1. Overview

This design reference manual describes a solution for a three-phase power meter based on the MKM34Z256VLQ7 microcontroller. This microcontroller is part of the Freescale Kinetis-M microcontroller family.

The three-phase power meter reference design was designed in compliance with the China state grid corporation Q/GDW 354—2012 Functional Specification for Smart Electricity Meters and standard Q/GDW 356—2012 Type Specification for Smart Poly-phase Electricity Meters.

The reference design aims at reducing the time to market for power meter customers and partners, and tailor the design to the customers’ unique needs in their powering system.

2. Key features

- Voltage Range: 3 x 220 V/380 V
- Current Range: 5(100)A
- Active Accuracy (−40°C ~ 70°C): 0.5 S
- Reactive Accuracy (−40°C ~ 70°C): 2 S
- RTC (−40°C ~ 70°C): 5 ppm
- Active, reactive, and apparent energy
- Active, reactive, and apparent power
Development environment

- RMS of voltage and current
- Line frequency measurement (for precision zero cross detection)
- Current transformers sensing circuit implementation
- Low-power modes effectively implemented, including the use of the built-in RTC
- LCD display, 8x32 segments
- LEDs pulse outputs (kWh, kVARh)
- Temper detection and records
- Phase missing detection
- Calibration UI tools
- Communication I/F (Isolated IR; Isolated RS232/485)
- ISO7816 smart card read and writes

3. Development environment

3.1. Hardware environment

- Debugger: P&E Micro multilink
- Test instrument: 3PH calibration instrument (Model No: KP-P2001-C)
- Initial Calibration Condition: Three-Phases, 220 V/5 A, initial angle is 60 degrees, 0.5 L
- MCU: KM34Z256

3.2. Software environment

- IDE: IAR 7.20
- OS: No OS
4. System architecture

Figure 1. System architecture
5. Software

5.1. Software workflow

Figure 2. Software workflow

5.2. Core modules

5.2.1. ZCD: Zero Cross Detect

Zero Cross Detect (ZCD) is used to calculate the power line frequency. It uses a comparator (CMP) to detect if the voltage reaches a certain threshold. When the CMP interrupt occurs, the trigger timer runs, and the timer capture function records the frequency.
5.2.2. **Sampling**

AFE (SD ADC) is used for sampling current. SAR ADC is used for sampling voltage. Compensation technology ensures that the voltage and current are sampled simultaneously.

5.2.3. **Algorithm lib**

Use the sampled data to calculate the active energy, reactive energy, apparent energy, and so on.

1. Input sampled value is 24 bits.

Example:

Voltage is 16 bits; please convert to 24 bits as below:
u24_sample = u24_sample << 8;

The current sample value is 24 bits. Its value should not change.

2. The filter coefficient is based on 1200 Hz sample rate, so data should be sent to meter lib with 1200 Hz.

To change the sample rate, use the “Filter-Based Metering Algorithms Configuration Tool” to create a new set of coefficients.
5.2.4. **RTC compensation lib**

RTC output accuracy reaches 5 ppm in a full temperature zone.

5.2.5. **Segment LCD display**

The segment LCD display shows values of active power, reactive power, voltage, and current of each phase.

![Segment LCD display](image)

*Figure 7. Segment LCD display*

5.2.6. **RS485 console**

The RS485 console shell can display current power and voltage, and compensate temperature.

![RS485 shell](image)

*Figure 8. RS485 shell*
5.2.7. **Smart card**

The smart card module provides a full stack of ISO7816 that you can perform smart card read, write, and so on.

5.2.8. **SPI Flash**

The SPI Flash module provides interfaces to easily access flash via the SPI.

