Application Note

AM243x/AM64x Single Chip Servo Motor Control Implementation and Benchmark

ABSTRACT

With the rise of Industrial 4.0 and smart factories, the precise, energy-efficient, safe and connected systems are in the great demanding. Electric motors account for 40 to 50 percent of global energy consumption and the industry is focused on improving the overall productivity of the motors. Constant-speed drives are inherently inefficient; these are being actively replaced with adaptable-speed drives controlled by processors. Intelligent industrial systems are adopting digital motor feedback systems. Coupled with industrial Ethernet for communication, they provide higher efficiency in factory and energy sectors. Connected drives have a variety of industrial communication protocols depending on the automation system used in the field.

The AM243x/AM64x Single Chip Servo Drive Demo showcases the AM243x/AM64x devices' ability to support a fully integrated real-time servo drive control and communication path. This path extends from receiving EtherCAT CiA402 target commands for torque/speed/position, to performing closed-loop FOC loop control of a connected motor, to passing the actual values back up to the EtherCAT PLC.

Project collateral and source code discussed in this document can be downloaded from the following URL: https://www.ti.com/lit/zip/sprad03.

Table of Contents

List of Figures

List of Tables

Trademarks

Code Composer Studio™ is a trademark of Texas Instruments.
Cortex® and Arm® are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
All trademarks are the property of their respective owners.

1 System Overview

The Single Chip Servo Demo showcases the AM64x and AM243x devices' ability to support a fully integrated real-time servo control and communication path. This path extends from receiving EtherCAT CiA402 target commands for torque/speed/position, to performing closed-loop FOC loop control of a connected motor, to passing the actual values back up to the EtherCAT PLC.

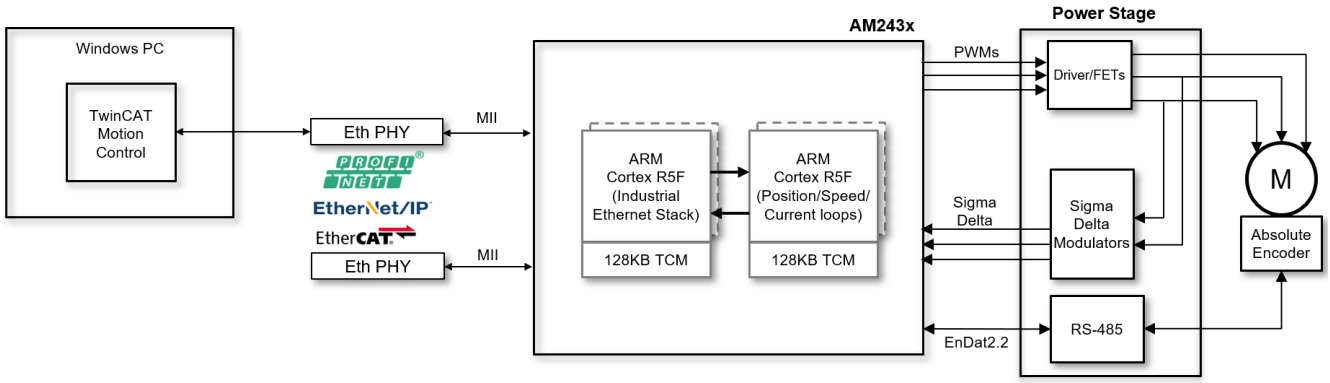

Figure 1-1. System Overview

2 Hardware Prerequisites

- Windows PC with TwinCAT
- TMDS243GPEVM or TMDS64GPEVM and power supply
- 3-Axis adapter interface board
- TIDEP-01015 3-axis board
- TIDA-01629 DRV8350 power stage board
- TIDA-00179 Absolute Encoder RS485 Transceiver board
- BLY342D-48V-3200 Anaheim Automation 3-phase Brushless DC motor
- ROQ-437 EnDat2.2 Encoder with cable
- 24 V barrel jack power supply for TIDEP-01015 board
- 24 V power supply for the screw terminal input of TIDA-01629
- Ethernet patch cable
- 2x micro USB cables to connect the PC to the emulator and UART ports on the EVM

3 Hardware Configuration

Figure 3-1 shows the demo setup.

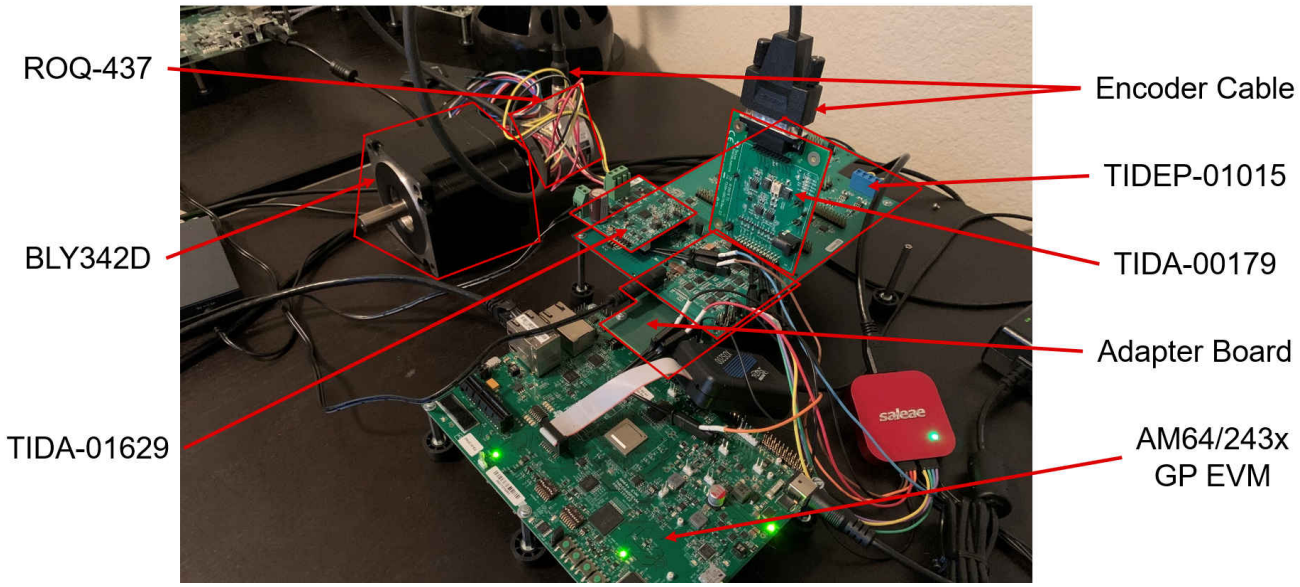

Figure 3-1. Hardware Configuration

4 HW Pinouts, Default Jumpers, and Connections

- Download https://www.ti.com/lit/zip/sprad03 and unzip it to <DEMO_INSTALL_ROOT>
- 3-Axis Adaptor Board: <DEMO_INSTALL_ROOT>\pinouts\3-Axis Adapter Enablement v2.pdf
- Pinout cheat sheet: <DEMO_INSTALL_ROOT>\pinouts\AM64x_3_Axis_Adapter_Interface_Board_v2.xlsx

5 Schematics

- Download and unzip it to <DEMO_INSTALL_ROOT>
- 3-axis adapter board: <DEMO_INSTALL_ROOT>\schematics\TIDEP-01015Rev E1.1_Sch.pdf
- TIDA-01629 Power Stage: <DEMO_INSTALL_ROOT>\schematics\TIDA-01629_E2_Sch.pdf
- TIDA-00179 Universal Encoder Board: <DEMO_INSTALL_ROOT>\schematics\TIDA-00179 E2.pdf
- TMDS243GPEVM or TMDS64GPEVM: <DEMO_INSTALL_ROOT>\schematics\PROC101E2_SCH.pdf
- Passive Adapter Board: <DEMO_INSTALL_ROOT>\schematics\MS_TI_AM64x_EVM_3-AXIS_INTERFACE_BOARD_SCH_REV_E1.pdf

6 Jumper Settings and Descriptions

Table 6-1 describes jumper settings and descriptions found on the TIDEP-01015 (rev 1.1) board.

Table 6-1. Jumper Settings and Descriptions

Ref ID	Description	Settings	Notes
J1, J3, J5	FOC loop feedback source	**(1,2): current fb (default)** 2,3: voltage fb open: open loop	pin numbering starts from bottom pin (opposite of most other pin groups on the board)
J2, J4, J6	ADC selection	closed: SAR**open: Sigma Delta (default)**	
J7	Safe torque off switch inputs	(1,2)(3,4)	pins 1,2 and pins 3,4 shorted is normal operation mode.
J8	Sigma Delta clock select	**(1,2): ECAP PWM (default)** 2,3: HW delay clk open: invalid	
J20	GND		
J21	3V3_D		

7 LEDs

Table 7-1 describes LEDs found on the TIDEP-01015 (rev 1.1) board.

Table 7-1. LEDs

Ref ID	Description	Notes
D1	M1 drive enable	LED on is error
D2	M2 drive enable	LED on is error
D3	M3 drive enable	LED on is error
D4	24V0	Should be always on
D5	3V3	Should be always on

8 Software Architecture

The Single-chip Servo Drive Demo was architected around a central real-time path that is made up of:

- ICSSG1 - EtherCAT Slave Controller Firmware
- R5FSS1_0 - EtherCAT Slave Stack implementing CiA402 using FreeRTOS
- IPC provided in the MCU+ SDK
- R5FSS0_0 - Closed loop FOC control capable of current, speed, or position closed-loop control
- EPWM - Enhanced PWM peripherals to generate waveforms based on the output of the FOC loop
- ICSSG0 - Sigma Delta firmware in PRU0 for phase current feedback and EnDat2.2 firmware in PRU1 for angle/position/speed feedback

This real-time path demonstrates the components needed to make up a bare-bones Servo Drive from receiving data from an EtherCAT master to closing the real-time control loops in order to precisely control a directly connected motor.

The Software Architecture diagram is shown in Figure 8-1.

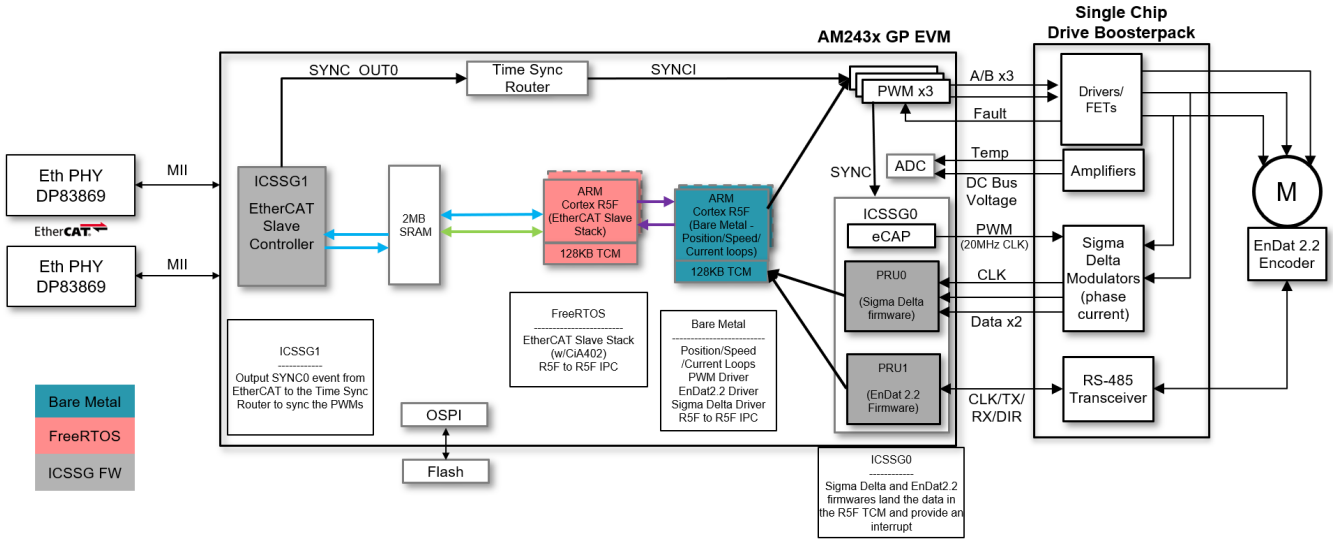

Figure 8-1. Software Architecture

9 Actuation and Feedback Timing

The precision of the actuation and feedback timing in the demo is crucial to the correct performance of the FOC loop calculations and in turn the overall motor control performance. To achieve the goal, this demo setup is as follow:

- 50 KHz PWM cycle time
  - PWM resolution of greater than 12 bits achieved due to EPWM peripheral clocking at 250 MHz
    - 50 KHz / 4 ns = 5000, which equates to approximately 12.28 bits of resolution
- 100 KHz FOC loop update
  - 2x updates per PWM cycle
  - Allows for 10 µs from trigger point until feedback can be captured, FOC loop close, and PWM values updated in the shadow registers
  - Achieved timing diagram here:

