

MSP430i2040 Embedded Metering EVM (EVM430-i2040SUBMTR)

User's Guide



Literature Number: SLAU587
August 2014

1	Getting Started	6
1.1	Preface	6
1.2	Safety and Precautions	6
1.3	Package Contents	7
1.4	Introduction	7
1.5	Features.....	7
1.6	Hardware.....	8
1.6.1	Top View of the EVM.....	8
1.6.2	Bottom View of the EVM.....	8
1.6.3	Hardware Setup Procedures	9
1.7	Calibrator Software.....	10
1.7.1	Software Package Content.....	10
1.7.2	Setting up the PC Software Tool	10
1.8	Instruments.....	13
2	Operating the PC Software Tool	14
2.1	Introduction.....	14
2.2	Start Using the EVM	14
2.3	Known Issues	18
3	Calibration Techniques	19
3.1	Introduction.....	19
3.2	Calibration Techniques	19
3.3	Calibration Procedures	20
3.3.1	Calibration of AC and DC Parameters	20
3.3.2	Calibration of Compensation Resistance and Capacitance	21
3.3.3	Calibration of Current AC Offset	22
3.3.4	Calibration of Voltage AC Offset	22
3.3.5	Calibration of Phase Correction	22
3.3.6	Calibration of DC Parameters	22
4	Serial Communication Commands	23
4.1	Introduction.....	23
4.2	Communication Protocol	23
4.2.1	Polling Mode	23
4.3	Commands	25
4.3.1	HOST_CMD_GET_METER_NAME	25
4.3.2	HOST_CMD_GET_METER_VER.....	25
4.3.3	HOST_CMD_GET_METER_CONFIGURATION.....	26
4.3.4	HOST_CMD_GET_RTC.....	27
4.3.5	HOST_CMD_ALIGN_WITH_CALIBRATION_FACTORS	27
4.3.6	HOST_CMD_SET_PASSWORD.....	27
4.3.7	HOST_CMD_GET_READINGS_PHASE_n	28
4.3.8	HOST_CMD_GET_EXTRA_READINGS_PHASE_n.....	29
4.3.9	HOST_CMD_SUMCHECK_MEMORY	29
4.3.10	HOST_CMD_CLEAR_CALIBRATION_DATA.....	30
4.3.11	HOST_CMD_SET_CALIBRATION_PHASE_n.....	30
4.3.12	HOST_CMD_GET_CALIBRATION_PHASE_n	31

4.3.13	HOST_CMD_SET_CALIBRATION_EXTRAS	31
4.3.14	HOST_CMD_GET_CALIBRATION_EXTRAS.....	32
5	Firmware and Embedded Metering Library API	33
5.1	Introduction.....	33
5.2	Firmware Structure.....	33
5.3	Toolkit Package.....	33
5.4	Metrology Computation Engine.....	34
5.4.1	Background Process	35
5.4.2	Foreground Process	36
5.5	Embedded Metering Library API	37
5.5.1	Embedded Metering Library Function calls	37
5.5.2	Embedded Metering Library Callbacks	40
5.5.3	Application-Level Calibration Functions	41
A	Example Application Code	48
A.1	Introduction.....	48
A.2	Prepaing the Application Code to Run	48
A.3	Downloading Without an IAR License	55
B	Hardware Design Files	58
B.1	Package	58
B.2	Schematic	59
C	EVM Specification and Performance	60
C.1	EVM Specifiction	60
C.2	EVM Performance.....	61
C.2.1	Power Accuracy at Room Temperature	61
C.2.2	Accuracy vs Temperature at 220 V, 5 A	63
D	Running on MSP430i2040 and MSP430i2041	64

List of Figures

1-1.	EVM Top View	8
1-2.	EVM Bottom View	8
1-3.	System Properties Window	11
1-4.	Device Manager Window	12
1-5.	Edit calibration-config.xml	13
2-1.	Calibrator Software Startup Window	15
2-2.	Meter Status Window	16
2-3.	Meter Calibration Factor Window	17
2-4.	Meter Features Window	17
2-5.	Meter Error Window	18
3-1.	Front-End Interface Model	19
4-1.	Polling Mode Data Frame Format	23
4-2.	Polling Mode Command Frame Format	24
4-3.	Polling Mode Respond Frame Format	24
A-1.	Project Options	49
A-2.	Optimization Options	50
A-3.	Debugger Options	51
A-4.	Download Options	52
A-5.	Compiling the Application	53
A-6.	Warnings	53
A-7.	Connecting EVM and FET	54
A-8.	Code Downloading	54
A-9.	Debugger Screen	55
A-10.	EVM Running	55
A-11.	Download File	56
A-12.	Select File	57
B-1.	EVM Schematic	59
C-1.	Power Percentage Error (%) vs Load Current (A)	62
C-2.	Typical Accuracy vs Temperature	63
D-1.	IAR 5.5 Launch Window	64
D-2.	Open Workspace Window	65
D-3.	Workspace Options	65
D-4.	Options Window	66
D-5.	Config Tab	66
D-6.	Selecting Default XCL File	67
D-7.	Finishing the Setting Changes	67
D-8.	Saving the Setting Changes	68

General Texas Instruments High Voltage Evaluation (TI HV EVM) User Safety Guidelines



Always follow TI's setup and application instructions, including use of all interface components within their recommended electrical rated voltage and power limits. Always use electrical safety precautions to help ensure your personal safety and those working around you. Contact TI's Product Information Center <http://support/ti.com> for further information.

Save all warnings and instructions for future reference.

Failure to follow warnings and instructions may result in personal injury, property damage, or death due to electrical shock and burn hazards.

The term TI HV EVM refers to an electronic device typically provided as an open framed, unenclosed printed circuit board assembly. It is **intended strictly for use in development laboratory environments, solely for qualified professional users having training, expertise and knowledge of electrical safety risks in development and application of high voltage electrical circuits. Any other use and/or application are strictly prohibited by Texas Instruments.** If you are not suitable qualified, you should immediately stop from further use of the HV EVM.

1. Work Area Safety

- (a) Keep work area clean and orderly.
- (b) Qualified observer(s) must be present anytime circuits are energized.
- (c) Effective barriers and signage must be present in the area where the TI HV EVM and its interface electronics are energized, indicating operation of accessible high voltages may be present, for the purpose of protecting inadvertent access.
- (d) All interface circuits, power supplies, evaluation modules, instruments, meters, scopes and other related apparatus used in a development environment exceeding 50Vrms/75VDC must be electrically located within a protected Emergency Power Off EPO protected power strip.
- (e) Use stable and nonconductive work surface.
- (f) Use adequately insulated clamps and wires to attach measurement probes and instruments. No freehand testing whenever possible.

2. Electrical Safety

As a precautionary measure, it is always a good engineering practice to assume that the entire EVM may have fully accessible and active high voltages.

- (a) De-energize the TI HV EVM and all its inputs, outputs and electrical loads before performing any electrical or other diagnostic measurements. Revalidate that TI HV EVM power has been safely de-energized.
- (b) With the EVM confirmed de-energized, proceed with required electrical circuit configurations, wiring, measurement equipment connection, and other application needs, while still assuming the EVM circuit and measuring instruments are electrically live.
- (c) After EVM readiness is complete, energize the EVM as intended.

WARNING: WHILE THE EVM IS ENERGIZED, NEVER TOUCH THE EVM OR ITS ELECTRICAL CIRCUITS AS THEY COULD BE AT HIGH VOLTAGES CAPABLE OF CAUSING ELECTRICAL SHOCK HAZARD.

3. Personal Safety

- (a) Wear personal protective equipment (for example, latex gloves or safety glasses with side shields) or protect EVM in an adequate lucent plastic box with interlocks to protect from accidental touch.

Limitation for safe use:

EVMs are not to be used as all or part of a production unit.

Getting Started

1.1 Preface

Thank you for selecting Texas Instruments [EVM430-i2040SUBMTR](#) development tool. This development tool provides you the resources to evaluate the performance of MSP430i2040 utilizing as embedded metering application. This documentation will guide you to get familiar with this development tool and will provide necessary information to start further development and evaluation. Before proceeding, read this guide to gain basic understanding and knowledge of this development tool and how to safely operate it.

More resources related to this development tool are available for download from the [Texas Instruments](#) web site, including more documentation, application notes, example application code, and software.

1.2 Safety and Precautions

The EVM is designed to operate by professionals who received appropriate technical training yet it is designed to operate from ac supply or high voltage dc supply please read the safety related documents that come with the EVM package and this user guide before operating this EVM.

CAUTION



Read user guide before use.

CAUTION



Do not leave EVM powered when unattended.

WARNING



HOT SURFACE : Contact may cause burns, do not touch.

WARNING



DANGER HIGH VOLTAGE : Electric shock possible when connecting board to live wires. Board should be handled with care by professional. For safety, use of isolated equipment with overvoltage and overcurrent protection is highly recommended.

1.3 Package Contents

- Quick start guide
- Safety guideline
- Terms and disclaimer
- Disc containing software and user guide
- EVM430-i2040SUBMTR

1.4 Introduction

The Texas instruments MSP430i2040 is an ultra-low-power mixed-signal microcontroller. It integrates four independent differential-input 24-bit sigma-delta analog-to-digital converters with programmable gain amplifiers, a 16-bit hardware multiplier, an eUSCI_A0 that supports UART and SPI communication interfaces, an eUSCI_B0 that supports SPI and I²C communication interfaces, two 16-bit timers, and 12 GPIO pins in a 28-pin TSSOP or 16 GPIO pins in a 32-pin QFN package. The peripheral set is a good combination for electricity power measurement.

This EVM is designed as an evaluation tool for using the MSP430i2040 in the application of embedded metering (sub-metering). In this application, the electricity measuring device is embedded in the end application and provides the user with information about the voltage, current, and power consumption of the device. Moreover, the EVM can compensate for the line resistance and EMI filter capacitance.

Embedded metering (sub-metering) is applicable in many areas such as:

- Home appliances
- Server or PC power supplies
- UPS
- Smart plugs or power strips
- Solar energy inverter
- Electrical vehicle charger
- Home monitoring, security, and automation

In this document, all the description of the configuration, operation, features, behaviors, functions, and interfaces is based on the default firmware that is pre-programmed on the EVM and the original EVM hardware design. Proper functionality is not ensured if changes made to the hardware or the firmware.

1.5 Features

Hardware features

- Spy-Bi-Wire debugging interface
- 14-pin debugger connector allows direct interface to MSP-FET430UIF without the need of an adaptor
- Built-in switching-mode power supply capable of supply from 85 to 265 VAC (47 Hz to 63 Hz) or 120 to 380 VDC simplifies evaluation setup
- Built-in RS232 external communication interface for reading measurements and performing calibration
- Seven built-in LEDs for customer debugging and visual monitoring

Software features

- Measurement of root mean square (rms) voltage, rms current, active power, reactive power, apparent power, power factor, ac frequency, voltage THD, current THD, fundamental voltage, fundamental current and fundamental active power
- Readings update every four ac cycles or 80 ms in case of dc input
- Capable of ac and dc measurement
- Capable of switching between ac and dc measurement mode automatically
- Capable of EMI filter capacitor and wire resistance compensation
- No separate dc calibration required

1.6 Hardware

This section introduces the parts on the EVM and describes the procedures to set up the hardware for evaluation.

1.6.1 Top View of the EVM

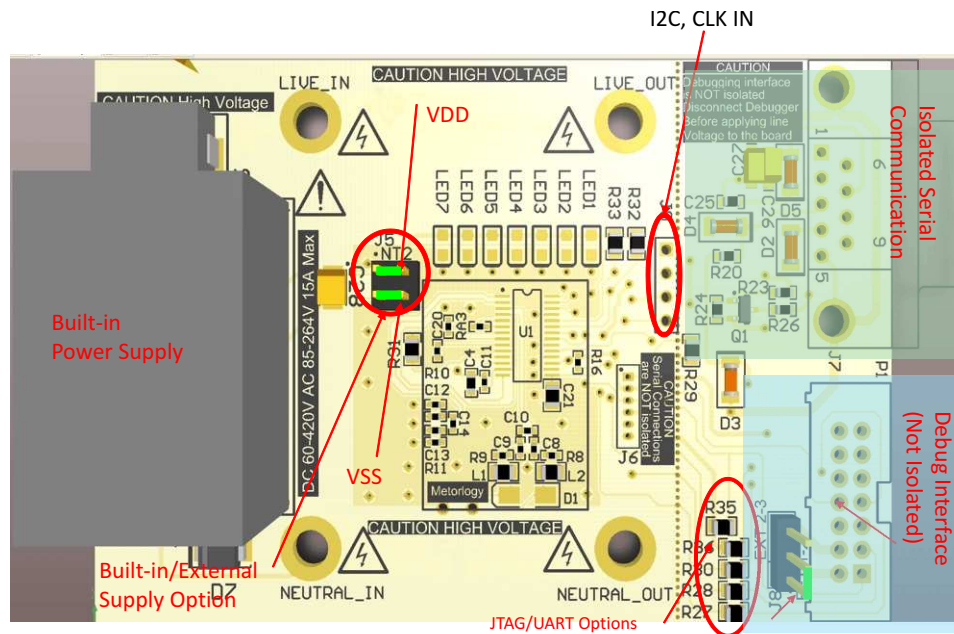


Figure 1-1. EVM Top View

1.6.2 Bottom View of the EVM

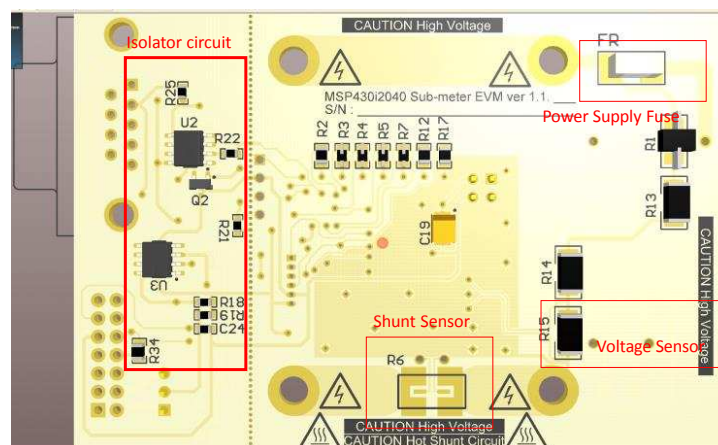


Figure 1-2. EVM Bottom View

1.6.3 Hardware Setup Procedures

1.6.3.1 Power Supply to EVM

The design of the EVM allows power to the EVM be supplied from the built-in power supply, which takes the line input to generate the power needed by the EVM. The EVM also allows power to the EVM be supplied from user source.

The EVM is shipped with the built-in power supply block installed. If power is to be supplied from external power supply, disconnect the built-in VSK-S1-3P3U power supply block. Remove the two jumpers on J5, and apply 3.3 V to VDD and VSS on J5 as shown in [Figure 1-1](#).

CAUTION

Do not supply power to the EVM until the hardware and software setup is completed.

1.6.3.2 Serial Communication Interface

Connect an RS232 extension cable to the DB-9 connector on the EVM and to a standard RS232 port on a computer.

1.6.3.3 Line Input and Load Output

- **Solder** the ac source live or dc source positive to LIVE_IN connector.
- **Solder** the ac source neutral or dc source negative to NEUTRAL_IN connector.
- **Solder** the load's live or positive power input to LIVE_OUT connector.
- **Solder** the load's neutral or negative power input to NEUTRAL_OUT connector.

1.6.3.4 Debugging Interface

Spy-Bi-Wire is used on this EVM as the debug interface to external debugger.

Before connecting to MSP-FET430UIF, if the power (while debugging) to the i2040 device is to be supplied by the FET then short pins 1 and 2 of J8 (toward board edge). If the power (while debugging) to the i2040 device is to be supplied by voltage on VDD pin, then short pins 2 and 3 of J8 (toward board center) (see [Figure 1-1](#)).

CAUTION

The debugging interface is NOT ISOLATED; make sure that proper isolation is in place between the EVM and the PC using for debugging.

NOTE: Connection to debugging interface is optional for the operation of the EVM. The EVM can operate standalone without debugger connected.

1.7 Calibrator Software

A package of software is needed to access full functionality of the EVM. The EVM package comes with the firmware pre-programmed into i2040 that allows the EVM to operate properly. The EVM also has a software package that is on the disc that comes with the EVM or the user can download the package from Texas Instruments web site. This section discusses the software in the software package and the procedure to setup the software to operate with the EVM

1.7.1 Software Package Content

The ZIP file for the EVM consist of several packages includes :

- PC Software tool to run on PC for reading and calibrating the EVM. The setup and operation of this PC software tool is described in [Section 1.7.2](#) and in [Chapter 3](#).
- Source code and Embedded Metering Library. This will be discussed in [Chapter 5](#) and [Appendix A](#).
- Hardware design files including schematic, layout and bill of materials (also see [Appendix B](#)).

1.7.2 Setting up the PC Software Tool

1.7.2.1 Minimum System Requirement

- PC with RS232 port or a RS232 port through USB
- PC running Microsoft Windows XP SP3 or Windows 7

The software tool has been tested to run properly on a Pentium M 1.4-GHz PC with 1.25GB of RAM installed; thus, it is assumed that most of the computers today meet the processing power requirement.

1.7.2.2 Installing the Software

Extract the file named calibrator-runtime.zip into any folder. A folder named *calibrator-runtime* that contains the necessary files to run the software tool is created. The file named calibrator-20121120.exe is the executable file of the software tool. The file named calibrator-config.xml contains the setup information for the software tool. Edit this XML file before calibrator-20121120.exe is launched (see [Section 1.7.2.3](#)).

1.7.2.3 Configuring the Software

Follow these steps to edit the XML for calibrator-20121120.exe to run properly.

1. Right click **My Computer** Icon and select **Properties** in the pop-up menu.
2. Select the **Hardware** tab in the **System Properties** Window, then Click the **Device Manager** to go to the **Device Manager** window.

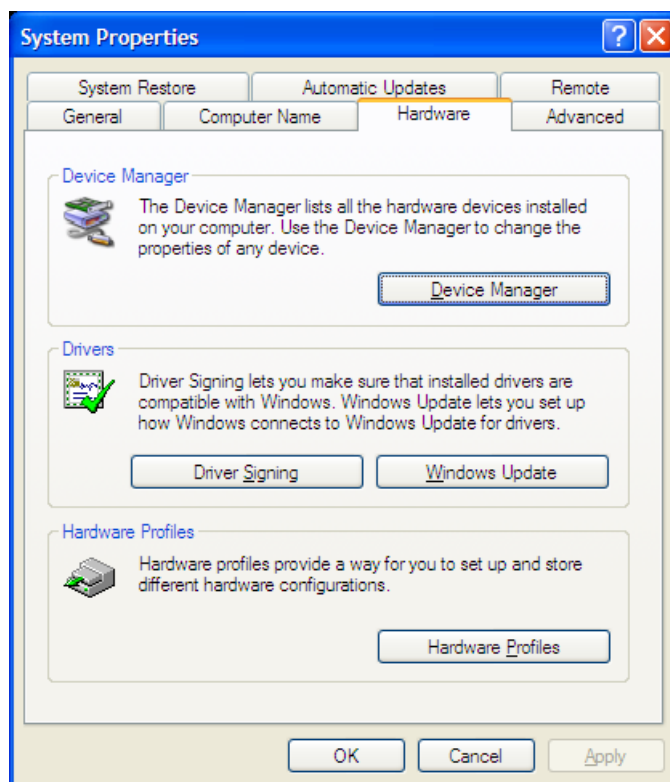


Figure 1-3. System Properties Window

3. Find the COM port number of the serial port that connecting the PC and the EVM.

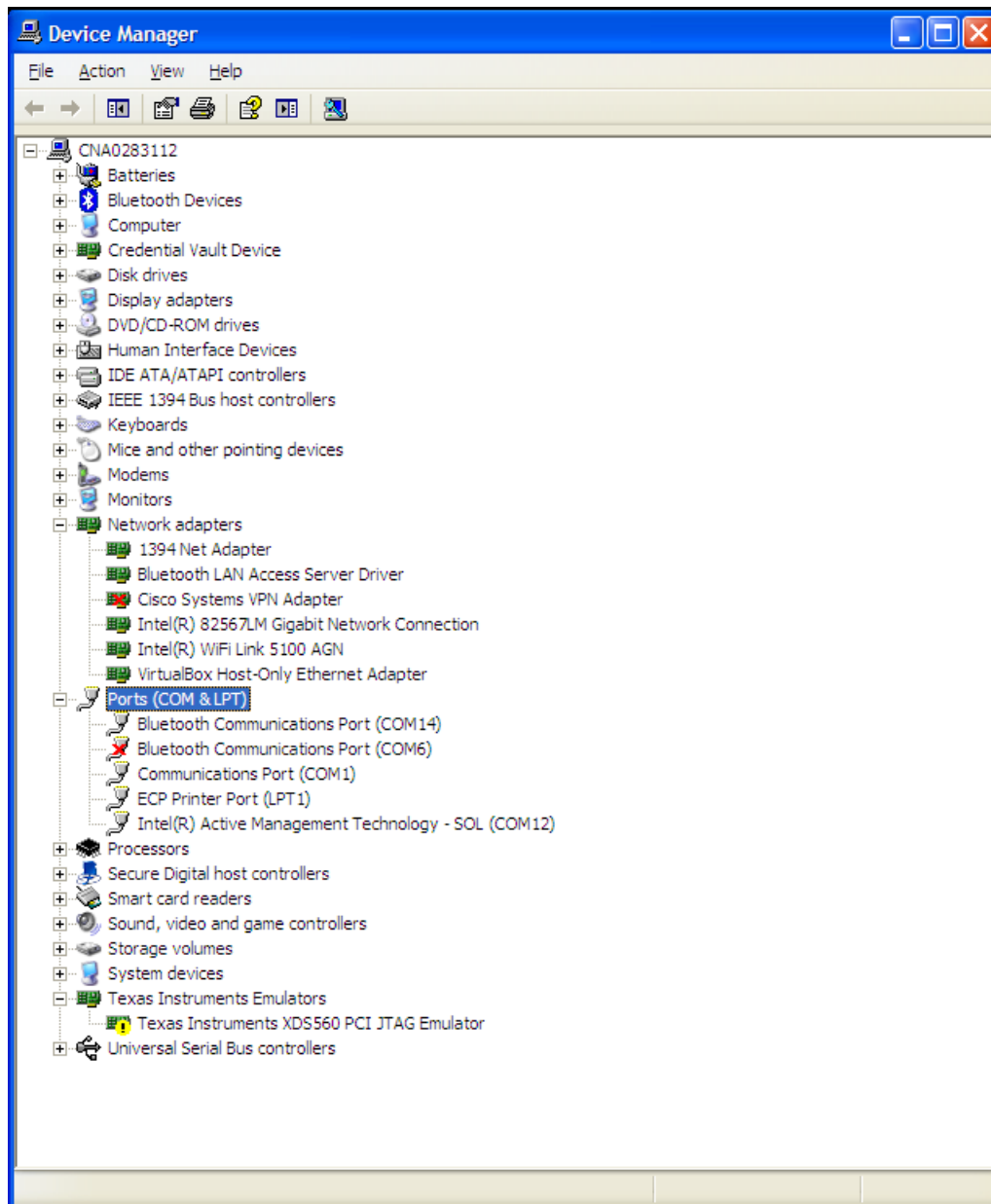


Figure 1-4. Device Manager Window

4. Open calibration-config.xml in the folder calibrator-runtime with a text editor or XML editor. Go to the line as shown below, enter the COM port number, and save the file.

