DesignDRIVE IDDK

User's Guide

![Texas Instruments logo]

Contents

List of Figures

Introduction

Trademarks

C2000, Code Composer Studio are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

DesignDRIVE IDDK Platform

The DesignDRIVE is a hardware and software platform that facilitates developing and evaluating solutions for many industrial drive and servo topologies. The DesignDRIVE kit and software offer an easy path to bgin exploring a wide variety of motor types, sensing technologies, encoder standards, and communications networks. The DesignDRIVE kit and software also offer easy expansion to develop with real-time Ethernet communications and functional safety topologies that enable more comprehensive and integrated system solutions.

With the C2000™ DesignDRIVE Software and Kit (IDDK), TI wants to help you spend more time differentiating your product in your core areas and less time evaluating new technologies that will eventually become table stakes in the industry. Using DesignDRIVE can help you deliver a more valuable product to market faster.

Based on the real-time control architecture of TI's C2000 microcontrollers, DesignDRIVE is ideal for the development of industrial inverter and servo drives used in robotics, computer numerical control (CNC) machinery, elevators, materials conveyance, and other industrial manufacturing applications.

This guide explains the steps required to run the motor with the IDDK using the MotorControl SDK. This SDK is a cohesive set of software infrastructure, tools, and documentation designed to minimize C2000 MCU based motor control system development time targeted for various three phase motor control applications. The software includes firmware that runs on C2000 motor control evaluation modules (EVMs) and TI designs (TIDs) which are targeted for industrial drive and other motor control applications.

The following projects are currently available for IDDK based on TMS320F28379x MCU and the MotorControl SDK:

* IDDK_PM_Servo_F2837x: Quick Response Control of PMSM Using Fast Current Loop

The features of this project include the following:

* Position, Speed, and Torque Control Loops based on QEP encoder
* Simultaneous Current Sensing Support for: Fluxgate/Hall, Delta-Sigma, and Shunt

NOTE: Read this guide and the *DesignDRIVE IDDK Hardware Reference Guide* before using the kit.

Ensure you understand the safety measures required to use the kit.

When MotorControl SDK is installed, you can find the Hardware Reference Guide at: *C:\ti\c2000\C2000Ware_MotorControl_SDK_1_00_00_00\solutions\tmdxiddk379d\docs*

The example software project is at: *C:\ti\c2000\C2000Ware_MotorControl_SDK_2_00_00_00\solutions\tmdxiddk379d*

Depending on the MCU used with the kit, the example software may further be located inside the folder named after the MCU. For example, if TMS320F28379D control card (TMDSCNCDF28379D) is used, then the example will be inside the sub-folder *\F2837x*

The basic guideline to develop software using this kit remains the same regardless of the MCU being used. The description in this guide is based on F28379D, but the same will apply to all MCUs that TI may use this kit for in future.

WARNING

TI intends this EVM to be operated in a lab environment only and does not consider it to be a finished product for general consumer use.

TI intends this EVM to be used only by qualified engineers and technicians familiar with risks associated with handling high-voltage electrical and mechanical components, systems, and subsystems.

This equipment operates at voltages and currents that can cause shock, fire, and/or injure you if not properly handled or applied. Use the equipment with necessary caution and appropriate safeguards to avoid injuring yourself or damaging property.

TI considers it the user's responsibility to confirm that the voltages and isolation requirements are identified and understood before energizing the board and or simulation. When energized, do not touch the EVM or components connected to the EVM.


The Hardware Configuration

1.1 The DesignDRIVE Development kit (IDDK)

To experiment with IDDK for digital motor control, you will need following components:

- An IDDK EVM
- A TMDSCNCD28379D control processor
- A USB A to Mini B cable
- A PMSM motor for evaluation
    - The motor is not included with the TMDXIDDK379D.
    - The motor is included with the TMDXIDDK379D-MTR-BNDL bundle.
    - The motor is available standalone from TI eStore. (The part number is HVPMSMMTR.)
- An incremental encoder or resolver (QEP included with motor available on the TI eStore)
- An external, isolated, 15-V power supply for MCU code development (TI recommends a power supply with a barrel connector [a DC-jack])
- An isolated high-voltage DC-power supply
- A PC with Code Composer Studio ™ (version 8 or greater) installed
- Additional instruments such as:
    - An oscilloscope
    - A current sensing probe
    - A digital multimeter (and optionally a function generator)

The experimental setup and connection are illustrated in the following sections. For details about the kit hardware, see the *DesignDRIVE IDDK Hardware Reference Guide*.

The schematic details of the IDDK EVM are available at
C:\ti\c2000\C2000Ware_MotorControl_SDK_<version>\solutions\tmdxiddk379d\hardware <version> here is 2_00_00_00 or above.
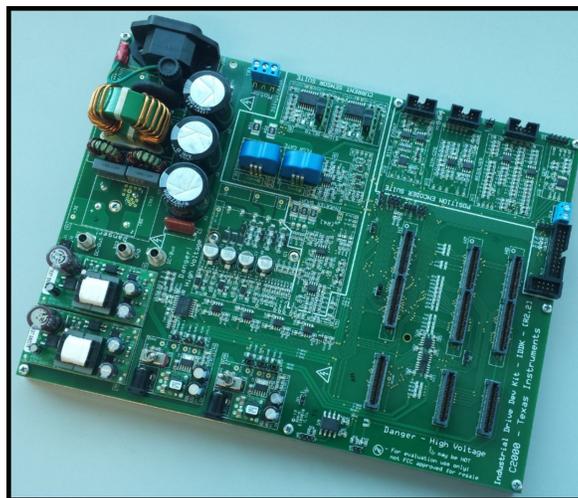
Figure 1-1. DesignDRIVE Kit (IDDK)

1.2 The Hardware Layout of IDDK

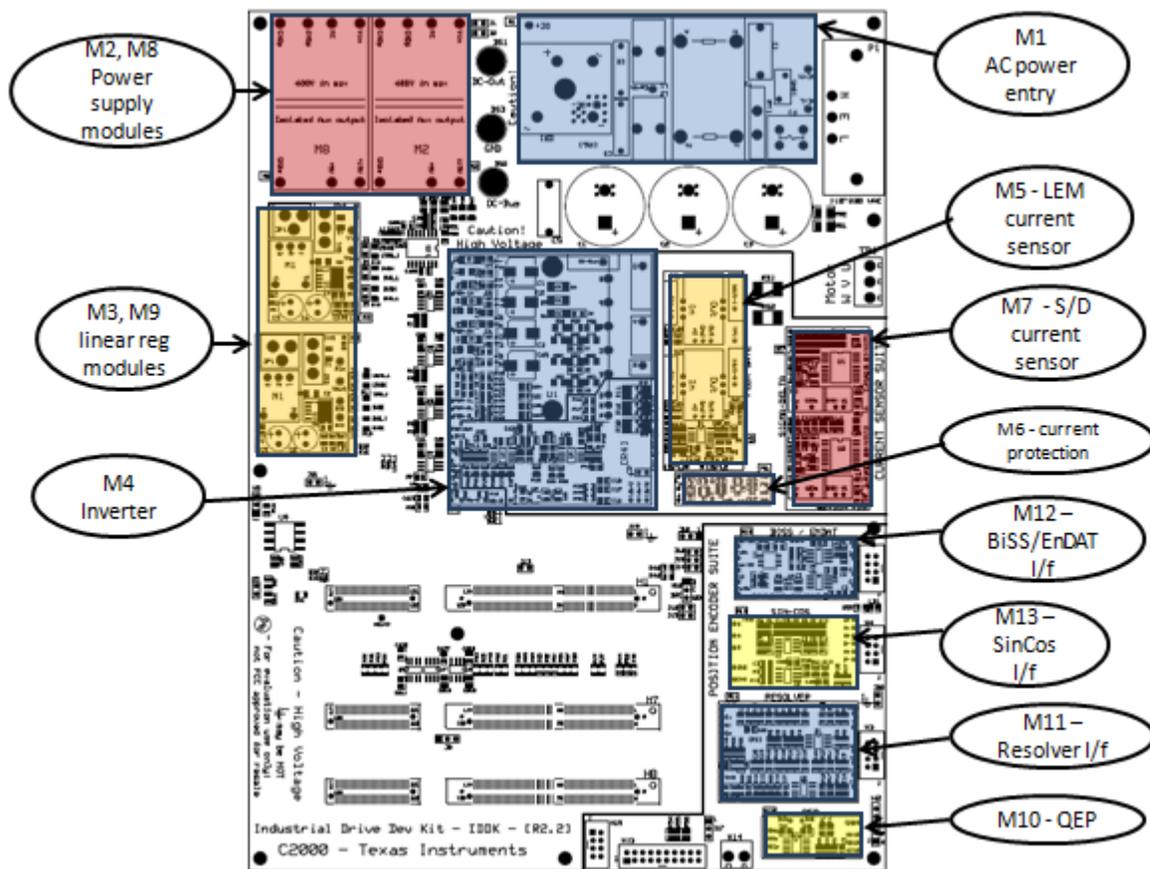Figure 1-2 shows the IDDK that TI designed by integrating various function specific macro blocks.

