EmonLibDB

This library provides a means of using advanced features of the Microchip AVR128DB48 microprocessor used in the emonTx V4 Energy Monitor.

It provides for up to 3 independent voltage inputs and up to 12 current inputs, allowing measurement of voltage, current and power on a single-phase, split-phase or three-phase electricity supply. The physical quantities are measured continuously, the average values are calculated and made available to the host sketch at user-defined intervals.

This document has the following main sections:

Key Properties – important points about this library

Using EmonLibDB – a very brief explanation of how to use this library

List of required supporting libraries

Initial configuration

Calibration notes

Example Sketches

Alphabetical index of Application Interface functions

Key Properties

- Continuous monitoring of up to three voltage channels and up to 12 current channels.
- Gives an accurate measure of rapidly varying loads.
- Better than 1480 sample sets per second – using 3 + 12 single-phase channels of an emonTx4 @ 50 Hz, which equates to 29 or 24 samples per mains cycle at 50 Hz or 60 Hz respectively.
- Calculates rms voltage, rms current, real & apparent power[1] & power factor[1].
- Accumulates Wh.
- User-defined reporting interval.
- Suitable for operation on a single phase, split phase or three phase supply (not exceeding 255 V line – neutral when used with the emonVs) at 50 or 60 Hz.
- Can be calibrated for any voltage and current up to a maximum of 32 kVA per input.
- Includes functions to enable on-line recalibration.
- Pulse counting on up to 3 inputs.
- D.C. voltage input available. .
- Temperature monitoring is not supported.

[1] Only when monitoring loads connected Line-Neutral, not available for loads connected Line-Line.

Using EmonLibDB

You will need a sketch that globally

Includes the emonLibDB library and any others that might be needed.

In **`setup( )`**

1. Sets up any parameters where the default value is not suitable.
2. Initialises emonLibDB.

and in **`loop( )`**

1. Checks that the library has 'logged' the data, then
2. Extracts, prints or forwards the data to wherever it is needed.

Setting parameters and options.

The way settings are made is significantly different to previous versions of emonLib or emonLibCM. The settings fall into two sections, one section defines the properties of the sensors connected to the physical inputs (the input voltage terminals or the current transformer sockets), while the second section defines which current input to use with which voltage input in order to calculate the power and energy values.

All settings are made with a "setter" function. For example, the default setting for the first voltage input is number 1, the amplitude calibration is 100, and the phase error is 0.16° This would be set as:

```
EmonLibDB_set_vInput(1, 100, 0.16);
```

Using the 3rd current input will need, for a 100 A c.t. an amplitude calibration of 100 (i.e. the c.t's rated current) and a phase error of 1.4°, which would be set as:

```
EmonLibDB_set_cInput(3, 100, 1.4);
```

To link those together and read the power, set:

```
EmonLibDB_set_pChannel(3, 1);
```

A list of all available setter functions, and a description of what they do, follows in the Application Interface sections.

Extracting the data.

The function

```
EmonLibDB_Ready( );
```

must be called very frequently inside **`loop( ).`** If no new data is ready, this returns immediately with the value **`false.`** If however new data is ready, it returns the value **`true`** and you may then use the "getter" functions to retrieve the data. For example, to retrieve the real power measured on channel 3 (CT3 & Voltage 1 in the example above) and assign the value to the floating point variable power3, you would use:

```
power3 = EmonLibDB_getRealPower(3);
```

A list of all available getter functions, and a description of what they do, follows in the Application Interface sections.

Example sketches are available: one shows the minimum sketch needed, and a second illustrates the use of every function available (even though in most cases the default value is set again, by way of illustration).

EmonLibDB Application Interface

Power & Energy

void EmonLibDB_set_vInput(uint8_t _input, double _amplitudeCal),
double _phase)
void EmonLibDB_set_cInput(uint8_t _input, double _amplitudeCal,
double _phase)
Sets the physical channels used for the inputs. The first sets the amplitude calibration constant for the designated voltage input, the second sets the amplitude calibration constant and phase error compensation for the designated current input.

Generally, **ALL** the input channels that will be used must be fully defined. Voltage and current inputs not in use need not be defined.

Voltage amplitude calibration constant: This is a unit-less ratio that depends on how the mains voltage is divided down to a low voltage suitable for the ADC input. For the emonVs with an emonTx4, the value is 100. Default: 100.0

Current amplitude calibration constant: This is the nominal rated current for a 0.333 V current transformer. Default: 100.0 The unit is the Siemens (1 / Ω).

Phase error: This is the phase lead of the transformer itself, in degrees. (It will be negative if the output lags the input.)

The physical channels must be set before the library is initialised with EmonLibDB_Init( ). To adjust the calibration while the library is running, use the functions below.

