Application Report

SLAA638–August 2014

Single Phase and DC Embedded Metering (Power Monitor) Using MSP430I2040

Mars Leung

ABSTRACT

This application report discusses an evaluation module (EVM) design utilizing the MSP430i2040 in the application of embedded metering (sub-metering).

Contents

List of Figures

List of Tables

1 Introduction

The Texas instruments MSP430i2040 is an ultra-low-power mixed signal microcontrollers. It integrates with four independent differential input 24-bit sigma-delta analog-to-digital converters (ADC) with a programmable gain amplifier, a 16-bit hardware multiplier, an eUSCI_A0 supports universal asynchronous receiver/transmitter (UART), serial peripheral interface (SPI) communication interface, an eUSCI_B0 supports SPI, inter-integrated circuit (I2C), two 16-bit timers and 12 general-purpose input/output 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 embedded electricity power measurement. In such application space, the electricity measuring device is embedded in the end application, which provides the device information about the voltage, current, power consumption, and so forth. Moreover, the EVM is enabled with the capability to compensate the line resistance and EMI filter capacitance. Embedded metering (sub-metering) is applicable in many areas such as:

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

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

1.1 Safety and Precautions

The design discussed in this document should be operated by professionals who have received appropriate technical training.

WARNING

Possibility of electric shock can occur when the board is connected to live wire. The board should be handled with care by professionals. For safety, use of isolated equipment with overvoltage and overcurrent protection is highly recommended.

1.2 Features

The software features are:

- Measurement of root mean square voltage, root mean square current, active power, reactive power, apparent power, power factor, AC frequency, voltage THD, current THD, fundamental voltage, fundamental current, fundamental active power
- Readings update every 4 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 doing EMI filter capacitor and wire resistance compensation
- No separate DC calibration required

IAR Embedded Workbench is a registered trademark of IAR Systems AB.
Microsoft, Windows are registered trademarks of Microsoft Corporation in the United States and/or other countries, or both.

2 Design Details

2.1 Block Diagram

Figure 1 shows the system block diagram of the design. Power, from the AC mains, passes through a shunt resistor. The voltage across the shunt resistor is tapped out. The AC mains voltage is divided to a smaller fraction by the resistor ladder and tapped out. The two signals are then passed to the two ADCs on the MCU for further processing. The two SD24 ADCs operate in a synchronous manner; the hardware allows the sampling times to be offset in finely controllable steps. This feature is used to correct any phase shift between the voltage and current signal, resulting from the external analog circuit.

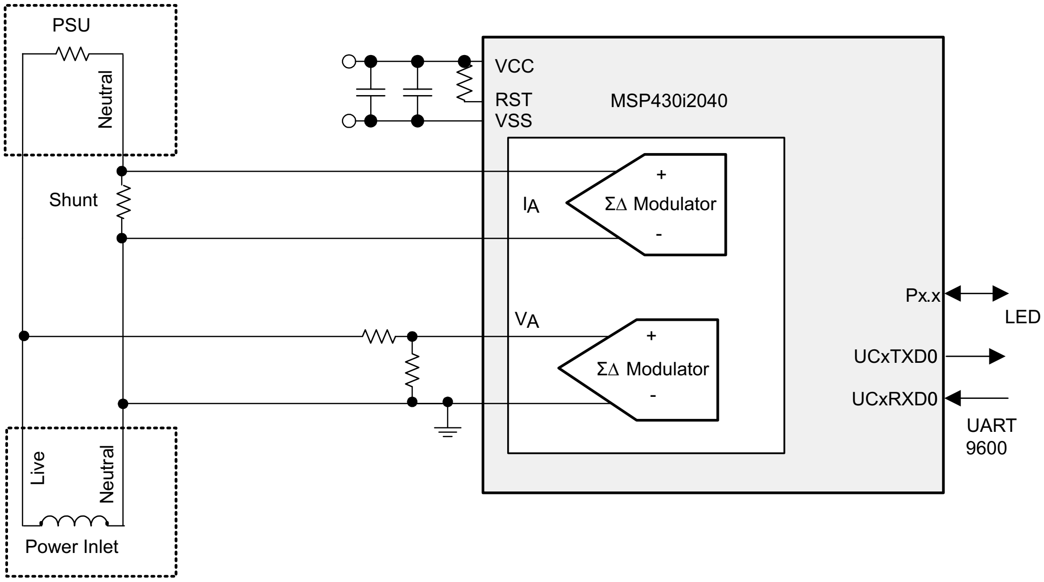


Figure 1. System Block Diagram

2.2 Hardware Design

2.2.1 Interface Circuit

In the design of a metering circuit, a key success factor is to yield sufficient accuracy and is dependant on a good sampling circuit. The 24 bits Sigma-delta AD Converter (SD24) on the MSP430i2040, together with the on-chip programmable gain amplifier provides the critical hardware for getting high quality AD conversion results from both current and voltage sensor. Moreover, the interface circuit, the component selection and the circuit layout also plays a crucial role for a successful design.

Due to the nature of a sigma-delta AD converter and of the voltage and current signal being measured, an external interface circuit is needed to interface the voltage and current to keep the SD24 ADC to work properly and provide proper filtering for band of interest.

The hardware of the interface circuit is simple, which is composed of passive components apart from the protection diode (D1, D8). Figure 2 shows the interface circuit designed. The top circuit is the interface to voltage sensor and the bottom circuit is for the current sensor – the shunt resistor.

In the current sensor interface circuit, R6 being the shunt sensors, D1 is the optional protection diode. The protection diode is not required unless, for example, a high shunt resistor value or a current transformer is in use instead of the current sensor, and the user anticipates a significant current surge to cause enough voltage across the sensor to damage the MCU.

L1, R9 and C9 forms a low-pass filter having bandwidth of a few MHz. The purpose of this filter is to reject radio frequency interference from going into the sigma-delta ADC. The filter formed by L2, R8 and C8 has the same characteristic which is balanced for the differential signal.

The filter formed by R9, R8 and C10 is a filter that gives the bandwidth of about 10 kHz, which is the filter for the band of interest.

It is important to note that the order of magnitude of these capacitors should be observed; even the actual value of the capacitance is not that critical.

The voltage sensor interface is a resistor ladder. The ratio of the resistance in the resistor ladder allows up to AC 308 V and DC 420 V to be measured. It is very similar to the current sensor filter except that the value of R10 is 100 Ω not 1 kΩ. The reason of this is the voltage divider circuit has an equivalent source resistance $1.5k \times \frac{990k}{(1.5k + 990k)} \approx 1.5k$ in series with R10. In order to have the filter balanced for the differential signal, a small R10 value is used. The combine effect is then of similar value to R11. Though there is a slight mismatch using the designed value but there is no observable impact to the accuracy.

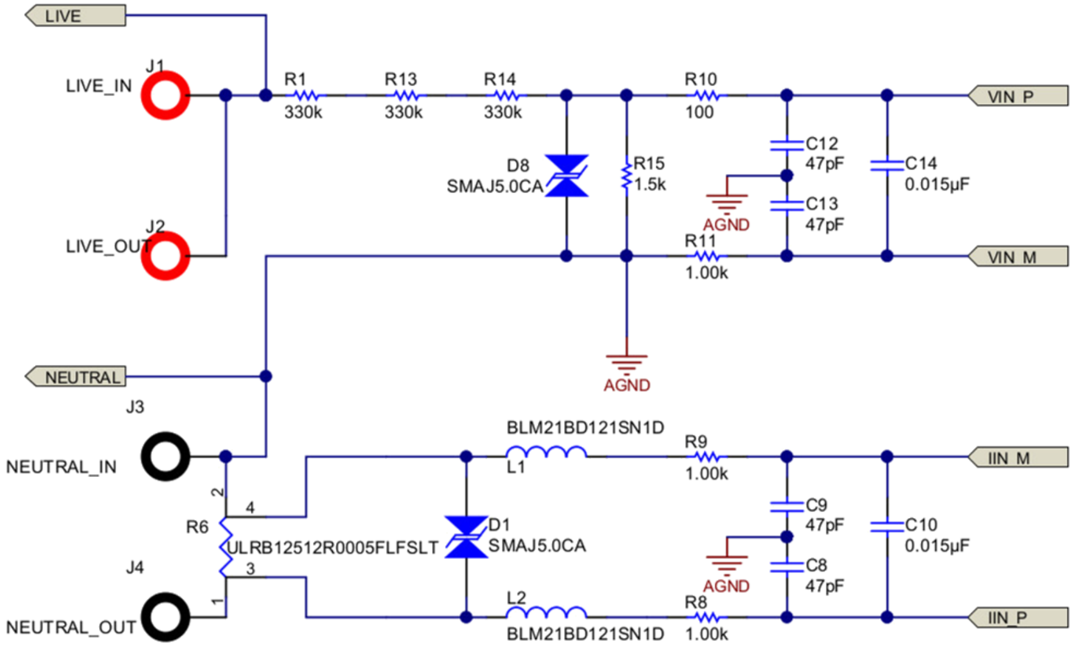


Figure 2. Voltage and Current Sensor Interface Circuit

2.2.2 Shunt Resistor

For an accurate measurement, the choice of shunt is another important factor. In general, it is suggested that a shunt of smaller value is a better choice. Smaller shunt value benefits the power dissipated on the shunt and the respective rise in temperature. The lower rise in temperature reduces the need for a very low temperature coefficient shunt resistor to sustain the accuracy over temperature. However, there is also drawback of using a small value shunt that the signal to noise ratio of a small value shunt is worse than using a larger value one. In this sense, there may be chance that a shunt of higher value is desired, for example when measuring a very small current and the range of current to be measured is also in a small range. In this design, a few factors are considered: the maximum current, the current dynamic range, the power dissipation.

In this application, the maximum current for each socket is 15A, the desired dynamic range of current keeping flat error percentage is 1000:1. It is preferred that the power dissipated on the shunt resistor be as small as possible.

To achieve the best accuracy, it is desired to have the current signal present to the SD24 ADC have maximum swing when measuring maximum current, since the SD24 ADC has an input range of approximately 900 mV at x1 gain. Using x16 gain, the input range for the AC root mean square is:

$$\frac{\left(\frac{900}{16}\right)}{\sqrt{2}} = 39.77\ mV_{rms} \tag{1}$$

With this number, 2mΩ shunt resistance should be good for this input range. Yet due to the limited area of copper attached to the shunt resistor, it is better to use a smaller valued shunt to keep the heat accumulation on the shunt resistor low; therefore, 0.5 mΩ shunt resistance is chosen.

In addition to value, the physical size of the shunt is also an important factor for accuracy. Not the size directly but the heat generated when current is flowing. A smaller sized shunt heats up more easily due to the limited surface area. If there is significant current to flow though the shunt, it is suggested to have a large enough sized shunt and there should also be sufficient ventilation on the PCB to prevent heat being accumulated. In this design, a 2512 size shunt is chosen. At 0.5 mΩ, the power dissipate on the shunt at maximum current is $16^2 \times 0.5 = 0.128$W, which is about 1/8 the rated power of a 2512 size 1W shunt resistor.

2.2.3 Voltage Divider Circuit

For the voltage sensor circuit, the matter is a little simpler. In this design, the major consideration is that the voltage divider should be able to prevent arcing between the terminals. To reduce the voltage difference between resistor terminals, this design uses 4 (instead of 2) resistors to form the voltage divider (R1, R13, R14 are 330 kΩ, and R15 is 1.5 kΩ), then the voltage across R1, R13 and R14 each takes about 1/3 of the AC voltage. Now at 220 V input, each resistor has a voltage drop about 75 V. Using 2512 resistor for R1, R13, R14 then provides greater equivalent distance of separation between the live voltage and R15.

The installation of D8 in the voltage sensor circuit is more important than D1 in the current circuit. The reason is that the resistor ladder on the voltage sensor circuit is more likely to see a voltage surge across R15 high enough to damage the MCU. In other cases, if R15 is an open circuit or in bad contact causing a higher resistance, the voltage across the input of the voltage channel ADC also causes damage to the MCU.

2.2.4 Layout Consideration

Though the circuit design of the external circuit is very simple, there are a few points to note in the PCB layout and component placement to achieve best accuracy. Figure 3 and Figure 4 shows the layout of top and bottom layer, respectively. The principle to keep in mind for the layout and placement is simple:

- The critical circuit are the two sets of interface filter circuit.
- The signal from the sensor to the input of the SD24 is differential no matter what.
- Separate the analog and digital ground, connect back together at power supply ground.
- Have ground plane around the analog signal as possible to protect against interference.

Now going back to the layout, note that inside the green rectangle on Figure 3 are the interface circuit for the voltage sensor and the shunt senor. The placement of this circuit (compose of R10, R11, C12, C13, C14 for voltage sensor, R8, R9, C8, C9, C10) should be arranged to place similar to the way it is drawn on the schematic. Doing this helps to protect the differential nature of the signal while passing through the interface filter circuit.

Inside the yellow rectangle on Figure 3 and Figure 4 shows that the differential signal pair leaving the interface filter should be closely side-by-side and surrounded by the analog ground plane on their side (of course, it is not always possible).

Although R15, R6, L1 has one terminal essentially connected to AGND, it is important to treat each as a separate signal connecting to AGND with a single point (Cyan circled in Figure 4 shows the connection of R15 to AGND just with a single track). In fact, this way of layout also applies to the connection to C12, C13, C8, C9. This helps the circuit to stay independent (neither the voltage signal nor the current signal has a path interfering the other) and free from the noise flowing in the ground plane.

The ground plane is designed such that it surrounds most of the analog circuit. Doing this with i2040 is easy since the analog signal pins are grouped together on a side of the package. Moreover, many via are added on the ground plane so that the top and bottom ground plane are closely tied together and provides stronger noise protection.

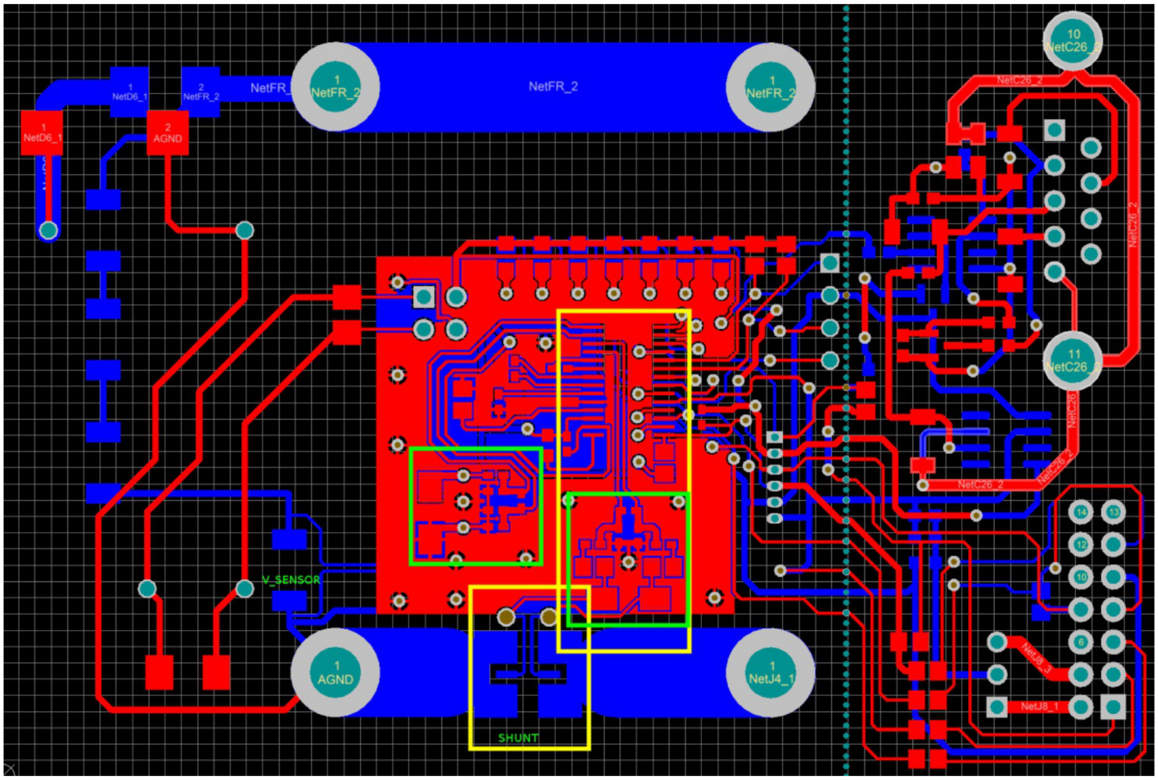


Figure 3. Top Layout

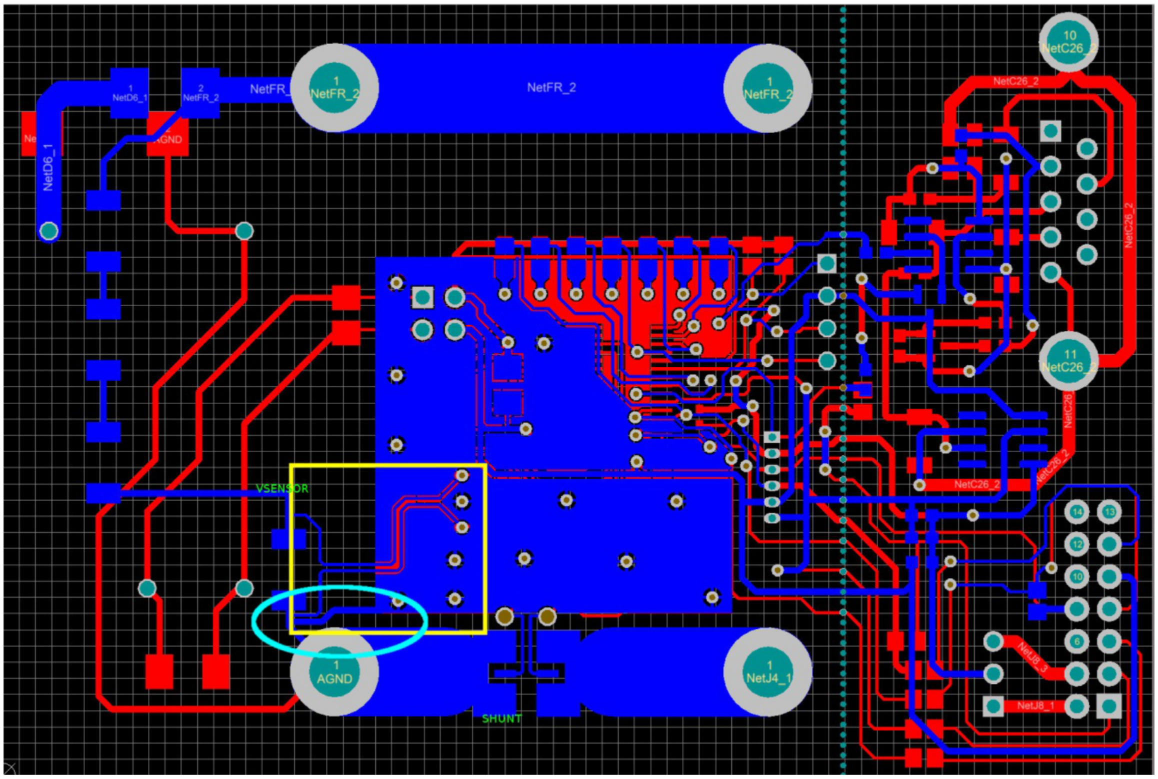


Figure 4. Bottom Layout

2.2.5 Shunt Sensor Pad Design

The design of the pad for the shunt sensor is designed around two principals:

- Large pad helps to dissipate the heat created on the shunt sensor
- The tapping point is free from current flow

In Figure 3, the shunt senor is labeled and is in a yellow rectangle. With the shape designed as shown, the current flowing path is forced onto the external “[” “]” shape. The tapping pad (the small pad inside the “[ ]” shape) is having a minimal contact to the shunt and is expected to have no current flowing between this point. This helps tap the shunt voltage the effect of solder and copper eliminated.

2.3 Software Design

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

The firmware is partitioned into three main blocks.

- The application contains:
  - System setup 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-i2041.r43” contains:

  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
- The toolkit packaged into a library named “emeter-toolkit-i2041.r43”, contains low-level computation routines.

2.3.1 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 DC offset removal (DC mode)
  – 16-bit high-pass filtering 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 multiply 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

2.3.2 Metrology Computation Engine

Three Sigma-Delta ADC on the MCU are used in this design: one for voltage measurement, one for current measurement, and the other one connects to the built-in temperature sensor for temperature measurement. The Sigma-Delta ADC is running at approximately 1.024 MHz modulator frequency fixed by the hardware, the sample frequency is derived from the modulator frequency divided by 128 equals 8 kHz that give us Nyquist bandwidth of 4 kHz. This bandwidth is sufficient to cover the 66th harmonic for 60Hz AC, 80th harmonic for 50Hz AC frequency as wider bandwidth is usually needed for server power monitoring due to the nature of the switching power supplies being monitored.

The background processing performs its actions in a time critical loop, which is triggered by the ADC at the sample rate. It is running in the interrupt services routine of the ADC and is processed automatically. It performs the actual sample computation based on the information collected from the voltage and current ADC channels. Hand over the less time critical process to foreground processing.