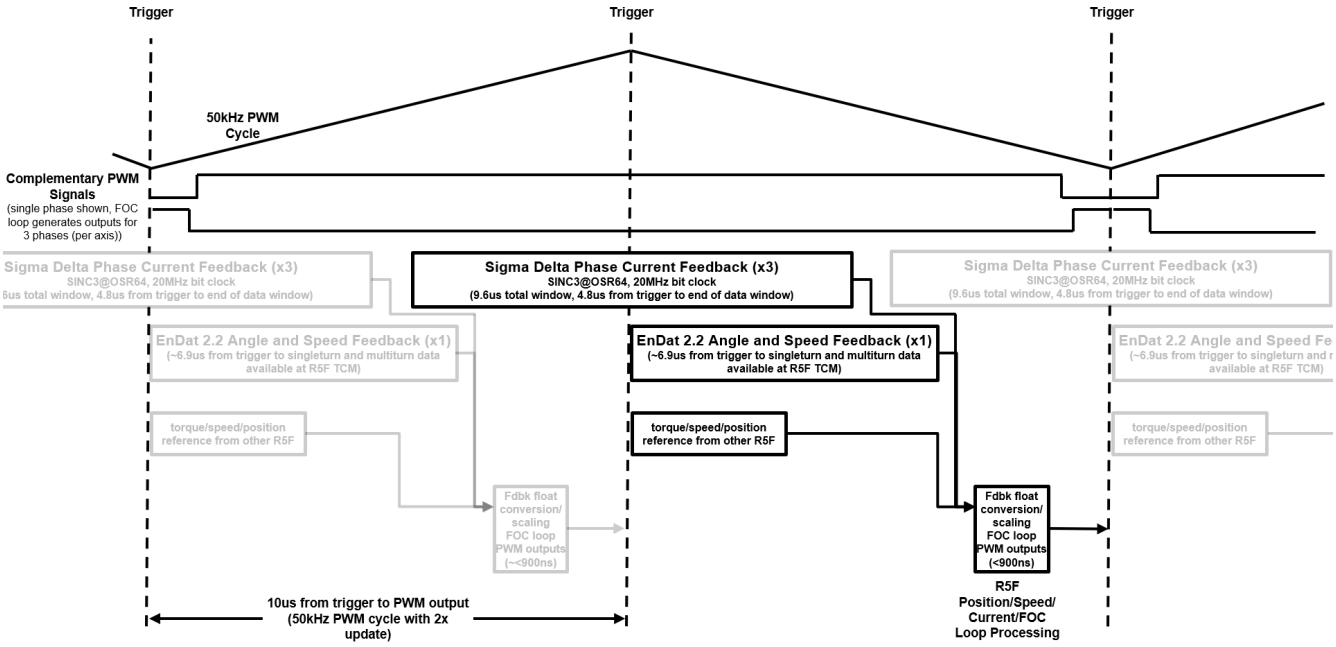

Figure 9-1. Timing Diagram

- Precise triggering for both EnDat2.2 and Sigma Delta feedback data
  - A synchronization technique between the PWM peripheral and the IEP Timer of ICSSG0 is used in order to precisely place the triggers anywhere along the PWM cycle
  - EPWM SYNCO signal resets the IEP Timer counter and CMP1 and CMP2 values are used to trigger the start points for both EnDat and Sigma Delta

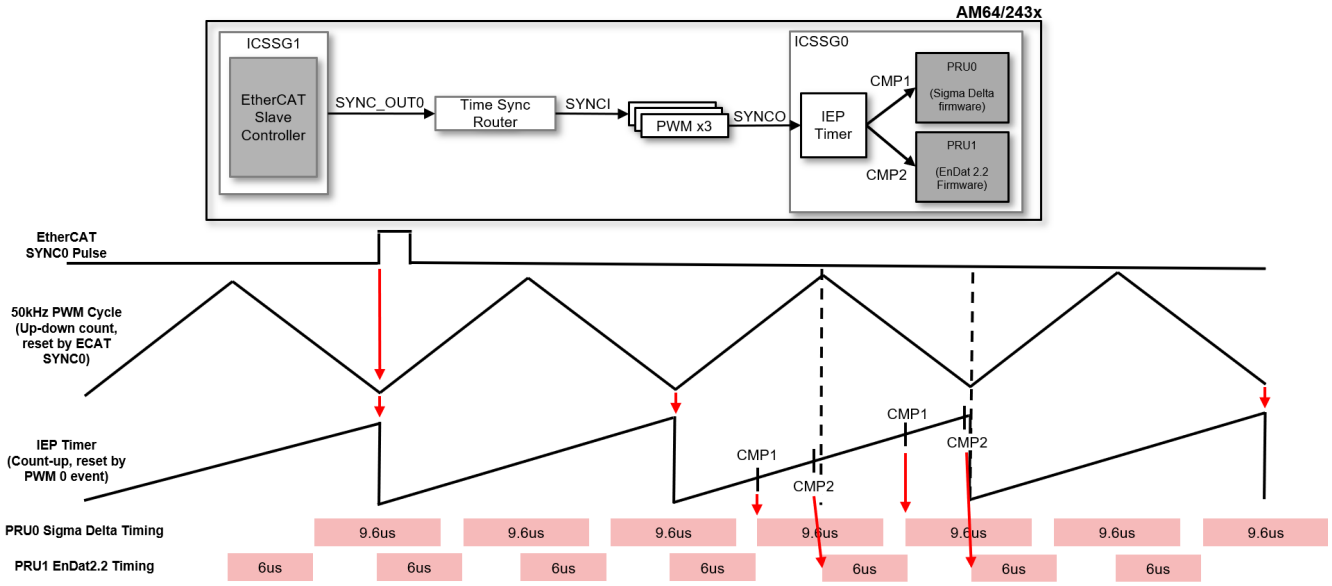

Figure 9-2. Synchronization Techniques

– Technique visualized below and results shown in 'Benchmark results' section

10 Benchmark Results

10.1 Motor Control R5F Processing Time

Measurements are taken for the duration of approximately 30 seconds (approximately 3 millions iterations) with motor spinning in the loop. Max values are used for all top-level reported results.

GPIO output toggle is measured at 180 ns. This value was subtracted from maximum measured times in the 'Max normalized' rows below in the Detailed results tables.

50 KHz PWM cycle with 2x update (100 KHz processing cycle time).

Compile flags: -mcpu=cortex-r5 -mfloat-abi=hard -mfpu=vfpv3-d16 -mlittle-endian -O3

Settings are used in 'settings.h':

Table 10-1. Build Options

BUILDLEVEL	OPEN_LOOP_IQ_ID	CLOSED_LOOP_IQ_ID	CLOSED_LOOP_SPEED	CLOSED_LOOP_POSITION	CLOSED_LOOP_CIA402
PID_TUNE_LEVEL	NO_TUNING	NO_TUNING	NO_TUNING	NO_TUNING	NO_TUNING
DEBUG_LEVEL	DEBUG_BUFFERS_OFF	DEBUG_BUFFERS_OFF	DEBUG_BUFFERS_OFF	DEBUG_BUFFERS_OFF	DEBUG_BUFFERS_OFF

Table 10-2. R5F Motor Control Processing Time

Function	Motor Control R5F Processing Time					Unit
	Open Loop Iq/Id	Closed Loop Iq/Id	Closed Loop Speed	Closed Loop Position	Closed Loop CiA402	
Angle/Position/Speed Calculations	132	128	156	156		ns
Phase Current Scaling, conversion to FP	20	180	316	236		ns
FOC loop and PI controllers	180	252	340	404		ns
Write to PWM outputs and save state for next cycle	84	84	84	84		ns
Total (Aggregate Added from above)	416	644	896	880		ns
Total (Measured all at once) (PRECOMPUTE_LEVEL = NO_PRECOMPUTE)	388	644	884	892	860	ns
Total (Measured all at once) ADC values and Clarke ran early (PRECOMPUTE_LEVEL = PRECOMPUTE_CLARKE	396	484	740	772		ns

For detailed motor control R5F processing time, see the Section 14

10.2 Trigger/Capture Point to R5F ISR Entry

EnDat position latch has been experimentally shown on a logic analyzer to be < 10 ns from the desired trigger point. Observed values have been between 0-8 ns from desired capture point. Figure 10-1 shows the trigger/capture points:

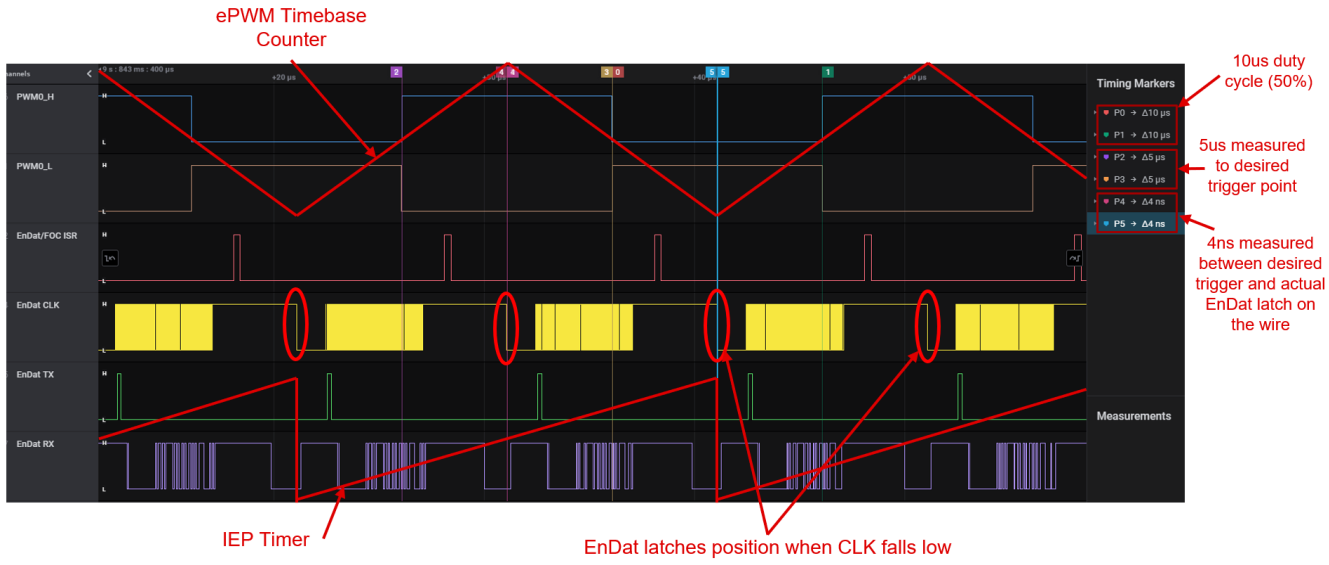

Figure 10-1. Trigger and Capture Points

11 Detailed Demo User's Guide

11.1 Step 0. Getting the Software and Building

- See the following link for details: Section 12

11.2 Step 1. Getting Started With the Hardware

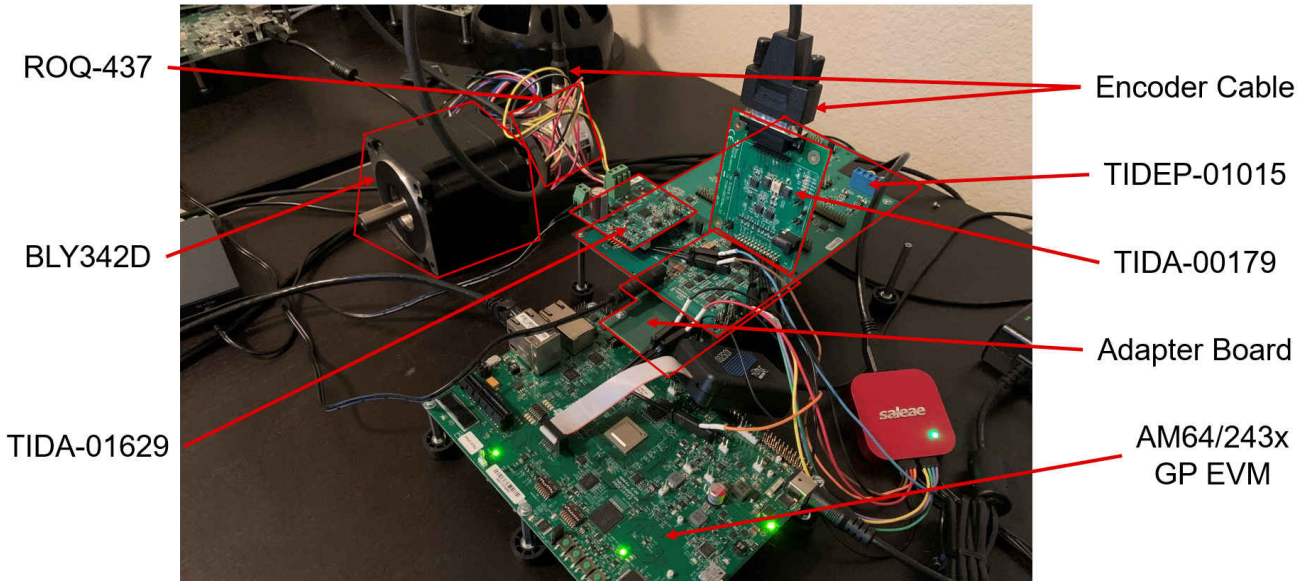

Figure 11-1. Hardware Setup

- Plug the 5 PCBs together, referencing the image above and the instructions below
  - The Adapter Board, the TIDEP-01015 board, and the TMDS243GPEVM can only be connect together in one way, seen above. Plug those three boards together and add all the standoffs as necessary
  - The TIDA-01629 power stage board should be plugged into the first boosterpack slot that is closest to the TMDS243GPEVM and the screw terminals should be facing the outside of the board. This first slot corresponds to PWM0-2 as well as SD channels 0-2
    - There should be NO jumpers populated on this board
  - The TIDA-00179 encoder transceiver board should be plugged into the first encoder slot that is closest to the TMDS243GPEVM with the right-angle connector and the DB-15 connector closest to the TMDS243GPEVM. This first slot corresponds to EnDat channel 0
    - There should be NO jumpers populated on this board
- Configure the jumpers on the TIDEP-01015 board as shown by the red rectangles below and described in the table. Also note that the screw terminals connectors in J7 need to be shorted as shown in Figure 11-2 as described in the table (this connector enables an STO type function but is currently being bypassed).