5.3. **Source code structure**

The below figure shows the root directory description:

![Root directory description](image)

**Figure 9. Root directory description**

- **build**: IAR project files.
- **km3x_bm_driv**: KM34Z256VLQ7 drivers.
- **src**: Source code of project.
- **template**: Template tool to create a new project.
- **utils**: Debug console utilities.
Project source code directory description, located in the src\3ph_power_metering:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>calib_data</td>
<td>2015/5/29</td>
<td>File folder</td>
</tr>
<tr>
<td>esd_test</td>
<td>2015/5/4</td>
<td>File folder</td>
</tr>
<tr>
<td>freq_detect</td>
<td>2015/5/26</td>
<td>File folder</td>
</tr>
<tr>
<td>key_handler</td>
<td>2015/6/4</td>
<td>File folder</td>
</tr>
<tr>
<td>led</td>
<td>2015/6/4</td>
<td>File folder</td>
</tr>
<tr>
<td>loading_calc</td>
<td>2015/6/2</td>
<td>File folder</td>
</tr>
<tr>
<td>metering_func</td>
<td>2015/4/1</td>
<td>File folder</td>
</tr>
<tr>
<td>nvm_config</td>
<td>2015/5/26</td>
<td>File folder</td>
</tr>
<tr>
<td>pit_trigger</td>
<td>2015/5/26</td>
<td>File folder</td>
</tr>
<tr>
<td>pulse_out</td>
<td>2015/5/26</td>
<td>File folder</td>
</tr>
<tr>
<td>rs485</td>
<td>2015/5/26</td>
<td>File folder</td>
</tr>
<tr>
<td>rtc_comp</td>
<td>2015/5/26</td>
<td>File folder</td>
</tr>
<tr>
<td>rtc_btn</td>
<td>2015/6/3</td>
<td>File folder</td>
</tr>
<tr>
<td>sf_access</td>
<td>2015/4/9</td>
<td>File folder</td>
</tr>
<tr>
<td>slcd_disp</td>
<td>2015/6/12</td>
<td>File folder</td>
</tr>
<tr>
<td>smart_card</td>
<td>2015/6/2</td>
<td>File folder</td>
</tr>
<tr>
<td>temp_sense</td>
<td>2015/4/14</td>
<td>File folder</td>
</tr>
<tr>
<td>uart_handler</td>
<td>2015/4/13</td>
<td>File folder</td>
</tr>
<tr>
<td>vref_comp</td>
<td>2015/3/17</td>
<td>File folder</td>
</tr>
<tr>
<td>work_mode</td>
<td>2015/6/4</td>
<td>File folder</td>
</tr>
<tr>
<td>3ph_power_metering.c</td>
<td>2015/6/4</td>
<td>C Source</td>
</tr>
<tr>
<td>3ph_power_metering.h</td>
<td>2015/5/27</td>
<td>C/C++ Header</td>
</tr>
<tr>
<td>appconfig.h</td>
<td>2015/4/18</td>
<td>C/C++ Header</td>
</tr>
<tr>
<td>freemaster_cfg.h</td>
<td>2015/4/13</td>
<td>C/C++ Header</td>
</tr>
<tr>
<td>metering_const.h</td>
<td>2015/4/22</td>
<td>C/C++ Header</td>
</tr>
<tr>
<td>metering_modules.h</td>
<td>2015/6/10</td>
<td>C/C++ Header</td>
</tr>
<tr>
<td>meterlib_cfg.h</td>
<td>2015/5/27</td>
<td>C/C++ Header</td>
</tr>
</tbody>
</table>

- **calib_data**  Calibration module. Calibration calculates phase compensation parameters.
- **esd_test**  Esd test module. Code for esd test.
- **freq_detect**  Frequency detect module.
- **key_handler**  Pin GPIO interrupt handler. The keys are SW3, SW4, SW5, smart card detect pin and battery pin.
- **led**  LED operation APIs, including on, off, toggle, and status.
- **loading_calc**  CPU loading calculation module.
- **metering_func**  Main metering functions, including sampling, metering, and so on.
- **nvm_config**  Flash configuration data access functions.
- **pit_trigger**  PIT interrupt handler.
- **pulse_out**  Energy accumulation and pulse out.
- **rs485**  RS485 console. A simple shell is supported.
- **rtc_comp**  RTC compensation module.
- rtc_evt  RTC interrupt handler.
- sf_access  SPI Flash access APIs. Including probe, read, write, and erase.
- slcd_disp  Segment LCD display APIs. SLCD display engine is included.
- smart_card  Smart card access functions.
- temp_sense  Temperature sensation and calculation.
- uart_handler  UART interrupt handler. Functions using uart interrupt are smart card, rs485, esam, and IrDA.
- vref_comp  VREF compensation logic code. As VREF module in KM3x is improved, these codes are not used.
- work_mode  Meter work mode switch functions such as VLPR to VLPS, RUN to VLPR, VLPR to RUN, and so on.
- 3ph_power_metering.c  Main function, normal initialization, and lowest priority operations.
- 3ph_power_metering.h  Configurations of all functions.
- appconfig.h  Driver configuration macros.
- freemaster_cfg.h  FreeMASTER configuration macros.
- metering_const.h  Initialization const for metering parameters.
- metering_modules  Contain all module header file including.
- meterlib_cfg.h  Meterlib configurations.