The foreground processing is triggered by the completion of background processing at the reporting and 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:

$$V_{RMS} = VGAIN \times \sqrt{\frac{1}{N} \sum_{i=1}^{N} V_{samp}(i) \times V_{samp}(i)} \tag{2}$$

$$I_{RMS} = IGAIN \times \sqrt{\frac{1}{N} \sum_{i=1}^{N} I_{samp}(i) \times I_{samp}(i)} \tag{3}$$

$$P_{active} = PGAIN \times \frac{1}{N} \sum_{i=1}^{N} V_{samp}(i) \times I_{samp}(i) \tag{4}$$

$$P_{reactive} = PGAIN \times \frac{1}{N} \sum_{i=1}^{N} V_{samp,90}(i) \times I_{samp}(i) \tag{5}$$

$$P_{apparent} = V_{RMS} \times I_{RMS} \tag{6}$$

$$PF = \cos\varphi = \frac{P_{active}}{P_{apparent}} \tag{7}$$

$$V_{RMS\,fund} = VGAIN \times \sqrt{\frac{1}{N} \sum_{i}^{N} V_{samp}(i) \times V_{pure}(i)} \tag{8}$$

$$P_{active\,fund} = PGAIN \times \frac{1}{N} \sum_{1}^{N} I_{samp}(i) \times V_{pure}(i) \tag{9}$$

$$P_{reactive\,fund} = PGAIN \times \frac{1}{N} \sum_{1}^{N} I_{samp}(i) \times V_{pure(\pi/2)}(i) \tag{10}$$

$$V_{THD} = \frac{\sqrt{V_{RMS}^2 - V_{RMS\,fund}^2}}{V_{RMS\,fund}} \tag{11}$$

$$I_{RMS\,fund} = \frac{\sqrt{P_{active\,fund}^2 + P_{reactive\,fund}^2}}{V_{RMS\,fund}} \tag{12}$$

$$I_{THD} = \frac{\sqrt{I_{RMS}^2 - I_{RMS\,fund}^2}}{I_{RMS\,fund}} \tag{13}$$

2.3.3 Background Process

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

- Capture data from voltage and current ADC channels
- Perform voltage sample processing
- Current sample processing
- Power processing
- Line frequency processing
- Trigger foreground process

Figure 5 shows the system functional diagram, the two SD24 ADCs take samples from the voltage and the current channel with the (calibrated and programmed) fractional delay. The samples are then passed through a DC removal filter. Each DC removal filter dynamically tracks the DC offset of the ADC and the signal by analyzing each sample and removes the DC (from signal and from ADC DC offset) before passing the sample to further processing. If the operation is in DC mode, the DC removal filter only removes the DC offset of the ADC.

The current sample and the voltage sample then performs a square and accumulates to get the root mean square current (I_{rms}) and voltage (V_{rms}). The current sample also multiplies and accumulates with the voltage sample with proper additional phase delay (if needed) to get the active power (P_{active}) and with the corresponding 90° shifted sample to get the reactive power ($P_{reactive}$).

An internally generated pure sine wave amplitude and phase synchronized with the AC voltage input together with the 90° shifted version are used to multiply and accumulate with the current sample to get the fundamental active and fundamental reactive power.

The cycle detection mechanism detects the presence of an AC signal by checking the number of zero crossing over a pre-defined time. If the number of zero crossing is less than the expected number of zero crossing minus 1, then it is determined to be DC mode otherwise it is in AC mode. This works well for the frequency from 25Hz when the pre-defined determination interval is set to 80 ms.

After 4 AC cycles (or 80 ms in DC mode), the system sets a flag showing that there are results that are needed to be processed. The system is then triggered by this flag to perform the scaling and convert those accumulated values into values of proper units (V, A, W, and so forth).

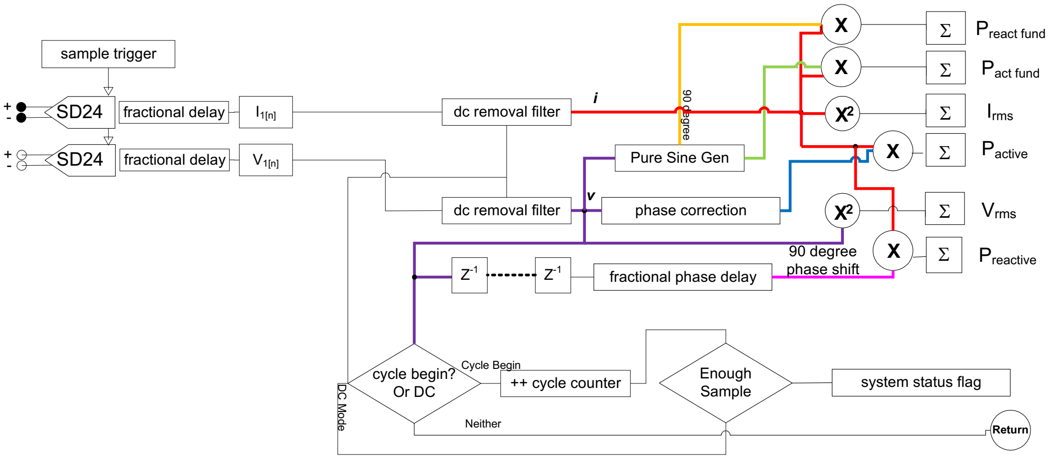


Figure 5. System Functional Diagram

2.3.4 Phase Correction

The sensors and the input circuit's passive components introduce a small but significant enough phase difference between the current and voltage signals. This phase mismatch varies from unit to unit, due to component tolerances. This needs to be corrected to achieve high accuracy, as even small phase errors can spoil the power measurement accuracy at low power factors. For example, if the power factor is 0.5, the phase error needed to produce a 0.1% error in the active power measurement is only 0.034°.

The SD24 has a built-in circuit to provide a fractional offset between the timing of the voltage and current samples, in 1/128th sample increments, when the OSR is 128. This is equivalent to 0.034° steps at 50Hz. The phase correction can be adjusted at any time by loading a value into one of the SD24PREx registers. This causes the following conversion, and only that one conversion, to be stretched by the 8 bit amount written to the register.

Figure 6 illustrates how the operation of phase correction is done.

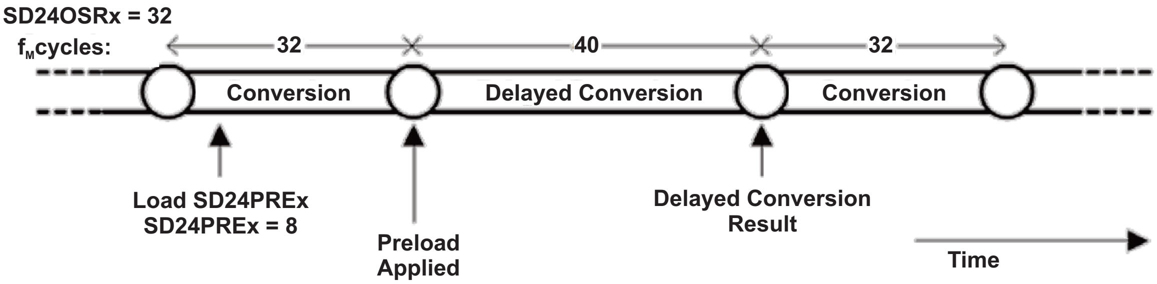


Figure 6. Phase Correction With SD24PRE Register

2.3.5 Frequency Measurement and Cycle Tracking

For frequency measurement, the integer number of samples per cycles is taken and filtered over many cycles to produce a mains period value with high resolution. However, this can take quite a long time to settle. To improve the settling time, a straight line interpolation between the voltage samples each side of a zero crossing is taken. To make this more robust in the presence of spike on the supply rail, unreasonable values from the estimation is ignored. Figure 7 illustrates this behavior.

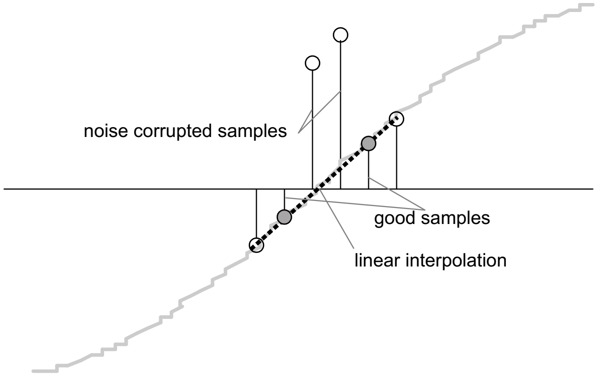


Figure 7. Frequency Measurement and Cycle Tracking

Since noise spikes can cause errors, a rate of change check is used to filter out the possible erroneous signals and so to make sure that the two points are interpolating from are genuine zero crossing points. (For example, if there are two negative samples, a noise spike can make one of them positive and, therefore, making the negative and positive pair looks as if there is a zero crossing).

The resultant cycle to cycle timing goes through a weak low pass filter to further smooth out cycle to cycle variations. The result is a stable and accurate frequency measurement result that is tolerant of noise.

2.3.6 Cycle Tracking and Forground Process Triggering

A cycle tracking counter and a sample counter keep track of how many samples have been accumulated together with a cycle counter. When 4 cycles worth of samples have been accumulated, the background process sets a flag to notify the foreground process, indicating it is a report cycle, that a new set of scaled results should be produced.

In the implementation, there are two sets of intermediate data registers: one set is using the background as intermediate data and the other set is using the foreground to calculate the readings. To save the processing time, when reached a report cycle, the background process passes the set of intermediate register to the foreground and uses another set to accumulate the next set of results.

2.3.7 Foreground Process

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

- Power processing: Calculate the active power, reactive power, fundamental active power, fundamental reactive power by scaling with number of samples and power scaling factor.
- Voltage processing: Calculate the RMS voltage, fundamental RMS voltage by scaling with number of samples, perform square root and multiply with the voltage scaling factor.
- Current processing:
  - Calculate the RMS current by scaling with number of samples, perform square root and multiply with the current scaling factor
  - Calculate fundamental RMS current from power and voltage
  - Calculate the apparent power by multiplying the root mean square voltage and root means square current
- Other processing:
  - Calculate the power factor
  - Calculate the frequency of the AC line

3 Calibration Techniques

3.1 Introduction

To maximize accuracy and to compensate component and manufacturing tolerance, it is necessary to go through a calibration process. The calibration techniques, required instruments, procedure and steps of calibration are discussed in this section.

3.2 Calibration Techniques

The calibration is defined based on the front-end interface model as shown in Figure 8.

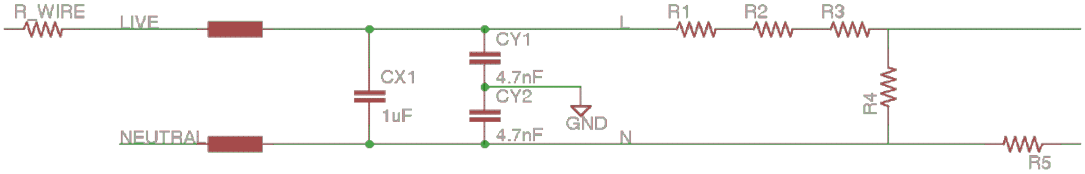


Figure 8. Front-End Interface Model

For the current design, a two-point calibration and characterization is required. VGAIN, IGAIN, PGAIN, CAP, RES, VDC_OFFSET, IDC_OFFSET are parameters that are calibrated during the process; an estimated value is put into the memory during design and characterization to help 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 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, 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. When using a user-calibration facility, the complete set of calibration values must be read, backed up, modified fields updated and the complete set written back to the EVM.

3.2.1 Calibration Setup

3.2.1.1 Apparatus

The following instrument list is suggested in order to perform calibration and test:

- AC source that could output sufficient power to drive the load at rated frequency (50 – 60 Hz), and rated voltage (110 V – 220 V) with 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 testset that could generate the rated loading can also be used.
- A reference meter that could give reading of AC parameters based on the voltage, current and phase setting from the AC meter testset

3.2.1.2 Instrument Link

AC Meter Test Set: http://en.3gcnkaipu.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/dcloads.php#6310a

3.2.1.3 Setup

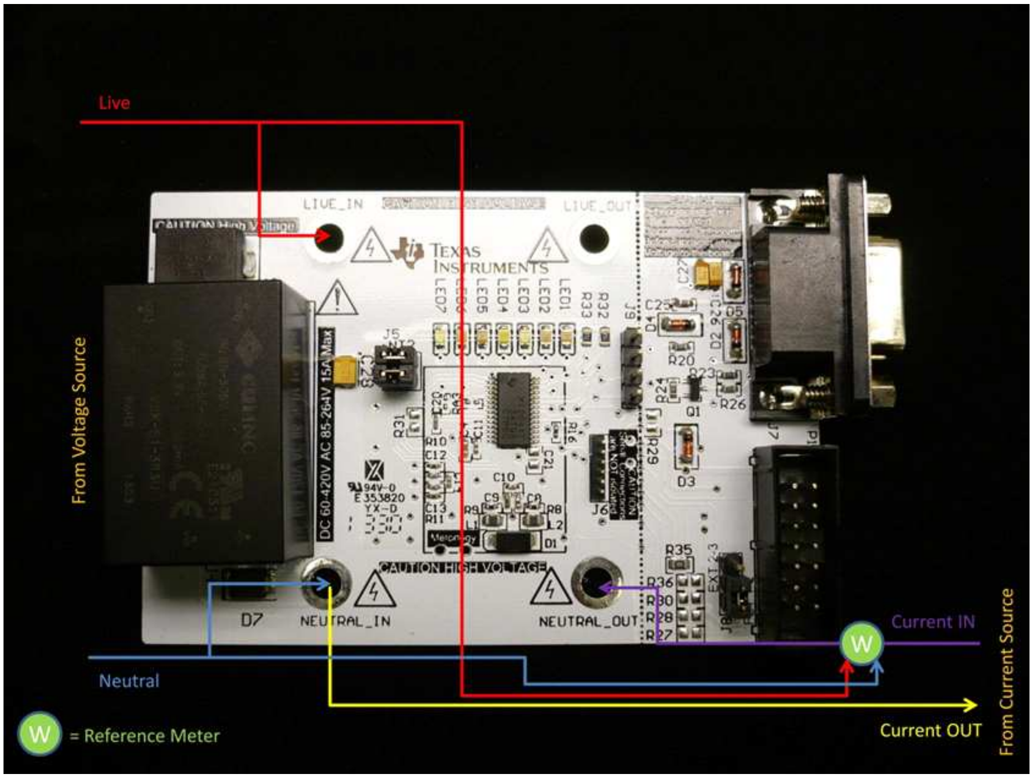


Figure 9. Calibration Setup

3.3 Calibration Procedures

3.3.1 Calibration of AC and DC Parameters

Use the following steps to calibrate the AC and DC parameters.

Calibrating VGAIN:

1. Set to the number of lowest possible load.
2. Set VIN to line voltage.
3. Calculate the value for VGAIN with the formula in Equation 14:

$$VGAIN_{n+1} = \frac{V_{REF}}{V_{UUT}} \times VGAIN_n \quad (14)$$

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 of error is used (as with the provided calibration software):

$$\%Error = \frac{V_{UUT} - V_{REF}}{V_{REF}} \times 100\% \quad (15)$$

4. Write and apply the calibrated VGAIN.

Calibrating IGAIN:

1. Set VIN to line voltage.
2. Set to high and highest possible load.
3. Calculate the IGAIN value with the formula in Equation 16:

$$IGAIN_{n+1} = \frac{I_{REF}}{I_{UUT}} \times IGAIN_{n} \tag{16}$$

where,

IGAIN$_{n+1}$ is the new voltage calibration factor

IGAIN$_{n}$ is the original voltage calibration factor

I$_{REF}$ is the reference meter voltage reading at the voltage set for IGAIN calibration

I$_{UUT}$ is the unit under test voltage reading at the voltage set for IGAIN calibration

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

$$\%Error = \frac{I_{UUT} - I_{REF}}{I_{REF}} \times 100\% \tag{17}$$

4. Write and apply the calibrated IGAIN.

Calibrating PGAIN:

1. Same condition as above and power factor = 1.
2. Note the percentage of error on voltage at this point.
3. Calculate PGAIN gain so that power is of the same percentage error as voltage at this point.

$$PGAIN_{n+1} = \frac{P_{REF}}{P_{UUT}} \times PGAIN_{n} \times \left(1 - \%\text{Error of } V_{UUT} \text{ at this load}\right) \tag{18}$$

where,

PGAIN$_{n+1}$ is the new power calibration factor

PGAIN$_{n}$ is the original power calibration factor

P$_{REF}$ is the reference meter voltage reading at the power set for PGAIN calibration

P$_{UUT}$ is the unit under test power reading at the power set for PGAIN calibration

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

$$\%Error = \left(1 + \%Error\right)\left(1 - \%\text{Error of } V_{UUT} \text{ at this load}\right) \tag{19}$$

4. Write and apply the calibrated PGAIN.

3.3.2 Calibration of Compensation Resistance and Capacitance

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

Calibrating RES:

1. Calculate RES with Equation 20.

$$R_{WIRE} = \frac{V_{REF}(\mathrm{Im}ax) - V_{UUT}(\mathrm{Im}ax)}{I_{\max} - I_{\min}} \approx \frac{V_{REF}(\mathrm{Im}ax) - V_{UUT}(\mathrm{Im}ax)}{I_{\max}} \tag{20}$$

where,

R$_{WIRE}$ is the estimated resistance

V$_{REF(Imax)}$ is the reference meter voltage reading at the current set for IGAIN calibration

V$_{UUT(Imax)}$ is the unit under test voltage reading at the current set for IGAIN calibration

I$_{max}$ is the current set for IGAIN calibration

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

2. Write and apply the calibrated RES (note the resistance is 1/256 - Ω unit).

Calibrating CAP:

1. Set to the number of lowest possible load.
2. Set VIN to low line voltage.
3. Calculate CAP with the formula in Equation 21:

$$C = \frac{1}{2\pi f V^2}\left(\sqrt{P_{APPERANT\_REF}{}^2 - P_{ACTIVE}{}^2} - \sqrt{P_{APPERANT\_UUT}{}^2 - P_{ACTIVE}{}^2}\right) \tag{21}$$

where,

C is the estimated EMI filter capacitance

V is the voltage set for calibration of C

$P_{APPERANT\_REF}$ is the apperant power reading of the reference meter at the setting for this calibration

$P_{APPERANT\_UUT}$ is the apperant power reading of the unit under test at the setting for this calibration

P_{ACTIVE} is active power reading of the reference meter

(suppose already calibrated to be the same as active power reading of UUT):

4. Write and apply the calibrated CAP (note the capacitance is in 1/64 – µF unit).

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 to the accuracy of power reading, this actually gives contribution to the current reading and its accuracy especially when current is small. To offset this, the EVM firmware has the mechanism to remove this from the current reading. The steps to remove 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 take an average as I_NOISE (in A).
4. Calculate the current AC offset value with Equation 22. (Note this is a big number even in case of a few mA of noise).

$$I\_AC\_OFFSET = \text{int}\left(I\_NOISE\left(\frac{1024 \times 10^6}{IGAIN}\right)\right)^2 \tag{22}$$

5. Write and apply the calibrated I_AC_OFFSET

3.3.4 Calibration of Voltage AC Offset

The voltage AC offset, in most cases, creates little effect to the voltage reading and does not require calibration.

3.3.5 Calibration of Phase Correction

1. Set the test set to generate rated voltage and set to calibration current (for example, 5 A).
2. Make sure calibrations are complete for PGAIN (at PF = 1).
3. Set the test set to output at power factor 0.5 (+ or – is not important at this point).
4. Note the power error.
5. Switch the test set to output at power factor 0.5 in the other direction to Step 3.
6. Note the power error.
7. At this point, both power errors should be approximately the same deviation but a different direction from the calibrated power error at PF = 1.
   (a) For example, at PF = 1, the calibrated error is 0.1%. If at PF = 0.5, the power error reads about 0.5%. Then at PF = –0.5, the power error should read about –0.3%.

8. Adjust the phase correction with time deviation from the current phase correction such that the power errors at PF = ±0.5 are minimized.
   (a) For example, if the current phase correction is 13 µs and 11 µs is desired, enter –2 into the phase correction box of the manual calibration window.

3.3.6 Calibration of DC Parameters

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

In fact, whenever the complete set of calibration value is read, the most update DC measurement parameters are also included in the set. When an update is done to the other parameters, the DC measurement parameter is also updated to the most updated value. However, the value is most accurate when the current is low (best with no current at all), thus, it is suggested to perform the 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 is modified and improved when the embedded metering customized calibration software is released.

4 Hardware Setup

This section introduces different parts on the EVM and describes the procedures of setting up the hardware for evaluation.

