PMSM3_1

System Document

C2000 Foundation Software

Table of Contents

1 System Overview

This document describes the “C” real control framework to demonstrate the PMSM3_1 demo implemented using Code Composer Studio (CCS) version 2.2. The “C” framework is designed to run on TMS320C281x and TMS320C280x based controllers on CCS V2.2.

The framework uses the following modules viz.,

1. EN_DRIVE
2. ILEG2_DCBUS
3. PWMDAC/PWMGEN
4. QEP
5. CLARKE
6. PID_REG3
7. PARK/IPARK
8. SPEED_FR
9. SVGEN_DQ

In this system, the sensored Field Oriented Control (FOC) of Permanent Magnet Synchronous Motor (PMSM) using QEP sensor will be experimented and explored the performance of speed control. The user can quickly start evaluating the performance of sensored FOC system by studying the speed responses.

The PWMDAC output is available in the case of TMS320X281X DSP series, which has two Event Managers. This is due to the fact that the PWMDAC_DRV uses Timer T3 available in Event Manager B (EVB) to generate 30 kHz PWM outputs. In case of TMS320X280X DSP series, the EPWM4-6 modules are used to generate the 30 kHz PWMDAC outputs.

The PMSM3_1 demo has the following properties

C Frame work		
System Name	**Program memory usage 281x/280x**	**Data memory usage[1] 281x/280x**
PMSM3_1 (IQ)	3987 words[2]/4393 words[2]	844 words/844 words

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

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)	20 kHz (PWMGEN, Timer 1-EVA), 30 kHz (PWMDAC, Timer 3-EVB)
(280x)	20 kHz (PWMGEN, EPWM1-3), 30 kHz (PWMDAC, EPWM4-6)
PWM Mode	Symmetrical with a programmable dead band (PWMGEN)
Interrupts (281x)	1 (Timer T1 underflow – Implements 20 kHz ISR execution rate)
(280x)	1 (EPWM1 Time Base CNT_zero – Implements 20 kHz ISR execution rate)
Peripheral Used (281x)	Timer T1/T3, PWM7/9/11
(280x)	EPWM1-6

The overall system for implementation of the 3-ph PMSM control can be depicted in figure 1. The PMSM is driven by the conventional voltage-source inverter. The TMS320x2812 or TMS320x2808 eZdsp is generating six pulse width modulation (PWM) signals by means of space vector PWM technique for six power switching devices in the inverter. Two input currents of the PMSM (i_a and i_b) are measured from the inverter and they are sent to the TMS320x2812 or TMS320x2808 eZdsp via two analog-to-digital converters (ADCs).

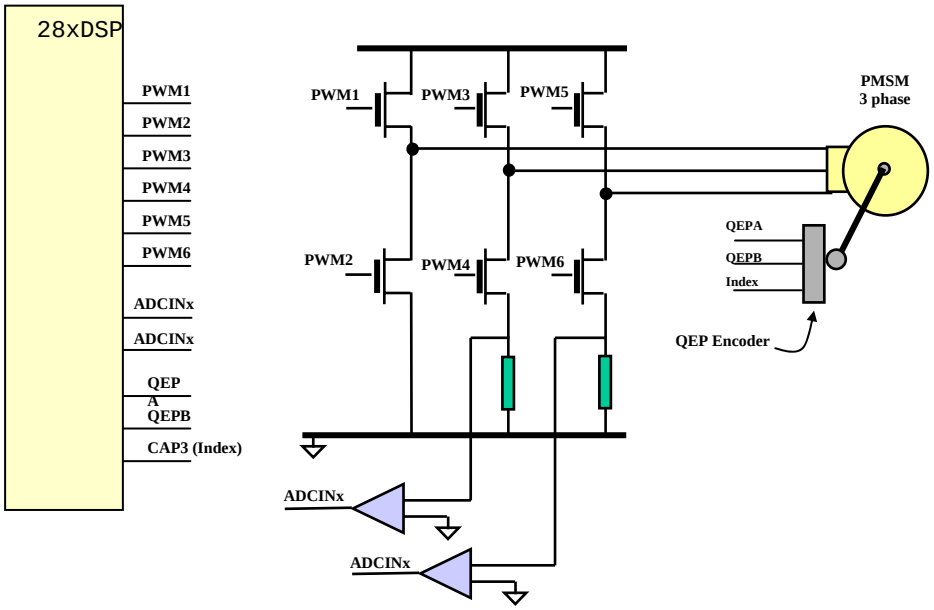


Figure 1: A 3-ph PMSM drive implementation

Theoretically, the field oriented control for the PMSM drive allows the motor torque be controlled independently with the flux like DC motor operation. On other words, the torque and flux are decoupled from each other. The rotor position is required for variable transformation from stationary reference frame to synchronously rotating reference frame. As a result of this transformation (so called Park transformation), q-axis current will be controlling torque while d-axis current is forced to zero. Therefore, the key module of this system is the information of rotor position from QEP encoder. The overall block diagram can be depicted in figure 2.

