COMPACT ANALYZER FOR POWER EQUIPMENT - CAPE

Meher Jain
Vivek Sankaranarayanan

Final Project Report
ECEN 5613 Embedded System Design
April 30, 2016

Contents

1 Introduction

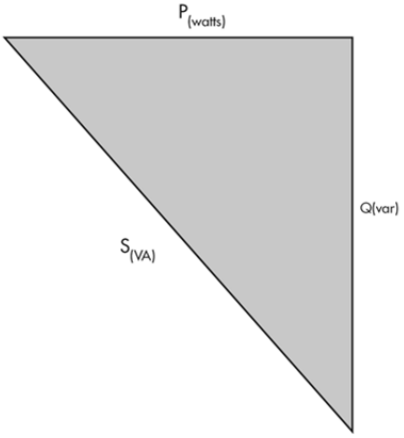
In this project an instrument to measure the power factor and harmonics of a load connected to grid was designed. The instrument is called CAPE (Compact Analyzer for Power Equipment).
Power factor and harmonics is an important design criterion for every device that draws power from the mains socket. As per [1], there are serious implications both for power generating utilities as well as the customer when a low power factor equipment is presented to the power grid. A few of them are [1]:

- Electric utilities must provision to deliver the peak voltage and current values instead of average power. A poor power factor can therefore lead to increased costs
- Grid stability is disrupted when high-order harmonics are injected into grid. There can also be heating and overload hazards for transmission equipment
- Increased electricity bills for user as electric utilities charge higher tariffs for a low power factor load

Keeping in mind the serious implications, there have been stringent legislations enacted around the world [1]. For example, the EU sets a limit to the 39th harmonic for equipment with input currents less that 16A per phase.

1.1 Power Factor, Total Harmonic Distortion (THD) and RMS value [2]

As per [2], Power factor is a figure of merit that measures how effectively energy is transmitted between a source and load network. It is defined in [2] as:



$$Power\ Factor = \frac{(average\ power)}{(rms\ voltage)(rms\ current)}$$

The power factor always has a value between 0 and 1.

As per [2], the ideal case, unity power factor, occurs for a load that obeys Ohm's Law. In this case, the voltage and current waveforms have the same shape, contain the same harmonic spectrum, and are in phase.

Figure 1.1 Power Factor [1]

Average Power is the net energy flown to the load over one cycle. It can be defined as:

$$P_{av} = \frac{1}{T}\int_0^T v(t)i(t)dt$$

Where $v(t)$ and $i(t)$ are instantaneous voltage and current values respectively. If $v(t)$ and $i(t)$ are periodic, then they may be expressed as Fourier series:

$$v(t) = V_0 + \sum_{n=1}^{\infty} Vn\cos(n\omega t - \varphi_n)$$

$$i(t) = I_0 + \sum_{n=1}^{\infty} In\cos(n\omega t - \theta_n)$$

The root mean squared (rms) value of a periodic waveform $v(t)$ with period T is defined as

$$(rms\ value) = \sqrt{\frac{1}{T}\int_0^T v^2(t)dt}$$

1.2 System overview

CAPE is a low cost instrument that can measure the power factor of an equipment connected to grid. It can also measure harmonics in voltage and current up to 1200Hz (fundamental * 20).

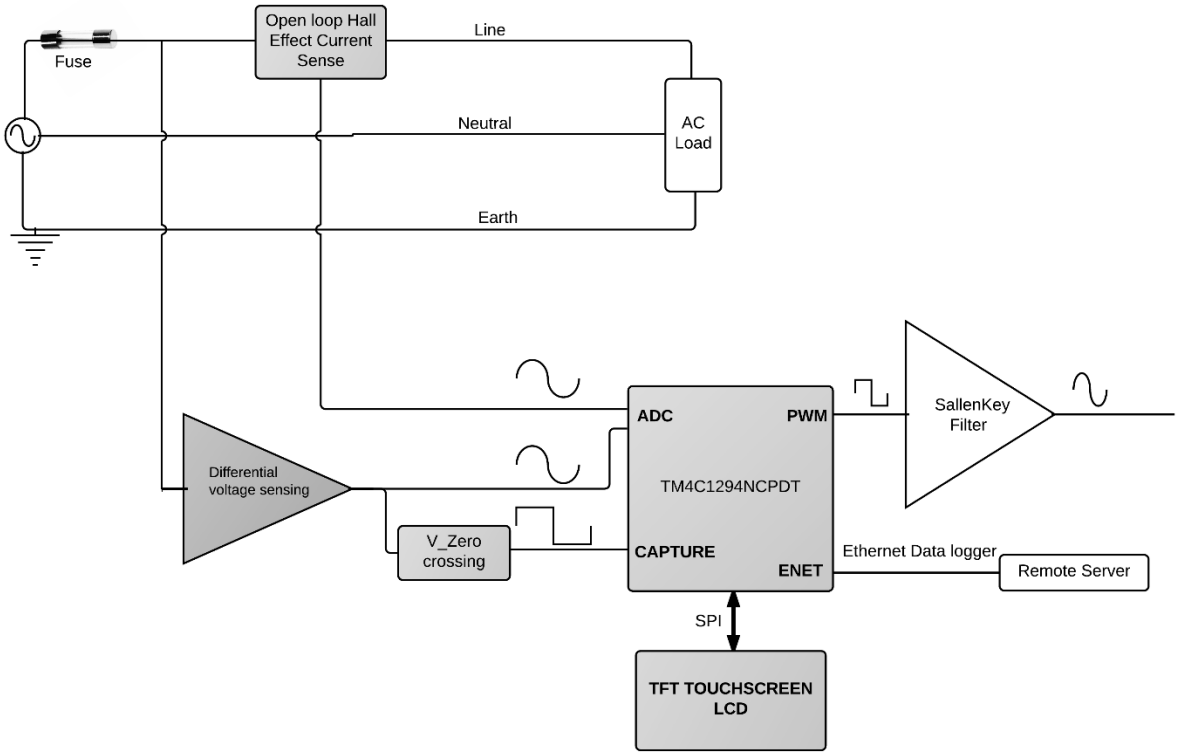


Figure 1.2 System Block Diagram

The system block diagram is shown in Figure 2. An ARM Cortex M4 processor (TM4C129NCPDT) from Texas Instruments is the CPU for the system. The line voltage is sensed using a differential sensing technique. For line current measurement, an open loop Hall-Effect sensor is used. A voltage zero crossing detector indicates changes in voltage polarity. A sallenkey Low Pass filter, a TFT touchscreen LCD to display waveform and metrics are the other major blocks in the system. CAPE also provides the capability to log data over the Ethernet port.

2 Technical description

The following sections detail the hardware and software design for CAPE.

2.1 Hardware Design

This sections describes the CAPE hardware and is divided into the following sections: MCU (outlines the Microcontroller selection), Sensors (describes Voltage and Current sensor, Zero crossing detector), Protection, Touchscreen and ENET interface. The whole project was built on perf-board. A PCB design was not preferred because of the time constraints in the project.

2.1.1 MCU

CAPE uses TM4C1294NCPDT as the main MCU. TM4C1294NCPDT offers a 32-bit, 120-MHz ARM Cortex-M4 processor with FPU. The key features of the chip are listed below. Refer to the [3] for the complete hardware description.

- **Core:** ARM Cortex-M4F, 120MHz, 150 DMIPS performance
- **Memory:** 1024KB flash, 256KB single-cycle SRAM, 6KB of EEPROM, Internal ROM
- **Communication Interface**8 UARTs, 4 SSI, 10 I2C, 2 CAN 2.0 A/B controllers, Ethernet MAC and PHY, USB
- **Analog:** 8 16 bit PWM outputs, 2 12bit ADC modules with 2MHz sampling rate
- **System:** 8 16/32 General purpose timers

The above processor was chosen for the application with a few things in mind:

- A high performance floating point processor for FFT and AC metrics computation
- Inbuilt 12bit ADC with a high sampling rate that allows us to take at least 256 samples in on line cycle
- High resolution Capture and PWM
- Ethernet module with both MAC and PHY layer
- Sufficient memory space and pins to interface with all the different components in the system
- JTAG debugging available

We used the Launchpad development board for the processor available here. The user guide for the Launchpad is available from here. The Launchpad brings out all the ports pins to the edge for easy access. It also has booster pack headers that is compatible with Kentec touchscreen display that we used.

2.1.2 Input/output connections

Input and output connections to line were made using 6.35mm quick connect male tabs. Link for the tabs here. 1 set of 3 tabs (Line, Neutral and Earth) were soldered on either end of the board. A standard AC Power Cord was cut into 2 halves and the cord with the male connector was soldered to the input tabs on the left. The cord with the female connector was soldered to the output tabs on the right. The corresponding input and the output tabs were shorted with a jumper wire. (The line jumper must pass through the current sensor before shorting the input and output tabs)

2.1.3 Sensing Circuitry

The sensing circuitry consists of 4 parts: Voltage sense, Opamp selection, Current sense and Zero crossing detector.

Voltage Sense Circuitry

Line voltage sensing is achieved by using differential sensing scheme. The differential voltage between line and neutral is divided using a resistor network and amplified using a differential opamp.

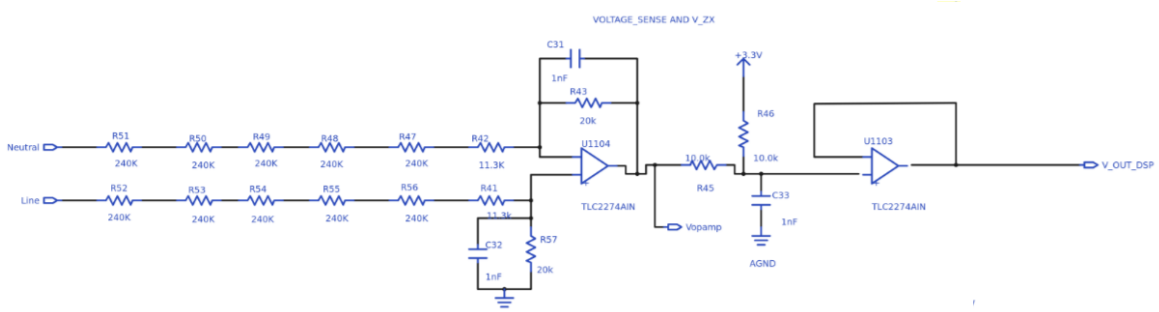


Figure 2.1 Voltage Sense Circuit

The resistor string at the input scales down the line voltages. All resistors used in the above circuit had 1% tolerance to minimize errors. The opamp then amplifies this differential signal to produce a bipolar output. The ADC of the MCU cannot take negative voltages therefore, we have to now level shift the signal such that the 0V is shifted to 1.65V (=VDDA/2. VDDA is Reference voltage to ADC. It has been set to 3.3V). The final opamp is used in a voltage follower configuration to improve the drive strength of the signal. The scaling calculations are shown below:

0 – 3.3V at the ADC input (V_OUT_DSP) corresponds to -3.3 to +3.3V at opamp output. This is as per the following equation:

$$V_{OUTDSP} = \frac{V_{OPAMP} + 3.3}{2}$$

Therefore, $$(V_{LINE} - V_{NEUTRAL})_{MAX} = V_{OPAMP} * \frac{240K*5+11.3K}{20K}$$

$$= 3.3 * 60.565V$$

$$= \sim 200V$$

Thus, a 0 - 3.3V on the MCU ADC pin corresponds to the change of -200V to +200V at the input.