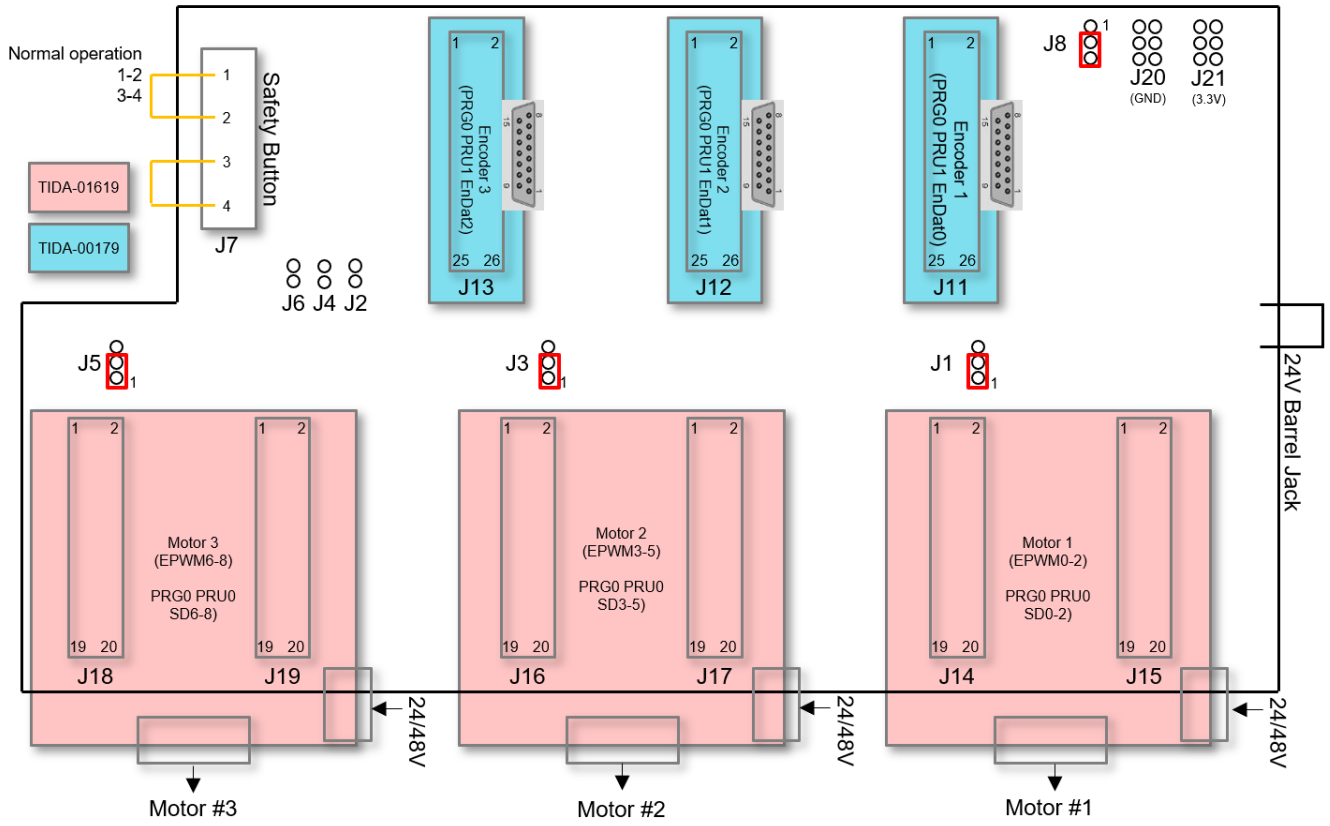

Figure 11-2. TIDEP-01015 Jumper Settings

- Jumper settings table

Table 11-1. Jumper Settings Table

Ref ID	Description	Settings	Notes
J1, J3, J5	FOC loop feedback source	**(1,2): current fb (default)** 2,3: voltage fb open: open loop	pin numbering starts from bottom pin (opposite of most other pin groups on the board)
J2, J4, J6	ADC selection	closed: SAR**open: Sigma Delta (default)**	
J7	Safe torque off switch inputs	(1,2)(3,4)	pins 1,2 and pins 3,4 shorted is normal operation mode.
J8	Sigma Delta clock select	**(1,2): ECAP PWM (default)** 2,3: HW delay clk open: invalid	
J20	GND		
J21	3V3_D		

- Connect the motor wirings to the TIDA-01629 power stage screw terminal at J6. **Be sure to notice that the motor phase wirings are the 6 thicker wires**.

Note

There are also 5 other smaller, single-colored wires that are for an incremental encoder that should be left unconnected.

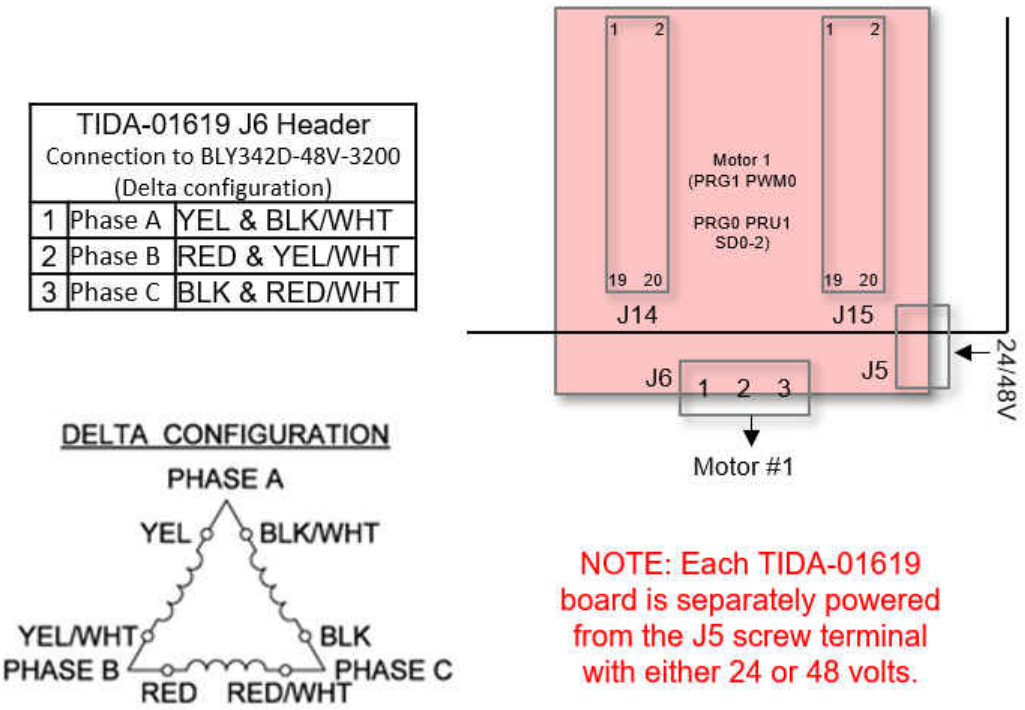

Figure 11-3. TIDA-01619 J6 Header 1

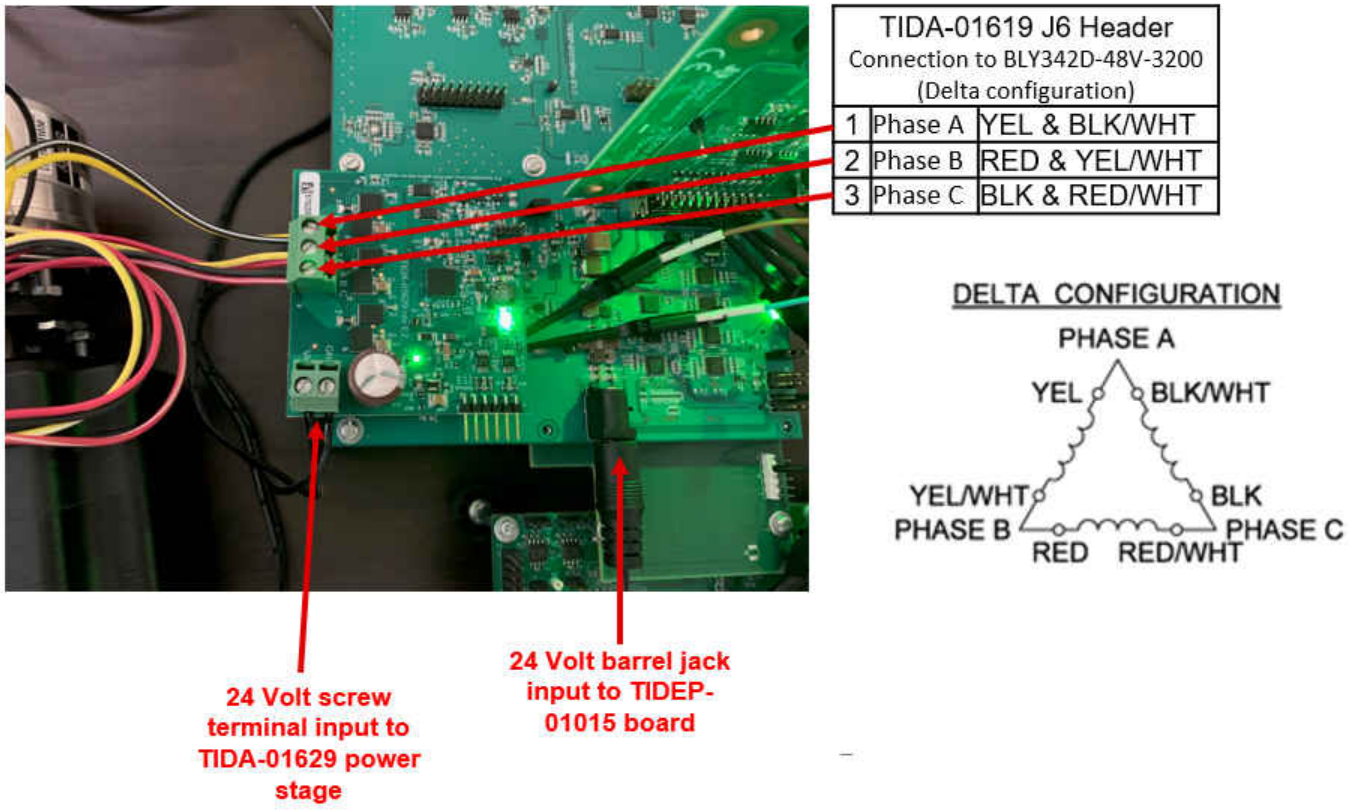

Figure 11-4. TIDA-01619 J6 Header 2

- Connect the encoder cable from the ROQ-437 EnDat2.2 encoder to the TIDA-00179 encoder transceiver board

- Connect 24V to the screw terminal of the TIDA-01629 board as well as to the barrel jack connector on the TIDEP-01015 board (pictured 2 steps above). DO NOT CONNECT power to the barrel jack of the TIDA-00179 board, this board receives power through the TIDEP-01015 board

11.3 Step 2. Configure ROQ437 EnDat2.2 Encoder for Faster EnDat 2.2 Recovery Time (only needs to be done once the first time you use the ROQ437 encoder)

The ROQ437 defaults to a slower EnDat2.1 recovery time between position latches. EnDat2.2 enables a faster recovery time but we need to set a non-volatile bit one time in the ROQ437 before running the rest of the demo to reduce the recovery time. This is a workaround for now that can be incorporated into the demo code at a later time.