4.1 Top View of the EVM

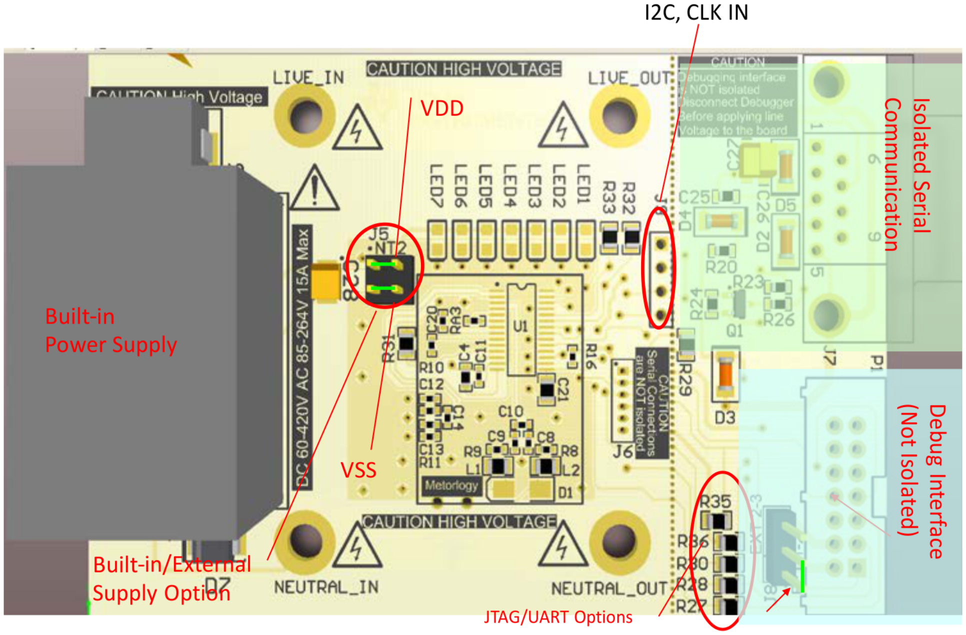


Figure 10. EVM Top View

4.2 Bottom View of the EVM

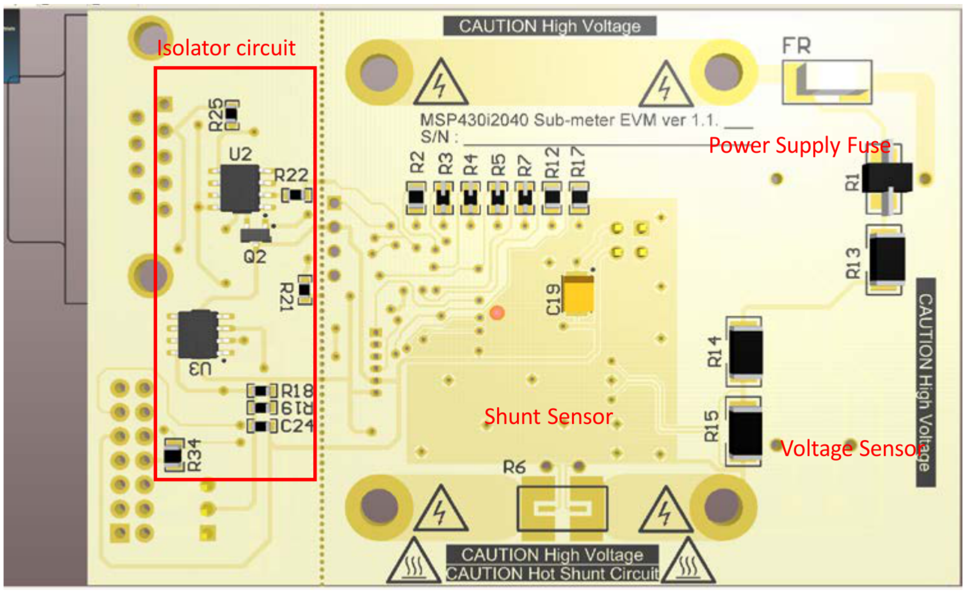


Figure 11. EVM Bottom View

4.3 Hardware Setup Procedures

4.3.1 Setting Up the Power Supply to EVM

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

The EVM is shipped with the built-in power supply block installed on the EVM. If power is to be supplied from an external power supply, the built-in VSK-S1-3P3U power supply block has to be disconnected. Remove the two jumpers on J5 and apply 3.3 V to V_{DD} and V_{SS} on J5 as shown in Figure 10.

WARNING

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

4.3.2 Setting Up the Serial Communication Interface

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

4.3.3 Setting Up 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

4.3.4 Setting Up the 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 i2040 is to be supplied by the FET, then short 1 – 2 of J8 (towards board edge). If the power (while debugging) to i2040 is to be supplied by voltage on V_{DD} pin, then short 2 – 3 of J8 (towards board center) (see Figure 10).

> **WARNING**
>
> **The debugging interface is NOT ISOLATED; make sure proper isolation is in place between the EVM and the PC used for debugging with AC or DC high voltage connected.**
>
> **Connection to debugging interface is optional for the operation of the EVM. The EVM can operate standalone without debugger connected.**

5 Calibrator Software

A package of software is needed to access full functionality of the EVM, which can be downloaded from the Texas Instruments website (TIDM-SERVER-PWR-MON). This section discusses the software in the software package and the procedure to setup the software to operate with the EVM.

5.1 Software Package Content

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

- Software tool to run on the PC for reading and calibrating the EVM. The setup and operation of this PC software tool is discussed in Section 5.2 and in Section 6.
- Source code and Embedded Metering Library. This is discussed in Section 8 and Appendix A.
- Hardware design files including schematic, layout and bill of materials are discussed in Section A.2

5.2 Setting Up the PC Software Tool

5.2.1 Minimum System Requirement

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

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

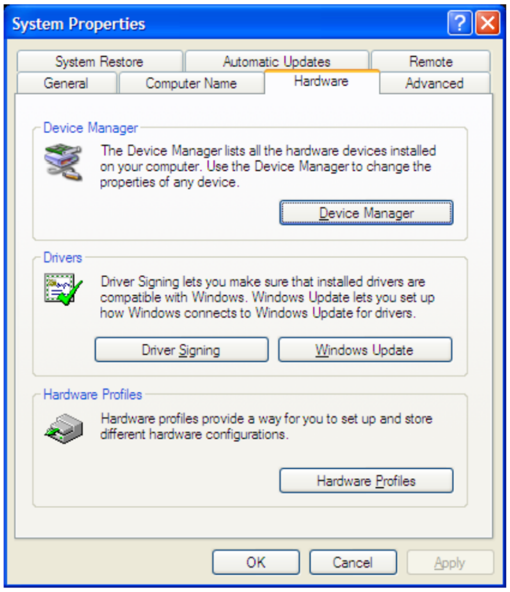
5.2.2 Installing the Software

Simply un-zip the file named "calibrator-runtime.zip” into any folder. A folder named "calibrator-runtime” containing the necessary files to run the software tool will be in that folder. 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. Editing to this XML file is required before calibrator-20121120.exe could be launched.

5.2.3 Configuring the Software

Use the following steps to setup 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 (see Figure 12), then Click on the Device Manager to go to the Device Manager window.



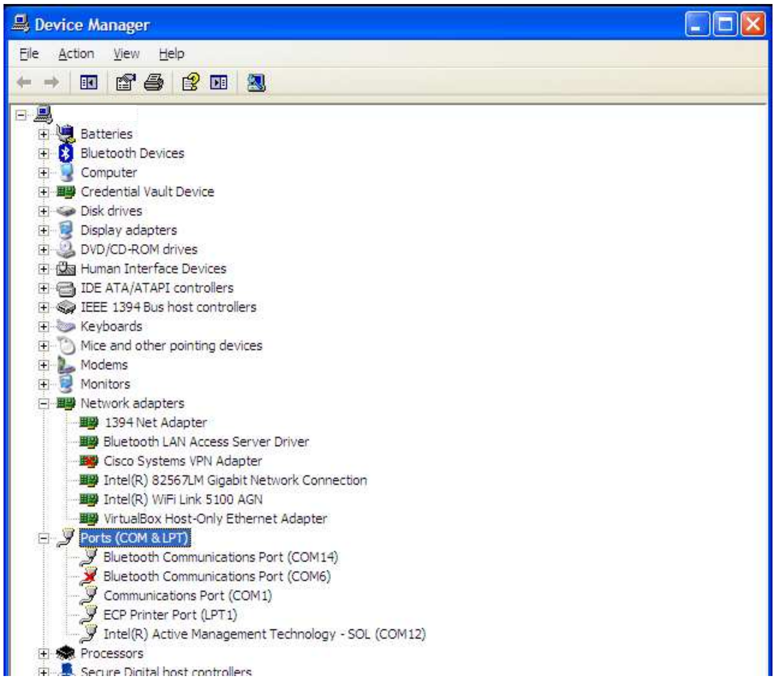
The Device Manager lists all the hardware devices installed on your computer. Use the Device Manager to change the properties of any device.

Driver Signing lets you make sure that installed drivers are compatible with Windows. Windows Update lets you set up how Windows connects to Windows Update for drivers.

Hardware profiles provide a way for you to set up and store different hardware configurations.

Figure 12. System Properties Window

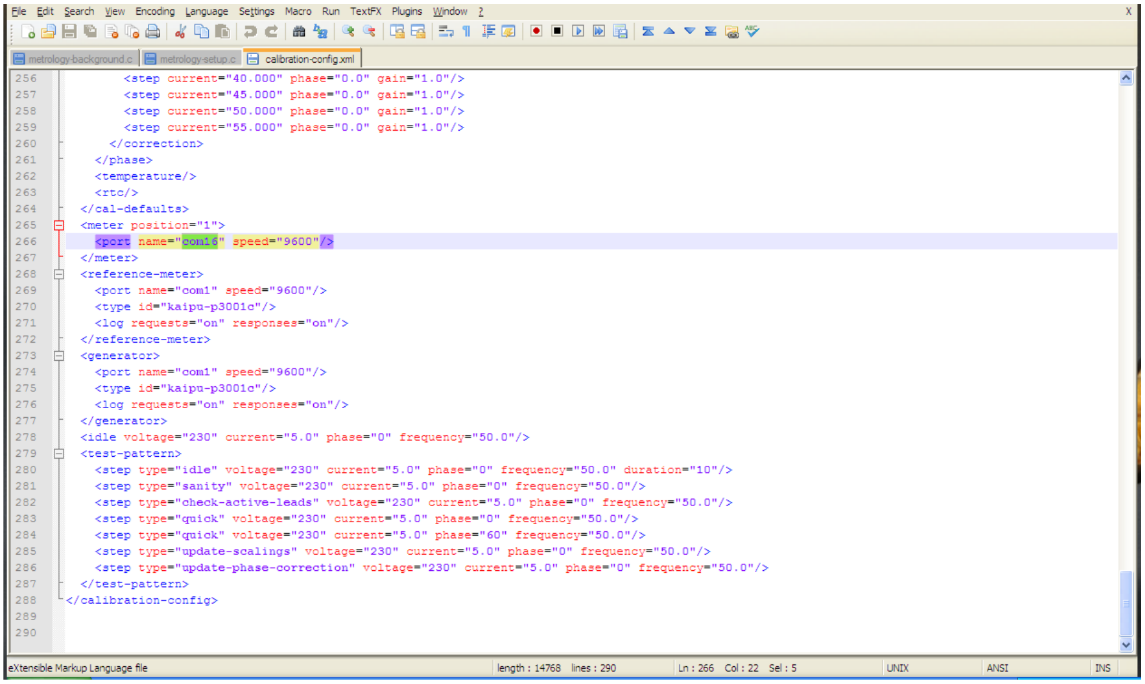
3. Check the COM port number of the serial port that is connecting the PC and the EVM.



- Batteries
- Bluetooth Devices
- Computer
- Credential Vault Device
- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Human Interface Devices
- IDE ATA/ATAPI controllers
- IEEE 1394 Bus host controllers
- Keyboards
- Mice and other pointing devices
- Modems
- Monitors
- Network adapters
  - 1394 Net Adapter
  - Bluetooth LAN Access Server Driver
  - Cisco Systems VPN Adapter
  - Intel(R) 82567LM Gigabit Network Connection
  - Intel(R) WiFi Link 5100 AGN
  - VirtualBox Host-Only Ethernet Adapter
- Ports (COM & LPT)
  - Bluetooth Communications Port (COM14)
  - Bluetooth Communications Port (COM6)
  - Communications Port (COM1)
  - ECP Printer Port (LPT1)
  - Intel(R) Active Management Technology - SOL (COM12)
- Processors
- Secure Digital host controllers

Figure 13. Device Manager Window

4. Open “calibration-config.xml” in the folder “calibrator-runtime” with a text editor or XML editor. Go to the line shown in Figure 14 and put in the COM port number in and save the file.


File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
metrology-background.c
metrology-setup.c
calibration-config.xml
eXtensible Markup Language file
length : 14768 lines : 290
Ln : 266 Col : 22 Sel : 5
UNIX
ANSI
INS

```
        <step current="40.000" phase="0.0" gain="1.0"/>
        <step current="45.000" phase="0.0" gain="1.0"/>
        <step current="50.000" phase="0.0" gain="1.0"/>
        <step current="55.000" phase="0.0" gain="1.0"/>
      </correction>
    </phase>
    <temperature/>
    <rtc/>
  </cal-defaults>
  <meter position="1">
    <port name="com16" speed="9600"/>
  </meter>
  <reference-meter>
    <port name="com1" speed="9600"/>
    <type id="kaipu-p3001c"/>
    <log requests="on" responses="on"/>
  </reference-meter>
  <generator>
    <port name="com1" speed="9600"/>
    <type id="kaipu-p3001c"/>
    <log requests="on" responses="on"/>
  </generator>
  <idle voltage="230" current="5.0" phase="0" frequency="50.0"/>
  <test-pattern>
    <step type="idle" voltage="230" current="5.0" phase="0" frequency="50.0" duration="10"/>
    <step type="sanity" voltage="230" current="5.0" phase="0" frequency="50.0"/>
    <step type="check-active-leads" voltage="230" current="5.0" phase="0" frequency="50.0"/>
    <step type="quick" voltage="230" current="5.0" phase="0" frequency="50.0"/>
    <step type="quick" voltage="230" current="5.0" phase="60" frequency="50.0"/>
    <step type="update-scalings" voltage="230" current="5.0" phase="0" frequency="50.0"/>
    <step type="update-phase-correction" voltage="230" current="5.0" phase="0" frequency="50.0"/>
  </test-pattern>
</calibration-config>
```

Figure 14. Editing “calibration-config.xml”

5.3 Instruments

A set of calibration factors that gives reading roughly to an accuracy of a few percent is defined in the example code. If more accurate result is preferred, the EVM needs to undergo a process of calibration. For more information on the calibration procedure, see Section 3. The following list of instruments are suggested in order to perform calibration:

- AC source that could output sufficient power to drive the load at rated frequency (50 – 60 Hz), and rated voltage (110 V – 220 V) or an AC testset that could 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 testset that could generate the rated loading.
- A reference meter that could give AC parameters of V_RMS, I_RMS, P_ACTIVE

The links below are the links to the instrument used for testing and calibrating the design:

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/dcloads.php#6310a

6 Operating the PC Software Tool

6.1 Introduction

Now the setup and configuration of hardware and software is completed. The operation of the EVM and the PC software tool is discussed here. Before proceeding, make sure the steps described in the previous section are completed and that power is applied to the EVM, and that line voltage to the load is applied.

6.2 Start Using the EVM

When the EVM has been powered, some of the seven LEDs will be flashing or turned on to indicate its operation status. Some of the LEDs are not used; make changes to the provided source code to make use of all the LEDs, as necessary. The list of LEDs' indication is listed in Table 1.

Table 1. LEDs' Indication Status

LED	State	Indication
LED1	ON	Background Operations Running
LED1	OFF	Background Operations Completed
LED2	ON	Voltage Negative Half Cycle
LED2	OFF	Voltage Positive Half Cycle
LED3	ON	Not used
LED3	OFF	Not used
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	Active Energy Pulse Pulsing
LED7	OFF	Active Energy Pulse Idle

The EVM is now ready to run, launch the software "calibrator-20121120.exe" in the folder "calibration-runtime" to start communicating with the EVM. A window as shown in Figure 15 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 the EVM from the PC is established and it flashes between read and green when the communication is taking place.

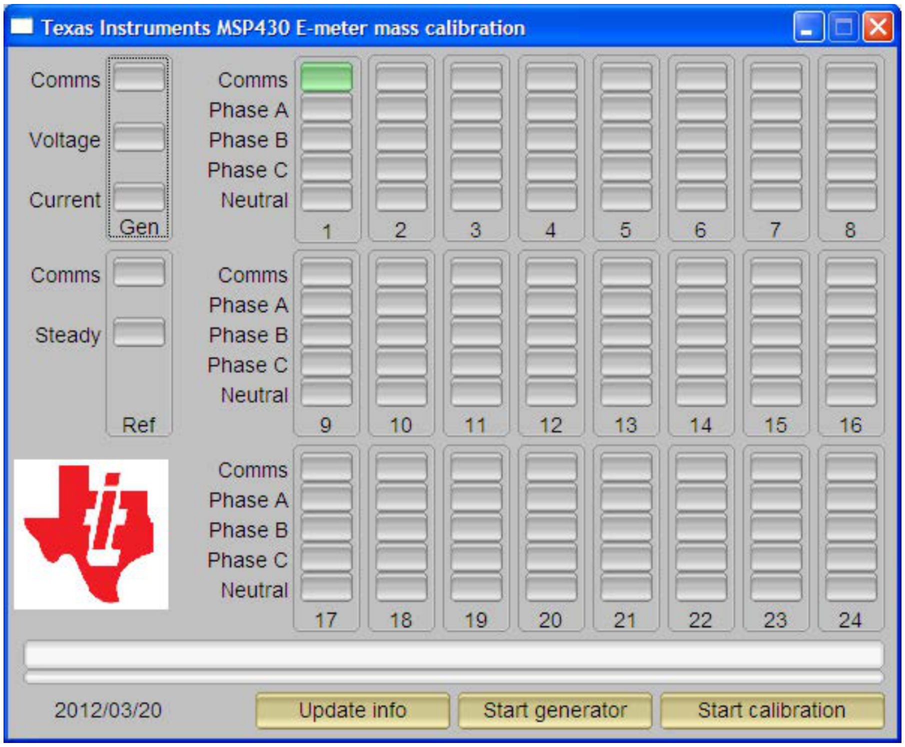
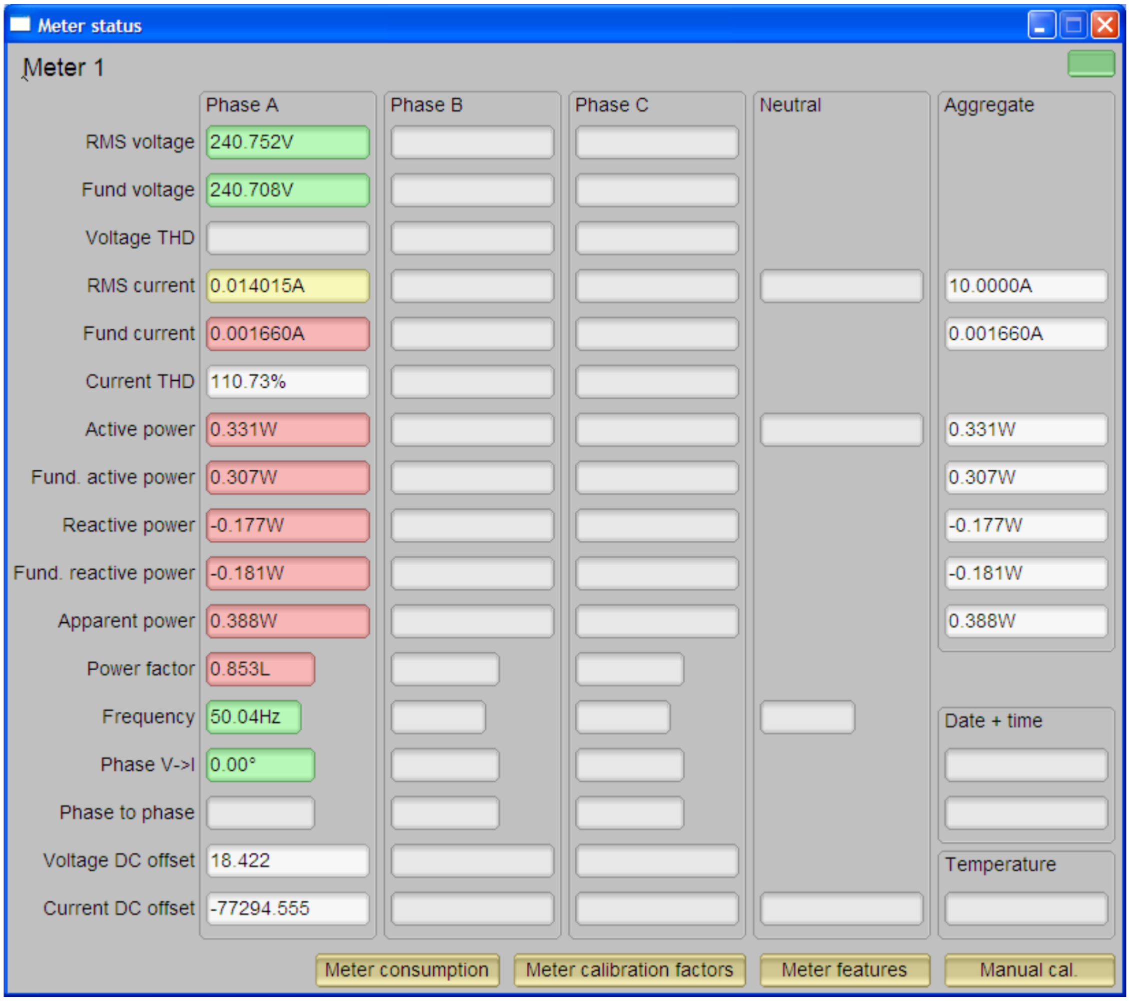


Figure 15. Calibrator Software Startup Window

Click on the [Comms] indicator in the Meter Status window as shown in Figure 16.