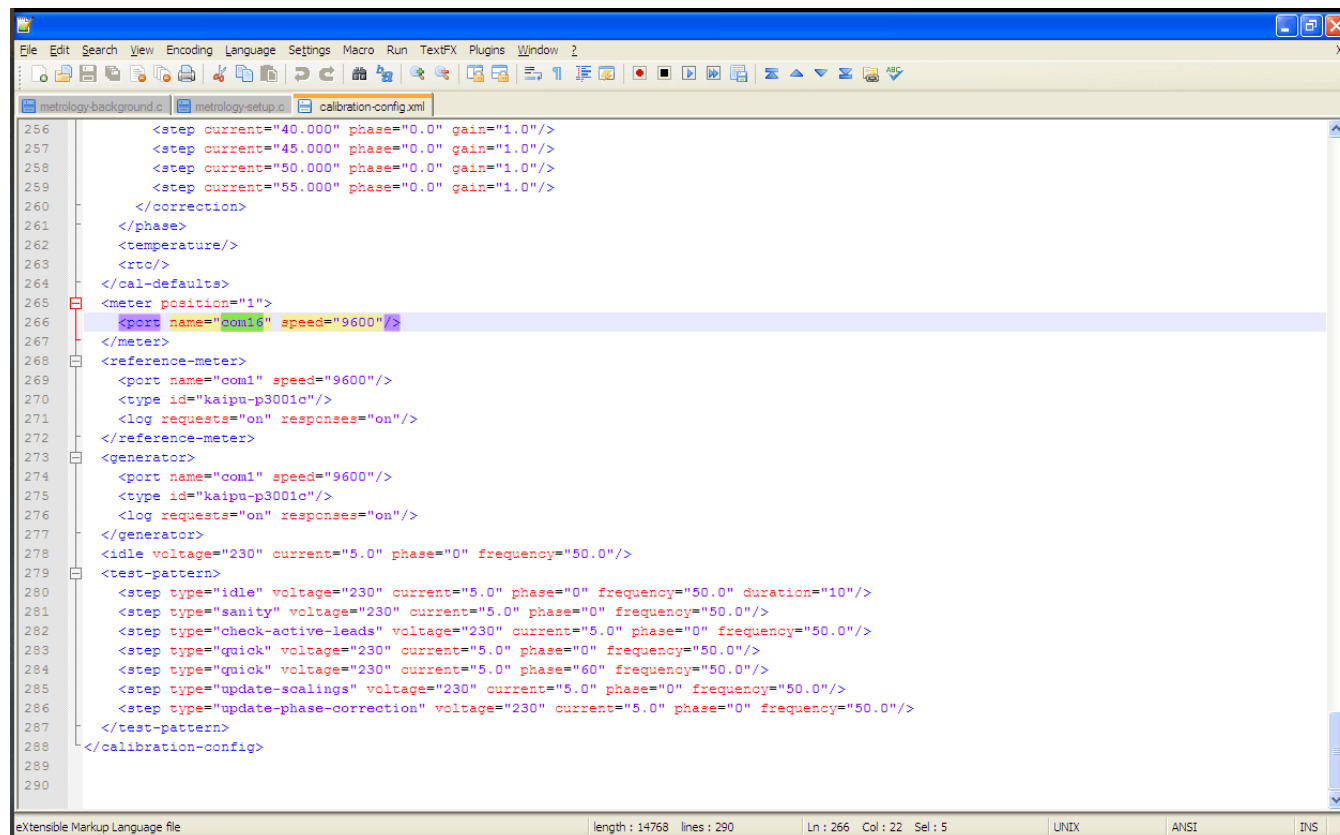


Figure 1-5. Edit calibration-config.xml

1.8 Instruments

The EVM has been programmed with a set of calibration factors that give readings accurate to within a few percent. If more accurate results are needed, the EVM must be calibrated (see [Chapter 3](#)). The following instruments are suggested to perform calibration:

- An ac source that can output sufficient power to drive the load at rated frequency (for example, 50 to 60 Hz) and rated voltage (for example, 110 to 220 V) or an ac test set that can generate the rated frequency and voltage.
- A variable ac load or the UUT. If DC measurement is needed, a variable dc load or the UUT is also recommended. Alternatively an ac test set that can generate the rated loading.
- A reference meter that can give ac parameters of V_RMS, I_RMS, and P_ACTIVE.

The following links provide information about the instruments that were used for testing and calibrating the design.

- AC Meter Test Set: http://en.3gcnpkaipu.com/products_list/&pmcId=10.html
- Reference Meter: http://www.hc.com.tw/portal_c1_cnt_page.php?owner_num=c1_142363&button_num=c1&folder_id=17560&cnt_id=124863&search_field=&search_word=&search_field2=&search_word2=&search_field3=&search_word3=&bool1=&bool2=&search_type=1&up_page=1
- AC Source: <http://www.chromausa.com/acpowersources/61500lo-ac-source.php>
- Reference Meter: <http://www.chromausa.com/powermeters/66201-66202-digital-power-meters.php>
- DC Electronic Load: <http://www.chromausa.com/dclloads.php#6310a>

Operating the PC Software Tool

2.1 Introduction

This section describes the operation of the EVM and the PC software tool. Before proceeding, make sure that the steps described in [Chapter 1](#) are completed, power is applied to the EVM, and line voltage to the load is applied.

2.2 Start Using the EVM

When the EVM has been powered, some of the seven LEDs flash or turn on to indicate its operation status. Some of the LED are not used, and you can make change to the provided source code to make use of all of the seven LEDs as needed in the application. [Table 2-1](#) shows the pre-programmed meaning of the LEDs.

Table 2-1. LED Indicators

LED	State	Indication
LED1	ON	Background operations running
	OFF	Background operations completed
LED2	ON	Negative voltage half cycle
	OFF	Positive voltage half cycle
LED3	ON	Not used
	OFF	Not used
LED4	ON	Foreground operations running
	OFF	Foreground operations completed
LED5	ON	EVM is in ac mode measurement
	OFF	EVM is not in ac mode measurement
LED6	ON	EVM is in dc mode measurement
	OFF	EVM is not in dc mode measurement
LED7	ON	Active energy pulse pulsing
	OFF	Active energy pulse idle

The EVM is now ready to run, so launch the software calibrator-20121120.exe in the folder calibration-runtime to start communicating with the EVM. A window as shown in [Figure 2-1](#) appears. As defined in the XML file calibration-config.xml, meter position 1 is assigned the serial port to communicate with the EVM. The **Comms** indicator turns green if communication to EVM from PC is established, and it flashes between read and green when communication is taking place.

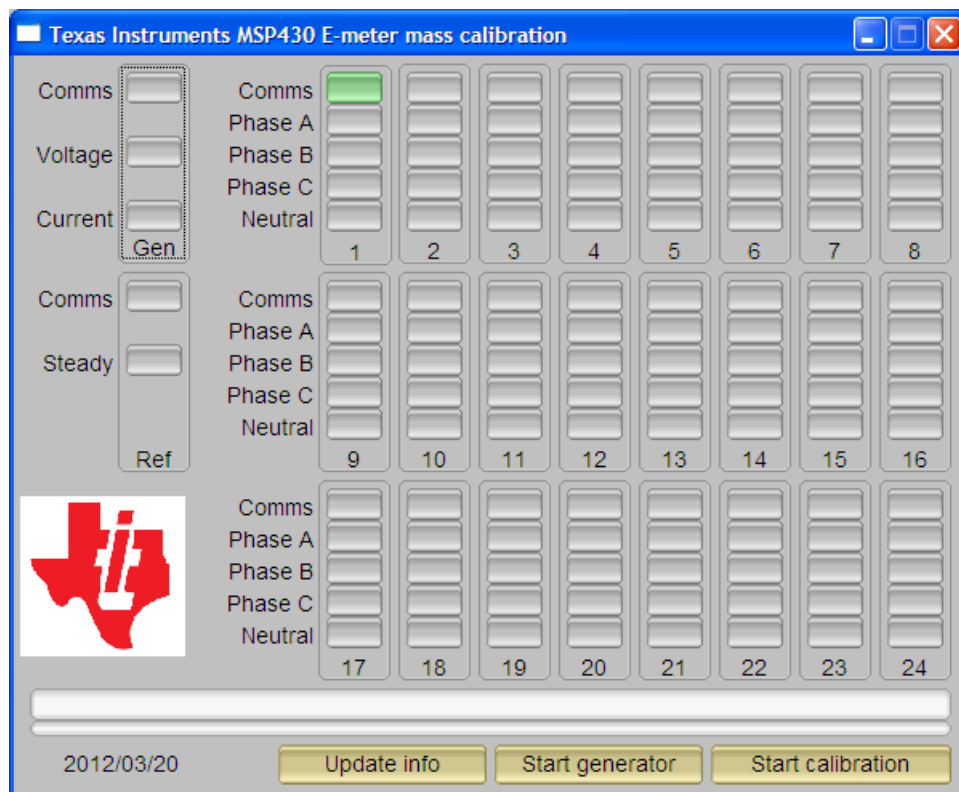


Figure 2-1. Calibrator Software Startup Window

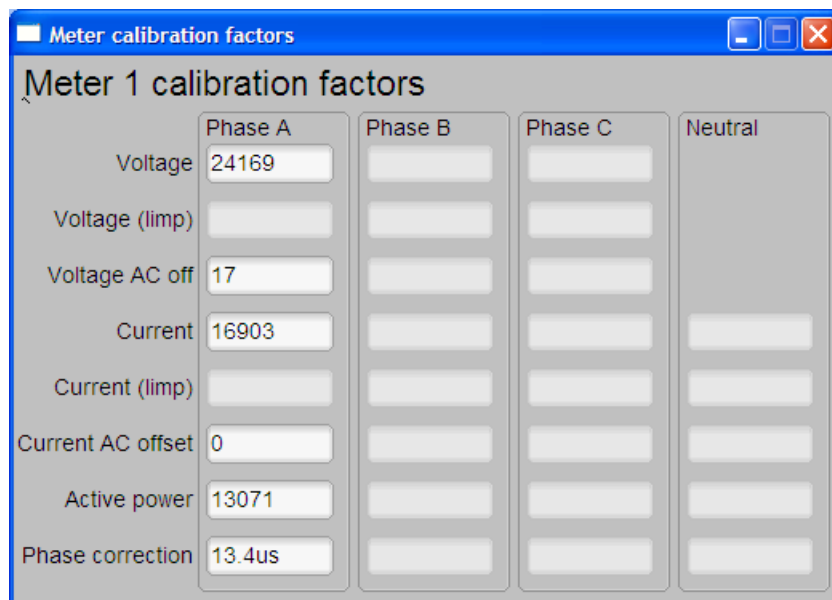
Click the **Comms** indicator to open the **Meter Status** window (see [Figure 2-2](#)).


Figure 2-2. Meter Status Window

This window shows the current reading of the meter. The background of a text box is gray if the EVM does not support that particular reading. The background turns red if the reading from EVM to that box has a large variance. The background turns yellow if the reading from EVM has a fairly low variance. The background turns green if the reading has a low variance. Note that the software on PC reads the EVM every second and also averages the data read; thus, the update rate is slower than the update rate of the EVM.

At the bottom of this window, there are four buttons.

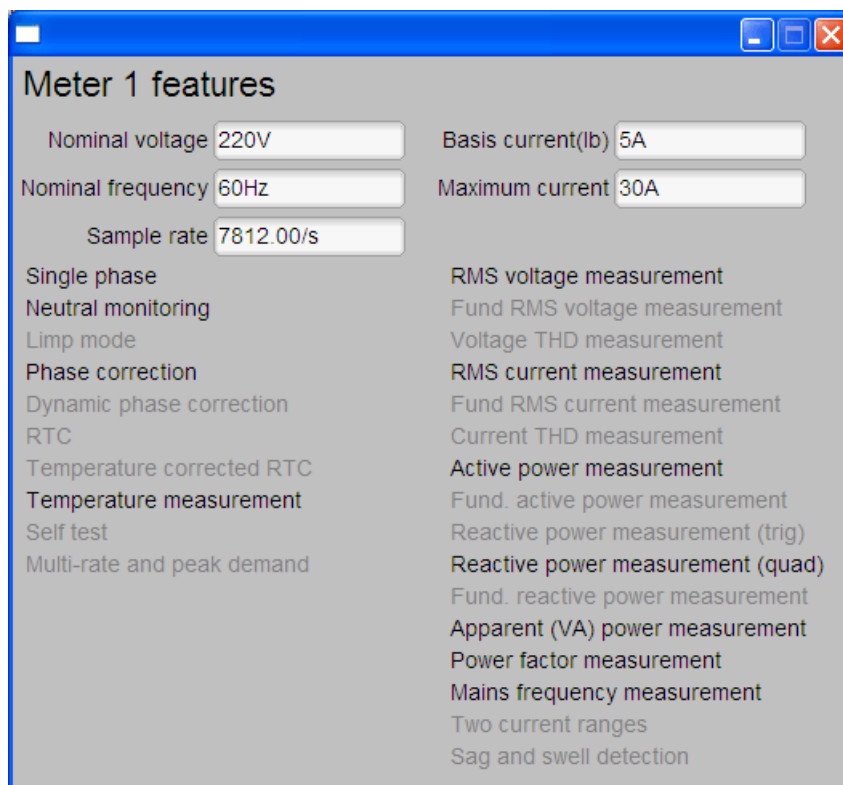
- Click the **Meter Consumption** button to open the **Meter Consumption** window. Because the EVM does not support this feature, the **Meter Consumption** window gives no useful information.
- Click the **Meter Calibration Factors** button to open the [Meter Calibration Factors] window (see [Figure 2-3](#)). This window shows the current calibration factor values.



	Phase A	Phase B	Phase C	Neutral
Voltage	24169			
Voltage (limp)				
Voltage AC off	17			
Current	16903			
Current (limp)				
Current AC offset	0			
Active power	13071			
Phase correction	13.4us			

Figure 2-3. Meter Calibration Factor Window

- Click the **Meter Features** button to open the **Meter Features** window (see Figure 2-4). This window shows the supported features of the EVM. Note that Figure 2-4 shows only the look and feel of the meter feature window, but this is not the exact feature of the EVM.

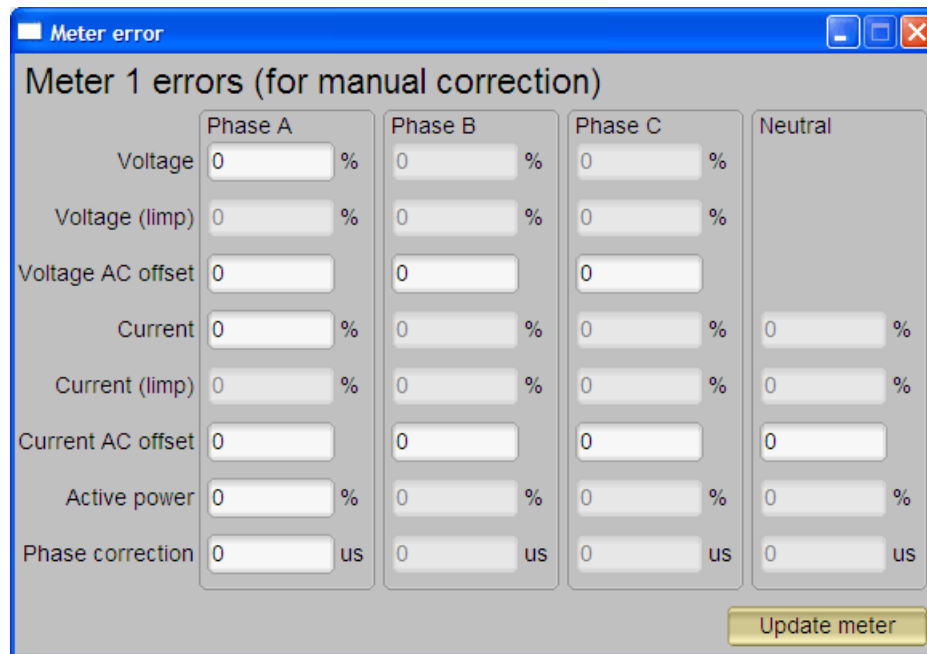


Nominal voltage	220V	Basis current(lb)	5A
Nominal frequency	60Hz	Maximum current	30A
Sample rate	7812.00/s		
Single phase		RMS voltage measurement	
Neutral monitoring		Fund RMS voltage measurement	
Limp mode		Voltage THD measurement	
Phase correction		RMS current measurement	
Dynamic phase correction		Fund RMS current measurement	
RTC		Current THD measurement	
Temperature corrected RTC		Active power measurement	
Temperature measurement		Fund. active power measurement	
Self test		Reactive power measurement (trig)	
Multi-rate and peak demand		Reactive power measurement (quad)	
		Fund. reactive power measurement	
		Apparent (VA) power measurement	
		Power factor measurement	
		Mains frequency measurement	
		Two current ranges	
		Sag and swell detection	

Figure 2-4. Meter Features Window

- Click the **Manual Cal** button brings up the **Meter Error** window. In this window, the adjustment to the calibration factor values could be done by entering the percentage error of the reading from the EVM compare to the reading from the reference meter. The technique and procedure of performing

calibration is discussed in [Chapter 3](#).



	Phase A	Phase B	Phase C	Neutral
Voltage	0 %	0 %	0 %	
Voltage (limp)	0 %	0 %	0 %	
Voltage AC offset	0	0	0	
Current	0 %	0 %	0 %	0 %
Current (limp)	0 %	0 %	0 %	0 %
Current AC offset	0	0	0	0
Active power	0 %	0 %	0 %	0 %
Phase correction	0 us	0 us	0 us	0 us

Update meter

Figure 2-5. Meter Error Window

To modify the calibration factor requires the correction to be entered in the percentage error. The percentage error is calculated as shown in [Equation 1](#).

$$\% \text{ Error} = \frac{\text{EVM Reading} - \text{Reference Meter Reading}}{\text{Reference Meter Reading}} \times 100\% \quad (1)$$

Type the percentage error in the corresponding box in **Meter Calibration** window. Click **Update meter**, and the updated calibration values are calculated and written to the EVM. The corresponding values are shown in the **Meter Calibration Factor** window.

2.3 Known Issues

The calibrator software is legacy software that operates with utility meters and has not been customized completely for embedded metering. The following list are the known issues with the existing software. These issues are scheduled to be fixed in the next version of calibrator software that is customized for embedded metering.

- Neutral Monitoring is shown in the **Meters Features** window as a supported feature, but this is not correct. When wire resistance compensation or inlet capacitor compensation is enabled, the calibrator interprets this as neutral monitoring support.
- Resistance value of wire resistance compensation and capacitance value of inlet capacitor compensation cannot be programmed with calibrator software.
- Voltage ac offset value cannot be written with the calibrator software. The value that is entered in the Voltage AC offset field is written to the Current AC offset, instead. The value that is entered in the Current AC offset box has no effect.
- DC Offset values cannot be written to the EVM with the calibration software. The current firmware takes the current and voltage dc offset value every time any calibration value is updated.
- Aggregate current in the [Meter Status] window always shows 10.0000 A.

Calibration Techniques

3.1 Introduction

The EVM is programmed with statistical calibration values that allow the EVM to measure and give roughly accurate readings; however the EVM has not been calibrated while shipped. To maximize accuracy and to compensate component and manufacturing tolerance it is necessary for the EVM to go through a calibration process. In this chapter the calibration techniques, required instruments, procedure and steps of calibration will be discussed.

3.2 Calibration Techniques

The calibration of the EVM is defined based on the front-end interface model as shown in [Figure 3-1](#).

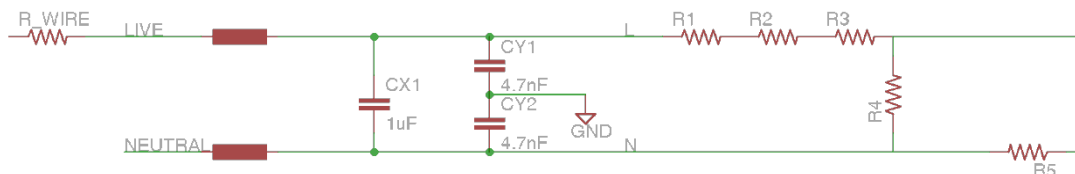


Figure 3-1. Front-End Interface Model

For the current design a 2 point calibration and characterization is required. VGAIN, IGAIN, PGAIN, CAP, RES, VDC_OFFSET, IDC_OFFSET are parameters that will be calibrated during the process; an estimated value is put into the memory during design/characterization to help to speed up calibration. (Note : Default CAP and RES should set to 0 before calibration).

