Using Position Manager SinCos Library on IDDK

User's Guide

Literature Number: SPRUI57
January 2016

Contents

List of Figures

List of Tables

User's Guide
SPRUI57–January 2016

Using Position Manager SinCos Library on IDDK

1 Introduction to IDDK

The DesignDRIVE Kit (IDDK) is a single platform that makes it easy to develop and evaluate design solutions for many industrial drive and servo topologies. The IDDK offers support for a wide variety of motor types, sensing technologies, encoder standards and communications networks, as well as easy expansion to develop with real-time Ethernet communications and functional safety topologies, enabling more comprehensive, integrated system solutions. Based on the real-time control architecture of TI's C2000™ microcontrollers (MCUs), the kit is ideal for the development of industrial inverter and servo drives used in robotics, computer numerical control machinery (CNC), elevators, materials conveyance and other industrial manufacturing applications.

The IDDK offers an integrated drive design with a full power stage to drive a three-phase motor, easing evaluation of a diverse range of feedback sensing and control topologies. The kit includes a 180-pin HSEC controlCARD based on the TMS320F28379D C2000 Delfino™ MCU, which integrates dual C28x real-time processing cores and dual CLA real-time co-processors, providing 800 MIPS of floating-point performance with integrated trigonometric and FFT acceleration.

The sophisticated sensing peripherals on the TMS320F28379D MCU, including sigma-delta filter modules with up to 8 input channels, four high-performance 16-bit ADCs and eight windowed comparators, enable the IDDK to support shunt, flux gate/ HALL, and sigma-delta current sensing simultaneously. For position feedback, the IDDK leverages integrated MCU support for resolver and incremental encoder interfaces. In addition, customers can also explore configuration options that allow the MCU to be placed on either side of the high voltage isolation barrier.

The kit is designed to plug into 110 V/220 V AC mains, delivers up to 8 Amps, and is rated to drive motors up to one horsepower.

This user's guide covers the kit contents and hardware details, and explains the functions and locations of various connectors present on the board. For more details, see the *DesignDRIVE Development Kit IDDK v2.2 Hardware Reference Guide* (SPRUI23).

2 Hardware Configuration

To evaluate and experiment with the Position Manager SinCos library, the following components are needed:

- IDDK EVM
- TMDXCNCD28379D
- Encoder and connectors (not included in the kit)
  - SinCos Encoder
  - Cable
- An external isolated 15 V power supply needed for MCU code development (preferably with a barrel connector commonly referred to as a “DC jack”).
- PC with Code Composer Studio™ (version 6 or greater) installed

The IDDK Hardware Reference Guide is available in controlSUITE at: *controlSUITE\development_kits\TMDSIDDK_v2.0\~Docs*

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

For schematic details of the IDDK EVM, see the schematic file available in controlSUITE at:
controlSUITE\development_kits\TMDSIDDK_v2.0\IDDK_HwDevPkg\IDDK_HwDevPkg_v2.2.1

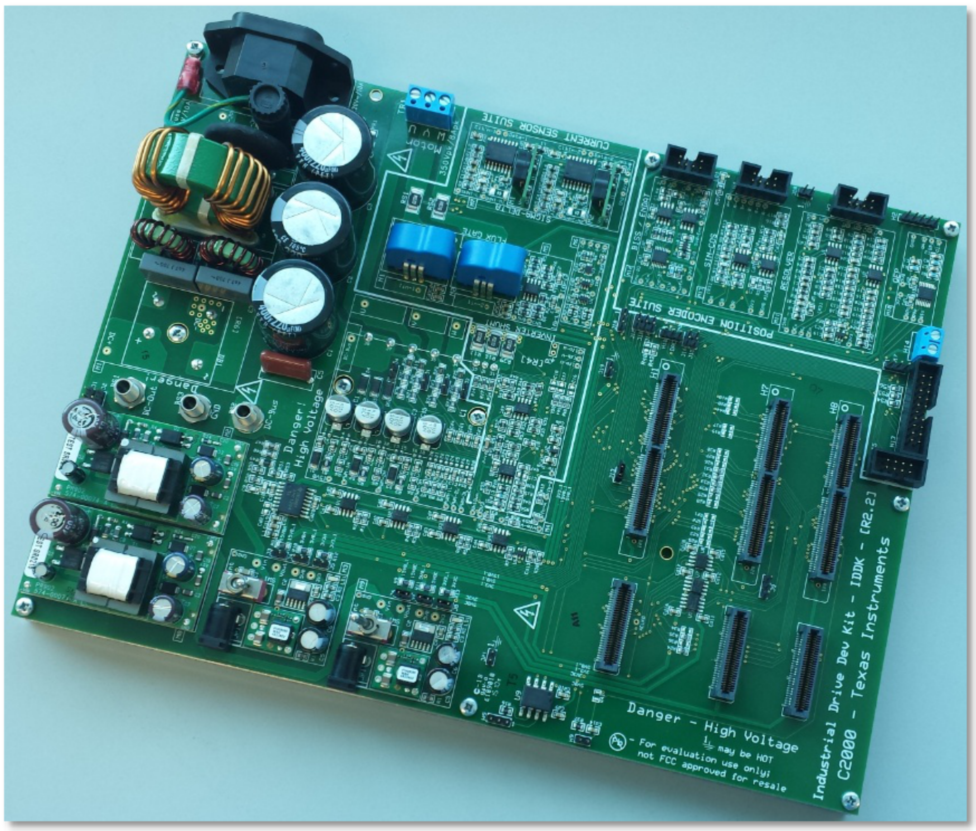


Figure 1. IDDK EVM Kit

3 Hardware Overview

This section describes the components required for evaluation of the Position Manager SinCos Library. A complete hardware overview of the kit can be obtained from DesignDRIVE Development Kit IDDK - Hardware Reference Guide, available in controlSUITE.

