UCD3138 E-Metering Solution

High Voltage Controller Solution

1 Introduction:

The real-time energy consumption measurement, including input real power and input RMS current measurement for off-line power supplies, is becoming ever more important nowadays. Traditionally the input power and current are measured by a dedicated power metering chip and extra sensing circuit. While the power metering chip proved to be sufficient, it adds extra cost and design effort. TI has developed an e-metering solution using the digital PFC controller UCD3138. This method uses the same existing PFC hardware, with simple two point calibration and optimized mathematical calculations. It significantly reduces the cost and design effort, and has no impact on normal PFC control. This application note summarizes how to use UCD3138 E-metering.

2 ADC assignment:

To get accuracy Pin measurement, Vin and Iin need to be sampled at the same time. UCD3138 provides dual sample and hold capability in which two channels can be configured to do sample at the same time. Here is the recommend configuration:

AD_00: Vac line
AD_02: Iin
AD_04: Vac neutral

Only 4 channels can be measured every 20us. If there are more than 4 channels needs to be measured in 20us, Vac line, Vac neutral and Iin need to be measured every time, the other channels can share the 4th sequence of the ADC measurement.

3 E-metering calibration:

Both Vin and Iin sense circuit need to be calibrated before using e-metering function. The Vin sense circuit is quite simple, it can be just a voltage divider as shown in Figure 1. There are usually clamp diodes to protect ADC pins, the reverse leakage current of the diodes will affect ADC measurement accuracy. Make sure to choose the diodes with low reverse leakage current.

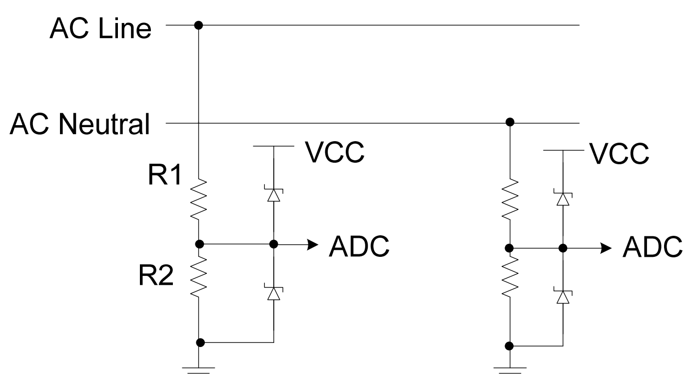


Figure 1. Input voltage sense circuit

For Iin sense, this method is based on using current shunt and amplifier. A low temperature coefficient current shunt and low offset amplifier is recommended.

Only 2 calibration points are needed. Follow these steps for e-metering calibration:

1. *emi_discharge_resistance*: EMI discharger resistor (Kohm). If there no such resistor, put R=10000
2. *emi_capacitance*: the toal capacitance of EMI filter (include the one right after bridge rectifier) (nF)
3. *ac_frequency*: input AC frequency (Hz)
4. Turn on PFC with 115V and 10% load, then wait for 30 seconds
5. Record the RMS value of input current Iin (mA), represented as 'step5'
6. Record the RMS value of input voltage Vin (V)
7. Read *iv.iin_filtered* from UCD3138, depicted as 'step7'
8. Read *iv.iin_squared_filtered* from UCD3138, represented as 'step8'
9. Read *iv.vin_squared_filtered* from UCD3138, represented as 'step9'
10. Keep Vin at 115V, increase load to 80% load, then wait for 30 seconds
11. Record the RMS value of input current Iin (mA), represented as 'step11'
12. Read *iv.iin_filtered* from UCD3138, represented as 'step12'
13. Read *iv.iin_squared_filtered* from UCD3138, represented as 'step13'
14. Set *iv.ipm_filter_shift = 11*
15. a = step8 / $2^{iv.ipm\_filter\_shift}$
16. b = step7 / $2^{iv.ipm\_filter\_shift}$
17. c = step13 / $2^{iv.ipm\_filter\_shift}$
18. d = step12 / $2^{iv.ipm\_filter\_shift}$
19. $$e = \left(step5 - \frac{Vin}{emi\_discharge\_resistance}\right)^2 - \left(\frac{2*3.14*ac\_frequency*Vin**\mathrm{emi\_capacitance}}{1000000}\right)^2$$
20. $$f = \left(step11 - \frac{Vin}{emi\_discharge\_resistance}\right)^2 - \left(\frac{2*3.14*ac\_frequency*Vin**\mathrm{emi\_capacitance}}{1000000}\right)^2$$
21. x: the slope of current sensor, it is calculated as:
    x = sqrt((a*e - c*e - 2*b*d*e + 2*d^2*e - a*f + 2*b^2*f + c*f - 2*b*d*f - 2*sqrt((b - d)^2*(d^2*e^2 - 2*b*d*e*f + c*e*(-e + f) + f*(a*e - a*f + b^2*f))))/(a^2 + c*(4*b^2 + c - 4*b*d) - 2*a*(c + 2*(b - d)*d)));

