VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Luis Alvarez Iglesias

HARMONICS OVER POWER LINES MEASUREMENT

Algorithm in C and testing environment

Information Technology, Embedded systems

2015

ACKNOWLEDGEMENTS

I would like to take this opportunity to show gratitude to my thesis supervisor Jukka Matila, Senior Lecturer in Vaasan Ammattikorkeakoulu, University of Applied Sciences, for his continuous encouragement to learn and support in any stage of this and any other project.

Also, I would like to present my most sincere gratitude to all VAMK's teachers for the valuable guidance in our learnings. Thank you for these years, certainly the most valuable and enrichment experience so far.

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Luis Alvarez Iglesias
Title	Measure Harmonics over power lines, algorithm in C, and testing environment
Year	2015
Language	English
Pages	105
Name of Supervisor	Jukka Matila

This paper is intended to give an overview of measuring harmonic phenomena over the neutral line in 3-phase 4-wire substation network when an earth fault is originated. It requires electrical and microcontroller interfacing background, but little knowledge of harmonics.

The company VASPEC Oy sponsors the research and belongs to a project owned by the same company to develop a reliable earth fault detection and location device. Is based on Seppo Pettissalo's original idea, and the result of the collaboration of the named person and company together with VAMK and Technobotnia laboratories during the last quarter of the year 2014.

The paper offers a solution to the need of a software algorithm in C language to measure transients and rms currents over the 3 phases, and fundamental, 3rd, 5th and 7th harmonic currents over the neutral line in 3-Phase 4-wire configuration. Store and transmit the information altogether with the event's timestamp given by a real-time clock. It includes hardware, software and a testing environment solution.

An approach to the use of a Renesas RX63N microcontroller, interfacing an ADC and an ADE7880 energy meter will be discussed in this document and alternative methods are discussed. It will gather the required information providing the values in less than 300 microseconds, process, transmit and serve the lectures to be displayed in a remote terminal in 0.6ms, repeating the process during a time period of

100ms is proved. The testing environment is provided by means of a desktop computer's soundcard and the use of MATLAB® to generate the fundamental and harmonic waves.

Keywords: fundamental, harmonics, energy meter, power lines, neutral line, driver.

CONTENTS

LIST OF FIGURES AND TABLES

LIST OF APPENDICES

LIST OF ABBREVIATIONS, ACRONYMS, SYMBOLS

ADC	Analog to Digital Converter
AFE	Analog Front-End
CAN	Controller Area Network
CLCK	Clock, refers to signal or pin
CMOS	Complementary metal–oxide–semiconductor
CMT	Compare Match Timer
CS	Channel select
CT	Current Transformer
dB	decibels
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DMIPS	Dhrystone Millions of Instructions per Second
DSP	Digital Signal Processor
ESD	Electrostatic discharge
FFT	Fast Fourier Transform
GND	Ground
HAL	Hardware Abstraction Layer
IAN	Current Phase A (where A can be A, B, C or Neutral) Negative
IAP	Current Phase A Positive

IDE	Integrated Development Environment
MCU or **µC**	Microcontroller unit
MISO	Master In Slave Out
MOSI	Master Out Slave In
MSB	Most Significant Bit
MSPS	Mega Samples per Second
NMOS	Negative-channel Metal-Oxide Semiconductor
PCB	Printed Circuit Board
RMS	Root Mean Square
RTC	Real Time Clock
RTU	Remote Terminal Unit
SMD	Surface Mounted Device
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter

1 INTRODUCTION

Measuring ground fault currents is part of the protection devices of the power lines in a substation network. The current magnitudes of these faults depend on the impedance of the fault and the grounding solidity and resistance. There are effective protection devices for low impedance faults, which produce a high fault current and require the isolation of the line to avoid any further damage. On the other hand, high impedance faults on multigrounded systems, still represent a challenge for protective devices. Its lower current may allow the line to continue to operate and the unbalanced currents may be tolerated by the asymmetry of the power lines. In any of the cases, a quick evaluation of the possible fault is necessary.

Measuring faulty current harmonics can extend the available information and exhibit detailed information about the power lines state. Harmonics are certainly a common measure to determine the quality of the power in the grid. They offer valuable information to evaluate a fault or the origin of an asymmetry. For this reason, a reliable harmonic measurement is needed in any protective device. Moreover, the faster the better. In case of a fault, should the line be isolated, there is a small time gap between the fault detection and the reaction, and it is vital to retrieve as much information as possible to determine the origin. And further, the more harmonic indexes under scope, the better.

A property of a 3-Phase 4-Wire network is that every disturbance in any single phase will appear reflected in the neutral line. This may be exploded to simplify the complexity of any sensing device by sensing the current and harmonics over the neutral line. Additionally, any ground fault, low or high impedance, generates transients. These transients, are spikes produced by the capacitance charge-discharge produced whenever a line contacts a grounded object, favouring the discharge of the line capacitance, and charging whenever the contact is removed. They can have a very high frequency and they have to be measured at a high sample rate (Imrs, 2006) and they will be used as the trigger event of the evaluation measurements.

The research establishes an accurate method to retrieve, in the case where transients appear followed by asymmetric faults, the mentioned information:

- fundamental *rms* current values of the 3 phases
- neutral line transients' peak instantaneous current value
- Total current, 3^{rd}, 5^{th} and 7^{th} harmonic currents over neutral line.

This information is fetched and made ready to evaluate in a time period lesser than 300µs.

The model presented here includes an energy meter as DSP to interface the power lines, although other approaches are reviewed. This DSP is controlled by a higher level MCU using SPI communications protocol. Other MCU functionalities implemented are to control a 16bit ADC that measures transients over the neutral line, and to transmit the recorded data, altogether with a time stamp given by a real time clock, to a remote terminal, the information is served to the remote terminal unit in 0.5ms.

2 LITERATURE REVIEW

The application of electronics to substitute the old electro-mechanic devices to measure power over the grid has brought accuracy and simplified grid quality control and management. It is not a long time ago when the MCUs became an important component in substation automation at the end of last century, and nowadays it is difficult to imagine a design without implementing them. Smart meters demand has grown rapidly and with it the development of energy meters.

Energy meters provide information about power and power quality and all the required information about the energy of the line under scope. More advanced meters provide more than one phase sensing and harmonic calculations.

Harmonics measurement over power lines in substation networks help not only to keep track of healthy lines, but provide useful information in case of failures that can help to determine their origin and location. As harmonics are spread all over the network in all directions, they can be read in the nearest substation or by a device for this purpose in the proximity.

Due to the high cost of a switch gear, whose life is reduced every time it trips, and the possible economical repercussion to the affected area, the false trips have to be avoided. Although the timeframe to trigger a counter action has to be granted as short as microseconds in the case of a low impedance fault, the gathered specific amount information is a valuable tool to provide the correct reaction. A high impedance fault allows more relaxed time of reaction and in many cases can be tolerated by the network, but damages may occur if the normality is not restored.

An important part of this information is given by the harmonics over the power lines, it is essential not only to measure the quality of the power, harmonics can be analysed for a wide range of reasons, like to avoid damages to power systems due to the overheating produced by the rise of the apparent power or to help to determine the origin of a fault. This latest is the area that drove to the case of study in this paper. Although a high impedance is initially assumed, and a timeframe of 150ms

is given, it is worth to consider lower impedance faults with a reduced reaction time interval.

2.1 Three-Phase four-Wire system

A system where the three phases have their independent lines and an additional neutral or returning line is known as three-phase four-wire system. It is a common method in distributed electric power networks. Is characterized for having three electrical conductors carrying symmetric alternate current where the phases are shifted one third of the period, or 120º, and an additional neutral line as a reference line where the sum of the three voltage vectors, in an ideal case of a balanced system, is zero.

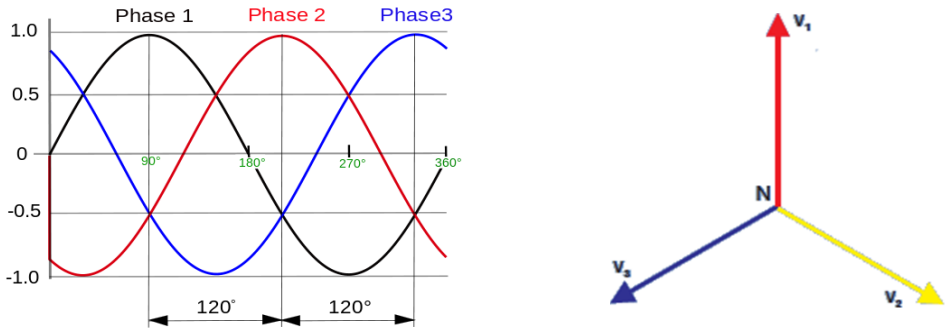


Figure 1 Three phase normalized waveforms and voltage vectors

This system have properties that favours its implementation. One of them is that any imbalanced load in any phase will reflect the unbalanced current over the neutral line. As a result, better voltage regulation is achieved and the system may continue working even in case of fault condition. In this manner, any fault can be exposed by sensing the neutral line.

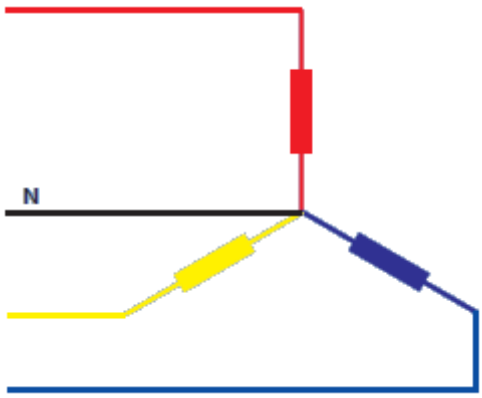


Figure 2 Three-phase four wire configuration

In the case of a low impedance fault, such as a shortcut, the grid requires to be protected in the shortest possible time due to the high current drawn and the damage that may occur. Actual protective devices may react in microseconds to secure the grid. When a high impedance fault occurs, the drawn current levels are lower and they may be in acceptable levels avoiding a shutdown of the faulty network. This behaviour difficult the diagnosis of the problem and the cause of the fault cannot be easily determined and as long as this exists, other signals may appear as a side effect that can reduce the efficiency of the transported power.

2.2 Interfacing power lines

From the power line to the RTU different components are involved, but all has to begin from a connection of the power line to the energy meter front-end. The methods are differentiated as direct (physically connected to the line), or indirect (no physical connection), and regarding the technology, as resistive (direct), transistor (direct) and Magnetic (indirect).

2.2.1 Indirect current measurement methods

Indirect current measurement means that there is no physical connection with the measured voltage line, the sensor is isolated from the line and it is an accurate method when the current is too high to be measured with a directly connected instrument. There are three main technologies where the measurement device provides isolation from the line, current transformers (CT), Rogowski Coil and Hall

Effect devices. All of them provide a voltage level signal at the output that is proportional to the current flowing through the line under scope

CTs are useful when measuring AC, transients or switching mode DC, since it

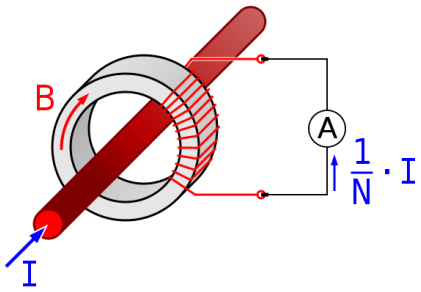


Figure 3 CT

senses the changing magnetic field produced by the AC oscillation. The CTs use the power line as the primary of a transformer, with 1 to a few turns that, its flowing current (I in Figure 1), induces an alternating magnetic field in the core (B in Figure 1), producing an alternating current in the secondary. As the induced current is the result of the relationship between the primary's number of turns (single turn), and secondary's turns (N), the output current (A in Figure 1) is calculated

$$I_{out} = \frac{1}{N} \cdot I$$

Equation (1)

where

I_{out} is the output current at both ends of the secondary in a closed circuit

N is the number of turns of the secondary

I is the current flowing through the power line

Placing a low value burden resistor in parallel with the load and closing the secondary winding circuit will convert the given current to a voltage signal that can be calculated, knowing the desired output voltage, by ohms law.

$$V_{out} = I_{out} \cdot R_{burden}$$

Equation (2)

where

V_{out} is the voltage drop over the burden resistor

I_{out} is the current flowing through the secondary winding

R_{burden} is the value in ohms of the resistor closing the secondary circuit

A voltage level is easier to read for any instrumentation device or the input of an ADC. Notice that if this burden resistor is not in place and the secondary winding terminals are left open circuit while there is a current flowing over the primary, the secondary will store the energy creating a high voltage and a dangerous situation.

Care is taken by the manufacturers' design to efficiently couple primary and secondary circuits and to avoid core saturation by choosing the wrong burden resistor. In this manner, CTs can provide in theory, a lossless current measurement, and the signal voltage in a power line is large providing a noise immunity measurement (Yarborough, 2012). The output has relatively low phase shift, from tenths of degrees to a few degrees in lower quality CTs allowing a direct connection with the measuring device. Although the phase shift does not affect the measurement when these are magnitudes. Designers have to ensure that the CT dynamic range is large enough according to the requirements.

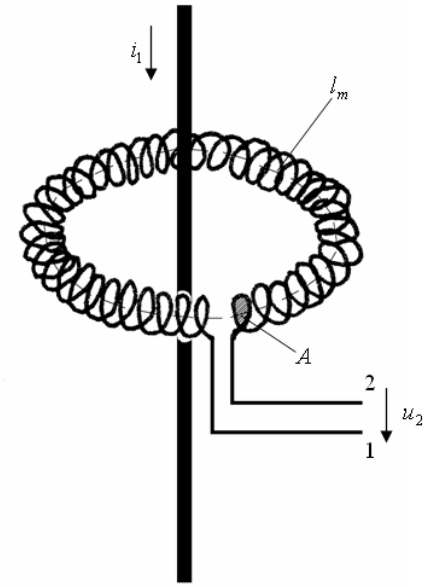


Figure 4 Rogowski Coil schematics

The Rogowski Coil shares the CT principles, an isolated line whose current flow induces a proportional current in a secondary coil. The main difference resides in the core, Rogowski Coil's core is air, with a lower inductance, faster signal response and very linear output.

Another useful property is that, theoretically, no matter the distance or location the Rogowski Coil is placed relative to the conductor line whenever the signal line passes through the toroid (Mäkinen, 2014) and it does not saturate. However, the output signal of a Rogowski coil is proportional to the time derivative of the current therefore requires an Integrator or using the non-integrated signal and process it to adjust magnitudes and phase shift the signal by 90º. This due to the properties of the Rogowski coil, where the induced voltage in the coil is proportional to the current rate of change, and integration is required to obtain a voltage level proportional to the current waveform.

$$v_{coil} = -\frac{\mu_0 AN}{l}\frac{di(t)}{dt}$$

Equation (3)

where

A is the turn Area
N is the number of turns
l is the length of the winding
μ_0 is the air permeability constant
$\frac{di(t)}{dt}$ is the rate of change of the current through the loop

Hall Effect devices, the last listed method, is not analysed in this document as it is not involved in this research.

2.2.2 Direct voltage measurement method

When directly measuring voltage levels, it is necessary to construct an attenuation network of resistors in a voltage divider implementation, to accommodate the voltage level to the required input and limit the current flow. Although for security

reasons it is recommended to split the value of the inline resistor in a series of resistors and calculate the related power dissipation.

2.2.3 Evaluating voltage level signals

In an ADC one or more input voltage levels who should be in between two input reference voltage values, are translated in digital information. This voltage level is compared in steps given by its resolution in bits, its value digitized in a binary format and output as a discrete value of an instantaneous input. A use may be applied to record individual instantaneous values which give a description of the element under scope. Nyquist-Shannon's sampling theorem (Smith, 1999) stablishes that all the sampling process maybe repeated at a sample rate at least twice faster than the maximum frequency of the signal under scope, to obtain data samples enough, forming a discrete signal with all the information of the original continuous signal, allowing any wave to be reconstructed.

Relevant characteristics have to be satisfied when selecting an ADC for energy metering purposes:

- The analog input bandwidth, will define the frequency limit above which the signal is attenuated. When measuring harmonics of higher orders, they should not be cut by the ADC limitations. A 50th order harmonic in 50Hz will require at least a 2.5 kHz bandwidth. (Moulin, 2003)
- The sampling frequency, should be at least twice the desired bandwidth or the signal will suffer aliasing, an effect for which higher frequencies cannot be correctly read and get a wrong digital equivalent sample. (Moulin, 2003)
- The LSB precision, accuracy and the noise floor. Known the voltage range under scope and the ADC resolution, we have the precision which is

$$V_{LSB} = \frac{V_r}{2^b - 1}$$

Equation (4)

where

V_r input voltage range

b bit resolution

This gives the value of the ideal resolution of the ADC in use. A different value is given by the Dynamic Range, or SNR, of the signal and values are better read in dB. In this manner, with the quantization error ideally uniformly distributed between $-\frac{1}{2}$ and $\frac{1}{2}$ of the LSB in all quantization levels, translated to dB, the Signal-to-quantization-noise ratio (SQNR) is

$$SQNR = 20log_{10}(2^b) \approx 6.02 \cdot b$$

Equation (5)

where

$SQNR$ Signal-to-quantization-noise ratio

b bit resolution

Which is the same as to say that each bit of resolution contributes with approximately 6dB to the Dynamic range.

An ideal meter with a dynamic range of 2000:1 with a precision of 0.1 of the units in use and a specified maximum error of 0.1% requires a minimum dynamic range of 146dB, or an analog ADC with at least 25 bits of resolution:

$$SNR \leq 20\log\left(\frac{2000}{0.1 \cdot 0.001}\right) = 146dB$$

Equation (6)

$$\frac{146dB}{6\,{}^{dB}/_{bit}} \approx 25bits$$

The noise floor of the system becomes relevant to satisfy the specifications. To have good accuracy, the noise floor of the selected system should lay over the bit resolution or the system won't satisfy the accuracy conditions. This is especially relevant when evaluating signals arriving from a CT or Rogowski Coil since their inducted voltage levels can be as low as microamperes, falling into the noise floor

level. For example a system like an energy meter DSP whose maximum ADC input voltage is $0.5V_p$ and the noise floor is $1\mu V_p$, then the dynamic range is 500075.52:1, or 114dB, requiring at least 19 bit ADC:

$$20\log\left(\frac{0.5}{1^{-6}}\right) = 114dB$$

$$\frac{114dB}{6\,{}^{dB}/_{bit}} \approx 19bits$$

A different ADC technique is the sigma-delta ADCs (1-bit ADC) where the final result comes from the successive bit approximation of the sampled value. These ADCs have the attribute of oversampling, sampling multiple times faster than a traditional ADC. This favours the sample quality because while the SNR is the same as before, its energy is spread over a larger frequency range. As a result of this, the RMS noise is less after filtering the signal (Maxim Integrated, 2003)

All the input signals should be filtered to avoid aliasing. An aliased signal is present in all sampled systems regardless the ADC architecture. It means that any sampled signal higher than the half of the sampling frequency will get a wrong sampled digital value in the frequency below half the sampling rate.

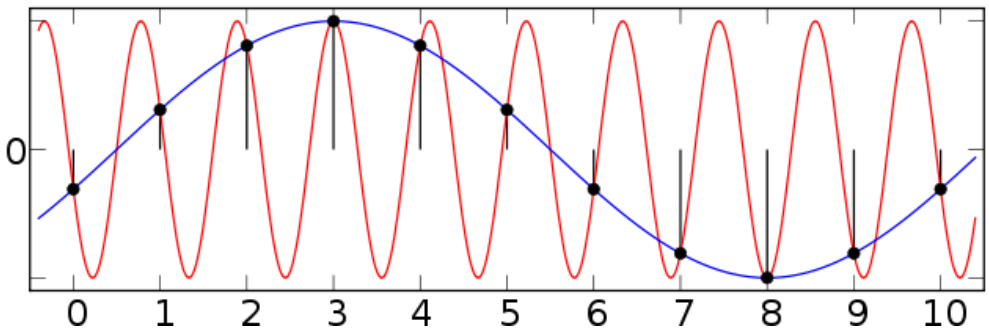


Figure 5 Aliasing: Two different waves having the same sampled values

When using a passive first order RC filter, one has to consider that their attenuation of 20dB/dec must be sufficiently high at the half of the sampling frequency. However, when using a Rogowski Coil, these sensors have a 20dB/dec gain that voids the 20dB/dec attenuation of the first order filter, thus the attenuation must

be offset again. Designing the LPF as a cascade set of filters or a second order filter will establish the attenuation again. On the other hand, it is known that Rogowski Coil output requires an integration of its output, yet an integration produces a result, in frequency domain, of 20dB/dec and a -90° phase shift that have to be accounted. Analog integrators design are easy to implement by means of an OpAmp, but its design requires special care and the environmental conditions have an important role in the operating lifetime.

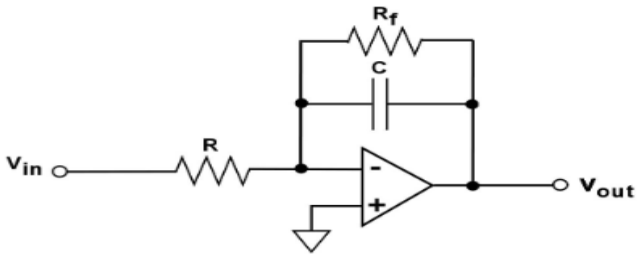


Figure 6 Analog integrator using OpAmp

Implementing a digital integrator is possible by processing the sampled output of the Rogowski coil. Having these models a closer output to the ideal, avoiding the need of extra analog circuitry thus more stable over the time. In this manner, allows its use as a CT but remembering to implement a second order filter at the input (William Koon, Analog Devices, Inc, 2001).

2.2.4 Analog Front-End (AFE)

The selection of the required front-end hardware presents different options and special considerations have to be taken when measuring harmonics. This will affect to the performance, accuracy, reliability, complexity and cost.

Whenever a decision has to be taken regarding the price, the MCU with built-in ADCs is the solution. It has the lowest prices and easy implementation due to a little number of extra components needed to implement reducing cost, complexity and the time to market. The selection of the MCU becomes important as it is required to have a good DSP processing capability and ADC high sampling rate. Whenever

using MCU DSP the software complexity grows as it is needed to have a good understanding of digital filtering and the manufacturer does not provide a DSP library or this is not free. Manufactures should provide with the information about it as well as detailed information of the DSP processing performance. As a result, the flexibility of the system is reduced as it may require an entire redesign to implement an upgraded MCU.

In case of aiming performance, the components can be selected individually for each function. ADC and DSP are implemented as individual entities, giving liberty to select the components given a required accuracy. Although a high quality and flexible device can be developed, this kind of implementation is expensive and more complicated.

