PROJECT REPORT

Micrium RTOS based Phase Locked Loop over Tiva-C LaunchPad for Grid-Tie Inverter

ECE 6336 - Advanced Microprocessor Systems

By: **Shreyas Rajendra Poyrekar – 1808531**

Vicky Bidawat – 1598755

Tata Aswini Kumar – 1805448

Naga Niktiha Kakani – 1792526

TABLE OF CONTENTS

SR. No.	TITLE	PAGE No.
1	ABSTRACT	3
2	CHAPTER 1	4
3	CHAPTER 2	6
4	CHAPTER 3	10
5	CHAPTER 4	14
6	EXPERIMENTAL RESILTS	16
7	CONCLUSION	18
8	ACKNOWLEDGEMENT	19
9	REFERENCES	20

ABSTRACT

During recent years, depletion of non-renewable energy resources, has put more emphasis on renewable resources. Grid tie solar micro inverter is one of such technique for renewable resources. Phase synchronization of inverter with the grid is required using the grid-tie inverter. In this project we talk about the implementation of a phase locked loop which can synchronize the phase as well as frequency of the inverter output and grid. The PLL algorithm has been implemented on TIVA-C TM4C123GH6PM using Micrium RTOS. Furthermore, the synchronized signal has been used to generate SPWM driving signals for the H- bridge inverter.

CHAPTER 1

1.1 INTRODUCTION

The conventional method of obtaining power from solar panels is combining the solar panels in either series or parallel and then to a DC-DC converter which in turn is given to an Inverter thus finally feeding it into the Grid. Instead of this Solar Grid-Tie Micro-Inverter is a panel having each of the above mentioned fundamental blocks built-in with a grid feeder as well. This system can be used in smart grid applications. As solar panels have low efficiency and out of the obtained there are losses due load impedance mismatch, external problems due to shading and other factors. For this purpose, to solve the problem of impedance matching and shading, maximum power point Tracking Algorithm (MPPT) is used. MPPT is basically adjusting the voltage or current w.r.t each other so that maximum power is achieved. Most of the time DC-DC converter are used (i.e. Voltage is controlled).

This Boosted supply is given to the inverter. The output of the Inverter is given to the feeder provided that the phase of the AC output is in synchronization with the AC supply in the Grid. So, Phase Locked Loop (PLL) is used to synchronize both and then feed to the grid. This is done by giving Sinusoidal PWM (SPWM) to the inverter which have the same phase as the grid (Generated by the PLL).

1.2 AIM OF THE PROJECT

Our aim in this project is to generate PWM signals for the DC-DC Converter and to implement the **PLL** to generate SPWM which are in synchronous with reference signal (Grid). We will use the **Micrium RTOS** to implement both Algorithms and generate the PWM signals for Converter and Inverter. We will make use of the ADC to sense the reference signal obtained by attenuating the Grid supply to voltage level of the controller. PLL and PWM are to be implemented real time. We will be designing and implementing an attenuator circuit which is used to create the reference sine wave for the SPWM. By using RTOS we can make changes in some parameters whenever needed instead of programing every time and add other fundamental algorithms in smart grid systems with the micro-inverter and make a compatible and upgradable system.

1.3 ORGANIZATION OF PROJECT REPORT

Chapter 1 provides the introduction to project followed by chapter 2 which describes the use of Micrium OS and other peripherals used on TIVA-C board. Further chapter 3 explains the PLL and its submodules along with mathematical models (equations). In chapter 4 the complete implementation of the project is then discussed. Finally, chapter 5 covers the experimental results achieved.

CHAPTER 2

We are using TM4C123GH6PM evaluation board also called TIVA-C with Micrium RTOS. We are using the ADC and PWM hardware blocks in the TIVA-C. we are trying to provide a brief explanation on some of the important features and explanation of the used blocks.

2.1 MICRIUM OS

Micrium OS (µC/OS-III) stands for micro-Controller Operating Systems. This is a real time operating system (RTOS) designed by embedded software developer, Jean J. Labrosse in 1991. It supports priority-based preemptive real-time kernel for microprocessors and is mainly written in C. In this chapter we will only discuss about the key API for task management and scheduler and its points.