	Phase A	Phase B	Phase C	Neutral	Aggregate
RMS voltage	240.752V				
Fund voltage	240.708V				
Voltage THD					
RMS current	0.014015A				10.0000A
Fund current	0.001660A				0.001660A
Current THD	110.73%				
Active power	0.331W				0.331W
Fund. active power	0.307W				0.307W
Reactive power	-0.177W				-0.177W
Fund. reactive power	-0.181W				-0.181W
Apparent power	0.388W				0.388W
Power factor	0.853L				
Frequency	50.04Hz				
Phase V->I	0.00°				
Phase to phase					
Voltage DC offset	18.422				
Current DC offset	-77294.555				

Date + time:

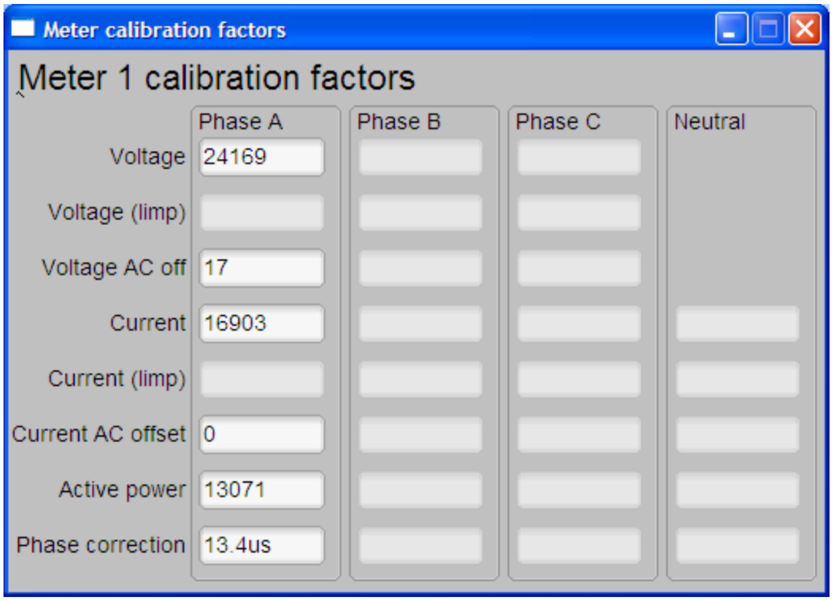
Temperature:

Figure 16. Meter Status Window

This window 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 the PC reads the EVM every second and it also does 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 four buttons:

1. Click on the Meter Consumption button, which brings up the Meter Consumption window, but the EVM does not support this feature. The Meter Consumption window provides no useful information.
2. Click on the Meter Calibration Factors button, which brings up the Meter Calibration Factors window (see Figure 17). The current calibration factor values are shown in this window.



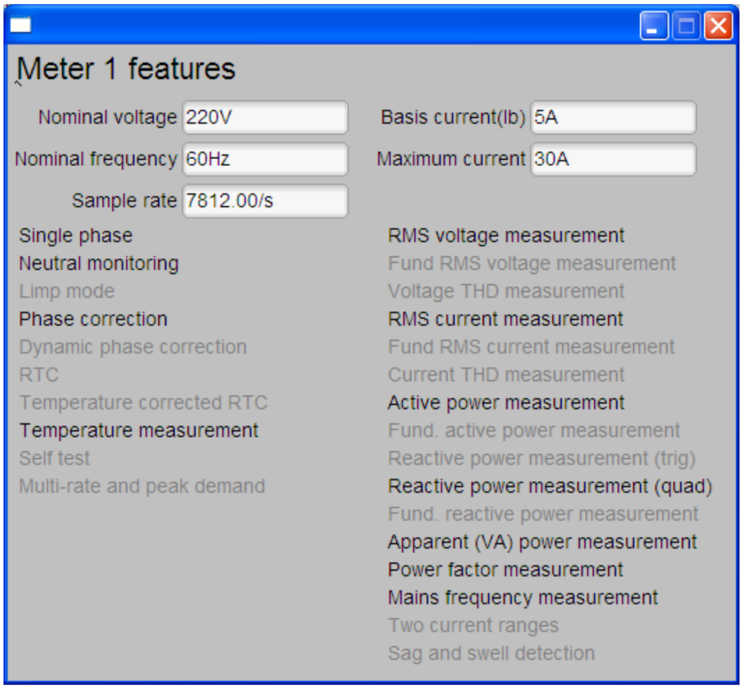
Meter calibration factors

Meter 1 calibration factors

	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 17. Meter Calibration Factor Window

3. Click on the Meter Features buttons, which brings up the Meter Features window (see Figure 18). In this window, the support feature of the EVM is reported in the Meters Feature window. Figure 18 only shows the look and feel of the meter feature window, but this is not the exact feature of the EVM.



Meter 1 features

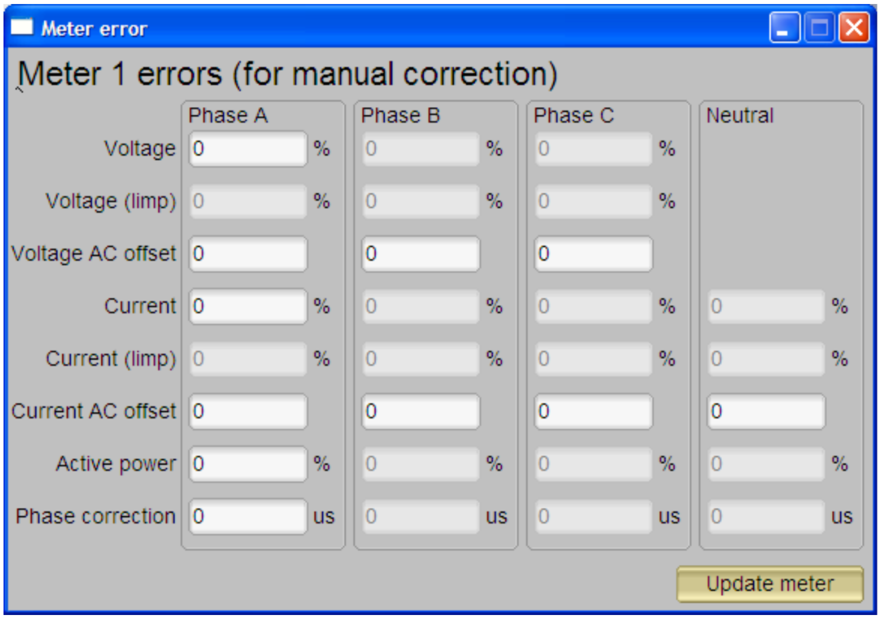
Nominal voltage: 220V
Basis current(Ib): 5A
Nominal frequency: 60Hz
Maximum current: 30A
Sample rate: 7812.00/s

Single phase
Neutral monitoring
Limp mode
Phase correction
Dynamic phase correction
RTC
Temperature corrected RTC
Temperature measurement
Self test
Multi-rate and peak demand

RMS voltage measurement
Fund RMS voltage measurement
Voltage THD measurement
RMS current measurement
Fund RMS current measurement
Current THD measurement
Active power measurement
Fund. active power measurement
Reactive power measurement (trig)
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 18. Meter Features Window

4. Click on the Manual Call button, which brings up the Meter Error window (see Figure 19). 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. For more details on the technique and procedure of performing calibration, see Section 3.



Meter 1 errors (for manual correction)

	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

Figure 19. 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 23:

$$\%Error = \frac{\text{EVM Reading} - \text{Reference Meter Reading}}{\text{Reference Meter Reading}} \times 100\% \tag{23}$$

Put the percentage error in the corresponding box in the Meter Error Window (see Figure 19). Click Update meter and the updated calibration values will be calculated and written to the EVM. The corresponding values will be reflected in the Meter Calibration Factor Window.

6.3 Known Issues

The calibrator software is legacy software that operates with utility meters that has not customized completely for embedded metering. The following 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 is interpreted 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 put into the voltage AC offset box is actually written to the 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 calibration software takes the current and voltage DC offset value every time any calibration value is updated.
- Aggregate current in the *Meter Status* windows always shows 10.0000A.

7 Serial Communication Commands

7.1 Introduction

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. This section discusses the protocol of the EVM sending out readings and commands.

NOTE: Change can be made to the provided source code to perform custom communication protocol, command, response and functions.

7.2 Communication Protocol

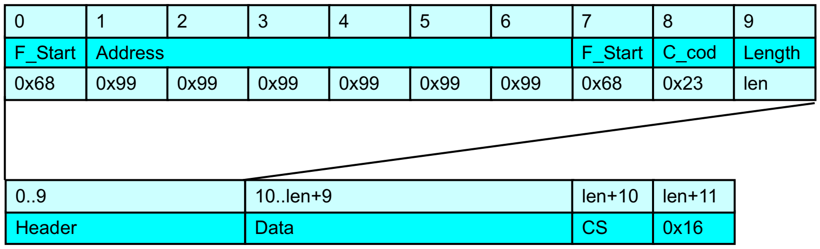
During the operation, the EVM communicates with the external host via the serial communication port with these parameters:

- 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.

7.2.1 Polling Mode

When the EVM is running, it waits for a command from the host through the serial port. When the command frame is received and passes frame checking, the EVM interprets the command and performs the requested action. The EVM communicates with the host using a data frame format as indicated in Figure 20.



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 20. 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.

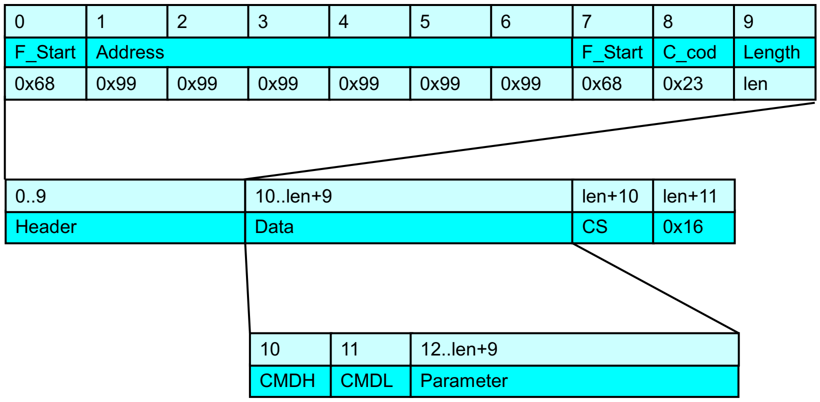
The header starts with 0x68, followed by an address field, which is fixed to 0x999999999999. The provided communication protocol source code can be modified (emeter-dlt645.c) to make the EVM respond to different address.

Followed by a fixed delimiter, 0x68 and 0x23 are the length bytes that indicate 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.

7.2.1.1 Command and Respond Frame

Figure 21 and Figure 22 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 are 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 (because of that CMDL cannot use value larger than 0x7F). The respond data is defined by the command received.

The definition of commands and the corresponding response are listed in the next section.



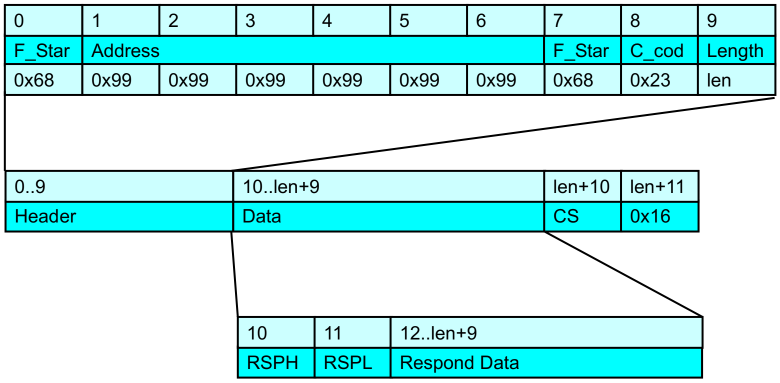
0	1	2	3	4	5	6	7	8	9
F_Start	Address						F_Start	C_cod	Length
0x68	0x99	0x99	0x99	0x99	0x99	0x99	0x68	0x23	len

0..9	10..len+9	len+10	len+11
Header	Data	CS	0x16

10	11	12..len+9
CMDH	CMDL	Parameter

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

Figure 21. Polling Mode Command Frame Format



0	1	2	3	4	5	6	7	8	9
F_Star	Address						F_Star	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

10	11	12..len+9
RSPH	RSPL	Respond Data

Figure 22. 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

7.3 Commands

7.3.1 HOST_CMD_GET_METER_NAME

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

7.3.1.1 Command Format

Table 2. 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

7.3.2 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

These can be found in “metrology-calibration-template.h”.

7.3.2.1 Command Format

Table 3. 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 bytes software version
				6	U8(4)	4 bytes hardware version
				10	U8(4)	4 bytes metrology version
				14	U8(4)	4 bytes protocol version

7.3.3 HOST_CMD_GET_METER_CONFIGURATION

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

7.3.3.1 Command Format

Table 4. 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*
				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				12	U16	Mains Basis Current
				14	U16	Mains Maximum Current
				16	U32	100 times the sample rate

7.3.3.2 Parameter Definition

Table 5. Parameter Definitions

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 I_{RMS}
Bit 3	Reserved	Bit 3	Measures V_{RMS}
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 I_{RMS}
Bit 3	RTC support	Bit 3	Measures fundamental V_{RMS}
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

7.3.4 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.

7.3.4.1 Command Format

Table 6. 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(6)	6 bytes dummy data
				8	S16	Temperature in 0.01°C

7.3.5 HOST_CMD_ALIGN_WITH_CALIBRATION_FACTORS

This commands the EVM to reload the calibration factors from Flash into the operation of its measurement activities.

7.3.5.1 Command Format

Table 7. 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

7.3.6 HOST_CMD_SET_PASSWORD

This command passes the password from the host to the EVM to enable calibration mode, which allows calibration and other functions to be executed. If auto report mode is enabled, this command also disables 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 3 = 0xDEF0. (The password can be changed by modifying the entry in emeter-template.h followed by a recompile).

7.3.6.1 Command Format

Table 8. HOST_CMD_SET_PASSWORD

	Command			**Respond**		
LEN	10			2		
	Offset	Width	Data	Offset	Width	Data
CMDH	0	U8	0x5A	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			

7.3.7 HOST_CMD_GET_READINGS_PHASE_N

This command reads the latest measurements from the EVM. 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.

7.3.7.1 Command Format

Table 9. 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 µA
				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 0.01Hz
				26	S32	Voltage channel DC offset
				30	S32	Current channel DC offset

7.3.8 HOST_CMD_GET_EXTRA_READINGS_PHASE_N

This command reads the latest extra measurements from the EVM. 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.

7.3.8.1 Command Format

Table 10. 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 µA
				18	U16	Voltage THD in 0.01%
				20	U16	Current THD in 0.01%
				22		Next 12 bytes reserved

7.3.9 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).

7.3.9.1 Command Format

Table 11. 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			

7.3.10 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.

7.3.10.1 Command Format

Table 12. 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

7.3.11 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 are read, saved and erase before executing this command.

NOTE: In the case of this EVM, only CMDH = 0xD1 is supported.

7.3.11.1 Command Format

Table 13. 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 = 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	0xdp (always 0x00 in this EVM)			
	4	U16	Inlet Capacitance in 1/64 µF			
	6	S32	Current Channel DC Offset			
	10	U32	Voltage Channel AC Offset			
	14	U32	Current Channel AC Offset			
	18	S16	Phase Correction in 1/1024Ts			
	20	U16	V_{rms} Scaling Factor			
	22	U16	Wire Resistance in 1/256 Ω			
	24	U16	I_{rms} Scaling Factor			
	26	U16	0x0000 reserved			
	28	U16	Power Scaling Factor			

7.3.12 HOST_CMD_GET_CALIBRATION_PHASE_N

This command reads the calibration value from the EVM.

NOTE: In the case of this EVM, only CMDH = 0xD6 is supported.

7.3.12.1 Command Format

Table 14. 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 µF
				6	S32	Current Channel DC Offset
				10	U32	Voltage Channel AC Offset
				14	U32	Current Channel AC Offset
				18	S16	Phase Correction in 1/1024Ts
				20	U16	V_{rms} Scaling Factor
				22	U16	Wire Resistance in 1/256 Ω
				24	U16	I_{rms} Scaling Factor
				26	U16	Reserved
				28	U16	Power Scaling Factor

7.3.13 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 value is read, saved and erase before executing this command.

7.3.13.1 Command Format

Table 15. 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	Intercept Temperature			
	8	U16	Temperature slope / degree			

7.3.14 HOST_CMD_GET_CALIBRATION_EXTRAS

This command reads the extra calibration value from the EVM.

7.3.14.1 Command Format

Table 16. 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

8 Firmware and Embedded Metering Library API

8.1 Introduction

There is a pre-tested firmware enables the EVM to operate. The source code together with the embedded metering library, as an example application, is provided. The summary of the APIs of the embedded metering library is discussed in this section.

8.2 Embedded Metering LIbrary API

The API of the embedded metering library are introduced in this section.

The metrology library communicates to the 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 the user applications to access calibration parameters and for the definition of default calibration parameter.

8.2.1 Embedded Metering Library Function Calls

Functions calls to interface to the metrology library are defined in “metrology-readings.h” and “metrology-foreground.h”, which provides access to:

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

8.2.1.1 Functions for Metrology Engine Control

Table 17. Functions for Metrology Engine Control

int metrology_init (void)

Description 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)

Description 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 EMI capacitor compensation.

Parameter None

Return Always 0x0000.

void align_metrology_with_calibration_data (void)

Description Re-initialize the metrology related ADC channels with proper phase correction.

Parameter None

Return None.

void metrology_switch_to_normal_mode (void)

Description 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)

Description Initialize the hardware of all metrology related ADC channels.

Parameter None

Return None.

void metrology_disable_analog_front_end (void)

Description Disable the hardware of all metrology related ADC channels.

Parameter None

Return None.

8.2.1.2 Procedure for Metrology Engine Initialization

- metrology_init () - Initializes uninitialized calibration parameter with default value
- metrology_disable_analog_front_end () - Prevents metrology ADCs from generating interrupt during initialization
- metrology_init_from_nv_data () – Initializes DC filter
- Does other setup operations that requires that the ADC interrupt be disabled
- metrology_switch_to_normal_mode () – Switches to normal, initialize metrology related ADCs

8.2.1.3 Functions for Calculate and Reading the Readings

Table 18. Functions for Calculate and Reading the Readings

Title **Page**

power_t calculate_phase_readings (void

Description 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 process. Upon returning all metrology reading is updated.

Parameter None

Return Active power reading.

power_t active_power (int ph)

Description Gets the latest active power reading of phase ph.

Parameter ph = 1

Return Latest active power of phase ph.

power_t reactive_power (int ph)

Description 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 active power of phase ph

power_t apparent_power (int ph)

Description 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 active power of phase ph

power_t fundamental_active_power(int ph)

Description 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.

power_t fundamental_reactive_power(int ph)

Description 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.

power_factor_t power_factor (int ph)

Description 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)

Description Returns the latest root mean square voltage of phase ph.

Parameter ph = 1

Return Latest root mean square voltage measured on phase ph.

rms_voltage_t fundamental_rms_voltage(int ph)

Description Returns the latest fundamental root mean square voltage of phase ph.

Parameter ph = 1

Return Latest fundamental root mean square voltage measured on phase ph.

thd_t voltage_thd(int ph)

Description Returns the latest THD on voltage of phase ph.

Parameter ph = 1

Return Latest THD on voltage of phase.

rms_current_t rms_current (int ph)

Description Returns the latest root mean square current of phase ph.

Parameter ph = 1

Return Latest returns the latest root mean square current of phase ph.

rms_current_t fundamental_rms_current(int ph)

Description Returns the latest fundamental root mean square current of phase ph.

Parameter ph = 1

Return Latest fundamental root mean square current of phase ph.

thd_t current_thd(int ph)

Description Returns the latest THD on current of phase ph.

Parameter ph = 1

Return Latest THD on current of phase.

int16_t mains_frequency (int ph)

Description 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)

Description Returns the status word of phase ph.

Parameter ph = 1

Return Status word of phase ph.

8.2.2 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 ZERO_CROSS_ON (void)**

  Called when voltage signal is zero crossing from positive to negative
- **void ZERO_CROSS_ON (void)**

  Called when voltage signal is zero crossing from negative to positive
- **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 mode
- **void active_energy_pulse_start (void)**

  Called when an energy pulse output is going from idle to pulsing state.
- **void active_energy_pulse_end (void)**

  Called when an energy pulse output is going from pulsing to idle

8.2.3 Application Level Calibration Functions

8.2.3.1 Functions for Reading and Writing Calibration Parameters

Table 19. Functions for Reading and Writing Calibration Parameters

int get_calibration_status (void)

Description

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)

Description

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)

Description Clears 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.

int16_t get_temperature_intercept (void)

Description Gets 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)

Description Gets the calibration value of the temperature slope.

