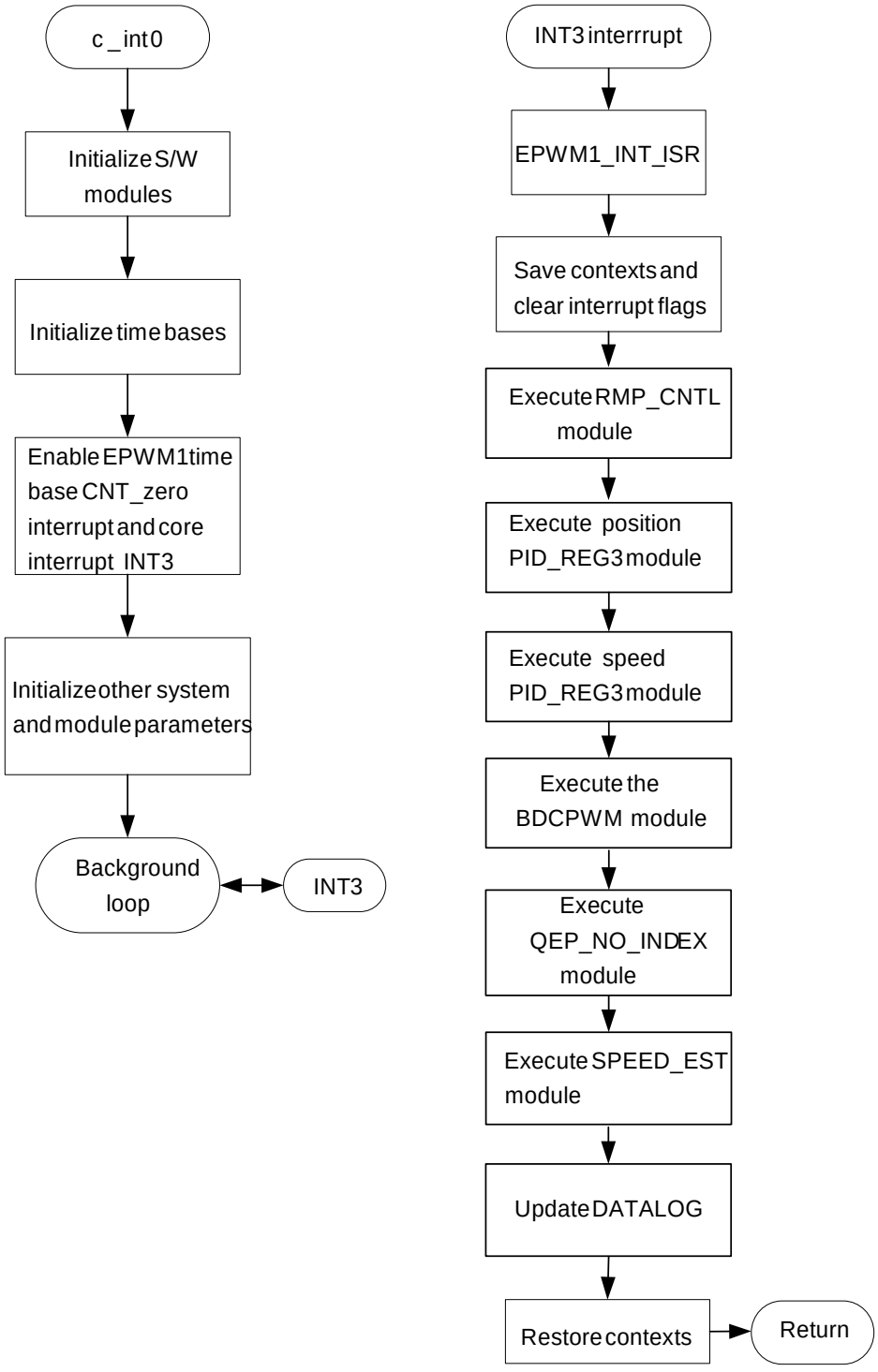


Figure 3b: Software flowchart (TMS320F280x series)

2 Hardware Configuration (DMC550 DRIVE)

The experimental system consists of the following hardware components:

1. Spectrum Digital DMC550 drive platform;
2. TMS320F2812 or TMS320F2808 eZdsp platform;
3. Brushed DC motor with QEP and a gearbox ratio;
4. IBM compatible development environment including an IBM compatible PC with Code Composer Studio (CCS) v2.2 (or above) installed;
5. Additional instruments such as oscilloscope, digital multi-meter, current sensing probe and function generator.

The experimental setup and connection can be illustrated in figure 4a for x2812 eZdsp and figure 4b for x2808 eZdsp. Notice that only major components in DMC550 and x28xx eZdsp are shown in this figure. For DMC1500 only, the JP27 should be installed to allow software to enable/disable PWM signals on DMC1500 (EN_DRIVE).

Refer to the User's Guides and or Manuals for configuration of each component and connection of the system for details.

2.1 Maximum Line Current

The software modules require that the line current variables be normalized with respect to their individual instantaneous maximum values and express these variables all as fractional numbers (i.e., Q15 format).

The choice of maximum line current depends on maximum motor current. This motor current again depends on multiple factors such as, motor drive ratings and load characteristics. In order to guarantee that the line current does not exceed the chosen maximum, a judgment factor can be applied to the selection. For example, if the maximum current is determined as 1A, then the line current can be normalized with a maximum value of 1A. The tradeoff of this large judgment factor is reduced resolution.