22. y: the offset of current sensor, it is calculated as:
    y = -((x*(d^2*e + b^2*f - b*d*(e + f) + sqrt((b - d)^2*(d^2*e^2 - 2*b*d*e*f + c*e*(-e + f) + f*(a*e - a*f + b^2*f)))))/((b - d)*(e - f)));

23. The calculated x and y are decimal. To maintain enough accuracy in the calculations, the small decimal values are multiplied by 2^N and then rounded to the closest integer. For example, if the current sense slope and offset for a PFC are calculated as:

    x = 1.59
    y = 229.04

    x will be multiplied by 2^8 and rounded to 407, y will be multiplied by 2^0. The following variables are defined:

    iin_slope = 407;
    iin_slope_shift = 8;

iin_offset = 229;
iin_offset_shift = 0;

24. The slope of voltage sensor is calculated as

$$z = \sqrt{\frac{Vin^2 * 2^{\text{iv.ipm\_filter\_shift}}}{step9}}$$

25. The offset of voltage sensor is zero
26. The calculated z is decimal. To maintain enough accuracy in the calculations, the decimal values is multiplied by 2^N and then rounded to the closest integer. For example, if the voltage sensor slope for a PFC is calculated as:

    z = 0.09863

    z will be multiplied by 2^{10} and rounded to 101. The following variables are defined:

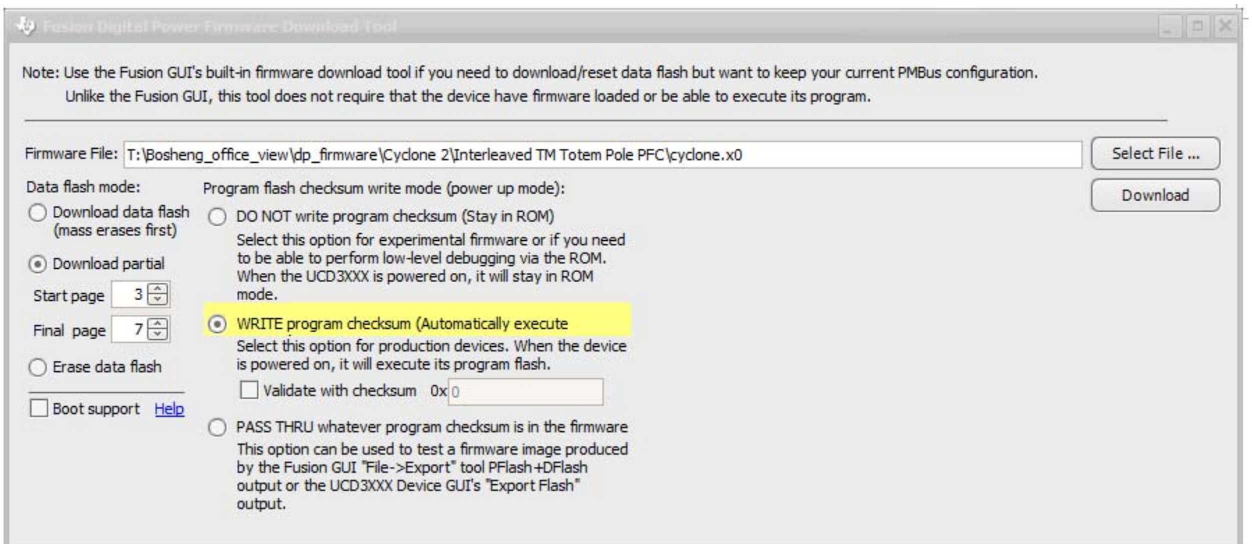
    vin_slope = 101;
    vin_slope_shift = 10;
    vin_offset = 0;
    vin_offset_shift = 0;

27. After calibration, assign these calibrated values to the corresponding global variables.
28. Make sure *iin_slope_shift* and *vin_slope_shift* are not bigger than 10 (avoid calculation overflow).

4 Store calibrated parameters into datafalsh

The calibrated values need to be stored in dataflash. The dataflash is separated into several sections, one section is used to store the PFC configuration parameters, such as PID coefficients. Another section is used to store the e-metering calibration parameters. In addition, a copy of them are also stored in other sections. During PFC power up, it will first read configuration parameters from dataflash and configure PFC control parametrers, then it will read e-metering calibration parameters for e-metering measurement. During firmware downloading, user can choose to download the whole dataflash, or just download a specific section. For example, the PFC firmware may need to be updated, the new firmware may has new PID coefficients, at the same time the e-metering calibration parameters need to be maintained. In this case, user can choose only download the sections in dataflash which contains PID coefficients, the other sections will be unchanged. UCD3XXX GUI supports this feature.