Explanation from the Heidenhain documentation (https://www.heidenhain.us/wp-content/uploads/Bidirectional_Interface_for_Position_Encoders-1.pdf): The extended EnDat interface version 2.2 is compatible in its communication, command set and time conditions with the previous version 2.1, but also offers significant advantages. It makes it possible, for example, to transfer additional information with the position value without sending a separate request for it. The interface protocol was expanded and the time conditions were optimized as follows: Increased clock frequency (CLOCK) (16 MHz), Optimized calculating time (position value acquisition within 5 µs), Minimized dead time (recovery time) (1.25 µs to 3.75 µs), Expanded power supply range (3.6 V to 5.25 V or 14 V at the encoder)

- Plug in all of the hardware to connect the EnDat2.2 encoder into the slot closest to the TMDS243GPEVM
- Import into CCS the 'endat_diagnostic' example from the MCU+ SDK located at '%SDK_INSTALL_DIR%\examples\motor_control\endat_diagnostic'
- Build the example and load it into MAIN_Cortex®_R5_0_0
- Setup the terminal in Code Composer Studio™ (CCS) to view the diagnostic output (my port is COM6 below but yours may differ):

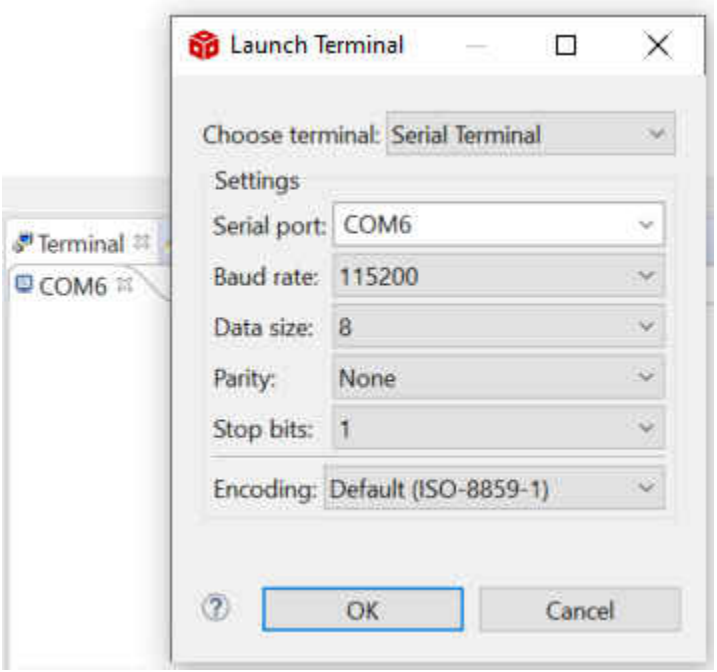

Figure 11-5. COM Port for Diagnostic Output

- Click 'Resume (F8)' to run the Diagnostic
- Select 'y' for Multi channel configuration, select 'y' for select channel 0, and select 'n' for channel 1 and channel 2.

```
COM6

EnDat firmware  : 0.2.0 (release)

Multi channel configuration ? [y/N]: y

selected multi channel configuration

select channels to be used in multi channel

select channel 0 [Y/n]: y

select channel 1 [Y/n]: n

select channel 2 [Y/n]: n

channels 0   selected


                              CHANNEL 0

                                        EnDat 2.2 rotary encoder       ID: 1042255 01  SN:   68423189

Position: 37 bits (singleturn: 25, multiturn: 12) [resolution: 33554432 M/rev]

Propagation delay: 335ns
```

Figure 11-6. EnDat 2.2 Diagnostic Output

- After these selections the diagnostics should communicate with the encoder and return ID, SN, and information about the resolution and propagation delay
- Connect a logic analyzer to to the EnDat CLK, TX, and RX pin (these pins can be accessed on the right angle connector of the TIDA-00179 board as shown below)

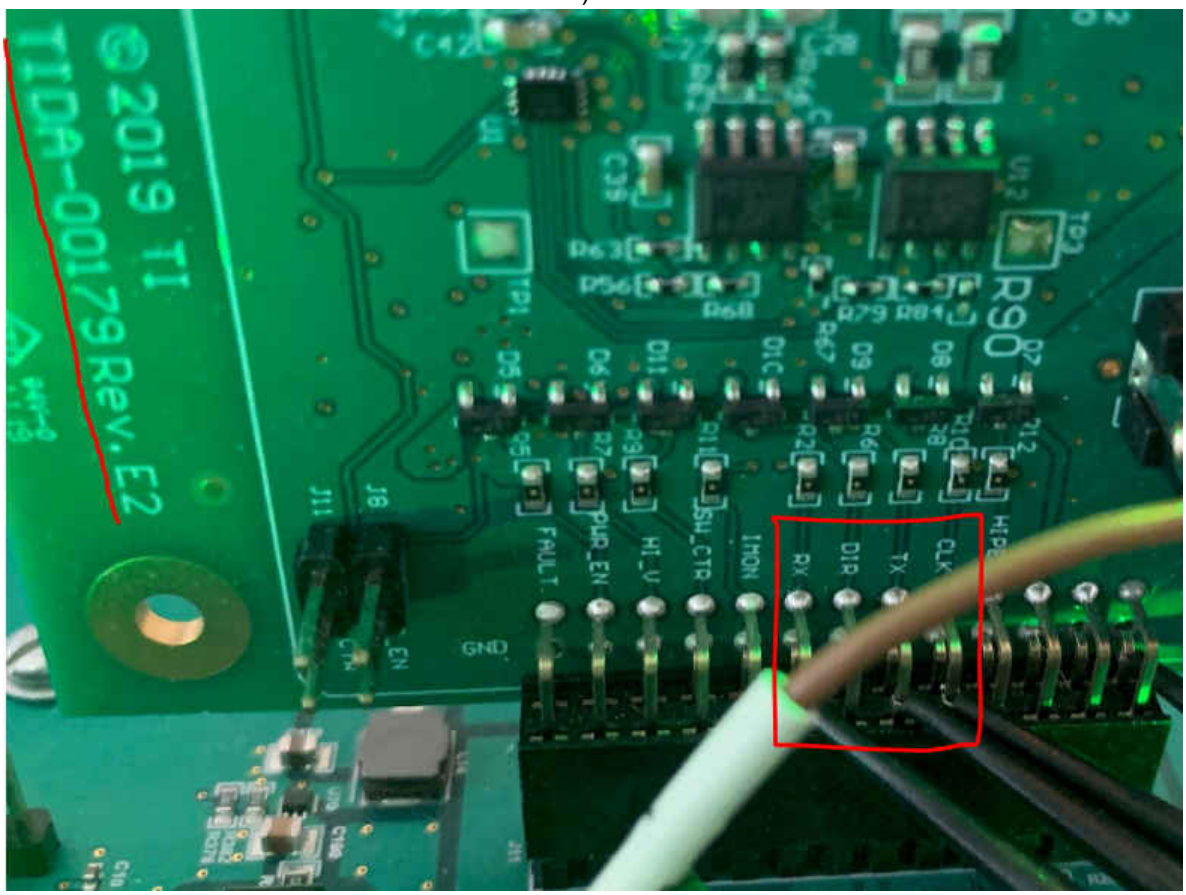

Figure 11-7. Measure EnDat CLK, TX and RX Pins

- Select option '107' in the diagnostic and enter '8000' for the frequency in Hz. This causes the program to continuously request data from the encoder like that shown in Figure 11-8.

```
enter value: 107
enter frequency in Hz: 8000

press enter to stop the position display


CHANNEL 0

         position,       revolution, crc errors, f1, f2
167.691802978516,               82,          1,  0,  1
167.691818237305,               82,          1,  0,  1
167.691802978516,               82,          1,  0,  1
167.691802978516,               82,          1,  0,  1
167.691787719727,               82,          1,  0,  1
167.691787719727,               82,          1,  0,  1
```

Figure 11-8. EnDat 2.2 Position Output

- Run the logic analyzer while the simulated position loop is running and look at the recovery time on the RX signal (the time the RX signal is held high after the CLK stops):

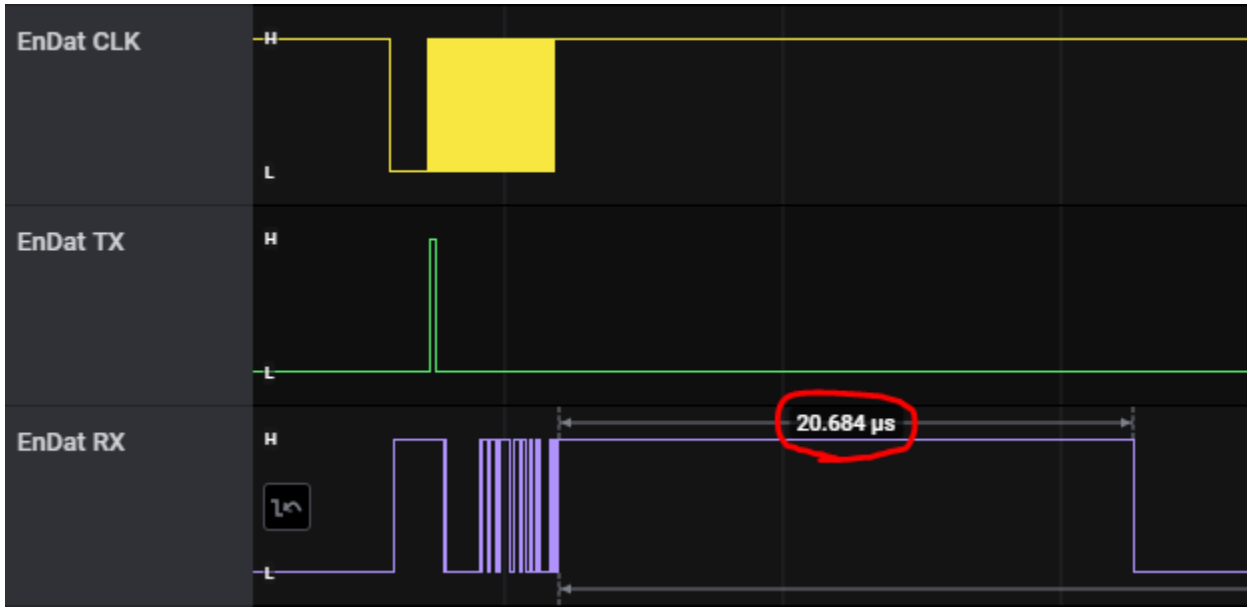

Figure 11-9. RX Signal (Before Change)

- To set the bit to enable EnDat2.2 recovery mode select diagnostic option: 10, parameter address: 3, and parameter value: 1

```
enter value: 10
enter parameter address (hex value): 3
enter parameter (hex value): 1
```

Figure 11-10. EnDat 2.2 Recovery Mode Change

- Then select diagnostic option '5' to reset the encoder in order for the setting to take effect
- Re-run option 107 with 8000 as the frequency to ensure that the setting has been saved and that the recovery time is less than 3.75 µs:

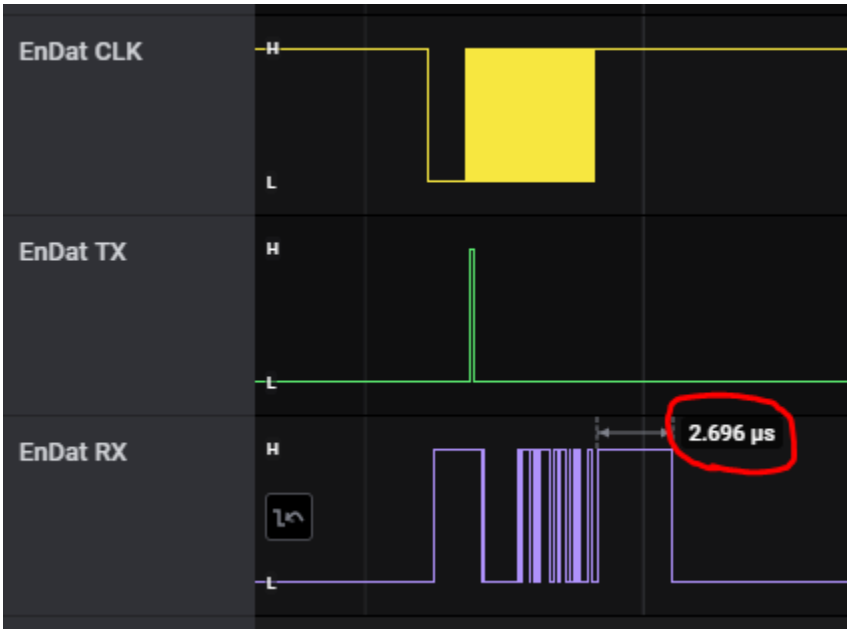

Figure 11-11. RX Signal (After Change)

- If the recovery time is still around 20 µs, you may need to try the previous steps one more time (option 10, address 3, value 1, then option 5 to reset)

At this point, your encoder should be configured to run in EnDat2.2 recovery mode, which allows for the tight timings that is needed throughout the rest of the demo.