Evaluation of the Position Manager SinCos Library requires usage of:

- Processor (CPU) block for control
- Position encoder suite
- On-board power supplies

3.1 Functional Blocks

Table 1 illustrates the subset of functional blocks on DesignDRIVE Development Kit IDDK Hardware along with the macro names used for Position Manager SinCos library evaluation.

Table 1. Hardware Macros in IDDK Used for SinCos Evaluation

Functional Block	Macro Reference	Macro Function
Power Supplies	M9	DC Power Supply – Linear Reg 15 V-5 V/3.3 V
Position Encoder Suite	M13	SinCos Encoder Interface
Processor/controlCARD	H1	All other functions

Each functional block and the macros that make them are presented in brief detail in the sections below. The layout of various macros in the board is given in Figure 2. Schematic details of the individual macros are available in controlSUITE at:

controlSUITE\development_kits\TMDSIDDK_v2.0\IDDK_HwDevPkg\IDDK_HwDevPkg_v2.2.1

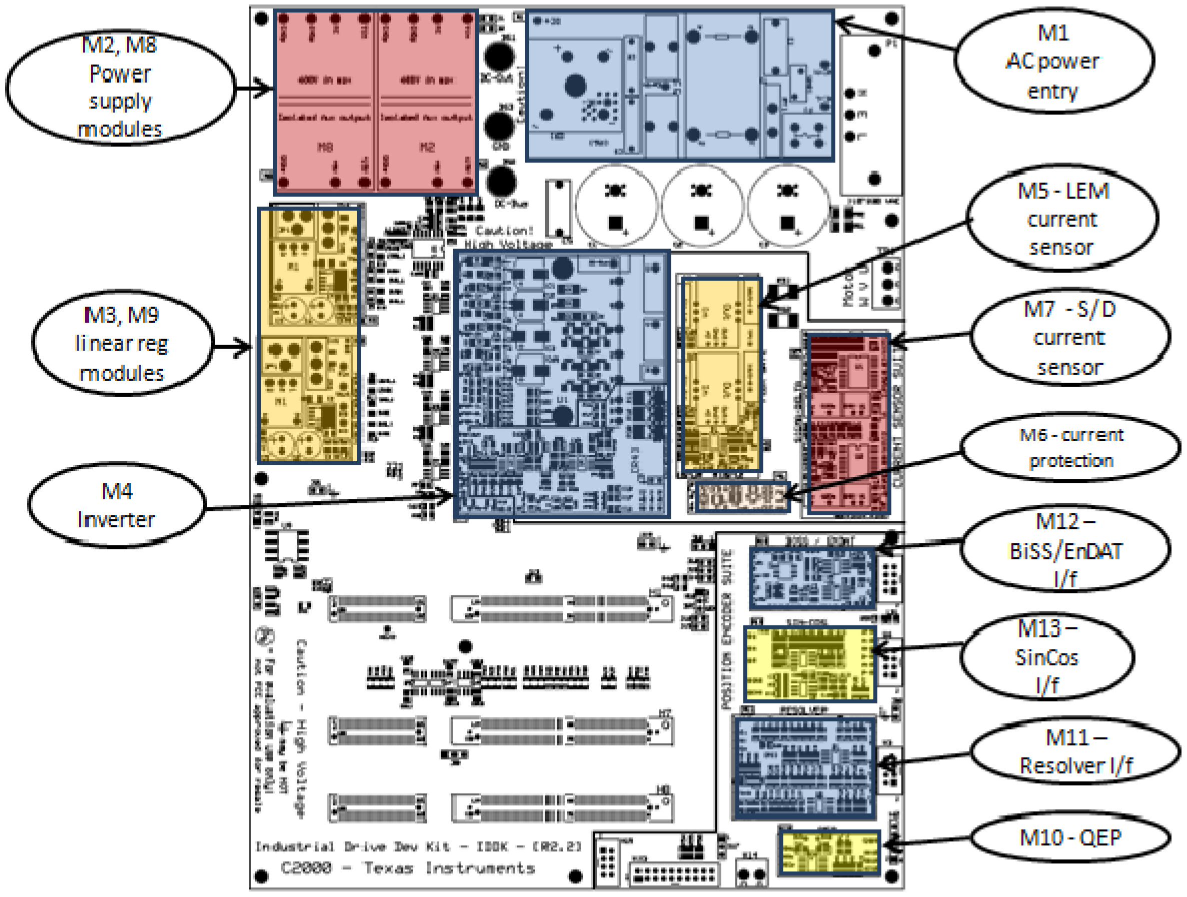


Figure 2. Layout of IDDK EVM With Its Functional Macros

3.2 *Processor Section (Control Processor Slot – H1)*

The IDDK is designed around the main control processor card in slot H1. This is designed to plug in a C2000 Delfino (TMS320F28379D) MCU control card TMDXCNCD28379D designed with a HSEC180pin edge connector.

