

Design Guide: TIDM-02006

Distributed Multi-axis Servo Drive Over Fast Serial Interface (FSI) Reference Design



Description

This reference design shows an example of distributed multi-axis servo drive over Fast Serial Interface (FSI) using C2000 real-time controllers. Multi-axis servo drives are used in many applications such as factory automation and robots. The cost per axis, performance, and ease of use are always high concerns for such systems. FSI is a cost-optimized and reliable high speed communication interface with low jitter that can daisy-chain multiple C2000 devices. In this design, each F28004x device serves as a real-time controller for a slave axis, running motor current control loop. A single F2838x device runs position and speed control loops for all axes. The same F2838x also executes main axis motor control plus EtherCAT communication, leveraging its multiple cores. The design uses TI's existing EVM kits, the software is released within C2000WARE MotorControl SDK.

Resources

TIDM-02006	Design Folder
LAUNCHXL-F280049C	Tool Folder
TMDXCNC28388D	Tool Folder
BOOSTXL-3PHGANINV	Tool Folder
TMDXIDDK379D	Tool Folder
C2000WARE-MOTORCONTROL-SDK	Tool Folder



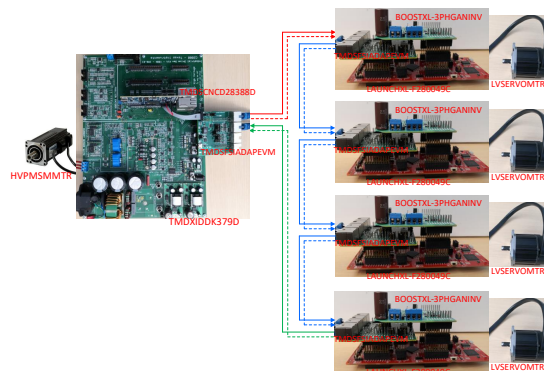
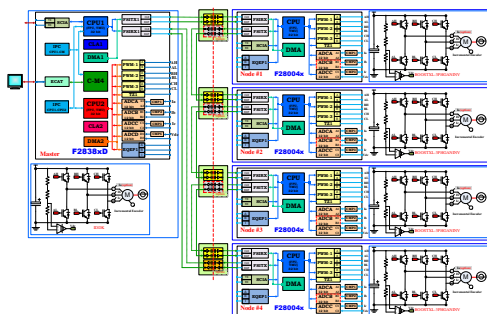
[Search Our E2E™ support forums](#)

Features

- Demonstrates high-speed communication using fast serial interface (FSI) for real-time speed, position and current command data communication between multiple devices.
- Implements position and velocity control loops for all slave axes on F2838x based master node, torque/current loop on each F28004x based slave node.
- Up to 16 axes can be managed with one F2838x. Start and stop of each axis can be controlled simultaneously. Achieves high bandwidth and higher precision control of distributed multi-axis motor drive system.
- Uses F2838x multi-core functionality to implement speed and position control and data exchange with all slave nodes over FSI, and executes main axis motor control and EtherCAT communication with PC.
- Integrated SFRA tool on both F28004x and F2838x to support online speed and current loop tuning.
- Incremental system builds to verify the different software modules step by step.

Applications

- [Industrial motor drives](#)
- [Factory automation and control](#)
- [AC drive control module](#)





An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 System Description

Motion control is a fundamental concern in many industrial applications, especially, multi-axis motion control systems which are found in many applications such as factory automation and robots. The evaluation criteria for multi-axis control systems includes control performance, connectivity, ease of use and cost. Modern multi-axis systems require high resolution and bandwidth within each motor controller as well as very fast communication between master and slaves. Servo inverters in multi-axis applications are often linked by Ethernet based field bus systems and controlled by a primary controller, but Ethernet is a high cost and complex communication peripheral. C2000's Fast Serial Interface (FSI) is implemented in this design to ensure a real-time data communication link between the host and slave devices.

1.1 Key System Specifications

Table 1 lists the system specification of the distributed multi-axis servo drive over fast serial interface (FSI) reference design.

Table 1. Key System Specifications

PARAMETER	SPECIFICATIONS	DETAILS
Input power supply voltage	BOOSTXL-3PHGANINV : 60VDC or less	Nominal is 48V, default is 24V for this design
	TMDXIDDK379D: 400VDC/220VAC or less	Default is 300VDC for this design
DC input current	BOOSTXL-3PHGANINV : 10A or less each board	Not intended for long run times with full current output
	TMDXIDDK379D: 10A or less	
Connected motor capacity	BOOSTXL-3PHGANINV : 400W or less each board	48V DC bus input voltage
	TMDXIDDK379D: 1000W or less	300VDC/220VAC bus input voltage
Maximum three-phase output current	BOOSTXL-3PHGANINV : 10A(rms) or less	Not intended for long run times with full current
	TMDXIDDK379D: 10A(rms) or less	
Switching PWM frequency	BOOSTXL-3PHGANINV : 40 kHz or less, default is 10kHz	CPU MIPS limits the maximum switching frequency
	TMDXIDDK379D: 20 kHz or less, default is 10kHz	Power module limits the maximum switching frequency
Maximum modulation index at 10kHz	BOOSTXL-3PHGANINV : 96%	Single sampling
	TMDXIDDK379D: 98%	Single sampling
PWM update latency	BOOSTXL-3PHGANINV : <2.06us	FCL code executes with F28004x CPU
	TMDXIDDK379D: <1.03us	FCL code executes with F2838x CPU
FOC execution time	BOOSTXL-3PHGANINV : <7.8us	FOC code executes with F28004x CPU
	TMDXIDDK379D: <4.12us	FOC code executes with F2838x CPU
Speed setting	0.1%~100%	Below 1% need to change the gain of speed controller.
Position detection method	Incremental encoder	Connect to QEP interface
Drive method	FOC with FCL	sensored-FOC based encoder
FSI speed	100Mbps @ 50MHz Clock	LVDS interface
Communication topology	Daisy chain (16 nodes or less)	1 master, 15 slaves
Communication cycle times	Free run mode, less than 1 ms every cycle	4 slave nodes, daisy-chain topology network
Working temperature range	-40°C to 85°C	Industrial temperature range -40°C to 85°C.

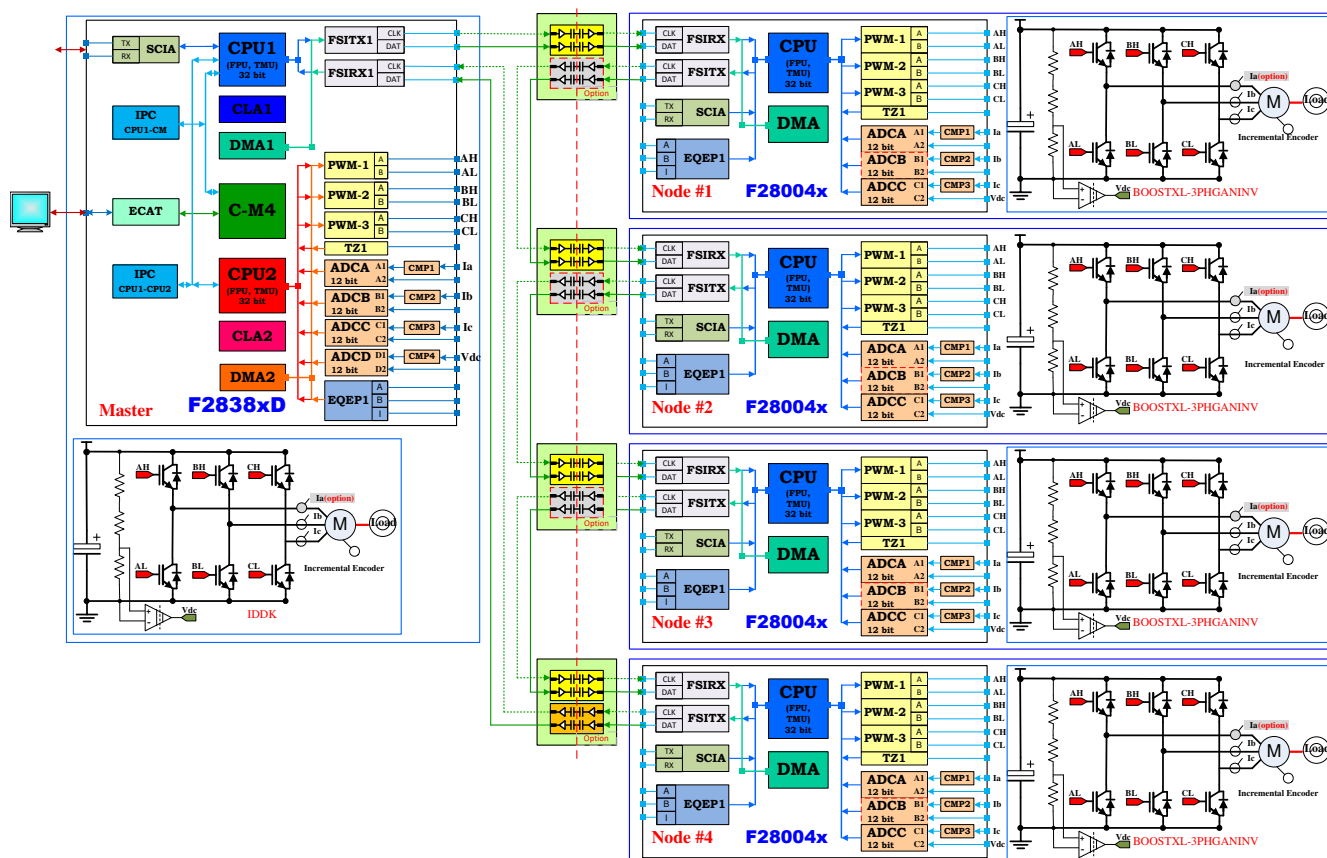
2 System Overview

2.1 Block Diagram

Figure 1 shows the system block diagram of TIDM-02006 reference design, which includes the following elements:

- Master node consists of one controller, one high voltage inverter, and one PM motor with incremental encoder. TMS320F28388D functions as the controller, which samples all the voltage and current inputs and generates the correct PWM signals for inverter. The controller also exchanges data as lead with slave controllers via FSI, and communication with EtherCAT master on PC.
- Four slave nodes, each node consists of one controller, one low voltage inverter, and one PM motor with incremental encoder. TMS320F280049C functions as the controller, which has all the voltage and current sensing inputs and generates the correct PWM signals for inverter. The controller also receives and transmits the data with master controller via FSI.

Figure 1. TIDM-02006 System Block Diagram



2.2 Highlighted Products

This distributed multi-axis servo drive system is based on a F2838x controller and multiple F28004x controllers. The F2838x device serves as the master and each F28004x serves as a slave node. The Fast Serial Interface (FSI) is used as the communication bus between master node and slave nodes.

2.2.1 TMS320F28004x

TMS320F28004x is a powerful 32-bit floating-point microcontroller unit (MCU) that lets designers incorporate crucial control peripherals, differentiated analog, and nonvolatile memory on a single device. The real-time control subsystem is based on TI's 32-bit C28x CPU, which provides 100 MHz of signal processing performance. The C28x CPU is further boosted by the new TMU extended instruction set, which enables fast execution of algorithms with trigonometric operations commonly found in transforms and torque loop calculations. The CLA allows significant offloading of common tasks from the main C28xCPU. The CLA is an independent 32-bit floating-point math accelerator that executes in parallel with the CPU. Additionally, the CLA has its own dedicated memory resources and it can directly access the key peripherals that are required in a typical control system. High-performance analog blocks are integrated on the F28004x MCU to further enable system consolidation. Three separate 12-bit ADCs provide precise and efficient management of multiple analog signals, which ultimately boosts system throughput. Seven PGAs on the analog front end enable on-chip voltage scaling before conversion. Seven analog comparator modules provide continuous monitoring of input voltage levels for trip conditions. The TMS320C2000™ devices contain industry-leading control peripherals with frequency independent PWM/HRPWM and eCAP allow for a best-in-class level of control to the system. A specially enabled device variant, TMS320F28004xC, allows access to the Configurable Logic Block (CLB) for additional interfacing features.

2.2.2 TMS320F2838x

The TMS320F2838x is a powerful 32-bit floating-point microcontroller unit (MCU) designed for advanced closed-loop control applications. The F2838x supports a dual-core C28x architecture along with a new Connectivity Manager that offloads critical communication tasks, significantly boosting system performance. The integrated analog and control peripherals with advanced connectivity peripherals like EtherCAT and Ethernet also let designers consolidate real-time control and real-time communications architectures, reducing requirements for multi controller systems. The dual real-time control subsystems are based on TI's 32-bit C28x floating-point CPUs, which provide 200 MHz of signal processing performance in each core. The C28x CPUs are further boosted by the TMU accelerator, which enables fast execution of algorithms with trigonometric operations common in transforms and torque loop calculations. The F2838x microcontroller family features two CLA real-time control coprocessors. The CLA is an independent 32-bit floating-point processor that runs at the same speed as the main CPU. The CLA responds to peripheral triggers and executes code concurrently with the main C28x CPU. This parallel processing capability can effectively double the computational performance of a real-time control system. By using the CLA to service time-critical functions, the main C28x CPU is free to perform other tasks, such as communications and diagnostics. The dual C28x+CLA architecture enables intelligent partitioning between various system tasks. For example, one C28x+CLA core can be used to track speed and position, while the other C28x+CLA core can be used to control torque and current loops. Performance analog and control peripherals are also integrated on the F2838x MCU to further enable system consolidation. Four independent 16-bit ADCs provide precise and efficient management of multiple analog signals, which ultimately boosts system throughput. The sigma-delta filter module (SDFM) works in conjunction with the sigma-delta modulator to enable isolated current shunt measurements. The Comparator Subsystem (CMPSS) with windowed comparators allows for protection of power stages when current limit conditions are exceeded or not met. Other analog and control peripherals include DACs, PWMs, eCAPs, eQEPs, and other peripherals. Peripherals such as EMIFs, CAN modules, EtherCAT, Ethernet, and MCAN (CAN-FD) extend the connectivity of the F2838x. Fast Serial Interface (FSI) with two transmitters and eight receivers is a serial communication peripheral capable of reliable high-speed communication across isolation devices.

2.3 System Design Theory

2.3.1 FOC and Control Loops in Servo Drive

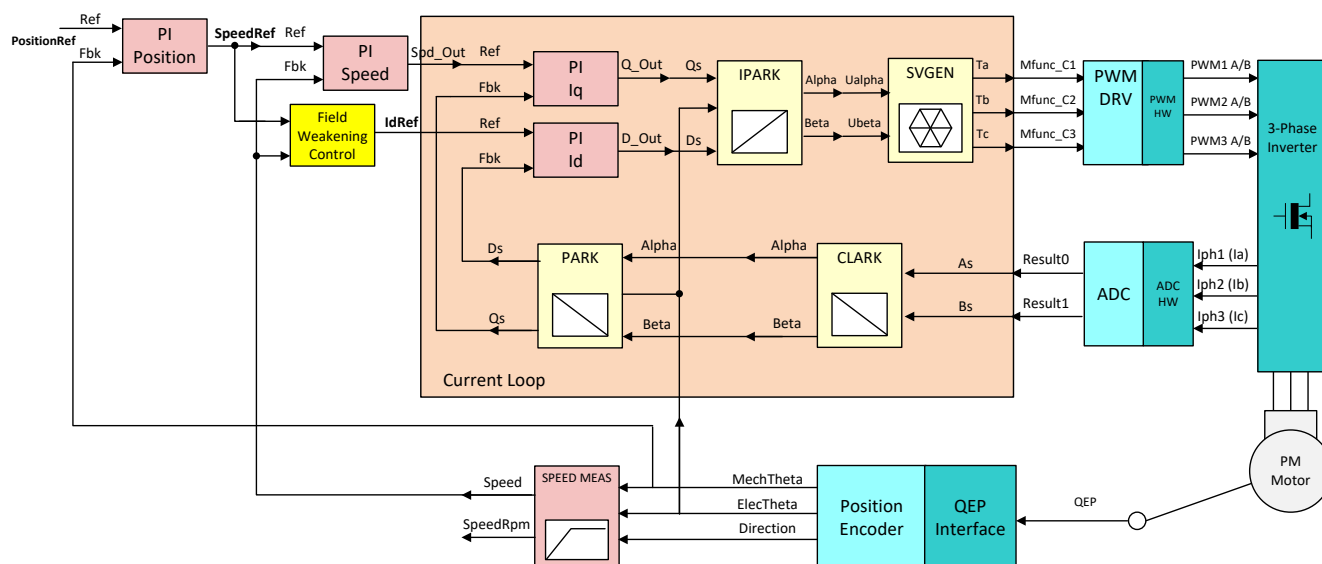
A servo drive system for a single motor axis generally consists of controller, inverter, motor and load, and feedback device. There are three closed loop control systems that are at work within the servo drive system: these are the position loop, speed loop and current loop as shown in [Figure 2](#).

The position loop compares position reference with the position feedback to determine the position error. This is used to generate a speed demand, also known as a speed reference, to regulate the position.

The speed loop compares the speed demand from the position loop to the speed feedback from the motor, and generates a torque reference.

The current loop is designed to regulate the torque producing output current to the motor. It takes the torque demand from the speed loop and current feedback signals from current sensors within inverter to generate the control signals to produce a power output from the power inverter module. In the current loop, any two of the motor phase currents are measured, while the third can be estimated from these two sensing currents. A minimal current loop time not only helps to improve the control bandwidth, but it also enables a higher modulation index for the inverter. A higher modulation index translates into the higher phase voltage that the inverter can apply on the motor. Higher current loop latency will reduce the maximum available voltage and restrict the rate of current change in the motor, thereby, adversely impacting the controller performance. To overcome these challenges, fast current loop (FCL) technology is used in this design for torque control loop.

Figure 2. Servo Drive Control System Block Diagram



2.3.2 Daisy-chain Communication Network Using FSI

The Fast Serial Interface (FSI) module is a serial communication peripheral capable of reliable high-speed communication across isolation devices. The FSI module consists of independent transmitter (FSITX) and receiver (FSIRX) cores. The FSITX and FSIRX cores are configured and operated independently. For more information on the FSI module on a specific C2000 devices, see the device-specific data sheet and technical reference manual.

There are a number of different types of communication network topology for connecting multiple devices via FSI, including point to point, bus, star, and ring. The ring topology is used in this design to connect multiple devices with FSI communication in a daisy-chain fashion as shown in [Figure 3](#).