Figure 1-2. Layout of IDDK EVM With Functional Macros

The following list provides a description of each block:

*   [Main] – Processor slots, jumpers, interprocessor communications, and digital-to-analog converters (DACs) and sections not covered by other macros
*   [M1] – AC-power entry rectifies AC power from the wall/mains power supply. The output of this block can then be the DC bus for the inverter.
*   [M2] – Auxiliary power supply–1: a 400-V to 5-V and 15-V module that can generate 15-V and 5-V for the inverter section of the board from rectified AC power. Through the proper jumper settings, this output of this block can provide 15 V to the entire board.
*   [M8] – Auxiliary power supply–2: a 400-V to 5-V and 15-V module that can generate 15-V and 5-V for the control section of the board from rectified AC power. Through the proper jumper settings, this block can provide 15 V to the entire board.
*   [M3] – DC-power entry 1 generates 15 V, 5 V, and 3.3 V for the HV section of the board from DC power fed through the DC-jack using an external or [M2] power supply. Through the proper jumper settings, this block can power the entire board.
*   [M9] – DC-power entry 2 generates 15 V, 5 V, and 3.3 V for the control section of the board from DC power fed through the DC-jack using an external or [M8] power supply. Through the proper jumper settings, this block can power the entire board.
*   [M4] – A 3-phase inverter that enables control of high voltage 3-phase motors
*   [M5] – A flux gate current sensor module that provides isolated voltage feedback of motor phase currents V and W

- [M6] – An overcurrent protection module that generates a TRIP signal to shutdown the inverter in the event of overcurrent through the motor
- [M7] – A sigma-delta interface module that provides motor phase current feedback to the CPU for digital control of motor
- [M10] – An interface module that translates 5-V logic signals from an incremental encoder into 3.3 V for the C2000 and its QEP peripheral
- [M11] – An analog interface module to work with the Resolver position sensor
- [M12] – A digital interface module to work with a BISS or EnDAT position encoder
- [M13] – An analog interface module to work with the Sin/Cos position encoder

Each component is referenced with their macro number in the brackets followed by a dash and the reference number.

The following is an example of this reference format:

- [M3] - J1 refers to jumper J1 in the macro M3.
- [Main] - J1 refers to jumper J1 on the board but outside of the macro blocks defined previously.

1.3 The IDDK Power Supplies

The IDDK has a low-voltage domain represented by the controller and a high-voltage domain represented by the rectifier and the inverter.

1.3.1 The Low-Voltage Power Domain

This domain represents the 15 V, 5 V, and 3.3 V to power the MCU, logic, sensing, and driver circuits on the IDDK.

The power input for this domain can be provided using the following methods:

- Connecting an isolated 15-V DC-power supply to the DC-jack ([M3]–JP1/[M9]–JP1) on the DC-power entry macro.
- Using an auxiliary power supply module ([M2]/M[8]) on the board that can generate 15 V and 5 V DC from the high-voltage DC link.

> **NOTE:** Because the input voltage range of this module is from 90 V to 400 V, the auxiliary modules are unsuitable if the DC-bus voltage goes below 90 V during the tests. In such cases, TI recommends using the first method.

- Configuring [M3] and [M9], which are identical macros that provide the same set of output voltages, to power the entire controller or the gate drives of the inverter. For more information, see the *DesignDRIVE IDDK Hardware Reference Guide*.

For the default configuration of the kit:

- Set up [M9] as the low-voltage power source.
- Both control GND and power GND planes are separated into COLD and HOT GNDs, respectively. In the previous release of IDDK [R2.2], they were tied together making them all HOT.
- Do not use shunt current and voltage sensing.

For applications requiring a HOT control GND:

- You can use the shunt current sensing method. For more information, see the *Power Supplies and GND Plane Configurations* section in the *DesignDRIVE IDDK Hardware Reference Guide*.

NOTE: The control GND and power GND can be separate or tied together. If control GND and power GND are tied together, the control GND is HOT. If they are not tied together, the control GND is COLD and will need a separate control power supply for the HOT and COLD sides.

For your safety and convenience during general code development, testing, and verification, use an external isolated 15-V power supply to power the control section of the board through DC-power entry macro [M9].

1.3.2 The High-Voltage Power Domain

This domain represents the DC-link section powering the inverter that generates three-phase voltages for the motor.

The following are ways to provide this voltage to the inverter:

- Connect an external isolated variable DC-power supply (maximum 350 V) using banana jacks to [Main]-BS2 and [Main]-BS3 of IDDK. TI recommends using this power supply during experiments. To power the auxiliary power supply modules [M2]/[M8], connect [Main]-BS3 and [Main]-BS1.
- Connect the kit to AC Mains through P1 (110 V or 220 V AC). An onboard rectifier converts AC to DC. For your safety, TI recommends using a variable AC transformer (variac) and isolator when using this power source.

NOTE: Keep power supplies at zero unless directed to energize.

The 3-phase induction motors are typically rated at 220-V AC. You need a 320-V DC-bus voltage. When using 110-V AC-power source to generate the DC bus for the inverter (150 V), the motor can run properly to a certain speed and torque range without saturating the PID regulators in the control loop. You can connect directly to a DC-power supply.

WARNING

The ground planes of both the power domains can be same or different depending on hardware configuration.

Meet proper isolation requirements before connecting any test equipment with the board to ensure the safety of yourself and your equipment.

Review the GND connections in the *DesignDRIVE IDDK Hardware Reference Guide* before powering the board.

Setting Up the Test Hardware

2.1 Bringing Up the Board

The experiment uses a separate control power supply for the CPU and interface circuits and another control power supply for inverter gate drive circuits. For more information, see entry 2 of the *Power Supply Connection Configuration* table in the *DesignDRIVE IDDK Hardware Reference Guide*.

For setting up the experimental hardware, perform the following steps:

1. Populate jumpers [Main]-J3, [Main]-J4, and [Main]-J5 in front of macro M3.
2. Mount jumpers [Main]-J6, [Main]-J7 and [Main]-J8 in front of macro M9 and resistors [Main]-R9, [Main]-R11 and [Main]-R13 and GND plane resistor R15 (on the bottom side). For more information, see the *Various GND Planes on the Bottom Side of Board* and *Default Connection of Various GND Planes* figures in the *DesignDRIVE IDDK Hardware Reference Guide*.
3. Ensure that resistors [Main]-R8, [Main]-R10 and [Main]-R12 are not populated.
4. Unpack the TMDXCNCD28379D control card.
5. Slide the card into the connector slot of [Main]-H1. (Push down using even pressure on both ends of the card until it cannot slide further. To remove the card, spread open the retaining clips and pull the card out, applying even force at the edges.)
6. Connect a USB cable to connector J1 on the control card. (The control card isolates the JTAG signals between the C2000 device and the computer. LED D2 on the control card should light.)
7. Ensure that toggle switch [M9]-SW1 is in Int position.
8. Connect an isolated 15-V DC-power supply to [M9]-JP1.
9. Turn on toggle switch [M9]-SW1.