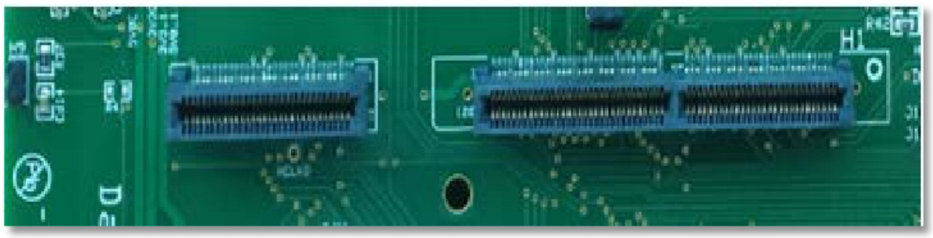


Figure 3. Processor Block

3.3 Position Encoder Suite

This block provides a range of position encoder and sensing interfaces, including:

- QEP
- Resolver
- SinCos
- EnDat / BiSS

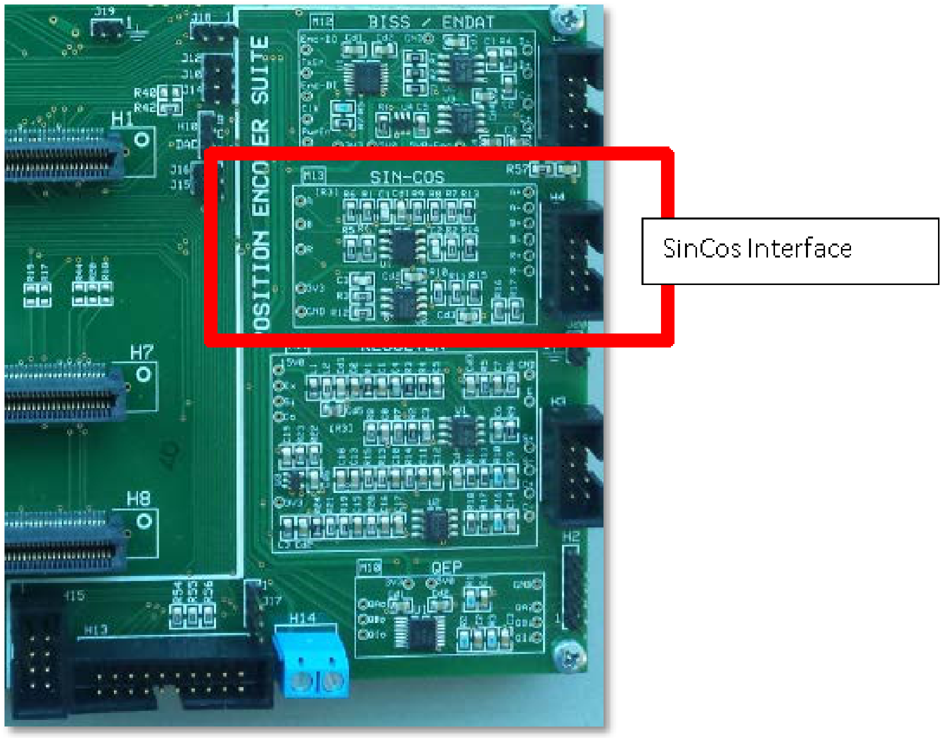


Figure 4. Position Sensor Suite

3.3.1 SinCos Encoder

This macro, designated M13, is a common interface for SinCos encoders. H4 is the external interface header with pin connections shown in Figure 5.

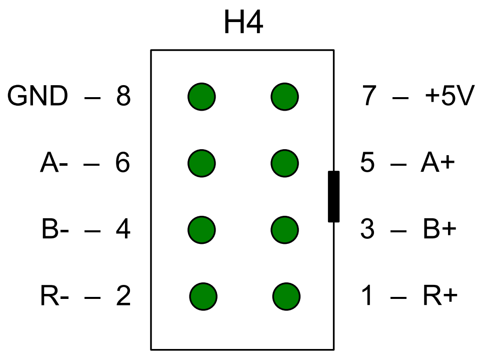


Figure 5. SinCos Interface Header (Top View)

3.4 Power Supply

An external power supply is preferred to power the controller during code development so that the board operates in low voltage. This ensures a safe environment as there is no high-voltage node present on the board.

The kit only needs 15 V of power and can receive this through M9 or M3. When the controller processor is to be on the cold-side, M9 should be used and this is what is assumed below.

M9 has a power supply jack ([M9]-JP1) and a toggle switch ([M9]-SW1). An external (15 V) power supply can be fed in through [M9]-JP1 while [M9]-SW1 should be turned away from the 'Int' position when power is needed

Settings needed to run when the controller processor is on the cold-side:

- [Main]-J6, J7, and J8 should be jumpered
- [Main]-J2 should NOT be jumpered
- [Main]-R9, R11, R13 should be populated
- [Main]-R8, R10, R12 should NOT be populated
- Ground resistors on the secondary side of the board should be configured for the controller to be on the cold-side. For more details, see the IDDK documentation (available in controlSUITE).

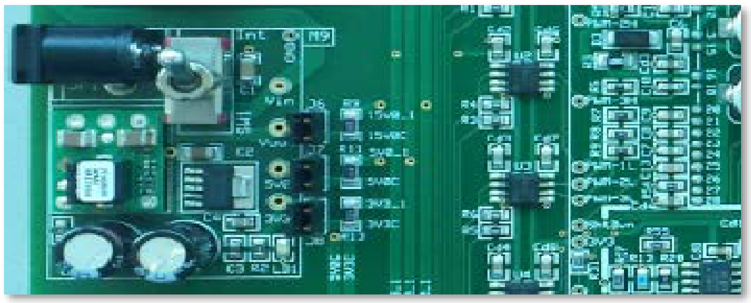


Figure 6. Power Supply Block

4 Hardware Resource Mapping

Section 4.1 shows the hardware resource requirements and signal mapping for PM SinCos library evaluation on the IDDK.

4.1 Signal Mapping

Table 2 shows the functional mapping of signals connected to the control processor on H1 that are relevant to PM SinCos library evaluation.