Figure 3. Daisy-chain Connection Topology

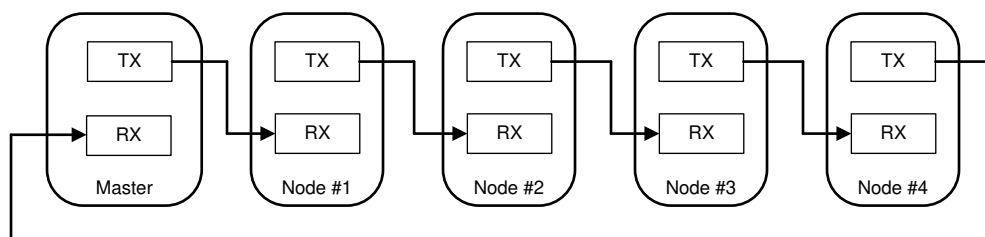
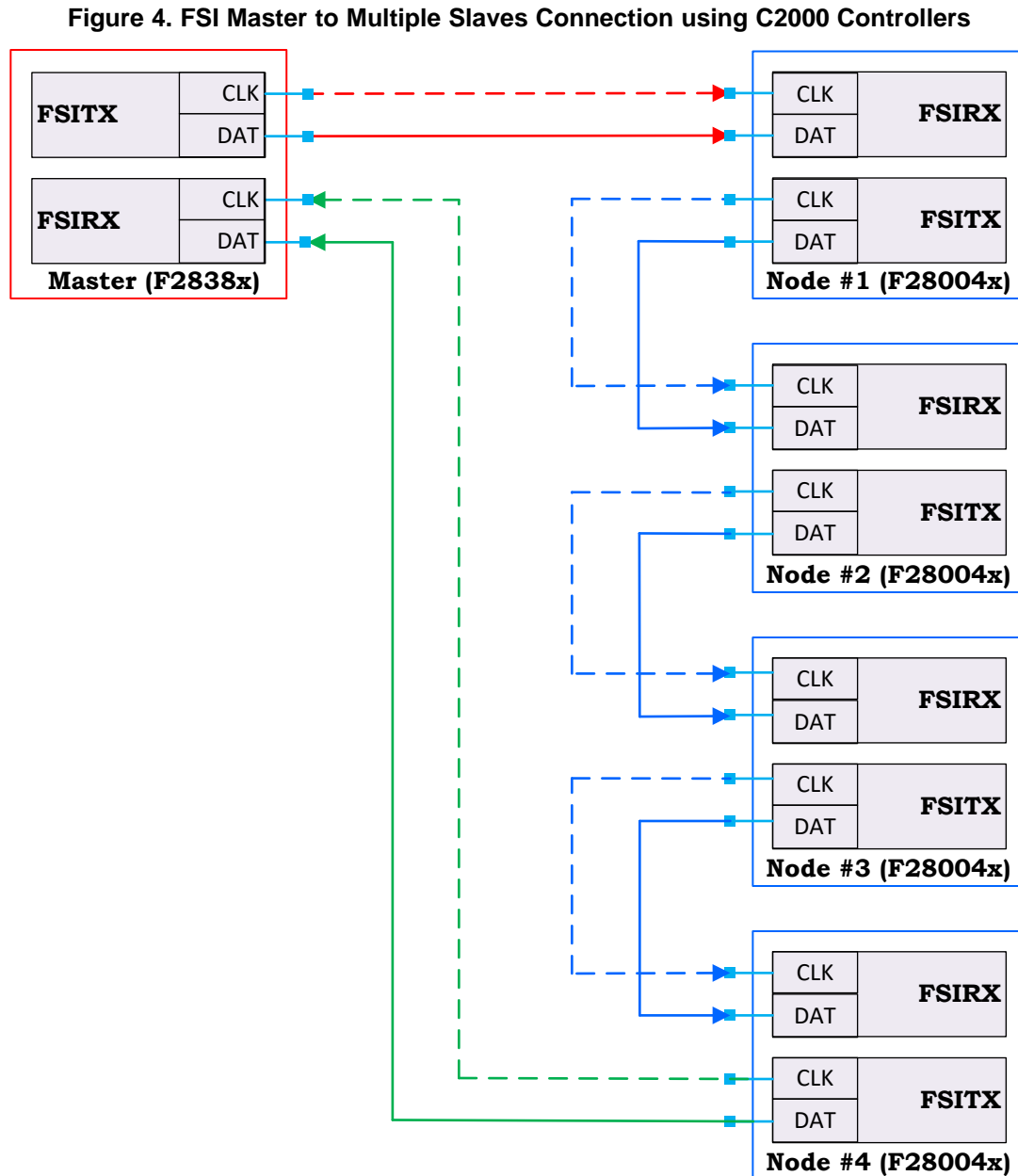


Figure 4 shows the implementation of the communication connect multiple C2000 devices in this design. The FSI module is operated using only one data line (RXD0 or TXD0). In this mode the RXCLK and RXD0 act as clock and data for receiver while RXD1 is unused, and the TXCLK and TXD0 act as clock and data for transmitter while TXD1 is unused. F2838x is the master node which will be the driver of the initialization sequence. The F28004x based nodes is the slave node, it will respond to the master's command.



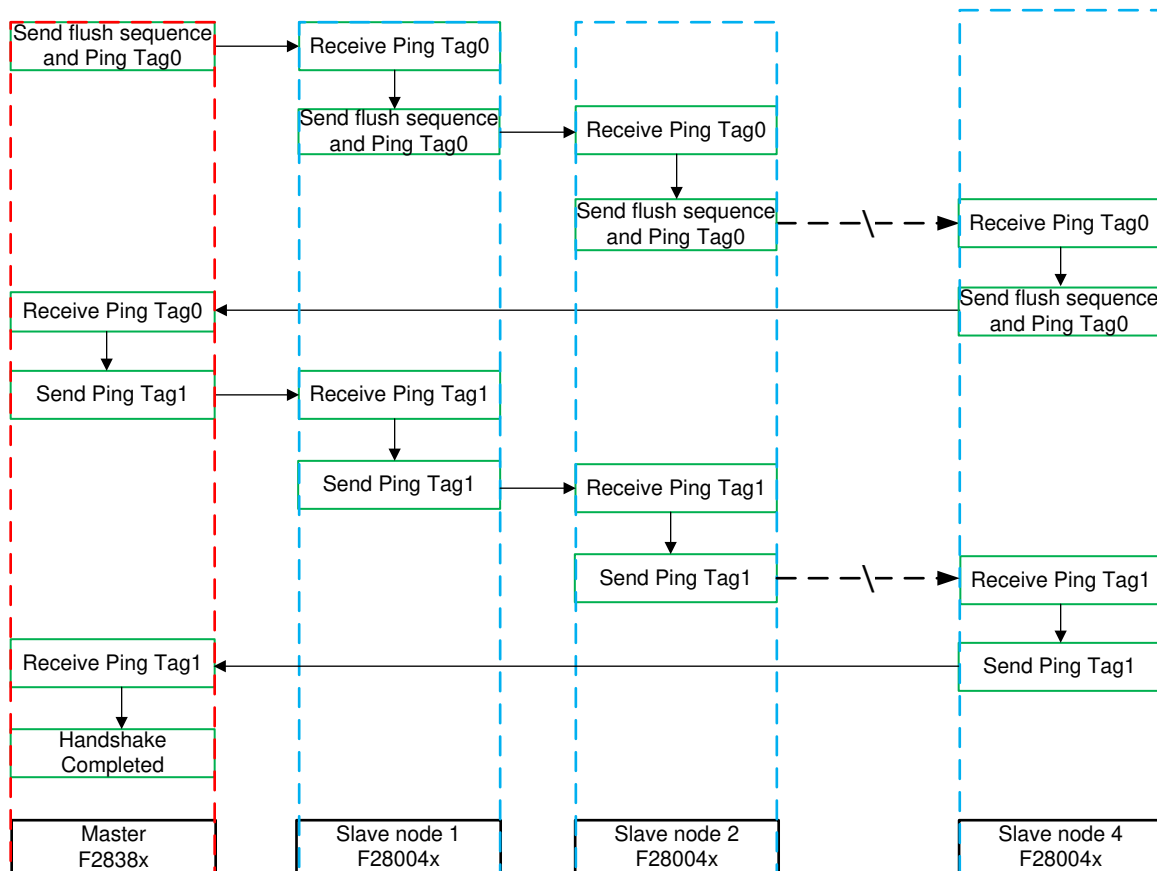
Use the following steps to establish FSI communication link for daisy-chain connection:

1. Configure the transmitter and receiver modules for synchronization.
2. Using the handshaking process to implement synchronization to ensure that the receiver is properly flushed and ready to receive a complete frame without error before the modules may exchange data. Figure 5 shows the data flow for handshake. There are two ping loops for the daisy-chain connection handshake.
 - a. Ping loop 0 has the purpose of establishing the communication path along the chain of devices and ping loop 1 acts as the acknowledgment to the nodes that the communication path is good. In ping loop 0, the node devices wait for receiving a Ping Tag0 from the previous device in the chain.

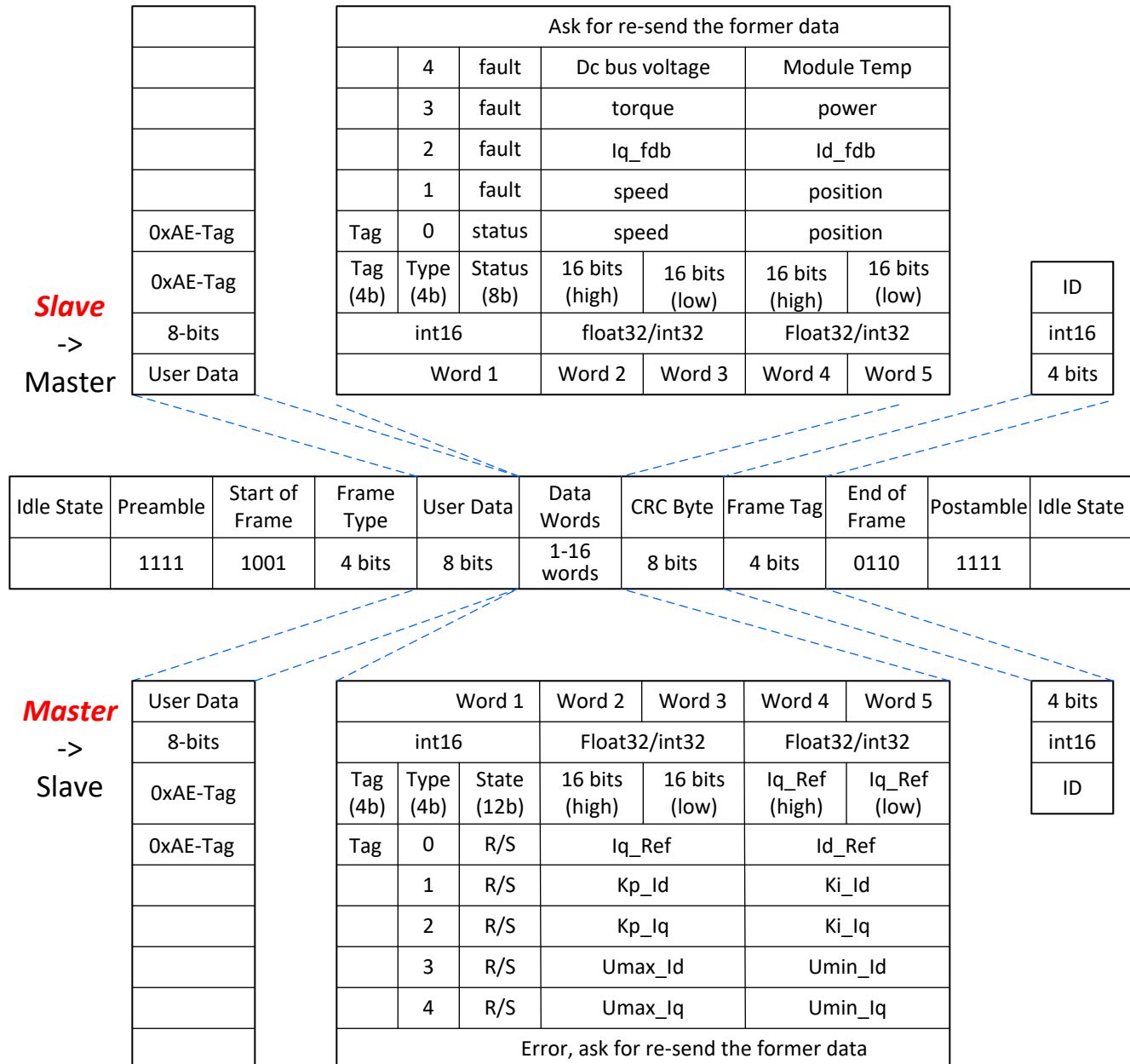
Once a Ping Tag0 is successfully received, it will be forwarded on to the next device in the chain. The ping loop 0 will fail if a device in the chain has not powered up or is not ready for the reception.

- b. Once ping loop 0 has succeeded, in which ping tag0 has made its way back to the lead device, ping loop 1 is initiated to inform the node devices that the handshake sequence has completed. At this point, both the transmit and receive modules have successfully received ping frames from their master counterparts. The link has been established and regular communication may now proceed.
3. Configure FSI for exchanging the regular data. The desired application configuration includes the frame type, frame tag, data words, number of data line.

Figure 5. FSI Handshake Data Flow



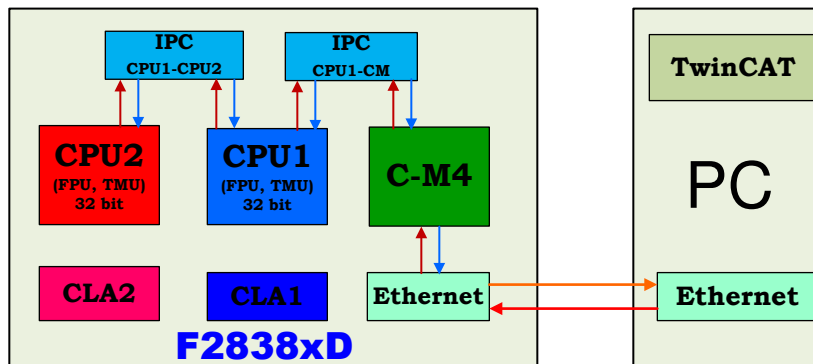
In this design, position and velocity control loops are implemented for all axes on the F2838x based master node, while the torque loop utilizing the fast current loop (FCL) technique is implemented on the F28004x based slave nodes for each individual axis. The running command or status, speed and position data are exchanged between master and slave nodes. [Figure 6](#) shows a simplified FSI communication protocol used in this design.

Figure 6. FSI Data Communication Protocol


2.3.3 EtherCAT Communication

The F2838x device offers enhanced connectivity options and increases control performance while enabling system-level flexibility in industrial and high-power-grid applications. Real-time communications are enabled by a dedicated Arm® Cortex-M4 based Connectivity Manager (CM), which offloads processing-intensive communications and optimizes connectivity. [Figure 7](#) shows the EtherCAT slave in F2838x device to link up with a PC using TwinCAT as master. Among the three cores of F2838x MCU (two C28x CPU cores (CPU1, CPU2) and one Cortex-M4 core), the EtherCAT peripheral can be connected to CPU1 or M4. In this design, M4 based connectivity manager is chosen to interact with the EtherCAT peripheral.

Figure 7. EtherCAT Configuration in Multi-Axis Servo System



2.3.4 Multi-Axis Servo Drive System

Figure 8 shows the layout of a typical multi-axis servo drive system, all control loops up to the position controller are implemented on the servo inverters. They are often linked via a high speed communication field bus to a high performance processor based primary controller. The primary controller distributes reference-values and receives feedback values from the servo inverter.

Figure 8. Typical Layout of a Multi-axis Servo Drive System

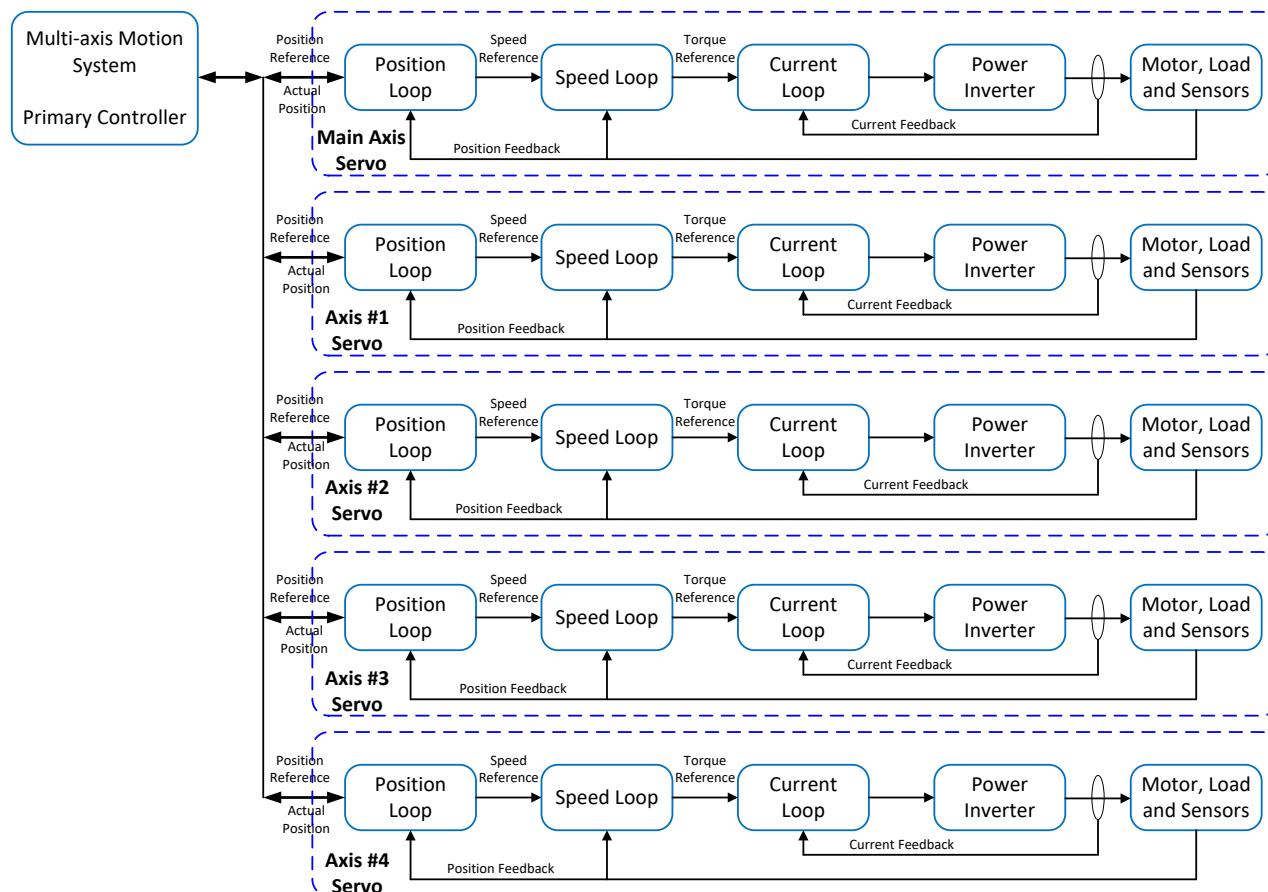


Figure 9 shows the control structure used in this design. The inverters only perform current control algorithms, and send the speed and position feedback signal to the position and speed controller via a high speed communication bus. All position and speed control loop functions are calculated on the position and speed controller for all axes.

