

# TI Designs

## Single-phase Energy Measurement Reference Design Guide




### TI Designs

TI Designs provide the foundation that you need including methodology, testing and design files to quickly evaluate and customize and system. TI Designs help you accelerate your time to market.

### Design Resources

[MSP430AFE253SUBMETEREVM](#) Tool Folder with Design Files



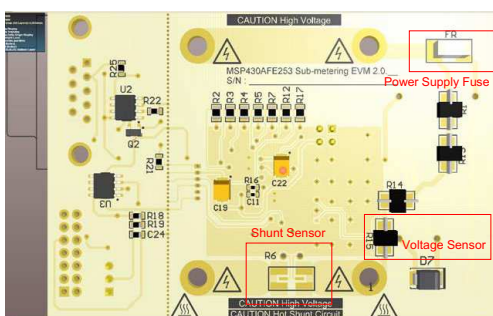
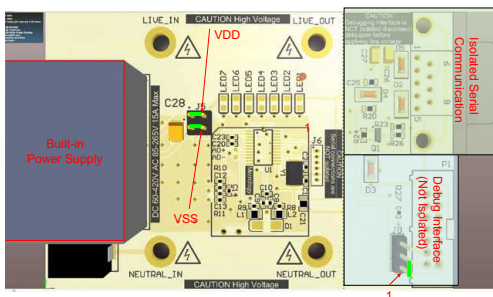
[ASK Our E2E Experts](#)  
[WebBench Calculator Tools](#)

### Featured Applications

- Home Appliances
- Server and PC Power Supplies
- UPS
- Smart Plugs or Power Strip
- Solar Energy Inverter
- Electrical Vehicle Charger
- Home Monitoring, Security and Automation

### Design 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 supplied by 85 – 265 VAC (47Hz – 63Hz) or 120 – 380 VDC simplifies evaluation setup
- Built-in RS232 external communication interface for reading measurements and performing calibration
- 7 Built-in LEDs for customer debugging and visual monitoring
- Measurement of root mean square voltage, root mean square current, active power, reactive power, apparent power, power factor, AC frequency
- Readings update every 4 AC cycles or 80ms in case of DC input
- Capable of automatic reporting measurements to UART port on every update at 9600bps
- Capable of AC and DC measurement
- Capable of switching between AC and DC measurement mode automatically
- Capable of doing EMI filter capacitor and wire resistance compensation
- No separate DC calibration required



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

All trademarks are the property of their respective owners.

## 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.

## 1 Getting Started

### 1.1 Preface

This development tool provides you the resources to evaluate the performance of MSP430AFE253 used as an embedded metering application. This documentation will guide you to get familiar with this development tool and will provide the necessary information for further development and evaluation. Before proceeding, please read this guide to gain a basic understanding and knowledge of this development tool and the requirements for safe operation.

More resources is available related to this development tool including more documentation, application notes, example application code, software etc is available for download from Texas Instruments web site, [www.ti.com](http://www.ti.com).

### 1.2 Safety and Precautions

The EVM is designed for professionals who have received the appropriate technical training, and is designed to operate from an AC power supply or a high-voltage DC supply. Please read this user guide and the safety-related documents that come with the EVM package before operating this EVM.



#### **CAUTION**

Do not leave EVM powered when unattended.



#### **WARNING**

**Hot Surface! Contact may cause burns. Do not touch!**



#### **WARNING**

**High Voltage! Electric shock is possible when connecting board to live wire. Board should be handled with care by a professional.**

**For safety, use of isolated test equipment with overvoltage and overcurrent protection is highly recommended.**

### 1.3 EVM Kit Contents

This kit contains the following:

- This User Guide
- Safety Instructions
- EVM430-AFE253SUBMTR Quick Start Guide
- Design Files and Software Disc

### 1.4 Introduction

The Texas Instruments MSP430AFE253 is an ultra-low-power mixed-signal microcontroller that integrates with 3 independent, 24-bit, sigma-delta A/D converters; a 16-bit hardware multiplier; a UART communication interface; and 11 I/O pins. The peripheral set is particularly well-purposed for electricity power-measurement. This EVM is designed as an evaluation tool for using the MSP430AFE253 device in an embedding metering (submetering) application. In the submetering application space, the electricity-measuring device is embedded in the end-application, and provides the user with the information about the voltage, current, power consumption, and other information about the device. Embedded metering is applicable in many areas, which includes the following:

- Home Appliances
- Server and PC Power Supplies
- UPS
- Smart Plugs or Power Strip
- Solar Energy Inverter
- Electrical Vehicle Charger
- Home Monitoring, Security and Automation

In this document, the descriptions of the setup, operations, features, behaviors, functions, and interfacing is based on the default firmware, pre-programmed on the EVM, and the original EVM hardware design. Proper functionality is not guaranteed if changes are made to hardware or the firmware.

### 1.5 Hardware

This section describes the different parts of the EVM and the procedure to set up the hardware for evaluation. The top and bottom views of the EVM are shown in [Figure 1](#) and [Figure 2](#).

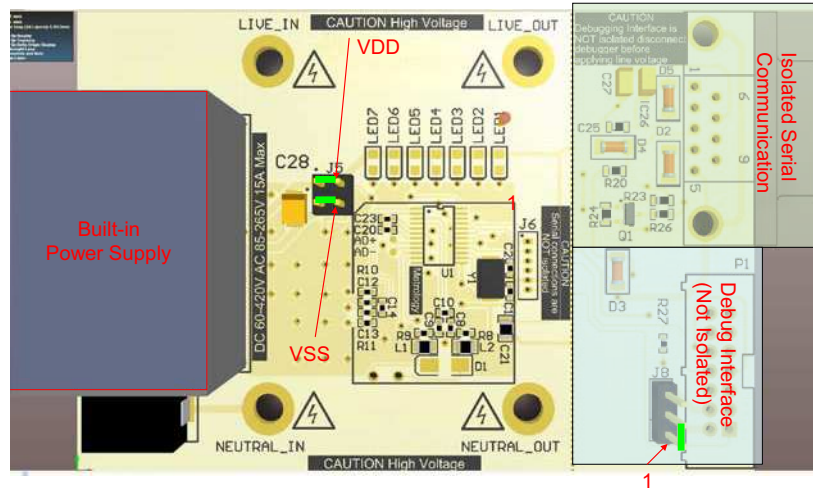


Figure 1. EVM Top View

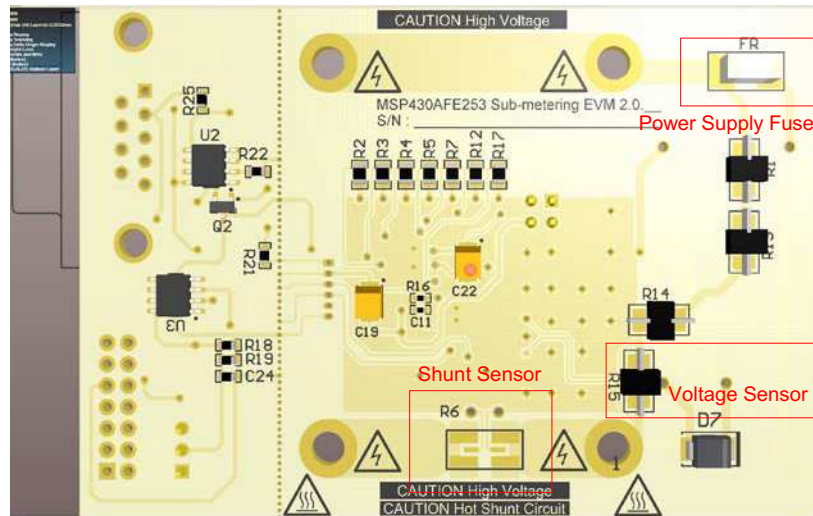


Figure 2. EVM Bottom View

## 1.5.1 Hardware Set-Up Procedures

### 1.5.1.1 Setting Up the Power Supply

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

The EVM is shipped with the built-in power supply block already installed on the EVM. If power is supplied from an external power supply, the user has to disconnect the built-in VSK-S1-3P3U power-supply block. Remove the 2 jumpers on J5, and apply 3.3V to VDD and VSS on J5, as shown in [Figure 1](#)



**Warning!** Do not supply power to the EVM until the hardware and software setup is completed

### 1.5.1.2 Setting up the Serial Communication Interface

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

### 1.5.1.3 Setting up Line Input and Load Output

**Solder** the following:

1. AC-source live or DC-source positive to the LIVE\_IN connector
2. AC-source neutral or DC-source negative to the NEUTRAL\_IN connector
3. The load's live or positive power input to the LIVE\_OUT connector
4. The load's neutral or negative power input to the NEUTRAL\_OUT connector

### 1.5.1.4 Setting up the Debugging Interface

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

If the power (while debugging) to AFE253 is to be supplied by the FET, then short 1 – 2 of J8 (towards board edge) before connecting to the MSP-FET430UIF. If the power (while debugging) to AFE253 is to be supplied by voltage on the VDD pin, then short 2 – 3 of J8 (towards board center). See [Figure 1](#)



**NOTE:** The debugging interface is **not isolated**; ensure proper isolation is in place between the EVM and the PC used for debugging

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

## 1.6 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 AFE253 that allows the EVM to operate properly. The EVM also has a software package that user can download from Texas Instruments web site. This section will discuss the software in the software package and the procedure to setup the software to operate with the EVM

### 1.6.1 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 will be discussed in [Section 1.6.2](#) and in [Section 3](#).
- Source code and Embedded Metering Library. This will be discussed in [Section 5](#) and [Appendix A](#)
- Hardware design files, including schematic, layout, and bill of materials, found in [Appendix B](#).

### 1.6.2 SETTING UP THE PC SOFTWARE TOOL

#### 1.6.2.1 Minimum System Requirement

The software tool has been tested run properly on a Pentium M 1.4GHz machine with 1.25GB RAM installed, thus it is assumed that most current computers can meet the processing power requirement.

For optimal performance, the minimum system requirements include:

- PC with built-in RS232 port or a RS232 port via USB
- PC running Microsoft Windows XP SP3 or Windows 7

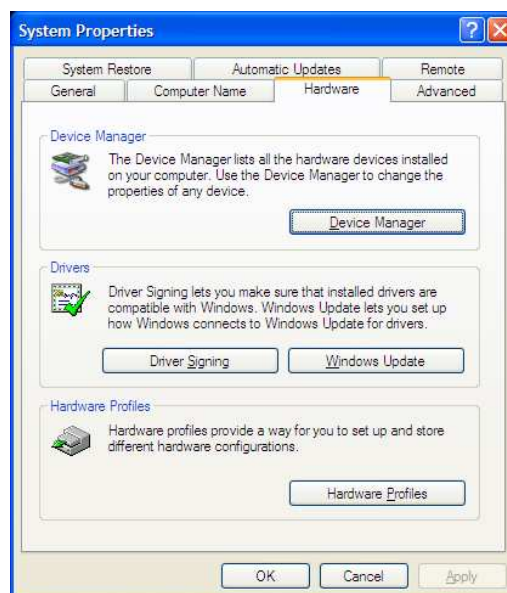
#### 1.6.2.2 Installing the Software

Simply un-zip the file named *calibrator-runtime.zip* into any folder. A folder named *calibrator-runtime* contains the necessary files to run the software tool. 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. A few user-made edits to this XML file are required before *calibrator-20121120.exe* can be launched

#### 1.6.2.3 Configuring the Software

Follow the steps below to setup the XML for *calibrator-20121120.exe* to run properly

1. Right-click the [My Computer] Icon and select [Properties] in the pop-up menu
2. Select the [Hardware] tab in the [System Properties] Window ([Figure 3](#)), then Click on the [Device Manager] to go to the [Device Manager] window



**Figure 3. System Properties Window**

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

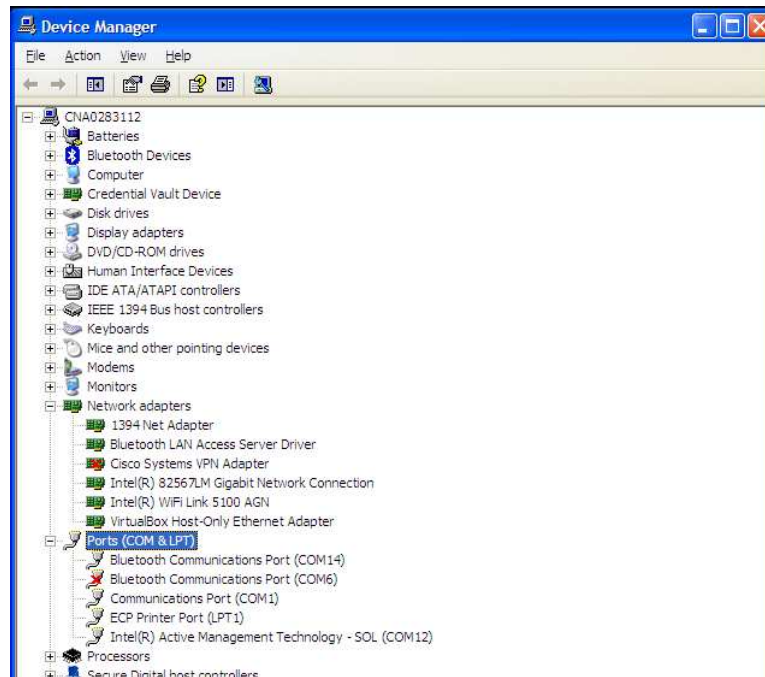


Figure 4. Device Manager Window

4. Open “calibration-config.xml” in the folder “calibrator-runtime” with a text editor or XML editor (see Figure 5). Go to the line as shown below and put in the COM port number in and save the file.