Table 2. Digital Signal Mapping on Control Card H1

IDDK Signal Name	MCU GPIO	MCU Peripheral Associated With GPIO	Function
SC-A-2	-	ADC-C/CMPSS-4	Sine input
SC-B-2	-	ADC-D/CMPSS-7	Cosine input
SC-R	-	ADC-D/CMPSS-8	Index input
CompOutSC-A	15-54	CMPSS-4/QEP-2	CMPSS - QEP channel A connection
CompOutSC-B	14-55	CMPSS-7/QEP-2	CMPSS - QEP channel B connection
CompOutSC-R	59-57	CMPSS-8/QEP-2	CMPSS - QEP channel I connection

4.2 Jumpers and Switches

The jumpers shown in Table 3 need to be populated for PM SinCos library evaluation.

Table 3. Purpose of Jumpers and Switches

Jumpers/Switches	Configuration	Description
[Main] - J6- J8	Populate	Jumpers to bring out linear regulator block M9 voltages
[Main] - J12	Do Not Populate	
[Main] - J15-16	Populate	Connect SinCos input signals to MCU
[Main] - J18	Populate (1-2)	Connects Encoder data out (En-Do-1 to the processor)
[Main] - J19-J21		GND headers for probe access
[M9] – SW1	Turn ON	Select 15 V supply from JP1 or onboard from M8 to generate 5 V and 3.3 V

4.3 Headers/Connectors

Table 4 shows headers and connectors that are relevant to SinCos configuration.

Table 4. Headers and Connectors

Headers/ Connectors	Description
[M9] – JP1	15 V DC power supply jack adapter
[Main] - H1	180 pin HSEC connector slot for control processor card
[Main] – H4	4x2 header for SinCos

5 Hardware Setup Instructions for PM_sincos Library Evaluation

1. Ensure default configuration – Make sure that jumpers [Main]-J6, [Main]-J7 and [Main]-J8 are in front of macro M9, that resistors [Main]-R9, R11, R13 are populated and that GND plane resistors R14 and R15 are mounted as shown in the *Various GND Planes on the Bottom of Board* and *Default Connection of Various GND Planes* figures in the *Power Supplies* section of the *DesignDRIVE Development Kit IDDK v2.2 Hardware Reference Guide* (SPRUI23).
2. Unpack the TMDXCNCD28379D control card and slide it down in the connector slot of [Main]-H1. Push down vertically using even pressure on both ends of the card until it cannot slide down further. To remove the card, spread open the retaining clips, if present, and pull the card out applying even force at the far edges.
3. 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.
4. Connect an encoder to connector H4 in accordance with the wiring description in the manufacturer’s data sheet.

   For more information, see the encoder/cable specifications and Hardware schematics, see the *Control Processor Slot - H1* section of the *DesignDRIVE Development Kit IDDK v2.2 Hardware Reference Guide* (SPRUI23).

   Insert the transducer cable into [Main] – H4 4x2 header of IDDK. Figure 7 shows a combined BiSS/SinCos transducer connected to the IDDK.

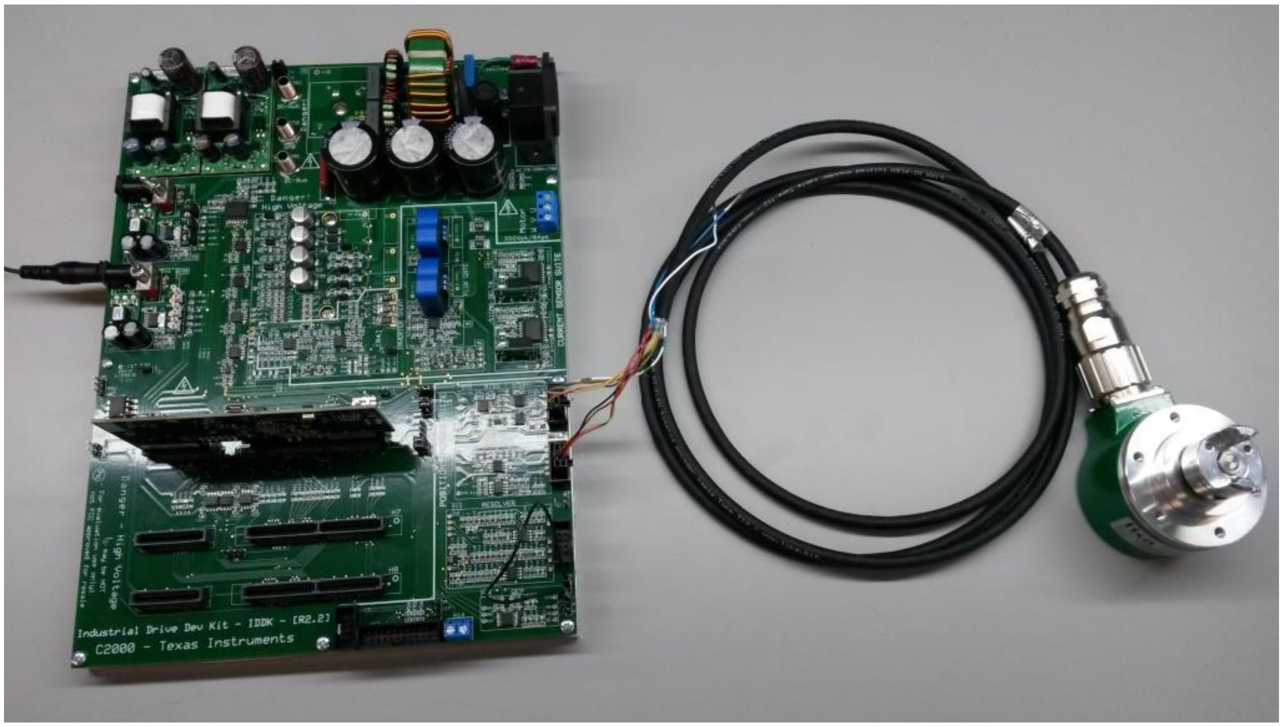

Industrial Drive Dev Kit - IDDK - [R2.2]
C2000 - Texas Instruments
Danger - High Voltage