A third option is a solid-state Energy Meter interfaced with an MCU. Since the apparition of electronic energy meters, they evolved from a single phase voltage meter to a polyphase multifunction energy metering with DSP and harmonic monitoring. These meters can be reduced to two types, analog front-end (AFE) ICs and System on Chip (SOC) meters. The first kind provide the front-end to the power line, allowing an external MCU to control them. The second includes a microcontroller. Each have benefits, as cost is in the case of a SOC and flexibility in AFE ICs and cons, like upgradeability, the SOC cannot be modified easily (Mani, 2013). And when referring to different manufacturers, they share similar architectures. A set of inputs to interface with voltage and current sensors, ADC to convert the signal value, a DSP to process the information and a MCU to manage the process and peripherals. Discarding the SOC, a single energy meter can be interfaced by a low cost MCU to handle the communications task.

2.3 Microcontroller Unit (MCU)

The MCU should be chosen after the AFE selection, if not embedded into it. Since it is not the same CPU load to receive the processed values from an Energy Meter and transmit it or to have to process harmonic calculations in a determined time frame. The first may be accomplished by a low cost 8-bit MCU and the second may require an MCU with DSP and floating point capabilities. Other aspects to consider,

regarding the performance, are like the use of an Operating System, if needed, with real time capabilities, fault handling and diagnosis code or communications CRC validation. Other constraints are cost, size or environment.

An implementation of an MCU interfacing an AFE requires peripherals and digital I/O of which is relevant to know its type, CMOS, NMOS or Open Drain Control, Pull-up Resistor control. Peripherals in a MCU include ADC that whenever planning to use these, should meet the requirements as mentioned. It is highly recommended that these peripherals include the required communications controllers that will be further used like UART, SPI, CAN, Ethernet, Parallel Data Capture unit and DMA that will release the CPU load. Other parameters are, regarding to the CPU architecture, the available memory space, word length, clock generation circuit, Interrupts Control unit, internal busses and pipelines. Whenever there are time constrains these parameters become crucial to control the application flow and perform the task in the given time frame. Whenever the MCU AFE are ADCs and having time constraints, it is important to count with DSP instructions that will reduce the processing time.

2.4 Evaluating harmonics

A harmonic component of a fundamental frequency is another frequency that is an integer multiple of this latest. Accordingly to this, any European power line transporting electricity at 50Hz of fundamental frequency, will present harmonics at 100Hz, 150Hz, 200Hz and so on. As a consequence of this, the harmonics may be identified by their index, that is the integer multiple of the fundamental, ergo the 3rd harmonic index of a fundamental of 50Hz refers to the 150Hz component. They are clearly differentiated from transients or spikes as a wave *x* times shorter than the reference wave.

Harmonic analysis of the currents in power lines is the best method to measure the quality of the transported energy. They give a description of the distortion of the fundamental frequency, and in an ideal environment they are not present. Hence in a real environment, harmonics indicate the real state of a transmission line. For this reason we can understand harmonics as a continuous source of valuable data about

the instantaneous real state of a power line. Harmonic's effects on the power line are traduced as an increased RMS current needed to source any load and therefore producing losses dissipated as heat. Another figure, related to the harmonics is the Harmonic Distortion and the Total Harmonic Distortion. The first gives the relative deviation of the signal respect the fundamental, the second is the percentage of the harmonics regarding the fundamental, or how much of the current or voltage belongs to the harmonics in a power line.

From the mathematical point of view, a harmonic answers to the question of how much component of an index *x* there is over a fundamental frequency *y*, and translating it to power lines, it gives the amount of energy that is carried by each frequency. This can be achieved by decomposing the original waveform into all of its frequency components by means of the Fourier series.

2.4.1 Fourier series background

Fourier series shows how any periodic function can be plotted by the sum of sinusoid functions. Whenever having a periodic function such that,

$$f(t+T) = f(t)$$

Equation (7)

For all t, maybe written as a Fourier series

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}[a_n \cos(n\omega t) + b_n \sin(n\omega t)]$$

Equation (8)

where

$$\omega = \frac{2\pi}{T} = 2\pi f$$

and the numbers $a_0, a_1, \dots, a_n\ b_1, b_2, \dots, b_n$ are known as the coefficients of the series. Having different coefficients for different functions $f(t)$.

The Fourier series coefficients can be calculated as follows,

$$a_n = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\cos(n\omega t)\,dt \qquad n = 0,1,2 \ldots$$

Equation (9)

and

$$b_n = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\sin(n\omega t)\,dt \qquad n = 0,1,2 \ldots$$

Equation (10)

If $f(t)$ is even, meaning $f(-t) = f(t)$,

$$a_n = \frac{4}{T}\int_{0}^{\frac{T}{2}} f(t)\cos(n\omega t)\,dt \qquad n = 0,1,2 \ldots$$
$$b_n = 0$$

and If $f(t)$ is odd, meaning $f(-t) = -f(t)$,

$$a_n = 0$$
$$b_n = \frac{4}{T}\int_{0}^{\frac{T}{2}} f(t)\sin(n\omega t)\,dt \qquad n = 0,1,2 \ldots$$

(Mäkelä, 2013)

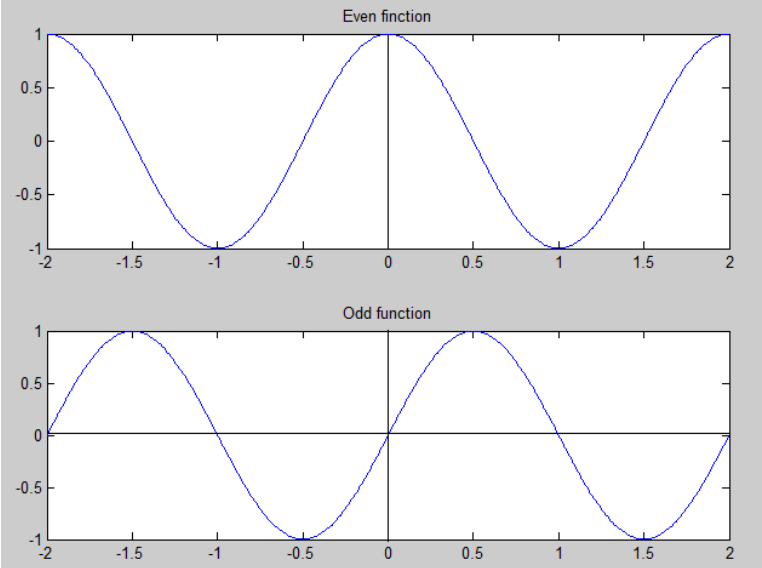


Figure 7 Even vs. Odd function. Matlab plot

in other words, as the electric current is an odd function, the Fourier series reduces to

$$f(t) = \sum_{n=1,3\ldots}^{\infty} \left[b_n \sin\left(\frac{n\pi t}{T}\right) \right]$$

Equation (11)

And coefficient b_n, where

$f(t)$ is the time domain function

n is the harmonic number (only odd values of n are required)

b_n is the coefficient or, regarding harmonics, amplitude of the nth harmonic component

T is the time period the length of one cycle in seconds

Whereas this introduction to the Fourier Transform as a tool to decompose a function in the sum of sinusoids, an extension of its idea applied to non-periodic functions is the Fourier Transform.

$$\mathcal{F}\{g(t)\} = G(f) = \int_{-\infty}^{\infty} g(t) e^{-i2\pi f t} dt$$

Equation (12)

As a result of the Fourier Transform, $G(f)$ gives the magnitude of $g(t)$ at a frequency f. For the most known applications there are free sets of look up Transform pair tables that one may use to get $G(f)$ from $g(t)$, to reduce the mathematical calculation time. To evaluate the Fourier transform, the Discrete Fourier Transform is widely used as it can be implemented in computer, MCU by numeric algorithms, or by dedicated hardware, by analysis of a finite amount of data, samples taken within the same period. It differs from Discrete Time Fourier Transform, also called continuous, in that is has a finite input, with N samples, and output, resulting in much easier and faster calculations.

$$\mathcal{F}\{x(n)\} = X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-\frac{i2\pi nk}{N}}$$

$$for\ k = 0, 1, \ldots, N-1$$

Equation (13)

where

$x(k)$ a complex number series of N samples such that $x_0, x_1, x_2 \ldots x_k \ldots x_{N-1}$ and $x_i = x_{real} + ix_{imag}$

N number of samples k ranging from 0 to $N-1$, repeated periodically such that $x(k) = x(k+N)$

To reduce the complexity of the equation above, one may refer to *Euler's* identity for complex numbers analysis, stating that for any real number x

$$e^{ix} = \cos(x) + i \cdot \sin(x)$$

Equation (14)

Using known fast Fourier transform (FFT) algorithms, result in shorter time and less processing power. Cooley and Turkey introduced this algorithm in 1965, limiting the input to a power of two size and generating two N/2 sequences to speed

up the process, the DFT of the even indexed part of $x(k)$ and the odd indexed part. The resultant lecture of this is such that $x(k)$ gives the value of the Nth frequency index.

Referring this to harmonic measurements, less calculations are required, since only the magnitude of a few frequencies of index N with value $x(k)$ are required. As $x(k)$ is a complex number, it has a real and an imaginary part

$$Z = x + iy$$
$$Z = |Z| \cdot e^{i\theta}$$
$$x = Re\left[|Z| \cdot e^{i\theta}\right] = |Z| \cdot cos\theta$$
$$y = Im\left[|Z| \cdot e^{i\theta}\right] = |Z| \cdot sin\theta$$

Equation (15)

where
Z is a complex number
x is the (Re) real part
y the (Im) imaginary part
$i = \sqrt{-1};\ i^2 = -1$
$|Z|$ or Z modulus or the magnitude
θ the phase angle

And, as in most of the cases when the harmonics are measured, not always the phase angle is needed, only the magnitude, having reduced calculations involved. As with the FFT, there are available algorithms to implement in this case. A remark to the Goertzel algorithm that can perform frequency detection using less computational power than the FFT (Banks, 2002)

2.5 Evaluating transients

Transients, in power lines, are transitory, non-lasting bursts or oscillations of energy as a response to a change from a previous state. Generally called spikes, they can be voltage, current or energy and they disappear if the power is disconnected for a short period of time. There are many situations that cause overvoltage transients,

lightning strikes, shortcut, trip of a circuit breaker, a tree occasionally touching the line or an animal contact among other. Earth faults, as the shortcuts, are sources of transients in their initial instants, from microseconds to a few milliseconds. The main characteristic is the overvoltage spike with a very high bandwidth and a very short time span.

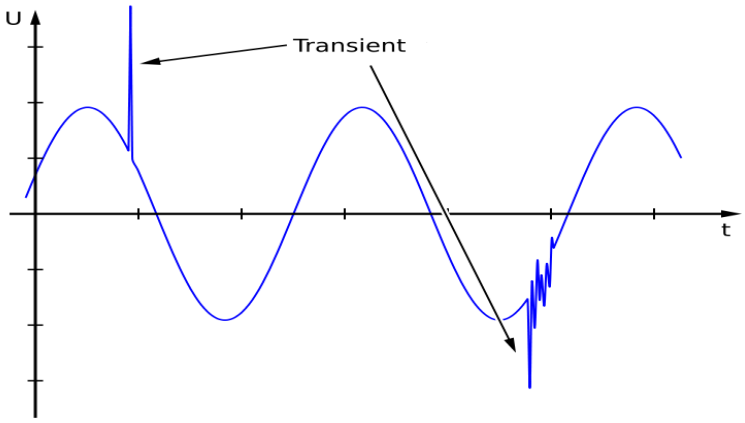


Figure 8 Plot of a transient

These overcurrent or overvoltage signals carry information that can be analysed by measuring devices. They require a very high bandwidth and amplitude tolerance, and in general, different circuitry and signal processing properties than the harmonics. Due to the random nature, the variability of the sources and the difficulties to define their amplitude, duration and energy content, they are not analysed in this paper, and only their amplitude is measured altogether with their time span.

3 SELECTED RESOURCES

Here is stated a method to solve the problem proposed by VASPEC Oy when trying to find an efficient low cost digital method to measure transient currents and harmonics in power lines on earth-fault conditions. The research had to include an MCU to perform the calculations and the communications, written in C language, interface the signals by ADCs and provide a low current testing environment, meaning to provide the signals at the ADC input voltage level.

The solution presented in this research was built by means of an ADE7880 solid-state Energy Meter, an Analog Devices Inc. product, interfaced with a Renesas YRDKRX63N board with a RX63N MCU by means of SPI communication at 2MSPS. The MCU transmits the data over UART to an RTU and the faulty conditions are simulated by means of a PC running MATLAB® and a soundcard used to output a wave signal composed of a fundamental frequency of 50Hz carrying a 3^{rd}, 5^{th} and 7^{th} harmonics generated by the SIMULINK® DSP and transmitted through a wire connected to the soundcard's jack input to the ADE7880. The software solution is Eclipse-based Renesas e2studio Integrated Development Environment (IDE) and the project's code written in C. The solution works as expected meeting all the requirements specified by VASPEC Oy.

3.1 Hardware and Software resources

The resources required for all the stages of the research are identified firstly, as related to the AFE and MCU, secondary as related to prototyping and PCB design and thirdly as related to the testing environment

First, the selected AFE to prototype in breadboard and later to implement in a PCB and a Renesas YRDKRX63N MCU board:

- ADE7880
- Renesas YRDKRX63N MCU board
- -CT Yhdc SCT-013-030 30A input 1V output

The selected MCU resources have a code implementation:

- MCU, start-up code
- SPI interface, 2 independent interfaces
- UART interface, 1 interface
- RTC
- CMT, 2 units
- ADC, 12 bit
- Digital I/O

The selected external resources have a code implementation:

- ADE7880 Energy Meter
- Switches
- LEDs
- Okaya display

Other tools related to the MCU used to code and debug were:

- Eclipse-based Renesas e2studio IDE
- Segger J-Link hardware debugger

Second, the testing environment, which is based in MATLAB® to generate the 50Hz fundamental with harmonic waves with a final implementation using SIMULINK®. A multichannel sound card, Realtek ACL650 (Appendix 7: Soundcard Datasheet appendix 7.5) is used to output the signals for at least as many AFE inputs as used, three phases and neutral line currents and one phase voltage.

Third, prototyping and PCB design. An initial design over breadboard is prototyped, all the components placed include:

- Input current passive low pass RC filters with a f_c of 14KHz and a maximum phase shift of -1.4 degrees in the frequency range of the 7^{th} harmonic, 350Hz
- Voltage line voltage divider circuit with a parasitic filter.
- Required ADE7880 circuitry as specified in Datasheet.

- 3.3 V linear regulated power supply, based in AMS1117-3.3 to power up the breadboard
- 3.3 V dc-dc isolated regulated power supply ISF 0503A to power up the AFE in the PCB
- Renesas YRDKRX63N board PMOD1 port interface
- CT will be used for testing purposes, Rogowski Coil will be used in a real environment, requiring an Integrator.

The PCB prototype was completely designed with PADS in a dual layer board of 75mm x 62mm and completely manually routed for efficiency, containing as mayor clusters:

- A socket to insert an ADE7880 previously soldered to a 2 x 20 inline pins board body.
- All the required input filters
- 3 high speed quad line digital Isolators, one ISO7240 with 4 out lines, and two ISO7241 with 1 input and 3 output lines, to isolate the ADE7880 side of the board from the MCU side.
- A 40 pin socket connector to interface with the Renesas PMOD connector and Raspberry Pi compatible
- Independent and isolated 5V to fixed 3.3V DC/DC converter ISF0503A to power the ADE7880

3.2 Selecting the AFE

The option of an MCU fetching the data with ADC and processing it to obtain fundamental and harmonic values is a valid option and reduced cost. Whenever the conditions of a high impedance fault, and consequently allowing longer time gaps for processing, are the target of the implementation, this can be the primary option. On the other hand, this choice requires a deeper level of understanding of harmonics, FFT and a DSP library for the selected MCU if available, and deserves to mention a word about the MCU ADC's quality, whenever a low floor noise and higher precision is required, the selectable number of MCU decreases and the price raises.

The ADE7880 energy meter was proposed and accepted before starting the development of the project by all the parts and its analysis in this paper is articled *post factum*. It has proven satisfactorily the energy and harmonic meter capabilities fulfilling all the demands, response time, precision and accuracy. Comprehensive understanding of the harmonics is required but no mathematical implementation is required as this AFE outputs the processed value.

The ADE7880 retail price of one unit in the market is 11.84€ (Farnell Oy, May 2015). The cost may represent the most important drawback of this IC and a further research to compare with the performance of the select MCU DSP instructions, regarding to the required selected input currents in the given timeframe, with the multiple outputs of the energy meter at a 125kSPS.

The ADE7880 requires isolated environment, since the neutral voltage input GND is internally connected to the IC GND. This means that the ADE7880 GND level is the same as the measured neutral power line level, and a serious risk since they represent a fatal health hazard. The result of this is the implementation of the ISF0503A, a single output dc to dc voltage regulator with isolation up to 1000Vdc that requires an input voltage from 4.5 to 5.5Vdc and outputs a 3.3V up to 0.3A, enough for the ADE7880, and having a cost rounding 6€ (Farnell Finland, May 2015). This will allow to externally supply the power from any popular 5V 0.5A converter. Moreover, all the external connections to the ADE7880 have to be isolated, being those the SPI communication paths and other signals with a high bandwidth. The addition of Silicon Dioxide Isolators with a high signal rate of 25MHz provides the required high voltage block and GND isolation barrier, preventing noisy currents to enter the other side of the circuitry.

3.2.1 Features of interest

The ADE7880 is compatible with 3-phase for 4-wire (Delta or Wye). This is especially relevant for the experimentation since the neutral line is the carrier of most of the information, but the three phases are sensed simultaneously as well. Having

an independent computational block for harmonic information on neutral current and phase data path, the output registers' content can be fetched simultaneously.

Supplies RMS, active, reactive, and apparent powers, power factor, THD, and harmonic distortion of all harmonics within 2.8 kHz pass band (up to the 63 harmonic) on phase or neutral current and voltage, which is beyond the 7^{th} harmonic as the highest index of interest. Although the harmonic calculations are limited to one phase or neutral at a time due DSP limitation, this is not a drawback since only neutral current harmonics are of the interest of this analysis. Supplies RMS and harmonic distortions of all harmonics within 2.8 kHz pass band on neutral current with less than 1% error in harmonic current and voltage RMS, harmonic active and reactive powers over a dynamic range of 2000 to 1 at $T_A = 25°C$

Regarding to the ADC, equips 7 Sigma-delta (Σ-Δ) 24bit ADCs with a sampling rate of 1.024MHz. The ADC outputs are signed twos complement 24-bit data-words and are available at a rate of 8 kSPS or every 125μs.

For communications it offers serial interfaces I2C, SPI or HSDC. For harmonics reading, HSDC in burst mode reading is recommended although not used since it burst into the line the content of all the registers in a row while only seven are needed, three harmonic indexes of neutral current, RMS neutral current and three phase RMS current.

Regarding its working modes of interest, as energy meter, where all fundamental, apparent, reactive or accumulated instantaneous or RMS phase powers and their components are calculated and many different properties of the signal, not relevant to the research, can be computed. As a harmonic meter contains a harmonic engine that analyses one phase at a time. Harmonic information is computed with a no attenuation pass band of 2.8 kHz (corresponding to a -3 dB bandwidth of 3.3 kHz) and it is specified for line frequencies between 45 Hz and 66 Hz. Neutral currents can also be analysed simultaneously with the sum of the phase currents. Figure 82 at p.58 of the Data-Sheet presents a synthesized diagram of the harmonic engine, its settings and its output registers. Working in its normal power mode, it draws a

maximum current of 5.8mA at 3.3V. Other reduced power modes can be selected but not used since the wake-up delay affects significantly to the measurements.

3.3 Selecting an MCU

The YRKRX63N board equipped with a Renesas RX63N (R5F5631BDDFP MCU) 100MHz 32-bit MCU with on-chip FPU, 165 DMIPS, 1.65 DMIPS/MHz, with a price rounding 13€ (Digi-key electronics Finland May 2015) was selected for prototyping. Joins all the required capabilities in one MCU, due to the availability at the moment of selection and the VASPEC's predilection for Renesas MCU for their long product longevity. This IC provides enough communication controllers, while UART and SPI are used and Ethernet planned for a possible future implementation. Includes PMOD connectors to support a variety of generic PMOD devices and will be the port in use to interface the AFE. A later independent PCB design was not necessary. The board is also equipped with a Segger J-Link hardware debugger that offers valuable information, for example, to optimize the hardware-software interface or to trace the SPI communication by visualizing the registers content.

A DSP library is provided by the manufacturer with 5 different categories of functions, like filtering or transforms like DFT or FFT, optimized to work with the MCU and their compilers. The code is light, the largest function is less than 1kByte and the largest stack memory requirements are less than 100 Bytes. These give optimized code for interfacing directly with the CPU DSP instructions resulting in the lesser CPU clock cycles per function.

Other relevant properties are its large built in flash memory of 1MB and RAM of 256kB, or the availability of several Real-time OS, like FreeRTOS, EmbOS or µC/OS II and µC/OS III. This option, an RT OS, is not in use at this level of the project. The retail cost of 1 unit of this MCU rounds 11€ in the market.

Other options are quickly assessed and, with a similar price and characteristics, was selected an ARM Cortex M4 (STM32F405RGT6 MCU), a 32bit RISC 168MHz MCU, 210 DMIPS, 1.25 DMIPS/MHz, perfectly capable for DSP instructions and

a price rounding 12€ (Farnell Finland May 2015). The immediate availability of the YRDK63N board for prototyping was the reason to select this latest.

3.4 Software environment

One of the project's requirements dictates the language, all MCU implementation has to be written in C. Although regarding MCUs' programming language, where the language limitations come due to the available compilers, C, or embedded C, is the most universal language in the industry environment (Blaza & Wilson, 2011) followed by C++ with less than the half of the designs compared to C. For this, one may find C/C++ compilers for all of the most used MCU on the market and this is the case for Renesas, having toolchains for these languages in their products.

Renesas provides the complete software environment by providing IDEs with their compiler and linker into them. One is their relatively new *eclipse* based *e²Studio IDE* integrating their proprietary compiler and linker into the large *eclipse* IDE environment and providing communication with the hardware debuggers. The *KPIT GNU toolchain* is a solution to the economic limitation of the proprietary solution.

A version control system, *Apache Subversion* ™ software mostly known as *SVN* is used as a centralized repository following the traces of all the changes, branches and trunks. The project files repository is stored in VAMKs server and no installation is required as it is already integrated in schools IT services.

3.5 The code

The code requires special attention whenever the source code has to be maintained along the time or delivered and maintained by third part teams. Most of the code is commented using block comments and the less by inline comments, this avoids the possibility to comment big portions of the code allowing a clear debugging. Every file has a header text introducing the name, the version and description of its content. After this, in the case of a source file, '.c' file, each section is divided by a comment block defining itself, system and project includes block, macro definitions block, local function declarations, global variables, and a new block per each function in the file. Not always all the blocks are needed. The function header block

comment includes the name, a description, list of arguments and return. A header file '.h' includes blocks for macro definitions, variable type definition, and public function declarations.

The ADE7880 driver is documented by means Doxygen to facilitate its maintainability and understanding. Doxygen facilitates the code documenting by means of selected tags around the inline comments. Although a powerful tool to ease the task, the code should be clearly architected from the beginning and changes carefully implemented by accounting the comments as well, otherwise Doxygen code might become a second maintenance task.

