Power Quality Meter

Software Description

Version: 0.2,
Data: 20-Jul-07
Project: Power Quality Meter
Author: Iliya D. Voronov
Http: http://powerdsp.narod.ru/
Email: powerdsp@narod.ru

Table of Contents

1 Introduction

This document containes general description of solftware developed according to Power Quality Meter project.

1.1 Primary software

Power Quality Meter project consist in developing software to implement 3-phase power metering functionality according to IEC 61000-4-30, IEC 61000-4-7 and IEC 61000-4-15 standards. The software is called below DSP Engine. DSP Engine measures following characteristic of power:

1. Phase-to-neutral RMS voltage and current, phase-to-phase RMS voltage, frequency.
2. Real, apparent and reactive power and energy, leading/lagging angle and power factor.
3. Harmonics and sub-grouped harmonics up to 50-th, inter-harmonics and sub-harmonics, above harmonics up to 9 kHz for voltage and current.
4. Total harmonic distortions, crest factor, transformer derating factor, K-factor.
5. Voltage and current imbalance.
6. Flicker.

DSP Engine implements following additional functionality:

1. Aggregation of measured values for intervals of 3 seconds, 10 minutes, 2 hours, auto and user specified interval, demand value calculation.
2. Event detection at measured value crosses specified thresholds.
3. Waveform capture on event detection and at user specified interval.

DSP Engine is developed for PC and for PQM instrument hardware; it is delivered as DSP Engine DLL and as firmware for PQM Instrument. DSP Engine DLL implements data processing functions only; it does not cover data input/output. It can be used for offline data processing with file input/output or for real-time data processing. DSP Engine DLL interface is described in . DSP Engine DLL project is `C:\PowerMeter\src\msvcEngineDll\EngineDll.dsp`, it generates `C:\PowerMeter\src\DspEngine.dll`.

DSP Engine for target hardware is integrated with real-time framework and runs on PQM hardware. Target DSP Engine, real-time framework and hardware constitute PQM Instrument. In most cases below, whole PQM Instrument is used as synonym to target DSP Engine software. In some cases, then PQM software is contrasted to PQM hardware, target DSP Engine is called PQM Firmware. PQM Firmware project is `C:\PowerMeter\src\5509TargFrw\ targFrw.pjt`, it generates `C:\PowerMeter\src\targFrw.out`. PQM Firmware include library project `C:\PowerMeter\src\5509EngineLib\engineLib.pjt`.

PQM Instrument process input samples from 13-bit 8-channel 200 kSPS per channel ADC in real time. Host application control PQM Instrument, read measured power characteristics, events and captured waveform by Serial Communication Protocol through USB. Serial Communication Protocol (below mentioned as SCP) is described in . PQM Instrument also implements hardware testing and PQM Firmware update.

1.2 Auxiliary software

In addition to DSP Engine DLL and target Firmware, following software was developed:

1. Test Bench
2. Host Application
3. Hardware Test Application
4. Firmware Update Application

Test Bench is designed to test DSP Engine performance in non-real-time. Test Bench parses configuration file, configure DSP Engine, read input samples from file to supply it to DSP Engine, save measurement results, event and captured waveforms to output files. DSP Engine calculation and input/output data processing performed on the same PC. Test Bench can be used as prototype for Power Quality Meter offline application. Test Bench project is `C:\PowerMeter\src\msvcTestBench\TestBench.dsp`, it generates `C:\PowerMeter\src\TestBench.exe`.

Host Application is designed to test Firmware performance in real-time on PQM instrument. It communicates with PQM instrument through USB. Host Application read and parse configuration file and configure Firmware according to it. Next it is continuously polling measurement results, events and captured waveforms from

Firmware and save them to output files. In contrast to Test Bench, DSP Engine calculation is performed on PQM Instrument, while file input/output operation is performed on PC. Host Application project is `C:\PowerMeter\src\msvcHostPoll\hostPoll.dsp`, it generates `C:\PowerMeter\src\hostPoll.exe`.

Hardware Test Application is designed to instruct PQM instrument to run hardware tests and display its results on PC. Hardware Test Application project is `C:\PowerMeter\util\usbTest\pollDspHost\pollDspHost.dsp`, it generates `C:\PowerMeter\util\usbTest\pollDspHost\pollDsp.exe`.