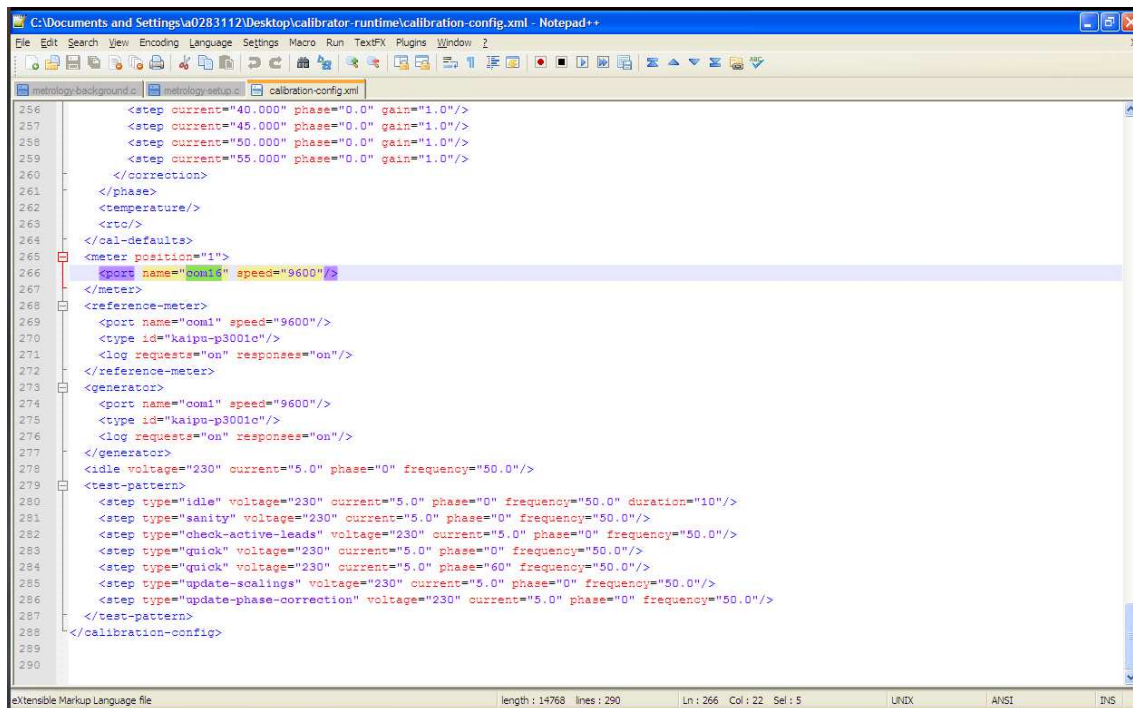


Figure 5. Editing “calibration-config.xml”



## 1.7 Instruments

The EVM has been programmed with a set of calibration factors which gives readings with an accuracy of a few percent. If more accurate result are preferred, the EVM needs to undergo a process of calibration (the procedure of calibration will be discussed in [Section 3](#). The following list of instruments are suggested to perform the calibration.

- AC source that can output sufficient power to drive the load at rated frequency (for example, 50 – 60 Hz), and rated voltage (for example, 110 – 220V)
- A variable AC load or the UUT. If DC measurement is needed, a variable DC load or the UUT is also recommended
- A reference meter that can provide AC parameters of V\_RMS, I\_RMS, P\_ACTIVE

## 2 Operating the PC Software Tool

### 2.1 Introduction

Now the setup and configuration of hardware and software is complete and this section discusses the operation of the EVM and the PC software tool. Before proceeding, make sure the steps described in [Section 1](#) are completed. Ensure power is applied to the EVM and line voltage is applied to the load.

### 2.2 Start Using the EVM

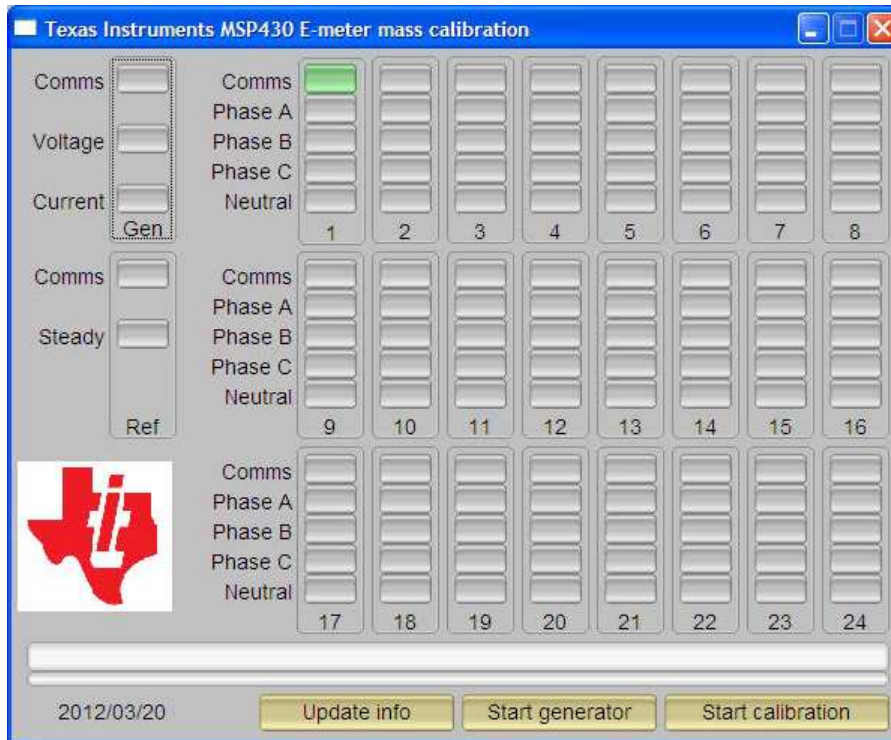
When the EVM is powered, some of the 7 LEDs will flash or turn on to indicate the operation status. Some of the LED are not used, user can make changes to the provided source code to use all 7 LEDs, as the user application requires. The list of LEDs' indication is listed in [Table 1](#).

**Table 1. LED Indication**

LED	State	Indication
LED1	ON	Background Operations Running
LED1	OFF	Background Operations Completed
LED2	ON	Negative Voltage Half Cycle
LED2	OFF	Positive Voltage Half Cycle
LED3	ON	Auto report mode
LED3	OFF	Polling mode
LED4	ON	Foreground Operations Running
LED4	OFF	Foreground Operations Completed
LED5	ON	EVM is in AC mode measurement
LED5	OFF	EVM is not in AC mode measurement
LED6	ON	EVM is in DC mode measurement
LED6	OFF	EVM is not in DC mode measurement
LED7	ON	Not used
LED7	OFF	Not used

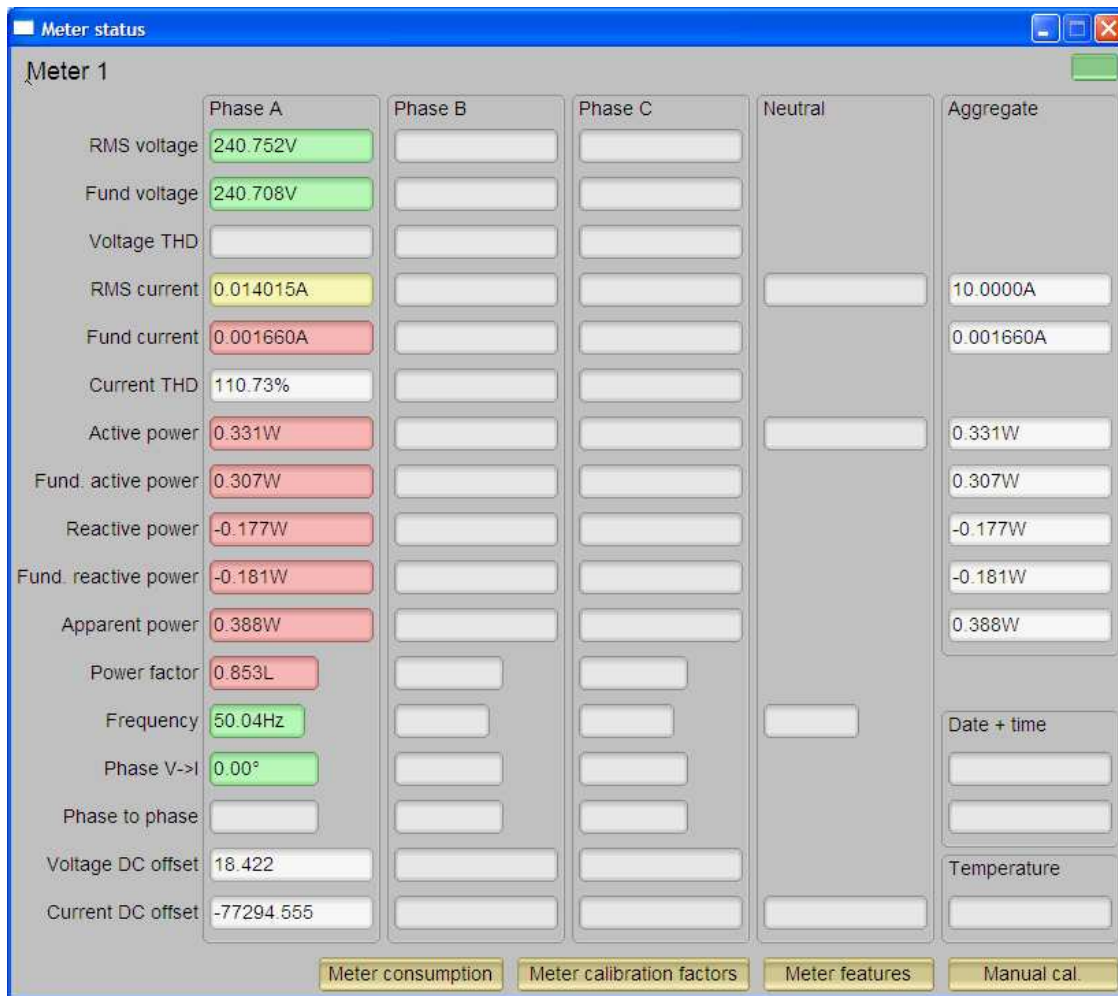
The EVM is now ready to run.

1. Launch the software “calibrator-20121120.exe” in the folder “calibration-runtime” to start communicating with the EVM. A window appears, as shown in [Figure 6](#). 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 will turn green if communication to the EVM from the PC is established and it will flash between read and green when communication is taking place.



**Figure 6. Calibrator Software Start-Up Window**

2. Click on the [Comms] indicator the [Meter Status] window will appear, as seen in [Figure 7](#)

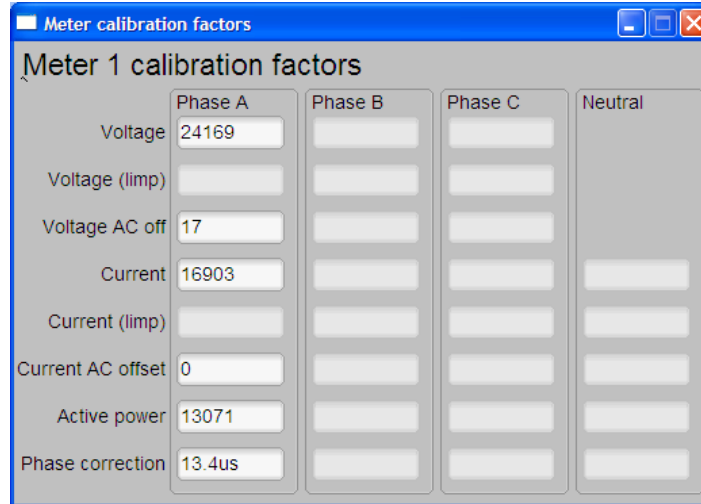


**Figure 7. Meter Status Window**

Figure 7 shows the current reading of the meter. The background of a reading box is gray if the EVM does not support that particular reading. It turns red if the reading from EVM to that box has a large variance. It turns yellow if the reading from EVM has a fairly low variance. It turns green if the reading has a low variance. Note that the software on PC reads the EVM every second and it also do some averaging to the data read, thus the update rate is slower than the update rate of the EVM.

At the bottom of this window, there are 4 buttons.

- Click on the [Meter Consumption] button to bring up the [Meter Consumption] window, but as the EVM does not support this feature, the [Meter Consumption] window will not give useful information
- Click on the [Meter Calibration Factors] button to bring up the [Meter Calibration Factors] window, shown in [Figure 8](#). In this window the current calibration factor values are shown



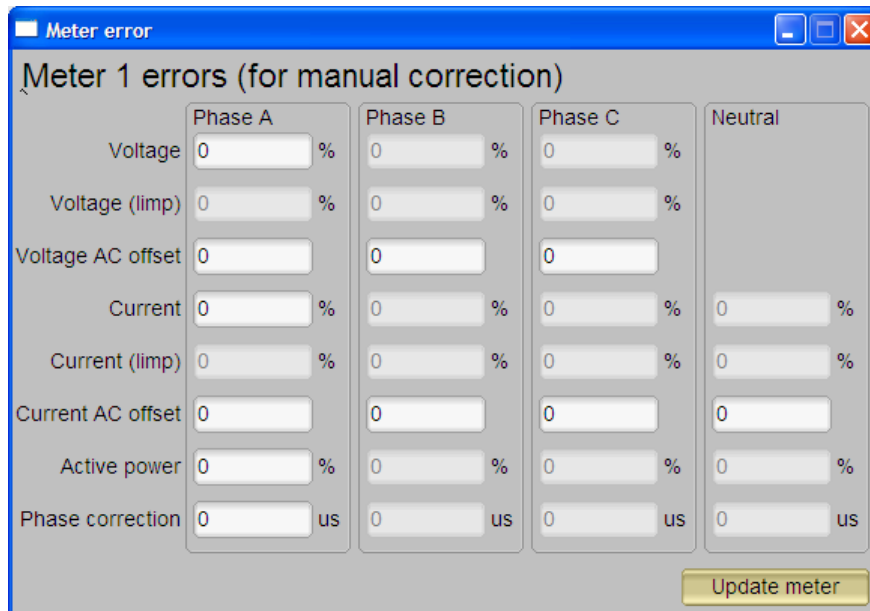
**Figure 8. Meter Calibration Factor Window**

- Click on the [Meter Features] buttons brings up the [Meter Features] window, as shown in [Figure 9](#). In this window, the support feature of the EVM is reported in the [Meters Feature] window.