2.1.1 OS INITIALIZATION

OSInit Initializes µC/OS-III this must be called prior to calling any other µC/OS-III function. Including OSStart () which will start the OS and scheduling all the tasks. OSInit () returns as soon as an error is detected. Provided OS_ERR is declared as a variable and passed as a parameter.

*Prototype: void OSInit (OS_ERR *p_err);*

2.1.2 TASK CREATION

Task can be created a by calling OSTaskCreate() by passing arguments specifying the TCB, stack, task function and other parameters needed for a task. Tasks are always created in the ready-to-run state. Tasks can be created anywhere in the main loop. A task cannot be created in the interrupt service routine.

```
Prototype:  void OSTaskCreate (OS_TCB        *p_tcb,
                               CPU_CHAR      *p_name,
                               OS_TASK_PTR    p_task,
                               void   *p_arg,
                                OS_PRIO       prio,
                               CPU_STK       *p_stk_base,
                               CPU_STK_SIZE   stk_limit,
                               CPU_STK_SIZE   stk_size
```

OS_MSG_QTY q_size,
OS_TICK time_quanta,
*void *p_ext, OS OPT opt,*
*OS_ERR *p_err);*

2.1.3 OS START

Calling this function starts the OS API's in µC/OS-III RTOS. This function is typically called from startup code. There are should be at least one task created. OSStart () will not return to the caller. Once µC/OS-III RTOS is running, calling OSStart () again will have no effect.

*Prototype: void OSStart (OS_ERR *p_err);*

2.1.3 ROUND ROBIN SCHEDULER

OSSchedRoundRobinCfg () can be used to enable or disable round-robin scheduling. OSSchedRoundRobinYield () is used to voluntarily give up a task's time slot, assuming that there are other tasks running at the same priority.

Prototype: void OSSchedRoundRobinCfg (CPU_BOOLEAN_EN,
OS_TICK dflt_time_quanta,
*OS_ERR *p_err);*

*void OSSchedRoundRobinYield (OS_ERR *p_err);*

2.2 ANALOG TO DIGITAL CONVERTER

TM4C123GH6PM microcontroller has two 12-bit ADC modules (ADC 0 &ADC1). Each ADC module has four programmable sequencers that can sample multiple analog inputs without the controller's intervention. Each sample sequencer is fully configurable for input source, trigger events, interrupt generation, and sequencer priority.

2.2.1 SAMPLE SEQUENCER

Sample sequencer handle's the control and data capture for the ADC input. There are four sample sequencers (SS0-SS3) in TIVA-C. All the sequencers are identical. The number of samples are different for each sequencer as well as the FIFO depth.

2.2.2 INITIALIZATION OF ADC

1. Enable the clock to the ADC module.
2. Enable the clock to the port.
3. Enable the alternate function of the GPIO pin.
4. Disable the digital function of the GPIO pin.
5. Configure the analog function on the GPIO
6. Select the ADC function.
7. Set the sequencer and its sampling frequency.
8. Set priority of the sequencer.
9. Configure the trigger event for each sample in the sample sequence
10. Connect the input of the ADC sequencer with the GPIO pin using the multiplexer.
11. Enable the sample sequencer logic.

2.3 PULSE WIDTH MODULATION

The TM4C123GH6PM has two PWM modules, each module has four PWM generator blocks and in total sixteen outputs. Each PWM generator block produces two PWM signals that share the same timer and frequency and can either be programmed with independent actions or as a single pair of complementary signals. It has One 16-bit counter in each generator. Each generator either runs in Down or Up/Down mode. The output frequency can be controlled by a 16-bit load value. Load value updates can be synchronized globally or locally. Produces output signals at zero and load value. Output PWM signal is constructed based on actions taken as a result of the counter and PWM comparator output signals. PWM generators can be operated independently or synchronized with other generators.

2.3.1 PWM INITIALIZATION

Enable the clock to the PWM module.

1. Enable the clock to the port.
2. Disable the analog function on GPIO pin.
3. Configure the digital function on the GPIO pin.
4. Select the clock to be used for the PWM & divide the system clock.
5. Enable PWM block.
6. Set either count down or count up-down mode
7. Set the generator actions when on load value or comparator value.