VAC_OFFSET, IAC_OFFSET, PHASE_CORRECT are parameters that may not need calibration but a characterization would be sufficient for embedded metering application except for high accuracy of < 0.1%.

NOTE: The calibration values are written in one flash page in the EVM. Therefore, when a new value needs to be written, the whole page is erased. The provided GUI performs the read, modify, and write operation automatically. If a custom calibration method is implemented, the application must read the complete set of calibration values, update the modified fields, and write the complete set back to the EVM.

3.3 Calibration Procedures

3.3.1 Calibration of AC and DC Parameters

To calibrate the ac and dc parameters:

1. Calibrating VGAIN

- (a) Set to No / Lowest possible load.
- (b) Set VIN to line voltage.
- (c) Calculate the value for VGAIN with [Equation 2](#).

$$VGAIN_{n+1} = \frac{V_{REF}}{V_{UUT}} \times VGAIN_n$$

where

- $VGAIN_{n+1}$ is the new voltage calibration factor
- $VGAIN_n$ is the original voltage calibration factor
- V_{REF} is the reference meter voltage reading at the voltage set for VGAIN calibration
- V_{UUT} is the unit under test voltage reading at the voltage set for VGAIN calibration

Or if percentage error is used (as with the provided calibration software):

$$\%Error = \frac{V_{UUT} - V_{REF}}{V_{REF}} \times 100\% \quad (3)$$

- (d) Write and apply the calibrated VGAIN.

2. Calibrating IGAIN

- (a) Set VIN to line voltage.
- (b) Set to High / Highest possible load.
- (c) Calculate IGAIN value with [Equation 4](#).

$$IGAIN_{n+1} = \frac{I_{REF}}{I_{UUT}} \times IGAIN_n$$

where

- $IGAIN_{n+1}$ is the new current calibration factor
- $IGAIN_n$ is the original current calibration factor
- I_{REF} is the reference meter current reading at the current set for IGAIN calibration
- I_{UUT} is the unit under test current reading at the current set for IGAIN calibration

Or if percentage error is used (as with the provided calibration software):

$$\%Error = \frac{I_{UUT} - I_{REF}}{I_{REF}} \times 100\% \quad (5)$$

- (d) Write and apply the calibrated IGAIN.

3. Calibrating PGAIN

- (a) Same condition as above (the power factor at this calibration point should be 1 or very close to 1).
- (b) Note the percentage error on voltage at this point.
- (c) Calculate PGAIN gain so that power is of the same percentage error as voltage at this point.

$$PGAIN_{n+1} = \frac{P_{REF}}{P_{UUT}} \times PGAIN_n \times (1 - \% \text{ Error of } V_{UUT} \text{ at this load})$$

where

- $PGAIN_{n+1}$ is the new power calibration factor
- $PGAIN_n$ is the original power calibration factor
- P_{REF} is the reference meter power reading at the power set for PGAIN calibration
- P_{UUT} is the unit under test power reading at the power set for PGAIN calibration

(6)

Or if percentage error is used (as with the provided calibration software):

$$\% \text{ Error} = (1 + \% \text{ Error}) (1 - \% \text{ Error of } V_{UUT} \text{ at this load})$$

(7)

- (d) Write and apply the calibrated PGAIN.

3.3.2 Calibration of Compensation Resistance and Capacitance

After the calibration of VGAIN, IGAIN and PGAIN, follow these steps to calibrate the compensation to wire resistance and EMI capacitance.

1. Calibrating RES

- (a) Calculate RES with the give equation.

$$R_{WIRE} = \frac{V_{REF(I_{max})} - V_{UUT(I_{max})}}{I_{max} - I_{min}} \approx \frac{V_{REF(I_{max})} - V_{UUT(I_{max})}}{I_{max}}$$

where

- R_{WIRE} is the estimated wire resistance
- $V_{REF(I_{max})}$ is the reference meter voltage reading at the current set for IGAIN calibration
- $V_{UUT(I_{max})}$ is the unit under test voltage reading at the current set for IGAIN calibration
- I_{max} is the current set for IGAIN calibration

(8)

- (b) Write and apply the calibrated RES (note that the resistance is in units of 1/256 Ω)

2. Calibrating CAP

- (a) Set to No / Lowest possible load.
- (b) Set VIN to low line voltage.
- (c) Calculate CAP with the given equation.

$$C = \frac{1}{2\pi fV^2} \left(\sqrt{P_{APPARENT_REF}^2 - P_{ACTIVE}^2} - \sqrt{P_{APPARENT_UUT}^2 - P_{ACTIVE}^2} \right)$$

where

- C is the estimated EMI filter capacitance
- V is the voltage set for calibration of C
- $P_{APPARENT_REF}$ is the apparent power reading of the reference meter at the setting for this calibration
- $P_{APPARENT_UUT}$ is the apparent power reading of the unit under test at the setting for this calibration
- P_{ACTIVE} is the active power reading of the reference meter

(9)

- (d) Write and apply the calibrated CAP (note that the capacitance is in units of 1/64 μF).

3.3.3 Calibration of Current AC Offset

Current ac offset is the result of noise pickup and generated on the shunt resistor circuit causes an illusion of having a finite current flowing when there is actually not current flowing through the shunt. Although this noise current has no effect on the accuracy of the power reading, it contributes to the current reading and its accuracy, especially when current is small. To offset this, the EVM firmware has a mechanism to remove this from the current reading. The steps to calibrate this current offset are:

1. Apply nominal voltage to make sure the EVM operates
2. Remove all loading from the EVM
3. Take for example 100 current readings and took an average as I_NOISE (in A)
4. Calculate the current ac offset value with the equation (note that this is a high value even in case of a few mA of noise).

$$I_AC_OFFSET = \text{int} \left(I_NOISE \left(\frac{1024 \times 10^6}{I_GAIN} \right) \right)^2 \quad (10)$$

5. Write and apply the calibrated I_AC_OFFSET .

3.3.4 Calibration of Voltage AC Offset

The voltage ac offset in most case creates little effect to the voltage reading and thus does not require calibration.

3.3.5 Calibration of Phase Correction

1. Set test set to generate rated voltage and set to calibration current (for example, 5 A).
2. Make sure calibration is completed for PGAIN (at PF = 1).
3. Set test set to output at power factor 0.5 (+ or – is not important at this point).
4. Note the error.
5. Switch test set to output at power factor 0.5 in the other direction used in step 3.
6. Note the error.
7. At this point, the two errors should be approximately the same deviation but different direction from the calibrated power error at PF = 1 (for example, at PF = 1 the calibrated error is 0.1%, and if at PF = +0.5 the error reads approximately 0.5%, then at PF = -0.5 the error should read approximately -0.3%).
8. Adjust the phase correction with time deviation from the current phase correction (for example, if current phase correction is 13 μs and if 11 μs is desired, enter -2 into the phase correction box of the manual calibration window) until the error at ± 0.5 is minimum.

3.3.6 Calibration of DC Parameters

The design of EVM allows the dc measurement parameters to be calibrated automatically at the same time that ac is being calibrated. When the result of the last step of ac calibration (current ac offset, [Section 3.3.3](#)) are written, the dc measurement parameters are automatically updated.

Whenever the complete set of calibration values is read, the most current dc measurement parameters are also included in the set. When an update is done to other parameters, the dc measurement parameters are also updated. However, the value is most accurate when current is low (best with no current); thus, it is recommended to perform dc measurement parameter update after the I_AC_OFFSET calibration.

NOTE: The procedure of having the best dc measurement parameters will be modified and improved when the embedded metering customized calibration software is released.

Serial Communication Commands

4.1 Introduction

To enable the EVM to be a useful tool, an external interface is necessary. The EVM uses its UART port connected to a RS232 DB-9 connector to communicate to the external PC. This section describes the protocol of the EVM sending out readings and commands.

NOTE: You can change the provided source code to perform custom communication protocols, commands, responses, and functions.

4.2 Communication Protocol

During the operation, the EVM communicates with the external host via the serial communication port with these parameters:

- 9600 bps, No parity, 8 data bit, 1 stop bit

The EVM supports polling mode communication which gives access to the full range of functionality, reading of complete set of measurement results, reading and writing calibration factors.

4.2.1 Polling Mode

When EVM is running, the EVM waits for a command from the host through the serial port. When the command frame receives and passes the frame checking, the EVM interprets the command and performs the requested action. The EVM communicates with the host using a data frame format as shown in [Figure 4-1](#).

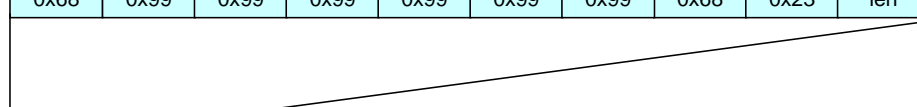
0	1	2	3	4	5	6	7	8	9
F_Start	Address						F_Start	C_cod	Length
0x68	0x99	0x99	0x99	0x99	0x99	0x99	0x68	0x23	len
									
0..9			10..len+9				len+10	len+11	
Header			Data				CS	0x16	

Figure 4-1. Polling Mode Data Frame Format

The frame starts with a 9 bytes header followed by a data field of 0 to 255 byte then a check sum byte and a frame ending byte.

The header starts with 0x68, followed by an address field which is fixed to 0x999999999999. User can modify the provided communication protocol source code (emeter-dlt645.c) to make the EVM respond to different address.

Followed by a fixed delimiter 0x68 and 0x23 is the length byte which indicates the number of bytes in the Data field. The end of the frame is a check sum byte which is a modulus 256 byte sum of each byte from the beginning of the header to the end of the data field followed by an end of frame marker 0x16.

4.2.1.1 Command and Respond Frame

Figure 4-2 and Figure 4-3 show the structure of the command and respond frame in polling mode. The command and respond frame has the same structure in the header and the frame end, the difference is in the data field. The first 2 bytes in the data field of a command is CMDH and CMDL which defines the command and the parameters that follows. After receiving a valid command frame from the host, the EVM responds to the host with a respond frame with the RSPH = CMDH and RSPL = CMDL | 0x80 (CMDL cannot use value larger than 0x7F). The respond data is defined by the command received.

See Section 4.3 for the definition of commands and the corresponding responses.

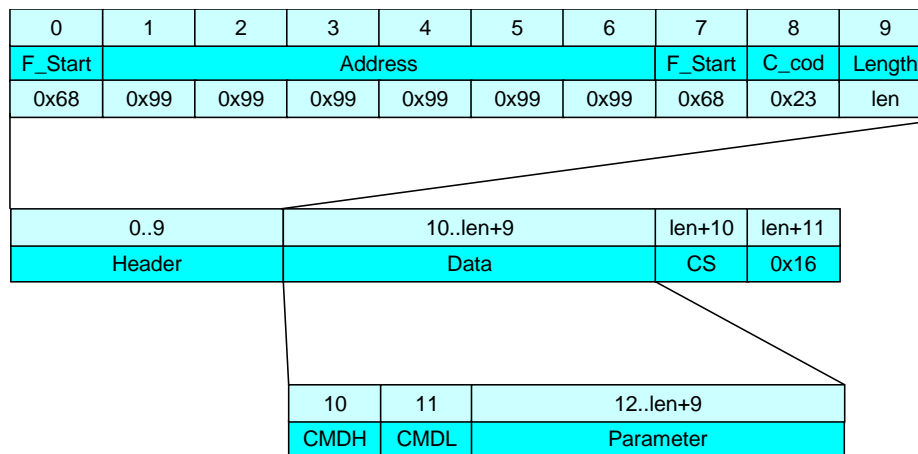
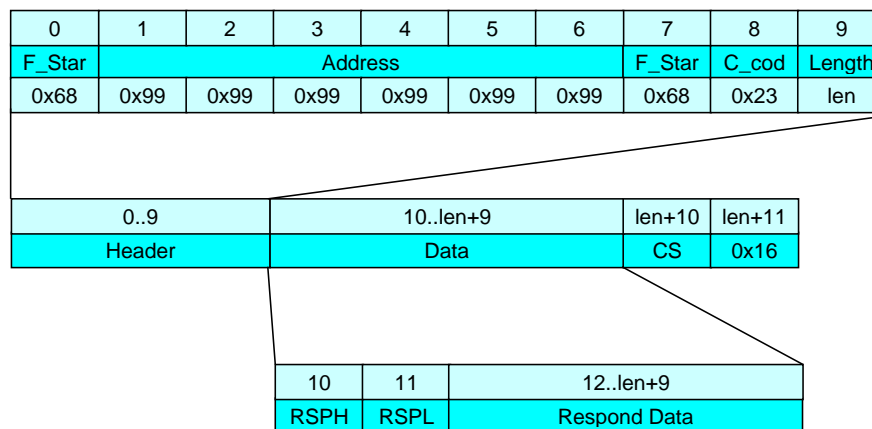


Figure 4-2. Polling Mode Command Frame Format



Respond is fitted into Data field as shown above
RSPH = CMDH, RSPL = CMDL | 0x80

Figure 4-3. Polling Mode Respond Frame Format

NOTE: If writing a custom protocol, make sure that the data buffer is sufficient to hold the data from the host.

4.3 Commands

4.3.1 HOST_CMD_GET_METER_NAME

Read the 32 byte meter name string as defined by #define METER_NAME in metrology-calibration-template.h

4.3.1.1 Command Format

HOST_CMD_GET_METER_NAME						
	Command			Respond		
LEN	2			34		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0x52	0	U8	0x52
CMDL	1	U8	0x00	1	U8	0x80
				2	U8(32)	32-byte meter name

4.3.2 HOST_CMD_GET_METER_VER

Read the four 32-bit version numbers as defined by the following lines in metrology-calibration-template.h.

```
#define METER_SOFTWARE_VERSION
#define METER_HARDWARE_VERSION
#define METER_METROLOGY_VERSION
#define METER_PROTOCOL_VERSION
```

4.3.2.1 Command Format

HOST_CMD_GET_METER_VERSION						
	Command			Respond		
LEN	2			18		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0x53	0	U8	0x53
CMDL	1	U8	0x00	1	U8	0x80
				2	U8(4)	4-byte software version
				6	U8(4)	4-byte hardware version
				10	U8(4)	4-byte metrology version
				14	U8(4)	4-byte protocol version

4.3.3 HOST_CMD_GET_METER_CONFIGURATION

This commands the EVM to return the parameter and the functionality the EVM supports

4.3.3.1 Command Format

HOST_CMD_GET_METER_CONFIGURATION						
	Command			Respond		
LEN	2			20		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0x56	0	U8	0x56
CMDL	1	U8	0x00	1	U8	0x80
				2	U8	Number of Phases
				3	U8	Features 0 (see Table 4-1)
				4	U8	Features 1 (see Table 4-1)
				5	U8	Features 2 (see Table 4-1)
				6	U8	Features 3 (see Table 4-1)
				7	U8	0x00
				8	U16	Mains Nominal Frequency
				10	U16	Mains Nominal Voltage
				12	U16	Mains Basis Current
				14	U16	Mains Maximum Current
				16	U32	100 times the sample rate

4.3.3.2 Parameter Definition

Table 4-1. HOST_CMD_GET_METER_CONFIGURATION Respond Parameters

Features 0		Features 2	
Bit 7	Reserved	Bit 7	Measures quadrature reactive power
Bit 6	Reserved	Bit 6	Measures mains frequency
Bit 5	Reserved	Bit 5	Measures power factor
Bit 4	Reserved	Bit 4	Measures IRMS
Bit 3	Reserved	Bit 3	Measures VRMS
Bit 2	Wire Resistance Compensation Support	Bit 2	Measures Apparent Power
Bit 1	Inlet Capacitor Compensation Support	Bit 1	Measures trigonometric reactive power
Bit 0	Neutral Monitor Support	Bit 0	Measures active Power
Features 1		Features 3	
Bit 7	Multi-rate Support	Bit 7	Measures sag and swell
Bit 6	Undefined	Bit 6	Measures current THD
Bit 5	Temperature Support	Bit 5	Measures voltage THD
Bit 4	Corrected RTC Support	Bit 4	Measures fundamental IRMS
Bit 3	RTC Support	Bit 3	Measures fundamental VRMS
Bit 2	Dynamic Phase Correction Support	Bit 2	Measures fundamental active power
Bit 1	Auto Report Support	Bit 1	Measures fundamental reactive power
Bit 0	Limp Mode Support	Bit 0	Undefined

4.3.4 HOST_CMD_GET_RTC

This command does not read anything related to RTC as there is no RTC in the EVM yet one of the data read from the meter by this command is the temperature. Hence this command is considered a command to read the temperature of the EVM using the internal temperature sensor on the MSP430i2040

4.3.4.1 Command Format

HOST_CMD_GET_RTC						
	Command			Respond		
LEN	2			10		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0x59	0	U8	0x59
CMDL	1	U8	0x00	1	U8	0x80
				2	U8(4)	6 bytes dummy data
				8	S16	Temperature in 0.01°C

4.3.5 HOST_CMD_ALIGN_WITH_CALIBRATION_FACTORS

This command cause the EVM to reload the calibration factors from flash into the operation of its measurement activities

4.3.5.1 Command Format

HOST_CMD_ALIGN_WITH_CALIBRATION_FACTORS						
	Command			Respond		
LEN	2			2		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0x5A	0	U8	0x5A
CMDL	1	U8	0x00	1	U8	0x80

4.3.6 HOST_CMD_SET_PASSWORD

This command passes from the host to the EVM the password to enable calibration mode which allows calibration and other functions to be executed. If auto report mode is enabled this command also disable the auto report mode and sets the EVM to polling mode. In the example code, the password default is Password 1 = 0x1234, Password 2 = 0x5678, Password 3 = 0x9ABC, Password 3 = 0xDEF0 (the password can be changed by modifying the entry in emeter-template.h followed by a recompile).

4.3.6.1 Command Format

HOST_CMD_SET_PASSWORD						
	Command			Respond		
LEN	10			2		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0x60	0	U8	0x60
CMDL	1	U8	0x00	1	U8	0x80
	2	U16	Password 1			
	4	U16	Password 2			
	6	U16	Password 3			
	8	U16	Password 4			

4.3.7 HOST_CMD_GET_READINGS_PHASE_n

This command reads from the EVM the latest measurements. If CMDL is specified a value other than 0x00 the CMDL value will be in priority to determine the phase number.

NOTE: Only CMDH = 0x61 is supported by this EVM.

4.3.7.1 Command Format

HOST_CMD_GET_READINGS_PHASE_n						
	Command			Respond		
LEN	2			34		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0x6n (n = 1, 2, 3)	0	U8	0x6n (n = 1, 2, 3)
CMDL	1	U8	0xdp (always 0x00 in this EVM)	1	U8	0x80 0xdp, d=device[0-7], p=phase[1-15]
				2	S32	Voltage in mV
				6	S32	Current in μ A
				10	S32	Active power in mW
				14	S32	Reactive power in mW
				18	S32	Apparent power in mW
				22	S16	Power factor in 0.001
				24	S16	Frequency in 0.01 Hz
				26	S32	Voltage channel dc offset
				30	S32	Current channel dc offset

4.3.8 HOST_CMD_GET_EXTRA_READINGS_PHASE_n

This command reads from the EVM the latest extra measurements. If CMDL is specified a value other than 0x00 the CMDL value will be in priority to determine the phase number.

NOTE: Only CMDH = 0x69 is supported by this EVM.

4.3.8.1 Command Format

HOST_CMD_GET_EXTRA_READINGS_PHASE_n						
	Command			Respond		
LEN	2			34		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0x68 + n (n = 1, 2, 3)	0	U8	0x68 + n (n = 1, 2, 3)
CMDL	1	U8	0xdp (always 0x00 in this EVM)	1	U8	0x80 0xdp, d=device[0..7], p = phase [1..15]
				2	S32	Fundamental active power in mW
				6	S32	Fundamental reactive power in mW
				10	S32	Fundamental voltage in mV
				14	S32	Fundamental current in μ A
				18	U16	Voltage THD in 0.01%
				20	U16	Current THD in 0.01%
				22		Next 12 bytes reserved

4.3.9 HOST_CMD_SUMCHECK_MEMORY

This command requests the EVM to perform a calculation and return of 16 bit checksum from the start flash address to the end flash address (inclusive).

4.3.9.1 Command Format

HOST_CMD_SUMCHECK_MEMORY						
	Command			Respond		
LEN	10			4		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0x75	0	U8	0x75
CMDL	1	U8	0x00	1	U8	0x80
	2	U32	Start Flash Address	2	U16	Check sum
	6	U32	End Flash Address			

4.3.10 HOST_CMD_CLEAR_CALIBRATION_DATA

Since the calibration data is stored in a flash page, the flash page needs to be cleared before any calibration data could be written. This command requests the EVM to erase the flash that contains the calibration data.

NOTE: Read back and save in the host the complete set of calibration data before executing this command or else the calibration data will not be able to retrieve again.

4.3.10.1 Command Format

HOST_CMD_CLEAR_CALIBRATION_DATA						
	Command			Respond		
LEN	2			2		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0xD0	0	U8	0xD0
CMDL	1	U8	0x00	1	U8	0x80

4.3.11 HOST_CMD_SET_CALIBRATION_PHASE_n

This commands the EVM to set the calibration values with the values listed. Make sure the flash page contains the calibration values is read, saved and erase before executing this command.

NOTE: Only CMDH = 0xD1 is supported by this EVM.

