Design Guide
SLAA580–February 2014

Multi-Phase Rogowski-Based E-Meter

Bart Basile *MCU Systems Application Europe*

ABSTRACT

This application report describes the implementation of a three-phase electronic electricity meter using the Texas Instruments MSP430F47197 system-on-chip (SOC) processor together with Rogowski-coils as sensors. It includes the necessary information with regard to the software and hardware solution used and details the calibration procedure for it.

WARNING

Failure to adhere to these steps and/or not heed the safety requirements at each step may lead to shock, injury, and damage to the hardware. Texas Instruments is not responsible or liable in any way for shock, injury, or damage caused due to negligence or failure to heed advice.

WARNING

EXPORT NOTICE:

Recipient agrees to not knowingly export or re-export, directly or indirectly, any product or technical data (as defined by the U.S., EU, and other Export Administration Regulations) including software, or any controlled product restricted by other applicable national regulations, received from Disclosing party under this Agreement, or any direct product of such technology, to any destination to which such export or re-export is restricted or prohibited by U.S. or other applicable laws, without obtaining prior authorization from U.S. Department of Commerce and other competent Government authorities to the extent required by those laws. This provision shall survive termination or expiration of this Agreement. According to our best knowledge of the state and end-use of this product or technology, and in compliance with the export control regulations of dual-use goods in force in the origin and exporting countries, this technology is classified as follows:

- US ECCN: 3E991
- EU ECCN: EAR99

And may require export or re-export license for shipping it in compliance with the applicable regulations of certain countries.

Contents

List of Figures

List of Tables

1 Introduction

The MSP430F471xx devices belong to the MSP430F4xx family of devices. These devices find its application in energy measurement and have the necessary architecture to support it. The MSP430F471xx devices have a powerful 16 MHz CPU with MSP430CPUx architecture. The analog front end consists of up to seven analog to digital converters (ADC) based on a second order sigma-delta architecture that supports differential inputs. The sigma-delta ADCs (SD16) that have a resolution of 16-bits can be configured and grouped together for simultaneous sampling of voltages and currents on the same trigger. Each SD16 supports a common mode voltage of up to -1 V and enables all sensors to be referenced to ground. In addition, it also has an integrated gain stage to support gains up to 32 for amplification of low-output sensors. A 32-bit x 32-bit HW multiplier on this chip can be used to further accelerate math intensive operations during energy computation. The software supports calculation of various parameters for total three phase and for each individual phases. The key parameters calculated during energy measurements are: RMS current and voltage, active and reactive energy and frequency. *Implementation of a Three-Phase Electronic Watt-Hour Meter Using MSP430F471xx* (SLAA409) has complete metrology source code provided as a zip file.

2 Block Diagram

Figure 1 depicts the block diagram that shows the high level interface used for a three-phase energy meter application. A three-phase four wire star connection to the mains is shown in this case. Rogowski-coils are connected to each of the current channels and a simple voltage divider is used for corresponding voltages. The choice of the Rogowski-coil and the necessary settings is made based on the manufacturer and current range required for energy measurements. The choice of voltage divider resistors for each voltage channel is selected to ensure the mains voltage is divided down to adhere to the normal input ranges that are valid for the MSP430 SD16. For these numbers, see the *MSP430F471x3, MSP430F471x6, MSP430F471x7 Mixed Signal Microcontroller Data Sheet* (SLAS626) and the *MSP430x4xx Family User's Guide* (SLAU056).

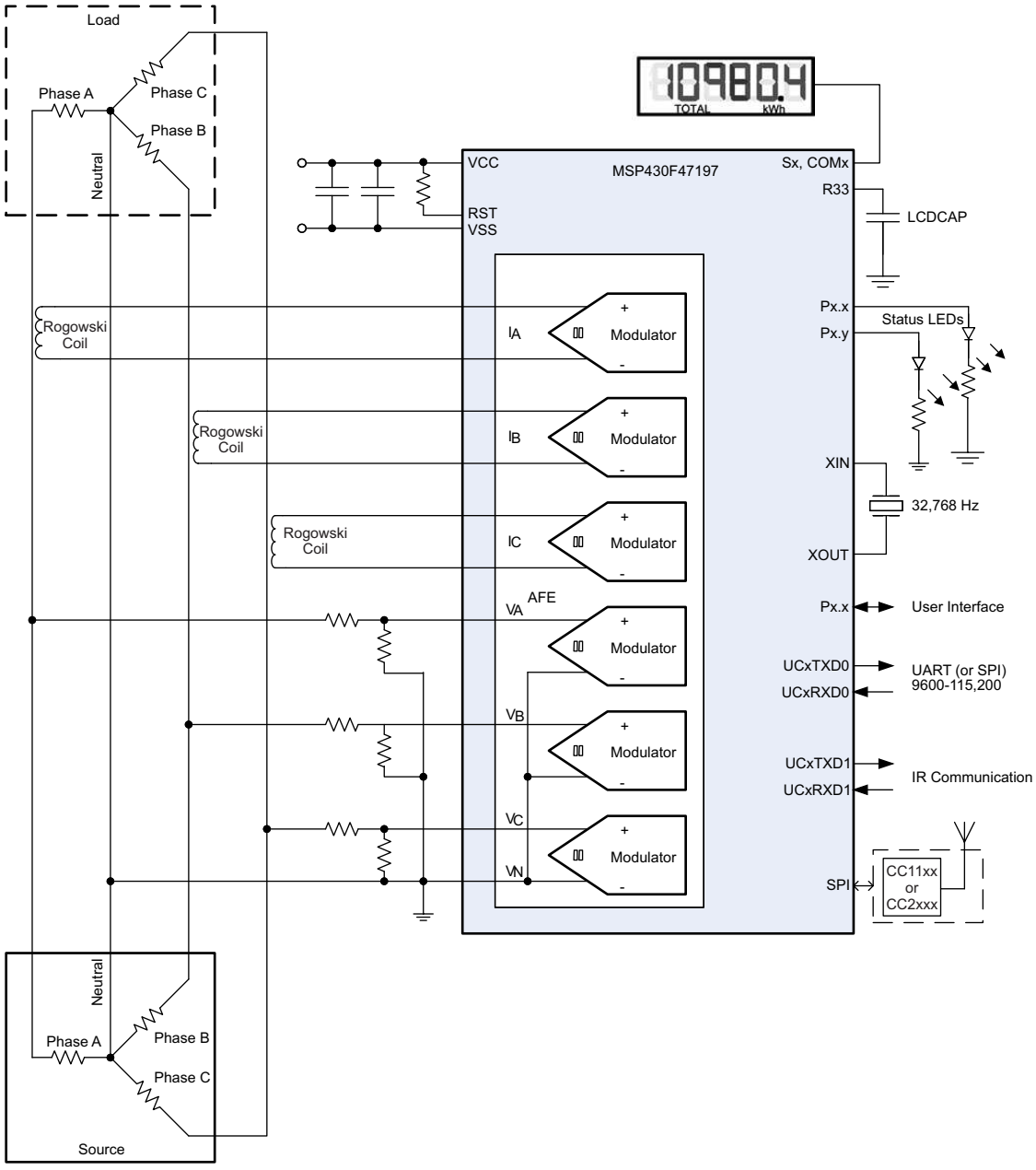


Figure 1. Three-Phase 4-Wire Star Connection Using the MSP430F47197

3 Hardware Implementation

This section describes various pieces that constitute the hardware for the working of an energy meter using the MSP430F47197.