Parameter None.

Return The calibrated ADC reading for increment per 1° Celsius in calibration parameter memory.

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

Description Writes temperature related calibration value into calibration parameter memory.

- temperature_at_calibration : the temperature in 0.01° Celsius which the calibration is done.
- temperature_sensor_intercept : the ADC reading at calibration temperature.
- temperature_sensor_slope : the ADC reading increment per 1° Celsius.

Parameter None.

calibration_scaling_factor_t get_P_scaling (int phx)

Description Gets 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)

Description Writes 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)

Description Gets 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)

Description Writes 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)

Description Gets 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)

Description Gets 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)

Description Writes 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)

Description Gets 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)

Description Writes the voltage channel AC offset of phase phx with value. Value should be calculated as show in Equation 24:

$$V\_AC\_OFFSET = \text{int}\left(V\_NOISE\left(\frac{1024 \times 10^3}{VGAIN}\right)\right)^2 \tag{24}$$

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.

calibration_scaling_factor_t get_I_rms_scaling(int phx);

Description Gets 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);

Description Writes 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);

Description Gets 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)

Description Gets 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);

Description Writes 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_estimate(int phx);

Description Gets 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)

Description Writes the current channel AC offset of phase phx with value. Value should be calculated as shown in Equation 25:

$$I\_AC\_OFFSET = \text{int}\left(I\_NOISE\left(\frac{1024 \times 10^6}{IGAIN}\right)\right)^2 \tag{25}$$

Parameter phx = 1

Value In square of ADC counts, he value to be written to current channel AC offset for phase phx in calibration parameter memory.

Return None.

uint16_t get_compensate_capacitor_value (int phx)

Description Gets 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.

void set_compensate_capacitor_value (int phx, uint16_t value)

Description Writes EMI filter capacitor value of phase phx with value.

Parameter phx = 1

Value The capacitance to be written (in 1/64 µF unit) to calibration parameter memory.

Return None.

uint16_t get_compensate_resistance (int phx)

Description Gets wire resistance value of phase phx.

Parameter phx = 1

Return The wire resistance to be compensate (in 1/256 Ω unit) in calibration parameter memory.

void set_compensate_resistance (int phx, uint16_t value)

Description Writes wire resistance value of phase phx with value.

Parameter phx = 1

Value The wire resistance to be written (in 1/256 Ω unit) to calibration parameter memory.

Return None.

int16_t get_phase_corr (int phx)

Description Gets phase correction of phx.

Parameter phx = 1

Return The phase correction value (in unit of 0.9765625 µs unit) in calibration parameter memory.

void set_phase_corr (int phx, int16_t value)

Description Writes phase correction of phase phx with value.

Parameter phx = 1

Value The phase correction value (in unit of 0.9765625 µs unit) to be written to calibration parameter memory.

Return None.

8.2.4 Setting Default Calibration Parameters

Being part of metrology but not the metrology itself, the calibration parameter default values reading and manipulation functions are 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 1° Celsius increment
- **DEFAULT_ROOM_TEMPERATURE**

  Calibration temperature in 0.01° 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 (so 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:

$$\text{DEFAULT\_I\_AC\_OFFSET} = \left(\frac{1024 \times 10^6}{\text{Current Scaling Factor}} \times \text{Estimated noise level}\right)^2 \tag{26}$$

- **DEFAULT_EMI_FILTER_CAP_UF_A**

  EMI filter capacitance in 1/64 µF unit, Maximum 1023 (15.984375 µF)

  Putting value greater than 0x8000 results 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_METE_NAME and is defined as in the provided example code "MSP430I2040SUBMETEREVM.
- **METER_SOFTWARE_VERSION**

  Defines the software version of the meter as reported by HOST_CMD_GET_METE_VER.
- **METER_METROLOGY_VERSION**

  Defines the Metrology library version of the meter as reported by HOST_CMD_GET_METE_VER.
- **METER_PROTOCOL_VERSION**

  Defines the Protocol version of the meter as reported by HOST_CMD_GET_METE_VER.

Appendix A Example Application Code

A.1 Introduction

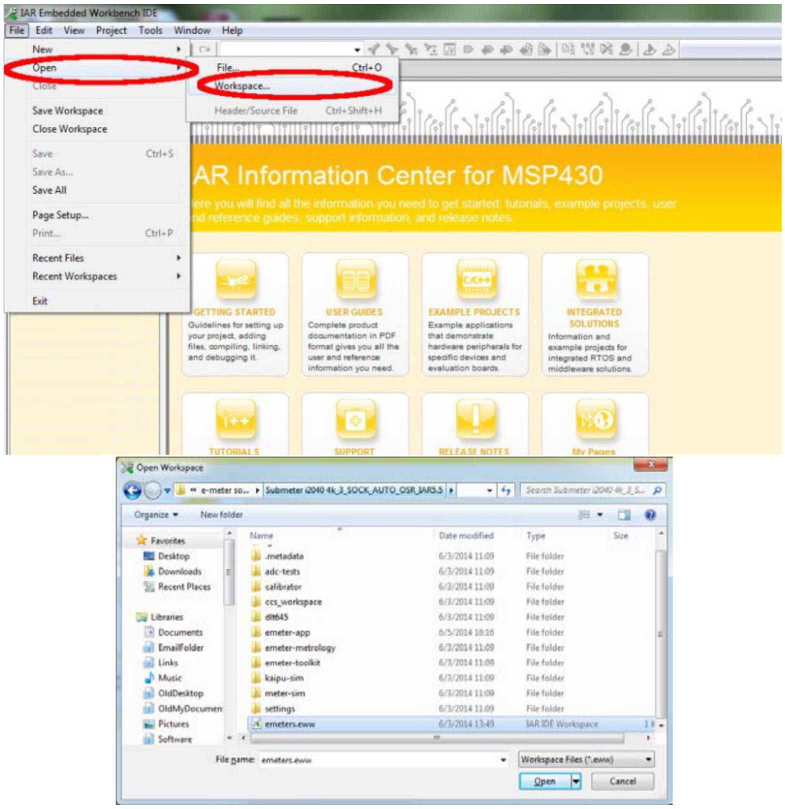
Project Structure:

- 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
- 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. Launch the IAR5.5 Embedded Workbench IDE, then click on File → Open → Workspace.
2. Select “emeters.eww” when prompted for Workspace to be opened.


IAR Embedded Workbench IDE
File
Edit
View
Project
Tools
Window
Help
New
Open
Close
Save Workspace
Close Workspace
Save Ctrl+S
Save As...
Save All
Page Setup...
Print... Ctrl+P
Recent Files
Recent Workspaces
Exit
File... Ctrl+O
Workspace...
Header/Source File Ctrl+Shift+H
IAR Information Center for MSP430
Here you will find all the information you need to get started: tutorials, example projects, user and reference guides, support information, and release notes.
GETTING STARTED
Guidelines for setting up your project, adding files, compiling, linking, and debugging it.
USER GUIDES
Complete product documentation in PDF format gives you all the user and reference information you need.
EXAMPLE PROJECTS
Example applications that demonstrate hardware peripherals for specific devices and evaluation boards.
INTEGRATED SOLUTIONS
Information and example projects for integrated RTOS and middleware solutions.
TUTORIALS
SUPPORT
RELEASE NOTES
My Pages
Open Workspace
e-meter so... ▸ Submeter i2040 4k_3_SOCK_AUTO_OSR_IAR5.5 ▸
Search Submeter i2040 4k_3_S...
Organize
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
EmailFolder
Links
Music
OldDesktop
OldMyDocumen
Pictures
Software
Open
Cancel

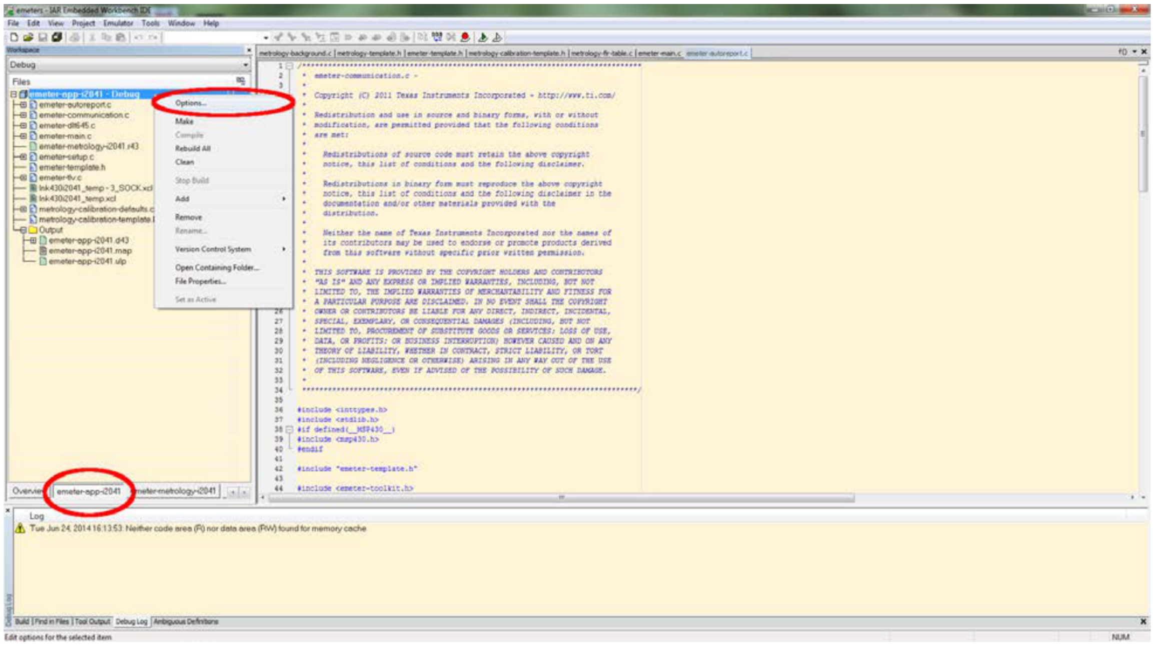
Name	Date modified	Type	Size
.metadata	6/3/2014 11:09	File folder	
adc-tests	6/3/2014 11:09	File folder	
calibrator	6/3/2014 11:09	File folder	
ccs_workspace	6/3/2014 11:09	File folder	
dlt645	6/3/2014 11:09	File folder	
emeter-app	6/5/2014 18:16	File folder	
emeter-metrology	6/3/2014 11:09	File folder	
emeter-toolkit	6/3/2014 11:09	File folder	
kaipu-sim	6/3/2014 11:09	File folder	
meter-sim	6/3/2014 11:09	File folder	
settings	6/3/2014 11:09	File folder	
emeters.eww	6/3/2014 13:49	IAR IDE Workspace	1

File name: emeters.eww

Workspace Files (*.eww)

Figure 23. Opening Workspace

3. Select the emeter-app-i2041 project tab at the bottom of the Workspace window.
4. Check the project options by right clicking on the project name and selecting Options… from the pop-up menu as shown in Figure 24.

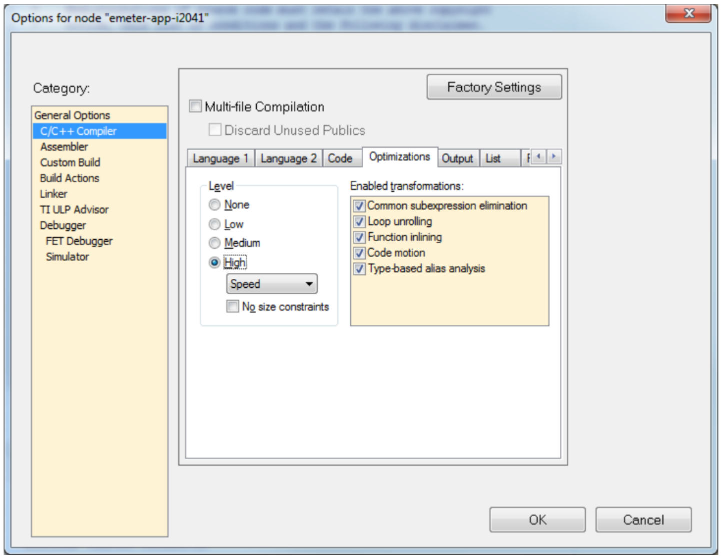


```
/**************************************************************************
 *  emeter-communication.c -
 *
 *  Copyright (C) 2011 Texas Instruments Incorporated - http://www.ti.com/
 *
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions
 *  are met:
 *
 *    Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 *    Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the
 *    distribution.
 *
 *    Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 *  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 *  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 *  A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 *  OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 *  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 *  LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 *  DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 *  THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 *  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 *  OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 **************************************************************************/

#include <inttypes.h>
#include <stdlib.h>
#if defined(__MSP430__)
#include <msp430.h>
#endif

#include "emeter-template.h"

#include <emeter-toolkit.h>
```

```
Tue Jun 24, 2014 16:13:53: Neither code area (R) nor data area (RW) found for memory cache
```

Figure 24. Project Tab

5. 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 25).



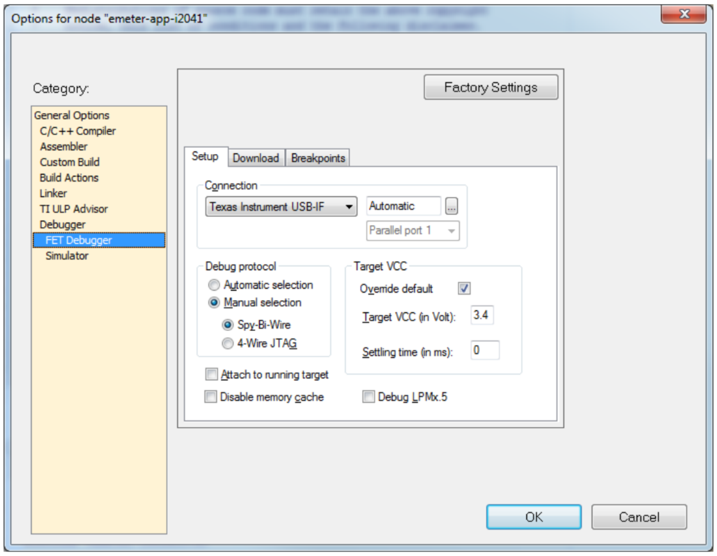
☐ Multi-file Compilation
☐ Discard Unused Publics

Level
○ None
○ Low
○ Medium
● High
Speed
☐ No size constraints

Enabled transformations:
☑ Common subexpression elimination
☑ Loop unrolling
☑ Function inlining
☑ Code motion
☑ Type-based alias analysis

Figure 25. Optimization Options

6. Select FET Debugger on the left hand column of the select Setup tab. The EVM uses Spy-Bi-Wire for its code downloading and debugging. Check to make sure the options are as shown in Figure 26.



Options for node "emeter-app-i2041"

Category:

- General Options
- C/C++ Compiler
- Assembler
- Custom Build
- Build Actions
- Linker
- TI ULP Advisor
- Debugger
- FET Debugger
- Simulator

Factory Settings

Setup | Download | Breakpoints

Connection: Texas Instrument USB-IF; Automatic; Parallel port 1

Debug protocol:
○ Automatic selection
● Manual selection
● Spy-Bi-Wire
○ 4-Wire JTAG

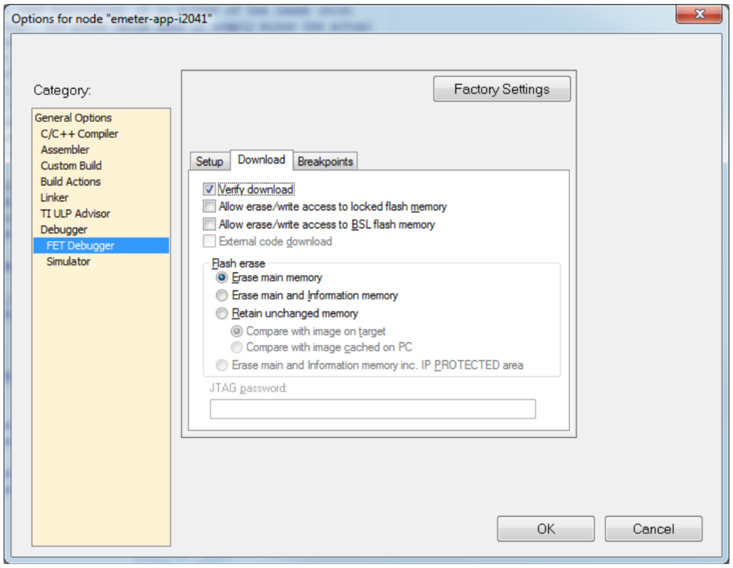
Target VCC:
Override default ☑
Target VCC (in Volt): 3.4
Settling time (in ms): 0

□ Attach to running target
□ Disable memory cache
□ Debug LPMx.5

OK Cancel

Figure 26. Debugger Options

7. Select the Download tab. Under Flash erase, do not choose Erase main memory and Information memory; this option erases both sets of data and cannot be recovered. Instead, choose Erase main memory as the download option to preserve these factory parameters: system clock calibration, ADC calibration, and internal reference calibration (Figure 27). However, metrology calibration stored in the main memory, such as VGAIN, IGAIN, PGAIN, and so on, are always erased after downloading. Select Erase main memory as the option for download to preserve the factory parameters.



Options for node "emeter-app-i2041"

Category:

- General Options
- C/C++ Compiler
- Assembler
- Custom Build
- Build Actions
- Linker
- TI ULP Advisor
- Debugger
- FET Debugger
- Simulator

Factory Settings

Setup | Download | Breakpoints

☑ Verify download
□ Allow erase/write access to locked flash memory
□ Allow erase/write access to BSL flash memory
□ External code download

Flash erase:
● Erase main memory
○ Erase main and Information memory
○ Retain unchanged memory
● Compare with image on target
○ Compare with image cached on PC
○ Erase main and Information memory inc. IP PROTECTED area

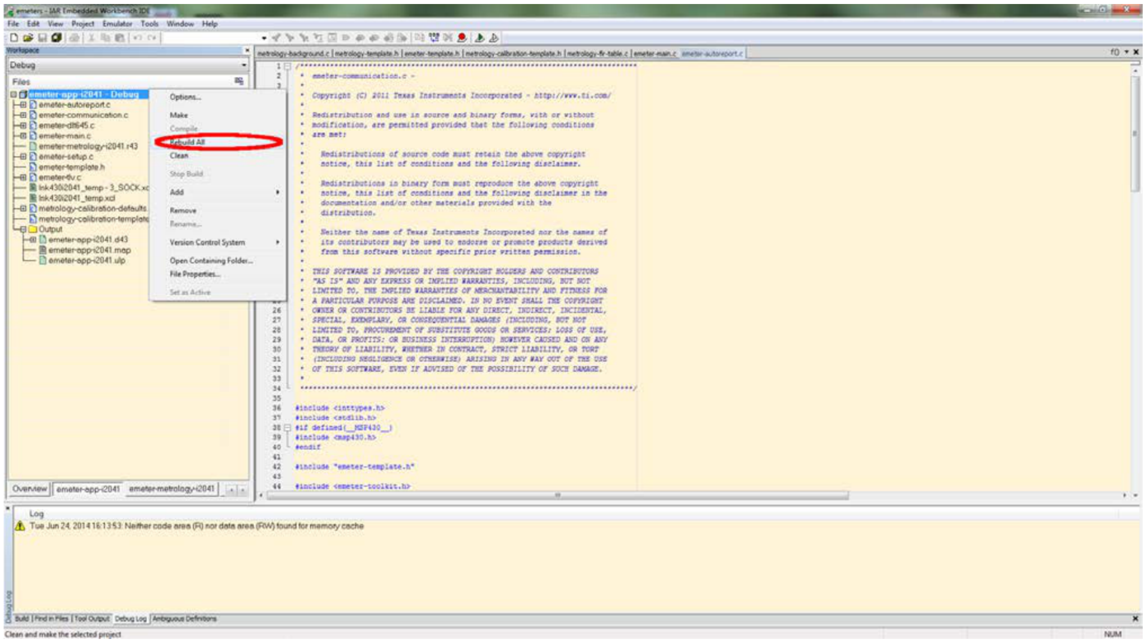
JTAG password:

OK Cancel

Figure 27. Download Options

8. Click OK after all of the changes are made.

9. Rebuild the project by right clicking on the project and selecting Rebuild All from the pop-up menu (see Figure 28). Three warnings are reported during rebuilding (see Figure 28), it is safe to ignore the three warnings. To open the project workspace, modify the code, compile, and download the code; IAR Embedded Workbench® 5.5 must be installed with a valid license. If a valid license is not available, the object code can still be downloaded. For downloading procedures, see Section A.3.