4.1 UCD3XXX Device GUI – Firmware download



4.1.1 Data flash download options

There are three options regarding downloading of data flash.

The option “Download data flash” writes the data flash portion defined in the .x0 file to the data flash location on the device. Before the writing of data flash, a mass erase is issued where all the pages are cleared simultaneously.

The option “Erase data flash” simply issues the mass erase without downloading the .x0 file.

The second option is “Download partial.” For this case the user must specify an initial start page index and a final page index of the pages defined in their .x0 they wish to download. The data flash pages outside the range of these indices on the device will not be edited.

4.1.2 Download partial flash clarification

Erase time: Before the continuous set of pages (defined by the start and final page indices) are written, the page erase command is issued sequentially beginning with the “Start page.” This erase is done sequentially, one page at a time, including the appropriate wait time after a page erase has been issued. Therefore, if there are 10 pages and “y” is the wait time per page erase, then the total wait time needed would be 10y. For the first option above, the wait time is only “y”, as the mass erase applies a simultaneous erase to all the pages as opposed to the sequential erase in this option.

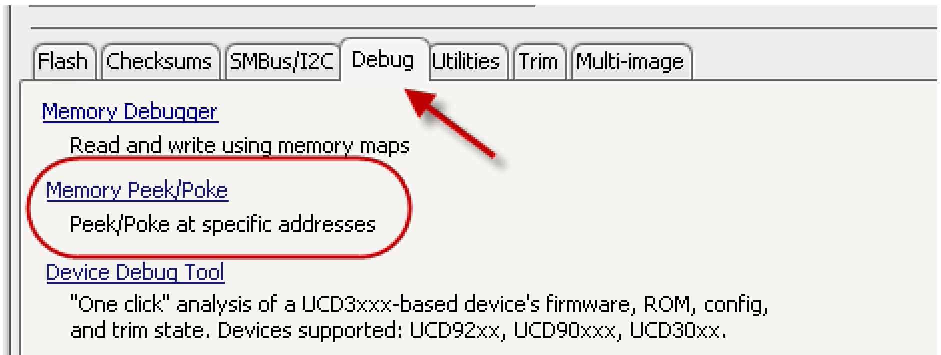
Identifying the pages: Once the data flash beginning address, and the address of the data variables with their respective data lengths are known then finding the start page index and final page index for a partial download can be found as follows:

Start_page_index = (data_variables_begin_address – data_flash_begin_address)/0x20
Final_page_index = Start_page_index + (sum_of_data_lengths/0x20) -1

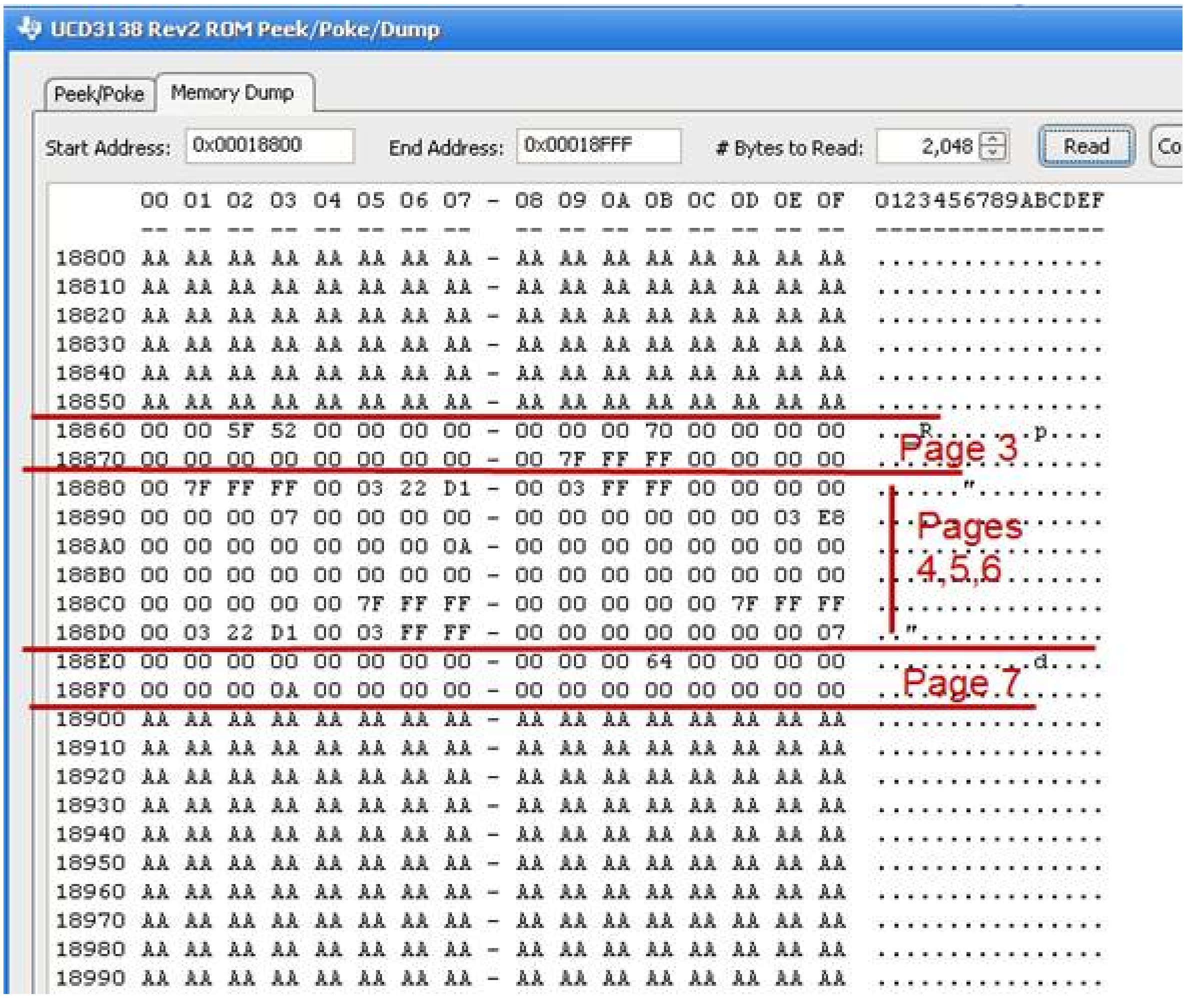
Note: usually the data that is being partially downloaded to the device is defined in the firmware along page boundaries.