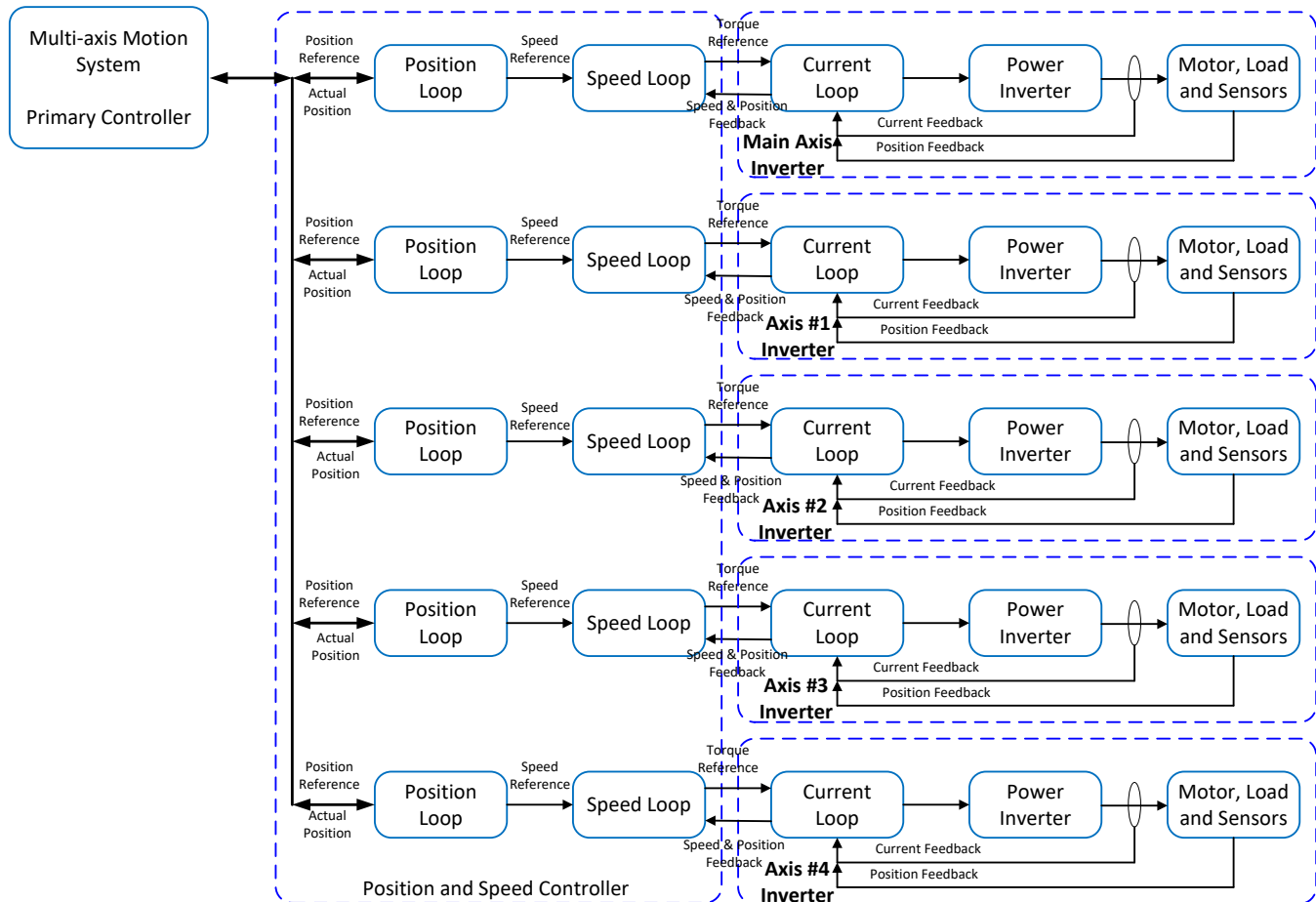
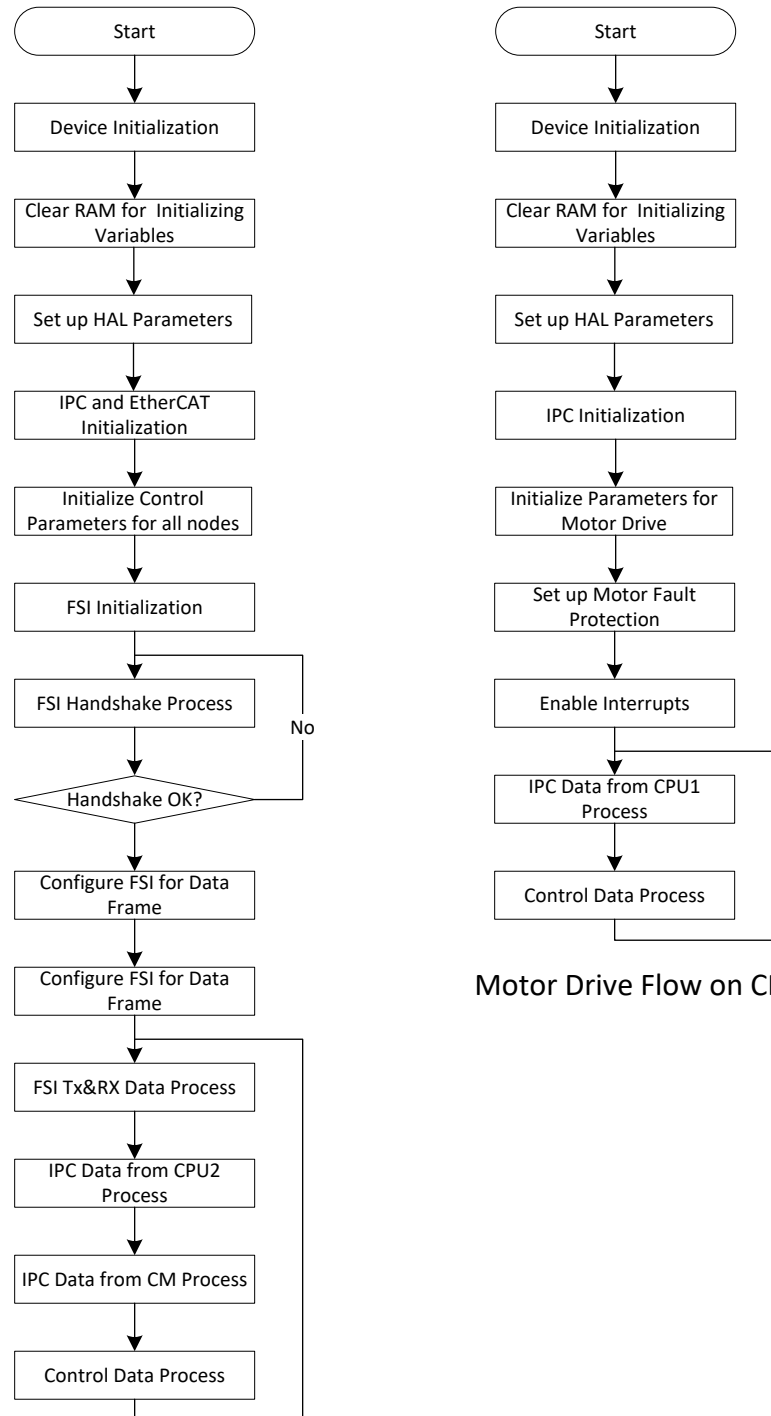
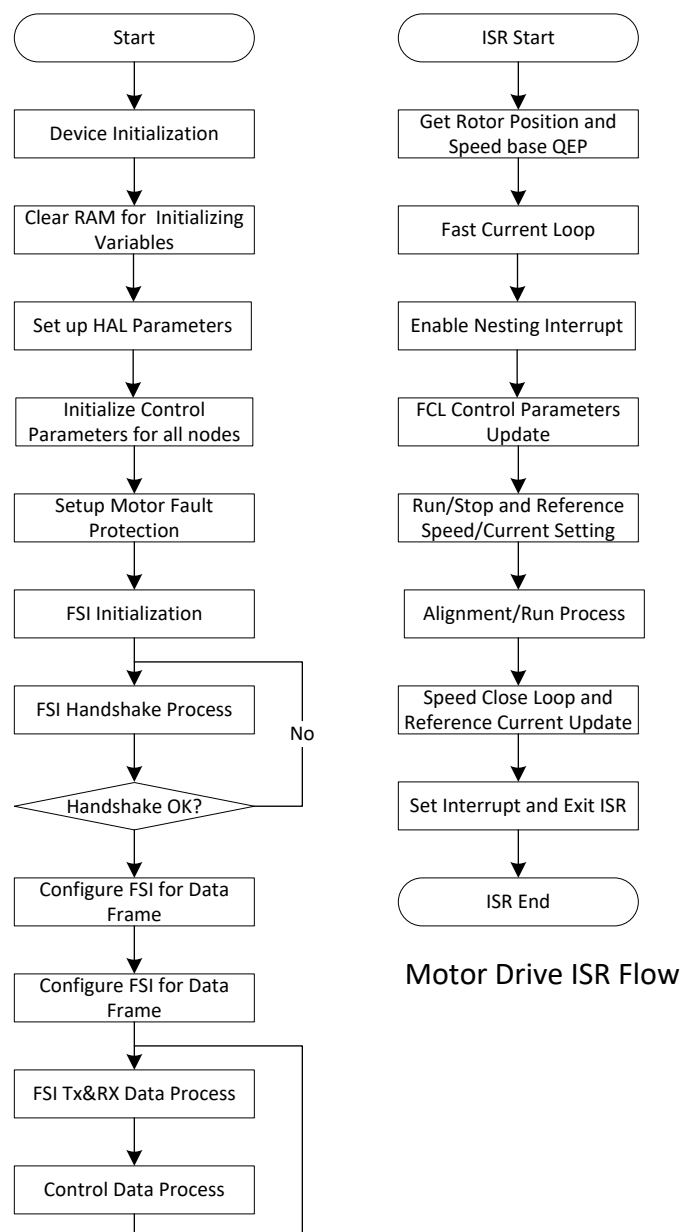
Figure 9. Layout of a Distributed Multi-axis Servo Drive System


Figure 10 shows the implementation of a distributed multi-axis servo drive system based on C2000 devices, where all position and speed control functions are executed on the master node, torque control loop runs on slave node. Position and speed feedback values are transmitted via FSI link between master and slave nodes. The distributed control architecture utilizes the high speed and low signal count of FSI to transfer more data with fewer channels, leading to reduced overall cost and improved reliability of the system.

Figure 11. Main Program Flowchart on CPU1 and CPU2 of F2838x

Motor Drive Flow on CPU2

Multi-Axis Motors Control Flow on CPU1

The software of each slave node runs on a F28004x device respectively, implements sensored-FOC and torque control with FCL. The main and ISR program flows are shown in [Figure 12](#).

Figure 12. Main and ISR Program Flowcharts on F28004x

Main Loop Program Flow on F28004x

3 Hardware, Software, Testing Requirements, and Test Results

3.1 Hardware

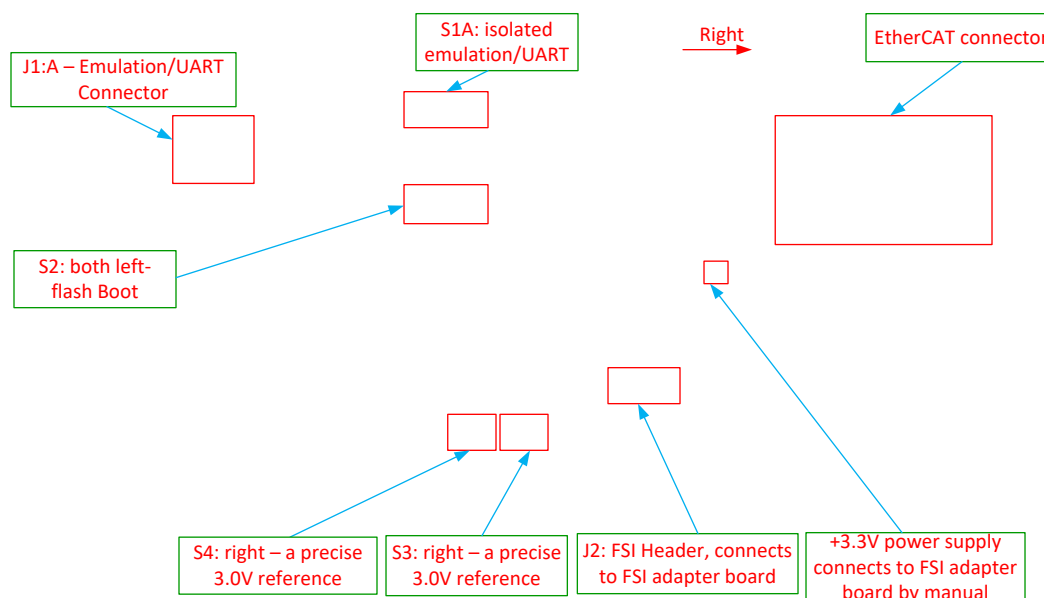
The reference design is based on the existing hardware tools, most of which are available from the TI Store. The details of the evaluation hardware and references to the user's guide are listed below:

- Controller
 - [TMDSCNCD28388D](#): F28388D controlCARD evaluation module
 - [LAUNCHXL-F280049C](#): C2000 Piccolo MCU F280049C LaunchPad™ development kit
- Inverter
 - [BOOSTXL-3PHGANINV](#): 48-V Three-Phase Inverter With Shunt-Based In-Line Motor Phase Current Sensing Evaluation Module
 - [TMDXIDDK379D](#): C2000 DesignDRIVE Development Kit for Industrial Motor Control
- Motor
 - [LVSERVOMTR](#): Low Voltage Servo (encoder) Motor and wiring harness for BOOSTXL-3PHGANINV
 - [HVPMSMMTR](#): High Voltage Permanent Magnet Synchronous Motor for TMDXIDDK379D
- Interface board
 - TMDSFSAIDAPEVM : FSI Adapter Board (will be available soon)
- DC Power Supply
 - A variable DC power supply rated at 100V/10A for BOOSTXL-3PHGANINV.
 - A variable DC power supply rated at 400V/2A for TMDXIDDK379D

3.1.1 TMDSCNCD28388D

Figure 13 shows the layout of TMS320F28388D controlCard, on-board switches setting as marked. Please refer to the [TMS320F28388D controlCARD Information Guide](#) for more details.

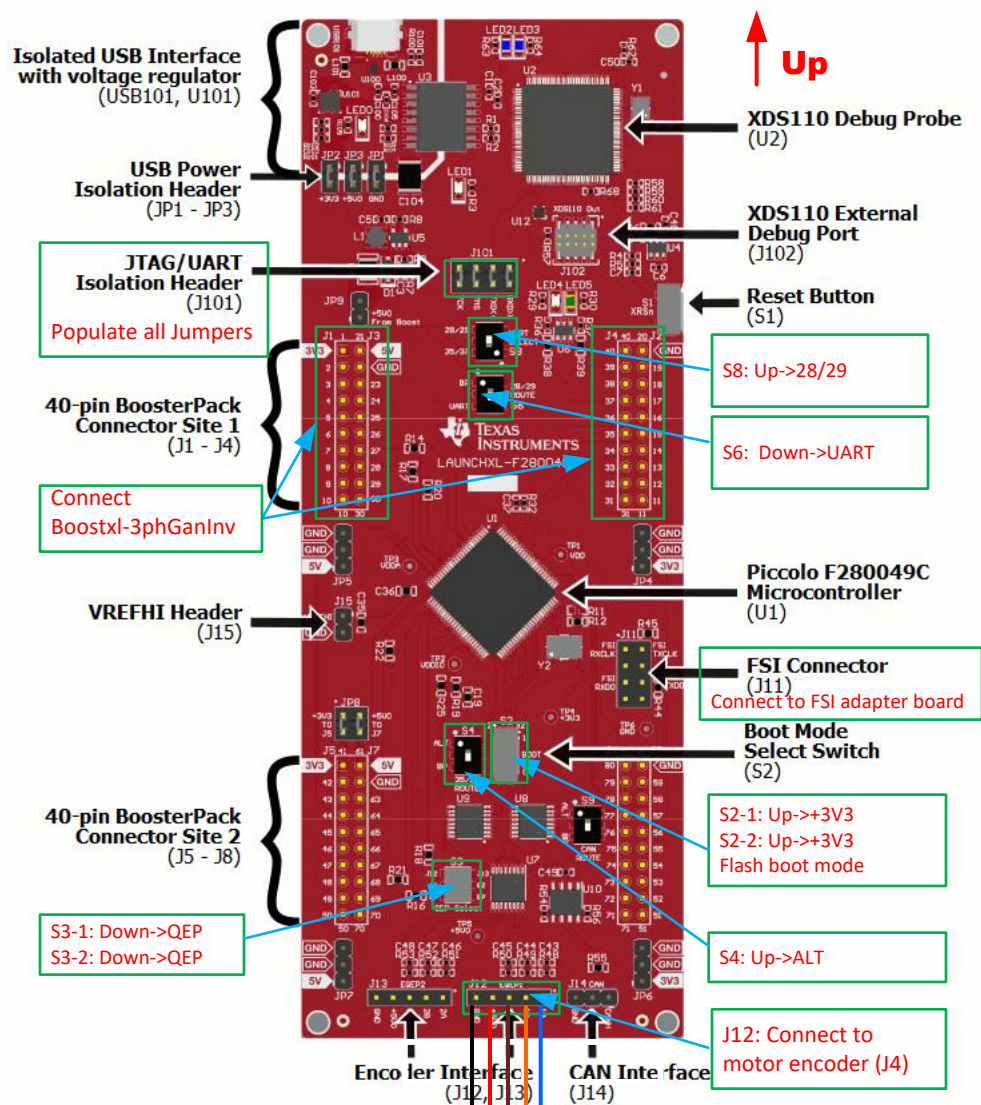
Figure 13. Layout of TMDSCNCD28388D and Switches Setting



3.1.2 LAUNCHXL-F280049C

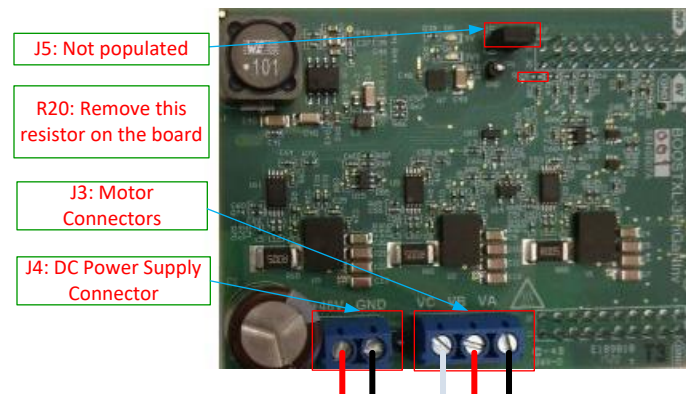
Figure 14 shows the layout of TMS320F20049C LaunchPad, on-board switches setting as marked. Please refer to the [C2000™ Piccolo™ F28004x Series LaunchPad™ Development Kit](#) for details.