3.1 Power Supply

The MSP430 is a family of ultra-low-power microcontrollers from Texas Instruments. These devices support a number of low-power modes and also have low-power consumption during active mode when the CPU and other peripherals are active. The low-power feature of this device family allows design of the power supply to be simple and inexpensive. The power supply allows the operation of the energy meter powered directly from the mains.

Figure 2 shows a simple capacitor power supply for a single output voltage of 3.3 V directly from the mains.

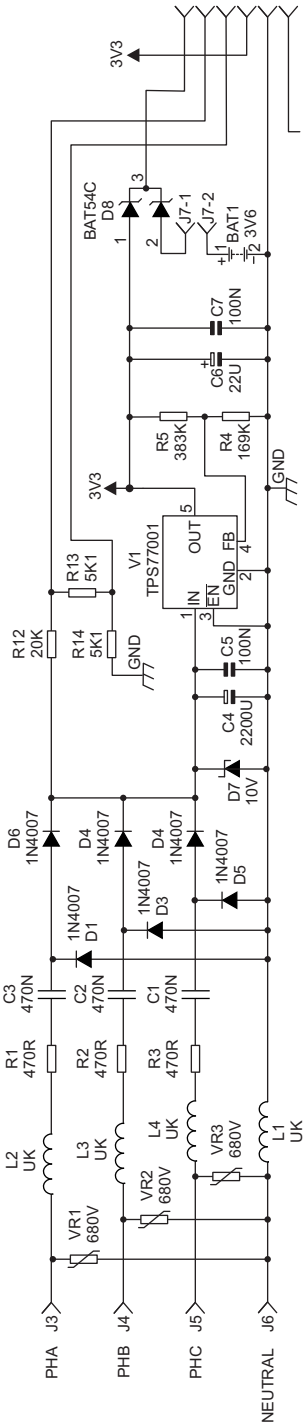


Figure 2. Simple Capacitive Power Supply for the MSP430 Energy Meter

Appropriate values of resistors (R1-R3) and capacitors (C3-C1) are chosen based on the required output current drive of the power supply. Voltage from mains is directly fed to a RC based circuit followed by a rectification circuitry to provide a DC voltage for the operation of the MSP430. This DC voltage is regulated to 3.3V for full speed operation of the MSP430. For the circuit above, the approximate drive provided by each phase is about 12mA. The design equations for the power supply are given in the *Capacitor Power Supplies* section of the *MSP430 Family Mixed-Signal Microcontroller Application Reports* (SLAA024) from Texas Instruments. The above configuration allows for all three phases to provide the required current drive, which would then be three times the drive available from each phase. If a need for additional drive is required, either an NPN output buffer or a transformer based power supply may be used.

3.2 Analog Inputs

The MSP430 analog front end that consists of the SD16 ADC is differential and requires that the input voltages at the pins do not exceed ±500 mV (gain = 1). To meet this specification, the current and voltage inputs must be divided down. In addition, the SD16 allows a maximum negative voltage of -1 V; therefore, the AC signals from the mains can be directly interfaced without the need for level shifters. The following sections describe the analog front end used for voltage and current channels.

3.2.1 Voltage

The voltage from the mains is usually 230 V or 110 V and needs to be brought down to a range of 500 mV. The analog front end for voltage consists of spike protection varistors followed by a simple voltage divider and a RC low-pass filter that acts like an anti-alias filter.

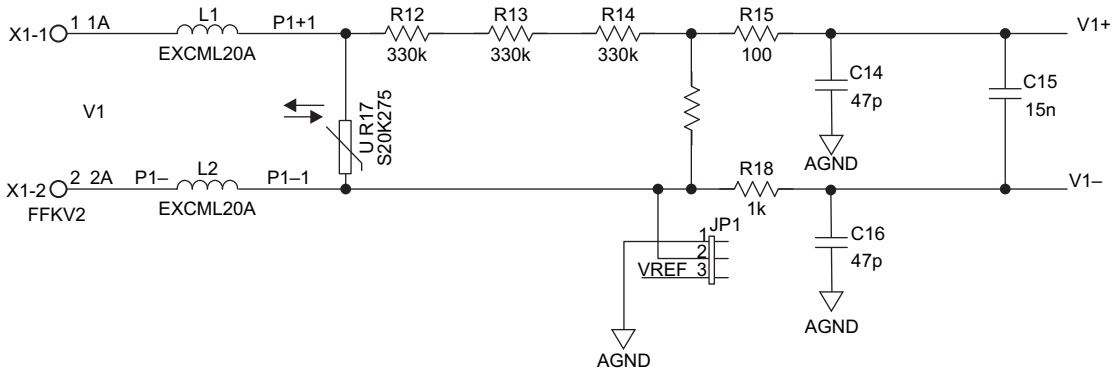


Figure 3. Analog Front End for Voltage Inputs

Figure 3 shows the analog front end for the voltage inputs for a mains voltage of 230 V. The voltage is brought down to approximately 350 mV RMS, which is 495 mV peak and fed to the positive input, adhering to the MSP430 SD16 analog limits. A common mode voltage of zero can be connected to the negative input of the SD16. In addition, the SD16 has an internal reference voltage of 1.2 V that can be used externally and also as a common mode voltage if needed.

It is important to note that the anti-alias resistors on the positive and negative sides are different because, the input impedance to the positive terminal is much higher and therefore a lower value resistor is used for the anti-alias filter. If this is not maintained, a relatively large phase shift of several degrees would result.

3.2.2 Current

The analog front-end for the current inputs is different from the analog front-end for the voltage inputs.

The current inputs are based on an implementation of Sentec's patented Mobius technology and shares the printed circuit board (PCB) with a capacitive dropper power supply. Care should be taken with soft magnetic materials in close proximity to the bus bars, for example nickel end caps on resistors and steel fixing bolts, as these will deform the magnetic field being measured. Non-magnetic equivalents should be used, such as brass bolts.