4.1.3 Helpful tools

The “Memory Peek/Poke” tool is helpful for observing the flash.

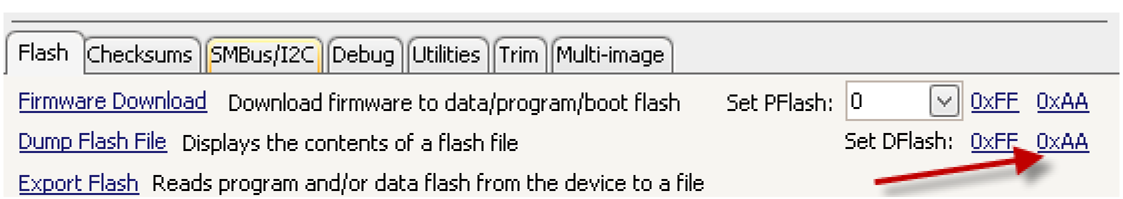


After the user specifies the begin and end address they can view the flash contents in the “Memory Dump” tab.



In the image above the data flash(0x18800-0x18FFF for UCD3138) was set initially to all 0xAA. Then a data flash partial download was done where the start and final page indices were defined to be 3 and 7 respectively.

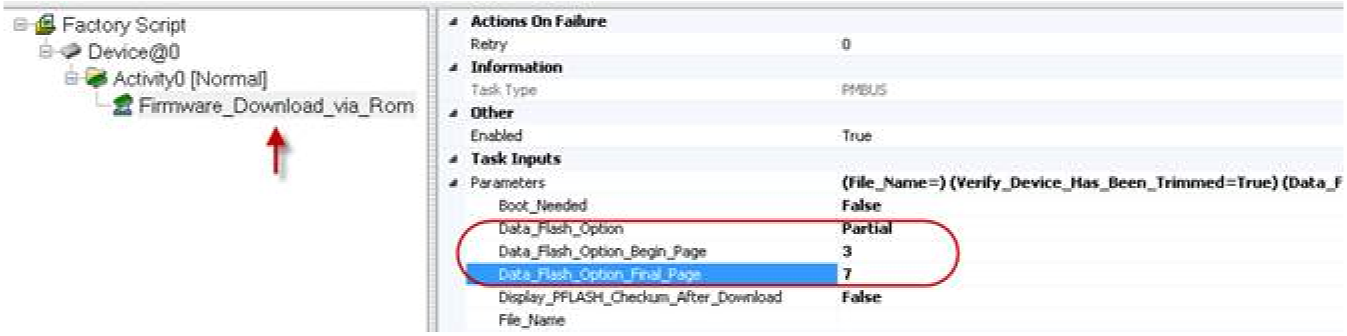
Note: To set the data flash to 0xAA click the 0xAA link found in the "Flash" tab as shown below:



.