> **NOTE:** More LEDs on the control card light up indicating that the control card is receiving power from the board.

10. Ensure that toggle switch [M3]-SW1 is in the Int position.
11. Connect an isolated 15-V DC-power supply to [M3]-JP1.
12. Turn on toggle switch [M3]-SW1. ([M3]-LD1 should turn on.)

> **NOTE:** Connect the motor to the [Main]-TB1 terminals only after finishing the first incremental build.

13. Apply the DC-bus power only when the guide instructs. Two options exist to get DC-bus power:
    - To use an external, variable DC-power supply, do the following:
        a. Set the power-supply output to zero.
        b. Connect [Main]-BS2 and [Main]-BS3 to the + and − terminals of the DC-power supply, respectively. (See Figure 2-1.)
    - To use AC-Mains power, do the following:
        a. Connect [Main]-BS1 to [Main]-BS2 using a banana plug cord.
        b. Connect the end of the AC-power cord to [Main]-P1.
        c. Set the output of the variac to zero.
        d. Connect the variac to the wall supply through an isolator. (See Figure 2-2.)
        e. Connect the other end fo the AC-power cord to the output of the variac.

Figure 2-1. Connection Diagram With External Variable DC-Power Supply Providing DC-Bus Voltage
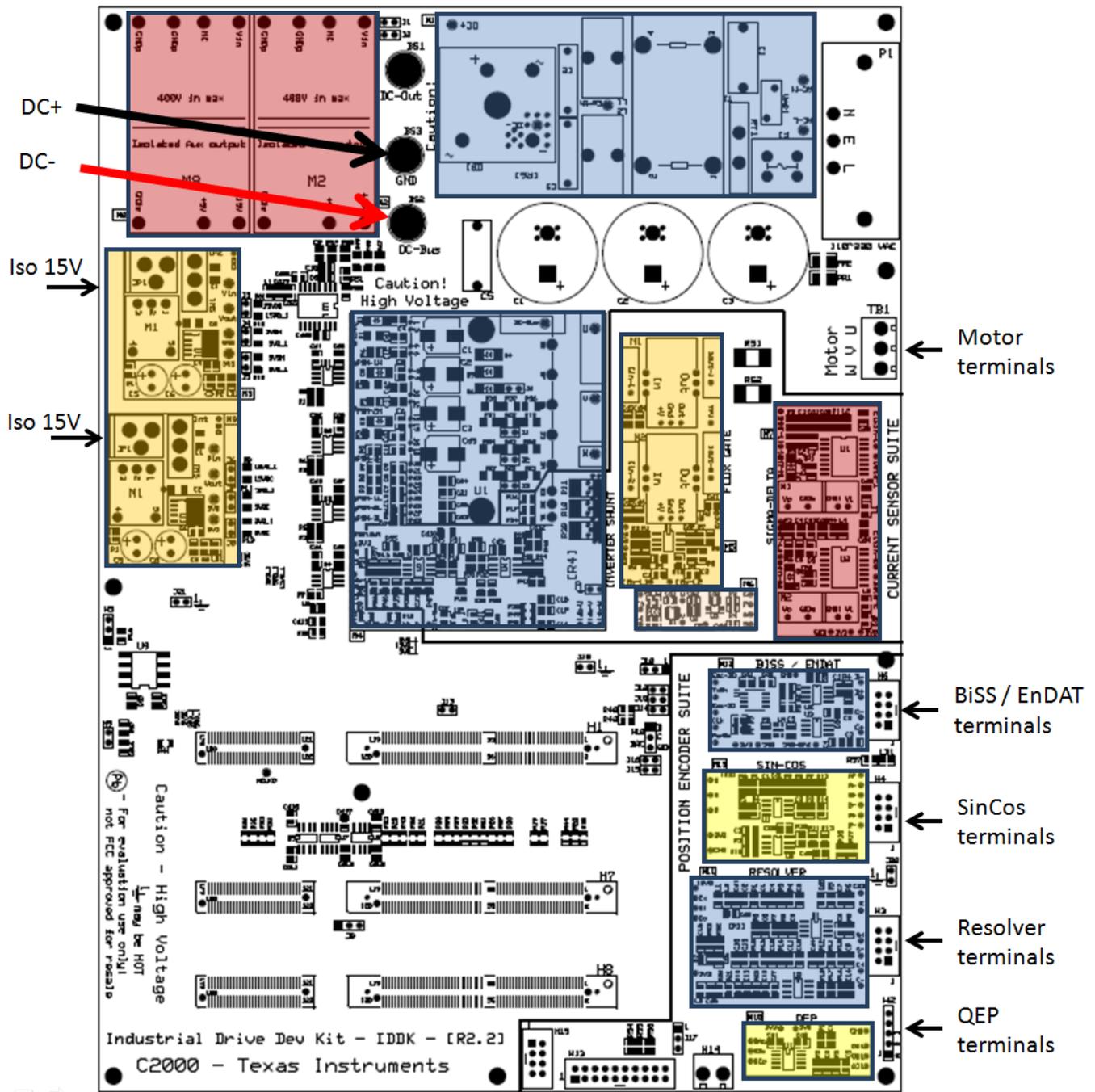
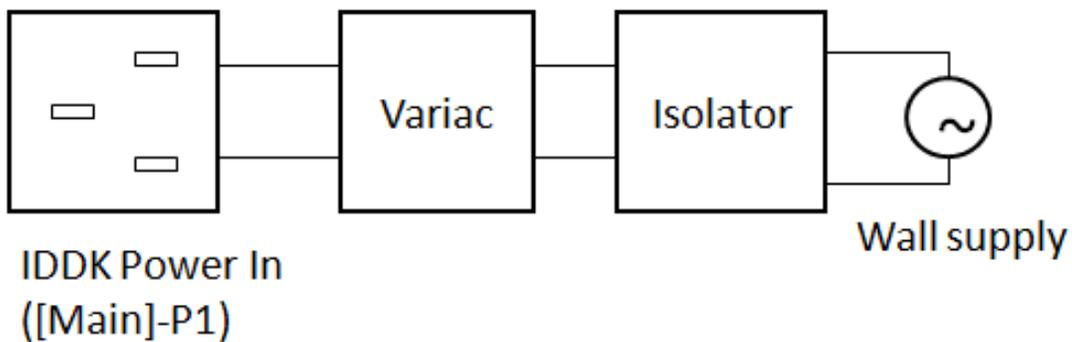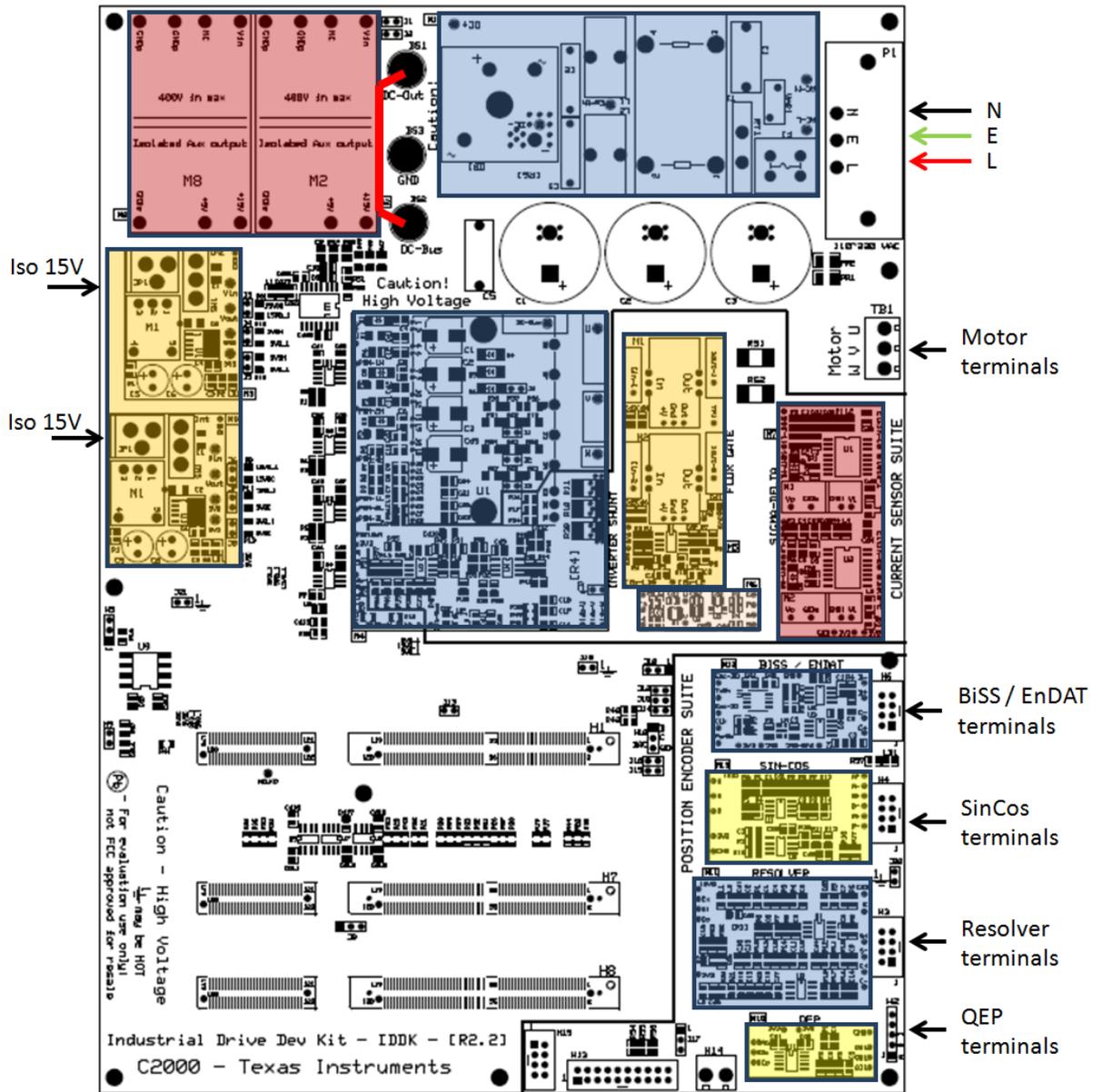**Figure 2-2. Connection Diagram With AC Input**