Opamp selection

The opamp used is Texas Instruments' TLC2274AIN. Its datasheet is available at [6]. It was selected based on a few important parameters:

1. Available on Digikey
2. Throughole package
3. Bipolar (+-5V) and rail to rail operation
4. High Bandwidth (~2.2MHz) We need to measure harmonics till 1200Hz
5. Low input offset voltage ~300uV

The opamp is biased and decoupling capacitors connected as shown below:

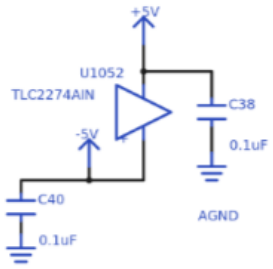


Figure 2.2 Opamp bipolar biasing and decoupling

Current Sense Circuitry

For current sensing, a non-contact technique was preferred. An open loop hall effect sensor was used. The part used was HO-6P. Link to datasheet [7]. Hall Effect sensing was preferred as opposed to resistive sensing because:

1. Easy to interface. Pass through wire sensing.
2. Very little additional circuitry required. Can be directly interfaced with MCU
3. Isolated sensing technique. MCU section is isolated from grid

The sensor was on line wire. Basically the jumper that shorts the input and the output line tabs passes through the sensor.

Hall Effect sensor does suffer from a few disadvantages. They have poorer accuracy (+- 1.35%) and are more non-linear (+-1%) compared to other techniques. A few key features are reproduced from the datasheet [7]:

- Primary nominal rms current (I_{PN}): 6A
- Primary current measuring range (I_{PM}): -20 to +20A
- Reference voltage: ~2.5V
- Output voltage range @ I_{PM} : -2 to +2V; Vout at 0A: +2.5V
- Frequency Bandwidth (-3dB): 250KHz; Accuracy @ I_{PN} (% of I_{PN}): +-1.35

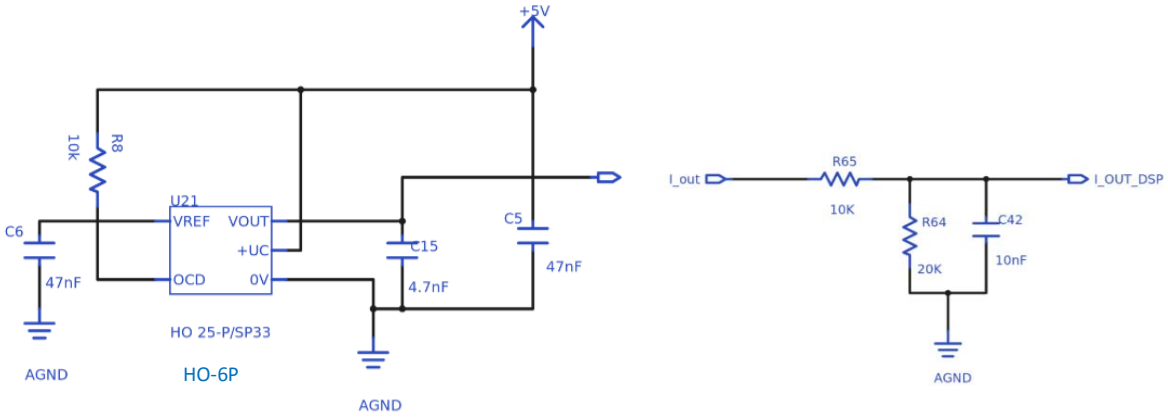


Figure 2.3 Current Sensing Circuitry [7]

The current sense output at 0A is equal to 2.5V i.e. the internal reference voltage of the sensor. For a full scale deviation of +-20A, the output varies from 0.5V (= 2.5 - 2) to 4.5V (= 2.5 + 2). We need to

scale this down since our ADC cannot accept voltages greater than VDDA (3.3V). Also at 0A we would like the input to the ADC to be centered at 1.65V instead of 2.5V. Hence we use a voltage divider set to ratio (2/3) to scale Vout to required value. Thus at 0A, the ADC sees 2.5V *2/3 = 1.6667V. Also the full scale seen by the ADC changes to (2-(-2) * 2/3 = 4*2/3 = 2.6667V.

Thus the full scale can be calculated as follows:
From the above paragraph: 2.6667V change on ADC corresponds to 40A change in current. Thus a 3.3V change on ADC corresponds to 49.5A (=40* 3.3/2.6667) change in current. Note than once again all resistors used are of 1% tolerance.

Zero Crossing detector

As mentioned earlier, CAPE has a zero crossing detector that indicates to the MCU when the voltage changes polarity. This indication is used by the MCU to keep track of a complete cycle and perform calculations such as rms measurements, phase difference, pll syncing etc. These algorithms will be covered in detail in the Software design section of this document. The basic zero crossing detector circuit is shown below:

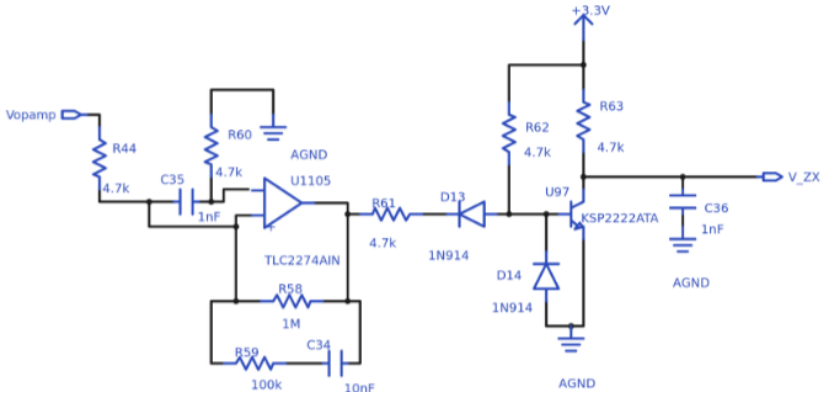


Figure 2.4 Voltage zero crossing detector

The above circuit works as follows: The first stage of the circuit is a comparator. The bipolar output of the differential voltage sensing opamp is connected to positive input. The negative input is connected to ground. Whenever, the input signal is positive (greater than ground) the output of the opamp saturates high. When the input goes negative (below ground), the output hits negative saturation. The positive feedback of the opamp is to provide hysteresis and prevent the opamp from switching state for noise transients.
The second stage of the zero crossing is a simple transistor switch that turns ON when output of the opamp is high. It is OFF otherwise. Thus the MCU sees a logic low in positive half cycle and a logic high in the negative cycle. The test waveforms have been produced in section 2.2.4.

2.1.4 SallenKey Filter

SallenKey filter is simply a 2nd order active low pass filter. This circuit is used to filter out the high frequency carrier frequency of the PWM and output the fundamental sine wave. This sine wave is locked with the grid voltage.

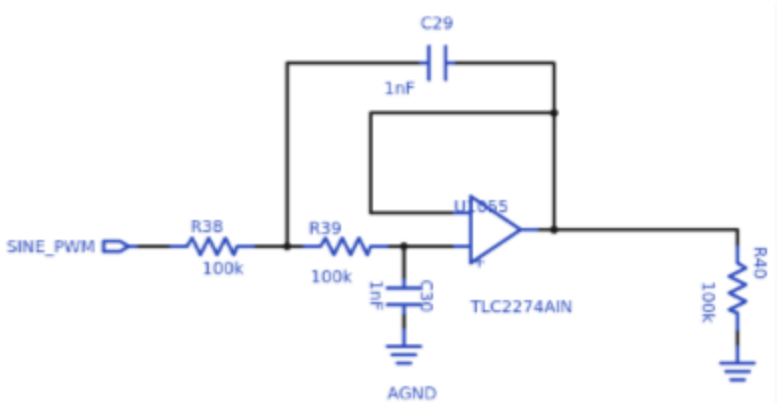


Figure 2.5 Sallenkey filter

The carrier frequency (switching frequency) of the PWM is 20KHz. The 2nd order filter gives -40db per decade from the cutoff frequency (Fc)

$$Fc = \frac{1}{2\pi\sqrt{(R_{38} * R_{39} * C_{29} * C_{30})}}$$

$$= 1591.54\text{Hz}$$

The fundamental band pass frequency is in the range of 55-65Hz

2.1.5 Protection

Line Protection

In this project we work with line voltages. Having protection in case of line shorts is extremely important. Therefore, a fast blowing 16A fuse was placed in the Line wire on the input cable before connecting to the tabs.

MCU ADC Pins protections

Any voltage, one diode drop above 3.3V can damage the MCU ADC pins. A back to back schottky diode as shown below was placed on the ADC pins for protection.

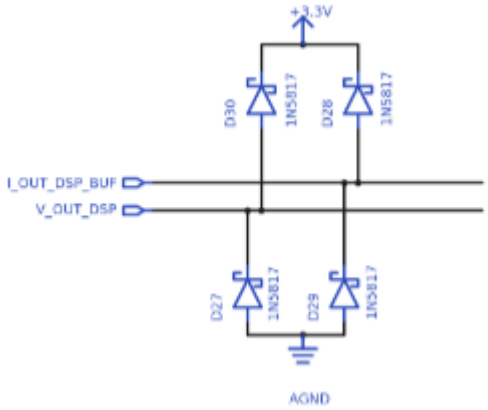


Figure 2.6 MCU ADC pins protection with back to back Schottky diodes

2.1.6 Kentec Touchscreen Display

A touchscreen display from Kentec provides the display functionality. The display is a 320x240 TFT LCD. It works on SPI and provides resistive touchscreen [8]. It uses an SSD2119 Solomon Systech LCD controller [9]. This display module is available as a boosterpack for the TIVA Launchpad that we use and plugs directly into the Boosterpack2 slot.