**Figure 9. Meter Features Window**

- Click on the [Manual Cal] button brings up the [Meter Error] window, as shown in [Figure 10](#). 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 will be discussed, see more details in [Section 3](#)



**Figure 10. Meter Error Window**

- To modify the calibration factor, it is required that the correction is entered in the percentage error. The percentage error is calculated as shown in [Equation 1](#).

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

- Put the percentage error into the corresponding box of the Meter Calibration Window.
- Click [Update meter]. The updated calibration values will be calculated and written to the EVM, and the corresponding values will be reflected on the [Meter Calibration Factor Window].

### 2.3 Known Issues

The calibrator software is legacy software that operates with utility meters that has not been completely customized for embedded metering. Here is a list of known issues with the existing software that will be fixed when the next version of calibrator software customized for embedded metering is released.

- Neutral Monitoring is shown in the [Meters Features] window as a supported feature; this is not a correct indication. When wire resistance compensation or inlet capacitor compensation is enabled the calibrator will interpreted this as neutral monitoring support.
- Resistance value of wire resistance compensation and capacitance value of inlet capacitor compensation cannot be programmed with the calibrator software.
- Voltage AC offset value cannot be written with the calibrator software. The value put into the Voltage AC offset box is actually written to Current AC offset instead. The value put into the Current AC offset box has no effect.
- DC Offset values cannot be written to the EVM with the calibration software, the current firmware will take the current and voltage DC offset value every time any calibration value is updated.

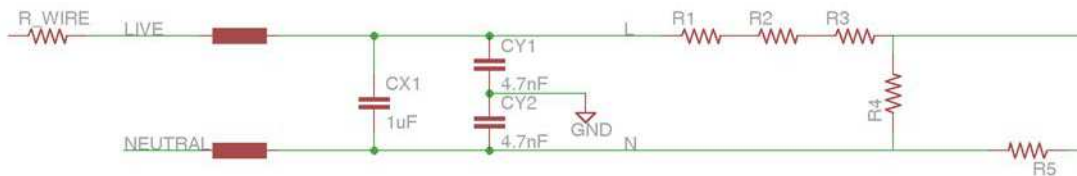
## 3 Calibration Techniques

### 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 is not calibrated before being shipped. To maximize accuracy and to compensate component and manufacturing tolerance, it is necessary for the EVM to go through a calibration process. This chapter explains the calibration techniques, required instruments, and the procedures and steps of calibration.

### 3.2 Calibration Techniques

The calibration of the EVM is defined based on the front-end interface model as shown in [Figure3-1](#)



**Figure 11. 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 and 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:** Since the calibration values are written in one flash page in the EVM, thus when a new value needs to be written the whole page is erased. The provided GUI will do the read, modify and write operation automatically. User must read the complete set of calibration values, back it up, update the modified field and write the complete set back to the EVM when using a user calibration facility.

---

### 3.3 Calibration Procedures

#### 3.3.1 Calibration of AC and DC Parameters

Use the steps listed below to calibrate the AC and DC parameters

##### 3.3.1.1 Calibrating VGAIN

1. Set to No load/ lowest possible load
2. Set VIN to line voltage
3. Calculate the value for VGAIN with the formula:

$$VGAIN_{n+1} = \frac{V_{REF}}{V_{UUT}} \times VGAIN_n \quad (2)$$

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

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

4. Write and apply the calibrated VGAIN

### 3.3.1.2 **Calibrating IGAIN**

1. Set VIN to line voltage
2. Set to High / Highest load
3. Calculate IGAIN value with the formula

$$IGAIN_{n+1} = \frac{I_{REF}}{I_{UUT}} \times IGAIN_n \quad (4)$$

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

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

4. Write and apply the calibrated IGAIN

### 3.3.1.3 **Calibrating PGAIN**

1. Same condition as above (assume at this point the power factor is very close to 1)
2. Note the percentage error on voltage at this point
3. Calculate PGAIN gain so that power is of the same percentage error as voltage at this point.

$$PGAIN_{n+1} = \frac{P_{REF}}{P_{UUT}} \times PGAIN_n \times (1 - \%Error \text{ of } V_{UUT} \text{ at this load}) \quad (6)$$

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

$$\%Error = (1 + \%Error)(1 - \%Error \text{ of } V_{UUT} \text{ at this load}) \quad (7)$$

4. Write and apply the calibrated PGAIN

## 3.3.2 **CALIBRATION OF COMPENSATION RESISTANCE AND CAPACITANCE**

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

1. Calibrating RES
  - (a) Calculate RES with the give equation

$$R_{WIRE} = \frac{V_{REF(I_{max})} - V_{UUT(I_{max})}}{I_{max} - I_{min}} \approx \frac{V_{REF(I_{max})} - V_{UUT(I_{max})}}{I_{max}} \quad (8)$$

- (b) Write and apply the calibrated RES (Note the resistance is in 1/256 – ohm unit)

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

$$C = \frac{1}{2\pi f V^2} \left( \sqrt{P_{APPERANT\_REF}^2 - P_{ACTIVE}^2} - \sqrt{P_{APPERANT\_UUT}^2 - P_{ACTIVE}^2} \right) \quad (9)$$

### 3.3.3 Calibration of Current AC Offset

Current AC offset is the result of noise picked up and generated on the shunt resistor circuit, causing an illusion of having a finite current flowing when there is not actually a current flowing through the shunt. Although this noise current does not affect the accuracy of the power reading, this actually gives contribution to the current reading and its accuracy, especially when the current is small.

To offset this noise current, the EVM firmware has the mechanism to remove this from the current reading, as detailed in the following steps:

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 take an average as I\_NOISE (in A)
4. Calculate the current AC offset value with the equation (Note: This is a big number, even in case of a few mA of noise)

$$I\_AC\_OFFSET = \text{int} \left( I\_NOISE \left( \frac{1024 \times 10^6}{IGAIN} \right) \right)^2 \quad (10)$$

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 DC Parameters

The design of the EVM allows the DC measurement parameters be calibrated automatically the same time AC is being calibrated. In the last step of calibration (Current AC offset) when it is updated the DC measurement parameters are updated automatically.

In fact, whenever the complete set of calibration value is read, the most update DC measurement parameters is also included in the set. When an update is done to other parameters the DC measurement parameter is also update to the most updated value. However the value is most accurate when current is low (best with not current at all) thus it is suggested to perform DC measurement parameter update after the I\_AC\_OFFSET calibration.

---

**NOTE:** This may not be a user friendly way of setting DC measurement parameters. The procedure of having the best DC measurement parameters will be modified and improved when the embedded metering customized calibration software is released.

---



## 4 Serial Communication Commands

To enable the EVM to be a useful tool external interface is a must have. The EVM uses its UART port connected to a RS232 DB-9 connector to communicate to the external. In this chapter the protocol of the EVM sending out readings and commands will be discussed.

**NOTE:** User could made change to the provided source code to perform custom communication protocol, command, response and functions.

### 4.1 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 2 communication modes, auto-reporting mode and polling mode. Without the need of host interaction, auto-reporting mode returns the basic measurement result from EVM. Polling mode, however, gives access to the full range of functionality, reading of complete set of measurement results, reading and writing calibration factors.

#### 4.1.1 Auto-Reporting Mode

When EVM is powered up or reset it is automatically set to auto-reporting mode. In this mode the EVM automatically sends out readings consist of voltage, current, power, power factor every 4 AC cycle when performing AC measurement, or every 80ms when performing DC measurement. The data format of the data sent during auto-reporting mode is shown in Figure 4 1

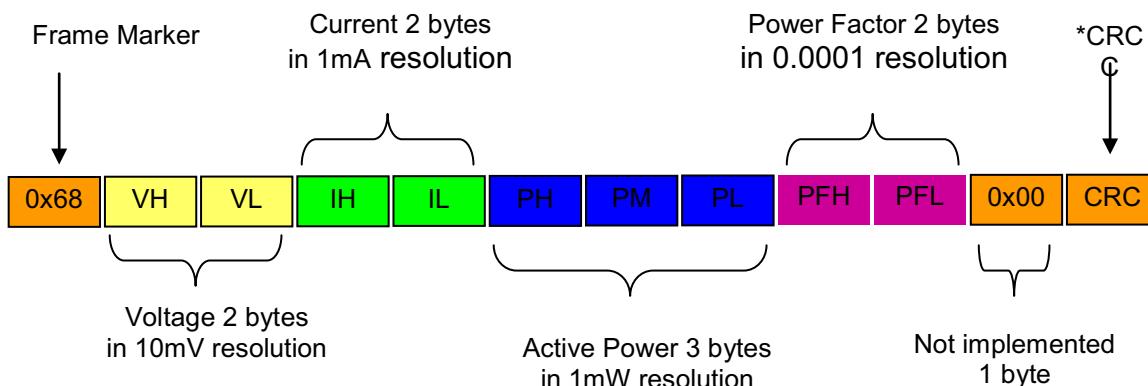


Figure 12. Auto-Report Mode Data Frame Format

In auto-reporting mode, frame start is indicated by the marker 0x68 and ended with a CRC. User should make use of this marker and the CRC byte to track the boundary of a frame.

The bit sequence 100000110 equivalent to the generator polynomial

$$x^8 + x^2 + x$$

is used for CRC calculation.

The source code for CRC calculation and an example calling is shown in Appendix D.

In auto-reporting mode, the host should only send `HOST_CMD_SET_PASSWORD` or `HOST_CMD_SET_POLLING_REPORT_MODE` command to the EVM. Sending other commands may cause unknown respond. See [HOST\\_CMD\\_SET\\_PASSWORD](#) and [HOST\\_CMD\\_SET\\_POLLING\\_REPORT\\_MODE](#) for more details on these commands. Upon receiving either command, the EVM will set itself to polling mode.

### 4.1.2 Polling Mode

When EVM is running in polling mode, the EVM waits for a command from the host through the serial port. When the command frame received passes the frame checking the EVM will interpret the command and performs the requested action. The EVM communicate with the host using a data frame format as indicated in [Figure 13](#).

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 13. Polling Mode Data Frame Format**

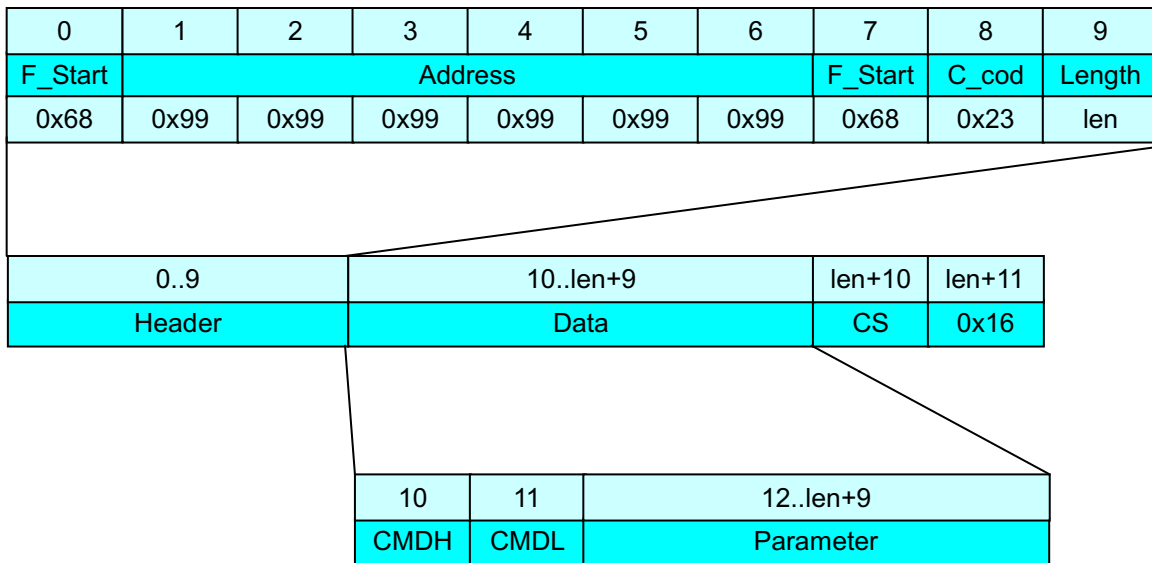
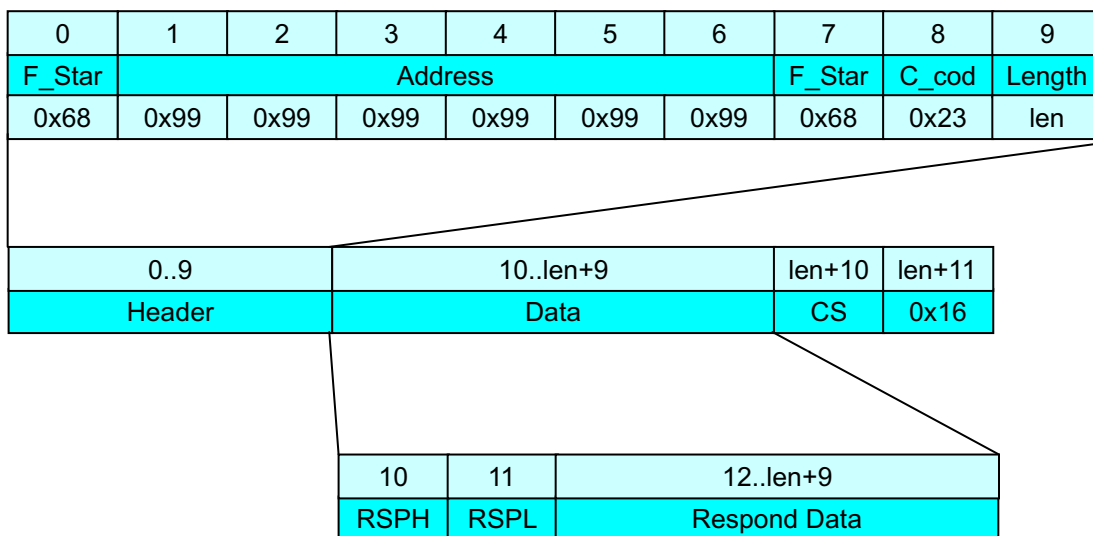
The frame starts with a 9 bytes header followed by a data field of 0 to 255 byte then a check sum byte and a frame ending byte.