Figure 4 shows the analog front end used for the current channels following the Rogowski-sensor used. It was designed in a way to make a good second order anti-alias filter.

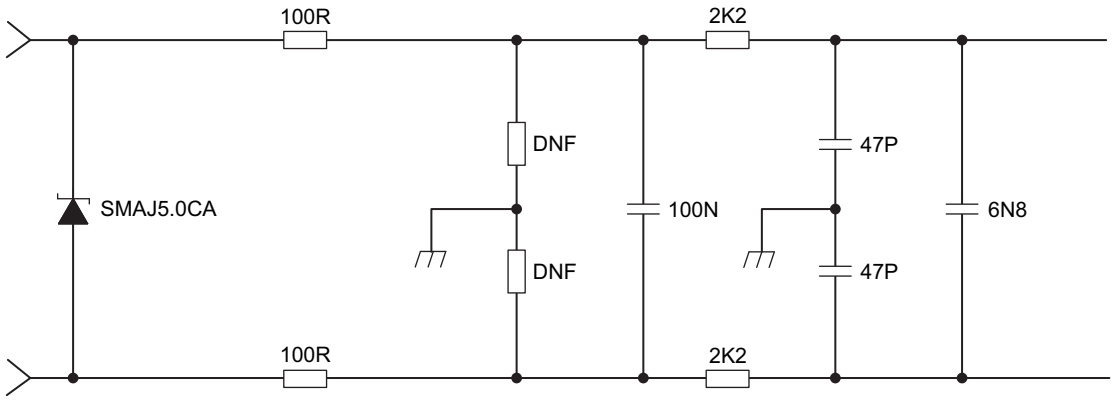


Figure 4. Analog Front End for Current inputs

4 Software Implementation

The software for the three-phase metrology using Rogowski coils is discussed in this section. The application programmer's interface (API) functions are described together with their function parameters and their return values.

The software itself is supplied as a library, which exposes a clearly defined API with C prototyped function entry points, and which can be linked against application code, making software development easy. The following functionality is accessible using these API calls:

- Configuration for run time adapting the library (calibration constants to include gain, phase, and zero offset trim, other run time options such as choice of test pulse output)
- Event capture for easy application interfacing (low line voltage, over current, reverse current)
- Signal processing chain hooks for advanced users (intermediate calculation values)

4.1 High Level Architecture of the Software

Figure 5 describes the basic signal flow through the software. Voltage samples acquired by the SD16 are passing through two high-pass filters before being corrected for the signal path delay. Current samples pass through a single high-pass filter for DC-removal before being integrated and passed on through an additional high-pass filter, which compensates for the cancellation of the pole of the first high-pass filter in the integrator. After that, the filtered voltage and current samples are used to calculate the active and reactive energy. They also can be accessed by the API calls. For the calculation of the reactive energy, the voltage samples are shifted by 90° before being multiplied with the current samples.

For both paths, the calibration values for offset and gain are applied and the results are finally scaled to the units published in the API documentation below.

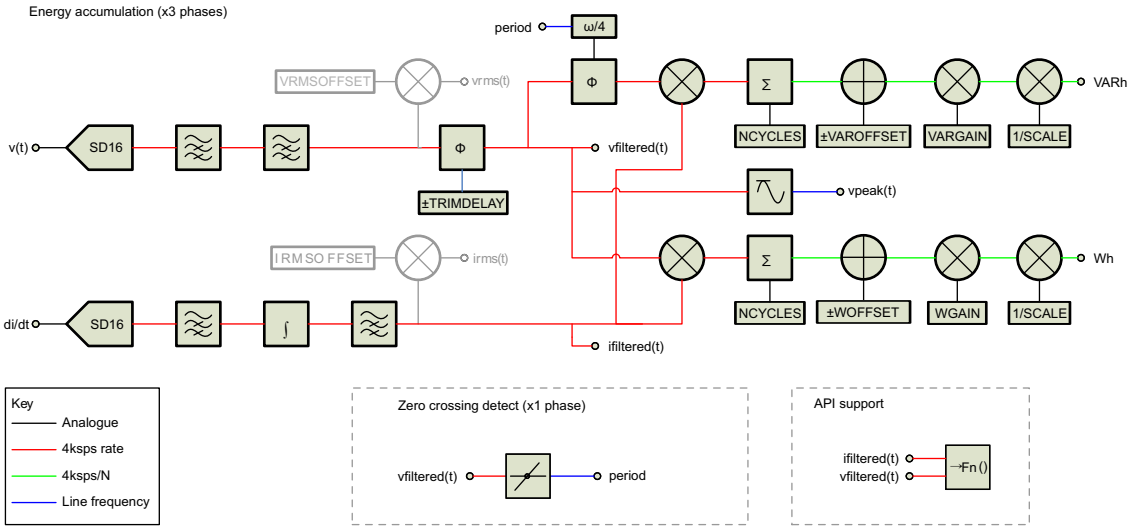


Figure 5. Software Flowchart

4.2 Energy Meter Software

This section details the contents of the package and the available API calls. Note that only minor adjustments for the sensor are necessary and that all other functions can be used as-is.

4.2.1 Contents of the Library Package

The package for the library contains the following files:

- The file softdidt.h in the directory Library\include. This is the file containing the export definitions for the API functions and their return codes and the setup definitions for the phases. The contents will be detailed later in this application report.
- The file Library.r43 in the directory Library\Release. This is the Rogowski-coil software library against which the application code needs to be linked.
- An example project in the directory Dummyapp is detailed later in this application report.

4.2.2 API Functions

Initialisation Functions

DiDtInitLibrary()

Prototype	uint16_t DiDtInitLibrary(void)
Parameters	None
Return Value	uint16_t version - library version number in the form 0xHHLL
Description	Performs any required software initialization of the library.
Comments	0xHH is the major version number and 0xLL is the minor version number; changes to the major version number represent incompatible feature set changes

DiDtInitHardware()

Prototype	status_t DiDtInitHardware(const sensor_routing_t *adcchannels)
Parameters	sensor_routing_t * adcchannels - filled with the mapping between ADC channel and sensor input, one for each phase
Return Value	status_t result – result code
Description	Informs the library of the hardware set up.
Comments	The library expects that the processor clock is already running at its high rate, and this function call will configure the ADC channels and one timer needed to run.