MCU driver and meter library, located in km3x_bm_drv\src:

![Figure 10. Driver code and meter lib description](image)

- common  Initialization and common code for MKM34Z256.
- drivers  Peripheral driver code.
- fraclib  Float mathematics library.
- freemaster  FreeMASTER code.
- freertos  FreeRTOS code.
- meterlib  Metering library.
5.4. **Meter configurations**

Meter configurations are a set of macros in 3ph_power_metering.h. The user can enable or disable macros based on the requirements of each.

5.4.1. **POWER_METERING_DEBUG**

This macro is used when the user uses Serial Wire Debug (SWD) for debugging.

5.4.2. **POWER_METERING_SC_DEBUG**

This macro enables log output of the smart card module.

5.4.3. **POWER_METERING_PASSWORD**

This macro defines the application password, which is needed by FreeMASTER. Enter the password to recalibrate and/or save parameters into flash in FreeMASTER.

5.4.4. **CAL_CURR**

This macro defines the value of calibration current. The test bench needs to output this current for calibration.

5.4.5. **CAL_VOLT**

This macro defines the value of calibration voltage. The test bench needs to output for calibration.

5.4.6. **METER_CL**

This macro is not used yet. This macro cannot be shown in the power meter, so it is currently reserved in text. This may be changed in this future to be used with LCD.

5.4.7. **METER_SN**

This macro is not used yet. This macro is reserved and supposed to be used in FreeMASTER. In the future, more FreeMASTER support may be added and this macro would be used.
5.4.8. **POWER_METERING_PHASE_NUM**

This macro defines the phase number of the meter. For a three-phase power meter, set this value to 3.

5.4.9. **POWER_METERING_SYS_CLK_FREQ**

This macro defines the system clock definition. The value of this macro can be set to 72000000, 48000000, or 24000000. The value of macro POWER_METERING_BUS_CLK_FREQ is set accordingly.

5.4.10. **POWER_METERING_BUS_CLK_FREQ**

This macro defines the bus clock macro. The value of the bus clock is up to 24 M, and is set automatically according to the settings of POWER_METERING_SYS_CLK_FREQ.

5.4.11. **POWER_METERING_SYS_CLK_FEE_MODE** and **POWER_METERING_SYS_CLK_FEI_MODE**

Only one of these two macros can be set to 1.

FLL Engaged External (FEE) uses an external crystal/oscillator as an FLL source to output the MCG core clock.

FLL Engaged Internal (FEI) uses an internal crystal/oscillator as an FLL source to output the MCG core clock.

POWER_METERING_SYS_CLK_FEE_MODE sets the MCU to FEE mode, and
POWER_METERING_SYS_CLK_FEI_MODE sets the MCU to FEI mode.

5.4.12. **POWER_METERING_ENABLE_ESD_TEST_MODE**

This macro enables ESD testing features, which is special for ESD testing. Enable this macro before doing ESD testing.

5.4.13. **POWER_METERING_ENABLE_CPU_LOADING_TST**

This macro enables CPU loading testing features. Enable this macro to get the CPU to load information.

5.4.14. **POWER_METERING_CALI_ENABLE_ITERATION**

This macro enables iteration mode in calibration. In iteration mode, the calibration does iteration on phase parameters provided from the previous calibration.