Once the maximum value is chosen, the offset and gain of the current sense amplifier circuit needs to be adjusted (by R15, R5, R6 on DMC550) for maximum output voltage (corresponding to 3.0V for x28xx ADC pins) at the selected maximum current.

2.2 Gain and Offset Adjustment for Line Current Sense Amplifier Circuits

Only two phase line currents are sensed through two leg resistors at the two lower power switches in the DMC550 (see schematics for details). The line currents are measured (or sampled) when all three upper power switches are turned off. The voltages across the leg resistors are shifted and amplified to an appropriate level by the associated current sense amplifier circuit (R15 for offset and R6, R5 for gains of IU, IV, respectively) before being applied to the ADC input channels of the DSP.

The knowledge of selected ADC channels and the corresponding gains/offset is required to properly configure the software modules. Refer to the User's Manual of DMC550 for details of setting the ADC circuit gains.

Note: The DC-bus voltage (optional) measurement has no gain adjustment on DMC550 board.

2.3 Jumper Settings

1. On DMC550 board, install the following jumpers:
   - JP3 (Current offset phase U)
   - JP10 (Current offset phase V)

2. Then, install position 2-3 for the following jumpers:
   - JP4 (Capture2/Hall Effect2) = Encoder 2 (B+) is mapped to Capture 2
   - JP5 (Capture1/Hall Effect1) = Encoder 1 (A+) is mapped to Capture 1
   - JP13 (VIO, Voltage Range Selection) = 3.3 Volt for 28xx DSP
   - JP14 (Voltage Control, Pot or P4) = Potentiometer R66 controls V control (optional)

   Note: when viewing the power input 5-v (P6) be upper right corner of DMC550 board, position 2-3 in each jumper is numbered as follows. (0 0 0) ===> (1 2 3)

2.4 Cautions

1. Pin #18 (at P1, Analog interface) on DMC550 must be cut. Because this same pin #18 (P9, Analog interface) on eZdsp2812/eZdsp2808 board might be shorted circuit with GND (any odd pin). This prevents the short circuited VIOANALOG on DMC550 board.

2. If the over-current trips PDPINT (pin #37, P2 on DMC550) bringing to low logic, then the DSP must be reset. The user might unplug 5-volt power supply, and wait a few seconds, then re-plug it again.

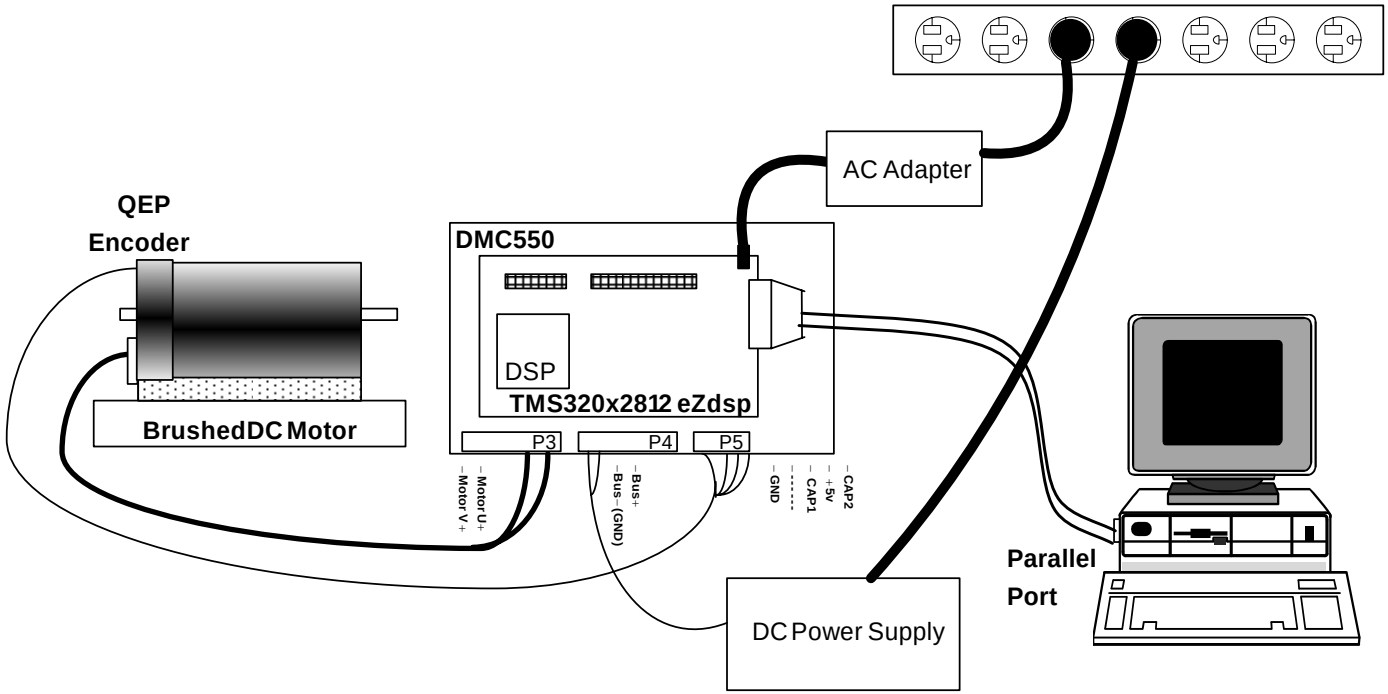


Figure 4a: Experimental setup and connection (TMS320F2812 eZdsp)