Setting Up the Software for IDDK Projects

3.1 Installing Code Composer and MotorControl SDK

1. If not already installed, download the installer for Code Composer Studio v8.x or later from http://www.ti.com/tool/CCSTUDIO and install it by following the prompts.

2. Likewise, download the installer for MotorControl SDK from http://www.ti.com/tool/C2000WARE-MOTORCONTROL-SDK and install it by following the prompts.

NOTE: Allow the installer to download and update any automatically-checked software for C2000.

3.2 Setting Up Code Composer Studio to Work With TMDXIDDK379D

The procedure given here is a basic guideline to develop software using this kit regardless of the C2000 MCU being used. The description in this guide is based on F28379D and the same will apply to all MCUs that TI may use this kit for in future. Any deviation to this will be appropriately addressed when required.

1. Open Code Composer Studio (Version 8 or later).

NOTE: When Code Composer Studio opens, the workspace launcher may appear that requests the selection of a workspace location on the hard drive where all the settings for the IDE, that is, which projects are open, what configuration is selected, and so forth. are saved. This can be anywhere on the disk. The location mentioned in the following steps is for reference. If this is not your first time to run Code Composer, the dialog in the following steps may not appear).

---

2. Click *Browse*.
3. Create the following path: *C:\c2000 projects\CCSv9_workspace*.
4. Uncheck the box labeled *Use this as the default and do not ask again*.
5. Click *OK*.
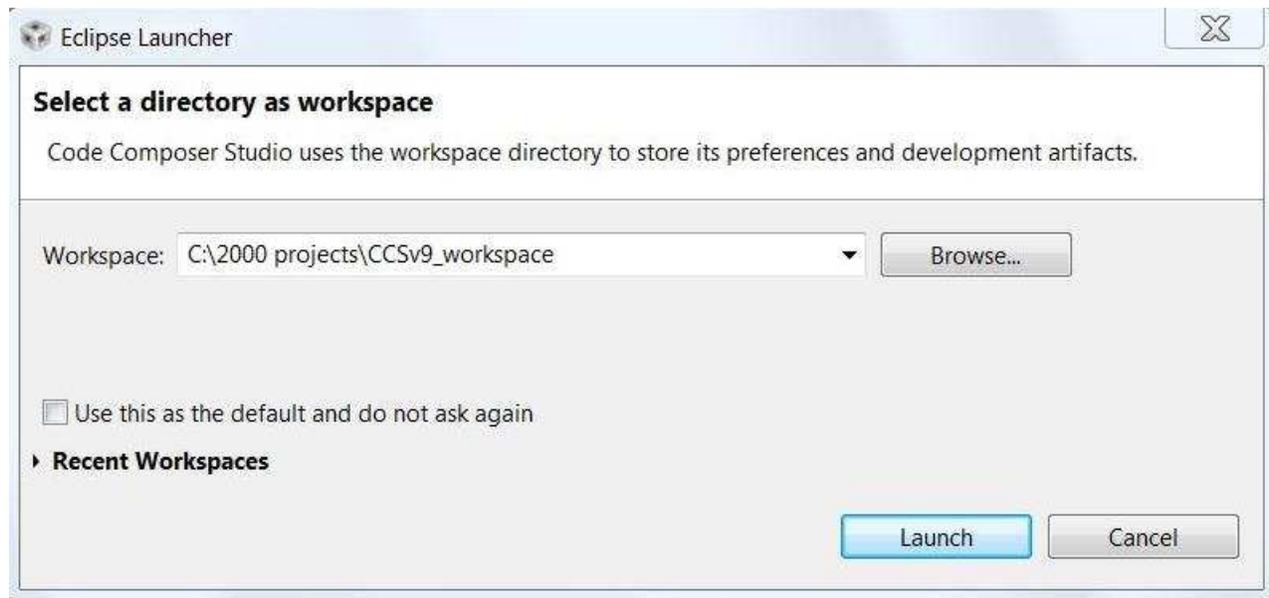
Figure 3-1. Workspace Launcher

> **NOTE:** A *Getting Started* tab opens with links to tasks for creating a new project, importing an existing project, and watching a tutorial on CCS.

> **NOTE:** You must set up the *Target Configuration* to configure Code Composer Studio to recognize the MCU to which it must connect. The target configurations are already set up in *TMS320F28379D.ccxml* in the project files.

6. If you intend to use the configuration file included in the project files, skip to step 28. To create your own configuration file, proceed to the next step.

7. Click *View* to set a new configuration file.

8. Click *Target Configuration* to open the *Target Configuration* window.

9. Click [icon] in the *Target Configuration* window.

10. Name the new configuration file based on the target device.

> **NOTE:** If *Use shared location* is checked, store this configuration file in a common location by CCS for use by other projects.

11. Click *Finish*.

> **NOTE:** A new tab opens. (See Figure 3-2.)