Firmware Update Application is designed to update PQM Firmware. Firmware Update Application project is `C:\PowerMeter\util\usbTest\flashProgHost\flashProgHost.dsp`, it generates `C:\PowerMeter\util\usbTest\flashProgHost\flashProg.exe`.

In general, auxiliary software can be considered as integration examples for DSP Engine DLL and as prototype software for PQM Instrument.

1.3 Development tools and utilities

PC software is developed under MS Visual C++ 6.0. Target software is developed under Code Composer Studio 3.1.

LibUsb-Win32 0.1.12 library was used as USB driver for all host applications, interfacing with PQM Instrument. More details on LibUsb can be found at http://libusb.sourceforge.net/.

At first connection of Power Quality Meter, Windows will ask for driver for the USB device. Windows wizard should be pointed to file stateVector.inf, \libusb-win32-device-bin-0.1.12.1\bin\libusb0.sys and \libusb-win32-device-bin-0.1.12.1\bin\libusb0.dll files in the ordinal way.

2 Testing DSP Engine DLL and PQM Instrument.

As shown above, auxiliary software Test Bench and Host Application is used for testing primary software. They functions are similar: both of them run on PC, read their configuration from configuration file, configure DSP Engine DLL or PQM Instrument according to it, save message log, measurement results, events and captured waveforms to output files. But they differ in following things: Test Bench runs offline, interfaces directly to DSP Engine DLL, feeds to it input samples from file; Host Application runs in real-time, controls PQM Instrument through SCP/USB, electrical input signals should be supplied to PQM Instrument externally.

2.1 Configuration

Configuration file is case insensitive text file with CR LF (0x0D 0x0A) line terminating. Comments starts from ; (semicolon) character, all character after ; ignored, one command per line is allowed only. Command starts from command name with following parameters, if several parameters are allowed, they should be separated by white character and/or , (comma) character. Unrecognized command are ignored, error message is generated. Commands are shown in Table 1. Configuration commands.

Command	Default	Description
`infile "string" <1-10000>`	`sample.pcm`	Specify input file name, and number of repetition of file, default 1. Is not applicable for Host Application.
`3phmode <0,1>`	0	Input sample file mode: 0 - single phase, 1- 3-phase + neutral
`enaevent <0-2>`	0	Enable Event generation 0 – disable all events 1 – enable all events, excepting user event 2 – enable all events
`logtime <1-10000>`	100	Period of reading measurement results and saving them to output files, in milliseconds
`logpar <parameter name>`	empty	List of parameter from measurement results structure, to be saved to output file. Full parameter list is shown in Table 3. Measurement results parameters name. Parameter lists for consecutive `logpar` commands are concatenated.

`run "string"`	empty	Run session, also specify session name

Table 1. Configuration commands.

Input file name specify file name, Test Bench get input samples, it is not used for Host application. If non single repetition number is specified, after file is completely read, it is rewind and reread specified number of time. There are two input mode: Single phase mode and 3-phase mode. Single phase mode simulates single phase connection to power circuit. 3-phase mode simulates 3-phaze 4 wire connection. Single phase mode also disables all events for B, C and N phase, phase-to-phase voltage and imbalance.

There are three level of event generation: all events are disabled, all events excepting periodic user events are disable, all events are enable. It is used to reduce number of generated files during testing procedure.

Specified Parameters fields of Measurement Results is periodically read from DSP Engine DLL or PQM Instrument and saved to Measurement Result files. Full list of parameters can be found in Table 3. Measurement results parameters name. It covers most of parameters for phase A and several Total parameters, it is enough to test and demonstrate DSP Engine DLL and PQM Instrument performance.

Run command indicates end of configuration and starts data processing session. Specified session name string is added to all input and output files to distinguish similar files for different sessions. Test Bench session duration is determined by input file length, thus several session can be processed sequentially from the same configuration file. Host Application session is infinite, only first session in file makes sense; session is interrupted by terminating application.

Here is example of configuration file (`C:\PowerMeter\src\testCases\707to10v.txt`):

```
infile 707to10v.pcm
logtime 200
logpar v_per, v_rms, v_under, v_over, v_sr, v_crest, v_derat
logpar c_rms, vpp_rms, apppwr
run 707to10v_
```

It specifies session name prefix `707to10v_`, input file name `707to10v.pcm`, `following` measured results for phase A saved to file: voltage for cycle, rms voltage for 10/12 cycles, under voltage, over voltage, voltage sliding reference, voltage crest factor, voltage derating factor, rms current for 10/12 cycles, phase-to-phase voltage, apparent power, saving interval is 200 ms.