The header starts with 0x68 as auto-reporting do, followed by an address filed which is fixed to 0x999999999999. User can modify the provided communication protocol source code 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.

#### 4.1.2.1 Command and Respond Frame

[Figure 14](#) and [Figure 15](#) Polling Mode Respond Frame Format shows 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 will respond the host with a respond frame with the RSPH = CMDH and RSPL = CMDL | 0x80 (because of that CMDL cannot use value larger than 0x7F). The respond data is defined by the command received.


**Figure 14. Polling Mode Command Frame Format**

**Figure 15. Polling Mode Respond Frame Format**


---

**NOTE:** When writing custom protocol, make sure that the data buffer is sufficient to hold the data byte from the host

---

## 4.2 COMMANDS

### HOST\_CMD\_SET\_AUTO\_REPORT\_MODE —

This command sets the EVM to return to auto-reporting mode from polling mode

**Table 2. HOST\_CMD\_SET\_AUTO\_REPORT\_MODE  
Command Format**

	Command			Respond		
<b>LEN</b>	2			2		
	<b>Offset</b>	<b>Width</b>	<b>Data</b>	<b>Offset</b>	<b>Width</b>	<b>Data</b>
<b>CMDH</b>	0	U8	0x50	0	U8	0x50
<b>CMDL</b>	1	U8	0x00	1	U8	0x80

### HOST\_CMD\_SET\_POLLING\_REPORT\_MODE —

This command set the EVM to polling mode from auto-reporting mode.

**Table 3. HOST\_CMD\_SET\_POLLING\_REPORT\_MODE  
Command Format**

	Command			Respond		
<b>LEN</b>	2			2		
	<b>Offset</b>	<b>Width</b>	<b>Data</b>	<b>Offset</b>	<b>Width</b>	<b>Data</b>
<b>CMDH</b>	0	U8	0x51	0	U8	0x51
<b>CMDL</b>	1	U8	0x00	1	U8	0x80

### HOST\_CMD\_GET\_METER\_NAME —

Read the 32 byte meter name string as defined by #define METER\_NAME in “metrology-calibration-template.h”

**Table 4. HOST\_CMD\_GET\_METER\_NAME  
Command Format**

	Command			Respond		
<b>LEN</b>	2			34		
	<b>Offset</b>	<b>Width</b>	<b>Data</b>	<b>Offset</b>	<b>Width</b>	<b>Data</b>
<b>CMDH</b>	0	U8	0x52	0	U8	0x52
<b>CMDL</b>	1	U8	0x00	1	U8	0x80
				2	U8(32)	32 byte meter name

**HOST\_CMD\_GET\_METER\_VER —**

Read the 4 x 32 bit version numbers as defined by:

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

In "metrology-calibration-template.h"

**Table 5. HOST\_CMD\_GET\_METER\_VER  
Command Format**

	Command			Respond		
<b>LEN</b>	2			18		
	<b>Offset</b>	<b>Width</b>	<b>Data</b>	<b>Offset</b>	<b>Width</b>	<b>Data</b>
<b>CMDH</b>	0	U8	0x53	0	U8	0x53
<b>CMDL</b>	1	U8	0x00	1	U8	0x80
				2	U8(4)	4 bytes of software version
				6	U8(4)	4 bytes of hardware version
				10	U8(4)	4 bytes of metrology version
				14	U8(4)	4 bytes of protocol version

**HOST\_CMD\_GET\_METER\_CONFIGURATION —**

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

**Table 6. HOST\_CMD\_GET\_METER\_CONFIGURATION Command Format**

	Command			Respond		
<b>LEN</b>	2			20		
	Offset	Width	Data	Offset	Width	Data
<b>CMDH</b>	0	U8	0x56	0	U8	0x56
<b>CMDL</b>	1	U8	0x00	1	U8	0x80
				2	U8	Number of Phases
				3	U8	Features 0*
				4	U8	Features 1*
				5	U8	Features 2*
				6	U8	Features 3*
				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

**Table 7. Parameter Definition**

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

**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 MSP430AFE253.

**Table 8. HOST\_CMD\_GET\_RTC  
Command Format**

	Command			Respond		
<b>LEN</b>	2			10		
	<b>Offset</b>	<b>Width</b>	<b>Data</b>	<b>Offset</b>	<b>Width</b>	<b>Data</b>
<b>CMDH</b>	0	U8	0x59	0	U8	0x59
<b>CMDL</b>	1	U8	0x00	1	U8	0x80
				2	U8(4)	4 bytes software version
				8	S16	Temperature in 0.01C

**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.

**Table 9. HOST\_CMD\_ALIGN\_WITH\_CALIBRATION\_FACTORS  
Command Format**

	Command			Respond		
<b>LEN</b>	2			2		
	<b>Offset.</b>	<b>Width</b>	<b>Data</b>	<b>Offset</b>	<b>Width</b>	<b>Data</b>
<b>CMDH</b>	0	U8	0x5A	0	U8	0x5A
<b>CMDL</b>	1	U8	0x00	1	U8	0x80

**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 is default to:

- Password 1 = 0x1234
- Password 2 = 0x5678
- Password 3 = 0x9ABC
- Password 4 = 0xDEF0.

**Table 10. HOST\_CMD\_SET\_PASSWORD  
Command Format**

	Command			Respond		
<b>LEN</b>	10			20		
	Offset	Width	Data	Offset	Width	Data
<b>CMDH</b>	0	U8	0x56	0	U8	0x56
<b>CMDL</b>	1	U8	0x00	1	U8	0x80
	2	U16	Password 1			
	4	U16	Password 2			
	6	U16	Password 3			
	8	U16	Password 4			

**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:** In the case of this EVM only CMDH = 0x61 is supported.

**Table 11. HOST\_CMD\_GET\_READINGS\_PHASE\_N  
Command Format**

	Command			Respond		
<b>LEN</b>	2			34		
	Offset	Width	Data	Offset	Width	Data
<b>CMDH</b>	0	U8	0x6n (n = 1, 2, 3)	0	U8	0x6n (n = 1, 2, 3)
<b>CMDL</b>	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 uA
				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



**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:** In the case of this EVM only CMDH = 0x69 is supported.

**Table 12. HOST\_CMD\_GET\_EXTRA\_READINGS\_PHASE\_N  
Command Format**

		Command			Respond		
<b>LEN</b>	2			34			
	<b>Offset</b>	<b>Width</b>	<b>Data</b>	<b>Offset</b>	<b>Width</b>	<b>Data</b>	
<b>CMDH</b>	0	U8	0x68 + n (n = 1, 2, 3)	0	U8	0x68 + n (n = 1, 2, 3)	
<b>CMDL</b>	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 uA	
				18	U16	Voltage THD in 0.01%	
				20	U16	Current THD in 0.01%	
				22		Next 12 bytes reserved	

**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).

**Table 13. HOST\_CMD\_SUMCHECK\_MEMORY  
Command Format**

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

**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.

**Table 14. HOST\_CMD\_CLEAR\_CALIBRATION\_DATA  
Command Format**

	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

**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.

**Table 15. HOST\_CMD\_SET\_CALIBRATION\_PHASE\_N  
Command Format**

	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 nF			
	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 ohm			
	24	U16	Irms Scaling Factor			
	26	U16	0x0000 reserved			
28	U16	Power Scaling Factor				

**HOST\_CMD\_GET\_CALIBRATION\_PHASE\_N —**

This command reads the calibration value from the EVM.

**Table 16. HOST\_CMD\_GET\_CALIBRATION\_PHASE\_N Command Format**

	Command			Respond		
<b>LEN</b>	2			30		
	Offset	Width	Data	Offset	Width	Data
<b>CMDH</b>	0	U8	0xD6 + n	0	U8	0xD6 + n ( n = 1, 2, 3)
<b>CMDL</b>	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 nF
				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 ohm
				24	U16	Irms Scaling Factor
				26	U16	Reserved
				28	U16	Power Scaling Factor

**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.

**Table 17. HOST\_CMD\_Get\_CALIBRATION\_PHASE\_N Command Format**

	Command			Respond		
<b>LEN</b>	10			2		
	Offset	Width	Data	Offset	Width	Data
<b>CMDH</b>	0	U8	0xD5	0	U8	0xD5
<b>CMDL</b>	1	U8	0x00	1	U8	0x80
	2	U16	Calibration Status			
	4	U16	Intercept Temperature			
	6	U16	Temperature Intercept			
	8	U16	Temperature slope and degree			

**HOST\_CMD\_GET\_CALIBRATION\_EXTRAS —**

This command reads the extra calibration value from the EVM

**Table 18. HOST\_CMD\_Get\_CALIBRATION\_EXTRAS  
Command Format**

	Command			Respond		
<b>LEN</b>	2			10		
	Offset	Width	Data	Offset	Width	Data
<b>CMDH</b>	0	U8	0xDA	0	U8	0xDA
<b>CMDL</b>	1	U8	0x00	1	U8	0x80
				2	U16	Calibration Status
				4	U16	Intercept Temperature
				6	U16	Temperature Intercept
			8	U16	Temperature Slope and Degree	

## 5 Firmware and Embedded Metering Library API

The EVM comes with pre-tested firmware that enables the EVM to operate. The source code together with the embedded metering library (emtere-metrology-afe253.r43), 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.

### 5.1 FIRMWARE STRUCTURE

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

In a user's view, the firmware is partitioned into 3 main blocks: the application, the metrology computation engine, and the toolkit.

- The application contains the following:
  - System set-up and initialization
  - Main loop
  - Communication protocol and command handling
  - Non-volatile parameters preset and manipulation
- The metrology computation engine, packaged into a library named "emeter-metrology-afe253.r43" contains the following:

---

**NOTE:** The source code of this library is not included in the package. Contact your Texas Instruments Sales Team in your area if you require the source code.

---

- ADC set-up
- Parameter initializations
- Sample-based background processing
- Reporting cycle-based foreground processing
- Reading application interface
- The toolkit packaged into a library named "emeter-toolkit-afe253.r43", contains the following:
  - Low-level computation routines

### 5.2 The 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, 32-bit by 16-bit multiplication
  - 16-bit, 32-bit, 64-bit square root, integer square root
  - 16-bit by 16-bit multiple accumulate into 48-bit accumulator
  - 16-bit by 24-bit multiply accumulate into 64-bit accumulator
  - Q1.15 fix-point number multiply
  - 16-bit square and accumulate into 48-bit accumulator
  - 24-bit square and accumulate into 64-bit accumulator

### 5.3 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 formulas in Equation 11 are used in the metrology computation.

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

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

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

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

$$P_{\text{apparent}} = V_{\text{RMS}} \times I_{\text{RMS}}$$

$$\text{PF} = \cos\phi = \frac{P_{\text{active}}}{P_{\text{apparent}}}$$

(11)

#### 5.3.1 The Background Process

In the background, time critical sample based process is performed. The background processing is quite straight forward. The process is started by a trigger, at the sample rate, sampling complete interrupt. The background process then

1. Captures 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) Adjust phase correction to voltage(i) sample
3. Current sample processing
  - (a) Current channel high-pass filtering and DC offset removal
  - (b) Adjust phase correction to current(i) sample
  - (c) Integration of square of current(i) sample

4. Power processing
  - (a) Integrate voltage(i) \* current(i)
  - (b) Integrate quadrature voltage(i) \* current(I)
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 fore the foreground to process

### 5.3.2 The Foreground Process

After the background has collected sufficient data the flag PHASE\_STATUS\_NEW\_LOG will be set by the background. The user main loop will check the status of this flag and will call to the foreground process `calculate_phase_readings ()` to perform the rest of the computation to deliver the measurement readings. The foreground process include

- Power processing
  - Calculate the active power by scaling with the number of samples and the power scaling factor
  - Calculate the reactive power by scaling with the number of samples and the power scaling factor
- Voltage processing
  1. Calculate the RMS voltage by scaling with the number of samples, perform square root and multiply with the voltage scaling factor
- Current Processing
  1. Calculate the RMS current by scaling with the number of samples, perform square root and multiply with the current scaling factor
  2. Calculate the amount of compensation to current due to the inlet capacitor
  3. Calculate the apparent power by multiplying the root mean square voltage and the root means square current
- Other processing
  1. Calculate the power factor
  2. Calculate the frequency of the AC line

## 5.4 EMBEDDED METERING LIBRARY API

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.

### 5.4.1 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, including
  1. Initialize the metrology library and the ADC hardware
  2. Initialize the metrology library with the default calibration parameter
- Performing metrology calculations with data collected by background processing
- Reading results from metrology calculations

### 5.4.1.1 Metrology Engine Control Functions

#### **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.

#### **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

#### **void align\_metrology\_with\_calibration\_data (void) —**

Reinitialize the metrology related ADC channels with proper phase correction.

**Parameter:**

none

**Return:**

none

#### **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

#### **void metrology\_init\_analog\_front\_end\_normal\_mode (void) —**

Initialize the hardware of all metrology-related ADC channels

**Parameter:**

none

**Return:**

none

#### **void metrology\_disable\_analog\_front\_end (void) —**

Disable the hardware of all metrology-related ADC channels

**Parameter:**

none

**Return:**

none



#### 5.4.1.1.1 **How to Initialize the Metrology Engine**

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. Complete other set-up operations that require the ADC interrupt to be disabled
5. [metrology\\_switch\\_to\\_normal\\_mode](#)

#### 5.4.1.2 **Calculate and Reading the Readings Functions**

##### **power\_t calculate\_phase\_readings (void) —**

This function must be called by the 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 processes. Upon returning all metrology reading is updated.