4.3.11.1 Command Format

HOST_CMD_SET_CALIBRATION_PHASE_n						
	Command			Respond		
LEN	30			2		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0xD0 + n	0	U8	0xD0 + n (n = 1, 2, 3)
CMDL	1	U8	0xdp (always 0x00 in this EVM)	1	U8	0x80 0xdp, d=device[0..7], p = phase [1..15]
	2	S16	Voltage channel dc offset			
	4	U16	Inlet capacitance in 1/64 μ F			
	6	S32	Current channel dc offset			
	10	U32	Voltage channel ac offset			
	14	U32	Current channel ac offset			
	18	S16	Phase correction in 1/1024 Ts			
	20	U16	Vrms scaling factor			
	22	U16	Wire resistance in 1/256 Ω			
	24	U16	Irms scaling factor			
	26	U16	0x0000 reserved			
	28	U16	Power scaling factor			

4.3.12 HOST_CMD_GET_CALIBRATION_PHASE_n

This command reads the calibration value from the EVM.

NOTE: Only CMDH = 0xD6 is supported by this EVM.

4.3.12.1 Command Format

HOST_CMD_GET_CALIBRATION_PHASE_n						
	Command			Respond		
LEN	2			30		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0xD5 + n	0	U8	0xD5 + n (n = 1, 2, 3)
CMDL	1	U8	0xdp (always 0x00 in this EVM)	1	U8	0x80 0xdp, d=device[0..7], p = phase[1..15]
				2	S16	Voltage channel dc offset
				4	U16	Inlet capacitance in 1/64 μ F
				6	S32	Current channel dc offset
				10	U32	Voltage channel ac offset
				14	U32	Current channel ac offset
				18	S16	Phase correction in 1/1024 Ts
				20	U16	Vrms scaling factor
				22	U16	Wire resistance in 1/256 Ω
				24	U16	Irms scaling factor
				26	U16	Reserved
				28	U16	Power scaling factor

4.3.13 HOST_CMD_SET_CALIBRATION_EXTRAS

This commands the EVM to set the extra calibration values with the values listed. Make sure the flash page contains the calibration values is read, saved and erase before executing this command.

4.3.13.1 Command Format

HOST_CMD_SET_CALIBRATION_EXTRAS						
	Command			Respond		
LEN	10			2		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0xD5	0	U8	0xD5
CMDL	1	U8	0x00	1	U8	0x80
	2	U16	Calibration status			
	4	U16	Intercept temperature			
	6	U16	Temperature intercept			
	8	U16	Temperature slope / degree			

4.3.14 HOST_CMD_GET_CALIBRATION_EXTRAS

This command reads the extra calibration value from the EVM

4.3.14.1 Command Format

HOST_CMD_GET_CALIBRATION_EXTRAS						
	Command			Respond		
LEN	2			10		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0xDA	0	U8	0xDA
CMDL	1	U8	0x00	1	U8	0x80
				2	U16	Calibration status
				4	U16	Intercept temperature
				6	U16	Temperature intercept
				8	U16	Temperature slope / degree

Firmware and Embedded Metering Library API

5.1 Introduction

The EVM comes with pre-tested firmware that enables the EVM to operate. The source code together with the embedded metering library, as an example application is provided. In this chapter the structure of the firmware, the summary of the APIs of the embedded metering library will be discussed.

5.2 Firmware Structure

The firmware the comes with the EVM is designed to use a layered approach isolating the user from the details of metrology and the associated computations involved so that the programming work is simplified.

The firmware is partitioned into three main blocks:

1. The application
 - System setup and initialization
 - Main loop
 - Communication protocol and command handling
 - Non-volatile parameters preset and manipulation
2. The metrology computation engine, packaged into a library named `emeter-metrology-i2041.r43`

NOTE: The source code of this library is not included in the package. Contact the Texas Instrument Sales Team in your area if the source code is needed.

- ADC setup
 - Parameter initializations
 - Sample based background processing
 - Reporting cycle based foreground processing
 - Reading application interface
3. The toolkit packaged into a library named `emeter-toolkit-i2041.r43`
 - Low level computation routines

5.3 Toolkit Package

The low-level computation engine provides speed optimized processing for common arithmetic operations in metrology. Functions include:

- 48-bit accumulation
- 24-bit high pass filtering and dc offset removal (dc mode)
- 16-bit high pass filtering and dc offset removal (dc mode)
- Reference pure sine wave generation
- 48-bit by 16-bit division
- 16-bit by 16-bit and 32-bit by 16-bit multiplication
- 16-bit, 32-bit, and 64-bit square root and integer square root
- 16-bit by 16-bit multiply-accumulate into 48-bit accumulator
- 16-bit by 24-bit multiply-accumulate into 64-bit accumulator

- Q1.15 fixed-point number multiply
- 16-bit square-and-accumulate into 48-bit accumulator
- 24-bit square-and-accumulate into 64-bit accumulator

5.4 Metrology Computation Engine

The metrology computation engine performs the actual sampling and computation based on the information collected from the voltage and current ADC channels. The computation engine performs its actions in a time critical background processing and a less time critical foreground processing.

The background processing is triggered by the ADC at the sample rate. It is running in the interrupt services routine of the ADC and is processed automatically.

The foreground processing is triggered by the completion of background processing at the reporting/update rate. Background process sets a flag PHASE_STATUS_NEW_LOG in variable phase_state to indicate there is data ready to be processed by the foreground process. The application then needs to monitor this flag to trigger the foreground process by calling to calculate_phase_readings().

In the actual computation, the following formulas are used in the metrology computation:

$$V_{RMS} = VGAIN \times \sqrt{\frac{1}{N} \sum_{i=1}^N V_{samp}(i) \times V_{samp}(i)} \quad (11)$$

$$I_{RMS} = IGAIN \times \sqrt{\frac{1}{N} \sum_{i=1}^N I_{samp}(i) \times I_{samp}(i)} \quad (12)$$

$$P_{active} = PGAIN \times \frac{1}{N} \sum_{i=1}^N V_{samp}(i) \times I_{samp}(i) \quad (13)$$

$$P_{reactive} = PGAIN \times \frac{1}{N} \sum_{i=1}^N V_{samp,90}(i) \times I_{samp}(i) \quad (14)$$

$$P_{apparent} = V_{RMS} \times I_{RMS} \quad (15)$$

$$PF = \cos \phi = \frac{P_{active}}{P_{apparent}} \quad (16)$$

$$V_{RMS\ fund} = VGAIN \times \sqrt{\frac{1}{N} \sum_i V_{samp}(i) \times V_{pure}(i)} \quad (17)$$

$$P_{active\ fund} = PGAIN \times \frac{1}{N} \sum_i I_{samp}(i) \times V_{pure}(i) \quad (18)$$

$$P_{reactive\ fund} = PGAIN \times \frac{1}{N} \sum_i I_{samp}(i) \times V_{pure(\pi/2)}(i) \quad (19)$$

$$V_{THD} = \frac{\sqrt{V_{RMS}^2 - V_{RMS\ fund}^2}}{V_{RMS\ fund}} \quad (20)$$

$$I_{RMS\ fund} = \frac{\sqrt{P_{active\ fund}^2 + P_{reactive\ fund}^2}}{V_{RMS\ fund}} \quad (21)$$

$$I_{THD} = \frac{\sqrt{I_{RMS}^2 - I_{RMS\ fund}^2}}{I_{RMS\ fund}} \quad (22)$$

5.4.1 Background Process

Time-critical sample-based processes are performed in the background. The process is started by a trigger, at the sample rate, sampling complete interrupt. The background process is:

1. Capture data from voltage and current ADC channels
2. Perform voltage sample processing
 - (a) Voltage channel high pass filtering/dc offset removal
 - (b) Wire resistance compensation
 - (c) Integration of square of voltage(i) sample
 - (d) Integration of voltage(i) to fundamental voltage
 - (e) Adjust phase correction to voltage(i) sample
3. Current sample processing
 - (a) Current channel high pass filtering/dc offset removal
 - (b) Adjust phase correction to current(i) sample
 - (c) Integration of square of current(i) sample
 - (d) Capacitor compensation
4. Power processing
 - (a) Integrate voltage(i) \times current(i)
 - (b) Integrate quadrature voltage(i) \times current(i)
 - (c) Integrate fundamental active power
 - (d) Integrate fundamental reactive power
5. Line frequency processing
 - (a) Update sample count since last report
 - (b) Determine the presence of a valid zero crossing
 - (c) Compute the period between zero crossing
 - (d) Check for dc mode or ac mode and switch over
6. Trigger foreground process
 - (a) Setting flag to indicate there is data for the foreground to process

5.4.2 Foreground Process

After the background has collected sufficient data, the flag PHASE_STATUS_NEW_LOG is set by the background. The main loop should then check the status of this flag and call the foreground process calculate_phase_readings() to perform the rest of the computation to deliver the measurement readings. The foreground process is:

1. Power processing
 - (a) Calculate the active power by scaling with number of samples and power scaling factor
 - (b) Calculate the reactive power by scaling with number of samples and power scaling factor
 - (c) Calculate fundamental active power by scaling with number of samples and power scaling factor
 - (d) Calculate fundamental reactive power by scaling with number of samples and power scaling factor
2. Voltage processing
 - (a) Calculate the RMS voltage by scaling with number of samples, perform square root and multiply with the voltage scaling factor
 - (b) Calculate the fundamental RMS voltage by scaling with number of samples, perform square root and multiply with the voltage scaling factor
- (c) Current processing
 - (i) Calculate the RMS current by scaling with number of samples, perform square root and multiply with the current scaling factor
 - (ii) Calculate fundamental RMS current from power and voltage
 - (iii) Calculate the apparent power by multiplying the root mean square voltage and root means square current
- (iv) Other processing
 - (i) Calculate the power factor
 - (ii) Calculate the frequency of the ac line

5.5 Embedded Metering Library API

In this section the API of the embedded metering library will be introduced.

The metrology library communicates to the user application utilizing function calls, callbacks and application level calibration functions. Functions calls are the actual call to access the functionalities and readings provided by the embedded metering. Callbacks are predefined functions called from the embedded metering library to the user application. Application-level calibration functions are the functions from user application to access calibration parameters and for the definition of default calibration parameter.

5.5.1 Embedded Metering Library Function calls

Functions calls to interface to metrology library is defined in metrology-readings.h and metrology-foreground.h, which provides access to:

- Metrology engine control, which includes:
 - Initialize the metrology library and ADC hardware
 - Initialize the metrology library with default calibration parameter
- Performing metrology calculations with data collected by background processing
- Reading result from metrology calculations

5.5.1.1 Functions for Metrology Engine Control

5.5.1.1.1 *int metrology_init (void)*

Check if PGAIN in the calibration parameter memory is 0xFFFF, if true copy the default calibration parameter to calibration parameter memory. Skip copying otherwise.

Parameter: none

Return: 0xFFFF if calibration parameter memory is initialized with default calibration parameters, 0x0000 if not.

5.5.1.1.2 *int metrology_init_from_nv_data (void)*

Initialize the metrology engine's dc filter with calibration parameter loaded to proper memory variables, initialize the scaling factor of wire resistance compensation and scaling factor of inlet capacitor compensation.

Parameter: none

Return: Always 0x0000

5.5.1.1.3 *void align_metrology_with_calibration_data (void)*

reinitialize the metrology related ADC channels with proper phase correction.

Parameter: none

Return: none

5.5.1.1.4 *void metrology_switch_to_normal_mode (void)*

Initialize the metrology related ADC channels with proper phase correction, Set system to run in normal mode

Parameter: none

Return: none

5.5.1.1.5 void metrology_init_analog_front_end_normal_mode (void)

Initialize the hardware of all metrology related ADC channels

Parameter: none

Return: none

5.5.1.1.6 void metrology_disable_analog_front_end (void)

Disable the hardware of all metrology related ADC channels

Parameter: none

Return: none

5.5.1.1.7 Procedure for Metrology Engine Initialization

1. metrology_init () - initialize uninitialized calibration parameter with default value
2. metrology_disable_analog_front_end () - prevent metrology ADCs from generating interrupt during initialization
3. metrology_init_from_nv_data () – initialize dc filter
4. Do other setup operations that requires the ADC interrupt be disabled
5. metrology_switch_to_normal_mode () – switch to normal, initialize metrology related ADCs

5.5.1.2 Functions for Calculate and Reading the Readings

5.5.1.2.1 power_t calculate_phase_readings (void)

This function must be called by application explicitly when an indication is received to ensure correct readings. This function performs the non-time critical metrology calculations with data collected in background process. Upon returning all metrology reading is updated.

Parameter: none

Return: Active Power reading

5.5.1.2.2 power_t active_power (int ph)

Get the latest active power reading of phase ph

Parameter: ph = 1

Return: latest active power of phase ph

5.5.1.2.3 power_t reactive_power (int ph)

Returns the latest reactive power reading of phase ph. The returned value of this function is invalid when the EVM is operating in dc measurement mode.

Parameter: ph = 1

Return: latest reactive power of phase ph

5.5.1.2.4 power_t apparent_power (int ph)

Returns the latest calculated apparent power reading of phase ph. The returned value of this function is invalid when the EVM is operating in dc measurement mode.

Parameter: ph = 1

Return: latest apparent power of phase ph

5.5.1.2.5 power_t fundamental_active_power(int ph)

Returns the latest calculated fundamental active power reading of phase ph. The returned value of this function is invalid when the EVM is operating in dc measurement mode.

Parameter: ph = 1

Return: latest fundamental active power

5.5.1.2.6 power_t fundamental_reactive_power (int ph)

Returns the latest calculated fundamental reactive power reading of phase ph. The returned value of this function is invalid when the EVM is operating in dc measurement mode.

Parameter: ph = 1

Return: latest fundamental reactive power

5.5.1.2.7 Power_factor_t power_factor (int ph)

Returns the latest power factor reading of phase ph. The returned value of this function is invalid when the EVM is operating in dc measurement mode.

Parameter: ph = 1

Return: latest power factor of phase ph

5.5.1.2.8 rms_voltage_t rms_voltage (int ph)

Returns the latest root mean square voltage of phase ph

Parameter: ph = 1

Return: latest root mean square voltage measured on phase ph

5.5.1.2.9 rms_voltage_t fundamental_rms_voltage (int ph)

Returns the latest fundamental root mean square voltage of phase ph

Parameter: ph = 1

Return: latest fundamental root mean square voltage of phase ph

5.5.1.2.10 thd_t voltage_thd (int ph)

Returns the latest THD on voltage of phase ph

Parameter: ph = 1

Return: latest THD on voltage of phase

5.5.1.2.11 rms_current_t rms_current (int ph)

Returns the latest root mean square current of phase ph

Parameter: ph = 1

Return: latest root mean square current measured on phase ph

5.5.1.2.12 rms_current_t fundamental_rms_current (int ph)

Returns the latest fundamental root mean square current of phase ph

Parameter: ph = 1

Return: latest fundamental root mean square current of phase ph

5.5.1.2.13 *thd_t current_thd (int ph)*

Returns the latest THD on current of phase ph

Parameter: ph = 1

Return: latest THD on current of phase

5.5.1.2.14 *int16_t mains_frequency (int ph)*

Returns the measured ac frequency of phase ph. The returned value of this function is invalid when the EVM is operating in dc measurement mode.

Parameter: ph = 1

Return: latest ac frequency measured on phase ph

5.5.1.2.15 *uint16_t phase_status (int ph)*

Returns the status word of phase ph.

Parameter: ph = 1

Return: status word of phase ph

5.5.2 *Embedded Metering Library Callbacks*

The embedded metering library uses callbacks to call to functions defined in the application program as an indication of events. The callbacks are declared in `emeter-metrology.h` and the function implementation must be provided in the application code (even an empty function is acceptable, in case there is no need to process the event). In the provided application example, the callback functions are implemented in `emeter-main.c`.

5.5.2.1 *void BACKGROUND_PROCESS_ON (void)*

Called when background process starts.

5.5.2.2 *void BACKGROUND_PROCESS_OFF (void)*

Called when background process finished.

5.5.2.3 *void FOREGROUND_PROCESS_ON (void)*

Called when foreground process starts.

5.5.2.4 *void FOREGROUND_PROCESS_OFF (void)*

Called when foreground process finished.

5.5.2.5 *void ZERO_CROSS_ON (void)*

Called when the background process detects that the voltage signal is zero crossing from positive to negative.

5.5.2.6 *void ZERO_CROSS_ON (void)*

Called when the background process detects that the voltage signal is zero crossing from negative to positive.

5.5.2.7 *void AC_MODE_ON (void)*

Called when the background process enters ac measurement mode.

5.5.2.8 void AC_MODE_OFF (void)

Called when the background process exits ac measurement mode.

5.5.2.9 void DC_MODE_ON (void)

Called when the background process enters dc measurement mode.

5.5.2.10 void DC_MODE_OFF (void)

Called when the background process exits dc measurement mode.

5.5.2.11 void active_energy_pulse_start (void)

Called when the background process is generating an energy pulse and the energy pulse output is going from idle to pulsing state.

5.5.2.12 void active_energy_pulse_end (void)

Called when the background process is generating an energy pulse and the energy pulse output is going from pulsing to idle.

5.5.3 Application-Level Calibration Functions

5.5.3.1 Functions for Reading and Writing Calibration Parameters

5.5.3.1.1 int get_calibration_status (void)

Returns the calibration status of the meter

The returned value is not defined by the embedded metering library. Interpretation is to be defined by application

5.5.3.1.2 void set_calibration_status (int value)

Sets the calibration status to a specific value into the calibration parameter memory

The parameter value is not defined by the embedded metering library. Interpretation is to be defined by application

5.5.3.1.3 int clear_calibration_data (void)

Clear all calibration data in calibration parameter memory

NOTE: Must do this before writing any calibration data since all calibration data is stored in the same flash page. A reading and backup of all calibration parameter should be done before clearing the calibration data. Writing any calibration data should write the modified values together with all other values that is backed up and not modified.

5.5.3.1.4 int16_t get_temperature_intercept (void)

Get the calibration value of the temperature intercept.

Parameter: none

Return: The calibrated ADC reading for calibration temperature in calibration parameter memory

5.5.3.1.5 *int16_t get_temperature_slope (void)*

Get the calibration value of the temperature slope

Parameter: none

Return: The calibrated ADC reading increment per 0.1 degree Celsius in calibration parameter memory

5.5.3.1.6 *void set_temperature_parameters (int16_t temperature_at_calibration, int16_t temperature_sensor_intercept, int16_t temperature_sensor_slope)*

Write temperature related calibration value into calibration parameter memory

Parameter:

- `temperature_at_calibration` : the temperature in 0.1 degree Celsius which the calibration is done.
- `temperature_sensor_intercept` : the ADC reading at calibration temperature.
- `temperature_sensor_slope` : the ADC reading increment per 0.1 degree Celsius.

Return: none

5.5.3.1.7 *calibration_scaling_factor_t get_P_scaling (int phx)*

Get power scaling factor of phase phx

Parameter: `phx = 1`

Return: Power scaling factor in calibration parameter memory (see `metrology-types.h` for type definition of `calibration_scaling_factor_t`)

5.5.3.1.8 *void set_P_scaling (int phx, calibration_scaling_factor_t value)*

Write power scaling factor of phase phx with value

Parameter:

- `phx = 1`
- `value` = the value to be written to power scaling factor in calibration parameter memory

Return: none

5.5.3.1.9 *calibration_scaling_factor_t get_V_rms_scaling (int phx)*

Get the voltage scaling factor of phase phx

Parameter: `phx = 1`

Return: Voltage scaling factor in calibration parameter memory (see `metrology-types.h` for type definition of `calibration_scaling_factor_t`)

5.5.3.1.10 *void set_V_rms_scaling (int phx, calibration_scaling_factor_t value)*

Write the voltage scaling factor of phase phx with value

Parameter:

- `phx = 1`
- `value` = the value to be written to voltage scaling factor in calibration parameter memory

Return: none

5.5.3.1.11 *int16_t get_v_dc_estimate (int phx)*

Get the active (dynamic) voltage channel ADC dc offset of phase phx

Parameter: `phx = 1`

Return: the dynamic voltage channel dc offset, in ADC counts, of phase phx

5.5.3.1.12 *int16_t get_initial_v_dc_estimate(int phx)*

Get the initial voltage channel ADC dc offset of phase phx

Parameter: phx = 1

Return: the voltage channel dc offset of phase phx, in ADC counts, stored in calibration parameter memory

5.5.3.1.13 *void set_v_dc_estimate(int phx, int16_t value)*

Write the voltage channel ADC dc offset of phase phx with value

Parameter:

- phx = 1
- value : the value to be written to voltage channel dc offset for phase phx, in ADC counts, in calibration parameter memory

Return: none

5.5.3.1.14 *int32_t get_v_ac_offset(int phx)*

Get the voltage channel ac offset of phase phx

Parameter: phx = 1

Return: the voltage channel ac offset of phase phx, in square of ADC counts, to be stored in calibration parameter memory

5.5.3.1.15 *void set_v_ac_offset(int phx, int32_t value)*

Write the voltage channel ac offset of phase phx with value. Value should be calculated as

$$V_AC_OFFSET = \text{int} \left(V_NOISE \left(\frac{1024 \times 10^3}{VGAIN} \right) \right)^2 \quad (23)$$

Parameter:

- phx = 1
- value : the value to be written to voltage channel ac offset for phase phx, in square of ADC counts, in calibration parameter memory

Return: none

5.5.3.1.16 *calibration_scaling_factor_t get_I_rms_scaling(int phx);*

Get the current scaling factor of phase phx

Parameter: phx = 1

Return: Current scaling factor in calibration parameter memory (see metrology-types.h for type definition of calibration_scaling_factor_t)

5.5.3.1.17 *void set_I_rms_scaling(int phx, calibration_scaling_factor_t value);*

Write the current scaling factor of phase phx with value

Parameter:

- phx = 1
- value = the value to be written to current scaling factor in calibration parameter memory

Return: none

5.5.3.1.18 **int32_t get_i_dc_estimate(int phx);**

Get the active (dynamic) current channel ADC dc offset of phase phx

Parameter: phx = 1

Return: the dynamic current channel dc offset of phase phx in ADC counts

5.5.3.1.19 **int32_t get_initial_i_dc_estimate(int phx)**

Get the initial current channel ADC dc offset of phase phx

Parameter: phx = 1

Return: the current channel dc offset of phase phx, in ADC counts, stored in calibration parameter memory

5.5.3.1.20 **void set_i_dc_estimate(int phx, int32_t value);**

Write the current channel ADC dc offset of phase phx with value

Parameter:

- phx = 1
- value : the value to be written to current channel dc offset for phase phx, in ADC counts, in calibration parameter memory

Return: none

5.5.3.1.21 **int32_t get_i_ac_offset(int phx);**

Get the current channel ac offset of phase phx

Parameter: phx = 1

Return: the current channel ac offset, in square of ADC counts, of phase phx to be stored in calibration parameter memory

5.5.3.1.22 **void set_i_ac_offset(int phx, int32_t value);**

Write the current channel ac offset of phase phx with value. Value should be calculated as

$$I_AC_OFFSET = \text{int} \left(I_NOISE \left(\frac{1024 \times 10^6}{IGAIN} \right) \right)^2 \quad (24)$$

Parameter:

- phx = 1
- value : in square of ADC counts, the value to be written to current channel ac offset for phase phx in calibration parameter memory

Return: none

5.5.3.1.23 **uint16_t get_compensate_capacitor_value (int phx)**

Get EMI filter capacitor value of phase phx

Parameter: phx = 1

Return: the capacitance to be compensate (in 1/64 μF unit) in calibration parameter memory

5.5.3.1.24 void set_compensate_capacitor_value (int phx, uint16_t value)

Write EMI filter capacitor value of phase phx with value

Parameter:

- phx = 1
- the capacitance to be written (in 1/64 μ F unit) to calibration parameter memory

Return: none

5.5.3.1.25 uint16_t get_compensate_resistance (int phx)

Get wire resistance value of phase phx

Parameter: phx = 1

Return: the wire resistance to be compensate (in units of 1/256 Ω) in calibration parameter memory

5.5.3.1.26 void set_compensate_resistance (int phx, uint16_t value)

Write wire resistance value of phase phx with value

Parameter:

- phx = 1
- the wire resistance to be written (in units of 1/256 Ω) to calibration parameter memory

Return: none

5.5.3.1.27 int16_t get_phase_corr (int phx)

Get phase correction of phx

Parameter: phx = 1

Return: the phase correction value (in unit of 97.65625 μ s) in calibration parameter memory

5.5.3.1.28 void set_phase_corr (int phx, int16_t value)

Write phase correction of phase phx with value

Parameter:

- phx = 1
- the phase correction value (in unit of 97.65625 μ s) to be written to calibration parameter memory

Return: none

5.5.3.2 Setting Default Calibration Parameters

Being part of metrology but not the metrology itself, the calibration parameter default values reading and manipulation functions is provided as source code in the application level. The user defined default parameter is defined in metrology-calibration-template.h and metrology-capacitor-compensation.h.