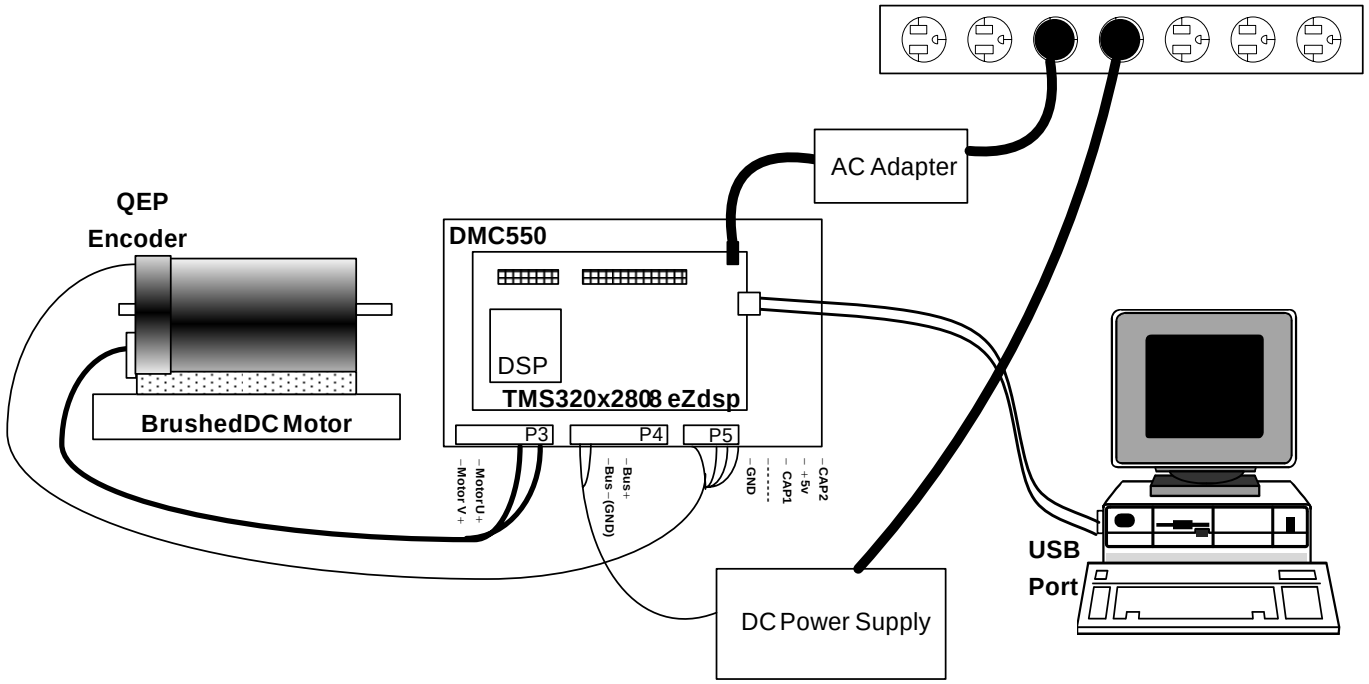
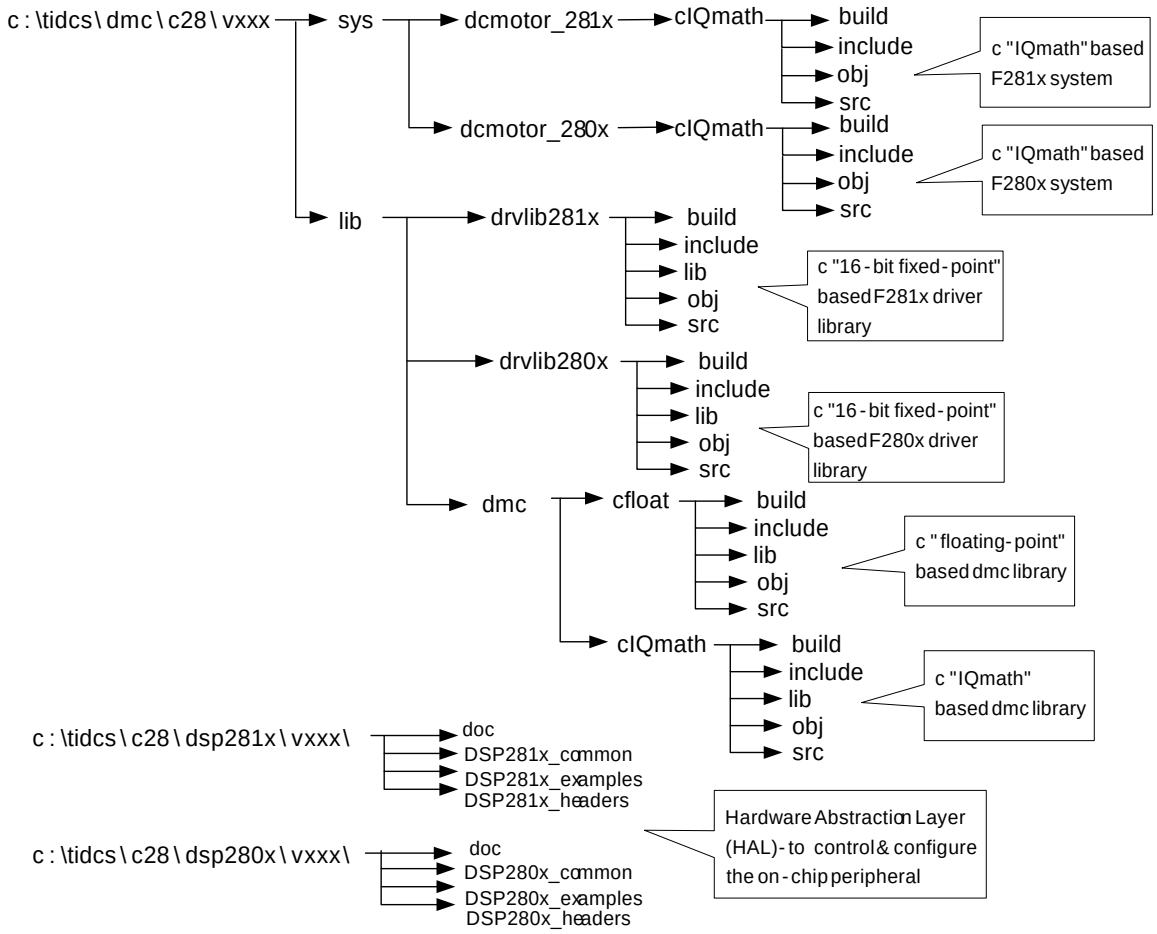