12. Select the connection.

13. Enter the connection. (Use the TI XDS100v2 USB Emulator or the TI XDS100v2 USB Debug Probe.)

14. Select the device.

15. Enter the device (for example, the C2000 MCU on the control card or TMS320F28379D).
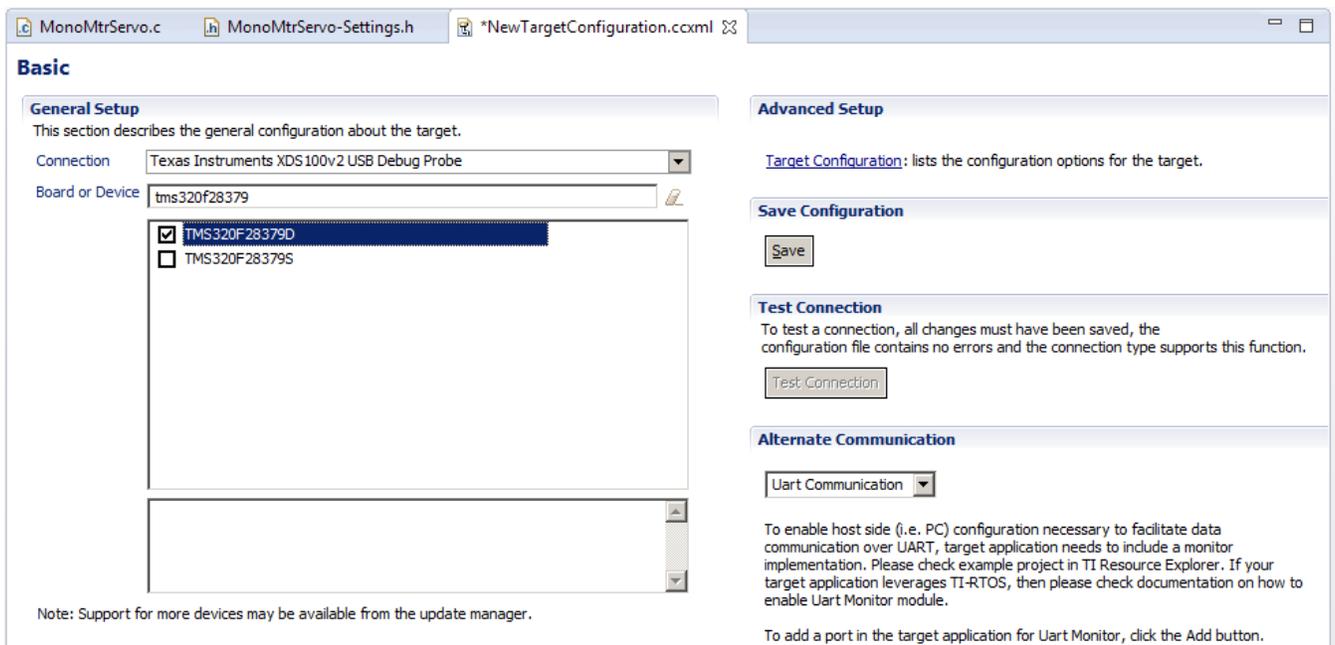
16. Click *Save* and close.

Figure 3-2. Configuring a New Target

17. Click *View*.
18. Click *Target Configurations*.
19. Find the file that you created in steps 7 to 16 in the *User Defined* section.
20. Right-click this file.
21. Select *Set as Default*.

NOTE: Alternatively, to use the *Target Configuration* file in the SDK project at any time, proceed to step 22 or skip to step 28.

---

22. Click *View*.
23. Click *Target Configurations*.
24. Click *Expand Projects*.
25. Select *fcl_qep_2837x_tmdxiddk*.
26. Right-click *TMS320F28379D.ccxml*.
27. Click *Set as Default*. (This tab also lets you to reuse existing target configurations and link them to specific projects. This step completes the target configuration setup.)
28. To add the motor control projects into the current workspace, click *Project*.
29. Click *Import CCS Project*.
30. To select the IDDK project, navigate to:*c:\ti\c2000\C2000Ware_MotorControl_SDK_<version>\solutions\tmdxiddk379d* (see Figure 3-3).
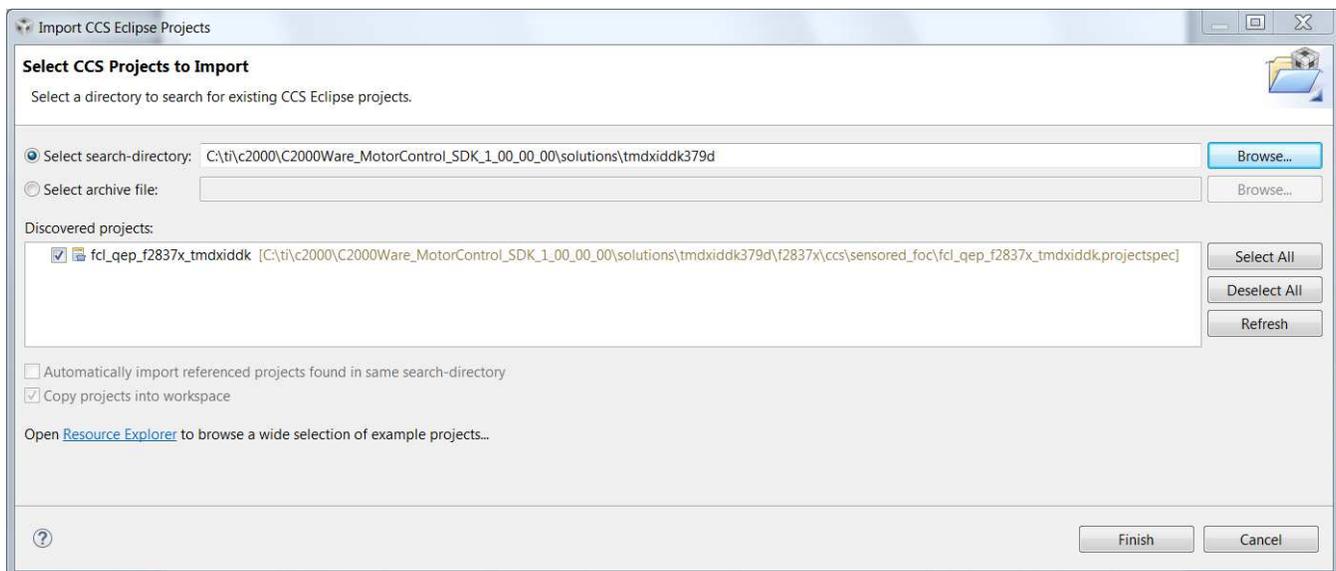
Figure 3-3. Adding an IDDK Project to Workspace

31. If there are multiple projects in this directory, do the following:

a. Click the projects to import. (Figure 3-3 shows one project in the directory.)

b. Click *Finish*. (This copies all the selected projects into the workspace.)

3.3 Configuring a Project

NOTE: If this is your first time using Code Composer, xds100v2-F2837x must be the default target configuration.

---

1. To verify if *TMS320F28379D.ccxml* is the default configuration file, view this file in the expanded project structure and check to see if [Active/Debug] status is marked next to it.

2. Click *Target Configurations* to edit existing target configurations or change the default or active configuration.

3. Right-click the name of the target configuration to link a target configuration to a project in the workspace.

4. Select *Link to Project*.

NOTE: The project can be configured to create code and run in either flash or RAM. For lab experiments, use the RAM configuration most of the time and use the flash configuration for production.