The portability of the ADE7880 driver becomes a high concern because the final MCU target might differ from the initial prototype. Regarding to the AFE, this is completely achieved by implementing a driver without specific CPU compiler instructions and isolating the hardware, both AFE and SPI communications by implementing their respective HAL. This is not performed for the other peripheral drivers since they are highly bound to the MCU and their specific registers with single access and single register access to retrieve data, for example, an ADC requires to write once several MCU specific registers to be configured and turned on, and only one MCU specific register is read in order to collect the information.

Other measurements are adopted to facilitate the reading and understanding of each part of it. Individual folders for each driver, HAL in those which require it, naming convention to identify each section. The code's file naming follows a predefined structure such as each driver is wrapped into its own folder where the name identifies the peripheral's driver and its dependence. A folder's name of a driver package related to a Renesas built-in SPI controller is called "r_spi_rx600". The initial "r_" means that the driver targets a Renesas RX peripheral, and the "spi"_rx600" is the name of the peripheral and is exclusive for the rx600 series. Other driver folder's name is "ade7880", identifies a driver for the ADE7880 IC. No reference notation at the beginning of the name means that the code inside is unbound to the hardware, portable, requiring a HAL implementation and providing the hook code, found as "r_ade7880" on the top of it.

3.6 Communications protocols

There are several different communication protocols required. In the first place, the communications that take place among MCU and peripherals in a very short range, less than 10cm that require of a protocol. In this case, SPI is selected for being versatile and common, providing high speed communications. SPI is used by the ADE7880 and the LCD. This latest is used for onsite debugging, displaying RT information and allowing to set the voltage threshold value, not needed in a final implementation. In the second place, the communications with remote devices, the computer used for the testing environment, that receives, plots and stores the data in case of an anomaly, and an RTU running SCADA that receives the information. The computer receives UART signals meanwhile the SCADA RTU requires Ethernet that is not required at the moment and so not implemented in this project.

3.7 Testing environment

A testing environment is required in order to assess the implementation on a reduced scale. For safety and legal reasons, students are not allowed to work with voltage levels over 50V and the conditions had to be simulated at a smaller scale. The test environment requires a computer running MATLAB® and a professional sound card to generate the significant waves.

The specifications state fixed frequencies, the fundamental of the AC current is 50Hz, and harmonic frequencies appear at decimal multiples of a fundamental frequency, being the harmonics of the 3^{rd} order 3 times 50Hz, resulting in the component of 150Hz, the 5^{th} resulting in 250Hz and 7^{th} equals to 350Hz.

The selected method was by means of SIMULINK® blocks. One block needed as DSP with a matrix of 4 frequencies with independent amplitudes and the output connected to a second block as the default audio device. Other methods were evaluated but discarded due to the simplicity and effectiveness of the chosen. Once the SIMULINK® model was implemented, only a cable, soldered to a Jack connector plugged into the soundcard's output, was needed to feed the AFE input pins.

4 METHODOLOGY AND IMPLEMENTATION

Four different frequencies are generated by Simulink® and output from the sound-card of a computer as an analog signal. Having then a fundamental of 50Hz and 3rd, 5th and 7th index harmonics, with a maximum amplitude of $1V_{p-p}$ and the possibility to control the amplitude of each frequency independently or to modify the harmonic indexes.

The signal is presented to the AFE neutral current differential input (INP and INN). The same signal is used to feed the phase A, B and C current differential inputs (IAP-IAN, IBP-IBN and ICP-ICN). This is only for testing purposes, to verify that the data from the phase inputs is correctly read. Since only one signal is generated, it is impossible to simulate a 3-phase 4-wire environment. In a real environment, reading each phase and neutral line will allow to determine the faulty line and analyse the missing current with the neutral line current, as well as the phase angle of each of them to approximate the origin of the fault. To finalize, the same signal is presented at one of the phases voltage differential inputs (phase A is selected, VAN-VAP). This is an ADE7880 requirement to use the signal as a time base for the harmonic calculations engine. All the required LPF are calculated and placed to each input.

The AFE is a SPI slave of the RX63N MCU with a 4 wire diagram. Only SPI protocol is used for all the communications despite the selected energy additionally offers I2C and HSDC (High Speed Data Capture). To configure the ADE7880 to perform harmonics calculations, one may, when from power off, follow the power up procedure, establish SPI as the communications protocol and follow the recommended approach for managing Harmonic Calculations.

Regarding to the RX63N MCU, and because the YRDKRX63N prototyping board is used, no MCU PCB is designed for this research. The RX63N PMOD port is configured to allow SPI communications by means of the SPI peripheral number 1 and the Signal Select Line 0, requiring four lines, CS, CLCK, MOSI and MISO. The ADE7880 power mode select input pins PM0 and PM1 are connected to and controlled by MCU digital output pins driven to the same PMOD port, as well as

the ADE7880 $\overline{reset}$ input and HREADY output pins. The same POMD port offers +3.3Vcc and GND to power the MCU side of the Isolators. The Raspberry Pi connector offers IRQ0, IRQ1 and CF1 pins as well. All these signals are interfaced through the required Isolators, and the PCB design presents 3 additional pins, ADE_CF1, ADE_CF2 and ADE_CF2 as energy-to-frequency conversion output pins that may be later used for calibrating purposes.

The software enables all the required hardware, reads an ADC and compares its value with a maximum threshold value. Two methods are implemented to get the ADC readings, by means of the external ADC peripheral or by means of the ADE7880, this can be selected by modifying a definition in the header file named '*definitions.h*'. Whenever the threshold value is crossed, a time stamp is retrieved from the RTC, the CMT is turned on counting up to 10ms, the time of half of a cycle of the fundamental frequency, triggering an interrupt at the end where all the required values are read from the ADE7880. This is repeated during a predetermined amount of time set to 100ms. During this time, the retrieved information is available to be evaluated, with the possibility to trigger a reaction. If no reaction is taken, the CMT will consume all the specified time frame. The information is placed in the transmissions buffer and transferred to an RTU as soon as it is made available. The RTU in this case is a computer receiving UART data that can be formatted to be displayed on a terminal screen by sending strings of ASCII characters, or retrieved, stored, and plot in Matlab® by sending floating point values.

Other peripherals are used. The LCD uses SPI peripheral channel 0 and SSL3 and it displays the RTC time, information about the last event and the threshold value. Regarding the switches are programmed to set a new threshold value and to simulate a spike triggering all the measurement events. Another CMT timer is configured additionally to measure the time spent in each SPI communication and the total time of each cycle, from the detection of an over voltage to the moment the last byte is shifted into the UART transmission register. A led is used to signal the events, staying on until the information is transmitted to the RTU.

The next figure shows the graphical representation of the relationship of different hardware parts and the MCU application with the software drivers to perform the required actions.

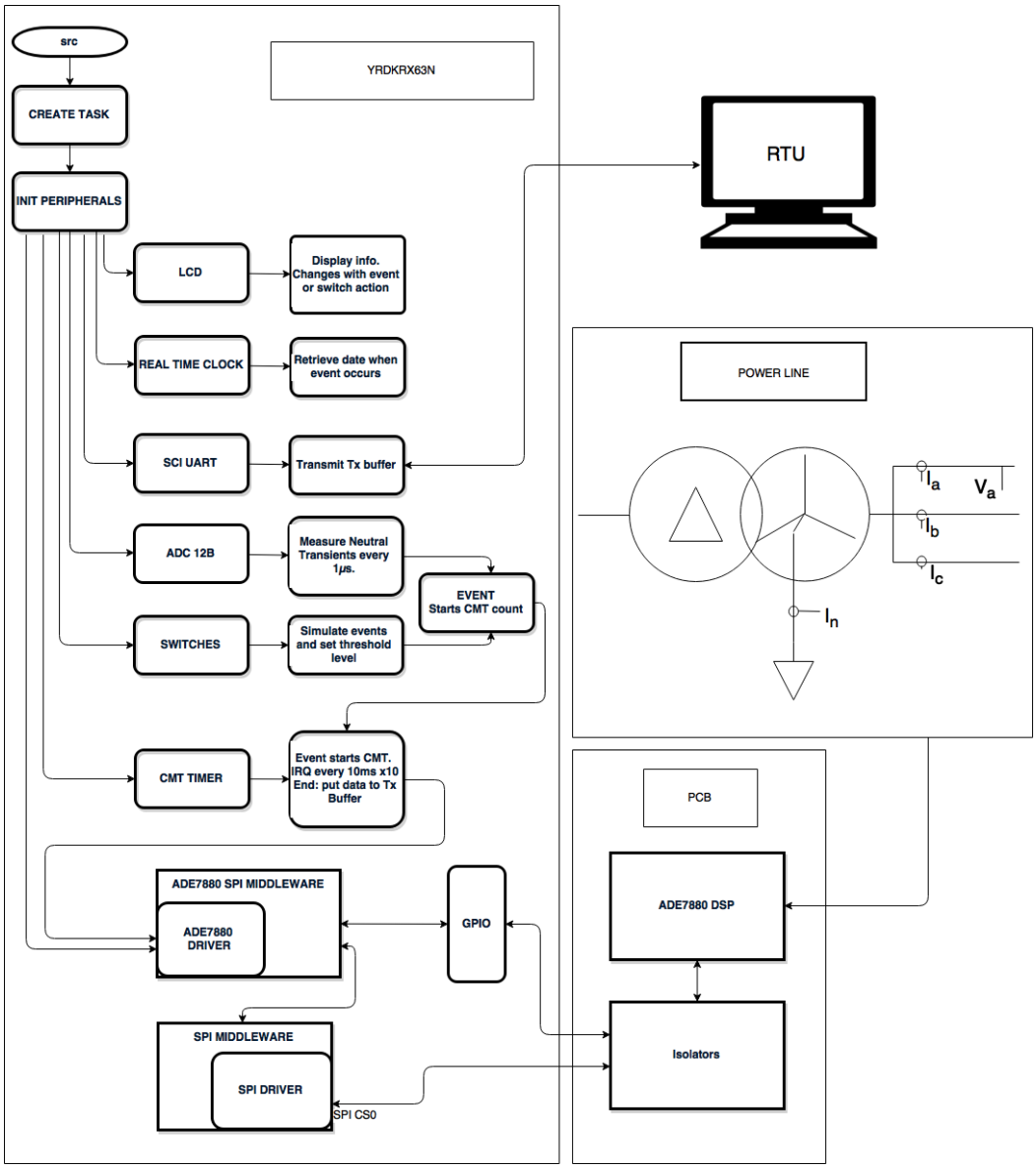


Figure 9 Graphical representation of the system

4.1 Explaining portability, Renesas MCU or ARM

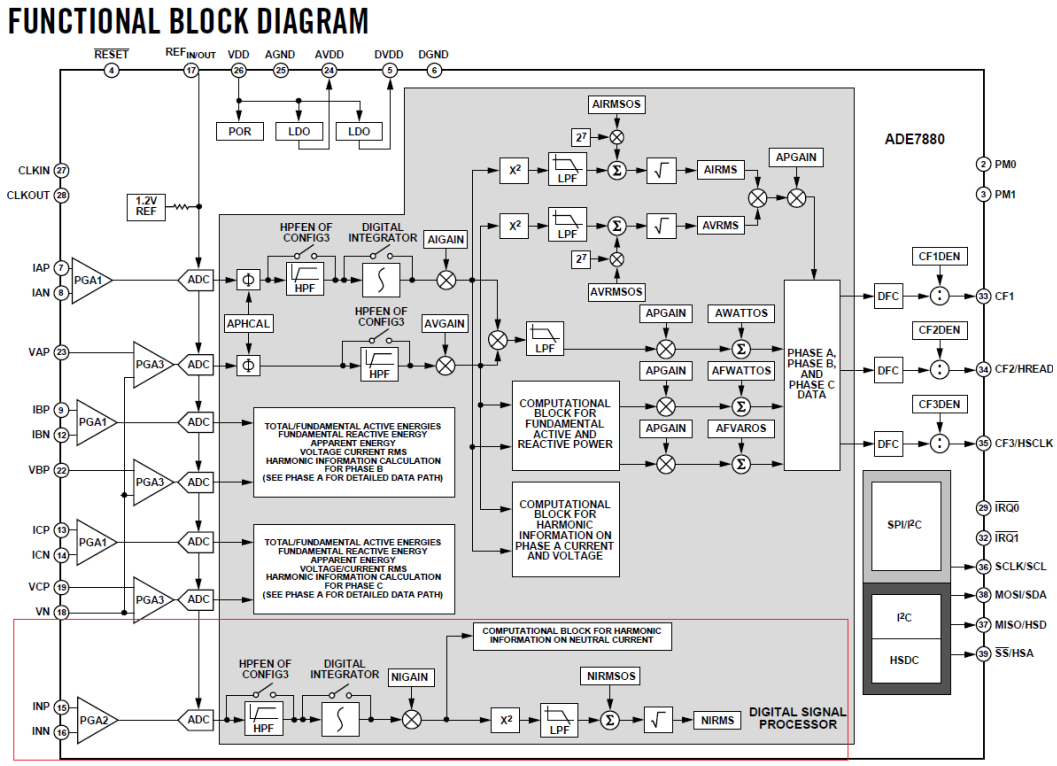
The ADE7880 driver prototyped here has its roots in a preliminary approach by Sami Mahamoud Mahamoed for interfacing the ADE7880 with a Raspberry Pi to retrieve power related values. Later was passed to in earlier stages of the project. This has affected the design of this implementation. The Raspberry Pi is widely used for educational purposes in VAMK, as it may serve as an introduction to MCU embedded coding, Linux and OS programming. Having this in mind, the AFE driver was designed to be completely portable and not bound to any particular CPU or specific compiler instruction. It can be implemented in a standalone system or in an OS. It can run in Renesas or ARM, Raspberry Pi is ARM v7, as it is not bound to any of them. Only requires a middleware layer implementation that can be easily written by following the same schema as shown here.

The PCB design reflects the same procedure and tries to follow the Raspberry Pi compatibility by having a similar size (62.5mm width and 74.5 height), with smaller area, and with a female 40 pin header that fits on the Raspberry Pi v2 IO male pin header. A prototype carrying a compatible connector, may be attached to a Raspberry Pi connector and the Renesas PMOD connector, using the latest, an adapter cable made by following the schematics in Appendix 5: Schematics

ADE7880 and connector socket).

4.2 Prototyping the AFE, ADE7880 input channels

Before the PCB was designed, a prototype of the AFE board, input filters, connectors and power source were located over a breadboard having all the required input and output pins clearly distributed. Seven differential inputs of which five are required, fourteen lines to interface the MCU PMOD port and all the filtering capacitors and resistors were placed on top of it to start the MCU interface. Care should be taken regarding ESD.



- Neutral current processing path

Figure 10 Functional block diagram, from ADE7880 datasheet

In the functional block one may observe that the neutral current measurements engine follows an independent path. Phase energy values share a common computational engine in which the neutral current does not interfere, and, regarding the harmonics, this last has its own. This is could be a relative drawback in other kind of implementations than this, since the ADE7880 has one DSP, meaning that either one phase or neutral line harmonics can be measured at a time. The computational block for harmonics calculations will allow to obtain a reading of the neutral current RMS value. This is important since the engine can output three harmonics indexes at a time but four are needed, the named 3rd, 5th and 7th, and the fundamental as well. The fundamental component and related THD are only calculated for the phases, they cannot be measured by the neutral line engine, obtaining the total RMS current instead by reading the NIRMS register. There is no output either to the THD value of the neutral current. Although there is no need to retrieve these values following

the given specifications, they can provide valuable information therefore appearing as a relevant limitation for data analytics.

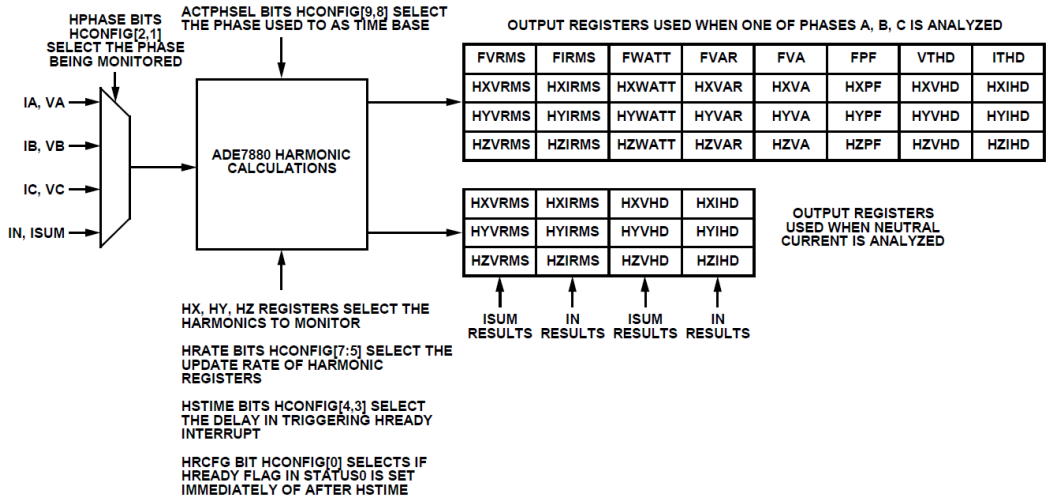


Figure 11 Harmonic engine block diagram, fromADE7880 datasheet

When using the ADE7880 the sample rate is fixed at 1024 kSPS, meaning that any frequency higher than 512 kHz will generate aliases and the antialiasing input filter has to take it into account, hence the input passive low pass filters are calculated at this stage. These filters have to meet two requirements, by one hand, cut the non-wanted high frequencies and, by the other, avoid as much as possible the phase shift.

With the help of LTSpice, these filters can be simulated to find the best approach. For testing purposes, an analog LPF with a *cutoff* frequency is calculated at 15 kHz having attenuation of 31dB at 512 kHz and a phase shift of -1.5º. The ADE7880 recommends a *cutoff* of 5kHz (Analog Devices, 2014), one may observe that such a filter can be built with an RC where R=15Ω and C=2.2µF. Its 0 attenuation bandwidth does not go further than 300Hz and producing a phase shift of 4º at the frequency of the 7th harmonic and up to 40º in the bandwidth of interest, while having a 40dB attenuation at 512kHz. Two different approaches may follow, a first one, the selected for testing purposes, having a relaxed RC values but allowing a higher bandwidth respecting the original signal properties.

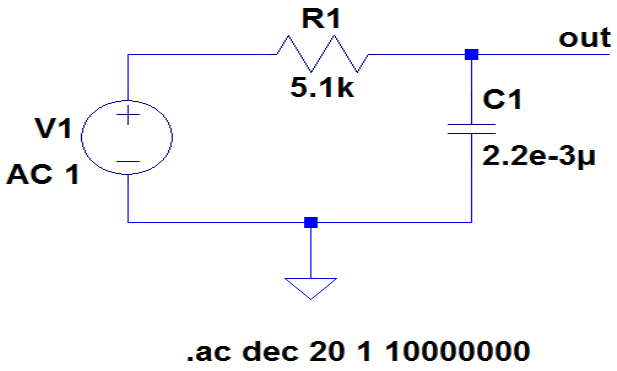


Figure 12 Input current path low pass filter simulation schematic

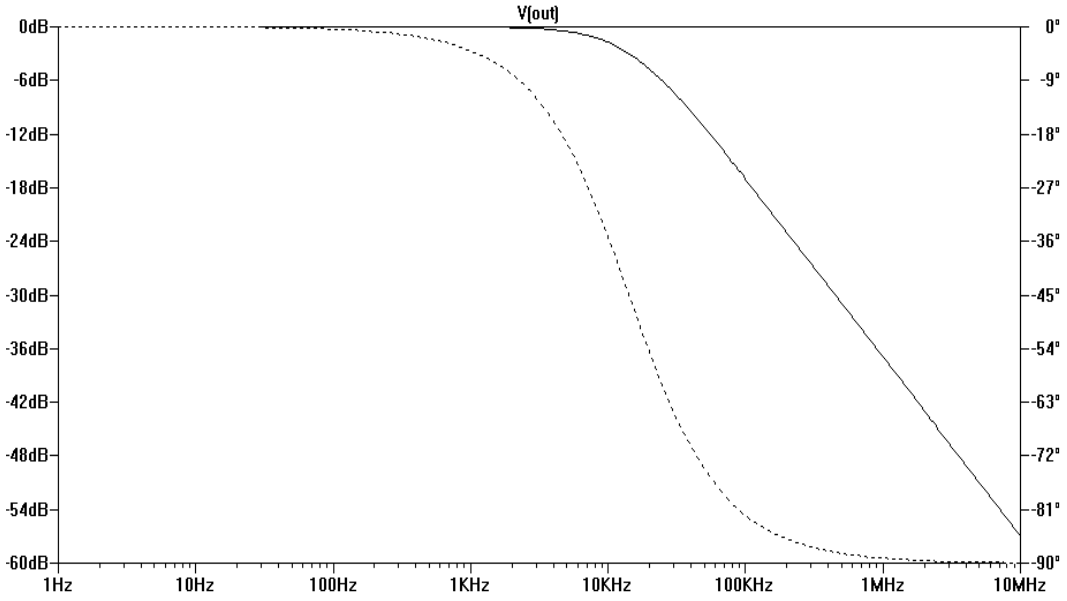


Figure 13 Current low pass filter simulation Bode diagram R=5.1k C=2.2nF

With the given RC values of 5.1kΩ and 2.2nF the cut-off frequency is 15 kHz, not affecting the input bandwidth of the IC of 3.3 kHz. Having 30dB attenuation. The target of these selected relaxed RC values is to avoid possible future testing environment limitations when the case is to select higher harmonics indexes than the 7th and keeping the original properties of the wave at higher frequencies.

In a second approach the RC values have been analysed to fit specifically a measurement up to the 7th harmonic. In a final design with a fixed value of the 7th index as the maximum harmonic, the recommended RC filter implements a resistor of

22kΩ and a capacitor of 2.2nF, having then an attenuation of 27mdB with a phase shift of -5.4º at 350Hz, and an attenuation of 44dB at 512kHz.