11.4 Step 3. Open Loop Iq Control (BUILDLEVEL == OPEN_LOOP_IQ_ID)

This build level verifies that Sigma Delta is working and its DC offsets are getting set properly, the EnDat mechanical theta offset is being calculated from EnDat feedback, and the PWM path is working properly to spin the motor in open loop control.

1. Go to the settings.h file and change the definitions to match the following (open loop control with debug buffers on):

```
#define  BUILDLEVEL              OPEN_LOOP_IQ_ID
#define  PID_TUNE_LEVEL          NO_TUNING
#define  DEBUG_LEVEL             DEBUG_BUFFERS_ON
#define  DEBUG_WRAP_TYPE         DEBUG_BUFFER_SINGLE
#define  PRECOMPUTE_LEVEL        NO_PRECOMPUTE

/* The number of ISR cycles to use the same target from
#define  CYCLES_PER_TARGET       6000
#define  TARGET_BUFF_SIZE        CYCLES_PER_TARGET * 8

/* Iq testing value for open loop */
#define  IQ_TESTING              0.2
/* Max speed change request allowed between updates */
#define  MAX_SPD_CHANGE          0.12
/* Max position change request allowed between updates
#define  MAX_POS_CHANGE          0.03        /* 500RPMs
```

Figure 11-12. Open Loop IQ ID - BUILDLEVEL

2. Build the project in Debug mode, then load it into MAIN_Cortex_R5_0_0.

3. Once the project is loaded, open the 'single_chip_servo.c' file and find the following line, then right-click and select 'Run to Line'.

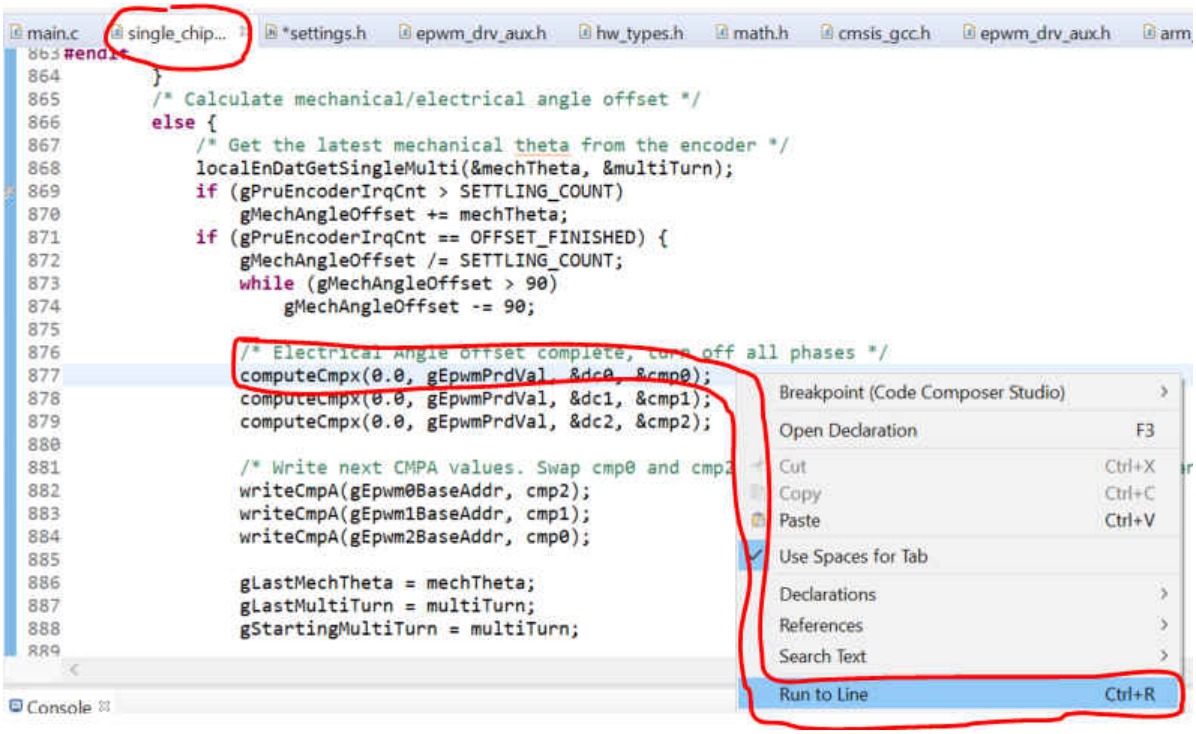

Figure 11-13. Open Loop IQ ID - Run to Line

4. The motor shaft should lock into place (at an electrical angle of 0) before the core reaches the line and halts.
5. Once the core has reached the line, the program has already set all phases to 0,0,0 and read all three phase currents SETTLING_COUNT times (defaults to 8192) and computed the DC offset from. Once the Sigma Delta offsets are found the program locks the shaft to an electrical angle of 0 by setting all phases to 1,0,0. Once locked the program reads the EnDat2.2 mechanical angle feedback value SETTLING_COUNT times (8192) and computes the offset between mechanical theta and electrical theta.
6. To ensure that your feedback paths (Sigma Delta and EnDat2.2) are working properly, open the 'Expressions' window and observe 'gSddfChOffsets' and 'gMechAngleOffset'.
   a. The Sigma Delta offsets should be somewhere in the range of -1000 to 1000. If your offsets are well outside this range, or stuck at mid-scale or full-scale values (131072 or 272144) then you need to check that your jumper settings are selecting Sigma Delta (J2, J4, J6 left open) and all 3 phase currents (J1, J3, and J5 connecting pins 1 and 2 (the two pins closest to the power stage)). If all jumper settings are correct and you still have a channel stuck outside of this range there may be a HW issue (has been observed on a single channel on one board so far).
   b. The gMechAngleOffset should be a value between 0 and 90 and represents the offset between the mechanical angle from the encoder and the electrical angle inside the motor windings. This value should stay relatively constant between multiple runs of the demo. If you notice this value is varying between different runs then the coupler between the motor and the encoder shafts may have become loose which will cause slippage. If there is slippage in the coupler while running the demo then the electrical angle given to the Park transform will be wrong and the motor will not spin smoothly (this was observed on the first motor tested).

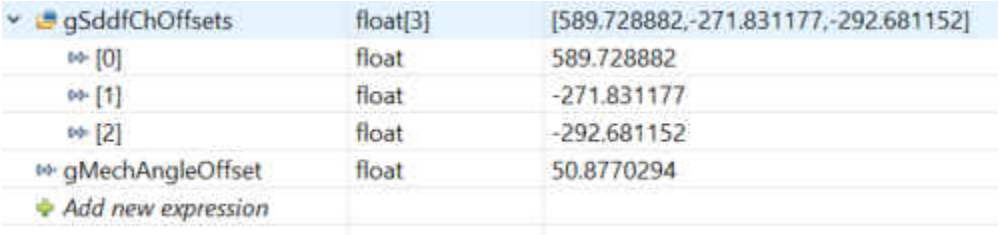

Figure 11-14. Open Loop IQ ID - Electrical Angle

7. Once the Sigma Delta and EnDat2.2 paths are confirmed above you can resume the core to start running the open loop control, the motor should start spinning.
8. As the motor is spinning you can view the debug outputs:
   a. Connecting to the OTHER R5F core 'R5_1_0' and load the SYMBOLS ONLY:

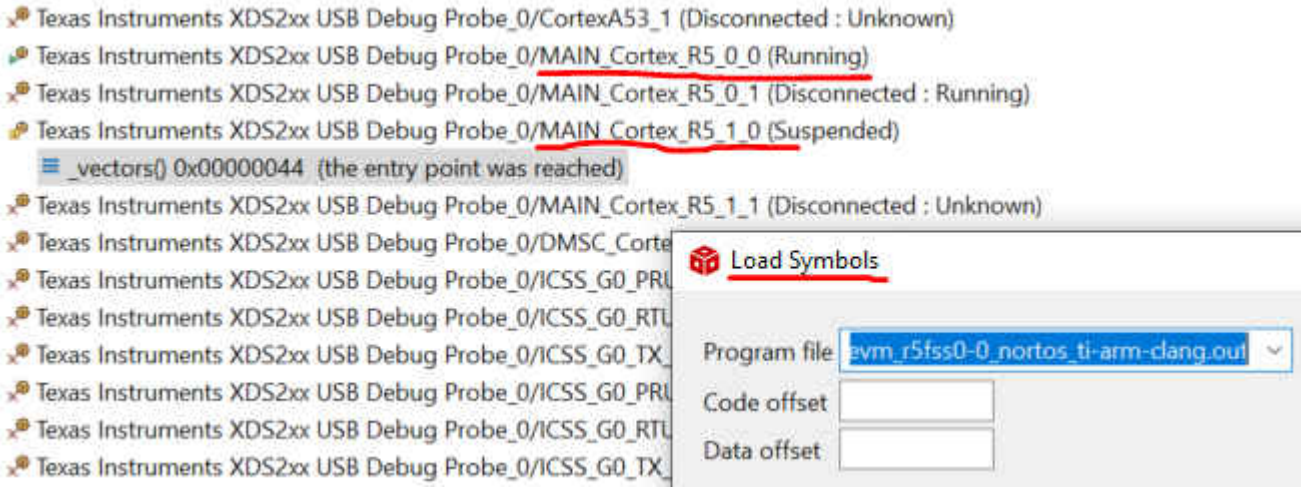

Figure 11-15. Open Loop IQ ID - Load Symbols

   b. Import the three CCS graphs (to show the Phase A, B, and C PWM output values being used in open loop control.