NOTE: The file metrology-calibration-defaults.c provides the source code level support for default values to be put into the flash. Unless the code is thoroughly understood do not modify the code in this file.

5.5.3.2.1 DEFAULT_TEMPERATURE_INTERCEPT

ADC reading at the calibration temperature in 16 bits scale

5.5.3.2.2 DEFAULT_TEMPERATURE_SLOPE

ADC counts per 0.1 degree Celsius increment

5.5.3.2.3 DEFAULT_ROOM_TEMPERATURE

Calibration temperature in 0.1 degree C unit

5.5.3.2.4 DEFAULT_BASE_PHASE_A_CORRECTION

The phase correction between voltage and current channel due to hardware

5.5.3.2.5 DEFAULT_P_SCALE_FACTOR_A

Power scaling factor

5.5.3.2.6 DEFAULT_V_RMS_SCALE_FACTOR_A

Voltage scaling factor

5.5.3.2.7 DEFAULT_V_DC_ESTIMATE_A

DC offset of voltage channel ADC

5.5.3.2.8 DEFAULT_V_AC_OFFSET_A

Square of estimated noise level on voltage channel ADC (so small to be noticeable, usually put 0)

5.5.3.2.9 DEFAULT_I_RMS_SCALE_FACTOR_A

Current scaling factor

5.5.3.2.10 DEFAULT_I_DC_ESTIMATE_A

DC offset of current channel ADC

5.5.3.2.11 DEFAULT_I_AC_OFFSET_A

Square of estimated noise level on voltage channel ADC

Calculated as

$$DEFAULT_I_AC_OFFSET = \left(\frac{1024 \times 10^6}{Current\ Scaling\ Factor} \times Estimated\ noise\ level \right)^2 \quad (25)$$

5.5.3.2.12 DEFAULT_EMI_FILTER_CAP_UF_A

EMI filter capacitance in units of 1/64 μ F, Maximum 1023 (15.984375 μ F)

Putting value greater than 0x8000 will result in no EMI filter compensation done

5.5.3.2.13 DEFAULT_WIRE_RESISTANCE_A

Estimate wire resistance in units of 1/256 Ω , Maximum 255 (0.99609375 Ω)

5.5.3.2.14 METER_NAME

Defines the name of the meter as reported by HOST_CMD_GET_METE_NAME and is defined as in the provided example code "MSP430I2040SUBMETEREVM "

5.5.3.2.15 METER_SOFTWARE_VERSION

Defines the software version of the meter as reported by HOST_CMD_GET_METE_VER

5.5.3.2.16 METER_HARDWARE_VERSION

Defines the hardware version of the meter as reported by HOST_CMD_GET_METE_VER

5.5.3.2.17 METER_METROLOGY_VERSION

Defines the Metrology library version of the meter as reported by HOST_CMD_GET_METE_VER

5.5.3.2.18 METER_PROTOCOL_VERSION

Defines the Protocol version of the meter as reported by HOST_CMD_GET_METE_VER

Example Application Code

A.1 Introduction

Project Structure

- emeter-communication.c – source code for low level UART communication routines including UART port setup, write and read from UART, interrupt services routine for byte-wise send and receive
- emeter-dlt645.c – source code for the polling mode protocol implementation
- emeter-main.c – source code for system initialization, main loop, callback functions implementation and interrupt vector placement.
- emeter-metrology-i2041.r43 – embedded metering library object code
- emeter-setup.c – source code for low level system initialization
- emeter-template.h – source code for configuration (will be discussed later)
- metrology-calibration-default.c – source code to put the user defined default calibration parameter into a proper data structure
- metrology-calibration-template.h – source code of user defined default calibration parameter
- emeter-autoreport.c – source code for performing auto reporting. (Note: To enable auto report support, uncomment the `#define AUTOREPORT_SUPPORT` definition in emeter-template.h.)

A.2 Preparing the Application Code to Run

1. Select the emter-app-i2041 project tab at the bottom of the Workspace window
2. Check project options by right clicking project name and select [Options...] from the pop-up menu (see [Figure A-1](#))

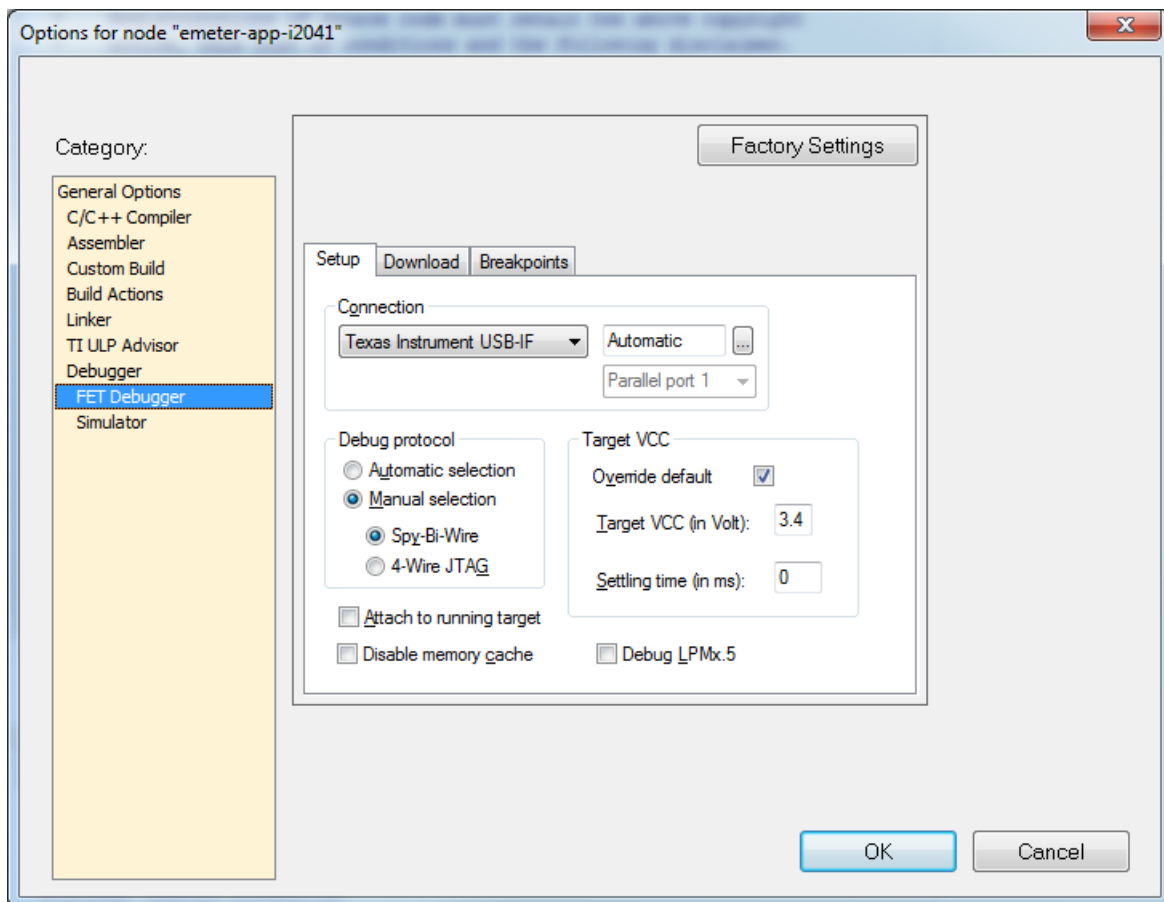


Figure A-1. Project Options

3. When the options appears select [C/C++ compiler] on the left hand column and then select the **Optimizations** tab on the right hand side and check that the optimization setting is as shown in Figure A-2.

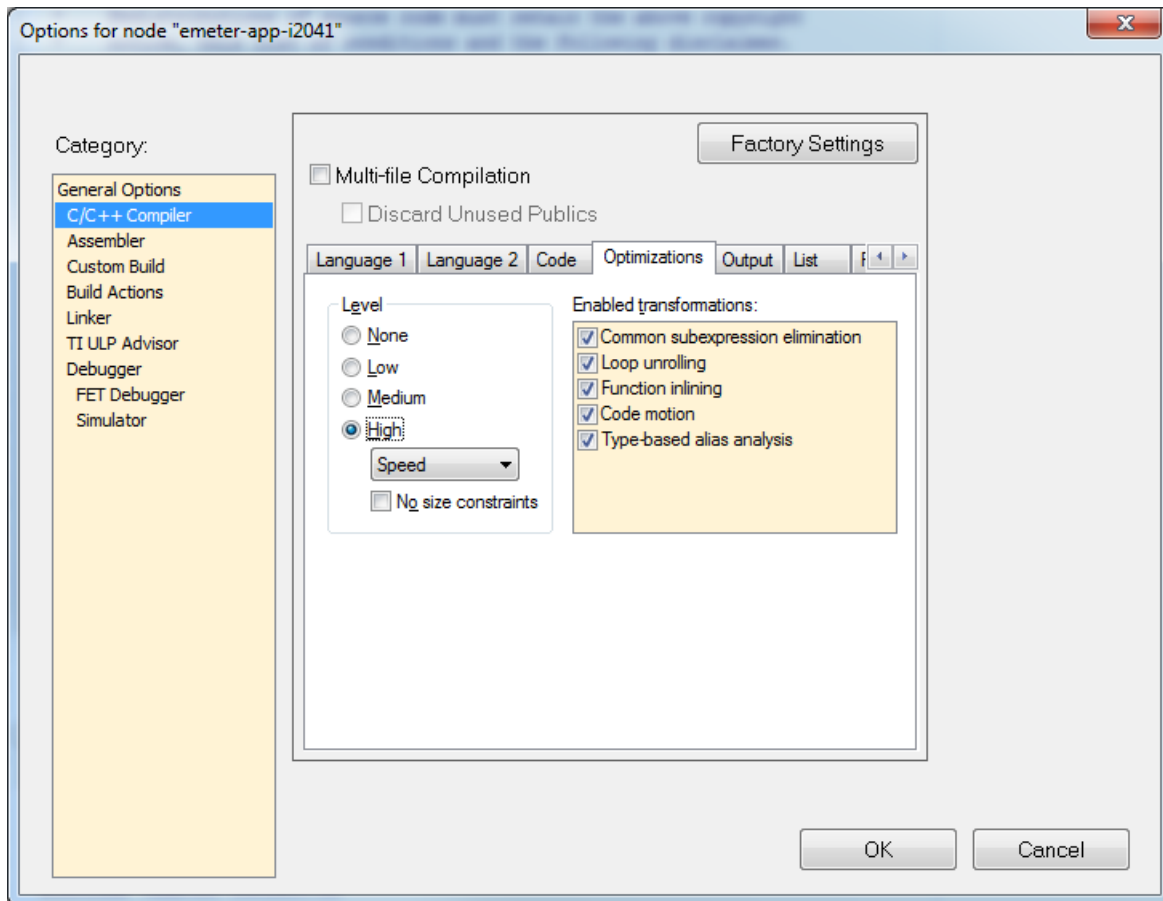


Figure A-2. Optimization Options

4. Select **FET Debugger** on the left hand column the select **Setup** tab. The EVM uses Spy-Bi-Wire for its code downloading and debugging. Check to make sure the options is as shown in [Figure A-3](#).

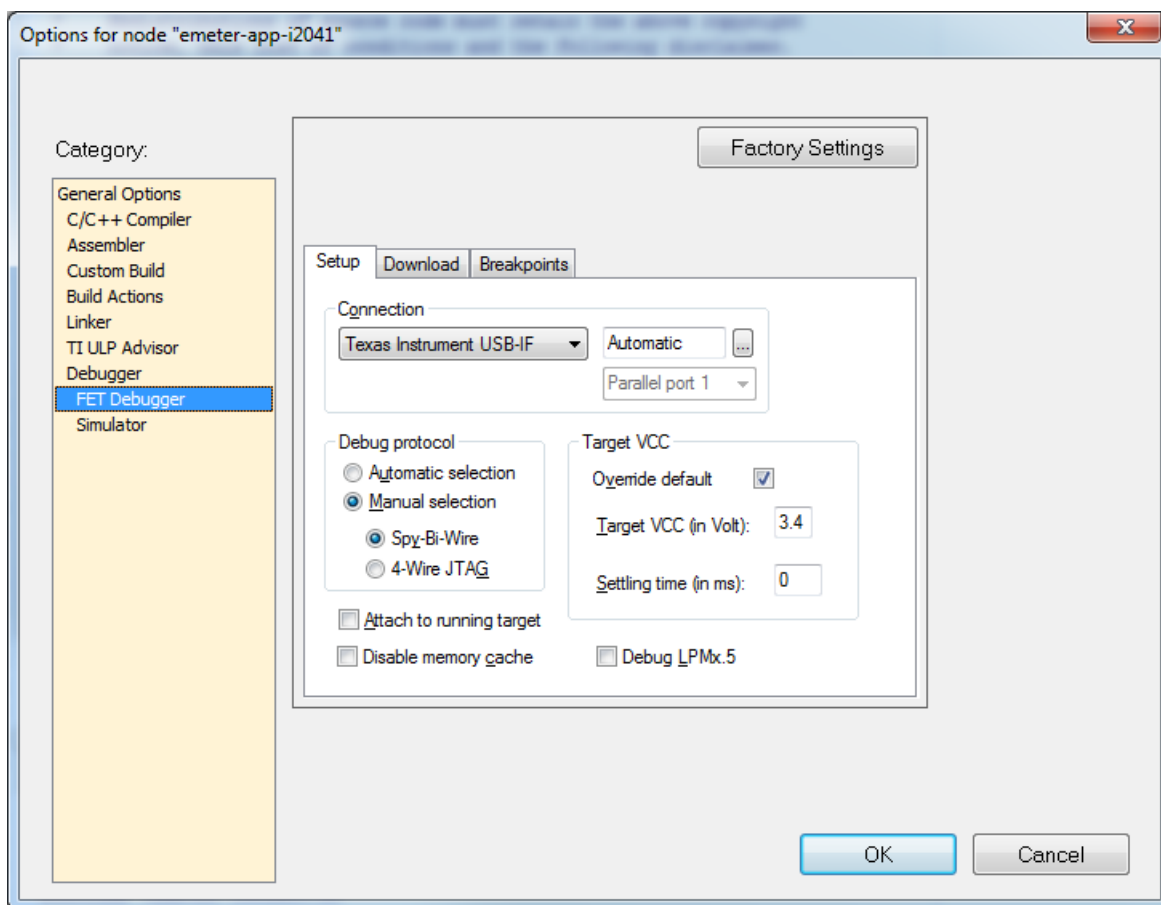


Figure A-3. Debugger Options

5. Select the **Download** tab. Due to the design of the firmware to protect against unintentional deletion of factory parameters like the calibration for the clock speed, the calibration values are stored in the main memory in flash page 0x8000 to 0x83FF. Thus the calibration value is NOT preserved when **Erase main memory** or **Erase main and information memory** is selected (see [Figure A-4](#)). The user should select **Erase main memory** as the option for download to preserve the factory parameters.

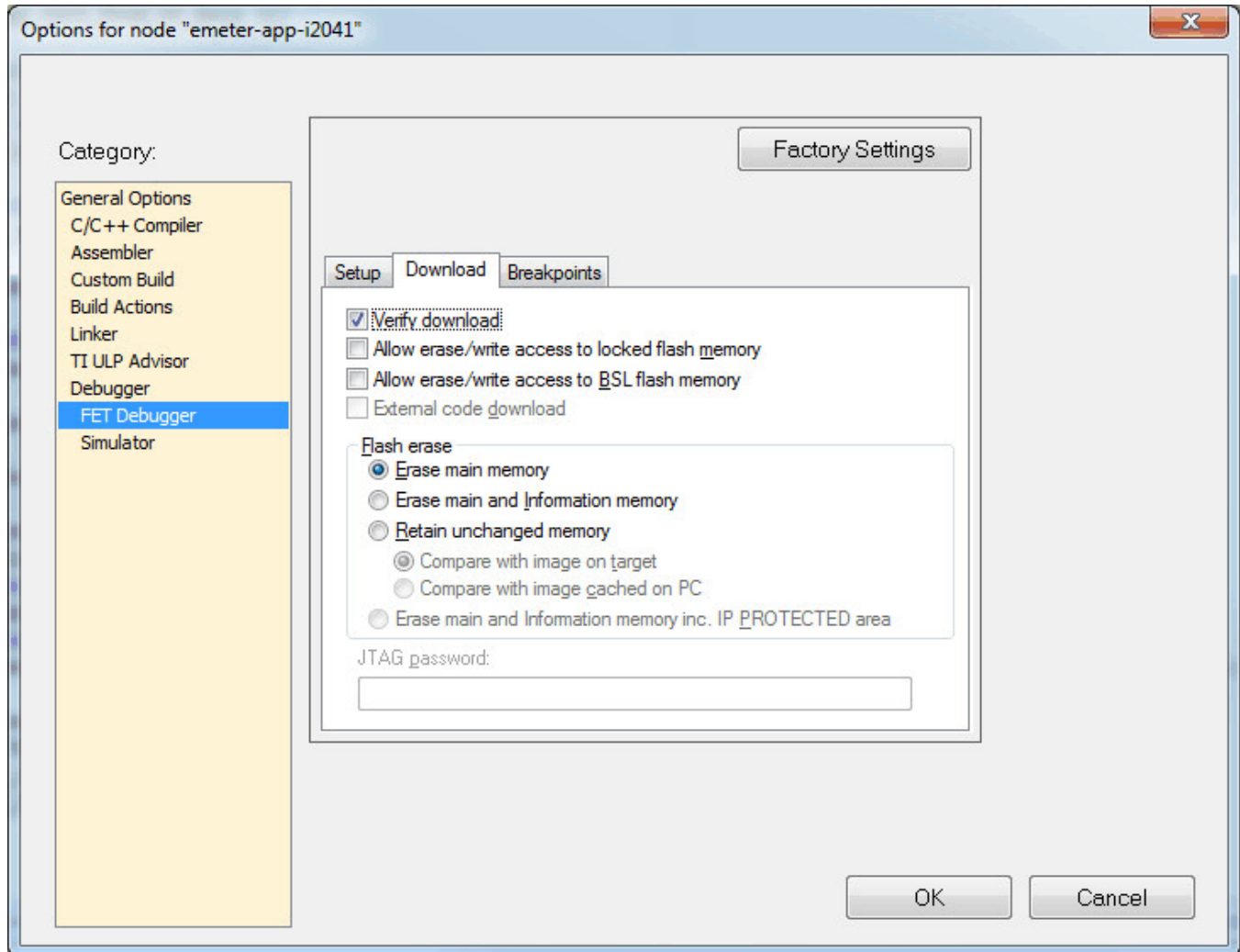


Figure A-4. Download Options

6. Click OK after all of the changes are made.
7. Rebuild the project by right clicking on the project and select [Rebuild All] from the pop-up menu (see [Figure A-5](#)). Three warnings will be reported during rebuilding (see [Figure A-6](#)), it is safe to ignore the three warnings. To open the project workspace, modify the code, compile, and download the code, the user must have IAR Embedded Workbench 5.5 installed with a valid license. If a valid license is not available, the user can still download the object code. See [Section A.3](#) for downloading procedures.