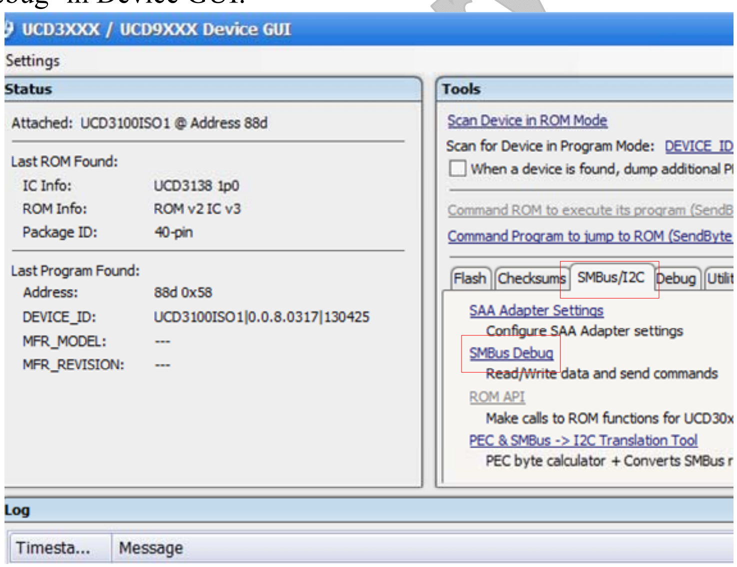
4.2 MFR GUI - Firmware download

The partial download feature described above is also supported in the MFR GUI. Both firmware download tasks, namely Firmware_Download_via_Rom and Firmware_Update, have support for partial download of the data flash. Below displays the related properties to be set if data flash download option "Partial" is selected.

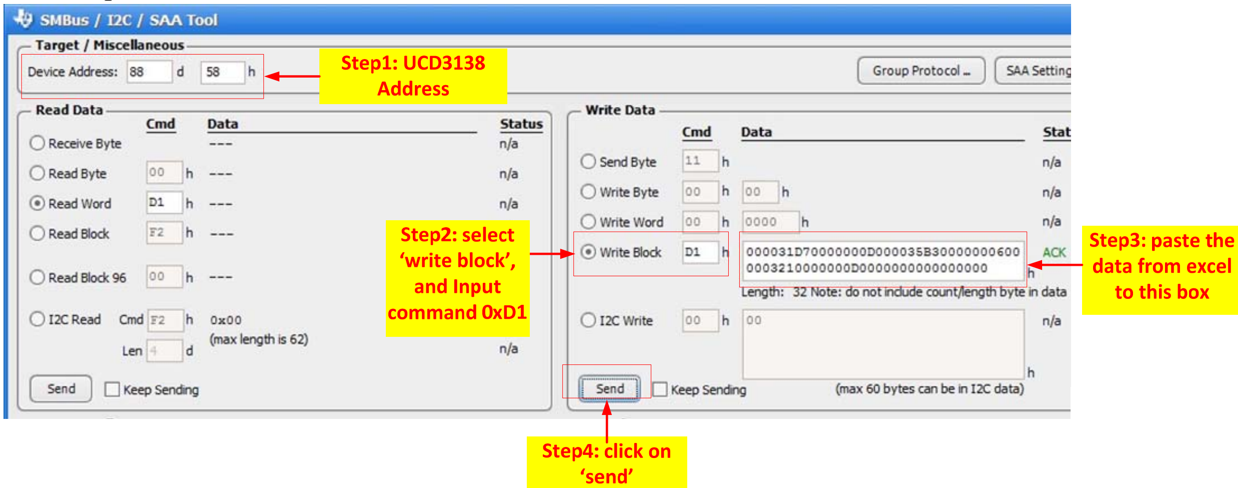


5 Send calibration parameters to UCD3138 through PMBus

It is customer's choose of how to send the calibrated parameters to UCD3138 in mass production. One example is: if the calibration equipment is on PFC side, the calibrated parameter can be sent to UCD3138 through PMBus. Click on 'SMBus Debug' in Device GUI.