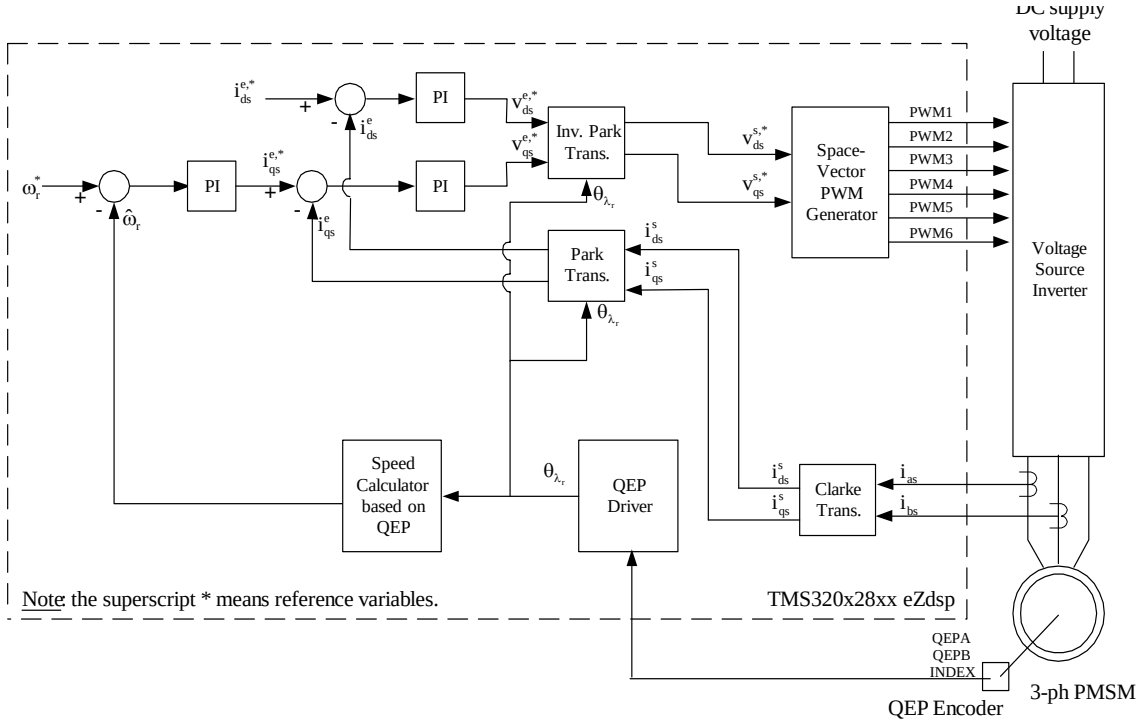


Figure 2: Overall block diagram of field oriented control of PMSM

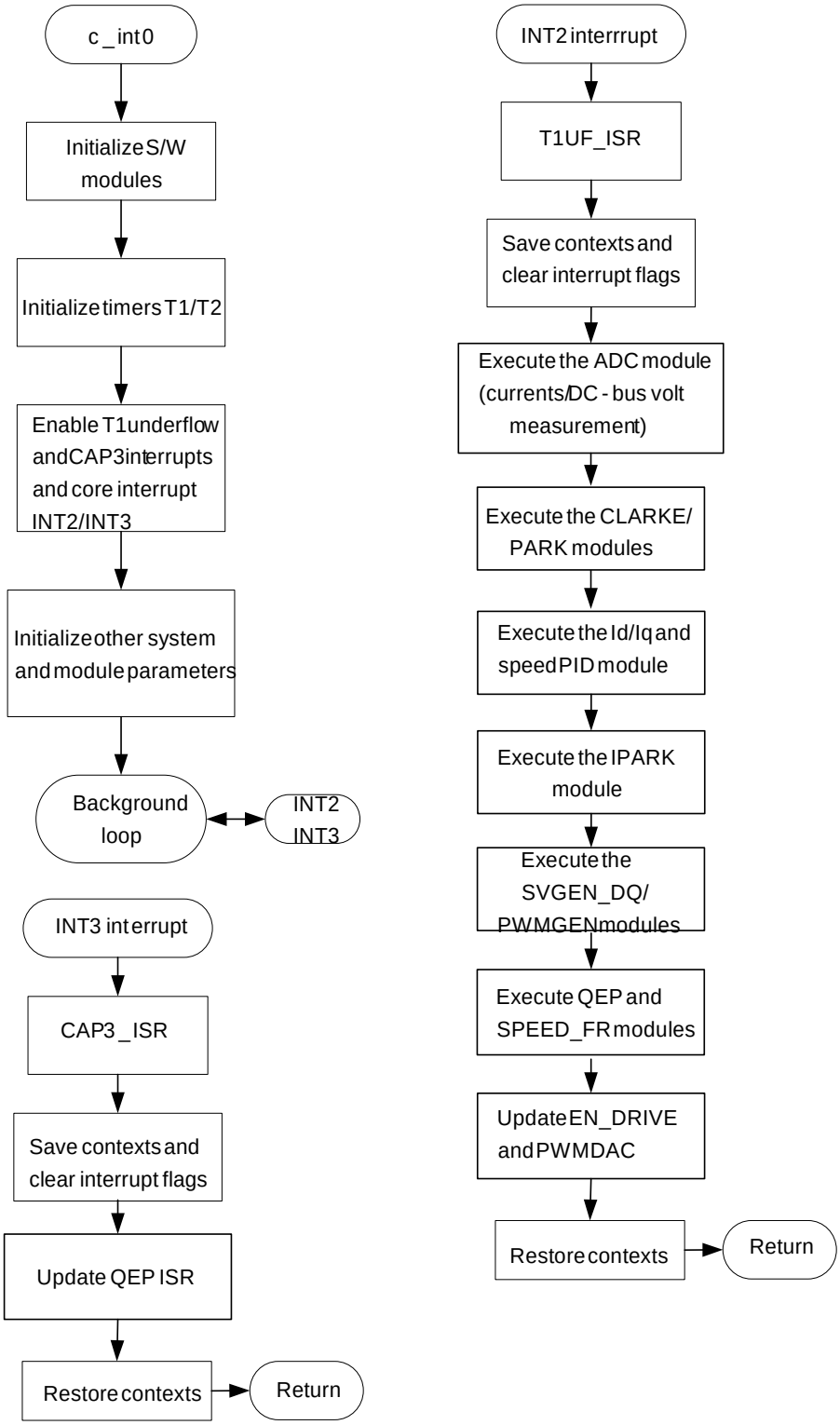


Figure 3a: Software flowchart (TMS320F281x series)

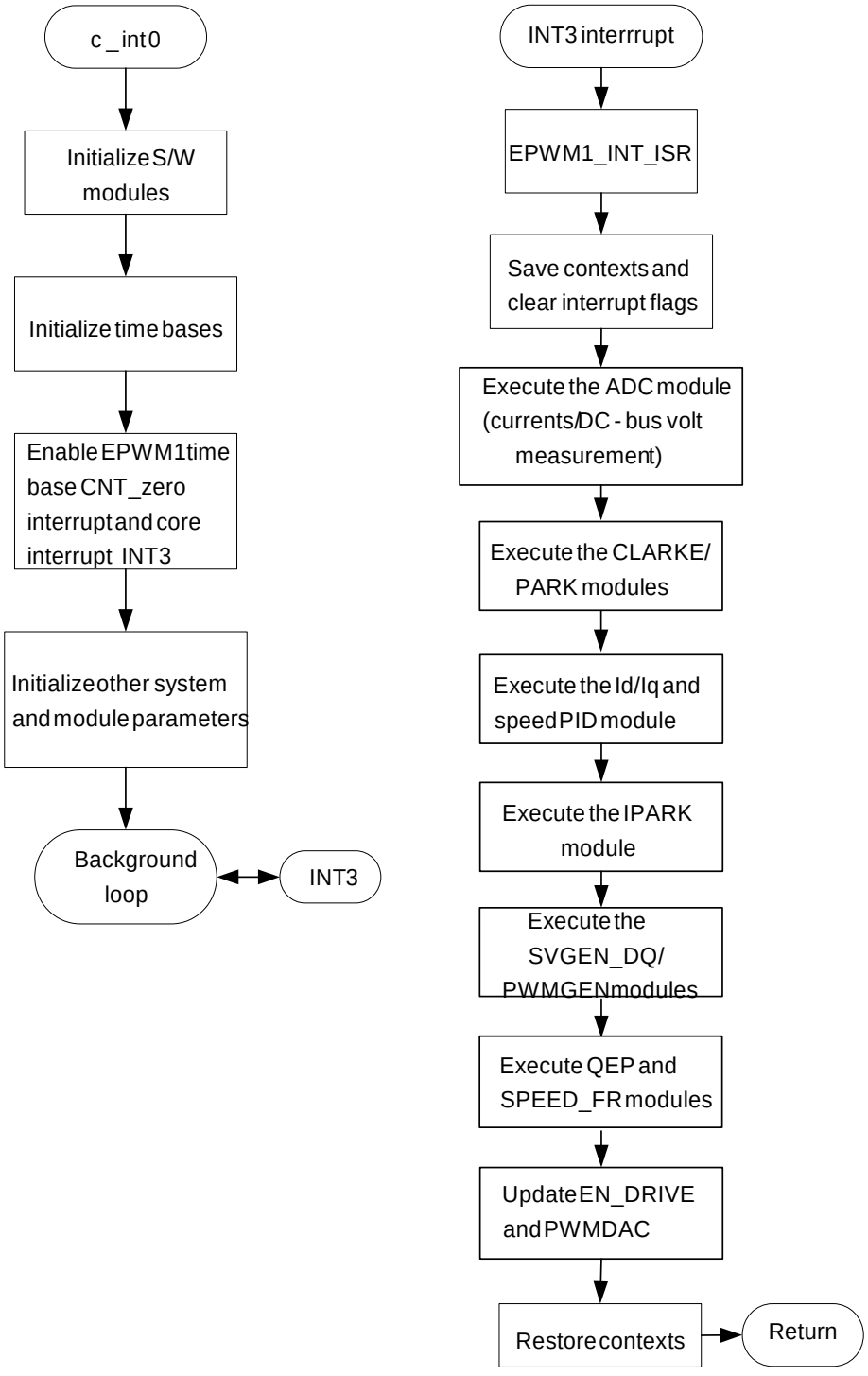


Figure 3b: Software flowchart (TMS320F280x series)

2 Hardware Configuration (DMC1500 DRIVE)

The experimental system consists of the following hardware components:

1. Spectrum Digital DMC1500 drive platform;
2. TMS320F2812 or TMS320F2808 eZdsp platform;
3. 3-ph PMSM with a QEP encoder;
4. IBM compatible development environment including an IBM compatible PC with Code Composer Studio (CCS) v2.2 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 DMC1500 and x28xx eZdsp are shown in these figures. 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.

At this point, the AC supply to DMC1500 should remain at zero output voltage pending step-by-step verification described later in this documentation. In order to check out the operation of the logic state, turn on the 18V DC source for DMC1500 and 5V DC source for TMS320F2812 or TMS320F2808 eZdsp. Since the 3-ph PMSM is rated at 220V AC, so the 320 V DC-bus voltage is needed and the input voltage doubler option (×2 setting) on DMC1500 should be used if the 1-ph supply voltage is fully at 110 V AC.

2.1 Maximum Line Current and DC-Bus Voltage

The software modules require that the line current and DC-bus voltage 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 10A, then the line current can be normalized with a maximum value of 12A. 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 R12, R16, R17, R19 in figure 4a/4b) for maximum output voltage (corresponding to 3.0V for x28xx ADC pins) at the selected maximum current.

Maximum DC-bus voltage will typically depend on the rating voltage of motor. For example, if the motor is rated at 220V(line-line), then the DC-bus voltages can be appropriately normalized with a maximum value of 310V (with ×2 setting seen in figure 4a/4b). Then, this maximum value will become the base line-line (peak) voltage for the system. The gain of the voltage sense amplifier circuits needs to be adjusted (by R10 in figure 4a/4b) for maximum output voltage (corresponding to 3.0V for x28xx ADC pins) at the selected maximum DC-bus voltage.