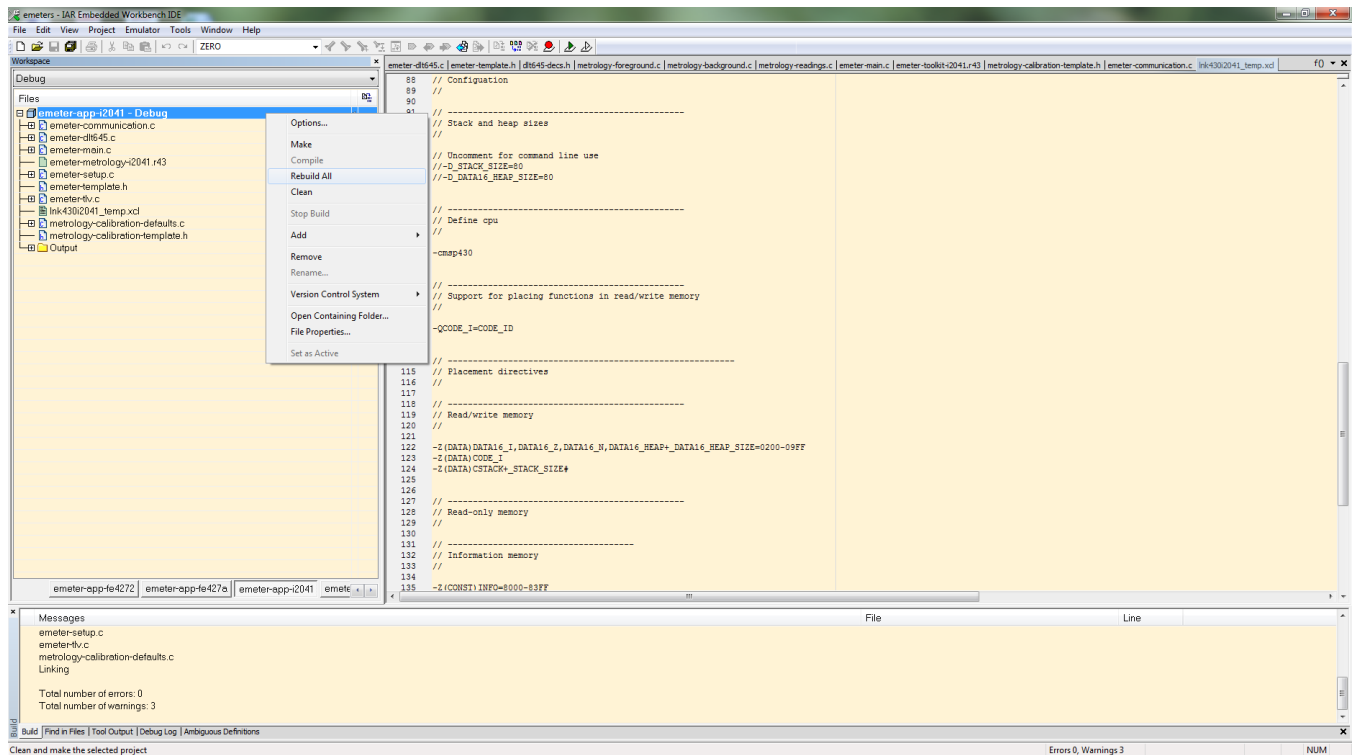


Figure A-5. Compiling the Application

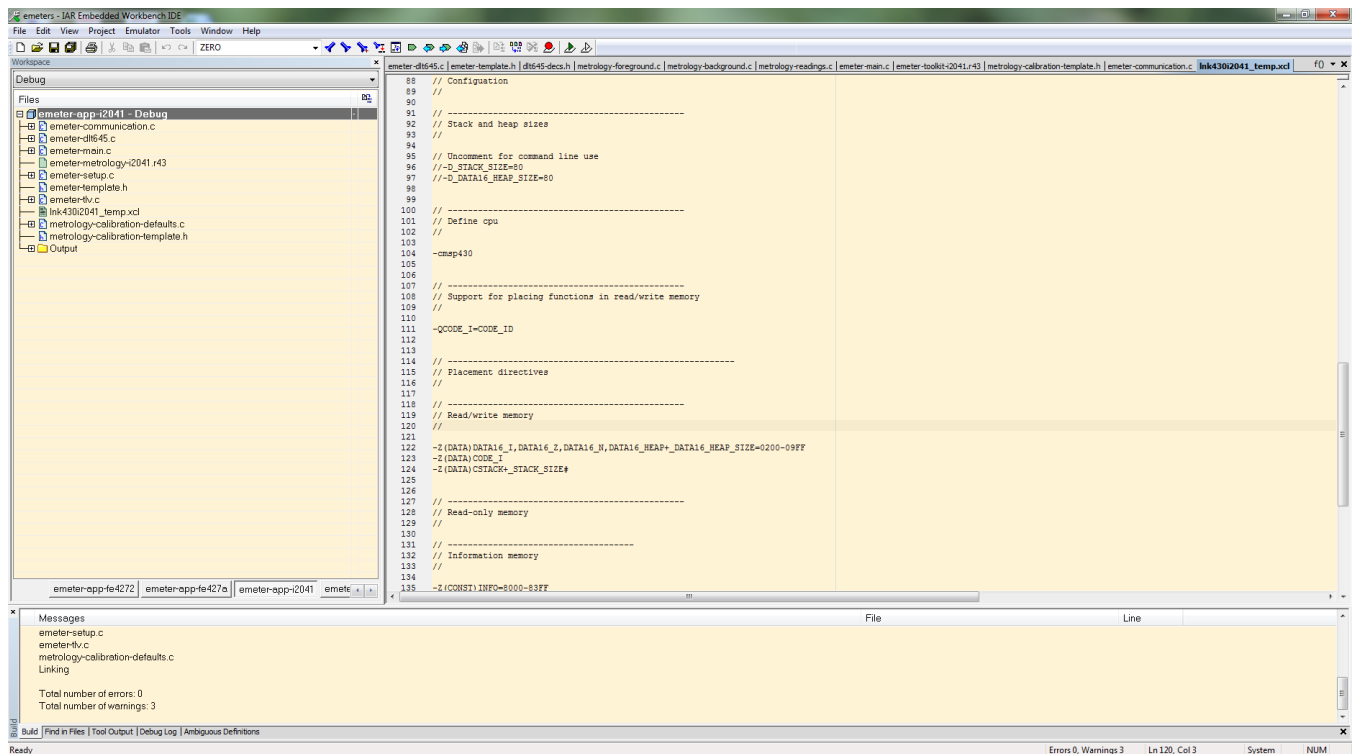


Figure A-6. Warnings

8. Make sure the jumper on J8 is short properly. Connect the 14-pin connector P1 to FET by a flat cable as shown in [Figure A-7](#).

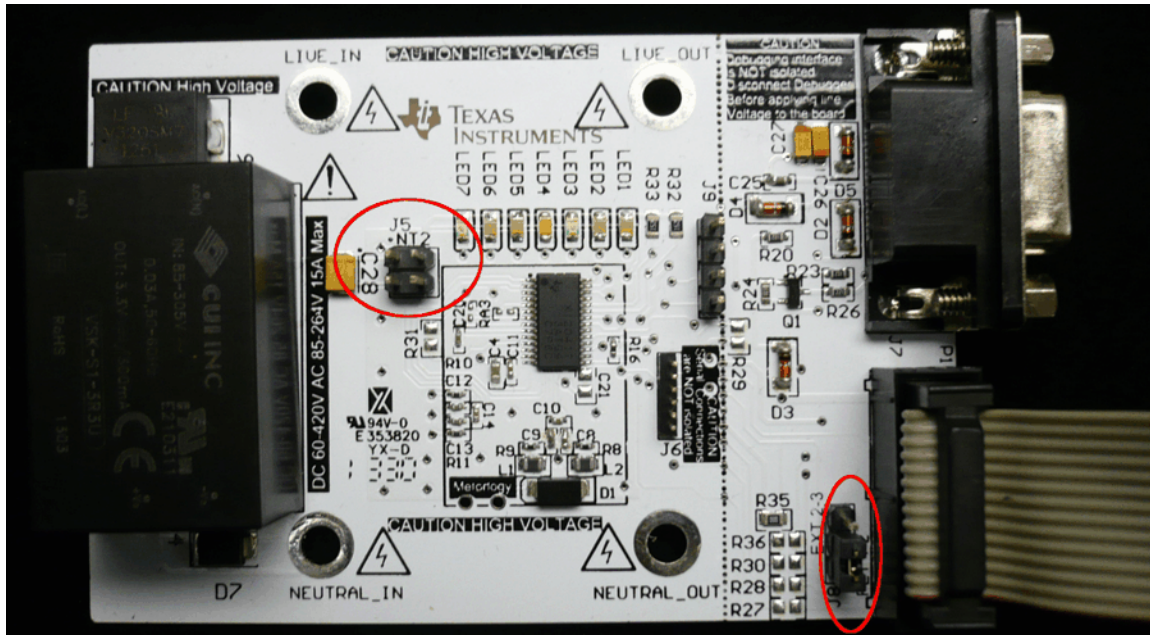


Figure A-7. Connecting EVM and FET

9. Click the **Download and Debug** button to begin download and debugging (see [Figure A-8](#)).

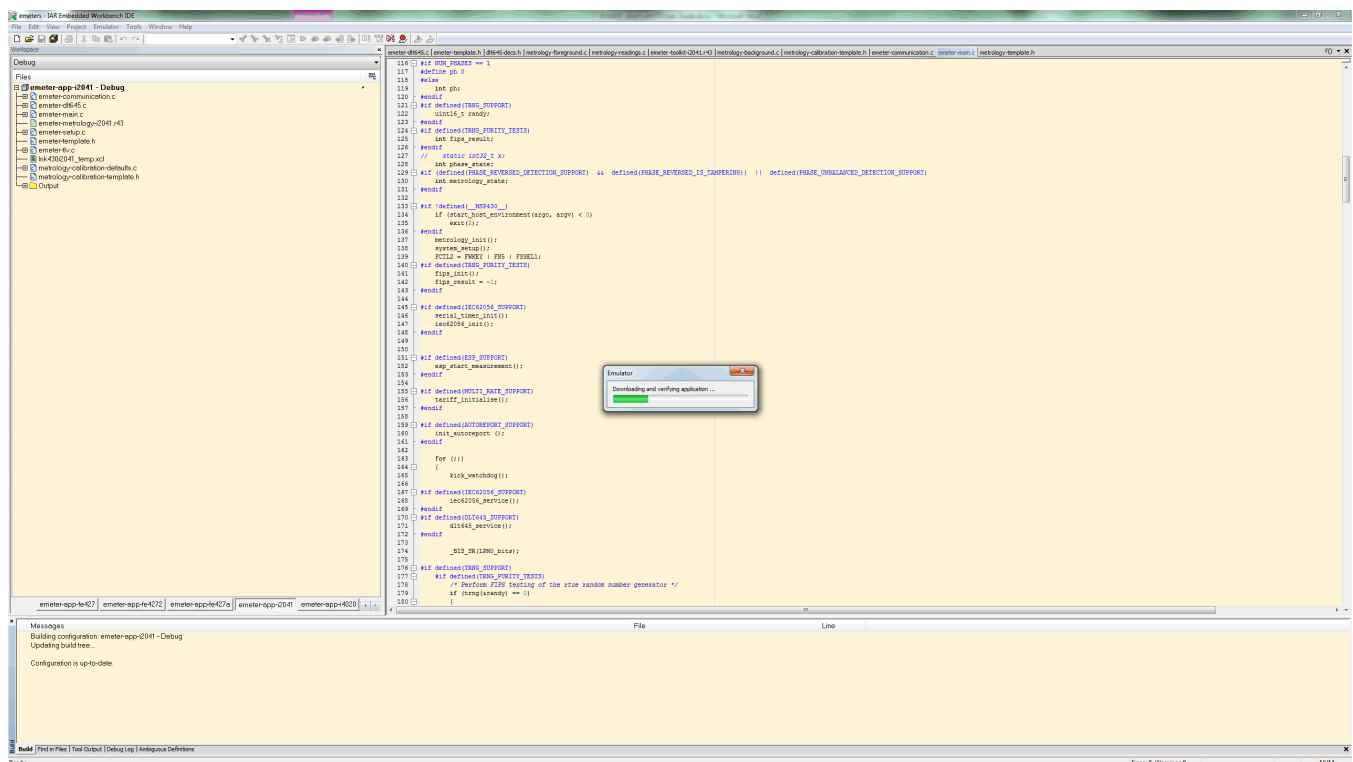


Figure A-8. Code Downloading

10. After downloading is completed and succeed [Figure A-9](#) appears. Click **Go** to start application running (see [Figure A-10](#)).

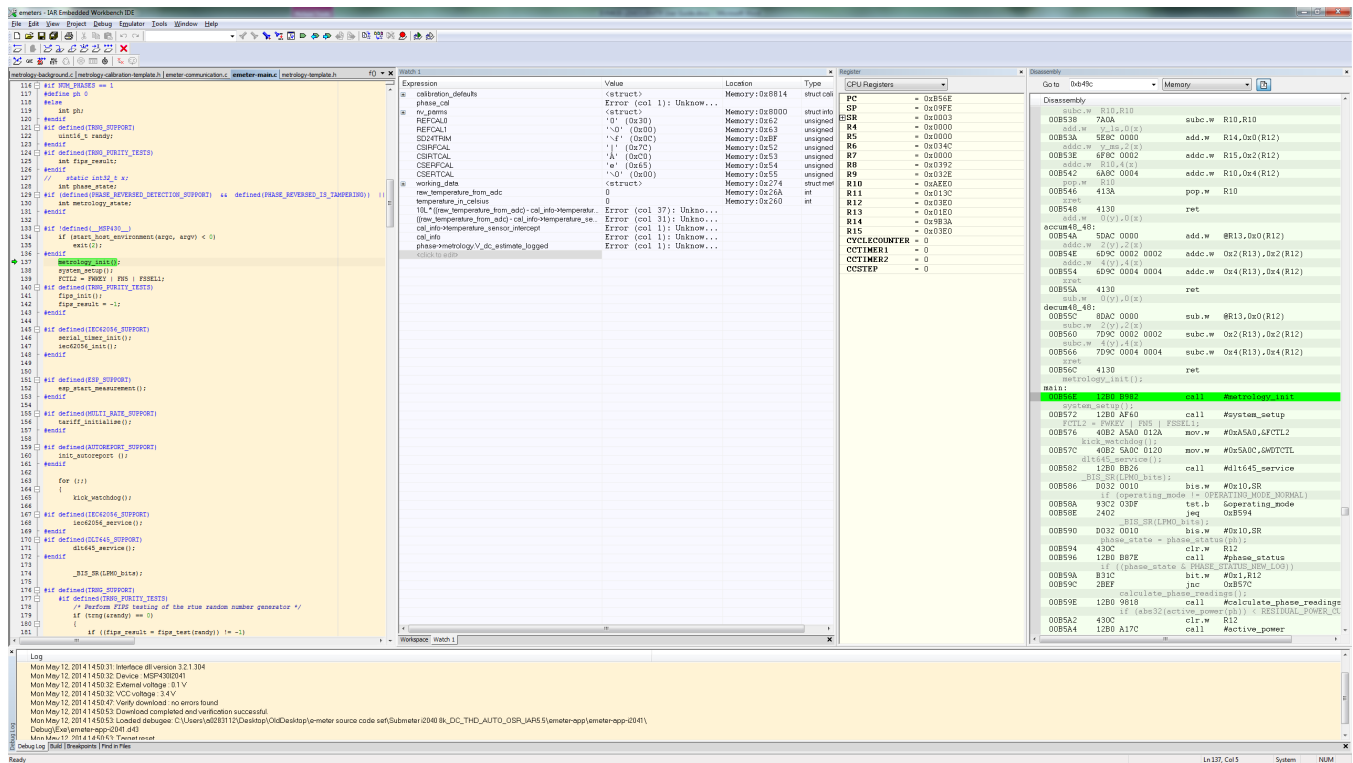


Figure A-9. Debugger Screen

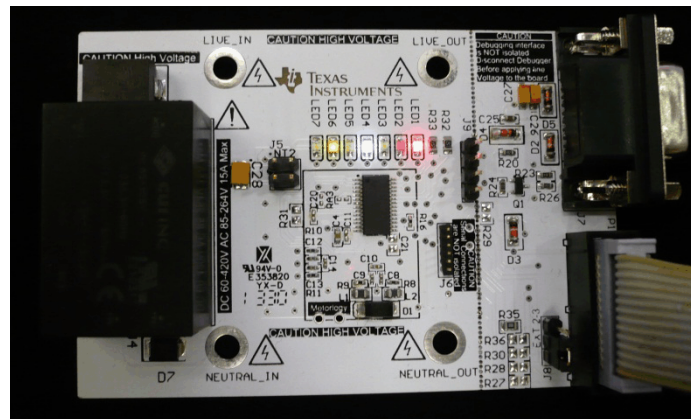


Figure A-10. EVM Running

A.3 Downloading Without an IAR License

If a valid IAR Embedded Workbench 5.5 license is not available, the executable code can still be downloaded to the board with the following steps using the installed IAR Embedded Workbench 5.5.

1. Open the project workspace as described in [Section A.2](#) steps 1 through 5. Then connect the board to the MSP-FET430UIF as described in step 8.
2. Click Project→Download→Download File... (see [Figure A-11](#)).

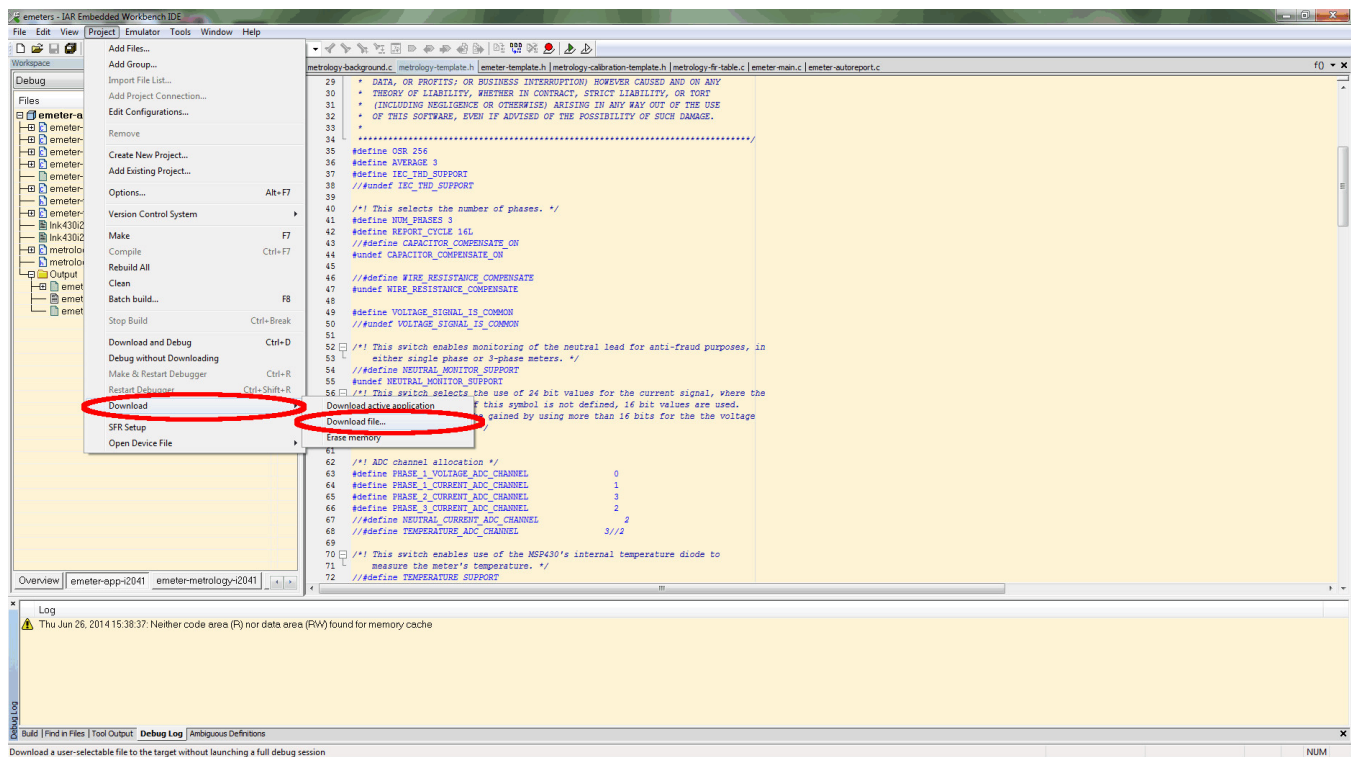


Figure A-11. Download File

- When prompted to select the file, select the file named `emeter-app-i2041.d43` from the folder `[Submeter i2040 4k_3 SOCK_AUTO_OSR_IAR5.5]\emeter-app\emeter-app-i2041\Debug\Exe` (see [Figure A-12](#)).