2.2 Running Test Bench and Host Application

Both Test Bench and Host Application are command line application; they take up to two command line arguments. First optional argument is configuration file name, `config.txt` is used by default. Second optional argument is log file name, `msg_log.txt` is used by default. `DspEngine.dll` file should be in the same directory as `TestBench.exe`.

Here is the example of command line: `TestBench.exe 707to10v.txt`

All debug messages, error messages and events, generated by DSP Engine DLL and Test Bench or PQM Instrument and Host Application are displayed on screen and stored to log file. Critical error messages starts from "!!!" mark. All messages are prefixed with time of message printing in milliseconds.

2.3 Input sample file

Input sample file format is 16-bit Intel Raw PCM Stereo, sample rate 20 kSPS. Left channel is the voltage of the particular phase, format S10Q5, full scale range +/- 1024V, step 0.03V. Right channel is the current, format S12Q3, full scale range +/- 8.192A, step 0.00025A. File can be edited in any sound editor, for example Adobe Audition.

DSP Engine DLL sample rate is 200 kSPS, it is 10 times more than input file sample rate. Input file samples are repeated 10 times before feeding to DSP Engine DLL to get required sample rate.

In Single phase mode samples from file comes to Phase A input of DSP Engine DLL. Zero samples are supplied to Phase B, Phase C and Neutral inputs. In 3-phase mode, 4 input files are used; samples from each file come to particular input of DSP Engine DLL. `_A`, `_B`, `_C`, `_N` suffixes are added to file name (before extension) to designate phase. In any case, name is prefixed with session name.

Input sample file name is used by Test Bench only. To run Host Application testing, real electrical signal should be supplied to input of PQM Instrument. Possible decision is connecting sound card output to PQM Instrument input and playing back the same files input sample file. Special attention should be paid on proper signal scaling and synchronization.

2.4 Measurement result file

Measurement results file is ASCII text file. Each line contains one parameter, delay from line to line defined by `logtime` value. Parameter can be single value or array, array values are delimited by white space. File name consist from session name, measurement result parameter name and extension `.dat`.

Here is example of single value measurement result file (`C:\PowerMeter\src\testCases\707to10v_v_rms.dat`):

```
0.000
689.250
690.672
690.656
…
```

It is voltage rms voltage for 10/12 cycles values, reported with interval 200 ms, as specify `logtime 200` in configuration file. So rms voltage is 0.000 V at 200 ms, 689.250 V at 400 ms, 690.672 V at 600 ms and so on.

Here is example of array measurement result file (`C:\PowerMeter\src\testCases\harm_v_harm_mag.dat`), line comprises all 50 harmonic values:

```
161.953 0.078 0.063 0.063 0.047 0.047 0.047 0.047 0.047 0.031 0.031 0.047 0.031
0.031 0.031 0.031 0.031 0.047 0.031 0.031 0.031 0.031 0.047 0.031 0.031 0.031
0.031 0.016 0.016 0.031 0.031 0.016 0.031 0.031 0.031 0.031 0.016 0.016 0.031
0.031 0.031 0.031 0.016 0.016 0.016 0.016 0.016 0.016 0.031 0.016
189.266 0.094 0.094 0.078 0.063 0.063 0.047 0.063 0.063 0.063 0.047 0.047 0.031
0.031 0.047 0.031 0.047 0.047 0.047 0.047 0.047 0.031 0.047 0.047 0.031 0.031
0.031 0.031 0.016 0.031 0.031 0.016 0.031 0.031 0.031 0.031 0.031 0.031 0.031
0.031 0.047 0.031 0.031 0.016 0.031 0.031 0.031 0.031 0.031 0.031
…
```

It is voltage harmonics magnitude. So at first time, 1-st (fundamental) harmonics magnitude is 161.953 V, 2-nd harmonics magnitude is 0.078 V, 3-rd harmonics magnitude is 0.063 V and so on.

2.5 Event and Waveform capture files

Notification about detected event saved to event file. It is text file, one detected event per line. For most events excepting User and Transition evnets, event beginning and event end considered to be different events. Following information about event are printed: beginning/end sign, event phase name (phase A formally assigned to imbalance event), voltage/current/phase-to-phase voltage designator (if applicable), event type (magnitude, frequency etc.), event number, event beginning time or event duration. If available, event characterization information is printed.