8. Set the load value.
9. Set the compare value.
10. Set the update value for the PWM signals.
11. Connect the GPIO pin with the generator output using multiplexer.

CHAPTER 3

3.1 PHASE LOCKED LOOP

A phase-locked loop or phase lock loop (PLL) is a control system that generates an output signal whose phase is related to the phase of an input signal. There are several different types; the simplest is an electronic circuit consisting of a variable frequency oscillator and a phase detector in a feedback loop. The oscillator generates a periodic signal, and the phase detector compares the phase of that signal with the phase of the input periodic signal, adjusting the oscillator to keep the phases matched.

Keeping the input and output phase in lock step also implies keeping the input and output frequencies the same. Consequently, in addition to synchronizing signals, a phase-locked loop can track an input frequency, or it can generate a frequency that is a multiple of the input frequency. These properties are used for computer clock synchronization, demodulation, and frequency synthesis.

ADC module as explained in chapter 2, gives a discrete sinusoidal output signal of the voltage range 0-3 volts and frequency 60Hz. This signal is given as input to the PLL which consist of three blocks as shown in the block diagram below *Fig1.*:

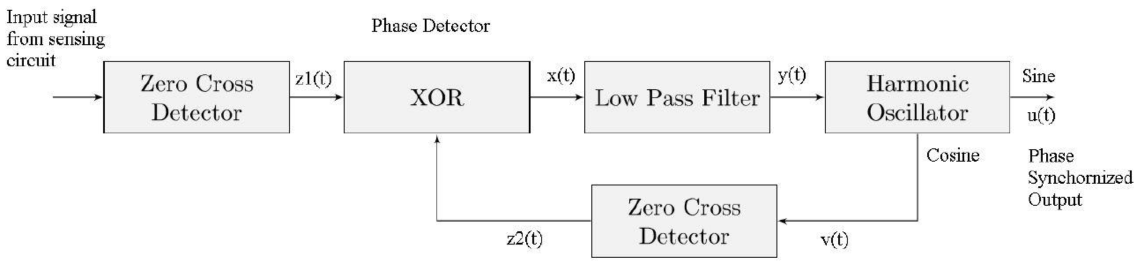

Fig 1. Block diagram of the implemented PLL module

The individual block with their mathematical model is further explained below:

3.1.1 PHASE DETECTOR

First block of PLL is a phase detector, which detects the phase of the input (grid) and compares it with the output of PLL. Basically, it checks whether the input and output signals are in phase or not. It consists of total two sub-blocks:

3.1.1.1 ZERO CROSS DETECTORS

This Block detects the zero crossing of input and output signal to check their phases. It gives high logic output only if the amplitude of the input is greater than or equal to zero. If both the zero cross detectors are giving the same output signals, then there is no phase difference between the input (grid) and output (PLL) signals.

3.1.1.2 XOR GATE

An XOR gate receives two input signals from the zero cross detectors. The low logic output from XOR gate indicates that there is no phase difference between the input and output of PLL.

3.2 LOW PASS FILTER

Second block is a low pass filter, whose output is used to generate"2*pi*f" which is required for the harmonic oscillator. There are two main functions of LPF in PLL. The primary function is to determine the loop stability of the system i.e. it will tell about loop disturbances, and changes in the input frequency. It determines how fast the loop achieves the synchronization as well as the locking range of the loop. It allows all the frequencies below the reference frequency. LPF gets input from XOR block whose output is a digital signal.

The mathematical model of the first order low pass filter implemented in this project is as follows:

Transfer function of 1^{st} order LPF is given by

$$\frac{\mathrm{Y(s)}}{\mathrm{X(s)}} = \frac{1}{T_c s + 1}$$

Where Y(s) is the filter output, X(s) is the output of XOR gate which is given as input to the filter and T_c is the Time constant.

$$(T_c s + 1)Y(\mathrm{s}) = \mathrm{X(s)}$$

By applying Inverse Laplace Transform on Eq.1

$$Tc * \dot{y(t)} + y(t) = x(t) \tag{1}$$

The Euler's Backward Differentiation approximation is given by

$$\dot{y}(t) = \frac{y(t) - y(t-1)}{h} \tag{2}$$

Where h is the time step which should be smaller than the filter time constant.