2.2 Gain and Offset Adjustment for Line Current and DC-Bus Voltage Sense Amplifier Circuits

Three phase line currents are sensed through three leg resistors at the three lower power switches in the DMC1500 (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 (R17 for offset and R12, R16, R19 for gains of IU, IV, IW, respectively) before being applied to the ADC input channels of the DSP. Similarly, DC-bus voltages are sensed through voltage divider circuits on DMC1500 and then amplified (R10 for gain) to an appropriate level before being applied to the ADC input channel of the DSP. Also, the jumper JP24 (see figure 4a/4b) is required to be connected according to the selected channels for these line currents and DC-bus voltage.

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 DMC1500 for details of setting the ADC circuit gains.

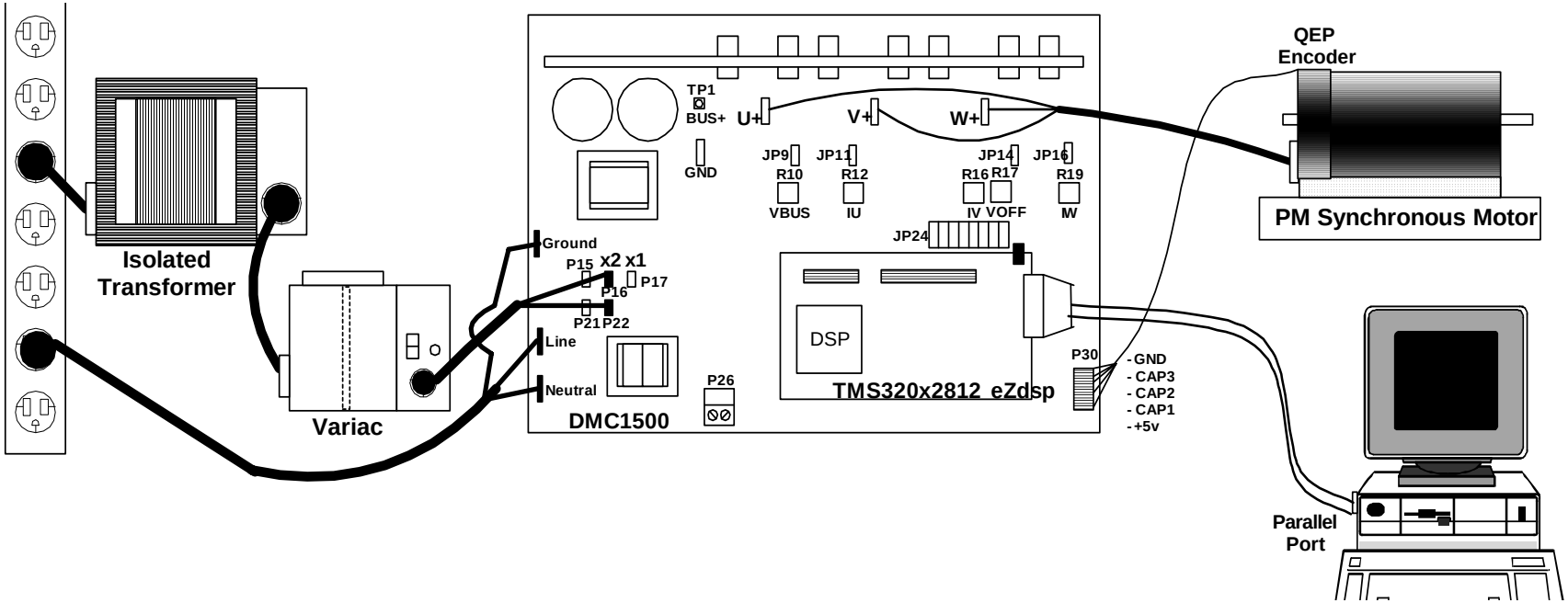


Figure 4a: Experimental setup and connection (TMS320F2812 eZdsp)

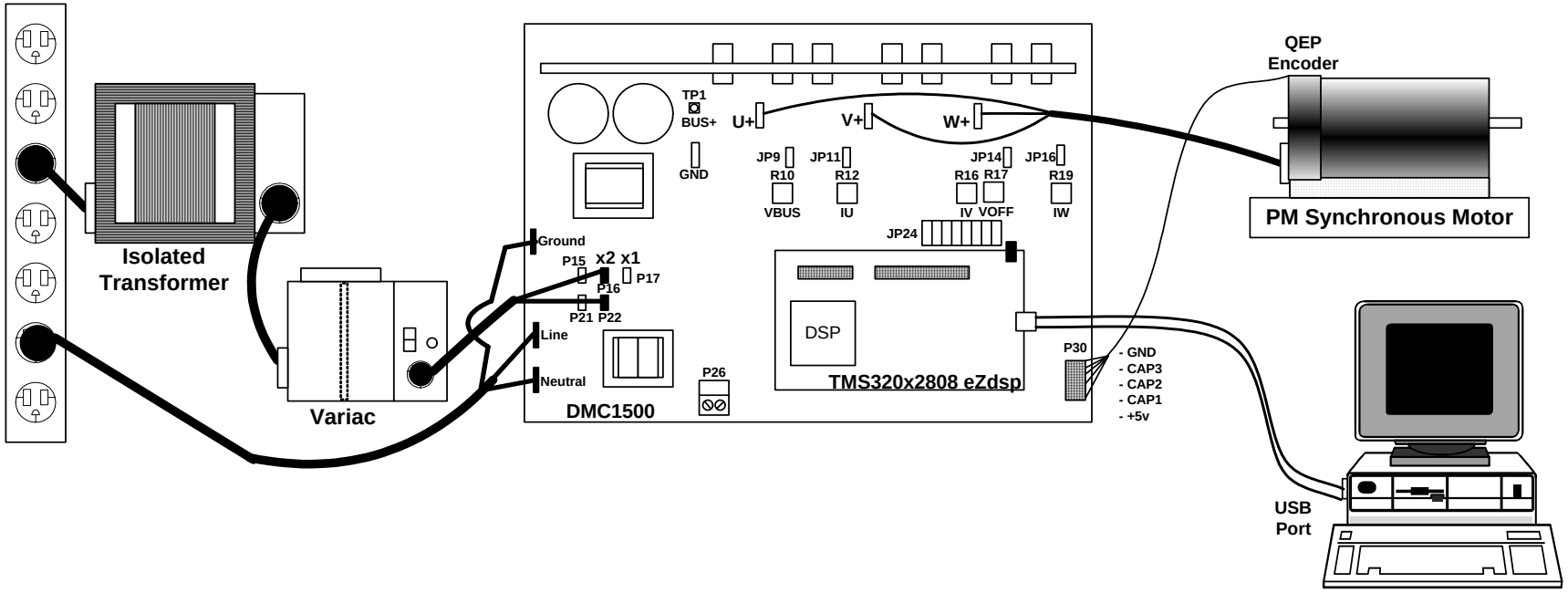


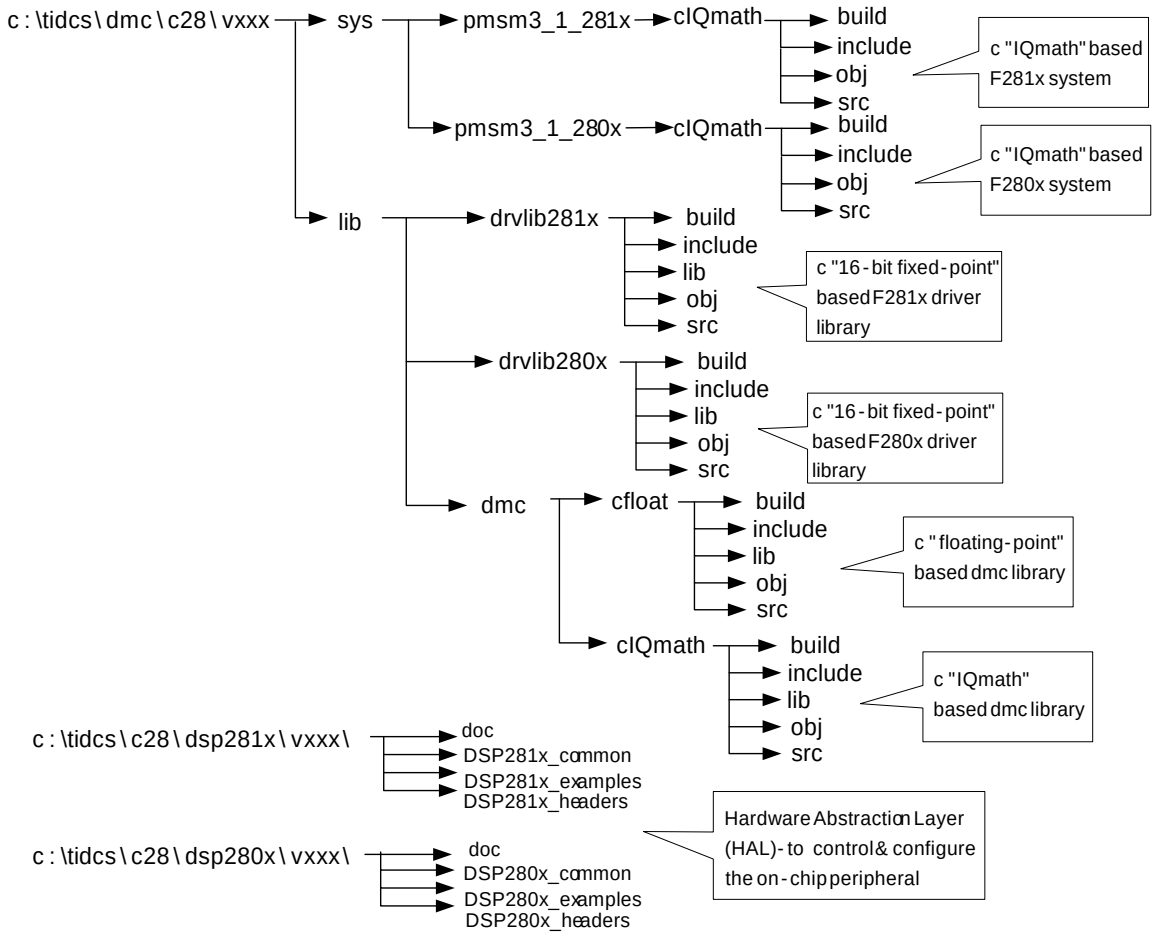
Figure 4b: Experimental setup and connection (TMS320F2808 eZdsp)