2.1.7 Ethernet Module

The Launchpad can connect directly to Ethernet network using an RJ45 connector. The microcontroller contains a fully integrated Ethernet MAC and PHY. In this application, we used LwIP stack for the Ethernet.

2.1.8 MCU pin assignments

The following table lists the GPIO pin assignments for the application.
A1, B1, C1, D1 headers refer to Boosterpack1 set of headers
A2, B2, C2, D2 headers refer to Boosterpack1 set of headers
X11 refers to the 98 pin edge header

Kentec Display	Description	MCU	Launchpad
LCD_SCS	chip select	PP3	D2-8
LCD_SDC	Cmd/Data	PP4	A2-8
LCD_SCL	Clock	PQ0	A2-7
LCD_SDI	Data Input	PQ2	D2-6
LCD_RST	Reset	PK6	C2-9
LED_PIN	Back Light	PG1	C2-1
LCD_X+	X positive_ADC	PB5 (AIN11)	B2-4

LCD_Y+	Y positive_ADC	PB4 (AIN10)	B2-3
LCD_X-	X Negative	PK7	C2-10
LCD_Y-	Y Negative	PM6	D2-10
Power Board	**Description**	**Tiva Board**	**Launchpad**
EXEC1	Execution test	PN0	X11-85
EXEC2	Execution test	PN1	X11-87
PWM Out	SallenKey Input	PF1	C1-1
V_Zero_Crossing	Capture for PLL	PL4	C1-5
V_OUT_DSP	Voltage ADC	PE0 (AIN3)	B1-3
I_OUT_DSP	Current ADC	PE1 (AIN2)	B1-4

2.2 Software Design

The firmware for CAPE has been developed using Code Composer Studio v6. The firmware can be divided into the following sections: Control section, FFT computation, Display section, Ethernet Data. There was also some front end software written in C to act as the server and perform data logging operations over Ethernet. Each section will be discussed in detail. Before that all the different libraries used in the project are listed below. All these libraries except for CMSIS library are available in TivaWare [18].

2.2.1 Library Usage

TIVA Driver Library [10]

The driver library provides API for each peripheral in the microcontroller. The driver library encapsulates the user from register level programming. This driver library is also referenced by some of the top-level libraries like graphic library and LwIP stack.

Although we have used the driver library for all the peripherals, the general purpose timer module used in capture mode was programmed directly at the register level. The driver library APIs for this module were a little ambiguous.

IQMath Library [11]

The IQ math library provides functions that perform floating point math using fixed point representation. This library provided basic math computations like multiplication, division, square root, trigonometric function etc. that were used in AC metrics calculations.

CMSIS library [12]

CMSIS library, developed by ARM provides routines for various mathematical and Signal Processing operations for Cortex-M devices. We used the DSP routines in the CMSIS library for FFT computations for voltage and current harmonics. To get the CMSIS library to work for TIVA platform, it had to be recompiled using the instructions in [13]

Graphics Library [14]

TivaWare Graphics library was used to draw all the graphics on the display. It provides a host of widgets like canvasses, buttons and also provides drawing abilities such as lines and shapes.

LwIP stack [15] [16][17]

LwIP is a lightweight implementation of TCP/IP protocol written by Adam Dunkels but now is being actively developed by worldwide team of developers led by Kieran Mansley. This stack was used to implement the data logger functionality over Ethernet.

2.2.2 Control section

The control section code deals primarily with ADC measurements, AC metrics computation, sine wave generation and (Phased Locked loop) PLL algorithm. In fact, the whole firmware architecture is managed by the control code.

To understand the control code section and firmware architecture, it is first important to understand the sine wave generation.

Sinewave generation

The sine wave was generated using a PWM whose duty cycle follows a sine wave pattern. The frequency of the PWM was set to 20KHz. Since an MCU cannot generate negative voltages, the sine wave generated is a level shifted one (i.e.),

- sin (nπ) corresponds to 1.65V (50% duty cycle)
- sin ((2n+1) π/2) corresponds to 3.3V (100% duty cycle)
- sin ((2n+2) π/2) corresponds to 0V (0% duty cycle)

The high frequency carrier wave is filtered out using a SallenKey filter.
To generate a sine wave of the required frequency, we use 512 different duty cycle values from 0 - 2π radians. Thus we have a systick timer that interrupts 512 times a line cycle; every interrupt cycle modifies the duty cycle to the next point in the sine.
A sine_index variable counts from 0 to 511 incrementing every interrupt. This variable generates the radians for the sine as per the following formula:

$$radians = \frac{sine\ index * 2 * \pi}{512}$$

The sine values are generated using IQmath using IQsin() function. The IQsin() takes radians and returns the result (in _IQ format) from +1 to -1. To level shift this range (0-1), we add 1 and divide by 2. The code snippet is shown below:

```
/* Calculate sin */
sin = _IQsin(radians);
/* Level shift */
sin += _IQ(1);
sin >>= 1;
/* sin is now between 0 & 1. This is the duty required */
```

The diagram below illustrates the concept of sine wave generation. In one cycle, sine_index varies from 0-511. Thus 128 corresponds to 90 degees, 256 corresponds to 180 degrees, 38

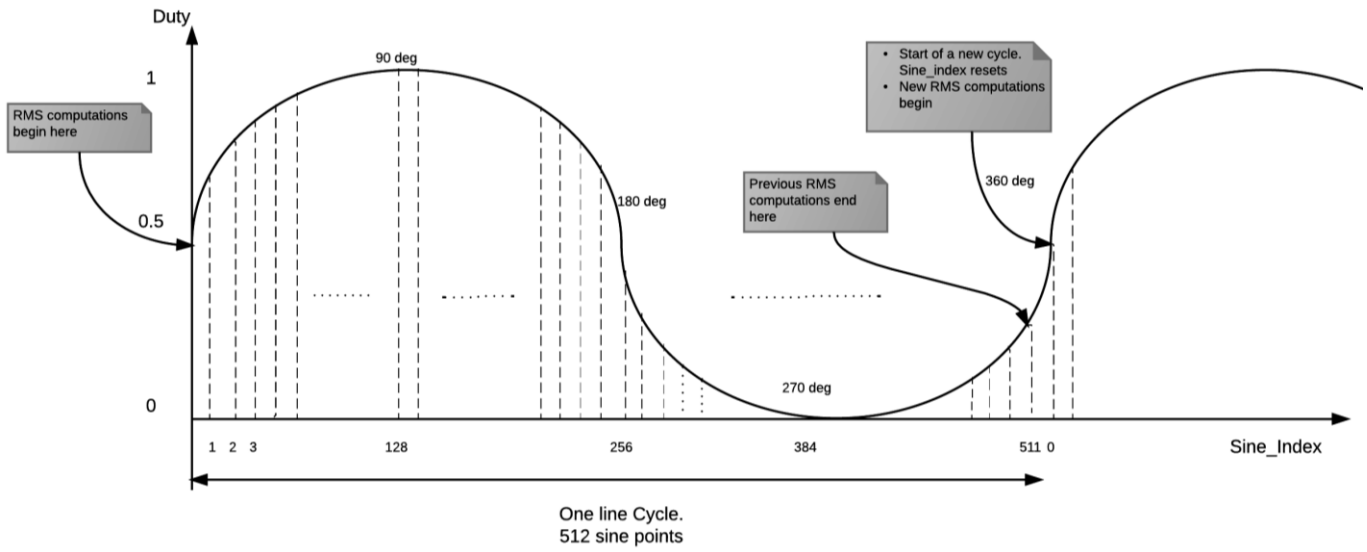


Figure 2.7 Sine Wave Generation

ADC sample, trigger and result acquisition

The ADC 1 samples 2 channels continuously in the interrupt – voltage and the current. The trigger and read mechanism is shown below:

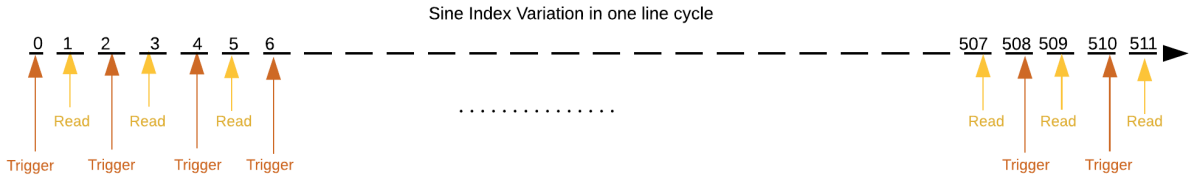


Figure 2.8 ADC Trigger and Read

The ADC trigger and read happen in alternate interrupts. The ADC is triggered whenever the sine_index is even. It is read in the next interrupt when the sine_index is odd. This kind of a scheme eliminates a busy wait scheme that would be if the ADC was triggered and read every interrupt. Thus, in one line cycle 256 ADC samples are taken.

ADC SAMPLING FREQUENCY

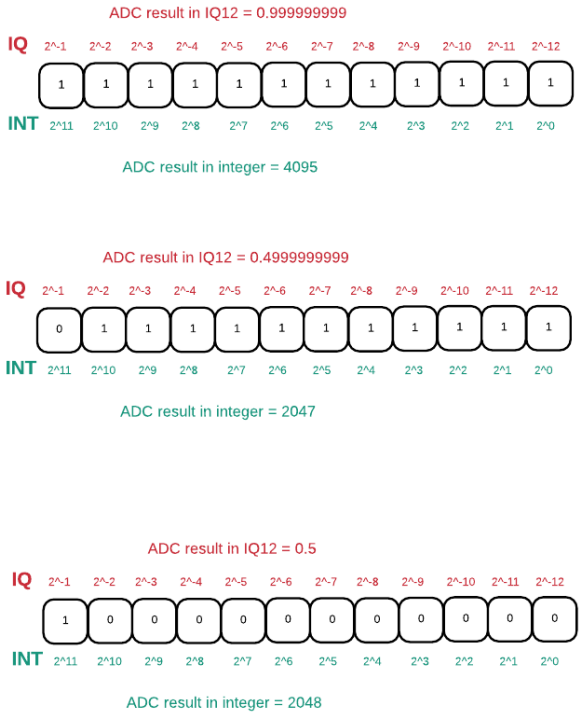
Assuming a line frequency of 60Hz, the ADC samples the current and voltage channels at 60*256 = 15,360HZ. The maximum harmonic frequency that we intend to measure with CAPE is 60*20 = 1200Hz. Thus the ADC sampling frequency is more than 10 times the frequency intended to be measured.