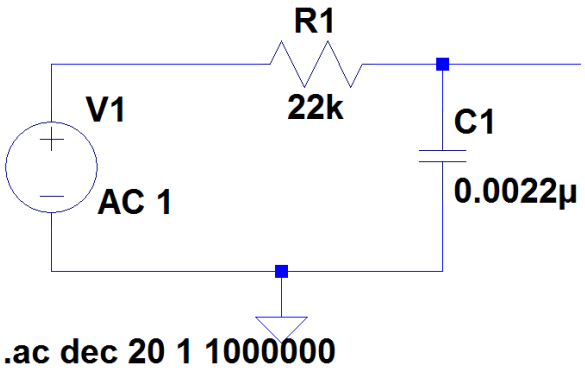


Figure 14 Optimal current input low pass filter for a highest 7th harmonic index

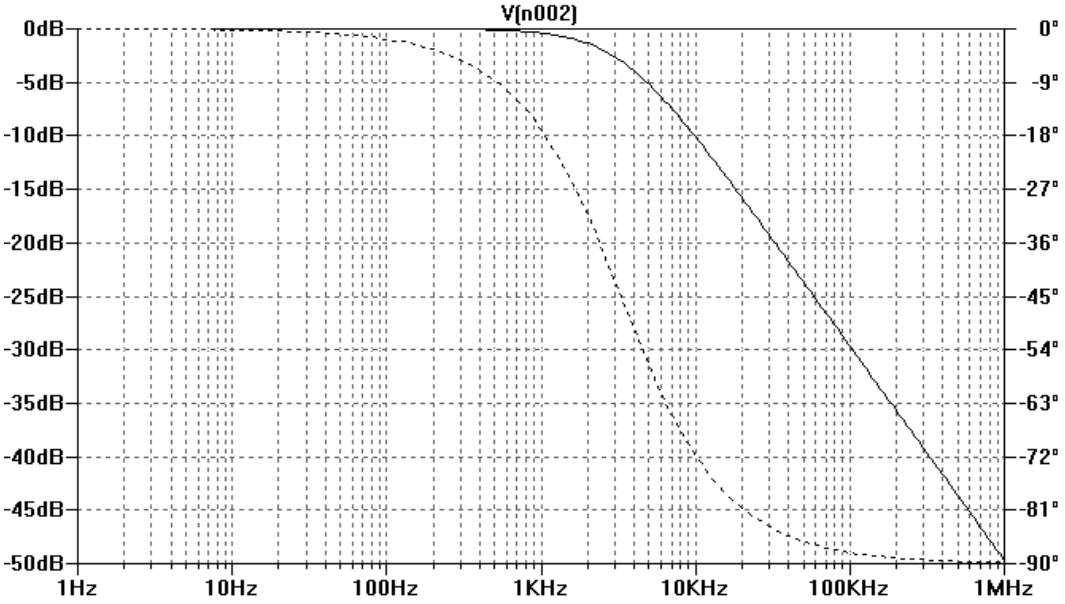
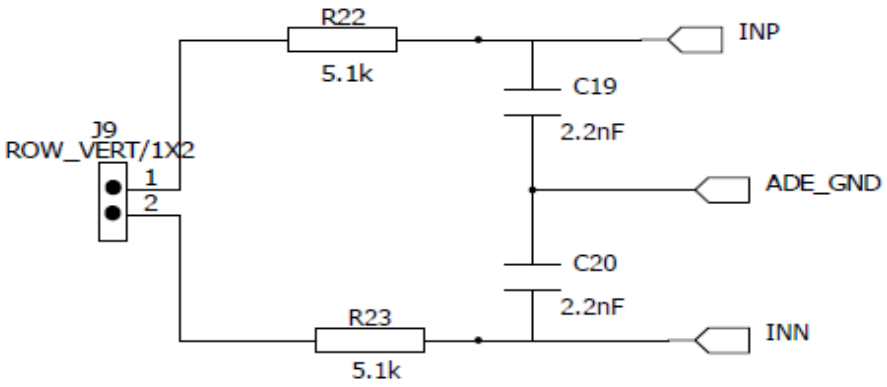


Figure 15 Optimal current low pass filter Bode diagram with cuttoff at 3.5kHz



Where

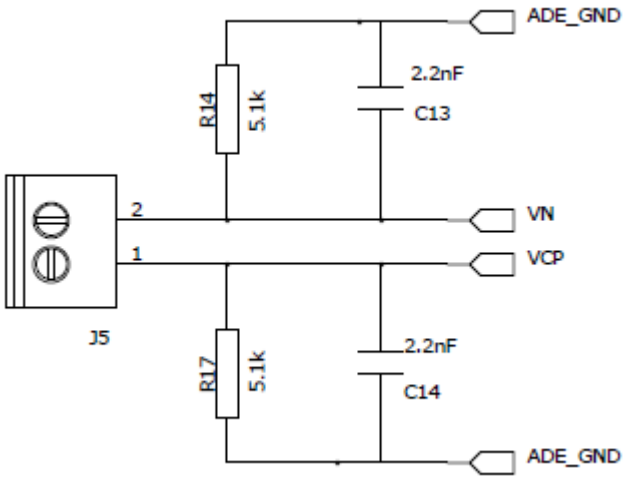
INP Neutral current positive differential input

INN Neutral current negative differential input

ADE GND ADE7880 GND Voltage level

IAP & IAN, IBP & IBN, and ICP & ICN have the same configuration

Figure 16 Current input antialiasing filters schematic



Where

VN Neutral line Voltage input

VCP Phase C Voltage input

VAP and VBP voltage inputs have the same configuration as VCP

Figure 17 Voltage input antialiasing filters schematic

The ADE7880 ADCs have differential inputs that accept a maximum range of $\pm 0.5V_p$ to preserve the precision and accuracy specified in its documentation. In

fact they support sporadic maximum differential voltages of 2V but anything above $\pm 0.5V_p$ cannot be accepted as an accurate measure. Hence the CT or Rogowski Coil outputs should be translated to values in this range before its output is sourced to the ADE7880 current input pins. And the voltage inputs require an external resistor placed in series and forming a voltage divider with the filter's resistor (R14 or R17 in Figure 17 Voltage input antialiasing filters schematic) calculated as follows,

$$R(k\Omega) = 10.2V_p - 5.1 \approx 10V_p$$
$$R(k\Omega) \approx 7V_{rms}$$

Where

R resistance in kΩ of the required input resistor

V_p maximum instantaneous input voltage

V_{rms} input rms voltage

In the testing environment they are unnecessary since the input signals are provided by SIMULINK® and the amplitudes are a software variable.

Other I/O of interest are the communication paths, MOSI, MISO, clock signal SCLK and signal select $\overline{SS}$ the power mode PM0 and PM1, interrupts , HREADY and $\overline{RESET}$ pins, all sourced to the MCU through the PMOD port.

Whenever this IC is measuring power lines requires isolation thus extra circuitry is required. Since in this project there is already a 5V power supply that provide the system with energy, the same is used, and a dc to dc isolated regulator is placed to power the energy meter. Additionally, Texas Instruments ISO7240 high speed Isolators are placed to protect the signal traces.

The required protection circuitry is not necessary when working with a testing environment. The test signals are, according to the ADE7880 ACD inputs, in the range of ±0.5V and absolutely harmless, thus the energy meter receives the power form the YRDKRX63N PMOD port which outputs 5V, and a common AMS1117 3.3V is used to convert the voltage level. With the same, the optoIsolators are not required

and not implemented in the testing prototype, establishing a direct connection between the MCU and ADE7880. Regarding the current inputs, in a real case, most of the CT and Rogowski coil in the market work with low output voltages, many inasmuch as 1V range making them suitable to be directly connected to the energy meter. This project has tested a CT which outputs 1V at a maximum current of 30A directly connected with controlled maximum currents up to 3 Amps.

The Appendix 6 OptoIsolators, in the PCB design section, shows the schematics of the required isolation circuitry as they are in the PCB which can work in any condition, testing or real, with the only need of implementing the series resistor to the voltage lines input.

4.3 Interfacing the Transients. Independent ADC

The transients appear whenever an event occur in the line, they are of unbalanced nature, and so they might occur on an individual line not affecting the other. As a 3phase 4 wire always tries to reach a steady state, these unbalances are always reflected over the neutral line. For this reason, one fast and sensitive ADC is sensing the neutral permanently. A threshold level is set by software that whenever crossed, sets a flag that will trigger an avalanche of measurements to evaluate the situation.

The ADE7880 ADCs can perform measurements at 1MHz, implementing internal registers to set threshold values to signal over currents and voltages in each phase. Associated to the threshold level, an interrupt is triggered when any phase drops below or grows more than the selected level. Additionally, voltage and current peak detection register hold the respective named values. Although the interrupt signal can be used to fetch the data immediately, the SPI communications might be a bottleneck in tight timing requirements, for which my recommendation is to select a MCU, with an ADC that meets the accuracy, precision and speed, among its peripherals and fetch the readings using a DMA channel. As the given timing specifications are relatively relaxed for the RX63N, there was no need to implement DMA in the design, fulfilling any task, under any circumstance, into the expected timing.

However, an independent 16bit ADC is given by the specifications as a valid resolution although a 12bit ADC, available on the YRDKR63N board is accepted and will be used for testing.

4.4 The software environment

The software solution is implemented in C language built in eclipse environment e^{2}studio using only generic C99 specification instructions and libraries. The proprietary RXC toolchain, the set of compiler, assembler and linker in use are provided by Renesas as an evaluation version with a limitation of 128KB of linkable object for non-commercial software, although a GNU solution is available under the name of GNURX by KPIT. The software project uses Apache Subversion SVN repository as a version control system stored in VAMK dependencies. It includes three different build options, Debug using a RX GDB simulator, Hardware debugging using Segger JLink hardware debugger and Release.

Hardware debugging is the most extensively used mode that, thanks to a connection to the embedded Segger JLink debugger in the YRDKRX63N board allows access not only to the software content but to the CPU and peripheral hardware registers content. It is completely integrated in eclipse and works altogether with GDB giving a detailed information. Hardware debugging requires of a connected device. When there was no possibility to have a connection with the hardware, Renesas e^{2}studio provides a RX simulation environment that emulates the RX63 hardware and uses GDB, useful thou limited compared with a hardware debugging session. The Release build is used only for demonstrating purposes.

To write the RX63N highly hardware dependent code and drivers the RX63N Group hardware user's manual (Renesas Electronics, 2014) is needed providing the information to all of the available MCU registers and their properties. The start-up environment, both, by Renesas or Kpit, provide with the lowest level hardware access, a set of functions and header files with IO definitions to program and start running the CPU in a way that the entry point to the programming becomes already the user code.

The following code manages all the required MCU hardware, ADE7880 communications and RTU communications. The required peripheral drivers are initialized and started immediately if needed. All the drivers but three, have very low level access and their flow diagram is very direct, retrieving the content of a specific register in a single function. Three other drivers require of a low level access to the hardware or HAL and middleware software layer to read or write to the peripheral, offering a set of public functions that can be call when inserting the driver in an application. They are ADE7880, SPI and LCD drivers, although LCD is not part of this research since it will not be part of any final solution.

The ADE7880 driver can be ported to another environment with little to nothing to modify. Requires to implant a software layer following the same routines shown by this code. Special care is taken with the SPI hardware driver, since has to meets the requirements to work in an OS or having several devices accessing to it at the same time. It is re-entrant and non-blocking at a basic level, two or more different devices, and with or without OS, can be calling seamlessly the SPI driver. There is no OS in this solution, but re-entrancy is required to handle the ADE7880 and the LCD screen SPI simultaneous access. It was successfully tested in Renesas RX63N, with no OS, and Raspberry Pi ARM v7 with Linux Raspbian OS.

4.4.1 Folders and files naming standard and structure

Following the same structure as all along the code, folders starting by ‘r_*’ are specific Renesas application drivers to operate the peripheral indicated by the tail name. As in this case ‘r_ade7880’ indicates that is the driver that provides control and a set of functions and callbacks to drive the ADE7880 IC.

Inside the top folder, one may find subfolders named by the peripheral's name, 'ade7880', indicating that this is an external peripheral with its HAL and driver. Files starting by the peripheral name 'ade7880_*' are specific HAL and hardware related code, build to be independent to the µC and portable C code, such in a way that only a set of files in the parent folder are required as middleware to allow other systems’ specific code interact with the driver. Files starting by 'r_*' followed by

the peripheral's name, as 'r_ade7880*' belong to the middleware that provides access to and from the Renesas MCU with the ADE7880 IC (see Appendix 1)

4.4.2 The data storage. A container for the information

A place to store it is provided as an object. This is a structure that provides memory location for all valuable data. This container offers a place for the data that may be collected from the energy meter and the time stamp of the measurement keeping an easy and known form to retrieve what it is needed. An example of it can be seen in the Appendix 2.

The structure is defined only once and local variable declarations to store the same information are not needed. The container is allocated when the application is created and is passed as a reference to other functions. One memory location provides light code, avoiding the allocation of extra memory to store the same data each function call reducing significantly the memory requirements of the application and the clock cycles required to process the allocation and deallocation of each variable and process, therefore obtaining a faster execution. Another consideration is that the object is allocated only once and in the stack memory, which is of faster access than the heap (Dobry, 1993).

Other positive aspect is the flexibility. Since the object is a parameter in all the functions calls to the ADE7880 driver access, the functions always return the error occurrence, therefore having more information than returning a single local variable value. Moreover, the container makes the data always fully available wherever needed, avoiding a long list of parameters in each function declarations with each individual value.

The object allocates memory for one event’s data, does not audit the history of events because each event is transmitted immediately to an RTU, where this will be processed, relying then in the communications buffer. When required, one may always declare the object as a buffer, as an array of objects, which its iterations may have to be handled. This will do the work of a local and volatile, storage requiring

larger amount of memory. If necessary, an external and non-volatile memory access, like an SD-card, where to save the information to a file, should be implemented.

There are drawbacks of having one object for everything that matters and all along the code. The access to this object has to be handled carefully to avoid data corruption. This may appear, for example, when having concurrent access requests, at the same time to the same memory location, one access may be writing what the other is reading. When non handled interrupts may occur or using an OS with multiple threads accessing the driver, one should protect the access to this memory location with any of the known techniques, i.e. disabling interrupts, setting flags or locking mechanisms like semaphores when using OS. This is not happening in the implementation proposed here since event interrupts don't occur while in the window frame of an event's information retrieval. The transients ADC readings relinquish the execution to the information retrieval algorithm segment and is not attained again until the process finish.

4.4.3 The ADE7880 Driver

Provides high level access to all the available registers of the ADE7880. While the individual registers can be read or write independently at the low level access, it provides two public functions at a higher level, that perform all the operations needed for this implementation and most of the common operations for an energy meter, not including operations like energy-to-frequency measurements, or calibration registers access, although any register can still be read individually.

4.4.4 ADE7880 low level access, HAL

The ADE7880 provides known hardware memory locations or registers, information which is provided by the manufacturer. These registers are mapped in the file '`ade7880_registers.h`' that follows the hardware datasheet registers list section preserving the naming convention. Registers bit length varies, 8, 16, 24 and 32 bits, signed or unsigned. Wherever possible, bit access level is provided to favour the

access to configuration registers. This is achieved by generating unions as new variable definitions as follows:

```
/** CONFIG Register address and structure   */
#define CONFIG              0xE618      // ADE7880 configuration register.
                                        // See Table 47 DS Rev. A| Page 97 of 104.
#define CONFIG_DEFAULT      0x0002      // Constant macro for its default value
typedef union
{
    uint16_t REG_ALL;
    struct
    {                                           //bit   Def.Val
        uint16_t    INTEN       :1;     //  0       0
        uint16_t    RESERVED    :1;     //  1       1
        uint16_t    CF2DIS      :1;     //  2       0
        uint16_t    SWAP        :1;     //  3       0
        uint16_t    MOD1SHORT   :1;     //  4       0
        uint16_t    MOD2SHORT   :1;     //  5       0
        uint16_t    HSDCEN      :1;     //  6       0
        uint16_t    SWRST       :1;     //  7       0
        uint16_t    VTOIA       :2;     //  9:8     0
        uint16_t    VTOIB       :2;     //  11:10   0
        uint16_t    VTOIC       :2;     //  13:12   0
        uint16_t    RESERVED2   :2;     //  15:14   0
    }bits;
}CONFIG_reg_u;
/* End of CONFIG  */
```

Figure 18. ADE7880 Hardware registers access example

Defining the address of the register with a name offers readability to the code and simplifies the hardware access. A default value can be defined simplifying the task of writing to the defined address. The bit access is provided by a new variable definition, '*CONFIG_reg_u*', resultant of the union of the two members, one for whole access and one for bit access. The first gives all bit range, named '*uint16_t REG_ALL*' in the previous figure, announces that it is a 16 bits register and provides access to the whole content at once. Next line after it is the declaration of the second member of the union, a structure named '*bits*' which provides a name to each bit available in the register.

4.4.5 Public methods, high level access

The two most important operations are writing the ADE7880 configurations and reading the measurement from it. For this reason, the driver offers two public functions to handle the access to the energy meter. These hide lower level access like registers or SPI communications form the user side. They are performed by the

functions '*ADE_command_handler()*', which perform configuration operations and *ADE_measure()*, that retrieves the measured values. Their signatures are as follows:

```
error_list_et    ADE_command_handler     (uint32_t pid, uint32_t cmd, uint8_t arg,...);

error_list_et    ADE_measure             (uint32_t pid, uint8_t cmd, uint8_t channel,
                                          uint16_t samples, void *result);
```

The public functions share common features in their signature. The output or return

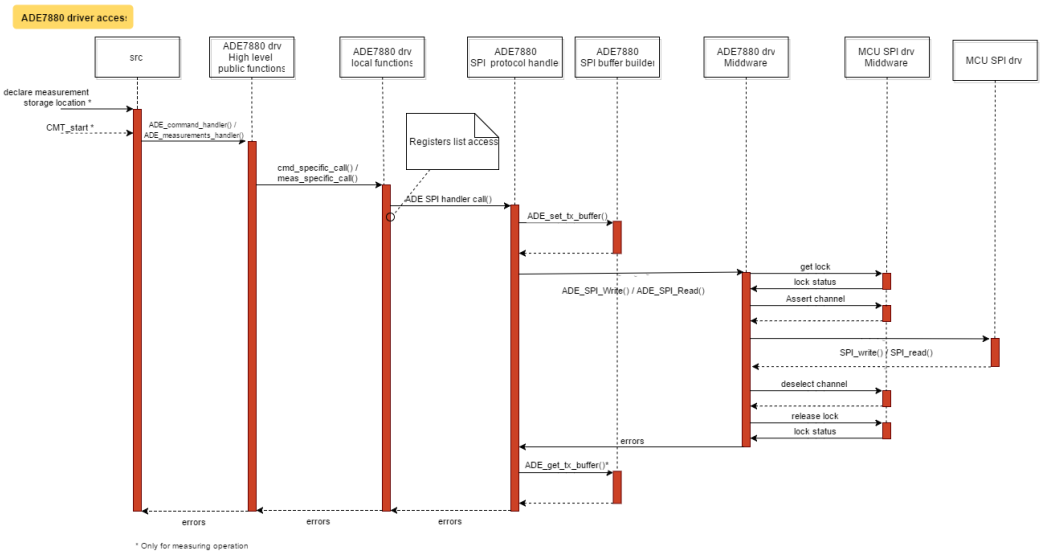


Figure 19 ADE7880 driver user sequence diagram

is the error occurrence which references are contained in '*error_list_et*' enumeration defined into '*ade7880_configuration.h*' for its identification, where 0 means no error and up to 14 different errors are defined. This output is propagated along the code from its origin to the caller, and by caller we understand the user application that originated the request to the energy meter. Regarding the input parameters, '*uint32_t pid*' belongs to the process identifier, a 32 bits integer unique ID that is exclusive of the caller and required to implement the multiple access to the SPI driver from the ADE7880 and the LCD. And '*uint32_t cmd*' which is a set of individual instructions for configuration processes defined by '*config_cmd_et*' enumeration or measurement options defined by '*measurement_cmd_et*' enumeration defined into '*ade7880_srv_cmd_handler.h*'. Regarding the command handler signa-

ture, it additionally requires a variable list of arguments. Declared after the command instruction, different configurations require a different number of o parameters and the code's documentation or a quick review to each case into this function gives the information about the correct set arguments. The distinct measurements parameters are unambiguous, requiring in this order, the channel, '*uint8_t channel*' referring to one of the three phases, A, B or C, or the neutral line, '*uint16_t samples*' referring to the number of times the measurement is read and later averaged and one last, '*float *result*' which is a float pointer to the memory location where the measured value is stored.

The measured values are always stored in the provided container, whose location is passed as a reference all along the code to the ADE7880 SPI received buffer, where the measured value is extracted from the received SPI bit stream and placed in the given location. Passing a pointer to the measurement as an input parameter gives the freedom to use the output return to signal the occurrence of errors and produces a lighter code. Precautions have to be taken when working with an OS to protect the access to this memory location with a semaphore to avoid data corruption or using atomic access. This is not the case in the RX63N application where the implementation follows a determined sequence after an IRQ occurrence, which finishes before the next IRQ can be triggered. Inside the IRQ, when reading the measurements, follows a sequence with no possibility to corrupt the data. Additionally gives the freedom to build an object, a structure variable in C language, with capacity to all the required information, avoiding the creation of individual variables with each portion of data.

The data container is a set of new type definitions linked from a lower to a higher level. The file '*definitions.h*' provides with a specific example of modelling this data object. The following is the structure, in a human readable form, for stored measurements and its related timestamp (Appendix 0):

```
container.which_phase.phase_specific_measurement.measured_value
container.which_phase.phase_specific_measurement.time_stamp
```

This favours the use of a pointer to the location of the '*container*', offering the access to store to or retrieve required data from anywhere the pointer is referenced. This pointer is casted as a null pointer allowing the user to build its own container without modifying the code, but requires to understand how the transmission and reception SPI buffers are built, otherwise follows the pattern given here.

4.4.6 SPI hardware access and Middleware layer

The ADE7880 driver has to be bounded with the hardware SPI driver. Having defined an action like read a measurement or write a configuration command, the ADE7880 driver's SPI buffer have to be build, the access to the SPI hardware driver has to be safely acquired and granted and the ADE7880 driver's transmission buffer transferred to the hardware SPI buffer registers in a command operation, and read back the hardware SPI registers from the ADE7880 receiver buffer for a measurement operation. Basically the first operation is unidirectional, the MCU sending a bit stream with a write command byte (1 byte as 0x00 in hexadecimal notation), the destination memory address byte and a value to write to the ADE7880 register address. Whereas the second is a bidirectional communication that follows the same pattern, but sending a read command byte (1 byte as 0x01 in hexadecimal notation), and an address to read from, expecting something in return from the ADE7880 immediately after this sequence, see Figure 20. This transmission is handled by the hardware registers of both devices, energy meter and MCU. The drivers provide a safe and exclusive access to these registers. Therefore, the AFE driver has to be aware of the MCU SPI hardware driver.