Note: 1. Make a +5V solder connection on JP4 for eZdsp2808. Otherwise, an additional DC power supply 5 volt is required (connected at P6 port).

2. For proper operation, the pin #18 between P19 on DMC1500 and P8 on eZdsp2808 should not be interfaced.

3 Software Configuration

3.1 C28x Real PMSM3_1 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 pmsm3_1 system are available in “C” only, they are located under **pmsm3_1_281x** (for F281x target) and **pmsm3_1_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 PMSM3_1 demo

The workspace file (*.wks) and project file (*.pjt) for C framework to demonstrate the "IQmath" real PMSM3_1 demo are located in the ***..\pmsm3_1_281x\cIQmath\build*** (for F281x target) or ***..\pmsm3_1_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 from ***..\pmsm3_1_281x\cIQmath\build*** (for F281x target) or ***..\pmsm3_1_280x\cIQmath\build*** (for F280x target) directory. The **pmsm3_1_281x.wks** is for TMS320F2812 eZdsp and **pmsm3_1_280x.wks** is for TMS320F2808 eZdsp.

  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 **pmsm3_1_281x.wks** or **pmsm3_1_280x.wks**.

- 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 20 kHz which can be easily changed in the header file **parameter.h** under ***..\pmsm3_1_281x\cIQmath\include*** (for F281x target) or ***..\pmsm3_1_280x\cIQmath\include*** (for F280x target) directory.

- The PMSM 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.