Figure 7. Combined BiSS/SinCos Transducer Connected to the IDDK

5. Ensure that toggle switch [M9]-SW1 is in “Int” position. Connect an isolated 15 V DC power supply to [M9]-JP1.
6. Turn on toggle switch [M9]-SW1. Then, [M9]-LD1 should turn on. Note that more LEDs on the control card should light up indicating that the control card is receiving power from the board.

6 Software Setup for PM SinCos Evaluation Example Project

6.1 Installing Code Composer and controlSUITE

1. Install Code Composer v6.x or later (if not already installed) from http://www.ti.com/tool/CCSTUDIO.
2. Run the http://www.ti.com/controlsuite installer. Allow the installer to download and update any automatically checked software for C2000.
3. The SinCos Library is available at the following location:

   <base> install directory is:

   C:\ti\controlSUITE\libs\app_libs\position_manager\sincos\v01_00_00_00\sincos

   The following sub-directory structure is used:

```
<base>\Doc        Documentation
<base>\Float      Implementation of the library and corresponding include file
<base>\examples   Example using SinCos library
```

6.2 Software Flow Diagram

An outline software flow of the example project PM_sincos_example is shown in Figure 8

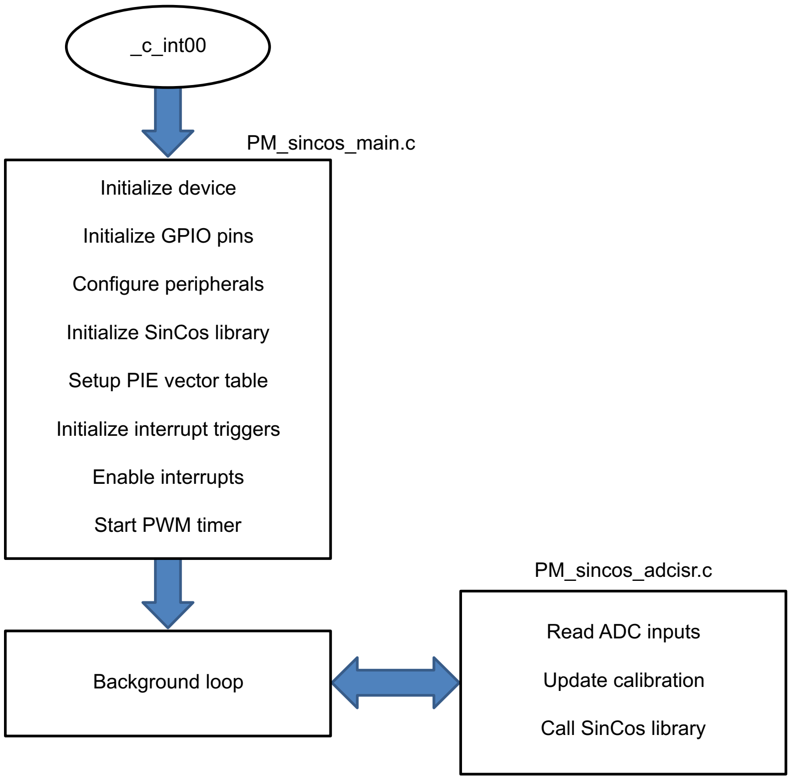
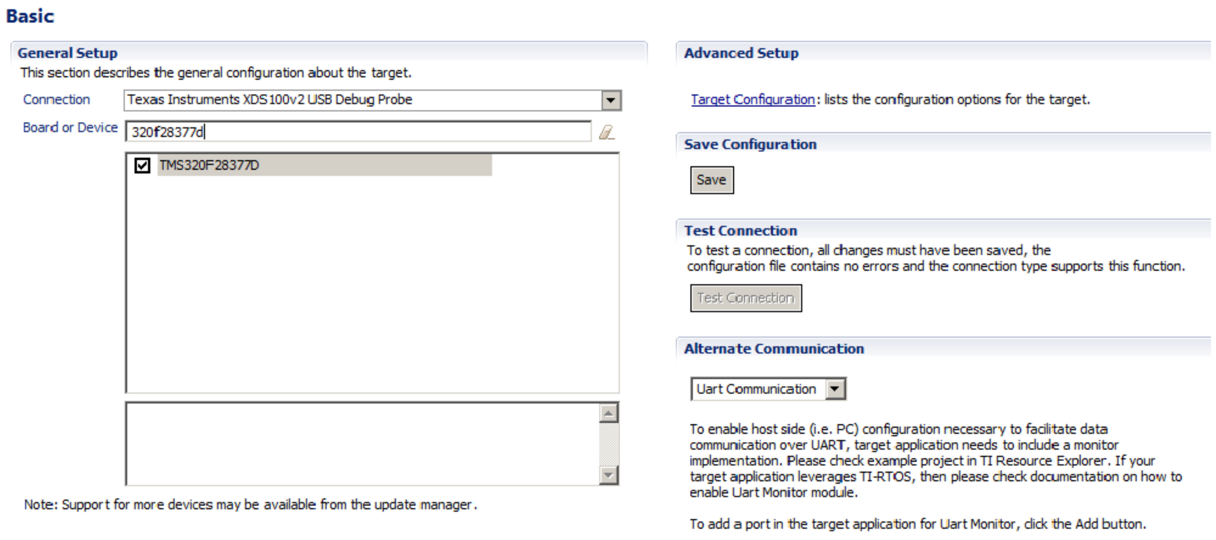