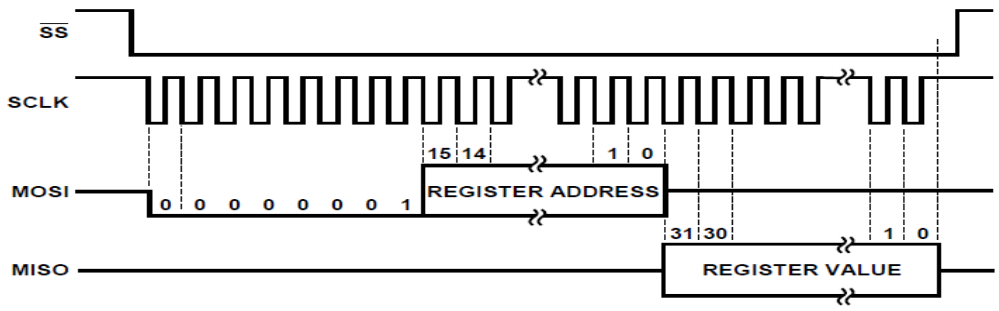


Figure 20 ADE7880 SPI Read operation of 32 bit register (Datasheet p.79)

The AFE driver is unaware of the hardware SPI driver, this is masked by an intermediate middleware layer between both of them. This layer is bound to the given hardware and its SPI driver, requiring some additional rework to port it to another platform. Provides GPIO translation and ADE7880 boot up sequence until SPI is stablished and locked into its configurations as the communication protocol. The major functions provided by this middleware layer are SPI driver read or write:

```
int8_t R_ADE_SPI_Read  (uint8_t *data, uint8_t usBytes, uint32_t pid);
int8_t R_ADE_SPI_Write (uint8_t *data, uint8_t usBytes, uint32_t pid);
```

They are implemented into '*r_ade7880_drv.c*' with other required functions and, under the same folder, '*r_ade7880_gpio.c*' holds the GPIO configuration. Other

The architecture of the SPI reads or write functions is simple. First step tries to obtain a channel's lock to the SPI peripheral, this lock mechanism is implemented in the peripherals driver returning true or false, and, as the LCD uses a different channel, this is true unless the call is originated by an IRQ or it is an OS with multiple applications requiring access to the same SPI channel. Secondly, when the channel's lock is obtained, assert the required RX SPI channel Signal Select pin, the LCD uses a different one, therefore the ADE7880 SPI channel CS pin should always be inactive before this operation. Third step proceeds to read or write to the SPI driver, and when this execution finalizes, the CS pin is deselected and, finally, the lock is released. The CS pin is asserted manually for the freedom that gives to choose any of the I/O pins, but it could be also triggered automatically by configuring the MCU SPI registers to perform that operation, skipping then steps two and four.

The SPI hardware access is controlled into the SPI hardware driver. After initialization, it offers direct hardware access in read or write functions. These functions receive a pointer to the location of the transmission buffer and in case of a reading operation, another pointer to the storage's location for the incoming data. Around these functions a simple traffic control is built that follows the simplicity of Peterson's algorithm. The turn is managed with a global variable as a flag that is set in

an atomic instruction and the inside control is given by an array as long as the number of channels, 3 in this case, that, if the semaphore is obtained, and its channel slot is free, stores the application ID that obtained the semaphore into its channel slot. In this manner, the driver is re-entrant and non-blocking as other SPI hardware channels are allowed to write to their independent hardware registers while their slot is free. More complex algorithms like Bakery algorithm (Shankar, 2013) are not implemented.

These middleware layer functions are made available to the AFE driver by means of function pointers and it is mandatory to initialize them before the driver is made available. The diagram shown in **¡Error! No se encuentra el origen de la referencia.** reveals the methodology implemented to maintain the driver unaware of the link to a dedicated SPI hardware driver.

The ADE7880 SPI handling occurs inside '*ade7880_spi_protocol.c*' and has as main functions:

```
error_list_et   ADE_SPI_read    (uint16_t target_register, uint8_t reg_len,
                                 int32_t *result, uint32_t pid);

error_list_et   ADE_SPI_write   (uint16_t target_register, uint32_t value,
                                 uint8_t reg_len, uint32_t pid);
```

These functions call, in their code lines, to the known middleware layer methods to access to the SPI hardware driver without knowing about them. To achieve this, two function pointers are declared locally with the names of '*ADE_SPI_WRITE_CALLBACK*' and '*ADE_SPI_READ_CALLBACK*' into the driver's file '*ade7880_spi_protocol.c*':

```
int8_t  (*ADE_SPI_WRITE_CALLBACK) (uint8_t *data, uint8_t usBytes, uint32_t pid) = 0;
int8_t  (*ADE_SPI_READ_CALLBACK)  (uint8_t *data, uint8_t usBytes, uint32_t pid) = 0;
```

Two other public functions receive the address of the targeted outsider function as parameter and assign it:

```
void ADE_SPI_WRITE_callback_set
                        (int8_t (*func)(uint8_t *data, uint8_t usBytes, uint32_t pid))
{
        ADE_SPI_WRITE_CALLBACK = func;
}

void ADE_SPI_READ_callback_set
```

```
                        (int8_t(*func)(uint8_t *data, uint8_t usBytes, uint32_t pid))
{
        ADE_SPI_READ_CALLBACK = func;
}
```

The compiler does not complain because they are locally known, but their memory addresses content are initially undetermined, remaining disconnected before they are initialized by calling '*ADE_SPI_READ_callback_set*' with the address of the function '*&R_ADE_SPI_Read*' as its only required parameter, and calling '*ADE_SPI_WRITE_callback_set*' with '*&R_ADE_SPI_Write*' as its parameter. Then at some point in the code, whenever the driver is initialized and before the SPI hardware is made available to it, the '*ADE_SPI_´xxxx´_callback_set*' functions have to be called in the ADE7880 SPI middleware layer:

```
void R_ADE_ADE7880_driverCallbacks (void)
{
        ADE_SPI_WRITE_callback_set(&R_ADE_SPI_Write);
        ADE_SPI_READ_callback_set(&R_ADE_SPI_Read);

        …
}
```

At this point '*ADE_SPI_read*', the driver's internal SPI read function, uses '*ADE_SPI_READ_CALLBACK*' to communicate with the SPI hardware, that its memory address targets '*R_ADE_SPI_Read*' that will execute and follow the five mentioned steps to achieve SPI hardware access.

4.4.7 RTU communications. UART control

The driver built on top of the UART hardware registers allows bidirectional buffered byte transfer at 115200sps. Interrupts are triggered when the transmission register is empty placing buffered data into it and when the reception buffer is full for retrieving the incoming byte, though there is no use for this latest at the moment.

Any data sent to the driver is treated as a byte stream, one may use it to transmit ASCII characters to display in a terminal screen or a binary transmission sent byte after byte. As the measured values are real numbers, the use of floating point variables is extensive to allocate them, and their ASCII representation and transmission may represent a challenge and inefficient communications. For that reason, the data

which is meant to be received and interpreted by MATLAB® software is sent as floating point of 32bit resolution in binary format, four bytes of binary data. To obtain a byte division, a new variable is defined as the union of a float member and a string of 4 bytes, giving the flexibility to manipulate the data as a float and send it to the UART buffer as a string. Otherwise, data transmitted to a terminal screen is initially formatted and buffered to a string by means of '*sprint()*' C library function and the resultant formatted string is thrown to the UART buffer.

The transmission buffer is initially set to 4092 bytes, this does not represent a problem for its allocation and can be safely increased as the MCU has up to 128KB of RAM memory and the code itself uses only 22KB. The buffer is calculated and big enough to handle all the possible incoming data. As the code inserts no delay between an event and the next while expecting the situation to be handled at any moment, data is continuously gathered and placed to the transmissions buffer in sequences of 10ms during 100ms. Data transmitted to MATLAB® consist of at an estimated amount of 7 different float values, the three phases plus neutral currents and three harmonic currents indexes, of 4 bytes each plus a delimiter character making a total of 35 bytes, while each data string transmitted to a terminal program have an average of 25 bytes times 7, making 175 bytes per event, having 11 events per sequence of 100ms, plus the formatting strings that add an extra overhead of about 400 bytes resolving to a total transferred data of 2.71KB. With a transmission speed of 115.2Ksps the buffer should empty in 23.5ms, in a window of 100ms between events.

The transmission buffer is a circular buffer where any string can be placed at any time. A global variable follows the index to position any incoming new byte and the number of bytes to transmit. When the transmission register is empty, it begins by placing the next byte in the index order into the UART transmission register and storing the remaining bytes. The UART bus is programmed to start transmitting automatically whenever a byte is placed in its register and triggering an interrupt whenever the last bit in the register has shifted out. The service of this interrupt, will place the next buffer's byte in the register following the index order and decreasing the number of bytes to transmit. As strings are transmitted, a flag controls

when any new string should be shifted immediately into the register or placed in the buffer. When the last byte in the buffer is put into the register by the interrupt, the flag is released. Under the previously estimated conditions, the buffer should never be full, although in that case, new information is hold in queue until the IRQ released a new position. This is a blocking situation that should never happen or should be handled as required.

4.4.8 Real time clock

The time the events occur is recorded according to the content of the RTC registers available as a MCU peripheral. Using the main clock or a sub-clock pre-scaled to 128Hz as counting source, it can generate an interrupt as fast as 1/256 seconds, sufficient precision for the 10Hz of maximum event frequency in the system, although the resolution in use is given in seconds.

The registers can be read at any moment. Having independent registers for each value, seconds, minute, hour, day, month and year, all of them are read sequentially with the events occurrence, no interrupt is required, despite a method was implemented for this purpose to program possible periodic actions. All the registers are of 8 bits unsigned or less but the year which is 16 bit unsigned, and a structure is defined to store them all and provide space for a string, built with all the registers content, formatted with the use of '*sprint()*' to retrieve and print or transmit the complete date time at once.

4.5 MATLAB® and SIMULINK® testing environment

The MATLAB® license of VAMK's school allowed its use for the research and its implementation took place by using the same concept. Setting timers to populate the result of the mathematical formulae of the respective sine waves to the soundcard. This showed a problem regarding to the timers reload and the output to the soundcard that could not be solved and the result was the disappearance of the output during an undefined interval of time while timers were probably, but not confirmed, reloading.

Before adopting this environment, other solutions where examined. A first testing environment was built with a second MCU and a 4 channel 16bit DAC with parallel input for quick data transfers. An interrupt sets the indexes of the sine lookup tables to feed the output pins with the resultant amplitude. The main drawback with this implementation is the additional required hardware and the limitations of a µC regarding high speed processes. Other proven issues were the asynchronies or delays added by the latencies of the interrupts handlers that distorted the wave generation. The idea, initially simple was having a concept problem, while an interrupt with a higher priority is being serviced, avoid other lower priority interrupts to take place, losing its turn and showing at a different frequency, lower than expected. This was practiced with a Renesas MC16/62P 24MHz MCU. A pre-emptive RT OS could be used but the available MC16 µC-OS II was not made precisely for the 62P series needing to port the code. A simplest approach with the same MCU was to calculate the least common multiple of the four involved frequencies, which for 50, 150, 250 and 350 is 5250 and use a single timer with a single interrupt where the four frequencies indexes where processed 5250 times per second, this means, once every 191.57µs. Although feasible, it is a challenging speed to the mentioned MCU. Changing the MCU for a faster model or use a DSP could be a solution too, but the idea was discarded in early stages. Additionally, this type of implementation is noise sensitive.

A first MATLAB® approach was built using timers to calculate the instantaneous vector of the waveform, and throwing its result to the soundcard. Some issues merged in this implementation, for an unknown reason, the wave generation was halted for a short period of time but enough to distort the result, perhaps the OS was affecting the process. And based in the previous schema, this time implementing a third party DSP library. Solving partially the problems of the previous, others arise with the use of a 32bit DSP in a 64bit machine. Additionally, a GUI was built to dynamically control the required frequencies and amplitudes.

Finally, SIMULINK® proved itself as effective as simple, although not exempt of issues that have no solution at the moment. Those do not affect the output wave and its apparent continuous signal, but limiting the available channels of the soundcard.

The odd frequencies in the array list were output through the front right channel of the soundcard and the even through the front left, while the remaining output channels were silent. This forced the use of both channels as only one to every input of the AFE, and, despite having a single multichannel soundcard, its output can be only considered as a mono signal, and the same single signal was connected in parallel to the AFE inputs affecting the signal filtering and therefore the measurements. MATLAB® and SIMULINK® software offered in Technobotnia facilities under VAMK's license are installed in a remote application server, a different machine with a different hardware and probably not recognising properly the sound hardware of the local machine where it is actually running.

The testing environment is built from the basis of MATLAB® with SIMULINK® and a soundcard Realtek 6 channels ALC662. It does produce the four waves, although not including advanced capabilities. As said, limitations in the soundcard's output, probably caused by having the software running in a remote machine, did not allow to output different signal through different output channels. Other issue is the noise appearing at the output of the soundcard signals, which is almost irrelevant since most of this noise has a very high bandwidth that is cut by the filters and although they add energy to the total current, the FFT result at the given frequencies is not altered.

To produce the signals, SIMULINK® is used implementing only two source block objects, a sine wave DSP and a To Audio Device. The former requires little configuration, see picture Figure 21 SIMULINK® implementation, where in the Amplitude section, the amplitudes may be input as an array of four variables [a b c d], the Frequency section receives an array of the same length with the values of the frequencies of interest [50 150 250 350]. The use of variables at this point will help the task of varying them dynamically, by assigning them directly in MATLAB's input console or by the help of a GUI built for the same purpose. In the sample time box the maximum allowed by the soundcard is set. The other block requires less configuration, where device can only be set to the one found by the OS, the OS's primary and Speakers selection, being them all located in the remote computer and not allowing to select the local machine's soundcard. The length of the queue and

the length the waves are played in time can be set, and pressing play in the toolbar will start generating the waves.

The amplitudes, as they are meant to play in the soundcard, are hardware dependent, therefore one signal of a determined amplitude will have a different level using a different soundcard requiring to be corrected to obtain the same output. The resultant wave is a voltage signal, therefore its output should never be higher than the AFE limitations, in the range of $\pm 0.5V_p$. Although the ADE7880 can stand up to ±1V as a maximum transient limit, nevertheless its ADC 24bit registers cannot go further than ±5,326,737 (from 0xAEB86F to 0x514791 nominal values) at the full-scaled input signal of $\pm 0.5V_p$. Hence the amplitude should be monitored to avoid AFE damages.

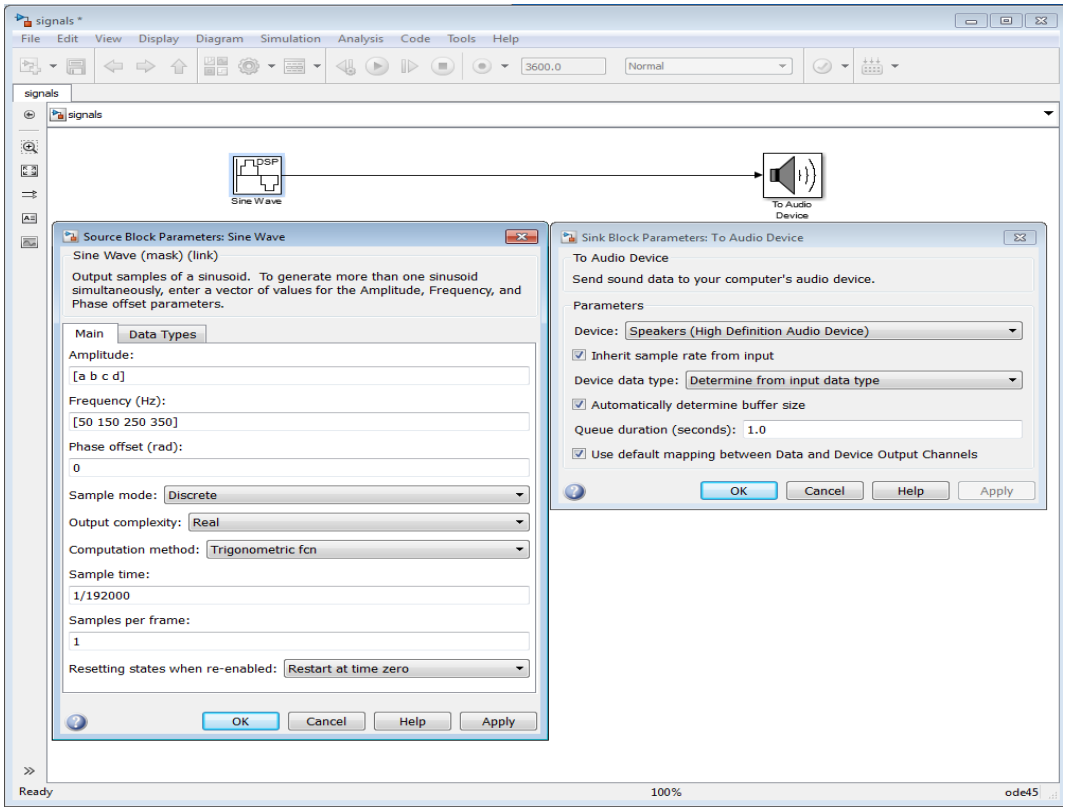


Figure 21 SIMULINK® implementation

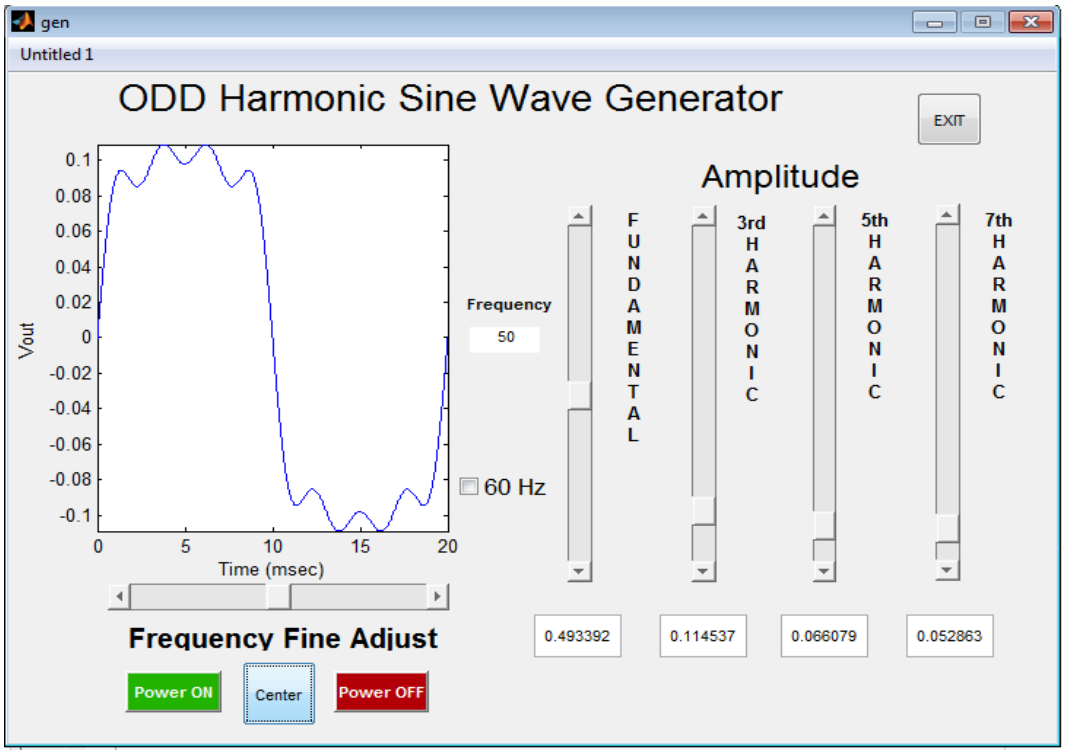


Figure 22 MATLAB® GUI to control frequencies a, b,c and d amplitude

The GUI, seen in the previous figure, facilitates the task of modifying the amplitudes by providing 4 vertical sliders and a graphical representation of the output signal with its maximum voltage output displayed in the y-axis. The GUI allows to power on and off the signal generation and although initially was providing the capability to modify the fundamental frequency from 50Hz to 60Hz and fine tune them with a horizontal slider, there is no use for them in this research.

To present the RTU's received data, two methodologies can be used. One is a terminal console window displaying the incoming information to the UART RTU port, the same MATLAB's computer using a UART to USB converter. The second, another MATLAB® code, used to plot in real time the received measures as a graph, whereas the energy carried by each frequency can be visually compared. The first option was selected as optimal having all the data as lines of strings in the screen.

idx	N IRMS	HX IRMS	HY IRMS	HZ IRMS	Ph_A IRMS	Ph_B IRMS	Ph_C IRMS
1	6.149097	0.00007	0.00057	0.00012	0.00346	0.00514	0.00458
2	6.149097	0.00007	0.00057	0.00012	0.00346	0.00514	0.00458
3	6.149097	0.00007	0.00057	0.00012	0.00346	0.00514	0.00458
4	6.149097	0.00007	0.00057	0.00012	0.00346	0.00514	0.00458
5	6.149097	0.00007	0.00057	0.00012	0.00346	0.00514	0.00458
6	6.149097	0.00007	0.00057	0.00012	0.00346	0.00514	0.00458
7	6.149097	0.00007	0.00057	0.00012	0.00346	0.00514	0.00458
8	6.149097	0.00007	0.00057	0.00012	0.00346	0.00514	0.00458
9	6.149097	0.00007	0.00057	0.00012	0.00346	0.00514	0.00458
10	6.149097	0.00007	0.00057	0.00012	0.00346	0.00514	0.00458

Figure 23 Terminal console showing the data of a sequence of measures

4.6 PCB design

The schematics and PCB design is performed by means of Mentor Graphics software PADS. The PCB is intended to be a portable platform, providing support to hold the energy meter, its isolated power source and every other component required to have an ADE7880 testing board and providing a 40 pin female connector compatible with Raspberry Pi and Renesas PMOD port, this latest requiring an additional adapter, all in a dual layer format of 75mm x 62mm.

Presented in a dual layer format, all components are SMD with the exception of the pin headers and high voltage input connectors. Both layers use intentionally a different plane, the bottom layer, where the AFE and its PSU altogether with power related traces are placed is flooded with ground plane to allow a same reference potential and reduce the conductive noise by reducing the ground connections impedance. The top layer where all communications traces and the fast acting optoIsolators are, is flooded with a power plane adding in this way a distributed interplane capacitance between both planes and therefore better high frequency decoupling. Each plane is divided in two different planes to provide electrical isolation. One part of each plane belongs to the AFE isolated powered supply side and another to the MCU powered side. The OptoIsolators electrically divide both, AFE and MCU, sides of the planes in both layers and therefore, the planes are physically divided under the location of the optoIsolators and being a mirror one to each other layer. Regardless these measurements, the PCB should be placed in an isolating case and avoid direct contact when in use due to the risk of shock. An EMI test was

not executed as this is a standalone device for testing purposes and bypass capacitors are placed regarding device manufacturer recommendations.