DiDtInitSensors()

Prototype	status_t DiDtInitSensors(const sensor_setups_t *sensors)
Parameters	sensor_ setups_t * sensors - array of setup structures, one for each phase
Return Value	status_t result – result code
Description	Sets up any sensor to sensor variations for the attached Rogowski coils.
Comments	None

DiDtInitAccumulators()

Prototype	status_t DiDtInitAccumulators(const sensor_accumulators_t *sensors)
Parameters	sensor_accumulators_t * sensors - accumulator structure to update
Return Value	status_t result – result code
Description	Initialise the accumulator set.
Comments	This function is required to set the continuation point for the accumulators after a power cycle, for example having retrieved them from non-volatile memory.

4.2.2.1 Obtaining Running Results

DiDtReadSnapshot()

Prototype	status_t DiDtReadSnapshot(sensor_results_t *sensors)
Parameters	sensor_results_t * sensors - result structure to update
Return Value	status_t result – result code
Description	Collects the last second snapshot of the sensor readings.
Comments	Calling this function at a rate faster than the accumulation period will merely return the last snapshot multiple times. Callers can check if the data has been updated since the last poll by inspecting the sequence number, which is incremented once per accumulation period.

DiDTReadAccumulators()

Prototype	status_t DiDtReadAccumulators(sensor_accumulators_t *sensors)
Parameters	sensor_accumulators_t * sensors - accumulator structure to update
Return Value	status_t result – result code
Description	Reads the current accumulator set.
Comments	Calling this function at a rate faster than the accumulation period has no effect, the accumulators are read only.

4.2.2.2 Runtime Changes

DiDtAdjustTestPulse()

Prototype	status_t DiDtAdjustTestPulse(const test_pulse_t *setting)
Parameters	Test_pulse_t setting - pointer to test pulse settings
Return Value	status_t result – result code
Description	Updates the test pulse settings.
Comments	A setting change also clears out the internal pulse accumulator since it would be in the wrong scale; this is therefore also the case at startup when calling this function for the first time.

4.2.2.3 Registering Callout Functions

DiDtRegisterMetrologyCallout()

Prototype	status_t DiDtRegisterMetrologyCallout(metrocallout_t function, void *param)
Parameters	status_t DiDtRegisterMetrologyCallout(metrocallout_t function, void *param) void * param - An opaque parameter to pass to the function when it is called
Return Value	status_t result – result code
Description	Registers a single function with the library that will be called when a metrology event occurs.
Comments	None

DiDtRegisterSampleCallout()

Prototype	status_t DiDtRegisterSampleCallout(samplecallout_t function, void *param)
Parameters	samplecallout_t function - Function to register, or NULL to deregister void * param - An opaque parameter to pass to the function when it is called
Return Value	status_t result – result code
Description	Registers a single function with the library that will be called when raw sample data is available for each phase.
Comments	None

4.2.3 Data Types

The following data types are used in addition to those in <stdint.h> per ISO9899:1999.

Table 1. Data Type

Data Type	Description
bool_t	A boolean taking the values TRUE (!FALSE) or FALSE (0) only
status_t	An alias of uint16_t for returning success or error codes
adcchannel_t	An alias of uint16_t for specifying an ADC channel number
adcgain_t	An alias of uint16_t for specifying the ADC gain value
metro_reason_t	An alias of type uint16_t containing reason codes which may be passed to callout functions to inform the application code that some interesting event has occurred within the metrology. The codes are detailed in Section 4.2.5.
metrocallout_t	A function pointer called when the library has interesting events to report, called with a subreason code indicating the event of interest, and the opaque handle that was supplied when the function pointer was registered
samplecallout_t	A function pointer called when the library has interesting events to report, called with the filtered voltage and current sample, and the opaque handle that was supplied when the function pointer was registered
phasemask_t	A bitmap of active phases when passing multiple structures around

4.2.4 Data Structures

The data structures used in the library and the API functions can be found in the header file softdidt.h. In this header file, also the scaling of the different results is explained. The two main structures used throughout the software, sensor_results_t and sensor_accumulators_t, contain the measurement results that are detailed in Section 4.2.4.1 and Section 4.2.4.2. The structures used during calibration of the system are outlined in their respective sections.

4.2.4.1 sensor_results_t

This structure contains the current results for each phase in a sub-structure “phases” of the type sensor_result_t and the phasemask defining the phase(s) used.

Table 2. Fields in sensor_result_t

Variable	Type	Scaling	Remark
Vrms	uint16_t	V * 2^{-7}	RMS voltage
Irms	uint16_t	A * 2^{-8}	RMS current
wh	uint16_t	Wh/s * 2^{-8}	Active energy
varh	uint16_t	Wh/s * 2^{-8}	Reactive energy
watts	uint16_t	W	Instantaneous active power
line_frequency	uint16_t	Hz * 2^{-10}	Frequency
wh_forward	uint8_t	-	Direction of Wh (forward == 1)
varh_forward	uint8_t	-	Direction of VARh (forward == 1)
sequence	uint_8	-	Free running sequence number

4.2.4.2 sensor_accumulators_t

This structure contains the current results for each phase in a sub-structure “phases” of the type sensor_accumulator_t and the phasemask defining the phase(s) used.

Table 3. Fields in sensor_accumulator_t

Variable	Type	Scaling	Remark
net_wh	int64_t	Wh * 2^{-8}	Net Wh (active power)
net_varh	int64_t	Wh * 2^{-8}	Net VARh (reactive power)

4.2.5 Callout Reasons

The following reason codes may be passed to the callout functions to inform the application code that some interesting event has occurred within the metrology.

Table 4. Callout Reasons

Definition	Description
SUBREASON_VOLTAGE_SAG	Voltage sag in progress
SUBREASON_OVER_CURRENT	Gross over-current detected
SUBREASON_REVERSE_CURRENT	Reverse current detected
SUBREASON_SUSPECT_FREQUENCY	Suspect mains frequency
SUBREASON_ACCURACY_LOST	Calculation accuracy error detected.
	NOTE: The library manages the intermediate accuracy internally;
	therefore, this code is not currently used.
SUBREASON_NEW_SNAPSHOT	New summation results are available