Figure 4b: Experimental setup and connection (TMS320F2808 eZdsp)

Note: Make a +5V solder connection on JP4 for eZdsp2808. Otherwise, an additional DC power supply 5 volt is required (connected at P6 port).

3 Software Configuration

3.1 C28x Real DCMOTOR Demo Directory Structure



Notice that the HAL and DMC software for F281x/F280x are located under the ***..\vxxx*** directory where xxx is the release version number.

All system-related files used in the real dcmotor system are available in “C” only, they are located under **dcmotor_281x** (for F281x target) and **dcmotor_280x** (for F280x target) directories. The workspace (*.wks)/project (*.pjt)/linker command files (*.cmd), source files (*.c) and header files (*.h) are also located in the separate directories as seen in above directory structure.

All module-related files are located under **drvlib281x** (for F281x target), **drvlib280x** (for F280x target), and **dmclib** directories. The driver modules located in **drvlib281x** and **drvlib280x** directories are implemented in 16-bit word-length. However, the dmc library located in **dmclib** directory has both floating-point and IQ formats (32-bit word-length).

3.2 Loading and Building CCS Project for C "IQmath" Real DCMOTOR demo

The workspace file (*.wks) and project file (*.pjt) for C framework to demonstrate the "IQmath" real DCMOTOR demo are located in the ***..\dcmotor_281x\cIQmath\build*** (for F281x target) or ***..\dcmotor_280x\cIQmath\build*** (for F280x target) directory. The CCS workspace file, contains the setup information for the whole project and the debugging environment such us the graph window properties, watch window parameters, break points and probe points etc. It facilitates the user to save and restore the same environment between debugging sessions instead of reconfiguring the working environment again and again for each debugging session. Notice that although the spectrum digital driver is named differently from the default one, "**sdgo2812eZdsp**" for TMS320F2812 eZdsp or "**F28xx XDS510USB Emulator (Spectrum Digital)**" for TMS320F2808 eZdsp, the CCS could bring up the workspace file successfully with a warning message.

- To quickly execute demo using the pre-configured work environment, load the correct workspace file according to the DSP target and CCS version from ***..\dcmotor_281x\cIQmath\build*** (for F281x target) or ***..\dcmotor_280x\cIQmath\build*** (for F280x target) directory as described below:

  For TMS320F2812 eZdsp, **dcmotor_281x_CCS2x.wks** and **dcmotor_281x_CCS3x.wks** are for CCS v2.x and v3.x, respectively.

  For TMS320F2808 eZdsp, **dcmotor_280x_CCS2x.wks** and **dcmotor_280x_CCS3x.wks** are for CCS v2.x and v3.x, respectively.

  Loading the workspace file will automatically open up the project file (*.pjt) for the corresponding project and show all the files relevant to the project in the FILEVIEW tab.

- From the **Project** menu choose **'Rebuild All'** or the **'Rebuild All'** shortcut on the toolbar to compile the program and load it to the target.

  Once this is done, the expanded project view as part of the CCS environment will be as shown in figures 5 and 6, if you have loaded the workspace file.

- To enable real-time mode, from the **Debug** menu choose **'Reset CPU'**, then select **'Real Time Mode'**. Then, click **'Yes'** when a message box asks "Do you want to allow realtime mode switching?: Can't enter real time mode unless debug events are enabled. Bit 1 of ST1 must be 0".

- After selecting Real Time Mode, run the software by choosing **Run** from the **Debug** menu or using the tool bar shortcut.

- The default ISR frequency is 40 kHz which can be easily changed in the header file **parameter.h** under ***..\dcmotor_281x\cIQmath\include*** (for F281x target) or ***..\dcmotor_280x\cIQmath\include*** (for F280x target) directory.

- The BDC motor parameters, base quantities, mechanical parameters, and sampling period time (i.e., ISR period) can be conveniently changed in the header file, **parameter.h**.

- The overall Q (called GLOBAL_Q, default GLOBAL_Q is set at 24) is adjustable in the header file, **IQmathLib.h** under ***..\lib\dmclib\cIQmath\include*** directory.