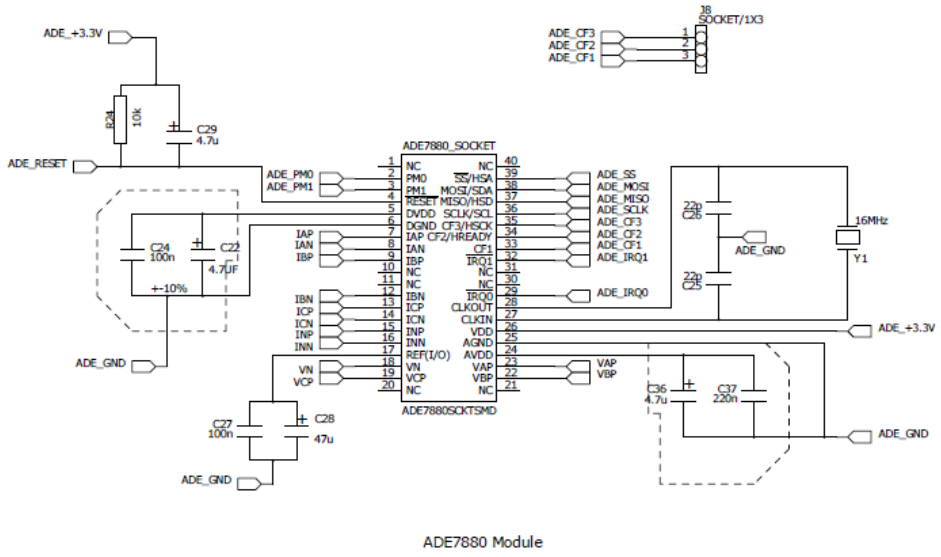


Figure 24 AFE connections schematic

Regarding the AFE connections, a number of issues have to be addressed. One has to account that the ADE7880 requires an external 16.384MHz crystal to set the clock of its DSP. To save space and keep it and its load capacitors near, this is placed in the bottom layer under the centre of the ADE7880 socket, having in this manner, a reduced path. The power mode pins (PM0 and PM1) are internally pulled, and therefore when setting each one of them to a low state by an MCU IO pin, current will be sank to ground, for that reason it is recommended use and enable pins with pull down resistors, although the current that ADE7880 IO draw is as low as 80nA. This is not a problem with many modern MCUs such as RX63N that use tristate IO pins, sinking out current to ground when low or sourcing it when high, having additional pull up and down resistors. This acquires relevance when booting up the energy meter since this pins have to be kept in high state during this process and before any communication can be established. In case of a simultaneous boot up, MCU has to grant that IO can be kept high or in high impedance, state not sinking current to ground during the ADE7880 power up time, otherwise unpredictable AFE behaviour may occur. If this condition cannot be granted, one may trigger

the ADE7880 hardware reset after MCU boot up by setting reset pin (ADE_RESET) to low at least 10µs. Said this, having this short period for a reset signal is very recommendable to pull up this pin up and filter any noise that may trigger a fake reset signal by adding additional circuitry, in Figure 24 AFE connections schematic, resistor R24 is placed to pull up the pin, and capacitor C29 as filter. Additionally, the socket J8 for ADE_CF1, ADE_CF2 and ADE_CF3 is placed as a pulse output, since the power measures may be converted to frequency and measured from these outputs, and helping the calibration process. Additionally AFE IO voltage high level is 3.3 volts, the same as RX63N and Raspberry Pi, for other MCUs with other voltage levels, they should be translated to this level to avoid IO hardware damage.

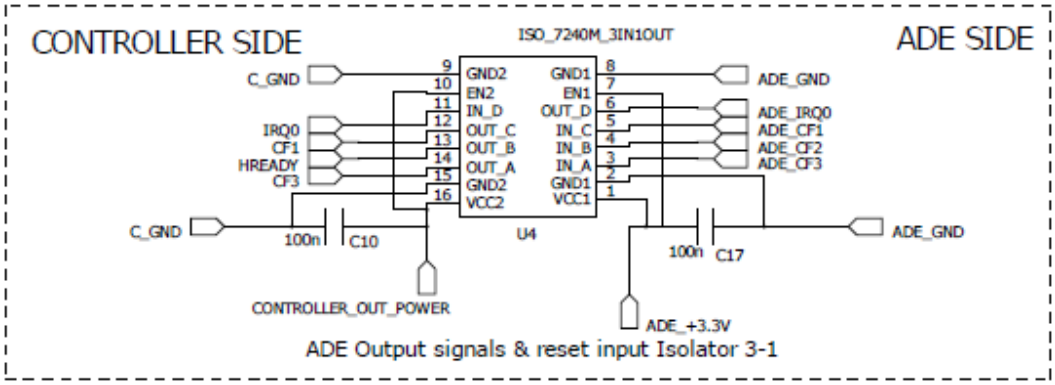


Figure 25 Fast acting optoIsolators, sample from Isolation circuits schematics

The four channels optoIsolators have a bandwidth of 25MHz while the selected SPI transmission speed is set to 2MHz although the maximum allowed by the ADE7880 is 2.5MHz. At this speed there were neither detected communications loss nor transmission errors. Figure 25 Fast acting optoIsolators, represents the schematics of one of the three in use, where both sides are clearly identified with their own voltages VCC1 and VCC2 and ground levels. Two of these Isolators are bidirectional, having three and one channel in each direction, selected due to the characteristics of the SPI protocol having signal select (SS), MOSI and clock as MCU outputs and MISO as input, as well as other input and output signals.

The 40 pin header is compatible with the new Raspberry Pi 2 and older Raspberry Pi 2 B, being always backwards compatible with the help of an adapter, needed as well to connect to the YRDKRX63N board.

The PCB board provides a standard female micro-usb type B jack connector as input power, and requires only one external power supply of 5V, of at least 500mA. The integrated voltage regulator provides an isolated GND and an output of 3.3V up to 300mA although the AFE requires only 28mA working in normal mode.

Traces are of two different widths, power related traces width is 0.500mm and all other traces are 0.250mm. Clearance rules allow a minimum separation of 0.250mm, the maximum angle for the traces is 45 degrees and pads include thermals where required.

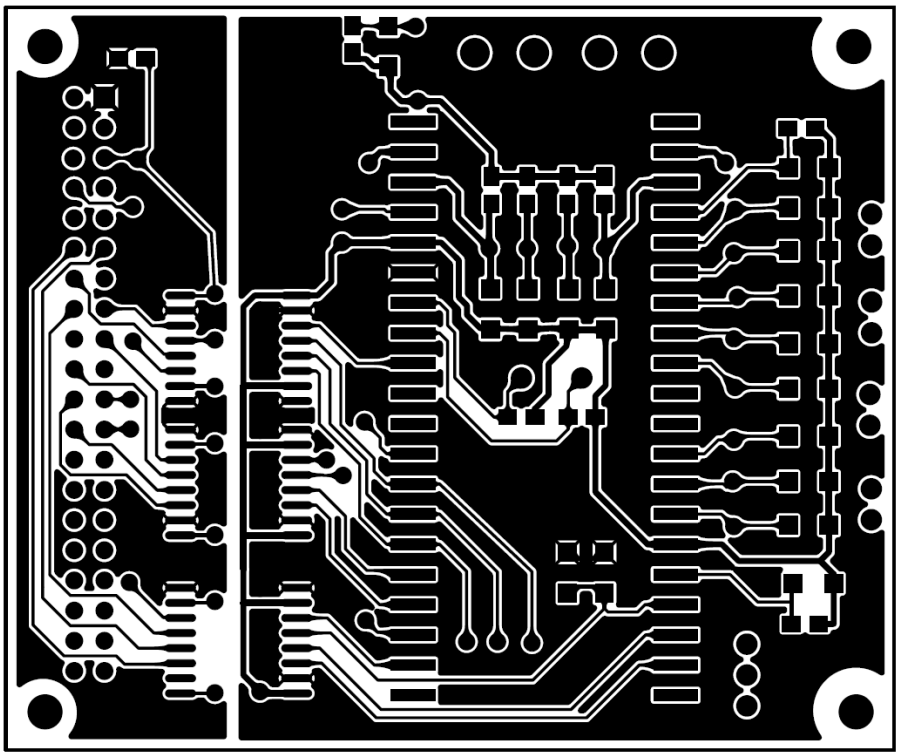


Figure 26 PCB Top layer

In Figure 26 PCB Top layer, the separated power planes are easily seen, with the MCU power plane at the left and the bigger ADE7880 power plane to the right side, separated under the fast acting optoIsolators. The ADE7880 comes in a 40-lead lead frame chip scale package (LFCSP), and the PCB offers a 0.5mm Pitch 40 Pin DIP

SMD socket to mount, with the ADE7880 previously soldered to a 0.5mm Pitch 40 Pin (20x2) QFP/QFN to DIP adapter that takes the most of the space but provides flexibility, easing the task of replacing an energy meter. This socket is pin to pin compatible with other energy meters, ADE7854, ADE7858, ADE7868 and ADE7878 from the same manufacturer.

5 ANALYSIS AND RESULTS

Measuring the effectiveness of the implementation is achieved by several processes. The goal is to measure the SPI speed and signal quality, the single measure time, the grouped measured time and the total window frame time, from the event is triggered to the moment the last byte of information is placed in the output UART register buffer. The testing environment generated wave is seen as well.

5.1 ADE7880 driver and UART driver performance

The SPI transmission is seen with the help of the digital analyser of a DSOX2012A oscilloscope where the speed of a single transmission, only one register reading from the AFE, and a group of them, seven registers in total, which are the total current, three harmonics and three phases current, are shown. A single register reading always involve three bytes of overhead and two or four bytes of payload depending on the targeted register. Measurements are stored in four bytes registers so a single SPI transmission always requires 7 bytes. The payload is always what the slave device places in the MISO line, in this case in the next clock cycle after the last bit of the MOSI has shifted in. the MOSI line carries the overhead bytes, consisting of one byte with the type of operation, read or write, and two bytes with the target register address. The next scope shows this sequence, where D_0, in red, is the Channel Select line which enables the slave device, the AFE, SPI transmission, and is active low. Above this, D_1 shows the SPI transmission clock, set to 2MHz, where from each bit can be counted in each byte. D_2 is the MOSI line, and D_3 is the MISO.

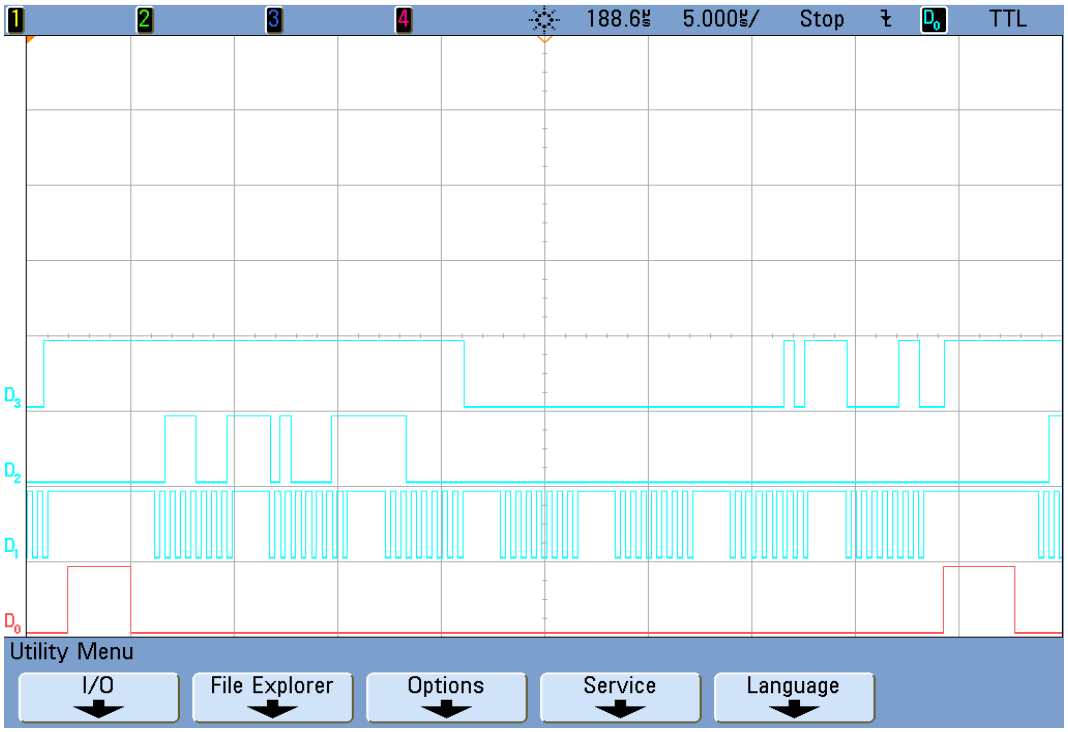


Figure 27 Single SPI reading scope by DSOX2012A

The transmission endianness is MSB first, one may read the transmission starting from left to the right as a human reads a number. As the transmission starts by CS shifting from high lo low, followed by the SCLK signal and the first MOSI byte, we start analysing the MOSI line, D_2. A look to D_1, the SCLK signal one may count the 7 bytes easily thanks to the delay introduced between each byte. The master sets data on the MOSI line starting with the first high-to-low transition of SCLK, and the SPI of the ADE7880 samples data on the low-to-high transitions of SCLK. The first byte in D_2, from the left to the right, after CS goes from high to low, indicates the operation type, bit 0, which is a 1, means a read operation, as the datasheet specifies, 1 for read operations and 0 for writing. The total bit sequence is 01110001b which corresponds to 0x71, this is because the device ID is assigned to be 0x70 in case of having multiple AFE and using I2C protocol. Notice that it is recommended to avoid using this combination, upper seven bits as 0111000b, to avoid confusion when not using I^2C protocol, although it is not relevant while only using SPI in 4 wire mode where CS pin exclusively targets one destination. The

next two bytes are the target register and they content is 01000011b and 11000000b which in hexadecimal notation is 0x43C0, and, as the datasheet indicates, refers to the AIRMS memory register, referring to phase A, current RMS value.

Immediately after the last address bit has shifted into the AFE, the MISO line starts transmitting back to the MCU when the next SCLK high-to-low transition occurs. While the first two bytes contain only 0, the next two are 00000101b and 00000110b representing 0x00000506, which in decimal notation represents the number 1286. The datasheet, in the registers list, specifies that this register is a 24-bit signed or unsigned register that is transmitted as a 32-bit word with four or eight MSBs, respectively, padded with 0s. And the Current RMS Calculation section specifies that this is a 24-bit signed registers accessed as 32-bit registers with the eight MSBs padded with 0s. The same section specifies that at the full-scale input signal of 0.5V, the ADCs produce an output code of approximately ±5,326,737, where the equivalent RMS is 3,766,572 (0x39792C) when the integrator is not in use and the gains are set to 0. Using the proportionality equation throws a result of 171µV at the AFE input which now have to be converted to the real RMS current by knowing the properties of the CT in use to interface the power line, 30A input, 1V output giving a result of 5.13mA flowing in the power line at the moment of the measurement.

In Figure 27 Single SPI reading scope by DSOX2012A one may see that the time for the SPI transmission required was, approximately 39µs, almost 8 divisions at 5µs/div. While the SPI clock is 2MHz and theoretically, 7 bytes at this speed, should be transferred in 28µs, what we see is the delay caused for the reload of the SPI registers whenever a byte is shifted in. This process time, if critical, may be reduced by using the whole SPI register space, where RX63N offers 4 long word registers of 32 bits, instead of shifting byte by byte, which is the simplest implementation but not the best in terms of performance.

One complete single cycle set of SPI requests, measuring the currents and harmonics, seven values once, can be seen in the next scope, where the total time for this cycle can be calculated as approximately 290µs, almost 6 divisions at 50µs/div.

Although compressed in the picture, the clock signal D_1 follows the same byte pattern, showing the MOSI and MISO bytes of each measure. The signal in CS D_0 shows the transitions from high to low whenever a new AFE register is retrieved. The slight delay that can be seen in between each low period is the transition time that the MCU spends in returning from the call to the ADE7880 driver to obtain a value to the next call to obtain another value, which is a fraction of each time division.

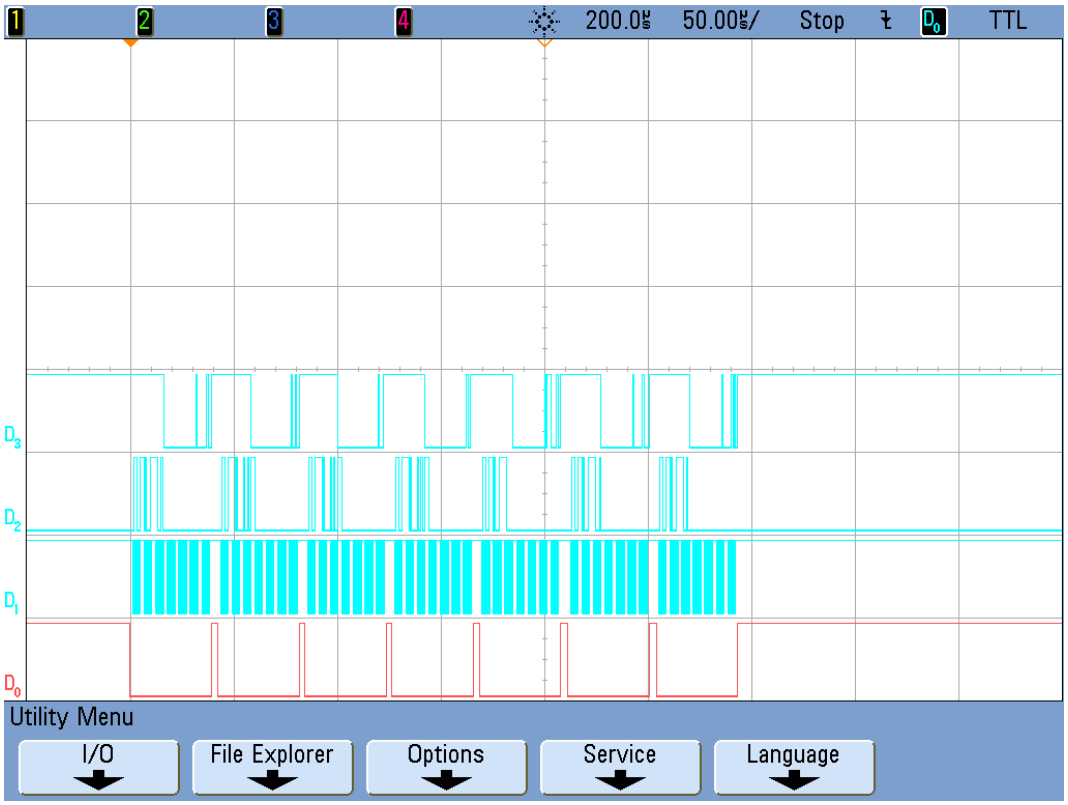


Figure 28 Grouped SPI readings in one cycle, scope by DSOX2012A

This scope Figure 28 Grouped SPI readings in one cycle, scope by DSOX2012Amay determine the effectivity of the implemented ADE7880 driver where the time that requires to access to a register, return and access again to another register in, both read, operations take less than 10µs.

To evaluate the performance of the communications to a RTU, the transmission buffer and UART communications, a timer counter is enabled to measure different actions' time of execution. By one hand, a first scope, let's call it 't1', where the

performance of the driver is measured by counting the time of one grouped SPI transmission, this serves to confirm, by another method, what so far was seen in Figure 27 Single SPI reading scope by DSOX2012A. Another scope, ‘t2’, is set to measure the time of the same grouped transmission until the UART driver returns after placing the last byte in the transmissions buffer, measuring in this way the time that takes to measure and make the information available in a RTU. A last scope, ‘t3’, is set from the event detection until the UART driver call returns after placing the last byte in the transmission buffer. This means the total time of the ten grouped transmissions, from the event triggering action, to the moment the last requested value of the last measurement cycle is placed in the output transmission buffer is shown.

The conditions are, for each test case ‘t’, as follows. Measuring ‘t1’, the fifth measurements cycle out of ten was randomly chosen. The time counter starts measuring when the measurements cycle function is called and ends after the last call to the ADE7880 driver has returned. To measure ‘t2’, the first cycle, right after an event is detected, is chosen to avoid any data from previous measurements in the UART transmission buffer that could have an influence in its performance. The third case, ‘t3’ measures from the moment an event is detected until the call to transmit measures to RTU function has returned. The LCD shows the results although the write operation is performed at the end of the whole process to avoid computing its processing time.

The additional MCU’s 16bit timer is configured as an up counter using the peripheral clock (PCLK 48MHz) prescaled by 8, giving a precision of 1/6 µs per clock period, counting up to 60000 clock cycles, triggering an interrupt every 10ms that, when serviced, adds one to the global variable that holds the number of timer interrupts counter. The values are shown in Figure 29 LCD showing the where ‘t1’, ‘t2’ and ‘t3’ match the scopes described above and their values are execution time in µs of each scope.

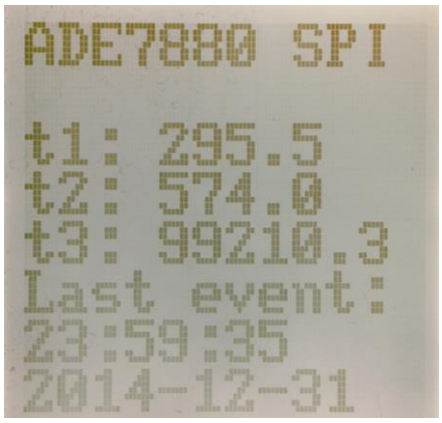


Figure 29 LCD showing the execution times of cases ‘t1’, ‘t2’ and ‘t3’

The first line displaying ‘t1= 295.5µs’ confirms the oscilloscope’s data, having additional µs, that may come from the extra processing clock cycles required by the calculations. Being ‘t2=574.0µs’ the most significant figure, showing that the time of 0.574ms to serve the seven measured values to an RTU. In this implementation means that there are 9.4ms available to evaluate the data and execute the desired action before the next avalanche of measurements is executed and transferred. The next row shows ‘t3=99210.3µs’ scope displaying the total time in µs of ten cycles of measuring and transmitting, corresponding to the chosen window frame of 100ms, measuring 10 times starting from time 0, plus the transmission time of all the measures and extra formatting characters. Although the requirements specify an available frame as long as 150ms and to effectively measure the harmonics at least once, this solution provides with ten times more information in 2/3 of the time. If only one measure is accounted, the time is reduced by 1/100. Despite complying with the requirements, the gap of 0.2ms between the data is gathered by SPI in approximately 0.3ms and served to the RTU in 0.5ms may be reduced significantly as later is explained in further improvements. The analysis of the data over 200 events offered the next statistical results:

Sample size n = 200

Frequency table for t1:

x_k	f_k	$\overline{f_k}$

295.5	200	1

Frequency table for t2:

x_l	f_l	$\overline{f_l}$
574.0	200	1

Frequency table for t3:

x_m	f_m	$\overline{f_m}$
99210.3	1	0.005
98338.8	190	0.95
98338.3	1	0.005
97150.7	1	0.005
97099.3	2	0.01
97098.7	3	0.015
97098.2	1	0.005
97096.5	1	0.005

Where

x_z is the list of observed values

f_z is the number of times the value is observed

$\overline{f_z} = \frac{f_m}{n}$ is the relative frequency

The mean value for t1:

$$\mu(\,t1\,) = \sum_k \overline{f_k} \cdot x_k = 1 \cdot 295.5 = 295.5ms$$

The mean value for t2:

$$\mu(\,t2) = \sum_l \overline{f_l} \cdot x_l = 1 \cdot 574.0 = 574.0ms$$

The mean value for t3:

$$\mu(\,t3\,) = \sum_m \overline{f_m} \cdot x_m$$
$$= (0.005\ \cdot 99210.3) + (0.95 \cdot 98338.8) + (0.005 \cdot 98338.3)$$
$$+ (0.005 \cdot 97150.7) + (0.01 \cdot 97099.3) + (0.015 \cdot 97098.7)$$
$$+ (0.005 \cdot 97098.2) + (0.005 \cdot 97096.5) = \ 98293.8035ms$$

The variance for t1:

$$var(t1) = \sum_k \overline{f_k} \cdot (x_k - \mu(t1))^2 = 1 \cdot (295.5 - 295.5)^2 = 0$$

The variance for t2:

$$var(t2) = \sum_l \overline{f_l} \cdot (x_l - \mu(t2))^2 = 0$$

The variance for t3:

$$var(t3) = \sum_m \overline{f_m} \cdot (x_m - \mu(t3))^2$$
$$= (0.005\ \cdot (99210.3 - 98293.8035)^2)$$
$$+ (0.95 \cdot (98338.8 - 98293.8035)^2)$$
$$+ (0.005 \cdot (98338.3 - 98293.8035)^2)$$
$$+ (0.005 \cdot (97150.7 - 98293.8035)^2)$$
$$+ (0.01 \cdot (97099.3 - 98293.8035)^2)$$
$$+ (0.015 \cdot (97098.7 - 98293.8035)^2)$$
$$+ (0.005 \cdot (97098.2 - 98293.8035)^2)$$
$$+ (0.005 \cdot (97096.5 - 98293.8035)^2) = 62674.096$$

Standard deviation of t1:

$$\delta(t1) = \sqrt{var(t1)} = 0$$

Standard deviation of t2:

$$\delta(t2) = \sqrt{var(t2)} = 0$$

Standard deviation of t3:

$$\delta(t3) = \sqrt{var(t3)} = 250.348ms$$

This information reveals that the SPI transfers use exactly the same amount of CPU clock cycles for each transmission. This is the expected result due to the fact that this operation is mostly performed by the SPI peripheral and its memory registers, involving little CPU load. By the other hand reveals that whenever the UART transmission is involved to transmit a single set of measurements, performs as good as the SPI peripheral. Although when several sets of measurements are performed, case 't3', making higher use of the UART transmission buffer, the performance is slightly reduced, increasing the total time in 8ms, from the expected 90.5ms (the tenth and last measurements cycle is triggered after 90ms of the event, expecting to add no more than 0.5ms of a single transmission) up to 98.3ms.