4.3 Energy Meter Configuration

For the library to function properly, it needs to be informed about the routing of the used SD16 channels and the gain necessary for the Rogowski coil used. This is done by populating a structure of the type sensor_routing_t and by calling the function DiDTInitHardware(), passing the populated structure. The small code example below shows how to achieve this.

```
/*****************************************************************************************/

/* Create a macro to tell the software that the following hardware connections exist and the */
/* gain necessary for the current channels: */
/* */
/* Voltage phase A connected to SD16 channel A2 */
/* Voltage phase B connected to SD16 channel A1 */
/* Voltage phase C connected to SD16 channel A0 */
/* */
/* Current phase A connected to SD16 channel A5 */
/* Current phase B connected to SD16 channel A4 */
/* Current phase C connected to SD16 channel A3 */
/* */
/* Gain needed for the used coils: 16 */
/* */
/*****************************************************************************************
/
#define HW_EVALKIT_ROUTING PHASE_ABC, /* 3 phases */ \
                           DIDT_HW_SD16_CH2,DIDT_HW_SD16_CH1,DIDT_HW_SD16_CH0,/*voltage inputs*/
\
                           DIDT_HW_SD16_CH5,DIDT_HW_SD16_CH4,DIDT_HW_SD16_CH3,/*di/dt inputs  */
\
                           DIDT_HW_SD16_PGA16                                  /* Preamp setting
*/
…
static const sensor_routing_t m_adcwiring[3] = { HW_EVALKIT_ROUTING };
…
void main(void)
{
     …
     /* Set up and calibrate the library and tell it how the sensors are wired up */
     DiDtInitLibrary();
     if (DiDtInitHardware(m_adcwiring) != DIDT_OK)
     {
        return;
     }
     …
}
```

Beside defining the routing of the ADC channels, also the PGA gain to be used for the SD16 channels used for the currents need to be set. Based on the sensitivity of the Rogowski coils connected this gain can be calculated once the reference voltage of the converters and the maximum current of the system is known.

In case of the software library, the build in SD16 converters are used with their internal 1.2V reference voltage, limiting the input range to ±0.6V. In a system with a Rogowski coil having a sensitivity of 66µV/A @ 50 Hz, a maximum current of 160Arms and a tolerance of 20% overcurrent, and Equation 1 is valid:

$$\begin{aligned} Vsensor_{pp} &= Vbusbar_{rms} * 2\sqrt{2} * k_{overrange} * k_{sensitivity} \\ &= 160\ A * 2\sqrt{2} * 120\% * \frac{66\,\mu V}{A} \\ &= 35.8\ mV_{pp} \end{aligned} \quad (1)$$

Knowing the sensor output during this overcurrent condition, it is possible to calculate the gain value to set the PGA in the SD16_A to. The allowable gains are 1, 2, 4, 8, 16, or 32, and the selected gain should maximize the input range used to enhance low current performance.

$$k_{pga} = floor\left(\frac{Vref_{pp}/2}{Vsensor_{pp}}\right) = floor\left(\frac{1.2V/2}{Vsensor_{pp}}\right) = 16 \quad (2)$$

NOTE: Additional harmonic content, such as a flat topped sine wave on a heavily loaded circuit, will result in additional output from the sensor as it is measuring di/dt. This translates to extra input range being required, in exchange for a loss of low current performance (because some of the bottom end bits are traded for spare top end bits in the ADC results).

5 Energy Meter Demo

5.1 EVM Overview

The hardware is based around the Texas Instruments EVM430-F47197 evaluation kit as described in *Implementation of a Three-Phase Electronic Watt-Hour Meter Using MSP430F471xx* (SLAA409). The EVM itself is solely a demonstrator and evaluation tool for the Rogowski-Coil software library from Texas Instruments and is not intended to be used as a Rogowski Coil technology demonstrator or EVM.

The three-phase MSP430 Rogowski evaluation kit includes an implementation of Sentec's patented Mobius technology, and shares the PCB with a capacitive dropper power supply similar to the one used in the EVM430-F47197 evaluation kit.

Care should be taken with soft magnetic materials in close proximity to the bus bars, for example nickel end caps on resistors and steel fixing bolts, as these will deform the magnetic field being measured.

The following figures of the EVM describe the HW. Figure 6 is the top view of the energy meter. The enclosure enables an easy evaluation and encapsulates the two boards, including the on-board current sensors.

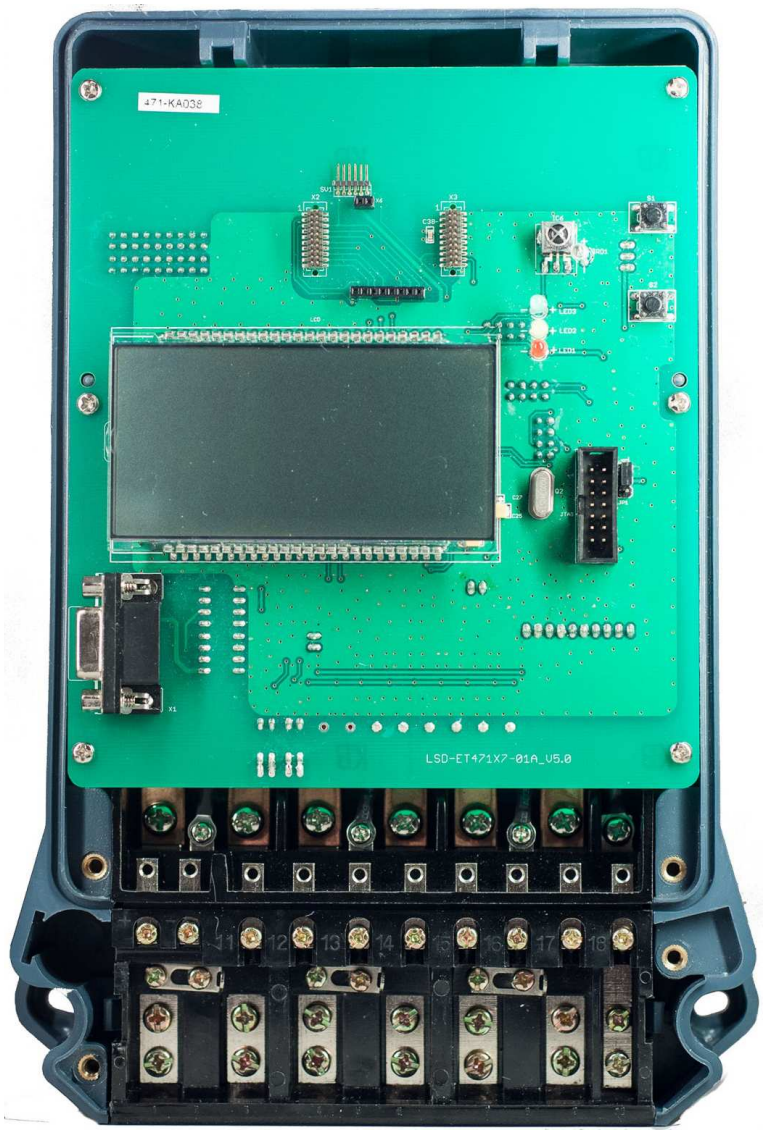


Figure 6. Top View of the Three-Phase Energy Meter EVM

The top of the EVM shows a 160 segment LCD that can be used to display energy, RMS voltage or current for all three phases. There is an isolated RS-232 connector available to interface to a PC. Also there is a JTAG-connector which can be used to program the MSP430.