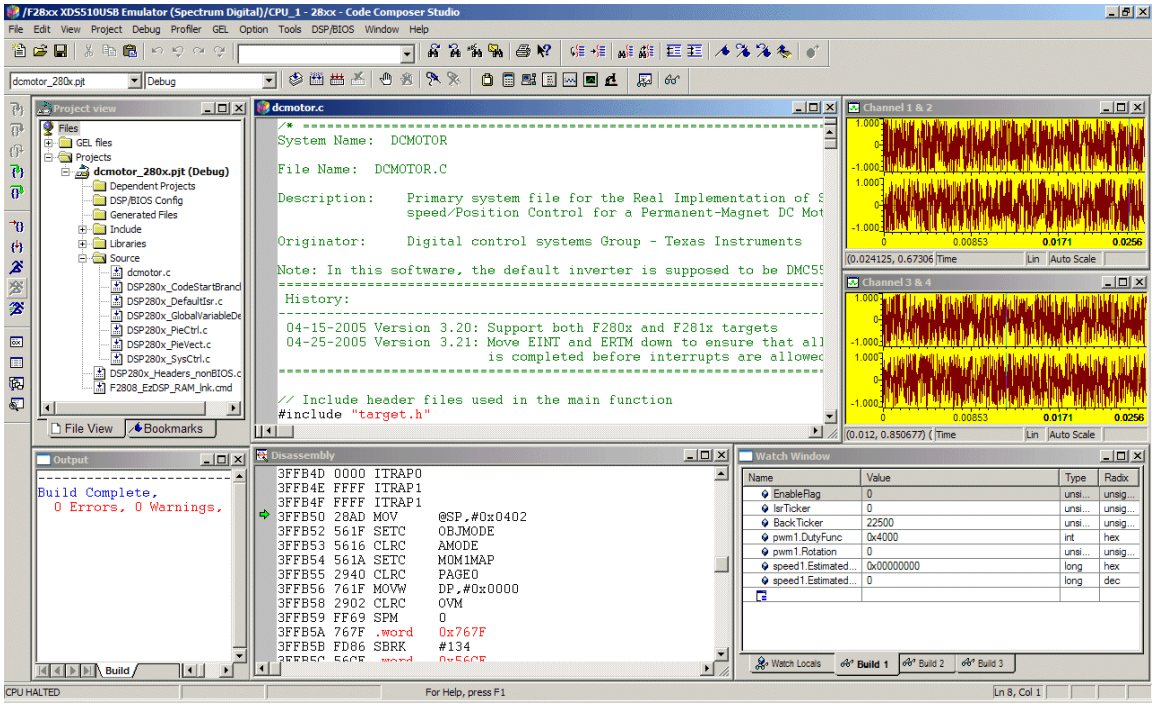


Figure 5: CCS project view of real DCMOTOR demo using C framework

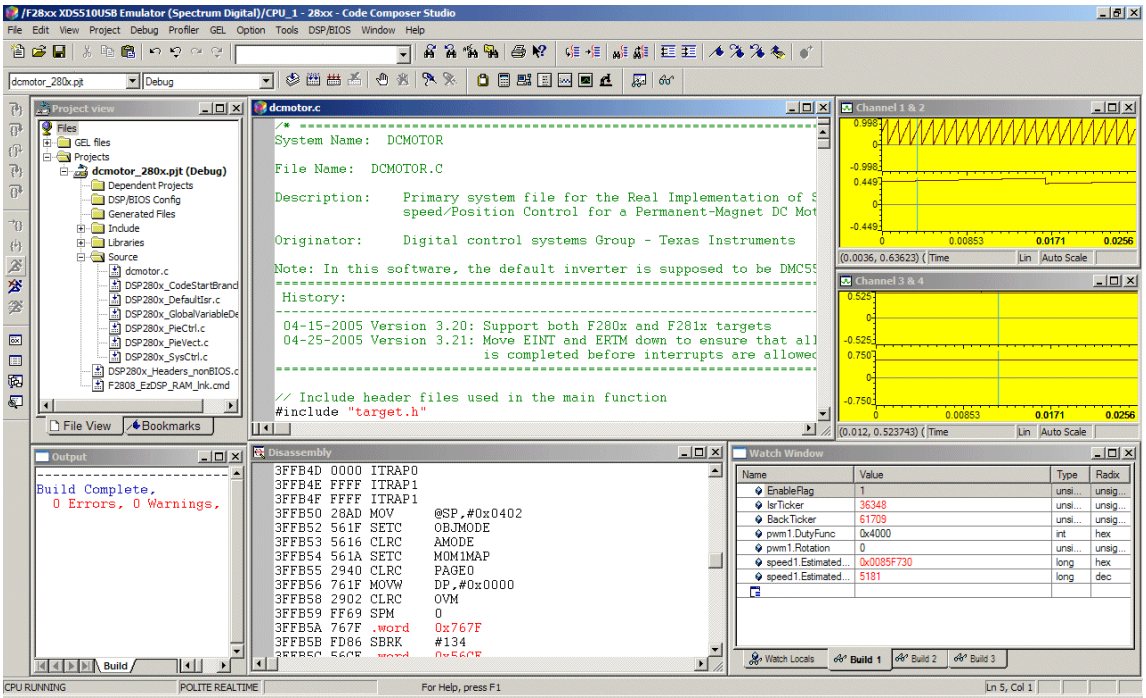


Figure 6: Run time view of real DCMOTOR demo using C framework

4 Incremental System Build

The system is gradually built up in order for the final system can be confidently operated. Three phases of the incremental system build are designed to verify the major software modules used in the system. Table 1 summarizes the modules testing and using in each incremental system build.

Software module	Phase 1	Phase 2	Phase 3
DATALOG	√	√	√
RMP_CNTL	√	√	√
BDCPWM_DRV	√√	√	√
QEP_NO_INDEX_DRV	√√	√	√
ESTIMATEDSPEED	√√	√	√
PID_REG3		√√	√√

Note: the symbol √ means this module is using and the symbol √√ means this module is testing in this phase.