5.4.15. **POWER_METERING_CALI_ITERATION_ONCE** and
Only one of these two macros can be set to 1. POWER_METERING_CALI_ITERATION_ONCE means calibration only does iteration once, while POWER_METERING_CALI_ITERATION_NO_LIMIT means calibration does iteration repeatedly until it reaches the POWER_METERING_CALI_Q_TARGET.

5.4.16. **POWER_METERING_CALI_SKIP_TIMEOUT**

This defines the calibration pre-processing time delay. Before calibration, the meter waits a while to receive a stable output from the test bench. This macro defines the time by seconds.

5.4.17. **POWER_METERING_CALI_INIT_ANGLE**

This defines the calibration start angle. In 1.0 L, the angle is zero degrees, while in 0.5 L, the angle is 60 degrees. For example, 0.5 L = 60 degrees, 60 = π/3 = 1.04719755119659.

The user can select zero degrees, 45 degrees, or another angle for this calibration. When calibrating, the value of this angle is removed.

5.4.18. **POWER_METERING_CALI_Q_TARGET**

This defines the calibration target of reactive power. In calibration, the value of POWER_METERING_CALI_Q_TARGET is a target. When the calculated reactive power is located in [POWER_METERING_CALI_Q_TARGET, POWER_METERING_CALI_Q_TARGET + 1], the calibration finishes. This macro is only useful when POWER_METERING_CALI_ITERATION_NO_LIMIT is set to 1.

5.4.19. **POWER_METERING_ENABLE_LOAD_CALIB_DATA**

The calibration data is loaded from flash every time the system is initialized. If this macro is set to 0, the default value is used.

5.4.20. **POWER_METERING_USE_PDB_TRIGGER_ADC and POWER_METERING_USE_QTMR_TRIGGER_ADC**

Only one of these two macros can be set to 1.

In sampling, the procedure is:

a. AFE is connected to PDB or QTmr via XBAR.

b. PDB or QTmr is connected to ADC via XBAR.

c. AFE does current sampling.

d. AFE COC event trigger PDB or QTmr.

e. PDB or QTmr trigger ADC for sampling.
f. ADC does voltage sampling.

This macro selects whether PDB or QTmr triggers ADC for sampling. QTmr is easier, but requires three QTmr channels to trigger three ADC channels. PDB is more complex, but only one PDB is required to trigger three ADC channels, so it is preferred. On-chip hardware resources can be saved.

Because QTmr has already been used in frequency detection and clock of the smart card, POWER_METERING_USE_QTMR_TRIGGER_ADC needs to disable the smart card.

5.4.21. **POWER_METERING_ENABLE_WATCHDOG**

This macro enables watchdog.

5.4.22. **POWER_METERING_STARTUP_DELAY**

This defines a delay that sets up a waiting time for data to be stable for calibration.

5.4.23. **POWER_METERING_SAR_CONT**

This defines a delay used in AFE. The interval between a two AFE channel sampling is $2 \times POWER_METERING_SAR_CONT$, which is $(2 \times POWER_METERING_SAR_CONT \times 1000000 / AFE_CLK)$ us. In the current design, POWER_METERING_SAR_CONT is 70, AFE clock is 6144000 Hz, so the interval is about 22.8 us.

5.4.24. **POWER_METERING_PDB_CONT**

This defines the PDB delay value of the channels’ pre-trigger for ADC to sample voltage. In AFE sampling, the interval between each AFE sampling is about 22 us, which in PDB, is $(2 \times POWER_METERING_SAR_CONT / 6.144) \times BUS_CLK / 1000000 = 22 \times 24 = 546.875 \sim 547$ counts. Therefore, this value is set to 547.

5.4.25. **POWER_METERING_CT_COMP_POLARITY**

This macro indicates if voltage is sampled ahead of the current, or if the current is sampled ahead of voltage. In most situations, the voltage is sampled ahead of current, so this value should be set to 1.

5.4.26. **POWER_METERING_VOLTAGE_ADC_CH_PHx**

ADC sampling channels for voltage.