To confirm the delay added by the UART transmission buffer, the same test is performed but 't1' case is executed in the second cycle of measurements execution, after 10ms of the event, and 't2' case is performed during the fifth cycle, after 40ms of the event. The result seen in Figure 30 LCD, execution times when 't2' incurs in delay shows the UART transmission buffer as the responsible of the delay, requiring almost 1ms more in the same operation.

Figure 30 LCD, execution times when 't2' incurs in delay

To a further improvement, the ADC may trigger an interrupt whenever a reading is made available, instead of reading in polling mode. Its resultant value evaluated

inside the interrupt which deactivates itself in case of the detection of an event. As long as the ADC measures will not be needed again upon the end of the measurements frame, when this interrupt has to be enabled again. It is recommended to keep the ADC running to avoid its own registers settling time each time it is started. This interrupt has to relinquish the control to the MCU's process. UART transmission is interrupt driven as well and should have a higher priority, allowing the UART transmission buffer to progress when required. Furthermore one may use the available DMA channels to transfer the measured data directly into the UART transmission buffer, releasing the processor cycles required to perform the operation and speeding up the process. This is not the case and these optimizations are not implemented being the processing time fast enough to the purpose of this research.

5.2 Testing environment, ADE7880 driver and UART driver accuracy

A set of signals are produced and measured by the energy meter, its data transmitted and displayed in the terminal console, while an oscilloscope, connected to the soundcard output gives the plot of the wave and shows the resultant FFT in the same scope.

The test conditions are, in all the cases, the mentioned signals interfaced to the *INP* and *INN* AFE's input pins and the same signal to the *VAP* and *VAN* input pins to serve as the base signal for the DSP. Nothing is connected to the phases' input. Console scopes show '*idx*' as the index of the of each set of measurements occurring every 10ms, followed by the seven measured values, *IN TOTAL* as neutral line total current, *HX*, *HY* and *HZ RMS* as the *RMS* values of the three harmonics indexes and *Ph_A*, *Ph_B* and *Ph_C IRMS* as the *RMS* current of the three phases, these latest are not connected for this test as only current and harmonics over the neutral are required. Oscilloscope scopes show the analog signal in probe no.1 in a yellow trace and its related FFT in magenta colour.

The test case shows the result of a measurement where the amplitudes of the frequencies a, b, c and d are set to the same dimensionless value of 0.5. This evidences the necessity of amplitudes calibration by showing slightly different results, again, probably due to hardware limitations, in this case, in the lower frequencies range as

these present a slight attenuation. The selected configuration simulates a total RMS current of 7.73 A, where the 3rd harmonic has a value of 0.55322, the 5th 0.74390 and the seventh 0.75348 as seen in Figure 31 Currents when input of a, b, c and d frequencies amplitude is 0.5.

idx	N TOTAL	HX IRMS	HY IRMS	HZ IRMS	Ph_A IRMS	Ph_B IRMS	Ph_C IRMS
1	7.732870	0.55322	0.74390	0.75348	0.00412	0.00655	0.00625
2	7.732870	0.55322	0.74390	0.75348	0.00412	0.00655	0.00625

Figure 31 Currents when input of a, b, c and d frequencies amplitude is 0.5

The next Figure 32 Scope of the test signal, four frequencies with an amplitude of 0.5 shows the plot of the signal and gives the values of the produced voltage RMS that simulated the current flowing through a Rogovski Coil or a current transformer. The collage made at the left of the picture shows two columns. The left one shows the data from the cursors. Cursors *X1-Y1* are placed over the fundamental at 50Hz and *X2-Y2* over *HZ* at 350Hz, shows ΔX of 300Hz as expected between both frequencies. ΔY reveals the attenuation of 4.375dB at the lower frequency with respect to the higher. The right column shows measures instead of cursors belonging to the same scope, where Freq(1) proves that the frequencies are correctly generated with a fundamental at 50Hz. Regarding the amplitudes, the FFT shows that the value of the highest frequency *Max(M)* is 17.4dBv.

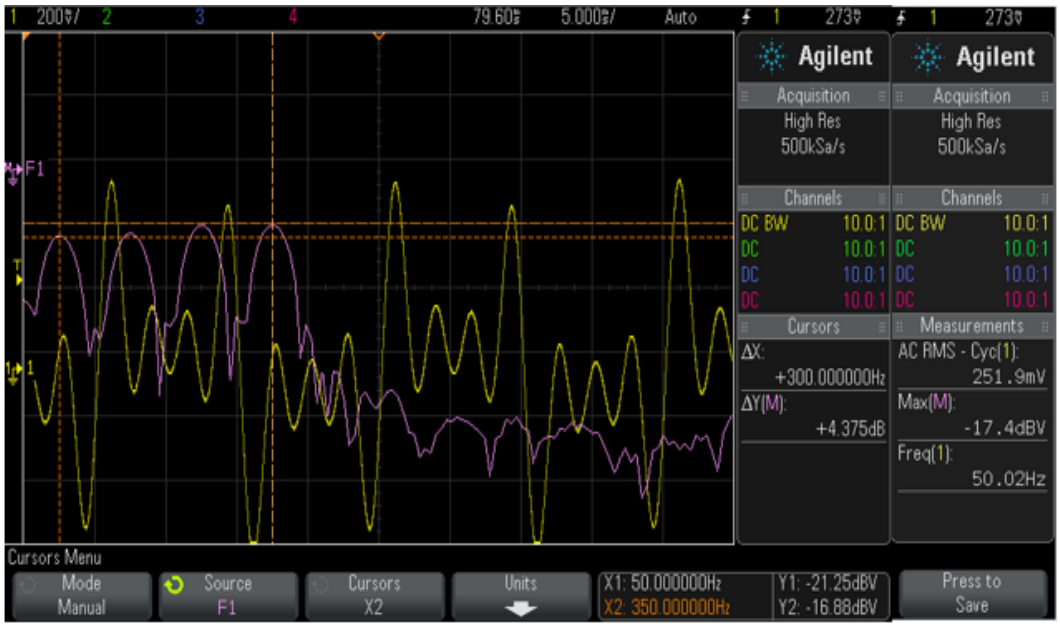


Figure 32 Scope of the test signal, four frequencies with an amplitude of 0.5

As there is no calibration performed during this test, the analysis is performed taking the total current as the reference for the value of the harmonics.

Conversion of total current to RMS:

$$\frac{7.733A}{\sqrt{2}} = 5.468A_{RMS}$$

Conversion of *Max(M)* in dBv to a scalar magnitude:

$$20log_{10}(G) = -17.4dBv$$

$$G = 10^{\frac{-17.4}{20}} = 0.135$$

Current RMS of the 7th harmonic component at 350Hz:

$$5.468x0.135 = 0.738A_{rms}$$

Current at 350Hz as the 7th harmonic shown by the energy meter is $0.753A_{rms}$ and analysis of the signal produced by the testing environment without calibration shows $0.738A_{rms}$. The error found is the 2% while the datasheet confirms that the maximum error is of 1% in a dynamic range of 1:1000. Although MATLAB is

producing the expected result, the linked soundcard hardware was not the ideal environment, having the software executed in a remote machine with a different hardware adding a delay to the response time. Moreover it is difficult to determine if errors are introduced by one or another as in the case of the frequencies, the span more than 50Hz having a high Q-factor, meaning that the frequency range of ±50Hz of the centre frequency has elevated energy when what it is needed is higher energy concentration in the frequencies of interest, this may be due to a probable output bandwidth limitation in the lower frequencies of the audio range of 20Hz to 20KHz. However, the use of a software solution to implement the testing environment provides portability and a quick adaptability to new test conditions.

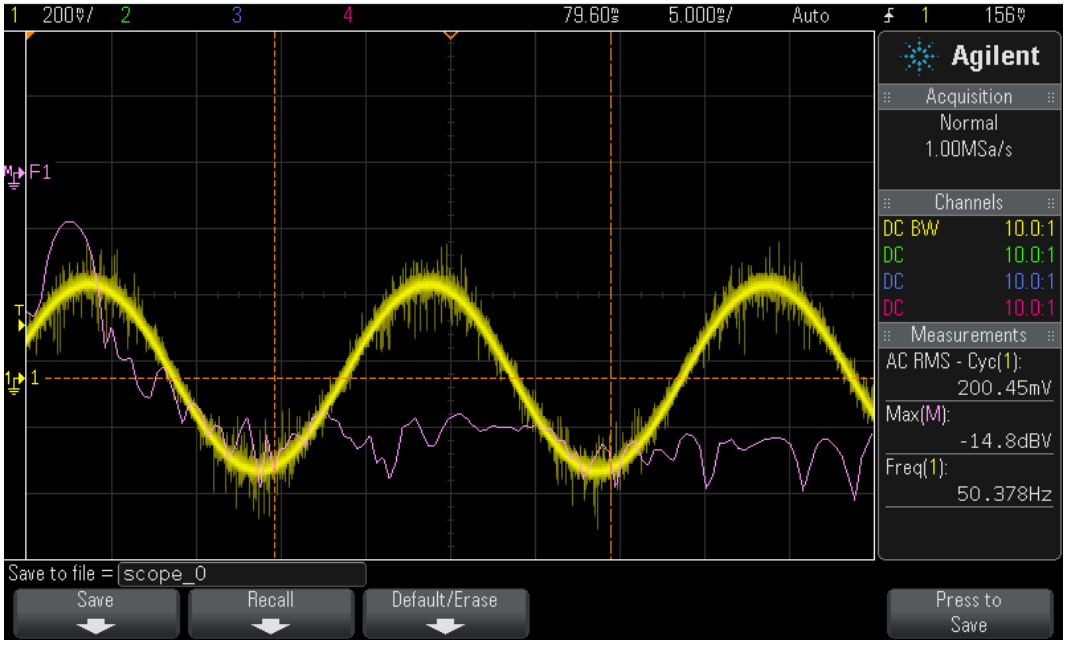


Figure 33 Not filtered testing signal

The scope above shows the output of the testing environment signal, only fundamental is generated, before the low pass antialiasing filter and in normal resolution acquisition mode. The noise is evident, and part of the energy is carried by it. The FFT shows the same property of a poor Q-factor as another cause of energy loss in the neighbour frequencies.

5.3 A word about MCU and ADE7880 AFE vs. MCU ADCs performance

The use of the ADE7880 offers the advantages already seen but at the expense of a higher cost. Moreover, regarding to frequency component analysis, its output is limited to the fundamental and its multiples due to the use of FFT in its DSP computational block. The next lines are intended to give a brief theoretical comparison of an AFE by means of an energy meter versus a MCU to directly interface the power lines.

The MCU interfacing the power lines through CTs and voltage dividers by means of its embedded ADCs represents a cheaper solution, and, when requiring to analyse specific frequencies other than harmonics of 45Hz-65HZ, the energy meter is not an option anymore. Another advantage is the speed at the data is served, interfacing the internal ADCs with a DMA channel provides much faster transmission speed than the energy meter SPI. On the other hand, other requirements have to be accounted in this option. The selected MCU embedded ADCs have to comply with the prerequisites for this task, precision, accuracy and speed, because if one has to interface external dedicated ADCs, the cost raises, been higher that a single energy meter. When the MCU is selected, the required front end circuitry has to be designed accordingly. The development process may be longer since more mathematical calculations are involved having a higher risk of error and buggy code.

Regarding to the processing time, most of the actual MCU come with an integrated DSP and public software libraries which will speed up the process, otherwise one may implement the analysis based in the theory presented in chapter 2.3 of this research. At this point, using an FFT library, which they usually implement an optimized version of the known algorithms, like Cooley and Turkey's algorithm, one is still limited by the number of points in the transform which is of the power of two number that defines the number indexes, and the frequencies under scope may lay in these indexes. The other approach, whenever only a few values and not a full FFT set of magnitudes is needed, and/or a frequency other than the harmonics is involved, is to find suitable sampling frequency, decide the indexes number, one may construct a set of constant cosine lookup tables (only the real part is needed),

and, by means of a timer triggering an interrupt at this given fundamental frequency, keep track of the index that, whenever it matches an index multiple of a frequency under scope, implement an algorithm that performs the Discrete Fourier Transform calculation shown in the theory chapter 2.4.1, to the value given by the ADC and obtain the magnitude of the frequency at that index.

According to RX63N, and the RX DSP library documentation, its FFT algorithm may perform a complete Discrete Fourier Transform of real-valued input array in 151110 clock cycles without error checking, and 175483 with error checking (RENESAS, n.d.). Analysing the worst scenario, from the detection of an event until a value is stored into the MCU memory, the time required, taking 50Hz as the base frequency, is 20ms to sample the data and another 1.83ms required by the DSP library with the MCU working at 96MHz, its maximum clock frequency. At this point 21.83ms are spent in the process. The process of data retrieval and storage may take additional CPU clock cycles that are not evaluated but accounting them by rounding up to 22ms. Whereas this is still in the range of 150ms provided by the specifications, representing a valid solution. Additional optimizations may be applied as the signal's fundamental is a periodic and odd function, the second half of the period is the same, but inverted, as the first half, therefore, discarding it, the sampling time may substantially be reduced to the half, 10ms, making a transform in 12ms.

As stated, this approach is theoretical and not a definite statement, no test was practiced, only tries to show a preview of a comparison of interfacing the power lines with a MCU interfacing an energy meter dedicated DSP versus directly with MCU. The comparison shows that this solution is valid for the purpose and at a lower cost[1]. If there exists a need to evaluate other frequencies than harmonics, the former is not a valid option, being this last, the choice. Although the drawbacks have to be accounted and the use of a DSP library has a learning curve. Moreover, there is a

[1] No research was procured,

greater need of knowing the theory or mathematical background behind it, furthermore, there is additional work to select the proper MCU and ADCs to keep the required accuracy and precision and other signal filtering study is required to properly interface these elements.

6 CONCLUSION

This paper has shown a method to solve the problem presented by VASPEC Oy by interfacing a power line with an ADE7880 energy meter as a front end and a Renesas RX63N microcontroller to retrieve and store in a short term the phase and neutral currents of the power lines fundamental (50Hz) and their 3^{rd}, 5^{th} and 7^{th} harmonic component values. The selected microcontroller facilitates the communications with the ADE7880 by using a SPI channel at 2Mbps and to a remote terminal unit by means of UART channel at 115200Kbps.

It is proved that the selected approach can effectively and precisely measure the fundamental and harmonics frequencies altogether with each phase RMS current and make the information available for processing in a RTU in 574 µs and repeat the measures every half a fundamental cycle until a selected timeframe is consumed or a counter measure action taken. However the measuring process can be dynamically adapted, for example until the tamper situation is finished or handled. The frequencies are generated by the testing environment provided with MATLAB® and SIMULINK®, and the RTU receives and displays the measurements triggered by the testing frequencies total current surpassing the selected threshold voltage. Although the use of Microsoft remote application server and MATLAB® in a remote session where both have a different hardware configuration is not recommendable, the related amplitudes require to be reconfigured and calibrated for each different hardware.

Other implantations were studied and, theoretically proved as effective as the one shown here, where the front end in use is provided by the same microcontroller, yet the solid and precise ADE7880 energy meter avoids further mathematical calculations simplifying the code and saving valuable processing time. However, the limitations of the ADE7880 regarding the neutral line harmonics engine and the use FFT to perform the harmonics analysis, represent a drawback whenever, i.e. other frequencies than multiples of the fundamental are involved, in which case, if the processing time is not an impediment, and MCU with adequate integrated ADCs is

a valid option, having a reduced cost while introducing a slightly longer developing time.

A further recommended research may be conducted by introducing an analysis of the phases and neutral line phase shifting, where the drift to capacitive or inductive shift offers richer information of the fault and the faulty line, helping to determine the origin and cause of the event.

REFERENCES

Analog Devices. (2014). ADE7880 Datasheet Rev. C. *Polyphase Multifunction Energy Metering IC with harmonic monitoring*.

Banks, K. (2002). *The Goertzel Algorithm*. Retrieved from embedded.com

Blaza, D., & Wilson, R. (2011). *Embedded Market Study*. Retrieved from Electronics Embedded Ecosystem: Embedded.com

Dobry, B. K. (1993). *Programming in C.* Hawaii: wiliki.eng.hawaii.edu .

Imrs, P. (2006). *TRANSIENT BASED EARTH FAULT LOCATION IN* . Espoo: Doctoral Dissertation.

Mäkelä, J. (2013). Fourier series. *Jarmo Mäkelä Lectures in Advanced Analysis*.

Mäkinen, S. (2014). *Toroidal Coil in Measuring Alternating Current at a Distance.* Retrieved from Scientific Research Publishing: http://www.scirp.org

Mani, H. (2013). *Analog Devices Products.* Retrieved from Application note AN-639 rev.A: www.analog.com

Maxim Integrated. (2003, 01 31). *MaximIntegrated*. Retrieved from Demystifying Delta-Sigma ADCs: http://www.maximintegrated.com/en/app-notes/index.mvp/id/1870

Moulin, E. (2003). *Measuring Harmonic energy with a solid-state energy meter.* Retrieved from METERING INTERNATIONAL issue 3-2003: www.metering.com

Renesas Electronics. (2014). *RX63N Group User's Hardwar Manual (Rev.1.80).* Retrieved from RENESAS 32-Bit MCU RX Family / RX600 series.

RENESAS. (n.d.). *RX DSP Library version 3.0 (CCRX) for High-performance Embedded Workshop (Application note R01AN1800ES0100 Rev.1.00).*

Smith, S. W. (1999). *The scientist and enginee's guide to Digital Signal Processing.* California: California Technical Publishing.

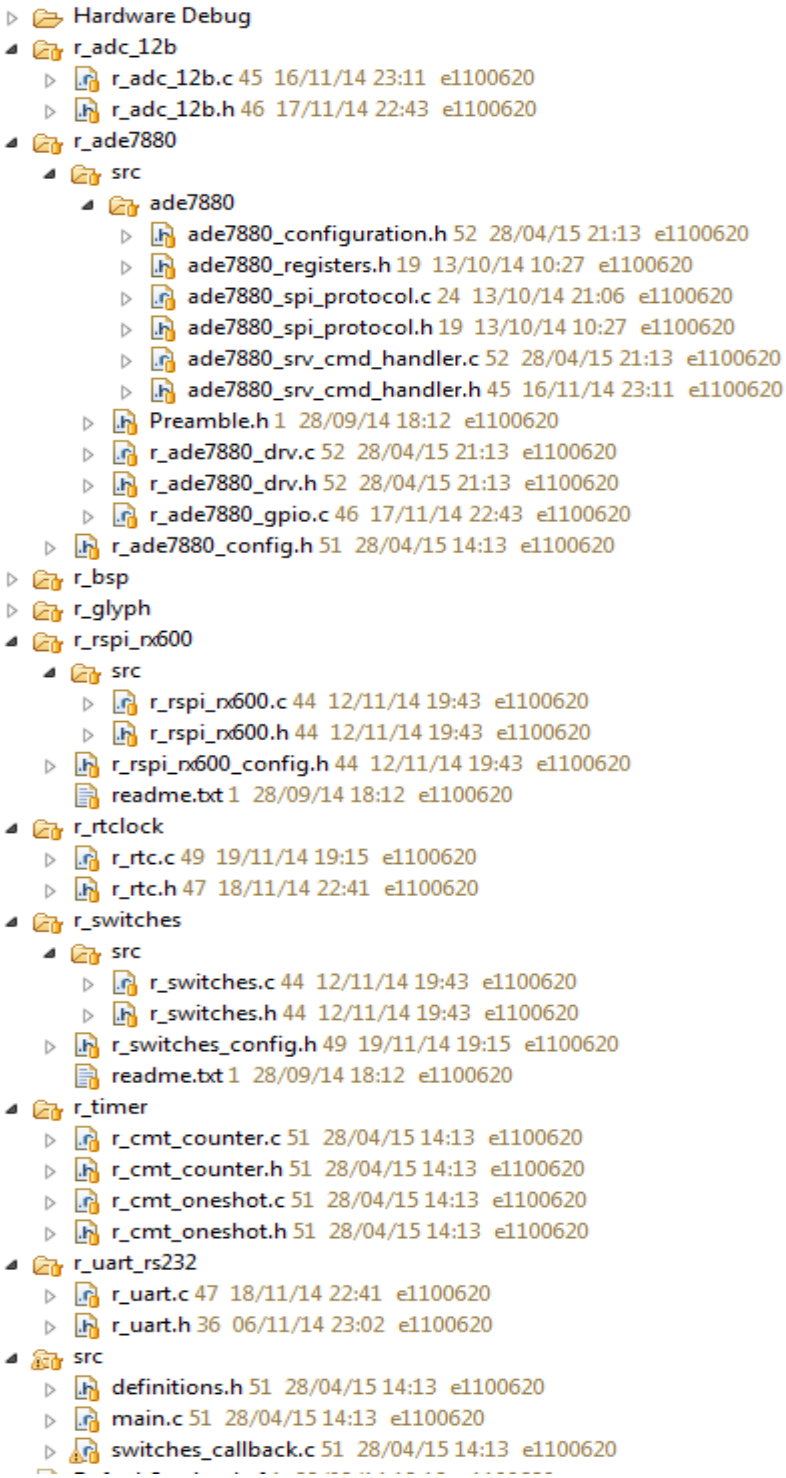
ToolsPractice, M. E. (2010). *IoT & Embedded Software Development.* Retrieved from What languages do you use to develop software?: http://blog.vdcresearch.com/embedded_sw/2010/09/what-languages-do-you-use-to-develop-software.html