There is no return value.

void EmonLibDB_reCalibrate_vInput(uint8_t _input, double _amplitudeCal,
double _phase)
void EmonLibDB_reCalibrate_cInput(uint8_t _input, double _amplitudeCal,
double _phase)
The first resets the amplitude calibration constant for the designated voltage input, the second resets the amplitude calibration constant and phase error compensation for the designated current input. These are intended for calibrating the sketch whilst it is running. They must only be called after the library had been initialised with EmonLibDB_Init( ). You should expect a brief interruption or disturbance to the output data when these functions are used.
The parameters are the same as for EmonLibDB_set_vInput(…) and EmonLibDB_set_cInput(…) above, there is no return value.

void EmonLibDB_set_pInput(uint8_t _cInput, uint8_t _vInput1 [, uint8_t _vInput2])
Sets the relationship between voltage and current inputs to enable the calculation of real power. For single-phase, one leg of a split-phase and 3-phase 4-wire measurements, voltage and current sensor must be on the same phase/leg. The second voltage is needed only for the line-line power calculation in split-phase and 3-phase systems; i.e. when the current measured is line-line.

void EmonLibDB_setPulseMinPeriod(uint16_t _period)
void EmonLibDB_setPulseMinPeriod(uint8_t channel, uint16_t _period, [uint8_t _edge=FALLING])
Sets the minimum period of time that the pulse must be active and in a steady state to be recognised, in ms. This must be shorter than the duration of the pulse. *(Note: this differs from the definition in previous releases, and if the period is specified, its value should be reviewed.)* For electronic switches that do not exhibit contact bounce, zero may be used. The channel number (one-based) defines the input to be used, and must be specified if edge is specified. Input channels for the emonTx V4 are: 1 – 'Pulse', 2 – 'Digital', 3 – 'Analog-Input' – alternatively the names 'Pulse', Dig' & 'ADC' may be used. The appropriate solder links must be made on the p.c.b, and the 'Analog-Input' pulse input is not available when the extender p.c.b. (c.t's 7 – 12) is fitted. The edge on which the pulse is detected may be specified if required (RISING or FALLING). Default: period = 20.

void EmonLibDB_setPulseEnable([uint8_t channel,] bool _enable)
Enables pulse counting. Pulse counting is initialised when the library is initialised, and thereafter may be turned on or off as required. The change is applied at the next datalogging. The pulse count is frozen whilst pulse counting is disabled. Input channels for the emonTx V4 are: 1 – 'Pulse', 2 – 'Digital', 3 – 'Analog-Input' – alternatively the names 'Pulse', Dig' & 'ADC' may be used. The appropriate solder links must be made on the p.c.b, and the 'Analog-Input' pulse input is not available when the extender p.c.b. (c.t's 7 – 12) is fitted. Defaults: 'Pulse' input, enable = false, edge=FALLING.

void EmonLibDB_setPulseCount([uint8_t channel,] uint32_t _pulseCount)
Initialises the pulse count. This will normally be used at start-up to restore a value saved prior to a supply interruption or reset; although it *may* be used to synchronise the value with another meter. Input channels for the emonTx V4 are: 1 – 'Pulse', 2 – 'Digital', 3 – 'Analog-Input' – alternatively the names 'Pulse', Dig' & 'ADC' may be used. The appropriate solder links must be made on the p.c.b, and the Analog input is not available as a pulse input when the extender board (c.t's 7 – 12) is fitted. The previous value is overwritten. Default: 0.

void EmonLibDB_setAnalogueEnable(bool _enable)
Enables the (d.c.) analogue input. The analogue input uses the same multiplexer channel as CT12, therefore it must not be enabled when the Expansion Board for CTs 7 – 12 is fitted. The analogue input may only be used when when the Expansion Board is not fitted. The voltage applied to the input connector should be within the range 0 – 1.024 V. Damage is likely if it is allowed to go outside the range -0.3 – +3.6 V. Default: false.

void EmonLibDB_ADCCal(double _Vref)
Sets the ADC reference voltage used by the library as part of the calibration process. If the precise voltage is known, that value can be used. This must appear in the sketch before any physical inputs are defined and calibrated. There is no return value. Default: 1.024

void EmonLibDB_fCal(double _frequencyCal)
Sets the calibration of the internal timebase – principally to enable accurate frequency

reporting. The value is a coefficient and should be expected to be very close to 1.0 Default: 1.0

void EmonLibDB_cyclesPerSecond(uint8_t _cycles_per_second)
Sets the nominal mains frequency, used to calculate phase error compensation. There is no return value. Default: 50.