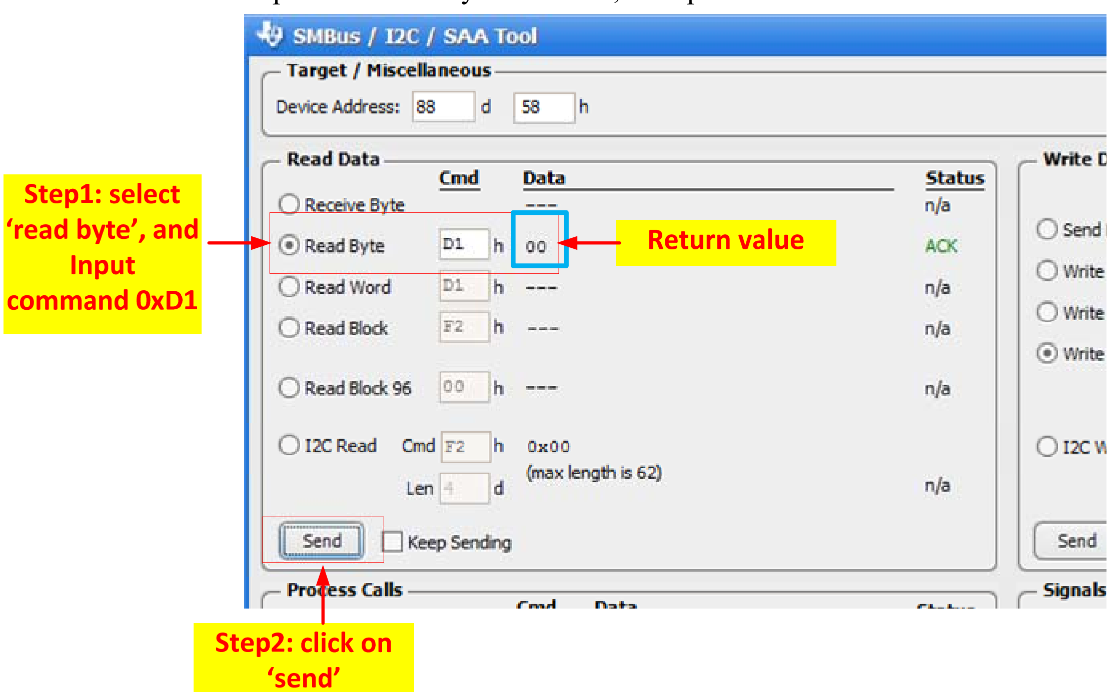
Follow below steps filled in associated information



After click the "Send" button, the latest e-metering parameters will successfully update to RAM and store into the data flash.

Verify to see if the update the e-metering coefficients is successful.

a. Click on read byte and command is still 0Xd1
b. The return data shows the status of update e-metering
   i. 00 – update flash and ram successful
   ii. 01 – input emetering coefficients number of byte is incorrect
   iii. 10 – input number of byte is correct, but update data flash is unsuccessful



6 Send calibration parameters to UCD3138 through UART

If calibration equipment is on the DC/DC side, the calibration parameter can be sent to UCD3138 via UART. The following 7 steps show the process of doing calibration in secondary side.

1. Secondary side(MFR GUI)send a command(0x01) to Primary side via UART, that requests primary send e-meter data for calibration. Here is the data format between primary side and secondary side through UART. Each package data contains header, command, 8 bytes valid data, and checksum.

Secondary to Primary

Header (0x55)	CMD(0x01)	Byte 2	Byte 3
Byte 4	Byte 5	Byte 6	Byte 7
Byte 8	Byte 9	Checksum	

2. After receiving the calibration request, primary side will send out the related data to secondary side. There are 3 variables needs to be sent out, and 2 of them are 8 bytes, one variable is 4 bytes. Since each package of data only contains 8 bytes valid data, so secondary side will send 3 times for all data.

Here is the data that secondary side sent to primary side:

a. Group 1: (CMD: 0X10)
   o **unsigned long long** vin_squared_filtered; //8bytes
b. Group 2: (CMD: 0X11)
   o Uint32 iin_filtered; //4bytes
c. Group 3: (CMD: 0X12)
   o **unsigned long long** iin_squared_filtered; //8bytes

Here is the format for each package. Byte 2 is the most significant byte, and byte 9 is the least significant byte.

Primary to secondary

Header (0x55)	CMD(0x10/ 0x11/0x12)	Byte 2	Byte 3
Byte 4	Byte 5	Byte 6	Byte 7
Byte 8	Byte 9	Checksum	

3. If secondary receive incorrect data, it should goes back to step1, request again. Otherwise, go ahead with the following step.
4. E-meter parameter will be calculated by MFR GUI in secondary side, then those data will be transferred to primary side via UART. The emeter parameter will be divided into 3 groups, each group contains slop and slope_shift value. It is also necessary to send 3 times to transfer all of the data.
   a. Group 1: (CMD:100)
   ```
   Uint32 vin_slope;
   Uint32 vin_slope_shift;
   ```
   b. Group 2: (CMD:101)
   ```
   Uint32 iin_slope;
   Uint32 iin_slope_shift;
   ```
   c. Group 3: (CMD:102)
   ```
   Uint32 iin_offset;
   Uint32 iin_offset_shift;
   ```

As for vin_offset and vin_offset_shift are always 0, so there is no need to transfer those variables.