William Koon, Analog Devices, Inc. (2001, 04 12). *Current sensing for energy metering.* Retrieved from analog.com media: http://www.analog.com/media/en/technical-documentation/technical-articles/16174506155607IIC_Paper.pdf

Yarborough, B. (2012, 01 06). *Components and Methods for Current Measurement.* Retrieved from Power Electronics: http://powerelectronics.com/power-electronics-systems/components-and-methods-current-measurement

APPENDICES

Appendix 1: Code structure and naming convention



```
/r_ade7880                                  (1)
|_r_ade7880_config.h                        (2)
|_/src                                      (3)
    |_ade_driver.h                          (4)
    |_ /ade7880                             (5)
            |_ade7880_registers             (6)
            |_ade7880_configuration         (7)
            |_ade_spi_protocol              (8)
            |_ade_src_cmd_handler           (9)
    |_r_ade7880_drv                         (10)
    |_r_ade_gpio                            (11)
```

1. Folder containing middleware, driver and HAL
2. Renesas SPI to ADE Hook configuration file: This file provides different definitions to serve as ADE7880 general standard configuration values for its two main different working modes, as an energy meter or as a harmonics meter. To understand what these values refer to, one should cross-check them with the datasheet. Any change here will directly affect to any ADE7880 measurement, for that reason is highly recommended to read the document before manipulating the values. The values here are standard to measure currents with a C.T. and one may use them as they are but, to a proper accuracy and performance, the device should be calibrated as mentioned in the datasheet.
3. Driver's source folder
4. Provides access to the driver from outside. To implement this driver in an application granting the access to the public functions, one should include the file "ade7880_driver.h" into the application requiring of it.
5. Driver and HAL folder
6. HAL definitions
7. Specific ADE7880 IC configuration definitions
8. Builds the transmission and Reception buffers and communicates via SPI by means of the function pointers received from the middleware layer
9. Provides public functions to interact with the IC.
10. Middleware driver layer to allow ADE7880 access to the SPI driver by means of callback functions passed as pointers to the driver
11. Middleware driver layer to allow ADE7880 driver access the GPIO pins

Appendix 2: Data container

```
/* ENERGY MEASUREMENT ATTRIBUTES: */
// Individual data
typedef struct
{
    float value;            // Measured Data
    uint64_t timestamp;     // Time stamp
}em_tval_st;


//Phase related data storage
typedef struct
{
    em_tval_st IRMS;    // Current RMS
    em_tval_st VRMS;    // Voltage RMS
    em_tval_st WH;      // Watts hour
    em_tval_st POWER;   // instantaneous power in Watts
} em_phase_t;


// Neutral line related data storage
typedef struct
{
    em_tval_st IN_IRMS;                         // Neutral current RMS value
    em_tval_st INSTANT_IN;                      // Instantaneous Neutral current
    // Neutral current Harmonics RMS values
    em_tval_st IN_HXIRMS;
    em_tval_st IN_HYIRMS;
    em_tval_st IN_HZIRMS;
    // Neutral current Harmonic distortion
    em_tval_st IN_HXIHD;
    em_tval_st IN_HYIHD;
    em_tval_st IN_HZIHD;
    // ISUM RMS values
    em_tval_st ISUM_HXVRMS;
    em_tval_st ISUM_HYVRMS;
    em_tval_st ISUM_HZVRMS;
    // ISUM Harmonics distortion
    em_tval_st ISUM_HXVHD;
    em_tval_st ISUM_HYVHD;
    em_tval_st ISUM_HZVHD;
} em_neutral_t;


// Global container
typedef struct
{
    em_phase_t phase_a;
    em_phase_t phase_b;
    em_phase_t phase_c;
    em_neutral_t neutral_line;
} em_data_t;
```

Appendix 3: Recommended approach for managing Harmonic Calculations with ADE7880

The recommended approach to manage the ADE7880 harmonic calculations is the following:

1. Follow boot up procedure and SPI selected as communications protocol
2. Set up Bit 2 (CF2DIS) in the CONFIG register. Set CF2DIS bit to 1 to use the CF2/HREADY pin to signal when the harmonic calculations have settled and are updated. The high to low transition of HREADY signal indicates when to read the harmonic registers. Use the burst reading mode to read the harmonic registers as it is the most efficient way to read them. This is done in the following way (1): -Set bit 2 of 'CONFIG' register, address 0xE618, as 1 so CONFIG = 0x0006 (CONFIG register is 16bits) -Write the register at least twice (in case you write the register individually, if sequentially, write twice only the last). -Read the last written register to check that it was a successful write.
3. Choose the harmonics to be monitored by setting HX, HY and HZ:
    a. Write 'HX' 0xEA08 (8bits) with the desired harmonic index (i.e. 1 for Fundamental)
    b. Write 'HY' 0xEA09 (8bits) as i.e. 5 to measure the 5th harmonic.
    c. Write 'HZ' 0xEA0A (8bits) as 7, 7th harmonic.
4. Select all the HCONFIG register bits.
5. Write 'HCONFIG' 0xE900 (16bit), i.e. with update rate of 1ms (HCONFIG = 0x0047)
6. Initialize the gain registers used in the calculations. Leave the offset registers to 0.
7. Write 'GAIN' 0xE60F 16bits register initially as 0.

Additionally, selecting the desired harmonics at each index is done here. One should pay attention that the EM Absolute Maximum Number of Harmonic indexes is 63 (actually 2800 / Fundamental line frequency). This is due to its properties in

which its Measurement Bandwidth (-3 dB) is 3.3KHz. Read the EM data-sheet for more information.

8. Select phase or neutral line to monitor. Options: NEUTRAL_LINE, PHASE_A, PHASE_B or PHASE_C.
9. Number of samples per reading: From one single reading to as many as desired and the average of the total readings is calculated in the following way: (sum up all the readings) / (number of readings)
10. Set CF2DIS bit in CONFIG to 1 to use CF2/HREADY pin to wait for a new calculation to be completed and its register has settle for HRCFG & HSTIME.
11. Set HRCFG bit in HCONFIG: 0(Default) = waits HSTIME to settle and triggers HREADY : 1 = Triggers HREADY immediately
12. Set HSTIME bits in HCONFIG: (Default) 01 = 750ms of settle time: 500ms (00): 1 sec (10): 1250 ms (11).
13. Set HRATE bits in HCONFIG:(Default) 000 = the update rate of the harmonic registers. 000 = 8kHz
14. Set ACTPHSEL bits in HCONFIG: (Default = 00 phase A) to select the phase voltage used as time base for harmonic calculations. The selected phase has to be connected to its input pins.

Regarding to the Integrator, in case of using a di/dt current sensor, the EM has its own integrator which can be activated.

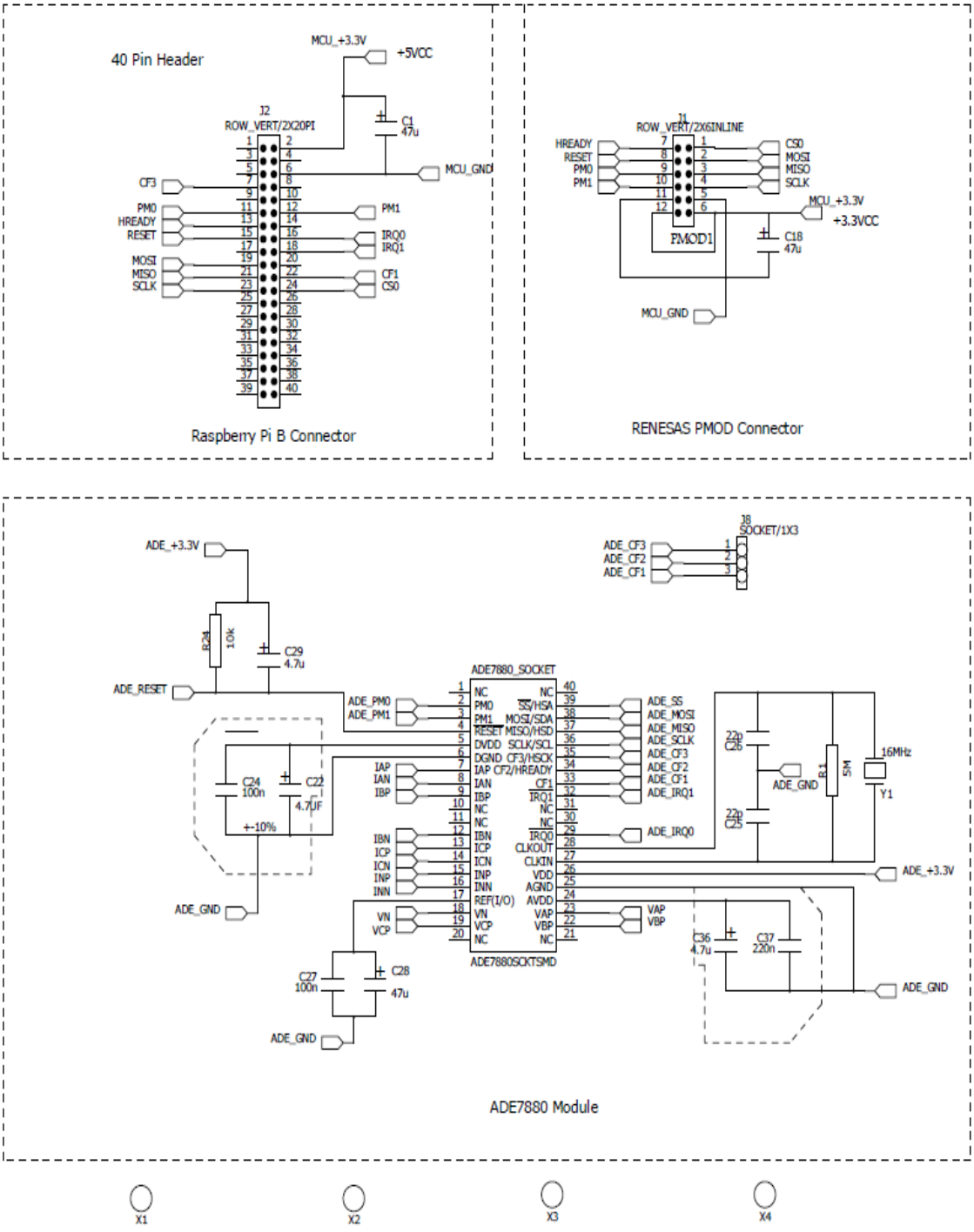
15. If Rogowski coils are used, enable the digital integrators in the phase and neutral currents: Bit 0 (INTEN) set to 1 in the CONFIG register. Initialize the DICOEFF register to 0xFF8000 before setting the INTEN bit in the CONFIG register.
16. Start the DSP by setting Run register to 1.
17. Read the registers in which the harmonic information is stored using the burst or regular reading mode at high to low transitions of CF2/HREADY pin.

18. CF2/HREADY pin can be programmed as an interrupt (read Datasheet Configuring Harmonic Calculations Update rate p.63-64) and its value is stored and can be read in 'STATUS0' register 0xE502 32bits.
19. For neutral line monitoring read to 'HXIRMS' 0xE889 (32bits), 'HYIRMS' 0xE891 (32bits) and 'HZIRMS' 0xE899 (32bits) (before proceeding with a reading one must follow the procedure to initialize and turn on the ADE7880 DSP)

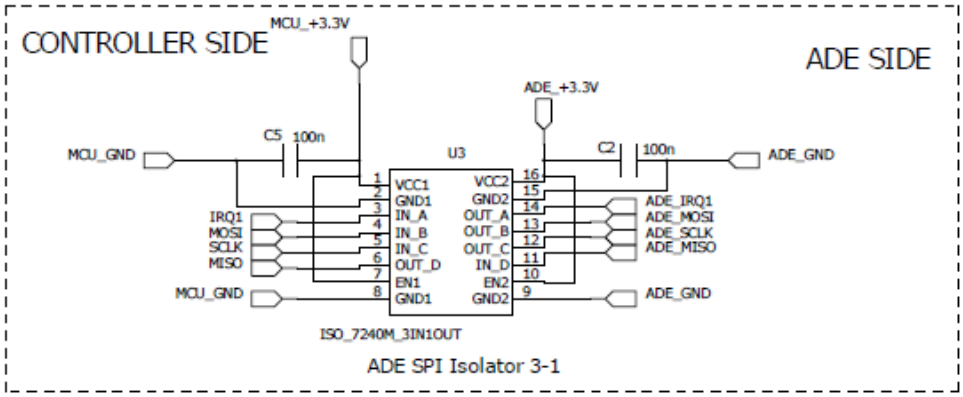
Appendix 4: SPI driver access from ADE7880 driver

Appendix 5: Schematics

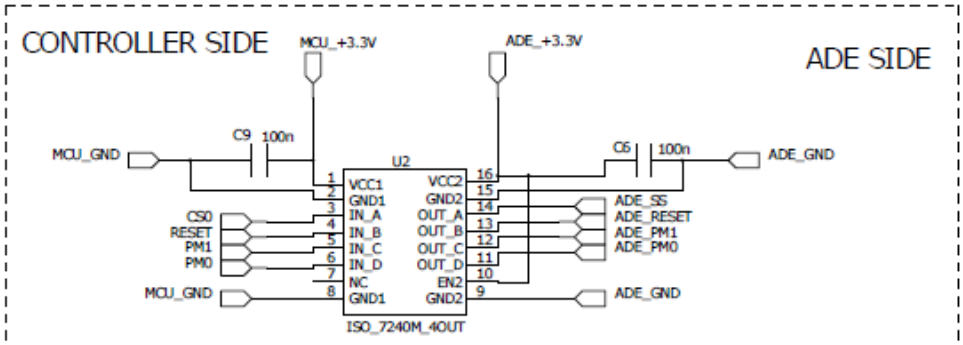
ADE7880 and connector socket



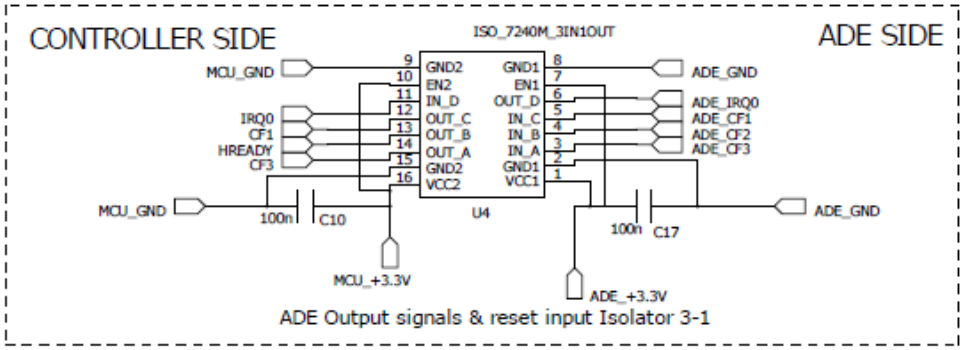
OptoIsolators



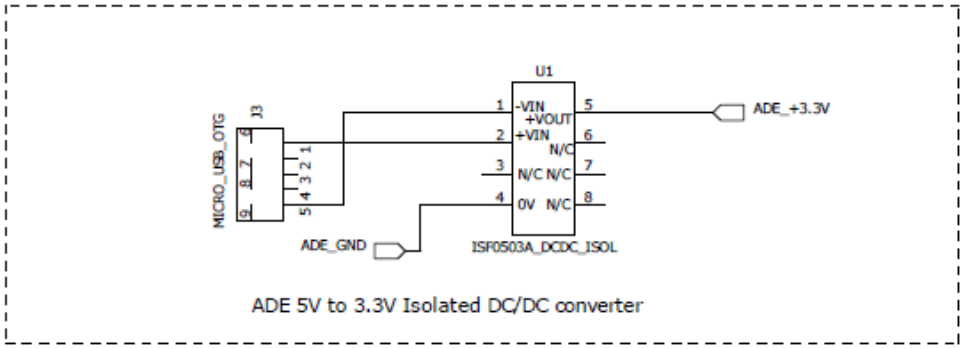
ADE SPI Isolator 3-1



Power Mode & IRQ signals Isolator



ADE Output signals & reset input Isolator 3-1



ADE 5V to 3.3V Isolated DC/DC converter

Appendix 6: PCB layout and silkscreens (real size 62.5mm x 7.45mm)

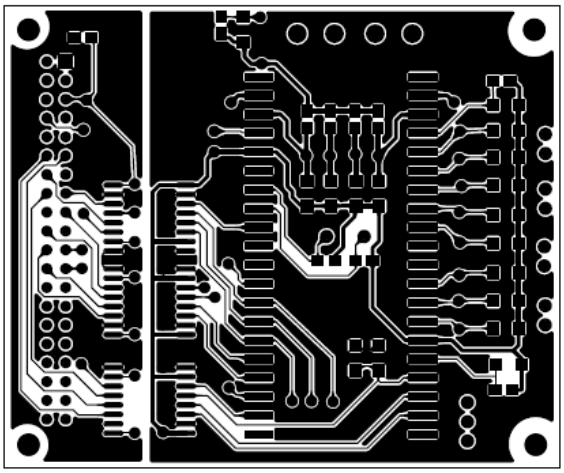


Figure 35 PCB Top layer

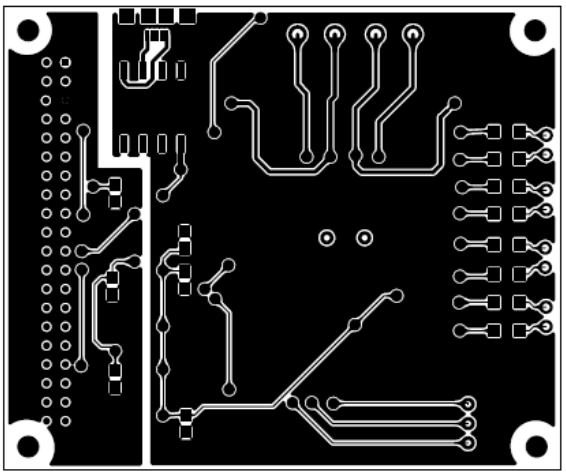


Figure 34 PCB Bottom layer

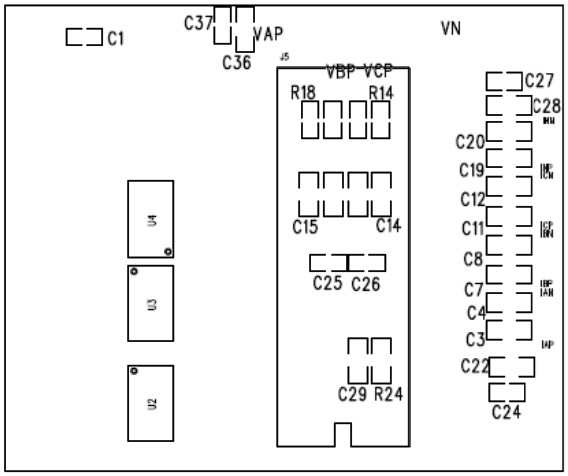


Figure 36 Silkscreen top layer

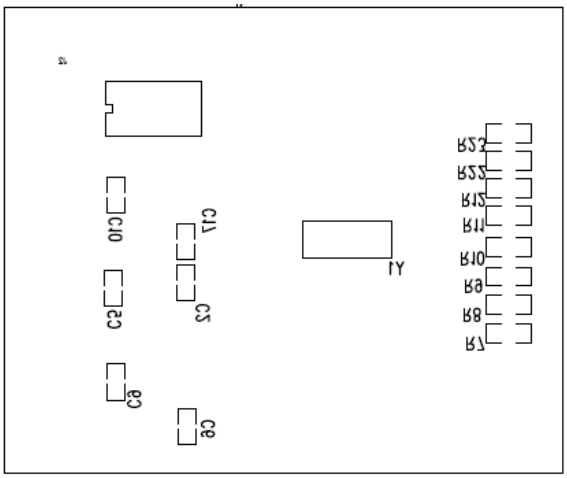


Figure 37 Silkscreen bottom layer

Appendix 7: Soundcard Datasheet

8. Analog Performance Characteristics

Standard test condition: $T_{ambient}$=25°C, Dvdd=5.0 or 3.3V ±5%,Avdd=5.0V±5%
Input Voltage Level: Logic Low=0.35*Vdd, Logic High=0.65Vdd
1KHz input sine wave; Sampling frequency=48KHz; 0dB=1Vrms
10KΩ/50pF load; Test bench characterization BW:20Hz~20KHz
0dB attenuation; tone and 3D disabled

Parameter	Minimum	Typical	Maximum	Units
Full scale input voltage				
Mixer (except for MIC)	-	1.6	-	Vrms
Mic input (gain=0dB)	-	1.6	-	Vrms
Mic input (gain=20dB)	-	0.16	-	Vrms
ADC	-	1.0	-	Vrms
Full scale output voltage				
Front DAC	-	1.5	-	Vrms
Front DAC (F version)	-	1.1	-	Vrms
Surround DAC, Center/LFE DAC	-	1.1	-	Vrms
S/N (A weighted)				
Analog Inputs to LINE_OUT	-	95	-	dB FSA
ADC	-	85	-	dB FSA
DAC (Front DAC with headphone amp)	-	85	-	dB FSA
DAC (Surround, Center, LFE DAC)	-	90	-	dB FSA
THD+N				
Analog Inputs to LINE_OUT	-	-85	-	dB FS
ADC	-	-80	-	dB FS
DAC (Front DAC with headphone amp)	-	-75	-	dB FS
DAC (Surround, Center, LFE DAC)	-	-80	-	dB FS
Frequency Response				
Mixers	10	-	22,000	Hz
ADC, DAC	16	-	19,200	Hz
Power Supply Rejection (DAC, ADC)	-	-68	-	dB
Total Out-of-Band Noise (28.8K~100KHz)	-	-63	-	dB
Mic 20dB gain is selected	18	20	22	dB
Crosstalk between inputs channels	-	-70		dB
Attenuation, Gain Step Size	-	1.5	-	dB
Input impedance (gain=0dB)				
MIC1, MIC2, PCBEEP, PHONE	-	16	-	KΩ
Others (LINE,CD,AUX,VIDEO)	-	32	-	KΩ
ADC	-	32	-	KΩ
Power Supply Current (normal operation)				
VA=5V / VD=3.3V	-	88 / 36	-	mA
VA=3.3V / VD=3.3V	-	71 / 36	-	
Power Supply Current (power down mode)				
VA=5V / VD=3.3V	-	0.5 / 10	-	mA
VA=3.3V / VD=3.3V	-	0.2 / 10	-	
Vrefout	2.25	2.50	2.75	V
Digital Filter Characteristics				
ADC Lowpass Filter				
Passband	0	-	19.2	KHz
Stopband	28.8	-	-	KHz
Stopband Rejection	-	-76.0	-	dB
Passband Frequency Response	-	+- 0.15	-	dB
DAC Lowpass Filter				
Passband	0	-	19.2	KHz
Stopband	28.8	-	-	KHz
Stopband Rejection	-	-78.5	-	dB
Passband Frequency Response	-	+- 0.15	-	dB