Figure 14. Layout of LAUNCHXL-F280049C and Switches Setting



3.1.3 BOOSTXL-3PHGANINV

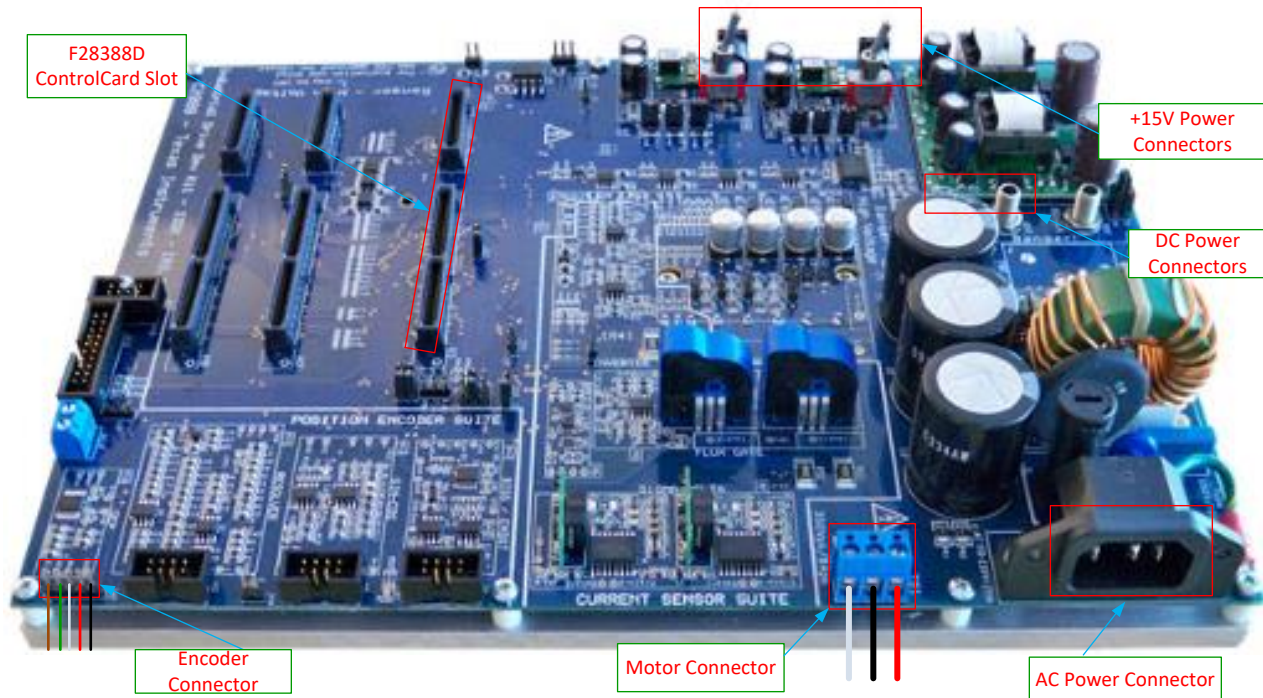
Figure 15 shows the layout of BOOSTXL-3PHGANINV, on-board switches setting and wires connection as marked. Please refer to the [BOOSTXL-3PhGaNInv Evaluation Module User's Guide](#) for details.

Figure 15. Layout, Switches Setting and Wires Connection of BOOSTXL-3PHGANINV

NOTE:

- Don't populate the jumper on the J5.
- Remove the R20 resistor on the board, it is near the Pin30 of J3 and TP1.

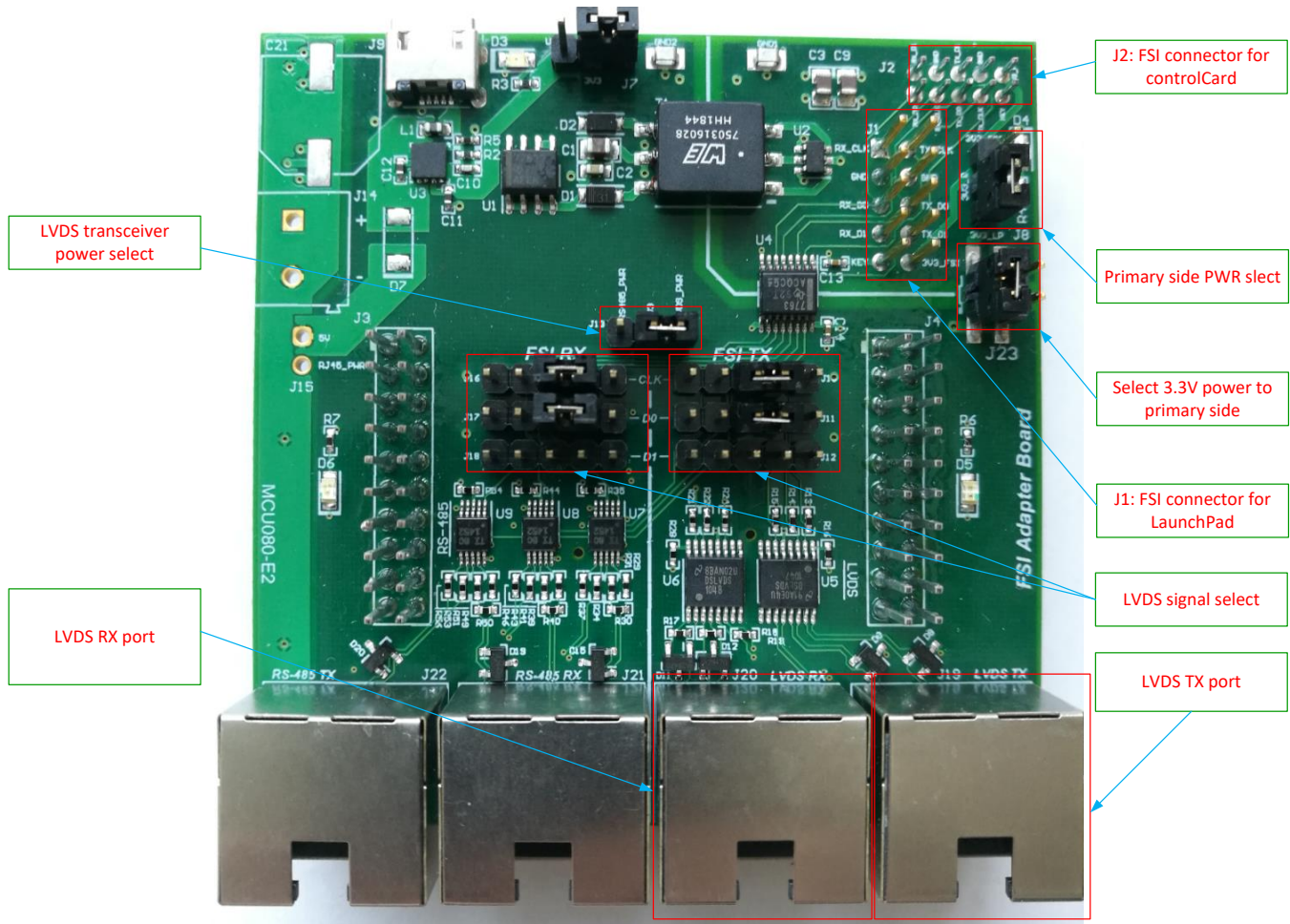
3.1.4 TMDXIDDK379D

Figure 16 shows the layout of TMDXIDDK379D, on-board switches setting and wires connection as marked. Please refer to the [DesignDRIVE Development Kit IDDK v2.2.1 - User's Guide](#) for details.

Figure 16. Layout, switches setting and wires connection of TMDXIDDK379D


3.1.5 TMDSFSIADAPEVM

Figure 17 shows the layout of TMDSFSIADAPEVM, on-board switches setting as marked. Please refer to the [TMDSFSIADAPEVM FSI Adapter Board User's Guide](#) for details.

Figure 17. Layout and Switches Setting of TMDSF5IADPEVM


3.2 Software

The firmware of the design is released within the [C2000Ware MotorControl SDK](#). To run the design the required software is the following:

- Download and install the latest version of [Code Composer Studio](#)
- Download and install the latest version of [C2000Ware MotorControl SDK](#).
- Download and install TwinCAT3 from the [Beckhoff](#)

3.2.1 Open project using CCS

The design includes multiple projects that run on F2838x for master node and on F28004x for slave nodes. Be sure to download and install [MotorControl software development kit \(SDK\) for C2000™ MCUs](#) software package revision 3.00.00.00 or later in the default install path.

- multi_axis_master_ctrl_f2838x_CPU1 - speed and position control loops for all axes on CPU1 of F2838x.
- multi_axis_master_drive_f2838x_CPU2 - main axis motor drive on CPU2 of F2838x.
- multi_axis_master_ecat_f2838x_cm - EtherCAT communication with PC on cortex-m4 of F2838x
- multi_axis_slave_node<n>_f28004x_cpu - slave axis #<n> motor drive on CPU of F28004x, where <n> is 1, 2, 3, or 4.

Use the following steps to get started:

1. Install the CCS software (version 9.2 or later).
2. Open the CCS software. Create the CCS workspace.
3. Go to **View > Resource Explorer**. Below the **Resource Explorer**, go to **Software > C2000Ware_MotorControl_SDK_<version>**.
4. Below **C2000Ware_MotorControl_SDK_<version>**, select **Solutions > tidm_02006_multi_axis_drive > F2838x for master node, or F28004x for slave nodes**.
5. After selecting the CCS project, click **Import to IDE** on the top right corner. This action imports the project into the CCS workspace environment.

3.2.2 Project Structure

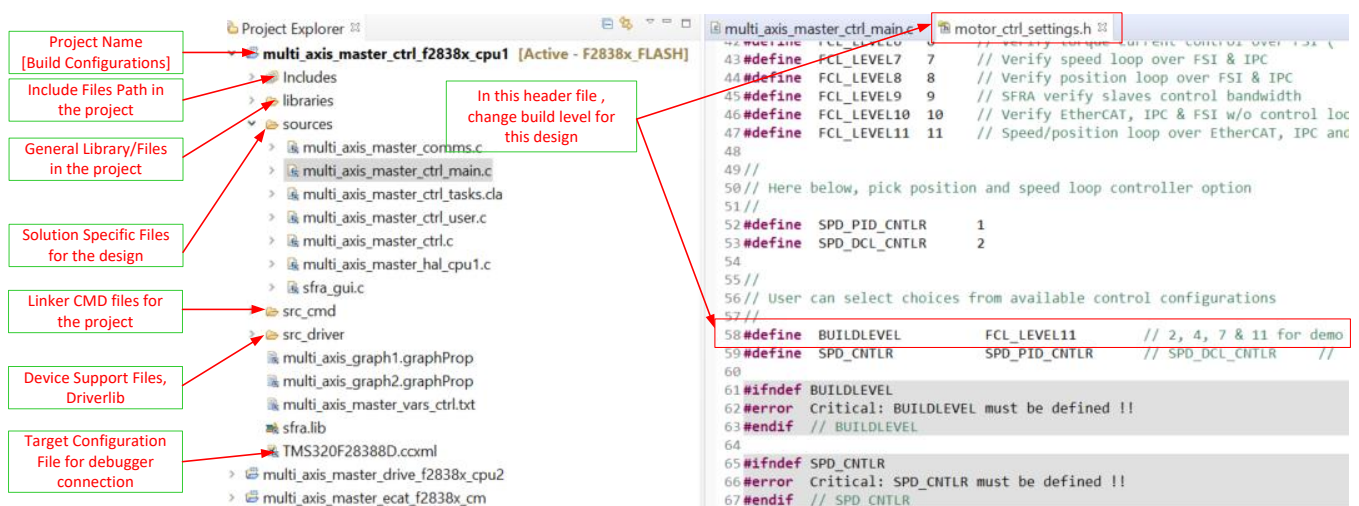
When the projects are imported in CCS respectively, the project includes device-independent and device-specific files.

The solution-specific and device-independent files are <solution>_<function>.c/.h, where <solution> is "multi_axis_", and <function> is ctrl, comm or drive. These files are responsible for the control, drive and communication functions of the solution.

The board-specific and device-specific files are <solution>_hal_<core>.c/.h, where <core> is CPU1, CPU2, or CM. These files consists of device-specific drivers to run the solution.

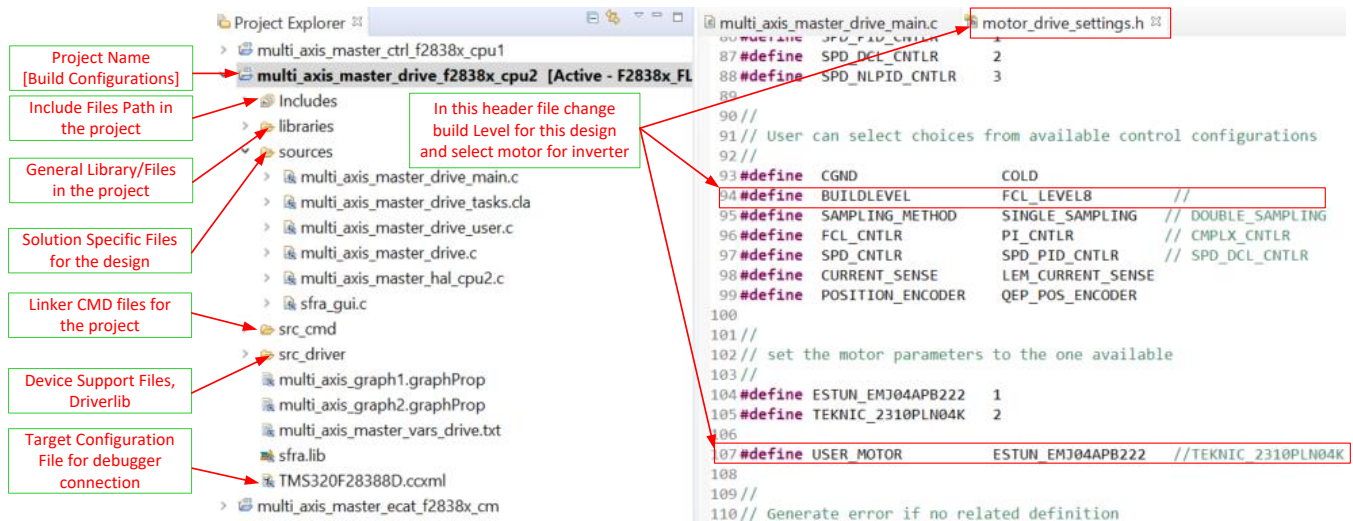
The project explorer displays the structure of "multi_axis_master_ctrl_f2838x_CPU1" project inside CCS as shown in Figure 18.

Figure 18. Project Explorer View of Master Control Project



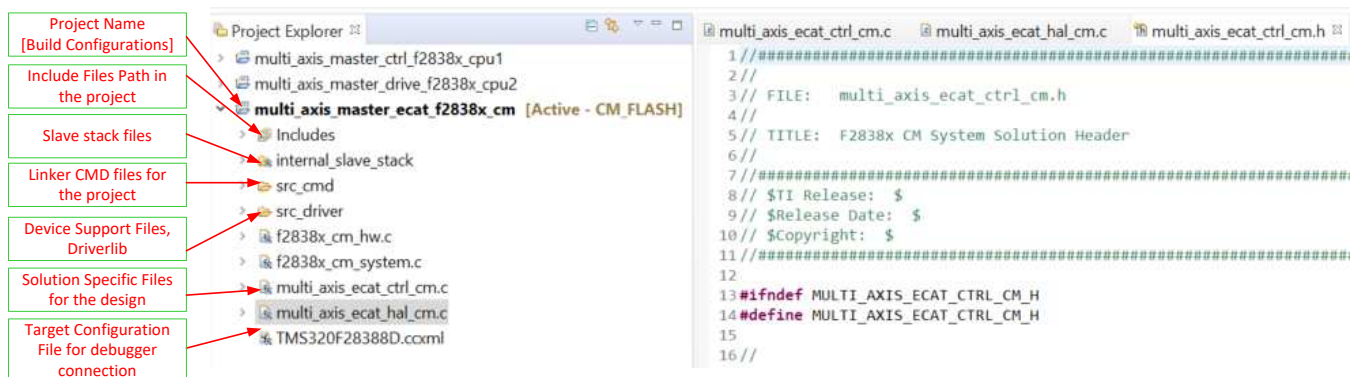
The project explorer displays the structure of **multi_axis_master_drive_f2838x_CPU2** project inside CCS as shown in Figure 19.