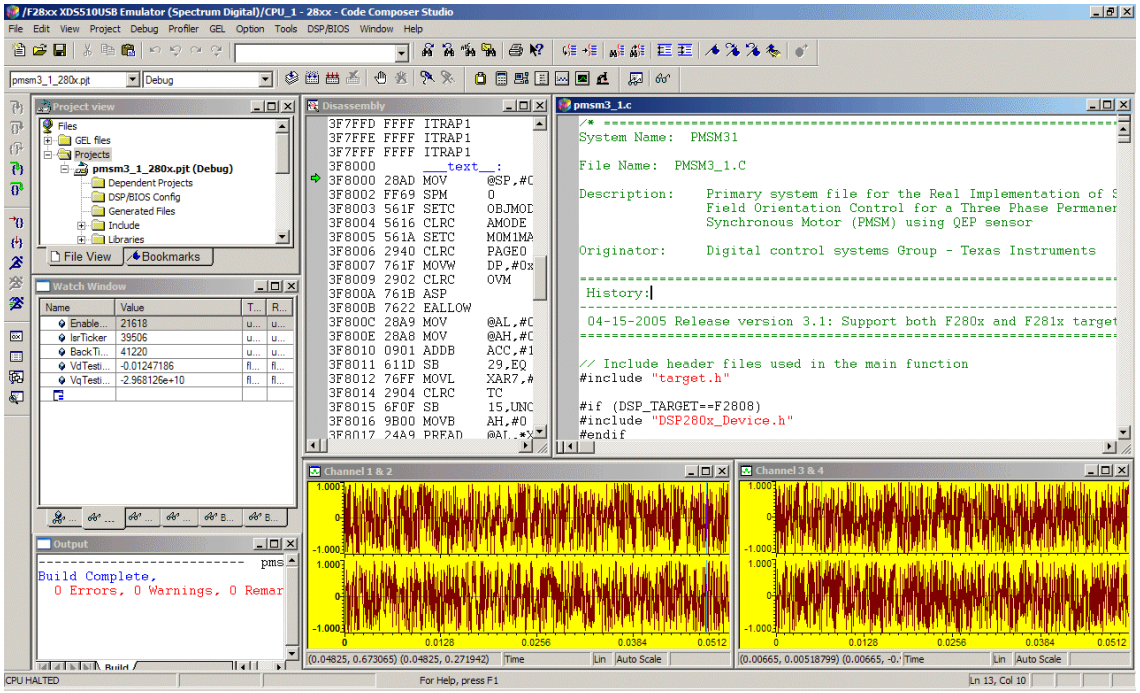


Figure 5: CCS project view of real PMSM3_1 demo using C framework

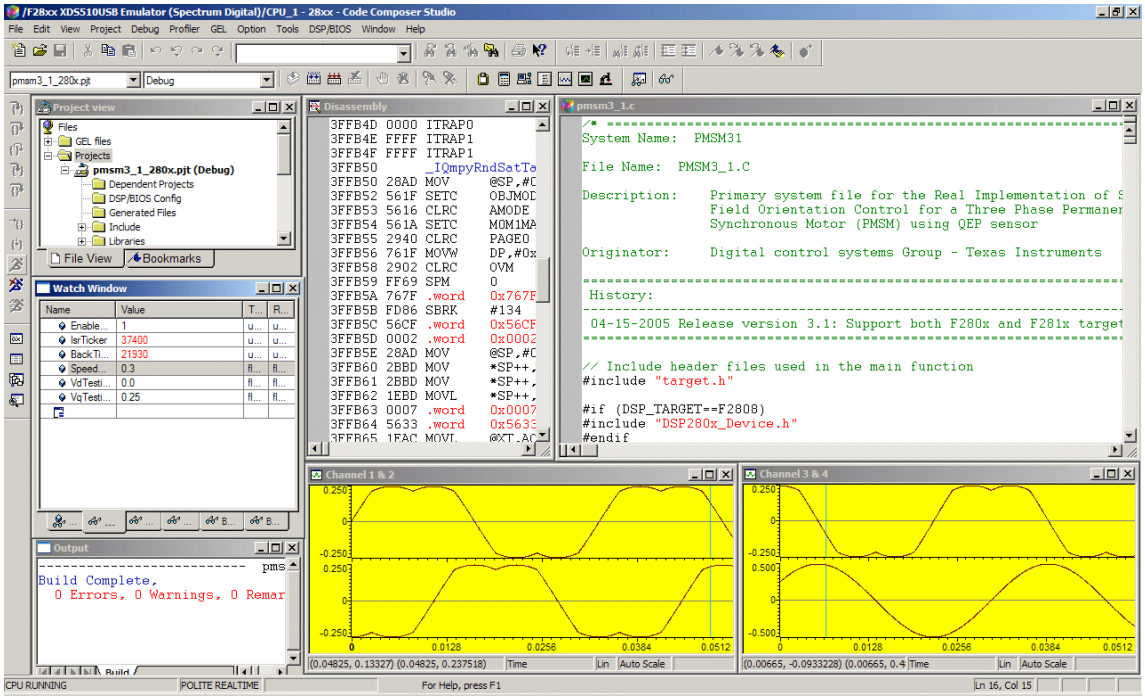


Figure 6: Run time view of real PMSM3_1 demo using C framework

4 Incremental System Build

The system is gradually built up in order for the final system can be confidently operated. Five 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	Phase 4	Phase 5
EN_DRIVE	√	√	√	√	√
PWMDAC	√	√	√	√	√
RAMP_GEN	√	√	√	√	
RAMP_CNTL	√	√	√	√	
I_PARK	√√ (1ab)	√	√	√	√
SVGEN_DQ	√√ (1a)	√	√	√	√
PWM_DRV	√√ (1b)	√	√	√	√
ILEG2_DCBUS_DRV		√√ (2a)	√	√	√
CLARKE		√√ (2b)	√	√	√
PARK		√√ (2b)	√	√	√
PID_REG3 (ID)			√√	√	√
PID_REG3 (IQ)			√√	√	√
QEP_DRV				√√	√
SPEED_FR				√√	√
PID_REG3 (SPEED)					√√
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
EN_DRIVE	EnableFlag	Q0	GPIOA6 GPIOA11	GPIO registers
PWMDAC_DRV	PWMDACINPOINTER0 PWMDACINPOINTER1 PWMDACINPOINTER2	Pointers to Q15 variables	CMPR4 CMPR5 CMPR6 T3PER	EVB registers
RAMP_GEN	Freq Offset Gain	IQ	Out	IQ
RAMP_CNTL	TargetValue	IQ	SetpointValue	IQ
I_PARK	Ds Qs Angle	IQ	Alpha Beta	IQ
SVGEN_DQ	Ualpha Ubeta	IQ	Ta Tb Tc	IQ
PWM_DRV	MfuncC1 MfuncC2 MfuncC3 MfuncPeriod	Q15	CMPR1 CMPR2 CMPR3 T1PER	EV registers
ILEG2_DCBUS_DRV	ADCINx/y/z	ADC H/W pins	ImeasA ImeasB ImeasC VdcMeas	Q15
CLARKE	As Bs	IQ	Alpha Beta	IQ
PARK	Alpha Beta Angle	IQ	Ds Qs	IQ
QEP_DRV	QEPA,B,I	EV H/W pin	ElecTheta DirectionQep	Q15 Q0
SPEED_FR	ElecTheta DirectionQep	IQ Q0	Speed SpeedRpm	IQ Q0
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 is completed successfully, this section describes the steps for a "minimum" system check-out which confirms operation of system interrupts, the peripheral & target independent I_PARK and SVGEN_DQ modules and the peripheral dependent PWM_DRV module. Notice that only the x2812 or x2808 eZdsp is used in this phase. The PMSM and DMC1500 board are not necessary to be connected yet.

In the **build.h** header file located under ***..\pmsm3_1_281x\cIQmath\include*** (for F281x target) or ***..\pmsm3_1_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.

- SpeedRef (floating format): for changing the rotor speed in per-unit.
- VdTesting (floating format): for changing the d-qxis voltage in per-unit.
- VqTesting (floating format): for changing the q-axis voltage in per-unit.

4.1a Phase 1a (SVGEN_DQ test)

The SpeedRef value is specified to the RAMP_GEN module via RAMP_CNTL module. The I_PARK module is generating the outputs to the SVGEN_DQ module. Three outputs from SVGEN_DQ module are monitored via the PWMDAC module with external low-pass filter and an oscilloscope. The expecting output waveform can be seen in figure 7 where Ta, Tb, and Tc waveform are 120^o apart from each other. Specifically, Tb lags Ta by 120^o and Tc leads Ta by 120^o.

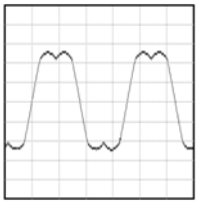


Figure 7: Ta, Tb, and Tc waveforms

4.1b Phase 1b (PWM_DRV test)

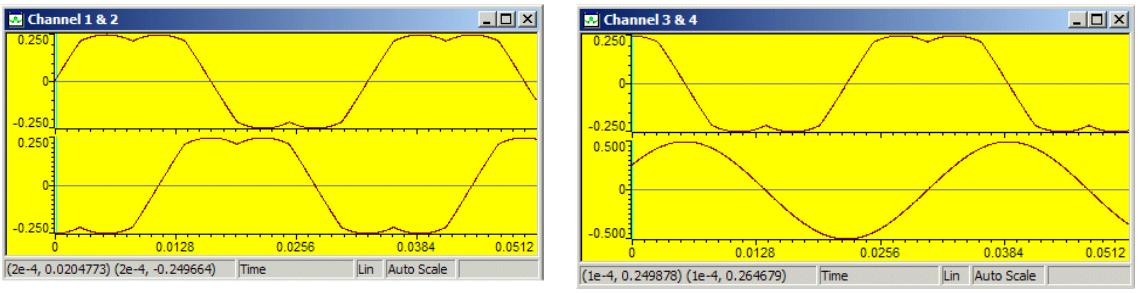
After verifying SVGEN_DQ module in phase 1a, the PWM_DRV module is tested by looking at the six PWM output pins. A simple 1^{st}–order low-pass filter RC circuit may be created to filter out the high frequency components. The selection of R and C value (or the time constant, τ) is based on the cut-off frequency (f_c), for this type of filter the relation is as follows:

$$\tau = RC = \frac{1}{2\pi f_c} \qquad (1)$$

For example, R = 1.8 kΩ and C = 100 nF, it gives f_c = 884.2 Hz. This cut-off frequency has to be below the PWM frequency.

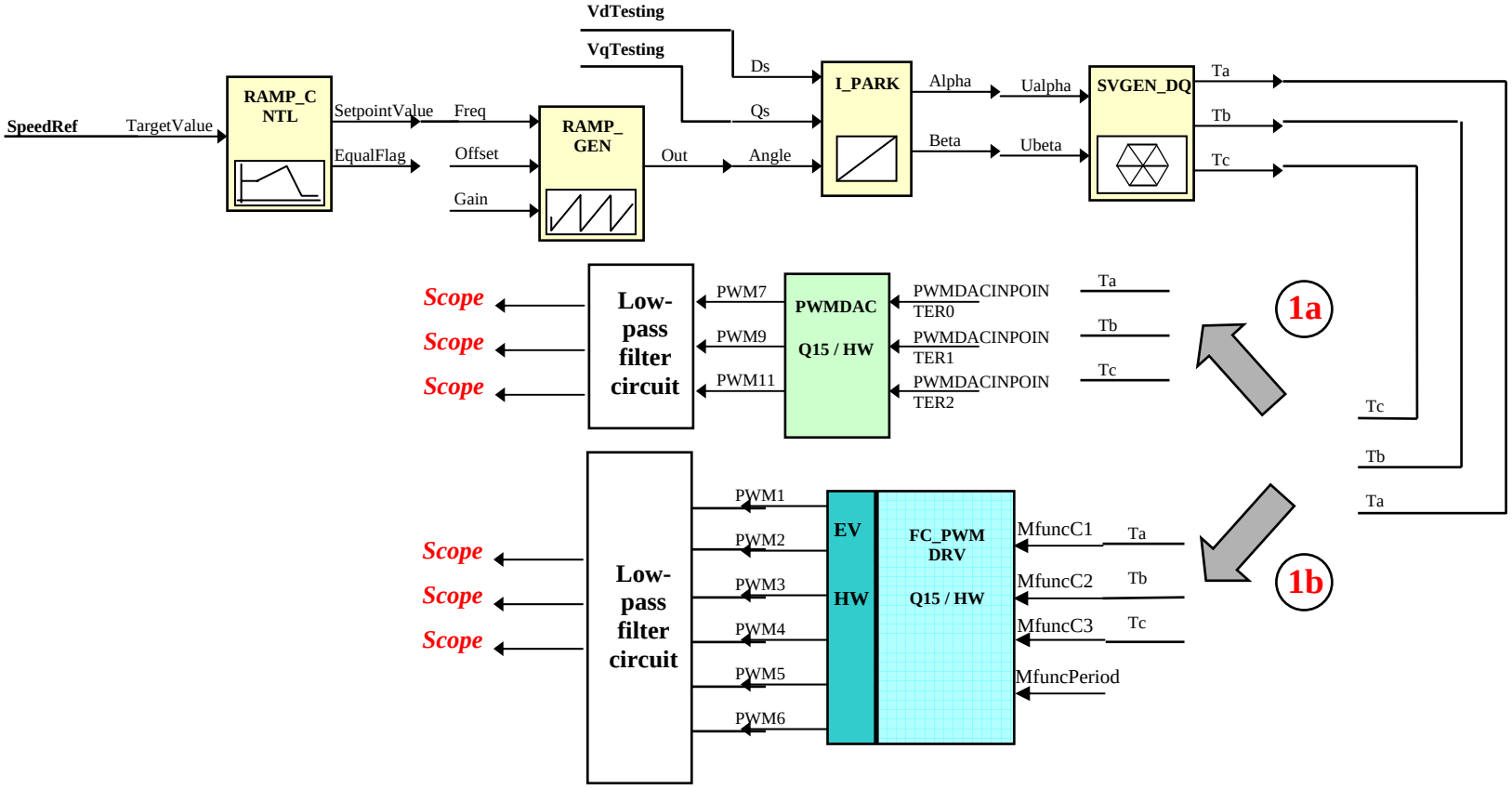
Once the low-pass filter is connected to the PWM pins of the x2812/x2808 eZdsp, the filtered version of the PWM signals are monitored by oscilloscope. The waveform shown on the oscilloscope should appear as same as one shown in figure 7. It is emphasized that the Ta waveform may be out of phase comparing with the filtered PWM1 signal. This means that the Ta waveform is the filtered PWM2 signal, which is complementary with PWM1 signal (i.e., PWM1=1−PWM2) as seen in figure 1.