**Parameter:**

none

**Return:**

Active power reading

##### **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

##### **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

##### **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

##### **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.

**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.

**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.

**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.

**uint16\_t phase\_status (int ph) —**

Returns the status word of phase ph

**Parameter:**

ph = 1

**Return:**

Status word of phase ph

#### 5.4.2 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*.

**void BACKGROUND\_PROCESS\_ON(void) —** Called when background process starts

**void BACKGROUND\_PROCESS\_OFF(void) —** Called when background process finished

**void FOREGROUND\_PROCESS\_ON(void) —** Called when foreground process starts

**void FOREGROUND\_PROCESS\_OFF(void) —** Called when foreground process finished

**void AC\_MODE\_ON(void) —** Called when entering AC measurement mode

**void AC\_MODE\_OFF(void) —** Called when leaving AC measurement mode

**void DC\_MODE\_ON(void) —** Called when entering DC measurement mode

**void DC\_MODE\_OFF(void) —** Called when leaving DC measurement mod

## 5.4.3 Application-Level Calibration Functions

### 5.4.3.1 Reading and Writing Calibration Parameters Functions

#### **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.

#### **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.

#### **int\_clear\_calibration\_data (void) —**

Clear all calibration data in calibration parameter memory.

**Note:** Must clear calibration data 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.

#### **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.

#### **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.

#### **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 at which the calibration was done

`temperature_sensor_intercept` – The ADC reading at calibration temperature.

`temperature_sensor_slope` – The ADC reading increment per 0.1 degree Celsius

**Return:**

none

**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*)

**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

**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*)

**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

**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

**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.

**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

**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.

**void set\_v\_ac\_offset (int phx, int32\_t value) —**

Write the voltage channel AC offset of phase phx with value.

**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. Value should be calculated as shown in [Equation 12](#)

$$V\_AC\_OFFSET = \text{int} \left( V\_NOISE \left( \frac{1024 \times 10^3}{V\_GAIN} \right) \right)^2 \quad (12)$$

**Return:**

none

**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*).

**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

**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.

**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.

**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

**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

**void set\_i\_ac\_offset (int phx, int32\_t value) —**

Write the current channel AC offset of phase phx with value.

**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. Value should be calculated as:

$$I\_AC\_OFFSET = \text{int} \left( I\_NOISE \left( \frac{1024 \times 10^6}{IGAIN} \right) \right)^2 \quad (13)$$

**Return:**

none

**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 uF units) in calibration parameter memory.

**void set\_compensate\_capacitor\_value (int phx, uint16\_t value) —**

Write EMI filter capacitor value of phase phx with value.

**Parameter:**

phx = 1

value = the capacitance to be written (in 1/64 uF units) to calibration parameter memory

**Return:**

none

**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 1/256  $\Omega$  units)

**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 1/256  $\Omega$  units) to calibration parameter memory

**Return:**

none

**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 us) in calibration parameter memory

**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 us) to be written to calibration parameter memory

**Return:**

None

### 5.4.3.2 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.

---

**DEFAULT\_TEMPERATURE\_INTERCEPT** —ADC reading at the calibration temperature in 16-bits scale

**DEFAULT\_TEMPERATURE\_SLOPE** —ADC counts per 0.1 degree Celsius increment

**DEFAULT\_ROOM\_TEMPERATURE** —Calibration temperature in 0.1 degree C unit

**DEFAULT\_BASE\_PHASE\_A\_CORRECTION** —The phase correction between voltage and current channel due to hardware

**DEFAULT\_P\_SCALE\_FACTOR\_A** —Power-scaling factor

**DEFAULT\_V\_RMS\_SCALE\_FACTOR\_A** —Voltage-scaling factor

**DEFAULT\_V\_DC\_ESTIMATE\_A** —DC offset of voltage channel ADC

**DEFAULT\_V\_AC\_OFFSET\_A** —Square of estimated noise level on voltage channel ADC (too small to be noticeable, usually put 0)

**DEFAULT\_I\_RMS\_SCALE\_FACTOR\_A** —Current scaling factor

**DEFAULT\_I\_DC\_ESTIMATE\_A** —DC offset of current channel ADC

**DEFAULT\_I\_AC\_OFFSET\_A** —Square of estimated noise level on voltage channel ADC (usually put 0)  
Calculated as

$$\text{DEFAULT\_I\_AC\_OFFSET} = \left( \frac{1024 \times 10^6}{\text{Current Scaling Factor}} \times \text{Estimated noise level} \right)^2 \quad (14)$$

**DEFAULT\_EMI\_FILTER\_CAP\_UF\_A** —EMI filter capacitance in 1/64 uF unit, Maximum 1023 (15.984375uF). Putting value greater than 0x8000 will result in no EMI filter compensation done

**DEFAULT\_WIRE\_RESISTANCE\_A** —Estimate wire resistance in 1/256 Ω unit, Maximum 255 (0.99609375 Ω)

**METER\_NAME** —Defines the name of the meter as reported by `HOST_CMD_GET_METER_NAME` and is defined as in the provided example code "MSP430AFE253SUBMETEREVM"

**METER\_SOFTWARE\_VERSION** —Defines the software version of the meter as reported by `HOST_CMD_GET_METER_VER`

**METER\_HARDWARE\_VERSION** —Defines the hardware version of the meter as reported by `HOST_CMD_GET_METER_VER`

**METER\_METROLOGY\_VERSION** —Defines the Metrology library version of the meter as reported by `HOST_CMD_GET_METER_VER`

**HOST\_PROTOCOL\_VERSION** —Defines the protocol version of the meter as reported by `HOST_CMD_GET_METER_VER`



## 6 Bill of Materials

To download the bill of materials (BOM) for each board, see the design files at [www.ti.com/tool/TIDM-MSP430AFE253SUBMETEREVM](http://www.ti.com/tool/TIDM-MSP430AFE253SUBMETEREVM). Table 19 shows the BOM for the SAT00xx.

**Table 19. Bill of Materials**

Comment	Description	Designator	Footprint	LibRef	Quantity	Value	Digikey Link
Printed Circuit Board	Printed Circuit Board	PCB		PCB	1		
GRM1555C1E180JA01D	CAP, CERM, 12pF, 25V, +/-5%, C0G/NP0, 0402	C1, C2	0402	GRM1555C1E180JA01D	2	18pF	<a href="#">Link</a>
GRM1555C1H470JZ01	CAP, CERM, 47pF, 50V, +/-5%, C0G/NP0, 0402	C8, C9, C12, C13	0402	GRM1555C1H470JZ01	4	47pF	<a href="#">Link</a>
GRM155R71H153KA12D	CAP, CERM, 0.015uF, 50V, +/-10%, X7R, 0402	C10, C14	0402	GRM155R71H153KA12D	2	0.015uF	<a href="#">Link</a>
C1005X5R1A104K	CAP, CERM, 0.1uF, 10V, +/-10%, X5R, 0402	C11, C20, C23	0402	C1005X5R1A104K	3	0.1uF	<a href="#">Link</a>
293D106X0010B2TE3	CAP, TANT, 10uF, 10V, +/-20%, 2.5 ohm, 3528-21 SMD	C19, C22	3528-21	293D106X0010B2TE3	2	10uF	<a href="#">Link</a>
0805ZC222JAT2A	CAP, CERM, 0.1uF, 25V, +/-10%, X7R, 0805	C21	0805	0805ZC222JAT2A	1	2.2n	<a href="#">Link</a>
C1608X7R1H104K	CAP, CERM, 0.1uF, 50V, +/-10%, X7R, 0603	C24, C25	0603	C1608X7R1H104K	2	0.1uF	<a href="#">Link</a>
F931D106KAA	CAP, TANT, 10uF, 50V, +/-10%, 0.8 ohm, 7343-43 SMD	C26, C27	3216-18	F931D106KAA	2	10uF	<a href="#">Link</a>
TCJB227M006R0200	CAP, TANT, 220uF, 6.3V, +/-10%, 0.6 ohm, 7343-31 SMD	C28	3528-12	TCJB227M006R0200	1	220uF	<a href="#">Link</a>
SMAJ5.0CA		D1, D8	DSO-C2/X1.8	SMAJ5.0CA	2		<a href="#">Link</a>
LL103A		D2, D3, D4, D5	SOD-80	LL103A	4		<a href="#">Link</a>
V320SM7		D6	V320SM7	V320SM7	1		<a href="#">Link</a>
SMBJ7.0A-13-F	Diode, TVS, Uni, 7V, 600W, SMB	D7	SMB	SMBJ7.0A-13-F	1	7V	<a href="#">Link</a>
6125FA1A	Fuse, 1A,125V, SMD	FR	6125FA	6125FA1A	1		<a href="#">Link</a>
A106735-ND	Header, TH, 100mil, 2x2, Gold plated, 230 mil above insulator	J5	A106735-ND	A106735-ND	1		<a href="#">Link</a>
1.27mm pitch socket		J6	6-pin header (1.27mm)	6 pin header (1.27mm)	1		<a href="#">Link</a>
D Connector 9	Receptacle Assembly, 9 Position, Right Angle	J7	DSUB1.385-2H9	A35109-ND	1		<a href="#">Link</a>
PBC03SAAN	Header, TH, 100mil, 1x3, Gold plated, 230 mil above insulator	J8	PBC03SAAN	PBC03SAAN	1	1x3	<a href="#">Link</a>
BLM21BD121SN1D	Inductor, [Technology], [Material], xxuH, xxA, x.xx ohm, [MountType]	L1, L2		BLM21BD121SN1D	2		<a href="#">Link</a>
Red		LED1, LED5	0805-LED	LED-0805	2		<a href="#">Link</a>
Green		LED2, LED6	0805-LED	LED-0805	2		<a href="#">Link</a>
Blue		LED3, LED7	0805-LED	LED-0805	2		<a href="#">Link</a>
White		LED4	0805-LED	LED-0805	1		<a href="#">Link</a>

**Table 19. Bill of Materials (continued)**

Used in BOM report		LIVE_IN, LIVE_OUT, NEUTRAL_IN, NEUTRAL_OUT	Hole for Banana Socket	Hole for Banana Socket	4		<a href="#">Link</a>
A104860-ND	Header, 7-Pin, Dual row	P1	2X7BOXHEADER	A104860-ND	1		<a href="#">Link</a>
BC857B	Transistor, PNP, 45V, 0.1A, SOT-23	Q1, Q2	SOT-23	BC857B	2	0.075V	<a href="#">Link</a>
ERJ-S1DF3303U	RES, 330kohm, 1%, 1W, 2010	R1, R13, R14	2010M	ERJ-S1DF3303U	3	330k	<a href="#">Link</a>
ERJ-6GEYJ391V	RES, 390 ohm, 5%, 0.125W, 0805	R2, R3, R4, R5, R7, R12, R17	0805	ERJ-6GEYJ391V	7	390	<a href="#">Link</a>
ULRB12512R0005FLFSLT	Shunt Resistor	R6	Current Sensing Pad	ULRB12512R0005FLFSLT	1		<a href="#">Link</a>
CRCW04021K00FKED	RES, 1.00k ohm, 1%, 0.063W, 0402	R8, R9, R11	0402	CRCW04021K00FKED	3	1.00k	<a href="#">Link</a>
CRCW0402100RFKED	RES, 100 ohm, 1%, 0.063W, 0402	R10	0402	CRCW0402100RFKED	1	100	<a href="#">Link</a>
MCR50JZHF1501	RES, 1.5kohm, 1%, 1W, 2010	R15	2010M	MCR50JZHF1501	1	1.5k	<a href="#">Link</a>
CRCW04024K70JNED	RES, 4.7k ohm, 5%, 0.063W, 0402	R16	0402	CRCW04024K70JNED	1	4.7k	<a href="#">Link</a>
ERJ-3GEYJ222V	[NoValue], RES, 2.2k ohm, x%, xW, [PackageReference]	R18, R21	0603	ERJ-3GEYJ222V	2	2.2k	<a href="#">Link</a>
Open		R19	0603	Resistor	1	Open	OPEN
ERJ-3GEYJ102V	[NoValue], RES, 1k ohm, x%, xW, [PackageReference]	R20, R23	0603	ERJ-3GEYJ102V	2	1k	<a href="#">Link</a>
ERJ-3GEYJ680V	RES, 68 ohm, x%, xW, [PackageReference]	R22	0603	ERJ-3GEYJ680V	1	68	<a href="#">Link</a>
ERJ-3GEYJ103V	RES, 10k ohm, x%, xW, [PackageReference]	R24	0603	ERJ-3GEYJ103V	1	10k	<a href="#">Link</a>
ERJ-3GEYJ152V	RES, 1.5k ohm, x%, xW, [PackageReference]	R25	0603	ERJ-3GEYJ152V	1	1.5k	<a href="#">Link</a>
ERJ-3GEYJ221V	RES, 220 ohm, x%, xW, [PackageReference]	R26	0603	ERJ-3GEYJ221V	1	220	<a href="#">Link</a>
ERJ-2GEJ331X	RES, 330 ohm, x%, xW, [PackageReference]	R27	0402	ERJ-2GEJ331X	1	330	<a href="#">Link</a>
MSP430AFE253	MSP430AFE253	U1	TSSOP-24	MSP430AFE253IPW	1		<a href="#">Link</a>
PS8802-1		U2, U3	SO-8	PS8802-1	2		<a href="#">Link</a>
VSK-S1-3R3U	On board block power supply	U4	VSK-S1	VSK-S1-3R3U	1		<a href="#">Link</a>
ABM3-12.000MHZ-B2-T	Crystal, 12Mhz, 18pF, SMD	Y1	XTAL_ABM3	ABM3-12.000MHZ-B2-T	1		<a href="#">Link</a>

## 7 Gerber Files

To download the Gerber files for each board, see the design files at [www.ti.com/tool/TIDM-MSP430AFE253SUBMETEREVM](http://www.ti.com/tool/TIDM-MSP430AFE253SUBMETEREVM)

## 8 Software Files

To download the software files for the reference design, see the design files at [www.ti.com/tool/TIDM-MSP430AFE253SUBMETEREVM](http://www.ti.com/tool/TIDM-MSP430AFE253SUBMETEREVM)