Table 1: Testing modules in each incremental system build

Table 2 conveniently shows the specified input/output variable names for each module. The formats of the variables are also indicated, accordingly.

Software module	Input		Output	
	Name	Format	Name	Format
DATALOG	*iptr1 *iptr2 *iptr3 *iptr4	Pointer to Q15 variables	N/A	Memory
RMP_CNTL	TargetValue	IQ	SetpointValue	IQ
BDCPWM_DRV	Rotation DutyFunc	Q0 Q15	CMPR1 CMPR2 T1PER	EV registers
QEP_NO_INDEX_DRV	CAP1,2	EV H/W pin	MechTheta OutputTheta DirectionQep	Q15 Q15 Q0
ESTIMATEDSPEED	EstimatedTheta	IQ	EstimatedSpeed	IQ
PID_REG3	Ref Fdb	IQ	Out	IQ

Table 2. Input/output variable names and corresponding formats for each software module

4.1 Phase 1 Incremental System Build

Assuming sections 2-3 are completed successfully, this section describes the steps for a "minimum" system check-out which confirms operation of system interrupts, some peripheral & target dependent modules. The position and speed are verified by the open-loop operation of motor.

In the **build.h** header file located under ***..\dcmotor_281x\cIQmath\include*** (for F281x target) or ***..\dcmotor_280x\cIQmath\include*** (for F280x target) directory, select phase 1 incremental build option by setting the build level to level 1. Use the 'Rebuild All' feature of CCS to save the program, compile it and load it to the target.

After running and setting real time mode, set "EnableFlag" to 1 in watch windows in order to enable interrupt T1UF (for x281x) and EPWM1 (for x280x). The variable named "IsrTicker" will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

In the software, the key variables to be adjusted are summarized below.

- pwm1.DutyFunc (Q15 format): for changing the PWM duty cycle in per-unit.
- pwm1.Rotation (Q0 format): for changing the rotational direction of DC motor.

- Compile/load/run program with real time mode.

- Set DutyFunc to a small value in Q15 (e.g., 0x0F00). The Rotation could be set either 0 or 1.

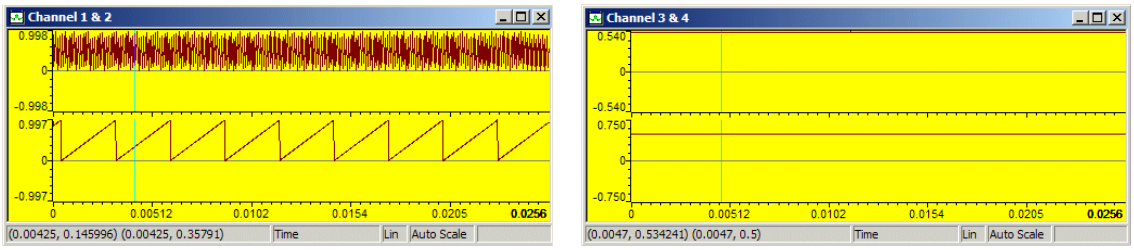
- Gradually increase voltage at DC power supply to get an appropriate DC-bus voltage and now the motor is running with a certain speed.

- Check the PWM outputs with an oscilloscope. For the Rotation = 0, the PWM1/4 are active, whereas PWM2/3 are inactive (forced OFF). For the Rotation = 1, the PWM2/3 are active, whereas PWM1/4 are inactive (forced OFF).

- Check the position (MechTheta) from QEP_NO_INDEX module by comparing the calculated speed (EstimatedSpeed) from ESTIMATEDSPEED module with the speed calculated by 1/Tp, where Tp is the period of the motor position.

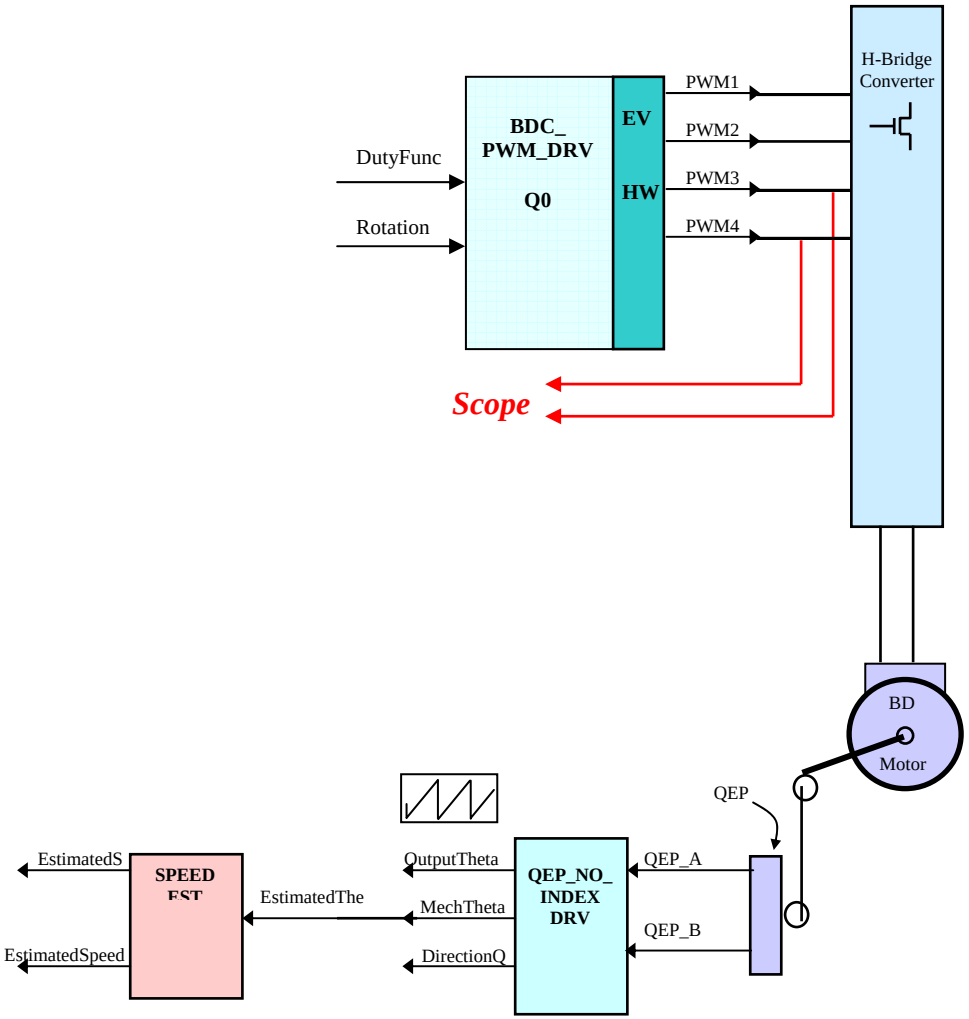
- Reduce voltage at DC power supply to zero, halt program and stop real time mode. Now the motor is stopping.