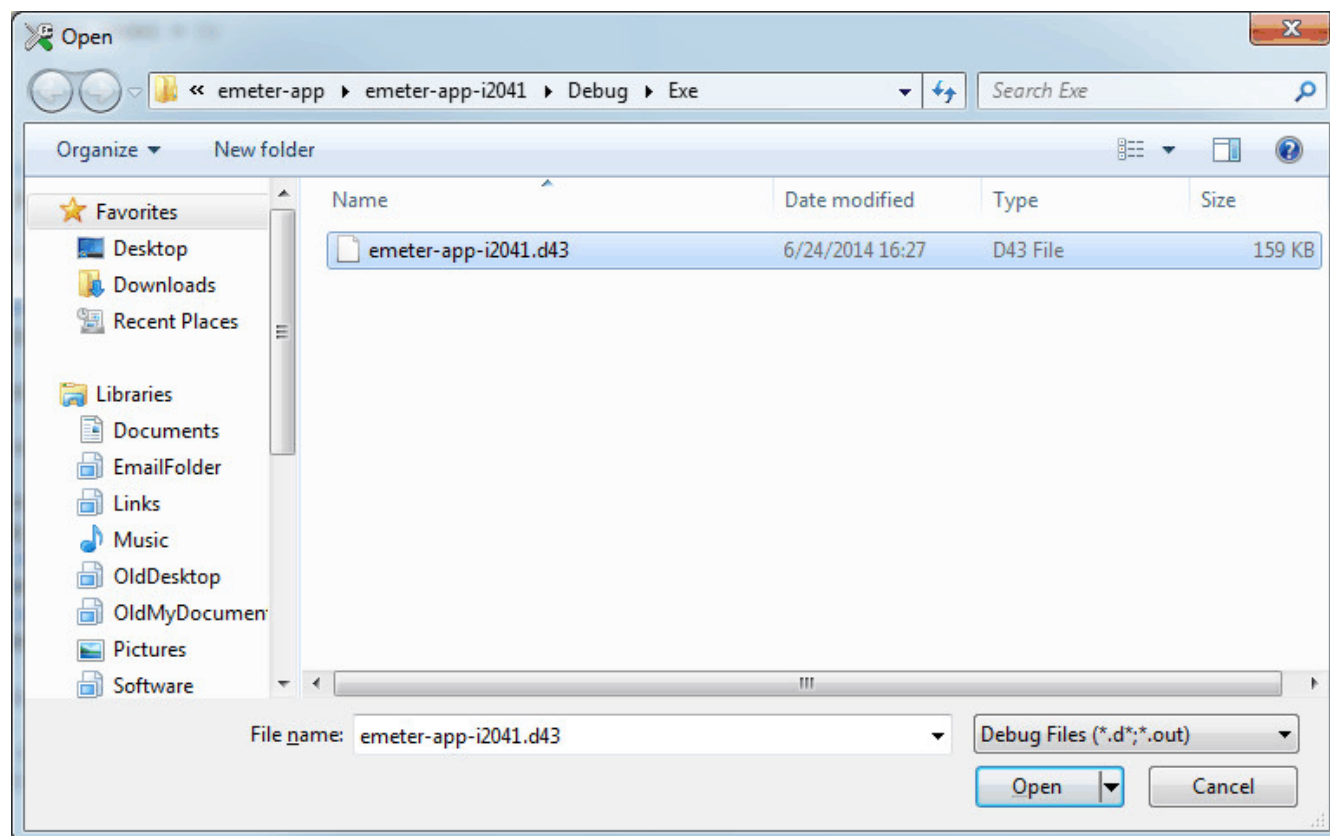


Figure A-12. Select File

4. The executable code is downloaded to the board.

Hardware Design Files

B.1 Package

The hardware related file is included in the downloadable package in a compressed file named hardware.zip which expanded to a folder called Hardware. The following files are included:

- Schematic in PDF format
- Schematic CAD file in Altium Designer format
- PCB layout CAD file in Altium Designer format
- Bill of materials
- PCB layout files in Gerber file format

B.2 Schematic

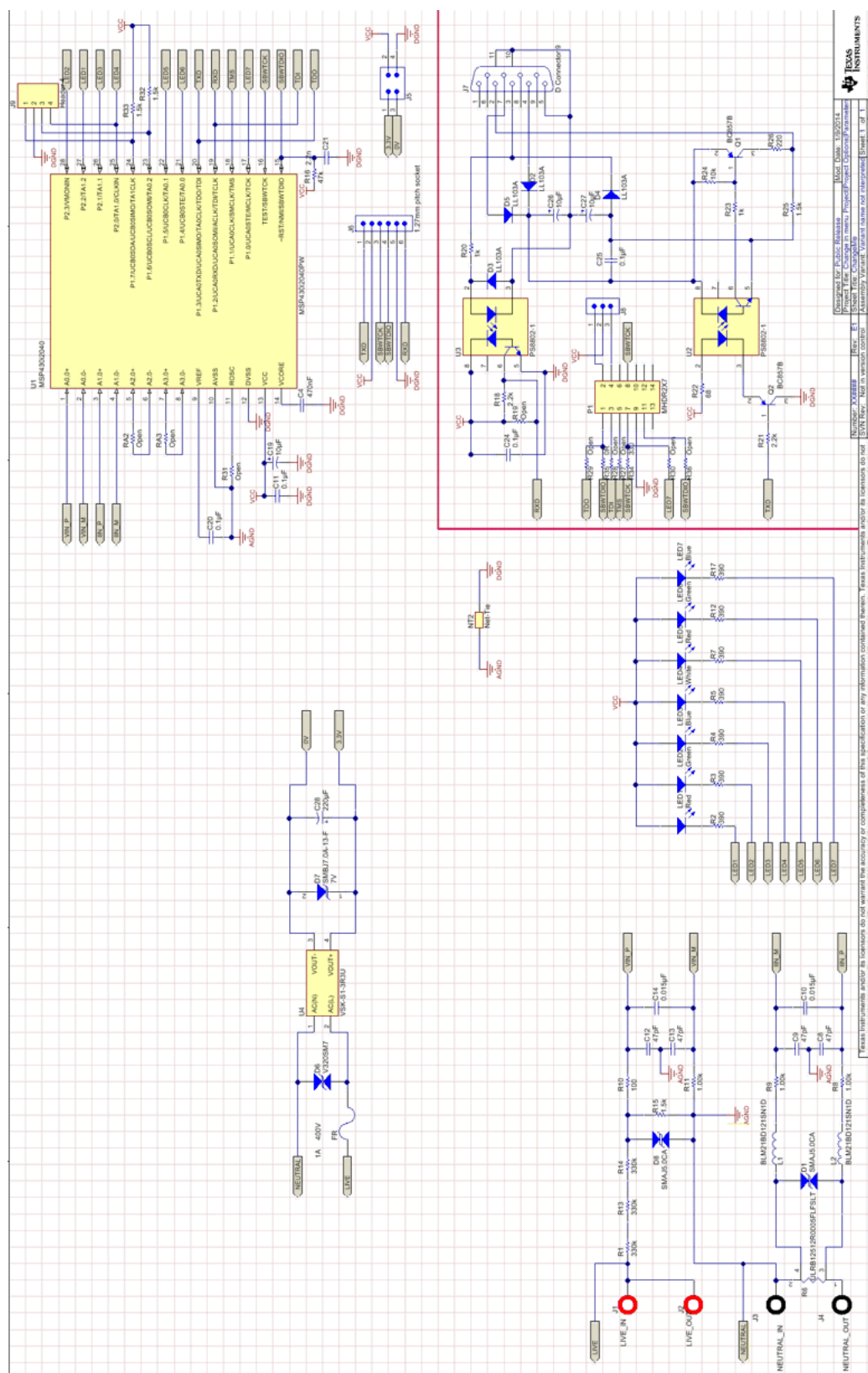


Figure B-1. EVM Schematic

EVM Specification and Performance

C.1 EVM Specification

- Voltage operating range (with supplied power supply) : 85 to 265 VAC, 120 to 380 VDC
- Sample rate : 8000 Hz
- Sampling bit depth : 24 bits
- Polling report supported
- Update rate : four ac cycles (ac mode), 80 ms (dc mode)
- UART communication data rate : 9600 bps
- AC/DC measuring mode switching : four ac cycles (ac to dc), 80 ms (dc to ac)
- Measurements : RMS voltage, RMS current, active power, reactive power, apparent power, power factor, line frequency, temperature
- Measurement voltage range : 0 – 265 Vrms AC, 0 to ± 380 VDC (using reference design circuit and component values)
- Measurement current range : 0 to 15 Arms AC, 0 to ± 22.5 ADC (using reference design circuit and 0.5-m Ω shunt values)
- Voltage resolution : 1 mV
- Current resolution : 1 μ A
- Active Power resolution : 1 mW
- Reactive and apparent power resolution : 1 mW
- Power factor resolution : 0.001
- Frequency resolution : 0.01 Hz

C.2 EVM Performance

C.2.1 Power Accuracy at Room Temperature

Table C-1. Output C, PF = 1, Calibrated at 7.50212 A, 1650.7 W

Current Ref	Current Read	Current Error	Current %	Power Ref	Power Read	Power Error	Power %
0.014555	0.014760	0.000205	1.408%	3.202	3.213	0.011	0.355%
0.029637	0.029737	0.000100	0.339%	6.517	6.515	-0.002	-0.029%
0.074836	0.074950	0.000114	0.152%	16.579	16.615	0.036	0.217%
0.145378	0.145391	0.000013	0.009%	31.984	31.992	0.009	0.027%
0.295964	0.295992	0.000028	0.009%	65.114	65.168	0.054	0.083%
0.747332	0.747569	0.000237	0.032%	165.104	165.256	0.152	0.092%
1.500050	1.500000	-0.000050	-0.003%	330.060	330.314	0.254	0.077%
2.988460	2.989000	0.000540	0.018%	662.136	662.586	0.450	0.068%
7.502390	7.500000	-0.002390	-0.032%	1650.380	1650.779	0.399	0.024%
14.350000	14.320000	-0.030000	-0.209%	3090.240	3084.544	-5.696	-0.184%
19.321400	19.242000	-0.079400	-0.411%	4162.290	4143.000	-19.290	-0.463%

Table C-2. Output C, PF = 0.5L, Calibrated at 7.50212 A, 1650.7 W

Current Ref	Current Read	Current Error	Current %	Power Ref	Power Read	Power Error	Power %
0.014558	0.014654	0.000096	0.659%	1.555	1.559	0.004	0.257%
0.029628	0.029774	0.000146	0.492%	3.173	3.168	-0.005	-0.158%
0.074823	0.074824	0.000001	0.001%	8.037	8.055	0.018	0.218%
0.145166	0.145141	-0.000025	-0.017%	13.287	13.298	0.011	0.083%
0.295495	0.295481	-0.000014	-0.005%	27.587	27.589	0.002	0.007%
0.747391	0.747393	0.000002	0.000%	71.399	71.492	0.093	0.130%
1.500100	1.500000	-0.000100	-0.007%	165.134	165.379	0.245	0.148%
2.988390	2.989000	0.000610	0.020%	319.346	319.745	0.399	0.125%
7.402320	7.400000	-0.002320	-0.031%	596.562	596.954	0.392	0.066%
14.371200	14.354000	-0.017200	-0.120%	890.268	889.965	-0.303	-0.034%
20.021600	19.914000	-0.107600	-0.537%	2197.230	2187.520	-9.710	-0.442%

Table C-3. Output C, PF = 0.5C, Calibrated at 7.50212 A, 1650.7 W

Current Ref	Current Read	Current Error	Current %	Power Ref	Power Read	Power Error	Power %
0.014543	0.014671	0.000128	0.879%	1.659	1.652	-0.007	-0.422%
0.029624	0.029638	0.000014	0.046%	3.388	3.390	0.002	0.056%
0.074827	0.074848	0.000021	0.029%	8.525	8.537	0.012	0.141%
0.145314	0.145267	-0.000047	-0.032%	18.491	18.503	0.012	0.068%
0.295834	0.295752	-0.000082	-0.028%	37.381	37.391	0.010	0.027%
0.747077	0.747067	-0.000010	-0.001%	94.044	94.030	-0.014	-0.015%
1.481450	1.481000	-0.000450	-0.030%	171.561	171.526	-0.035	-0.020%
2.988510	2.989000	0.000490	0.016%	345.256	345.311	0.055	0.016%
7.401750	7.399000	-0.002750	-0.037%	1025.610	1025.054	-0.556	-0.054%
14.334700	14.318000	-0.016700	-0.117%	2196.370	2190.308	-6.062	-0.276%
19.304000	19.241000	-0.063000	-0.326%	2965.780	2951.400	-14.380	-0.485%

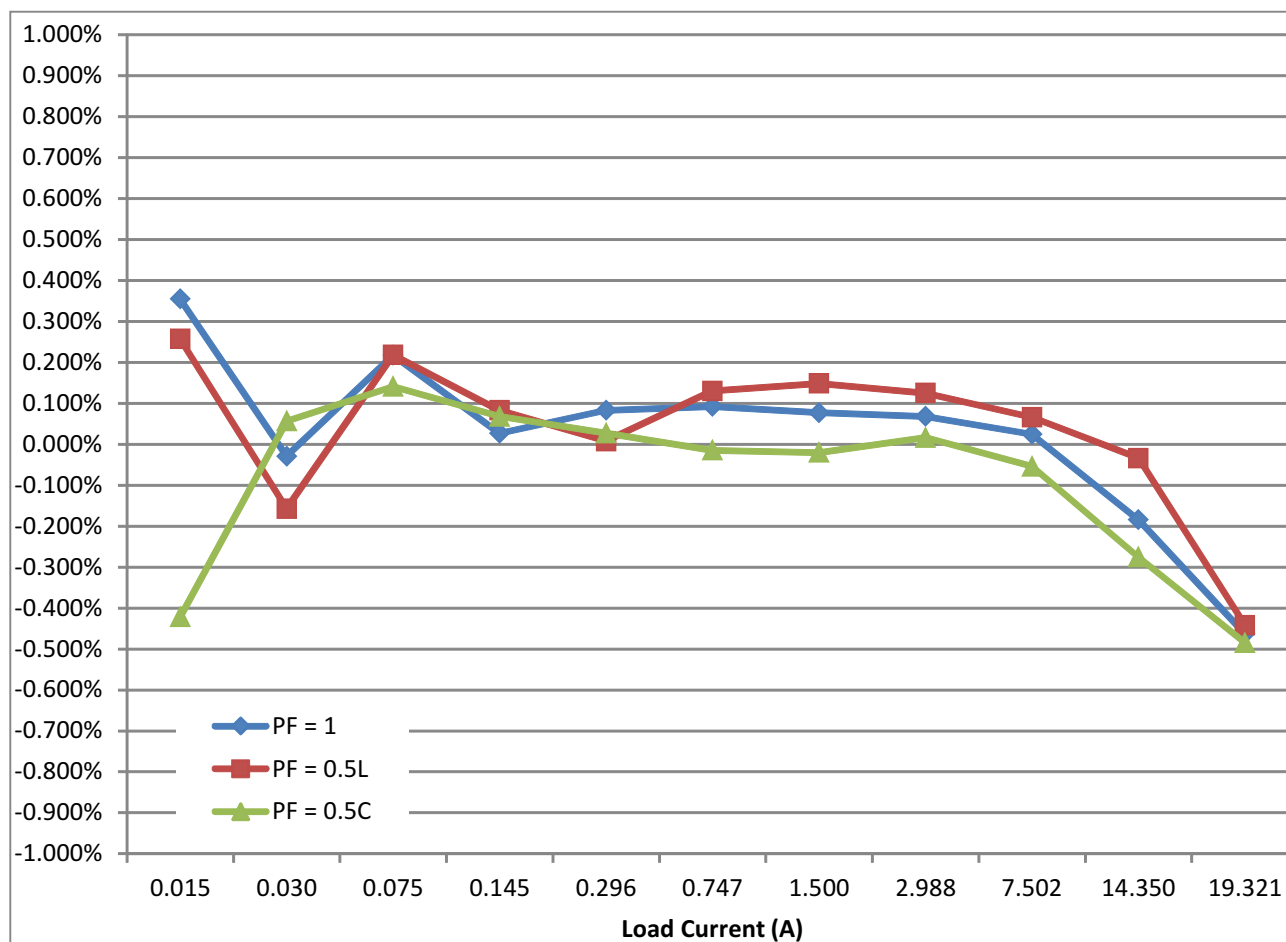


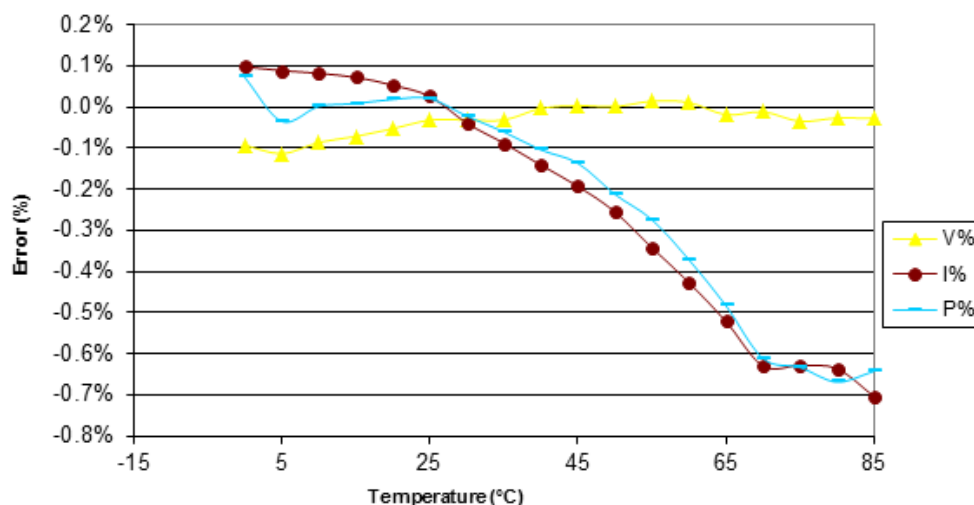
Figure C-1. Power Percentage Error (%) vs Load Current (A)

NOTE: It can be seen in [Figure C-1](#) that the error begins to go negative at higher current. This phenomenon occurs because the shunt is heated by the current flowing through the traces on the board. At low current, the higher error is mostly caused by noise that causes instability of the readings.

C.2.2 Accuracy vs Temperature at 220 V, 5 A

Table C-4. Accuracy vs Temperature

Temp (°C)	Voltage (V)			Current (A)			Power (W)		
	V	Vref	V%	I	Iref	I%	P	Pref	P%
0	219.786	219.991	-0.093%	5.005	5.000	0.097%	1100.730	1099.940	0.072%
5	219.728	219.978	-0.114%	5.005	5.000	0.088%	1099.470	1099.860	-0.035%
10	219.801	219.992	-0.087%	5.005	5.000	0.082%	1100.050	1100.040	0.001%
15	219.839	219.997	-0.072%	5.004	5.000	0.072%	1100.110	1100.020	0.008%
20	219.896	220.013	-0.053%	5.003	5.000	0.053%	1100.120	1099.920	0.018%
25	219.883	219.954	-0.032%	5.002	5.000	0.025%	1100.050	1099.830	0.020%
30	219.951	220.021	-0.032%	4.998	5.000	-0.040%	1099.680	1099.930	-0.023%
35	219.990	220.060	-0.032%	4.996	5.000	-0.088%	1099.290	1099.960	-0.061%
40	220.012	220.021	-0.004%	4.993	5.000	-0.142%	1098.910	1100.060	-0.105%
45	220.060	220.054	0.003%	4.991	5.000	-0.193%	1098.330	1099.840	-0.137%
50	220.054	220.053	0.000%	4.988	5.000	-0.256%	1097.580	1099.920	-0.213%
55	220.026	219.996	0.014%	4.983	5.000	-0.343%	1096.890	1099.910	-0.275%
60	220.050	220.027	0.010%	4.979	5.000	-0.427%	1095.800	1099.890	-0.372%
65	220.009	220.051	-0.019%	4.974	5.000	-0.522%	1094.700	1100.020	-0.484%
70	219.968	219.995	-0.012%	4.969	5.000	-0.630%	1093.210	1099.940	-0.612%
75	219.896	219.973	-0.035%	4.969	5.000	-0.629%	1093.060	1100.030	-0.634%
80	219.892	219.952	-0.027%	4.969	5.000	-0.637%	1092.600	1099.960	-0.669%
85	219.895	219.957	-0.028%	4.968	5.004	-0.704%	1092.840	1099.910	-0.643%


Figure C-2. Typical Accuracy vs Temperature

NOTE: It can be seen in [Figure C-2](#) that the error begins to go negative at higher temperature. This phenomenon occurs because the board is heated and the combined temperature coefficient of the whole board is affected.