ADC RESULT ACQUISTION

TIVA has 2 12bit SAR ADCs. The result obtained from ADC conversion are in counts from 0-4095, where 4095 corresponds to 3.3V and 0 corresponds to 0V. Now since all the math calculation is performed

using IQmath, it is advantageous to treat the ADC result as a normalized IQ12 value (between 0 & 1). The 3 figures below explain this concept.

Figure 2.9 Normalizing ADC results



ADC read all 1's

ADC read 11 least significant bits as 1

ADC read the 12th bit as 1

The above figure shows how ADC results can be treated as normalized IQ numbers. In all our math, we are currently using IQ24. The normalized ADC results are in IQ12 as we see above. To convert to IQ24, we simply shift the ADC result left 12 times (=24-12). For further info on IQmath refer to [19].

AC Metrics calculations

Another important function of the control code is make all AC metrics calculations. A structure with all the metrics is shown below:

```
typedef struct
{
  rms_struct_t Iac;
  rms_struct_t Vac;
  _iq P_apparent;
  _iq P_active;
  _iq P_inst_acc;
  _iq P_PowerFactor;
  _iq Phase_shift;
  _iq frequency;
  float32_t Vthd;
  float32_t Ithd;
  _iq V_Peak;
  _iq I_Peak;
} ac_metrics_t;
```

RMS CALULATIONS

As mentioned earlier in 1.1 rms in time domain is given as:

$$(rms\ value) = \sqrt{\frac{1}{T}\int_0^T v^2(t)dt}$$

Converting this to z-domain,

$$(rms\ value) = \sqrt{\frac{1}{N}\sum_{n=1}^{N} v^2(n)}$$

Where N is the number of samples taken in one complete cycle. In this case N = 512. The rms computation starts in the interrupt with sine_index equal to 1. The ADC results are read, offset removed, squared and accumulated into a variable. In the last interrupt of the cycle, where sine_index equal to 511, all rms results are available. The mean and square root of the accumulated value is taken and the accumulated value is cleared. This computation is restarted the next cycle.

As mentioned earlier, ADC results are treated as normalized IQ values. Thus the whole rms calculation is performed on normalized values. In fact, the final rms result available is in normalized IQ24 format. To get actual scaled values, we multiply the normalized results with appropriate full scale values.

- Voltage channel full scale: 400V
- Current channel full scale: 49.5A

The derivations for the full scale values has already been discussed in section 2.1.3.

POWER CALCULATIONS

To compute power factor, we need both the apparent power and the average power delivered to the load.

Apparent Power

Apparent power is simply the product of Vrms and Irms. Since, the rms values are available once in a line cycle, average power is also computed once a line cycle.

Average Power

The equation for average power in time domain is given in section 1.1 converting this to z-domain, we get:

$$P_{av} = \frac{1}{N}\sum_{n=1}^{N} v(n)i(n)$$

The product of every sample of voltage and current is taken and accumulated. Every last interrupt of the cycle (sine_index = 511), average is taken and the accumulated value cleared.

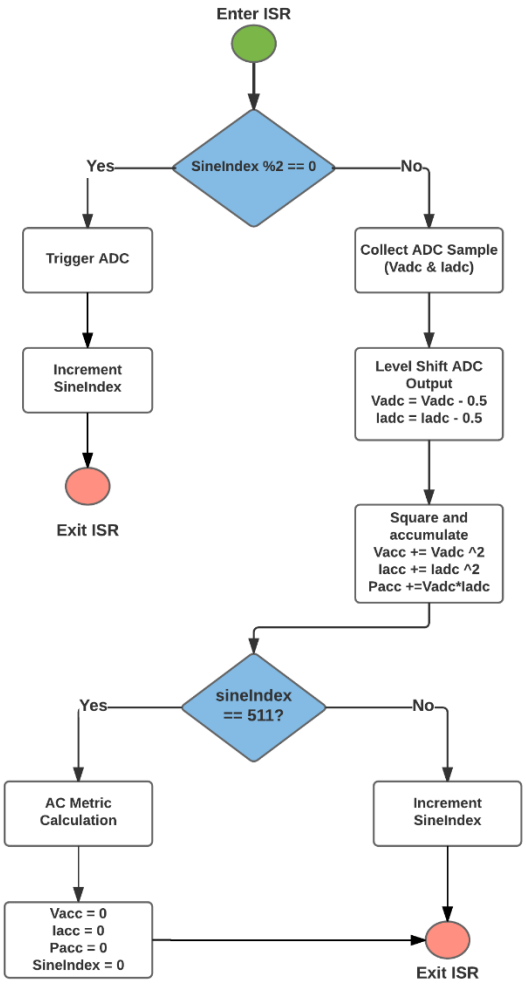
PEAK VOLTAGE, CURRENT CALCULATIONS

The maximum value among all the samples (after removing the level-shift offset) of voltage and current is the peak value for that cycle. The absolute value of the samples must be taken so that, the negative cycle is not eliminated from the computation.

Also, the peak values are cleared at the end of every cycle so that a new peak for the next cycle is obtained.

Below, is the overall algorithm flowchart for the computations discussed above.

Figure 2.10 AC metrics computation



2.2.3 Capture and PLL

This section is discussed separately even though it's part of the control section.

Capture Interrupt

As mentioned in section 2.1.3, we have a zero crossing detector that indicates when the voltage changes polarity. The typical waveform generated from this section is shown below.

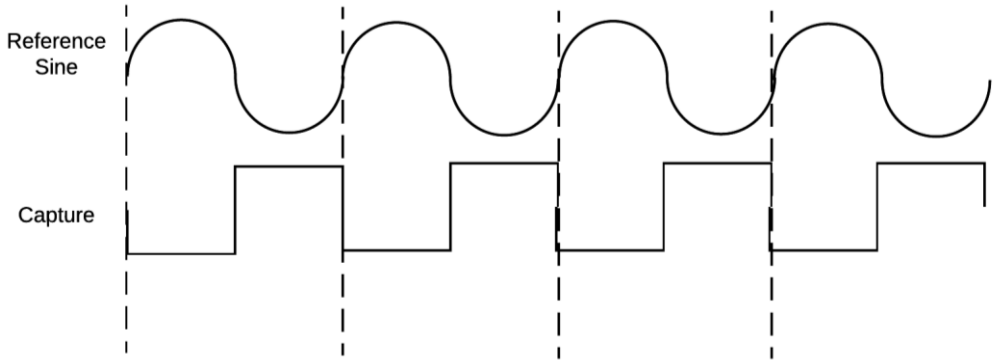


Figure 2.11 Capture Waveform

The general purpose timer in TIVA is configured in a 24-bit falling edge capture. The TIMER0_TAR register stores the period between consecutive interrupts in counts. To get the line frequency, the clock frequency (120MHz) has to be divided by the counts stored in the TAR register.
The timer is the only peripheral which was directly configured using registers. The configuration code snippet is shown below:

```
/* Timer 0 Capture Initialization */
ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
TIMER0_CC_R = 0x00; /* Clocked with SYSCLK (120MHz) */
TIMER0_CTL_R = 0; /* Timer disabled */
TIMER0_CFG_R = TIMER_CFG_16_BIT; /* 16-bit configuration */
/* Up count, Capture mode, Edge Time, Capture mode, Capture mode interrupt enable */
TIMER0_TAMR_R = TIMER_TAMR_TACDIR + TIMER_TAMR_TACMR + TIMER_TAMR_TAMR_CAP;
TIMER0_CTL_R = TIMER_CTL_TAEVENT_NEG; /* Capture on falling edge */
TIMER0_IMR_R = TIMER_IMR_CAEIM; /* Capture mode event interrupt enable */
TIMER0_ICR_R = 0xFFFFFFFF; /* Clear interrupt status */
TIMER0_TAILR_R = 0xFFFF; /* 2^16 is the upper bound */
TIMER0_TAPR_R = 0xFF; /* 2^8 is upper bound */
TIMER0_CTL_R |= TIMER_CTL_TAEN; /* Enable Timer */
```

PLL algorithm

Phased locked loop (PLL) algorithms synchronizes the internally generated sine wave with the reference sine wave in frequency and phase. This algorithm runs in the main loop once a line cycle.

Need for PLL

- PLL is one of the most basic algorithms in a grid connected inverter, where the current pushed out must be in phase with the line voltage.
- In CAPE, we don't actually require a sine wave generation, but a PLL algorithm is still useful to measure the start and end of a line cycle. Most of the AC metrics calculations like rms, power etc. depend on the information of line cycle.

The Algorithm

The PLL algorithm requires pieces of information:

- The frequency counts for the last line cycle obtained from the last capture interrupt

- The phase difference of the internally generated sine at the instant the capture interrupt occurs
- It is also important that the PLL algorithm be run once a line cycle

To PLL algorithm basically adjusts the value of the systick interrupt such that the generated sine wave matches the reference wave. The PLL algorithm consists of 2 stages: frequency sync and phase sync.

Frequency Sync

In this stage, all that needs to be done is to adjust the value of systick timer so that, the we still get 512 systick interrupts for the new reference frequency. The following code snippet explains this concept.

```
systick_present = SysTickPeriodGet();  /* Get the present systick period */
new_capture = pll->freq_in_cap_counts; /* freq of last line cycle updated in last
                                          capture interrupt */
systick_final = new_capture/SINE_SAMPLE_SIZE; /* next value of sytick period */

/**
 * FREQUENCY SYNC
 * The change in systick allowed is restricted to MAX_INCREMENTAL_CHANGE_IN_COUNTS to
 * ensure a smooth transition to new frequency */
change = _IQsat((systick_final - systick_present),MAX_INCREMENTAL_CHANGE, -
                MAX_INCREMENTAL_CHANGE);
systick_next = systick_present + change;
SystickPeriodSet(systick_next);  /* Set the new systick period */
```

Phase Sync

Once the frequency sync is completed, phase sync is done. Phase sync involves adjusting the systick timer by one count on either side base on phase difference to slightly decrease/increase the frequency of the generated waveform so that it slowly slews toward the reference waveform.

The first step is to obtain the phase shift. This is done in the capture isr. Phase shift is nothing but the value of sine_index when the capture interrupt occurs. We know that the capture interrupt occurs at the positive zero crossing of the reference waveform. If we are perfectly in sync, at this instant the sine_index would be 0. If sine_index is 1-256, we are leading the reference waveform because we have already entered the positive half cycle at the capture instant. If sine_index is 257-511 we are lagging the reference waveform because we are in negative half cycle at capture instant. The following table summarizes this concept.