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



Channel 1: Ta, Channel 2: Tb, Channel 3: Tc, Channel 4: Ta - Tb

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 analog-to-digital conversion (ILEG2_DCBUS_DRV), clarke/park transformations (CLARKE/PARK). Now the PMSM motor and DMC1500 are ready to be connected since the PWM signals are successfully generated from phase 1 incremental build.

In the **build.h** header file located under ***..\pmsm3_1_281x\cIQmath\include*** (for F281x target) or ***..\pmsm3_1_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 rotor speed in per-unit.
- VdTesting (floating format): for changing the d-qxis voltage in per-unit.
- VqTesting (floating format): for changing the q-axis voltage in per-unit.

4.2a Phase 2a (ILEG2_DCBUS_DRV test)

The following steps are the major things in order to properly set up the ADC channels for two line currents and DC-bus voltage. The DMC1500 design will mainly be explained.

At JP24, install Ad0, Ad1 for ISenseU, ISenseV, respectively and Ad7 for VSenseBus. The analog outputs Ad0, Ad1, and Ad7 from DMC1500 are corresponding to the ADCIN0, ACDIN1, ADCIN7 channels, respectively, for x2812 or x2808 eZdsp. These ADCIN channels are necessary to be configured in the ILEG2_DCBUS_DRV init function.

Next, the offset and gain settings have to be properly made according to the base line current (I_b) and base DC-bus voltage (V_b), i.e., base peak line-line voltage. For example, let base line current be 5 amp and base DC-bus voltage be 320 volt. These following steps do not need to open CCS and actually to run the motor.

Line currents

- Install JP11, JP14, JP16 for offset of ISenseU, ISenseV, ISenseW, respectively.

- Adjust R17 (VOFFSET) such that at JP11, JP14, or JP16 it gives 0.1 volt. For different base line current, the voltage at these jumpers is computed as follows.

$$V_{JP11,JP14,JP16} = \frac{0.04\Omega \times I_b}{2} \quad \text{volt}$$

  where the leg resistance is 0.04 Ω (see schematics).

- Adjust R12 (IU) such that at JP24, ISenseU gives 1.5 volt for x2812 or x2808 eZdsp.

- Adjust R16 (IV) such that at JP24, ISenseV gives 1.5 volt for x2812 or x2808 eZdsp.

- Adjust R19 (IW) such that at JP24, ISenseW gives 1.5 volt for x2812 or x2808 eZdsp.

DC-bus voltage

- Install ×2 position for double DC-bus voltage (for rated 220-volt motor).
- Remove JP9 for disabling Vref.
- Power on the variac and increase voltage up to 160 volt (i.e., $0.5 \times V_b$) by measuring this voltage between BUS+ (TP1) and GND on the DMC1500 (see figure 4a/4b).
- Adjust R10 (VBUS) such that at JP24, VSenseBus gives 1.5 volt for x2812 or x2808 eZdsp.

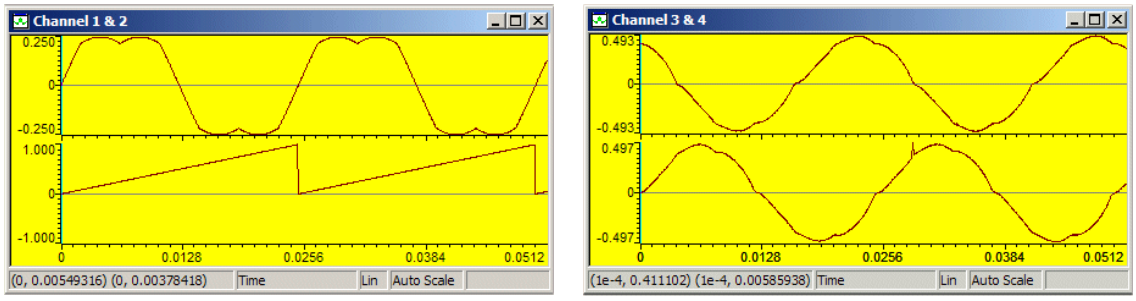
Once completed, the PMSM can be run and the actual line currents/DC-bus voltage can be measured by using ILEG2_DCBUS_DRV module. Assuming the ADCIN channels are correctly setup in ILEG2_DCBUS_DRV init function as explained before. Now the PMSM is run at a particular value of SpeedRef, appropriate value of VdTesting/VqTesting, and at the appropriate DC-bus voltage.

4.2b Phase 2b (CLARKE/PARK test)

Three measuring line currents are transformed to two phases dq currents in stationary reference frame. The outputs of this module can be checked via the PWMDAC module with external low-pass filter and an oscilloscope as follows:

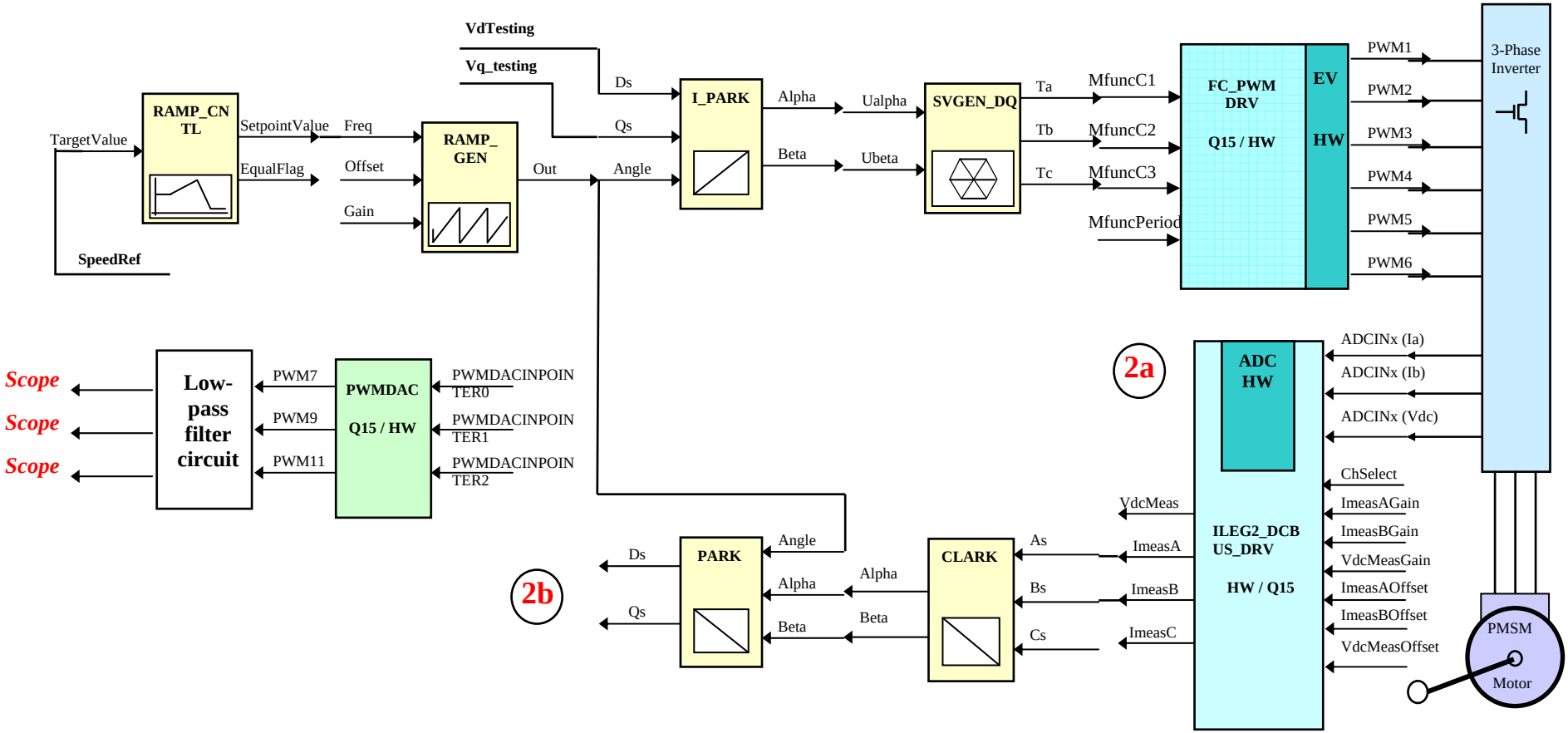
- The clark1.Alpha waveform should be same as the clark1.As waveform.
- The clark1.Alpha waveform should be leading the clark1.Beta waveform by 90^o at the same magnitude.