Figure 7 shows the front view of the EVM with connections that need to be made to the current outputs from the test system. Starting from the right going left connections for the current is made. GND on the extreme right can also be connected from the top with the voltages as shown in Figure 8.

Figure 7. Front View With Current Connections

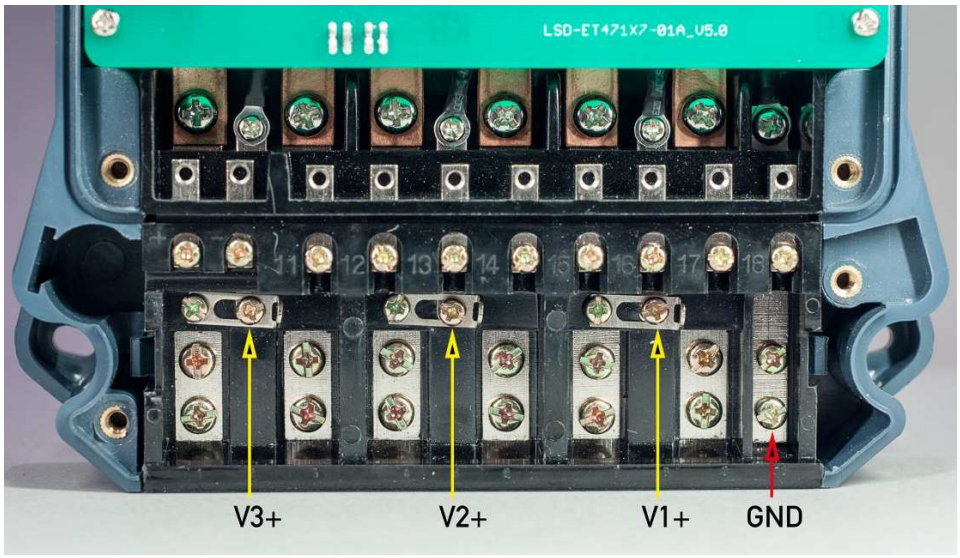


Figure 8. Top View With Voltage Connections

5.2 Jumper Settings

5.2.1 Top PCB (top and back side)

- Jumper JP1: Power selector
  - Jumper on [1-2]: Board is powered from JTAG (if the EVM is to be connected to a non-isolated voltage and current source, an isolated emulator must be used)
  - Jumper on [2-3]: Board is powered from the mains
- Jumper JP6: Selector for V+ (voltage for opto-couplers, and so forth). In order to operate properly, this jumper must be in the [1-2] position.
  - Jumper on [1-2]: D_{VCC} is used for V+
  - Jumper on [2-3]: V_{COMMS} is used for V+
- Jumper JP11: V_{REF} selector
  - Jumper must be open (coupled to GND)

- Jumper JP19: Ground selector for current paths. This jumper needs to be set depending on the test-equipment used.
  - Open: Currents are referenced to ground inside the test-equipment
  - Closed: Currents are not referenced to ground inside the test-equipment, so this connection needs to be done externally

5.2.2 Bottom PCB (top side only)

- Jumper JP7: Used to enable the battery backup BAT1, which is not fitted by default. Battery backed operation is not supported by the software, so this jumper needs to be left open in the default configuration:
  - Open: Backup battery is not enabled
  - Closed: Backup battery is enabled
- Pads R15, R16, R17: Options for surface mount 0603 size 0 Ohms resistors to join the center tap of the Mobius sensor coils to ground. They are not fitted by default.

6 Example Project and Calibration

6.1 Loading the Example Project

The source code is developed in the IAR™ IDE using IAR compiler version 6.x. The project files cannot be opened in earlier versions of IAR. If the project is loaded in a version later than 6.x, a prompt to create a backup is displayed, and you can click YES to proceed. The source code of the example project together with the binary of the library is available as a zip file with *Implementation of a Three-Phase Electronic Watt-Hour Meter Using MSP430F471xx* (SLAA409). After the zip is decompressed, two folders and the IAR workspace-file will be shown: one folder contains the dummy (example) application and the other one the library objects and the API include file. The source folder structure is shown in Figure 9.

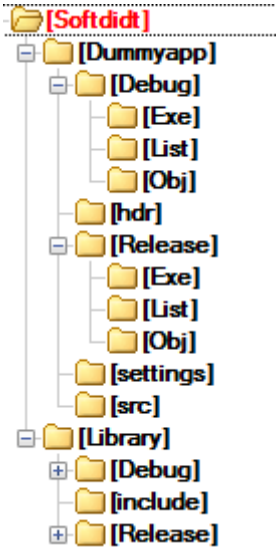


Figure 9. Source Folder Contents

For the first time use, it is recommended to completely rebuild the project.

6.1.1 Opening the Project

Open IAR Embedded Workbench®, find and load the project Dummyapp.ewp, and rebuild all (see Figure 10).

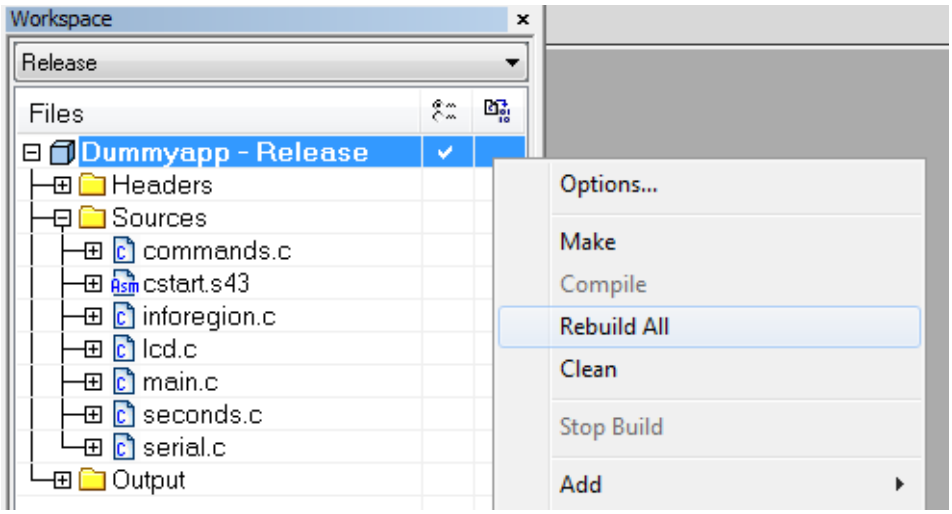


Figure 10. Project Build in IAR

Load the program onto the EVM and select “Download and Debug” from the Project drop-down list. This will start the application.