Figure 19. Project Explorer View of Main Axis Drive Project

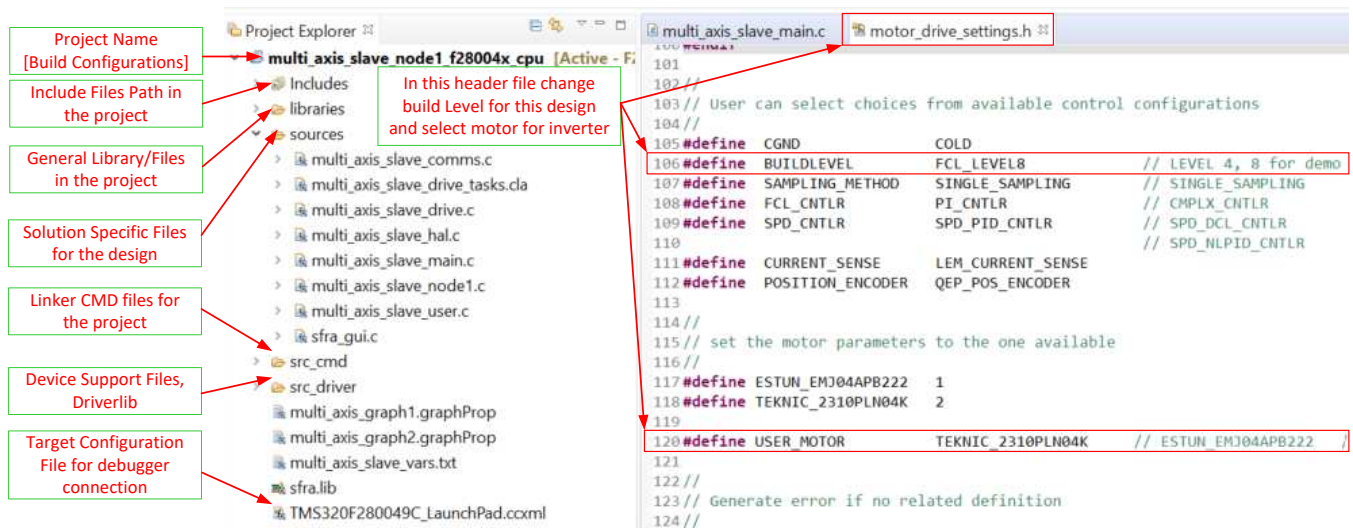


The project explorer displays the structure of **multi_axis_master_ecat_f2838x_cm** project inside CCS as shown in Figure 20.

Figure 20. Project Explorer View of EtherCAT Communication Project

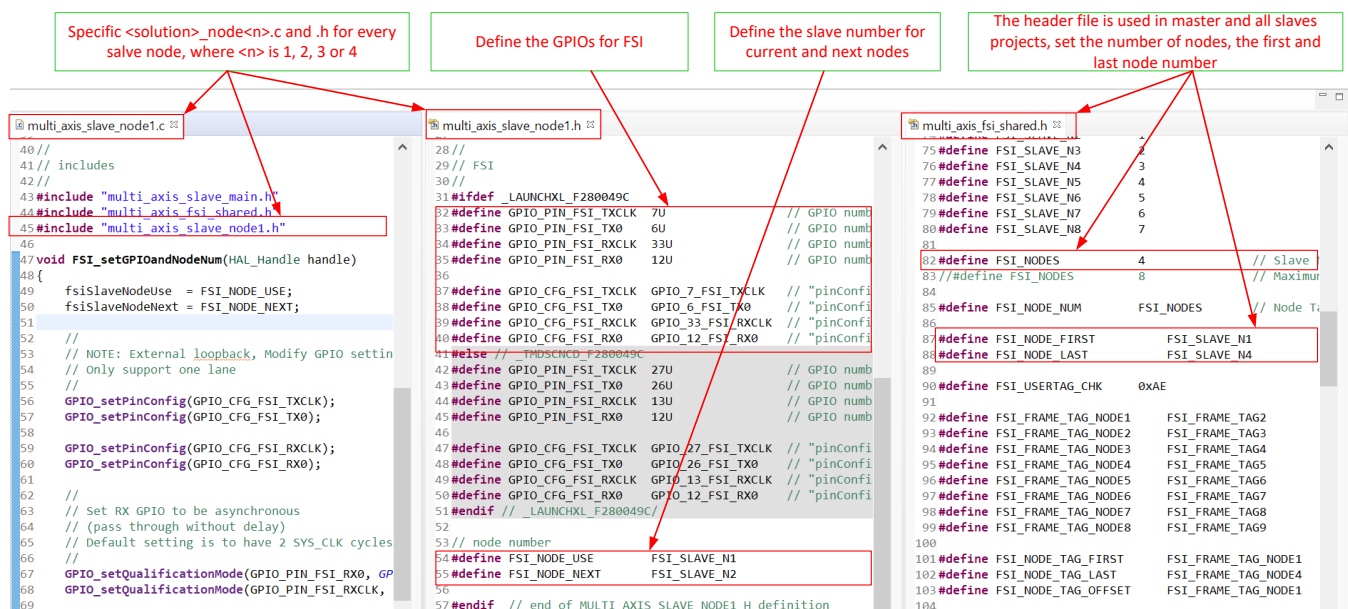


The project explorer displays the structure of **multi_axis_slave_node<n>_f28004x_cpu** project inside CCS as shown in Figure 21, where <n> is 1, 2, 3 or 4. Each slave drive project uses the same project structure and shares most of files, the only different files are multi_axis_slave_node<n>.c and the included header file multi_axis_slave_node<n>.h, where n is 1, 2, 3, or 4.

Figure 21. Project Explorer View of Slave Axis Drive Project


In this reference design, the number of the supported nodes is 4. The designer may increase or decrease the number as shown in [Figure 22](#).

1. Open the multi_axis_slave_node<n>.h.
2. Change the GPIO assignment for the FSI signals according to the hardware board.
3. Change the FSI_NODE_USE and FSI_NODE_NEXT definition. Define the FSI_NODE_USE to FSI_SLAVE_N<n>, where n represents the slave number in the daisy-chain loop (i.e. 1, 2, 3, etc.). Define FSI_NODE_NEXT to FSI_SLAVE_N<m>, where m must be equal to n+1. Both FSI_NODE_USE and FSI_NODE_NEXT can be set to FSI_SLAVE_N1 if there is only one slave node in the system.
4. Open multi_axis_fsi_shared.h
5. Define FSI_NODES to the maximum number of the supporting nodes which must be less than 15.
6. Define FSI_NODE_LAST to FSI_SLAVE_N<n>, where n equals to the total number of nodes in the chain. For example, the n should be equal to 4 if there are 4 slave nodes in the system.

Figure 22. The Files for Slave Nodes


3.2.3 Running the Project

The system is gradually built up in order for the final system to be confidently operated. Multiple phases of the incremental system build are designed to verify the major software modules used in the system. [Table 2](#), [Table 3](#), and [Table 4](#) summarize the core modules testing and use at each build level in the incremental system build approach.

Table 2. Functions Verified at Each Build Level in Master Control Project

Level 1	Verify HAL, peripherals configuration, control ISR, IPC with CPU2
Level 2	verify main-axis drive via IPC, master (IDDK) runs standalone
Level 3	verify EtherCAT and IPC for main-axis without running motor, master (IDDK) runs standalone
Level 4	verify main-axis drive via EtherCAT&IPC
Level 5	verify FSI functions, need setup the system with master node and at least one slave node. (No EtherCAT)
Level 6	verify the torque current control over FSI. (No EtherCAT)
Level 7	verify speed loop control over FSI (No EtherCAT)
Level 8	verify position loop control over FSI (No EtherCAT)
Level 9	integrate SFRA to verify the speed loop for slave node on master (No EtherCAT)
Level 10	verify EtherCAT, IPC and FSI integration without control loop
Level 11	verify speed or position loop over EtherCAT, IPC and FSI

NOTE:

- Select the build level in motor_ctrl_settings.h
- Make sure that main-axis drive is verified by running level1~level5 one by one in [Table 3](#) before run the level 2, 3, 4, 6, 7, 8, 9, 10, and 11 as shown in [Table 2](#).
- Make sure that all slave axes drives are verified independently by running level1~level5 one by one as shown in [Table 4](#) before run level 6, 7, 8, 9, 10 and 11 as shown in [Table 2](#).

Table 3. Functions Verified at Each Build Level in Main Axis Drive Project

Level 1	verify HAL configuration, SVGEN and PWM generation
Level 2	verify ADC and offset calibration, QEP and speed measurement with open loop running
Level 3	verify closed current (torque) loop suing FCL with the position angle from encoder and FCL
Level 4	verify closed speed loop and FCL
Level 5	verify closed position loop
Level 6	integrate SFRA and verify closed speed loop
Level 7	verify IPC without drive control
Level 8	torque control on CPU2, the reference speed/position from CPU1 over IPC
Level 9	integrate SFRA for tuning current loop over IPC

NOTE:

- Run level1~level5 one by one in [Table 3](#) before run level 8 and level 9 to implement main axis drive over IPC

Table 4. Functions Verified at Each Build Level in Slave Axis Drive Project

Level 1	verify HAL configuration, SVGEN and PWM generation
Level 2	verify ADC and offset calibration, QEP and speed measurement with open loop running
Level 3	verify closed current (torque) loop suing FCL with the position angle from encoder and FCL
Level 4	verify closed speed loop and FCL
Level 5	verify closed position loop

Table 4. Functions Verified at Each Build Level in Slave Axis Drive Project (continued)

Level 6	integrate SFRA and verify closed speed loop
Level 7	verify IPC without drive control
Level 8	torque control on slave, receive the reference speed/position from master over FSI
Level 9	integrate SFRA for tuning current loop. Torque control on slave, the reference speed/position from master over FSI

NOTE:

- Run level1~level5 one by one in [Table 4](#) before run level 8 and level 9 to implement slave axis drive over FSI.

3.2.3.1 System Software Integration Configuration

Each project has multiple build levels, so there are many combinations of builds in the design. The following are some key combination selections at build level to run the design.

- Verify IPC between CPU1 and CPU2 without motor drive
 - Don't connect the motor to TMDXIDDK379D. It's not necessary to connect slave drive at this point.
 - Set build level to level 1 in "motor_ctrl_settings.h" in "multi_axis_master_ctrl_f2838x_CPU1" project. Build, load and run the program on CPU1 of F2838x.
 - Set build level to level 7 in "motor_drive_settings.h" in "multi_axis_master_drive_f2838x_cpu2" project. Build, load and run the program on CPU2 of F2838x.
- Verify EtherCAT between F2838x and TwinCAT on PC
 - Don't connect the motor to TMDXIDDK379D. It's not necessary to connect slave drive at this point.
 - Set build level to level 3 in "motor_ctrl_settings.h" in "multi_axis_master_ctrl_f2838x_CPU1" project. Build, load and run the program on CPU1 of F2838x.
 - Build, load and run the "multi_axis_master_ecat_f2838x_cm" project program on CM of F2838x.
- Verify EtherCAT and IPC-CPU2&CPU1
 - Don't connect the motor to TMDXIDDK379D. It's not necessary to connect slave drive at this point.
 - Set build level to level 3 in "motor_ctrl_settings.h" in "multi_axis_master_ctrl_f2838x_CPU1" project. Build, load and run the program on CPU1 of F2838x.
 - Set build level to level 7 in "motor_drive_settings.h" in "multi_axis_master_drive_f2838x_cpu2" project. Build, load and run the program on CPU2 of F2838x.
 - Build, load and run the "multi_axis_master_ecat_f2838x_cm" project program on CM of F2838x.
- Verify FSI between master and slaves without motor drive
 - Don't connect the motor to both TMDXIDDK379D and BOOSTXL-3PHGANINV.
 - Connect the FSI using TMDSFSAIDAPEVM between master and slaves.
 - Set build level to level 5 in "motor_ctrl_settings.h" in "multi_axis_master_ctrl_f2838x_CPU1" project. Build, load and run the program on CPU1 of F2838x.
 - Set build level to level 7 in "motor_drive_settings.h" in "multi_axis_slave_node1_f28004x_cpu" project for each slave node. Build, load and run the program on F28004x.
- Verify EtherCAT, IPC, and FSI
 - Don't connect the motor to both TMDXIDDK379D and BOOSTXL-3PHGANINV.
 - Connect the FSI using TMDSFSAIDAPEVM between master and slaves.
 - Set build level to level 10 in "motor_ctrl_settings.h" in "multi_axis_master_ctrl_f2838x_CPU1" project. Build, load and run the program on CPU1 of F2838x.
 - Set build level to level 7 in "motor_drive_settings.h" in "multi_axis_master_drive_f2838x_cpu2" project. Build, load and run the program on CPU2 of F2838x.
 - Build, load and run the "multi_axis_master_ecat_f2838x_cm" project program on CM of F2838x.
 - Set build level to level 7 in "motor_drive_settings.h" in "multi_axis_slave_node1_f28004x_cpu"

project for each slave node. Build, load and run the program on F28004x.

6. Verify multi-axis drive over FSI without EtherCAT
 - a. Don't connect the motor to TMDXIDDK379D.
 - b. Connect the FSI using TMDSFSAIDAPEVM between master and slaves.
 - c. Connect the motor BOOSTXL-3PHGANINV and the encoder to LAUNCHXL-F280049C.
 - d. Set build level to level 7 in "motor_ctrl_settings.h" in "multi_axis_master_ctrl_f2838x_CPU1" project. Build, load and run the program on CPU1 of F2838x.
 - e. Set build level to level 8 in "motor_drive_settings.h" in "multi_axis_slave_node1_f28004x_cpu" project for each slave node. Build, load and run the program on F28004x.

NOTE: Each slave axes must be verified by using level 1 to level 5 independently before run this combination.

7. Verify multi-axis drive over FSI with EtherCAT
 - a. Connect the motor and its encoder to TMDXIDDK379D.
 - b. Connect the FSI using TMDSFSAIDAPEVM between master and slaves.
 - c. Connect the motor BOOSTXL-3PHGANINV and the encoder to LAUNCHXL-F280049C.
 - d. Set build level to level 11 in "motor_ctrl_settings.h" in "multi_axis_master_ctrl_f2838x_CPU1" project. Build, load and run the program on CPU1 of F2838x.
 - e. Set build level to level 8 in "motor_drive_settings.h" in "multi_axis_master_drive_f2838x_cpu2" project. Build, load and run the program on CPU2 of F2838x.
 - f. Build, load and run the "multi_axis_master_ecat_f2838x_cm" project program on CM of F2838x.
 - g. Set build level to level 8 in "motor_drive_settings.h" in "multi_axis_slave_node1_f28004x_cpu" project for each slave node. Build, load and run the program on F28004x.

NOTE:

- Main axis and slave axes motor drive must be verified by using level 1 to level 5 independently before control the motor over FSI or IPC.
 - The detailed operation steps about combination 7 as show in [Section 3.2.3.2](#), [Section 3.2.3.3](#), [Section 3.2.3.4](#), and [Section 3.2.3.5](#). The other combinations can refer to these sections also, only need to set the specific build level as above lists.
-

3.2.3.2 Run the Master Control Code on CPU1 of F2838x

1. Open the CCS workspace of the master, and import the master control project by clicking **Project > Import CCS Projects...** in CCS menu. Select the **multi_axis_master_ctrl_f2838x_CPU1** by browsing to the folder: \ti\c2000\C2000Ware_MotorControl_SDK_<version>\solutions\tidm_02006_multi_axis_drive\2838x\ccs\sensored_foc.
2. Open **motor_ctrl_settings.h** and then select the build level 11 by setting the BUILDLEVEL to FCL_LEVEL11 (#define BUILDLEVEL FCL_LEVEL11).

NOTE:

- This reference design only shows the final build level, FCL_LEVEL11 to test the whole functions. To understand and operate the design, the builds must be done individually.
 - The detailed operation of CCS are introduced in [Dual-Axis Motor Control Using FCL and SFRA On a Single C2000™ MCU](#).
 - Control power supply to the IDDK is only needed for this build level, therefore, you can turn off the high voltage dc power at this point.
-

3. Right-click the project name and then click **Rebuild Project**.
4. When the build is completed, click the **Debug** button or **Run > Debug**, that launches a debugging session. In the case of dual CPU devices, a window might appear to select the CPU debug must be performed. In this step, select **CPU1**. The project then loads on the device for CPU1, and CCS debug perspective becomes active. The code halts at the start of the main routine.

5. To add the variables in the watch and expressions window by right-clicking within the Expressions Window and importing the **multi_axis_master_vars.txt** file from the debug directory at:
\\ti\c2000\C2000Ware_MotorControl_SDK_<version>\solutions\common\sensored_foc\debug.
6. With appropriate variables required to debug the system. Click the **Continuous Refresh** button on the watch window to enable continuous update of values from the controller.
7. To enable real-time mode by hovering the mouse on the buttons on the buttons on thee horizontal toolbar and clicking the "Enable Silicon Real-time Mode" button.
8. Run the project by clicking on "Resume" button.
9. Set "enableCtrlFlag" to 1 in watch window. In the build levels without enabling FSI, such as level1~level4, the variable named "sysVars.isrTicker" is incrementally increased as seen in the watch windows, to confirm the interrupt working properly. In other levels, the variable is only increased after FSI handshake completed successfully.

NOTE: In *F2838x_FLASH* build configuration, the *enableCtrlFlag* is set to 1 automatically after a delay.

10. Monitor *fsiHandShakeState* in the watch window if the value of the variable equals to *FSI_HANDSHAKE_DONE*, that means the handshake process completes.
11. Set *sysVars.ecatCtrlSet* to enable or disable the setting command or reference value from TwinCAT3 over EtherCAT.
 - a. If set *sysVars.ecatCtrlSet* to *ECAT_CTRL_DISABLE*, the control command and speed/position reference values are set in the watch window.
 - i. If set *sysVars.ctrlSynSet* to *CTRL_SYN_DISABLE*, the main axis and each slave axis are controlled respectively. The main axis is controlled by setting *ctrlVars[0].ctrlStateSet* and *ctrlVars[0].ctrlSpeedRef*, the slave axis #1, #2, #3 or #4 is control by setting *ctrlVars[n].ctrlStateSet* and *ctrlVars[n].ctrlSpeedRef* (n= 1, 2, 3, or 4 accordingly).
 - ii. If set *sysVars.ctrlSynSet* to *CTRL_SYN_ENABLE*, the main axis and all slave axes are controlled simultaneously by setting *sysVars.ctrlStateSet* and *sysVars.speedSet*.
 - b. If set *sysVars.ecatCtrlSet* to *ECAT_CTRL_ENABLE*, the control command and speed/position reference values are set by the communication data from TwinCAT3 on PC as [Section 3.2.3.4](#).

3.2.3.3 Run the Main Axis Drive Code on CPU2 of F2838x

1. Open the CCS workspace of the master, and import main axis drive project by clicking **Project > Import CCS Projects...** in CCS menu. Select the **multi_axis_master_drive_f2838x_cpu2** by browsing to the folder: *\\ti\c2000\C2000Ware_MotorControl_SDK_<version>\solutions\tidm_02006_multi_axis_drive\f2838x\ccs\sensored_foc*
2. Open *motor_drive_settings.h* and select the build level 8 by setting the *BUILDLEVEL* to *FCL_LEVEL8* (*#define BUILDLEVEL FCL_LEVEL8*).