5.4.27. **POWER_METERING_PDB_TRIG_ADC_DIRECTLY**

If the set SIM_MISC_CTL[PDBADCTRIG] bit uses the PDB to trigger the ADC directly, XBAR does not need to be used.
For PDB module in RM 42.2.6, the counter is paused when the processor is in Debug mode, and the counter for the ADC trigger is also paused in Debug mode. The PDB and ADC counter are paused. Setting this macro to 1 causes the ADC to not work in IAR Debug mode.

5.4.28. **POWER_METERING_ENABLE_OSC32K_LOAD_CAPS**

Enabling this macro selects load capacitors for operation with a 32 KHz RTC oscillator.

5.4.29. **POWER_METERING_FLASH_CONFIG_ADDR**

This macro sets an address in flash to save compensation parameters.

5.4.30. **POWER_METERING_RUNTIME_DATA_VALID_FLAG**

This macro defines a flag that is used to verify if the runtime data is valid.

5.4.31. **POWER_METERING_ENABLE_FMSTR**

This macro enables freemaster.

5.4.32. **POWER_METERING_ENABLE_VREF_COMP**

This macro enables VREF compensation.

5.4.33. **POWER_METERING_USE_EXTERNAL_VREF**

Enable this macro when using external VREF. The External VREF uses a different configuration and does not need to be trimmed.

5.4.34. **POWER_METERING_ENABLE_RTC_COMP**

This macro enables RTC compensation.

5.4.35. **POWER_METERING_EXTERNAL_TEMP_AD**

This macro defines which AD is used for external temperature.

5.4.36. **POWER_METERING_TEMP_SAMPLE_NUM**

In temperature calculation, use an array to record a number of samples for averaging. This macro defines the number of samples used for averaging.
5.4.37. **POWER_METERING_TEMP_COMP_ARRAY_SIZE**

This macro defines the array size of the temperature compensation table. The temperature compensation table contains compensation values from -40 degrees C to +70 degrees C. In total, there are 111 items.

5.4.38. **POWER_METERING_SET_RESET_PIN_TO_GPIOOUT**

This macro sets the RESET pin to GPIO, and is driven low to improve EMC susceptibility.

5.4.39. **POWER_METERING_ENABLE_CLKOUT_SRC**

This macro defines the clock source that is routed to PTF7.

The value of this macro should use macros defined in sim.h.

```c
#define CLKOUT_DISABLED (uint32)0x00 ///< Disabled
#define CLKOUT_SRC1     (uint32)0x01 ///< Gated core clock
#define CLKOUT_SRC2     (uint32)0x02 ///< Bus clock
#define CLKOUT_SRC3     (uint32)0x03 ///< LPO clock from PMC
#define CLKOUT_SRC4     (uint32)0x04 ///< IRC clock from MCG
#define CLKOUT_SRC5     (uint32)0x05 ///< Source selected in SIM_SOPT1 [OSC32KSEL];
                               /// See @ref SIM_SelOsc32kClk
#define CLKOUT_SRC6     (uint32)0x06 ///< OSCERCLK external reference clock
#define CLKOUT_SRC7     (uint32)0x07 ///< PLL clock output from MCG
```

5.4.40. **POWER_METERING_SLCD_REFRESH_FREQ_HZ**

This macro defines the refresh frequency of segment LCD. Normally, a frequency of 1 Hz is enough.

5.4.41. **POWER_METERING_PULSE_NUM**

This macro defines the pulse number constant, which is also known as grid constant (C). This macro decides how many active and reactive pulses are the output for 1 KWh.

5.4.42. **POWER_METERING_PULSE_WIDTH**

This macro defines default pulse width. This value is found from grid specification, and the unit is µs. If the pulse interval is less than 80 ms, use 50% of the duty waveform.

5.4.43. **POWER_METERING_PULSE_PIT_INTERVAL**

This macro defines the energy accumulate interval. The default interval is 20 µs, or 50 KHz. In an experiment, the maximum value for the interval may be 50 KHz.
5.4.44. **POWER_METERING_PULSE_PIT_FREQ**

This macro defines the energy accumulation frequency. The value can be obtained by \( \frac{1000000}{\text{POWER_METERING_PULSE_PIT_INTERVAL}} \).