Sine_index at ↓ capture	Phase diff with ref wave	Phase_sync action	Impact of sine wave
0	In Phase	No Action	No Impact
1-256	CAPE leads	Increment systick by 1	Generated Sine frequency slightly lower than reference
257-511	CAPE lags	Decrement systick by 1	Generated Sine frequency slightly higher than reference

Table 2.1 PLL phase sync

The third column is the action that is taken to correct phase and 4th describes the impact on the generated sine wave. The code snippet is shown below:

```
    /** PHASE SYNCING **/
    if(PHASE_LEAD(shift_index))
      systick_next++;
    else if(PHASE_LAG(shift_index))
      systick_next--;

SystickPeriodSet(systick_next);  /* Set the new systick period */
```

PLL Corner Cases

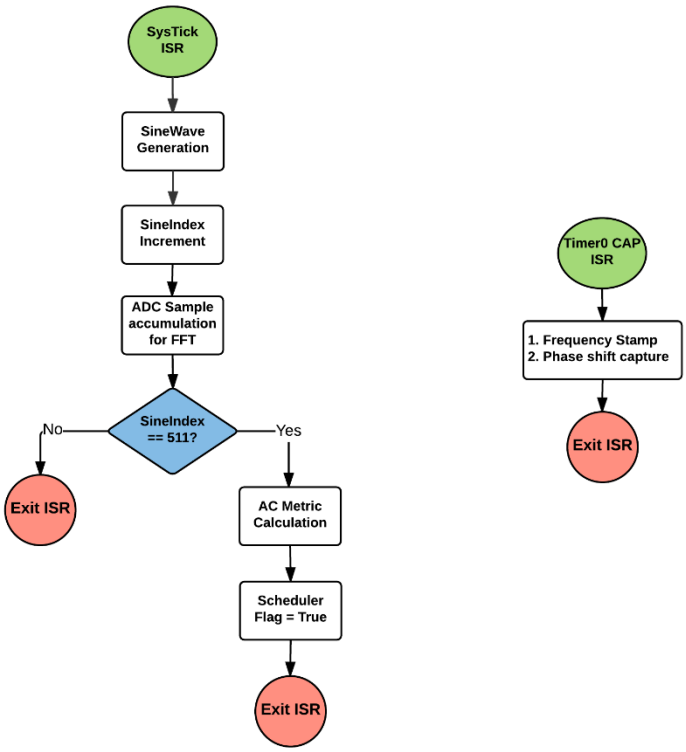
Two corner cases below need to be handled

a) *No capture interrupt:* This means there is grid voltage is missing (assuming no hardware faults). In this case PLL algorithm is not run and previous state is maintained

b) *Frequency out of range:* Typically grid frequency varies very little (1-2% max) and the generated waveform locks to whatever change in frequency. But we still place hardware limits on the max change in frequency that we sync to. Currently these limits have been set to +- 5Hz. Thus our range of operation is 55-65HZ. If reference frequency goes beyond these limits, we simply stay at either ends till the reference frequency comes back in range

As mentioned earlier, the control code determines the software architecture. There are 3 main interrupts running. (Systick, ISR, Ethernet). Events like display, FFT and PLL are scheduled events executing from main loop. These events are scheduled to run by the systick interrupt.

Figure 2.12 ISR map



The main loop events architecture flowchart is shown below

Figure 2.13 Main loop architecture

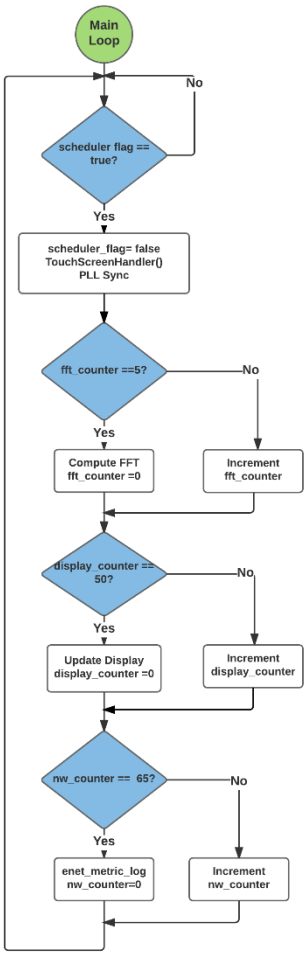


Table 2.2 Interrupt summary table

ISR name	Occurrence (Hz)	Description
Systick	Line*512 Hz	Sine wave generation, AC metrics
Timer0	Once in line frequency	Frequency stamp, phase shift
Ethernet	H/W or 60Hz	Ethernet timer

Events summary table

Event name	Occurrence	Description
TouchscreenHandler()	Line Frequency	Touchscreen events
Phase_locked_loop()	Line frequency	PLL algorithm
FFT_Compute()	Once in 5 line cycles	FFT and THD calculations
Display_update()	Once in 50 line cycles	Graphics Display
ENET_metrics_log()	Once in 65 line cycles	Ethernet data logger

2.2.4 FFT section

To measure the THD, we need to resolve the voltage and current into individual frequency components. The formula for Fourier series given in section 1.1 is repeated here for convenience:

$$v(t) = V_0 + \sum_{n=1}^{\infty} Vn\cos(n\omega t - \varphi_n)$$

$$i(t) = I_0 + \sum_{n=1}^{\infty} In\cos(n\omega t - \theta_n)$$

In z-domain, we take a 256 point FFT to resolve the frequency spectrum. A 256 point FFT for real values yields 128 complex values (the 2nd 128 are simply complex conjugates of 1st 128) with each point spaced at $\frac{Fs*n}{256}$, where n = 0,1,2…128,
Since we are sampling at line frequency * 256 Hz, our frequency point would be f = 0, 60, 120, 180 … 7680Hz (assuming line frequency = 60Hz). We are interested in ranges from 0 to 1200Hz only.

The CMSIS DSP library provides routines to compute the FFT. Notes on usage of the library is available in [20]. The instantaneous ADC samples of voltage and current are stored in 256 sized array and passed through the FFT routine. The output buffer now stores 128 complex values in 256 locations. This is then passed to a function that computes the magnitude of the complex buffer.
The final output buffer contains 128 amplitude values for 128 frequency points starting from index 0.
The FFT event as mentioned earlier occurs once in 5 cycles. The below flowchart details the process:

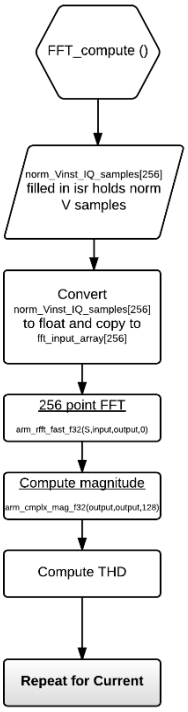


Figure 2.14 FFT computation flow

2.2.5 THD computation

The FFT magnitudes in the output buffer from the CMSIS library are scaled by 128 (FFT_LENGTH/2). To compute the THD we first rescale them to normalized values by dividing with 128. Then the THD is computed using the following formula [21]

$$\mathrm{THD_F} = \frac{\sqrt{V_2^2 + V_3^2 + V_4^2 + \cdots}}{V_1}$$

Where V1 is the magnitude of the fundamental component. V2, V3, V4 ... are magnitudes of the harmonics.

2.2.6 Display Section

The 240x320 Kentec TFT resistive touch screen is used display all the information and waveforms. The Kentec works on SPI communication. The graphics library[14] by TI provides both the high level application layer as well as device drivers to interface with the display. Currently the display has 4 pages:

- AC metrics: Displays all the metrics in one screen
- Voltage spectrum: Draws the voltage spectrum on the screen from FFT output array
- Current spectrum: Draws the current spectrum on the screen from FFT output array
- Time domain: Draws the voltage and current waveforms

See 3 and images folder for pictures of the screens.

Creating Widgets [14]
Many of the graphics objects on the screen are created as widgets. The graphics library organises widgets in a tree structure that contains a root called WIDGET_ROOT. Widgets can be created at run-time or in compile time. Widgets can also be added or removed from the tree dynamically.

Two kinds of widgets were used in this project: Canvas and Buttons.

CANVAS
The canvas widget provides a simple drawing surface that provides no means for interaction with the user. The canvas has the ability to be filled with a color, outlined with a color, have an image drawn in the center, have text drawn within it, and allow the application to draw into the canvas.

BUTTONS
The push button widget provides a button that can be pressed, causing an action to be performed.

The widget map of the project looks as shown below:
WIDGET_ROOT

- TitlePanel
- NextButton
- PreviousButton
- ACMetricsWidget

- Vrms
  - VrmsValue
    - IrmsValue
      - and so on…

The metrics displayed in the AC_metrics page are added as child widgets to ACMetricsWidget. This widget is removed and added to WIDGET_ROOT in runtime depending on what page is currently displayed on the screen.

Touch Screen Control

Kentec offers a resistive touchscreen. It requires 4 pins for interface. 2 ADC and 2 GPIO pins. ADC0 is used for touchscreen control (since ADC1 is being used for voltage and current sensing). The pins are continuously sensed in the while(1) loop and TouchScreenIntHandler() is invoked to detect key presses.

Drawing Waveforms on the screen

The voltage and current waveforms as measured by the ADC are plotted on the screen in Time Domain page. First two line cycles of voltage and current sample are accumulated in two different buffers. The actual plotting on the screen is performed using the library function GrLineDraw() that takes arguments X1, X2, Y1, Y2 (x and y co-ordinates of the line)

To draw a sine wave X1 and X2 get incremented linearly (X1 always lags X2 by 1). The Y axis points are a little trickier. Y1 is simply previous Y2. Y2 is computed as follows:

$$Y2 = Ycenter - Ycenter * Vinst$$

If Vinst follows a sine profile then Y2 follows a sine profile as well. The waveform is plotted in 2 steps. In the first step, The previous plotted waveform is erased by replotting it with the background color (black). In the second step the actual waveform for this update is plotted with the desired color. The previous set of Y2 pixels are stored for erasing.

The current waveform is plotted in the similar way.

2.2.7 Ethernet Data logger

Originally, it was intended that CAN would the protocol choice for data logging. The protocol was changed to Ethernet because of the faster speeds and ease of use. The MCU already provides the MAC and the PHY layers. LwIP stack developed by Adam Dunkels the provides the libraries for the upper layers.

A TCP/IP based remote server was running on a linux machine connected to CAPE via Ethernet port.

An example project from TivaWare enet_lwip was used as a reference project for porting lwIP stack into the CAPE project. For establishing connection steps mentioned in [17] was followed.