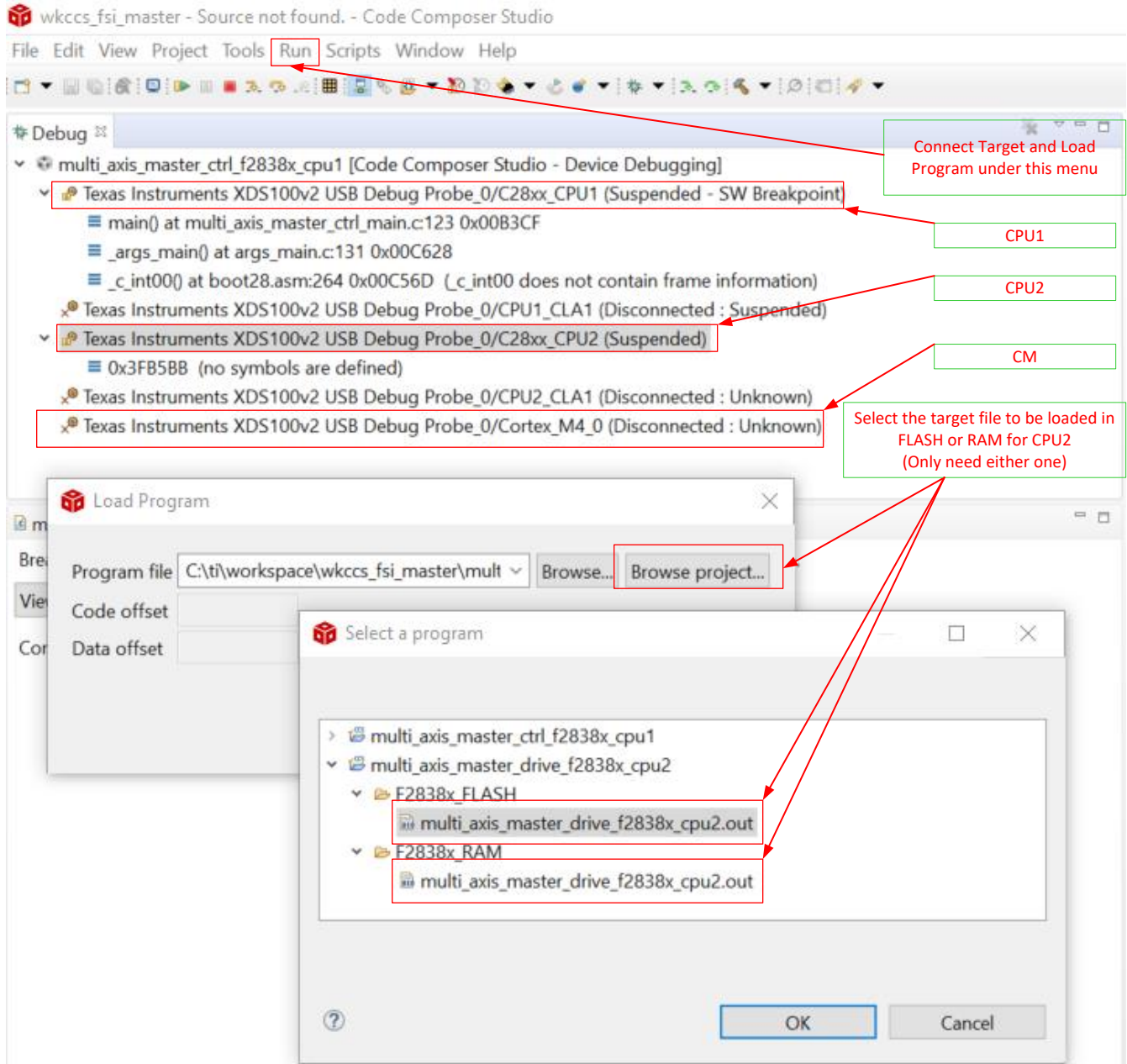
NOTE:

- In this design guide, just shows the final build level, *FCL_LEVEL8* to test whole system functions. The builds from *LEVEL1* to *LEVEL6* must be done one by one for the main axis drive code to validate motor drive modules on the inverter. Refer to [Dual-Axis Motor Control Using FCL and SFRA On a Single C2000™ MCU](#) to do these steps to build and run the code with different incremental builds.
 - Do not "turn on" the high voltage dc power until indicated. Control power supply to the IDDK is alone needed for this build level, therefore, you can turn off the high voltage dc power at this point.
-

3. Right-click the project name and click **Rebuild Project**.
4. When the build is completed, go to **Debug Perspective**. From within the debug window, click **Debug Probe/C28x_CPU2**, right click and select **Connect Target**.
5. After connected the target, select **Run > Load > Load Program...**, and then select the related program

for CPU2 by clicking **Browse project..** as shown in Figure 23. Click the **OK** button on the pop-up window, the program then loads on the device for CPU2. The code halts at the start of the main routine.

Figure 23. Load Program to Memory for CPU2 or CM



6. Run the project by clicking the **Resume** button.
7. Set enableDriveFlag to 1 in watch window. The variable named motorVars.isrTicker is incrementally increased as seen in the watch windows, to confirm the interrupt working properly.

NOTE: In F2838x_FLASH build configuration, the enableDriveFlag will be set to 1 automatically after a delay.

8. Unlike the previous build levels (LEVEL1 ~ LEVEL6), here the commands received by the controller are not looped back and are used to actually set the operational mode or references for the main axis drive. The operational command and status of the main-axis drive is set via CPU1 > IPC > CPU2 and

sent back via CPU2 > IPC > CPU1.

NOTE: Since this build level attempts to drive the motor, high voltage dc power to IDDK can be turned on at this point.

9. Check motorVars.faultStatusFlag value in watch window, if the value equals to 0, that means there is no any fault trip.

3.2.3.4 Run the EtherCAT Communication Code on CM of F2838x

1. Open the CCS workspace of the master, and import main axis drive project by clicking the **Project > Import CCS Projects...** in the CCS menu. Select the multi_axis_master_ecat_f2838x_cm by browsing to the folder: `\\ti\\c2000\\C2000Ware_MotorControl_SDK_<version>\\solutions\\tidm_02006_multi_axis_drive\\f2838x\\ccs\\sensored_foc`
2. Ensure that CPU1 is actively running the project described in the previous section handing off the EtherCAT ownership to CM.
3. Besides USB connection between controlCARD and computer for JTAG purposes, connect an Ethernet cable between controlCARD RJ45 Port 0 and computer
4. Right-click the project name and click Rebuild Project.
5. When the build is completed, go to Debug Perspective. From within the Debug Probe/Cortex_M4_0, right-click and select Connect Target.
6. After connecting the target, select **Run > Load > Load Program..**, and then select the related program for CM by clicking **Browse project..** as similarly shown in [Figure 23](#). Click the OK button on the pop-up window, the program then loads on the device for CM. The code halts at the start of the main routine.
7. Run the project by clicking the Resume button. This would configure the F2838x EtherCAT slave controller and also the IPC on CM side for data transfer between CM and CPU1.
8. Set enableECATFlag to 1 in watch window. The variable named mainLoopCntr is incrementally increased as seen in the watch windows, to confirm the program working properly.
Now the EtherCAT slave controller is ready to connect to EtherCAT master, and in this design, the EtherCAT master is TwinCAT3 running on PC.
9. Please refer to Section 12. (Setup TwinCAT), Section 12.4 (Scanning for EtherCAT Devices via TwinCAT), and Section 12.5 (Program ControlCard EEPROM for ESC) in [EtherCAT-Based Connected Servo Drive Using Fast Current Loop on PMSM](#). In this step, use the ESI file (`\\ti\\c2000\\C2000Ware_MotorControl_SDK_<version>\\solutions\\tidm_02006_multi_axis_drive\\f2838x\\ssc_configuration`).

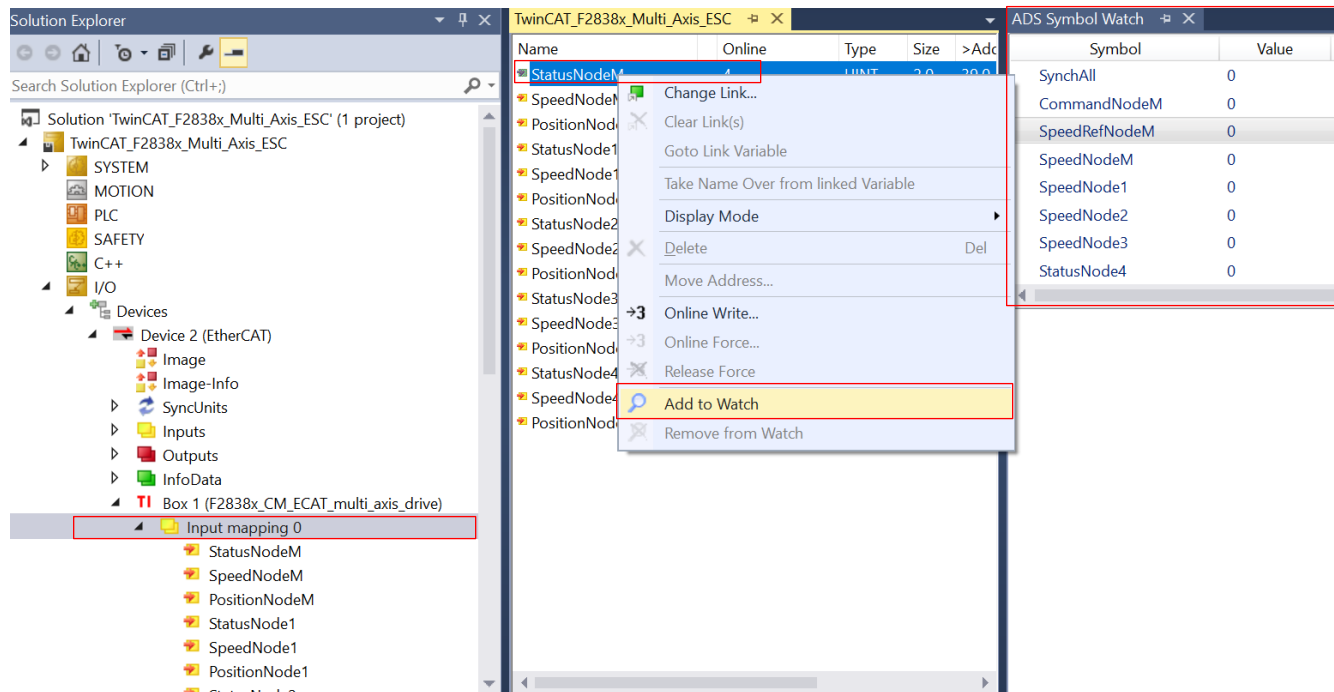
NOTE: Please refer to the EtherCAT Slave Controller Software User's Guide details details how to setup the EtherCAT master software (TwinCAT) on your compute. The guide available at `\\ti\\c2000\\C2000Ware_<version>\\libraries\\communications\\Ethercat\\f2838x\\docs`

10. Within TwinCAT, expand the explorer to the EtherCAT box (**I/O > Devices > Device 2 (EtherCAT) > TI Box 1** (F2838x_CM_ECAT_multi_axis_drive)). The Input mapping and Output mapping as shown in [Figure 24](#).

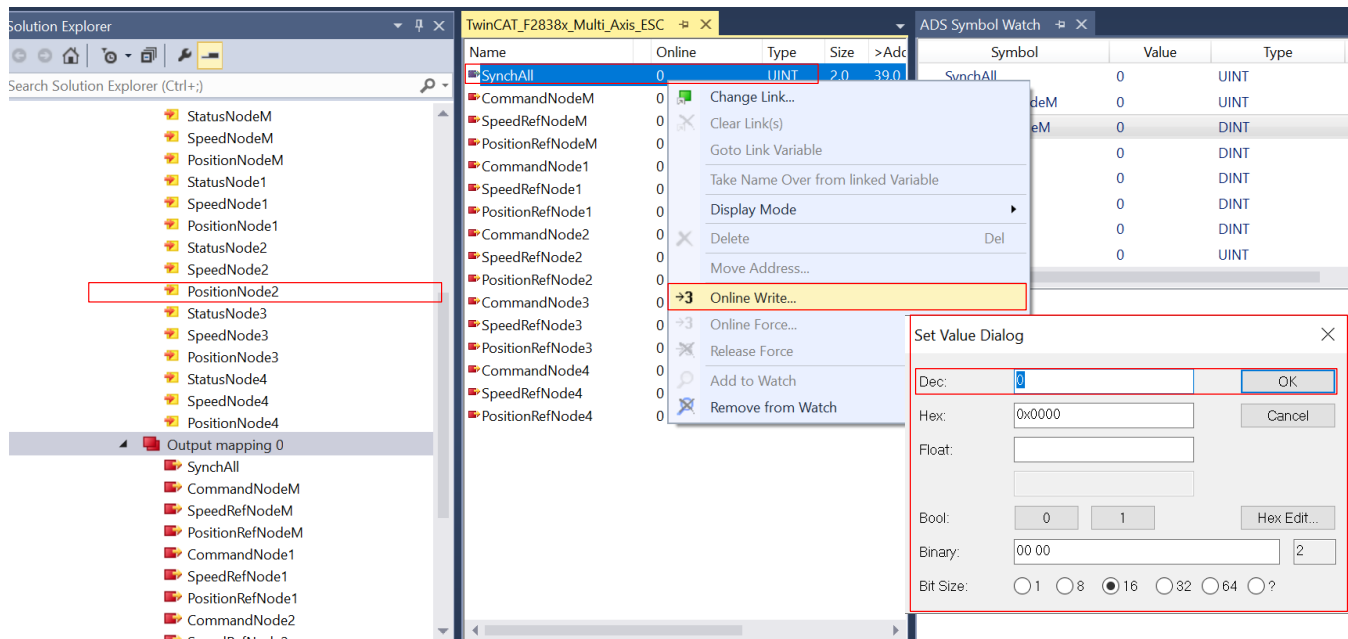
Figure 24. Inputs and Outputs in TwinCAT Solution Explorer

Name	Online	Type	Size	>Addr...	In/Out	User I
StatusNodeM	0	UINT	2.0	39.0	Input	0
SpeedNodeM	0	DINT	4.0	41.0	Input	0
PositionNodeM	4645	DINT	4.0	45.0	Input	0
StatusNode1	0	UINT	2.0	49.0	Input	0
SpeedNode1	0	DINT	4.0	51.0	Input	0
PositionNode1	5480	DINT	4.0	55.0	Input	0
StatusNode2	0	UINT	2.0	59.0	Input	0
SpeedNode2	0	DINT	4.0	61.0	Input	0
PositionNode2	5790	DINT	4.0	65.0	Input	0
StatusNode3	0	UINT	2.0	69.0	Input	0
SpeedNode3	0	DINT	4.0	71.0	Input	0
PositionNode3	8337	DINT	4.0	75.0	Input	0
StatusNode4	0	UINT	2.0	79.0	Input	0
SpeedNode4	0	DINT	4.0	81.0	Input	0
PositionNode4	3829	DINT	4.0	85.0	Input	0
WcState	0	BIT	0.1	1522.1	Input	0
InputToggle	1	BIT	0.1	1524.1	Input	0

11. Select Input mapping 0 to view all the feedback from the system. Or add the inputs to the Symbol Watch as shown in Figure 25. Right click the selected input, and then click **Add to Watch**. The following inputs are the feedback parameters monitored via EtherCAT.
 - a. StatusNode<n>, the status from the main axis or slave axes, where n is M (Main Axis), 1 (Axis #1), 2 (Axis #2), 3 (Axis #3), 14(Axis #4). 0-Stop the motor, 1-Speed control, 2-Position control, 3-Reset the state, 4-Had a fault from the axis controller.
 - b. SpeedNode<n>, the feedback speed from main and slave nodes.
 - c. PositionNode<n>, the feedback position from main and slave nodes.

Figure 25. Add Inputs to Symbol Watch


12. Select Output mapping 0 to set the control command or reference value to the main axis and slave axes as shown in [Figure 26](#). Right click the output, select Online Write, set the target value for the output, and then click the **OK** button.
 - a. SynchAll, 0-disable synchronization to control all axes respectably. 1- enable synchronization to control simultaneously. All reference command/speed/position will be set by CommandNodM/SpeedRefNodM/PositionRefNodeM.
 - b. CommandNode<n>, the command to the nodes, where n is M (Main Axis), 1 (Axis #1), 2 (Axis #2), 3 (Axis #3), 14(Axis #4). 0-Stop the motor, 1-Speed control, 2-Position control, 3-Reset the state.
 - c. SpeedRefNode<n>, the reference speed for the related node
 - d. PositionRefNode<n>, the reference speed for the related node

Figure 26. Set Control Command or Reference to Main and Slave Axes


3.2.3.5 Run the slave axis drive code on CPU of F28004x

1. Open a CCS workspace of the slave, and import slave axis drive project by clicking **Project->Import CCS Projects...** in CCS menu. Select the multi_axis_slave_node<n>_f28004x_cpu by browsing to the folder: `\ti\c2000\C2000Ware_MotorControl_SDK_<version>\solutions\tidm_02006_multi_axis_drive\28004x\ccs\sensored_foc`.
2. Open `motor_drive_settings.h.h` and select the build level 8 by setting the BUILDLEVEL to FCL_LEVEL8 (#define BUILDLEVEL FCL_LEVEL8)
3. Right-click the project name and click **Rebuild Project**.
4. When the build is completed, click the **Debug** button or **Run > Debug**, which launches a debugging session and CCS debug perspective becomes active. The code halts at the start of the main routine.

NOTE:

- This design guide only shows the final build level, FCL_LEVEL8 to test whole system functions. The builds from LEVEL1 to LEVEL6 must be done one by one for the slave axis drive code to validate motor drive modules on the inverter. Refer to [Dual-Axis Motor Control Using FCL and SFRA On a Single C2000™ MCU](#) to do these steps to build and run the code with different incremental builds.
 - Must *turn on* the dc power to the BOOSTXL-3PHGANINV at this point.
5. To add the variables in the watch and expressions window by right-clicking within the Expressions Window and importing the multi_axis_slave_vars.txt file from the debug directory at: `\ti\c2000\C2000Ware_MotorControl_SDK_<version>\solutions\common\sensored_foc\debug`.
 6. With appropriate variables required to debug the system. Click the **Continuous Refresh** button in the watch window to enable continuous update of values from the controller.
 7. Enable real-time mode by hovering the mouse on the buttons of the horizontal toolbar and clicking the Enable Silicon Real-time Mode button.
 8. Run the project by clicking the **Resume** button.
 9. Set enableFlag to 1 in watch window. In the build levels without enabling FSI, such as level1~level6, the variable named motorVars.isrTicker is incrementally increased as seen in the watch windows, to

confirm the interrupt working properly. In other levels, the variable is only increased after FSI handshake completed successfully.