During running this build, the waveforms in the CCS graphs should be appeared as follow:



Channel 1: motor mechanical angle, Channel 2: output angle, Channel 3: motor speed, Channel 4: PWM duty function

Phase 1 Incremental System Build Block Diagram



4.2 Phase 2 Incremental System Build

Assuming section 4.1 is completed successfully, this section verifies the closed loop speed PI controller.

In the **build.h** header file located under ***..\dcmotor_281x\cIQmath\include*** (for F281x target) or ***..\dcmotor_280x\cIQmath\include*** (for F280x target) directory, select phase 2 incremental build option by setting the build level to level 2. Use the 'Rebuild All' feature of CCS to save the program, compile it and load it to the target.

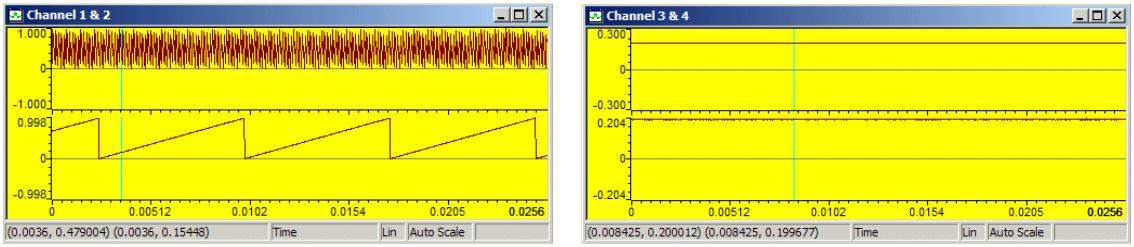
After running and setting real time mode, set "EnableFlag" to 1 in watch windows in order to enable interrupt T1UF (for x281x) and EPWM1 (for x280x). The variable named "IsrTicker" will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

In the software, the key variables to be adjusted are summarized below.

- SpeedRef (floating format): for changing the reference rotor speed in per-unit.