Here are example event messages:

```
_/''' AC_magn_N4 beg:1.530
'''\_ AC_magn_N4 dur:0.090 Instantaneous Under mean:1.472 min:1.471 max:1.475
```

It means:

1. At phase A current Magnitude Event number 4 begins at 1.53 second

2. At phase A current Magnitude Event number 4 ends, event duration 90 ms, Event qualified as Instantaneous Under-deviation, mean/min/max current value are 1.472/1.471/1.475 A.

Event file name consists from session name, suffix "_evt" and extension `.txt`.

Entire captured waveform is divided to packets, before sending to host. Notification about each received packets saved to waveform packet file. It is text file, one packet per line. Waveform packet information is similar to its initiating event information. Following waveform packet information in addition to event information is printed: waveform phase (some event can initiate waveform capture for several phases), voltage/current/phase-to-phase voltage designator (some current event can initiate voltage waveform capture and vice verse), packet length in samples, decimation rate (reference to 50 kSPS), packet generation time, lost packet flag and last packet in the waveform flag.

Here is example packet information:

```
AC_magn_N4_AC len-812 dec-8 at-1.620 END
```

Waveform capture packet for phase A current is got, initiating event detected at phase A current Magnitude Event number 4. Packet length is 812 samples, decimation rate is 8 or sample rate is 50000 / 8 = 6250 SPS or 125 samples per cycle, packet send at 1.62 second, it is last packet for this waveform.

Waveform packet file name consists from session name, suffix "_wave" and extension `.txt`.

All received waveform capture packets for particular event are assembled and saved to file, file format is 16-bit Intel Raw PCM Mono, sample rate depends on specified decimation rate for particular event waveform capture. File name consists from session name, initiating event, waveform phase and voltage/current/phase-to-phase voltage designator and extension `.pcm`. For example `evmagn_AC_magn_N4_AC.pcm`.

3 Utilities

3.1 Hardware testing

Hardware Test Application is designed to instruct PQM Instrument run hardware tests and display results. It is command line application; it takes up to two command line arguments. First optional argument is number of tests, to be run, any positive value is allowed, 0 or c means continues test until application is terminated, 1 is default value. Second optional argument is comma separated list of tests to be run; by default all tests are run. Test numbers are listed in Table 2. Hardware Test number. Bicolor LED on PQM Instrument is getting red during hardware test processing.

Test number	Test description
0	Read phase A voltage raw ADC value
1	Read phase B voltage raw ADC value
2	Read phase C voltage raw ADC value
3	Read Neutral voltage raw ADC value
4	Read phase A current raw ADC value
5	Read phase B current raw ADC value
6	Read phase C current raw ADC value
7	Read Neutral current raw ADC value
8	Read power supply voltage
9	Read board temperature
10	External memory test
11	Oscillator frequency test
12	WDT test, it is not actually test. It initiates WDT reset.

Table 2. Hardware Test number.

Batch files `runMemTest.bat`, `readAdc.bat`, `readTempVccInt.bat`, `wdtReset.bat` in directory `C:\PowerMeter\util\makeBoot` can be considered as examples.

Here is example program output:

```
C:\PowerMeter\util\usbTest\pollDspHost> pollDsp.exe
found 5 busses
Raw A Volt ADC value:  -5
Raw B Volt ADC value:  -6
Raw C Volt ADC value:  -7
Raw N Volt ADC value:  -6
Raw A Curr ADC value:  -4
Raw B Curr ADC value:  -5
```

```
Raw C Curr ADC value:  -3
Raw N Curr ADC value:  -4
Supply voltage:        4768 mV
Board temperature:     42.50 C
Memory test result:    OK
Interrupt rate:        320028
WDT test result:        WDT will reset CPU in 1.5s
c:\PowerMeter\util\makeBoot>pause
Press any key to continue . . .
```

3.2 Firmware update

Firmware Update Application is designed to update PQM Instrument Firmware in internal flash memory. It is command line application; it takes two command line arguments. First argument file name to be downloaded to instrument. Second argument firmware update password – 32-bit hexadecimal number.

Firmware Update Application download binary image of PQM Firmware. A firmware update operation is possible only for PQM Instruments with already downloaded to flash and running healthy firmware. Bicolor LED on PQM Instrument is getting red during firmware updating.