NOTE:

- In F28004x_FLASH build configuration, the enableFlag will be set to 1 automatically after a delay.
-

10. Monitor fsiHandShakeSate in watch window if the value of the variable equals to FSI_HANDSHAKE_DONE, that means the handshake process completes.
 11. Unlike the previous build levels (LEVEL1 ~ LEVEL6), here the commands received by the controller are not looped back and are used to actually set the operational mode or references for the slave axis drive. The operational command and status of the slave axis drive is set over FSI.
 12. Check the motorVars.faultStatusFlag value in watch window, if its value equals to 0, that means there is no any fault trip.
- In this design, there are four slave nodes, follow the same operation steps for open and load these four projects in the slave controller.

3.2.3.6 Run the system

As [Section 3.2.3.2](#) , [Section 3.2.3.3](#), [Section 3.2.3.4](#), and [Section 3.2.3.5](#), load the related project into the master and slave controllers. And then follow the power-up or running sequencing to run the system.

1. If the code runs in FLASH standalone, the enable<nn>Flag will be set automatically after a delay. The system requires strict power-up sequencing as one of the following two options.
 - a. Power on the master board first, and then power on the last slave node (#4) to the first slave node (#1) one by one.
 - b. Or, power on all slave nodes first, wait for a delay, and then power the master board.
2. If the code runs in RAM with CCS, the enable<nn>Flag need to set to 1 by manual in CCS.
 - a. Run the multi_axis_master_ctrl_f2838x_cpu1 project and set the enableCtrlFlag to 1 in the debug window of CCS.
 - b. Run the multi_axis_master_drive_f2838x_cpu2 project and set the enableDriveFlag to 1 in the debug window of CCS.
 - c. Run the multi_axis_master_ecat_f2838x_cm, the enableECATFlag is set automatically by the program.
 - d. Run all slave nodes one by one from the last slave node (#4) to first slave node (#1). Set the enableFlag to 1 in CCS accordingly.

The LED (D1) on F28388D controlCard flashes quickly that means the motor drive program on CPU2 works well via IPC with CPU1. The LED (D2) flashes slowly that means handshake completes on master. A LED (D4) on LaunchPad flashes quickly that means handshake completes on this node. If these LED don't work as above, the system needs reset to make the handshaking work.

3.3 Testing and Results

3.3.1 Test Setup

CAUTION

Before you begin to test, to prevent damage to the motor drive board, ensure all boards are turned off, and then follow [Section 3.2.3](#) running projects for the design.

Before mounting the BoosterPack and FSI adapter board to the F280049C Launchpad, ensure that jumpers on LaunchPad, FSI Adapter board and BoosterPack are set correctly as shown in [Figure 14](#), [Figure 15](#) and [Figure 17](#). The motor is a PMSM motor with both QEP and HALL sensors available on its headers J4 and J10, respectively. The control scheme is based on QEP feedback; therefore, only the QEP header J4 is fed into the LaunchPad. The BoosterPack and FSI adapter board suggested for this evaluation will mount directly on to the LAUNCHXL-F280049C. Make sure to match the orientation of inverter BoosterPacks as shown in Figure 9 before mounting. Mount the inverter BoosterPack on LaunchPad connectors J1-J4, mount the FSI adapter board on LaunchPad connectors J5-J8. Until instructed, leave the BOOSTXL-3PhGaNIInv output headers and QEP headers open. When instructed to connect motor, connect the motor terminal to BOOSTXL-3PhGaNIInv connector terminal and motor's encoder header to J12(EQEP1) connector on LaunchPad listed in [Table 5](#). The slave node system setup as shown in [Figure 27](#).

Figure 27. BOOSTXL-3PhGaNIInv and TMDSFSAIDAPEVM Assembly with LAUNCHXL-F280049C

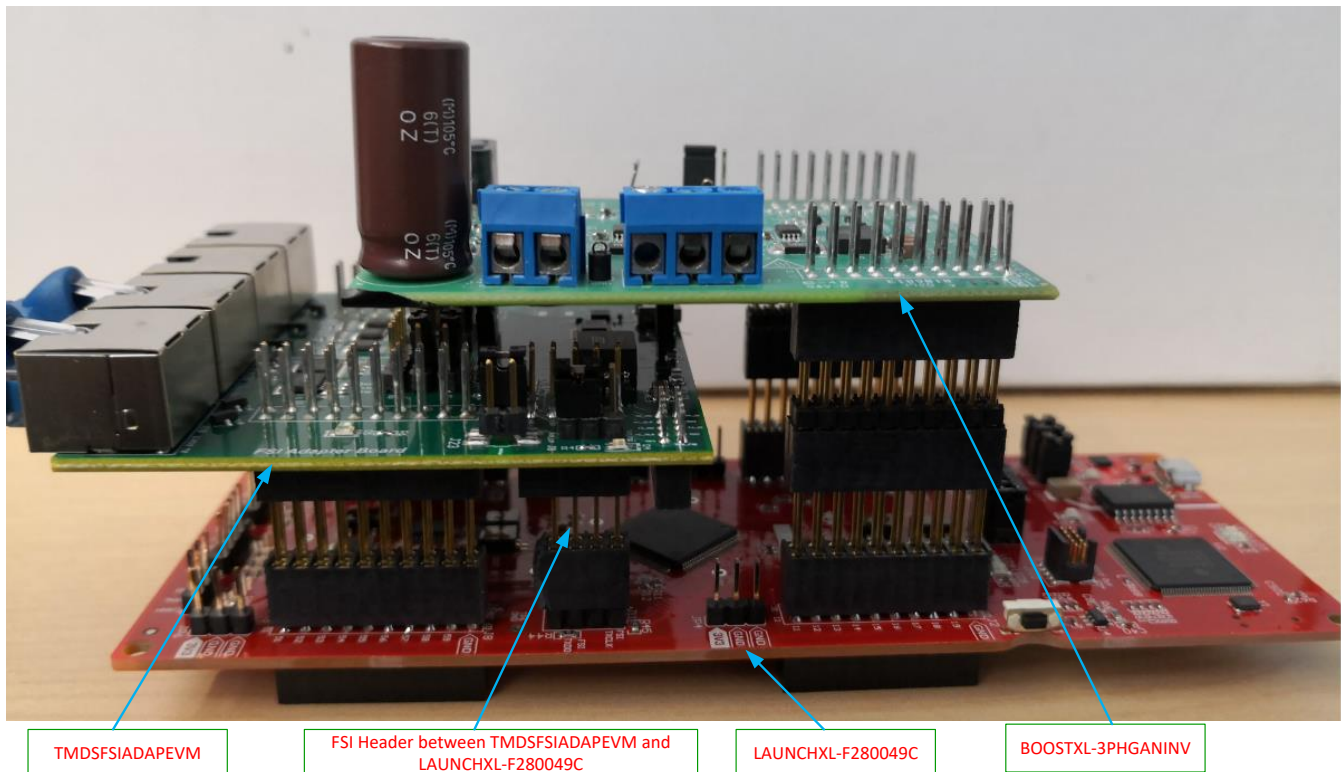


Table 5. Motor Power Lines and Encoder Wires Connections for Slave Node

PIN NUMBER	SIGNAL	PIN NUMBER	SIGNAL	COLOR
J3 on BOOSTXL-3PhGaNIInv		LVSERVOMTR Motor Connector		
1	VA	1	U	Black

Table 5. Motor Power Lines and Encoder Wires Connections for Slave Node (continued)

2	VB	2	V	Red
3	VC	3	W	White
J12 on LAUNCHXL-F280049C		Encoder Connector (J4)		
1	QEP1A	1	ENC A+	Blue
2	QEP1B	2	ENC B+	Orange
3	QEP1I	3	ENC I+	Brown
4	+5V	4	+5VDC IN	Red
5	GND	5	GND	Black

Before assembling TMDXCNC28388D and TMDSF5IADAPEVM with TMDXIDDK379D, ensure that jumpers on these three boards are set correctly as shown in [Figure 13](#), [Figure 16](#), and [Figure 17](#). When instructed to connect motor, connect the motor terminal to connector terminal and motor's encoder header to QEP connector TMDXIDDK379D listed in [Table 6](#). The master node system setup as shown in [Figure 28](#).

Figure 28. TMDXCNC28388D and TMD5FSIADAEVM Assembly with TMDXIDDK379D

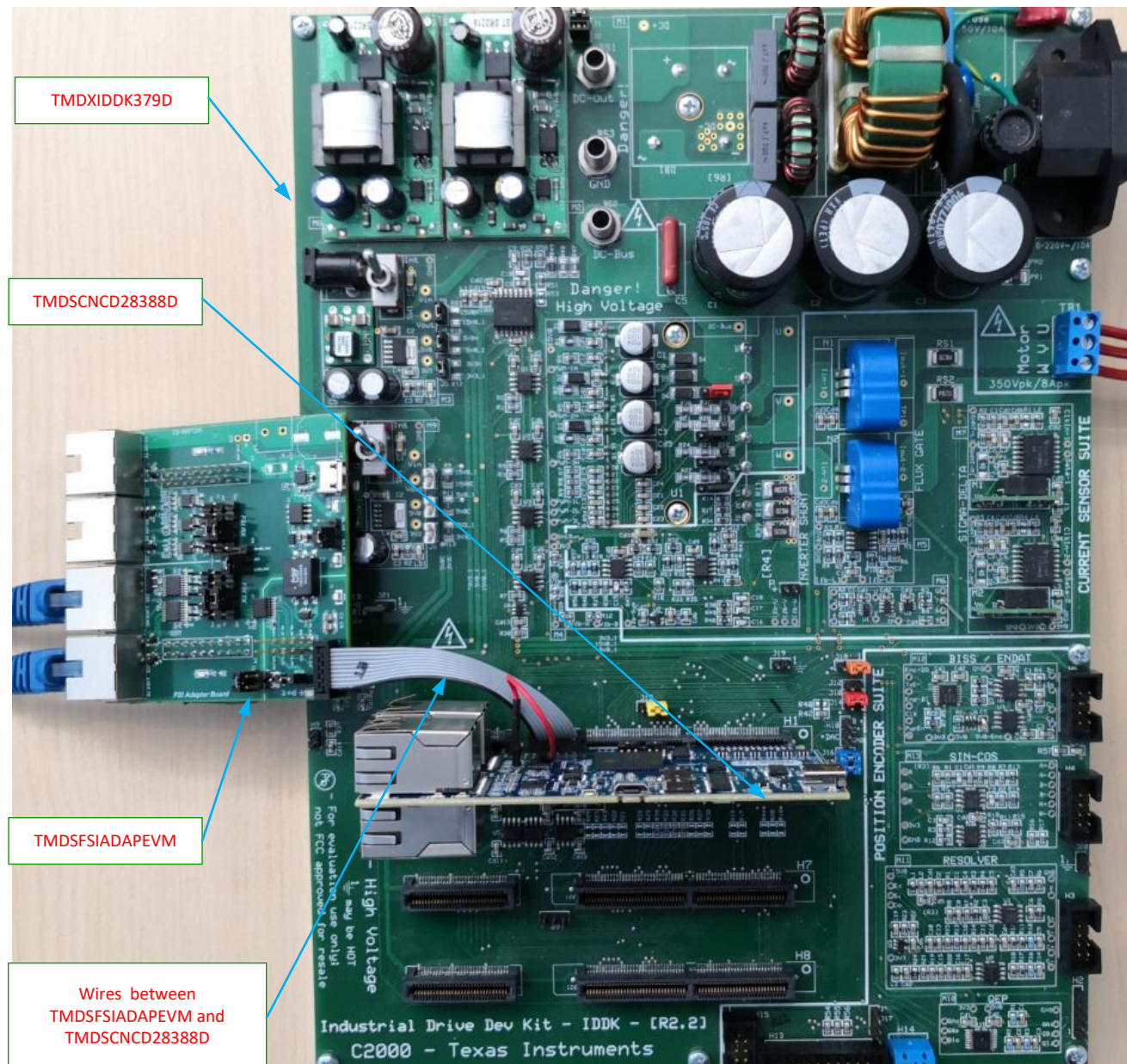
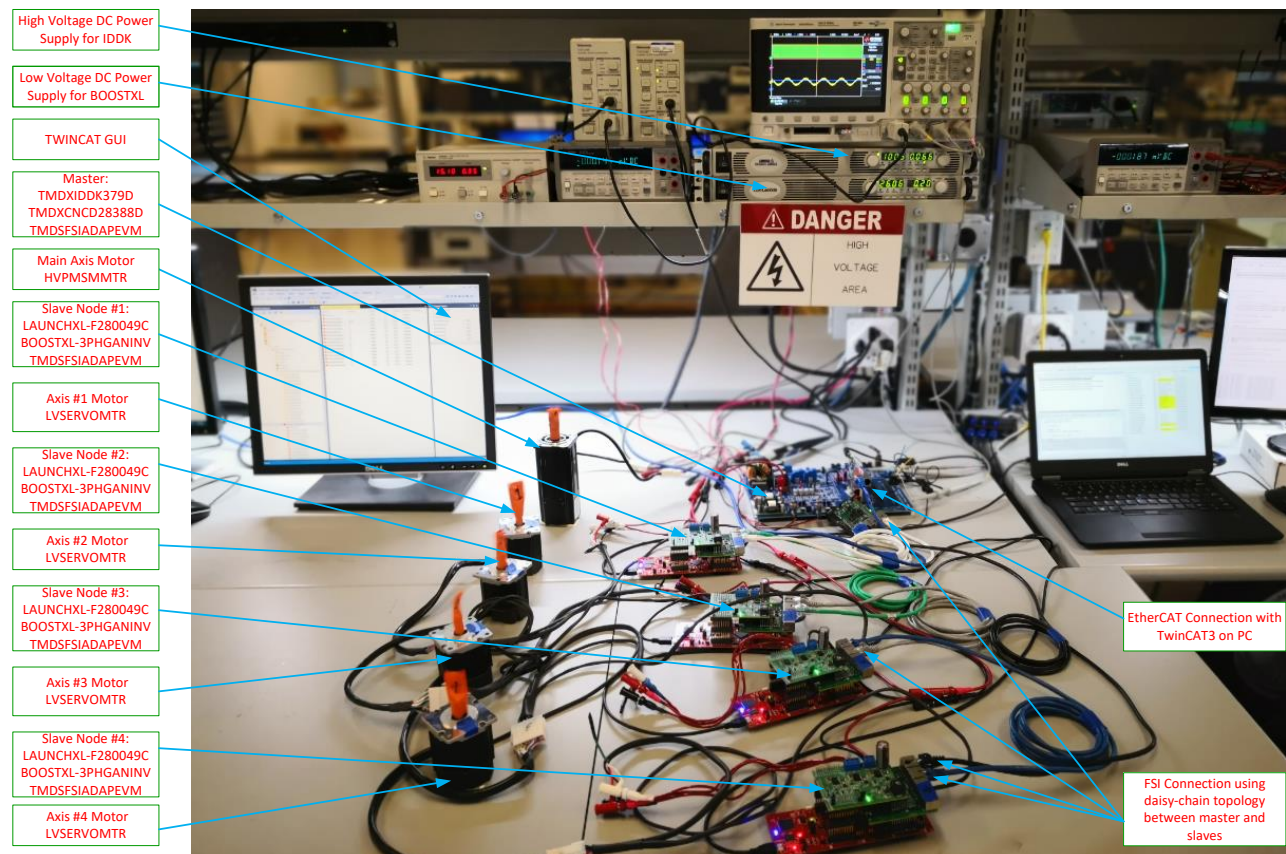


Table 6. Motor Power Lines and Encoder Wires Connections for Master Node

Pin No.	Signal	Pin No.	Signal	Color
TB1 on TMDXIDDK379D		HVPMSMMTR Motor Connector		
1	U	1	U	Red
2	V	2	V	Black
3	W	3	W	White
H2 on TMDXIDDK379D		Encoder Connector		
1	QA	1	A+	Brown
2	QB	2	B+	Yellow
3	QI	3	C+	White
4	+5V	4	PG 5V	Red
5	GND	5	PG 0V	Black

Figure 29 shows the test platform setup for this design. The master and slave nodes of multi-axis drive system are connected over FSI using daisy-chain topology.

Figure 29. Test Platform for Multi-axis Drive over FSI



3.3.2 Test Results

The operation and test result at those build levels (LEVEL1~LEVEL6) for motor drive in main or slave axis are described in application report [Dual-Axis Motor Control Using FCL and SFRA On a Single C2000™ MC](#) in detail.

The following results are tested in the final system by setting the build level in each project separately as below

- multi_axis_master_ctrl_f2838x_cpu1 - define BUILDLEVEL to FCL_LEVEL11
- multi_axis_master_drive_f2838x_cpu2 - define BUILDLEVEL to FCL_LEVEL8
- multi_axis_master_ecat_f2838x_cm - only one build level
- multi_axis_slave_node(n)_f28004x_cpu - define BUILDLEVEL to FCL_LEVEL8

[Figure 30](#) shows the speed curve of each slave axis and main axis using CCS graph tool with datalog function. The result shows there is no significant difference between all the axes.