---

## Appendix A Example Application Code

### A.1 Example Application Code

#### A.1.1 Introduction

##### Project Structure

**emeter-autoreport.c**— Source code for auto-reporting frame assembly

**emeter-autoreport.h**— Header file for "emeter-autoreport.c" and is necessary for the "emeter-main.c" to access the auto-report function

**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-afe253.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

#### A.1.2 Preparing the Application Code

1. Select *emeter-app-afe253* project tab at the bottom of the Workspace window
2. Check the project options by right-clicking project name and selecting [Options...] from the pop-up menu ([Figure 16](#))

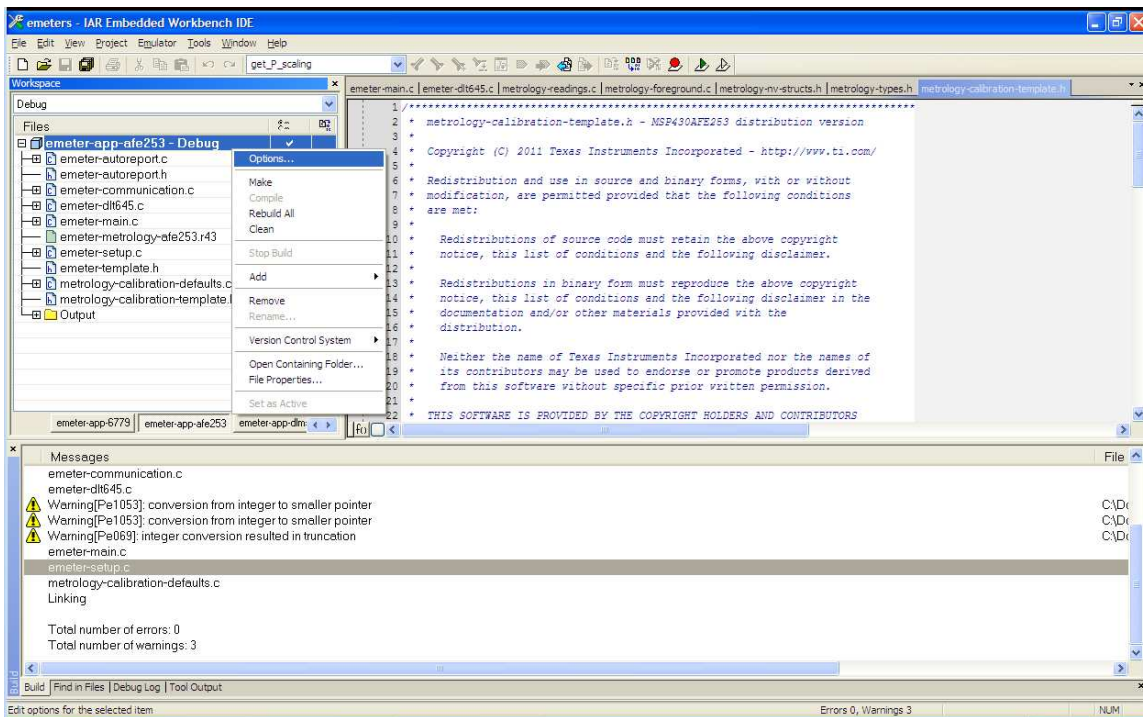


Figure 16. Project Options

- 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 17

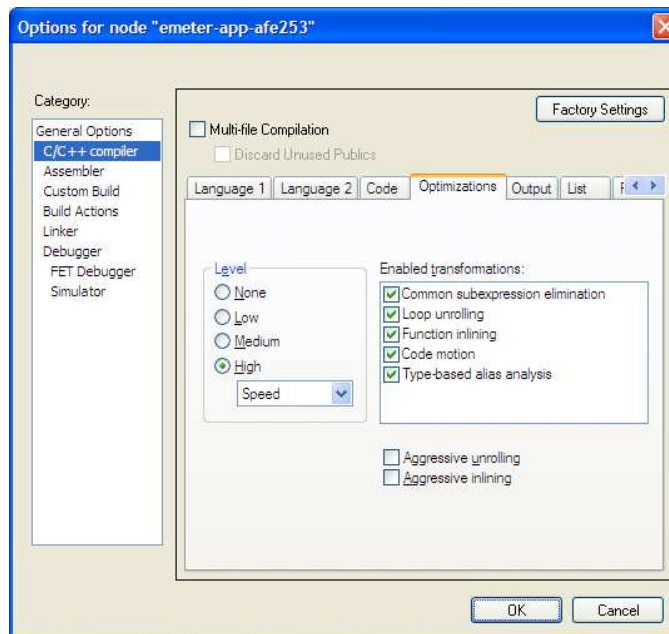
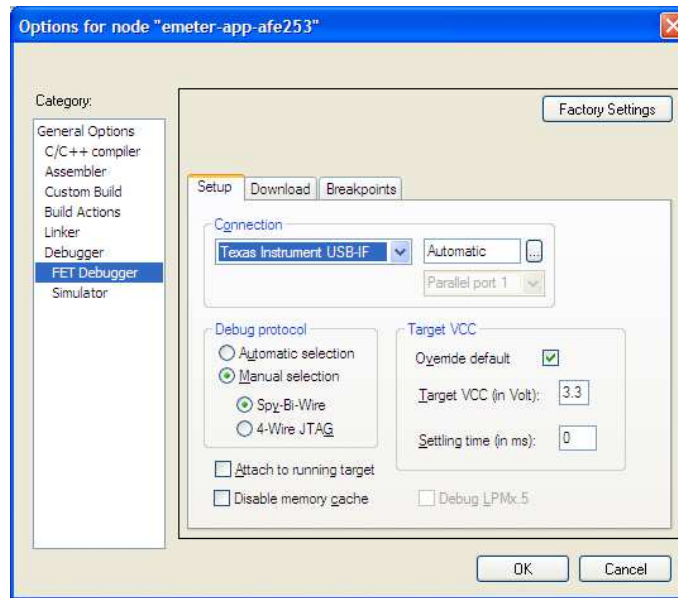
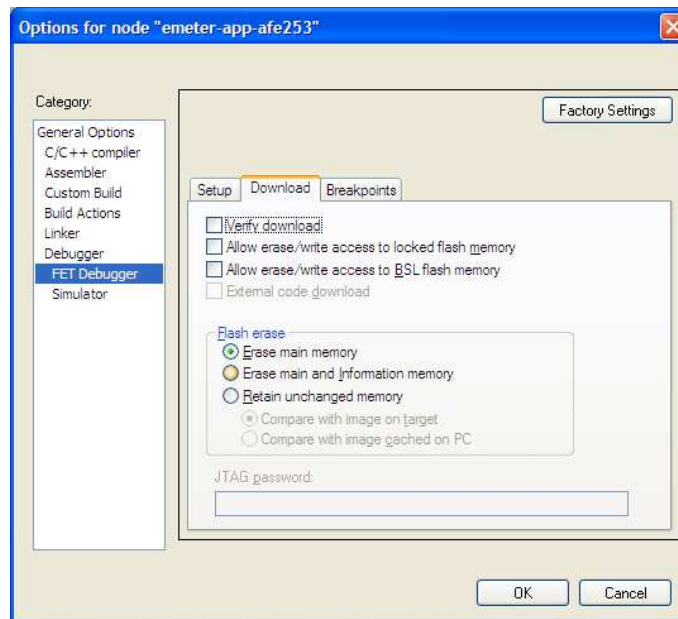


Figure 17. Optimization Options

- 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 18.


**Figure 18. Debugger Options**

5. Select the [Download] tab. If verify of the downloaded code is preferred then check the [Verify download] check box. If it is preferred to preserve the calibration values in calibration parameter memory while updating the code select [Erase main memory] radio button. If calibration values needed to be erased (e.g. the default value is changed) select [Erase main and Information memory] radio button [Figure 19](#).


**Figure 19. Download Options**

6. Click OK after all the changes made
7. Rebuilding the project by right clicking on the project and select [Rebuild All] from the pop-up menu ([Figure 20](#)). 3 warnings will be reported during rebuilding ([Figure 21](#)), it is safe to ignore the 3 warnings.