Here is the format for each package:

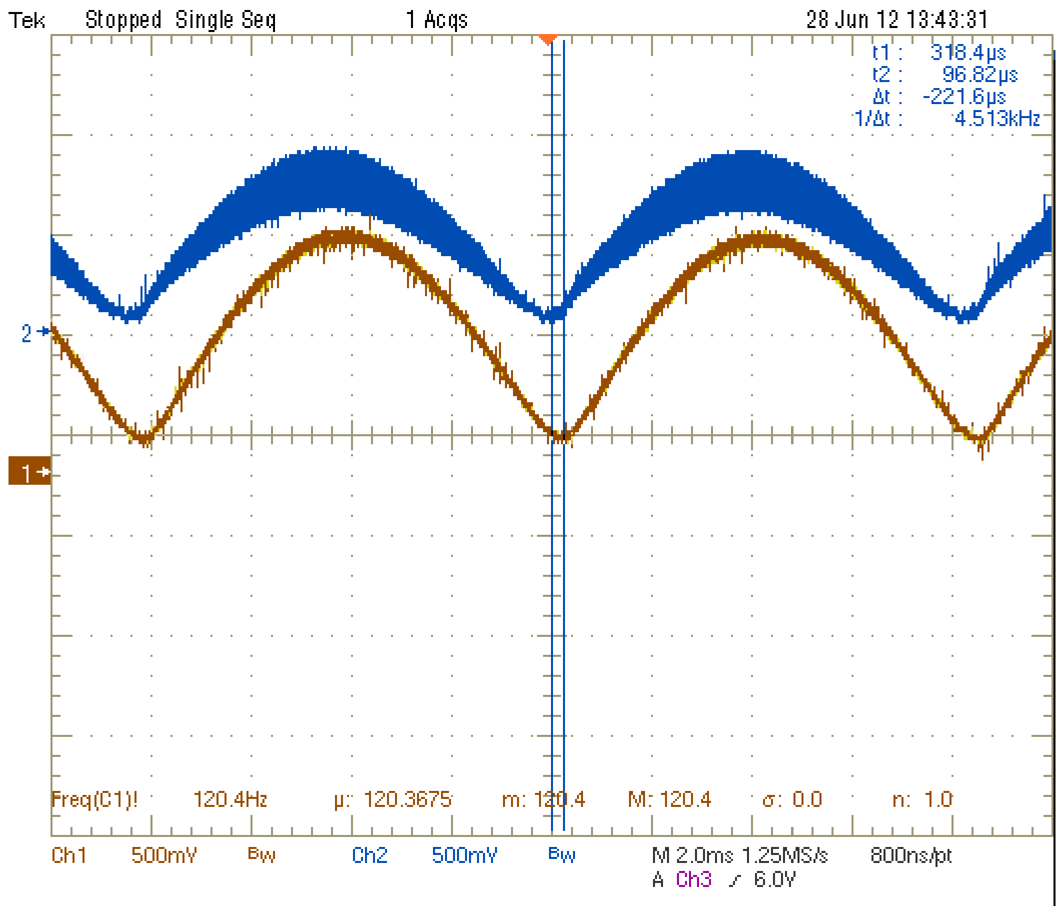
Secondary to Primary

Header (0x55)	CMD(100/ 101/102)	Byte 2	Byte 3
Byte 4	Byte 5	Byte 6	Byte 7
Byte 8	Byte 9	Checksum	

5. After primary side got the correct data, then store it into data flash.
6. In order to make sure that the data stored in data flash is correct, primary side will send the values programmed in data flash to secondary side.
7. Secondary side check the value to see if it is correct, if no, then go back to step 4.

7 Current Sense Circuit Phase Shift Compensation

Due to the low pass filter in current sense circuit, the measured current signal is delayed and out of phase with actual current, an example is shown in figure below:



Channel 2 is signal comes out of current shunt. Channel 1 is current sense signal connected to ADC. It has about 220µs phase delay. To compensate this delay, set:

iv.ipm_buff_delay = 11; //220/20=11

8 EMI resistance

Measure the resistance from AC input to the point where AC voltage is sampled. It is usually very small value, make sure you use kelvin sensing measurement.

emi_resistance = 20;//the total resistance of EMI filter in mohm

9 Integrate e_metering() function in your project

9.1 call input_power_measurement() from standard interrupt

9.2 call e-metering() from background loop in main.c

The *e_metering()* function in *main.c* actually contains three sub-functions, with *pmbus_handler()* in between. Make sure you have *pmbus_handler()* in your code. If not, instead of calling *e_metering(),* you can call *input_voltage_calculation(), input_current_calculation(), input_power_calculation()* separately.

```
void e_metering(void)
{
        input_voltage_calculation();

        pmbus_handler();

        input_current_calculation();

        pmbus_handler();

        input_power_calculation();
}
```

10 E-metering execution and response time

The execution time for input_power_measurement() is about 4.54us. The *e_metering()* function is called in background loop, its execution time is not critical, it will not affect PFC control.