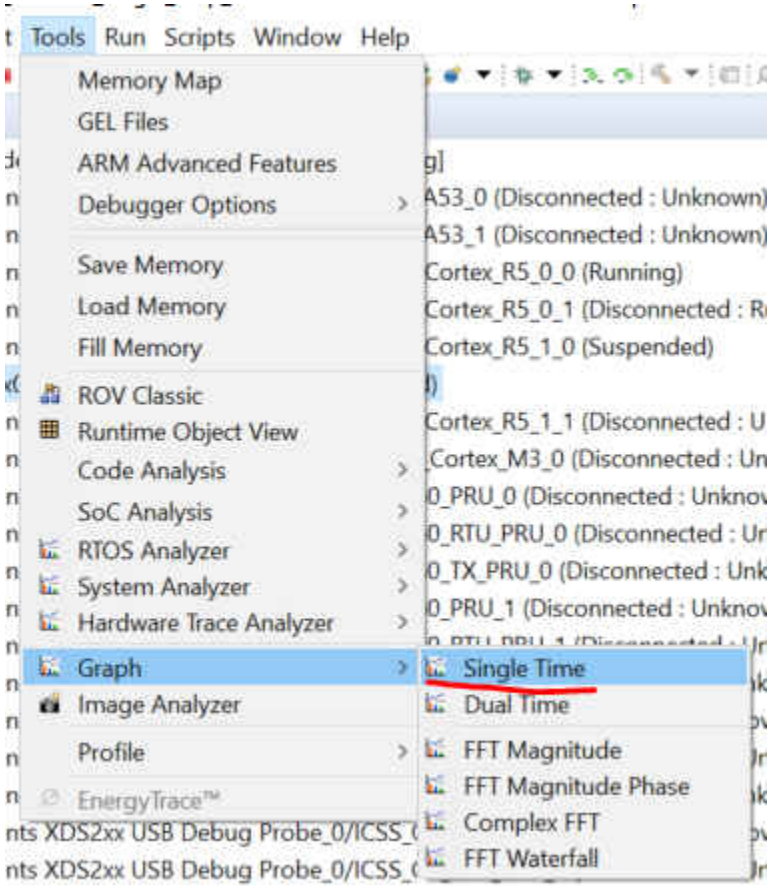

Figure 11-16. Open Loop IQ ID - PWM Graph 1

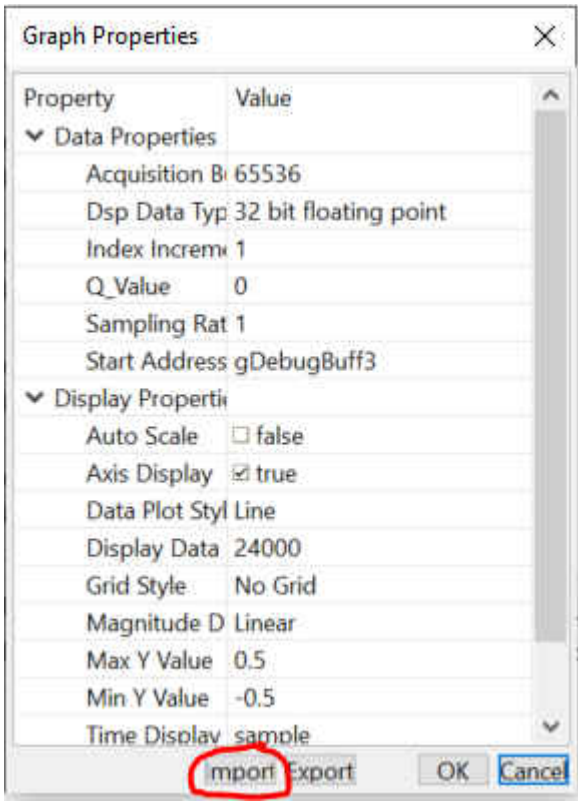

Figure 11-17. Open Loop IQ ID - PWM Graph 2

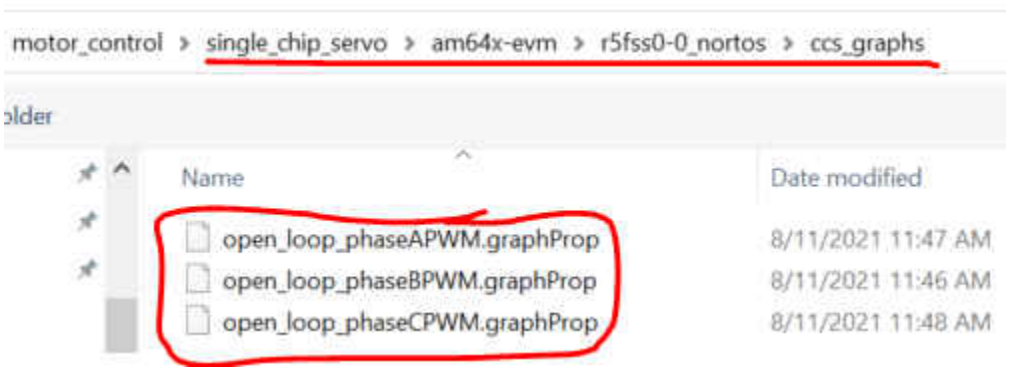

Figure 11-18. Open Loop IQ ID - PWM Graph 3

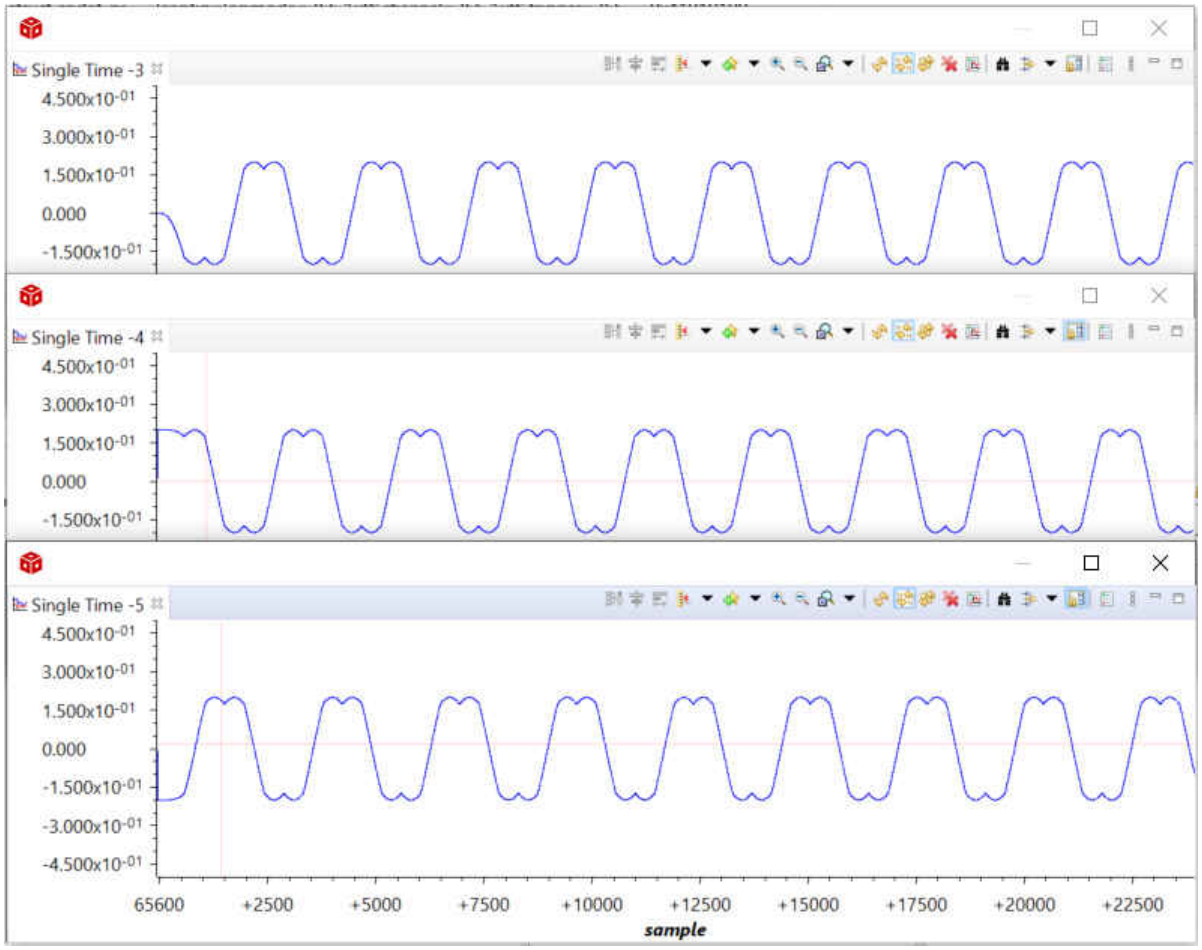

Figure 11-19. Open Loop IQ ID - PWM Graph 4

c. This shows the Space Vector Modulation PWM values being output while in open loop control. You can tell that the three phases are each offset in the graphs above. This data can also be dumped through the CCS memory browser if you would like to manipulate or graph together in Matlab or Excel.
d. This is the end of the Open Loop control demonstration, the Feedback paths and output paths are now validated assuming that your offset values looked good above and your motor was spinning.
e. To stop the motor spinning in open loop control, you can simply switch the EVM off.

11.5 Step 4. Closed Loop Iq/Id Control (BUILDLEVEL == CLOSED_LOOP_IQ_ID)

This build level is the first to close the loop using the Phase Current and Angle/Position feedback.

1. Go to the settings.h file and change the definitions to match the following (closed loop current control, debug buffers on, 8000 cycles per set point):

```
#define  BUILDLEVEL             CLOSED_LOOP_IQ_ID
#define  PID_TUNE_LEVEL         NO_TUNING
#define  DEBUG_LEVEL            DEBUG_BUFFERS_ON
#define  DEBUG_WRAP_TYPE        DEBUG_BUFFER_SINGLE
#define  PRECOMPUTE_LEVEL       NO_PRECOMPUTE

/* The number of ISR cycles to use the same target from
#define  CYCLES_PER_TARGET      8000
#define  TARGET_BUFF_SIZE       CYCLES_PER_TARGET * 8

/* Iq testing value for open loop */
#define  IQ_TESTING             0.2
/* Max speed change request allowed between updates */
#define  MAX_SPD_CHANGE         0.12
/* Max position change request allowed between updates
#define  MAX_POS_CHANGE         0.03        /* 500RPMs
```

Figure 11-20. Closed Loop IQ ID - BUILDLEVEL

2. If desired, go to the single_chip_servo.c file and modify the Iq set point array. Default values shown below.

```
volatile float gIqArray[8] = {0,0.5,1.0,1.5,0,-.5,-1,0};
```

3. Build the project in Debug mode, then load it into MAIN_Cortex_R5_0_0.
4. Once the project is loaded, open the 'single_chip_servo.c' file and find the following line, then right-click and select 'Run to Line'.

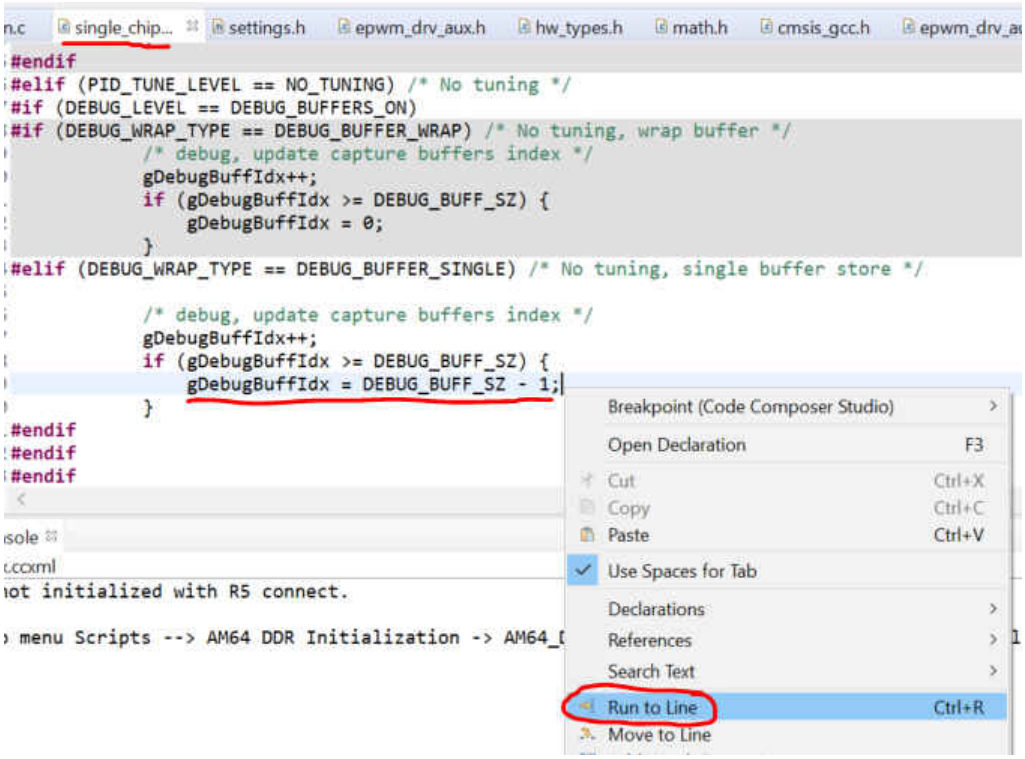

Figure 11-21. Closed Loop IQ ID - Run to Line

5. Running to this line allows the control code to run through the eight different setpoints and fill up the debug buffer so that we can view the graph.

6. Check the output by importing the following graphs:

Figure 11-22. Closed Loop IQ ID - Alpha, Beta, IQ and ID Graph 1

7. Example output shown below (Alpha, Beta, Iq, and Id graphed from top to bottom). Notice how the Iq graph matches the setpoints given above in the setpoint array and that Id is controlled close to 0 as requested:

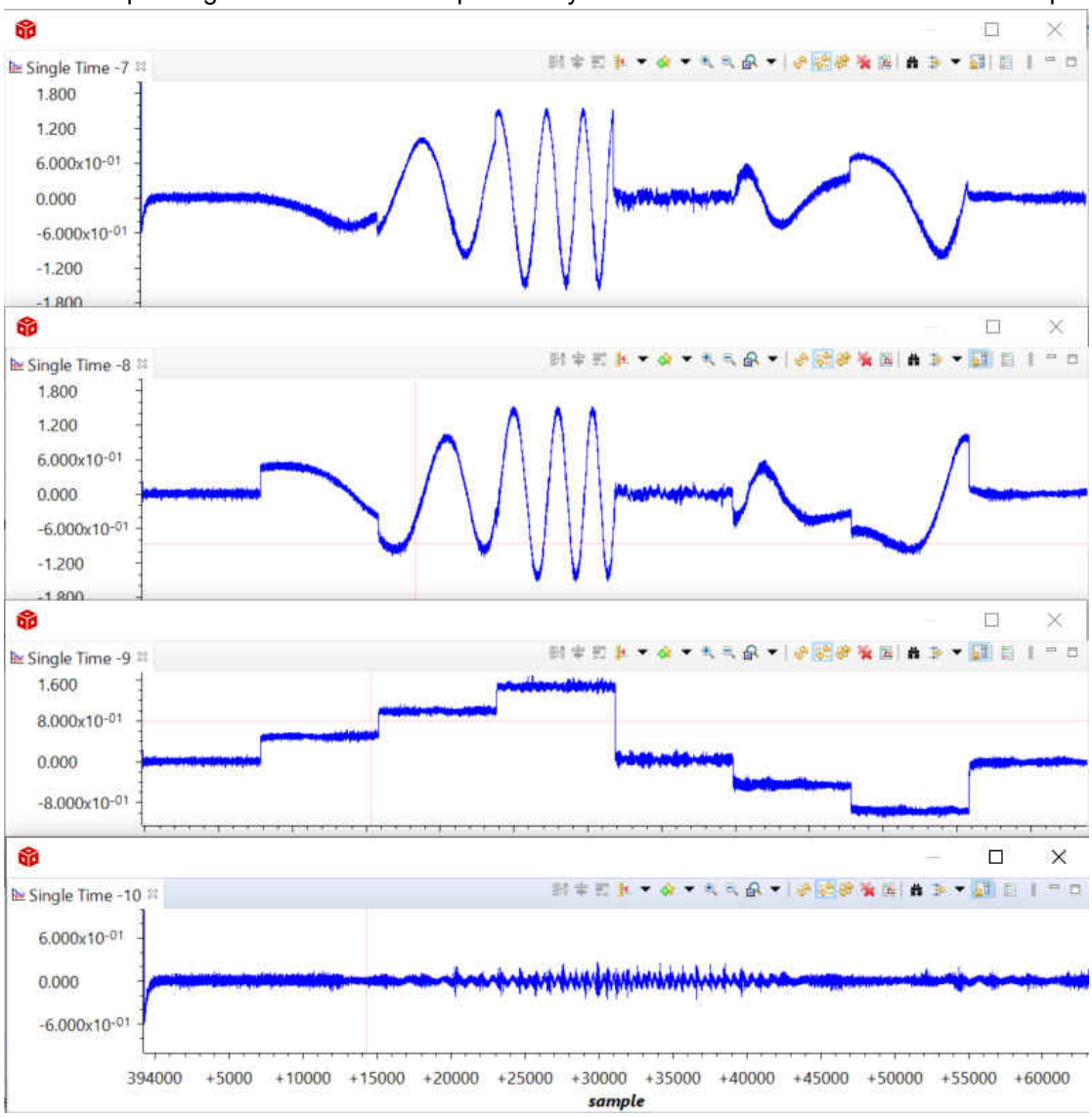

Figure 11-23. Closed Loop IQ ID - Alpha, Beta, IQ and ID Graph 2

11.6 Step 5. Closed Loop Speed Control (BUILDLEVEL == CLOSED_LOOP_SPEED)

This build level closes the speed loop in addition to the current loop.

1. Go to the settings.h file and change the definitions to match the following (closed loop speed control, debug buffers on, 8000 cycles per set point):

```
#define  BUILDLEVEL            CLOSED_LOOP_SPEED
#define  PID_TUNE_LEVEL        NO_TUNING
#define  DEBUG_LEVEL           DEBUG_BUFFERS_ON
#define  DEBUG_WRAP_TYPE       DEBUG_BUFFER_SINGLE
#define  PRECOMPUTE_LEVEL      NO_PRECOMPUTE

/* The number of ISR cycles to use the same target fro
#define  CYCLES_PER_TARGET     8000
#define  TARGET_BUFF_SIZE      CYCLES_PER_TARGET * 8

/* Iq testing value for open loop */
#define  IQ_TESTING            0.2
/* Max speed change request allowed between updates */
#define  MAX_SPD_CHANGE        0.12
/* Max position change request allowed between updates
#define  MAX_POS_CHANGE        0.03        /* 500RPMs
```

Figure 11-24. Closed Loop Speed - BUILDLEVEL

2. If desired, go to the single_chip_servo.c file and modify the Speed set point array. Default values shown below.

```
volatile float gSpdArray[8] = {0,500,750,-500,-750,0,0,0};
```

3. Build the project in Debug mode, then load it into MAIN_Cortex_R5_0_0.
4. Once the project is loaded, open the 'single_chip_servo.c' file and find the following line, then right-click and select 'Run to Line'.

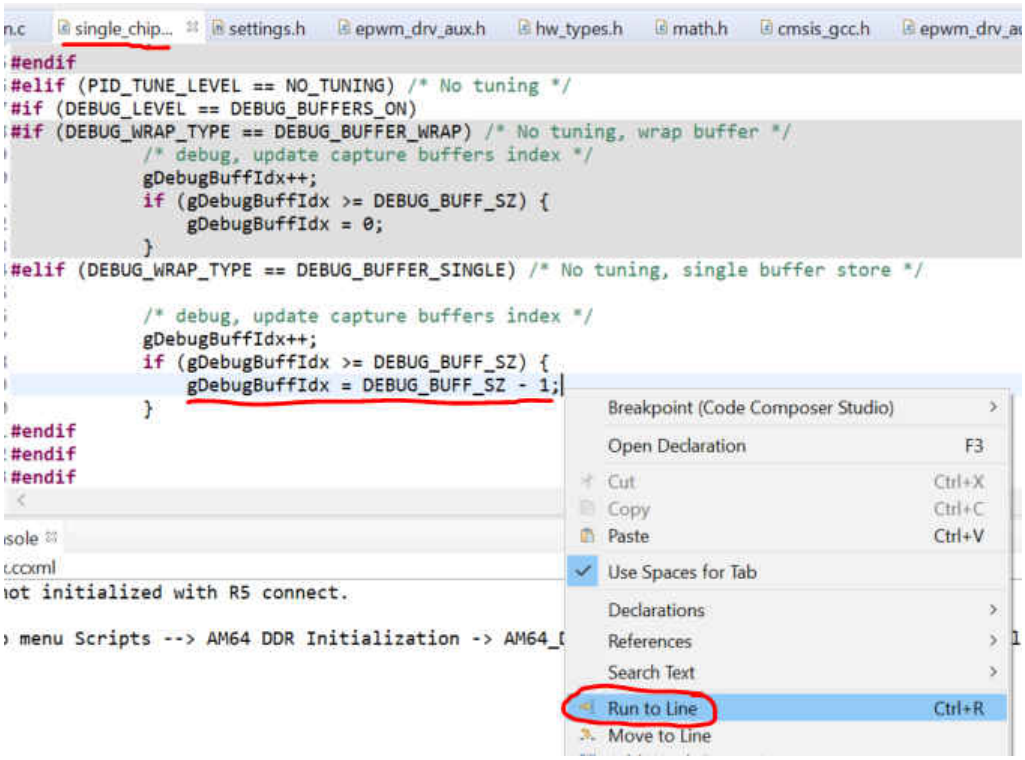

Figure 11-25. Closed Loop Speed - Run to Line

5. Running to this line allows the control code to run through the 8 different setpoints and fill up the debug buffer so that we can view the graph.

6. Check the output by importing the following graphs:

Figure 11-26. Closed Loop Speed - ID, IQ, Speed Graph 1

7. Example output shown below (Iq, Id, and speed graphed from top to bottom). Notice how Iq ramps and stays high during the speed ramps and goes close to zero to maintain speed. The speed graph at the bottom shows that the speed ramps to the requested value and then levels out. The ramp is slope is determined by the MAX_SPD_CHANGE definition in the settings.h file. You'll notice that during the ramp from 750 to -500 and then -750 that -500 was basically skipped because the ramp wasn't steep enough to reach it within the cycles allowed.

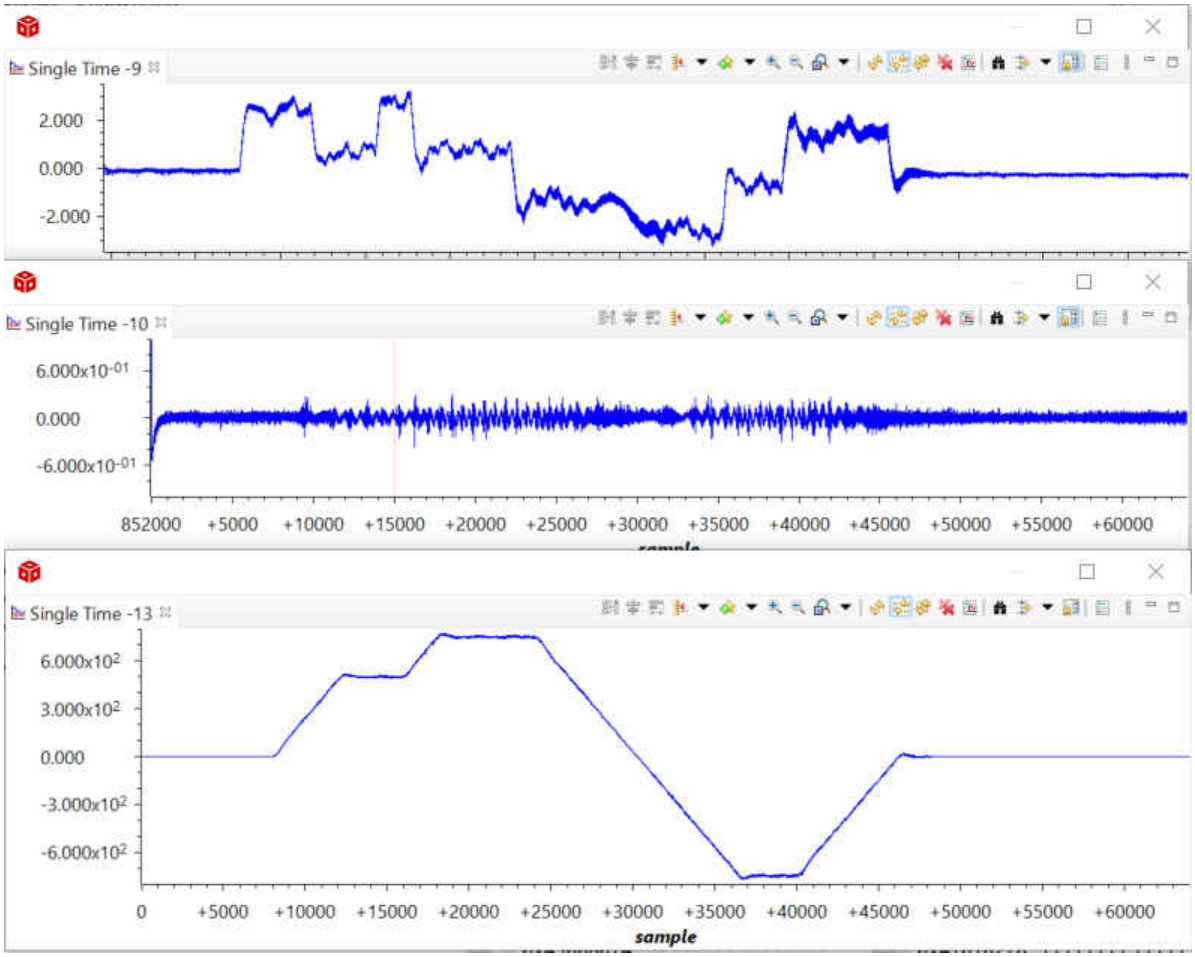

Figure 11-27. Closed Loop Speed - ID, IQ, Speed Graph 2

11.7 Step 6. Closed Loop Position Control (BUILDLEVEL == CLOSED_LOOP_POSITION)

This build level closes the position loop in addition to the speed and current loops.

1. Go to the settings.h file and change the definitions to match the following (closed loop position control, debug buffers on, 8000 cycles per set point):

```
#define  BUILDLEVEL              CLOSED_LOOP_POSITION
#define  PID_TUNE_LEVEL          NO_TUNING
#define  DEBUG_LEVEL             DEBUG_BUFFERS_ON
#define  DEBUG_WRAP_TYPE         DEBUG_BUFFER_SINGLE
#define  PRECOMPUTE_LEVEL        NO_PRECOMPUTE

/* The number of ISR cycles to use the same target fro
#define  CYCLES_PER_TARGET       8000
#define  TARGET_BUFF_SIZE        CYCLES_PER_TARGET * 8

/* Iq testing value for open loop */
#define  IQ_TESTING              0.2
/* Max speed change request allowed between updates */
#define  MAX_SPD_CHANGE          0.12
/* Max position change request allowed between updates
#define  MAX_POS_CHANGE          0.03        /* 500RPMs
```

Figure 11-28. Closed Loop Position - BUILDLEVEL

2. If desired, go to the single_chip_servo.c file and modify the Speed set point array. Default values shown below.

```
volatile float gPosArray[8] = {180,180,359.5,359.5,0.5,0.5,180,180};
```

   a. Values are duplicated to give the motor time to get to the position before the setpoint cycle count is up.

3. Once the project is loaded, open the 'single_chip_servo.c' file and find the following line, then right-click and select 'Run to Line'.