6.1.2 Viewing the Results

The results of the measurements can be viewed in two ways: first using the serial interface as described in Section 6.2 or using the LCD display of the board. By default, the display only shows the reactive energy per phase, but this can easily be changed to any other value by modifying the code at the bottom of the DummAppEntry() function in the file main.c.

6.2 Calibration

6.2.1 Overview

Calibration is best performed using a meter test station and test pulse, but a rough calibration can be performed using a known current and voltage source via the serial terminal. Connect a terminal program to the display PCB via the isolated serial port using the following settings:

- Baudrate: 9600 bps
- Databits: 8
- Parity: None
- Stopbits: 1
- Handshaking: None
- Handshaking: None
- Emulation: ASCII
- Local Echo: Off
- CR Translation: Off (for both in and out)

This describes a zero offset calibration of just one phase, but in a production situation all three phases could be zero offset calibrated in parallel, with the addition of more complex software. Also note that all the commands mentioned in this section are not part of the library itself, but part of the dummy application.

6.2.2 Variables Available for Calibration

The calibration values for each sensor can be described in the structure sensor_setup_t, together with the thresholds for creep, sag and high load. Together with the phase mask, the structure sensor_setups_t defines an array of three variables of the type sensor_setup_t. All calibration values should be placed in a local structure of the type sensor_setups_t, which, then should be passed on the API function DiDtInitSensors() for inclusion in the calculations.

Table 5 shows the variables used to calibrate the sensors.

Table 5. Variables for Calibration

Variable	Type	Scaling	Remark
vrms_gain	uint16_t	factor * 2^{-8}	(Vrms gain)2
irms_gain	uint16_t	factor * 2^{-8}	(Irms gain)2
wh_gain	uint16_t	factor * 2^{-16}	Used to calibrate the gain of the active power
wh_offset	uint16_t	factor * 2^{-8}	Used to correct any leakage between I and V channels. To be calibrated before the calibration of the wh_gain gain
varh_gain	uint16_t	factor * 2^{-16}	Used to calibrate the gain of the reactive power
varh_offset	uint16_t	factor * 2^{-8}	Used to correct any leakage between I and V channels. To be calibrated before the calibration of the varh_gain
delay	uint16_t	Units of modulator frequency	Used to compensate for any phase shift the board and the sensor might introduce. Defines the shift between the sampling of the voltage and the current. Must be in the range of 0 .. 255. If not, DiDtInitSensors() will return the error DiDt_ERR_EXCESSIVE_DELAY. One unit equals about 954 ns (if the default timings are used)
vrms_gain	uint16_t	factor * 2^{-8}	Used to calibrate the gain of the RMS voltage
irms_gain	uint16_t	factor * 2^{-8}	Used to calibrate the gain of the RMS current

6.2.3 Variables to be Set

The software also includes a couple of variables which allow to be set the thresholds for error messages (callouts) to be sent from the library (variables creep_threshold, sag_threshold and high_load_threshold) and what the starting energy is for accumulating. These variables are:

Table 6. Fields in sensor_accumulator_t

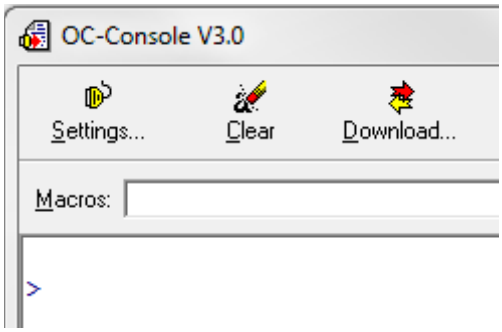
Variable	Type	Scaling	Remark
creep_threshold	uint16_t	Wh/s * 2^{-12}	Energy, which must be present to start the accumulation. Max. is 4096 steps, equating to a power of 5 W
sag_threshold	uint16_t	Volts	If the peak voltage measured is below the value set in this variable, a voltage sag in progress will be reported in the below_sag field of the sensor_results_t structure, which is returned by the DiDtReadSnapshot() API function and the registered callout function is called with the error (reason) code SUBREASON_VOLTAGE_SAG (0)

Table 6. Fields in sensor_accumulator_t (continued)

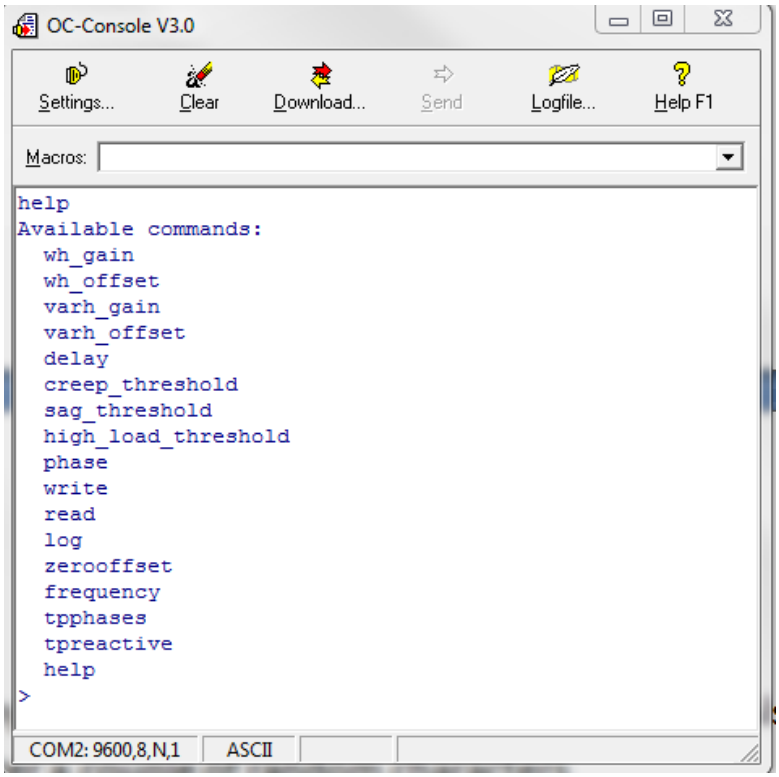
Variable	Type	Scaling	Remark
high_load_threshold	uint16_t	Wh/s * 2^{-8}	If the energy measured exceeds the value given in this variable the registered callout function is called with the error (reason) code SUBREASON_OVER_CURRENT (1)

6.2.4 Starting

Check if the communication with the board is properly established. For that, hit the return key and a prompt '>' should show. Note that after power-up, a couple of random characters could be displayed. This does not indicate a problematic connection by itself.