Figure 8. Software Flow Diagram for the Example Project PM_sincos_example

7 Setup Code Composer Studio to Work With TMDXIDDK379D

1. Open “Code Composer Studio” (note that this document assumes version 6 or later).
2. Once Code Composer Studio opens, the workspace launcher may appear prompting the selection of a workspace location; the workspace is a location on the hard drive where all the user settings for the ID, which projects are open, what configuration is selected, and so forth are saved. This can be anywhere on the disk, the location mentioned below is just for reference. Also, note that if this is not your first-time running Code Composer Studio, the dialog below may not appear:
   (a) Click the “Browse…” button
   (b) Create the path below by making new folders as necessary:
   C:\c2000 projects\CCSv6_workspaces\PM_sincos_eval_workspace
   (c) Uncheck the box that says “Use this as the default and do not ask again”.
   (d) Click “OK”
3. This opens a ‘Getting Started’ tab with links to various tasks, such as creating a new project, importing an existing project, to watching a Tutorial on CCS. Click the ‘Import Project’ icon that skips the procedure to step 1 in Section 7.1, or close the ‘Getting Started’ Tab and go to the next step.
4. Configure Code Composer to know which MCU it will connect to. This is done by setting up the ‘Target Configuration’. All these are already set up and configured in “xds100v2_F2837x.ccxml” provided as part of the files in project, and you can skip to step . However, for general information regarding setting up this configuration file, steps 6, 7 and 8 can be used.
5. A new configuration file can be set by clicking “View → Target Configuration. This opens the Target Configuration window, click on . Give a name to the new configuration file depending on the target device. If “Use shared location” checkbox is checked, then this configuration file can be stored in a common location by CCS for use by other projects as well. Then click Finish.

6. This should open up a new tab as shown in Figure 9. Select and enter the options as shown:
   (a) Connection – Texas Instruments XDS100v2 USB Emulator or Texas Instruments XDS100v2 USB Debug Probe
   (b) Device – the C2000 MCU on the control card, TMS320F28379D, for example
   (c) Click Save and close


Basic

General Setup
This section describes the general configuration about the target.
Connection: Texas Instruments XDS100v2 USB Debug Probe
Board or Device: 320f28377d
☑ TMS320F28377D
Note: Support for more devices may be available from the update manager.

Advanced Setup
Target Configuration: lists the configuration options for the target.

Save Configuration
Save

Test Connection
To test a connection, all changes must have been saved, the configuration file contains no errors and the connection type supports this function.
Test Connection

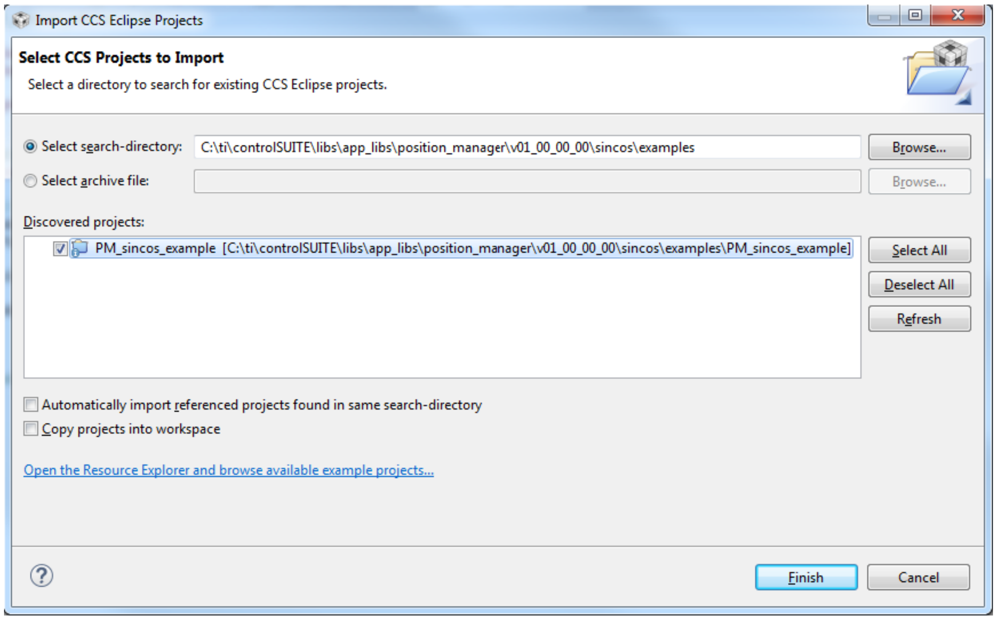
Alternate Communication
Uart Communication
To enable host side (i.e. PC) configuration necessary to facilitate data communication over UART, target application needs to include a monitor implementation. Please check example project in TI Resource Explorer. If your target application leverages TI-RTOS, then please check documentation on how to enable Uart Monitor module.

To add a port in the target application for Uart Monitor, click the Add button.