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



Channel 1: Ta, Channel 2: RMPGEN output, Channel 3: phase-a current, Channel 4: phase-b current

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 dq-axis current regulation performed by PID_REG3 modules. To confirm the operation of current regulation, the gains of these two PID controllers are necessarily tuned for proper operation.

In the **build.h** header file located under ***..\pmsm3_1_281x\cIQmath\include*** (for F281x target) or ***..\pmsm3_1_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.

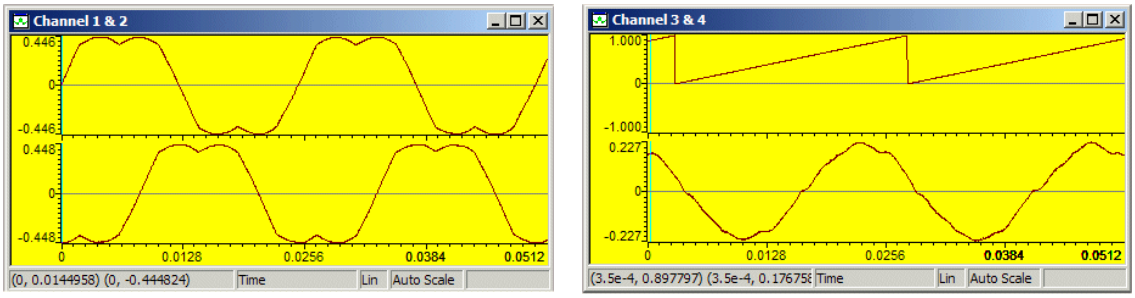
In the software, the key variables to be adjusted are summarized below.

- SpeedRef (floating format): for changing the rotor speed in per-unit.
- IdRef (floating format): for changing the d-qxis reference current in per-unit.
- IqRef (floating format): for changing the q-axis reference current in per-unit.

In this build, the motor is supplied by AC input voltage and the (AC) motor current is dynamically regulated by using PID_REG3 module through the park transformation on the motor currents. The steps are explained as follows:

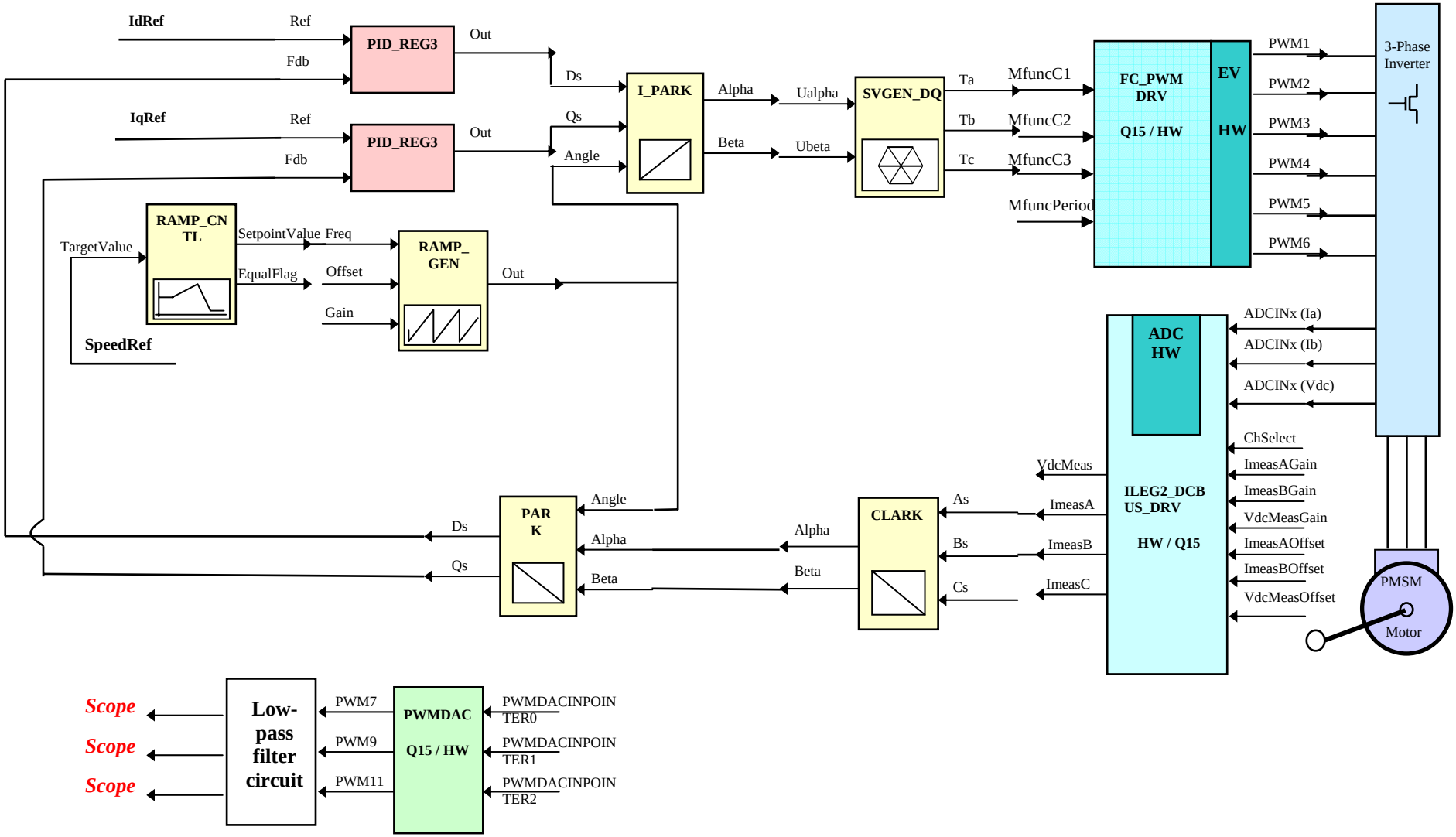
- Compile/load/run program with real time mode.
- Set SpeedRef to 0.2 pu (or another suitable value if the base speed is different).
- Gradually increase voltage at variac to get an appropriate DC-bus voltage.
- Check pid1_id.Fdb in the watch windows with continuous refresh feature whether or not it should be keeping track pid1_id.Ref for PID_REG3 module. If not, adjust its PID gains properly.
- Check pid1_iq.Fdb in the watch windows with continuous refresh feature whether or not it should be keeping track pid1_iq.Ref for PID_REG3 module. If not, adjust its PID gains properly.
- To confirm these two PID modules, try different values of pid1_id.Ref and pid1_iq.Ref or SpeedRef.
- For both PID controllers, the proportional, integral, derivative and integral correction gains may be re-tuned to have the satisfied responses.
- Reduce voltage at variac 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: Ta, Channel 2: Tb, Channel 3: RMPGEN output, Channel 4: phase-a current

Phase 3 Incremental System Build Block Diagram



4.4 Phase 4 Incremental System Build

Assuming section 4.3 is completed successfully, this section verifies the QEP driver and its speed calculation.

In the **build.h** header file located under ***..\pmsm3_1_281x\cIQmath\include*** (for F281x target) or ***..\pmsm3_1_280x\cIQmath\include*** (for F280x target) directory, select phase 4 incremental build option by setting the build level to level 4. 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 rotor speed in per-unit.
- IdRef (floating format): for changing the d-qxis reference current in per-unit.
- IqRef (floating format): for changing the q-axis reference current in per-unit.

The purpose of this step is to check out the encoder (QEP) interface driver software and SPEED modules. The steps are as follows:

- Compile/load/run program with real time mode.
- Set SpeedRef to 0.2 pu (or another suitable value if the base speed is different).
- Set the LockRotorFlag in watch window to 0 to use emulated rotor angle.
- Gradually increase voltage at variac to get an appropriate DC-bus voltage and now the motor is running with this reference speed (0.2 pu). Check that the Speed should be closed to SpeedRef.
- Use oscilloscope to view the electrical angle output, ElecTheta, from QEP_THETA_DRV module and the emulated rotor angle, Out, from RAMPGEN at PWMDAC outputs with external low-pass filters.
- Check that both ElecTheta and Out are of saw-tooth wave shape and have the same period.
- Check from Watch Window that qep1.IndexSyncFlag is set back to 0xF0 every time it reset to 0 by hand. Add the variable to the watch window if it is not already in the watch window.
- Reduce voltage at variac to zero, halt program and stop real time mode. Now the motor is stopping.