---

5. Right-click the project you wish to work with if there are multiple projects in the workspace.

6. Click *View*.

7. Select *Active Build Configuration*.

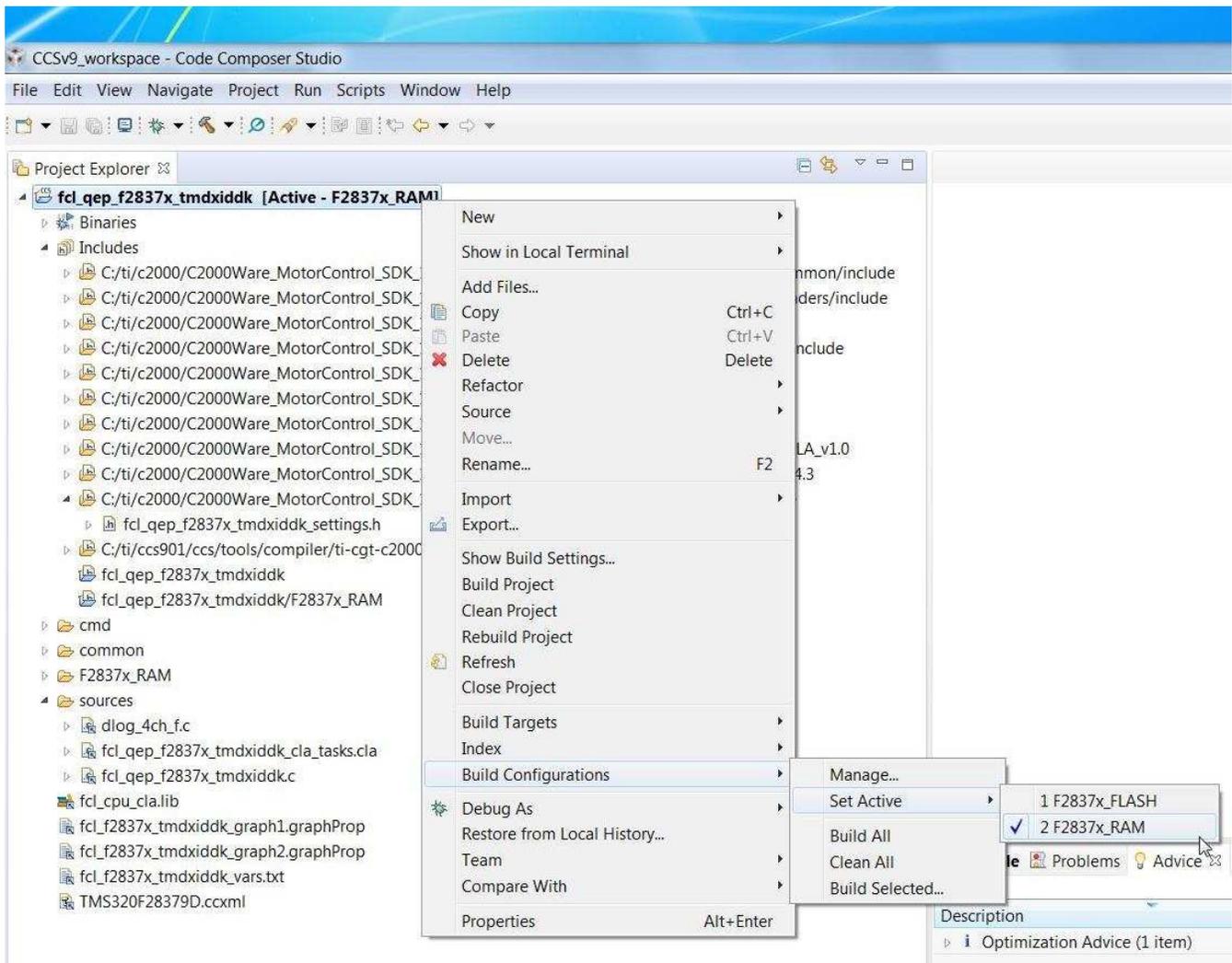8. Click the *F2837x_RAM* configuration. (See Figure 3-4.)

Figure 3-4. Selecting the F2837x_RAM Configuration

3.4 Building and Loading the Project

The TI motor control software has incremental builds where different components and macro blocks of the system are pieced together, one-by-one, to form the entire system. (This build structure helps to easily debug and understand the system.)

1. Open the file *fcl_qep_f2837x_tmdxiddk_settings.h* from the CCS Edit Perspective.

2. Set the BUILDLEVEL to LEVEL1.

3. Save this file.

> **NOTE:** After testing build 1, redefine BUILDLEVEL to LEVEL2 for the next level tests. Increment the BUILDLEVEL and perform the tests for each level until all builds are complete.

4. Open the *fcl_qep_f2837x_tmdxiddk.c* file.

5. Navigate to the *motorControlISR()* (or *MainISR()* in general) function.

6. Locate the following piece of code in incremental build 1.

```
DlogCh1=rg1.Out;
DlogCh2=svgen1.Ta;
DlogCh3=svgen1.Tb;
DlogCh4=svgen1.Tc;
```

7. Confirm that the datalog buffers point to the correct variables.

> **NOTE:** These datalog buffers are large arrays that contain value-triggered data that can be displayed in a graph. In other incremental builds, variables may be put into this buffer to be graphed. The previous code is an example where the datalog buffers are pointed to the space vector generator module.

8. Right-click on the project name.

9. Click *Rebuild Project*.

10. Ensure the console window is error free.

11. When the build completes, click ⚙(Debug).

> **NOTE:** A window as shown in Figure 3-5 may pop up if this is the first time opening the debug
> window. The window requests that you select one of the two CPUs in F28379x with which to
> connect.
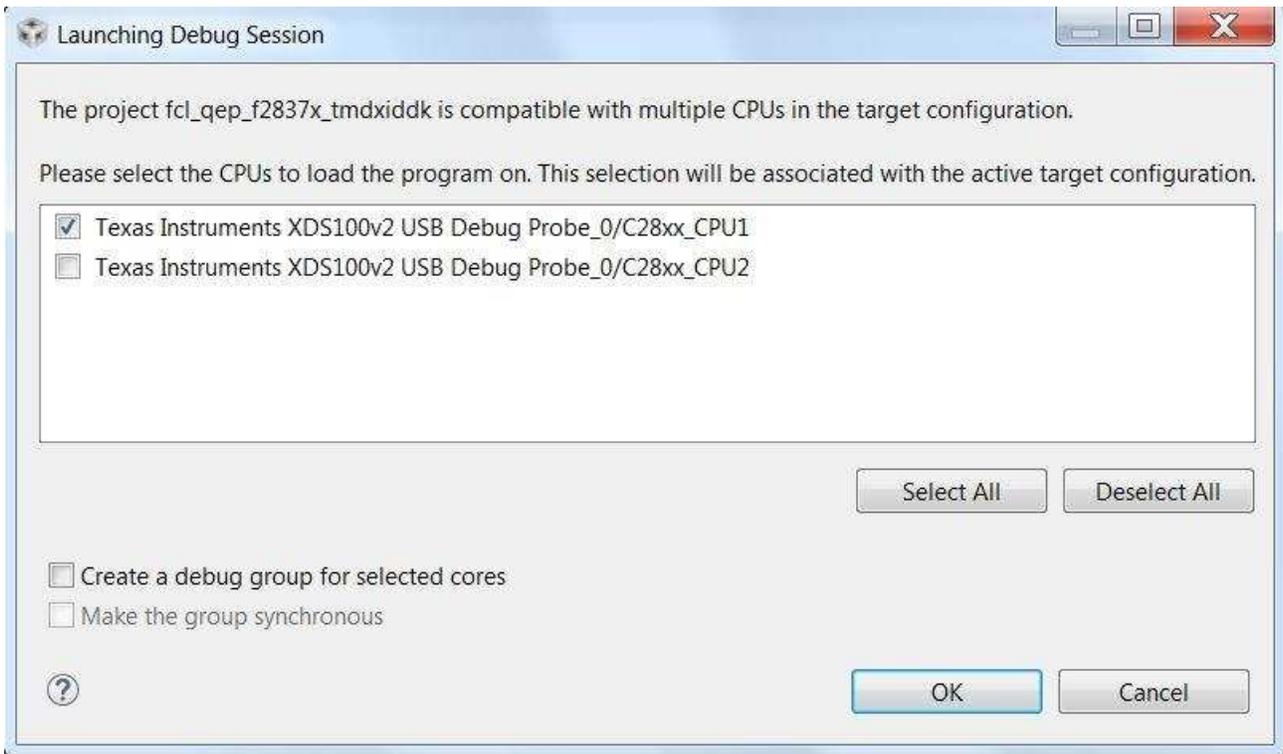
Figure 3-5. Selecting the CPU to Connect

12. Click the box next to CPU1.

> **NOTE:** The IDE automatically connects to the target.