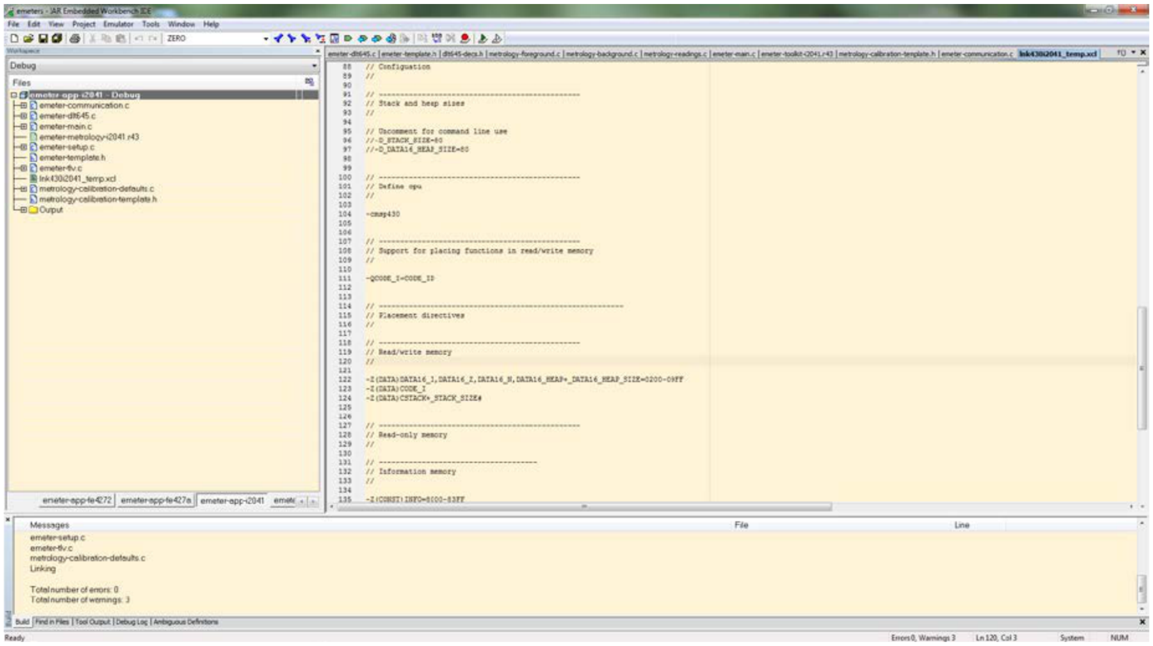


```
/*******************************************************************************
 *  emeter-communication.c -
 *
 *  Copyright (C) 2011 Texas Instruments Incorporated - http://www.ti.com/
 *
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions
 *  are met:
 *
 *    Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 *    Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the
 *    distribution.
 *
 *    Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 *  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 *  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 *  A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 *  OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 *  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 *  LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 *  DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 *  THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 *  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 *  OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ******************************************************************************/

#include <inttypes.h>
#include <stdlib.h>
#if defined(__MSP430__)
#include <msp430.h>
#endif

#include "emeter-template.h"

#include <emeter-toolkit.h>
```

```
Log
Tue Jun 24, 2014 16:13:53: Neither code area (R) nor data area (RW) found for memory cache
```

Clean and make the selected project

Figure 28. Compiling the Application



```
// Configuation
//

// ---------------------------------------------
// Stack and heap sizes
//

// Uncomment for command line use
//-D_STACK_SIZE=80
//-D_DATA16_HEAP_SIZE=80


// ---------------------------------------------
// Define cpu
//

-cmsp430


// ---------------------------------------------
// Support for placing functions in read/write memory
//

-QCODE_I=CODE_ID


// --------------------------------------------------------
// Placement directives
//

// ---------------------------------------------
// Read/write memory
//

-Z(DATA)DATA16_I,DATA16_Z,DATA16_N,DATA16_HEAP+_DATA16_HEAP_SIZE=0200-09FF
-Z(DATA)CODE_I
-Z(DATA)CSTACK+_STACK_SIZE#


// ---------------------------------------------
// Read-only memory
//

// -----------------------------------
// Information memory
//

-Z(CONST)INFO=8000-83FF
```

```
Messages                                   File          Line
emeter-setup.c
emeter-tlv.c
metrology-calibration-defaults.c
Linking

Total number of errors: 0
Total number of warnings: 3
```

Ready   Errors 0, Warnings 3   Ln 120, Col 3

Figure 29. Warnings

10. Make sure the jumper on J8 is short properly. Connect the 14 pin connector P1 to MSP-FET430UIF by a flat cable as shown in Figure 30.

> ## WARNING
>
> **The debugging interface is NOT ISOLATED; make sure proper isolation is in place between the EVM and the PC using for debugging with AC or DC high voltage connected.**
>
> **Connection to debugging interface is optional for the operation of the EVM. The EVM can operate standalone without debugger connected.**

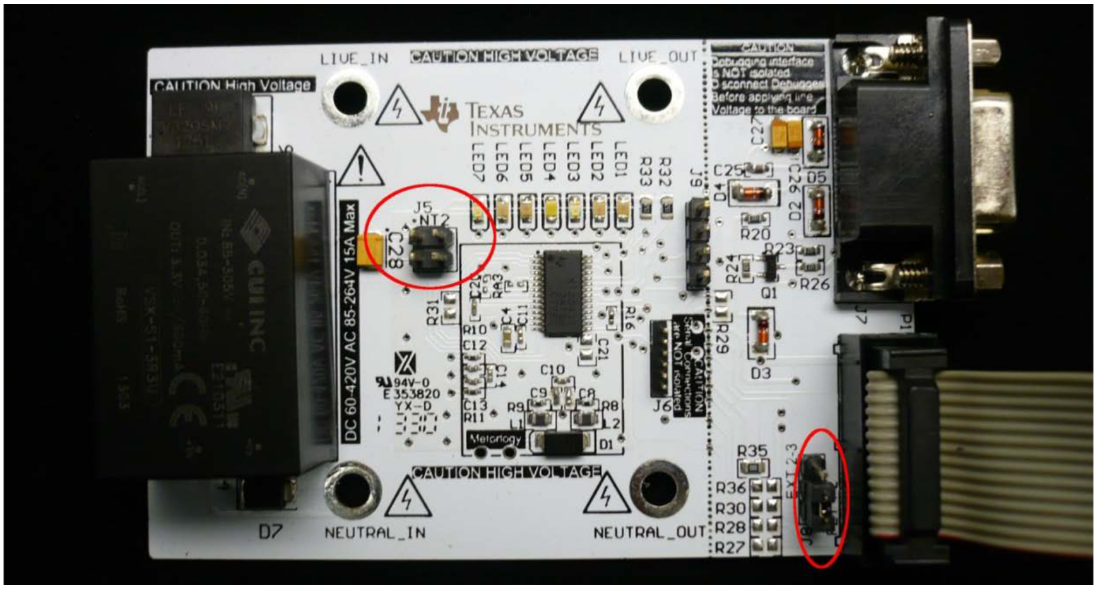
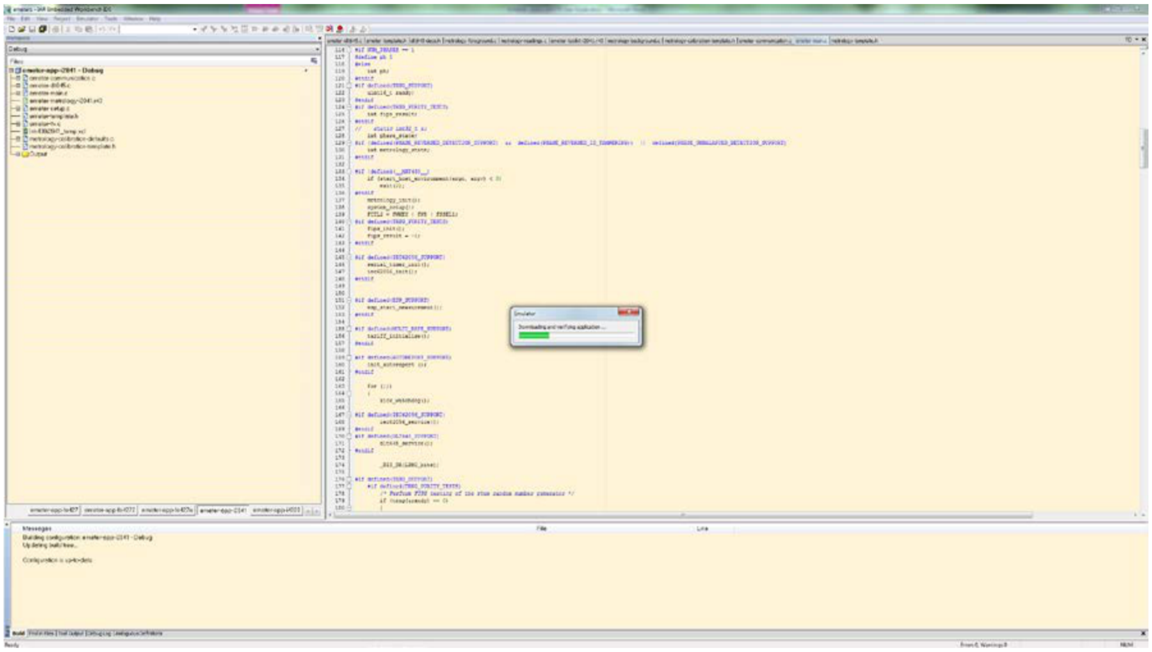


Figure 30. Connecting EVM and FET

11. Click the Download and Debug button to begin download and debugging as shown in Figure 31.



```
Messages
Building configuration: emeter-app-2041 - Debug
Updating build tree...

Configuration is up-to-date.
```

Figure 31. Code Downloading

12. After downloading is completed and successful, a screen like that shown in Figure 32 will appear. Click Go to start application running.

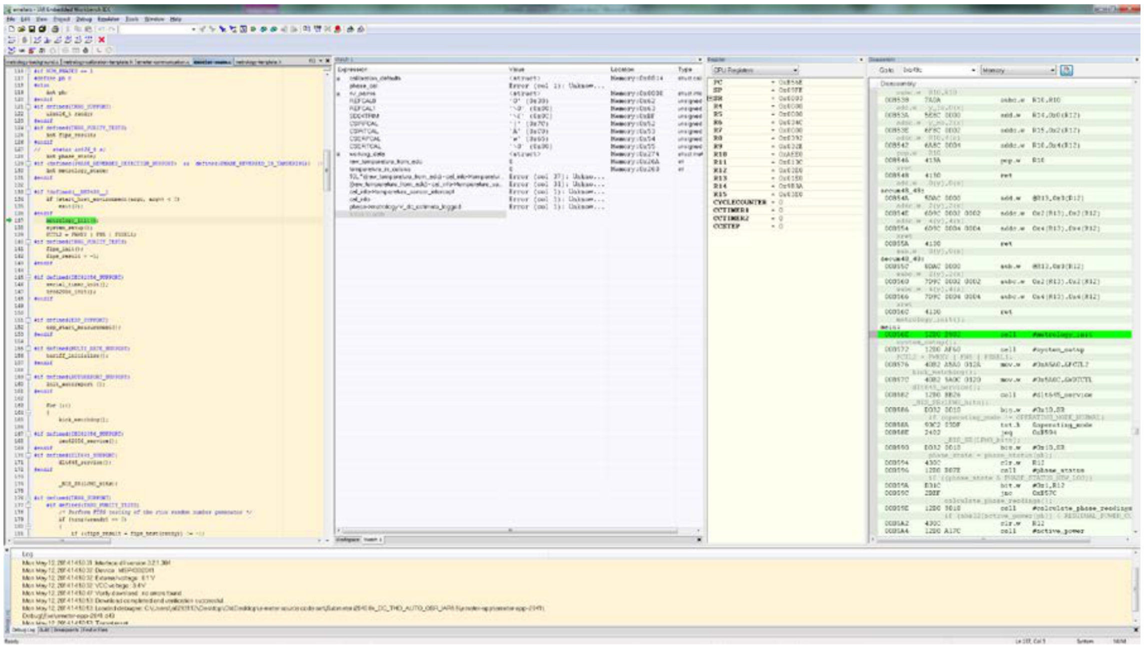


Figure 32. Debugger Screen

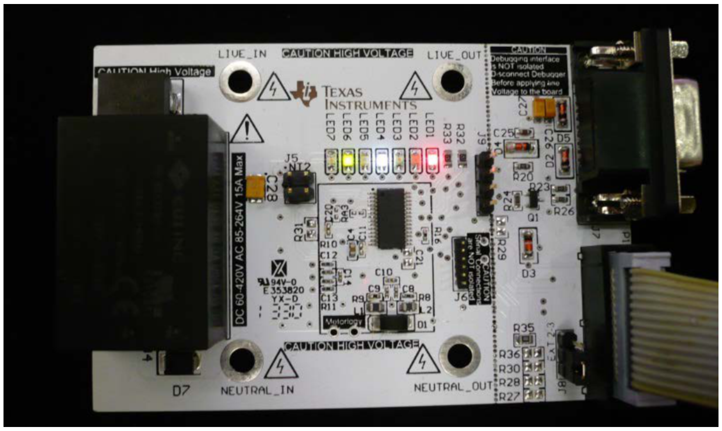
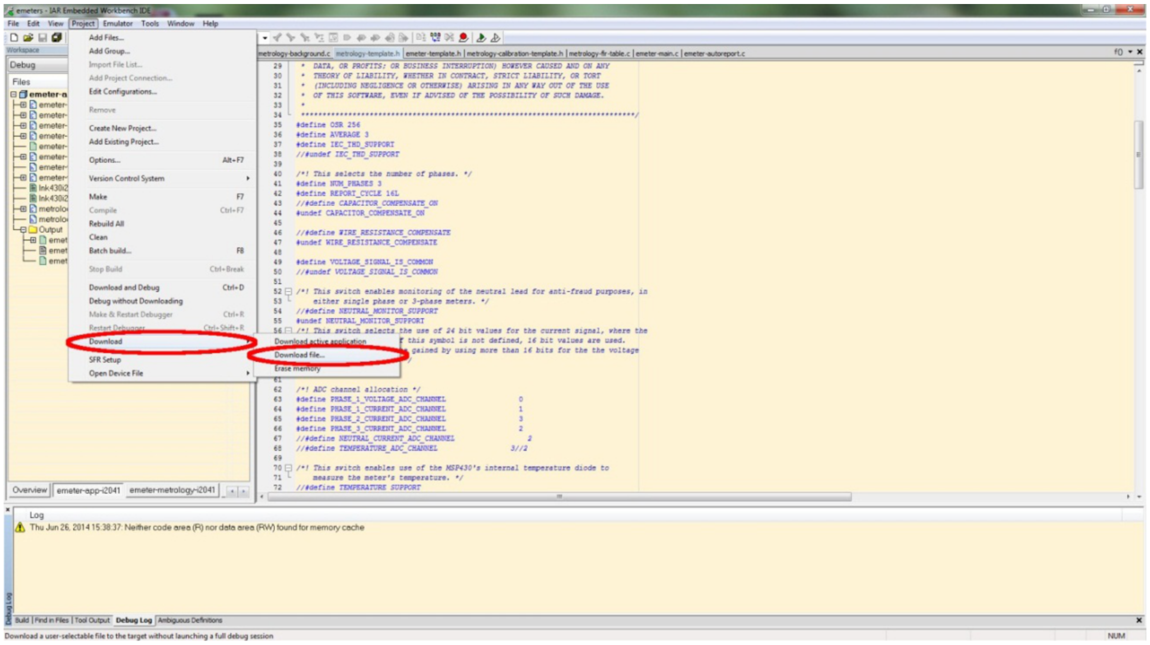


Figure 33. EVM running

A.3 Downloading Without an IAR License

Use the following steps and the installed IAR Embedded Workbench 5.5 (if a valid IAR Embedded Workbench 5.5 license is not available) to download the executable code:

1. Open the project workspace as described in Section A.2: steps 1 – 8. Connect the board to the MSP-FET430UIF as described in step 10 in Section A.2.
2. Then select Project → Download → Download File… from the menu, as shown in Figure 34.


emeters - IAR Embedded Workbench IDE
File Edit View Project Emulator Tools Window Help
Add Files...
Add Group...
Import File List...
Add Project Connection...
Edit Configurations...
Remove
Create New Project...
Add Existing Project...
Options... Alt+F7
Version Control System
Make F7
Compile Ctrl+F7
Rebuild All
Clean
Batch build... F8
Stop Build Ctrl+Break
Download and Debug Ctrl+D
Debug without Downloading
Make & Restart Debugger Ctrl+R
Restart Debugger Ctrl+Shift+R
Download
SFR Setup
Open Device File
Download active application
Download file...
Erase memory
Workspace
Debug
Files
emeter-a
Output
metrology-background.c | metrology-template.h | emeter-template.h | metrology-calibration-template.h | metrology-fir-table.c | emeter-main.c | emeter-autoreport.c
Overview | emeter-app-i2041 | emeter-metrology-i2041
Log
Build | Find in Files | Tool Output | Debug Log | Ambiguous Definitions
NUM

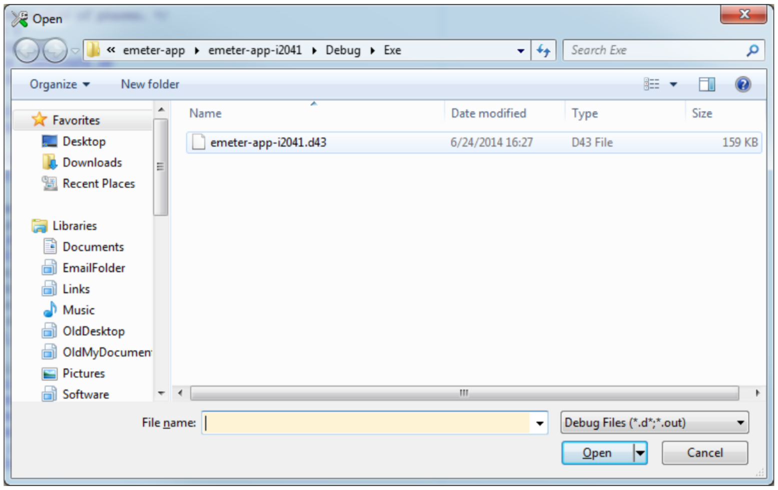
```
 *  DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 *  THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 *  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 *  OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ******************************************************************************/
#define OSR 256
#define AVERAGE 3
#define IEC_THD_SUPPORT
//#undef IEC_THD_SUPPORT

/*! This selects the number of phases. */
#define NUM_PHASES 3
#define REPORT_CYCLE 16L
//#define CAPACITOR_COMPENSATE_ON
#undef CAPACITOR_COMPENSATE_ON

//#define WIRE_RESISTANCE_COMPENSATE
#undef WIRE_RESISTANCE_COMPENSATE

#define VOLTAGE_SIGNAL_IS_COMMON
//#undef VOLTAGE_SIGNAL_IS_COMMON

/*! This switch enables monitoring of the neutral lead for anti-freud purposes, in
    either single phase or 3-phase meters. */
//#define NEUTRAL_MONITOR_SUPPORT
#undef NEUTRAL_MONITOR_SUPPORT
/*! This switch selects the use of 24 bit values for the current signal, where the
                  f this symbol is not defined, 16 bit values are used.
                  gained by using more than 16 bits for the the voltage

/*! ADC channel allocation */
#define PHASE_1_VOLTAGE_ADC_CHANNEL              0
#define PHASE_1_CURRENT_ADC_CHANNEL              1
#define PHASE_2_CURRENT_ADC_CHANNEL              3
#define PHASE_3_CURRENT_ADC_CHANNEL              2
//#define NEUTRAL_CURRENT_ADC_CHANNEL              2
//#define TEMPERATURE_ADC_CHANNEL              3//2

/*! This switch enables use of the MSP430's internal temperature diode to
    measure the meter's temperature. */
//#define TEMPERATURE_SUPPORT
```

```
Thu Jun 26, 2014 15:38:37: Neither code area (R) nor data area (RW) found for memory cache
```

Download a user-selectable file to the target without launching a full debug session

Figure 34. Download Executable File

3. When prompted for selecting the file, select from the folder Submeter i2040 4k_3_SOCK_AUTO_OSR_IAR5.5 \emeter-app\emeter-app-i2041\Debug\Exe. The file is named “emeter-app-i2041.d43”, as shown in Figure 35.