Substituting this in Eq.2

$$T_c * \frac{y(t) - y(t - 1)}{h} + y(t) = x(t)$$

Solving for gives

$$y(t) = \frac{T_c}{T_c + h} * y(t-1) + \frac{h}{T_c + h} * x(t)$$

$$if\ a = \frac{h}{T_c} + h = \frac{h}{\frac{1}{w_c} + h}$$

Then *y(t)* becomes

$$y(t) = (1 - a) * y(t-1) + a * x(t)$$

Considering, f = 60Hz and h = 25 ∗ 10−6 (step size) ∴ a = 0.0012

3.3 HARMONIC OSCILLATOR

Harmonic oscillator is implemented using bilinear transformation. This block receives the input from LPF and it continuously generates two signals (a sinusoidal signal and a cosine signal). This sinusoidal signal is synchronized with the grid. The cosine output signal is fed back through zero cross detector to XOR.

The mathematical model of the first order low pass filter implemented in this project is as follows:

Sine and Cosine signals have been generated using Bilinear Transform method. Output of LPF $y(t)$ is multiplied by a gain to obtain the required frequency of harmonic oscillator.

$$\therefore \omega = G * y(t)\ where\ G = gain\ \&\ k = \omega * T$$

Where T is the sampling period, $y(t)$ is the LPF output and ω is the required frequency of harmonic oscillator.

$$v(t) = \frac{\frac{(1-k^2)}{4}}{\frac{(1+k^2)}{4}} * v(t-1) + \frac{k}{1+\frac{k^2}{4}} * u(t-1)$$

$$u(t) = v(t-1) - \frac{k}{2} * v(t-1) - \frac{k}{2} * v(t)$$

Where $u(t)$ and $y(t)$ are sine and cosine signals respectively. This $u(t)$ signal will be in phase with the grid signal.

CHAPTER 4

4.1 IMPLEMENTATION

For implementation of the PLL algorithm using Micrium RTOS we have used the an ADC0 module with sequencer 3 on Port E pin 4 of the TIVA-C to read the DC offset sine signal. Generator 0 and Generator 1 in PWM0 module each having 2 PWM signals, in total 4 PWM signals are generated for the inverter on Port B pin 4,5,6,7. Out of the four signals two signals are SPWM.

We have made one AppStarter Task which is to create another task. This task creates ADC0_Seq3 Task, PWMDutySet Task & InverterPll Task. AppStarter Task can be termed as main task or idle task. ADC Initialization and PWM Initialization is done in the ADC0_Seq3 & PWMDutySet Tasks. The block diagram describes the architecture of the implementation on the TIVA-C. All the tasks and structure are connected as shown below in the block diagram.