Incorrect file will destroy PQM Instrument flash contents. Just as attention should be paid to avoid destroying PQM Instrument flash contents by failed firmware update operation due to accidental Firmware Update Application termination or PQM Instrument power or USB connector disconnection.

For downloading firmware in empty flash instrument, or for recovering from failed firmware update, USB boot loading procedure should be used first.

Here is example program output:

```
C:\PowerMeter\util\usbTest\flashProgHost> flashProg.exe usbTest.bin 12345678
found 5 busses
Programming file usbTest.bin of size 65422, Password 12345678
Flash was successfully erased
Packet 0 was successfully written to flash
Packet 1 was successfully written to flash
Packet 2 was successfully written to flash
Packet 3 was successfully written to flash
Packet 4 was successfully written to flash
Packet 5 was successfully written to flash
Packet 6 was successfully written to flash
Packet 7 was successfully written to flash
Flash was successfully programmed, DSP will restart in a second
c:\PowerMeter\util\makeBoot>pause
Press any key to continue . . .
```

3.3 USB boot loading

USB boot loading is used to start PQM Instrument with empty of destroyed internal flash memory. This operation boot loads DSP but do not updates flash, so ordinal firmware update operation should be used after USB boot loading to download firmware to flash.

Thesycon USBIO demo application 2.40 is used for downloading boot image to DSP from USB. Application itself and Thesycon user manual “USB Driver Manual” and other information can be found at http://www.thesycon.com/.

To boot loading DSP from USB follow instructions:

1. power off PQM Instrument
2. connect pins 11 and 12 of P3 connector on DSP board with jumper
3. power on PQM Instrument and connect USB to host computer

4. follow instruction in Chapter 6 “USB Boot Using the Thesycon (USBIO) Driver and Demo Application” from TI document “Using the TMS320C5509/C5509A USB Bootloader“ literature reference number SPRA840A.
5. remove jumper from pins 11 and 12 of P3 connector

Annex A. Measurement result parameters name

Parameter name	Field of tENG_fullRes structure see .
v_per	rec.rms.aVo[0].per
v_rms	rec.rms.aVo[0].rms
v_under	rec.rms.aVo[0].under
v_over	rec.rms.aVo[0].over
v_crest	rec.rms.aVo[0].crest
v_derat	rec.rms.aVo[0].derat
v_sr	rec.rms.aVo[0].sr
vpp_per	rec.rms.aVpp[0].per
vpp_rms	rec.rms.aVpp[0].rms
vpp_under	rec.rms.aVpp[0].under
vpp_over	rec.rms.aVpp[0].over
vpp_sr	rec.rms.aVpp[0].sr
vn_rms	rec.rms.aVo[3].rms
cn_rms	rec.rms.aCu[3].rms
v_harm_mag	rec.harm.aPh[0].harm.aVoMagn[0]
v_harm_sq	rec.harm.aPh[0].harm.aVoSq[0]
v_harm_ang	rec.harm.aPh[0].harm.aVoAng[0]
v_harm_perc	rec.harm.aPh[0].harm.aVoPerc[0]
v_harmsg_mag	rec.harm.aPh[0].harmSg.aVoMagn[0]
v_harmsg_sq	rec.harm.aPh[0].harmSg.aVoSq[0]
v_harmsg_ang	rec.harm.aPh[0].harmSg.aVoAng[0]
v_harmsg_perc	rec.harm.aPh[0].harmSg.aVoPerc[0]
v_inth_mag	rec.harm.aPh[0].inth.aVoMagn[0]
v_inth_sq	rec.harm.aPh[0].inth.aVoSq[0]
v_inth_ang	rec.harm.aPh[0].inth.aVoAng[0]
v_inth_perc	rec.harm.aPh[0].inth.aVoPerc[0]
v_abvh_mag	rec.harm.aPh[0].abvh.aVoMagn[0]
v_abvh_sq	rec.harm.aPh[0].abvh.aVoSq[0]
v_abvh_ang	rec.harm.aPh[0].abvh.aVoAng[0]
v_abvh_perc	rec.harm.aPh[0].abvh.aVoPerc[0]
v_subh_mag	rec.harm.aPh[0].subh.voMagn
v_subh_sq	rec.harm.aPh[0].subh.voSq
v_subh_ang	rec.harm.aPh[0].subh.voAng
v_subh_perc	rec.harm.aPh[0].subh.voPerc
v_dcoffs	rec.harm.aPh[0].voDcoffs
v_fft_harm	rec.rawFft.aPh[0].voHarm[0]
v_fft_abv	rec.rawFft.aPh[0].voAbv[0]
v_thd_thd	rec.harm.aPh[0].voThd.thd
v_thd_ohd	rec.harm.aPh[0].voThd.ohd
v_thd_ehd	rec.harm.aPh[0].voThd.ehd