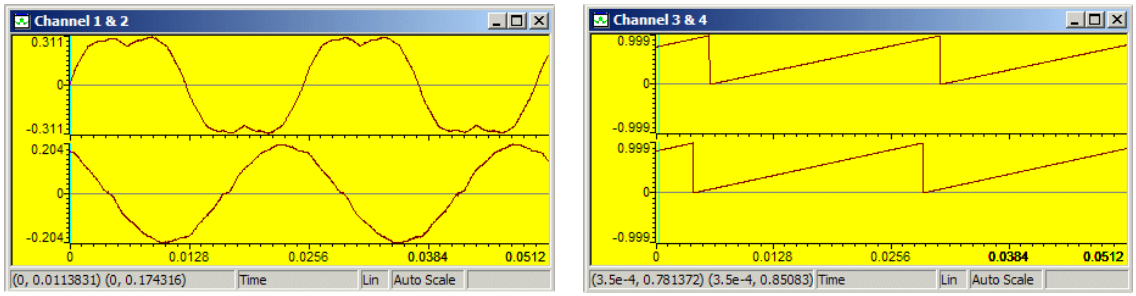
Next, the following steps are to verify and or perform calibration of the encoder. The steps are as follows:

- Make sure Timer 2 counter, EvaRegs.T2CNT, calibration angle, qep1.CalibratedAngle, and lock rotor flag, LockRotorFlag, are displayed in watch window
- Set the LockRotorFlag in watch window to 1 to lock the rotor to 0 rotor angle

- Write down the value of EvaRegs.T2CNT in watch window.

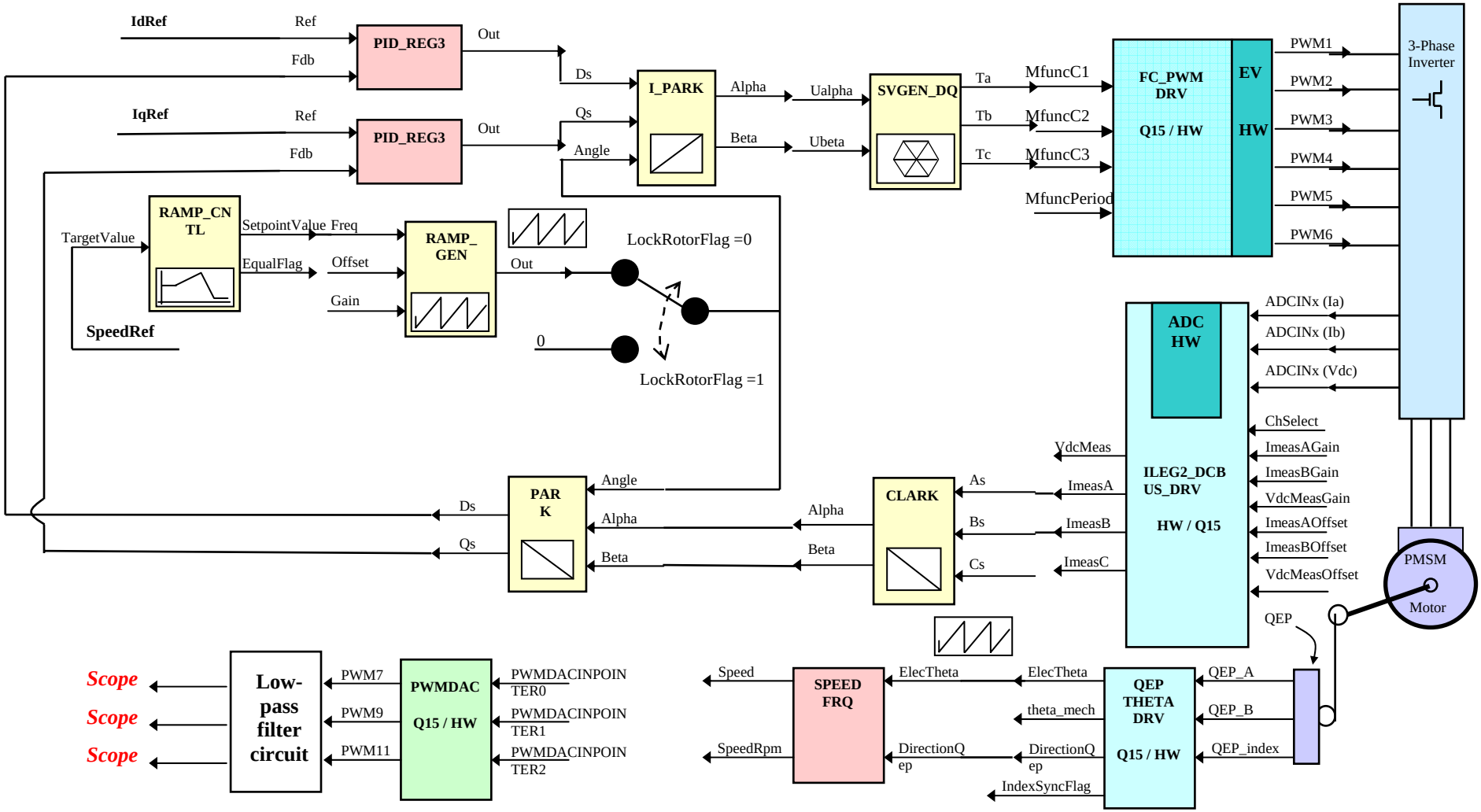
  Add any variable mentioned above to the watch window if it is not already in the watch window. The value of EvaRegs.T2CNT at this point is the calibration angle, i.e. offset of index pulse from 0 rotor angle. Negate the value and compare it to qep1.CalibratedAngle parameter in the main program, pmsm3_1.c. The difference should be close to K*2000 for some positive or negative integer K. Set qep1.CalibratedAngle to the negative of EvaRegs.T2CNT, if this is not true. 2000 is the timer count corresponding to 360 electrical degrees.

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



Channel 1: Ta, Channel 2: phase-a current, Channel 3: QEP angle, Channel 4: RMPGEN output

Phase 4 Incremental System Build Block Diagram



4.5 Phase 5 Incremental System Build

Assuming section 4.4 is completed successfully, this section verifies the speed regulator performed by PID_REG3 module. The system speed loop is closed by using the measured speed as a feedback.

In the **build.h** header file located under ***..\pmsm3_1_281x\cIQmath\include*** (for F281x target) or ***..\pmsm3_1_280x\cIQmath\include*** (for F280x target) directory, select phase 5 incremental build option by setting the build level to level 5. 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.

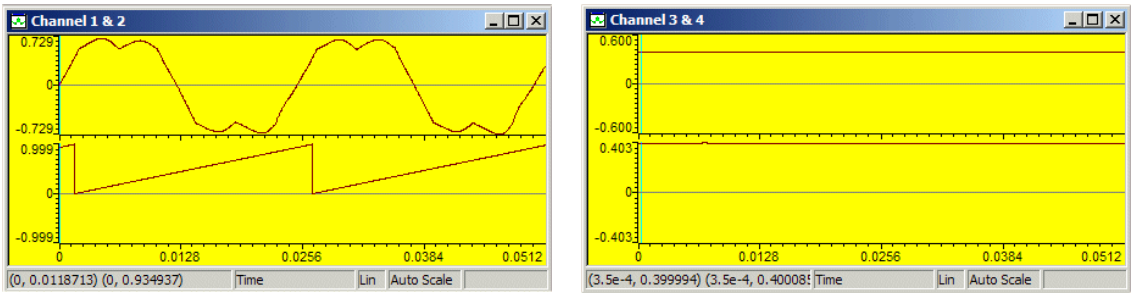
In the software, the key variables to be adjusted are summarized below.

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

The key steps can be explained as follows:

- Compile/load/run program with real time mode.
- Set SpeedRef to 0.2 pu (or another suitable value if the base speed is different).
- Gradually increase voltage at variac to get an appropriate DC-bus voltage and now the motor is running with this reference speed (0.2 pu).
- Compare Speed with SpeedRef 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 (positive or negative).
- For speed PID controller, the proportional, integral, derivative and integral correction gains may be re-tuned to have the satisfied responses.
- At very low speed range, the performance of speed response relies heavily on the good rotor position angle provided by QEP encoder.
- Reduce voltage at variac 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: Ta, Channel 2: QEP angle, Channel 3: speed reference, Channel 4: speed feedback

Phase 5 Incremental System Build Block Diagram