Figure 30. Experiment of Main Axis and 4 Slave Axes Speed Synchronization

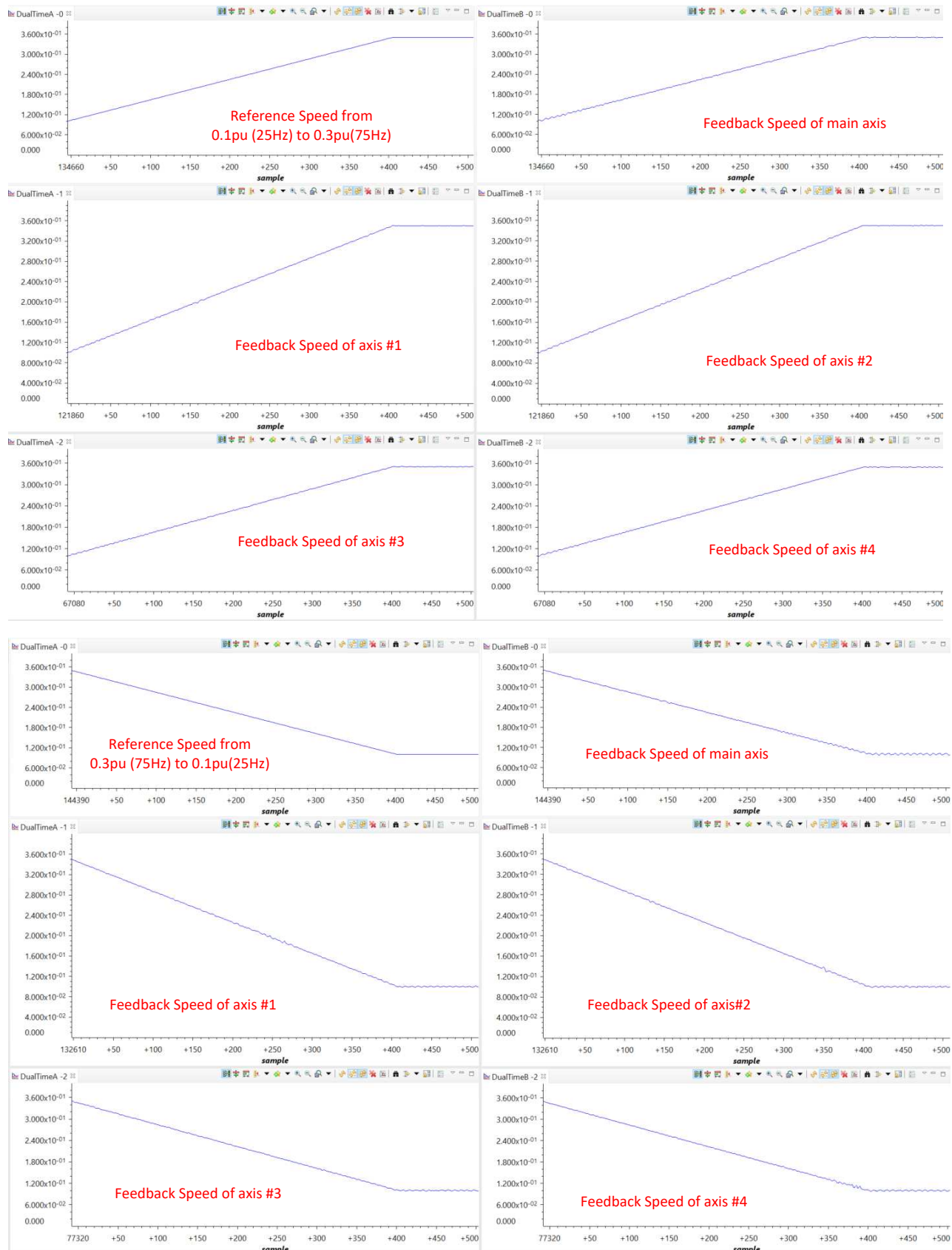


Figure 31 shows the speed curve of a slave axis and main axis, or reference and feedback speed of a motor using buffer DAC and oscilloscope. The result shows there is no significant difference between the two axes, or no significant delay between the reference and real running speed of the motor.

Figure 31. Experiment of Speed Synchronization

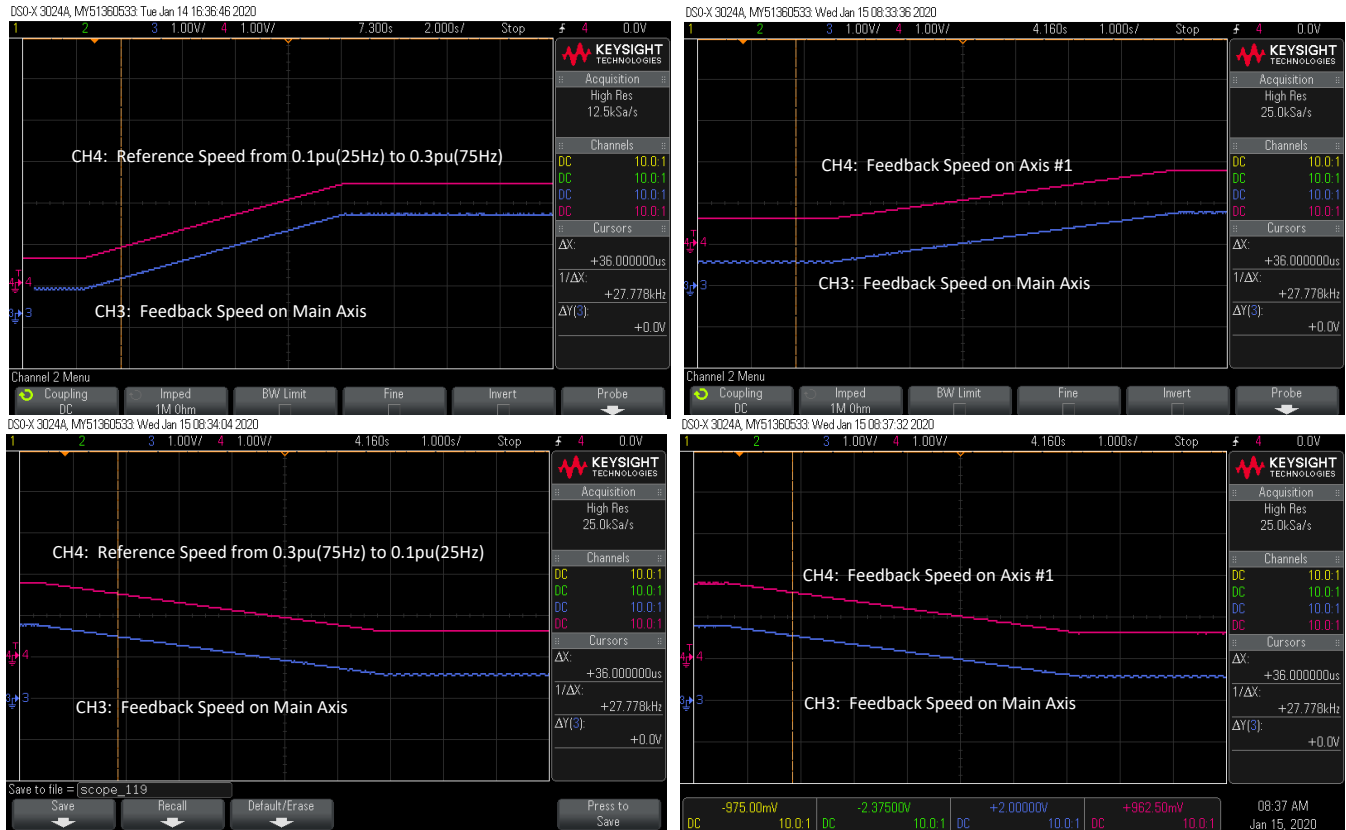


Figure 32 shows the speed curve of a slave axis and main axis, or reference and feedback speed of a motor using buffer DAC and oscilloscope.

Figure 32. The Reference Speed and Feedback Position of Different Motor

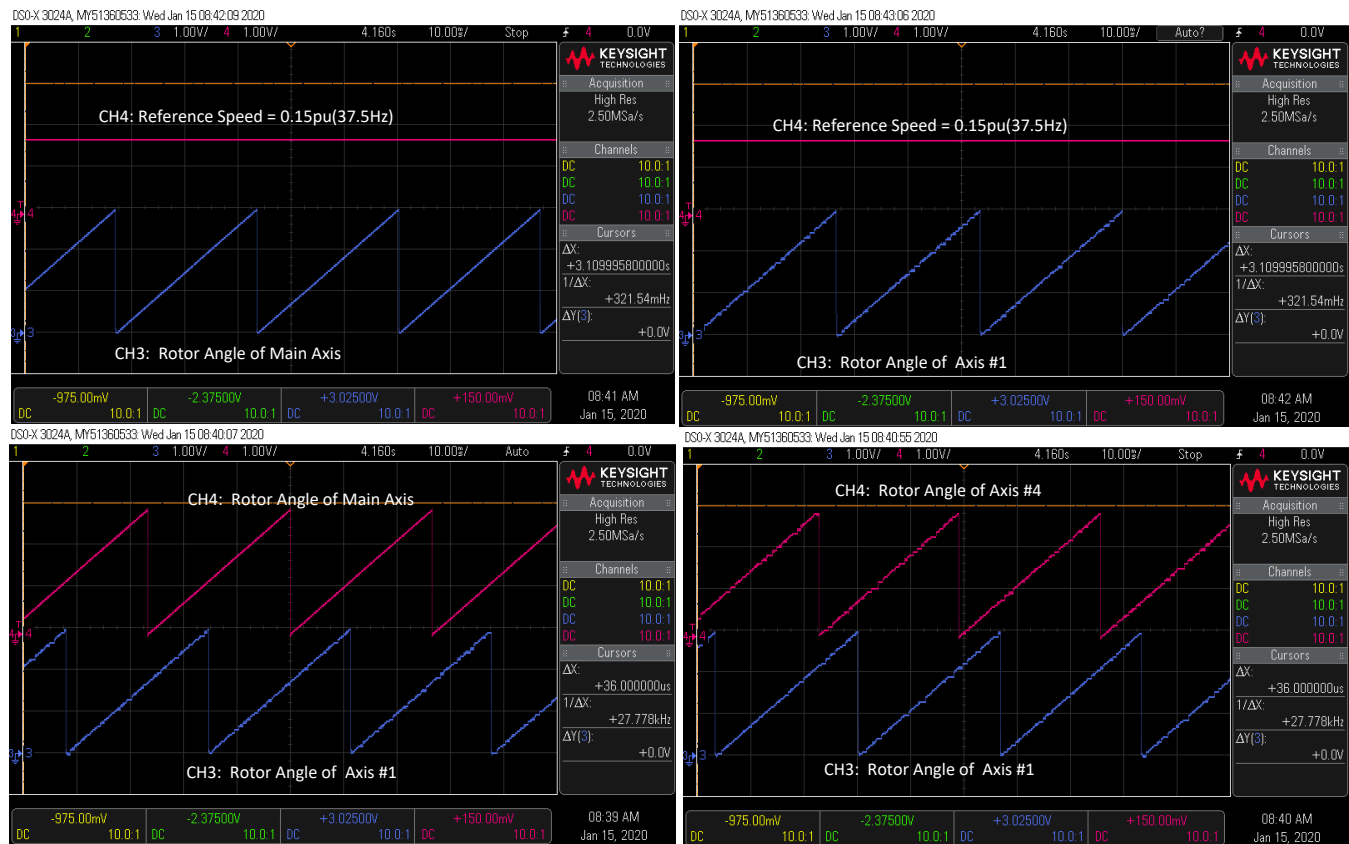


Figure 33 shows a snapshot of the expressions window in CCS. The variable, `motorVars.fclUpdateLatency_us` indicates the PWM latency that is the amount of time elapsed between the feedback sampling and PWM updating. Another variable, `motorVars.focExecutionTime_us` indicates the amount of time executed by the whole FOC algorithm.

Figure 33. PWM latency and FOC execution time

motorVars.lsw	enum <unna...	ENC_CALIBRATION_DONE	
motorVars.ctrlStateUse	enum <unna...	CTRL_RUN	
motorVars.speedWe	float	0.300094098	
motorVars.fclUpdateLatency_us	float	1.02999997	Latency on F2838x CPU2 (200MHz) for main axis drive
motorVars.focExecutionTime_us	float	4.11000013	
motorVars.fclClrCntr	unsigned int	0	
motorVars.focClrCntr	unsigned int	0	
motorVars.fclUpdateLatency_us	float	2.05999994	Latency on F28004x CPU (100MHz) for slave axis drive without FSI interrupt nesting
motorVars.focExecutionTime_us	float	7.69999981	
motorVars.fclClrCntr	unsigned int	0	
motorVars.focClrCntr	unsigned int	0	
motorVars.fclUpdateLatency_us	float	4.42000008	Latency on F28004x CPU (100MHz) for slave axis drive with FSI interrupt nesting
motorVars.focExecutionTime_us	float	13.21	
motorVars.fclClrCntr	unsigned int	0	
motorVars.focClrCntr	unsigned int	0	

Figure 34 shows a snapshot of the expressions window in CCS. These variables indicate the number of transmission times and failures. The result shows that the failure rate is very low after running over a long time.

Figure 34. Long Time Experiment of FSI Communication

Expression	Type	Value			
enableCtrlFlag	unsigned int	1			
fsiRxFrameCntr	unsigned long	0x8415FF97 (Hex)			
fsiTxFrameCntr	unsigned long	0xBAE8577A (Hex)			
fsiRxInt2Received	unsigned long	0			
fsiTxInt2Received	unsigned long	0			
fsiFrameActiveCntr	unsigned lon...	[5780085,7037768,58256...			
[0]	unsigned long	5780056			
[1]	unsigned long	7037725			
[2]	unsigned long	5825602			
[3]	unsigned long	8385494			
fsiFrameTagErrCntr	unsigned lon...	[0,0,0,0]			
[0]	unsigned long	0			
[1]	unsigned long	0			
[2]	unsigned long	0			
[3]	unsigned long	0			
fsiFrameCrcErrCntr	unsigned lon...	[0,0,1,0]			
[0]	unsigned long	0			
[1]	unsigned long	0			
[2]	unsigned long	1			
[3]	unsigned long	0			
fsiFrameDataErrCntr	unsigned lon...	[0,0,0,0]			
[0]	unsigned long	0			
[1]	unsigned long	0			
[2]	unsigned long	0			
[3]	unsigned long	0			
sysVars.ecatCtrlSet	enum <unna...	ECAT_CTRL_DISABLE			
sysVars.ctrlSynSet	enum <unna...	CTRL_SYN_ENABLE			
sysVars.ctrlStateSet	enum <unna...	CTRL_RUN			
sysVars.ctrlModeSet	enum <unna...	CTRL_MODE_SPEED			
sysVars.speedSet	float	0.300000012			
ctrlVars[0].speedWe	float	0.299219728			
ctrlVars[1].speedWe	float	0.299299985			
ctrlVars[2].speedWe	float	0.299899995			
ctrlVars[3].speedWe	float	0.299600005			
ctrlVars[4].speedWe	float	0.299899995			
fsiRxFrameCntr	unsigned long	0x5DA6AB72 (Hex)			
fsiTxFrameCntr	unsigned long	0x8438980E (Hex)			
fsiRxInt2Received	unsigned long	0			
fsiTxInt2Received	unsigned long	0			
fsiFrameActiveCntr	unsigned lon...	[4010388,4864654,40852...			
[0]	unsigned long	4010399			
[1]	unsigned long	4864662			
[2]	unsigned long	4085286			
[3]	unsigned long	5916598			
fsiFrameTagErrCntr	unsigned lon...	[0,0,0,0]			
[0]	unsigned long	0			
[1]	unsigned long	0			
[2]	unsigned long	0			
[3]	unsigned long	0			
fsiFrameCrcErrCntr	unsigned lon...	[3,3,2,3]			
[0]	unsigned long	3			
[1]	unsigned long	3			
[2]	unsigned long	2			
[3]	unsigned long	3			
fsiFrameDataErrCntr	unsigned lon...	[0,0,0,0]			
[0]	unsigned long	0			
[1]	unsigned long	0			
[2]	unsigned long	0			
[3]	unsigned long	0			
sysVars.ecatCtrlSet	enum <unna...	ECAT_CTRL_DISABLE			
sysVars.ctrlSynSet	enum <unna...	CTRL_SYN_ENABLE			
sysVars.ctrlStateSet	enum <unna...	CTRL_RUN			
sysVars.ctrlModeSet	enum <unna...	CTRL_MODE_SPEED			
sysVars.speedSet	float	0.300000012			
ctrlVars[0].speedWe	float	0.300359279			
ctrlVars[1].speedWe	float	0.299899995			
ctrlVars[2].speedWe	float	0.300099999			
ctrlVars[3].speedWe	float	0.300399989			
ctrlVars[4].speedWe	float	0.300199986			

4 Design Files

This reference design is based on the released C2000 development kits and Evaluation Module, which include [LAUNCHXL-F280049C](#), [TMDSCNCD28388D](#), [BOOSTXL-3PHGANINV](#) and [TMDXIDDK379D](#). The latest FSI adapter board (TMDSFISIADAEVM) design files are as shown in subsections, the official release TMDSFISIADAEVM board will be available at a later time.

4.1 Schematics

To download the schematics of TMDSFISIADAEVM, see the design files at [TIDA-02006](#).

4.2 Bill of Materials

To download the bill of materials (BOM) of TMDSFISIADAEVM, see the design files at [TIDA-02006](#).

5 Software Files

To download the software files, see the design files at [C2000WARE-MOTORCONTROL-SDK](#) the multi-axis servo supporting version 3.00.00.00 will be released at a later date.

6 Related Documentation

1. Texas Instruments, [C2000™ Piccolo™ F28004x Series LaunchPad™ Development Kit](#)
2. Texas Instruments, [BOOSTXL-3PhGaNInv Evaluation Module User's Guide](#)
3. Texas Instruments, [TMS320F28388D controlCARD Information Guide](#)
4. Texas Instruments, [TMDSFISIADAEVM FSI Adapter Board User's Guide](#) (released at a later date)
5. Texas Instruments, [DesignDRIVE Development Kit IDDK v2.2.1 - User's Guide](#)
6. Texas Instruments, [DesignDRIVE Development Kit IDDK v2.2.1 - Hardware Reference Guide](#)
7. Texas Instruments, [TMS320F28004x Piccolo Microcontrollers Technical Reference Manual](#)
8. Texas Instruments, [TMS320F2838x Microcontrollers TRM](#)
9. Texas Instruments, [Fast Serial Interface \(FSI\) Skew Compensation](#)
10. Texas Instruments, [Using the Fast Serial Interface \(FSI\) With Multiple Devices in an Application](#)
11. Texas Instruments, [Dual-Axis Motor Control Using FCL and SFRA On a Single C2000™ MCU](#)
12. Texas Instruments, [Dual-Axis Motor Drive Using Fast Current Loop \(FCL\) and SFRA on a Single MCU Reference Design](#)
13. Texas Instruments, [EtherCAT-Based Connected Servo Drive Using Fast Current Loop on PMSM](#)

6.1 Trademarks

C2000, E2E, BoosterPack, LaunchPad, TMS320C2000, Piccolo are trademarks of Texas Instruments. All other trademarks are the property of their respective owners.

7 Terminology

ADC - Analog-to-Digital Convert

CLA - Control Law Accelerator

CM - Cortex-M4 core

CMPSS - Comparator Subsystem Peripheral

CNC - Computer Numerical Control

DAC - Digital to Analog Converter

eQEP - Enhanced Quadrature Encoder Pulse Module

EtherCAT - Ethernet for Control Automation Technology

FCL - Fast Current Loop

FOC - Field-Oriented Control

FSI - Fast Serial Interface

MCU - Microcontroller Unit

PMSM - Permanent Magnet Synchronous Motor

PWM - Enhanced Pulse Width Modulator

SFRA- Software Frequency Response Analyzer

TMU - Trigonometric Mathematical Unit

8 About the Author

YANMING LUO is a Systems Application Engineer in the system solutions team of C2000 at Texas Instruments, where he is responsible for developing reference design solutions for motor-drive applications based C2000 controllers. Yanming has been with TI since 2003 and earned an M.S. degree from Northeastern University, China in 1998.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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, 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 (www.ti.com/legal/termsofsale.html) 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.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2020, Texas Instruments Incorporated