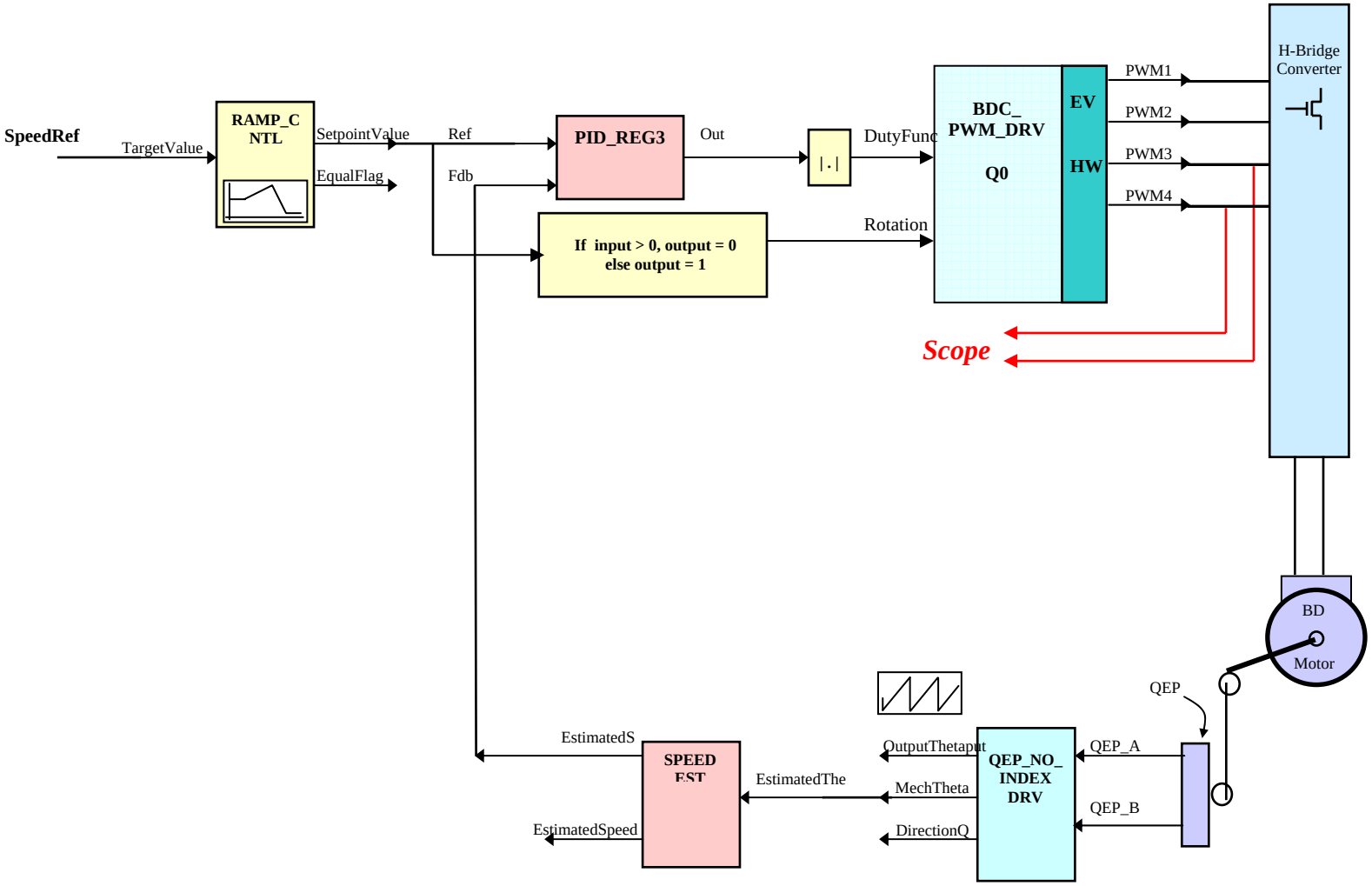
- Compile/load/run program with real time mode.
- Set SpeedRef to 0.5 pu (or another suitable value if the base speed is different).
- Gradually increase voltage at DC power supply to get an appropriate DC-bus voltage and now the motor is running with this reference speed (0.5 pu).
- Compare pid1_spd.Fdb with pid1_spd.Ref in the watch windows with continuous refresh feature whether or not it should be nearly the same.
- To confirm this speed PID module, try different values of SpeedRef.
- For speed PID controller, the proportional, integral, derivative and integral correction gains may be re-tuned to have the satisfied responses.
- Reduce voltage at DC power supply to zero, halt program and stop real time mode. Now the motor is stopping.

During running this build, the waveforms in the CCS graphs should be appeared as follow:



Channel 1: motor mechanical angle, Channel 2: output angle, Channel 3: speed reference, Channel 4: speed feedback

Phase 2 Incremental System Build Block Diagram



4.3 Phase 3 Incremental System Build

Assuming section 4.2 is completed successfully, this section verifies the closed loop position PI controller.

In the **build.h** header file located under ***..\dcmotor_281x\cIQmath\include*** (for F281x target) or ***..\dcmotor_280x\cIQmath\include*** (for F280x target) directory, select phase 3 incremental build option by setting the build level to level 3. Use the 'Rebuild All' feature of CCS to save the program, compile it and load it to the target.

After running and setting real time mode, set "EnableFlag" to 1 in watch windows in order to enable interrupt T1UF (for x281x) and EPWM1 (for x280x). The variable named "IsrTicker" will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

In the software, the key variables to be adjusted are summarized below.

- PositionRef (floating format): for changing the reference output position in per-unit.

- Compile/load/run program with real time mode.

- Set PositionRef to 0.5 pu.

- Gradually increase voltage at DC power supply to get an appropriate DC-bus voltage and now the motor is turning to this reference position (0.5 pu) from the initial position of zero degree.

  Note that the controlled position is the output position, not the shaft motor position.

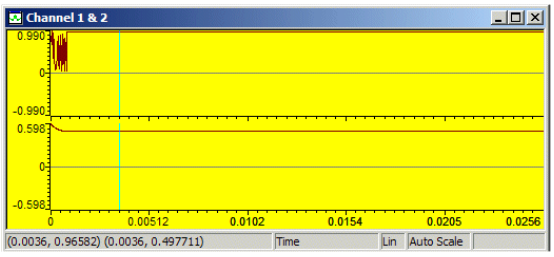
- Compare pid1_pos.Fdb with pid1_pos.Ref in the watch windows with continuous refresh feature whether or not it should be nearly the same.

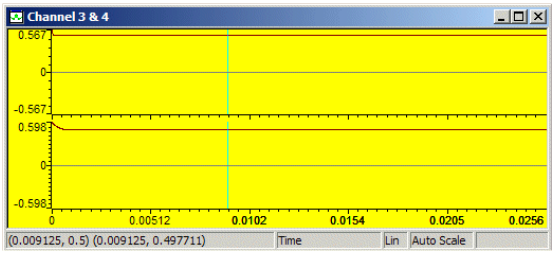
- To confirm this position PID module, try different values of PositionRef.

- For position PID controller, the proportional, integral, derivative and integral correction gains may be re-tuned to have the satisfied responses.

- Reduce voltage at DC power supply to zero, halt program and stop real time mode. Now the motor is stopping.

  During running this build, the waveforms in the CCS graphs should be appeared as follow:





Channel 1: motor mechanical angle, Channel 2: output angle, Channel 3: output position reference, Channel 4: output position feedback

Phase 3 Incremental System Build Block Diagram