v_thd_thdsg	rec.harm.aPh[0].voThd.thdSg
v_thd_ohdsg	rec.harm.aPh[0].voThd.ohdSg
v_thd_ehdsg	rec.harm.aPh[0].voThd.ehdSg
v_thd_tihd	rec.harm.aPh[0].voThd.tihd
v_thd_tahd	rec.harm.aPh[0].voThd.tahd
v_thd_tshd	rec.harm.aPh[0].voThd.tshd
v_kfact	rec.harm.aPh[0].voThd.kFact
c_per	rec.rms.aCu[0].per
c_rms	rec.rms.aCu[0].rms
c_under	rec.rms.aCu[0].under
c_over	rec.rms.aCu[0].over
c_crest	rec.rms.aCu[0].crest
c_derat	rec.rms.aCu[0].derat
c_sr	rec.rms.aCu[0].sr
c_harm_mag	rec.harm.aPh[0].harm.aCuMagn[0]
c_harm_sq	rec.harm.aPh[0].harm.aCuSq[0]
c_harm_ang	rec.harm.aPh[0].harm.aCuAng[0]
c_harm_perc	rec.harm.aPh[0].harm.aCuPerc[0]
c_harmsg_mag	rec.harm.aPh[0].harmSg.aCuMagn[0]
c_harmsg_sq	rec.harm.aPh[0].harmSg.aCuSq[0]
c_harmsg_ang	rec.harm.aPh[0].harmSg.aCuAng[0]
c_harmsg_perc	rec.harm.aPh[0].harmSg.aCuPerc[0]
c_inth_mag	rec.harm.aPh[0].inth.aCuMagn[0]
c_inth_sq	rec.harm.aPh[0].inth.aCuSq[0]
c_inth_ang	rec.harm.aPh[0].inth.aCuAng[0]
c_inth_perc	rec.harm.aPh[0].inth.aCuPerc[0]
c_abvh_mag	rec.harm.aPh[0].abvh.aCuMagn[0]
c_abvh_sq	rec.harm.aPh[0].abvh.aCuSq[0]
c_abvh_ang	rec.harm.aPh[0].abvh.aCuAng[0]
c_abvh_perc	rec.harm.aPh[0].abvh.aCuPerc[0]
c_subh_mag	rec.harm.aPh[0].subh.cuMagn
c_subh_sq	rec.harm.aPh[0].subh.cuSq
c_subh_ang	rec.harm.aPh[0].subh.cuAng
c_subh_perc	rec.harm.aPh[0].subh.cuPerc
c_dcoffs	rec.harm.aPh[0].cuDcoffs
c_fft_harm	rec.rawFft.aPh[0].cuHarm[0]
c_fft_abv	rec.rawFft.aPh[0].cuAbv[0]
c_thd_thd	rec.harm.aPh[0].cuThd.thd
c_thd_ohd	rec.harm.aPh[0].cuThd.ohd
c_thd_ehd	rec.harm.aPh[0].cuThd.ehd
c_thd_thdsg	rec.harm.aPh[0].cuThd.thdSg
c_thd_ohdsg	rec.harm.aPh[0].cuThd.ohdSg
c_thd_ehdsg	rec.harm.aPh[0].cuThd.ehdSg
c_thd_tihd	rec.harm.aPh[0].cuThd.tihd
c_thd_tahd	rec.harm.aPh[0].cuThd.tahd
c_thd_tshd	rec.harm.aPh[0].cuThd.tshd