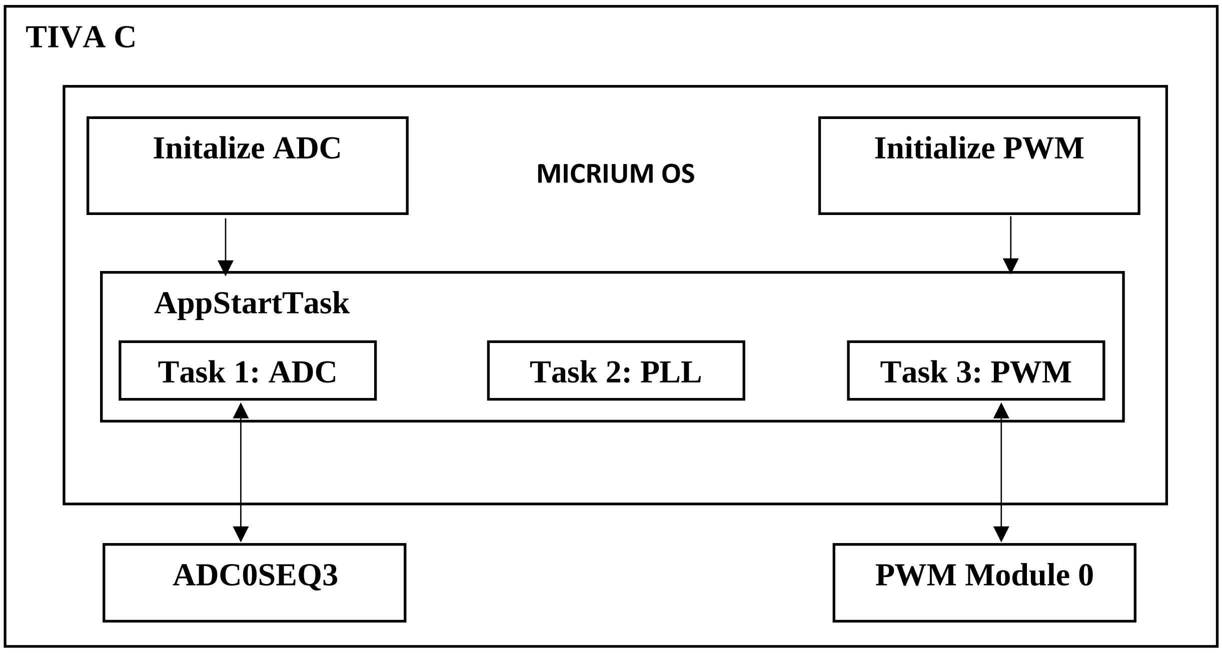

Fig 2. Complete Software Architecture

ADC0_Seq3 Task calls the ADC_ReadData function which reads the ADC input on the GPIO and stores into the global variable PllInput. InverterPll Task gets the global variable and processes the instance to generate output in phase with the input and store this in global variable PllOutput. PWMDutySet Task reads this instance, compares and determines the instance whether in positive of negative cycle and sets the CMP A & B value in the PWM0 Generator 0 & 1 to generate the 4 PWM signals for the inverter.

All the 4 tasks AppStarter Task, ADC0_Seq3 Task, PWMDutySet Task & InverterPll Task have the same priority and round robin scheduler is used for scheduling. We have set the time quanta for the round robin to 10 ticks for each task, but this does not work for priority if the task does not yield. So, we yield each task after one iteration of the while loop in each loop.