Figure 9. Configuring a New Target

7. Click “View → Target Configurations”. In the “User Defined” section, find the file that was created in step 6 and 7. Right-click on this file and select “Set as Default”. To use the configuration file supplied with the project, click “View → Target Configurations, then expand “Projects → PM_sincos_example and right-click on the file “xds100v2_F2837x.ccxml” and “Set as Default”. This tab also allows you to reuse existing target configurations and link them to specific projects.
8. Add the PM SinCos evaluation example project into the current workspace by clicking “Project → Import CCS Project”.
   (a) Select the project by browsing to:

      C:\ti\controlSUITE\libs\app_libs\position_manager\v01_00_00_00\sincos\examples\PM_sincos_example”



Select CCS Projects to Import

Select a directory to search for existing CCS Eclipse projects.

(●) Select search-directory: C:\ti\controlSUITE\libs\app_libs\position_manager\v01_00_00_00\sincos\examples

( ) Select archive file:

Discovered projects:

[✓] PM_sincos_example [C:\ti\controlSUITE\libs\app_libs\position_manager\v01_00_00_00\sincos\examples\PM_sincos_example]

[ ] Automatically import referenced projects found in same search-directory

[ ] Copy projects into workspace

Open the Resource Explorer and browse available example projects...

Figure 10. Adding the SinCos Example Project to the CCS Workspace

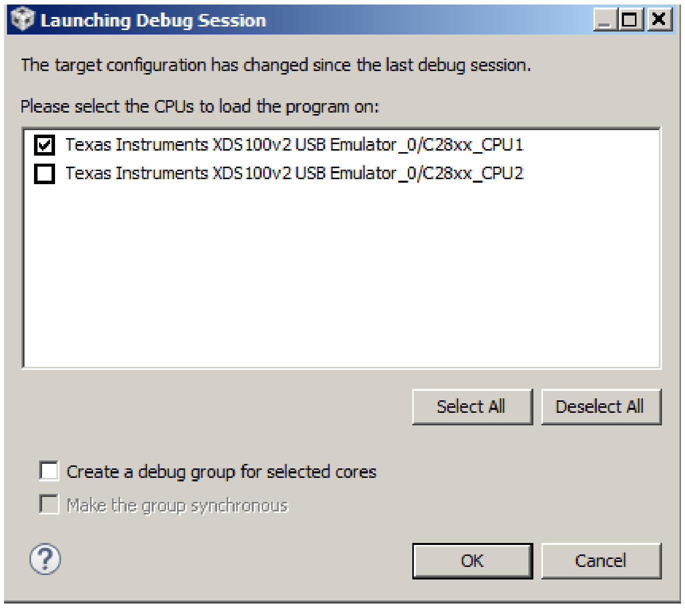
If there are multiple projects in this directory, click and choose the projects to import, and then click “Finish”. This copies all the selected projects into the workspace. Figure 10 shows one project only; select it and then click “Finish”.

7.1 Configure the Project

1. Assuming this is your first time using Code Composer Studio, the xds100v2-F2837x should have been set as the default target configuration. Verify this by viewing the xds100v2-f2837x.ccxml file in the expanded project structure and a [Active/Default] status written next to it. By going to “View → Target Configurations” you can edit existing target configurations or change the default or active configuration. You can also link a target configuration to a project in the workspace by right clicking on the Target configuration name and selecting “Link to Project”.

7.2 Build and Load the Project

1. Open the file PM_sincos_lib.h and ensure the definition “LINECOUNT” matches the number of electrical cycles per mechanical rotation for the chosen transducer. Save the file.
2. Right Click on the Project Name and click on “Rebuild Project” and watch the Console window. Any errors in the project will be displayed in the Console window.
3. On successful completion of the build, click the “Debug” button located in the top-left side of the screen. The window shown in Figure 11 may pop up if being used for the first time or requesting the user to select which of the two CPUs to connect to. Choose CPU1 by clicking the box next to CPU1.



The target configuration has changed since the last debug session.

Please select the CPUs to load the program on:

- [✓] Texas Instruments XDS100v2 USB Emulator_0/C28xx_CPU1
- [ ] Texas Instruments XDS100v2 USB Emulator_0/C28xx_CPU2