1. Cape behaves like a client with static IP: 192.168.1.3 and connects to a server with IP 192.168.1.2 on port 5000.
2. It logs the complete AC_metrics to the server approximately once every 1.08 seconds
3. A program written in C runs on the remote server and logs the received data into a CSV file with system time stamp

				AC METRICS							
Time	VRMS	IRMS	Frequency	PowerFactor	Apparent Power(VA)	Active Power(W)	THD(V)	THI(A)	Phase	Vpeak	Ipeak
Wed Apr 27 16:45:52 2016	120.4 V	0.9 A	60.0 Hz	0.55	103.3	57.1	0.0 %	1.4 %	56.4 dg	200.0 V	24.8 A
Wed Apr 27 16:45:53 2016	120.4 V	0.8 A	60.0 Hz	0.56	101.6	57.3	0.0 %	1.4 %	55.7 dg	171.2 V	2.6 A
Wed Apr 27 16:45:54 2016	120.4 V	0.9 A	60.0 Hz	0.55	103.1	56.8	0.0 %	1.4 %	56.6 dg	170.2 V	2.6 A
Wed Apr 27 16:45:55 2016	120.4 V	0.9 A	60.0 Hz	0.56	104	57.9	0.0 %	1.5 %	56.2 dg	171.0 V	2.6 A
Wed Apr 27 16:45:57 2016	120.4 V	0.9 A	60.0 Hz	0.55	104.2	57.7	0.0 %	1.4 %	56.4 dg	171.2 V	2.7 A
Wed Apr 27 16:45:58 2016	120.3 V	0.9 A	60.0 Hz	0.55	104.2	57.4	0.0 %	1.4 %	56.6 dg	170.2 V	2.7 A
Wed Apr 27 16:45:59 2016	120.4 V	0.8 A	60.0 Hz	0.56	102.2	56.9	0.0 %	1.4 %	56.1 dg	169.8 V	2.6 A
Wed Apr 27 16:46:00 2016	120.4 V	0.9 A	60.0 Hz	0.57	103	58.5	0.0 %	1.3 %	55.4 dg	170.0 V	2.6 A
Wed Apr 27 16:46:01 2016	120.4 V	0.9 A	60.0 Hz	0.55	102.6	56.7	0.0 %	1.4 %	56.4 dg	169.9 V	2.6 A
Wed Apr 27 16:46:02 2016	120.5 V	0.9 A	60.0 Hz	0.55	104.1	57.7	0.0 %	1.4 %	56.3 dg	170.3 V	2.9 A
Wed Apr 27 16:46:04 2016	120.5 V	0.9 A	60.0 Hz	0.55	103	57.1	0.0 %	1.4 %	56.4 dg	170.8 V	2.6 A
Wed Apr 27 16:46:05 2016	120.5 V	0.9 A	60.0 Hz	0.56	102.9	57.5	0.0 %	1.4 %	56.0 dg	170.1 V	2.7 A
Wed Apr 27 16:46:06 2016	120.4 V	0.9 A	60.0 Hz	0.56	103	57.4	0.0 %	1.4 %	56.2 dg	169.8 V	2.6 A
Wed Apr 27 16:46:07 2016	120.4 V	0.9 A	60.0 Hz	0.55	103.9	57.1	0.0 %	1.4 %	56.7 dg	170.1 V	2.7 A
Wed Apr 27 16:46:08 2016	120.4 V	0.9 A	60.0 Hz	0.55	103	56.7	0.0 %	1.4 %	56.6 dg	169.9 V	2.7 A
Wed Apr 27 16:46:10 2016	120.4 V	0.9 A	60.0 Hz	0.56	103.9	58.5	0.0 %	1.4 %	55.7 dg	171.5 V	2.8 A

Figure 2.15 Ethernet AC metrics logs

3 Hardware Testing & Results

The system was incrementally tested in three steps with firmware being developed in parallel at each stage.

1. Initial tests
2. System bring up tests
3. Integrated tests

Following sections describe each step in detail and then the final results are discussed.

3.1 Initial Tests

Initially the sensing circuitry was verified using the signal from arbitrary waveform generator. Two independent waveform generators were used for voltage and current sensing circuitry. These waveform generators were synchronized using master-slave configuration with master providing 10MHz clock to the slave. The block diagram of the initial test setup is shown below

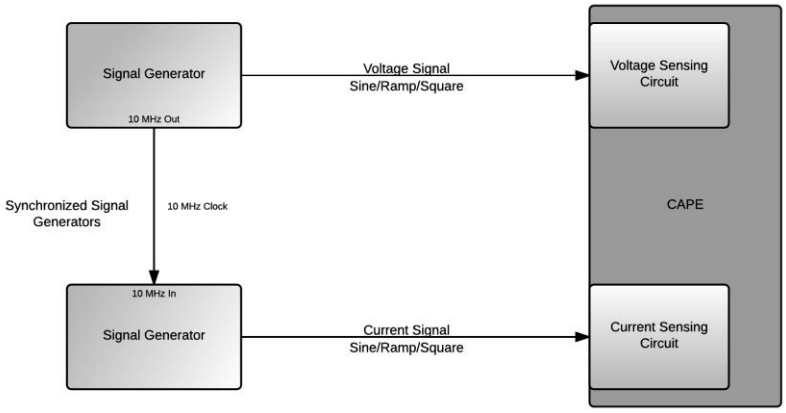


Figure 3.1 Initial test with function generators

In this stage the systick interrupt, ADC acquisition and scaling, PLL algorithms were verified. The hall-effect sensor was tested using a DC power supply in current mode

3.2 System bring up tests

Once the sensing circuitry was verified, line testing was initiated. To isolate the board and have a current limit in case of short circuit, Isolation transformer and variac were used. Variac also provides an option to gradually increase the line voltage. The block diagram for this stage is shown below.

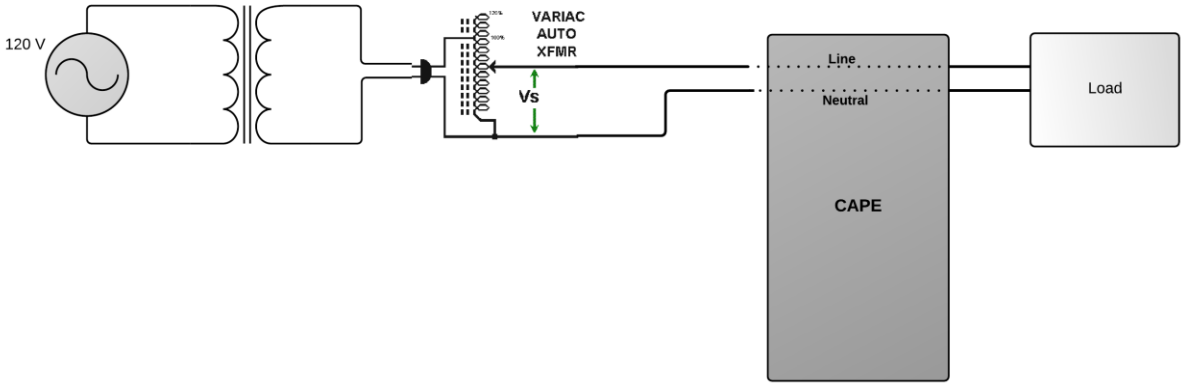


Figure 3.2 Initial line testing

1. All waveforms and voltage levels to the MCU board was verified before plugging in the MCU Launchpad. Because of unavailability of differential probes, comparative measurements with line voltages could not be taken. Measurements were taken independently with the waveform show below.

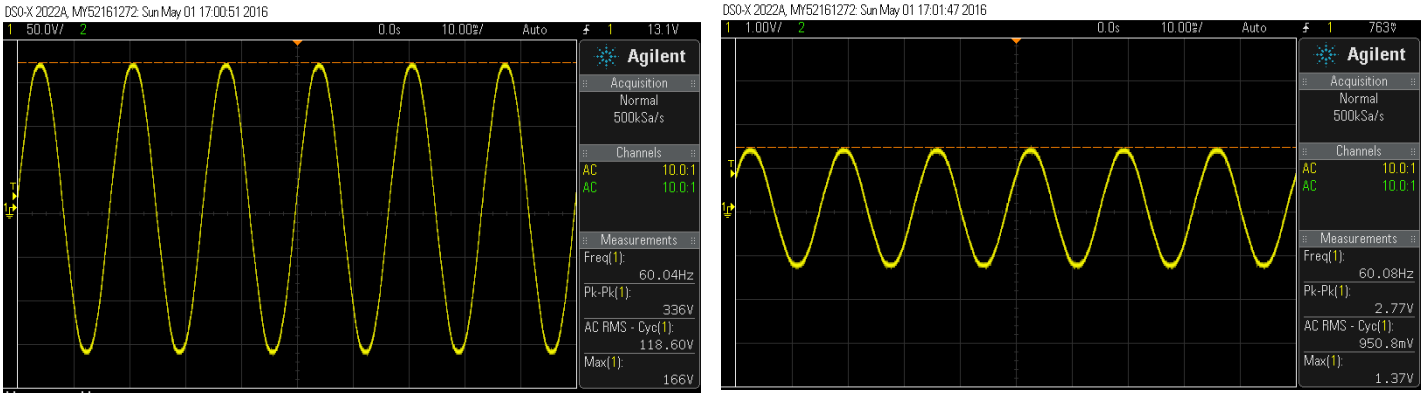


Figure 3.3 Line Waveform vs MCU Analog pin

2. The MCU was plugged in at this stage and ADC measurements of Voltage parameters was verified with multimeter and was found to be within 1%. The tabular column is shown below:

Line Voltage (Vrms)	ADC measured (Vrms)	Error on full scale (%)
20.12	19.53	0.42
50.50	49.97	0.37
75.98	74.67	0.93
119.76	118.66	0.78

Table 3.1 Line variation vs ADC measurements

3. Load was connected and current measurements were verified. Unfortunately, since current probes were not available, the accuracy could not be determined.
4. Zero crossing detector was verified and PLL algorithm was verified and syncing was successfully observed