void EmonLibDB_minStartupCycles(uint8_t _min_startup_cycles)
Sets the number of complete mains cycles to be ignored before recording starts. There is no return value. Default: 10

void EmonLibDB_datalogPeriod(float _datalog_period_in_seconds)
Sets the minimum interval (seconds) over which the power, voltage and current values are averaged and reported. There is no return value. The interval ends at the next voltage zero crossing, so the actual period will generally be longer. Accuracy depends on the processor's ADC clock, and only a 'typical' value is specified, therefore the actual datalog period should be checked and adjusted if necessary. The minimum is 0.5 s, it has not been tested above 5 minutes (300 s). Note that this *may* be called after the library has been initialised with EmonLibDB_Init( ) so that the datalog period can be changed whilst the sketch is running. Note also that emoncms.org will not accept data faster than once every 10 s. Default: 9.8

void EmonLibDB_setWattHour(uint8_t channel, int32_t _wh)
Sets the value of the Watt-hour (energy) counter. There is no return value. 'channel' is the current/power channel (one-based) for the physical CT Input defined by the pcb legend and front panel engraving. This will normally be used at start-up to restore a value saved prior to a supply interruption or reset; although it *may* be used to synchronise the value with another meter. The previous value is overwritten. Default: 0

bool EmonLibDB_acPresent(int8_t channel = 1)
Returns true when the a.c. voltage on input 1 has been detected (greater than approx. 17 V, well below the voltage at which the emonVs can operate). Voltage Input 1 is used for timing purposes and this voltage is required for proper operation of the library.
N.B. Only works on channel 1.

double EmonLibDB_getVrms(uint8_t input)
Returns the decimal value of the rms average voltage in volts over the reporting period, for the physical input (1 – 3, depending on the version of the emonVs used). A value of 0.0 is returned if the input is not in use, irrespective of the voltage present on the input.

bool EmonLibDB_getVinputInUse(uint8_t input)
Returns true if the input has been defined in any setPChannel( ) statement, otherwise false.

double EmonLibDB_getLineFrequency(void)
Returns the decimal value of the average power line frequency over the reporting period. The accuracy is dependent upon the processor's 24 MHz clock, the frequency of which is

specified as “typical”, and the value is liable to jitter when the reporting period is very short (less than 1 s). If an a.c. voltage is not detected, zero is returned.

bool EmonLibDB_getCinputInUse(uint8_t input)
Returns true if the input has been defined in any setPChannel( ) statement, otherwise false.

double EmonLibDB_getIrms(uint8_t input)
Returns the decimal value of the rms average current in amperes over the reporting period for the physical CT Input defined by the pcb legend and front panel engraving. A value of 0.0 is returned if the input is not in use.

int16_t EmonLibDB_getRealPower(uint8_t input)
Returns the nearest integer value of the average real power in watts over the reporting period for the physical CT Input defined by the pcb legend and front panel engraving.

int32_t EmonLibDB_getWattHour(uint8_t input)
Returns the integer value of the accumulated energy in watt-hours since the library was initialised, for the physical CT Input defined by the pcb legend and front panel engraving.

int16_t EmonLibDB_getApparentPower(uint8_t input)
Returns the nearest integer value of the average apparent power in volt-amperes over the reporting period for the physical CT Input defined by the pcb legend and front panel engraving. Apparent power is not available for line-line connected loads.

double EmonLibDB_getPF(uint8_t input)
Returns the decimal value of the average power factor over the reporting period for the physical CT Input defined by the pcb legend and front panel engraving. If an a.c. voltage is not detected, zero is returned. Power factor is not available for line-line connected loads.

double EmonLibDB_getDatalog_period(void)
Returns the decimal value of the interval (seconds) over which the energy, power, voltage, current, etc, values are reported.

uint32_t EmonLibDB_getPulseCount(uint8_t input)
Returns the accumulated count of pulses on that channel since the library was initialised, for the period that pulse counting has been enabled. ‘input’ (one-based, or the names ‘Pulse’, ‘Dig’ & ‘ADC’) is only needed if more than the first (‘Pulse’) interrupt channel is available and being used.

uint16_t EmonLibDB_getAnalogueCount(void)
Returns the mean of the values recorded on each scan over the reporting period, in ADC counts. 1 count = one 4096^{th} part of the ADC reference voltage (nominally 1.024 V) i.e. 0.25 mV. Zero is returned when the analogue input is disabled.

EmonLibDB_Init( )
Initialise the library. This function must be called once only, and then only after all other set-up functions have been called, typically it will be the last line of setup( ). It assigns all

the set-up and calibration constants to the appropriate internal variables and starts the ADC in free-running mode. There is no return value.

bool EmonLibDB_Ready( )
Returns *true* when a new result is available following the end of the reporting period, else returns *false*. Typically, it will be used in loop( ) to control a conditional branch that includes the 'get' functions that extract the required values. It must be called every time that loop( ) executes to ensure correct operation.