« emeter-app ▸ emeter-app-i2041 ▸ Debug ▸ Exe

Name	Date modified	Type	Size
emeter-app-i2041.d43	6/24/2014 16:27	D43 File	159 KB

Debug Files (*.d*;*.out)

Figure 35. Select File to Download

4. The executable code will be downloaded to the board.

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 is 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

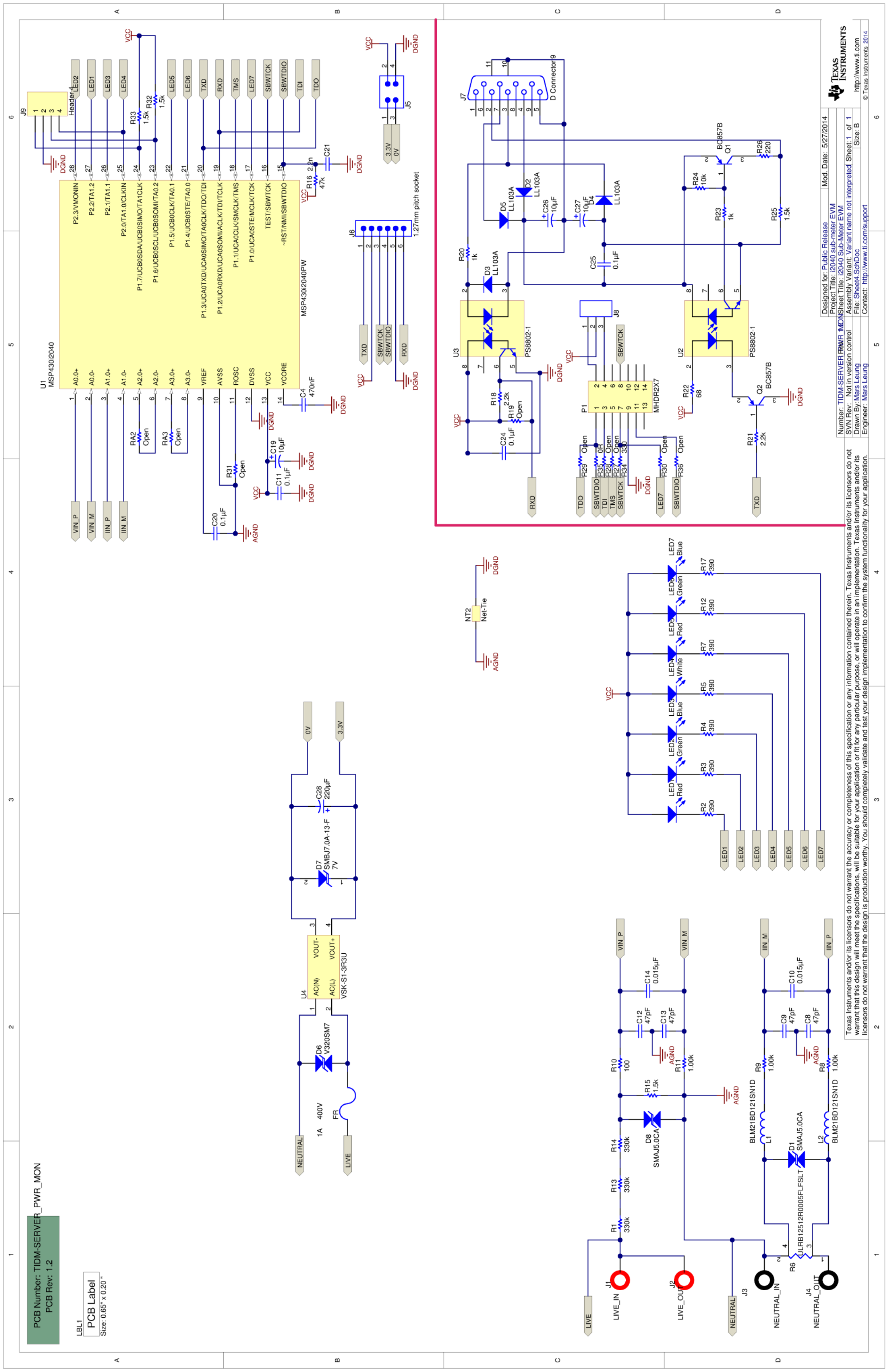

U1
MSP430I2040
MSP430I2040PW
J9
Header 4
J5
J6
1.27mm pitch socket
J7
D Connector 9
J8
P1
MHDR2X7
U2
U3
PS8802-1
Q1
Q2
BC857B
D2
D3
D4
D5
LL103A
U4
VSK-S1-3R3U
D6
V320SM7
D7
SMBJ7.0A-13-F
D1
D8
SMAJ5.0CA
C28
220µF
NT2
Net-Tie
L1
L2
BLM21BD121SN1D
R6
LRB12512R0005FLFSLT
J1
LIVE_IN
J2
LIVE_OUT
J3
NEUTRAL_IN
J4
NEUTRAL_OUT
LIVE
NEUTRAL
VIN_P
VIN_M
IIN_P
IIN_M
VCC
DGND
AGND
3.3V
0V
TXD
RXD
TDO
TDI
TMS
SBWTCK
SBWTDIO
LED1
LED2
LED3
LED4
LED5
LED6
LED7
Red
Green
Blue
White
390
PCB Number: TIDM-SERVER_PWR_MON
PCB Rev: 1.2
LBL1
PCB Label
Size: 0.65" x 0.20"
Designed for: Public Release
Mod. Date: 5/27/2014
Project Title: i2040 sub-meter EVM
Sheet Title: i2040 Sub-Meter EVM
Number: TIDM-SERVER_PWR_MON
SVN Rev: Not in version control
Drawn By: Mars Leung
Engineer: Mars Leung
Sheet: 1 of 1
Size: B
File: Sheet4.SchDoc
Contact: http://www.ti.com/support
http://www.ti.com
© Texas Instruments 2014
Texas Instruments and/or its licensors do not warrant the accuracy or completeness of this specification or any information contained therein. Texas Instruments and/or its licensors do not warrant that this design will meet the specifications, will be suitable for your application or fit for any particular purpose, or will operate in an implementation. Texas Instruments and/or its licensors do not warrant that the design is production worthy. You should completely validate and test your design implementation to confirm the system functionality for your application.

Figure 36. EVM Schematic

Appendix C EVM Specification and Performance

C.1 EVM Specifiction

- Voltage operating range (with supplied power supply): 85 – 265VAC, 120 – 380VDC
- Sample rate: 8000 Hz
- Sampling bit depth: 24 bits
- Polling report supported
- Update rate: 4 AC cycles (AC mode), 80 ms (DC mode)
- UART communication data rate: 9600 bps
- AC/DC measuring mode switching: 4 AC cycle (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 V – 265 V_{rms} AC, 0 – ±380 VDC (using reference design circuit and component values)
- Measurement current range: 0 – 15 Arms AC, 0 – ±22.5 ADC (using reference design circuit and 0.5m-Ω shunt values)
- Voltage resolution: 1mV
- Current resolution: 1 µA
- Active power resolution: 1 mW
- Reactive and apparent power resolution: 1 mW
- Power factor resolution: 0.001
- EVM performance
- Power accuracy at room temperature

PF = 1		Cal at 7.50212A, 1650.7W					
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%

PF = 0.5L		Cal at 7.50212A, 1650.7W					
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%

PF = 0.5C		Cal at 7.50212A, 1650.7W					
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%

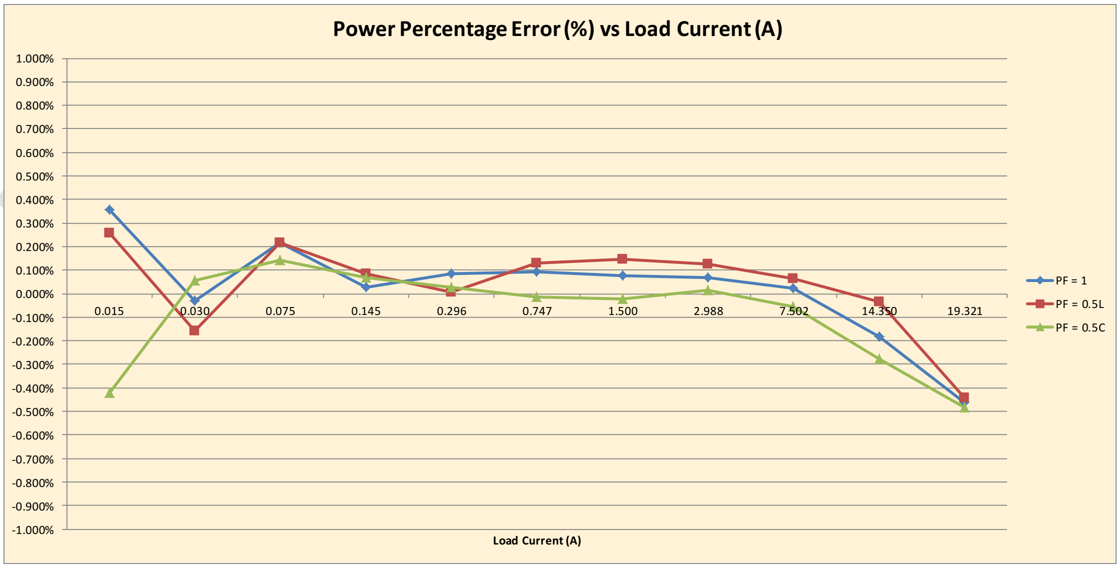


Figure 37. Power Percentage Error % Load Current (A)

NOTE: In Figure 37, the graph indicates that at a higher current the error begins to go negative. This phenomenon is due to the fact that the shunt is heated up by the current flowing through the traces on the board. At low current, the higher error is mostly caused by noise causing instability of readings.

Table 20. Accuracy vs Temperature at 220 V, 5A

	Voltage (V)			Current (A)			Power (W)		
Temp/C	**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%

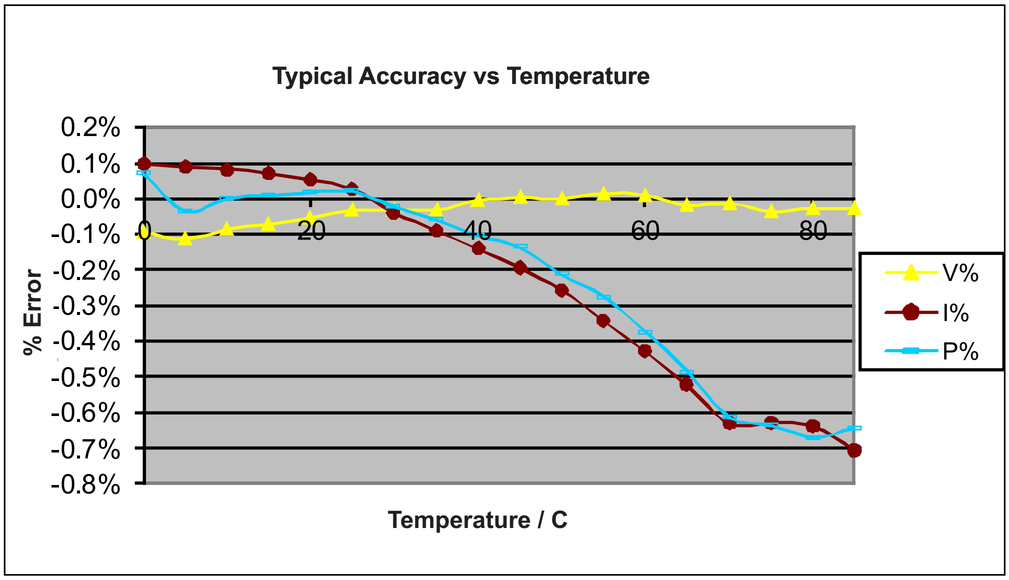


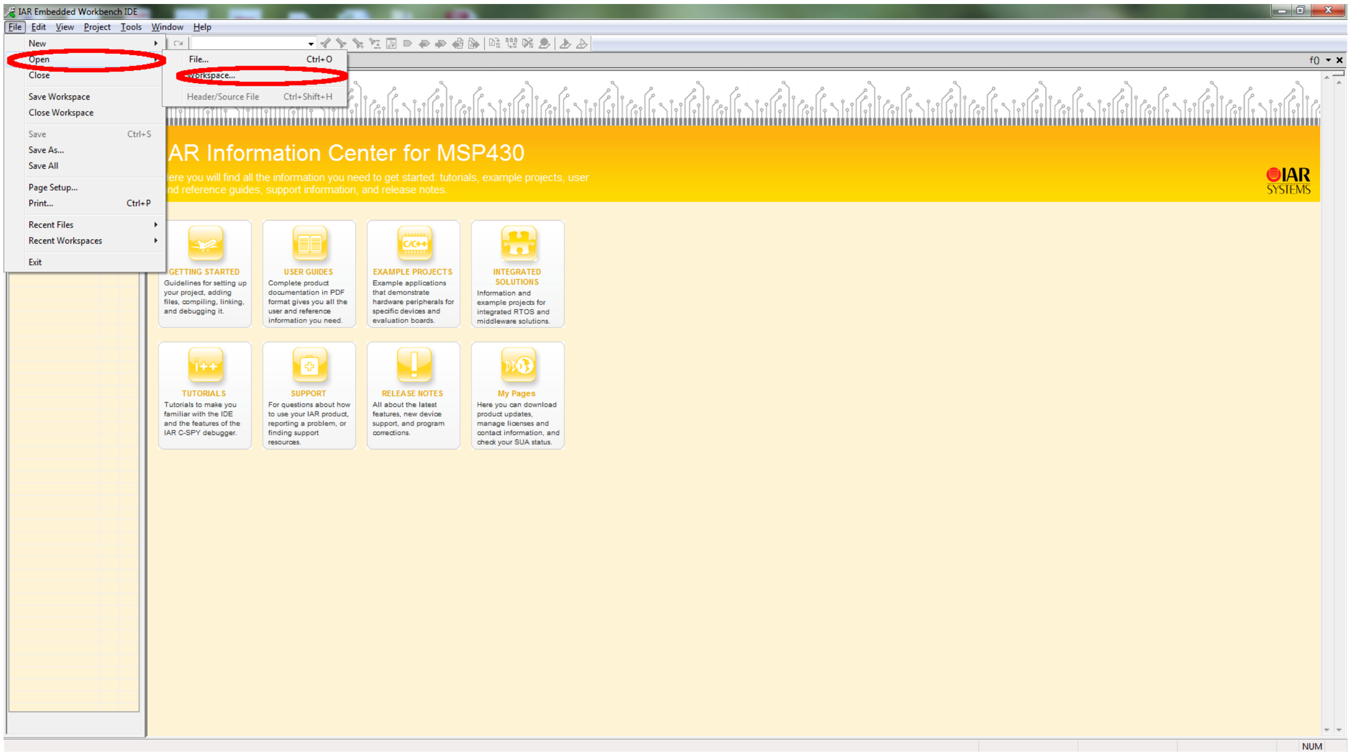
Figure 38. Typical Accuracy vs Temperature

NOTE: In Figure 38, the graph indicates that at higher temperature the error begin to go negative. The phenomenon is due to the board heated up and the combined temperature coefficient of the whole board is in effect.

Appendix D 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.*



AR Information Center for MSP430

ere you will find all the information you need to get started: tutorials, example projects, user nd reference guides, support information, and release notes.

GETTING STARTED
Guidelines for setting up your project, adding files, compiling, linking, and debugging it.

USER GUIDES
Complete product documentation in PDF format gives you all the user and reference information you need.

EXAMPLE PROJECTS
Example applications that demonstrate hardware peripherals for specific devices and evaluation boards.

INTEGRATED SOLUTIONS
Information and example projects for integrated RTOS and middleware solutions.

TUTORIALS
Tutorials to make you familiar with the IDE and the features of the IAR C-SPY debugger.

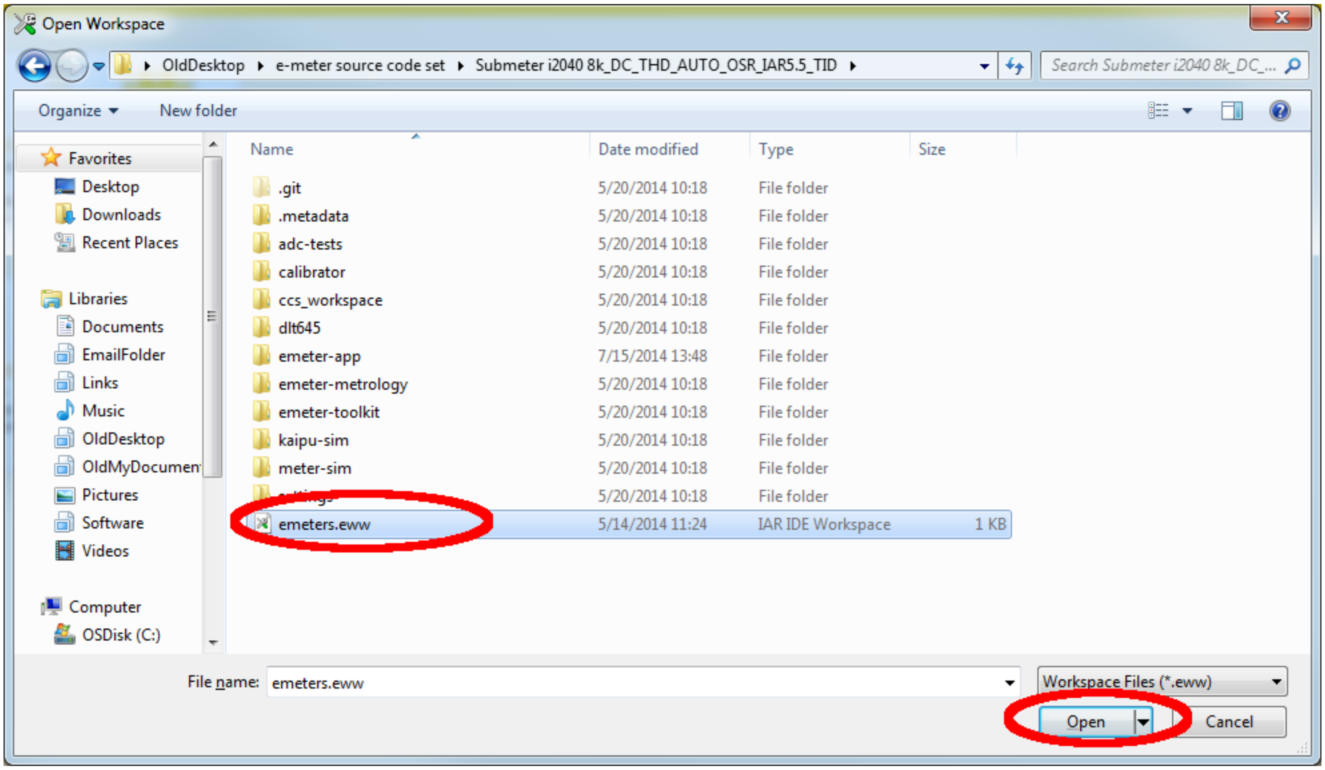
SUPPORT
For questions about how to use your IAR product, reporting a problem, or finding support resources.

RELEASE NOTES
All about the latest features, new device support, and program corrections.

My Pages
Here you can download product updates, manage licenses and contact information, and check your SUA status.

Figure 39. IAR 5.5 Launch Window

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



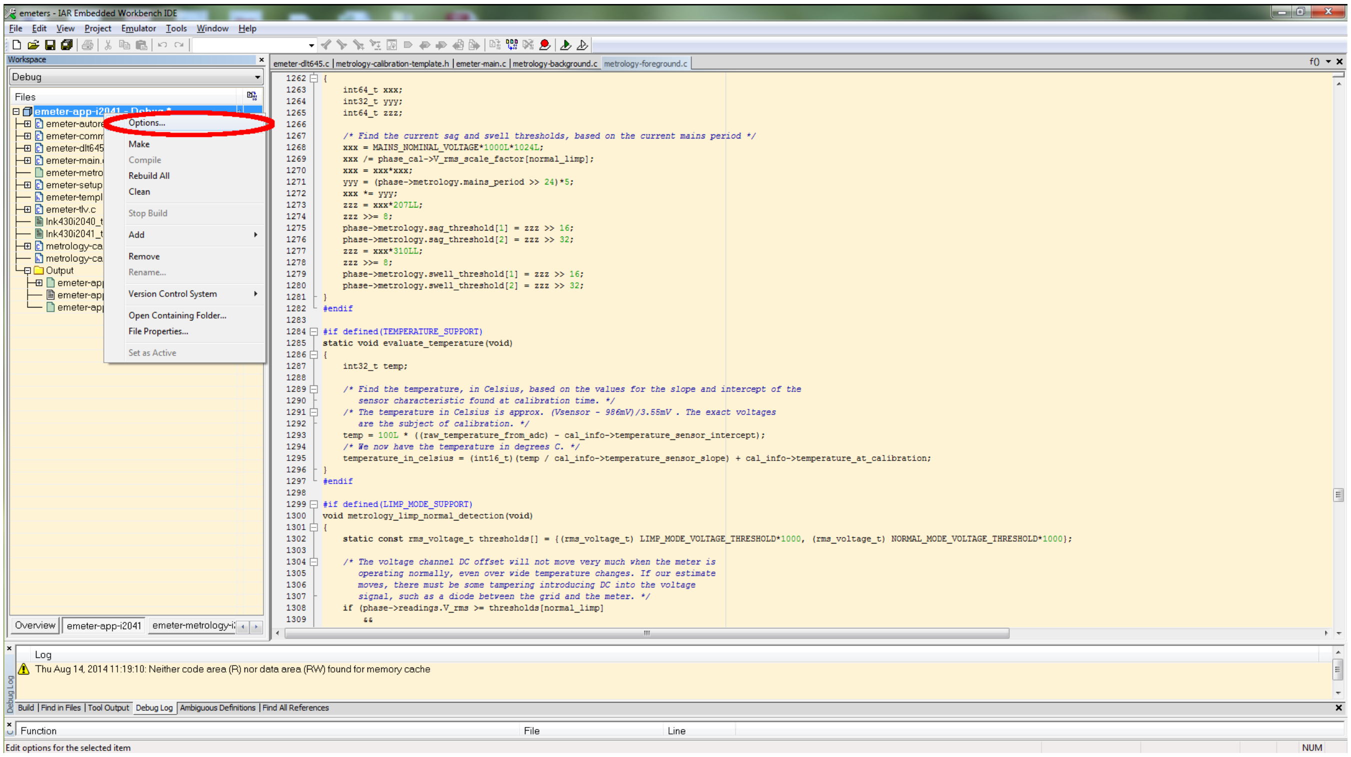
Name	Date modified	Type	Size
.git	5/20/2014 10:18	File folder	
.metadata	5/20/2014 10:18	File folder	
adc-tests	5/20/2014 10:18	File folder	
calibrator	5/20/2014 10:18	File folder	
ccs_workspace	5/20/2014 10:18	File folder	
dlt645	5/20/2014 10:18	File folder	
emeter-app	7/15/2014 13:48	File folder	
emeter-metrology	5/20/2014 10:18	File folder	
emeter-toolkit	5/20/2014 10:18	File folder	
kaipu-sim	5/20/2014 10:18	File folder	
meter-sim	5/20/2014 10:18	File folder	
settings	5/20/2014 10:18	File folder	
emeters.eww	5/14/2014 11:24	IAR IDE Workspace	1 KB