5.4.45. **POWER_METERING_PULSE_PIT_LDVAL**

This macro defines the PIT LDVAL value for pulse interval. This value can be obtained by \( \frac{\text{POWER_METERING_BUS_CLK_FREQ}}{\text{POWER_METERING_PULSE_PIT_FREQ}} \).

5.4.46. **POWER_METERING_PULSE_HOLDING_COUNT**

This macro defines an 80 ms holding count value for PIT. This value can be obtained by \( \frac{\text{POWER_METERING_PULSE_WIDTH}}{\text{POWER_METERING_PULSE_PIT_INTERVAL}} \).

5.4.47. **POWER_METERING_PULSE_THRES**

This macro defines a pulse threshold that, when accumulated energy exceeds the threshold, a pulse is sent out. The value of this macro is calculated in below procedure.

Each KWH has \( \text{POWER_METERING_PULSE_NUM} \) pulses, so each pulses indicates \( \frac{(60 \times 60 \times 1000)}{\text{POWER_METERING_PULSE_NUM}} \).

Our energy increases every \( \text{POWER_METERING_PULSE_PIT_INTERVAL} \) by \( \frac{\text{P}}{1000000} \).

Multiply two sides by \( \frac{1000000}{\text{POWER_METERING_PULSE_PIT_INTERVAL}} \), so \( \text{POWER_METERING_PULSE_THRES} = \frac{(3600 \times 1000 \times 1000000)}{\text{POWER_METERING_PULSE_PIT_INTERVAL} \times \text{POWER_METERING_PULSE_NUM}} \).

5.4.48. **POWER_METERING_PERIOD_TOTAL**

This macro defines the total power that exists in one pulse.

5.4.49. **POWER_METERING_POWER_100MS**

When the pulse interval is less than 100 ms, meters output 50% of duty pulses. This macro defines the energy threshold for 100 ms.

5.4.50. **POWER_METERING_CALIB_GRID_FREQ**

This macro defines the grid frequency used for calibration. It means that, for calibration, the test bench needs to set the grid frequency to this value.

5.4.51. **POWER_METERING_GRID_FREQ_MIN and**


**POWER_METERING_GRID_FREQ_MAX**
This macro defines the maximum and minimum value for the grid frequency. This is needed in frequency detection to know whether the frequency is valid or not.

5.4.52. **POWER_METERING_FREQ_DET_CHx**
These macros define a series of channel used for frequency detection. If one channel fails to get a valid frequency, another channel could be used.

5.4.53. **POWER_METERING_FREQ_CALC_TMR_DIV**
This macro defines the clock divisor for TMR, which is used for frequency detection.

5.4.54. **POWER_METERING_SMART_CARD_CLOCK_FREQ**
This macro defines smart card clock frequency. This clock is sourced from QTmr3, and the range is from 1 M to 5 M.

5.4.55. **POWER_METERING_SMART_CARD_ISO7816_CARD**
This macro defines if need to support ISO7816 card.

5.4.56. **POWER_METERING_RS485_UART_BAUD_RATE**
This macro defines the baud rate of RS485 port.

5.4.57. **POWER_METERING_RS485_PARSE_CMD_NUM**
This macro defines number of commands supported in shell.

5.4.58. **POWER_METERING_RS485_SHELL_BUFFER_SIZE**
This macro defines the size of shell buffer. This buffer is to store user input temporarily.

5.4.59. **POWER_METERING_RS485_SHELL_MAX_ARGS**
This macro defines maximum arguments supported in shell command.

5.4.60. **POWER_METERING_RS485_SHELL_PROMPT**
This macro defines prompt of shell.
6. Revision history

The revision history of the design reference manual is shown in the following table.

Table 1. Revision history

<table>
<thead>
<tr>
<th>Revision number</th>
<th>Date</th>
<th>Substantive changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>07/2015</td>
<td>Initial release</td>
</tr>
</tbody>
</table>
Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts.

Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, and Kinetis are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All rights reserved.

© 2015 Freescale Semiconductor, Inc.