REQUIRED LIBRARIES

No additional libraries are required to support emonLibDB.

INITIAL CONFIGURATION

The following settings should be checked and included in your sketch as necessary. The default values – given in the Application Interface sections above – should be suitable in most cases.

Set the correct I/O channels according to your hardware, using an instance of **EmonLibDB_set_vInput** or **EmonLibDB_set_cInput** for each voltage and current input in use. For best performance, you should not include any voltage or current channels that will not be used.

Link current and voltage inputs in pairs to define how to calculate the power and energy, using an instance of **EmonLibDB_set_pInput** for each current input and defining one or two voltage inputs to use as necessary. The same voltage input may be associated with many current inputs.

Set the interval at which you wish the values to be reported, using **EmonLibDB_datalog_period**
For emoncms.org, this may not be less than the default value of 10 s. For emoncms running on a private server, any value is permissible, provided that the remainder of the system is compatible. For a local emonCMS running on an emonPi, a value of 9.85 s gives good results. Short periods will give a large amount of data. The library has not been tested below the minimum value of 0.5 s, nor with values greater than 300 s (5 mins).

If you are **not** in the 50 Hz world, set the mains frequency, using **EmonLibDB_cycles_per_second**.

If you are using the pulse input(s):

Set the debounce period and operating sense using **EmonLibDB_setPulseMinPeriod**, and finally enable pulse counting with **EmonLibDB_setPulseEnable**.

If you are using the ADC input:

Enable the ADC input using **EmonLibDB_setAnalogueEnable.**

CALIBRATION

Before calibrating a sketch that uses this library, read (but do not do) the calibration instructions in Docs > Electricity Monitoring > Current and Voltage > Calibration Procedure. Those instructions contain the general procedure and safety warnings, which you must be familiar with. The detailed instructions that follow apply only to sketches using this library. The default values are given in the description of each function above. Follow *these* instructions for the order in which to make the adjustments and how to apply the values in the sketch, but follow the *general instructions* for how to proceed with the measurements.

Check the ADC reference voltage. If the actual value can be determined with accuracy, this should be set with **EmonLibDB_ADCCal(Vref)** so that the calibration coefficients will be closer to the nominal values. It is not essential to do this, as any discrepancy will be taken up by the individual voltage and current calibration constants.

Set the voltage calibration constant for each voltage input circuit that is in use, using **EmonLibDB_set_vInput(uint8_t input, double _amplitudeCal, double _phase)** The amplitudeCal value should be very close to 100 for any system voltage when using the emonVs. A value of 0.16 for the phase error should be reasonably accurate for the emonVs.

Set the current calibration constant and the phase error compensation for each current input that is in use, using **EmonLibDB_set_cInput(byte _input, double _amplitudeCal, double _phaseCal)** The phase error compensation for the voltage inputs does not need to be used for single-phase use, the phase error of the current input can compensate for both errors. It will only be important when using two voltages together on different sets of current inputs – as when monitoring a 3-phase delta system; or when the emonVs is changed (when it will be easier to calibrate 1 – 3 voltage inputs in preference to up to 12 current inputs; or when you are unable to calibrate each channel individually and you are relying on the nominal published phase error).

If an accurate measurement of line frequency is required and an accurate means of obtaining the line frequency is available, then a correction to the indicated frequency returned by **EmonLibDB_getLineFrequency( )** can be made using **EmonLibDB_fCal(double _frequencyCal).** The correction needed might be in the order of 0.5% (i.e. between 0.995 & 1.005).

EXAMPLE SKETCHES

Three example sketches are provided.

EmonTx4DB_min.ino

This is the absolute minimum sketch that is needed to use the library. As the comment at the beginning of the sketch states, the sketch assumes that all the default values for the emonTx V3.4 are applicable, that no input calibration is required, the mains frequency is 50 Hz and the data logging period interval is 10 s, pulse counting is not required, and that 6 'standard' 100 A CTs and the emonVs power supply/monitor from the OEM Shop are being used as the input sensors.

This should be your starting point for using the library. If you find that you need to adjust any of the default settings, consult the Application Interface section and then copy the appropriate function either from there or from the full sketch `EmonTxV34DB_max.ino`, changing parameters as necessary.

EmonTx4DB_max.ino

This provides an example of every Application Interface function. Many will be redundant in normal circumstances as they simply set again the default parameters, many of which are likely to be correct and will not need changing. If you do need to change a value, the Application Interface section above gives full details.

EmonTx4DB_rf.ino

This provides an example of 12 power and energy values being sent in two radio packets using two NodeIDs. Particular attention should be paid to the number of retries and the timing of and between the calls to **rf.sendWithRetry(…)**

Alphabetical Index of Application Interface Functions

(Note: The function name has been abbreviated for clarity)