EXPERIMENTAL RESULTS

The results of the complete project are as follows

1. SPWM waveform for single leg of H-Bridge inverter

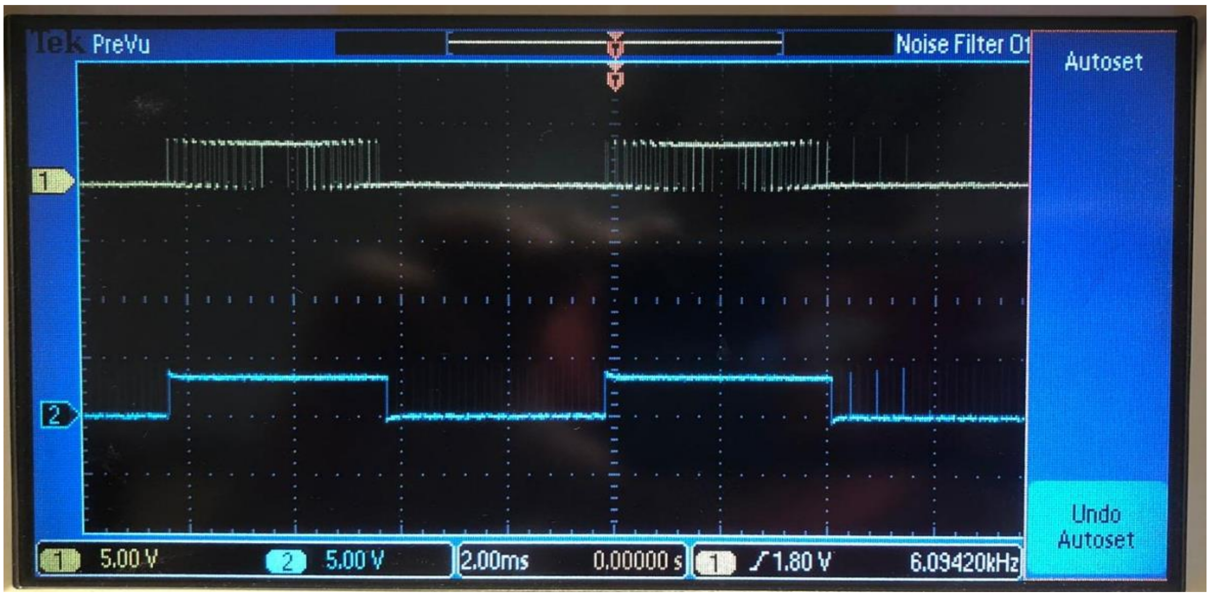

Fig 3. SPWM for one single leg of H-Bridge

2. SPWM waveform for both the legs of H-Bridge inverter which are perfectly 180^o out of phase.

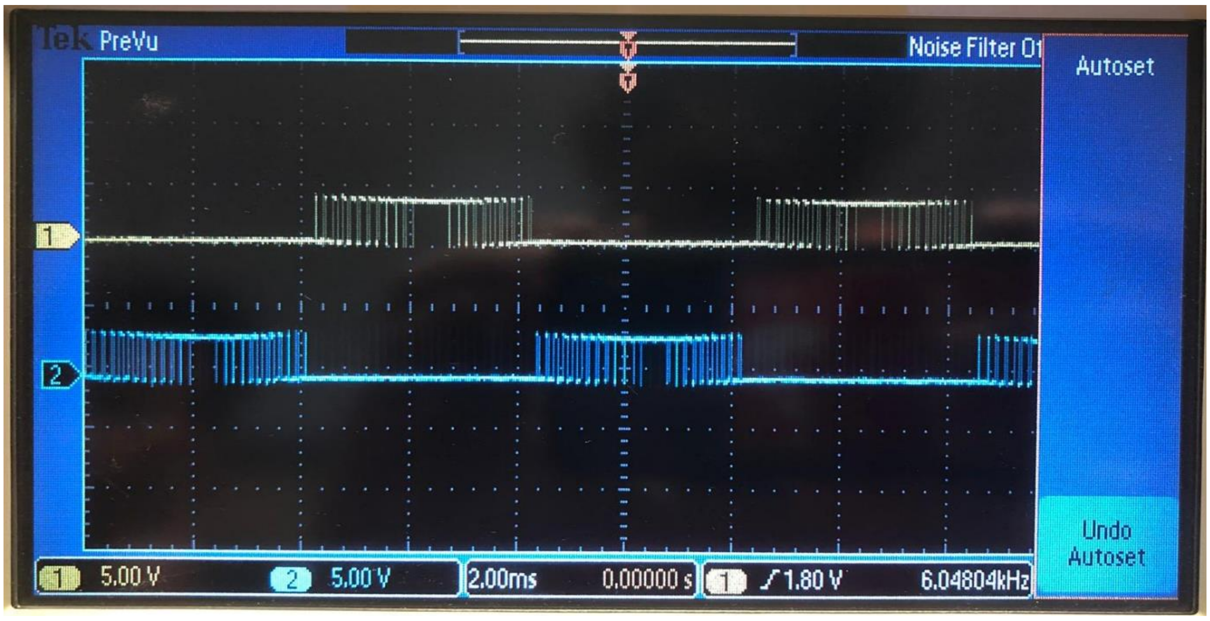

Fig 4. SPWM for both legs of H-Bridge

3. SPWM waveform is in synchronization with the input reference signal to ADC module indicating the proper phase locking of PLL module