c_kfact	rec.harm.aPh[0].cuThd.kFact
ldangl	rec.harm.aPh[0].leadAng
freq	rec.pwr.aPh[0].freq
fft_sync	rec.harm.isSync
rlpwr	rec.pwr.aPh[0].rlPwr
apppwr	rec.pwr.aPh[0].appPwr
rctpwr	rec.pwr.aPh[0].rctPwr
displpf	rec.pwr.aPh[0].displPf
truepf	rec.pwr.aPh[0].truePf
kwh	rec.pwr.aPh[0].kwh
kvah	rec.pwr.aPh[0].kvah
kvarh	rec.pwr.aPh[0].kvarh
ifl	rec.pwr.aPh[0].ifl
pst	rec.pwr.aPh[0].pst
plt	rec.pwr.aPh[0].plt
harm_pwr	rec.harm.aPh[0].harm.aPwr[0]
harmsg_pwr	rec.harm.aPh[0].harmSg.aPwr[0]
inth_pwr	rec.harm.aPh[0].inth.aPwr[0]
abvh_pwr	rec.harm.aPh[0].abvh.aPwr[0]
subh_pwr	rec.harm.aPh[0].subh.pwr
harm_dir	rec.harm.aPh[0].aHarmDir[0]
harmsg_dir	rec.harm.aPh[0].aHarmSgDir[0]
inth_dir	rec.harm.aPh[0].aInthDir[0]
abvh_dir	rec.harm.aPh[0].aAbvhDir[0]
subh_dir	rec.harm.aPh[0].subhDir
t_ldangl	rec.harm.totalLeadAng
t_freq	rec.pwr.total.freq
t_rlpwr	rec.pwr.total.rlPwr
t_apppwr	rec.pwr.total.appPwr
t_rctpwr	rec.pwr.total.rctPwr
t_displpf	rec.pwr.total.displPf
t_truepf	rec.pwr.total.truePf
t_kwh	rec.pwr.total.kwh
t_kvah	rec.pwr.total.kvah
t_kvarh	rec.pwr.total.kvarh
t_ifl	rec.pwr.total.ifl
t_pst	rec.pwr.total.pst
t_plt	rec.pwr.total.plt
v_seqpos	rec.seq.aSeq[ENG_VCPN_V].pos
v_seqneg	rec.seq.aSeq[ENG_VCPN_V].neg
v_seqzero	rec.seq.aSeq[ENG_VCPN_V].zero
c_seqpos	rec.seq.aSeq[ENG_VCPN_C].pos
c_seqneg	rec.seq.aSeq[ENG_VCPN_C].neg
c_seqzero	rec.seq.aSeq[ENG_VCPN_C].zero
v_imneg	rec.seq.aSeq[ENG_VCPN_V].imNeg
v_imzero	rec.seq.aSeq[ENG_VCPN_V].imZero

c_imneg	rec.seq.aSeq[ENG_VCPN_C].imNeg
c_imzero	rec.seq.aSeq[ENG_VCPN_C].imZero
v_imnema	rec.seq.voImNema
c_imnema	rec.seq.cuImNema
vpp_imnema	rec.seq.vppImNema
v_rms_aggr_3s	ag3s.aggr.rms.aVo[0].rms
c_rms_aggr_3s	ag3s.aggr.rms.aCu[0].rms
rlpwr_aggr_3s	ag3s.aggr.pwr.aPh[0].rlPwr
v_harm_mag_aggr_3s	ag3s.aggr.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_aggr_3s	ag3s.aggr.harm.aPh[0].harm.aCuMagn[0]
v_rms_min_3s	ag3s.min.rms.aVo[0].rms
c_rms_min_3s	ag3s.min.rms.aCu[0].rms
rlpwr_min_3s	ag3s.min.pwr.aPh[0].rlPwr
v_harm_mag_min_3s	ag3s.min.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_min_3s	ag3s.min.harm.aPh[0].harm.aCuMagn[0]
v_rms_max_3s	ag3s.max.rms.aVo[0].rms
c_rms_max_3s	ag3s.max.rms.aCu[0].rms
rlpwr_max_3s	ag3s.max.pwr.aPh[0].rlPwr
v_harm_mag_max_3s	ag3s.max.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_max_3s	ag3s.max.harm.aPh[0].harm.aCuMagn[0]
v_rms_aggr_10m	ag10m.aggr.rms.aVo[0].rms
c_rms_aggr_10m	ag10m.aggr.rms.aCu[0].rms
rlpwr_aggr_10m	ag10m.aggr.pwr.aPh[0].rlPwr
v_harm_mag_aggr_10m	ag10m.aggr.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_aggr_10m	ag10m.aggr.harm.aPh[0].harm.aCuMagn[0]
v_rms_min_10m	ag10m.min.rms.aVo[0].rms
c_rms_min_10m	ag10m.min.rms.aCu[0].rms
rlpwr_min_10m	ag10m.min.pwr.aPh[0].rlPwr
v_harm_mag_min_10m	ag10m.min.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_min_10m	ag10m.min.harm.aPh[0].harm.aCuMagn[0]
v_rms_max_10m	ag10m.max.rms.aVo[0].rms
c_rms_max_10m	ag10m.max.rms.aCu[0].rms
rlpwr_max_10m	ag10m.max.pwr.aPh[0].rlPwr
v_harm_mag_max_10m	ag10m.max.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_max_10m	ag10m.max.harm.aPh[0].harm.aCuMagn[0]
v_rms_aggr_2h	ag2h.aggr.rms.aVo[0].rms
c_rms_aggr_2h	ag2h.aggr.rms.aCu[0].rms
rlpwr_aggr_2h	ag2h.aggr.pwr.aPh[0].rlPwr
v_harm_mag_aggr_2h	ag2h.aggr.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_aggr_2h	ag2h.aggr.harm.aPh[0].harm.aCuMagn[0]
v_rms_min_2h	ag2h.min.rms.aVo[0].rms
c_rms_min_2h	ag2h.min.rms.aCu[0].rms
rlpwr_min_2h	ag2h.min.pwr.aPh[0].rlPwr
v_harm_mag_min_2h	ag2h.min.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_min_2h	ag2h.min.harm.aPh[0].harm.aCuMagn[0]
v_rms_max_2h	ag2h.max.rms.aVo[0].rms