13. Load the output file into the device.
14. Change to the debug perspective.
15. Click *Tools*.
16. Click *Debugger Options*.
17. Click *Program*.
18. Click *Memory Load Options*.
19. Check *Reset the target when loading or restarting the program* to enable the debugger to reset the processor each time it reloads the program,.
20. Click *Remember My Settings* to make this setting permanent,.
21. Click 🔧(Enable silicon real-time mode).

> **NOTE:** Clicking this mode automatically selects 🔧 (Enable polite real-time mode). This action lets
> you edit and view variables in real-time. Do not reset the CPU unless you disable these real-
> time options.

22. If a message box appears, select yes to enable the debug events.

NOTE: This action sets bit 1 (the DGBM bit) of status register 1 (ST1) to 0. The DGBM is the debug enable mask bit. When the DGBM bit is set to 0, memory and register values can pass to the host processor to update the debugger windows.

3.5 Setting Up the Watch Window and Graphs

1. Click *View*.
2. To open a watch window to view the variables used in the project, click *Expressions* on the menu bar.
3. Add variables to the watch window (see Figure 3-6).
4. To select a number format for the variable, do the following:
   a. Right-click the variable.
   b. Choose the format (see Figure 3-6).

NOTE: The watch window uses the same number format with which the variable is declared.
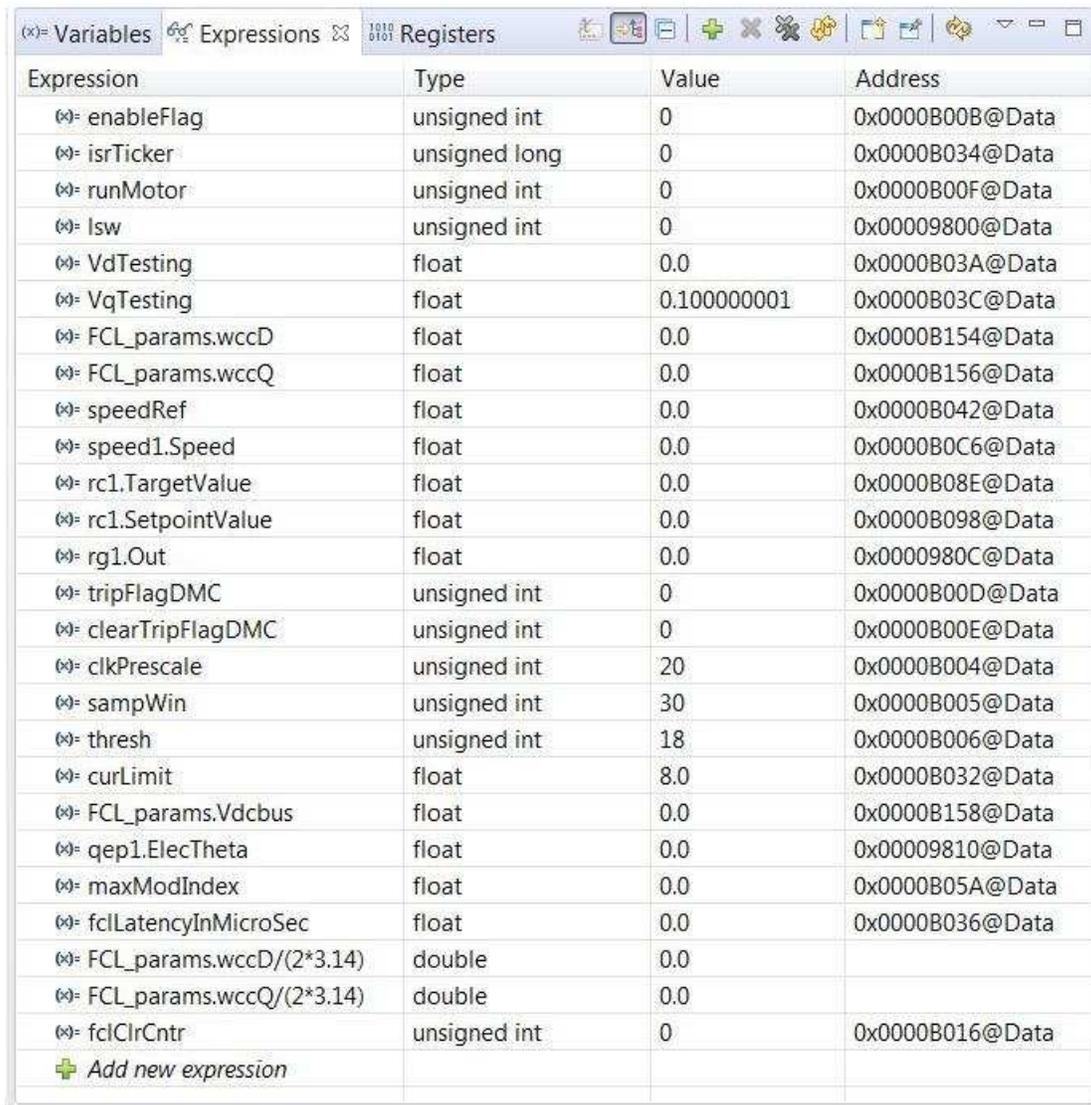
Expression	Type	Value	Address
enableFlag	unsigned int	0	0x0000B00B@Data
isrTicker	unsigned long	0	0x0000B034@Data
runMotor	unsigned int	0	0x0000B00F@Data
lsw	unsigned int	0	0x00009800@Data
VdTesting	float	0.0	0x0000B03A@Data
VqTesting	float	0.100000001	0x0000B03C@Data
FCL_params.wccD	float	0.0	0x0000B154@Data
FCL_params.wccQ	float	0.0	0x0000B156@Data
speedRef	float	0.0	0x0000B042@Data
speed1.Speed	float	0.0	0x0000B0C6@Data
rc1.TargetValue	float	0.0	0x0000B08E@Data
rc1.SetpointValue	float	0.0	0x0000B098@Data
rg1.Out	float	0.0	0x0000980C@Data
tripFlagDMC	unsigned int	0	0x0000B00D@Data
clearTripFlagDMC	unsigned int	0	0x0000B00E@Data
clkPrescale	unsigned int	20	0x0000B004@Data
sampWin	unsigned int	30	0x0000B005@Data
thresh	unsigned int	18	0x0000B006@Data
curLimit	float	8.0	0x0000B032@Data
FCL_params.Vdcbus	float	0.0	0x0000B158@Data
qep1.ElecTheta	float	0.0	0x00009810@Data
maxModIndex	float	0.0	0x0000B05A@Data
fclLatencyInMicroSec	float	0.0	0x0000B036@Data
FCL_params.wccD/(2*3.14)	double	0.0	
FCL_params.wccQ/(2*3.14)	double	0.0	
fclClrCntr	unsigned int	0	0x0000B016@Data
Add new expression			

Figure 3-6. Configuring the Expressions Window

5. Alternately, you can import a group of variables into the Expressions window by doing the following:
    a. Right-click in the Expressions window.
    b. Click *Import*.
    c. Navigate to the debug directory of the project*C:\ti\c2000\C2000Ware_MotorControl_SDK_1_00_00_00\solutions\tmdxiddk379d\f2837x\debug*
    d. select *fcl_f2837x_tmdxiddk_vars.txt* file containing the list of variables.
    e. Click *OK* to import the variables. (See Figure 3-6.)

6. Click ⚙ (Continuous Refresh) in the watch window.

NOTE: This action enables the window to run with real-time mode. If you click the down arrow in this watch window, you can select *Customize Continuous Refresh Interval* and edit the refresh rate of the watch window. Choosing an interval that is too fast can affect performance.

The datalog buffers point to different system variables depending on the build level. The buffers let you inspect the variables and judge the performance of the system. Figure 3-7 shows open and setup time graph windows to plot the data log buffers.