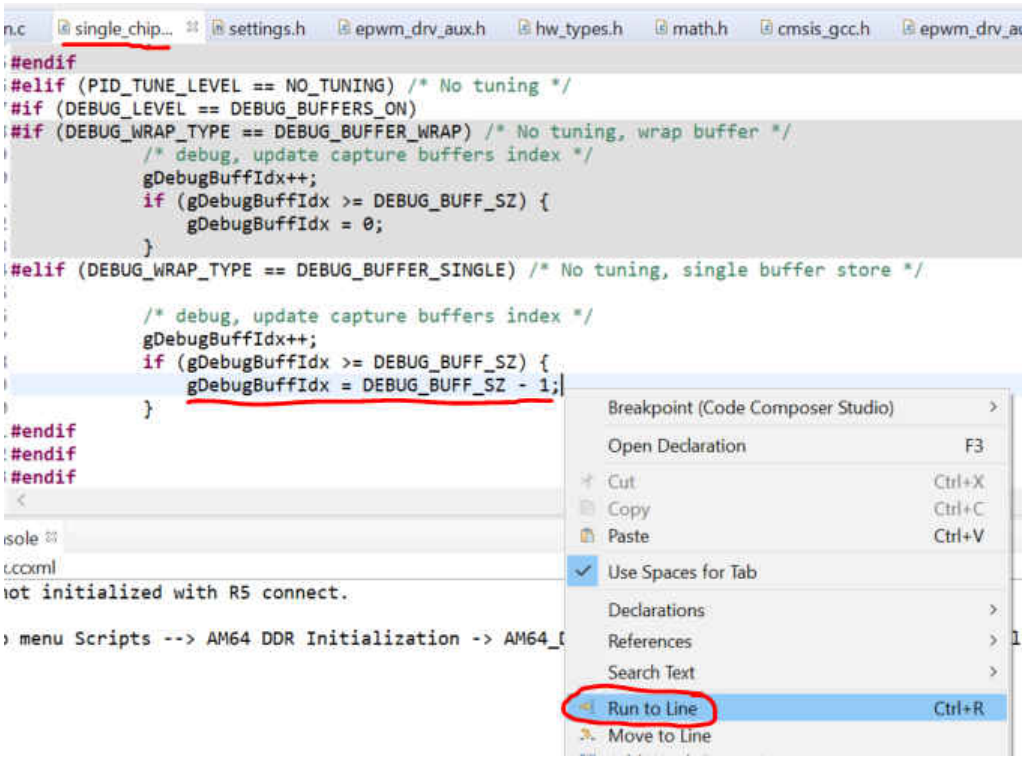

Figure 11-29. Closed Loop Position - Run to Line

4. Running to this line allows the control code to run through the eight different setpoints and fill up the debug buffer so that the grap can be viewed.

5. Check the output by importing the following graphs:

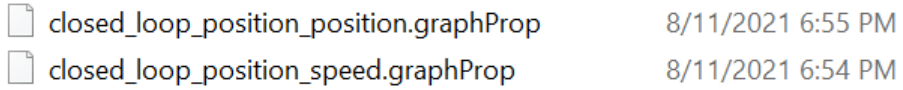

Figure 11-30. Closed Loop Position - Position and Speed Graph 1

6. Example output shown below (speed and position graphed from top to bottom). Notice how speed goes to +-500RPMs during the position changes and stays there until the position is reached. The position ramp slope is determined by the MAX_POS_CHANGE definition in the settings.h file.

Figure 11-31. Closed Loop Position - Position and Speed Graph 2

7. It is a difficult to see how accurate the position is in the CCS graph above, so you can also use the CCS memory browser to dump 'gDebugBuff5' to a text file in order review in Excel or Matlab. Once you do that, you can see how accurate the position actually is: typically +-0.001° from the requested target once the position has settled (shown in Figure 11-32 for two of the requested positions):

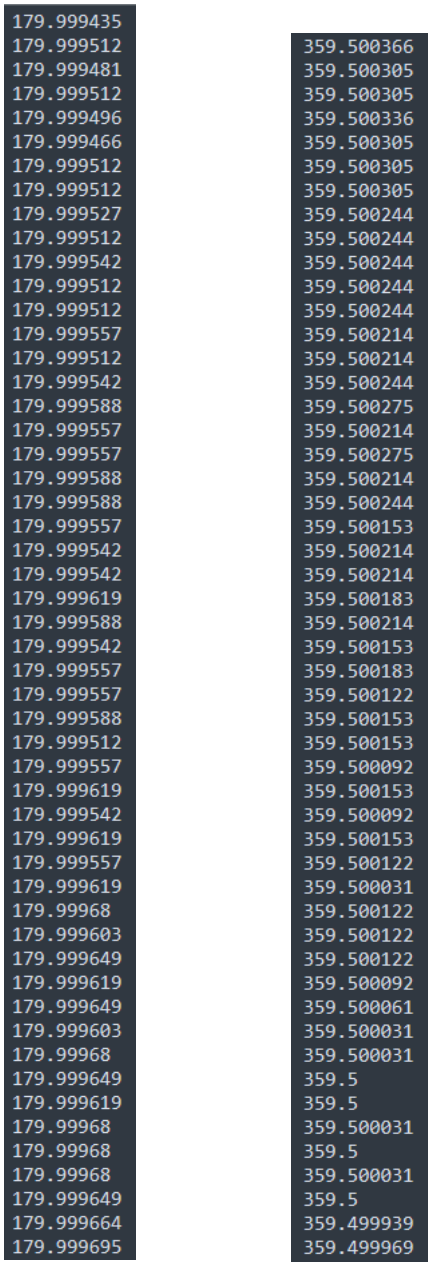

Figure 11-32. Closed Loop Position - Position Control Accuracy

12 Build Using MCU+SDK 08.00.00.21 & CCS 10.3.1

1. Downlaod and install MCU+SDK 08.00.00.21 for AM64x at c:\ti.
2. Download () and unzip it at <DEMO_INSTALL ROOT>.
3. Copy and Overwrite:
   a. Copy <DEMO_INSTALL ROOT>\mcu_plus_sdk_servo_drive_code\examples\motor_control\sddf\ to C:\ti\mcu_plus_sdk_am64x_08_00_00_21\examples\motor_control
   b. copy <DEMO_INSTALL ROOT>\mcu_plus_sdk_servo_drive_code\examples\motor_control\single_chip_servo\ to C:\ti\mcu_plus_sdk_am64x_08_00_00_21\examples\motor_control
   c. copy <DEMO_INSTALL ROOT>\mcu_plus_sdk_servo_drive_code\source\motor_control\ to C:\ti\mcu_plus_sdk_am64x_08_00_00_21\ource\motor_control

4. Re-build the EnDat Library:
   a. cd C:\ti\mcu_plus_sdk_am64x_08_00_00_21
   b. gmake -s -f makefile.am64x motorcontrol_endat_r5f.ti-arm-clang_clean
   c. gmake -s -f makefile.am64x motorcontrol_endat_r5f.ti-arm-clang
   d. gmake -s -f makefile.am64x motorcontrol_endat_r5f.ti-arm-clang_clean PROFILE=debug
   e. gmake -s -f makefile.am64x motorcontrol_endat_r5f.ti-arm-clang PROFILE=debug
5. Import and Build CCS projects:
   a. Import and build the single_chip_servo_am64x-evm_system_freertos_nortos from the folder: C:\ti\mcu_plus_sdk_am64x_08_00_00_21\examples\motor_control\single_chip_servo\am64x-evm.
   b. The system CCS project will import three sub CCS projects:
      i. single_chip_servo_am64x-evm_r5fss0-0_nortos_ti-arm-clang
      ii. single_chip_servo_am64x-evm_r5fss1-0_freertos_ti-arm-clang
      iii. single_chip_servo_am64x-evm_m4fss0-0_nortos_ti-arm-clang
   c. You can build the system CCS project. It will build all three sub CCS projects.
6. Load and Run single_chip_servo_am64x-evm_r5fss0-0_nortos_ti-arm-clang:
   a. Load and Run single_chip_servo_am64x-evm_r5fss0-0_nortos_ti-arm-clang in R5F_0_0 using CCS and JTAG on AM64x/AM243x EVM.
   b. The motor should start spin at 6000 cycles/min.

13 Summary

New guidelines and system requirements for servo drives are being introduced with Industry 4.0, making it important for designers to select a solution that fits the needs of current and future servo drives. Precision, connectivity, efficiency, and safety are the key considerations. Devices like Sitara AM243x MCU, which includes up to 4x R5F @ 800 MHz, industrial communication subsystem and real-time control peripherals, can be a good fit for the connected, high-end motor drive system. This app note provides a step-by-step guidance to demonstrate AM243x in the servo drive application, and help users understand the details of the implementation and performance. Key features are summarized as below:

- Flexibility to support multiple popular industrial communication protocols
- Unique synchronization architecture of the AM243x/AM64x for easy implementation of synchronization among system events
- A dedicated M4F core on AM243x/AM64x for functional safety
- Use dedicated 800 Mhz R5F core to achieve much higher complete motor control loop frequency (50Khz) in comparison to the usually 8K to 20 Khz control loop frequency, Thanks to the low latency, more deterministic, more reliable nature of the Arm® R5F core.
- Use the highly efficient MCU+ SDK, take advantage of the single-cycle TCM and large on-chip RAM to build highly efficient and DDRless motor control system.
- Use dedicated 800 Mhz R5F core to handle the Industrial Communication Stack to improve the reliability and capable to handle gigabit Profinet.
- Easy to extend to multiple axis motor control – Easy to add functional safety. Thanks to the dedicated M4F core for implementing functional safety.
- The FOC loop time reduced from 6us to <1us.
- The Sigma Delta Filtering on PRU_ICSSG can handle up to 9 channels (enough for 3 axis)
- The EnDat 2.2 decoding on PRU_ICSSG can handle up to three channels (enough for 3 axis)
- AM243x consume less 1W

14 Appendix A: Detailed Motor Control R5F Processing Time

Here are the detailed motor control R5F processing time:

Table 14-1. Angle/Position/Speed Calculation Time

	Angle/Position/Speed Calculations					
	Open Loop Iq/Id	Closed Loop Iq/Id	Closed Loop Speed	Closed Loop Position	Closed Loop CiA402	Unit
Min	252	252	268	268		ns
Max	312	308	336	336		ns
Mean	274.462	269.271	291.303	295.503		ns
Max normalized	132	128	156	156		ns

Table 14-2. Phase Current Scaling and Conversion Time

	Phase Current Scaling and Conversion to Floating Point					
	Open Loop Iq/Id	Closed Loop Iq/Id	Closed Loop Speed	Closed Loop Position	Closed Loop CiA402	Unit
Min	188	316	460	380		ns
Max	200	360	496	416		ns
Mean	185.139	328.791	474.26	394.462		ns
Max normalized	20	180	316	236		ns

Table 14-3. FOC Loop Time

	FOC (Clarke, Sin/Cos, Park, PI controllers, Inverse Park, Space Vector Generation					
	Open Loop Iq/Id	Closed Loop Iq/Id	Closed Loop Speed	Closed Loop Position	Closed Loop CiA402	Unit
Min	308	380	444	468		ns
Max	360	432	520	584		ns
Mean	331.301	398.24	476.313	494.554		ns
Max normalized	180	252	340	404		ns

Table 14-4. PWM Output Time

	Write to PWM Output	Unit
Max normalized	84	ns
Min	244	ns
Mean	252.666	ns
Max	264	ns

Table 14-5. Full ISR Measured At Once

	Full ISR Measured at Once					
	Open Loop Iq/Id	Closed Loop Iq/Id	Closed Loop Speed	Closed Loop Position	Closed Loop CiA402	Unit
Min	516	700	932	892	854	ns
Max	568	824	1064	1072	1040	ns
Mean	535.97	741.272	983.271	947.688	888.707	ns
Max normalized	388	644	884	892	860	ns

Table 14-6. Full ISR Measured At Once (Clarke Precomputed)

	Full ISR Measured at Once (Clarke Precomputed)					
	Open Loop Iq/Id	Closed Loop Iq/Id	Closed Loop Speed	Closed Loop Position	Closed Loop CiA402	Unit
Min	516	572	820	820		ns
Max	576	664	920	952		ns
Mean	543.08	599.084	851.557	874.515		ns
Max normalized	396	484	740	772		ns

15 References

- Texas Instruments: *Highly integrated industrial drive to connect, control and communicate*
- Texas Instruments: *Distributed Multi-axis Servo Drive Over Fast Serial Interface (FSI) Reference Design*
- AM2434 product folder page
- AM6442 product folder page
- Texas Instruments: *48V/500W Three-phase Inverter with Smart Gate Driver Reference Design for Servo Drives*
- Texas Instruments: Universal Digital Interface to Absolute Position Encoders Reference Design
- Texas Instruments: Utilizing Sitara Processors and Microcontrollers for Industry 4.0 Servo Drives
- Texas Instruments: *Industrial Communication Protocols Supported on Sitara™ Processors*
- Connect and control the factory floor, AM64x/AM243x multi-protocol demo video: (https://training.ti.com/connect-and-control-factory-floor)

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265