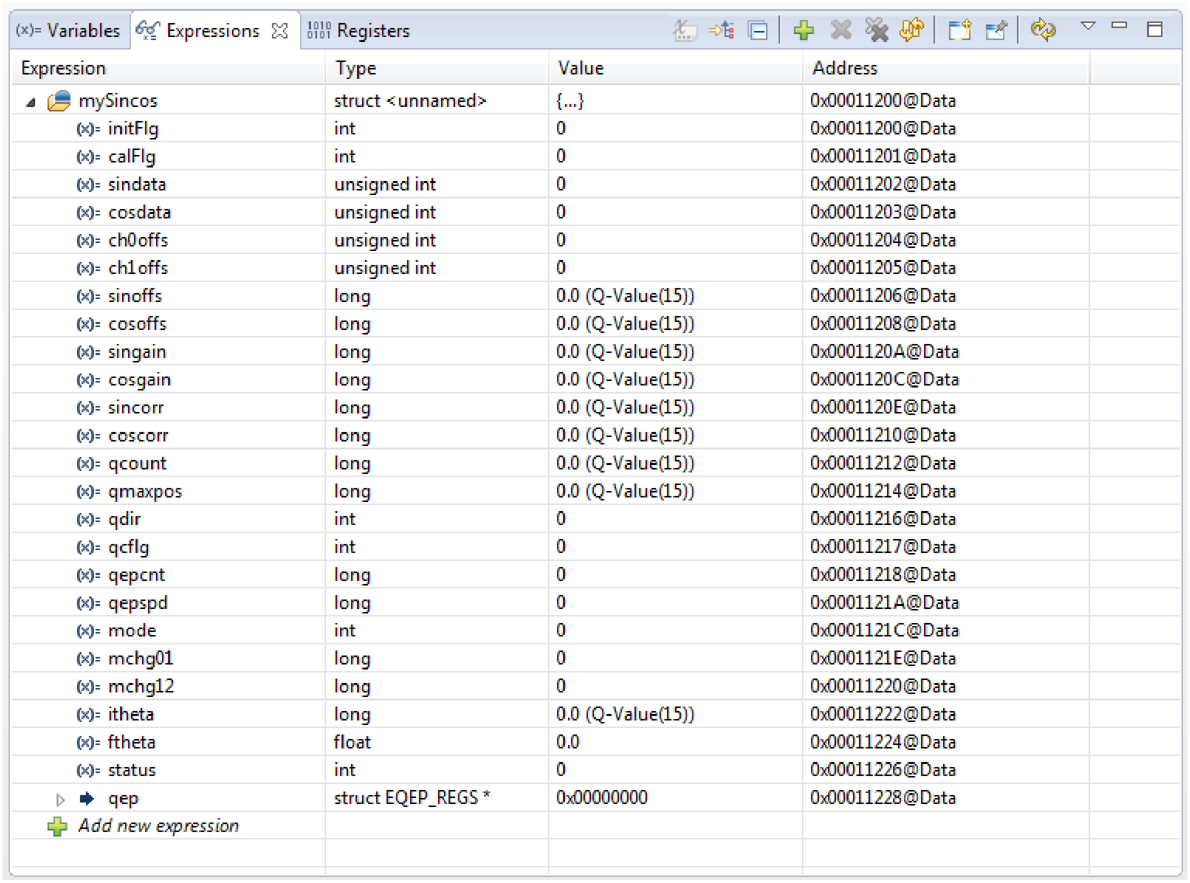
- [ ] Create a debug group for selected cores
- [ ] Make the group synchronous

Figure 11. Selecting the CPU(s) to Connect

4. The IDE will now automatically connect to the target, load the output file into the device and change to the Debug perspective.
5. Click “Tools → Debugger Options → Program/Memory Load Options”. You can enable the debugger to reset the processor each time it reloads program by checking “Reset the target on program load or restart” and click “Remember My Settings” to make this setting permanent.
6. Click on the “Enable silicon real-time mode” button that auto selects the “Enable polite real-time mode” button. This allows you to edit and view variables in real-time. Do not reset the CPU without disabling these real-time options!
7. A message box may appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a “0”. The DGBM is the debug enable mask bit. When the DGBM bit is set to “0”, memory and register values can be passed to the host processor for updating the debugger windows.

7.3 Setup Watch Window

1. Click: View → Expressions on the menu bar to open a watch window to view the variables being used in the project. Add the “mySincos” structure to the watch window and expand it as shown in Figure 12.



Expression	Type	Value	Address
mySincos	struct <unnamed>	{...}	0x00011200@Data
initFlg	int	0	0x00011200@Data
calFlg	int	0	0x00011201@Data
sindata	unsigned int	0	0x00011202@Data
cosdata	unsigned int	0	0x00011203@Data
ch0offs	unsigned int	0	0x00011204@Data
ch1offs	unsigned int	0	0x00011205@Data
sinoffs	long	0.0 (Q-Value(15))	0x00011206@Data
cosoffs	long	0.0 (Q-Value(15))	0x00011208@Data
singain	long	0.0 (Q-Value(15))	0x0001120A@Data
cosgain	long	0.0 (Q-Value(15))	0x0001120C@Data
sincorr	long	0.0 (Q-Value(15))	0x0001120E@Data
coscorr	long	0.0 (Q-Value(15))	0x00011210@Data
qcount	long	0.0 (Q-Value(15))	0x00011212@Data
qmaxpos	long	0.0 (Q-Value(15))	0x00011214@Data
qdir	int	0	0x00011216@Data
qcflg	int	0	0x00011217@Data
qepcnt	long	0	0x00011218@Data
qepspd	long	0	0x0001121A@Data
mode	int	0	0x0001121C@Data
mchg01	long	0	0x0001121E@Data
mchg12	long	0	0x00011220@Data
itheta	long	0.0 (Q-Value(15))	0x00011222@Data
ftheta	float	0.0	0x00011224@Data
status	int	0	0x00011226@Data
qep	struct EQEP_REGS *	0x00000000	0x00011228@Data
Add new expression			

Figure 12. Configuring the Expressions Window

2. Click on the Continuous Refresh button in the watch window. This enables the window to run with real-time mode. By clicking the down arrow in this watch window, you can select “Customize Continuous Refresh Interval” and edit the refresh rate of the watch window. Note that choosing too fast an interval may affect performance.

7.4 Run the Code

1. Run the code by pressing the Run Button in the Debug Tab.
2. The project should now run, and the values in the watch window should continuously update. If encoder is not mounted on a spinning Motor, users can manually rotate the encoder shaft and observe the position changing in the watch window.
3. Once complete, reset the processor (Run → Reset → CPU Reset) and then terminate the debug session by clicking (Run → Terminate). This will halt the program and disconnect Code Composer from the MCU.

7.5 Closing the Project

It is not necessary to terminate the debug session each time you change or run the code again. Instead, the following procedure can be followed.

1. After rebuilding the project, (Run → Reset → CPU Reset), (Run → Restart), enable the real-time options.
2. Once complete, disable real-time options and reset the CPU.
3. Terminate the project if the target device or the configuration is changed, and prior to shutting down CCS.

8 References

DesignDRIVE Development Kit IDDK v2.2 Hardware Reference Guide (SPRUI23)

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		