At the prompt, type: "help" and hit return. A list of commands available should appear.



Now, as the communication has been established, the calibration can start.

Select the phase to operate on here, for example, using phase A (the command phase = 1 would select phase B and phase = 2 would select phase C, respectively):

```
phase=0
```

and read back the active settings using

```
read
```

If the Flash memory has been erased, the values will all be 0xFFFF; a sensible initial set of values to start with would be to enter the following assignments:

```
wh_gain=16384
wh_offset=0
varh_gain=16384
varh_offset=0
delay=0
creep_threshold=0
write
```

6.2.5 Gain Trim

Apply a known current of 15 A and 240 V with a phase shift of 0 degrees; this is convenient, because it is 3600 W per phase (which in internal units corresponds to 256 × 2-8 Wh/s). The actual current should merely be selected to be well away from zero.

Turn logging on and observe the output values. Use the value in the first column (scaled Wh/s) as value for the variable $Wh_{measured}$ in Equation 3 together with the default of the Wh_{gain} (current) of 16384 set earlier.

$$Wh_{gain} = \frac{Wh_{expected}}{ScalingFactor * 3600 * Wh_{measured}} * Wh_{gain}(current) \quad (3)$$

With ScalingFactor = 2^{-8}

For example, with a setting of 240 V and 16 A, which results in a value of 3600 Wh, as $Wh_{expected}$:

```
log=1
A, +230, -2, 3234
B, +260, +1, 3656
C, +255, -3, 3585
wh_gain=18236
```

$$Wh_{gain} = \frac{3600}{2^{-8} * 3600 * 230} * 16384 = 18236 \quad (4)$$

If a better precision is needed, the average of several measured Wh/s values can be used for the $Wh_{measured}$.

Another possibility is to use the test pulse LEDs of the board together with an e-meter tester. For this, bring the test pulse LED into sight of the optical test probe of the meter test station, or equivalent isolated connection. Two commands set that load contributes to the test pulse are:

```
tpreactive=0
tpphases=7
```

The 'tpphases' command accepts a bitmask of phases to sum into the test pulse, phase A contributes 1, phase B contributes 2, phase C contributes 4, therefore, all phases are selected in the example above. The choice of active or reactive energy is made by using '1' to mean 'yes' (reactive) and '0' to mean 'no' (active).

With this setup, read the error returned from the tester and use Equation 5 to calculate the new Wh gain setting:

$$Wh_{gain} = Wh_{gain}\,(actual) * (1 - error\%) \tag{5}$$

The new gain factor for the active energy is then entered using the following command:

```
wh_gain=Wh_gain
```

6.2.6 Zero Offset

Apply zero current to the meter and accumulate energy over a reasonable period of time; in this example, 30 seconds is selected:

```
zerooffset=30
```

After this time, the dummy application will calculate the correction factor and update the local RAM copy.

If the dummy application is not used, it is possible to calculate the correction factor following these steps:

1. Accumulate the results for the active (wh) and reactive (varh) energy over the desired period of time.
2. Scale the accumulated values and divide them by the gain (wh_gain) and the test length (in the example above: 30).
3. Update the offsets in the sensor setup structure.

These steps are based on the fact that over a longer period of time the following is valid:

$$\frac{N\,(data + offset)}{N}\ is\ identical\ to\ \frac{Ndata}{N} + offset \tag{6}$$

An example of the complete procedure in 'C' can be found in the function CommandRun_zerooffset() inside the file commands.c.

6.2.7 Phase Correction

Set the test load to 15 A and 240 V and 60°, adjust the delay register in the range 0-255 to achieve an output of 128 × 2-8Wh/s. This can be achieved by iteration or by using Equation 7:

$$delay_{new} = delay_{old} + \frac{OSR * f_{sampling}}{2 * \pi * f_{line}} * (\cos^{-1}\frac{1 + E}{2} - \frac{\pi}{3}) \tag{7}$$

With:

- *OSR*: Oversampling rate, that is, 256
- $f_{sampling}$: sampling rate, that is, 4096 Hz
- f_{line}: line frequency, either 50 Hz or 50 Hz
- *E*: Measured error in %

6.2.8 RMS Gains

When the RMS option is enabled, with the current still at 15 A and voltage still at 240 V the gains can be adjusted to give the desired output in the log data. Note that the gain is applied before the square root, so any gain factors must first be squared.

For example, when calibrating the voltage to 30720 × 2^{-7} V the output is reported as 30000 with a gain of 12340. The corrected calibration factor should be (30720 / 30000) × (30720 / 30000) × 12340 = 12939.

In general terms, Equation 8 and Equation 9 would look as follows:

$$Voltage_{gain} = \frac{Voltage_{expected}}{ScalingFactor_{voltage} * Voltage_{measured}} * Voltage_{gain}(actual) \quad (8)$$

and

$$Current_{gain} = \frac{Current_{expected}}{ScalingFactor_{current} * Current_{measured}} * Current_{gain}(actual) \quad (9)$$

With:

- ScalingFactorvoltage = 2^{-7}
- ScalingFactorcurrent = 2^{-8}

6.2.9 Final Check

Apply a known current of 15 A and 240 V and 0 V, check offset.

Having applied the new calibration values, the results can be committed to flash using the command. In the event of a mistake, the modified values can be discarded by either selecting another phase with the ‘phase’ command or reading back the stored settings with ‘read’. It is important to note that if the board is powered down before the write command is issued, the new calibration values will be lost and a new calibration has to be performed.

```
write
```

7 Important Notes

- This document is preliminary and is subject to change when the next board revision is made available.
- Never use the mains at the same time as debug, unless you are using isolated-FET USB FETs.
- Three LEDs, one for the active energy of each phase, are present to test the accuracy of the meter via pulse generation.

WARNING

Failure to adhere to these steps and/or not heed the safety requirements at each step may lead to shock, injury, and damage to the hardware. Texas Instruments is not responsible or liable in any way for shock, injury, or damage caused due to negligence or failure to heed advice.

8 References

- *MSP430F471x3, MSP430F471x6, MSP430F471x7 Mixed Signal Microcontroller Data Sheet* (SLAS626)
- *MSP430x4xx Family User's Guide* (SLAU056)
- *Implementation of a Three-Phase Electronic Watt-Hour Meter Using MSP430F471xx* (SLAA409)
- *MSP430 Family Mixed-Signal Microcontroller Application Reports* (SLAA024)

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