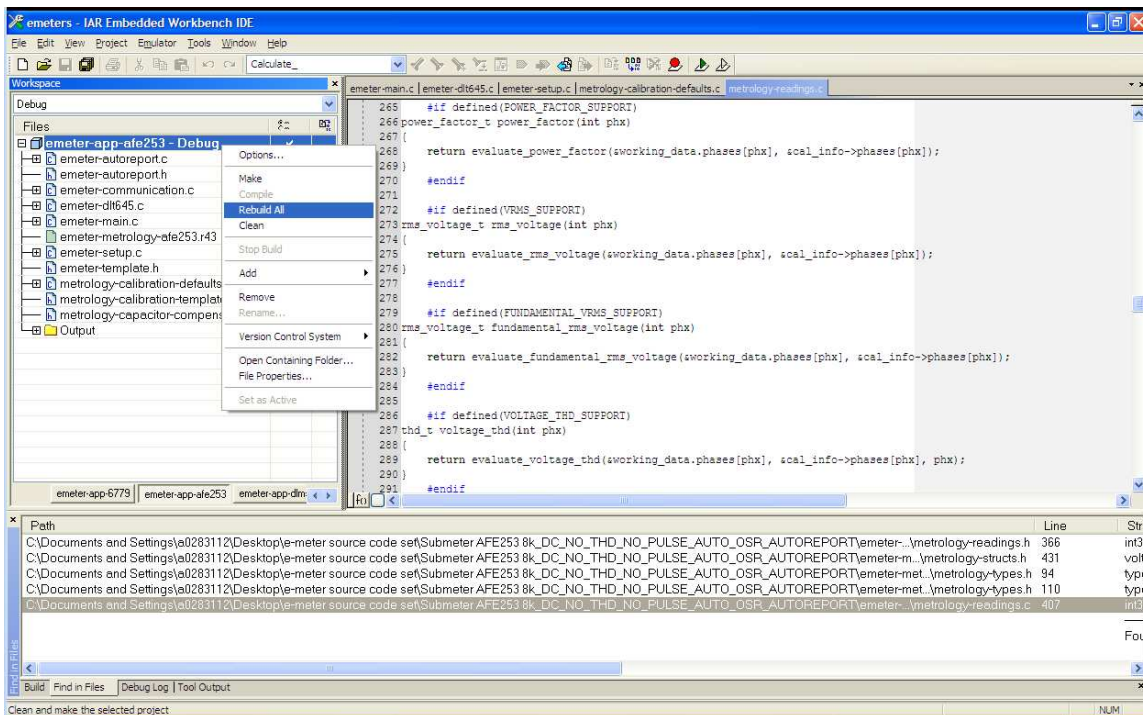


Figure 20. Compiling the Application

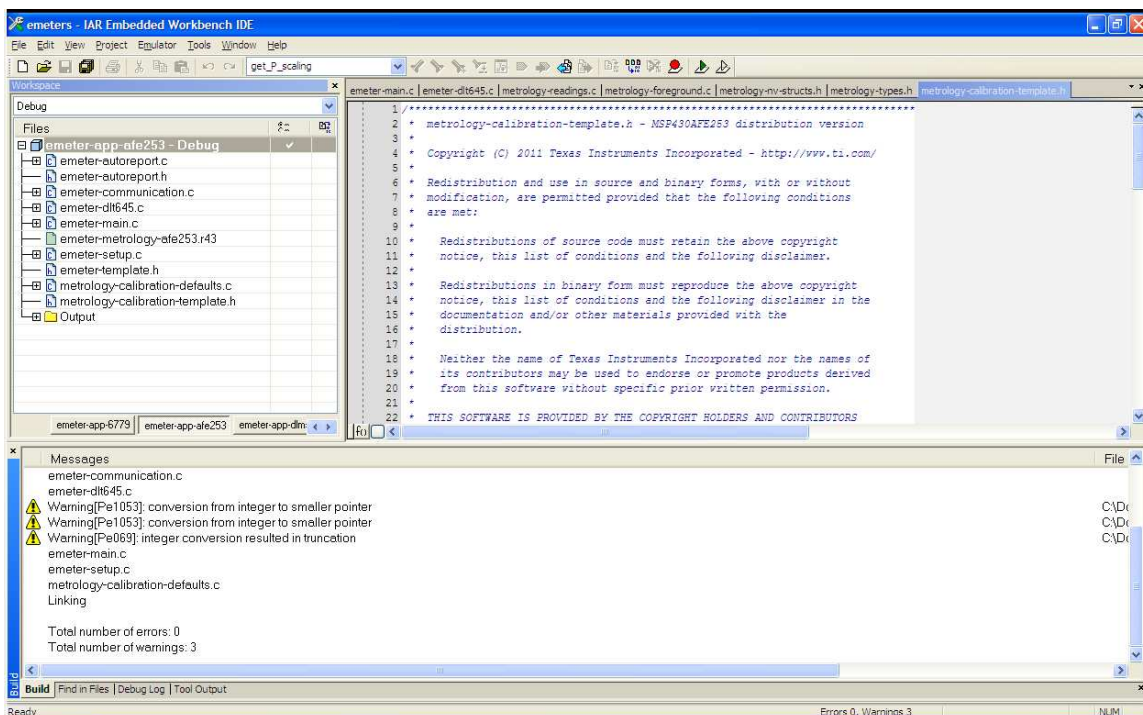


Figure 21. Warnings

8. Remove the 2 jumpers on J5, 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 22](#)

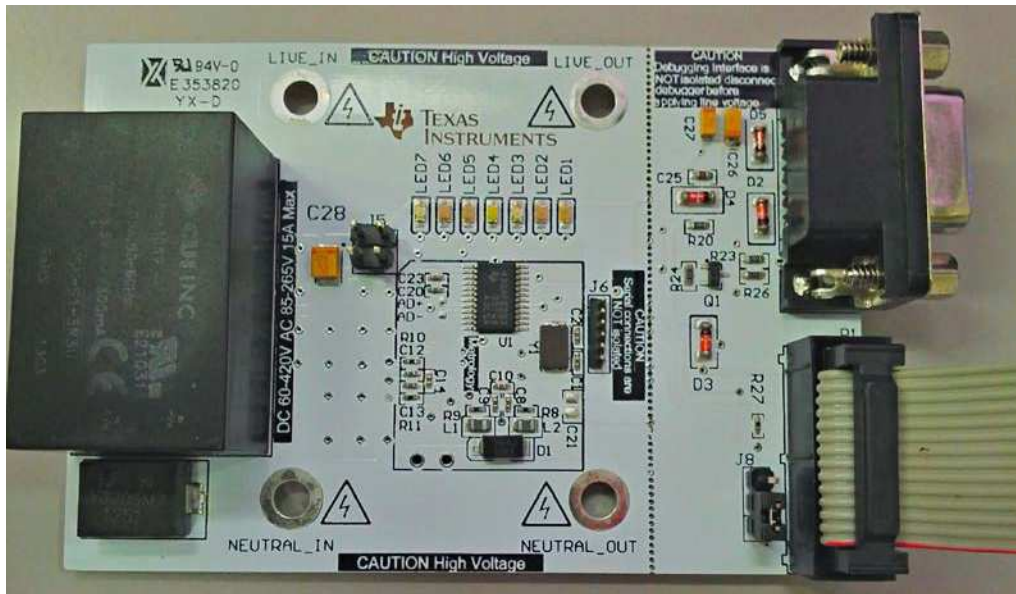


Figure 22. Connecting the EVM and FET

9. Click the [Download and Debug] button to begin download and debugging (Figure 23).

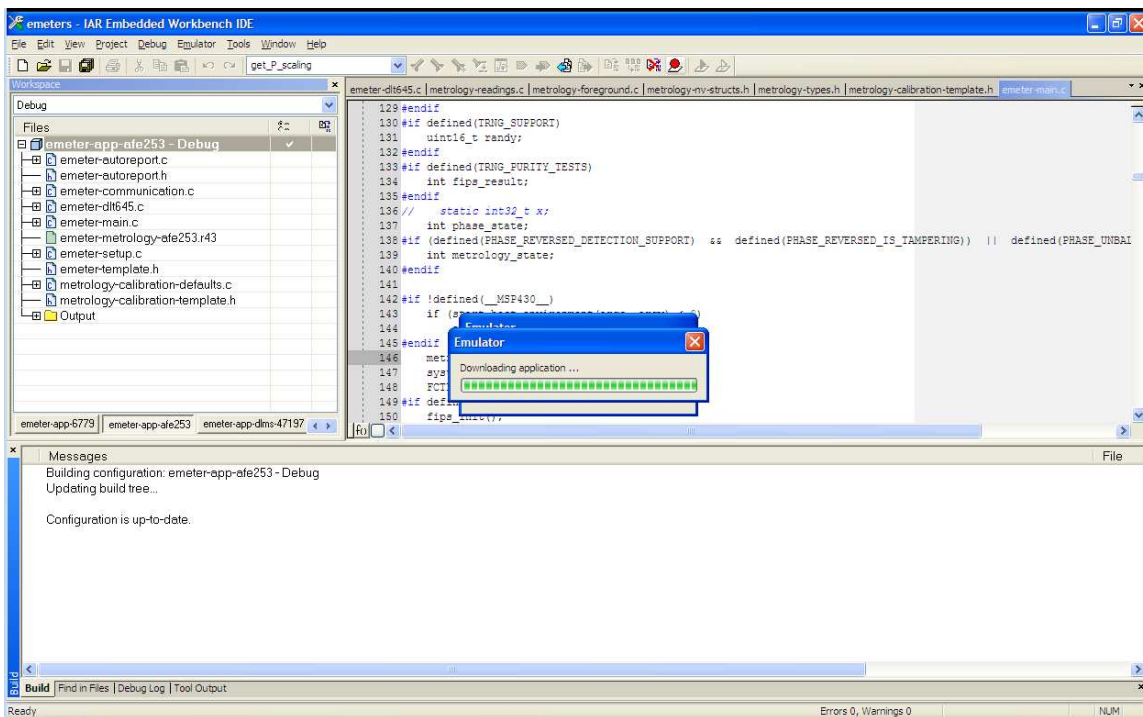


Figure 23. Code Downloading

10. After downloading is completed and succeed Figure 24 will appear. Click [Go] to start application running. (See Figure 25)



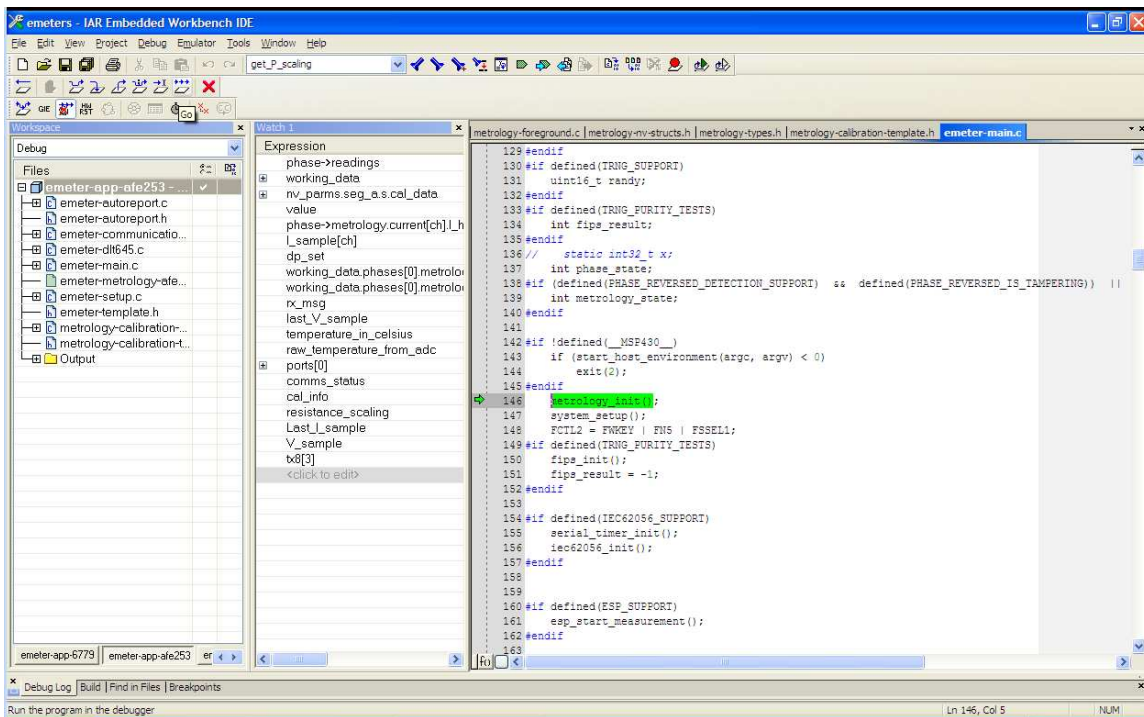


Figure 24. Debugger Screen

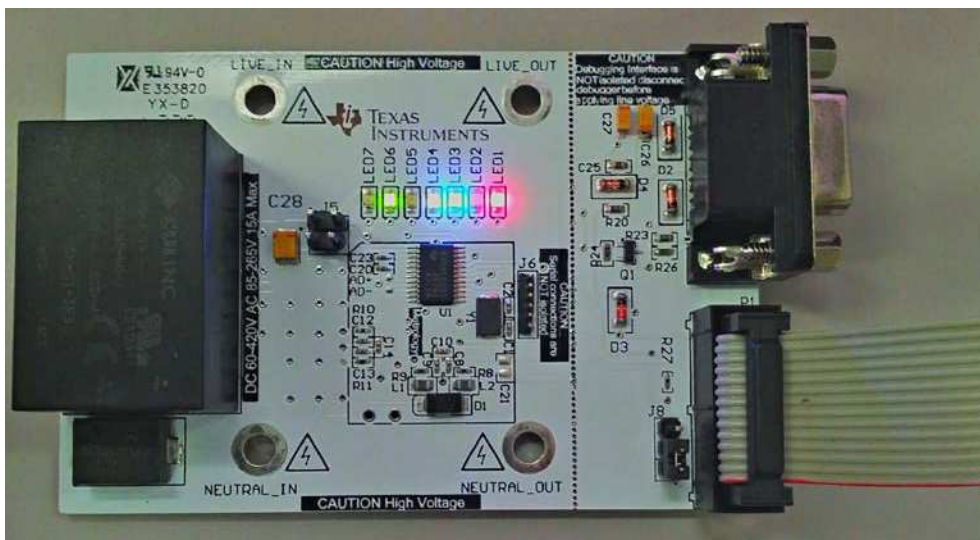


Figure 25. EVM Running

## Appendix B 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

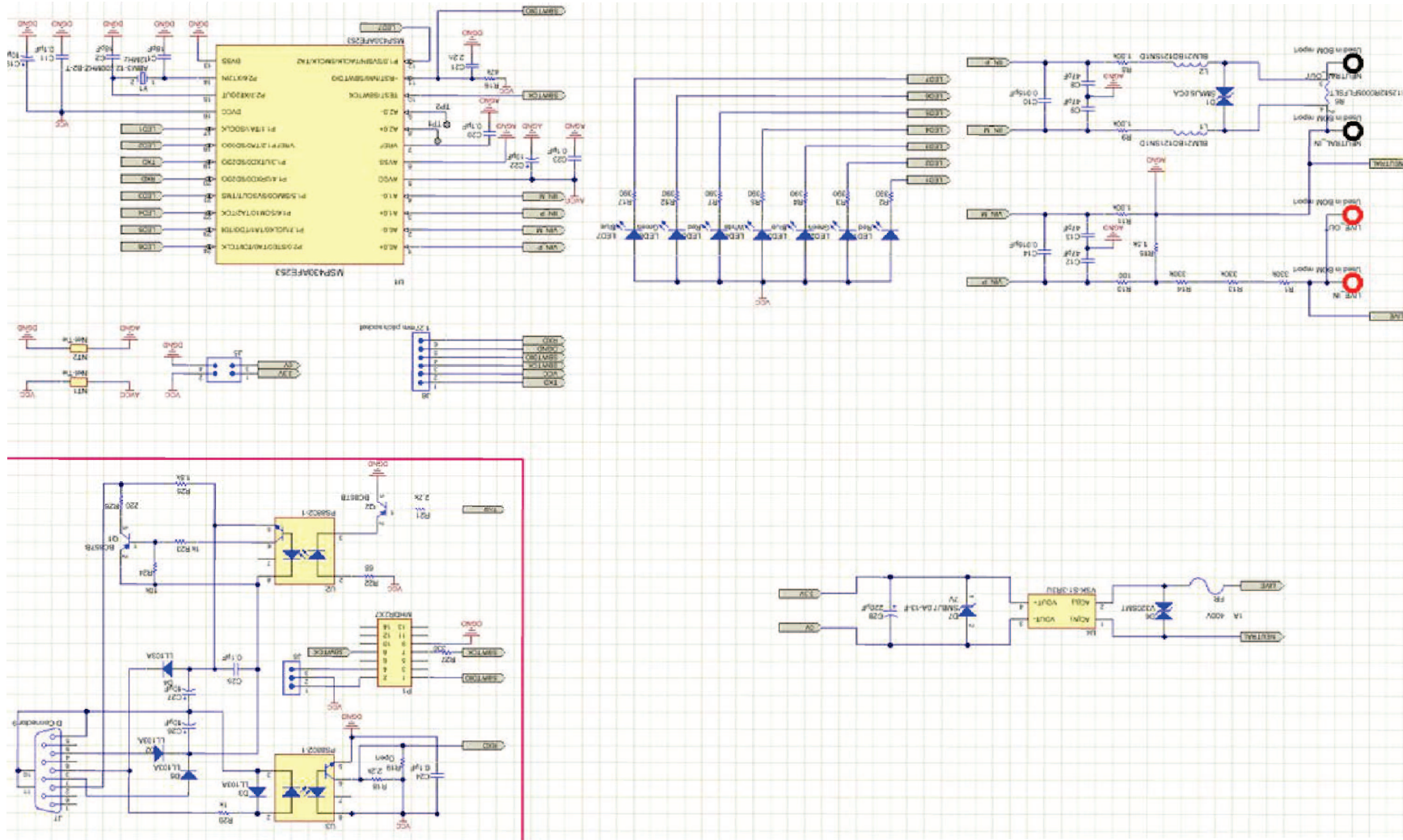


Figure 26. EVM Schematic

## Appendix C EVM Specification and Performance

### C.1 EVM Specification

- Voltage operating range (with supplied power supply) : 85 – 265VAC, 120 – 380VDC
- Sample rate : 7812 Hz
- Sampling bit depth : 24 bits
- Automatic reporting and polling report supported
- Reporting/Update rate : 4 AC cycles (AC mode), 80ms (DC mode)
- UART communication data rate : 9600 bps
- AC/DC measuring mode switching : 4 AC cycle (AC to DC), 80ms (DC to AC)
- Measurements : RMS Voltage, RMS Current, Active Power, Reactive Power, Apparent Power, Power Factor, Line Frequency, Temperature
- Measurement voltage range : 0 – 265Vrms AC, 0 – +/-380VDC (using reference design circuit and component values)
- Measurement current range : 0 – 15Arms AC, 0 – +/-22.5ADC (using reference design circuit and 0.5m-ohm shunt values)
- Voltage resolution : 1mV (Polling mode) / 10mV (auto-reporting mode)
- Current resolution : 1uA (Polling mode) / 1mA (auto-reporting mode)
- Active Power resolution : 1mW (Both mode)
- Reactive/Apparent Power resolution : 1mW (Polling mode only)
- Power factor resolution : 0.001 (Both mode)
- Frequency resolution : 0.01Hz (Polling mode only)

### C.2 EVM Performance

Power accuracy at room temperature can be found in [Table 20](#)