---

7. Alternatively, you can import graph configurations files in the project folder by doing the following:
    a. Click *Tools*.
    b. Click *Graph*.
    c. Click *DualTime...*
    d. Click *Import.*
    e. Browse to *C:\ti\c2000\C2000Ware_MotorControl_SDK_1_00_00_00\solutions\tmdxiddk379d\f2837x\debug*
    f. Select *fcl_f2837x_tmdxiddk_graph1.graphProp*. (The Graph Properties window should look like Figure 3-7.)
    g. Click *OK* to add graphs to your debug perspective.
   h. Click ⚙ (Continuous Refresh).

NOTE: If using a second graph window, import *fcl_f2837x_tmdxiddk_graph2.prop* (the beginning address for this should be DBUFF_4CH3 and DBUFF_4CH4).

The default dlog.prescaler is set to 5, which lets the dlog function to only log one out of every five samples.

Set the default dlog.trig_value to the correct value to generate trigger for the plot similar to using an oscilloscope within a debug environment.
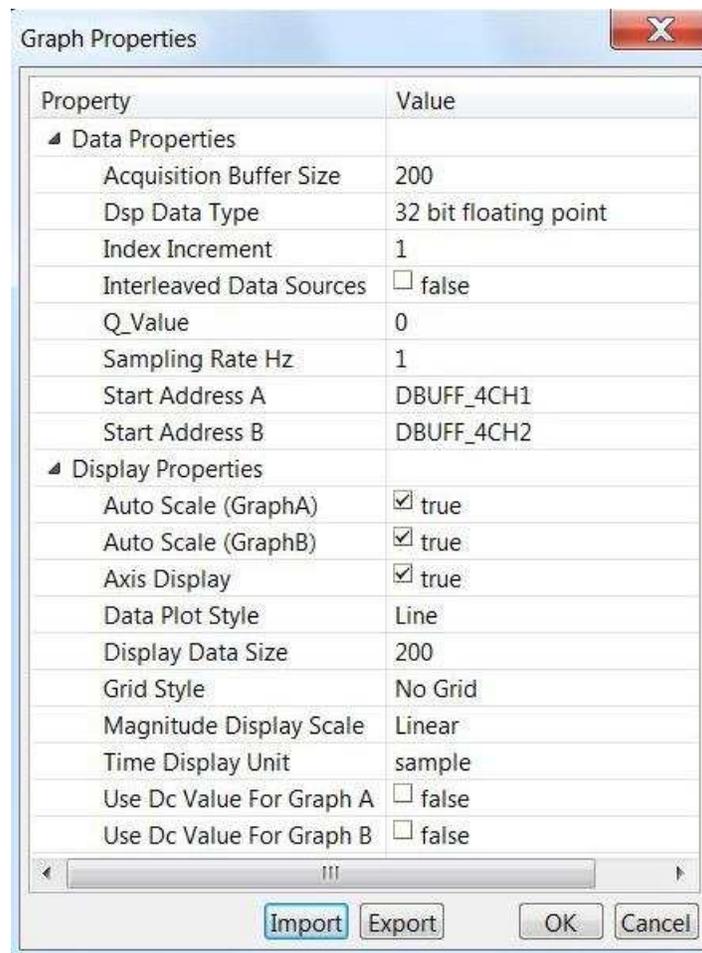
Figure 3-7. Graph Window Settings

3.6 Running the Code

1. Click *Run* in the Debug tab.

2. In the *Expressions* window, set the EnableFlag to 1.

> **NOTE:** The project runs and the values in the graphs and watch window continuously update. The following screen captures are typical CCS perspectives while using this project. You may resize the windows.

---

3. When complete, click (Run).

4. Click *Reset.*

5. Click (CPU Reset) to reset the processor.

6. Click *Run.*

7. Click (Terminate) to terminate the debug session.

> **NOTE:** This action halts the program and disconnects Code Composer from the MCU.

Edit perspective

Project files

Console window to display Build progress / errors

Summary of Build Errors / Warnings
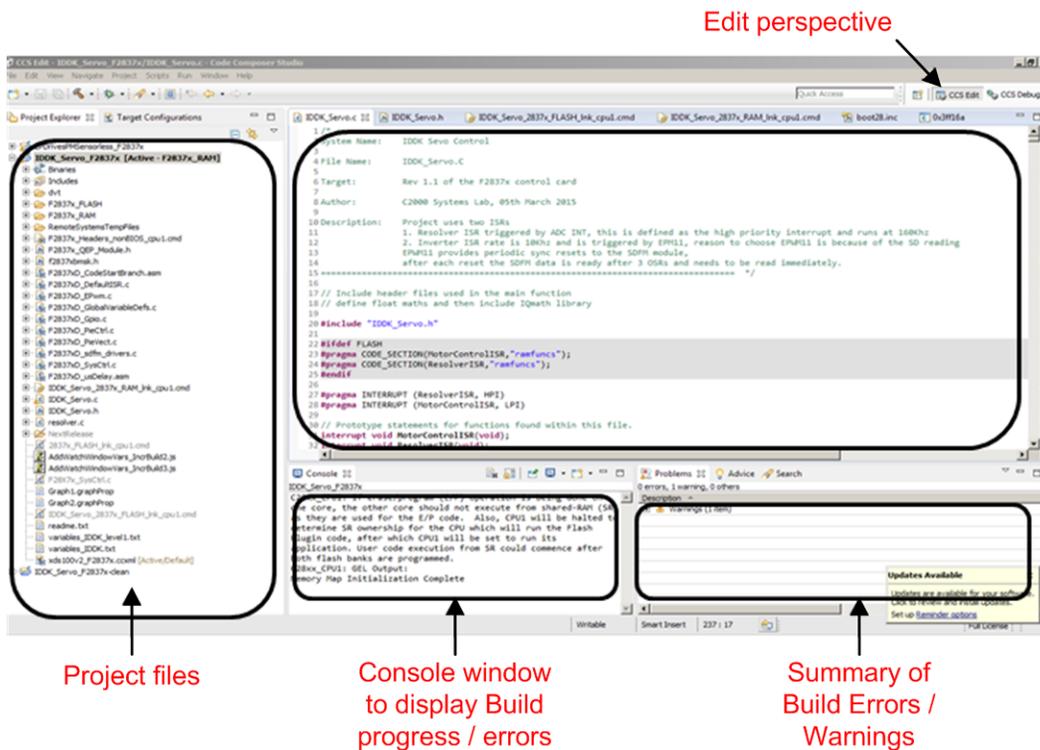
Figure 3-8. CCS IDE Showing Edit Perspective

Terminating the debug session each time you change or run the code is unnecessary. Instead of terminating, do the following after rebuilding the project:

1. Click *Run*.
2. Click *Reset*.
3. Click ![icon] (CPU Reset).
4. Click *Run*.
5. Click ![icon] (Restart).
6. Enable real-time options.

> **NOTE:** After completing the tests, end the debug session as follows.

7. Disable real-time options.
8. Reset the CPU.
9. Terminate the project if you change the target device or the configuration from RAM to Flash or Flash to RAM and before shutting down CCS.
10. Customize the project to meet your motor specifications and control options by editing the *fcl_qep_f2837x_tmdxiddk_settings.h* file at *C:\ti\c2000\C2000Ware_MotorControl_SDK_1_00_00_00\solutions\tmdxiddk379d\f2837x\include*

> **NOTE:** Feel free to change the PWM switching frequency (ISR frequency). Choose the ISR frequency as a submultiple of the PWM frequency for testing purposes or as required.

11. Open the lab manual in: *C:\ti\c2000\C2000Ware_MotorControl_SDK_1_00_00_00\solutions\tmdxiddk379d\docs*

> **NOTE:** For any other MCU and/or project, refer to the lab manual within the project folder under \docs folder

12. Start experimenting.

3.7 References

- *DesignDRIVE IDDK Hardware Reference Guide*

IMPORTANT NOTICE AND DISCLAIMER