Running on MSP430i2040 and MSP430i2041

The example IAR project has been configured for MSP430i2041. To run on MSP430i2040, follow these steps once:

1. Open the workspace. After launching IAR 5.5, select *File*→*Open*→*Workspace*.

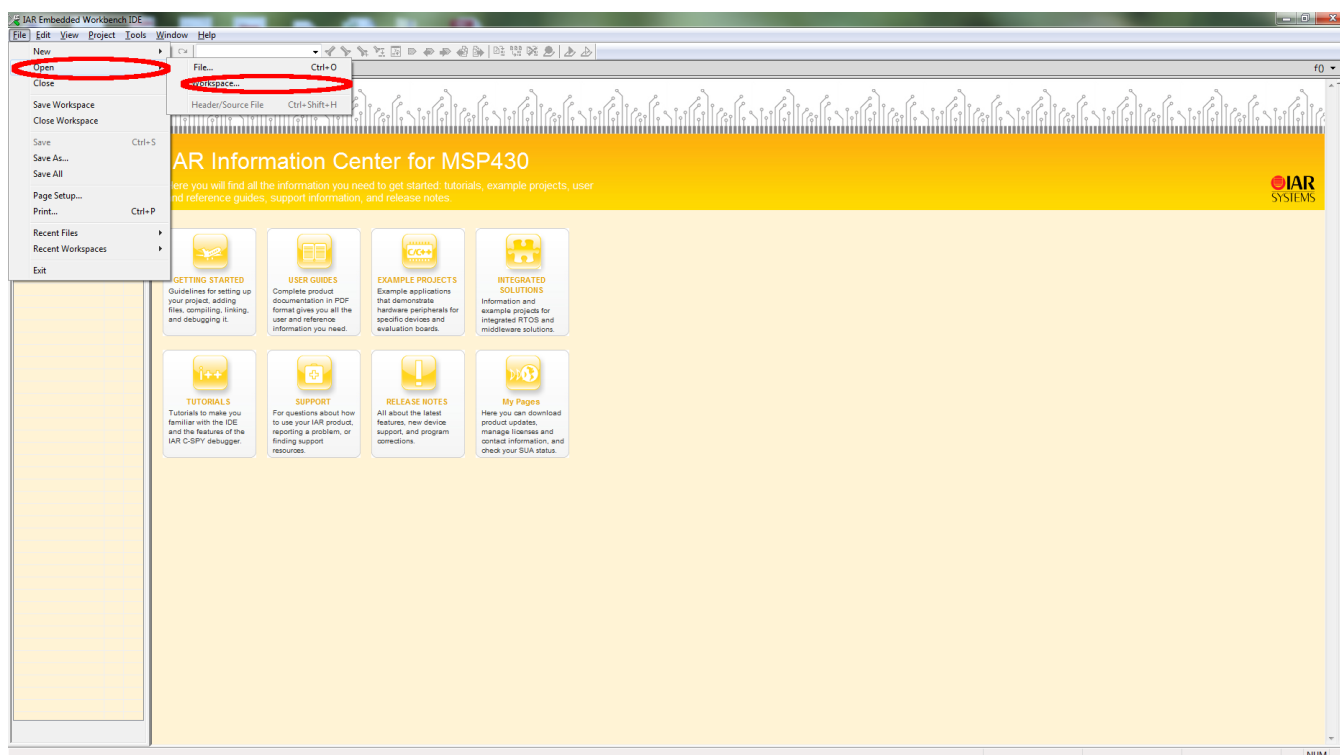


Figure D-1. IAR 5.5 Launch Window

When prompted to open the project, select *emeters.eww* from the project directory of the example code.

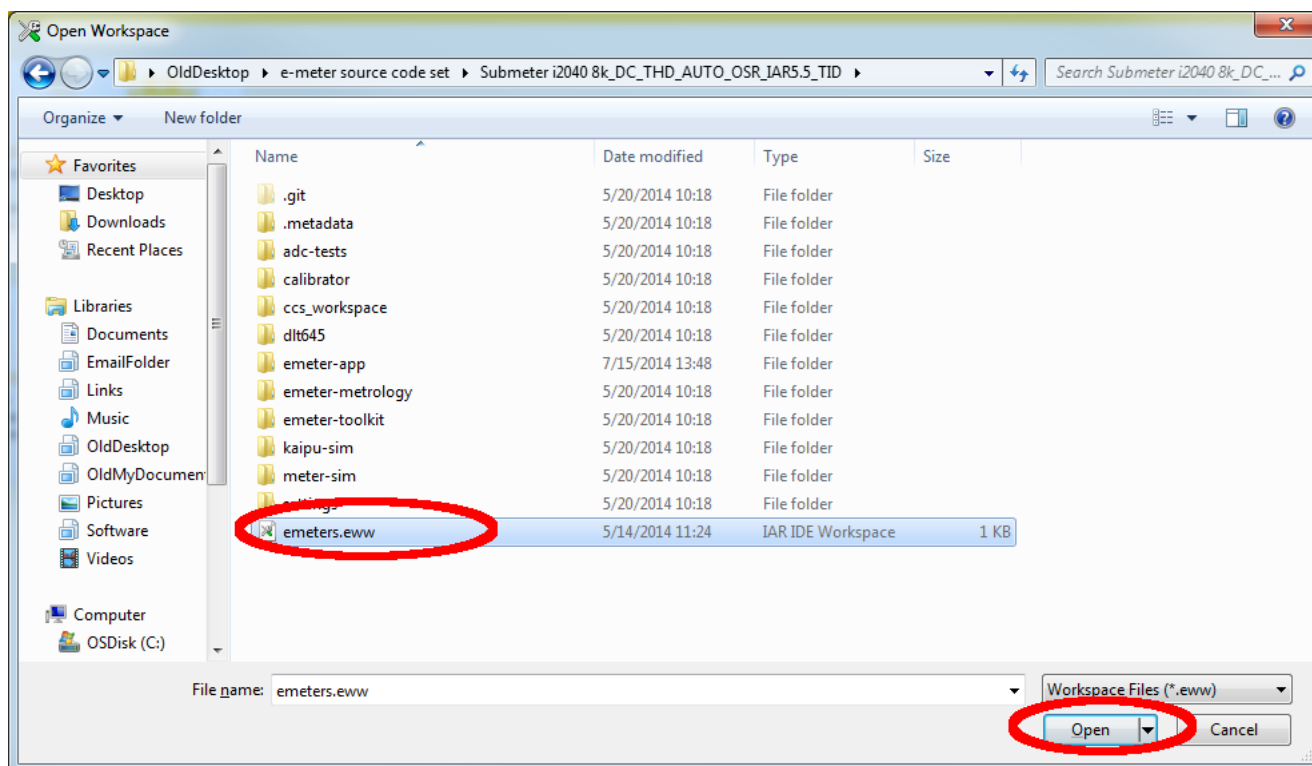


Figure D-2. Open Workspace Window

2. Set the option. Right click on the project *emeter-app-i2041* and select *Options...*

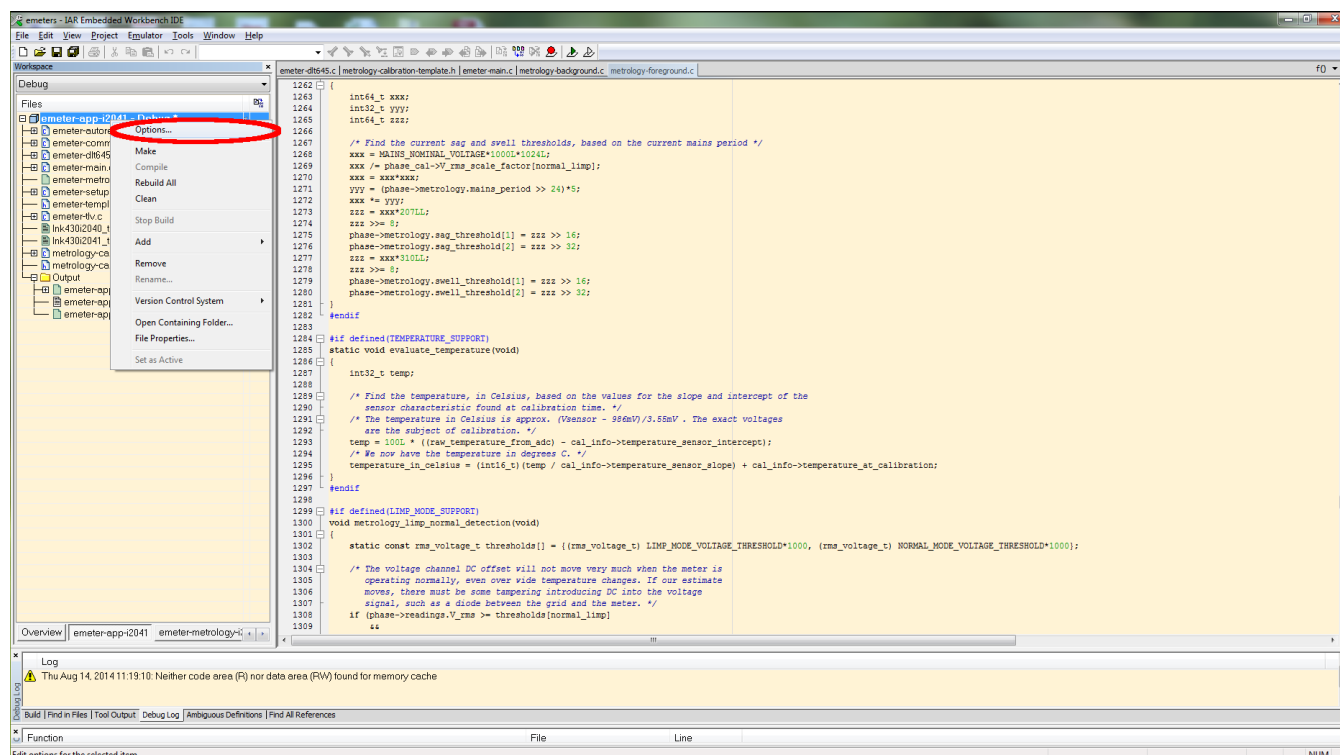


Figure D-3. Workspace Options

In the *Options* window, click on *General Options* on the left-hand column and then select the *Target* tab. Then, in the *Device* section, click on the right-hand side button of the entry box and select *MSP430Ixxxx Family*→*MSP430I2040*.

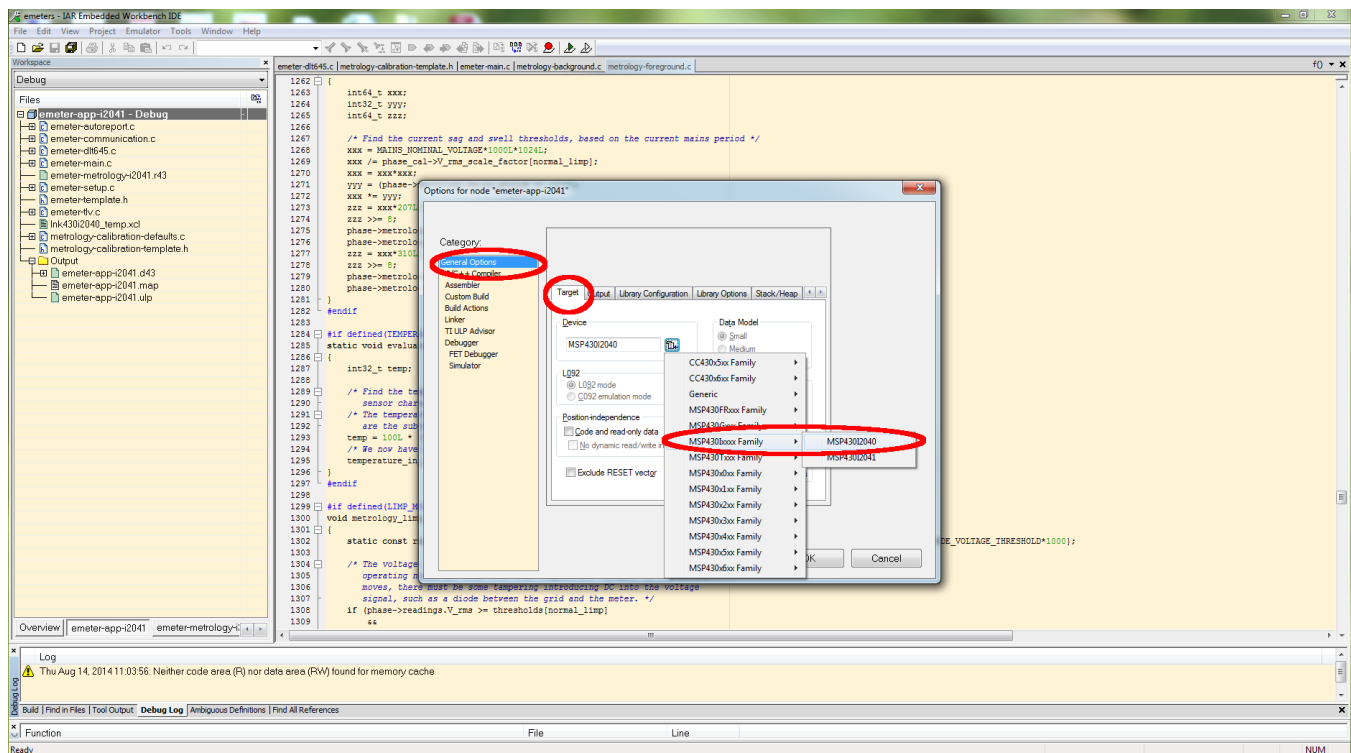


Figure D-4. Options Window

Click on *Linker* in the left-hand column and select the *Config* tab. Then click on the right-hand button in the *Linker configuration file* box.

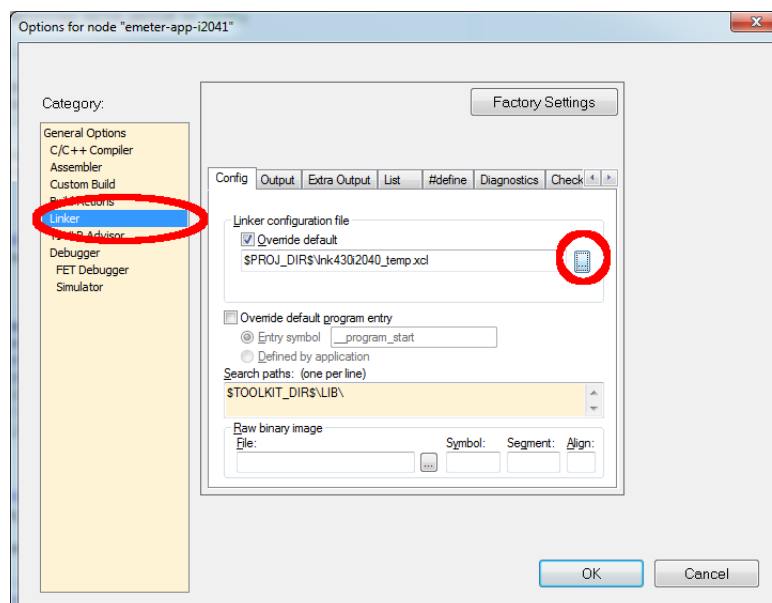


Figure D-5. Config Tab

Select *Ink430i2040_temp.xcl* when prompted and click *Open*.

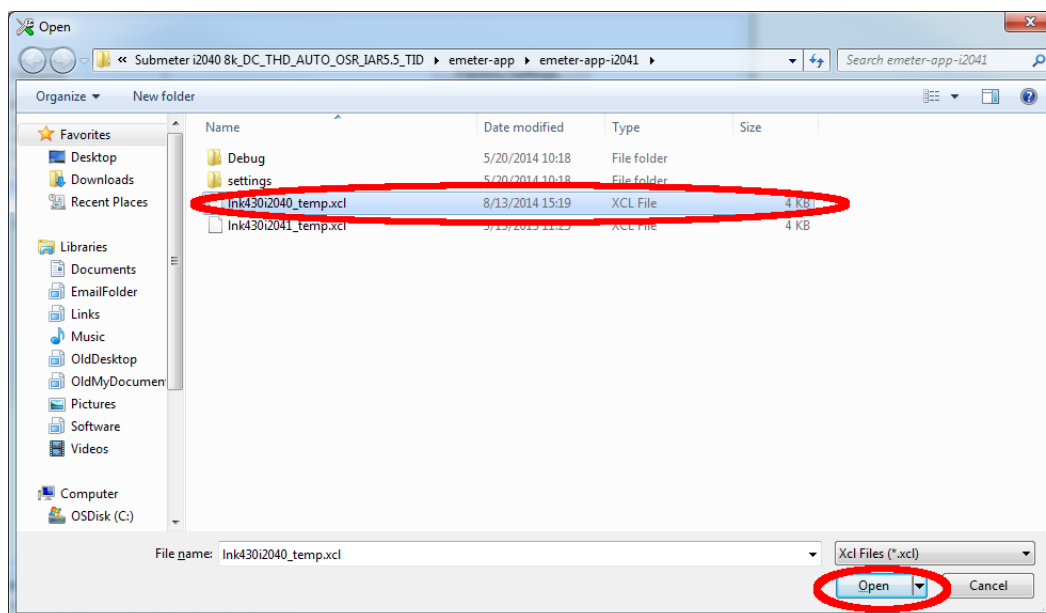


Figure D-6. Selecting Default XCL File

When brought back to the *Options* window, click *OK*.

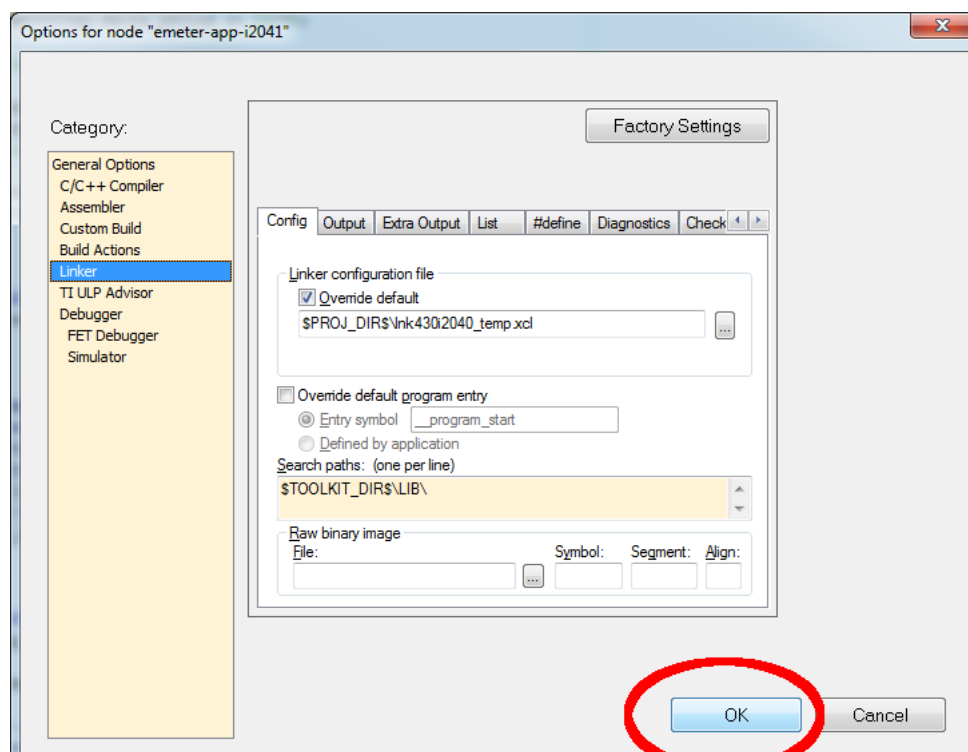


Figure D-7. Finishing the Setting Changes

When brought back to the main screen, select **File**→**Save** to save the setting.

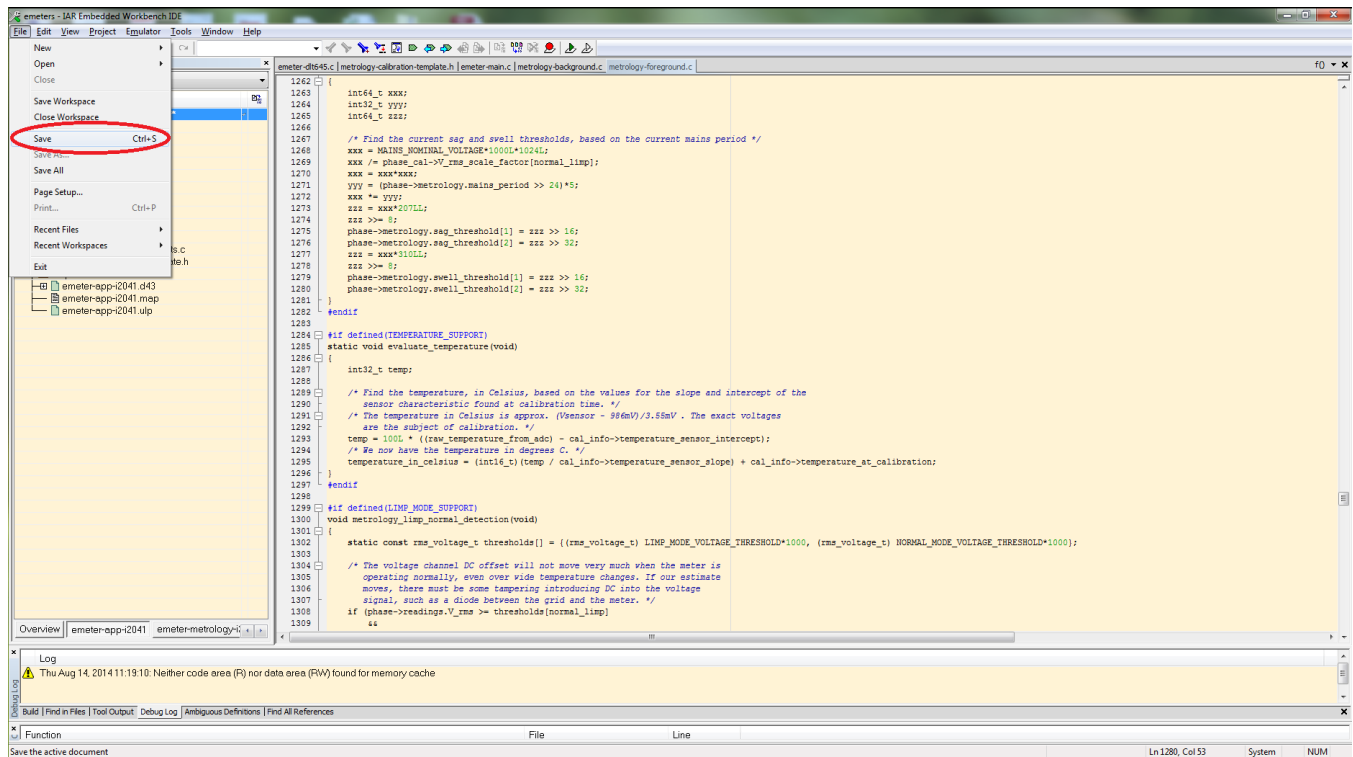


Figure D-8. Saving the Setting Changes

The project is now set for run on MSP430i2040. To have the project to run back on MSP430i2041, follow the same steps except to select MSP430lxxx Family→MSP430i2041 in the Options window and select *Ink430i2041_temp.xcl* when prompted in the *Linker configuration file* box.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
OMAP Applications Processors	www.ti.com/omap
Wireless Connectivity	www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation	www.ti.com/automotive
Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Video and Imaging	www.ti.com/video

TI E2E Community

e2e.ti.com