It needs a little time for e-metering to measure Vin, Iin, Pin information and then calculate RMS value. In general, the e-metering can report the correct value very fast. In the extreme case, for example transient response, the e-metering will takes a longer time to report the accurate new value. Here is the response time measured from TI EVM at extreme transient response condition (the operation condition change is a jump, with almost infinite slew rate):

	from	jump to	time takes to report correct value
Vin	115V	220V	180ms
	220V	115V	175ms
Iin	15%	85%	110ms
	85%	15%	198ms
Pin	15%	85%	216ms
	85%	15%	220ms

11 Variable definition

There is a struct define in *variables.h*. The first part of the variables are reserved for e-metering, please do not change or modify. If you want to add new variables in this struct, please add in the bottom.

```
struct INTERRUPT_VARIABLES
{
        //stuff for ADC measurement
        Uint32 adc_raw[NUMBER_OF_ADC_CHANNELS_ACTIVE];
        Uint32 adc_avg[NUMBER_OF_ADC_CHANNELS_ACTIVE];
```

```
        //stuff for Iin
        Uint32 iin_raw;
        Uint32 iin_filtered;
        Uint32 iin_filtered_reporting;
        Uint32 iin_squared;
        unsigned long long iin_squared_filtered;
        unsigned long long iin_squared_filtered_reporting;

        //stuff for Vin
        Uint32 vin_raw;
        Uint32 vin_sum;
        Uint32 vin_filtered;
        Uint32 vin_filtered_reporting;
        Uint32 vin_average;
        Uint32 vin_squared;
        Uint32 vin_squared_slow_average;
        Uint32 vin_squared_average;
        unsigned long long vin_squared_filtered;
        unsigned long long vin_squared_filtered_reporting;

        //stuff for pin
        Uint32 pin_raw;
        unsigned long long pin_filtered;
        unsigned long long pin_filtered_reporting;

        Uint32 ipm_filter_shift;
        Uint32 half_cycle_counter_filtered;
        Uint16 cir_buff[64]; //64buffer for vin
        Uint8 ipm_buff_delay; //Iin_sense has delay, to sompensate that, delay vac_sense for accurate input power
    measurement
        Uint8 ipm_pointer;//pointer for current used measurement for IPM compensation

    /****** IN THIS STRUCT, DO NOT ADD ANY NEW VARIABLES ABOVE THIS LINE ************/
    /******* you can add new variable or delete variable from here after ******************************/
    …
    …
    …
}
```


I. These global variables are used in ipm.lib, do not modify:

Input parameters	bit(Value range)	description
Uint32 iin_raw	12	ADC measurement result for Iin
Uint32 vin_raw;	12	ADC measurement result for Vin
Uint32 half_cycle_counter_filtered;	32	how many 100us intervals in half AC cycle
Uint16 cir_buff[64];	12	buffer for vin ADC measurement
Uint8 emi_pointer;	6	pointer to the buffer
Uint32 ipm_filter_shift;	4	Recommend ipm_filter_shift = 11

EXTERN Uint32 emi_capacitance;		the total capacitance of EMI filter(include the one right after bridge rectifier) in nF
EXTERN Uint32 emi_resistance;		the total resistance of EMI filter in mohm
EXTERN Uint32 emi_discharge_resistance;		discharge resistor in EMI filter in Kohm, put 10000 if there is no such resistor
EXTERN Uint32 iin_slope;	10	calibration value
EXTERN Uint32 iin_slope_shift;	4	calibration value
EXTERN Uint32 iin_offset;	10	calibration value
EXTERN Uint32 iin_offset_shift;	4	calibration value
EXTERN Uint32 vin_slope;	10	calibration value
EXTERN Uint32 vin_slope_shift;	4	calibration value
EXTERN Uint32 vin_offset;	10	calibration value
EXTERN Uint32 vin_offset_shift;	4	calibration value
Output parameters	**bit(Value range)**	**description**
EXTERN Uint32 iin_rms;	16	Iin RMS value (mA)
EXTERN Uint32 pin;	32	Pin value (0.1W)
EXTERN Uint32 vin_rms;	16	Vin RMS value (V)
Intermedia parameters		**description**
Uint32 iin_filtered;		intermediate variable, only used in ipm.lib
Uint32 iin_squared;		intermediate variable, only used in ipm.lib
unsigned long long iin_squared_filtered;		intermediate variable, only used in ipm.lib
Uint32 iin_filtered_reporting;		intermediate variable, only used in ipm.lib
Uint32 vin_filtered_reporting;		intermediate variable, only used in ipm.lib
Uint32 pin_filtered_reporting;		intermediate variable, only used in ipm.lib
unsigned long long iin_squared_filtered_reporting;		intermediate variable, only used in ipm.lib
unsigned long long vin_squared_filtered_reporting;		intermediate variable, only used in ipm.lib
Uint32 vin_filtered;		intermediate variable, only used in ipm.lib
Uint32 vin_squared;		intermediate variable, only used in ipm.lib
unsigned long long vin_squared_filtered;		intermediate variable, only used in ipm.lib
Uint32 pin_raw;		intermediate variable, only used in ipm.lib
unsigned long long pin_filtered;		intermediate variable, only used in ipm.lib

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