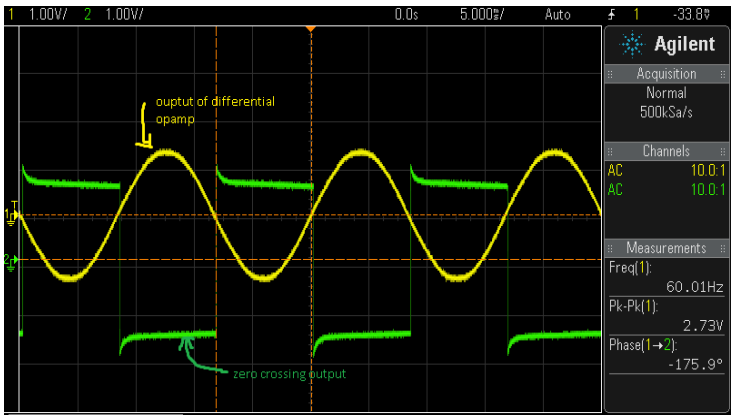


Figure 3.4 Zero Crossing detector

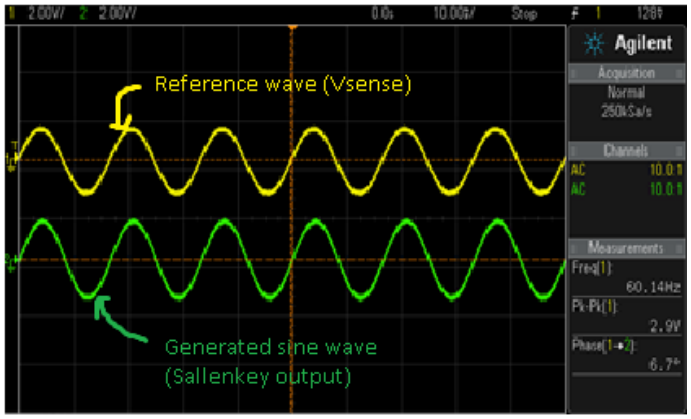


Figure 3.5 PLL syncing with grid waveform

3.3 Integrated Tests

- CAPE was tested by directly plugging into the line eliminating the variac and isolation transformer
- CAPE was tested with 2 different types of load.
  - poor power factor HP logic analyzer
  - high power factor desktop HP CPU
- FFT algorithm was added and computations were compared with oscilloscope's FFT computations
- TFT Display code was developed and incrementally tested
- Ethernet data logging was tested with remote server

3.4 Final results

3.4.1 HP logic Analyzer load (poor power factor, High current distortion)

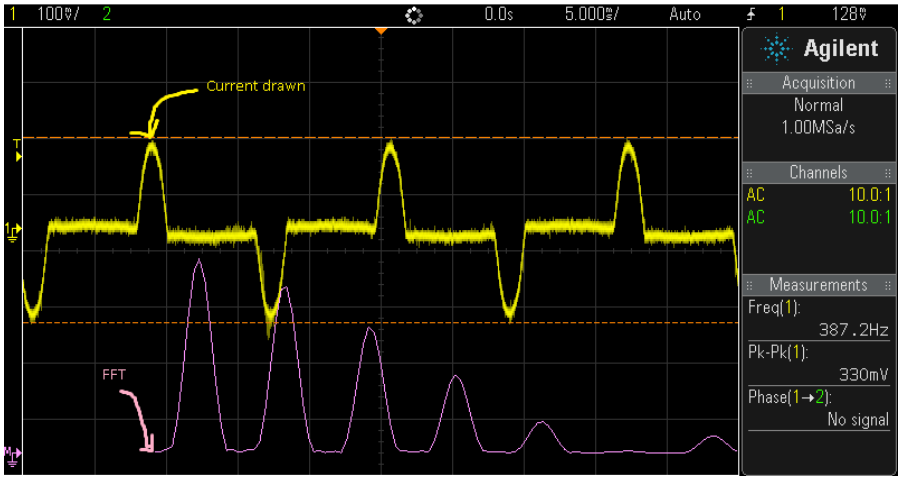


Figure 3.6 Current waveform and FFT (from scope)

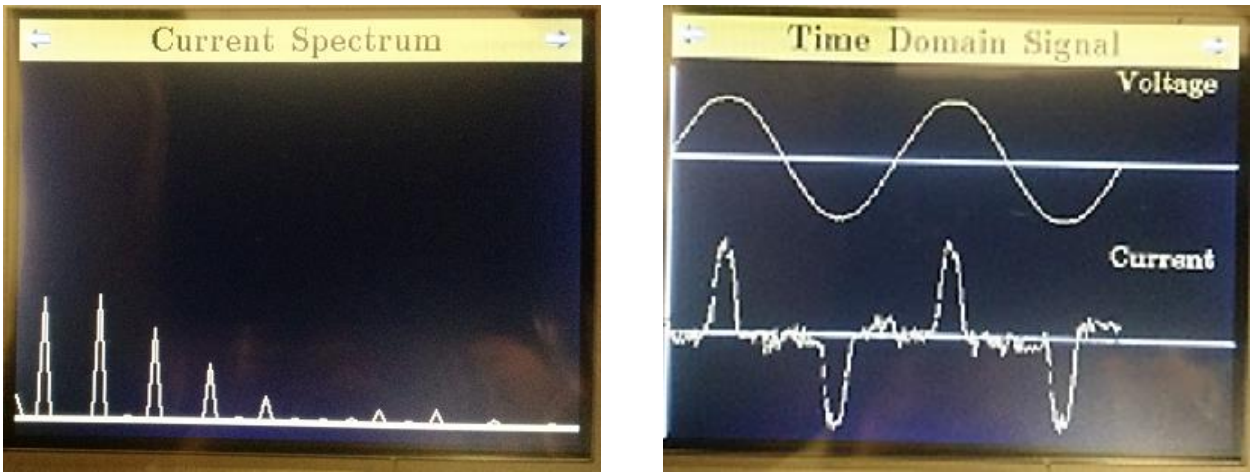


Figure 3.7 FFT and current waveforms (from CAPE)

3.4.2 HP CPU load (high power factor, Low current distortion)

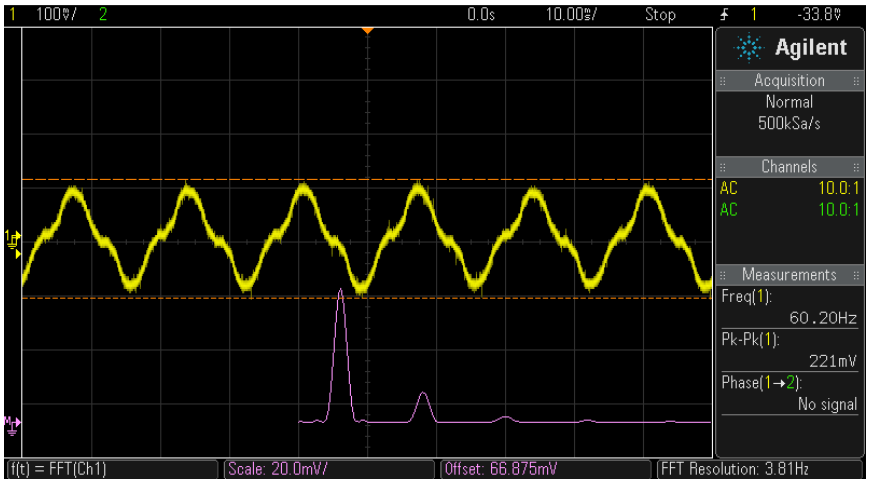


Figure 3.8 Current Waveform and FFT (from scope)

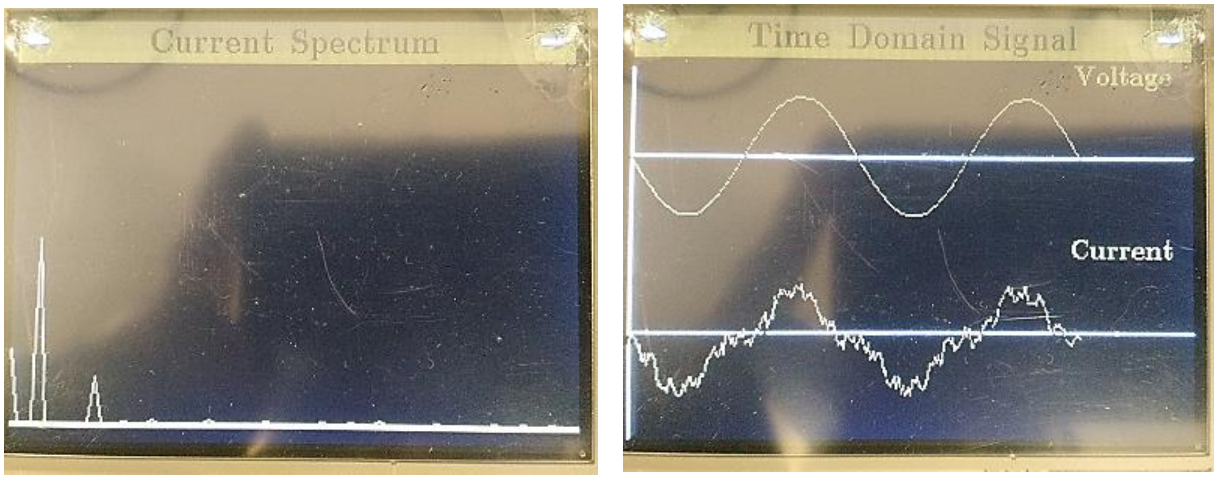


Figure 3.9 FFT and current waveforms (from CAPE)

3.4.3 AC metrics

AC metric page is shown just for the high power factor case

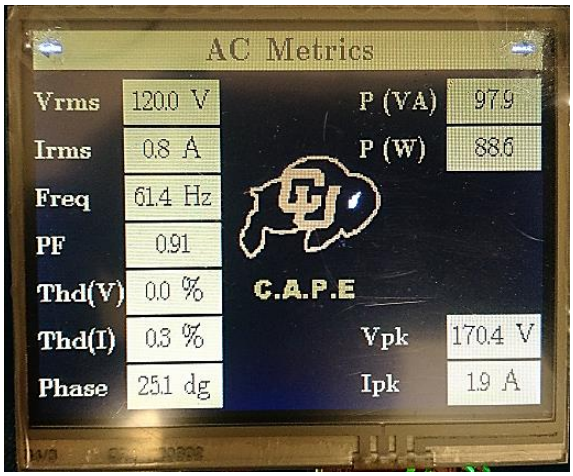


Figure 0.1 AC metrics page

4 Conclusion

In this project, a low cost harmonic analyzer for power equipment was designed and its performance was characterized. The performance was found to be comparable with an off-the-shelf measurement equipment (multi meter and oscilloscope).

The project incorporated concepts from both embedded systems, power electronics and signal processing. Working with line voltages, rms measurements, PLL algorithm were the aspects of power electronics. The embedded firmware had aspects of both real time and non-real time design. There was also elements of signal processing. Overall it was a perfect blend of various embedded system domains. The project if developed further has a good potential to become an actual product.

5 Future Development

The CAPE project has the potential to become a complete product. To get to that stage a few changes and enhancements have to be made.