**Table 20. EVM Performance at PF = 1**

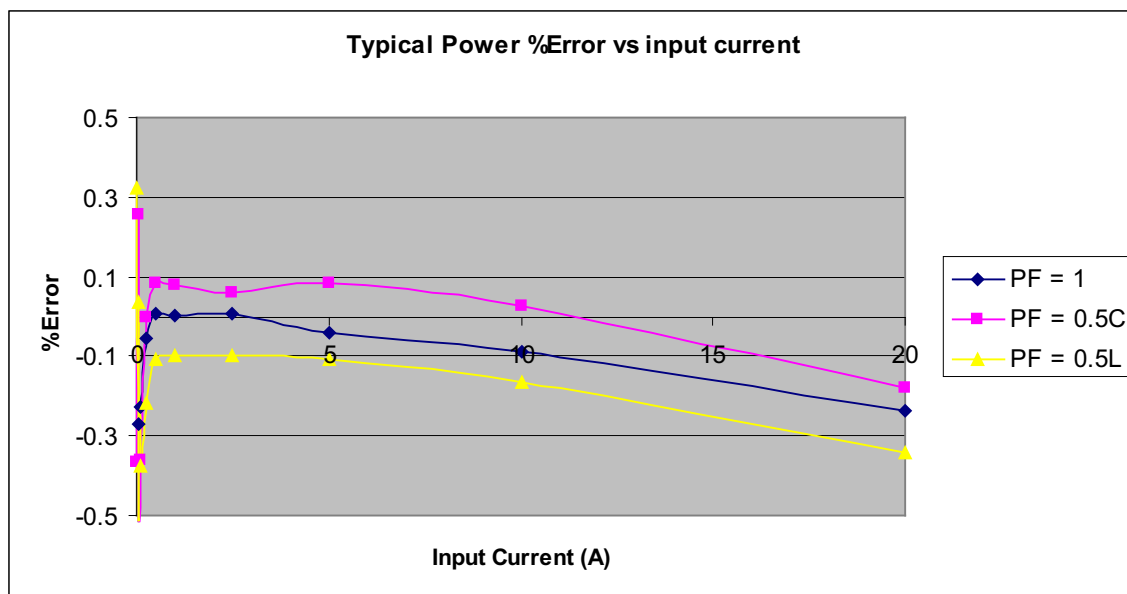
Current			Power		
AFE253	Ref	% Error	AFE253	Ref	%Error
0.011	0.010	3.981	2.164	2.220	-2.523
0.025	0.025	0.552	5.461	5.496	-0.637
0.050	0.050	-0.270	10.985	11.015	-0.271
0.100	0.100	0.000	21.950	22.000	-0.227
0.250	0.250	-0.040	54.940	54.970	-0.055
0.500	0.500	0.000	110.020	110.010	0.009
1.000	1.000	0.000	219.997	219.996	0.000
2.500	2.500	0.000	549.900	549.860	0.007
4.998	5.000	-0.040	1099.300	1099.760	-0.042
9.995	10.000	-0.050	2199.200	2201.200	-0.091
19.959	20.000	-0.205	4388.800	4399.300	-0.239

**Table 21. EVM Performance at PF = 0.5C**

Current			Power		
AFE253	Ref	% Error	AFE253	Ref	%Error
0.011	0.010	8.646	1.109	1.113	-0.368
0.025	0.025	0.840	2.757	2.750	0.255
0.050	0.050	0.160	5.470	5.509	-0.708
0.100	0.100	-0.036	10.960	11.000	-0.364
0.250	0.250	0.009	27.480	27.480	0.000
0.500	0.500	-0.044	55.060	55.014	0.084
1.000	1.000	-0.006	110.100	110.013	0.079
2.500	2.500	-0.014	274.640	274.470	0.062
5.000	5.000	-0.002	551.470	551.010	0.083
10.005	10.008	-0.026	1099.700	1099.400	0.027
19.964	19.999	-0.177	2191.000	2194.900	-0.178

**Table 22. EVM Performance at PF = 0.5L**

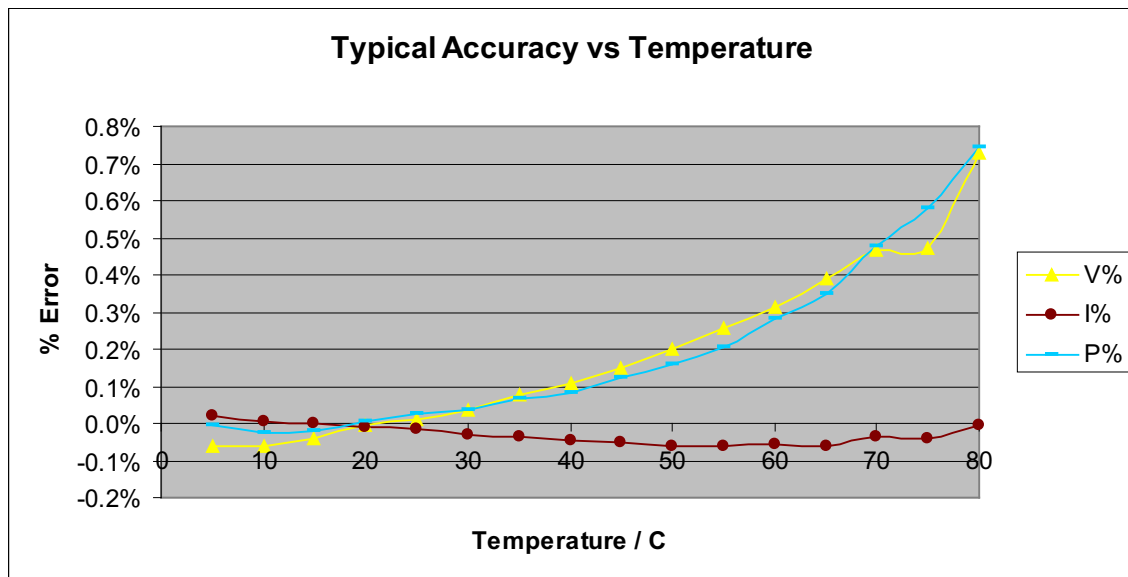
Current			Power		
AFE253	Ref	% Error	AFE253	Ref	%Error
0.011	0.010	5.951	1.115	1.111	0.324
0.025	0.025	-0.764	2.731	2.745	-0.510
0.050	0.050	-1.018	5.507	5.505	0.036
0.100	0.100	-0.036	10.930	10.971	-0.376
0.250	0.250	-0.160	27.390	27.450	-0.219
0.500	0.500	-0.041	54.890	54.950	-0.109
1.000	1.000	-0.006	109.800	109.906	-0.096
2.500	2.501	-0.021	275.200	275.469	-0.098
5.000	5.000	-0.008	549.260	549.850	-0.107
10.005	10.009	-0.035	1101.600	1103.400	-0.163
19.959	19.994	-0.175	2197.900	2205.400	-0.340



**Figure 27. Typical Power %Error vs Input Current**

**Table 23. Accuracy vs Temperature at 220V, 5A**

Temp/C	Voltage/V			Current/A			Power/ W		
	V	Vref	V%	I	Iref	I%	P	Pref	P%
5	219.820	219.961	-0.064%	5.001	5.000	0.023%	1099.440	1099.500	-0.005%
10	219.819	219.957	-0.063%	5.000	5.000	0.004%	1099.440	1099.700	-0.024%
15	219.852	219.942	-0.041%	5.000	5.000	0.000%	1099.470	1099.670	-0.018%
20	219.949	219.962	-0.006%	5.000	5.000	-0.008%	1099.790	1099.720	0.006%
25	219.995	219.972	0.010%	4.999	5.000	-0.014%	1099.890	1099.620	0.025%
30	220.020	219.941	0.036%	4.998	5.000	-0.031%	1099.950	1099.550	0.036%
35	220.133	219.964	0.077%	4.998	5.000	-0.037%	1100.250	1099.520	0.066%
40	220.170	219.931	0.109%	4.998	5.000	-0.046%	1100.550	1099.650	0.082%
45	220.286	219.960	0.148%	4.998	5.000	-0.050%	1100.900	1099.520	0.126%
50	220.394	219.955	0.200%	4.997	5.000	-0.062%	1101.350	1099.630	0.156%
55	220.525	219.957	0.258%	4.997	5.000	-0.059%	1101.850	1099.610	0.204%
60	220.664	219.972	0.315%	4.997	5.000	-0.057%	1102.760	1099.640	0.284%
65	220.787	219.934	0.388%	4.997	5.000	-0.062%	1103.390	1099.560	0.348%
70	220.993	219.966	0.467%	4.998	5.000	-0.037%	1104.850	1099.630	0.475%
75	221.000	219.962	0.472%	4.998	5.000	-0.040%	1105.950	1099.570	0.580%
80	221.550	219.950	0.727%	5.000	5.000	-0.007%	1107.610	1099.420	0.745%



**Figure 28. Typical Accuracy vs Temperature**

## Appendix D Auto-Reporting CRC Calculation Source Code

### D.1 Auto-Reporting CRC Calculation Source Code

```

const unsigned int CRC_Table[]=
{
    0x00,0x07,0x0E,0x09,0x1C,0x1B,0x12,0x15,
    0x38,0x3F,0x36,0x31,0x24,0x23,0x2A,0x2D,
    0x70,0x77,0x7E,0x79,0x6C,0x6B,0x62,0x65,
    0x48,0x4F,0x46,0x41,0x54,0x53,0x5A,0x5D,
    0xE0,0xE7,0xEE,0xE9,0xFC,0xFB,0xF2,0xF5,
    0xD8,0xDF,0xD6,0xD1,0xC4,0xC3,0xCA,0xCD,
    0x90,0x97,0x9E,0x99,0x8C,0x8B,0x82,0x85,
    0xA8,0xAF,0xA6,0xA1,0xB4,0xB3,0xBA,0xBD,
    0xC7,0xC0,0xC9,0xCE,0xDB,0xDC,0xD5,0xD2,
    0xFF,0xF8,0xF1,0xF6,0xE3,0xE4,0xED,0xEA,
    0xB7,0xB0,0xB9,0xBE,0xAB,0xAC,0xA5,0xA2,
    0x8F,0x88,0x81,0x86,0x93,0x94,0x9D,0x9A,
    0x27,0x20,0x29,0x2E,0x3B,0x3C,0x35,0x32,
    0x1F,0x18,0x11,0x16,0x03,0x04,0x0D,0x0A,
    0x57,0x50,0x59,0x5E,0x4B,0x4C,0x45,0x42,
    0x6F,0x68,0x61,0x66,0x73,0x74,0x7D,0x7A,
    0x89,0x8E,0x87,0x80,0x95,0x92,0x9B,0x9C,
    0xB1,0xB6,0xBF,0xB8,0xAD,0xAA,0xA3,0xA4,
    0xF9,0xFE,0xF7,0xF0,0xE5,0xE2,0xEB,0xEC,
    0xC1,0xC6,0xCF,0xC8,0xDD,0xDA,0xD3,0xD4,
    0x69,0x6E,0x67,0x60,0x75,0x72,0x7B,0x7C,
    0x51,0x56,0x5F,0x58,0x4D,0x4A,0x43,0x44,
    0x19,0x1E,0x17,0x10,0x05,0x02,0x0B,0x0C,
    0x21,0x26,0x2F,0x28,0x3D,0x3A,0x33,0x34,
    0x4E,0x49,0x40,0x47,0x52,0x55,0x5C,0x5B,
    0x76,0x71,0x78,0x7F,0x6A,0x6D,0x64,0x63,
    0x3E,0x39,0x30,0x37,0x22,0x25,0x2C,0x2B,
    0x06,0x01,0x08,0x0F,0x1A,0x1D,0x14,0x13,
    0xAE,0xA9,0xA0,0xA7,0xB2,0xB5,0xBC,0xBB,
    0x96,0x91,0x98,0x9F,0x8A,0x8D,0x84,0x83,
    0xDE,0xD9,0xD0,0xD7,0xC2,0xC5,0xCC,0xCB,
    0xE6,0xE1,0xE8,0xEF,0xFA,0xFD,0xF4,0xF3
};

static inline uint8_t FW_CRC (uint8_t temp_buf, uint8_t LastCRC)
{
    LastCRC = CRC_Table[LastCRC ^ temp_buf];

    return LastCRC;
}

```

### D.2 CRC Calculation Usage Example

```

{
    // frame is prepare here
    // ...
    ...
    // prepare crc after all bytes in the frame is ready
    Frame [FRAME_SIZE-1] = 0x00;
    for (ByteCount = 0 ; ByteCount < FRAME_SIZE-1 ; ByteCount++)
    {
        Frame [FRAME_SIZE-1] = FW_CRC(Frame [ByteCount], Frame [FRAME_SIZE-1]) ;
    }
    // ...
    ...
}

```

## IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Texas Instruments Incorporated ("TI") reference designs are solely intended to assist designers ("Buyers") who are developing systems that incorporate TI semiconductor products (also referred to herein as "components"). Buyer understands and agrees that Buyer remains responsible for using its independent analysis, evaluation and judgment in designing Buyer's systems and products.

TI reference designs have been created using standard laboratory conditions and engineering practices. **TI has not conducted any testing other than that specifically described in the published documentation for a particular reference design.** TI may make corrections, enhancements, improvements and other changes to its reference designs.

Buyers are authorized to use TI reference designs with the TI component(s) identified in each particular reference design and to modify the reference design in the development of their end products. HOWEVER, NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, IS GRANTED HEREIN, including but not limited to 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.

TI REFERENCE DESIGNS ARE PROVIDED "AS IS". TI MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. TI DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUIET ENJOYMENT, QUIET POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO TI REFERENCE DESIGNS OR USE THEREOF. TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY BUYERS AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON A COMBINATION OF COMPONENTS PROVIDED IN A TI REFERENCE DESIGN. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF TI REFERENCE DESIGNS OR BUYER'S USE OF TI REFERENCE DESIGNS.

TI reserves 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 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 for TI components 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.

Reproduction of significant portions of TI information in TI data books, data sheets or reference designs 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.

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 that anticipate dangerous failures, monitor failures and their consequences, lessen the likelihood of dangerous failures 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 Buyer's 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 an agreement specifically governing such use.

Only those TI components that 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 that have **not** been so designated is solely at Buyer's risk, and 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.