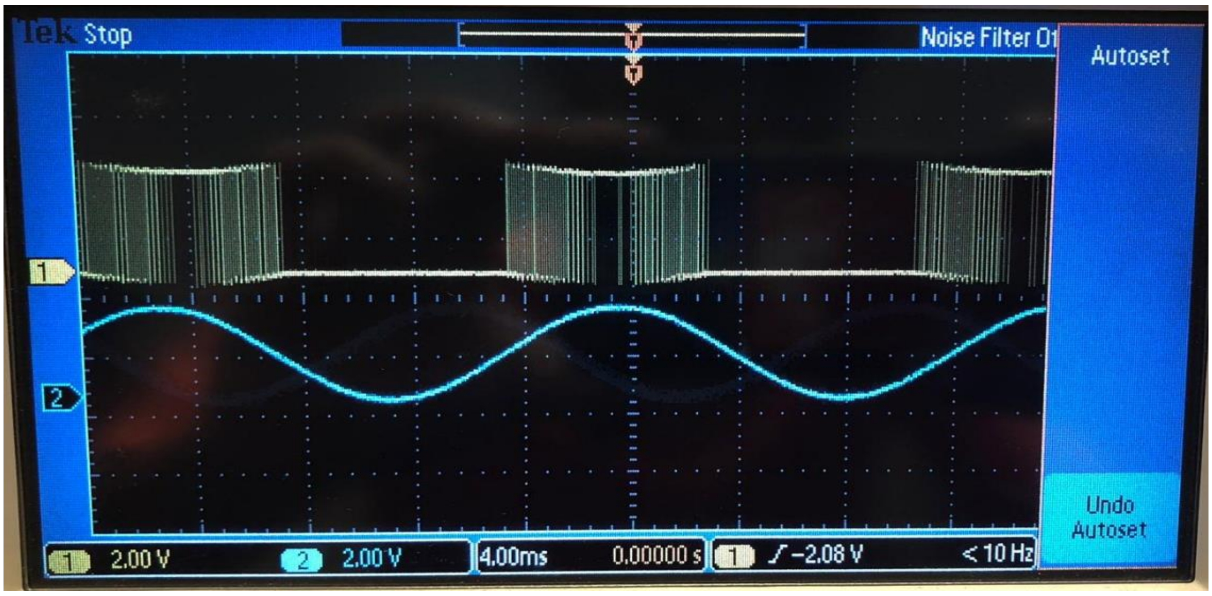

Fig 5. SPWM in synchronization with input reference signal

CONCLUSION

Phase synchronization for grid tie solar micro inverter using Micrium RTOS at small scale is described in detail including simulation and real-time implementation. We can draw the following conclusions form the results obtained from experimentation: Phase synchronization is achieved using Micrium RTOS for given sine input on TIVA-C. Portability is improved using RTOS as well as made flexible in changing the algorithm. Hardware cost to implement PLL is minimized. The presented results conclude that phase synchronization for grid tie solar micro-inverter can be achieved using any controller and RTOS.

ACKNOWLEDGEMENT

We would like to express our gratitude to all those who have contributed and motivated during our project work. We would like to thank our course instructor Dr. Harry Le for his constant help in technical and non-technical issues. We would also like to express our gratitude to him for delivering informative lecture and course content which helped out to implement our project.

Lastly, we are thankful to College of Technology department as well as Cullen College of Engineering for giving access to the labs for implementing our project.

REFERENCES

1. Qingxin Guan, Yu Zhang, Yong Kang, Josep M. Guerrero. Single-Phase Phase-Locked Loop Based on Derivative Elements IEEE Transactions on Power Electronics, Volume: 32, Issue: 6, Pages 4411 - 4420 June 2017
2. Sanjay Tolani, Tuhin Subhra Sasmal, Parthasarathi Sensarma. Low-Cost Digital Realization of Phase Synchronization for Grid Tied Micro Inverter Energy Conversion Congress and Exposition (ECCE), Pages 5991 - 5997, DOI: 10.1109
3. Digital Phase Lock Loops by Saleh R. Al-Araji, Zahir M. Hussain and Mahmoud A. Al-Qutayri (Publisher- Springer US, ISBN: 978-0-387-32863-8)
4. Microprocessor Applications in Power Electronics[Online]. Available: https://www.ee.iitb.ac.in/student/ cmouli/index files/ee675 slides 20%20OCT%2008.pdf
5. Derivation of a Discrete-Time LPF [Online].Available: http://techteach.no/simview/lowpass filter/doc/filter algorithm.pdf
6. http://www.ti.com/lit/ds/symlink/tm4c123gh6pm.pdf
7. https://doc.micrium.com/display/osiiihwosdoc/uCOS-III+HWOS+User+Manual
8. Embedded Systems: Real-Time Operating Systems for Arm Cortex-M Microcontrollers - Jonathan Valvano
9. https://en.wikipedia.org/wiki/Phase-locked_loop