c_rms_max_2h	ag2h.max.rms.aCu[0].rms
rlpwr_max_2h	ag2h.max.pwr.aPh[0].rlPwr
v_harm_mag_max_2h	ag2h.max.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_max_2h	ag2h.max.harm.aPh[0].harm.aCuMagn[0]
v_rms_aggr_user	agUser.aggr.rms.aVo[0].rms
c_rms_aggr_user	agUser.aggr.rms.aCu[0].rms
rlpwr_aggr_user	agUser.aggr.pwr.aPh[0].rlPwr
v_harm_mag_aggr_user	agUser.aggr.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_aggr_user	agUser.aggr.harm.aPh[0].harm.aCuMagn[0]
v_rms_min_user	agUser.min.rms.aVo[0].rms
c_rms_min_user	agUser.min.rms.aCu[0].rms
rlpwr_min_user	agUser.min.pwr.aPh[0].rlPwr
v_harm_mag_min_user	agUser.min.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_min_user	agUser.min.harm.aPh[0].harm.aCuMagn[0]
v_rms_max_user	agUser.max.rms.aVo[0].rms
c_rms_max_user	agUser.max.rms.aCu[0].rms
rlpwr_max_user	agUser.max.pwr.aPh[0].rlPwr
v_harm_mag_max_user	agUser.max.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_max_user	agUser.max.harm.aPh[0].harm.aCuMagn[0]
v_rms_aggr_auto	agAuto.aggr.rms.aVo[0].rms
c_rms_aggr_auto	agAuto.aggr.rms.aCu[0].rms
rlpwr_aggr_auto	agAuto.aggr.pwr.aPh[0].rlPwr
v_harm_mag_aggr_auto	agAuto.aggr.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_aggr_auto	agAuto.aggr.harm.aPh[0].harm.aCuMagn[0]
v_rms_min_auto	agAuto.min.rms.aVo[0].rms
c_rms_min_auto	agAuto.min.rms.aCu[0].rms
rlpwr_min_auto	agAuto.min.pwr.aPh[0].rlPwr
v_harm_mag_min_auto	agAuto.min.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_min_auto	agAuto.min.harm.aPh[0].harm.aCuMagn[0]
v_rms_max_auto	agAuto.max.rms.aVo[0].rms
c_rms_max_auto	agAuto.max.rms.aCu[0].rms
rlpwr_max_auto	agAuto.max.pwr.aPh[0].rlPwr
v_harm_mag_max_auto	agAuto.max.harm.aPh[0].harm.aVoMagn[0]
c_harm_mag_max_auto	agAuto.max.harm.aPh[0].harm.aCuMagn[0]

Table 3. Measurement results parameters name.

References

[1] DSP Engine Interface, version 0.2
[2] Serial Communication Protocol, version 0.2
[3] Test Procedure, version 0.2