File name: emeters.eww

Workspace Files (*.eww)

Figure 40. Open Workspace Window

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



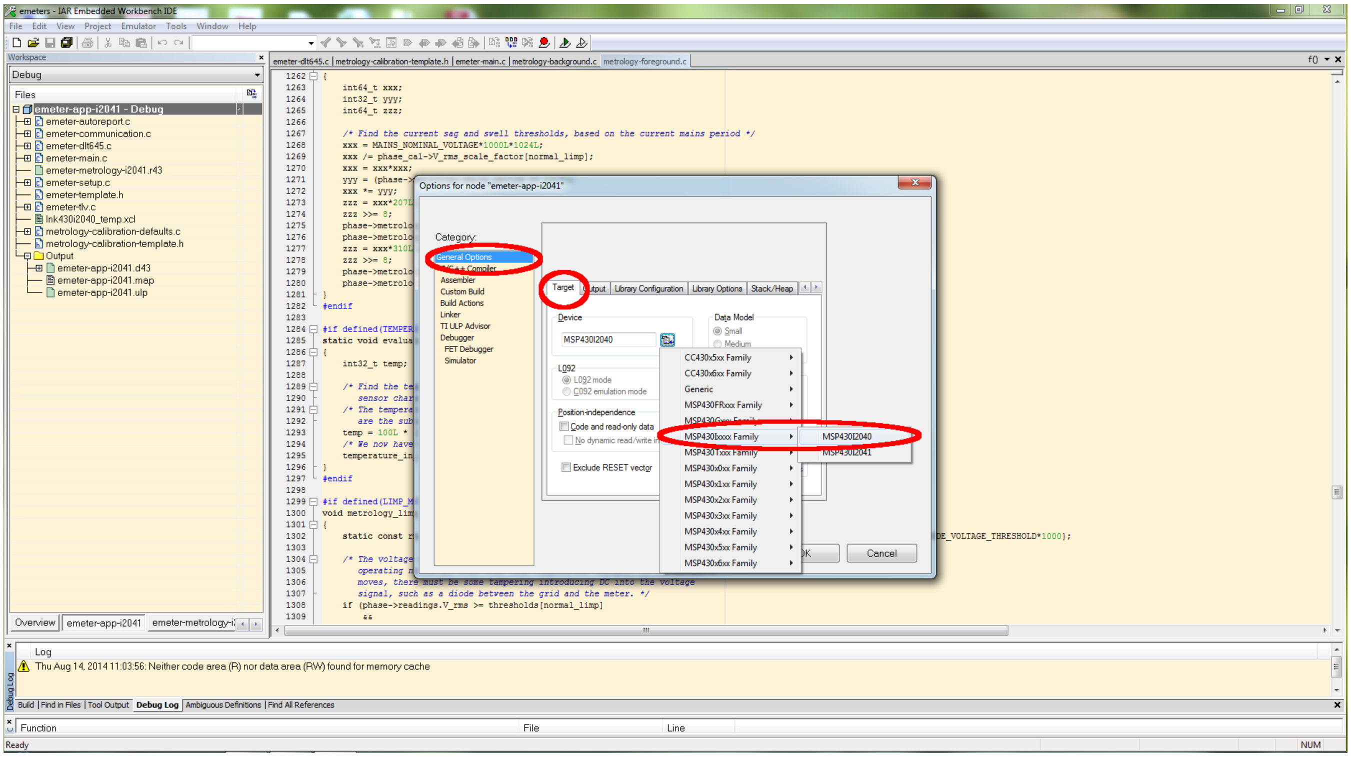
```
{
    int64_t xxx;
    int32_t yyy;
    int64_t zzz;

    /* Find the current sag and swell thresholds, based on the current mains period */
    xxx = MAINS_NOMINAL_VOLTAGE*1000L*1024L;
    xxx /= phase_cal->V_rms_scale_factor[normal_limp];
    xxx = xxx*xxx;
    yyy = (phase->metrology.mains_period >> 24)*5;
    xxx *= yyy;
    zzz = xxx*207LL;
    zzz >>= 8;
    phase->metrology.sag_threshold[1] = zzz >> 16;
    phase->metrology.sag_threshold[2] = zzz >> 32;
    zzz = xxx*310LL;
    zzz >>= 8;
    phase->metrology.swell_threshold[1] = zzz >> 16;
    phase->metrology.swell_threshold[2] = zzz >> 32;
}
#endif

#if defined(TEMPERATURE_SUPPORT)
static void evaluate_temperature(void)
{
    int32_t temp;

    /* Find the temperature, in Celsius, based on the values for the slope and intercept of the
       sensor characteristic found at calibration time. */
    /* The temperature in Celsius is approx. (Vsensor - 986mV)/3.55mV . The exact voltages
       are the subject of calibration. */
    temp = 100L * ((raw_temperature_from_adc) - cal_info->temperature_sensor_intercept);
    /* We now have the temperature in degrees C. */
    temperature_in_celsius = (int16_t)(temp / cal_info->temperature_sensor_slope) + cal_info->temperature_at_calibration;
}
#endif

#if defined(LIMP_MODE_SUPPORT)
void metrology_limp_normal_detection(void)
{
    static const rms_voltage_t thresholds[] = {(rms_voltage_t) LIMP_MODE_VOLTAGE_THRESHOLD*1000, (rms_voltage_t) NORMAL_MODE_VOLTAGE_THRESHOLD*1000};

    /* The voltage channel DC offset will not move very much when the meter is
       operating normally, even over wide temperature changes. If our estimate
       moves, there must be some tampering introducing DC into the voltage
       signal, such as a diode between the grid and the meter. */
    if (phase->readings.V_rms >= thresholds[normal_limp]
        &&
```

```
Thu Aug 14, 2014 11:19:10: Neither code area (R) nor data area (RW) found for memory cache
```

Edit options for the selected item

Figure 41. 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.*



```
{
    int64_t xxx;
    int32_t yyy;
    int64_t zzz;

    /* Find the current sag and swell thresholds, based on the current mains period */
    xxx = MAINS_NOMINAL_VOLTAGE*1000L*1024L;
    xxx /= phase_cal->V_rms_scale_factor[normal_limp];
    xxx = xxx*xxx;
    yyy = (phase->
    xxx *= yyy;
    zzz = xxx*207L
    zzz >>= 8;
    phase->metrolo
    phase->metrolo
    zzz = xxx*310L
    zzz >>= 8;
    phase->metrolo
    phase->metrolo
}
#endif

#if defined(TEMPER
static void evalua
{
    int32_t temp;

    /* Find the te
       sensor char
    /* The tempera
       are the sub
    temp = 100L *
    /* We now have
    temperature_in
}
#endif

#if defined(LIMP_M
void metrology_lim
{
    static const r                                                            DE_VOLTAGE_THRESHOLD*1000];

    /* The voltage
       operating n
       moves, there must be some tampering introducing DC into the voltage
       signal, such as a diode between the grid and the meter. */
    if (phase->readings.V_rms >= thresholds[normal_limp]
        &&
```

Options for node "emeter-app-i2041"

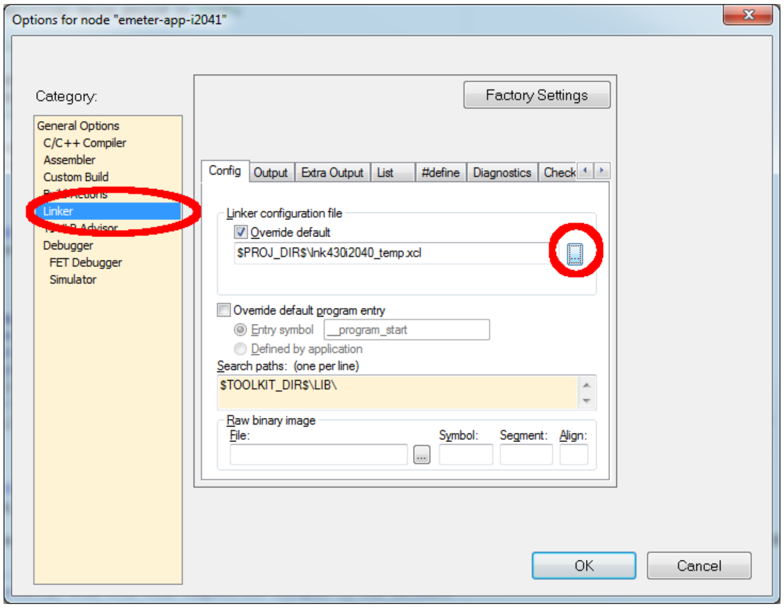
Category: General Options

Target tab — Device: MSP430I2040 — MSP430Ixxxx Family → MSP430I2040

Log: Thu Aug 14, 2014 11:03:56: Neither code area (R) nor data area (RW) found for memory cache

Figure 42. 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.



Options for node "emeter-app-i2041"

Category: Linker

Config tab

Linker configuration file

☑ Override default

$PROJ_DIR$\lnk430i2040_temp.xcl

☐ Override default program entry

○ Entry symbol __program_start

○ Defined by application

Search paths: (one per line)

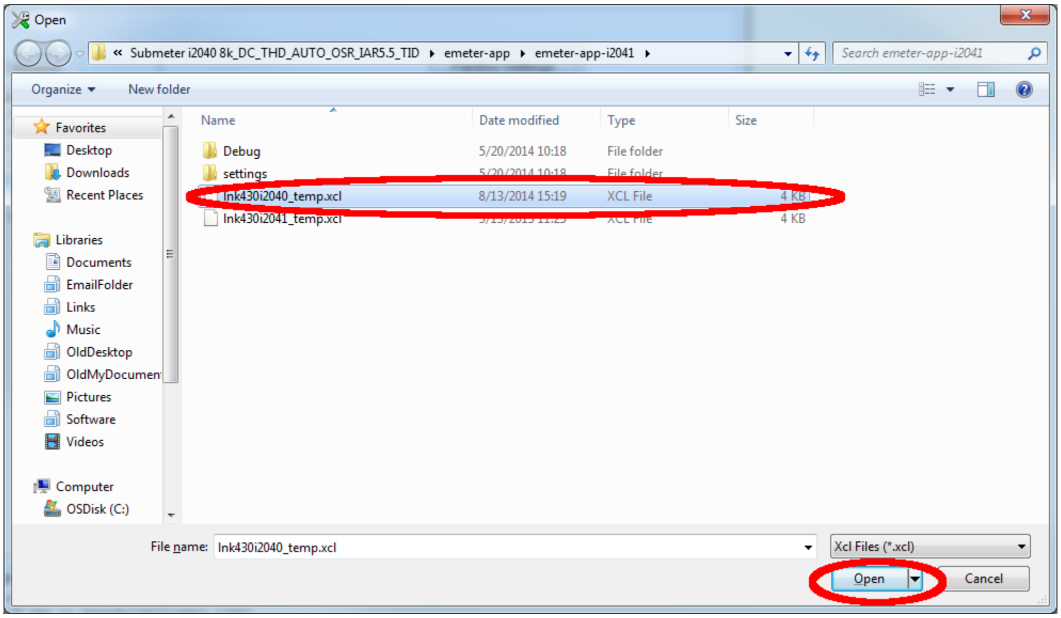
$TOOLKIT_DIR$\LIB\

Raw binary image

File: Symbol: Segment: Align:

Figure 43. Config Tab

Select *lnk430i2040_temp.xcl* when prompted and click *Open.*



Open

« Submeter i2040 8k_DC_THD_AUTO_OSR_IAR5.5_TID ▸ emeter-app ▸ emeter-app-i2041 ▸

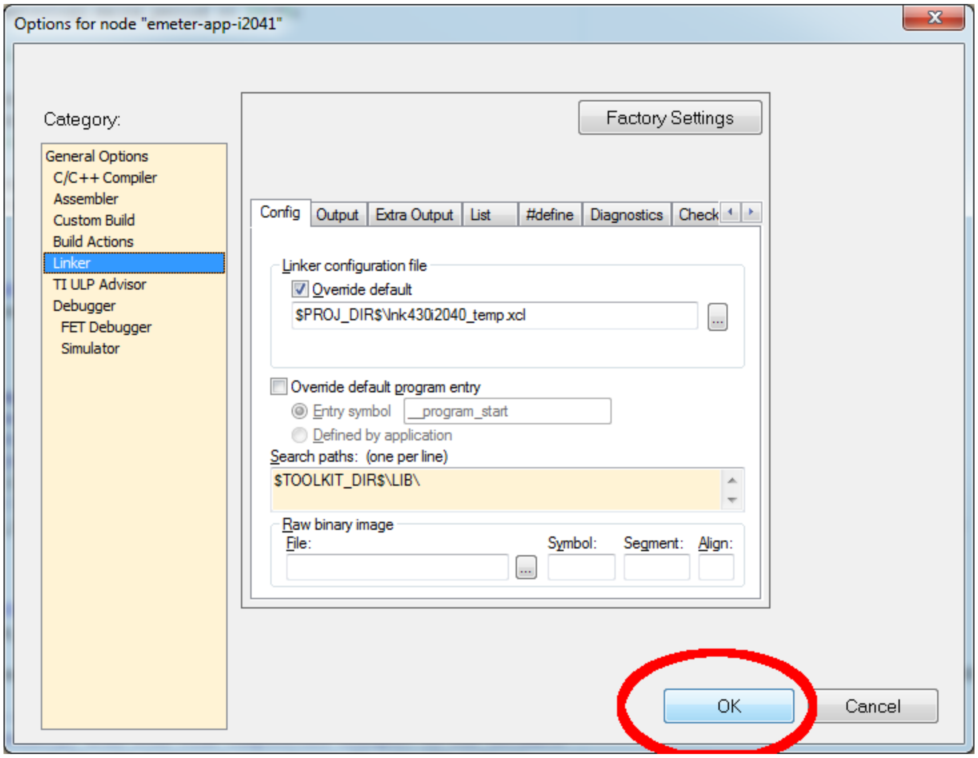
Name	Date modified	Type	Size
Debug	5/20/2014 10:18	File folder	
settings	5/20/2014 10:18	File folder	
lnk430i2040_temp.xcl	8/13/2014 15:19	XCL File	4 KB
lnk430i2041_temp.xcl	5/13/2015 11:23	XCL File	4 KB

File name: lnk430i2040_temp.xcl

Xcl Files (*.xcl)

Figure 44. Selecting Default XCL File

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



Options for node "emeter-app-i2041"

Category: Linker

Config

Linker configuration file

☑ Override default

$PROJ_DIR$\lnk430i2040_temp.xcl

☐ Override default program entry

○ Entry symbol __program_start

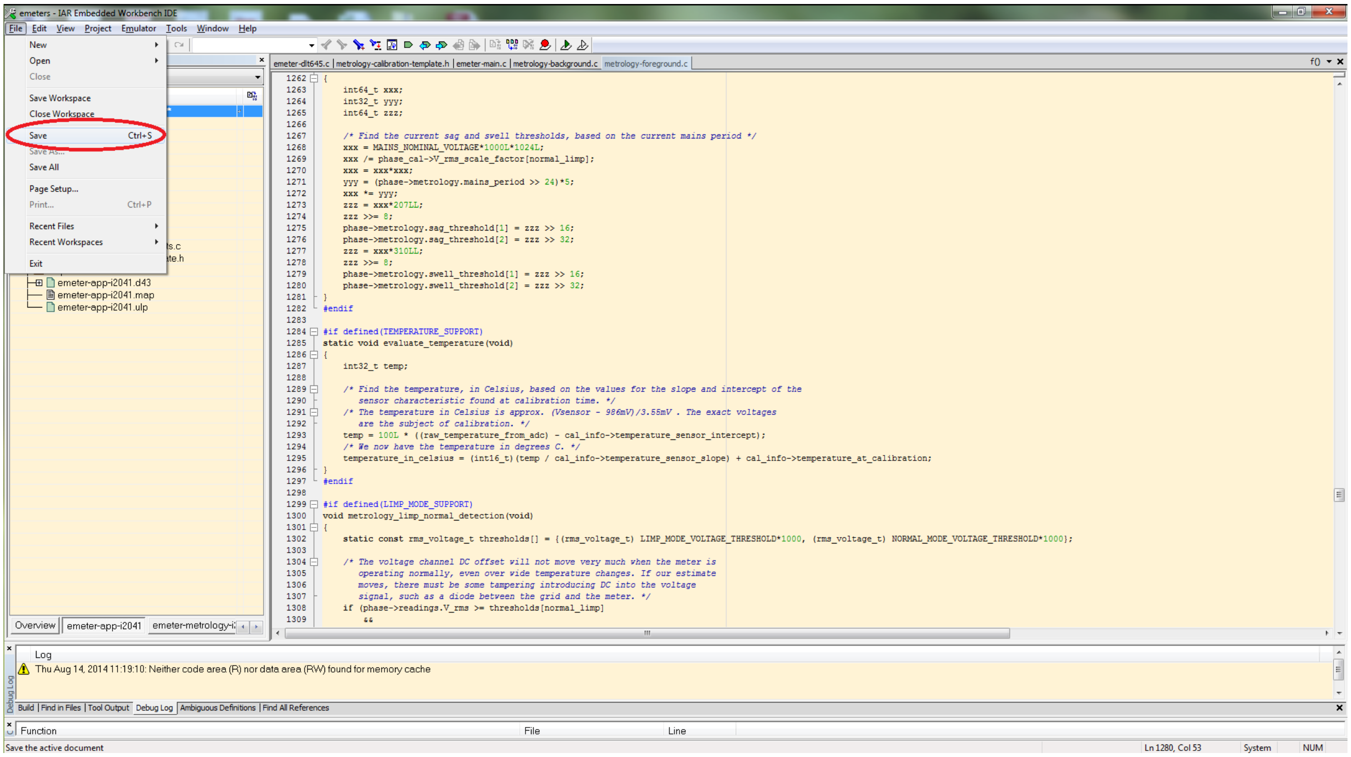
○ Defined by application

Search paths: (one per line)

$TOOLKIT_DIR$\LIB\

Figure 45. Finishing the Setting Changes

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



```
{
    int64_t xxx;
    int32_t yyy;
    int64_t zzz;

    /* Find the current sag and swell thresholds, based on the current mains period */
    xxx = MAINS_NOMINAL_VOLTAGE*1000L*1024L;
    xxx /= phase_cal->V_rms_scale_factor[normal_limp];
    xxx = xxx*xxx;
    yyy = (phase->metrology.mains_period >> 24)*5;
    xxx *= yyy;
    zzz = xxx*207LL;
    zzz >>= 8;
    phase->metrology.sag_threshold[1] = zzz >> 16;
    phase->metrology.sag_threshold[2] = zzz >> 32;
    zzz = xxx*310LL;
    zzz >>= 8;
    phase->metrology.swell_threshold[1] = zzz >> 16;
    phase->metrology.swell_threshold[2] = zzz >> 32;
}
#endif

#if defined(TEMPERATURE_SUPPORT)
static void evaluate_temperature(void)
{
    int32_t temp;

    /* Find the temperature, in Celsius, based on the values for the slope and intercept of the
       sensor characteristic found at calibration time. */
    /* The temperature in Celsius is approx. (Vsensor - 986mV)/3.55mV . The exact voltages
       are the subject of calibration. */
    temp = 100L * ((raw_temperature_from_adc) - cal_info->temperature_sensor_intercept);
    /* We now have the temperature in degrees C. */
    temperature_in_celsius = (int16_t)(temp / cal_info->temperature_sensor_slope) + cal_info->temperature_at_calibration;
}
#endif

#if defined(LIMP_MODE_SUPPORT)
void metrology_limp_normal_detection(void)
{
    static const rms_voltage_t thresholds[] = {(rms_voltage_t) LIMP_MODE_VOLTAGE_THRESHOLD*1000, (rms_voltage_t) NORMAL_MODE_VOLTAGE_THRESHOLD*1000};

    /* The voltage channel DC offset will not move very much when the meter is
       operating normally, even over wide temperature changes. If our estimate
       moves, there must be some tampering introducing DC into the voltage
       signal, such as a diode between the grid and the meter. */
    if (phase->readings.V_rms >= thresholds[normal_limp]
        &&
```

```
Log
Thu Aug 14, 2014 11:19:10: Neither code area (R) nor data area (RW) found for memory cache
```

Figure 46. 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 MSP430Ixxxx Family→MSP430I2041 in the Options window and select *lnk430i2041_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		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265