5.1 Modifications

1. *Design a PCB board:* Due to the time constraints in the project, a PCB was not designed for CAPE. The complete hardware was realized on a perf board. Also, a separate MCU development board and a pluggable display was used. The whole solution can be optimized for cost by making a tightly integrated PCB.
2. *Add a connect/disconnect switch:* Currently there is no switch to disconnect the input and output mains connection. When CAPE is plugged power is always available at the output terminal. A mechanical switch is required to turn ON/OFF power delivery

5.2 New features

1. *Redesign the product for 3-phase input*: CAPE can be redesigned to allow measurement of 3-phase systems. Two different models (one for single phase and another for 3 phase) can be designed.
2. *UI interface on the screen can be redesigned*: Display and UI interface is an integral component of CAPE. It can be redesigned to look more clean and offer more features. An oscilloscope functionality can be implemented.
3. *CAN/RS485 logging functionalities*: CAN and RS485 are commonly used industrial protocols. They can be designed into CAPE
4. *Inbuilt webserver*: Currently CAPE acts as client. For future versions, CAPE can be the server hosting an inbuilt webserver. The data can be pulled from a browser on a host computer.

6 Acknowledgments

Firstly, we would like to thank Prof. Linden McClure, for a wonderful course and for providing us with an opportunity to work on this project.
We would also like to thank all the TAs of the course – Aniket, Ashwin and Virag for helping us find the parts for the project, answering all our doubts and helping us out of many sticky situations.
We would like to extend our heartfelt gratitude to Prof. Robert Erickson for reviewing the schematics and for lending us equipment from power electronics lab to test our project. We would also like to thank Prof. Dragan Maksimovic for his invaluable guidance on the project.
We would like to thank Mr. Swaminathan B – Infineon, Bangalore for reviewing the schematics and helping with component selection.
Finally, we would like to acknowledge and thank the authors of the various sources cited in the reference below.
This project would not have been successful without the help of all of these people.

7 References

[1] http://powerelectronics.com/power-electronics-systems/back-basics-power-factor-and-why-we-correct-it
[2] Robert W. Erickson, Dragan Maksimovic - Fundamentals of Power Electronics (Second Edition)
[3] http://www.ti.com/lit/ds/symlink/tm4c1294ncpdt.pdf
[4] http://www.ti.com/tool/ek-tm4c1294xl
[5] http://www.ti.com/lit/ug/spmu365b/spmu365b.pdf
[6] http://www.ti.com/lit/ds/symlink/tlc2274.pdf
[7] http://www.lem.com/docs/products/ho-p_series.pdf
[8] http://www.ti.com/lit/ug/slau601a/slau601a.pdf
[9] http://www.crystalfontz.com/controllers/SSD2119.pdf
[10] http://www.ti.com.cn/cn/lit/ug/spmu298a/spmu298a.pdf
[11] http://www.ti.com/lit/sw/sprc990/sprc990.pdf
[12] http://www.keil.com/pack/doc/CMSIS/General/html/index.html
[13] http://www.ti.com.cn/cn/lit/an/spma041g/spma041g.pdf
[14] http://www.ti.com/lit/ug/spmu300a/spmu300a.pdf
[15] http://lwip.wikia.com/wiki/LwIP_Wiki
[16] http://lwip.wikia.com/wiki/LwIP_Application_Developers_Manual
[17] http://lwip.wikia.com/wiki/Raw/native_API
[18] http://www.ti.com/tool/sw-tm4c
[19] http://processors.wiki.ti.com/images/8/8c/IQMath_fixed_vs_floating.pdf
[20] http://www.keil.com/pack/doc/CMSIS/DSP/html/group___fast.html
[21] https://en.wikipedia.org/wiki/Total_harmonic_distortion
[22] http://savannah.nongnu.org/projects/lwip/

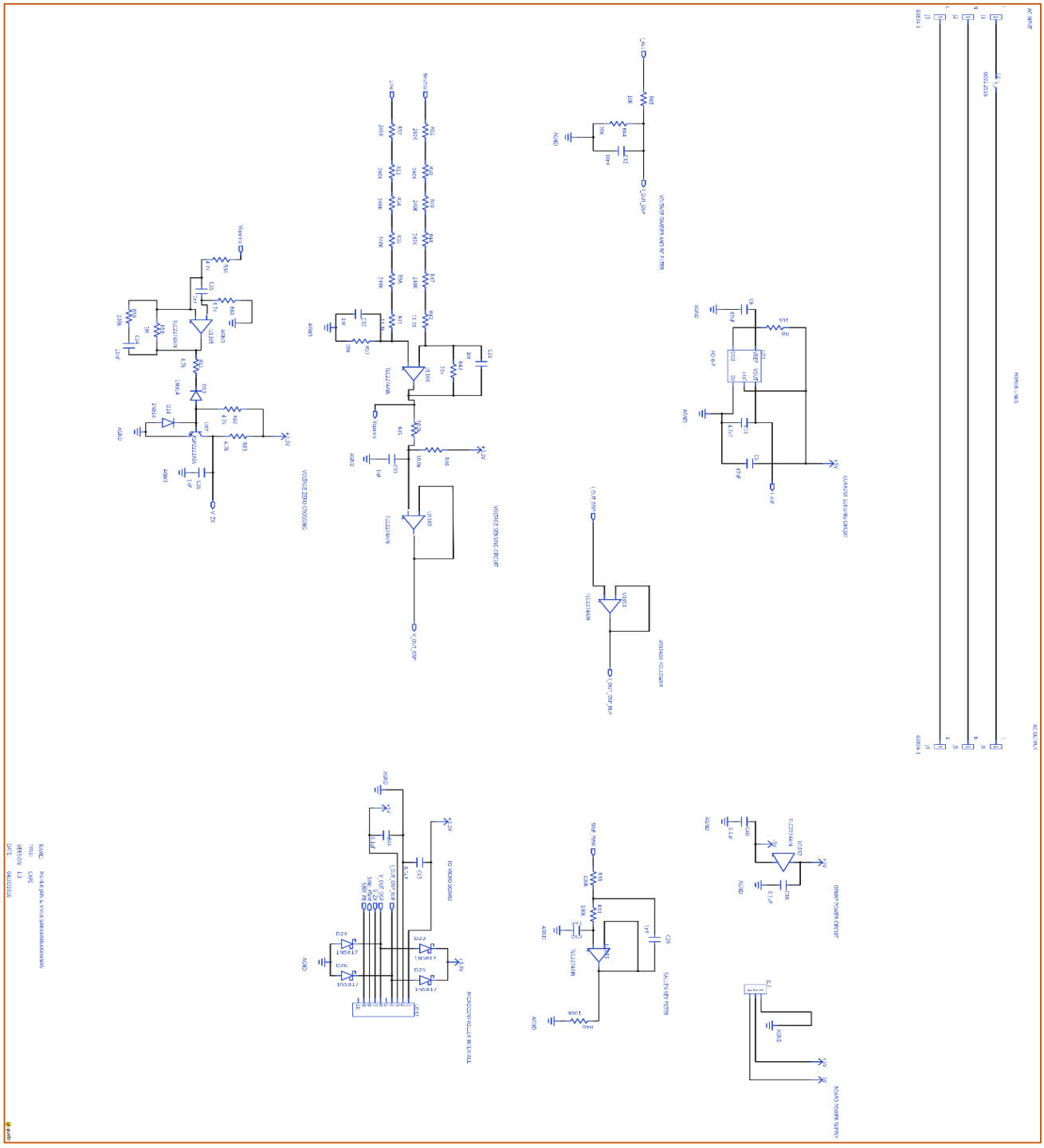
8 APPENDIX

Appendix consists of the following section: BOM, Schematics, Firmware, Datasheets

8.1 Bill of Materials (BOM)

S No.	Part Description	Qty	Source	Cost (in $)
1.	Tiva C series connected launchpad Board (TM4C1294)	1	http://www.ti.com/tool/ek-tm4c1294xl	19.99
2.	Kentec QVGA Display BoosterPack BOOSTXL-K350QVG-S1	1	http://www.ti.com/tool/BOOSTXL-K350QVG-S1	31.98
3.	Perf Board 4.0X6.0 inches	1	Digikey http://www.digikey.com	11.52
4.	Sensor Current HALL 6A AC/DC	1	Digikey http://www.digikey.com	26.99
5.	IC OPAMP GP 2.25MHz Texas Instrument (TLC2274AIN)	2	Digikey http://www.digikey.com	4.72
6.	4.7nf Ceramic Capacitor	1	Digikey http://www.digikey.com	0.19
7.	1nf Ceramic Capacitor	7	Digikey http://www.digikey.com	0.889
8.	10nf Ceramic Capacitor	2	Digikey http://www.digikey.com	0.68
9.	0.047uf Ceramic Capacitor	2	Digikey http://www.digikey.com	0.44
10.	0.1uf Ceramic Capacitor	6	Embedded Systems Lab	0.762
11.	Small Signal Diode (IN914)	2	Embedded Systems Lab	0.2
12.	Schottky Diode (IN5817)	4	Digikey http://www.digikey.com	1.72
13.	Fuse 16A, 250VAC, 125VDC, 5x20	1	Digikey http://www.digikey.com	0.59
14.	Fuse Holder SMD	1	Digikey http://www.digikey.com	1.29
14.	100k Resistor	4	Digikey http://www.digikey.com	0.16
15.	11.3K Resistor	2	Digikey http://www.digikey.com	0.2
16.	20K Resistor	2	Digikey http://www.digikey.com	0.2
17.	4.7K Resistor	5	Digikey http://www.digikey.com	0.2
18.	10K Resistor	4	Embedded Systems Lab	0.4
19.	240K Resistor	10	Digikey http://www.digikey.com	2.74
20.	1M Resistor	1	Digikey http://www.digikey.com	0.04
21.	PN222 NPN Transistor	1	Embedded Systems Lab	0.25
22.	Power Cord	1	Digikey http://www.digikey.com	3.54
23.	16 Pin DIP Socket	2	Embedded Systems Lab	3
24.	16 Pin Strip Header	1	Embedded Systems Lab	1
25.	Ethernet Cable	1	ITLL	
26.	Digikey Shiping Fee			12.65
	Total Cost			**126.251**

8.2 Schematics



TIVA Launchpad schematics can be found here [5]
Kentec display boosterpack schematics can be found here [8]

8.3 Datasheets

1. HALL Current Sensor data sheet can be found here
2. Texas Instrument (TLC2274AIN) OPAMP data sheet can be found here
3. Tiva C series connected Launchpad (TM4C1294) data sheet can be found here
4. Kentec QVGA Display BoosterPack (K350QVG-S1) data sheet can be found here
5. Kentec LCD controller (SSD2119) datasheet can be found here

8.4 Source code

Source code is linked as a separate attachment “Appendix_Software_SourceCode.pdf”